



ESCUELA DE INGENIERÍAS
INDUSTRIALES



Universidad de Valladolid

UNIVERSIDAD DE VALLADOLID
ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

**PANEL LUMINOSO DE EMERGENCIA EN CARRETERA
CONTROLADO POR PANTALLA TÁCTIL Y DE FORMA
INALÁMBRICA**

Autor:

Sergio Carrasco Hernández

Tutor:

Luis Carlos Herrero de Lucas

Departamento de Tecnología Electrónica

Valladolid, junio 2024



Resumen

Este documento describe el proceso de mejora de un panel luminoso de emergencia en carretera desarrollado durante las prácticas en empresa en el departamento de Tecnología Electrónica de la Universidad de Valladolid. El proyecto surge de las limitaciones del panel original, con el objetivo de generar mensajes de advertencia más claros para los usuarios de la vía mediante texto y efectos dinámicos.

Se ha realizado un estudio de mercado obteniendo las principales características de los modelos encontrados y a partir de éstas se seleccionarán los componentes electrónicos que cumplan los objetivos de este proyecto.

Se programará un microcontrolador que permite controlar el panel luminoso formado por matrices de puntos LED según una interfaz de usuario creada en una pantalla táctil. También se programará una aplicación Android que permita controlar el dispositivo de forma inalámbrica mediante bluetooth. Todo el sistema estará protegido por una envolvente diseñada e impresa en 3D.

Palabras clave

Panel LED, ESP32, pantalla táctil, Bluetooth, Android.

Abstract

This document describes the process of improving a road emergency panel which was developed previously in the electronic technology department in the university of Valladolid. This project arises from the limitations of the previous panel with the objective of generate clearly message to the road users.

The main characteristics of the project are taken from a market study from which the selected components will be selected to meet certain objectives of this project.

In this project a microcontroller will be programmed so it can control the emergency panel according to a user interface on a touch screen. Additionally, an Android application will be developed to control the device remotely using Bluetooth. All these systems will be protected by some 3D printed designs.

Keywords

LED panel, ESP32, touch screen, Bluetooth, Android



Palabras abreviadas

CI – Circuito Integrado	GPIO – General Purpose Input Output
SRAM - Static Random Access Memory	PCB – Printed Circuit Board
EEPROM - Electrically Erasable Programmable Read-Only Memory	RGB – Red, Green, Blue
ADC – Analog to Digital Converter	MSB – Most Significant Bit
SPI - Serial Peripheral Interface	LSB – Least Significant Bit
UART - Universal Asynchronous Receiver-Transmitter	LED – Light Emitting Diode
I2C - Inter-Integrated Circuit	PWM – Pulse Width Modulation
LDO - Low Dropout Regulator	QSPI – Quad Serial Peripheral Interface
SoC – System on Chip	App – Aplicación
IoT – Internet of Things	UPDI – Unified Program and Debug Interface
Wi-Fi – Wireless Fidelity	LCD – Liquid Crystal Display
ROM – Read-Only Memory	DVP – Digital Video Port
ULP – Ultra Low Power	RF – Radio Frequency
CPU – Central Processing Unit	DAC – Digital-to-Analog Converter
RTC – Real-Time Clock	TFT – Thin Film Transistor
MCU – Microcontroller Unit	IDE – Integrated Development Environment

Índice de contenido

Capítulo 1. INTRODUCCIÓN.....	13
Capítulo 1.1. MOTIVACIÓN Y ORIGEN DEL PROYECTO.....	13
Capítulo 1.2. OBJETIVOS.....	14
Capítulo 1.2.1. Objetivos generales.	14
Capítulo 1.2.2. Objetivos específicos.	15
Capítulo 1.3. METODOLOGÍA.	15
Capítulo 2. ANÁLISIS DEL HARDWARE.....	17
Capítulo 2.1. ANÁLISIS DEL PROBLEMA.....	17
Capítulo 2.2. ANÁLISIS DE PRODUCTOS EN EL MERCADO.	17
Capítulo 2.2.1. PANEL MENSAJE VARIABLE TEXTO – PROIN	18
Capítulo 2.2.2. PANEL MENSAJE VARIABLE GRÁFICO – PROIN	18
Capítulo 2.2.3. Sistema Combinado Luminoso – SEBA.	19
Capítulo 2.2.4. POLVIS TXT – SEBA.....	19
Capítulo 2.2.5. Panel mensaje variable LED – TECNIVIAL.	20
Capítulo 2.2.6. Panel de mensaje variable LED hidráulico – TECNIVIAL.....	21
Capítulo 2.2.7. Letrero de desplazamiento LED – VEVOR.....	22
Capítulo 2.3. PRINCIPALES CARACTERÍSTICAS DE LOS MODELOS ESTUDIADOS.....	23
Capítulo 2.4. ALTERNATIVAS PARA EL PANEL LUMINOSO.	23
Capítulo 2.4.1. Panel flexible basado en WS2812B.....	24
Capítulo 2.4.2. LDM-6432 – LUMEX.....	26
Capítulo 2.4.3. LED Matrix – Adafruit.....	28
Capítulo 2.4.4. Panel basado en MAX7219/21.....	34
Capítulo 2.4.5. Full Color RGB LED Matrix Driver Shield + RGB Matrix Screen – SUNFOUNDER.....	37
Capítulo 2.5. ALTERNATIVAS PARA EL CONTROLADOR.....	38
Capítulo 2.5.1. Placas de desarrollo basado en Atmel/AVR.....	39
Capítulo 2.5.2. Placas de desarrollo basados en ESP32.	49
Capítulo 2.5.3. Placas de desarrollo basados en ESP32-S2.	53
Capítulo 2.5.4. Placas de desarrollo basados en ESP32-S3.	61
Capítulo 2.5.7. Otras placas de desarrollo.	67
Capítulo 2.6. INCORPORACIÓN DE PERIFÉRICOS.	71
Capítulo 2.6.1. Hardware para la interfaz con el usuario.....	71
Capítulo 2.6.2. Sensor de intensidad luminosa.....	75

Capítulo 2.6.3. Accionador para inclinar el panel.....	75
Capítulo 2.7. ALIMENTACIÓN DEL PROTOTIPO.....	77
Capítulo 2.8. ELEMENTOS SELECCIONADOS.....	80
Capítulo 2.9. CONEXIONES.....	81
Capítulo 3. PROGRAMACIÓN DEL PROTOTIPO.....	83
Capítulo 3.1. LIBRERÍAS PARA EL CONTROL DEL PANEL LUMINOSO MEDIANTE MAX7219 ..	83
Capítulo 3.2. LIBRERÍAS PARA LA PANTALLA TÁCTIL.....	86
Capítulo 3.3. LIBRERÍAS PARA LA COMUNICACIÓN INALÁMBRICA (BLUETOOTH/BLE/WIFI/SIMILARES).	90
Capítulo 3.4. IMPLEMENTACIÓN DEL PROGRAMA EN LENGUAJE C++.....	90
Capítulo 3.4.1. Software empleado para la programación final del prototipo.....	90
Capítulo 3.4.2. Metodología empleada.....	91
Capítulo 3.4.3. Programación de la pantalla.....	93
Capítulo 3.4.4. Programación del panel.....	99
Capítulo 3.4.5. Programación de la comunicación bluetooth e interpretación de los comandos.....	99
Capítulo 3.4.6. Detalles de otras librerías.....	101
Capítulo 3.5. IMPLEMENTACIÓN DEL PROGRAMA EN LENGUAJE KOTLIN EN ANDROID STUDIO.....	101
Capítulo 3.5.1. Software empleado para la programación de la aplicación.....	101
Capítulo 3.5.2. Metodología empleada en el desarrollo de la App.	101
Capítulo 3.5.3. Disposición de los menús y funciones de la App.	102
Capítulo 3.5.4. Estructura del programa de la App de Android.	105
Capítulo 4. FABRICACIÓN DE LA PCB.....	107
Capítulo 5. ENVOLVENTE DEL PROTOTIPO.....	108
Capítulo 5.1. OBJETIVOS DE LA ENVOLVENTE.....	108
Capítulo 5.2. ENVOLVENTE PANEL LED.	108
Capítulo 5.3. ENVOLVENTE DEL PANEL DE CONTROL.	110
Capítulo 5.4. ENVOLVENTE DE ELEMENTOS AUXILIARES.....	112
Capítulo 5.5. MONTAJE DEL PROTOTIPO.	113
Capítulo 6. PRUEBAS Y VALIDACIÓN DEL PROTOTIPO.	114
Capítulo 6.1. PLAN DE PRUEBAS.	114
Capítulo 6.2. RESULTADOS Y ANÁLISIS DE LAS PRUEBAS REALIZADAS.	114
Capítulo 6.3. VALIDACIÓN DEL PROTOTIPO.	115
Capítulo 7. COSTE DEL EQUIPO.	115



Capítulo 7.1. DESGLOSE DE COSTOS	115
Capítulo 7.2. EVALUACIÓN DE LA VIABILIDAD ECONÓMICA.....	117
Capítulo 8. EVALUACIÓN GENERAL Y PLANES DE FUTURO	117
Capítulo 8.1. CONCLUSIONES.....	117
Capítulo 8.2. LÍNEAS A FUTURO	118
Capítulo 9. BIBLIOGRAFÍA.....	121
Capítulo 10. ANEXOS.....	127

Índice de figuras

Figura 1 Panel simple de ocho módulos.....	13
Figura 2 Interior panel simple de ocho módulos.	13
Figura 3 Ejemplo de funcionamiento del panel realizado en las prácticas de empresa.....	14
Figura 4 PANEL MENSAJE VARIABLE TEXTO - PROIN. Vista lateral.	18
Figura 5 PANEL MENSAJE VARIABLE TEXTO - PROIN. Vista frontal.	18
Figura 6 PANEL MENSAJE VARIABLE GRÁFICO – PROIN. Señal de estrechamiento izquierdo....	18
Figura 7 PANEL MENSAJE VARIABLE GRÁFICO - PROIN. Señal de velocidad limitada a 80km/h.18	
Figura 8 Sistema Combinado Luminoso - SEBA.....	19
Figura 9 PolVis TXT - SEBA. Vista con el panel retraído.....	20
Figura 10 PolVis TXT - SEBA. Vista con el panel desplegado.	20
Figura 11 PolVis TXT - SEBA. Panel de control.....	20
Figura 12 PolVisTXT - SEBA. Distintas señales.....	20
Figura 13 Panel de mensaje variable LED - TECNIVIAL. Vista lateral.....	20
Figura 14 Panel de mensaje variable LED - TECNIVIAL. Vista frontal.	20
Figura 15 Panel de mensaje variable LED - TECNIVIAL. Detalle del panel.	21
Figura 16 Panel de mensaje variable LED - TECNIVIAL. Controladora del producto.....	21
Figura 17 Panel de mensaje variable LED hidráulico - TECNIVIAL. Vista de panel sin desplegar.21	
Figura 18 Panel de mensaje variable LED hidráulico - TECNIVIAL. Vista del panel desplegado. 21	
Figura 19 Panel de mensaje variable LED hidráulico - TECNIVIAL. Señalización desvío carril izquierdo.....	22
Figura 20 Panel de mensaje variable LED hidráulico - TECNIVIAL. Señal de advertencia.	22
Figura 21 Letrero de desplazamiento LED - VEVOR. Letrero encendido.	22
Figura 22 Letrero de desplazamiento LED - VEVOR. Letrero apagado.....	22
Figura 23 Paneles flexibles basados en WS2812B. Tamaños disponibles.....	24
Figura 24 Paneles flexibles basados en WS2812B. Ejemplos de configuración.	24
Figura 25 Componente WS2812B.	24
Figura 26 Empaquetado 5050 del WS2812B.....	24
Figura 27 Tabla de tiempos para la transferencia de datos del WS2812B.	25
Figura 28 Codificación de los niveles lógicos para el WS2812B.	25
Figura 29 Método de transmisión para el WS2812B.	26
Figura 30 Composición del mensaje para un LED organizado en 24 bits.....	26
Figura 31 Panel LDM-6432 - LUMEX.	27
Figura 32 Panel LMD-6432 - LUMEX.	27
Figura 33 LMD-12864 - LUMEX. Composición de los distintos paneles.	28
Figura 34 Ejemplo de matriz LED 8x8.	28
Figura 35 Conexiones de las matrices LED 8x8.	28
Figura 36 Matriz LED 8x8 con LEDs cuadrados.....	29
Figura 37. Adafruit Mini 8x8 LED Matrix. Parte delantera.	29
Figura 38. Adafruit Mini 8x8 LED Matrix. Parte trasera.	29
Figura 39. Adafruit Mini 8x8 LED Matrix. Placa sin matriz LED.	30
Figura 40 Detalle de la placa Adafruit Mini 8x8.	30
Figura 41. Diagrama de bloques del HT16K33.	30
Figura 42. Empaquetado HT16K33A 20 pines.....	31
Figura 43. Empaquetado HT16K33A 24 pines.....	31



Figura 44. Empaquetado HT16K33A 28 pines.....	31
Figura 45. Esquema de la placa de Adafruit Mini [60].	31
Figura 46. Comparativa de tamaño Small (izquierda) y Mini (derecha).	32
Figura 47. Adafruit 16x8 1.2" LED Matrix + Backpack.....	32
Figura 48. Adafruit 18x8 1.2" LED Matrix + Backpack. Distintas partes.....	32
Figura 49. Esquema de la placa Adafruit 16x8 1.2" LED Matrix + Backpack [60].....	33
Figura 50. Adafruit Bicolor LED Square Pixel Matrix. Funcionamiento con 3 colores.....	33
Figura 51. Adafruit Bicolor LED Square Pixel Matrix. Parte trasera.....	33
Figura 52. Adafruit Bicolor LED Square Pixel Matrix. Parte delantera.....	33
Figura 53. Adafruit Bicolor LED Square Pixel Matrix. Esquema de la placa. [60]	34
Figura 54. Aplicación típica del MAX7219/21.	34
Figura 55. Configuración de las patillas del CI MAX7219/21.	34
Figura 56. Módulo matriz LED basado en MAX7219 con CI en encapsulado DIP [61].....	35
Figura 57. Módulo matriz LED basado en MAX7219 con CI en encapsulado SO [62].....	35
Figura 58. Módulo matriz LED basado en MAX7219. 4 módulos [63].	35
Figura 59. Esquema de conexiones de la matriz LED basada en MAX7219.....	35
Figura 60. Características eléctricas MAX7219 [22].....	36
Figura 61. Resistencias limitadoras de corriente en MAX7219 [22].	36
Figura 62. Descripción detallada de los pines del MAX7219 [22].	37
Figura 63. Full Color RGB LED Matrix Driver Shield + RGB Matrix Screen – SUNFOUNDER. Conjunto shield y matriz RGB.....	37
Figura 64. Full Color RGB LED Matrix Driver Shield + RGB Matrix Screen – SUNFOUNDER. Montaje sobre Arduino UNO.	37
Figura 65. Full Color RGB LED Matrix Driver Shield + RGB Matrix Screen – SUNFOUNDER. Detalle placa de desarrollo y montaje de la matriz.	38
Figura 66 Full Color RGB LED Matrix Driver Shield + RGB Matrix Screen – SUNFOUNDER. Esquema eléctrico de la placa de desarrollo.....	38
Figura 67 Arduino Nano Every – ATMega4809. Vista frontal.	39
Figura 68 Arduino Nano Every – ATMega4809. Vista trasera.....	39
Figura 69 Arduino Nano Every – ATMega4809. Diagrama de árbol de potencia.	39
Figura 70 Arduino Nano Every – ATMega4809. Esquema de la alimentación.....	40
Figura 71 Arduino Nano Every – ATMega4809. Pines del microcontrolador.	41
Figura 72 Arduino Nano Every – ATMega4809. Pinout de la placa.....	41
Figura 73 Arduino Nano Every – ATMega4809. Esquema genérico de la placa.	42
Figura 74 Arduino Nano Every – ATMega4809. Detalle de los Level Shifters.....	42
Figura 75 Arduino Nano Every – ATMega4809. Detalle protección línea USB.	42
Figura 76 Arduino Leonardo – ATMega32u4. Parte frontal.....	43
Figura 77 Arduino Leonardo – ATMega32u4. Parte trasera.	43
Figura 78 Arduino Leonardo – ATMega32u4. Esquema 5V.	43
Figura 79 Arduino Leonardo – ATMega32u4. Auto selector de 5V.	44
Figura 80 Arduino Leonardo – ATMega32u4. Esquema alimentación por micro USB.	44
Figura 81 Arduino Leonardo – ATMega32u4. Pines del microcontrolador.....	45
Figura 82 Arduino Leonardo – ATMega32u4. Pines de la placa de desarrollo.	45
Figura 83 Arduino Leonardo – ATMega32u4. Pines ISCP de la placa de desarrollo.	46
Figura 84 Arduino Leonardo – ATMega32u4. Pines ISCP de la placa de desarrollo en el esquema.....	46



Figura 85 Arduino Leonardo – ATMega32u4. LEDs de la placa de desarrollo	46
Figura 86 Arduino Micro – ATMega32U4. Vista Frontal.	47
Figura 87 Arduino Micro – ATMega32U4. Seleccionador de 5V.	47
Figura 88 Arduino Nano – ATMega328. Vista Frontal.....	48
Figura 89 Arduino Nano – ATMega328. Vista Trasera.	48
Figura 90 Arduino Nano – ATMega328. Esquema interfaz USB a UART.....	48
Figura 91 Arduino Nano – ATMega328. Pinout de la placa de desarrollo.	48
Figura 92 ESP32 Series. Diagrama de bloques.	49
Figura 93 ESP32 Series. Tablas de resumen de consumo.	50
Figura 94 ESP32 Series. ESP32-WROOM-32.....	51
Figura 95 ESP32 Series. ESP32-WROOM-32. Esquema interno del chip.....	51
Figura 96 ESP32 Series. ESP32-WROOM-32. Esquemas de la placa de desarrollo.	52
Figura 97 ESP32 Series. ESP32-WROOM-32. Pinout.	53
Figura 98 ESP32-S2 Series. Comparación entre los distintos modelos.	53
Figura 99 ESP32-S2 Series. Diagrama de bloques.	54
Figura 100 ESP32-S2 Series. Consumo en modo activo.	54
Figura 101 ESP32-S2 Series. Consumo en modo de suspensión del módem.....	55
Figura 102 ESP32-S2 Series. Consumo en los modos ligero, profundo y apagado.	55
Figura 103 ESP32-S2 Series. ESP32-S2-DevkitM-1. Vista frontal y diagrama de bloques del encapsulado del µC.	56
Figura 104 ESP32-S2 Series. ESP32-S2-DevkitM-1U. Vista frontal y diagrama de bloques del encapsulado del µC.	56
Figura 105 ESP32-S2 Series. ESP32-S2-DevkitM-1. Pinout.	57
Figura 106 Series. ESP32-S2-DevkitM-1. Esquema del puente USB a UART.	58
Figura 107 ESP32-S2 Series. ESP32-S2-DevkitM-1. Esquema del microcontrolador.	58
Figura 108 ESP32-S2 Series. ESP32-S2-DevkitM-1. Esquema de la alimentación.	58
Figura 109 ESP32-S2 Series. ESP32-S2-DevKitC-1. Vista frontal.....	59
Figura 110 ESP32-S2 Series. ESP32-S2-DevKitC-1. Diagrama de bloques del sistema.	60
Figura 111 ESP32-S2 Series. ESP32-S2-DevKitC-1. Pinout.	60
Figura 112 ESP32-S3 Series. Comparación entre los distintos modelos.	61
Figura 113 ESP32-S3 Series. Diagrama de bloques.	61
Figura 114 ESP32-S3 Series. Consumo en modo activo.	62
Figura 115 ESP32-S3 Series. Consumo en modo de suspensión del módem.....	62
Figura 116 ESP32-S3 Series. Consumo en los modos ligero, profundo y apagado.	63
Figura 117 ESP32-S3 Series. Tabla resumen de las funcionalidades en los diferentes modos de operación.	63
Figura 118 ESP32-S3 Series. ESP32-S3-DevKitM-1. Diferencias entre los dos encapsulados del microcontrolador.	64
Figura 119 ESP32-S3 Series. ESP32-S3-DevKitM-1. Esquema eléctrico del encapsulado ESP32-S3-MINI-1.	64
Figura 120 ESP32-S3 Series. ESP32-S3-DevKitM-1. Pinout.	65
Figura 121 ESP32-S3 Series. ESP32-S3-DevKitC-1. Esquema interno del ESP32-S3-WROOM.	66
Figura 122 ESP32-S3 Series. ESP32-S3-DevKitC-1. Pinout.	66
Figura 123 Raspberry Pi Pico Series. Placas de desarrollo de la serie.....	67
Figura 124 Raspberry Pi Pico y Pico H pinout.....	68
Figura 125 Raspberry Pi Pico W y Pico WH pinout.....	68



Figura 126 Arduino Uno R4 WiFi. Vista frontal.	69
Figura 127 Arduino Uno R4 WiFi. Pinout.	69
Figura 128 Tabla resumen de las placas de desarrollo.	70
Figura 129 Membrana de botones.	71
Figura 130 Pantalla LCD.	71
Figura 131 Adafruit 3.2" TFT LCD touchscreen – ILI9341. Vista frontal.	72
Figura 132 Adafruit 3.2" TFT LCD touchscreen – ILI9341. Vista trasera.	72
Figura 133 Adafruit 3.2" TFT LCD touchscreen – ILI9341. Esquema de conexiones.	72
Figura 134 Adafruit 3.5" TFT LCD touchscreen – HXD8357D. Vista frontal.	73
Figura 135 Adafruit 3.5" TFT LCD touchscreen – HXD8357D. Vista trasera.	73
Figura 136 Adafruit 3.5" TFT LCD touchscreen – HXD8357D. Esquemas eléctricos.	74
Figura 137 Pantalla TFT SPI 480x320. 3.5".	74
Figura 138 LDR.	75
Figura 139 Divisor de voltaje empleando una LDR.	75
Figura 140 Motor paso a paso 28byj-48 con driver.	75
Figura 141 Motor CC JGA16-050-12120.	76
Figura 142 Doble puente H para el control de motores de CC.	76
Figura 143 Servomotor MG995.	77
Figura 144 Servomotor MG995. Conexiones.	77
Figura 145 Tabla resumen de consumos máximos del prototipo.	78
Figura 146 Esquema eléctrico típico del buck MAX20008EAFOD/VY+.	78
Figura 147 Placa con Buck DC-DC de 5A basado en XL4005.	79
Figura 148 Esquema placa Buck basada en XL4005.	79
Figura 149 Adaptador AC/DC marca RS Pro. Modelo 66JYH4Z-0500800-BY-RS.	79
Figura 150 Esquema de conexiones del prototipo para el panel de emergencia en carretera.	81
Figura 151 Esquema eléctrico de la placa de interconexión.	82
Figura 152 Unión de los módulos MAX7219 en forma de Z.	83
Figura 153 Unión de los módulos MAX7219 en forma escalonada.	83
Figura 154 Reloj de doble altura con tipo de hardware FC16_HW.	84
Figura 155 Reloj de doble altura con tipo de hardware: PAROLA_HW.	84
Figura 156 Tabla resumen de las funciones de la librería MD_Parola.	84
Figura 157 Tabla resumen de las funciones de la librería MD_Parola (continuación).	85
Figura 158 Tabla resumen de algunas funciones de la librería TFT_eSPI.	87
Figura 159 Prueba gráfica en la pantalla TFT. Patrón círculos.	88
Figura 160 Prueba gráfica en la pantalla TFT. Triángulo.	88
Figura 161 Prueba de texto en la pantalla TFT.	88
Figura 162 Prueba 1 función táctil de la pantalla TFT.	88
Figura 163 Prueba 2 función táctil de la pantalla TFT.	88
Figura 164 Prueba 3 función táctil de la pantalla TFT.	88
Figura 165 Configuración del programa GUIslicebuilder para la pantalla ILI9488.	89
Figura 166 Fichero de configuración Visual Studio Code.	91
Figura 167 Fuente en panel LED con doble altura.	91
Figura 168 Fuente en panel LED con simple altura.	91
Figura 169 Ejemplos de interfaz de usuario usando GUIslice.	92
Figura 170 Pantalla principal.	93
Figura 171 Pantalla del menú principal de la pantalla TFT.	95



Figura 172 Pantalla del submenú de mensajes de texto predeterminados de la pantalla TFT	95
Figura 173 Pantalla del submenú de efectos dinámicos de la pantalla TFT.	96
Figura 174 Pantalla del submenú de inclinación del panel LED.....	96
Figura 175 Pantalla del submenú de nivel de brillo del panel LED.	97
Figura 176 Pantalla del submenú de mensajes predeterminadas con una sola fila.....	97
Figura 177 Pantalla del submenú de mensajes personalizados en dos filas distintas.	98
Figura 178 Pantalla del submenú de ajustes de la pantalla TFT.	98
Figura 179 Tabla resumen del pseudoprotocolo de comunicación bluetooth del panel.	101
Figura 180 Icono de la aplicación Android.	103
Figura 181 Pantalla principal de la aplicación Android.	103
Figura 182 Pantalla de mensajes predeterminados de texto en la aplicación Android.....	103
Figura 183 Pantalla de efectos dinámicos predeterminados en la App de Android.....	104
Figura 184 Pantalla de inclinación del panel luminoso en la App de Android.....	104
Figura 185 Pantalla de iluminación del panel luminoso en la App de Android.	104
Figura 186 Pantalla de mensaje personalizado de una fila en la App de Android.....	105
Figura 187 Pantalla de mensaje personalizado de dos filas en la App de Android.....	105
Figura 188 Pantalla de ajustes bluetooth en la App de Android.....	105
Figura 189 Estructura del programa Android.....	106
Figura 190 Diseño de la PCB principal del prototipo.....	107
Figura 191 Diseño de la PCB auxiliar para las conexiones del panel LED.....	107
Figura 192 PCBs del prototipo con los componentes soldados.	108
Figura 193 Diseño de la envolvente del panel LED.	109
Figura 194 Diseño de la placa transparente del panel LED.....	109
Figura 195 Distintas piezas impresas para el panel.	110
Figura 196 Envolvente panel LED.	110
Figura 197 Envolvente del panel LED completo.....	110
Figura 198 Vista frontal y lateral del diseño del panel de control con el soporte.	111
Figura 199 Vista trasera y lateral del diseño del panel de control con el soporte.....	111
Figura 200 Vista frontal del panel de control sin montar en el soporte.	111
Figura 201 Vista trasera del panel de control sin la tapa.	111
Figura 202 Mecanismo piñón cremallera del panel luminoso.	112
Figura 203 Soporte de ángulo ajustable.	112
Figura 204 Montaje final del panel de emergencia en carretera - Vista frontal.....	113
Figura 205 Montaje final del panel de emergencia en carretera - Vista trasera.	113
Figura 206 Montaje final del panel de emergencia en carretera - Vista lateral.	113
Figura 207 Error producido al intentar mostrar la palabra TEXTO PRUEBA.	115
Figura 208 Resumen de costes físicos del panel de emergencia en carretera.	116
Figura 209 Resumen de costes del firmware del panel de emergencia en carretera.	116
Figura 210 Matriz de 8x8 LEDs de 5mm espaciados 1,5cm.	118
Figura 211 Pantalla formada por WS2812B.	119
Figura 212 ESP32-S3 SPI TFT con sensor táctil – Vista frontal.	119
Figura 213 ESP32-S3 SPI TFT con sensor táctil – Vista trasera.	119

Capítulo 1. INTRODUCCIÓN.

Capítulo 1.1. MOTIVACIÓN Y ORIGEN DEL PROYECTO.

Este proyecto surge de las prácticas en empresa realizadas para la asignatura de Métodos y Herramientas de Diseño Electrónico (MHDE) en el departamento de Electrónica Aplicada, donde se realizó el diseño electrónico y la fabricación del sistema de control de un panel luminoso de poca complejidad, el cual se puede observar en la Figura 1 y Figura 2. Este panel luminoso se basaba en ocho módulos distintos formados por recortes de tiras LED que se podían encender de forma independiente.

El control se basaba en la amplificación de señales almacenadas en una memoria EEPROM mediante una etapa de potencia que permitían el encendido individual de cada módulo. Para lograr efectos en el panel se empleaba un reloj y un contador que permitía variar los registros de la memoria y por tanto las señales que se amplificaban. Además, se permitía cambiar de tipo de mensaje empleando un pulsador que mediante otro contador modificaba los registros más altos de la memoria. Se permitía también variar la velocidad en la que se mostraban los distintos mensajes variando la frecuencia del reloj empleando un potenciómetro. En la Figura 3 se puede ver un ejemplo de funcionamiento con la placa electrónica realizada durante las prácticas.



Figura 1 Panel simple de ocho módulos.



Figura 2 Interior panel simple de ocho módulos.

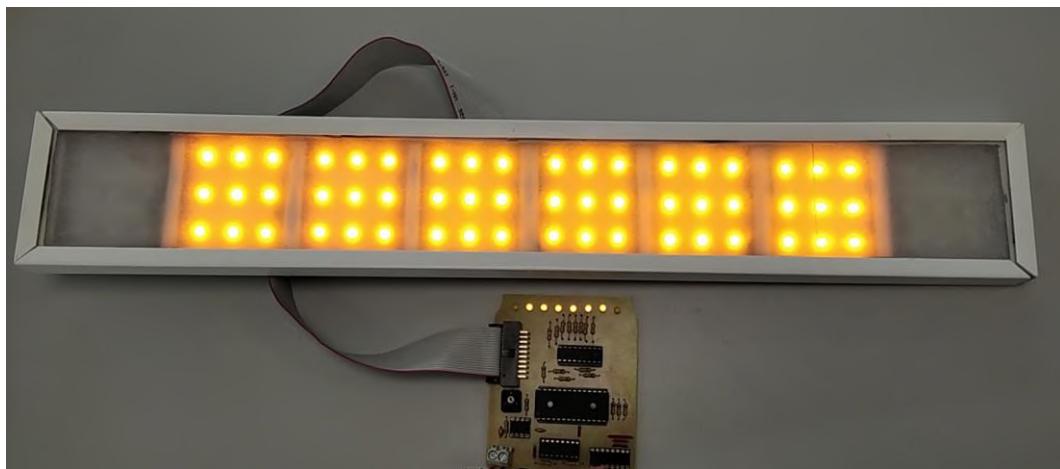


Figura 3 Ejemplo de funcionamiento del panel realizado en las prácticas de empresa.

Tras la fabricación y comprobación del funcionamiento del dispositivo anterior, se observaron grandes limitaciones sobre el prototipo como la escasa cantidad de mensajes que se pueden mostrar, la simpleza de dichos mensajes y el difícil manejo del prototipo a la hora de implementar el mensaje deseado al tratarse de una implementación secuencial, teniendo que pasar por cada uno de los distintos efectos hasta llegar al deseado, entre otras.

Por tanto, la motivación del proyecto es generar un panel de emergencia en carretera con la capacidad de mejorar las prestaciones del modelo anterior, empleando para ello un microcontrolador que dote al proyecto de mayor flexibilidad.

Capítulo 1.2. OBJETIVOS.

Capítulo 1.2.1. Objetivos generales.

El objetivo de este proyecto es realizar un panel de emergencia en carretera que mejore drásticamente el prototipo anterior, de forma que se puedan implementar señales más complejas y que pase de ser un simple panel de ocho módulos a una matriz capaz de mostrar texto y señales luminosas más complejas. Otro de los objetivos es mejorar la interfaz del usuario, ya que anteriormente solo se podían seleccionar diez señales distintas que se realizaba mediante un único pulsador, pasando secuencialmente por cada uno de los efectos hasta encontrar el deseado. Además, mediante esta interfaz se podrá cambiar la velocidad de cada efecto de forma más cómoda en vez de mediante un potenciómetro que variaba la frecuencia del reloj que controlaba los distintos estados.

En resumen, se pretende cambiar de un diseño basado en una electrónica analógica a un diseño más digital donde se puede conseguir un resultado más profesional y complejo pensado para mejorar la comunicación de situaciones de emergencia en la carretera, consiguiendo también una mejor interfaz de usuario y una mayor variedad de mensajes de emergencia.

Por tanto, dado que el prototipo está pensado para ser incorporado en el techo de un vehículo, deberá incorporar un circuito que permita alimentar al dispositivo mediante la propia batería del vehículo en el que vaya instalado y se deberá generar un primer diseño de envolvente que permita dicha instalación y transporte.

Capítulo 1.2.2. Objetivos específicos.

Los objetivos específicos de este proyecto serán:

- Implementación de mensajes de texto empleando para ello matrices de LEDs donde se podrán mostrar textos, caracteres o símbolos específicos que ayuden a transmitir un mensaje más claro de forma visual.
- Mejora en la interfaz de usuario para facilitar al operario la elección del tipo de mensaje que se desea mostrar por el panel, empleando para ello algún tipo de pantalla táctil.
- Implementar un control de luminosidad en el panel para poder variar el brillo dependiendo de la situación climática y luminosa de la carretera, contemplando posibles deslumbramientos.
- Se buscará también cumplir con algunas de las características más significativas que se encuentren durante el análisis de mercado que se realizará más adelante, con el fin de realizar un producto competitivo en el mercado.
- Implementación de una aplicación para móvil que permita controlar de forma inalámbrica el dispositivo.
- Además de la programación, se realizará un primer diseño de la envolvente para el producto con la intención de asegurar la integridad de las conexiones eléctricas y protegerlo de los golpes, además de facilitar así su transporte.

Capítulo 1.3. METODOLOGÍA.

Tras detallar los objetivos a los que se pretende llegar durante el desarrollo de este proyecto, se establecerá la metodología seguida para el desarrollo e investigación de las posibles soluciones que se pueden abarcar durante el desarrollo de este trabajo de fin de grado.

Inicialmente se realizará un estudio de mercado con la intención de obtener alguna idea más clara de lo implementado actualmente en un dispositivo de estas características. Se analizarán con detalle los distintos elementos, funcionalidades y modos de control de los modelos encontrados.

Tras el análisis de mercado, se buscarán diferentes elementos que permitan cumplir con las características más señaladas, buscando dispositivos para generar la matriz de LEDs que formarán el panel luminoso, la interfaz de usuario, el control del dispositivo y otros elementos auxiliares. Se realizará una selección de los diferentes dispositivos que

permitan cumplir con los objetivos específicos señalados anteriormente, así como el microcontrolador que llevará a cabo las tareas de control del prototipo.

Una vez obtenidos los distintos elementos se realizará la programación individual de cada dispositivo para comprender las distintas funcionalidades que pueden proporcionar al dispositivo final y sus distintas limitaciones.

Tras obtener los conocimientos necesarios sobre cada uno de sus módulos tanto en hardware como en software se realizará la programación en conjunto de los distintos elementos comprobando que no existan interferencias entre los diferentes dispositivos y sus librerías de programación. Para esta comprobación se realizará el montaje en algún sistema que permita hacer pruebas como placas de desarrollo, protoboards, placas de cobre perforadas, simuladores eléctricos, etc.

Si la comprobación no resulta exitosa se analizarán las causas de estas interferencias y errores y se buscará una solución para éstos, ya sea variando en software, hardware o el propio montaje del prototipo. Tras lograr el correcto funcionamiento se realizará la programación final del producto obteniendo así una primera versión que pueda ser implementada en el producto final, permitiendo ver las posibles mejoras que se puedan realizar en cuanto al montaje, fabricación y programación.

Tras conseguir el software prácticamente definitivo del producto se realizará el diseño y fabricación de la placa de circuito impreso (PCB) que garantizará las conexiones de los diferentes elementos, incluyendo elementos adicionales como pueden ser conectores, mangueras y otros elementos necesarios en la envolvente. A la par que el diseño electrónico de la PCB se realizará el diseño mecánico e impresión 3D cada elemento de la envolvente que sea necesario, realizando el montaje final con cada uno de los elementos.

Obtenido ya el software, el hardware y la envolvente que garanticen el correcto funcionamiento del panel de emergencia en carretera, se realizarán los ajustes necesarios en los elementos anteriores para corregir los detalles que se observen durante una etapa de pruebas de funcionamiento.

Una vez programado el microcontrolador y comprobado su funcionamiento se desarrollará una aplicación que permita controlar el panel luminoso de forma inalámbrica usando para ello un smartphone, realizando los cambios necesarios en la programación anterior.

Tras la validación del diseño se realizará un estudio del coste aproximado teniendo en cuenta costes de los distintos elementos del hardware, el coste de la programación y la viabilidad final del producto.

Una vez finalizado este proceso se analizarán las posibles mejoras y líneas futuras para este proyecto, quedando la posibilidad de aumentar la complejidad de este.

Capítulo 2. ANÁLISIS DEL HARDWARE.

Capítulo 2.1. ANÁLISIS DEL PROBLEMA.

Dado que el objetivo de este proyecto es la construcción de un panel que se instalará en un vehículo con la finalidad de mostrar mensajes al resto de usuarios, se debe plantear la forma en la que se transmite dicho mensaje, su modo de control y la interfaz de comunicación con el operario que controlará el panel de emergencia.

Dado que los mensajes deben ser legibles en situaciones de alta y baja luz ambiental se decide implementar la comunicación mediante señales luminosas, de tal forma que en el panel se puedan trazar distintos avisos en función de la situación en la vía y lo que un operario desee transmitir. Dado que la luz ambiental puede variar, se debe poder aumentar o disminuir la luminosidad del panel para evitar que el mensaje pase desapercibido o bien deslumbrar al resto de conductores. Por otra parte, el tamaño del panel debe ser suficiente para que se pueda percibir a larga distancia y además debe tener la suficiente resolución para que el mensaje a transmitir se pueda identificar fácilmente.

En cuanto al control del panel, este control se hará mediante algún tipo de microcontrolador que permita modificar las señales que se transmiten por el panel en tiempo real. Para seleccionar los mensajes o indicaciones que se mostrarán en el panel se deberá disponer de un dispositivo fijo en el interior del vehículo que, para dar robustez al producto, transmitirá la información seleccionada por el operario mediante conexiones eléctricas al controlador del dispositivo.

Además de esta interfaz de comunicación, se podrá plantear otro tipo de comunicaciones empleando algún tipo de software en otro dispositivo inteligente de tal forma que a distancia se pueda gobernar el panel luminoso. Por lo tanto, el microcontrolador debe disponer de interfaces para la comunicación inalámbrica de tal forma que su manejo se pueda realizar mediante otro dispositivo que se encuentre en las cercanías circuito de control.

Capítulo 2.2. ANÁLISIS DE PRODUCTOS EN EL MERCADO.

Dado que la meta del proyecto es la creación de un producto que pueda competir en el mercado actual en cuanto a la señalización en carretera se realizará un estudio de los productos ya existentes. El fin de este estudio será señalar las características y funciones que otros dispositivos poseen, de tal forma que el prototipo desarrollado se pueda hacer un lugar entre los posibles consumidores. A continuación, se desarrollarán las principales características de algunos de los dispositivos encontrados.

Capítulo 2.2.1. PANEL MENSAJE VARIABLE TEXTO – PROIN

Panel de tecnología led alfanumérico de una línea que muestra el mensaje que deseemos para informar del estado del tráfico o avisar de posibles peligros en la circulación. (Figura 4 y Figura 5).

Se puede colocar sobre cualquier tipo de vehículo, en la carretera o en una calle. Ideal para proporcionar información y avisos a los conductores en zonas de obras, en accidentes, en ciudad, en carretera, etc. [1].



Figura 4 PANEL MENSAJE VARIABLE TEXTO - PROIN.
Vista lateral.



Figura 5 PANEL MENSAJE VARIABLE TEXTO - PROIN. Vista frontal.

Como funciones presenta 75 mensajes básicos pregrabados con la posibilidad de insertar un número ilimitado (vía PDA). Permite elegir el tiempo entre mensajes, velocidad, intermitente o fijo, 100 niveles de intensidad luminosa de forma manual o automática.

Capítulo 2.2.2. PANEL MENSAJE VARIABLE GRÁFICO – PROIN.

Panel formado por leds rojos y ámbar para poder mostrar diferentes señales e informar del estado del tráfico o avisar de posibles peligros en la circulación. Se puede colocar en remolques, vehículos industriales, en la carretera o en una calle (Figura 6 y Figura 7).



Figura 6 PANEL MENSAJE VARIABLE GRÁFICO – PROIN.
Señal de estrechamiento izquierdo.



Figura 7 PANEL MENSAJE VARIABLE GRÁFICO - PROIN.
Señal de velocidad limitada a 80km/h.

Como funciones presenta 139 señales pregrabadas entre las que se permite elegir velocidad, intermitente o fijo y nivel de intensidad luminosa. [2]

Capítulo 2.2.3. Sistema Combinado Luminoso – SEBA.

Panel de Mensajes Variables para informar a los automovilistas (Figura 8). Panel de alta visibilidad aprobado en la norma EN 12966 con programación de hasta 100 mensajes, 12 caracteres por palabra con 3 palabras por mensaje mostrando el mensaje de forma fija, intermitente o deslizante [3].



Figura 8 Sistema Combinado Luminoso - SEBA.

Para el control se emplea una pantalla gráfica táctil con un menú de desplazamiento que indica los elementos de seguridad activos. Consta con opciones adicionales como focos rotativos, protección contra el viento, sistema de antirrobo y control en tableta o smartphone.

Capítulo 2.2.4. POLVIS TXT – SEBA

Se trata de un panel articulado (Figura 9 y Figura 10) de mensaje variable con cambio de señalización vial, desarrollado con colaboración de la policía alemana. El panel PolVis TXT puede ser programado con hasta 12 señales de tránsito mediante un panel de control (Figura 11 y Figura 12) o mensajes de texto que se complementan con dos focos azules o ámbar.

Los LEDs del panel están encapsulados individualmente en sus lentes para lograr una mayor claridad. Los colores de los LEDs del panel son rojo, ámbar, blanco y azul con un sistema anti-reflexión para evitar el efecto de luz fantasma, dificultando la creación de destellos o manchas [4].

El voltaje de alimentación del dispositivo es de 12V.

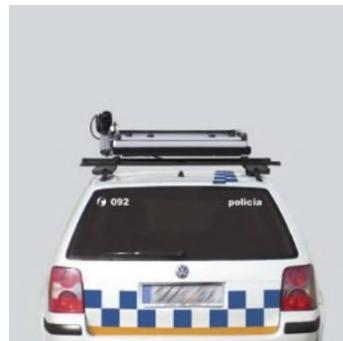


Figura 9 PolVis TXT - SEBA. Vista con el panel retraído.



Figura 10 PolVis TXT - SEBA. Vista con el panel desplegado.



Figura 11 PolVis TXT - SEBA. Panel de control.

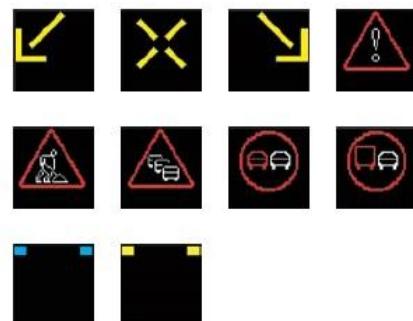


Figura 12 PolVisTXT - SEBA. Distintas señales.

Capítulo 2.2.5. Panel mensaje variable LED – TECNIVIAL.

Panel de mensaje variable alfanumérico con pictogramas (Figura 13 y Figura 14) con un tamaño de 1550x300x100mm para equiparlo en vehículos. Con posibilidad de incorporar cortavientos y sistema abatible. Peso de 14kg [5].



Figura 13 Panel de mensaje variable LED - TECNIVIAL.
Vista lateral.



Figura 14 Panel de mensaje variable LED - TECNIVIAL.
Vista frontal.

Se trata de una pantalla con tecnología LED de alto brillo con luz ámbar que permite visualizar mensajes con caracteres de espaciado proporcional en diferentes secuencias de texto y simulación de flechas bidireccionales. Los caracteres se muestran en tamaños de 24cm en una sola línea o bien en dos líneas de 10cm con 12 tipografías distintas.



Figura 15 Panel de mensaje variable LED - TECNIVIAL.
Detalle del panel.



Figura 16 Panel de mensaje variable LED - TECNIVIAL.
Controladora del producto.

La resolución del panel es de 96 por 16 pixeles, donde cada pixel está formado por 3 LEDs ámbar de 5mm de diámetro, formando un panel de 1536 LEDs (Figura 15). La intensidad luminosa del panel es de 4800mcd controlada de forma automática con una célula fotoeléctrica, con un ángulo de visión superior a 45°.

La alimentación del dispositivo se realiza a través de 12V DC con una potencia máxima de 250W y una media de 120W. Consumo medio de 10A/h.

Control del panel mediante botonera con display (Figura 16) o Software PC para puerto RS232, con más de 85 mensajes preprogramados.

Capítulo 2.2.6. Panel de mensaje variable LED hidráulico – TECNIVIAL.

Panel de mensaje LED de colores rojo y blanco, con mástil hidráulico fabricado en aluminio que permite recoger el panel (Figura 17 y Figura 18). [6] [7]



Figura 17 Panel de mensaje variable LED hidráulico - TECNIVIAL.
Vista de panel sin desplegar.



Figura 18 Panel de mensaje variable LED hidráulico -
TECNIVIAL. Vista del panel desplegado.



Figura 19 Panel de mensaje variable LED hidráulico - TECNIVIAL.
Señalización desvío carril izquierdo.



Figura 20 Panel de mensaje variable LED hidráulico -
TECNIVIAL. Señal de advertencia.

En la Figura 19 se pretende señalizar un desvío hacia la izquierda, en la que se emplean únicamente los LEDs blancos y en la Figura 20 se emplean ambos colores.

Capítulo 2.2.7. Letrero de desplazamiento LED – VEVOR.

El producto en cuestión es un letrero de desplazamiento LED RGB (Figura 21) con un tamaño de 100 x 20 cm (Figura 22). El letrero admite tres formas de control: a través de PC, unidad flash USB y teléfono móvil. Puede reproducir contenido editado desde una computadora o teléfono móvil mediante conexión de datos o wifi, o también puede almacenar el contenido en un dispositivo USB y luego reproducirlo [8].

El letrero LED ofrece varias funciones de pantalla, incluyendo 6 efectos de animación, 40 efectos para textos en movimiento y 39 niveles de velocidad. Esto permite mostrar textos dinámicos con diferentes efectos, como textos estáticos, de desplazamiento o parpadeantes. Tiene una resolución de 96 x 96 y cambios de color.

El precio del producto es de 85.99€.



Figura 21 Letrero de desplazamiento LED - VEVOR.
Letrero encendido.

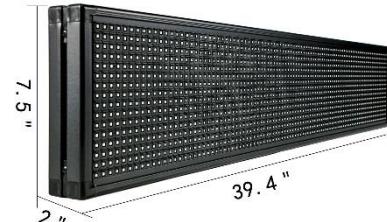


Figura 22 Letrero de desplazamiento LED - VEVOR.
Letrero apagado.

Capítulo 2.3. PRINCIPALES CARACTERÍSTICAS DE LOS MODELOS ESTUDIADOS.

Como se ha estudiado en el Capítulo 2.2 existen ciertos requisitos que debe tener el prototipo que se desea diseñar con el fin de lograr un producto competitivo en el mercado. Estos requisitos se basan principalmente en el panel que mostrará las indicaciones que se deseen y en el modo de control de este.

Comenzando por el panel, la resolución, el tamaño y ángulo de visión han de ser elevados para que se pueda percibir la señal desde grandes distancias. Existen en el mercado algunos paneles que permiten la señalización en varios colores para lograr enviar información gráfica más compleja y detallada. Además, la intensidad luminosa del panel debe poder configurarse para que se ajuste todas las situaciones posibles de la vía y evitar deslumbrar al resto de usuarios. Igualmente se deberá poder configurar el número de líneas de texto que se pueden mostrar en el panel, permitiendo mostrar mensajes más complejos en una misma pantalla.

Algunos de los productos incorporan un sistema que permite abatir el panel para reducir el rozamiento del aire a altas velocidades y evitar que el panel sufra desperfectos en los casos en los que no sea necesario que esté funcionando.

Otros dispositivos constan de rotativos que permiten dar presencia en todas direcciones del vehículo o del panel.

En cuanto al control, éste puede hacerse mediante dispositivos físicos conectados al controlador como una botonera o una pantalla táctil o empleando un smartphone o una tableta que permitan la conexión inalámbrica del dispositivo empleando bluetooth o wifi. De esta forma se deberá poder establecer el tipo de señal que se desea mostrar y la forma de mostrarse: fija, intermitente o deslizante, pudiendo configurar los tiempos y la intensidad luminosa de forma manual o automática. Hay equipos que contienen ciertos mensajes pregrabados que se podrán seleccionar y la posibilidad de implementar unos nuevos de forma cómoda mediante las interfaces del usuario.

La alimentación del dispositivo ha de ser compatible con la batería del propio vehículo. En algunos casos se menciona que el dispositivo funciona con una tensión de 12V y potencias de hasta 250W, pero existen vehículos cuya tensión nominal en la batería puede ser diferente.

Capítulo 2.4. ALTERNATIVAS PARA EL PANEL LUMINOSO.

Para la construcción del panel se parte de la necesidad de crear una matriz de puntos luminosos formados por LEDs que se permita controlar cada punto de manera individual.

Tras una investigación sobre las posibles soluciones que se pueden adoptar para la construcción del panel se obtienen los siguientes equipos, que podrían cumplir con las especificaciones descritas en el capítulo anterior.

Capítulo 2.4.1. Panel flexible basado en WS2812B.

Este modelo de panel se encuentra en formatos de 8x8, 16x16 y 8x32 LEDs RGB (Figura 23 y Figura 24), en un tamaño de 100x100, 170x170 y 330x100 milímetros respectivamente. Se alimenta con una tensión de 5V y consume aproximadamente 0,3W por cada píxel. El control del panel es externo y es compatible con plataformas tipo Arduino. Su precio oscila entre los 7 y 13€ dependiendo del tamaño [9].

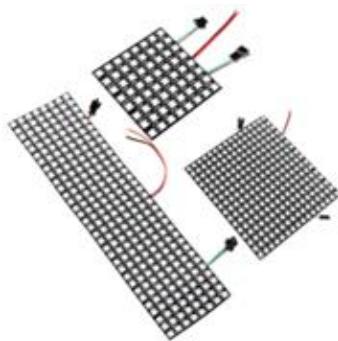


Figura 23 Paneles flexibles basados en WS2812B.
Tamaños disponibles.

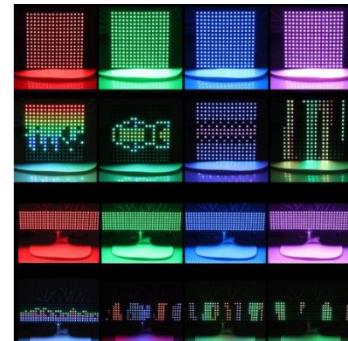


Figura 24 Paneles flexibles basados en WS2812B.
Ejemplos de configuración.

Se trata de un panel flexible basado en el componente WS2812B, una fuente de luz LED en la que el circuito de control y los LEDs RGB están incluidos en un empaquetado tipo 5050 (Figura 25 y Figura 26). Cada uno de los píxeles del panel puede tener de forma individual 256 niveles de brillo.



Figura 25 Componente WS2812B.

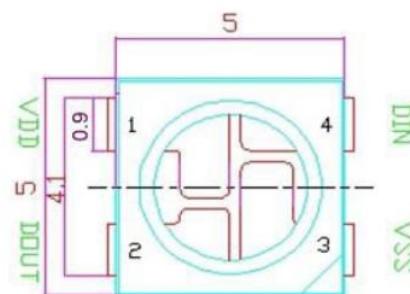


Figura 26 Empaquetado 5050 del WS2812B

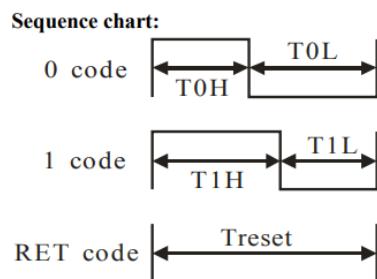
El propio encapsulado contiene a su vez los tres LEDs de colores primarios y un pequeño microcontrolador que permite variar la composición de la luz generada en este LED, permitiendo variar tanto el brillo como el color final. La comunicación con el exterior se realiza mediante uno de los pines del dispositivo que analiza la información recibida y la retransmite mediante otro pin, permitiendo así una comunicación serie. El protocolo que emplea para la comunicación es el protocolo NZR (Non Return to Zero) que permite almacenar los datos transmitidos por un controlador y transmitirlos por un pin de salida al resto de dispositivos que se conecten en cadena. Se trata de una comunicación a través de una sola línea en el que cada bit se representa por la duración de una tensión alta o baja. Un cero lógico es representado por un pulso corto, mientras que un uno lógico es representado por un pulso largo, tal y como se representa en la Figura 27.

Data transfer time(TH+TL=1.25μs±600ns)

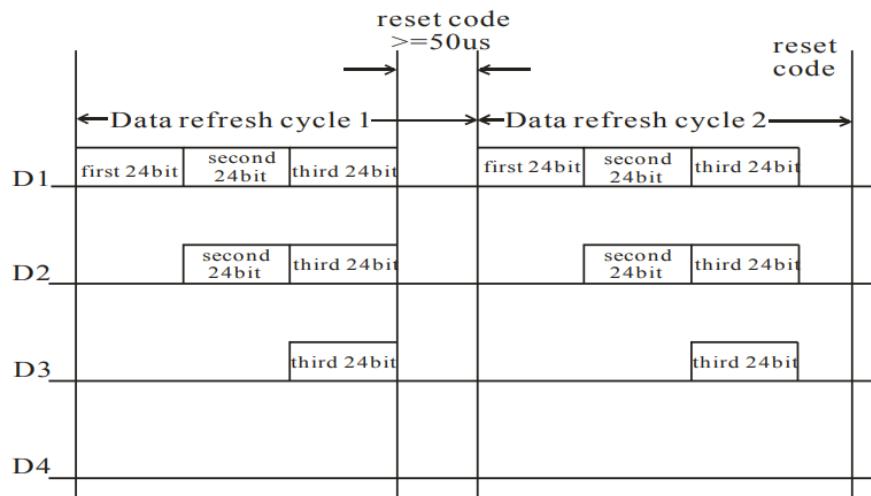
TOH	0 code ,high voltage time	0.4us	$\pm 150\text{ns}$
T1H	1 code ,high voltage time	0.85us	$\pm 150\text{ns}$
T0L	0 code , low voltage time	0.85us	$\pm 150\text{ns}$
T1L	1 code ,low voltage time	0.4us	$\pm 150\text{ns}$
RES	low voltage time	Above 50μs	

Figura 27 Tabla de tiempos para la transferencia de datos del WS2812B.

En este protocolo, típicamente un cero lógico es codificado cuando se produce un pulso de una duración de en torno a 400-500ns para la parte alta de tensión y de 800-900ns para la parte baja, mientras que un uno lógico es codificado con la configuración contraria, una duración de entre 800-900ns para la parte alta y entre 400-500ns para la parte baja como se observa en la Figura 28.


Figura 28 Codificación de los niveles lógicos para el WS2812B.

El formato de los datos para el WS2812B se basa en una serie de pulsos, empezando por una señal de reset y seguido por la información de cada dispositivo en la cadena de forma secuencial, como se indica en la Figura 29. La información consiste normalmente en una secuencia de 24bits por dispositivo en la que se representa el color deseado y su nivel de brillo.


Data transmission method:

Figura 29 Método de transmisión para el WS2812B.
Composition of 24bit data:

G7	G6	G5	G4	G3	G2	G1	G0	R7	R6	R5	R4	R3	R2	R1	R0	B7	B6	B5	B4	B3	B2	B1	B0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Note: Follow the order of GRB to sent data and the high bit sent at first.

Figura 30 Composición del mensaje para un LED organizado en 24 bits.

El orden de transmisión empieza por el bit más significativo (MSB) y progresiva hacia el menos significativo (LSB) donde típicamente se envía el brillo de cada led en el orden de verde, rojo y azul como se indica en la Figura 30.

Para el control de estas matrices de LEDs se pueden emplear librerías como FastLED [10] o Adafruit NeoPixel [11], compatibles con plataformas como Arduino y ESP32.

Capítulo 2.4.2. LDM-6432 – LUMEX.

Existen en el mercado paneles de matrices LED que ya implementan parte del control en su estructura interna. Es el caso del LDM-6432 de LUMEX, una matriz de 64 por 32 LEDs RGB (Figura 31) alimentado a 5V que incorpora control por bluetooth y por puerto serie o UART [12]. Su precio ronda los 97€ [13] y tiene unas dimensiones de 192x96x34mm (Figura 32).

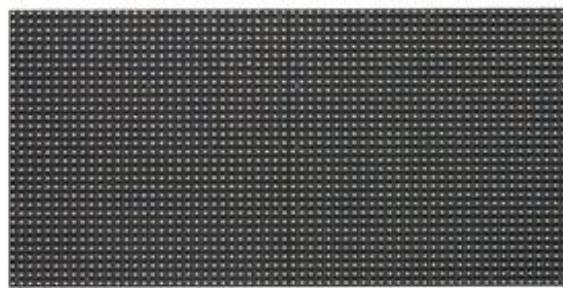


Figura 31 Panel LDM-6432 - LUMEX.

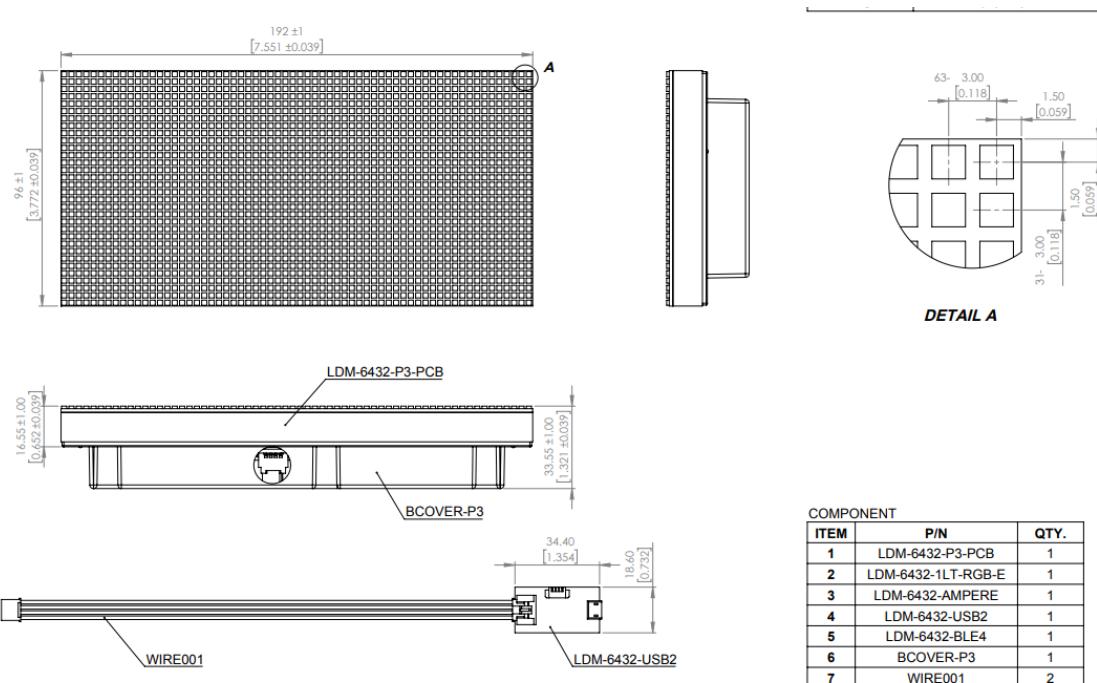


Figura 32 Panel LMD-6432 - LUMEX.

Del propio fabricante existen distintos tamaños y opciones como el LDM-12864 [14], un panel de 128 por 64 LEDs que se forma juntando varios paneles LDM-6432 (Figura 33) empleando para ello conectores que poseen los paneles de menor tamaño en su parte trasera.

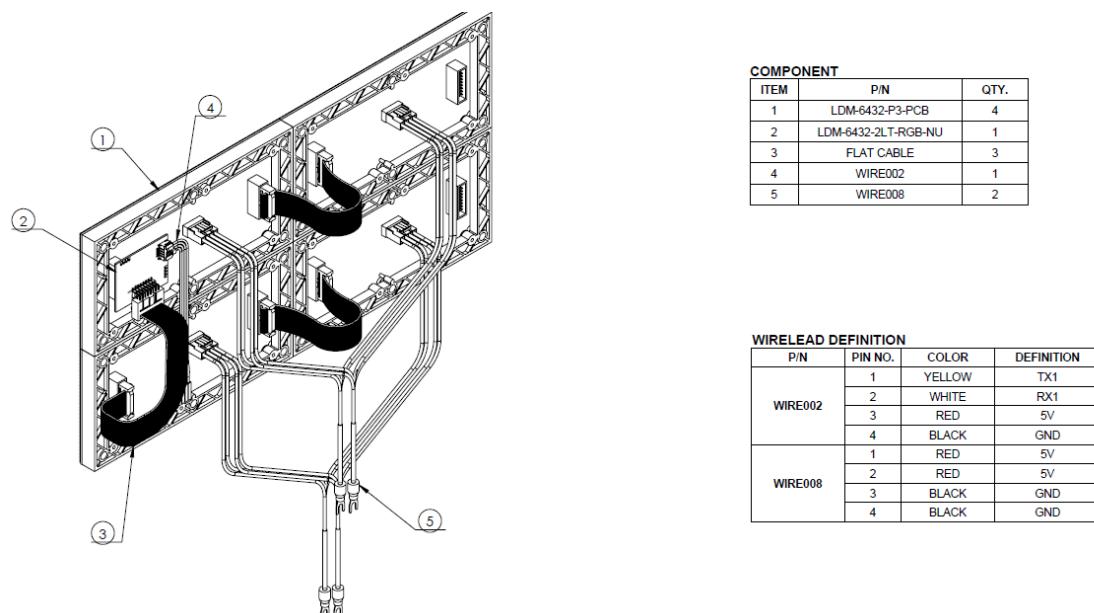


Figura 33 LMD-12864 - LUMEX. Composición de los distintos paneles.

Si bien su precio es elevado, se pueden encontrar modelos similares de otros fabricantes con un precio muy inferior al anterior (13€) [15].

Capítulo 2.4.3. LED Matrix – Adafruit.

Como alternativas a pequeña escala existen distintos modelos de la empresa Adafruit [16]. Estos modelos se basan en matrices LED que mediante técnicas de multiplexación permiten encender los distintos LEDs eligiendo la columna y fila adecuados (Figura 34). Existen dos tipos de matrices, las de ánodo común y las de cátodo común (Figura 35).

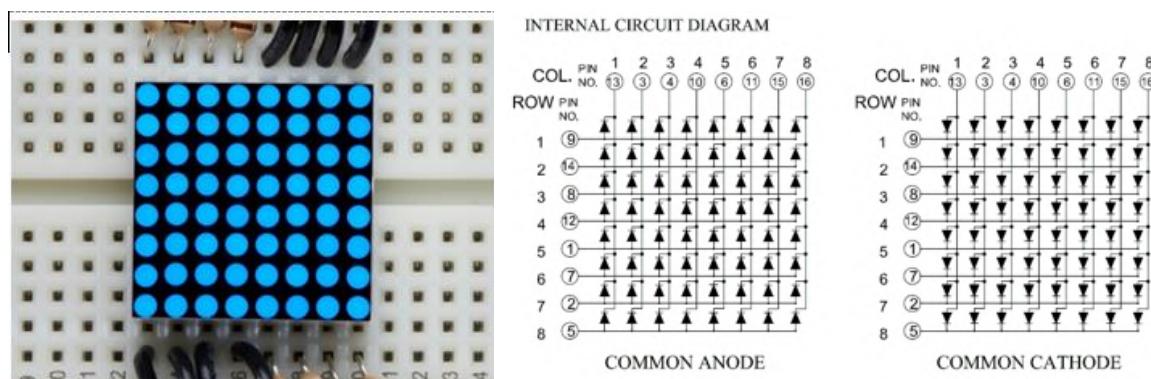


Figura 34 Ejemplo de matriz LED 8x8.

Figura 35 Conexiones de las matrices LED 8x8.

Estas matrices podrán estar dispuestas en forma de puntos o bien en forma de cuadrados (Figura 36) y en multitud de colores. Entre ellos se pueden encontrar matrices de alta intensidad luminosa (modelos Ultra Bright).

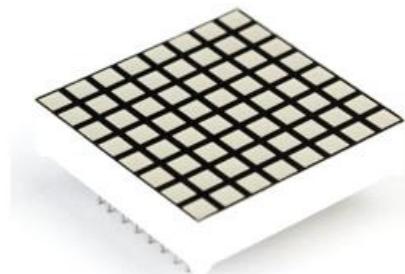


Figura 36 Matriz LED 8x8 con LEDs cuadrados.

El brillo de las matrices se podrá controlar de forma genérica empleando resistencias para variar la corriente que circula por cada una de las columnas o filas, pero los modelos que se estudiarán a continuación permitirán regular esta corriente con un único circuito integrado.

Algunos de los modelos del fabricante Adafruit Industries son:

Adafruit Mini 8x8 LED Matrix w/I2C Backpack:

Se trata de una matriz LED de 8 filas y 8 columnas de un único color (Figura 37) controlada por un circuito integrado que permite controlar la matriz empleando únicamente cuatro conexiones, dos de alimentación y dos de datos (Figura 38, Figura 39 y Figura 40).

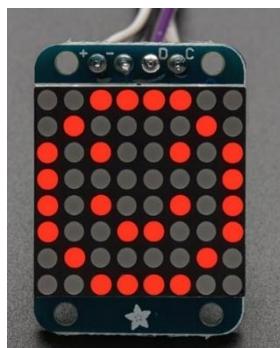


Figura 37. Adafruit Mini 8x8 LED Matrix. Parte delantera.

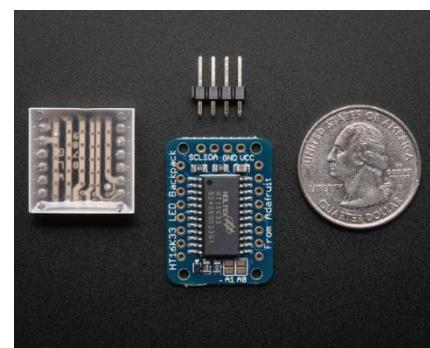


Figura 38. Adafruit Mini 8x8 LED Matrix. Parte trasera.

El circuito integrado empleado para el control es el HT16K33 [17] (Figura 41), un controlador multifunción LED cuyo máximo número LEDs que puede controlar es de 128 (16 filas y 8 columnas) pudiendo controlar la corriente y por tanto el brillo de los LEDs. El CI es compatible con la mayoría de los microcontroladores y se comunica mediante un bus bidireccional I2C. El tamaño del producto es de 20x28x4mm [18].

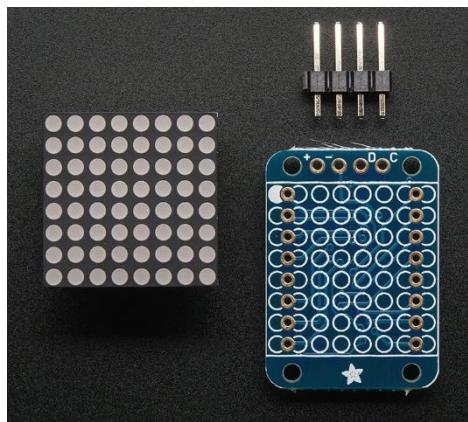


Figura 39. Adafruit Mini 8x8 LED Matrix. Placa sin matriz LED.



Figura 40 Detalle de la placa Adafruit Mini 8x8.

Block Diagram

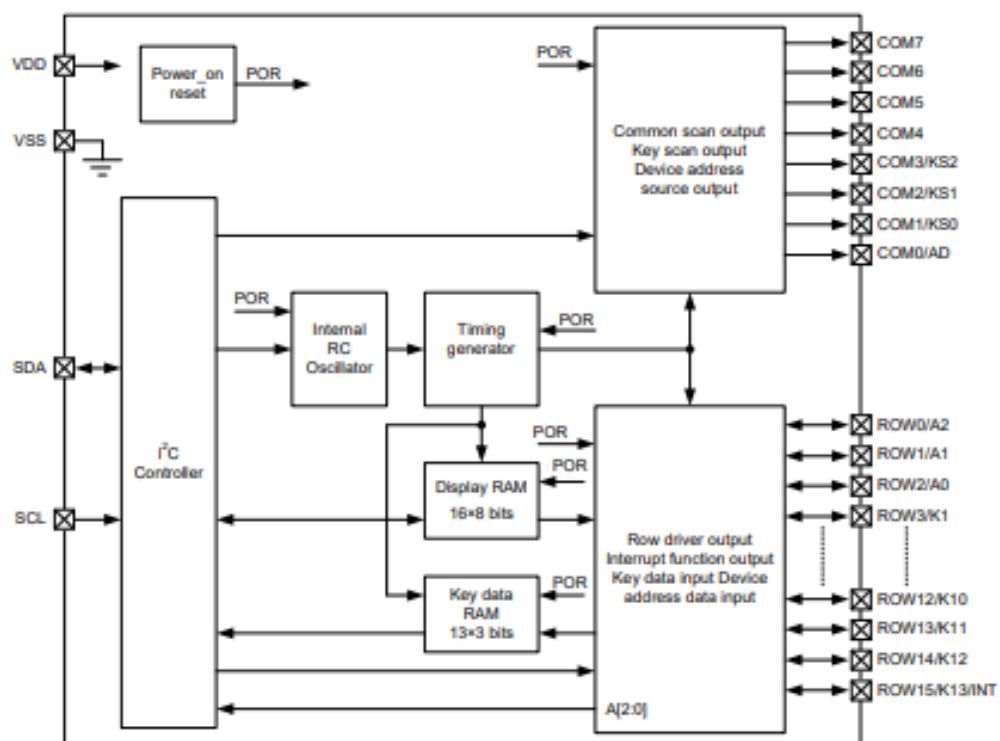


Figura 41. Diagrama de bloques del HT16K33.

El chip está disponible en tres formatos distintos en los que variará el número de pines dependiendo del tamaño de la matriz de LEDs que se desee controlar (Figura 42, Figura 43 y Figura 44).

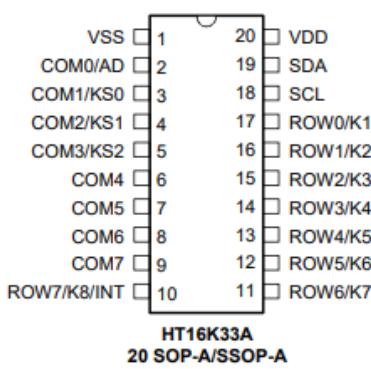


Figura 42. Empaquetado HT16K33A 20 pines.

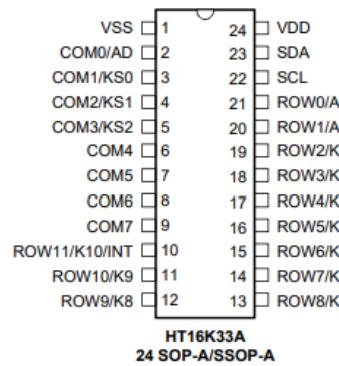


Figura 43. Empaquetado HT16K33A 24 pines.

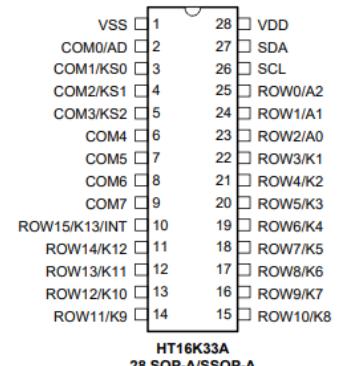


Figura 44. Empaquetado HT16K33A 28 pines.

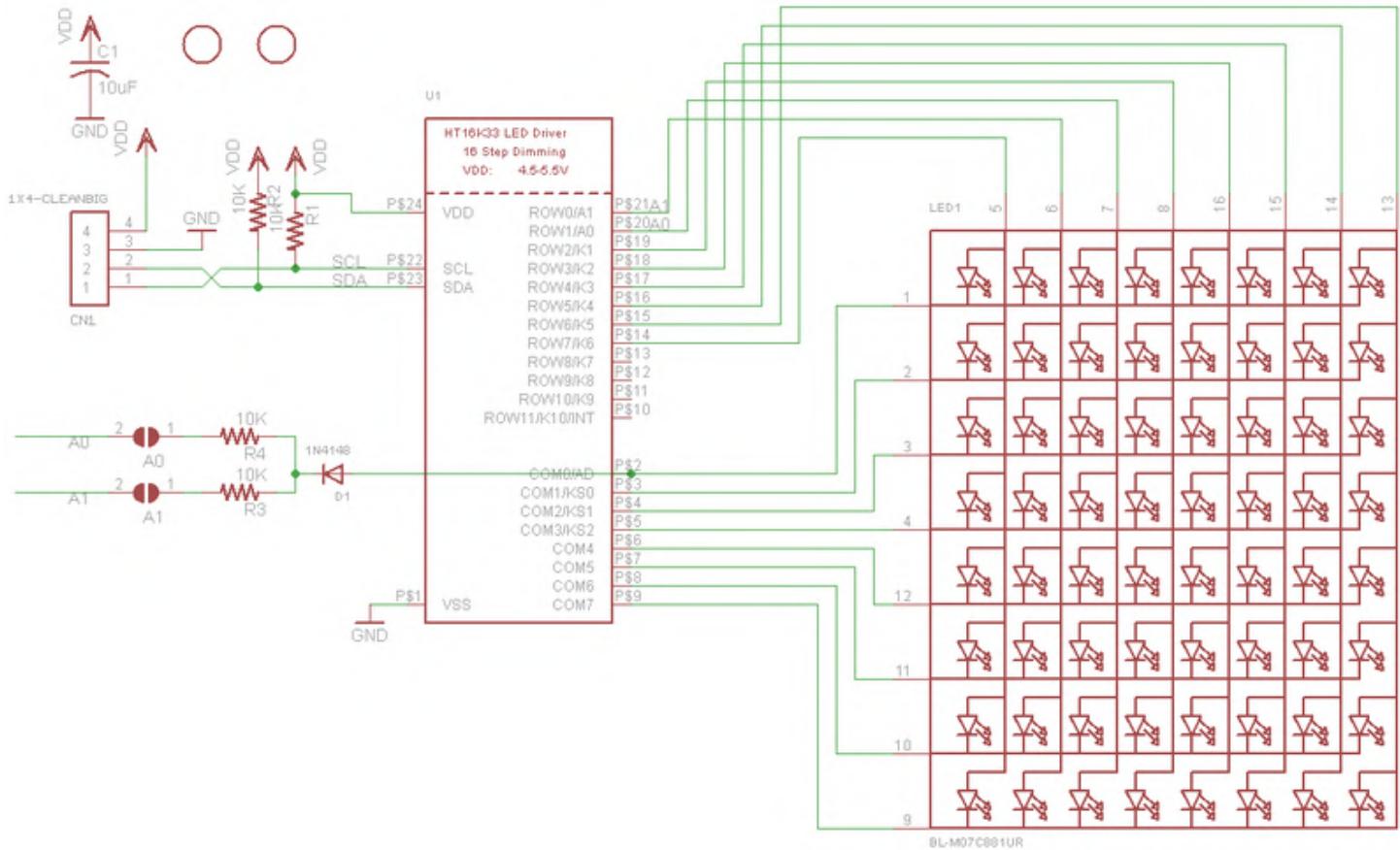


Figura 45. Esquema de la placa de Adafruit Mini [60].

Adafruit Small 1.2" 8x8 LED Matrix w/I2C Backpack:

Se trata de un modelo similar al anterior, pero con un tamaño superior (Figura 46) y un jumper adicional para la selección de la dirección I2C, por lo que se permite conectar hasta ocho dispositivos [19]. Este modelo permite mayor conexión de dispositivos ya que se está usando el controlador HT16K33 de 28 pines, mientras que en el modelo Mini se usa el de 24.

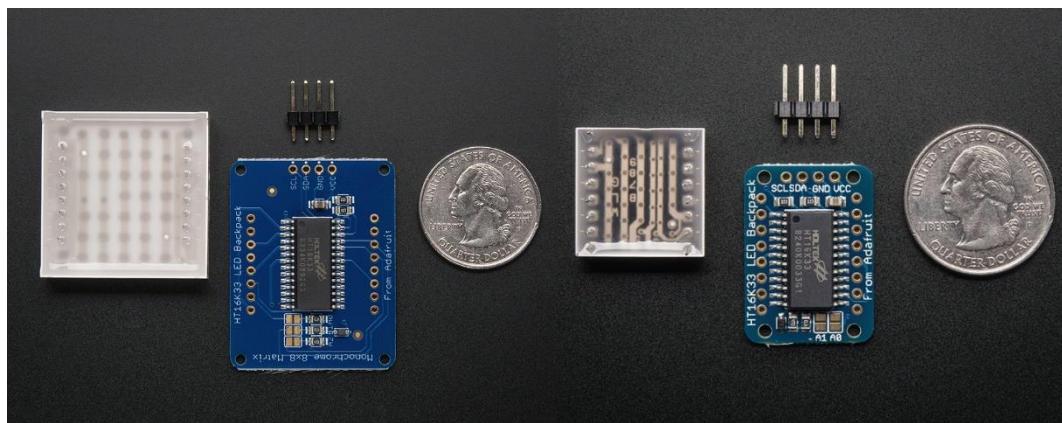


Figura 46. Comparativa de tamaño Small (izquierda) y Mini (derecha).

Adafruit 16x8 1.2" LED Matrix + Backpack:

Se trata de una variante del modelo Small en la que se sitúan en la placa dos matrices LED de 8x8 (Figura 47 y Figura 48), formando una matriz de mayores dimensiones [20]. Para esto se emplea el mismo circuito integrado, pero se conectan todas las patillas del CI (HT16K33 28 pines) como se observa en el esquema de la Figura 49. También permite la conexión de hasta 8 dispositivos, pudiendo formar un panel de 64x16 LEDs.

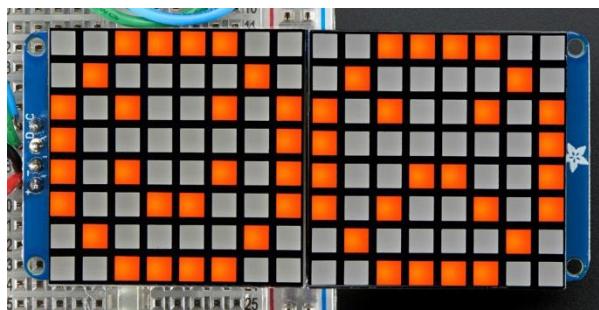


Figura 47. Adafruit 16x8 1.2" LED Matrix + Backpack.

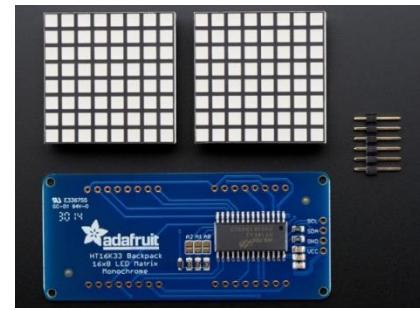


Figura 48. Adafruit 18x8 1.2" LED Matrix + Backpack. Distintas partes.

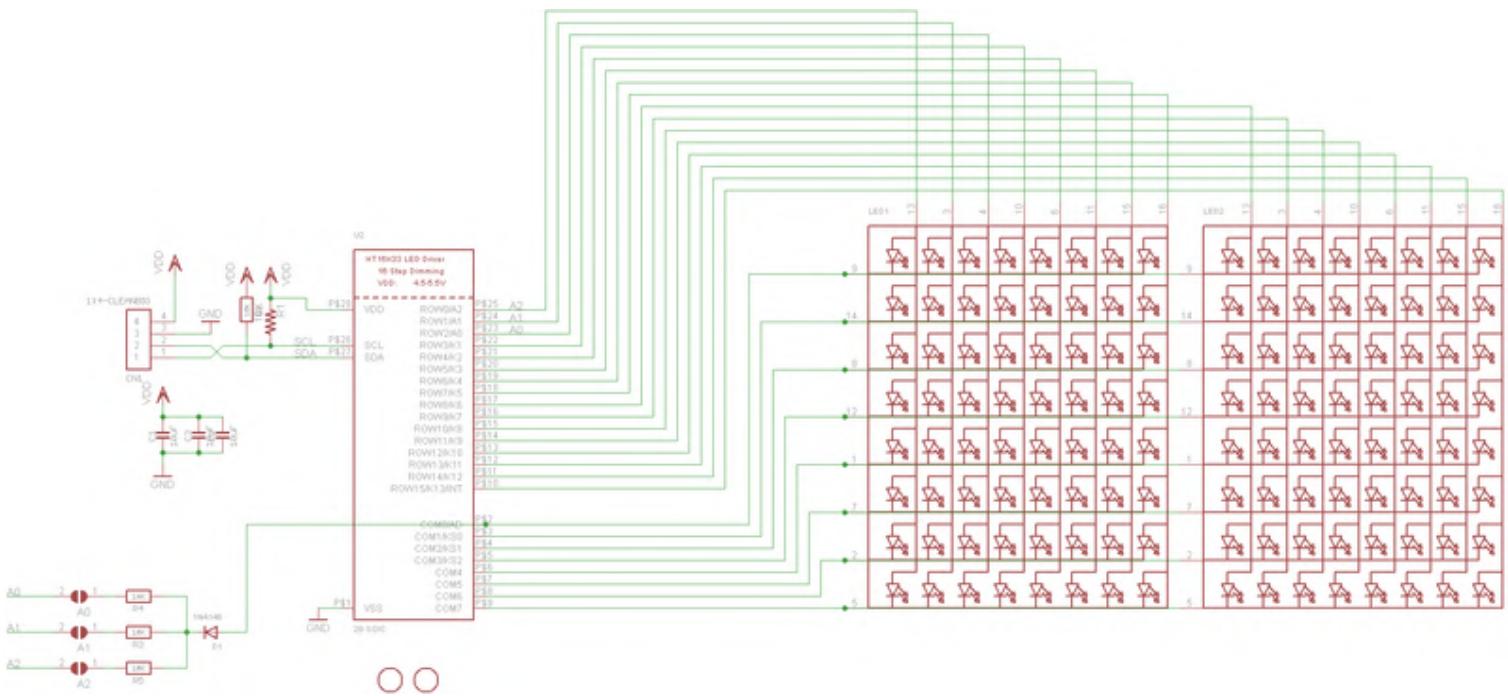


Figura 49. Esquema de la placa Adafruit 16x8 1.2" LED Matrix + Backpack [60].

Adafruit Bicolor LED Square Pixel Matrix with I2C Backpack – Qwiic / STEMMA QT:

Es otra variación del modelo Small 1.2" 8x8 donde se usan todos los pines del circuito integrado para controlar una única matriz que contiene en su interior dos LEDs por píxel. De esta forma mediante dos pines para cada columna y uno para cada fila se pueden obtener cuatro estados distintos del píxel en cuestión: apagado, verde, rojo y la fusión de los dos colores, amarillo como se ve en la Figura 50 [21]. Más detalles sobre la placa de este modelo se encuentran en la Figura 51 y Figura 52. El esquema de conexión interno de la matriz es el mostrado en la Figura 53.

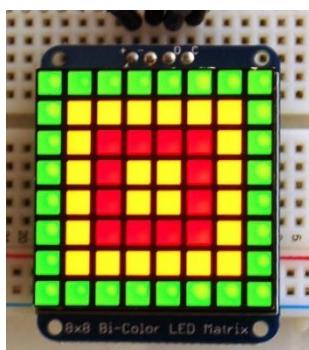


Figura 50. Adafruit Bicolor LED Square Pixel Matrix. Funcionamiento con 3 colores.

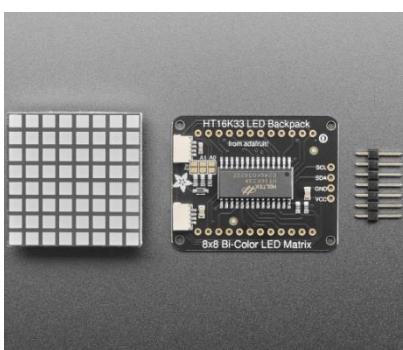


Figura 51. Adafruit Bicolor LED Square Pixel Matrix. Parte trasera.

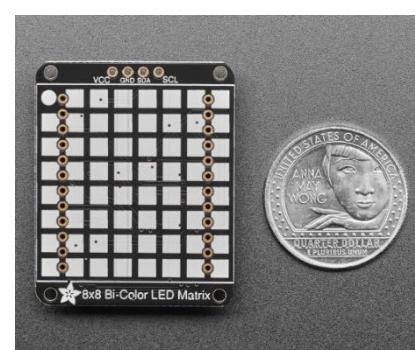


Figura 52. Adafruit Bicolor LED Square Pixel Matrix. Parte delantera.

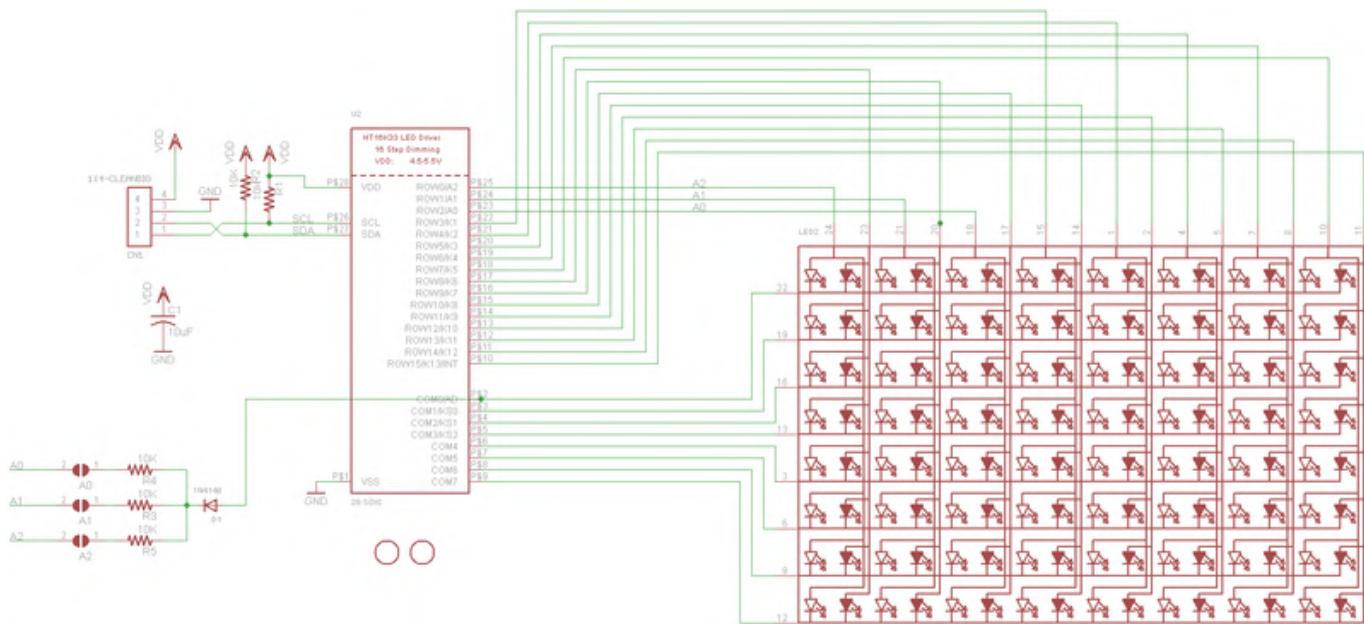


Figura 53. Adafruit Bicolor LED Square Pixel Matrix. Esquema de la placa. [60]

Capítulo 2.4.4. Panel basado en MAX7219/21.

Los circuitos integrados MAX7219 y MAX7221 son unos controladores para displays de siete segmentos (hasta 8 dígitos) de cátodo común que mediante comunicación SPI permite controlar el encendido de los distintos LEDs empleando un microprocesador (Figura 54). También es posible utilizarlo para displays gráficos en forma de barra y para controlar matrices LED de hasta 64 LEDs individuales [22].

Incluido en el CI se encuentra un decodificador BCD para siete segmentos y una memoria RAM estática de 8x8 para almacenar cada dígito. Se necesita una resistencia externa para establecer la corriente de todos los LEDs.

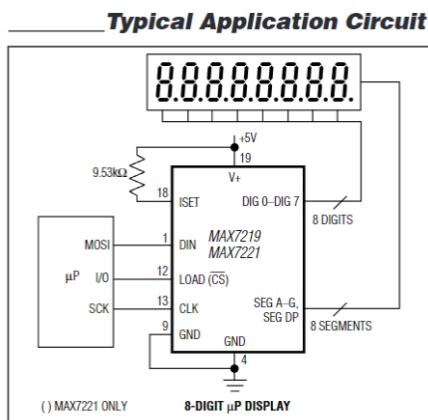


Figura 54. Aplicación típica del MAX7219/21.

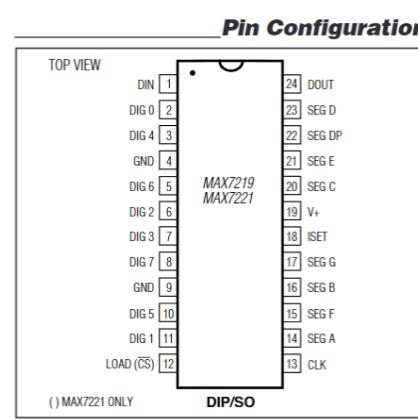


Figura 55. Configuración de las patillas del CI MAX7219/21.

Algunos de los modelos encontrados en el mercado que se basan en este circuito integrado son los mostrados en la Figura 56, Figura 57 y Figura 58. El esquema de conexión de las placas de desarrollo de estos equipos se observa en la Figura 59.

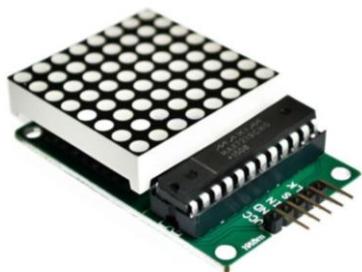


Figura 56. Módulo matriz LED basado en MAX7219 con CI en encapsulado DIP [61].

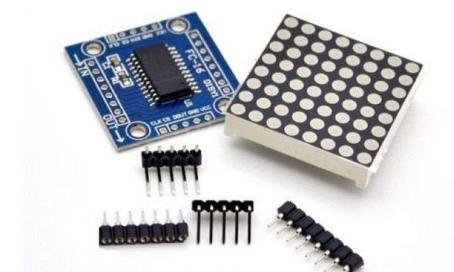


Figura 57. Módulo matriz LED basado en MAX7219 con CI en encapsulado SO [62].

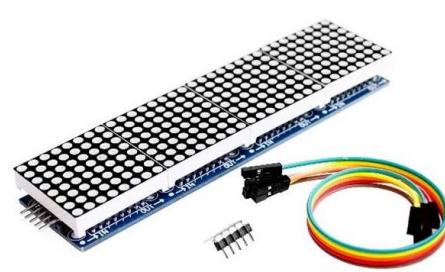


Figura 58. Módulo matriz LED basado en MAX7219.
4 módulos [63].

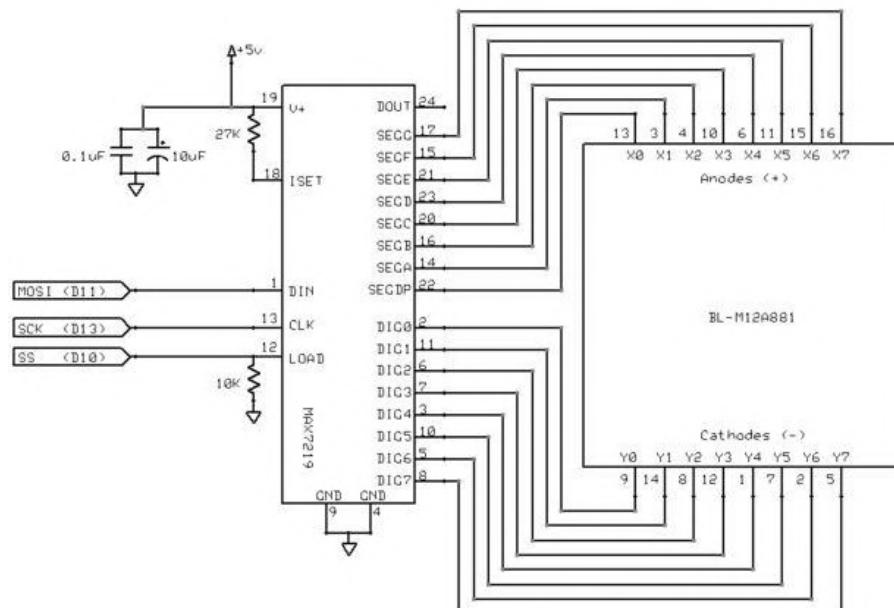


Figura 59. Esquema de conexiones de la matriz LED basada en MAX7219.

Según las especificaciones eléctricas del componente, se ha de conectar a una tensión de entre 4 y 5.5V, y la corriente por cada segmento no puede superar los 45mA (Figura 60). Para establecer esta corriente se seguirá la tabla de la Figura 61, donde en función de la corriente deseada y del voltaje de operación de la matriz LED que se conecte se deberá seleccionar una resistencia u otra para conectarla entre los pines V+ y ISET.


Electrical Characteristics

($V_+ = 5V \pm 10\%$, $R_{SET} = 9.53k\Omega \pm 1\%$, $T_A = T_{MIN}$ to T_{MAX} , unless otherwise noted.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Operating Supply Voltage	V_+		4.0	5.5		V
Shutdown Supply Current	I_+	All digital inputs at V_+ or GND, $T_A = +25^\circ C$		150		μA
Operating Supply Current	I_+	R_{SET} = open circuit		8		
		All segments and decimal point on, $I_{SEG_} = -40mA$		330		mA
Display Scan Rate	f_{OSC}	8 digits scanned	500	800	1300	Hz
Digit Drive Sink Current	I_{DIGIT}	$V_+ = 5V$, $V_{OUT} = 0.65V$	320			mA
Segment Drive Source Current	I_{SEG}	$T_A = +25^\circ C$, $V_+ = 5V$, $V_{OUT} = (V_+ - 1V)$	-30	-40	-45	mA
Segment Current Slew Rate (MAX7221 only)	$\Delta I_{SEG}/\Delta t$	$T_A = +25^\circ C$, $V_+ = 5V$, $V_{OUT} = (V_+ - 1V)$	10	20	50	$mA/\mu s$
Segment Drive Current Matching	ΔI_{SEG}			3.0		%
Digit Drive Leakage (MAX7221 only)	I_{DIGIT}	Digit off, $V_{DIGIT} = V_+$			-10	μA
Segment Drive Leakage (MAX7221 only)	I_{SEG}	Segment off, $V_{SEG} = 0V$		1		μA
Digit Drive Source Current (MAX7219 only)	I_{DIGIT}	Digit off, $V_{DIGIT} = (V_+ - 0.3V)$		-2		mA
Segment Drive Sink Current (MAX7219 only)	I_{SEG}	Segment off, $V_{SEG} = 0.3V$	5			mA

Figura 60. Características eléctricas MAX7219 [22].
Table 11. R_{SET} vs. Segment Current and LED Forward Voltage

I_{SEG} (mA)	V_{LED} (V)				
	1.5	2.0	2.5	3.0	3.5
40	12.2	11.8	11.0	10.6	9.69
30	17.8	17.1	15.8	15.0	14.0
20	29.8	28.0	25.9	24.5	22.6
10	66.7	63.7	59.3	55.4	51.2

Note: R_{SET} values are in Kilo Ohms ($k\Omega$)

Figura 61. Resistencias limitadoras de corriente en MAX7219 [22].

Una descripción más detallada de los pines se encuentra en la Figura 62, donde el pin 1 es la entrada de datos, los pines 2,3,5,6,7,8,10 y 11 se deben conectar a los cátodos de los LEDs, dos pines de masa 4 y 9, el pin 12 para la carga de datos o el chip select dependiendo del modelo, el 13 para la señal de reloj, los pines 14, 15, 16, 17, 20 y 23 para los ánodos de los LEDs, el pin 18 de ajuste la corriente, el pin 19 para la alimentación a 5V del dispositivo y el pin 24 será la salida de datos. En este último pin se mostrará la información recibida en el pin de entrada después de 16.5 ciclos de la señal de reloj, permitiendo conectar en cadena varios MAX7219/21.

Pin Description

PIN	NAME	FUNCTION
1	DIN	Serial-Data Input. Data is loaded into the internal 16-bit shift register on CLK's rising edge.
2, 3, 5-8, 10, 11	DIG 0-DIG 7	Eight-digit drive lines that sink current from the display common cathode. The MAX7219 pulls the digit outputs to V+ when turned off. The MAX7221's digit drivers are high-impedance when turned off.
4, 9	GND	Ground. Both GND pins must be connected.
12	LOAD (MAX7219)	Load-Data Input. The last 16 bits of serial data are latched on LOAD's rising edge.
	CS (MAX7221)	Chip-Select Input. Serial data is loaded into the shift register while CS is low. The last 16 bits of serial data are latched on CS's rising edge.
13	CLK	Serial-Clock Input. 10MHz maximum rate. On CLK's rising edge, data is shifted into the internal shift register. On CLK's falling edge, data is clocked out of DOUT. On the MAX7221, the CLK input is active only while CS is low.
14-17, 20-23	SEG A-SEG G, DP	Seven Segment Drives and Decimal Point Drive that source current to the display. On the MAX7219, when a segment driver is turned off it is pulled to GND. The MAX7221 segment drivers are high-impedance when turned off.
18	ISET	Connect to V _{DD} through a resistor (R_{ISET}) to set the peak segment current (Refer to Selecting R_{ISET} Resistor and Using External Drivers section).
19	V+	Positive Supply Voltage. Connect to +5V.
24	DOUT	Serial-Data Output. The data into DIN is valid at DOUT 16.5 clock cycles later. This pin is used to daisy-chain several MAX7219/MAX7221's and is never high-impedance.

Figura 62. Descripción detallada de los pines del MAX7219 [22].

Hay que añadir que este dispositivo solo permite controlar matrices LED monocromáticas, aunque permite generar paneles de grandes dimensiones gracias a la comunicación serie y al pin DOUT del que dispone el circuito integrado.

Capítulo 2.4.5. Full Color RGB LED Matrix Driver Shield + RGB Matrix Screen – SUNFOUNDER.

Existe una placa de desarrollo de la empresa SUNFOUNDER que permite controlar una matriz de 8x8 LEDs RGB conectándolo directamente a Arduino [23]. Algunas imágenes sobre este dispositivo se muestran en la Figura 63, Figura 64 y Figura 65.

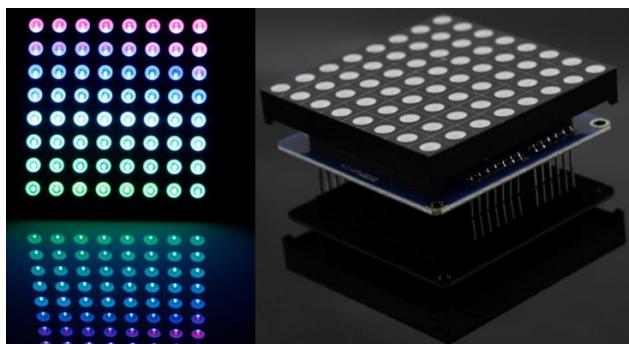


Figura 63. Full Color RGB LED Matrix Driver Shield + RGB Matrix Screen – SUNFOUNDER. Conjunto shield y matriz RGB.

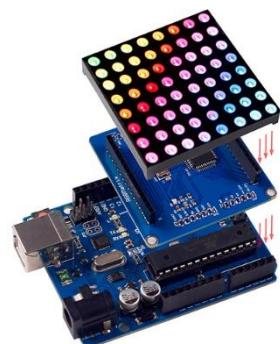


Figura 64. Full Color RGB LED Matrix Driver Shield + RGB Matrix Screen – SUNFOUNDER. Montaje sobre Arduino UNO.

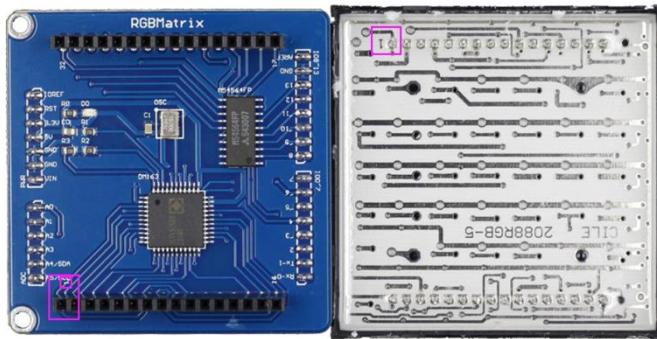


Figura 65. Full Color RGB LED Matrix Driver Shield + RGB Matrix Screen – SUNFOUNDER.
Detalle placa de desarrollo y montaje de la matriz.

Este dispositivo se basa principalmente en el controlador DM163 [24], un controlador para LEDs que consta de registros de desplazamiento, 8x3 canales de corriente seleccionable con tres resistencias externas y 64x256 niveles de PWM para controlar la intensidad lumínica de los LEDs de forma digital. Cada canal puede proporcionar hasta 60mA. El CI M54564FP [25] que se observa en el esquema de la placa de desarrollo (Figura 66) es un conjunto de transistores Darlington que sirven para alimentar los ánodos de la matriz RGB.

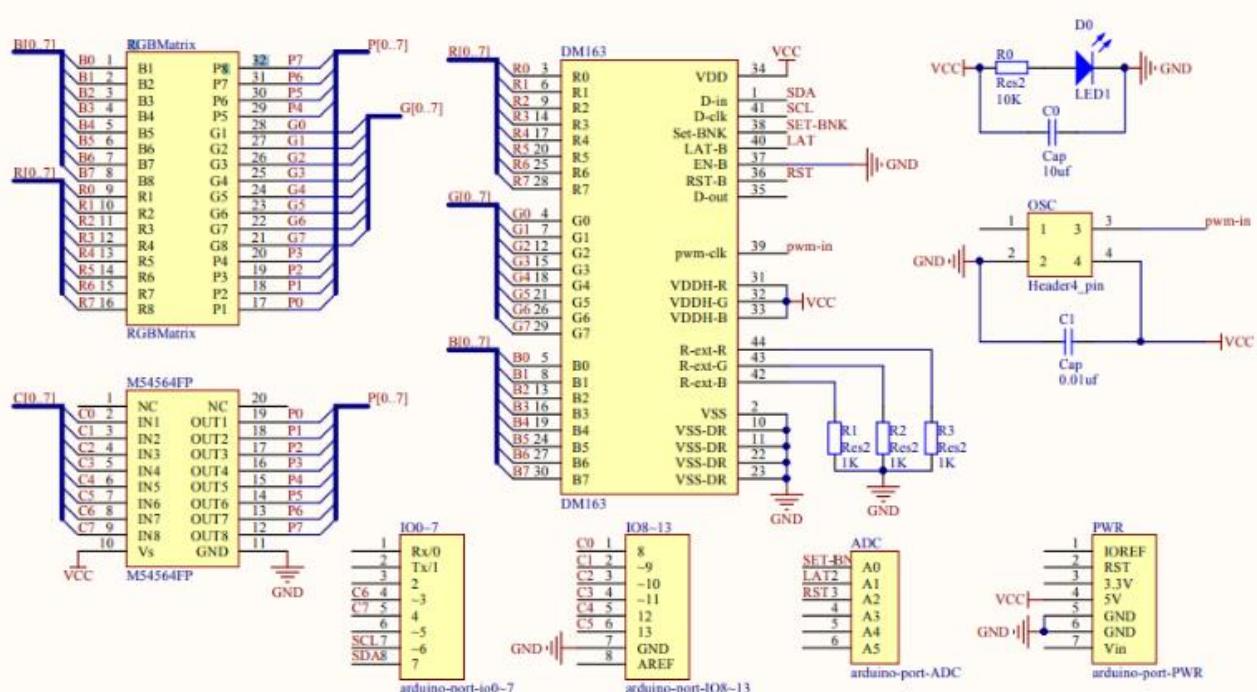


Figura 66 Full Color RGB LED Matrix Driver Shield + RGB Matrix Screen – SUNFOUNDER. Esquema eléctrico de la placa de desarrollo.

Capítulo 2.5. ALTERNATIVAS PARA EL CONTROLADOR.

Para cumplir con las especificaciones del Capítulo 2.3 se debe buscar una forma de controlar el panel y que permita la conexión de forma inalámbrica y mediante

periféricos. Estos controladores deberán tener suficientes entradas y salidas como para controlar el panel y algunos periféricos que se deseen introducir al panel de emergencia en carretera.

Las características eléctricas de los controladores deberán ser compatibles tanto con el panel luminoso como con los diferentes periféricos que se puedan añadir al prototipo evitando así introducir elementos intermedios que los hagan compatibles.

Algunos de los modelos de microcontroladores más populares actualmente se describirán a continuación, analizando las distintas particularidades de cada uno.

Capítulo 2.5.1. Placas de desarrollo basado en Atmel/AVR.

En este apartado se analizarán las diferentes placas de desarrollo que se basan en distintos microcontroladores Atmel de la compañía Microchip.

Arduino Nano Every – ATMega4809. Precio de 12.5€.

Se trata de una placa de desarrollo basada en el microcontrolador ATMega4809 que emplea un procesador ATSAMD11D14A para actuar de puente entre la conexión USB y el microcontrolador. El ATSAMD11D14A está cargado con un firmware que implementa una conversión de USB a comunicación serie que permite programar el ATMega4809 a través de la interfaz UPDI. El firmware también permite ser reprogramado para expandir las posibilidades de la placa (Figura 67 y Figura 68). [26]

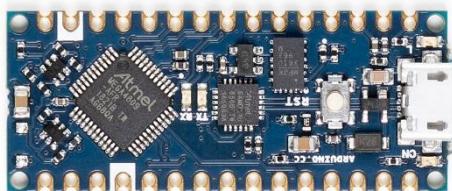


Figura 67 Arduino Nano Every – ATMega4809.
Vista frontal.

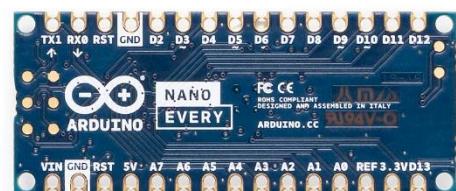


Figura 68 Arduino Nano Every – ATMega4809.
Vista trasera.

En cuanto a la alimentación de la placa, en el datasheet se encuentra el diagrama de árbol sobre potencia de la Figura 69.

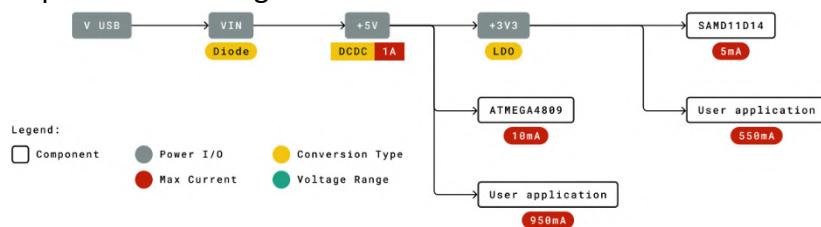


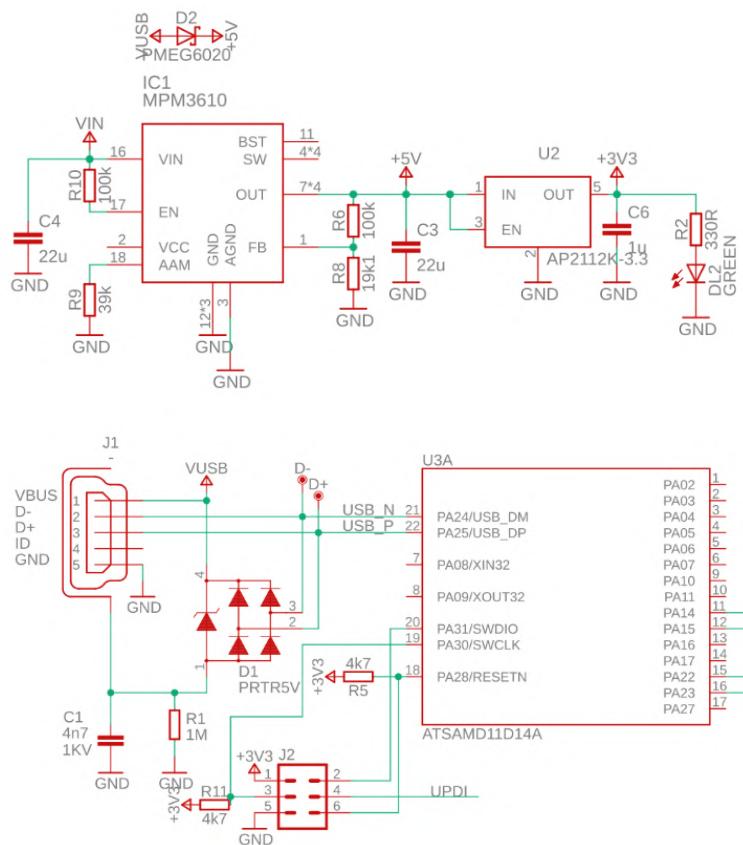
Figura 69 Arduino Nano Every – ATMega4809. Diagrama de árbol de potencia.

Se puede alimentar la placa empleando 5V a partir del puerto USB o bien se puede emplear una alimentación externa de entre 7 y 21V. El puerto USB y la tensión de 5V se separan en la placa empleando un diodo Schottky PMEG6020 que actuará como barrera para proteger el dispositivo USB al que se conecte el Arduino.

La alimentación externa se convierte la tensión de 5V empleando un MPM3610, un regulador reductor asíncrono que para esta aplicación permite obtener máximo 1A de corriente con una eficiencia mínima del 65% alimentado a 21V.

Con la tensión de 5V se permite alimentar el microcontrolador con una corriente de 10mA. Posteriormente se obtiene una tensión de 3,3V a partir del regulador de baja caída de tensión (LDO) AP2112K-3,3 que sirve para alimentar el procesador SAMD11D14 y permite alimentar otros dispositivos con hasta 550mA.

El esquema de la alimentación del circuito se puede observar en la Figura 70.



El cerebro principal de la placa es el microcontrolador ATmega4809 de la serie de microcontroladores megaAVR que puede funcionar con una frecuencia de hasta 20MHz con una memoria Flash de hasta 48KB, hasta 6KB de memoria SRAM y 256 bytes de memoria EEPROM en un empaquetado TQFP48. Los ciclos de escritura y borrado de estas memorias son de 10.000 ciclos para la Flash y 100.000 para la EEPROM, con una retención de datos estimada de 40 años.

Este microcontrolador incluye una arquitectura de bajo consumo empleando un sistema de eventos y SleepWalking. Permite tres modos de bajo consumo o “Sleep”:

- Inactivo con todos los periféricos en funcionamiento para una reactivación inmediata.



- En espera:
 - Operación configurable con los periféricos seleccionados.
 - Periféricos en modo SleepWalking, que permite que algunos periféricos puedan monitorizar el entorno y despertar al microcontrolador.
- Apagado con funcionalidad de reactivación limitada.

En cuanto a los puertos de entradas y salidas, el microcontrolador permite 41 puertos I/O (Figura 71), que la placa de desarrollo limita a un total de 22, donde 14 son pines digitales y 8 son pines analógicos cuya máxima corriente de salida es de 20mA. Una descripción más detallada se encuentra en la Figura 72, donde los puertos GPIO son los indicados como D0 a D13 dónde D3, D5, D6, D9 y D10 permiten obtener una salida PWM con 256 niveles distintos (8 bits de resolución). Además, están las entradas analógicas que convierten a una señal digital empleando un ADC con una resolución de 10 bits. Estos pines son los indicados como A0 a A7, que también pueden ser usados también como puertos GPIO. Esta placa permite comunicaciones empleando protocolos UART (pines RX y TX), SPI (pines D11, D12, D13) e I2C (pines D18 y D19). Se tiene acceso directo también a la tensión de 5V con una salida máxima de 950mA y a 3,3V con una salida máxima de 550mA, aunque es recomendable no exceder los 50mA.

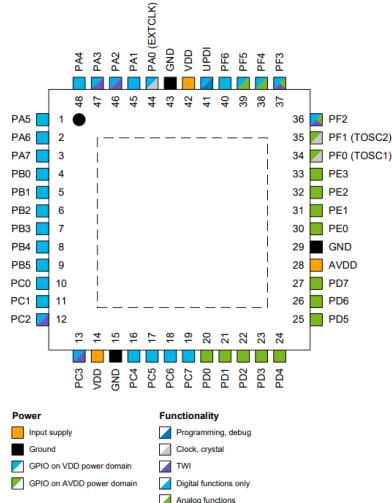


Figura 71 Arduino Nano Every – ATmega4809.
Pines del microcontrolador.

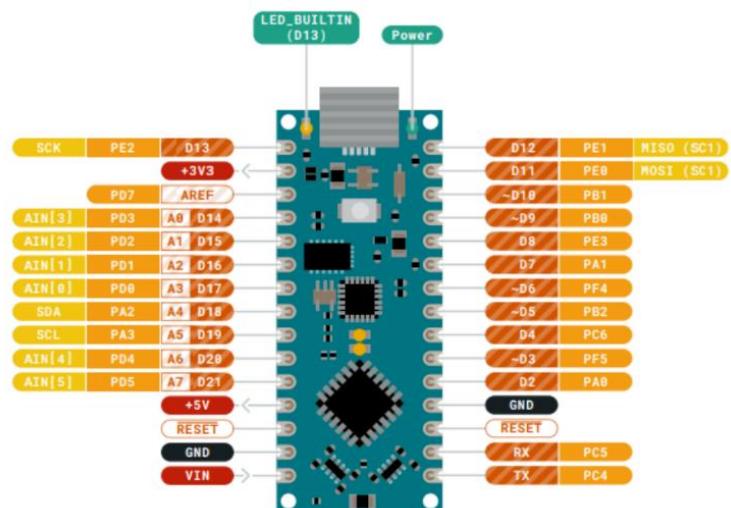


Figura 72 Arduino Nano Every – ATmega4809. Pinout de la placa.

Otros aspectos que señalar sobre esta placa de desarrollo se encuentran en el esquema de la Figura 73 y Figura 74, donde aparecen tres transistores BSS18PS (Q1-1, Q1-2 y Q2-1) que se emplean como Level Shifters para permitir la comunicación entre el procesador SAMD11D14 que trabaja a 3,3V y el microcontrolador ATmega4809 que trabaja a 5V.

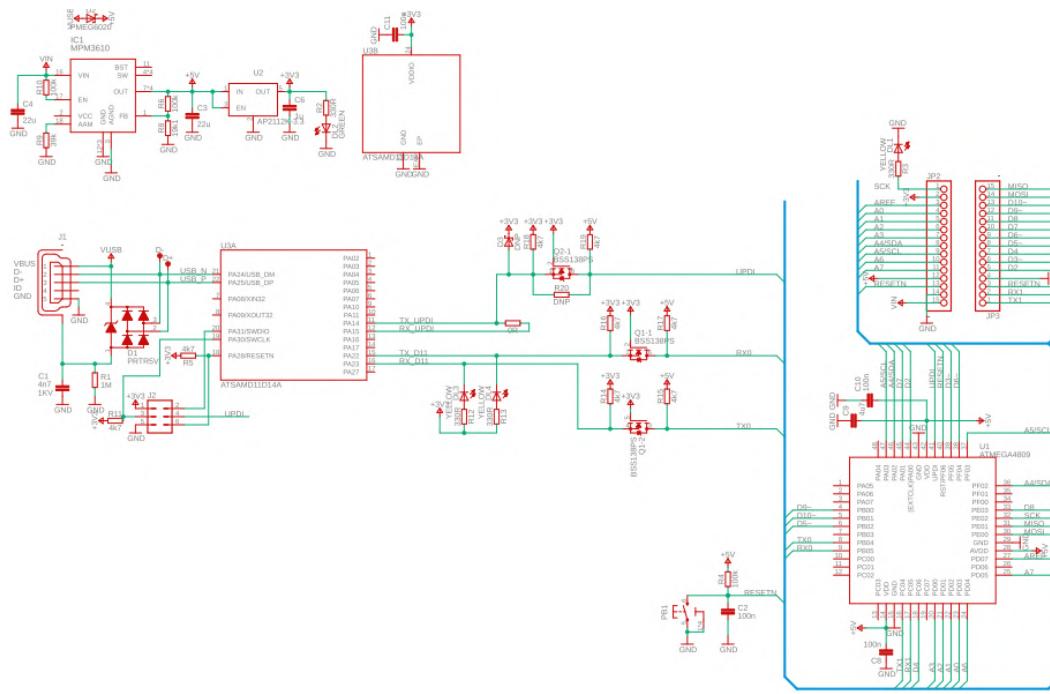


Figura 73 Arduino Nano Every – ATmega4809. Esquema genérico de la placa.

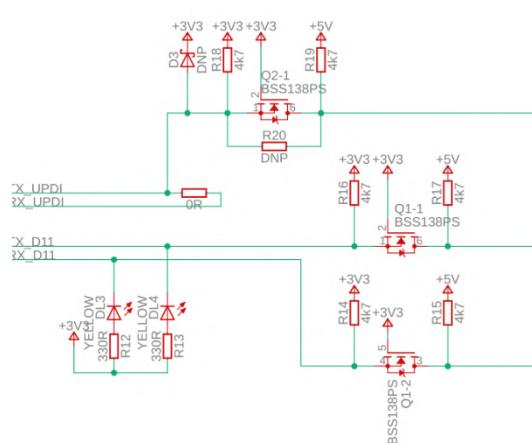


Figura 74 Arduino Nano Every – ATmega4809. Detalle de los Level Shifters.

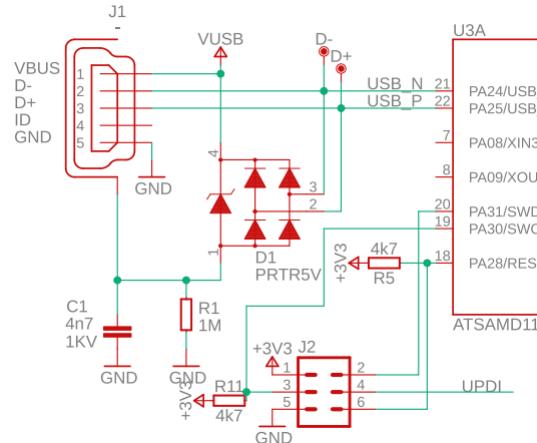


Figura 75 Arduino Nano Every – ATmega4809. Detalle protección línea USB.

Se observa también en la Figura 74 la existencia de dos LEDs que indicarán cuando se está produciendo una comunicación empleando los puertos UART.

Otro de los circuitos integrados del esquema es el PRTR5V0U2AX (Figura 75), un circuito protector de baja capacitancia y salida rail-to-rail contra descargas electrostáticas, diseñado para proteger dos líneas de alta velocidad de transmisión de datos.

Adicionalmente en la Figura 74 se observan dos conectores que son los pines de salida de la placa de desarrollo, un LED conectado al pin D13, un LED que indica que llega tensión al módulo y un botón que se emplea para reiniciar el microcontrolador.

Arduino Leonardo – ATMega32u4. Precio de 19.2€.

En este caso la placa de desarrollo se basa en el microcontrolador ATMega32u4 que no necesita de un CI adicional para la comunicación por USB, pues lo lleva incorporado en su estructura. [27] [28]

Figura 76 Arduino Leonardo – ATMega32u4. Parte frontal.

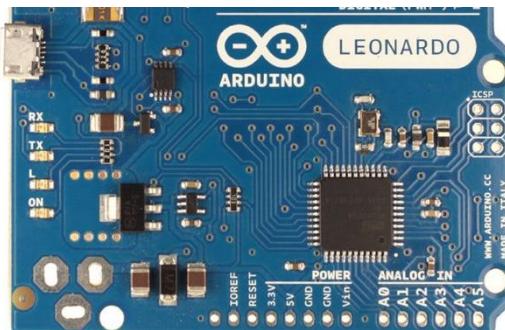


Figura 77 Arduino Leonardo – ATMega32u4. Parte trasera.



En cuanto a la alimentación del dispositivo se emplea un NCP117ST50T3G (Figura 78), un LDO que alimenta al resto de circuitos con una tensión fija de 5V con una corriente de hasta 800mA.

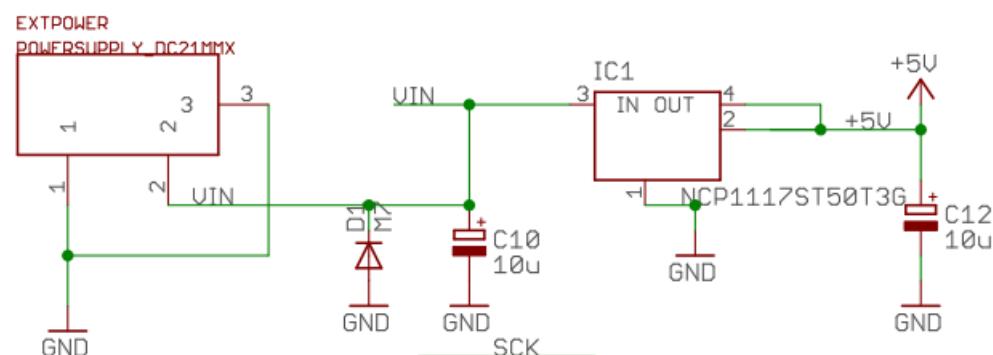


Figura 78 Arduino Leonardo – ATMega32u4. Esquema 5V.

La placa tiene un auto selector de alimentación para seleccionar que tensión se aplica al microcontrolador, si la tensión obtenida por el regulador de la Figura 78 o la del conector micro USB. Este circuito está formado por un FDN3409, un MOSFET de canal P que corta la alimentación del USB cuando VIN es superior a 6.6V, para ello se emplea uno de los dos amplificadores operacionales que tiene el chip LMV358IDGKR, el que se compara mediante un divisor de tensión el voltaje VIN con una referencia de 3,3V obtenida a partir de los 5V empleando un LDO LP2985-33DBVR. La máxima corriente extraíble por el pin de 3,3V es de 50mA. El esquema de este selector se puede observar en la Figura 79.

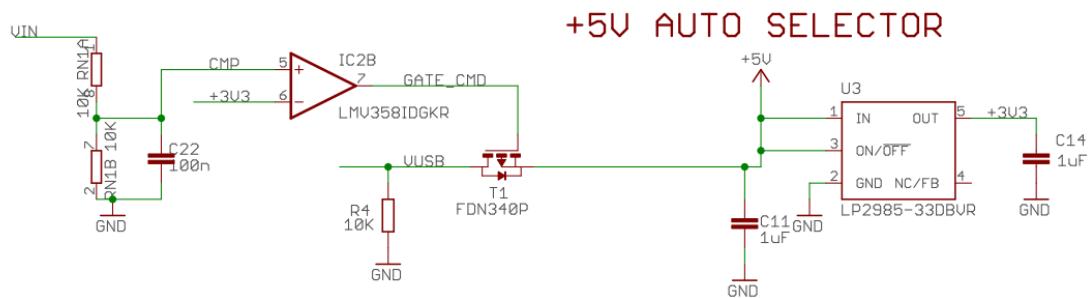


Figura 79 Arduino Leonardo – ATMega32u4. Auto selector de 5V.

Algunos elementos de protección que presenta esta placa es un diodo en la Figura 80 que previene contra la inversión de polaridad y un fusible de 500mA a la entrada de la alimentación por USB.

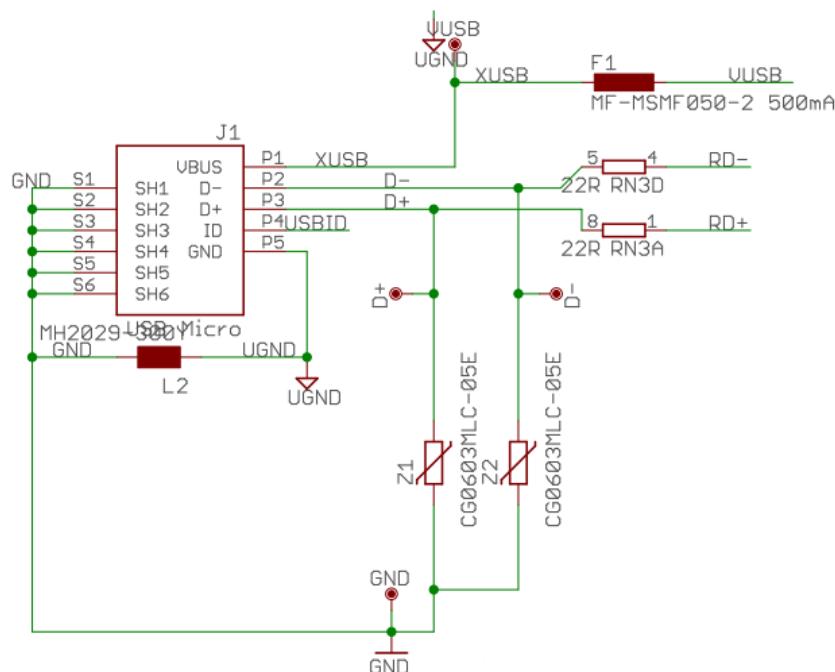


Figura 80 Arduino Leonardo – ATMega32u4. Esquema alimentación por micro USB.

Volviendo al microcontrolador, se basa en un ATMega32u4 que funciona con una frecuencia de 16MHz, 32KB de memoria Flash (4KB están reservados para el bootloader), 2.5KB de memoria SRAM y 1KB de memoria EEPROM. Como se ha mencionado anteriormente, este CI tiene un sistema de comunicación USB incorporado, por lo que no es necesario un segundo procesador para programarlo.

El microcontrolador cuenta con 6 modos de bajo consumo: inactivo, reducción de ruido en el ADC, ahorro de energía, apagado, en espera y en espera extendida.

Para los puertos de entrada y salida, el microcontrolador tiene 26 programables (Figura 81), de los cuales se utilizan 20 como puertos GPIO en la placa de desarrollo. La corriente máxima de salida para estos puertos es de 40mA. De los puertos GPIO anteriores, 7 se pueden emplear como salidas PWM con diferentes características: D3 tiene una resolución de hasta 8 bits, D5 y D5 son PWM de alta velocidad, D9, D10 y D11 tienen

una resolución de 16 hasta bits. También, de los 20 puertos, 6 de ellos pueden ser programados como entradas analógicas mediante un ADC con una resolución de 10 bits. Los pines corresponden a los mostrados en la Figura 82.

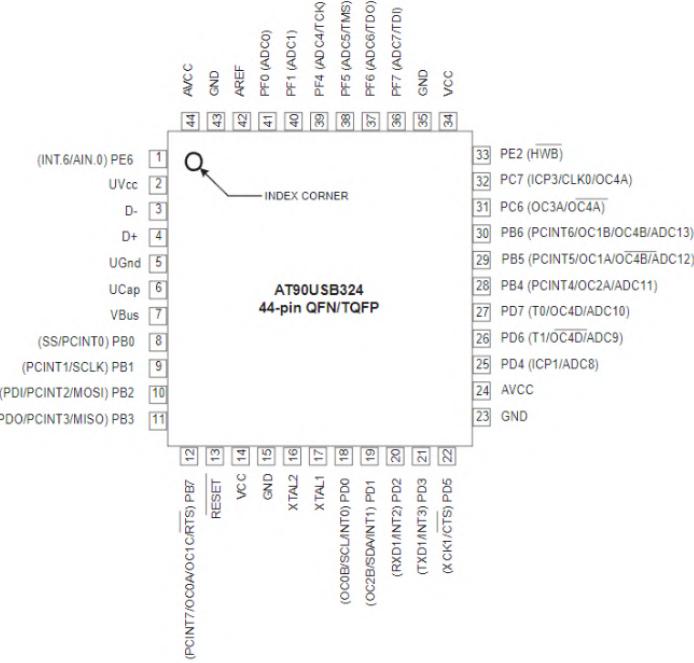


Figura 81 Arduino Leonardo – ATMega32u4. Pines del microcontrolador.

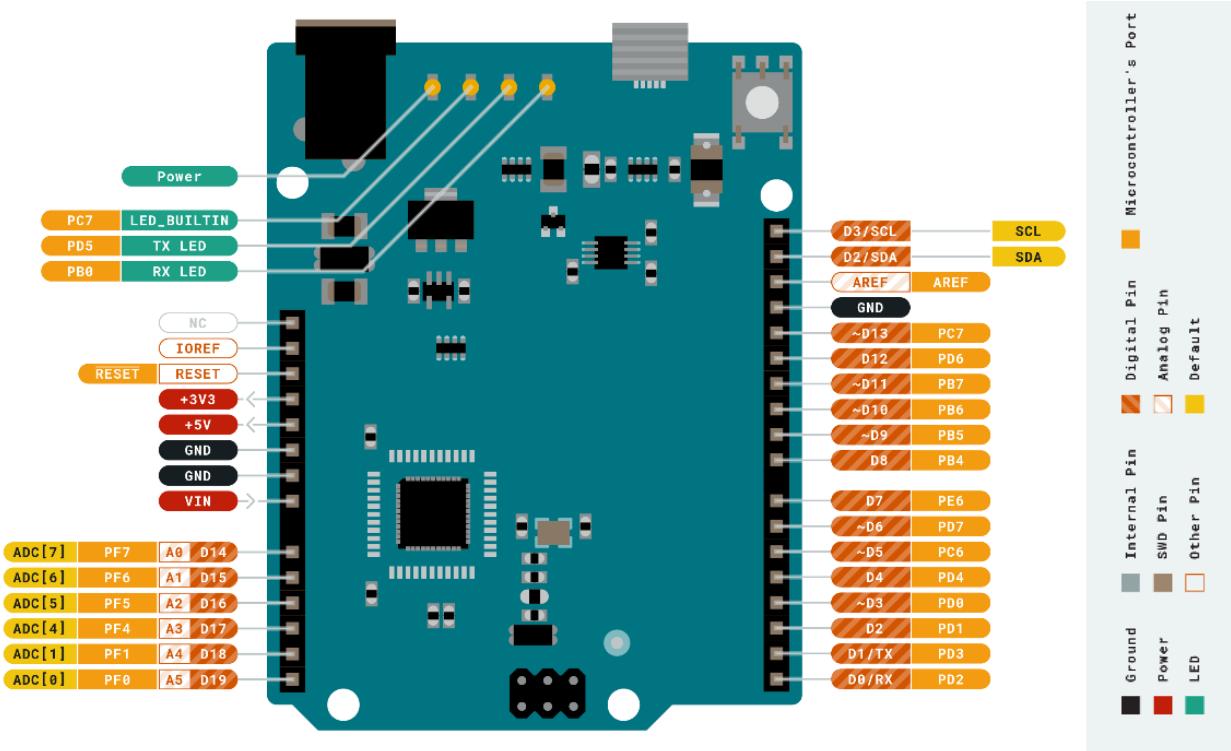


Figura 82 Arduino Leonardo – ATMega32u4. Pines de la placa de desarrollo.

Además de la funcionalidad como puertos GPIO, se pueden emplear los pines D0 y D1 como puerto UART, los pines D2 y D3 como puerto I2C y emplear tres de los seis pines del conector ISCP de la parte inferior como puertos ISP, como se observa en la Figura 83 y la Figura 84.

Por último, hay que indicar que existen varios LEDs para indicar visualmente la comunicación por UART, que la placa se encuentra conectada y un led conectado al pin D13 de libre configuración, como se muestra en la Figura 85.

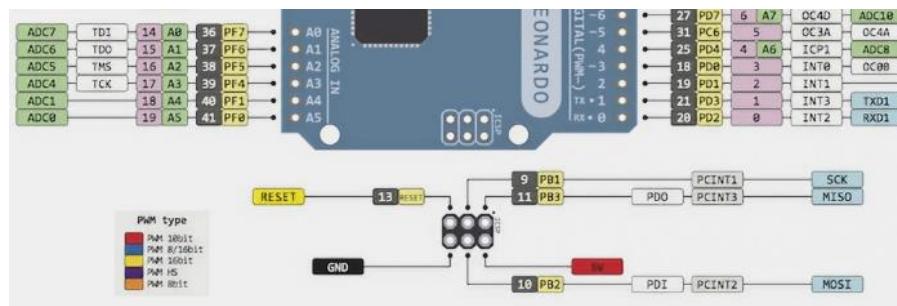


Figura 83 Arduino Leonardo – ATMega32u4. Pines ISCP de la placa de desarrollo.

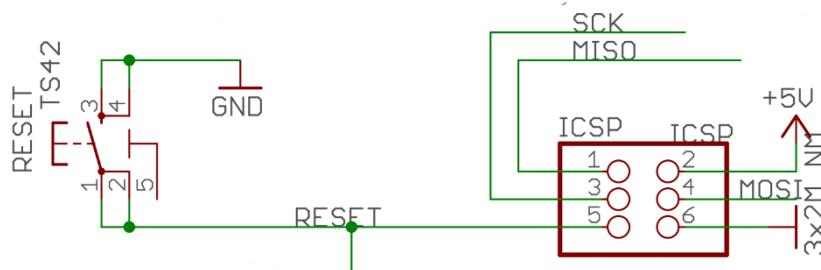


Figura 84 Arduino Leonardo – ATMega32u4. Pines ISCP de la placa de desarrollo en el esquema.

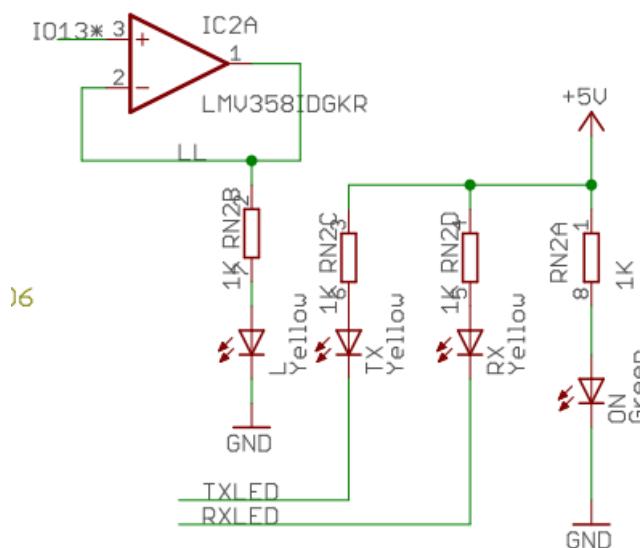


Figura 85 Arduino Leonardo – ATMega32u4. LEDs de la placa de desarrollo.

Arduino Micro – ATMega32u4. Precio de 19.2€.

Se trata de un modelo muy similar al Arduino Leonardo pues emplea el mismo microcontrolador. La principal diferencia es el tamaño (Figura 86), siendo este mucho más pequeño. Otra de las diferencias es la corriente máxima de salida por cada GPIO de 20mA, siendo inferior a las prestaciones del modelo ya mencionado. En cuanto al seleccionador de tensión de 5V, el circuito es algo más simple que el anterior, como se observa en el esquema de la Figura 87. [29]

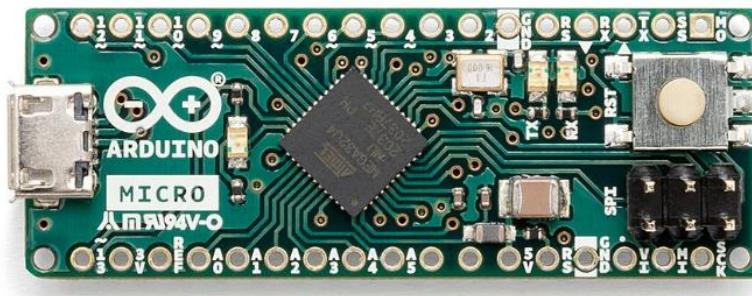


Figura 86 Arduino Micro – ATMega32U4. Vista Frontal.

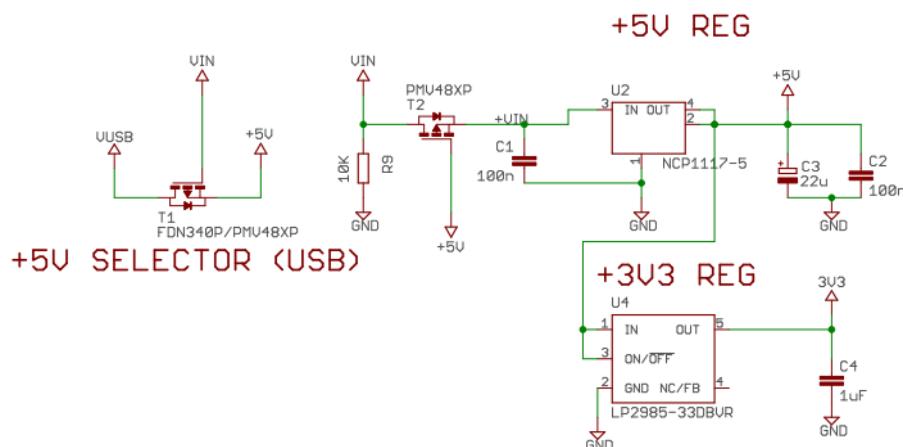


Figura 87 Arduino Micro – ATMega32U4. Seleccionador de 5V.

Arduino Nano – ATMega328. Precio de 21.6 €.

Se trata de un modelo parecido al Arduino Nano Every pero cambiando el microcontrolador por un ATMega328 (Figura 88 y Figura 89). Este modelo necesita de un CI que hace de interfaz entre el protocolo de comunicación USB y el UART, en concreto se emplea el FT23RL (Figura 90).

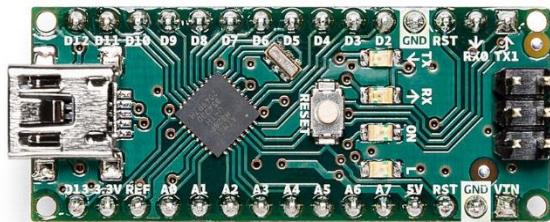


Figura 88 Arduino Nano – ATMega328. Vista Frontal.



Figura 89 Arduino Nano – ATMega328. Vista Trasera.

El microcontrolador trabaja con una frecuencia de hasta 20MHz y posee una memoria Flash de 32KB (2KB usados para el bootloader), una memoria SRAM de 2KB y una EEPROM de 1KB. Tiene un total de 22 puertos hágiles en la placa de desarrollo de los cuales 6 son salidas PWM y 8 pueden ser configuradas como GPIO o entradas analógicas (Figura 91). La máxima corriente de salida de cada puerto GPIO es de 40mA [30].

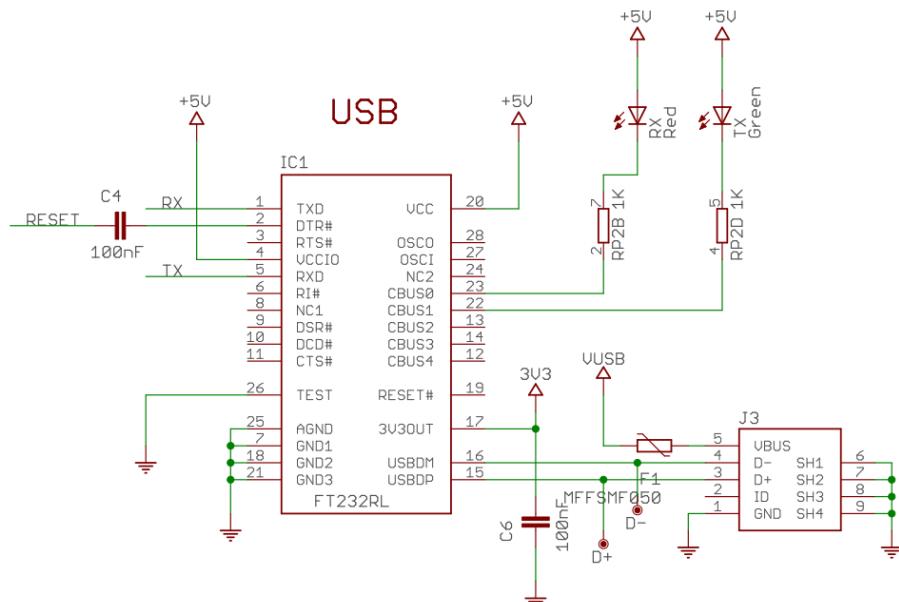


Figura 90 Arduino Nano – ATMega328. Esquema interfaz USB a UART.

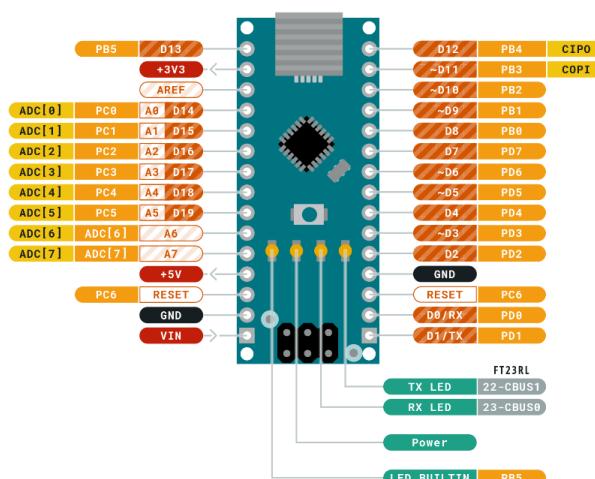


Figura 91 Arduino Nano – ATMega328. Pinout de la placa de desarrollo.

Capítulo 2.5.2. Placas de desarrollo basados en ESP32.

La serie de microprocesadores ESP32 se trata de un circuito integrado que contiene tecnología Wi-Fi de 2.4GHz y Bluetooth que ha sido diseñado específicamente para aplicaciones del internet de las cosas (IoT) en un sistema de bajo consumo. El subsistema Wi-Fi que integra es compatible con el protocolo IEEE 802.11b/g/n trabajando con velocidades de hasta 150Mbps. En cuanto al bluetooth, es compatible con la versión 4.2 y permite aplicar protocolos de Bluetooth Low Energy (BLE) con potencias de hasta 12dBm.

Sobre la CPU, se trata de un microprocesador de un solo núcleo o doble de 32 bits con memoria ROM y memoria SRAM. Soporta además la implementación de varias memorias flash empleando protocolos de QSPI. La frecuencia interna de funcionamiento es de hasta 240MHz, aunque puede variar según el modelo.

Las interfaces para la incorporación de periféricos consisten en 34 puertos GPIO programables, 18 convertidores analógico-digitales (ADC) con una resolución de 12 bits, 2 convertidores digital analógicos (DAC) de 8 bits de resolución, 10 sensores táctiles, 4 puertos SPI, 2 puertos I2C, 2 puertos I2S, 3 puertos UART y la posibilidad de implementar salidas PWM entre otras funcionalidades. Un resumen de todas las características se muestra en el diagrama de bloques de la Figura 92.

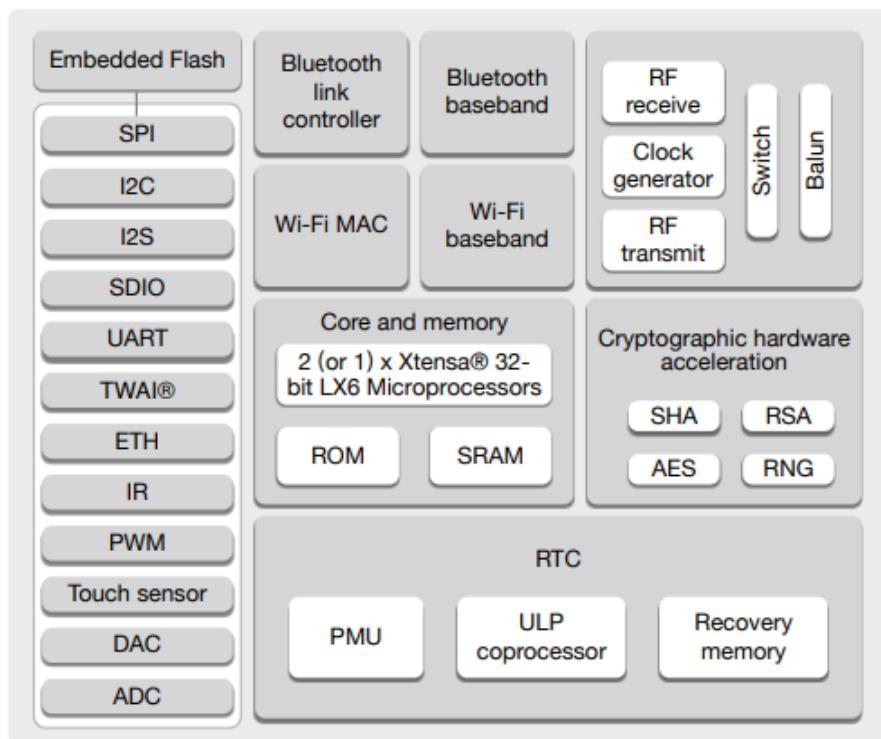


Figura 92 ESP32 Series. Diagrama de bloques.

Este microcontrolador posee 6 modos de funcionamiento:

- Modo activo: El chip de radio está encendido y puede recibir, transmitir o escuchar.

- Modo de suspensión del módem: La CPU está operativa y el Wi-Fi y el Bluetooth están desactivados.
- Modo de suspensión ligera: La CPU está pausada. La memoria y los periféricos RTC así como el procesador de ultra baja energía (ULP) están activos. Cualquier evento puede reactivar el sistema.
- Modo de suspensión profunda: Únicamente la memoria y los periféricos RTC están activos. Los datos obtenidos por la conexión Wi-Fi y Bluetooth son almacenados en dicha memoria. El procesador ULP está operativo.
- Modo de hibernación: El oscilador de 8MHz y el procesador ULP están deshabilitado. La memoria de recuperación del RTC está desactivada. Únicamente el temporizador lento y ciertos GPIOs están activos y pueden sacar al chip de este estado.

Los consumos de estos modos se observan en la Figura 93.

Power mode	Description			Power consumption	
Active (RF working)	Wi-Fi Tx packet			Please refer to Table 15 for details.	
	Wi-Fi/BT Tx packet				
	Wi-Fi/BT Rx and listening				
Modem-sleep	The CPU is powered on.	240 MHz [*]	Dual-core chip(s)	30 mA ~ 68 mA	
		Single-core chip(s)	N/A		
	160 MHz [*]	Dual-core chip(s)	27 mA ~ 44 mA		
		Single-core chip(s)	27 mA ~ 34 mA		
Power mode	Description			Power consumption	
		Normal speed: 80 MHz	Dual-core chip(s)	20 mA ~ 31 mA	
			Single-core chip(s)	20 mA ~ 25 mA	
Light-sleep	-			0.8 mA	
Deep-sleep	The ULP co-processor is powered on.			150 µA	
	ULP sensor-monitored pattern			100 µA @1% duty	
	RTC timer + RTC memory			10 µA	
Hibernation	RTC timer only			5 µA	
Power off	CHIP_PU is set to low level, the chip is powered off.			1 µA	

Figura 93 ESP32 Series. Tablas de resumen de consumo.

Según la página oficial del fabricante, existen varios tipos principales de placas de desarrollo basadas en este chip, cambiando algunas peculiaridades como el tamaño de la memoria flash.

ESP32-WROOM-32E o ESP32-WROOM-32UE [31]_[32]: Se trata de una placa de desarrollo basada en un ESP32 como se muestra en la Figura 94 donde la mayoría de los pines de entrada y salida del microcontrolador están conectados con el exterior. En la placa de desarrollo se observan varios elementos que permiten programar e interactuar con el microprocesador que se encuentra protegido en el encapsulado ESP32-WROVER, que contiene el chip del microprocesador, la memoria externa y varias resistencias, condensadores, bobinas y un reloj oscilador como se puede ver en el esquema de la Figura 95.

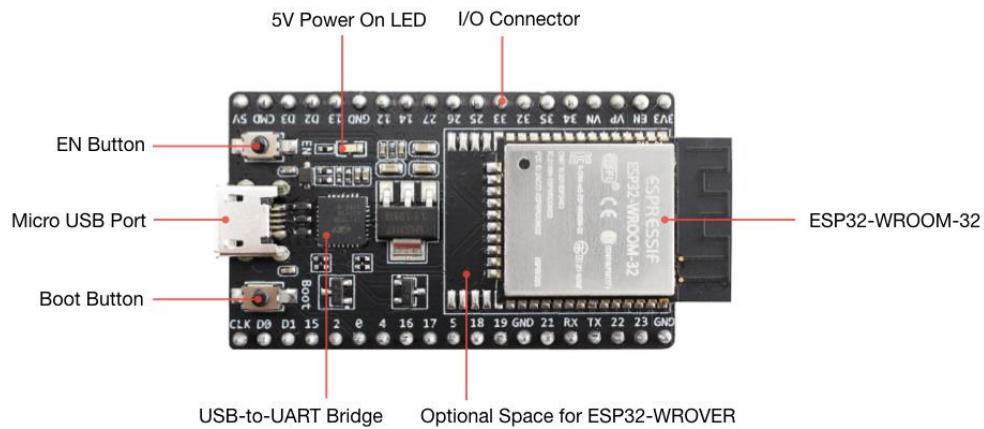


Figura 94 ESP32 Series. ESP32-WROOM-32.

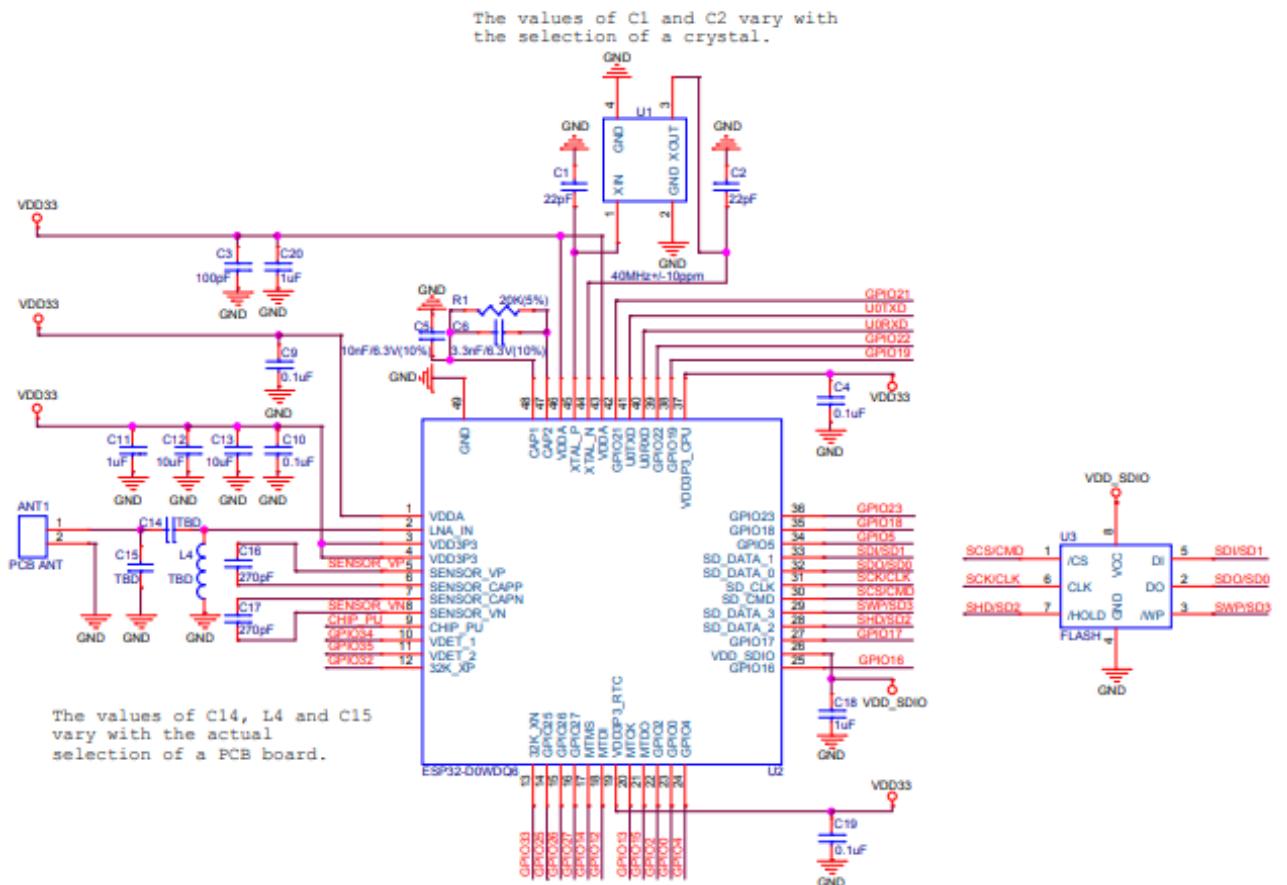


Figura 95 ESP32 Series. ESP32-WROOM-32. Esquema interno del chip.

El esquema de la placa de desarrollo se puede ver en la Figura 96, donde se puede ver que el microprocesador se alimenta con 3V a partir de una alimentación de 5V empleando para ello un regulador AMS1117-3.3. Existe también una interfaz entre el puerto USB y los puertos UART del microcontrolador formado por el CI CP2102N-A01-GQFN28 que permite su programación a partir de un conector micro USB.

Los transistores de la Figura 96 que se conectan a los pines EN y IO0 sirven para activar la programación automática siguiendo la tabla de verdad síncrona indicada.

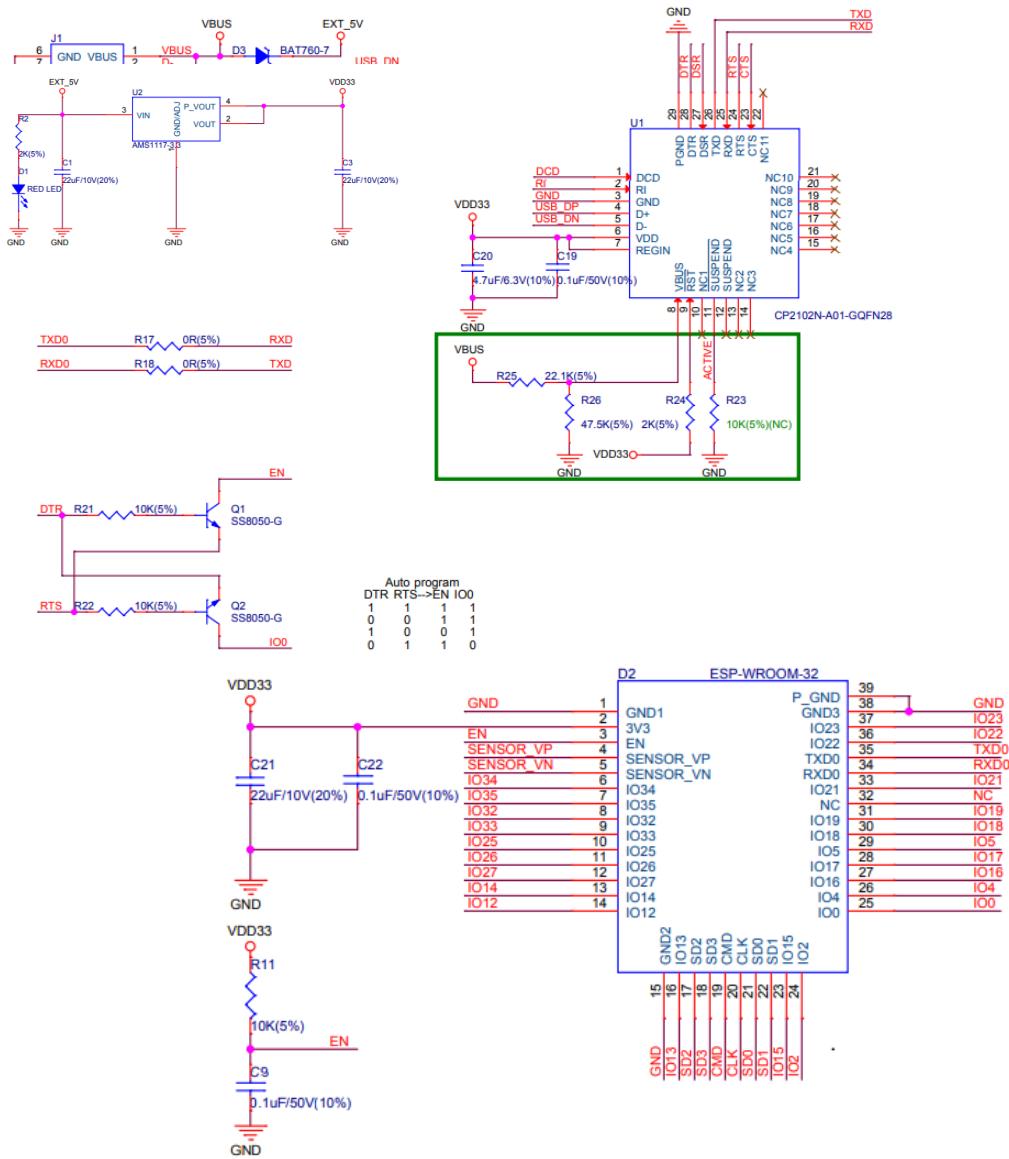
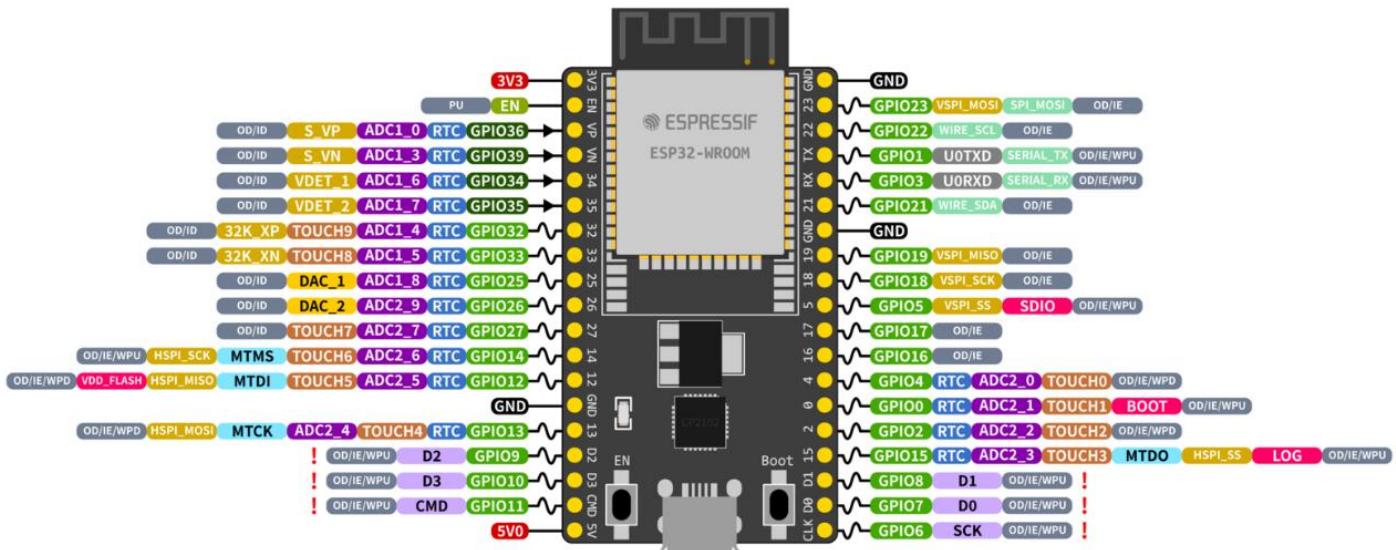


Figura 96 ESP32 Series. ESP32-WROOM-32. Esquemas de la placa de desarrollo.

Los pines de salida de esta placa son los descritos en la Figura 97, donde se encuentran disponibles 26 puertos GPIO, 3 puertos SPI, 3 puertos UART, 2 puertos I2C, 16 entradas analógicas y 10 sensores táctiles entre otras funcionalidades.

ESP32-DevKitC



ESP32 Specs

32-bit Xtensa® dual-core @240MHz
Wi-Fi IEEE 802.11 b/g/n 2.4GHz
Bluetooth 4.2 BR/EDR and BLE
520 KB SRAM (16 KB for cache)
448 KB ROM
34 GPIOs, 4x SPI, 3x UART, 2x I2C,
2x I2S, RMT, LED PWM, 1 host SD/eMMC/SDIO,
1 slave SDIO/SPI, TWAI®, 12-bit ADC, Ethernet

	PWM Capable Pin
	GPIO Input Only
	GPIO Input and Output
	Digital-to-Analog Converter
	JTAG for Debugging
	External Flash Memory (SPI)
	Analog-to-Digital Converter
	Touch Sensor Input Channel
	Other Related Functions
	Serial for Debug/Programming
	SERIAL
	ARDUINO
	Arduino Related Functions
	STRAP
	RTC Power Domain (VDD3P3_RTC)
	Ground
	Power Rails (3V3 and 5V)
	Pin Shared with the Flash Memory
	Can't be used as regular GPIO
	WPU: Weak Pull-up (Internal)
	WPD: Weak Pull-down (Internal)
	PU: Pull-up (External)
	IE: Input Enable (After Reset)
	ID: Input Disabled (After Reset)
	OE: Output Enable (After Reset)
	OD: Output Disabled (After Reset)

Figura 97 ESP32 Series. ESP32-WROOM-32. Pinout.

Capítulo 2.5.3. Placas de desarrollo basados en ESP32-S2.

La serie ESP32-S2 es un sistema que integra en el mismo chip (SoC) comunicación Wi-Fi de 2.4Ghz compatible con el protocolo IEEE 802.11b/g/n y además que y mecanismos de bajo consumo, por lo que junto a su potencia es una opción ideal para aplicaciones relacionadas con el internet de las cosas (IoT) [33].

El chip trabaja a frecuencias de hasta 240MHz e incluye 320KB de memoria SRAM, 128KB de memoria ROM y permite conectar memorias flash y RAM empleando un bus SPI. Las características en cuanto a memoria pueden variar en función del modelo concreto de módulo escogido, como recoge la Figura 98.

Ordering Code	Embedded Flash	Embedded PSRAM	Ambient Temperature (°C)
ESP32-S2	—	—	-40 ~ 105
ESP32-S2FH2	2 MB	—	-40 ~ 105
ESP32-S2FH4	4 MB	—	-40 ~ 105
ESP32-S2FN4R2	4 MB	2 MB	-40 ~ 85
ESP32-S2R2	—	2 MB	-40 ~ 85

Figura 98 ESP32-S2 Series. Comparación entre los distintos modelos.



Este dispositivo, al igual que el modelo básico anterior; incluye un amplio set de interfaces de periféricos en los que se incluyen SPI, I2S, UART, I2C, LED_PWM, interfaz LCD, interfaz para cámara, sensor táctil, sensor de temperatura interno y 43 puertos GPIO. Incluye también una interfaz de comunicación USB de alta velocidad.

Todas las características definidas anteriormente y algunas adicionales quedan resumidas en el diagrama de bloques de la Figura 99.

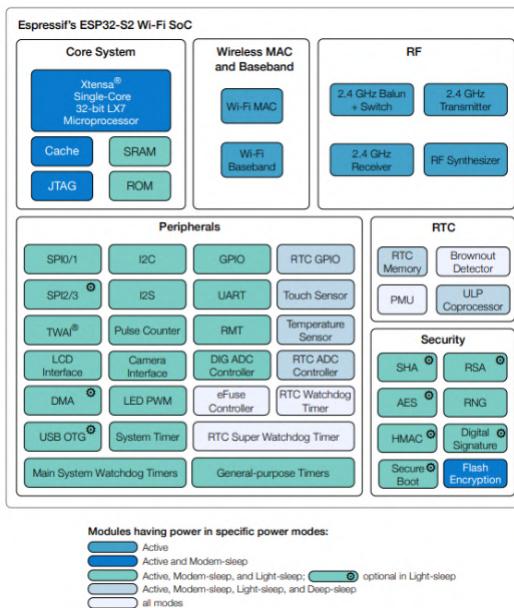


Figura 99 ESP32-S2 Series. Diagrama de bloques.

En cuanto al consumo de la serie ESP32-S2 se debe distinguir entre los distintos modos de funcionamiento que presenta el chip:

- Modo activo: La CPU y el chip de radio están encendidos. El chip puede recibir, transmitir y escuchar. El consumo de las transmisiones de radio frecuencia o RF no superará en este modo los 370mA, aunque dependerá del modo de comunicación que se esté empleando como se define en la Figura 100.

Mode	Description	Peak (mA)
Active (RF working)	802.11b, 20 MHz, 1 Mbps, @19.5 dBm	310
	802.11g, 20 MHz, 54 Mbps, @15 dBm	220
	802.11n, 20 MHz, MCS7, @13 dBm	200
	802.11n, 40 MHz, MCS7, @13 dBm	160
	802.11b/g/n, 20 MHz	63
	802.11n, 40 MHz	68

Figura 100 ESP32-S2 Series. Consumo en modo activo.

- Modo de suspensión del módem (Modem Sleep Mode): La CPU está operativa y la velocidad del reloj puede ser reducida para disminuir el consumo. La banda base Wi-Fi y la radio están deshabilitadas, pero la conexión Wi-Fi puede permanecer activa. El consumo no superará los 32mA, aunque dependerá de la frecuencia de varios factores como se ve reflejado en la Figura 101.



Mode	CPU Frequency (MHz)	Description	Typ	
			All Peripherals Clocks Disabled (mA)	All Peripherals Clocks Enabled (mA) ¹
Modem-sleep ^{2,3}	240	CPU is idle	20.0	28.0
		CPU is running	23.0	32.0
	160	CPU is idle	14.0	21.0
		CPU is running	16.0	24.0
	80	CPU is idle	10.5	18.4
		CPU is running	12.0	20.0

Figura 101 ESP32-S2 Series. Consumo en modo de suspensión del módem.

- Modo de suspensión ligera (Light-sleep mode): La CPU está pausada. Los periféricos y el coprocesador de ultra baja energía (ULP) están activos. Cualquier evento de activación despertará al chip. La conexión Wi-Fi permanecerá activa. El consumo máximo del chip en este modo de operación es de 750µA (Figura 102).
- Modo de suspensión profundo (Deep-sleep mode): Únicamente la memoria y los periféricos RTC están activos. Los datos de la conexión Wi-Fi son almacenados en la memoria RTC. El procesador ULP es funcional. El consumo máximo en este modo de operación es de 190µA (Figura 102).
- Modo de hibernación: El oscilador interno de 8MHz y el coprocesador ULP están desactivados. La memoria RTC de recuperación está apagada. Solamente el temporizador RTC, un reloj de baja velocidad y determinados RTC GPIOs están activos. El temporizador RTC puede activar el chip desde el modo de hibernación. El consumo en este modo es muy limitado, siendo tan solo 1µA (Figura 102).

Mode	Description		Typ (µA)
Light-sleep ¹	VDD_SPI and Wi-Fi are powered down, and all GPIOs are high-impedance		750
Deep-sleep	The ULP co-processor is powered on ²	ULP-FSM	170
		ULP-RISC-V	190
	ULP sensor-monitored pattern ³		22
	RTC timer + RTC memory		25
	RTC timer only		20
Power off	CHIP_PU is set to low level, the chip is powered off		1

Figura 102 ESP32-S2 Series. Consumo en los modos ligero, profundo y apagado.

En base a esta familia de microcontrolador se encuentran tres placas principales con pequeñas variaciones:

ESP32-S2-DevKitM-1/ESP32-S2-DevKitM-1U[34]: Dos placas de desarrollo muy similares que se diferencian por la forma que tiene el encapsulado del microcontrolador, empleándose en el primero el encapsulado ESP32-S2-MINI-1 en que viene la antena incorporada (Figura 103) y en el segundo el encapsulado ESP32-S2-MINI-1U, donde viene un conector para que el usuario conecte una antena externa (Figura 104).

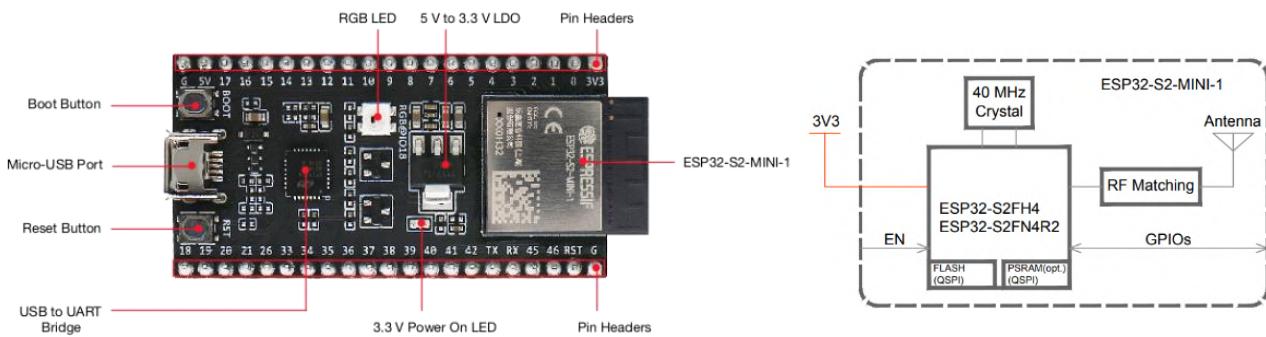


Figura 103 ESP32-S2 Series. ESP32-S2-DevkitM-1. Vista frontal y diagrama de bloques del encapsulado del µC.

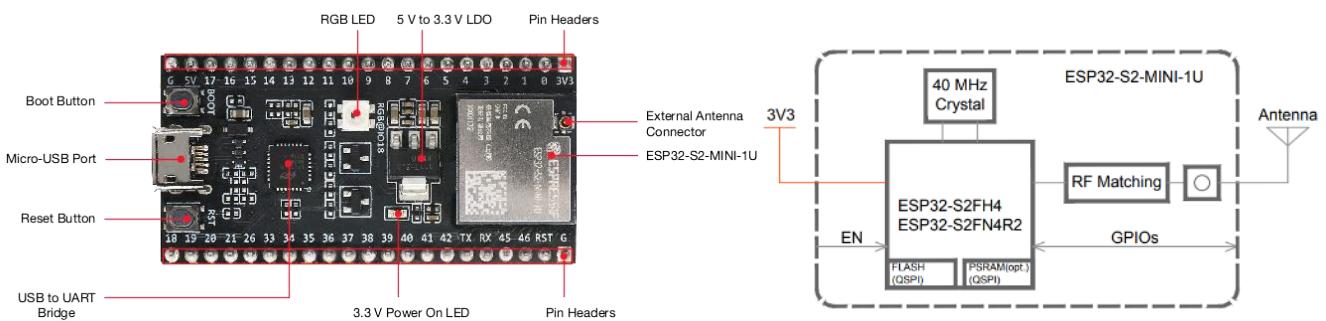


Figura 104 ESP32-S2 Series. ESP32-S2-DevkitM-1U. Vista frontal y diagrama de bloques del encapsulado del µC.

Presenta 43 puertos GPIO, aunque el IO26 estará conectado a la memoria PSRAM en los modelos ESP32-S2-MINI-1-N4R2 y ESP32-S2-MINI-1U-N4R2 y otros se emplearán para comunicarse con ciertas partes del circuito de la placa de desarrollo, por lo que no todos ellos están disponibles. El puerto IO46 está destinado únicamente como entrada. Más información de los pines se ve resumida en la Figura 105.

Analizando el circuito de la placa de desarrollo se observa que se emplea el CI CP2102N-A02-GQFN28 como puente entre el protocolo USB y el UART que permite conectar la placa a un ordenador y programar su microcontrolador empleando el conector micro USB de la placa (Figura 106).

ESP32-S2-DevKitM-1

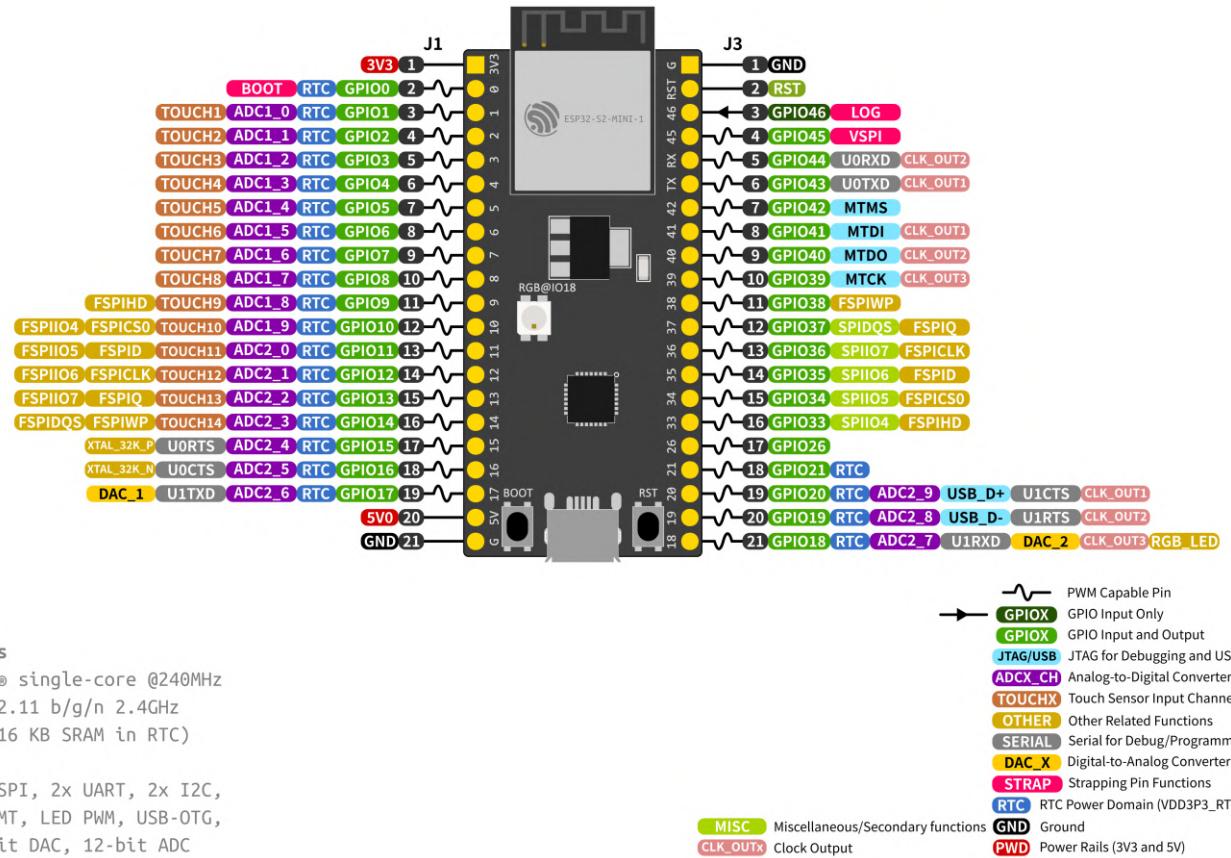


Figura 105 ESP32-S2 Series. ESP32-S2-DevkitM-1. Pinout.

ESP32-S2 Specs

32-bit Xtensa® single-core @240MHz
 Wi-Fi IEEE 802.11 b/g/n 2.4GHz
 320 KB SRAM (16 KB SRAM in RTC)
 128 KB ROM
 43 GPIOs, 4x SPI, 2x UART, 2x I2C,
 Touch, I2S, RMT, LED PWM, USB-OTG,
 TWAI®, 2x 8-bit DAC, 12-bit ADC

Al igual que el modelo anterior, el microprocesador se alimenta con una tensión de 3,3V, por lo que se emplea un reductor que obtiene esta tensión a partir de los 5V del USB. Esto se consigue empleando el LDO SGM2212-3.3XKC3G/TR (Figura 108), que permite reducir tensiones de hasta 26.4V.

El dispositivo presenta otros elementos adicionales como un par de botones que permitirán reiniciar el módulo o descargar el firmware a partir del puerto serie, un LED RGB SK68XXMINI-HS conectado al GPIO18 y un LED que indicará que el módulo está encendido.

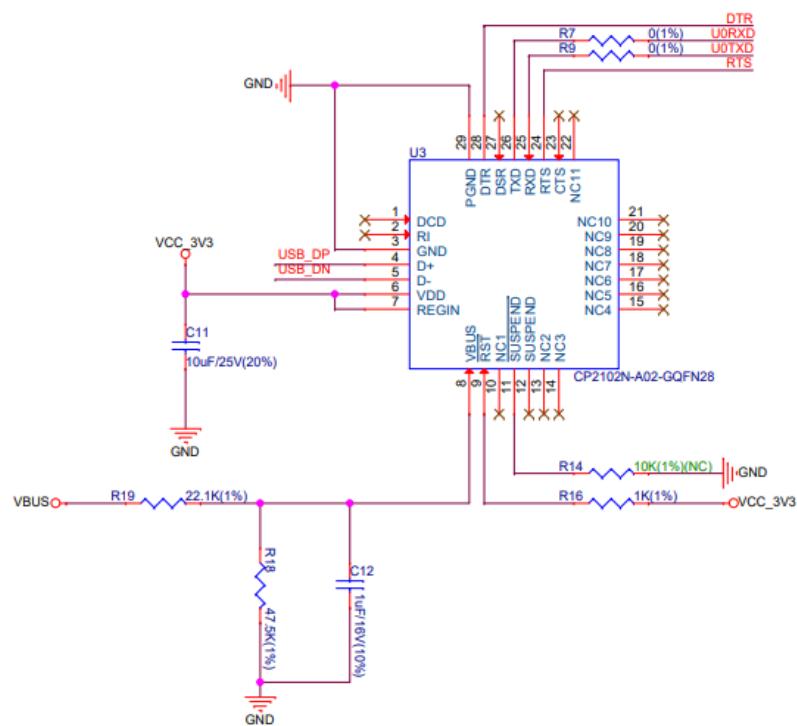


Figura 106 Series. ESP32-S2-DevkitM-1. Esquema del puente USB a UART.

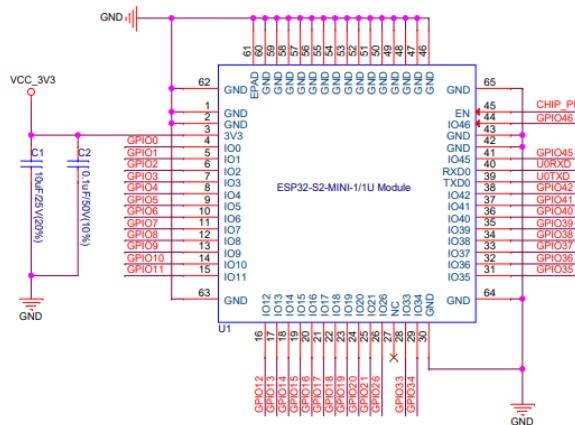


Figura 107 ESP32-S2 Series. ESP32-S2-DevkitM-1. Esquema del microcontrolador.

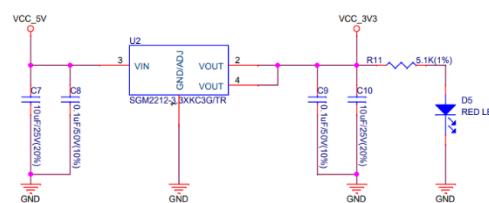


Figura 108 ESP32-S2 Series. ESP32-S2-DevkitM-1. Esquema de la alimentación.

ESP32-S2-DevKitC-1/ESP32-S2-DevKitC-1U [35]: Se trata de dos placas de desarrollo muy similares a las anteriores, diferenciándose en el microcontrolador y en la aparición de un segundo conector micro USB (Figura 109, Figura 110 y Figura 111). El controlador empleado es el ESP32-S2-SOLO o su versión con conector para antena externa ESP32-S2-SOLO-U que como características principales presenta las básicas de la serie ESP21-S2, 128KB de memoria ROM, 320KB de SRAM y 2MB de PSRAM en el modelo ESP32-S2R2, con compatibilidad Wi-Fi en frecuencias de 2,4G y que trabaja hasta frecuencias de 240MHz, trabajando por defecto a 40MHz.

El segundo conector USB se emplea para alimentar la placa, para cargar aplicaciones y comunicarse con el chip empleando los protocolos de USB1.1, por lo que no sería necesario emplear el puente USB a UART que proporciona el CP2102N-A02-GQFN28 como en el caso anterior. En esta placa se presentan las dos opciones.

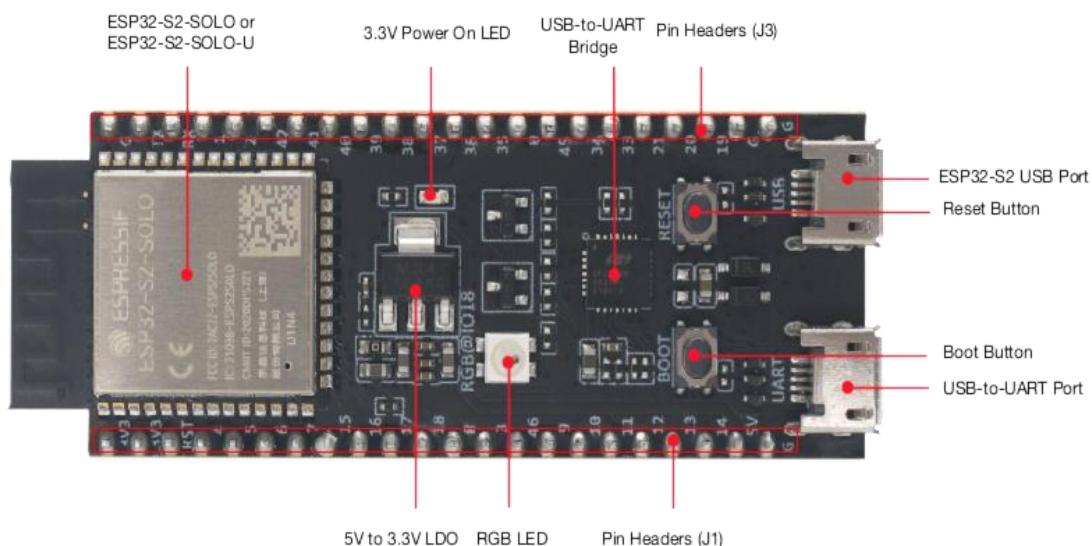


Figura 109 ESP32-S2 Series. ESP32-S2-DevKitC-1. Vista frontal.

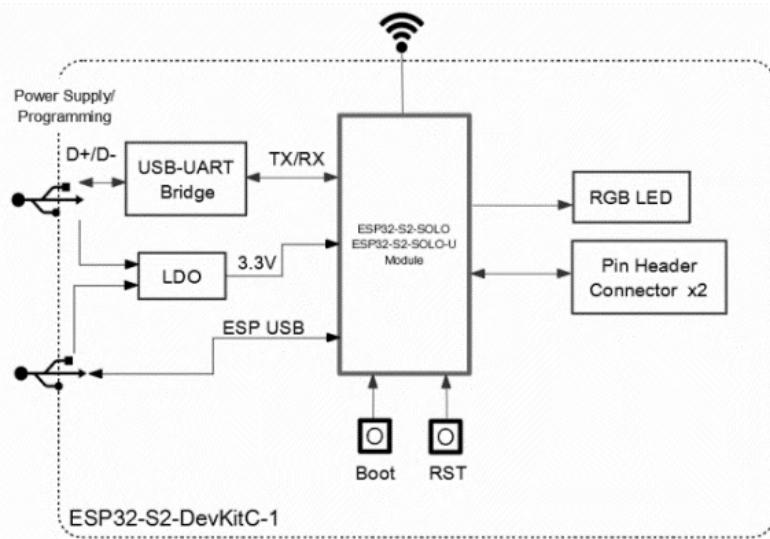
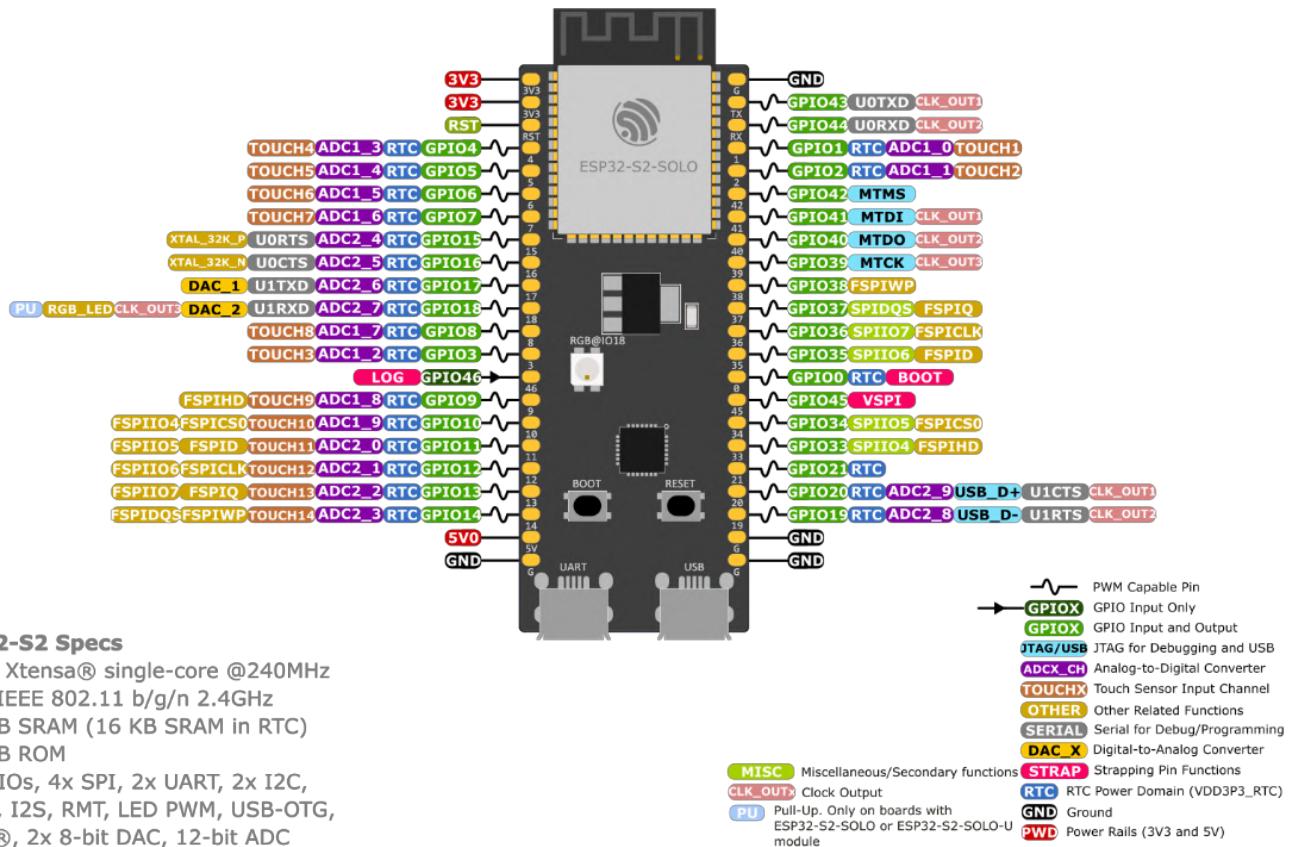


Figura 110 ESP32-S2 Series. ESP32-S2-DevKitC-1. Diagrama de bloques del sistema.

Los puertos de salida de este modelo son similares a los del modelo anterior, quedando reflejados en la Figura 111.

ESP32-S2-DevKitC-1



ESP32-S2 Specs

32-bit Xtensa® single-core @240MHz
 Wi-Fi IEEE 802.11 b/g/n 2.4GHz
 320 KB SRAM (16 KB SRAM in RTC)
 128 KB ROM
 43 GPIOs, 4x SPI, 2x UART, 2x I2C,
 Touch, I2S, RMT, LED PWM, USB-OTG,
 TWAI®, 2x 8-bit DAC, 12-bit ADC

Figura 111 ESP32-S2 Series. ESP32-S2-DevKitC-1. Pinout.

Capítulo 2.5.4. Placas de desarrollo basados en ESP32-S3.

La familia ESP32-S3 es un microcontrolador con tecnología Wi-Fi 2,4 GHz y Bluetooth Low Energy (BLE) integrado. Se trata de un microprocesador de doble núcleo con un módulo RF y numerosos periféricos.

El subsistema Wi-Fi trabaja en anchos de banda de 20 o 40MHz en la banda de 2.4GHz y es compatible con el protocolo IEE 802.11b/g/n.

El sistema BLE tiene un modo de alta potencia a 20dBm con velocidades de 125kbps0 500kbps, 1Mbps y 2Mbps.

En cuanto a la CPU, puede trabajar con frecuencias de hasta 240MHz con un doble núcleo como se ha mencionado anteriormente. Cuenta con una memoria ROM de 384KB, 512KB de memoria SRAM y 16KB de memoria SRAM en RTC. Permite la conexión con una memoria flash externa por QSPI. Las diferencias entre los distintos modelos de la familia S3 se ven reflejadas en la tabla de la Figura 112.

Ordering Code ¹	In-Package Flash	In-Package PSRAM	Ambient Temp. ² (°C)	VDD_SPI Voltage ³
ESP32-S3	—	—	-40 ~ 105	3.3 V/1.8 V
ESP32-S3FN8	8 MB (Quad SPI) ⁴	—	-40 ~ 85	3.3 V
ESP32-S3R2	—	2 MB (Quad SPI)	-40 ~ 85	3.3 V
ESP32-S3R8	—	8 MB (Octal SPI)	-40 ~ 65	3.3 V
ESP32-S3R8V	—	8 MB (Octal SPI)	-40 ~ 65	1.8 V
ESP32-S3FH4R2	4 MB (Quad SPI)	2 MB (Quad SPI)	-40 ~ 85	3.3 V

Figura 112 ESP32-S3 Series. Comparación entre los distintos modelos.

Las interfaces de los periféricos consisten en 45 puertos GPIO programables con interfaces digitales como SPI (x4), interfaz LCD (x1), interfaz DVP de 8 bits (16 bit en interfaz para cámara), UART (x3), I2C (x2), I2S (x2), entradas analógicas (ADC) de 12 bits (x2), sensores táctiles (x14) y puertos USB de alta velocidad entre otros.

Todas las capacidades de este microcontrolador se ven reflejadas en el diagrama de bloques de la Figura 113.

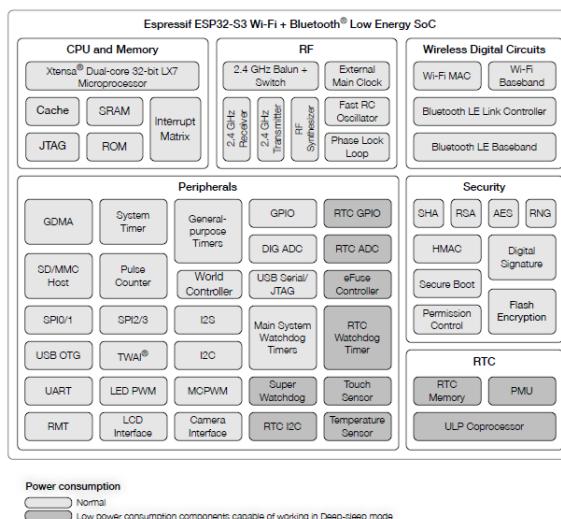


Figura 113 ESP32-S3 Series. Diagrama de bloques.

En cuanto al consumo de la serie ESP32-S3 se debe distinguir entre los distintos modos de funcionamiento que presenta el chip:

- Modo activo: La CPU, los circuitos RF y los periféricos están activos. El chip puede procesar datos, recibir, transmitir y escuchar. El consumo de los circuitos RF no superará en este modo los 340mA, aunque dependerá del modo de comunicación que se esté empleando como se define en la Figura 114.

Work Mode ¹	Description	Peak (mA)
Active (RF working)	802.11b, 1 Mbps, @21 dBm	340
	802.11g, 54 Mbps, @19 dBm	291
	802.11n, HT20, MCS7, @18.5 dBm	283
	802.11n, HT40, MCS7, @18 dBm	286
	802.11b/g/n, HT20	88
	802.11n, HT40	91

Figura 114 ESP32-S3 Series. Consumo en modo activo.

- Modo de suspensión del módem (Modem Sleep Mode): La CPU está operativa y la velocidad del reloj puede ser reducida. Las conexiones inalámbricas pueden ser configuradas para mantener activos los circuitos de RF de forma periódica cuando sean necesarios. El consumo del microcontrolador en este modo dependerá de diversos parámetros como se refleja en la Figura 115, donde el mayor consumo es de 107.9mA.

Work mode	Frequency (MHz)	Description	Typ ¹ (mA)	Typ ² (mA)
Modem-sleep ³	40	WAITI (Dual core in idle state)	13.2	18.8
		Single core running 32-bit data access instructions, the other core in idle state	16.2	21.8
		Dual core running 32-bit data access instructions	18.7	24.4
		Single core running 128-bit data access instructions, the other core in idle state	19.9	25.4
		Dual core running 128-bit data access instructions	23.0	28.8
		WAITI	22.0	36.1
	80	Single core running 32-bit data access instructions, the other core in idle state	28.4	42.6
		Dual core running 32-bit data access instructions	33.1	47.3
		Single core running 128-bit data access instructions, the other core in idle state	35.1	49.6
		Dual core running 128-bit data access instructions	41.8	56.3
		WAITI	27.6	42.3
	160	Single core running 32-bit data access instructions, the other core in idle state	39.9	54.6
		Dual core running 32-bit data access instructions	49.6	64.1
		Single core running 128-bit data access instructions, the other core in idle state	54.4	69.2
		Dual core running 128-bit data access instructions	66.7	81.1
		WAITI	32.9	47.6
	240	Single core running 32-bit data access instructions, the other core in idle state	51.2	65.9
		Dual core running 32-bit data access instructions	66.2	81.3
		Single core running 128-bit data access instructions, the other core in idle state	72.4	87.9
		Dual core running 128-bit data access instructions	91.7	107.9

Figura 115 ESP32-S3 Series. Consumo en modo de suspensión del módem.

- Modo de suspensión ligera (Light-sleep mode): La CPU deja de funcionar y puede ser accionada de forma opcional. Los periféricos y el coprocesador ULP pueden ser reactivados periódicamente por un temporizador. Cualquier evento de activación despertará al chip. La conexión Wi-Fi podrá permanecer activa. El consumo máximo del chip en este modo de operación es de $240\mu A$ (Figura 116).
- Modo de suspensión profundo (Deep-sleep mode): Únicamente el RTC está activo. Los datos de las conexiones inalámbricas se almacenan en la memoria RTC. El consumo en este modo es de $8\mu A$ (Figura 116).

Work mode	Description	Typ (μA)
Light-sleep ¹	VDD_SPI and Wi-Fi are powered down, and all GPIOs are high-impedance.	240
Deep-sleep	RTC memory and RTC peripherals are powered up. RTC memory is powered up. RTC peripherals are powered down.	8 7
Power off	CHIP_PU is set to low level. The chip is shut down.	1

Figura 116 ESP32-S3 Series. Consumo en los modos ligero, profundo y apagado.

Una tabla resumen sobre las funcionalidades activas en los diferentes modos de operación se encuentra en la Figura 117.

Power Domain Power Mode	RTC Optional RTC Periph	Digital			Analog		
		CPU	Optional Digital Periph	Wireless Digital Circuits	RC_ FAST_ CLK	XTAL_ CLK	PLL RF Circuits
Active	ON	ON	ON	ON	ON	ON	ON
Modem-sleep	ON	ON	ON	ON	ON ¹	ON	ON
Light-sleep	ON	ON	ON	OFF ¹	ON ¹	OFF ¹	OFF ¹
Deep-sleep	ON	ON	OFF	OFF	OFF	OFF	OFF

Figura 117 ESP32-S3 Series. Tabla resumen de las funcionalidades en los diferentes modos de operación.

En base a esta familia de microcontrolador se encuentran dos placas principales con pequeñas variaciones:

ESP32-S3-DevKitM-1 [36]: Se trata de una placa de desarrollo equipada con un ESP32-S3-MINI-1 o bien un ESP32-S3-MINI-1U que se diferencian entre sí por la antena empleada para las comunicaciones Wi-Fi y Bluetooth. En el primero la antena está construida en el propio encapsulado mientras en el modelo 1U presenta un conector para que se acople una antena externa como se ve en la Figura 118.



ESP32-S3-MINI-1



ESP32-S3-MINI-1U

Figura 118 ESP32-S3 Series. ESP32-S3-DevKitM-1. Diferencias entre los dos encapsulados del microcontrolador.

El encapsulado anterior recoge el microcontrolador, el oscilador y algunos componentes discretos como condensadores, bobinas y resistencias. El esquema del circuito interno de la placa que reúne estos componentes es el mostrado en la Figura 119.

Las salidas de la placa de desarrollo son las indicadas en la Figura 120, donde se pueden ver además dos conectores micro USB de los cuales uno es empleado para comunicarse directamente con el microcontrolador (marcado como USB) y el otro emplea un CP2102N-A02-GQFN28 como interfaz entre el protocolo USB 1.1 y el protocolo UART para comunicarse con el ESP32-S3.

El resto de los componentes del circuito son los mismos que en las placas homónimas formadas por un ESP32-S2.

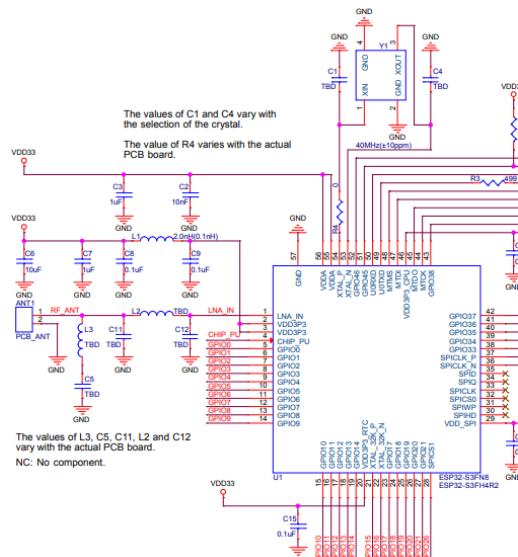


Figura 119 ESP32-S3 Series. ESP32-S3-DevKitM-1. Esquema eléctrico del encapsulado ESP32-S3-MINI-1.

ESP32-S3-DevKitM-1

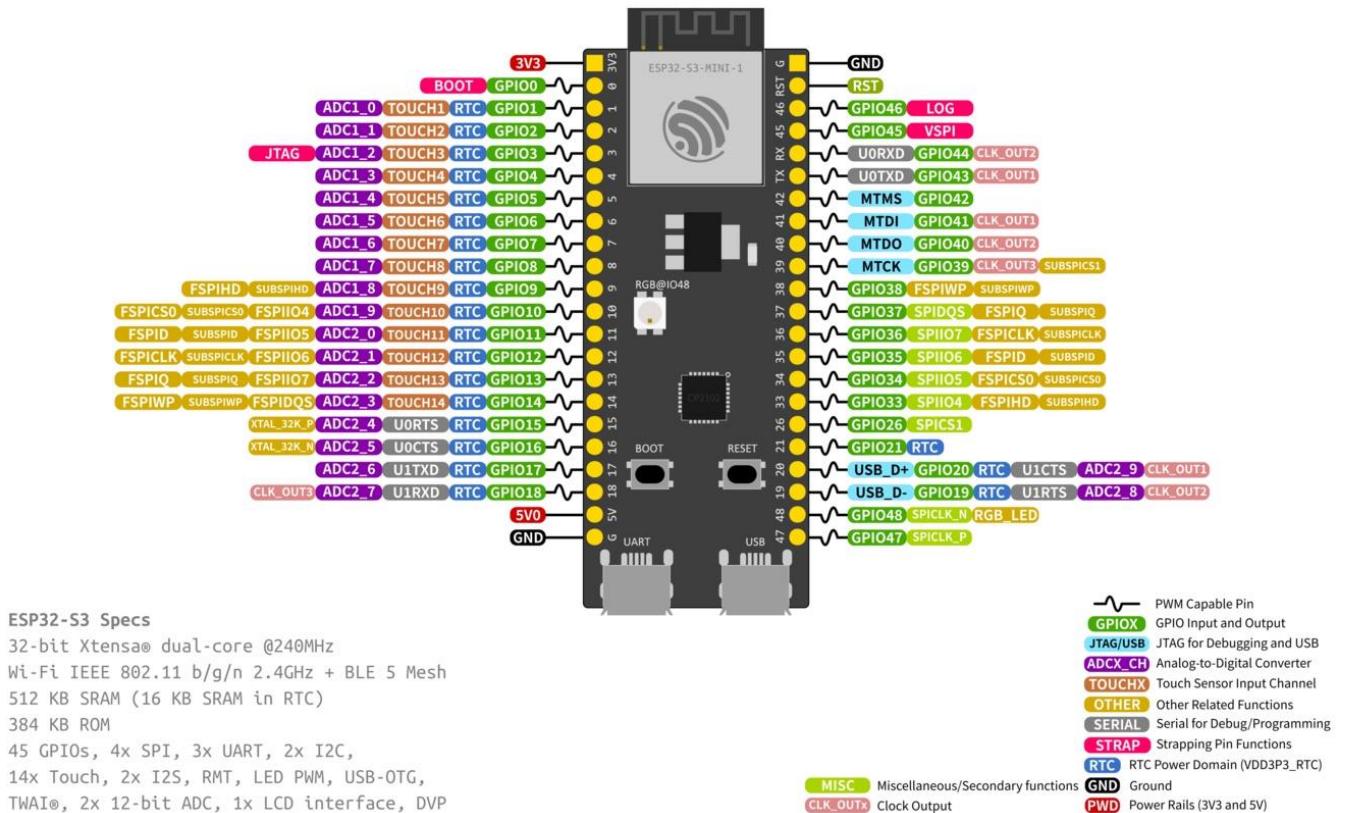


Figura 120 ESP32-S3 Series. ESP32-S3-DevKitM-1. Pinout.

ESP32-S3-DevKitC-1 [37]: Se trata de un modelo muy similar al anterior salvo porque emplea un encapsulado diferente para el MCU, un ESP32-S3-WROOM. Este nuevo encapsulado contiene el CI del microcontrolador junto con el oscilador, resistencias, bobinas, condensadores y una pequeña memoria flash de hasta 16 MB que se comunica con el microcontrolador empleando un puerto SPI. Esto se ve reflejado en el esquema eléctrico de la Figura 121. Dado que se emplean varios pines para comunicarse con la memoria flash, el número total de GPIOs disponibles para su uso en la placa de desarrollo disminuye hasta 36. El Pinout de la placa se muestra en la Figura 122, en la que se pueden observar algunas diferencias con el Pinout de la placa anterior.

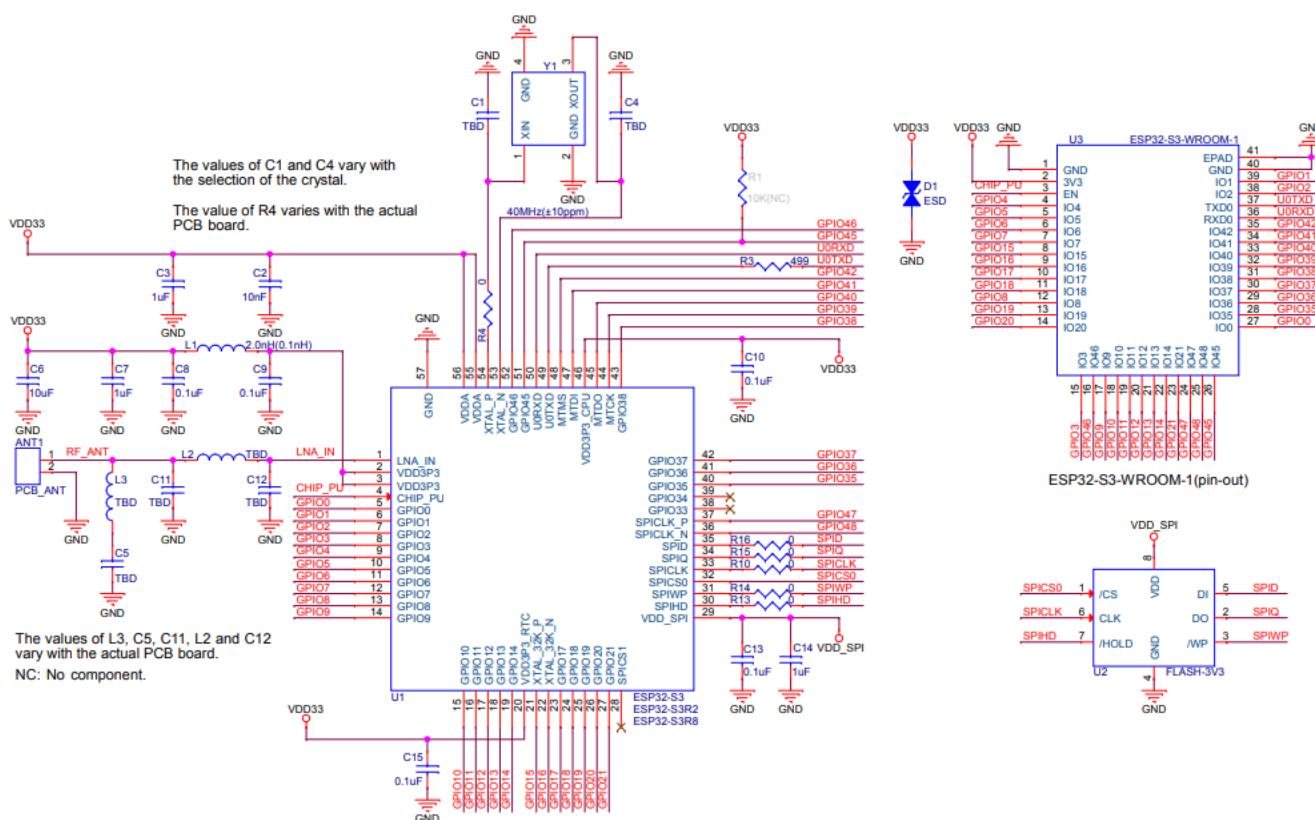
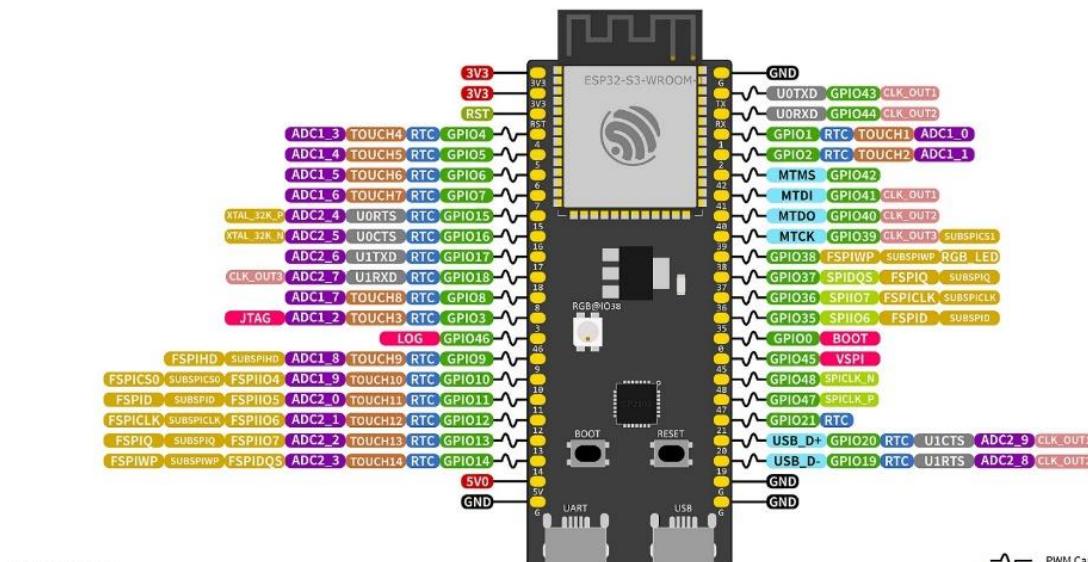


Figura 121 ESP32-S3 Series. ESP32-S3-DevKitC-1. Esquema interno del ESP32-S3-WROOM.

ESP32-S3-DevKitC-1



	PWM Capable Pin
	GPIOx Input and Output
	JTAG/USB JTAG for Debugging and USB
	ADCx_CH Analog-to-Digital Converter
	TOUCHx Touch Sensor Input Channel
	OTHER Other Related Functions
	SERIAL Serial for Debug/Programming
	STRAP Strapping Pin Functions
	RTC RTC Power Domain (VDD3P3_RTC)
	GND Ground
	PWD Power Rails (3V3 and 5V)

Figura 122 ESP32-S3 Series. ESP32-S3-DevKitC-1. Pinout.

Capítulo 2.5.7. Otras placas de desarrollo.

Además de las opciones estudiadas anteriormente, existen otras placas de desarrollo que empleando un microcontrolador podrían cumplir con las funcionalidades necesarias para este proyecto.

Raspberry Pi Pico Series:

Se trata de una serie de placas de desarrollo basadas en el microcontrolador RP2040. Este chip de dos núcleos que puede trabajar con una velocidad de hasta 133MHz. Cuenta con una memoria RAM incorporada de 264KB pero sin una memoria flash interna, como algunos de los dispositivos ya vistos anteriormente. El PR2040 viene con multitud de pines GPIO así como puertos UART, SPI, I2C, PWM y ADC. Incluye también un controlador USB 1.1 que permite comunicar directamente el microcontrolador con un ordenador sin necesidad de un chip adicional. [38]

De esta serie existen cuatro placas que se pueden observar en la Figura 123, donde de izquierda a derecha están Raspberry Pi Pico, Pico H, Pico W y Pico WH.

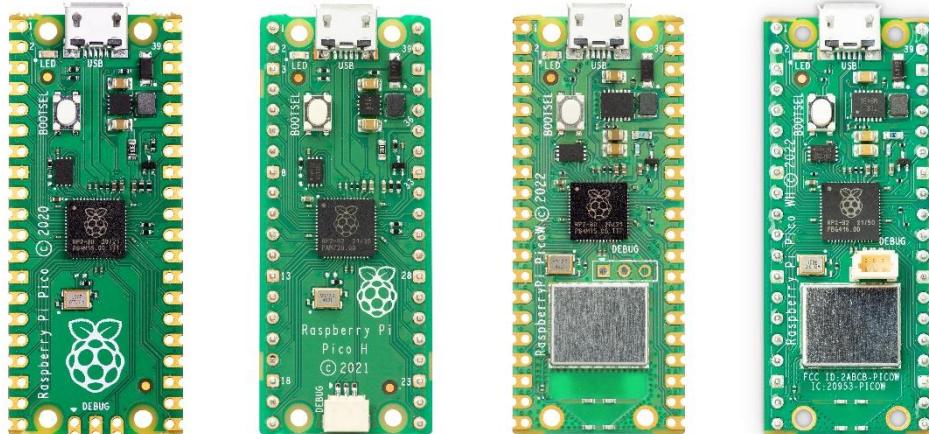


Figura 123 Raspberry Pi Pico Series. Placas de desarrollo de la serie.

Las dos primeras placas presentan las características del propio microcontrolador, resultando en 26 puertos GPIO, 2 puertos SPI, 2 puertos I2C, 2 interfaces UART, 3 canales ADC de 12-bits de resolución y 16 canales PWM de salida. En la Figura 124 se ve una descripción más detallada de los pines donde se indican también las funciones que pueden realizar. Se diferencian entre ellas en la forma de la placa, pues la Pico está pensada para ser soldada sobre otra placa y para conectarla empleando conectores de agujero pasante mientras que la Pico H solo puede ser conectada mediante estos conectores.

Las placas Pico W y Pico WH presentan una interfaz inalámbrica de 2.4G compatible con el protocolo 802.11n empleando para ello el módulo CYW43439 que permite comunicaciones Bluetooth 5.2 con modos de bajo consumo (BLE). En la Figura 125 se observa una descripción detallada de los pines y sus distintas funcionalidades, aunque son prácticamente idénticas a las de los modelos anteriores. Al igual que las dos placas anteriores, se diferencian en la forma de conectar las salidas del microcontrolador empleando soldadura sobre otra placa o bien conectores de agujero pasante.

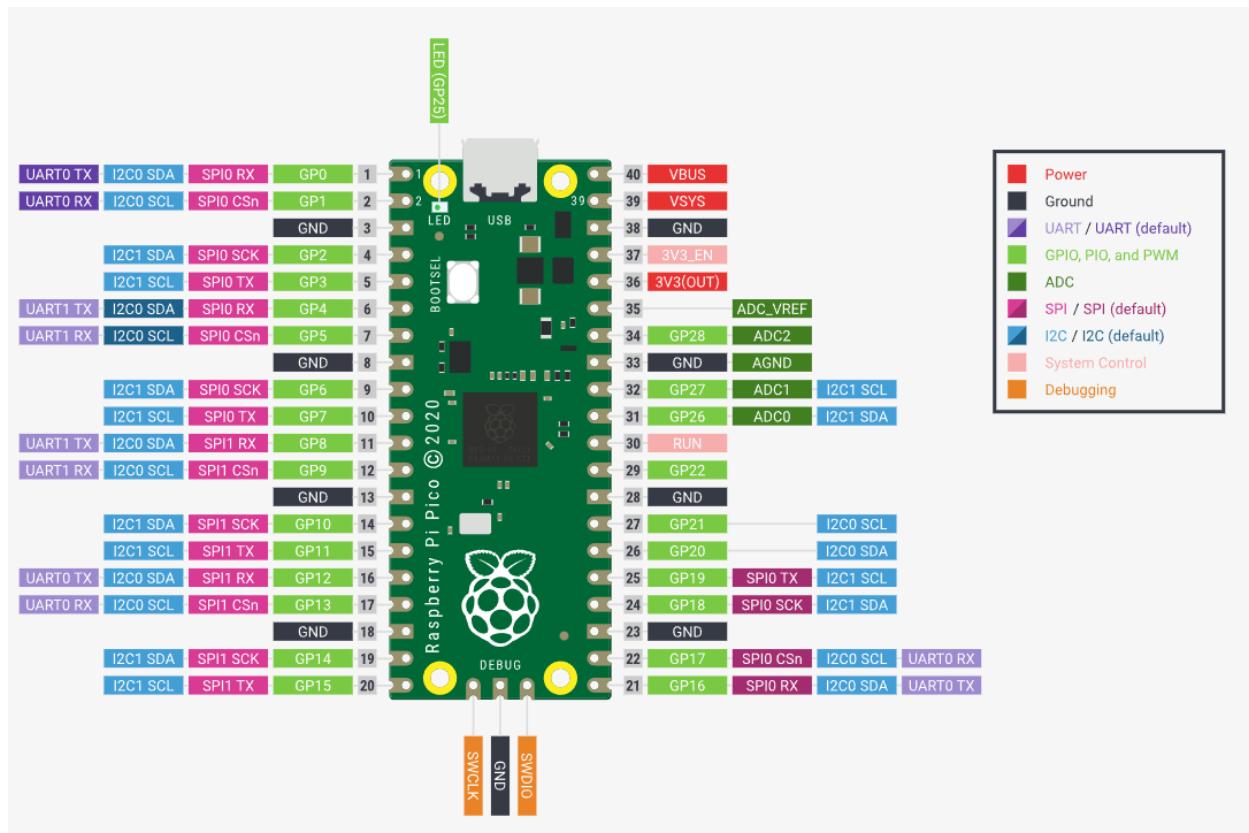


Figura 124 Raspberry Pi Pico y Pico H pinout.

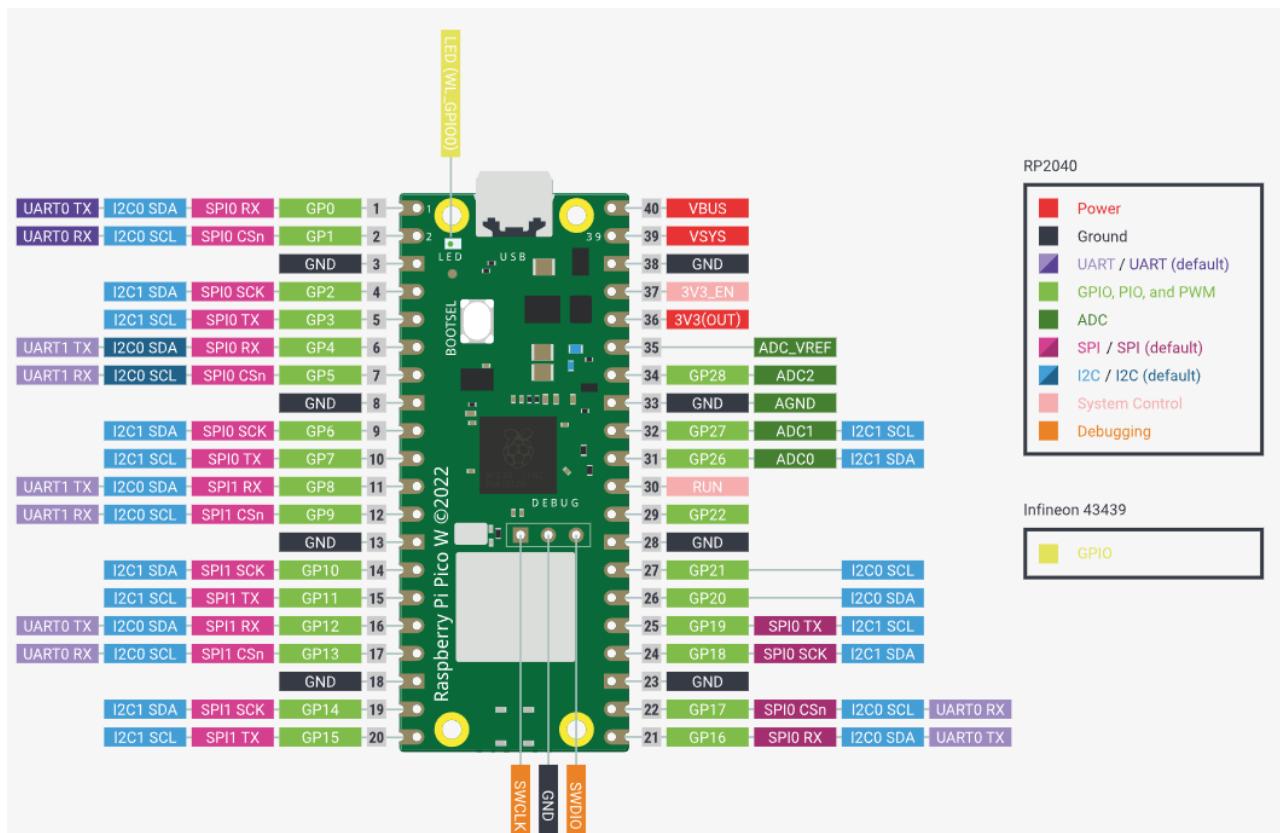


Figura 125 Raspberry Pi Pico W y Pico WH pinout.

Arduino WiFi R4:

El Arduino UNO R4 WiFi (Figura 126) fusiona el microprocesador RA4M1 de Renesas con el ESP32-S3 de Espressif. Integra comunicaciones Wi-Fi y Bluetooth gracias al ESP32 y cuenta con una matriz de LED de 12x8 integrada y un conector Qwiic para las comunicaciones I2C. [39]. En cuanto a los puertos presenta 14 entradas o salidas digitales con una corriente máxima de 8mA, 6 entradas analógicas, un conversor DAC, 6 pines PWM, 1 interfaz UART, 1 interfaz I2C, 1 interfaz SPI y 1 interfaz CAN (Figura 127). La velocidad de trabajo del RA4M1 es de hasta 48MHz, mientras que la del ESP32-S3 es mucho más elevada, 240MHz. Permite alimentarlo entre 6 y 24V.



Figura 126 Arduino Uno R4 WiFi. Vista frontal.

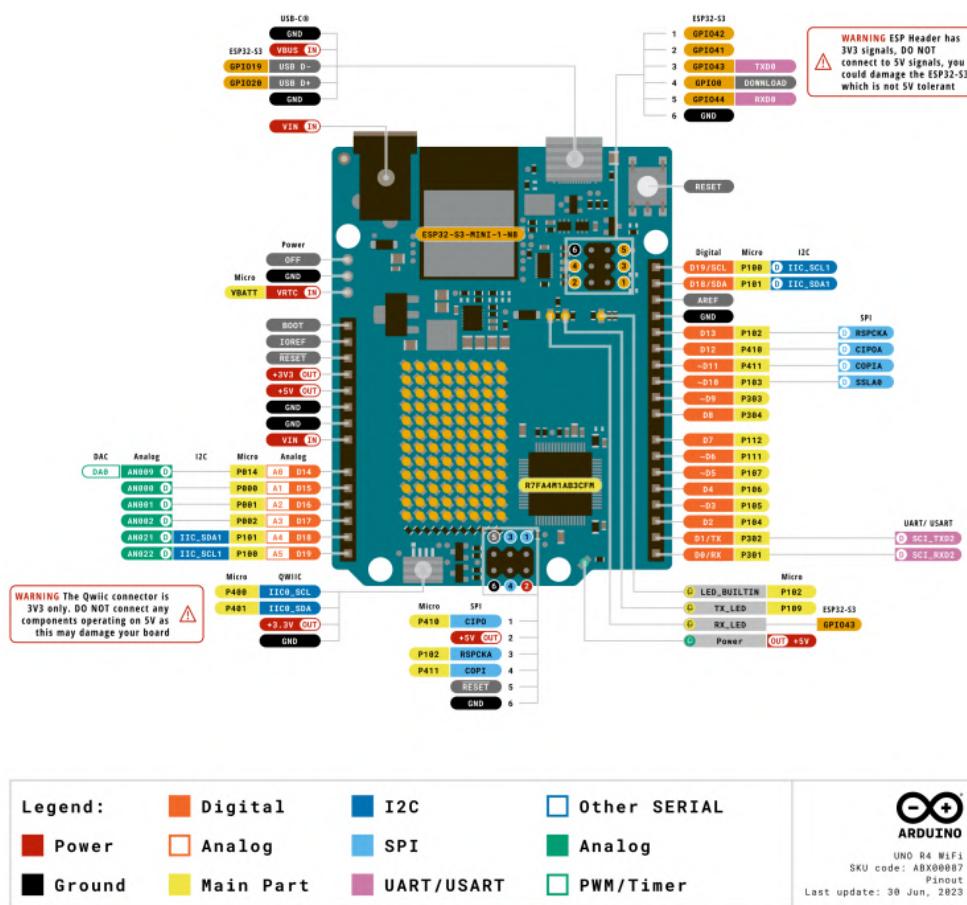


Figura 127 Arduino Uno R4 WiFi. Pinout.



Además de los microcontroladores anteriores, existen en el mercado otros que son bastante populares como los STM32 (Familia de microcontroladores de STMicroelectronics), MSP430 (Familia de microcontroladores de Texas Instruments) y los PIC (Familia de microcontroladores de Microchip) aunque en este proyecto no serán analizados.

En la Figura 128 se muestra una tabla comparativa de las principales características de las placas de desarrollo que han sido analizadas en mayor profundidad.

Placa	Arduino Nano Every	Arduino Leonardo	ESP32	ESP32-S2	ESP32-S3
μC	ATMega4809	ATMega32u4	ESP32	ESP32-S2	ESP32-S3
Nº núcleos	1	1	2	1	2
V(μC)	5V	5V	3.3V	3.3V	3.3V
Consumo μC máx	10mA	20mA	380mA	370mA	340mA
VIN	7-21V	6-20V (recomendado 7-12V)	5V	5V	5V
I_{max} 5V	950mA	800mA	-	-	-
I_{max} 3,3V	50mA	50mA	50mA	50mA	50mA
Frecuencia trabajo	Hasta 20MHz	16MHz	Hasta 240MHz	Hasta 240MHz	Hasta 240MHz
Flash	48KB	32KB (4KB bootloader)	Hasta 8MB	4MB	Hasta 8MB
SRAM	6KB	2.5KB	520KB	320KB	512KB
EEPROM	256B	1KB	-	-	-
ROM	-	-	448KB	128KB	384KB
Bajo consumo	3 modos	6 modos	5 modos	4 modos	3 modos
GPIO	14+8	14+6	26	39/36	39/36
V (GPIO)	5V	5V	3.3V	3.3V	3.3V
I_{max} GPIO	20mA	40mA	40mA	40mA	40mA
PWM	5	7	26	36/39	39/36
Resolución PWM	8 bits	8/10/16 bits	8 bits	8 bits	8 bits
Analog IN	8	6	16	20	20
Resolución ADC	10 bits	10 bits	12 bits	12 bits	12 bits
UART	1	1	3	2	3
SPI	1	1	3	4	4
I_CC	1	1	2	2	2
Interfaces Inalámbricas	-	-	Wi-Fi 2.4G Bluetooth 4.2	Wi-Fi 2.4G	Wi-Fi 2.4G Bluetooth 5
Sensores táctiles	-	-	10	14	14
Precio	12.5€	19.2€	9€	7.2€/9€	13.53€

Figura 128 Tabla resumen de las placas de desarrollo.

Capítulo 2.6. INCORPORACIÓN DE PERIFÉRICOS.

Capítulo 2.6.1. Hardware para la interfaz con el usuario.

Como se ha observado en el Capítulo 2.2 hay algunos modelos que incorporan una interfaz de control que permite al operario manipular los mensajes que se pueden mostrar por el panel luminoso. Esta interfaz está formada en algunos casos por una pantalla gráfica táctil o bien una botonera que permite seleccionar modos o introducir caracteres para que éstos se muestren en el panel.

Como opción más sencilla se podría implementar una interfaz con una botonera o membrana de botones (Figura 129) y una pequeña pantalla LCD (Figura 130) que permitiera elegir el mensaje deseado.



Figura 129 Membrane de botones.



Figura 130 Pantalla LCD.

Debido a las limitaciones de los elementos anteriores debido a su poca flexibilidad, se propone emplear una pantalla gráfica táctil que proporcionará mayores prestaciones al producto que una botonera. Se buscarán distintos componentes que proporcionen estos requisitos y que además sean compatibles con los controladores estudiados anteriormente.

De la marca Adafruit se pueden encontrar gran variedad de pantallas en función del tamaño y del controlador empleado. En este caso solo se estudiarán dos modelos.

Pantalla táctil LCD TFT de 3.2" con placa de conexión y conector MicroSD - ILI9341.

Se trata de una pantalla TFT (Thin Film Transistor) LCD (Liquid Crystal Display) de gran tamaño (3.2 pulgadas o 8,1cm en la diagonal) de alto brillo y a color con una pantalla de 240x320 píxeles RGB (Figura 131). Esta pantalla tiene una memoria RAM construida en el propio circuito por lo que el microcontrolador al que se conecta no consumirá prácticamente recursos para su control. Además, consta de cuatro salidas que gracias a la pantalla resistiva permitirá identificar la posición en la que se produce un toque [40]. La pantalla se puede usar en dos modos: 8-bit y SPI. Empleando el primer modo serán necesarias 8 líneas de datos y 4 o 5 líneas para escribir y leer del display (12 líneas en total). En el modo SPI solo son requeridas 5 líneas de datos y permite comunicarse con una tarjeta SD para mostrar imágenes, pero la comunicación es más lenta. Se necesitan además 4 pines para la pantalla táctil. En la placa de la Figura 132 se observan los



conectores para comunicación SPI en un lateral y los conectores para la comunicación de 8-bits en el otro. El precio de este periférico es de 34.95\$ (31.70€).



Figura 131 Adafruit 3.2" TFT LCD touchscreen – ILI9341. Vista frontal.

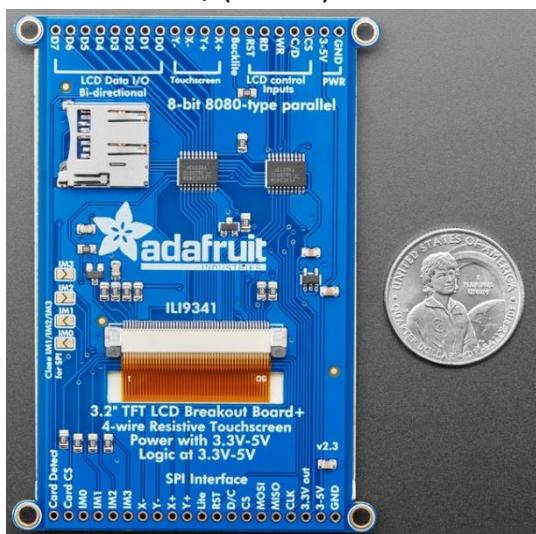


Figura 132 Adafruit 3.2" TFT LCD touchscreen – ILI9341. Vista trasera.

El esquema de la placa de desarrollo es el mostrado en la Figura 133, donde se muestra el CI del controlador de la pantalla, el ILI9341 y varios componentes.

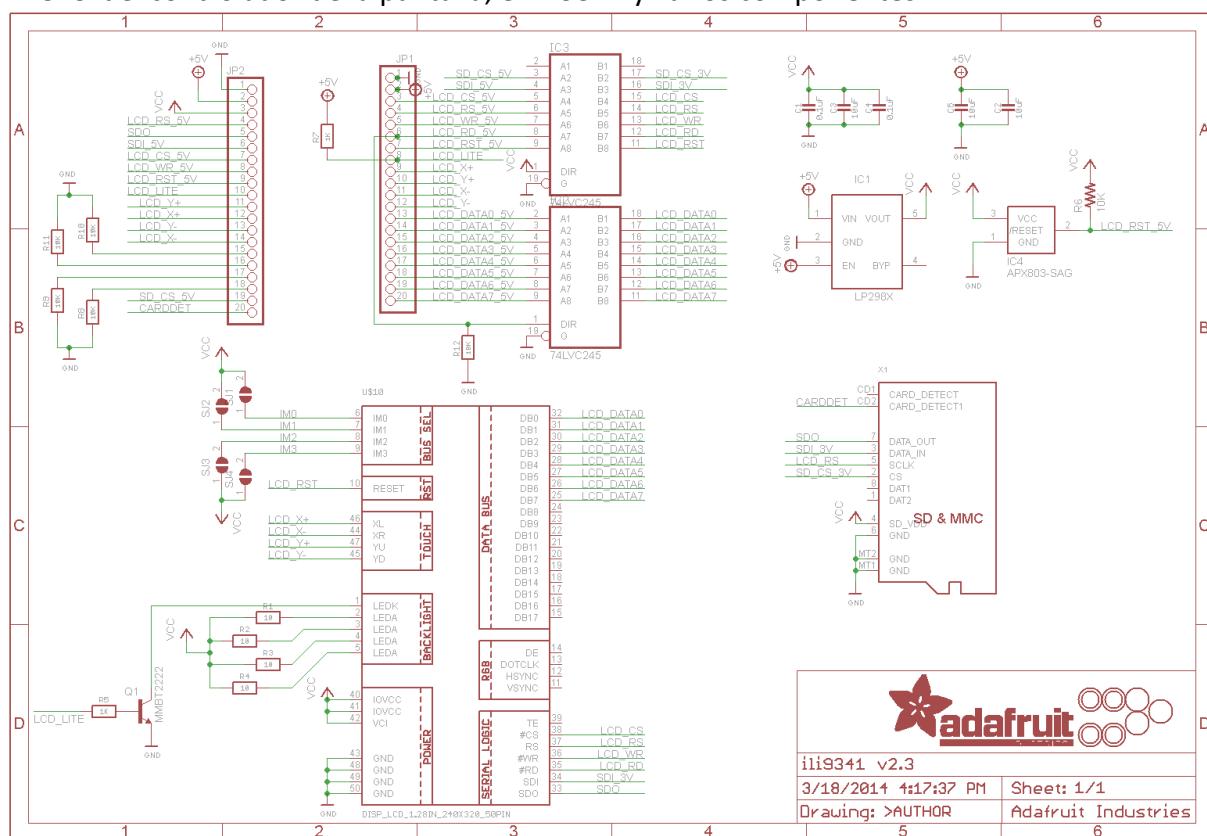


Figura 133 Adafruit 3.2" TFT LCD touchscreen – ILI9341. Esquema de conexiones.

El circuito integrado ILI9341 es el driver que controlará la pantalla TFT con una resolución de 240x320 puntos que tiene una memoria GRAM (Graphics RAM) 172800

bytes. Puede operar con tensiones de entre 1.65 y 3.3 voltios. Consta de dos modos de funcionamiento de bajo consumo, en reposo ligero y profundo.

Los CI 74LV245 son transceptores de ocho canales y tres estados que se emplean para transferir los datos en ambos sentidos entre dos buses de datos con diferentes voltajes. El CI LP2980 es un regulador de bajo voltaje de hasta 50mA que tiene como salida una tensión de 3.3V.

Por último, aparece un APX803-SAG que se emplea para la supervisión de microprocesadores en la monitorización de las fuentes de alimentación. Este chip tiene la función de enviar una señal de reset al microcontrolador cuando se detecta una sobretensión o la tensión disminuye demasiado, protegiendo al sistema.

Pantalla táctil LCD TFT de 3.5" con placa de conexión y conector MicroSD - HXD8357D.
 Se trata de un modelo muy similar al estudiado previamente, salvo que el driver que emplea para el control de la pantalla táctil es el HXD8357. En cuanto a forma es un poco más grande (3.5") como se puede observar en la Figura 134 y la circuitería adicional se puede observar en la Figura 135 y en los esquemas de la Figura 136. Otra de las diferencias que se observan con el modelo anterior es que tiene un circuito adicional formado por un FAN5333BSX y que los LEDs traseros de la pantalla se alimentan a 24V [41].



Figura 134 Adafruit 3.5" TFT LCD touchscreen – HXD8357D. Vista frontal.

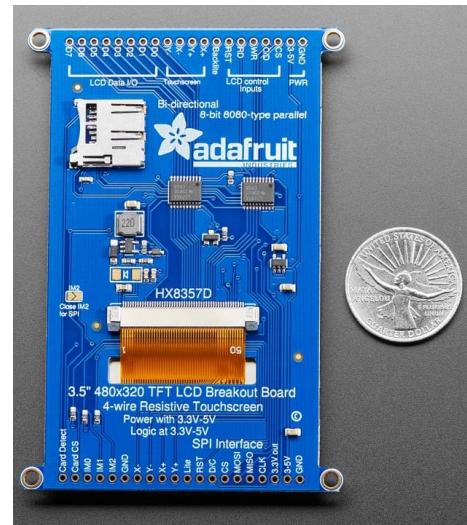


Figura 135 Adafruit 3.5" TFT LCD touchscreen – HXD8357D. Vista trasera.

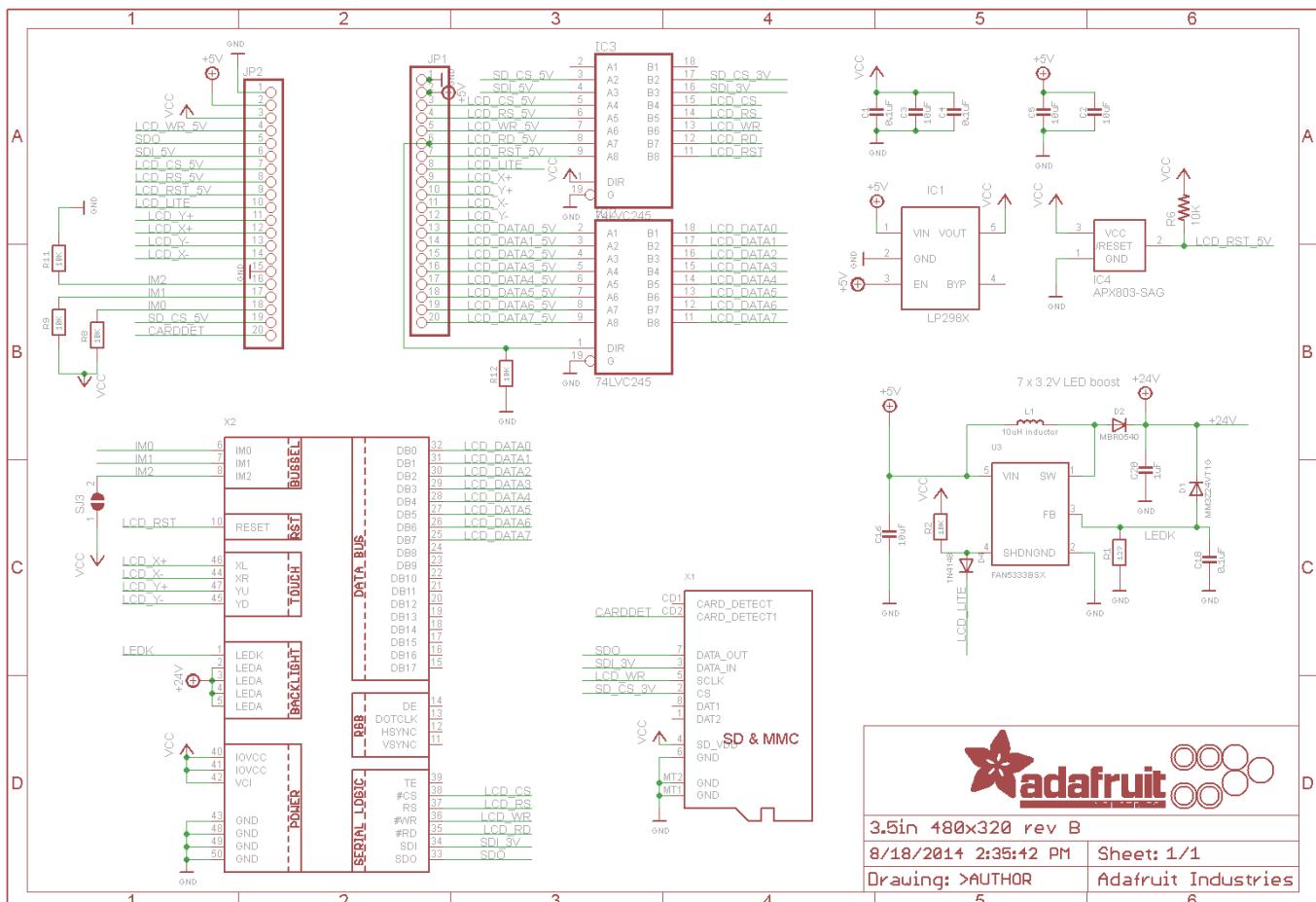


Figura 136 Adafruit 3.5" TFT LCD touchscreen – HXD8357D. Esquemas eléctricos.

El FAN5333BSX es un driver de propósito general que se emplea para alimentar LEDs a altas frecuencias de conmutación y a elevadas potencias minimizando el ruido de conmutación. De esta forma se consigue modificar el brillo de la pantalla empleando un único pin.

El precio indicado en la propia web del fabricante es de 39.95\$ (36.23€).

Existen otros modelos similares a los anteriores, pero más asequibles, como la pantalla de 3.5 pulgadas de la Figura 137.



Figura 137 Pantalla TFT SPI 480x320. 3.5".

Capítulo 2.6.2. Sensor de intensidad luminosa.

Para poder medir la luz externa y poder variar la intensidad luminosa del panel de forma automática será necesario un sensor que mida este parámetro. Dado que no se necesita una gran precisión ni sensibilidad se plantea usar una resistencia variable con la luz, fotorresistor o LDR (Figura 138) empleado en un divisor de tensión que mida la variación de este parámetro mediante una de las entradas analógicas del microcontrolador. Un esquema típico de montaje para esta aplicación es el de la Figura 145, donde a mayor luz recibida en la fotorresistencia, mayor será la tensión de salida pues menor será la resistencia de este componente. El valor de $10\text{k}\Omega$ de la resistencia inferior es elevado para que el consumo sea menor.



Figura 138 LDR.

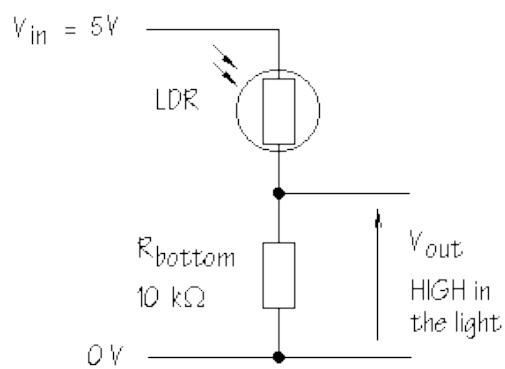


Figura 139 Divisor de voltaje empleando una LDR.

Capítulo 2.6.3. Accionador para inclinar el panel.

Como una de las características estudiadas es la posibilidad de recoger el panel luminoso si no se desea mostrar ningún mensaje. Una de las opciones estudiadas ha sido emplear un sistema piñón cremallera mediante un accionador que permita rotar sobre un eje al panel luminoso de tal forma que quede oculto si no se quiere usar. Como accionador se puede emplear un pequeño motor paso a paso, en este caso el 28byj-48 (Figura 140) que mediante un driver formado por transistores Darlington puede ser fácilmente controlado mediante un microcontrolador. Este tipo de motor permite saber en qué posición angular se encontraría el motor mediante el número de pasos o movimientos completados en el motor.



Figura 140 Motor paso a paso 28byj-48 con driver.

Otra de las opciones más comunes podría ser emplear un motor de corriente continua con una reductora que permita transmitir un mayor par. Un posible modelo de pequeña escala podría ser el motor JGA16-050-12120 de la Figura 141, que tiene una reductora que proporciona bajas revoluciones y alto par en el extremo del eje. Este motor se alimentará con una tensión de 6V y su control se podría realizar con un puente H como el L9110S (Figura 142). La dificultad de este montaje consistiría en controlar la posición exacta del eje debido a que solo se puede controlar la dirección de forma directa. Para solventar esto se podría implementar un sensor tipo encoder para medir su posición angular y detectar con qué inclinación está configurado el panel.



Figura 141 Motor CC JGA16-050-12120.



Figura 142 Doble puente H para el control de motores de CC.

Otro posible accionador sería emplear un servomotor de alto par que incluye el control en posición de forma directa. Un posible modelo podría ser el MG995 de la Figura 143, que transmite un par de 12kg/cm. El control de la posición de este tipo de accionamientos es mediante un PWM cuyo ciclo de servicio permitirá modificar la posición angular del eje del servomotor (Figura 144).



Figura 143 Servomotor MG995.



Figura 144 Servomotor MG995. Conexiones.

Capítulo 2.7. ALIMENTACIÓN DEL PROTOTIPO.

Partiendo de que el prototipo se diseñará para la implementación en un vehículo y que la electrónica de éste se alimenta por la batería instalada en su interior, se deben analizar las tensiones nominales más comunes en el mercado actual.

Por lo general, en los coches de combustión convencionales la batería es de 12V y en algunos vehículos más grandes, como los camiones; se emplean baterías de 24V para proporcionar una mayor potencia.

En cuanto a los coches eléctricos, presentan una batería principal que se encarga de dar tracción a los motores cuya tensión varía normalmente entre los 360V y los 400V y una batería secundaria de 12V de menor tamaño que se encarga de alimentar al resto de circuitos del vehículo como las luces, la dirección asistida, los limpiaparabrisas, la radio, etc. [42]

En algunos coches híbridos donde el motor eléctrico apoya al motor de combustión se suelen emplear baterías de 48 voltios que son más livianas y permiten aportar una mayor potencia en la tracción. Al igual que en el caso anterior, pueden contar con una segunda batería de 12 voltios que alimente al resto de circuitos del sistema eléctrico. [42]

Por tanto, si se emplean baterías para alimentar al producto, es necesario buscar soluciones para obtener una tensión de 5V a partir de tres casos posibles, tensiones de en torno a 12, 24 y 48 voltios.

Para obtener esta característica se pueden emplear reguladores de voltaje lineal o convertidores reductores continua-continua de alta eficiencia para evitar pérdidas de potencia. Para ello se debe hacer una estimación del consumo del prototipo, por lo que atendiendo a los capítulos anteriores se puede resumir el consumo en la tabla de la Figura 145 donde el consumo máximo posible es de aproximadamente 3.7 amperios.

Componente	Tensión [V]	Consumo máximo [mA]
Panel LED	5	160mA/modulo·20 módulos=3200
Microcontrolador	3.3	400
Pantalla Táctil	3.3	90
LDR	5	0.5
TOTAL	-	3690.5

Figura 145 Tabla resumen de consumos máximos del prototipo.

Dado el valor elevado de consumo de este prototipo, no es posible emplear un regulador de voltaje lineal (LDO) ya que estos están pensados para potencias mucho menores. Por tanto, se deberá emplear un convertidor reductor continua-continua que cumpla con estas necesidades.

Tras una pequeña búsqueda se encuentra el CI MAX20008EAFOC/VY+, un convertidor Buck que puede suministrar hasta 8A con voltajes de entrada desde 3.5V hasta 36V empleando para ello una corriente sin carga de 15µA. Esta serie ofrece tensiones de salida fijas de 3.3V, 3.9V y 5V y tensiones ajustables entre 1V y 5V empleando componentes discretos. Se trata de un modelo diseñado para aplicaciones punto a punto en automoción, sistemas de distribución de potencia en corriente continua y para sistemas de radio y navegación. Por estas características se podrá emplear el mismo convertidor para automóviles cuya batería sea de 12V o de 24V, dándole bastante flexibilidad al producto final. Una aplicación típica para este modelo es el mostrado en la Figura 146 donde se muestran los condensadores y resistencias necesarias. [43]

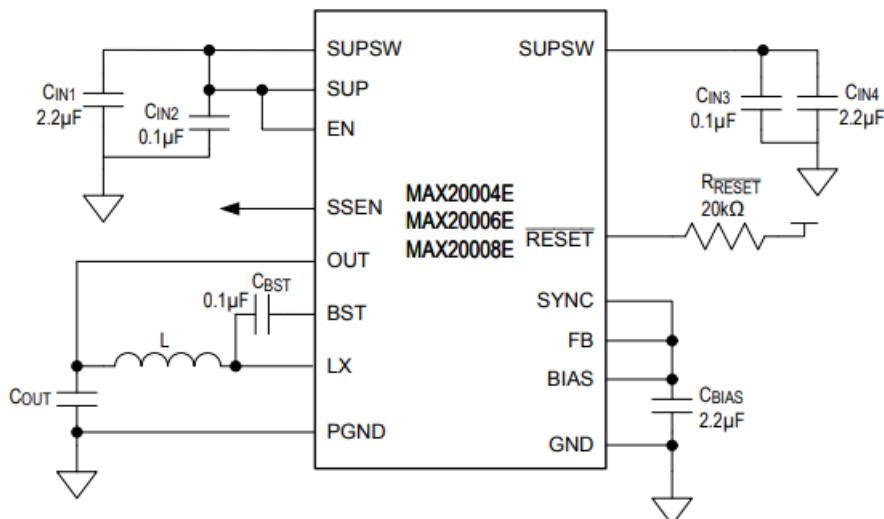


Figura 146 Esquema eléctrico típico del buck MAX20008EAFOC/VY+.

Los valores sin parametrizar del circuito anterior dependerán de las tensiones de entrada y de salida, así como la corriente de salida. Estos valores se obtendrán consultando las ecuaciones de diseño descritas en la hoja de características del componente.

Por otra parte, existen en el mercado tarjetas de desarrollo con los componentes necesarios basados en el CI XL4005 que son capaces de proporcionar 5A (Figura 147 y Figura 148).



Figura 147 Placa con Buck DC-DC de 5A basado en XL4005.

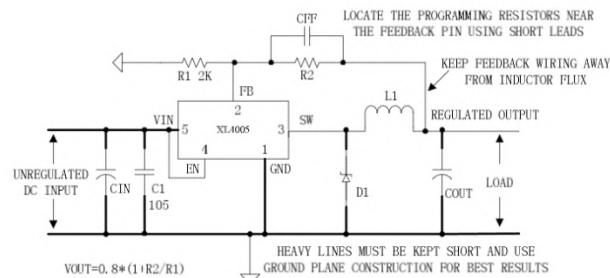


Figura 148 Esquema placa Buck basada en XL4005.

Otra de las posibilidades de alimentar este proyecto es a partir de una toma de corriente alterna, empleando para ello un adaptador AC/DC que emplee un transformador y un circuito de regulación que a su salida proporcione una salida estable de 5V con la corriente necesaria. Un modelo válido podría ser adaptador RS Pro CLASS I Desktop Power Supply 66JYH4Z-0500800-BY-RS de 40W mostrado en la con una salida de 5V cuya corriente máxima es de 8A. Este modelo tiene un precio de 28.62€ sin contar con el cable de alimentación a corriente alterna, que deberá incluir el conector macho IEC-320-C14 [44].



Figura 149 Adaptador AC/DC marca RS Pro. Modelo 66JYH4Z-0500800-BY-RS.

Capítulo 2.8. ELEMENTOS SELECCIONADOS.

En cuanto a los paneles LED investigados en el Capítulo 2.4 se decide emplear varios módulos basados en los controladores MAX7219 descritos en el Capítulo 2.4.4 debido a que presentan una interfaz SPI que permite encadenar varios de estos modelos y formar un panel de grandes dimensiones a bajo coste. Dado que esta característica es modular, se podrá plantear el proyecto para que el panel pueda variar en dimensiones cambiando únicamente algunos parámetros en la programación, empleando para ello las interfaces propuestas con el usuario. Estos controladores tendrán seleccionada una resistencia para LEDs rojos con una tensión de 1.6V y una corriente máxima de 20mA por segmento con una resistencia de 29.8 Ohmios, por lo que el consumo máximo de cada matriz de LEDs será de 160mA.

En este caso se emplearán 20 de estos módulos para formar un panel de dos filas y diez columnas de matrices LED de 8x8 puntos, formando un panel de 16x80 píxeles, permitiendo así mostrar caracteres en una sola fila en un tamaño mayor (empleando las dos franjas horizontales) o en dos filas para mostrar frases en dos alturas.

Una desventaja que tiene elegir este tipo de módulos es que permite mostrar mensajes en un solo color, en este caso rojo. Es por esto que como mejora del producto se podría emplear alguno de los otros modelos estudiados que permitan mostrar varios colores, aunque aumentando el coste y complejidad del producto.

En cuanto a los microcontroladores estudiados en el Capítulo 2.5 se escoge la placa de desarrollo basada en ESP32 debido a que presenta interfaces inalámbricas Wi-Fi y Bluetooth a diferencia del resto de placas estudiadas. En concreto se usará la placa ESP32-WROOM-32E ya que se trata de una de las placas más baratas y que permite cumplir con todas las especificaciones del producto, además se tiene familiaridad con este producto gracias a trabajos previos en este ámbito.

La pantalla elegida es el modelo más barato de los estudiados, el formado por el controlador ILI9488; ya que no presenta mucha diferencia entre los demás objetos estudiados y ya que gracias a su conexión SPI se podrán emplear pocos puertos del microcontrolador.

En cuanto al actuador que permitirá inclinar el panel luminoso será el servomotor MG995, ya que tras varias pruebas realizadas se ha concluido que es el más fácil de controlar y con el par necesario para inclinar el panel mediante el mecanismo de piñón cremallera.

Como sensor de luz se incorporará un divisor de tensión formado por la LDR descrita anteriormente de tal forma que el sensor esté colocado en el exterior del panel luminoso.

Para la alimentación del prototipo se decide emplear el convertidor continua-continua de la placa basada en el CI XL4005, que permite lograr el nivel de tensión de 5V para la carga máxima de este dispositivo sin suponer un alto coste.

Capítulo 2.9. CONEXIONES.

Las conexiones del prototipo se ven reflejadas en el esquema eléctrico de la Figura 150, donde se han añadido elementos externos como conectores JST y un conector tipo Barrel Jack de 2.1x5.5mm para la alimentación.

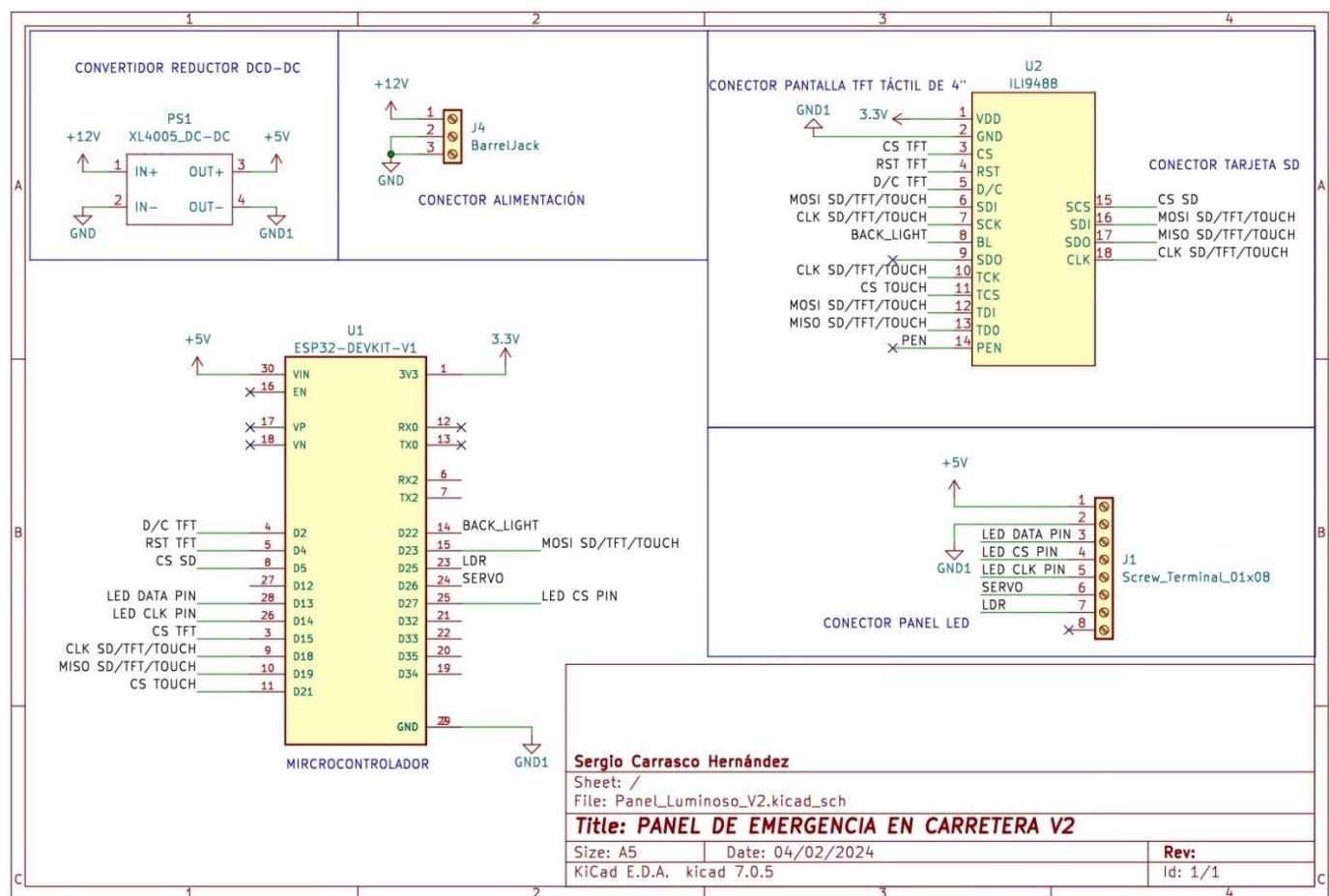


Figura 150 Esquema de conexiones del prototipo para el panel de emergencia en carretera.

Adicionalmente al esquema anterior, se suma un esquema para una pequeña placa de conexión que permitirá conectar tanto el actuador, los sensores y el panel de emergencia usando una única manguera. El esquema de conexiones es el mostrado en la Figura 151. En este esquema se observarán conectores para el servomotor, los módulos del panel luminoso y el divisor de tensión formado por la LDR para poder medir la luz que hay en el ambiente.

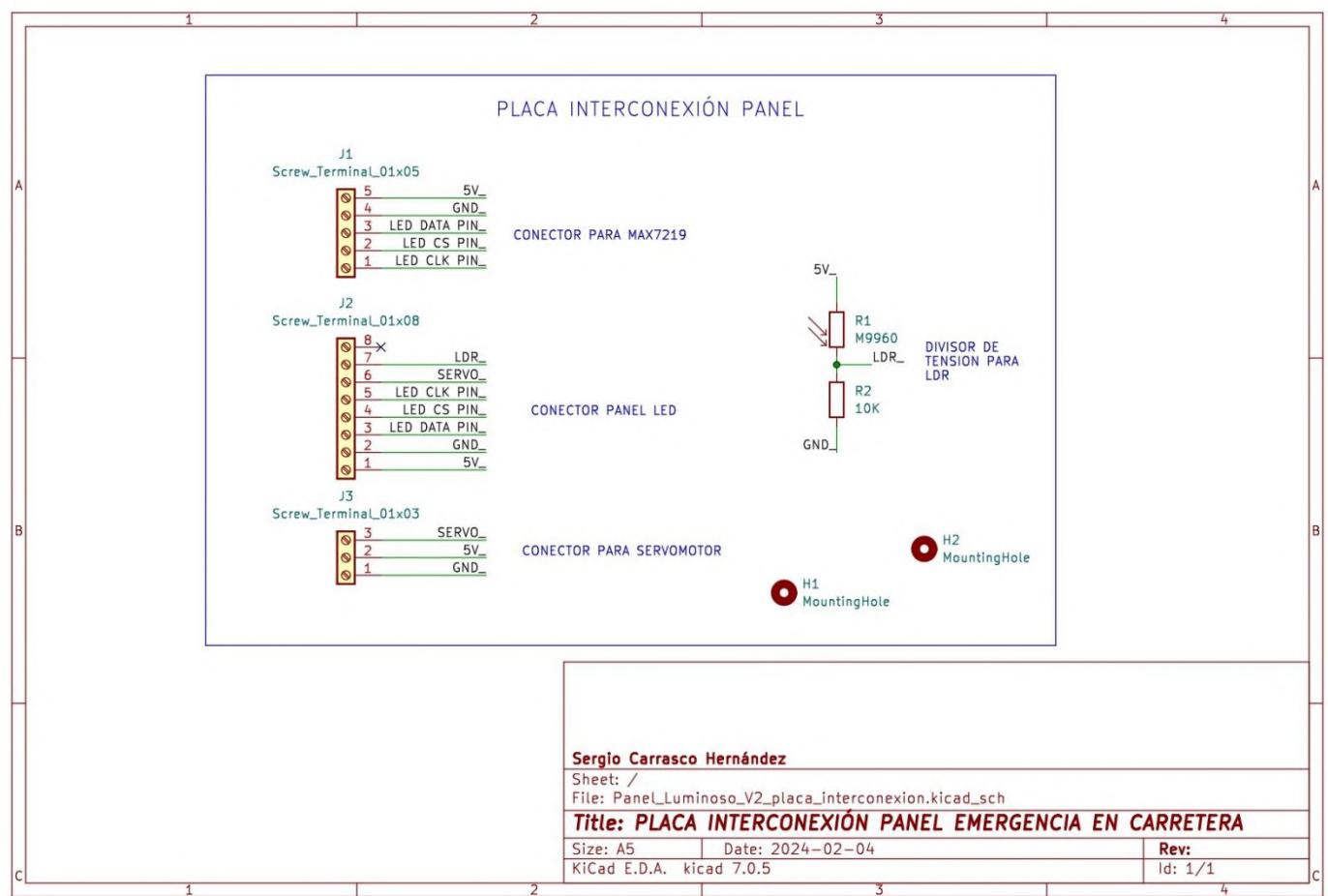


Figura 151 Esquema eléctrico de la placa de interconexión.

Con este tipo de montaje lo que se busca es que el microcontrolador, la pantalla TFT y los circuitos de alimentación queden reducidos en una placa de circuito impreso (PCB) del que salga un conector con una manguera de ocho hilos que lleve la tensión necesaria para el panel luminoso, el servomotor y el divisor de tensión, así como sus señales. De esta forma se obtendrá un prototipo más cómodo de manejar y de colocar en la parte superior del vehículo sin necesidad de colocar demasiadas mangas de cables para cada uno de los distintos elementos.

Capítulo 3. PROGRAMACIÓN DEL PROTOTIPO.

Se comenzará analizando las principales funciones de las librerías empleadas para el control de cada uno de los distintos elementos que forman el panel luminoso de emergencia en carretera.

Capítulo 3.1. LIBRERÍAS PARA EL CONTROL DEL PANEL LUMINOSO MEDIANTE MAX7219.

Para controlar el panel LED basado en los circuitos integrados MAX7219 se empleará la librería MD_Parola [45] (Versión 3.7.1) que permite una fácil implementación del control empleando funciones para justificar el texto, efectos de desplazamiento, control en las animaciones y en la velocidad y múltiples filas del display entre otras funcionalidades. Esta librería se basa a su vez en la librería MD_MAX72xx [46] del mismo autor que implementa funciones básicas que permiten al usuario controlar las matrices de 64 LEDs como un dispositivo de píxeles.

Para probar esta librería se carga inicialmente el programa de ejemplo *Parola_Double_Height_Clock* que permite configurar un reloj en un display de doble altura. Tras configurar los pines referentes al bus SPI del microcontrolador se debe establecer el tipo de hardware que se está empleando y la forma en la que se conectan las diferentes filas del panel (Figura 152 y Figura 153) pues esto afectará en la forma en la que se mostrará la información en el panel. En este caso el montaje del panel será siguiendo la forma en Z que permitirá añadir mayor altura al panel empleando consecutivamente la misma estructura en las conexiones.

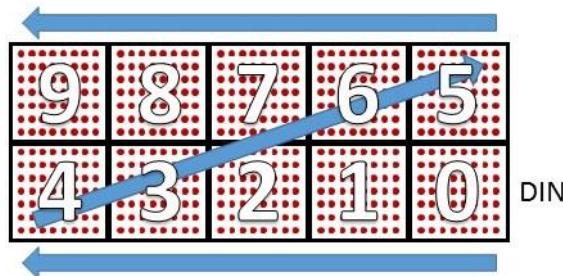


Figura 152 Unión de los módulos MAX7219 en forma de Z.

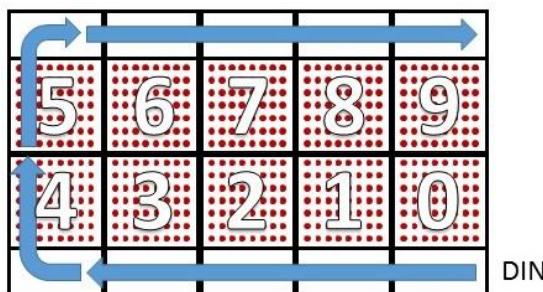


Figura 153 Unión de los módulos MAX7219 en forma escalonada.

Con esta forma de construcción se debe declarar el tipo de hardware como FC16_HW de tal forma que el reloj se muestre de forma correcta sobre el panel como en la Figura 154 y no se obtengan fallos en la visualización de los caracteres o símbolos como en Figura 155.



Figura 154 Reloj de doble altura con tipo de hardware FC16_HW.



Figura 155 Reloj de doble altura con tipo de hardware: PAROLA_HW.

Analizando el archivo de la cabecera de la librería se obtienen las diferentes funciones con las que se puede mostrar información en el panel. En la tabla de la Figura 156 y Figura 157 se encuentran recogidas las funciones principales de esta librería con una breve descripción del propósito que tienen.

FUNCIÓN O PARÁMETRO	DESCRIPCIÓN
PA_LEFT PA_CENTER PA_RIGHT	Diferentes posiciones en las que se puede alinear el texto.
PA_NO_EFFECT PA_PRINT PA_SCROLL_UP PA_SCROLL_DOWN PA_SCROLL_LEFT PA_SCROLL_RIGHT PA_SLICE PA_MESH PA_FADE PA_DISSOLVE PA_BLINDS PA_RANDOM PA_WIPE PA_WIPE_CURSOR	Diferentes efectos a la hora de mostrar el texto.
PA_SCAN_HORIZ PA_SCAN_HORIZX PA_SCAN_VERT PA_SCAN_VERTX PA_OPENING PA_OPENING_CURSOR PA_CLOSING PA_CLOSING_CURSOR PA_SCROLL_UP_LEFT PA_SCROLL_UP_RIGHT PA_SCROLL_DOWN_LEFT PA_SCROLL_DOWN_RIGHT PA_GROW_UP PA_GROW_DOWN	Diferentes efectos para usar en las zonas (filas) del display.
MD_Parola(MOD, MOSI_PIN, CK_PIN, CS_PIN, NumDev) MD_Parola(MOD, CS_PIN, NumDev) MD_Parola(MOD, SPIClass, CS_PIN, NumDev)	Constructores de la clase MD_Parola.
bool begin() bool begin(numZones);	Inicializadores del objeto.

Figura 156 Tabla resumen de las funciones de la librería MD_Parola.



FUNCIÓN O PARÁMETRO	DESCRIPCIÓN
<code>bool displayAnimate();</code>	Función para animar todas las zonas en el display según los parámetros definidos con anterioridad.
<code>void displayClear() void displayClear(z)</code>	Limpia todas las zonas en el display o la indicada en el argumento.
<code>void displayReset() void displayReset(z)</code>	Reinicia la animación de todas las zonas o de la zona indicada en el argumento.
<code>void displayShutdown(bool b)</code>	Apaga hardware del display. Si se llama a animate() se reinicia.
<code>void displaySuspend(bool b)</code>	Suspender o resume las actualizaciones del display.
<code>bool isAnimationAdvanced()</code>	Comprueba si la animación es de tipo avanzado.
<code>inline void getZone(z, &moduleStart, &moduleEnd)</code>	Obtiene los límites de una zona.
<code>bool setZone(z, moduleStart, moduleEnd);</code>	Define los límites de una zona.
<code>inline void displayScroll(*pText, align, effect, speed)</code>	Crea un display con el efecto de desplazamiento deseado.
<code>inline void displayText(*pText, align, speed, pause, effectIn, effectOut)</code>	Crea un display con el efecto de entrada y de salida deseado.
<code>void displayZoneText(z, *pText, align, speed, pause, effectIn, effectOut)</code>	Crea un texto sobre la zona con el efecto de entrada y de salida deseado.
<code>inline uint8_t getCharSpacing() inline uint8_t getCharSpacing(z)</code>	Obtiene el espacio entre caracteres de todo el display o de una zona en número de columnas.
<code>inline bool getInvert() inline bool getInvert(z)</code>	Obtiene el estado de inversión del display o de la zona en concreto.
<code>inline uint8_t getIntensity() inline uint8_t getIntensity(z)</code>	Obtiene la intensidad actual del display o la zona en concreto.
<code>inline uint16_t getPause() inline uint16_t getPause(z)</code>	Obtiene el tiempo de pausa actual del display o la zona en concreto.
<code>inline uint16_t getScrollSpacing()</code>	Obtiene el espaciado del desplazamiento.
<code>inline uint16_t getSpeed() inline uint16_t getSpeed(z)</code>	Obtiene la velocidad de la animación que se está ejecutando en el display o en la zona en concreto.
<code>inline uint16_t getSpeedIn(z) inline uint16_t getSpeedOut(z)</code>	Obtiene la velocidad de entrada de la animación de la zona. Obtiene la velocidad de salida de la animación de la zona.
<code>inline textPosition_t_getTextAlignment() inline textPosition_t_getTextAlignment(z)</code>	Obtiene la posición de alineamiento del texto del display o de la zona en concreto.
<code>inline uint16_t_getTextColumns(*p) inline uint16_t_getTextColumns(z, *p)</code>	Obtiene el ancho del texto del display o de la zona en columnas.
<code>inline boolean getZoneEffect(z, ze)</code>	Devuelve verdadero si el efecto en el argumento está configurado.
<code>void setCharSpacing(cs) inline void setCharSpacing(z, cs)</code>	Configura el espacio en columnas entre los caracteres para todas las zonas o la zona específica.
<code>inline void setIntensity(intensity) inline void setIntensity(z, intensity)</code>	Configura la intensidad de todo el display o de la zona en concreto.
<code>inline void setInvert(invert) inline void setInvert(z, invert)</code>	Invierte el display o la zona en concreto.
<code>inline void setPause(pause) inline void setPause(z, pause)</code>	Configura la pausa entre la entrada y salida de las animaciones para todas las zonas o la zona en concreto.
<code>inline void setScrollSpacing(space)</code>	Configura la distancia entre mensajes para todas las zonas.
<code>inline void setSpeed(speed) inline void setSpeedInOut(speedIn, speedOut) inline void setSpeed(z, speed)</code>	Configura la velocidad de entrada y de salida de las animaciones para todas las zonas o la zona en concreto.
<code>inline void setSpeedInOut(z, speedIn, speedOut)</code>	
<code>void setData(*inData, inWidth, inFrames, *outData, outWidth, outFrames) void setData(z, *inData, inWidth, inFrames, *outData, outWidth, outFrames)</code>	Configura los efectos de Sprites para todo el display o la zona en concreto. Un Sprite está hecho de un número de frames que corren de forma secuencial para realizar una animación en el display. Una vez realizado se reinicia desde el primer frame.
<code>inline void setTextAlignment(ta) inline void setTextAlignment(z, ta)</code>	Configura la alineación del texto para todo el display o la zona en concreto.
<code>inline void setTextBuffer(*pb) inline void setTextBuffer(z, *pb)</code>	Configura el puntero en el buffer de texto para todo el display o para la zona en concreto.
<code>inline void setTextEffect(effectIn, effectOut) inline void setTextEffect(z, effectIn, effectOut)</code>	Configura el efecto de entrada y de salida del display o de la zona en concreto.
<code>inline void setZoneEffect(z, b, ze)</code>	Configura el efecto del display para la zona específica si b es true y lo resetea si es false.
<code>inline void synchZoneStart()</code>	Sincroniza el inicio de las animaciones en todas las zonas.
<code>inline void addChar(code, *data) inline void addChar(z, code, *data)</code>	Añade un carácter definido por el usuario en todas las zonas o en la zona en concreto.
<code>inline void delChar(code) inline void delChar(z, code)</code>	Borra un carácter definido por el usuario en todo el display o en una zona en concreto.
<code>inline MD_MAX72XX::fontType_t* getFont(z) inline MD_MAX72XX::fontType_t* getFont()</code>	Obtiene la fuente usada en el display o en la zona en concreto.
<code>inline void setFont(*fontDef) inline void setFont(z, *fontDef)</code>	Configura la fuente del display o de una zona en concreto.
<code>inline MD_MAX72XX *getGraphicObject()</code>	Obtiene un puntero hacia el objeto gráfico instanciado.
<code>inline void getDisplayExtent(&startColumn, &endColumn) inline void getDisplayExtent(z, &startColumn, &endColumn)</code>	Obtiene el inicio y final del display o de la zona en concreto.
<code>inline void getTextExtent(&startColumn, &endColumn) inline void getTextExtent(z, &startColumn, &endColumn)</code>	Obtiene el inicio y el final del texto en el display o en la zona en concreto.

Figura 157 Tabla resumen de las funciones de la librería MD_Parola (continuación).

Capítulo 3.2. LIBRERÍAS PARA LA PANTALLA TÁCTIL.

Para el control de la pantalla táctil se usará la librería TFT_eSPI (Versión 2.5.0) [47] que permite implementar todas las funcionalidades de varios modelos de pantalla TFT, entre las que se encuentra la pantalla elegida para este proyecto. Permite controlar empleando un único puerto SPI las diferentes funcionalidades de la pantalla, como es el mostrar gráficos por pantalla, su funcionalidad táctil y acceder a la tarjeta SD que incorpora el propio módulo.

Se trata de una librería de fuentes y gráficos compatible con el IDE de Arduino pensada para procesadores de 32 bits, optimizando su rendimiento para microcontroladores como el ESP32 entre otros. En esta librería existe un fichero de configuración llamado *User_Setup.h* donde se declaran los pines y sus funcionalidades, no se deben incluir dentro del sketch del IDE pues pueden producir errores en las librerías.

Esta librería incluye un Sprite que permite actualizaciones sin parpadeos de la pantalla. Para ello, primero se almacenan en la memoria RAM del controlador los gráficos que se desean proyectar en la pantalla y posteriormente se traza en la pantalla el dibujo requerido, de tal forma que se emplea la RAM como un buffer.

El controlador de la parte táctil de la pantalla, el XPT2046; solo es compatible con SPI a diferencia de la pantalla que se puede controlar mediante otros métodos.

En cuanto a las fuentes que proporciona, se pueden habilitar y deshabilitar diferentes tamaños a la hora de compilar el programa para optimizar el uso de la memoria Flash. La selección de las fuentes se realiza de nuevo en el fichero *User_Setup.h* comentando o descomentando líneas de código.

Esta librería usa en gran parte la librería Adafruit_GFX [48], facilitando su implementación. En cuanto al contenido de se puede organizar en varias secciones dependiendo de su objetivo, entre las que se encuentran: cargar cabeceras requeridas por la librería como Arduino.h, Print.h y SPI.h, cargar librerías y cabeceras específicas de cada procesador (en este caso los drivers para el ILI9488) y los pines que se emplearán para la comunicación con este módulo, configuración de la interfaz, configuración de las fuentes de texto declaradas en el fichero *User_Setup.h*, definición de etiquetas para las diferentes posiciones del texto y colores más comunes, definición de estructuras y funciones para detectar posibles errores y definición de las funciones principales de esta librería cuyo nombre y descripción de algunas de ellas se mostrarán en la tabla de la Figura 158 a modo de estudio previo.

Función	Descripción
<code>drawPixel</code>	Dibuja un píxel de un color determinado en la posición indicada.
<code>drawChar</code>	Dibuja un carácter en la posición y fuente indicada, incluyendo glifos.
<code>drawLine</code>	Dibuja una línea entre las dos posiciones indicadas.
<code>drawFastVLine</code>	Dibuja una línea vertical en la posición indicada.
<code>drawFastHLine</code>	Dibuja una línea horizontal en la posición indicada.
<code>fillRect</code>	Dibuja un rectángulo lleno en la posición indicada y del ancho y alto deseado.
<code>height</code>	Obtiene la altura de la pantalla en píxeles.
<code>width</code>	Obtiene el ancho de la pantalla en píxeles.
<code>readPixel</code>	Lee el color de un píxel en la posición indicada y lo devuelve en formato 565.
<code>setWindow</code>	Define una ventana para recibir un flujo de píxeles. Empleado para la memoria RAM.
<code>pushColor</code>	Introduce (escribe un píxel) colores en la ventana creada con anterioridad.
<code>setRotation</code>	Rota la orientación de la pantalla.
<code>getRotation</code>	Lee la rotación actual de la pantalla.
<code>setOrigin</code>	Establece la posición de origen de los gráficos en el punto deseado.
<code>invertDisplay</code>	Invierte los colores de la pantalla.
<code>setAddWindow</code>	Define un área para recibir un flujo de píxeles.
<code>setViewport</code>	Establece la región de recorte para la pantalla TFT.
<code>checkViewport</code>	Comprueba si alguna parte del área especificada se encuentra visible en la ventana gráfica
<code>getViewPortWidth</code>	Obtiene el ancho de la ventana gráfica.
<code>getViewPortHeight</code>	Obtiene el alto de la ventana gráfica.
<code>frameViewport</code>	Dibuja un marco dentro o fuera de la ventana gráfica.
<code>resetViewport</code>	Reestablece la ventana gráfica a toda la pantalla TFT.
<code>clipAddWindow</code>	Recorta la dirección de la ventana a la pantalla y a la ventana gráfica.
<code>clipWindow</code>	Recorta la ventana a la pantalla y a la ventana gráfica.
<code>pushColor</code>	Introduce un solo píxel o varios en una longitud determinada. Del mismo color.
<code>pushColors</code>	Introduce una serie de píxeles para dibujar imágenes en formato de 16 bits.
<code>pushBlock</code>	Escribe un bloque de un único color sólido.
<code>pushPixels</code>	Escribe un conjunto de píxeles almacenados en la memoria.
<code>fillScreen</code>	Limpia la pantalla al color definido.
<code>drawRect</code>	Dibuja el contorno de un rectángulo.
<code>drawRoundRect</code>	Dibuja el contorno de un rectángulo con esquinas redondeadas.
<code>fillRoundRect</code>	Dibuja un rectángulo lleno con esquinas redondeadas.
<code>fillRectVGradient</code>	Dibuja un rectángulo lleno con un gradiente de color en vertical.
<code>fillRectHGradient</code>	Dibuja un rectángulo lleno con un gradiente de color en horizontal.
<code>drawCircle</code>	Dibuja el contorno de un círculo.
<code>fillCircle</code>	Dibuja un círculo lleno.
<code>drawEllipse</code>	Dibuja el contorno de una elipse.
<code>fillEllipse</code>	Dibuja una elipse llena.
<code>drawTriangle</code>	Dibuja el contorno de un triángulo.
<code>fillTriangle</code>	Dibuja un triángulo lleno.
<code>drawSmoothArc</code>	Dibuja un arco suave en el sentido de las agujas del reloj desde las 6 en punto.
<code>drawArc</code>	Dibuja un arco en el sentido de las agujas del reloj desde las 6 en punto.
<code>DrawSmoothCircle</code>	Dibuja el contorno de un círculo suave.
<code>fillSmoothCircle</code>	Dibuja un círculo suave lleno.
<code>drawSmoothRoundRect</code>	Dibuja el contorno de un rectángulo suave con esquinas redondeadas.
<code>drawSpot</code>	Dibuja un círculo lleno suavizado del radio indicado.
<code>drawWideLine</code>	dibuja una línea suavizada con extremos redondeados.
<code>drawWedgeLine</code>	dibujar una línea suavizada con extremos redondeados de diferente ancho
<code>drawBitmap</code>	Dibuja una imagen almacenada en una matriz en la pantalla.
<code>setBitmapColour</code>	Establece el color de primer plano y del fondo.
<code>pushRect</code>	Introduce píxeles de colores en un área definida.
<code>pushImage</code>	Traza un objeto o imagen en color en formato 16 bits en la pantalla.
<code>pushMaskedImage</code>	Renderiza una imagen en color de 16 bits con una máscara.
<code>drawNumber</code>	Dibuja un entero grande.
<code>drawFloat</code>	Dibuja un número decimal
<code>drawString</code>	Dibuja una cadena de caracteres.
<code>drawCentreString</code>	Dibuja una cadena de caracteres centrada en una posición.
<code>drawRightString</code>	Dibuja una cadena de caracteres alineada a la derecha.
<code>setCursor</code>	Establece la posición del puntero de texto.
<code>getCursor</code>	Obtiene la posición del puntero de texto.
<code>setTextColor</code>	Establece el color de la fuente.
<code>setTextSize</code>	Establece el tamaño de la fuente.
<code>setTextWrap</code>	Establece si el texto debe ajustarse al final de la línea.
<code>setTextDatnum</code>	Establece la referencia de posición del texto.
<code>setTextPadding</code>	Establece el ancho del relleno.
<code>getTextPadding</code>	Obtiene el ancho del relleno.
<code>setFreeFont</code>	Establece la fuente GFX
<code>setTextFont</code>	Establece la fuente del texto.
<code>textWidth</code>	Obtiene el ancho en píxeles de un string en la fuente elegida.
<code>fontHeifht</code>	Obtiene la altura de la fuente.

Figura 158 Tabla resumen de algunas funciones de la librería TFT_eSPI.

Dado que la placa de desarrollo cuenta con un adaptador para incorporar una tarjeta SD, se puede realizar la programación de este periférico en base a imágenes predefinidas que se alojarán en ella. Para esto será necesario incluir la librería SD (Versión 2.0.0) y una librería llamada JPEGDecoder (Versión 1.8.1) de Bodmer [49] para poder mandar la información a la pantalla.

Para la prueba inicial se carga el programa de ejemplo *TFT_graphistest_one_lib* que mostrará por pantalla varios gráficos y textos en distintos colores para asegurar que la pantalla funciona correctamente. Los resultados obtenidos se muestran en la Figura 159, Figura 160 y Figura 161.

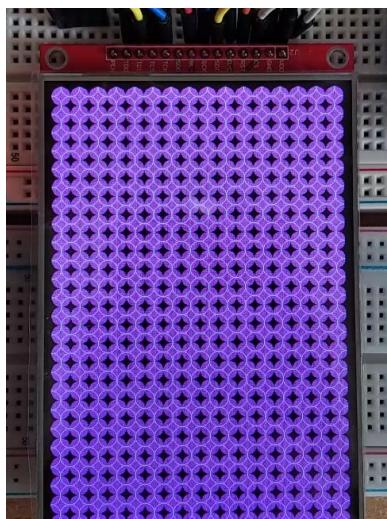


Figura 159 Prueba gráfica en la pantalla TFT. Patrón circulos.



Figura 160 Prueba gráfica en la pantalla TFT. Triángulo.



Figura 161 Prueba de texto en la pantalla TFT.

Para probar la función táctil se usa el programa de ejemplo *Touch_calibrate* que mediante cuatro flechas en las esquinas permite calibrar la pantalla y posteriormente dibujar sobre ella, como se ve en la Figura 162, Figura 163 y Figura 164.



Figura 162 Prueba 1 función táctil de la pantalla TFT.



Figura 163 Prueba 2 función táctil de la pantalla TFT.



Figura 164 Prueba 3 función táctil de la pantalla TFT.

Para facilitar la creación de la interfaz de usuario mediante la pantalla se usará la librería *GUIslice.h* de ImpulseAdventure [50]. Para la programación además se usará la herramienta *GUIslicebuilder* que mediante un programa permite generar los menús, botones y otros elementos gráficos simplemente arrastrándolos a la zona de la pantalla. Para ello hay que configurar el programa para que emplee la librería *TFT_eSPI.h* y el tamaño de la pantalla, como se observa en la Figura 165.

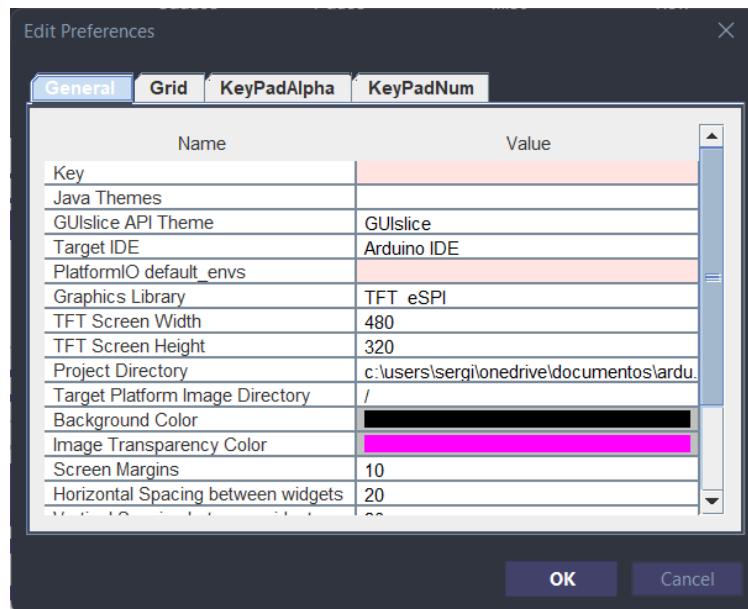


Figura 165 Configuración del programa *GUIslicebuilder* para la pantalla ILI9488.

El propio programa permite generar un código en formato .ino que permitirá mostrar los elementos en la pantalla, pero sin crear ningún tipo de acción. Eso se tendrá que realizar modificando cada elemento de forma individual para realizar las acciones de control correspondientes sobre el panel luminoso.

Capítulo 3.3. LIBRERÍAS PARA LA COMUNICACIÓN INALÁMBRICA (BLUETOOTH/BLE/WIFI/SIMILARES).

Para este proyecto se decide usar el protocolo bluetooth para permitir la recepción de comandos desde un terminal inalámbrico como puede ser un smartphone. Se usará este protocolo ya que la comunicación WiFi con el dispositivo crearía una red local a la que se tendría que conectar el usuario, impidiendo al smartphone conectarse a la red de Internet mientras el dispositivo se encuentra encendido. Existen otros protocolos como el Bluetooth Low Energy (BLE) que permiten realizar acciones similares a las del bluetooth clásico, pero como inconveniente se encuentra una mayor complejidad en la programación con respecto a su otra versión.

Para programar esta característica del dispositivo se emplea la librería *BluetoothSerial.h*, que permite crear una comunicación de tipo serial con el microcontrolador, pudiendo dar un nombre y contraseña al dispositivo para que sea detectado por otros dispositivos bluetooth y puedan conectarse. De esta forma se puede establecer una comunicación bidireccional entre el smartphone y el panel de emergencia. Esto permite establecer un protocolo de interpretación de comandos para mandar de forma inalámbrica información desde el smartphone hasta el microcontrolador y que éste interprete y realice las acciones necesarias y viceversa.

Capítulo 3.4. IMPLEMENTACIÓN DEL PROGRAMA EN LENGUAJE C++.

Capítulo 3.4.1. Software empleado para la programación final del prototipo.

El proceso de programación del prototipo se llevará a cabo usando el editor de código *Visual Studio Code* y la extensión *PlatformIO* donde se descargarán las librerías de espressif (versión 6.4.0) para el microcontrolador. Se emplea este software de programación ya que permite navegar entre los distintos ficheros de forma más cómoda y permite localizar mejor los posibles errores en el código en comparación con el IDE de Arduino.

Para ello se configura el proyecto en el fichero *platformio.ini* con la plataforma, modelo del microcontrolador empleado y el marco en el que se trabajará, tal y como se muestra en la Figura 166.

```

platformio.ini
1 ; PlatformIO Project Configuration File
2 ;
3 ; Build options: build flags, source filter
4 ; Upload options: custom upload port, speed and extra flags
5 ; Library options: dependencies, extra library storages
6 ; Advanced options: extra scripting
7 ;
8 ; Please visit documentation for the other options and examples
9 ; https://docs.platformio.org/page/projectconf.html
10
11 [env:esp32dev]
12 platform = espressif32
13 board = esp32dev
14 framework = arduino

```

Figura 166 Fichero de configuración Visual Studio Code.

Capítulo 3.4.2. Metodología empleada.

Como se ha observado en los apartados anteriores, la metodología empleada parte de la programación de los distintos dispositivos por separado que permite comprobar que las librerías y los distintos elementos funcionan correctamente. El siguiente paso será generar un programa de test donde se vayan añadiendo una a una las librerías y funcionalidades de cada elemento que permitirá comprobar si existen interferencias entre las distintas comunicaciones y las conexiones realizadas. Se decide no implementar por el momento el control inalámbrico y dejar su control para cuando el software del proyecto esté más avanzado ya que no influirá en gran medida en el hardware y las librerías de los dispositivos.

Para la realización de estas pruebas se crea el circuito eléctrico del Capítulo 2.9 en una protoboard para realizar las conexiones de forma rápida y sencilla y que permitan cambios de ser necesario.

Este programa de prueba parte de la librería MD_Parola en el que inicialmente se programará para controlar los mensajes que se muestran empleando para ello distintos botones que se dibujarán en la pantalla TFT con una única interfaz SPI donde se diferencian los dispositivos por el pin CS (Chip Select). El código creado permite mostrar texto con dos fuentes distintas, una que permite visualizar en los dos niveles una sola palabra o frase y otra que maneja cada fila de forma independiente como se puede ver en la Figura 167 y Figura 168.



Figura 167 Fuente en panel LED con doble altura



Figura 168 Fuente en panel LED con simple altura

Tras programarlo se observa que no hay respuesta de pantalla TFT tras un corto periodo de tiempo y que existen interferencias en los mensajes mostrados ya que algunos módulos MAX7219 no responden, formando huecos en el panel. Para solventar esta problemática se decide separa las interfaces SPI de los dos dispositivos, formando una interfaz SPI para el control del panel LED y otra para el resto de los dispositivos. Además,

se divide el programa en dos tareas distintas que permitan controlar los periféricos en un núcleo del microcontrolador y el panel LED en el otro, empleando para ello lenguaje concurrente en base a tareas o hilos.

Tras los cambios anteriores se sigue observando que algunos módulos MAX7219 no responden cuando se solicita un brillo mayor en ellos, por lo que se probará a alimentar el prototipo con una fuente de mayor amperaje y a cambiar la frecuencia del bus SPI, así como revisar las conexiones entre los distintos módulos.

A continuación, se introduce la funcionalidad de la tarjeta SD, mostrando en la pantalla los logos de la universidad de Valladolid y la Escuela de Ingenierías Industriales.

Tras comprobar que todos los elementos del hardware funcionan en conjunto se empezará a programar el primer prototipo. Se empieza creando un menú que se mostrará en la pantalla TFT que mediante su función táctil permitirá realizar las acciones de control o ajuste necesarias.

Esta tarea se llevará a cabo usando el programa *GUIslicebuilder* y las librerías *GUIslicer* [50] que permiten crear una interfaz de usuario (Figura 169) de forma cómoda, sencilla y bastante completa empleando para ello algunas funciones que engloban a la librería *TFT_eSPI* usada para controlar la pantalla.



Figura 169 Ejemplos de interfaz de usuario usando *GUIslice*.

El programa *GUIslicebuilder* se encarga de generar el esqueleto principal de interfaz de usuario, obteniendo de esta forma un programa en Arduino (.ino) de características muy simples que será modificado posteriormente para añadir complejidad al proyecto. De nuevo habrá que modificar el fichero de configuración *GUIslice_config.h*, parametrizando la librería para el caso de la pantalla basada en el controlador ILI9488.

Se plantea una interfaz de usuario donde en la pantalla principal se muestre el estado actual del panel (encendido, apagado, modo de funcionamiento, etc...) seguido de diversos botones que permitirán al usuario moverse entre las diferentes opciones, como pueden ser los ajustes o la selección del mensaje que se quiera mostrar por el panel LED (Figura 170).

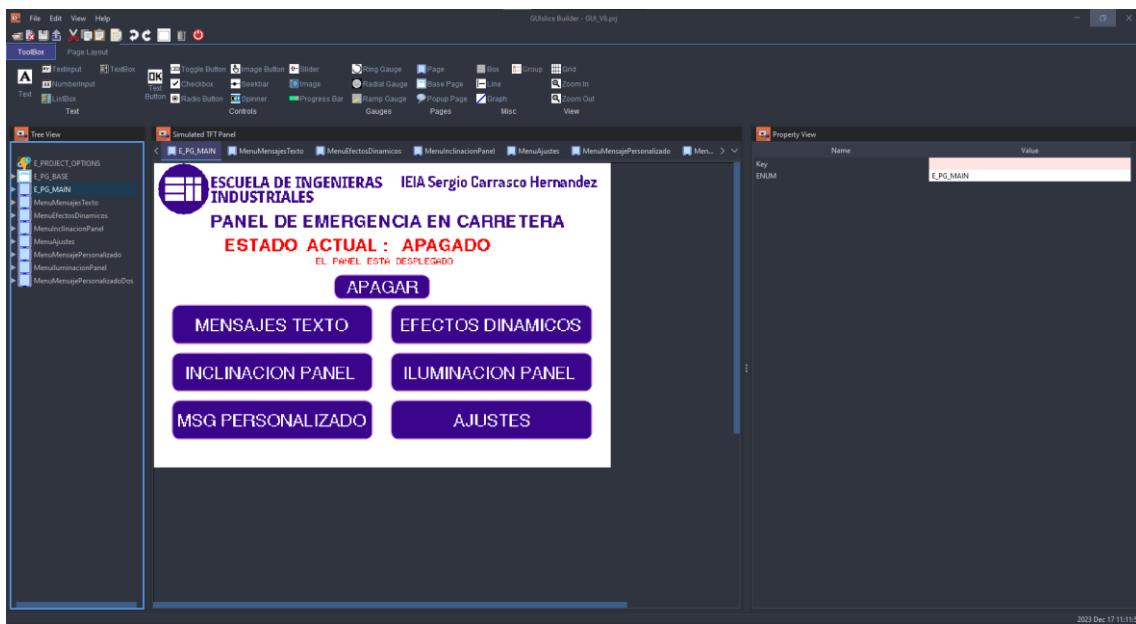


Figura 170 Pantalla principal.

Una vez diseñadas todas las pantallas se modificará el código generado para añadir las capacidades de control sobre el panel LED y poder hacer algo con esta interfaz. Para ello se ordenará el código generado en un nuevo fichero y se introducirá en un proyecto de Visual Studio Code para que el programa sea más ordenado, legible y mantenable.

Cuando se ha introducido el esqueleto de la pantalla táctil se creará una librería específica que englobará los distintos mensajes del panel mediante una máquina de estados en uno de los núcleos del ESP32. Los estados se modificarán desde el otro núcleo empleando para ello la interfaz de usuario creada con anterioridad.

Tras comprobar que los mensajes se muestran de forma correcta, se añaden el resto de las funcionalidades como inclinar el panel mediante una barra deslizante o unos botones, la lectura del nivel de luz externa, ajustes de brillo de la pantalla y la calibración de la función táctil entre otras. Cada una de estas acciones se separa en la tarea del núcleo correspondiente según sea su finalidad para mantener el código limpio y ordenado.

Capítulo 3.4.3. Programación de la pantalla.

Para la pantalla se ha creado un fichero de cabecera llamado *PantallaTFT.h* donde se recogen las principales funciones que hacen referencia a la pantalla como son su inicialización, calibración, guardado de datos en la memoria SPIFFS del microcontrolador, funciones para dibujar imágenes almacenadas en la memoria SPIFFS, funciones para manejar las acciones correspondientes de los botones, teclados para las entradas de texto, barras deslizantes y listas de opciones entre otras. Además, se ha creado una carpeta *PantallaTFT* donde se encuentran recogidas las fuentes de texto empleadas, un fichero de configuración inicial de la pantalla específico para el proyecto donde se recogen los colores y formas de cada uno de los elementos implementados en



la interfaz de usuario generada en la pantalla. Estos últimos se han generado con la herramienta GUIslicebuilder y han sido modificadas para añadir las funcionalidades necesarias de los botones, como el manejo de menús y realizar las acciones necesarias sobre el panel LED y la propia pantalla.

Para ejecutar el control de la pantalla TFT se ha creado un segundo hilo aprovechando los dos núcleos del ESP32 de tal forma que se pueda controlar de forma independiente al propio panel luminoso. En este segundo núcleo se ejecuta de forma continua una tarea que actualiza los distintos estados de la pantalla que según las acciones del usuario se comunicará con el hilo de programación del luminoso cambiando su estado.

Dado que este tipo de programación usa un lenguaje concurrente, ha sido necesario implementar algunos semáforos que impiden a ambos núcleos acceder simultáneamente a las mismas direcciones de memoria para leer o escribir las variables evitando así errores en la lectura y escritura de estas.

En cuanto a la implementación de la tarjeta SD, finalmente se ha decidido no hacer uso de esta característica ya que la visualización de los ficheros almacenados introducía un retraso en la generación de los distintos menús de la interfaz de usuario debido a la lectura de los datos desde la tarjeta SD y la posterior escritura en la pantalla. Por ello se ha decidido almacenar los pocos archivos en la memoria SPIFFS y leerlas directamente desde la memoria flash del microcontrolador. Este paso se ha realizado creando una carpeta llamada data en el directorio del proyecto de Visual Studio Code y empleando la extensión de platform.io subir estos archivos a la memoria no volátil del microcontrolador. Es importante que si se emplean imágenes como en este caso, las imágenes estén en formato BMP.

En cuanto a la interfaz de usuario realizada se puede dividir en varias pantallas o menús:

Pantalla principal, Figura 171: Se muestran siete botones donde seis de ellos llevarán a un submenú con más opciones y otro de ellos permitirá apagar de forma directa el panel luminoso. Consta de dos textos, uno de ellos cambiará de color y el mensaje según el panel luminoso esté encendido o no y el otro aparecerá o no en función de si el panel está inclinado o completamente recogido, avisando así al usuario de que tiene el panel desplegado sin necesidad de mirar al propio panel.



Figura 171 Pantalla del menú principal de la pantalla TFT.

Pantalla del menú de mensajes de texto, Figura 172: En esta pantalla se incorpora una caja con una lista de mensajes que junto con una barra deslizante permitirán al usuario elegir el texto que se desea mostrar en el panel numinoso. Consta de un botón que permite volver al menú principal. Los textos por introducir pueden estar en tamaño doble empleando ambas filas del panel luminoso o en una sola fila, pudiendo mostrar una frase o advertencia de mayor tamaño. Algunos de estos textos cuentan con efectos especiales como simulación de lluvia o nieve y algunos constarán con símbolos creados específicamente mediante fuentes como flechas, símbolos y señalizaciones.



Figura 172 Pantalla del submenu de mensajes de texto predeterminados de la pantalla TFT.

Pantalla del menú de efectos dinámicos, Figura 173: Al igual que en el submenu anterior, en esta pantalla se muestra una caja con una serie de efectos dinámicos que permitirán elegir señalizaciones más llamativas. Un botón específico permitirá volver al menú principal.



Figura 173 Pantalla del submenú de efectos dinámicos de la pantalla TFT.

Pantalla del menú de inclinación, Figura 174: En ella se encuentran cuatro botones y una barra deslizante que permitirán seleccionar la inclinación del panel. Además, se muestra de forma gráfica la inclinación actual del panel empleando para ello una barra de progreso en forma de anillo, simulando la posición real del panel. Un botón en la parte inferior permite volver al menú principal.



Figura 174 Pantalla del submenú de inclinación del panel LED.

Pantalla de iluminación del panel, Figura 175: En ella se mostrará el nivel de iluminación recibido en el panel luminoso y un interruptor que permitirá cambiar el brillo de este en función de forma automática según el valor leído. Este interruptor se desactivará de forma automática si el usuario acciona la barra de iluminación manual, permitiendo ajustar el brillo según se deseé. Un botón en la parte inferior permite volver al menú principal.



Figura 175 Pantalla del submenú de nivel de brillo del panel LED.

Pantalla de mensaje personalizado con una sola fila, Figura 176: En esta pantalla se ha incorporado un recuadro que permite introducir texto empleando un teclado que saldrá en la propia pantalla. El texto introducido será mostrado en una fuente de doble altura y mediante una lista de efectos y barra deslizante se permitirá modificar la velocidad a la que aparece y la forma en la que aparece. Un botón permite llegar a otra pantalla de mensaje personalizado. Al igual que en las pantallas anteriores, un botón permite regresar a la pantalla principal.

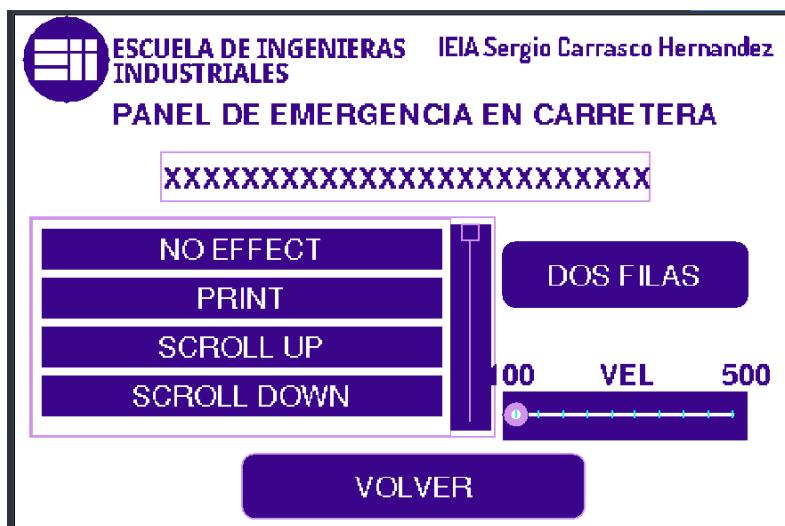


Figura 176 Pantalla del submenú de mensajes predeterminadas con una sola fila.

Pantalla de mensaje personalizado en dos filas, Figura 177: En este submenú se puede introducir un texto para la fila inferior y otro para la fila superior, permitiendo seleccionar efectos y velocidades separadas para cada fila. Un botón permite volver a la pantalla de mensaje personalizado en una fila.



Figura 177 Pantalla del submenu de mensajes personalizados en dos filas distintas.

Pantalla de ajustes, Figura 178: En esta pantalla se permite ajustar el brillo de la propia pantalla y realizar una calibración automática de la pantalla táctil. Se podrían introducir funcionalidades de la comunicación inalámbrica.



Figura 178 Pantalla del submenu de ajustes de la pantalla TFT.

Algunas de las modificaciones que se realicen empleando para ello la pantalla serán guardadas en la memoria no volátil (SPIFFS) empleando para ello la librería Preferences.h para que la próxima vez que se ejecute el dispositivo se mantengan estos valores y el usuario no tenga que volver a incorporarlos. Algunos de estos valores son la iluminación de la pantalla TFT, los datos obtenidos de la calibración manual y la inclinación y el brillo del panel luminoso.

Como marca de pertenencia a la Escuela de Ingenierías Industriales y a la Universidad de Valladolid en el encendido de la pantalla se mostrará el logo de la universidad y en cada una de las distintas pantallas se mostrará el logo de la escuela, el nombre del prototipo y el autor del proyecto.

Capítulo 3.4.4. Programación del panel.

Para la programación del panel se emplea principalmente la librería MD_Parola.h que como se ha mencionado anteriormente engloba a la librería MAX72XX.h. En este caso se ha decidido crear una clase objeto llamada PanelLED que permite englobar, recoger y organizar de mejor forma las distintas funciones que permiten el control del dispositivo de veinte matrices. Esta clase se organiza mediante un fichero de cabecera llamado PanelLED.h donde se recogen las distintas variables y funciones de la clase y un fichero PanelLED.cpp donde se han desarrollado las distintas funciones.

La clase creada está pensada para funcionar mediante una máquina de estados que permite modificar el estado de funcionamiento empleando para ello algunas funciones llamadas setters. Cambiando este parámetro se podrá implementar el mensaje deseado empleando la fuente, efecto, velocidad, brillo, mensaje e inclinación del panel luminoso.

Como se ha mencionado anteriormente, es posible implementar mensajes usando solo una única fila de doble altura para hacer el mensaje más grande y también en dos filas con caracteres que ocupen solo una matriz y así escribir mensajes de mayor longitud de forma simultánea en el panel. Los mensajes predeterminados que se pueden elegir según la pantalla de mensajes de texto permiten implementar ambos tipos de mensajes dependiendo del que se elija y además permiten incorporar símbolos específicos según la ocasión empleando para ello fuentes personalizadas descritas en el fichero FontData.h que se almacena en una carpeta de la librería MD_Parola.h. Algunos de estos caracteres especiales son distintos efectos dinámicos que simulan lluvia, nieve, niebla o símbolos especiales como señales de prohibido el paso, triángulos de advertencia, flechas, etc.

Se han incorporado también efectos especiales como puede ser un barrido del panel completo o encendidos simultáneos dos a dos de las matrices haciendo referencia a las funciones con las que ya contaba el primer modelo realizado en las prácticas en empresa descrito en el Capítulo 1.1. En cuanto a la ejecución del hilo que controla al panel, se realiza en el otro núcleo disponible, por lo que se realiza con una programación concurrente empleando hilos y semáforos para las variables como ya se ha mencionado. En este caso también se emplea una sola función que permite actualizar los distintos estados del panel según se modifiquen los parámetros en la pantalla táctil, modificando en todo momento unas variables que definen el menú en el que se encuentran y el estado del mensaje que se quiere mostrar.

Hay algunas funciones específicas de esta clase que empleando el servomotor permiten modificar la inclinación del panel.

Capítulo 3.4.5. Programación de la comunicación bluetooth e interpretación de los comandos.

Para la comunicación mediante bluetooth se ha creado un nuevo fichero llamado "BluetoothManager.h" que contiene las funciones y declaraciones necesarias para su



fin. Una de estas funciones es el setup de la comunicación, donde se indica el nombre del dispositivo como *Panel Emergencia* y se inicia la comunicación bluetooth. La lectura del canal bluetooth se realiza en una función llamada *RunBluetooth()* que, como si se tratara de una comunicación serial típica de Arduino o del ESP32 monitorizará el canal y en el caso de que haya datos disponibles se almacenarán los caracteres recibidos en una variable tipo String y se procesará.

El procesamiento de los comandos recibidos se realiza mediante un pseudo protocolo creado para este proyecto que consiste en detectar un carácter específico, en este caso la almohadilla (#) y separar unas cadenas de caracteres o Strings de otras y poder realizar diferentes acciones. De esta forma se establecen los comandos de la tabla de la Figura 179 para la comunicación con el panel de emergencia y que el microcontrolador sea capaz de ejecutar acciones en función del mensaje recibido.

PRIMER COMANDO RECIBIDO EN EL ESP32	SEGUNDO COMANDO RECIBIDO EN EL ESP32	ACCIÓN
TEXTO PREDeterminado#	(número entero indicador del texto)#	En función del número entero recibido por este protocolo se mostrará por pantalla uno de los textos predeterminados como avisos por niebla, lluvia, accidente, etc.
EFFECTO PREDeterminado#	(número entero indicador del efecto)#	En función del número entero recibido por este protocolo se mostrará por pantalla uno de los efectos predeterminados como parpadeos dos a dos, desvíos hacia la izquierda, parpadeo total del panel, etc.
TEXTO UNA FILA#	(string que se desea escribir en el panel)#	Escribirá en una fila de doble altura el string recibido por el puerto de comunicación.
VELOCIDAD UNA FILA#	(número entero)#	Establece la velocidad a la que se ejecutan los efectos cuando se está mostrando un mensaje personalizado en una fila de doble altura.
EFFECTO UNA FILA#	(número entero)#	Establece el efecto con el que se mostrará el texto personalizado de doble altura que se haya establecido anteriormente.
TEXTO FILA SUPERIOR#	(string que se desea escribir en el panel)#	Escribirá en la fila superior del panel el string recibido por el puerto de comunicación.
VELOCIDAD FILA SUPERIOR#	(número entero)#	Establece la velocidad a la que se ejecutan los efectos cuando se está mostrando un mensaje personalizado en una fila superior del panel.
EFFECTO FILA SUPERIOR#	(número entero)#	Establece el efecto con el que se mostrará el texto personalizado de la fila superior que se haya establecido anteriormente.
TEXTO FILA INFERIOR#	(string que se desea escribir en el panel)#	Escribirá en la fila inferior del panel el string recibido por el puerto de comunicación.
VELOCIDAD FILA INFERIOR#	(número entero)#	Establece la velocidad a la que se ejecutan los efectos cuando se está mostrando un mensaje personalizado en una fila inferior del panel.
EFFECTO FILA INFERIOR#	(número entero)#	Establece el efecto con el que se mostrará el texto personalizado de la fila inferior que se haya establecido anteriormente.
INCLINACION#	(número entero de 0 a 180)#	Establece la inclinación del panel según el número entero recibido.
BRILLO AUTO#	FALSO# ó VERDADERO#	En función del string recibido se iluminará el panel de forma automática según la lectura del sensor o de forma manual según haya establecido el usuario.
BRILLO#	(número entero del 1 al 10)#	Establecerá el brillo del panel de forma manual.
APAGAR#	#	Apagará el panel.
SOLICITA DATOS INCLINACION#	#	Devuelve la inclinación actual del panel.
SOLICITA DATOS BRILLO#	#	Devuelve el brillo manual actual en el panel, el nivel de luz leído por el sensor y si el modo de iluminación es automático o manual.
MENSAJE UNA FILA#	#	Devuelve el string del texto personalizado almacenado correspondiente al texto de doble altura.

MENSAJE DOS FILAS#	#	Devuelve los strings de los textos personalizados de las zonas superior e inferior.
--------------------	---	---

Figura 179 Tabla resumen del pseudoprotocolo de comunicación bluetooth del panel.

La distinción entre los comandos se realiza en la función *ProcesarComandoBT()* al que se le pasa el comando recibido como argumento y se encarga de actualizar las variables de las clases correspondientes mediante sus getters y setters.

Capítulo 3.4.6. Detalles de otras librerías.

Adicional a las librerías anteriores, se ha creado un fichero de cabecera llamado Memoria.h que contiene funciones para guardar datos en la memoria flash del microcontrolador, de tal forma que una vez se apaga el dispositivo las opciones de configuración e información importante como la posición actual del panel se puede volver a cargar. De esta forma se logra una mayor calidad en el producto final.

Otro de los ficheros importantes es el de Pinout.h donde se describen todos los pines empleados en la programación, lo que hace muy sencillo el cambiar si fuese necesario el puerto GPIO del microprocesador al que se conecta el periférico en cuestión.

Capítulo 3.5. IMPLEMENTACIÓN DEL PROGRAMA EN LENGUAJE KOTLIN EN ANDROID STUDIO.

Capítulo 3.5.1. Software empleado para la programación de la aplicación.

Para el desarrollo de la aplicación se usará Android Studio [51], un entorno de desarrollo integrado (IDE) pensado específicamente para el desarrollo de aplicaciones Android.

En cuanto al lenguaje de programación, este IDE ofrece dos posibilidades: Java y Kotlin. Con respecto a Java, Kotlin es un lenguaje más conciso, seguro y moderno además de que cuenta con funciones adicionales que Java no concibe. Por estas razones se decide programar la app en lenguaje Kotlin.

Capítulo 3.5.2. Metodología empleada en el desarrollo de la App.

En cuanto a la metodología del programa, al tratarse de un lenguaje y un entorno de trabajo totalmente desconocido antes de realizar este proyecto, se ha empezado aprendiendo sobre este nuevo entorno y lenguaje de programación siguiendo algunos cursos descriptivos sobre la programación [52] y creación de las distintas pantallas y botones principales [53].

Una vez familiarizado con el entorno y lenguaje se comienza creando la base de la aplicación, para el cual se definen las distintas pantallas, botones e indicadores que serán necesarios para este proyecto. El diseño que se ha seguido es muy similar al implementado en la pantalla TFT del dispositivo permitiendo así que el usuario no tenga que acostumbrarse a dos sistemas totalmente distintos.

Una vez se ha creado la disposición o *layout* de los elementos se añadirán las funciones necesarias como la navegación entre las pantallas y algunos efectos como cambios de textos y variables según se pulsen los botones, barras deslizantes o listas de elementos.

Estos elementos se han creado con la librería Jetpack Compose [54], una librería de interfaz de usuario que permite simplificar y acelerar la construcción de interfaces de usuario programándolas de una forma más intuitiva y sencilla.

Una vez se ha obtenido la base y funcionalidades en el manejo del *layout* se crea la conexión bluetooth con el panel de emergencia, permitiendo así mandar mensajes desde el teléfono móvil hasta el ESP32. Para lograr esto se ha partido de algún programa de ejemplo [55] que ha sido necesario adaptar debido a las diferencias entre versiones y a las diferencias entre el tipo de programación, ya que muchos de estos ejemplos se programan sin usar la librería de Jetpack Compose.

Capítulo 3.5.3. Disposición de los menús y funciones de la App.

La app creada ha sido configurada con un icono que permita identificarla claramente entre el resto de las aplicaciones, como se ve en la Figura 180. Al abrir la aplicación se observará una pantalla principal (Figura 181) donde se muestra el estado de funcionamiento actual del panel entre ENCENDIDO o APAGADO, cambiando el color entre rojo y verde; el estado de inclinación del panel entre RECOGIDO Y DESPLEGADO, también cambiando el texto de color entre rojo y verde según la situación. A estos dos indicadores le sigue un botón que permitirá mandar un comando al ESP32 mediante bluetooth para apagar el panel y posteriormente otros seis botones que permitirán navegar al resto de pantallas. Cada vez que se pulsa uno de los botones que lleva a una nueva pantalla la aplicación solicitará los datos que debe mostrar en la pantalla al microcontrolador del panel de emergencia para obtener en todo momento los valores actualizados.

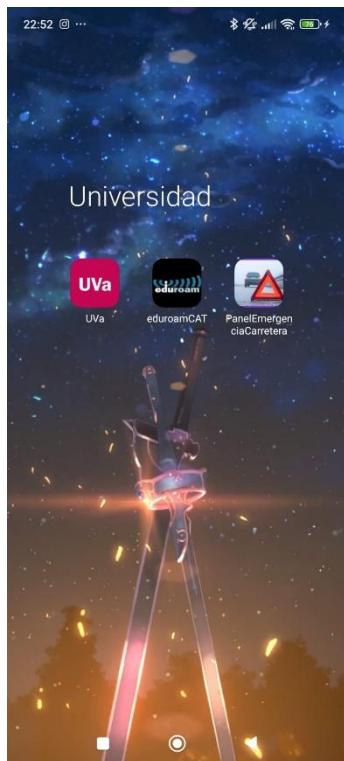


Figura 180 Icono de la aplicación Android.



Figura 181 Pantalla principal de la aplicación Android.



Figura 182 Pantalla de mensajes predeterminados de texto en la aplicación Android.

En la Figura 182 se muestra la pantalla que se obtiene al pulsar el botón MENSAJE DE TEXTO de la pantalla principal. En ella se muestran una lista de opciones de mensajes que se pueden poner en el propio panel. Cuando se pinche en alguno de estos botones el dispositivo mandará el comando en cuestión al botón (ver tabla de la Figura 158) por el puerto serie bluetooth, permitiendo al ESP32 mostrar en el panel LED el mensaje seleccionado.

De igual forma se muestra en la Figura 183 una lista de efectos dinámicos que al pulsar algún botón mandará un comando mediante bluetooth al ESP32.



Figura 183 Pantalla de efectos dinámicos predeterminados en la App de Android.



Figura 184 Pantalla de inclinación del panel luminoso en la App de Android.



Figura 185 Pantalla de iluminación del panel luminoso en la App de Android.

En la Figura 184 se muestran los controles relacionados con la inclinación del panel luminoso. Al entrar en esta pantalla se cargarán los valores actuales almacenados en el ESP32 sobre la inclinación, es decir, se tendrá el valor actualizado de la inclinación en el dispositivo móvil. Al accionar alguno de los botones o la barra deslizante se mandará el comando bluetooth adecuado para que el panel se incline. El valor mostrado ahora en la pantalla se actualizará sin necesidad de solicitarlo al ESP32 de forma inalámbrica por la lógica interna del programa en Kotlin.

En la Figura 185 se observa la pantalla correspondiente a la iluminación del panel luminoso, donde se mostrará el nivel de luz leído por el sensor del propio panel, si se está usando la iluminación automática o no y qué nivel de iluminación manual está almacenado en el microcontrolador. Al igual que en el caso anterior, los valores se actualizan al entrar en la pantalla solicitando estos datos al ESP32 mediante bluetooth. Si alguno de los controles se modifica, es decir, se cambia de modo manual a automático o se modifica el valor de luz mediante la barra deslizante se mandará el comando adecuado para modificar dicho valor en el panel luminoso.

En la Figura 186 se encuentra la pantalla de mensaje personalizado para mostrar en el panel un mensaje de doble altura con la velocidad y efecto deseado. La disposición permite elegir el efecto deseado empleando para ello una lista de opciones deslizante. Al igual que los casos anteriores al entrar en la página aparecerán los valores almacenados en el ESP32, manteniendo la aplicación continuamente actualizada. El botón de la parte inferior lleva a la pantalla de la Figura 187, que realiza la misma función, pero permitiendo establecer un mensaje para cada una de las filas de una sola altura del panel de emergencia.

Por último, la pantalla de la Figura 188 muestra los ajustes de bluetooth. Si queremos hacer funcionar la aplicación en conjunto con el panel de emergencia en carretera será necesario vincular el dispositivo, para lo cual primero será necesario accionar el botón de activar bluetooth para activar el bluetooth en caso de tenerlo desactivado y dar los permisos necesarios para que la aplicación escanee los dispositivos bluetooth asociados al teléfono en cuestión. Una vez aceptados los permisos se debe accionar el botón de buscar dispositivos, apareciendo debajo los dispositivos a los que está asociado. Estos dispositivos se muestran sobre botones en una lista deslizable donde accionado el indicado se conectará el dispositivo, apareciendo un texto en la parte inferior indicando que la conexión se ha realizado con éxito.

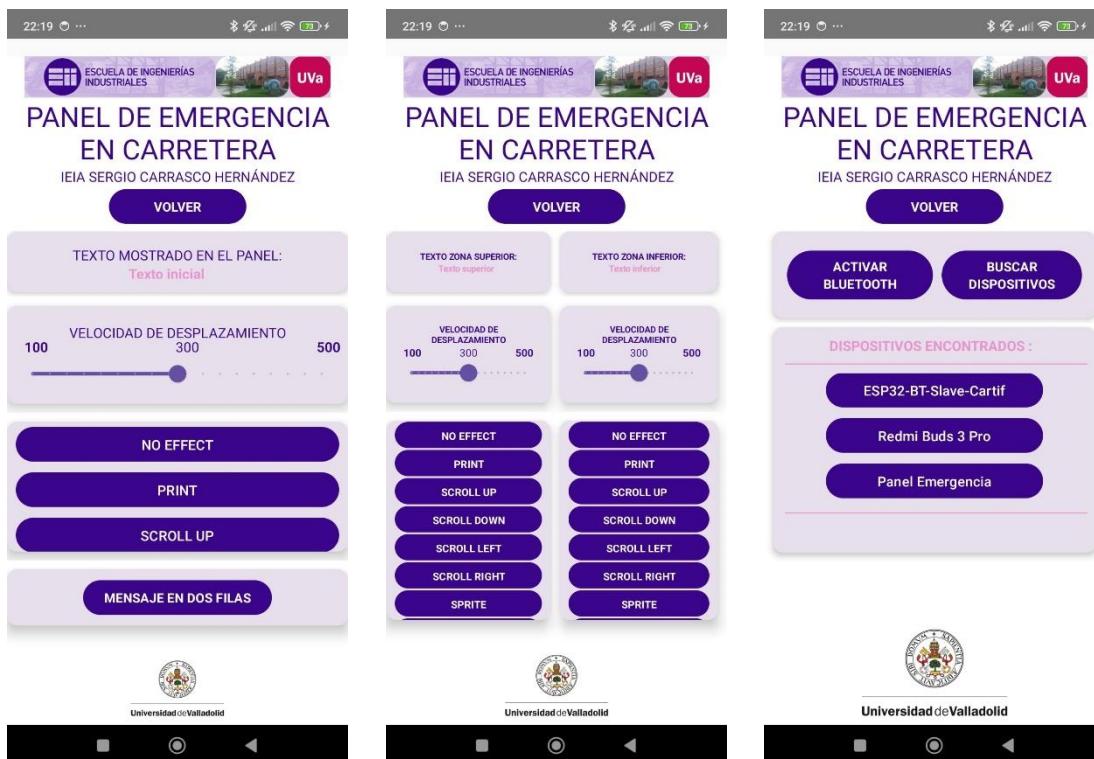


Figura 186 Pantalla de mensaje personalizado de una fila en la App de Android.

Figura 187 Pantalla de mensaje personalizado de dos filas en la App de Android.

Figura 188 Pantalla de ajustes bluetooth en la App de Android.

Capítulo 3.5.4. Estructura del programa de la App de Android.

Para la programación de la aplicación Android se ha estructurado el programa como se muestra en la Figura 189. En ella se observa un primer fichero llamado *AndroidManifest.xml* donde se deben declarar todos los permisos necesarios para usar la comunicación bluetooth del dispositivo. En la parte inferior se encuentra un fichero llamado *build.gradle.kts(Module:app)* que es donde se deben incluir las librerías que se emplearán para este proyecto de Android, entre las que se encuentra la librería *Jetpack Compose*, encargada de crear los distintos elementos como botones, barras deslizantes, imágenes etc; y la librería *Navigation Compose* [56], que permitirá navegar entre las distintas pantallas de la aplicación.

La estructura principal de la aplicación se ha organizado en distintos ficheros de programación en Kotlin, cuya extensión es *.kt*. El fichero análogo al *main.cpp* en programación de C++ es el *MainActivity.kt*, donde se definen y crean las rutinas principales de la app. En la función *onCreate* correspondiente a la del *main* en C++ se llamará a la función encargada de navegar y mostrar por las distintas pantallas que se organizarán en distintos ficheros *.kt* para lograr un programa ordenado según el *Destinos.kt*.

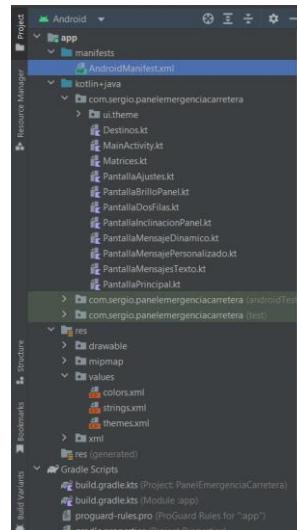


Figura 189 Estructura del programa Android.

En el propio *MainActivity.kt* se creará un hilo de trabajo para manejar la recepción de comandos bluetooth en el dispositivo Android, almacenando los datos recibidos en las variables que permitirán mostrar los valores actualizados en todo momento.

La comunicación desde la app hasta el ESP32 se realizará mandando los comandos según el protocolo descrito anteriormente cuando algún botón o barra deslizante sea accionado desde alguna de las pantallas, para lo que se usa la función *sendCommand()* implementada en el *MainActivity.kt*.

En cuanto a la prueba del dispositivo, se puede conectar el dispositivo móvil en el que se desea descargar la aplicación mediante el puerto USB y activando la depuración USB e Instalar mediante USB en los ajustes de Android se puede instalar y lanzar la app en el dispositivo. De esta forma se ha ido comprobado y navegando entre las distintas pantallas y comprobando la comunicación bluetooth con el ESP32.

Capítulo 4. FABRICACIÓN DE LA PCB.

Para probar el prototipo ha sido necesario fabricar las placas de circuito impreso basadas en el esquema de conexiones del Capítulo 2.9. Para ello se ha usado el software *KiCad 7.0* con la herramienta de edición de placas. En concreto ha sido necesario crear dos placas, una para alojar tanto el microcontrolador como la pantalla TFT y el circuito de alimentación y otra que permite conectar todos los elementos del panel luminoso mediante una serie de conectores. El diseño de estas placas se puede observar en la Figura 190 y Figura 191.

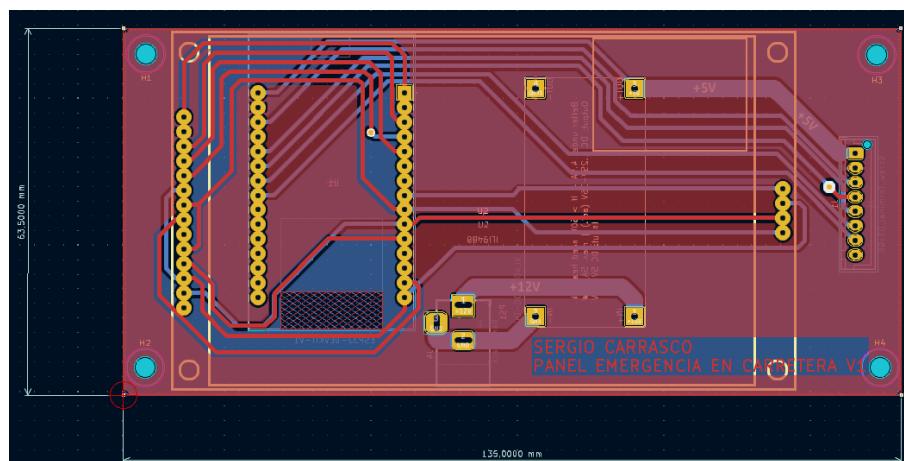


Figura 190 Diseño de la PCB principal del prototipo.

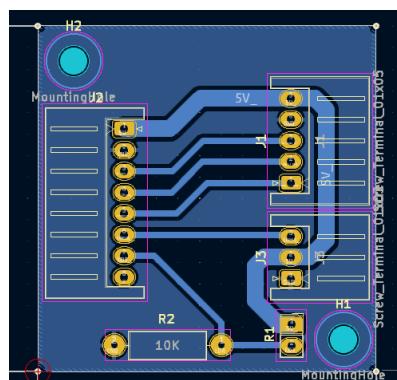


Figura 191 Diseño de la PCB auxiliar para las conexiones del panel LED.

Tras realizar el diseño, se fabricarán las placas mediante el procedimiento establecido en el laboratorio de Métodos y Herramientas de Diseño Electrónico, pasando por el insolado de la resina fotosensible de las PCBs vírgenes, atacado de la resina mediante una disolución de sosa caustica y agua, atacado del cobre expuesto mediante una disolución de agua y ácido clorhídrico y peróxido de hidrógeno. Una vez fabricadas, se sueldan los distintos componentes, resultando en las PCBs de la Figura 192.

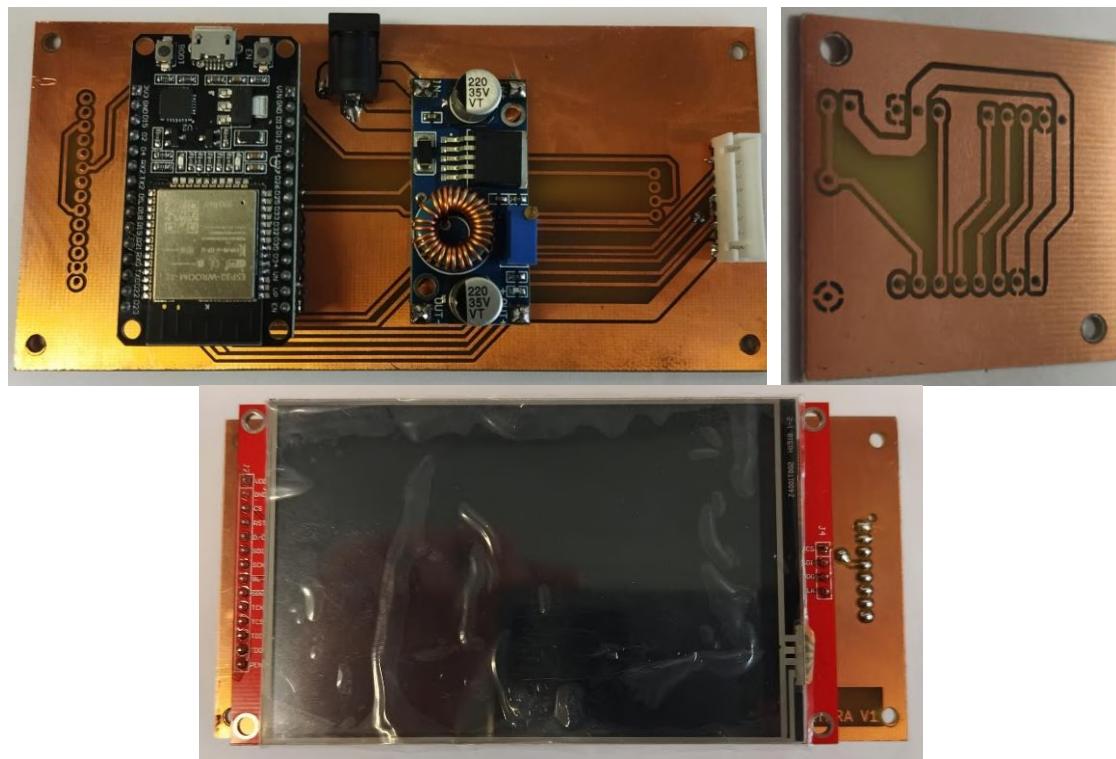


Figura 192 PCBs del prototipo con los componentes soldados.

Capítulo 5. ENVOLVENTE DEL PROTOTIPO.

Capítulo 5.1. OBJETIVOS DE LA ENVOLVENTE.

El objetivo principal de la envolvente es el de proporcionar una estructura sólida al prototipo para poder manejarlo con comodidad y seguridad. Además, esta envolvente proporcionará protección a cada uno de los elementos evitando en cierta medida daños debido a caídas o golpes. Se buscará crear una envolvente práctica a la vez que estética.

Para diseñar la envolvente se empleará el software Autodesk Inventor que permitirá hacer una carcasa o caja a medida en función de las necesidades del prototipo. Una vez hecho el diseño se imprimirá en una impresora 3D empleando la técnica FDM (Modelado por Deposición Fundida) que consiste en ir construyendo el objeto mediante capas de material fundente (en este caso PLA). Debido a la forma de impresión, el prototipo no contará con un grado de IP elevado pues entre las diferentes capas es posible que se filtren líquidos o incluso polvo, por lo que en caso de querer sacar el producto al mercado será necesario pensar en otra forma de fabricar la envolvente, como mediante técnicas de inyección o de moldeo.

Capítulo 5.2. ENVOLVENTE PANEL LED.

Para la envolvente del panel LED se debe tener en cuenta su tamaño. En este caso, debido a sus grandes dimensiones; se ha optado por diseñar dos partes que se unirán

mediante tornillos e insertos en la parte trasera para poder imprimirla en una impresora 3D convencional cuyo tamaño máximo es de 24x24cm. El diseño realizado en Inventor se observa en la Figura 193, donde se distinguen dos piezas de color gris oscuro que serán el marco donde irá alojado el panel y una pieza de color gris claro que permitirá la unión de las otras dos. En la Figura 194 se observa el diseño de una placa transparente de metacrilato transparente que se ha mecanizado empleando una CNC laser para que se ajuste a las medidas del prototipo. De esta forma se obtiene una mayor protección en el panel y se pueden visualizar de forma clara las indicaciones que se muestran a través de él. En la parte superior de la envolvente se ha realizado un orificio para poder exponer a la luz la LDR encargada de medir la luz exterior, tal y como se ha explicado con anterioridad. Además, se creará un orificio de métrica 12 en un lateral para poder atornillar un prensaestopas que mantenga fija la manquera de cables que procede del panel de control.

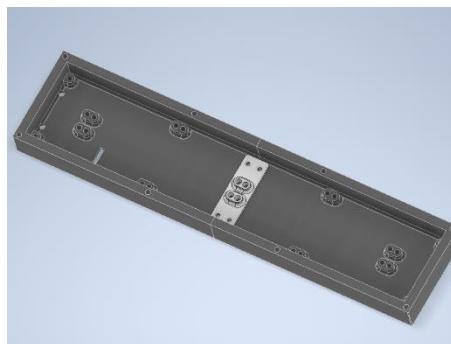


Figura 193 Diseño de la envolvente del panel LED.

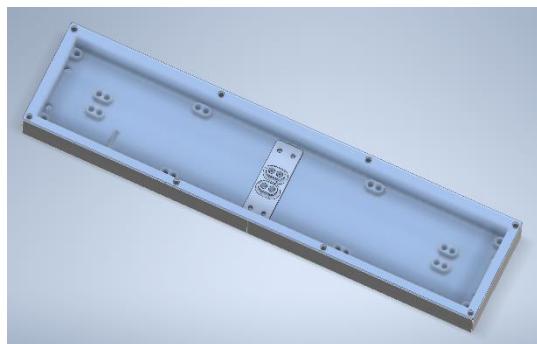


Figura 194 Diseño de la placa transparente del panel LED.

En la Figura 195, Figura 196 y Figura 197 se pueden observar las diferentes piezas de esta parte de la envolvente, así como el resultado final una vez se ha ensamblado todo.



Figura 195 Distintas piezas impresas para el panel.



Figura 196 Envolvente panel LED.



Figura 197 Envolvente del panel LED completo.

Capítulo 5.3. ENVOLVENTE DEL PANEL DE CONTROL.

Para la envolvente del panel de control se creará una caja que permita el acceso a la pantalla táctil y a los puertos de alimentación y de programación del dispositivo. Además, al igual que la envolvente del panel LED; tendrá un orificio de métrica 12 para poder acoplar un prensaestopas que mantenga fija la manquera que conecta ambos dispositivos. Dado que se trata de una pantalla táctil también se añadirá un soporte para un lápiz para dicha pantalla y un soporte para poder colocar ésta en algún lugar del salpicadero del vehículo. Este soporte permitirá rotar la pantalla y soltarla del soporte para poder manejar el dispositivo. La caja finalmente se cerrará con una tapa que terminará de proteger al resto de elementos.

El diseño final realizado en Inventor se observa en la Figura 198 y Figura 199. Una vez diseñado se imprimen las distintas piezas y se ensambla el conjunto junto con la PCB principal del dispositivo, donde se encuentra el controlador y la pantalla. El resultado final se puede observar en la Figura 200 y Figura 201.

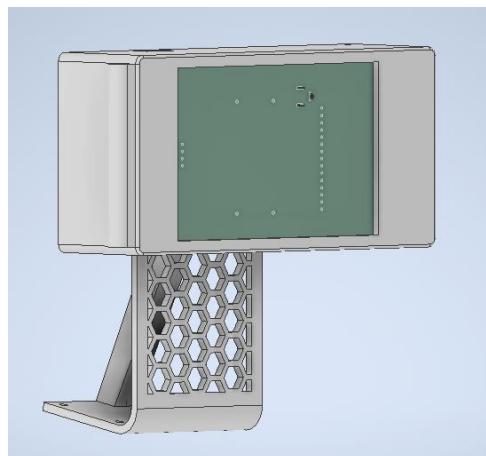


Figura 198 Vista frontal y lateral del diseño del panel de control con el soporte.

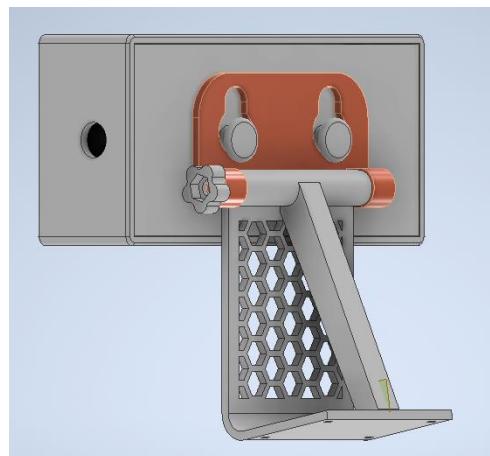


Figura 199 Vista trasera y lateral del diseño del panel de control con el soporte.

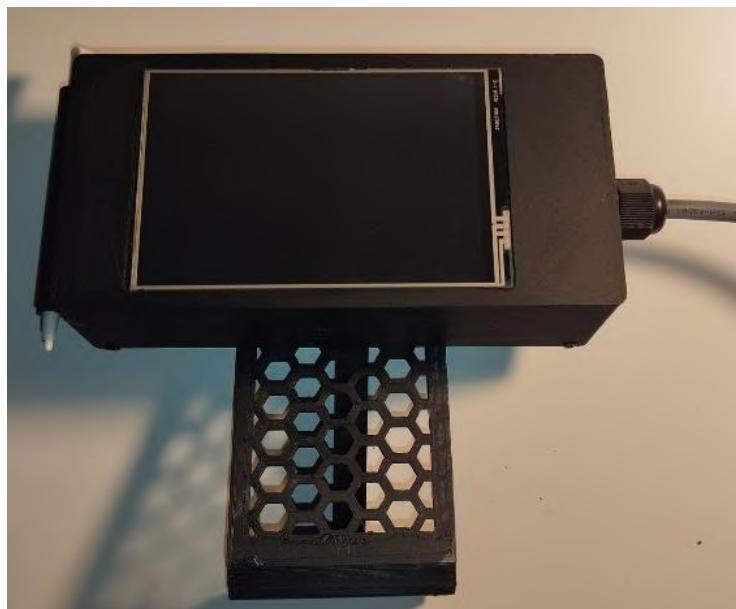


Figura 200 Vista frontal del panel de control sin montar en el soporte.



Figura 201 Vista trasera del panel de control sin la tapa.

Capítulo 5.4. ENVOLVENTE DE ELEMENTOS AUXILIARES.

Dado que algunas de las características de los modelos estudiados la capacidad de ocultar el panel luminoso se ha propuesto poder modificar el ángulo de éste para evitar añadir resistencia aerodinámica si se sitúa el panel sobre el techo de un vehículo. También este diseño permitirá que el prototipo ocupe un menor espacio y sea más fácil de almacenar.

Se propone que la inclinación se pueda ajustar de forma automática empleando para ello alguna función disponible en la interfaz de usuario o de forma manual.

Para la forma automática se empleará un servomotor con un mecanismo de piñón cremallera que engrane con un engranaje solidario al panel LED y que permita la rotación respecto a un eje solidario a la base sobre la que se apoya el panel. El diseño creado se observa en la Figura 202.

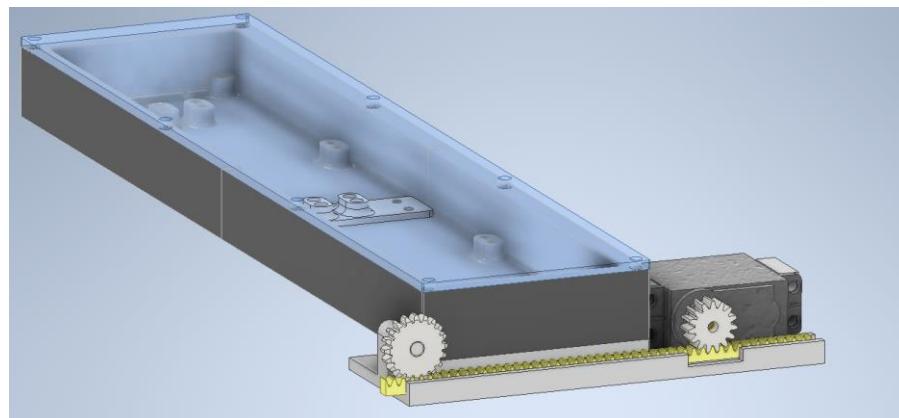


Figura 202 Mecanismo piñón cremallera del panel luminoso.

Otra de las formas propuestas para inclinar el panel es emplear un soporte de ángulo ajustable mediante una barra que se ancla en la base, permitiendo el giro libre sobre un eje fijo también a la base. Este mecanismo se puede observar en la Figura 203.

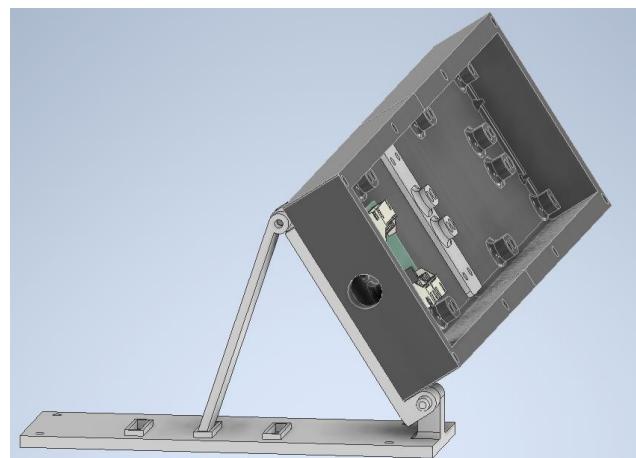


Figura 203 Soporte de ángulo ajustable.

Capítulo 5.5. MONTAJE DEL PROTOTIPO.

Una vez se tienen todos los elementos se anclan en un banco de trabajo para poder probar que todo funciona correctamente y tener una maqueta que sirva como expositor del trabajo realizado. El prototipo final se muestra en la



Figura 204 Montaje final del panel de emergencia en carretera - Vista frontal.

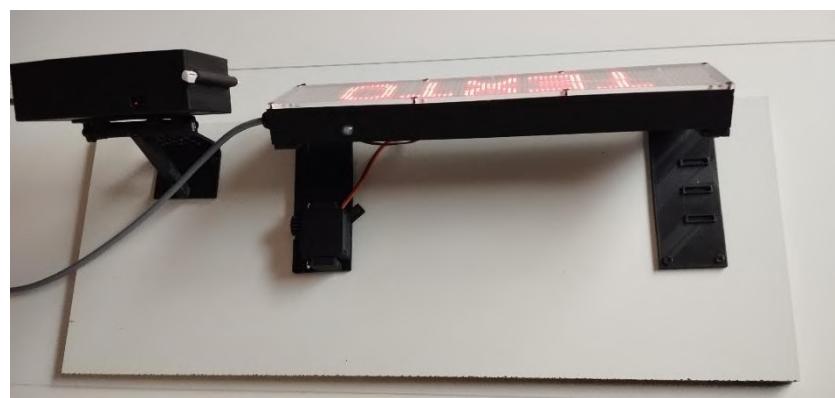


Figura 205 Montaje final del panel de emergencia en carretera - Vista trasera.



Figura 206 Montaje final del panel de emergencia en carretera - Vista lateral.

Capítulo 6. PRUEBAS Y VALIDACIÓN DEL PROTOTIPO.

Capítulo 6.1. PLAN DE PRUEBAS.

A lo largo del desarrollo de este proyecto se han ido probando todas y cada una de las funcionalidades conforme se iban incorporando al programa, empezando por controlar de forma simple el panel formado por matrices LED, siguiendo por la pantalla táctil y por último con la aplicación de Android y la comunicación bluetooth. Pese a esto se debe comprobar la integridad del dispositivo una vez se ha programado completamente para garantizar que los diferentes elementos y partes del programa no interfieren entre sí. Para ello se probará de nuevo el dispositivo, empezando por probar cada uno de los botones correspondientes a la interfaz de usuario creada en la pantalla TFT y observando que lleva a cada subpantalla y que cada botón y barra deslizante implementa las acciones de control correspondientes sobre el panel LED, incluyendo la inclinación del propio panel mediante el servomotor y la obtención del nivel de luz por parte del sensor.

Una vez comprobado el funcionamiento de la pantalla TFT se comprobará la comunicación y control del panel LED empleando para ello la aplicación desarrollada, comprobando que la información almacenada en el ESP32 llega hasta el teléfono móvil y que los botones de la aplicación implementan las acciones correctas sobre el panel LED.

Una vez se ha comprobado esta funcionalidad se vuelve a comprobar que los datos enviados desde el móvil se han almacenado y se muestran de nuevo en la pantalla TFT.

Capítulo 6.2. RESULTADOS Y ANÁLISIS DE LAS PRUEBAS REALIZADAS.

Durante la comprobación de resultados se ha obtenido que la transmisión de datos mediante bluetooth hacia y desde el ESP32 es satisfactoria y es muy complicado romper el protocolo de comunicaciones gracias a la autodetección de comandos erróneos implementado en el propio código, limpiando el buffer de llegada cuando esto sucede.

En cuanto al sensor de nivel de luz en el ambiente también se obtienen resultados satisfactorios pudiendo variar la luminosidad del panel de forma automática según reciba más o menos luz.

En cuanto al control de inclinación del panel existe la posibilidad de que el mecanismo de piñón cremallera acabe saltándose algún diente debido a la poca precisión de la impresora 3D y el tipo de montaje, pero sin ser realmente preocupante.

La iluminación del panel LED presenta algunos problemas más serios en algunas ocasiones, llegando a apagarse algunos de los módulos superiores sin ser intencionado. Este fenómeno se ve incrementado con el aumento del brillo de las matrices LED, por lo que puede que se deba al ruido electrónico introducido por el aumento de corriente por los módulos. En concreto el error encontrado sucede en las matrices 12 a 19 que, pese a que no permiten el encendido de esas matrices LED sí que permiten el paso de la

comunicación de datos, por lo que también puede deberse a defectos en las pistas o montaje de dicho módulo en concreto, aunque se ha revisado y reforzado el conexionado eléctrico de los mismos. Un ejemplo de este suceso se muestra en la Figura 207.



Figura 207 Error producido al intentar mostrar la palabra TEXTO PRUEBA.

Capítulo 6.3. VALIDACIÓN DEL PROTOTIPO.

Pese a que existen errores en el propio prototipo se puede validar el funcionamiento de este, permitiendo así pasar a la siguiente fase que sería generar el esquema y placas del panel LED al completo, evitando así realizar uniones entre distintos módulos que pueden ocasionar errores en la transmisión de datos por el bus SPI como se sospecha que sucede en el prototipo. En cuanto a los errores en la inclinación del panel, no se les da mucha importancia pues se deben a las tolerancias y la degradación de las piezas impresas en 3D con el uso debido a que la dureza del material empleado no es muy elevada.

Por estas razones y debido a que el resto de las pruebas han resultado satisfactorias se puede validar el prototipo para continuar con el desarrollo técnico del mismo.

Capítulo 7. COSTE DEL EQUIPO.

En este capítulo se describirá el coste del equipo tanto por los componentes físicos como el desarrollo del firmware correspondiente a la programación del microcontrolador y de la aplicación Android desarrollada. Este capítulo podrá marcar la diferencia entre la viabilidad económica del prototipo para continuar su posible desarrollo y posterior industrialización.

Capítulo 7.1. DESGLOSE DE COSTOS.

En la tabla de la Figura 208 se muestra el desglose de costes relacionado con los propios componentes que lleva el prototipo, especificando unidades, coste unitario y coste total del prototipo físico.

Tipo de coste	Resumen	Cantidad	Precio	Total
---------------	---------	----------	--------	-------

Componente electrónico	ESP32 dev module	1 u.	11.22€	11.22€
	Pantalla táctil SPI 4" de 480x320 TFT	1 u.	40.66€	40.66€
	Fuente de alimentación regulable, 5A	1 u.	2.44€	2.44€
	Fotorresistencia	1 u.	0.36€	0.36€
	Matriz LED 8x8 MAX7219	4 uds.	2.83€	11.32€
	Matriz LED 8x8 MAX7219 (4 matrices)	4 uds.	7.58€	30.32€
	Placa de circuito impreso doble cara grosor 35µm, 220 x 100 x 1.6mm	1 u.	13.23€	13.23€
	Conectores JST	2 uds.	0.36€	0.72€
	Servomotor	1 u.	13.99€	13.99€
Impresión 3D	Envolvente panel	123g	20€/kg	2.47€
	Soporte del panel	74g	20€/kg	1.48€
	Envolvente pantalla TFT	103g	20€/kg	2.06€
	Soporte pantalla TFT	32g	20€/kg	0.64€
Tornillería	Tornillos M3x10mm	10 uds.	0.1€	1€
	Tornillos M3x5mm	30 uds.	0.1€	3€
	Tornillos M5x20mm	4 uds.	0.3€	1.2€
	Insertos M3	40 uds.	0.1€	4€
	Insertos M5	2 uds.	0.3€	0.6€
Base	Tablero aglomerado blanco 30x60x1cm	1 u.	3.99€	3.99€
Otros	Metacrilato	1 u.	2€	2€
TOTAL				152.19€

Figura 208 Resumen de costes físicos del panel de emergencia en carretera.

En la tabla de la Figura 209 se muestra una estimación del número de horas dedicadas a la programación del dispositivo como de desarrollo de la documentación y ficheros necesarios para la fabricación y construcción del dispositivo. El precio establecido para el cálculo del coste por hora está basado en el sueldo medio en España para un ingeniero electrónico, de 30.000€ anuales.

Resumen	Cantidad	Precio	Total
Programación C++	150h	15.38€/h	2307€
Programación Android	50h	15.38€/h	769€
Desarrollo de documentación y fabricación de las PCBs	10h	15.38€/h	153.8€
Diseño 3D de los ficheros STL para impresión en 3D	30h	15.38€/h	461.4€
Investigación y definición de componentes	30h	15.38€/h	461.4
TOTAL			4152.6€

Figura 209 Resumen de costes del firmware del panel de emergencia en carretera.

Capítulo 7.2. EVALUACIÓN DE LA VIABILIDAD ECONÓMICA.

Si bien el coste de desarrollo del prototipo para el panel de emergencia en carretera es de 4304.79€, la mayor parte de éste se genera en la programación y desarrollo de documentación y ficheros, suponiendo un 96.5% del coste total. Por tanto, dado que el firmware y la documentación necesaria estarían ya generados y no sería necesaria una gran modificación de cara a la fabricación en masa de ser necesario se puede concluir que este producto podría resultar viable económicamente para salir a mercado una vez pasada la etapa de industrialización y mejora del producto.

Capítulo 8. EVALUACIÓN GENERAL Y PLANES DE FUTURO.

Capítulo 8.1. CONCLUSIONES.

En este documento se ha descrito el proceso seguido para la construcción de un panel de emergencia en carretera para situarlo en la parte superior de un vehículo. El prototipo consiste en un panel LED formado por una matriz de 16x80 puntos que permiten mostrar mensajes de texto y diferentes efectos, lo que supone una mejora con el proyecto desarrollado en las prácticas en empresa descrito en el Capítulo 1.1. Además, se han incorporado dos interfaces de usuario, una mediante una pantalla táctil que sirve como panel de control junto con el microcontrolador y otra que emplea un smartphone y comunicación inalámbrica mediante bluetooth. Por otra parte, se han incluido funciones adicionales como el autoajuste del brillo y el ajuste de inclinación del panel y se ha llevado a cabo la fabricación íntegra del prototipo. En esta fabricación se han diseñado e impreso en 3D las distintas envolventes del producto y se han fabricado las placas de circuito impreso correspondientes que permiten dotar al prototipo de suficiente robustez para la aplicación que se le desea dar.

El coste unitario del producto en esta fase del proyecto ronda los 152.2€ en cuanto a componentes físicos, por lo que abaratando costes y solventando algunos problemas que puedan surgir en su etapa de industrialización podría tratarse de un producto bastante rentable comparándolo con el estudio de mercado realizado en capítulos anteriores.

Como conclusión, se ha creado un producto que ha cumplido con todos los objetivos establecidos, logrando una mejora sustancial del panel del que se partía en las prácticas en empresa, tratándose de un panel de emergencia en carretera de mucha flexibilidad y de fácil uso, aunque de gran complejidad de desarrollo.

La dificultad de este proyecto ha recaído en gran parte en la programación de la pantalla táctil al tratarse de un elemento con el que nunca se había trabajado y del cual no existe mucha documentación pues es un modelo relativamente raro debido a sus grandes dimensiones, teniendo que llegar incluso a adaptar librerías para su uso. Ha existido también una elevada dificultad en la parte de la programación de la aplicación Android creada para el control del panel de emergencia ya que ha requerido aprender un nuevo

lenguaje de programación además de aprender como implementar la comunicación inalámbrica con el dispositivo mediante bluetooth. En cuanto a la programación de las matrices que forman el panel LED, la dificultad ha residido en las primeras etapas de desarrollo hasta que se han encontrado las librerías adecuadas para el uso que se le pretendía dar.

Capítulo 8.2. LÍNEAS A FUTURO.

Si bien se han cumplido todos los requisitos descritos en el Capítulo 1.2 existen algunas limitaciones de este producto lo que permite abrir nuevas líneas a futuro que permitirán lograr alcanzar una mayor calidad y complejidad en el producto final entre las que se encuentran:

- Dado que se trata de un panel formado por 20 matrices de 8x8 LEDs con un tamaño de 32x32 mm cada una, se logra un panel LED de 16x80 puntos con unas dimensiones totales de 6,4x32cm por lo que no se trata de un panel de grandes dimensiones que se pueda observar a largas distancias. Esto podría solventarse añadiendo más matrices de 8x8 aumentando el número total de puntos que se tendrían en el panel, por lo que habría que realizar algún pequeño cambio en la programación declarando de nuevo el número de elementos que contiene y el número de filas del panel. Otra de las opciones podría ser emplear el mismo número de matrices, pero de dimensiones superiores, creando la matriz propia usando LEDs de 5mm dispuestos en una matriz y controlados por el mismo chip como muestra la Figura 210, donde se observa una matriz de 64 LEDs equiespaciados 1.5cm, logrando una matriz de 12x12cm, lo que podría llegar a dar lugar a un panel LED de 24x120cm si se usaran 20 módulos como los del prototipo. Si bien se usaría el mismo chip para su control (MAX7219) sería necesario realizar un estudio sobre el consumo de este tipo de montaje.



Figura 210 Matriz de 8x8 LEDs de 5mm espaciados 1,5cm.

- Dado que el panel construido está formado por LEDs de color rojo, no es posible mostrar indicaciones o mensajes en varios colores, por lo que se podría optar

por usar el LED RGB con microcontrolador incorporado WS2812B. Este elemento, dado que se encuentra en tiras LED de grandes dimensiones, podría cumplir con el aumento del tamaño del propio panel que se comenta en el punto anterior de forma más sencilla además de lograr mostrar diversos colores. Existen por internet ejemplos de pantallas construidas con este elemento como la de la Figura 211, donde se emplean 8000 de estos dispositivos para mostrar vídeos o imágenes [57]. Esta característica cambiaría toda la programación realizada en este proyecto, aumentando significativamente la complejidad.



Figura 211 Pantalla formada por WS2812B.

- Otra de las líneas futuras podría ser la creación una PCB que no use las placas de desarrollo del microcontrolador, de la pantalla TFT y del convertido reductor de la alimentación, sino que se suelden directamente los componentes realmente necesarios sobre una placa de cobre. Esto reduciría significativamente los costes en la producción en masa y además dotaría al sistema de un acabado mucho más profesional y la envolvente de la pantalla y el controlador resultaría mucho más fino, por lo que el producto ocuparía mucho menos. Un ejemplo de lo que se propone podría ser el mostrado en la Figura 212 y Figura 213 , donde en una sola placa se encuentran los componentes discretos para el funcionamiento de esa pantalla.



Figura 212 ESP32-S3 SPI TFT con sensor táctil – Vista frontal.



Figura 213 ESP32-S3 SPI TFT con sensor táctil – Vista trasera.

- Otro camino que seguir sería la implementación de comunicaciones usando Bluetooth Low Energy (BLE), ya que en términos de consumo será mejor que usando el protocolo clásico, aunque se trate de un protocolo algo más complejo.



Capítulo 9. BIBLIOGRAFÍA.

- [1] PROIN S.L, «PROIN REMOLQUES Y EQUIPAMIENTOS LUMINOSOS. PANEL MENSAJE VARIABLE TEXTO,» 5 Junio 2023. [En línea]. Available: <https://proinbal.es/project/panel-mensaje-variable-texto/>.
- [2] PROIN S.L, «PROIN - REMOLQUES Y EQUIPAMIENTOS LUMINOSOS. PANEL VARIABLE GRÁFICO.,» 5 Junio 2023. [En línea]. Available: <https://proinbal.es/project/panel-mensaje-variable-grafico/>.
- [3] SEBA S.L, «SEBA Sistema Combinado Luminoso. Señalización,» [En línea]. Available: <http://sebasl.com/sistema-combinado-luminoso-23-focos/>. [Último acceso: 5 Junio 2023].
- [4] SEBA S.L, «SEBA POLVIS TXT,» [En línea]. Available: <http://sebasl.com/polvis-txt/>. [Último acceso: 5 Junio 2023].
- [5] TECNIVIAL, «TECNIVIAL Panel Mensaje Variable LED,» [En línea]. Available: <https://www.tecnivial.es/product/p-m-v-para-vehiculo/>. [Último acceso: 5 Junio 2023].
- [6] TECNIVIAL, «TECNIVIAL Panel de mensaje variable LED Hidráulico,» [En línea]. Available: <https://www.tecnivial.es/product/panel-de-mensaje-variable-led/>. [Último acceso: 5 Junio 2023].
- [7] TECNIVIAL S.A., «Panel de Mensaje Variable Portatil Tecnivial,» [En línea]. Available: https://www.youtube.com/watch?v=tO75uOuaziY&ab_channel=Tecnivial%2CS.A.. [Último acceso: 5 Junio 2023].
- [8] VEVOR, «VEVOR LEtrero De Desplazamiento Led Mensaje 100x20 Cm 2 Modos Para Publicidad,» [En línea]. Available: https://www.vevor.es/led-desplazamiento-de-mensaje-c_10919/letrero-de-desplazamiento-led-mensaje-100-x-20-cm-2-modos-para-publicidad-p_010227274051?adp=gmc&utm_source=google&utm_medium=cpc&utm_id=16396408319&utm_term=&gclid=Cj0KCQjwj_ajBhCqARIsAA37s0yqb8R. [Último acceso: 5 Junio 2023].
- [9] WORLDSEMI CO., «WS2812B Datasheet andSpecifications,» [En línea]. Available: https://voltiq.ru/datasheets/WS2812B_datasheet_EN.pdf. [Último acceso: 6 Junio 2023].
- [10] M. K. Daniel Garcia, «FastLED Animation Library,» 2013. [En línea]. Available: <http://fastled.io/>. [Último acceso: 6 Junio 2023].
- [11] P. Burgess, Adafruit , [En línea]. Available: <https://chat.openai.com/c/8e559ab4-3221-4d1c-963f-15dc948c932f>. [Último acceso: 6 Junio 2023].

- [12] LUMEX INC, «LDM-6432-P3-BLE4-1.PDF,» [En línea]. Available: <https://www.farnell.com/datasheets/2693593.pdf>. [Último acceso: 7 Junio 2023].
- [13] Farnell Components SL, «Farnell LDM-6432-P3-BLE4-1,» [En línea]. Available: <https://es.farnell.com/lumex/ldm-6432-p3-ble4-1/led-dot-matrix-display-64-x-32/dp/2946522>. [Último acceso: Junio 7 2023].
- [14] LUMEX INC, «LUMEX LDM-12864-P3-UR,» [En línea]. Available: <https://www.lumex.com/datasheet/LDM-12864-P3-UR>. [Último acceso: 7 Junio 2023].
- [15] Alibaba, «Alibaba Color rgb smd p2.5 Pantalla de panel de led módulo de 320*160mm,» [En línea]. Available: https://spanish.alibaba.com/p-detail/Rgb-60634647423.html?spm=a2700.pccps_detail.0.0.14d813a0AbGD6J&s=p. [Último acceso: 7 Junio 2023].
- [16] Adafruit Industries, «Adafruit LED Backpacks. Overview,» 14 Junio 2012. [En línea]. Available: <https://learn.adafruit.com/adafruit-led-backpack>. [Último acceso: 17 Junio 2023].
- [17] Holtek Semiconductor Inc., «HT16K33A,» [En línea]. Available: <https://www.holtek.com/productdetail/-/vg/HT16K33A>. [Último acceso: 17 Junio 2023].
- [18] Adafruit Industries, «Adafruit Mini 8x8 LED Matrix w/I2C Backpack - Red,» [En línea]. Available: <https://www.adafruit.com/product/870>. [Último acceso: 17 Junio 2023].
- [19] Adafruit Industries, «Adafruit Small 1.2" 8x8 LED Matrix w/I2C Backpack,» [En línea]. Available: <https://www.adafruit.com/product/1049>. [Último acceso: 18 Junio 2023].
- [20] Adafruit Industries, «16x8 1.2" LED Matrix + Backpack - Ultra Bright Square Amber LEDs,» [En línea]. Available: <https://www.adafruit.com/product/2041>. [Último acceso: 18 Junio 2023].
- [21] Adafruit Industries, «Adafruit Bicolor LED Square Pixel Matrix with I2C Backpack - Qwiic / STEMMA QT,» [En línea]. Available: <https://www.adafruit.com/product/902>. [Último acceso: 18 Junio 2023].
- [22] Analog Devices Inc., «Analog Devices MAX7219. Serially Interfaced.8-Digit, LED Display Drivers.,» [En línea]. Available: <https://www.analog.com/en/products/max7219.html>. [Último acceso: Junio 19 2023].
- [23] SUNFOUNDER, «Full Color RGB LED Matrix Driver Shield + RGB Matrix Screen,» 20 Marzo 2017. [En línea]. Available: http://wiki.sunfounder.cc/index.php?title=Full_Color_RGB_LED_Matrix_Driver_Shield_%2B_RGB_Matrix_Screen. [Último acceso: 19 Junio 2023].
- [24] SILICON TOUCH TECHNOLOGY INC., «DM163 8X3-CHANNEL CONSTANT CURRENT LED DRIVER,» [En línea]. Available:



http://wiki.sunfounder.cc/images/4/44/DM163_datasheet.pdf. [Último acceso: 19 Junio 2023].

- [25] MITSUBISHI ELECTRIC, «M54564P/FP,» [En línea]. Available: http://wiki.sunfounder.cc/images/6/66/M54564P_datasheet.pdf. [Último acceso: 19 junio 2023].
- [26] Arduino S.r.l, «Arduino Nano Every,» [En línea]. Available: <https://store.arduino.cc/collections/boards/products/arduino-nano-every>. [Último acceso: 31 Julio 2023].
- [27] Arduino S.r.l, «Arduino Leonardo without Headers,» [En línea]. Available: <https://store.arduino.cc/collections/boards/products/arduino-leonardo-without-headers>. [Último acceso: 1 Agosto 2023].
- [28] Farnell Components SL, «Datasheet Arduino Leonardo,» [En línea]. Available: <https://www.farnell.com/datasheets/1682240.pdf>. [Último acceso: 1 Agosto 2023].
- [29] Arduino S.r.l, «Arduino Micro,» [En línea]. Available: <https://store.arduino.cc/collections/boards/products/arduino-micro-without-headers>. [Último acceso: 1 Agosto 2023].
- [30] Arduino S.R.I, «Arduino Nano,» [En línea]. Available: <https://store.arduino.cc/collections/boards/products/arduino-nano>. [Último acceso: 1 Agosto 2023].
- [31] Espressif Systems Co, «ESP32-WROOM-32E Datasheet,» [En línea]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf. [Último acceso: 3 Agosto 2023].
- [32] Espressif Systems Co., «ESP32-DevKitC V4 Getting Started Guide,» [En línea]. Available: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc.html>. [Último acceso: 3 Agosto 2023].
- [33] Espressif Systems Co, «ESP32S2 Series Datasheet,» [En línea]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-s2_datasheet_en.pdf. [Último acceso: 3 Agosto 2023].
- [34] Espressif Systems Co, «ESP32-S2-DevKitM-1,» [En línea]. Available: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32s2/hw-reference/esp32s2/user-guide-devkitm-1-v1.html>. [Último acceso: 2 Agosto 2023].
- [35] Espressif Systems Co, «ESP32-S2-DevKitC-1,» [En línea]. Available: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32s2/hw-reference/esp32s2/user-guide-s2-devkitc-1.html>. [Último acceso: 2 Agosto 2023].



- [36] Espressif Systems CO, «ESP32-S3-DevKitM-1,» [En línea]. Available: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32s3/hw-reference/esp32s3/user-guide-devkitm-1.html>. [Último acceso: 3 Agosto 2023].
- [37] Espressif Systems CO, «ESP32-S3-DevKitC-1,» [En línea]. Available: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32s3/hw-reference/esp32s3/user-guide-devkitc-1.html>. [Último acceso: 3 Agosto 2023].
- [38] Raspberry Pi Foundation, «Raspberry Pi Documentation,» [En línea]. Available: <https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html>. [Último acceso: 4 Agosto 2023].
- [39] Arduino S.r.l, «Arduino Unio R4 WiFi,» [En línea]. Available: <https://store.arduino.cc/collections/boards/products/uno-r4-wifi>. [Último acceso: 4 Agosto 2023].
- [40] Adafruit Industries, «3.2" TFT LCD with Touchscreen Breakout Board w/MicroSD Socket - ILI9341,» [En línea]. Available: <https://www.adafruit.com/product/1743>. [Último acceso: 4 Agosto 2023].
- [41] Adafruit Industries, «3.5" TFT 320x480 + Touchscreen Breakout Board w/MicroSD Socket - HXD8357D,» [En línea]. Available: <https://www.adafruit.com/product/2050>. [Último acceso: 5 Agosto 2023].
- [42] Volkswagen, «Voltaje de la batería de un coche eléctrico,» 27 Julio 2022. [En línea]. Available: <https://www.volkswagen.es/es/revista/innovacion/voltaje-bateria-coche-electrico.html#:~:text=Entonces%20el%20voltaje%20o%20tensi%C3%B3n,potencia%20tiene%20dicho%20coche%20el%C3%A9ctrico..> [Último acceso: 7 Agosto 2023].
- [43] RS Components, «Convertidor de reducción síncrono MAX20008EAFOC/VY+, Convertidor reductor síncrono, 8A FC2QFN, 17 pines, Fijo,» [En línea]. Available: <https://es.rs-online.com/web/p/convertidores-buck/2149159?gb=s>. [Último acceso: 8 Agosto 2023].
- [44] RS Components, «Adaptador AC/DC RS PRO 90 → 264V ac, 5V dc, 1 salida, 8A, 40W, IEC 320 C14,» [En línea]. Available: <https://es.rs-online.com/web/p/adaptadores-ac-dc/2047562?gb=s>. [Último acceso: 10 Agosto 2023].
- [45] MajicDesings, «Github - MajicDesigns/MD_Parola,» [En línea]. Available: https://github.com/MajicDesigns/MD_Parola. [Último acceso: 15 Septiembre 2023].
- [46] MajicDesingd, «Github - MajicDesigns/MD_MAX72XX,» [En línea]. Available: https://github.com/MajicDesigns/MD_MAX72XX. [Último acceso: 15 Septiembre 2023].
- [47] Bodmer, «Github TFT_eSPI Library,» [En línea]. Available: https://github.com/Bodmer/TFT_eSPI. [Último acceso: 4 Septiembre 2023].
- [48] Adafruit Industries, «Github Adafruit_GFX Library,» [En línea]. Available: <https://github.com/adafruit/Adafruit-GFX-Library>. [Último acceso: 4 Septiembre 2023].

- [49] Bodmer, «Github JPEGDecoder,» [En línea]. Available: <https://github.com/Bodmer/JPEGDecoder>. [Último acceso: 08 Septiembre 2023].
- [50] ImpuseAdventure, «Github-GUIslice,» 4 Octubre 2021. [En línea]. Available: <https://github.com/ImpuseAdventure/GUIslice>. [Último acceso: 17 Diciembre 2023].
- [51] Android Developpers, «Android Studio,» [En línea]. Available: <https://developer.android.com/studio?hl=es-419>. [Último acceso: 20 Enero 2024].
- [52] N. Courses, «Youtube - Create the User Interface in Android Studio FULL COURSE || A Step-by-Step Guide,» 15 Febrero 2023. [En línea]. Available: https://www.youtube.com/watch?v=EVo1Ni5KE0&ab_channel=Nerd%27sCourses. [Último acceso: 12 Febrero 24].
- [53] C. W. Farhan, «Youtube - Android Food Recipe App Kotlin,» 7 Enero 2021. [En línea]. Available: https://www.youtube.com/playlist?list=PLpc1_FLg4LiMzUDQO6ALRgQwZ5SMzEXo0. [Último acceso: 20 Enero 2024].
- [54] Android Developpers, «Jetpack Compose,» [En línea]. Available: <https://developer.android.com/jetpack/compose?hl=es-419>. [Último acceso: 12 Febrero 2024].
- [55] I. DOMOTICS, «Kotlin - Comunicación Bluetooth con Android Studio y Arduino - Parte 1,» Youtube, 22 Noviembre 2022. [En línea]. Available: https://www.youtube.com/watch?v=ben6IruNVVg&ab_channel=INNOVADOMOTICS. [Último acceso: 8 Marzo 2024].
- [56] Android Developers, «Navigation con Compose,» [En línea]. Available: <https://developer.android.com/jetpack/compose/navigation?hl=es-419>. [Último acceso: 20 Febrero 2024].
- [57] Zep Labs, «DIY Led Display: WS2812b (8000 of them!),» Youtube, 3 Mayo 2018. [En línea]. Available: https://www.youtube.com/watch?app=desktop&v=gFQwZyjwEG0&ab_channel=ZepLabs. [Último acceso: 16 Marzo 2024].
- [58] PROIN S.L, «PROIN REMOLQUES Y EQUIPAMIENTOS LUMINOSOS.,» 5 Junio 2023. [En línea]. Available: <https://proinbal.es/soluciones/remolques-y-equipamientos-luminosos/>.
- [59] SEBA S.L, «SEBA Flechas y paneles luminosos para vehículos. Señalización,» [En línea]. Available: <http://sebasl.com/flechas-y-paneles-luminosos-para-vehiculos/?cn-reloaded=1>. [Último acceso: 5 Junio 2023].
- [60] Adafruit Industries, «Adafruit LED Backpacks. Downloaded,» [En línea]. Available: <https://learn.adafruit.com/adafruit-led-backpack/downloads>. [Último acceso: 18 Junio 2023].

- [61] e-ika S.L., «e-ika Matriz LED 8x8 MAX7219 para Arduino.,» [En línea]. Available: <https://www.e-ika.com/matriz-led-8x8-max7219-para-arduino-cable>. [Último acceso: 19 Junio 2023].
- [62] e-ika S.L., «e-ika Matriz LED 8x8 MAX7219 para Arduino, para soldar.,» [En línea]. Available: <https://www.e-ika.com/matriz-led-8x8-max7219-para-arduino-para-soldar>. [Último acceso: 19 Junio 2023].
- [63] e-ika S.L., «e-ika Matriz LED 8x8 mAX7219, (4 matrices) para Arduino,» [En línea]. Available: <https://www.e-ika.com/matriz-led-8x8-max7219-para-arduino-4-matrices>. [Último acceso: 19 Junio 2023].
- [64] Espressif Systems Co, «Espressif Development Boards,» [En línea]. Available: <https://www.espressif.com/en/products/devkits>. [Último acceso: 2 Agosto 2023].
- [65] Espressif Systems CO, «ESP32-S2-Saola-1,» [En línea]. Available: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32s2/hw-reference/esp32s2/user-guide-saola-1-v1.2.html>. [Último acceso: 3 Agosto 2023].
- [66] marco_c, «Parola A to Z – Double Height Displays,» 15 marzo 2017. [En línea]. Available: <https://arduinoplusplus.wordpress.com/2017/03/15/parola-a-to-z-double-height-displays/>. [Último acceso: 19 noviembre 2023].

Capítulo 10. ANEXOS.

En los documentos anexos de este proyecto se encontrará:

1. Carpeta del proyecto en Visual Studio Code que contiene el software del ESP32 para controlar el panel de emergencia en carretera.
2. Carpeta con la programación realizada en lenguaje Kotlin correspondiente a la aplicación creada para dispositivos Android en Android Studio.
3. Carpeta con los esquemas eléctricos y ficheros de fabricación realizados para la fabricación de la PCB.
4. Carpeta con los objetos 3D creados para la envolvente en formato STL.
5. Carpeta con los datasheet consultados a lo largo de este proyecto.
6. Carpeta con el diseño de la pantalla TFT realizado en GUIslicer.