Grado en Ingeniería Electrónica Industrial

Escuela Politécnica Superior de Jaén

UNIVERSIDAD DE JAÉN
ESCUELA POLITÉCNICA SUPERIOR DE JAÉN

Trabajo Fin de Grado

SISTEMA BÁSICO DE PROGRAMACIÓN Y DESARROLLO DE MICROCONTROLADOR ATMEGA

ALUMNO: JAVIER GARCÍA JIMÉNEZ

Tutor: Prof. D. Ángel Gaspar González Rodríguez

Dpto: Ingeniería Electrónica y Automática

Junio, 2019

INTRODUCCIÓN

Este proyecto tiene el objetivo de proporcionar al lector unos conocimientos sobre diseño y realización de una placa básica de entrenamiento para la iniciación al desarrollo de aplicaciones basadas en microcontroladores ATmega y el uso de herramientas para depuración de código de aplicaciones.

No se persigue la producción en masa de la placa que se va a diseñar, o de su utilización en entorno profesional, sino proporcionar una forma sencilla de diseño y montaje de una placa de entrenamiento para cualquier persona estudiante o aficionada del mundo de los microcontroladores.

ÍNDICE

ÍNDICE DE IMÁGENES

ÍNDICE DE TABLAS

1. ¿QÚE ES UN MICROCONTROLADOR?

Un microcontrolador es un circuito integrado digital programable que puede ser utilizado para diversos propósitos, por ejemplo, creación de prototipos de proyectos. Está compuesto de una unidad central de proceso (CPU), memorias (ROM y RAM) y líneas de entrada y salida (periféricos) [1].

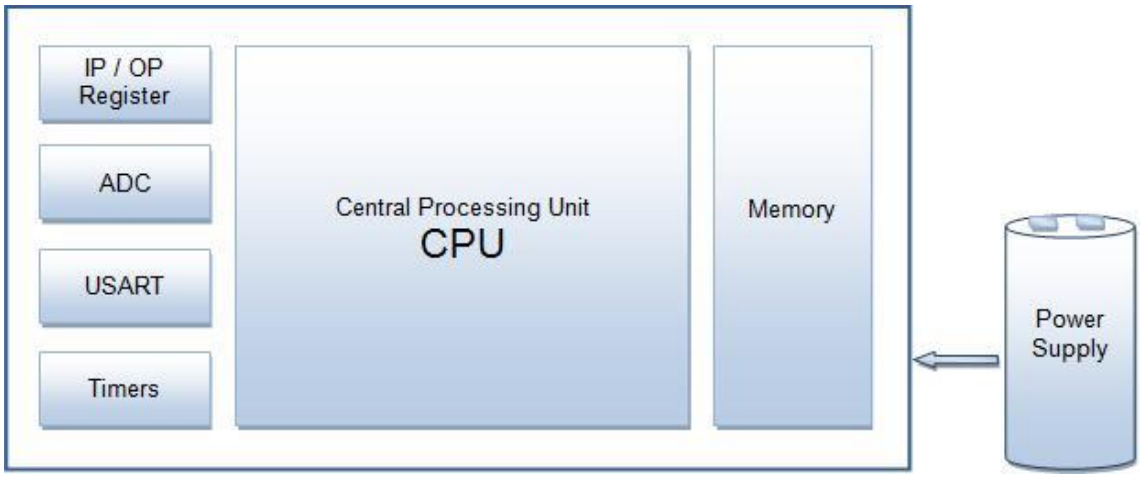


Imagen 1. Estructura básica de un microcontrolador.

Una vez que el microcontrolador está programado, el objetivo es realizar una o varias tareas específicas, a diferencia de un ordenador que se utiliza para realizar diversas tareas en una sola máquina.

1.1. CARACTERÍSTICAS DE UN MICROCONTROLADOR

ARQUITECTURA: Inicialmente todos los microcontroladores utilizaban arquitectura Von Neumann, caracterizada por disponer de una sola memoria y un sistema de buses único, pero actualmente se ha impuesto la arquitectura Harvard, que dispone de dos memorias independientes, con sus respectivos sistemas de buses. Una de ellas contiene solo instrucciones y datos constantes, y la otra solo datos.

CPU: Es el elemento más importante del microcontrolador. Se encarga de direccionar la memoria, recibe, decodifica y ejecuta la operación que implica la instrucción en curso, así como el almacenamiento en memoria del resultado. Existen dos tipos de CPUs según su arquitectura interna, RISC (Reduced Instruction Set Computer) y CISC (Complex Instruction Set Computer).

En la arquitectura RISC, el número de instrucciones es menor, más simple y rápido en ejecución que en la arquitectura CISC. En la arquitectura CISC, se realiza una tarea compleja empleando varios ciclos de reloj, mientras que la arquitectura RISC, se divide la tarea compleja en varias tareas más sencillas, la mayoría capaces de ser ejecutadas en un ciclo de reloj. La complejidad de la arquitectura RISC reside en el compilador, que debe determinar las tareas que son independientes para que estas puedan ser ejecutadas paralelamente, lo que se traduce en un funcionamiento más rápido y eficiente. En la arquitectura CISC, una instrucción compleja debe ser decodificada en instrucciones más simples que no son independientes, y no permiten ejecutarse en paralelo, además, el número de instrucciones que acceden a la memoria es mayor que en RISC, lo que reduce la velocidad y eficiencia [2].

MEMORIA: Está integrada en el propio chip. Una parte de la memoria es de tipo ROM y contiene el programa de instrucciones que gobierna la aplicación, la otra parte es de tipo RAM y contiene los datos manejados en el transcurso del programa.

RELOJ: Todos los microcontroladores necesitan una señal de reloj para funcionar. Esta señal es una cuadrada de alta frecuencia. Generalmente, el circuito de reloj está incorporado en el propio microcontrolador, y solo se necesitan unos pocos componentes externos para seleccionar y estabilizar la frecuencia de trabajo. Algunos microcontroladores disponen de un reloj integrado y no necesitan ningún componente externo.

PUERTOS DE ENTRADA Y SALIDA: Se utilizan para monitorizar y controlar dispositivos externos. Algunos de ellos están multiplexados, lo que aporta flexibilidad al microcontrolador, permitiendo seleccionar la función de cada pin como entrada/salida de propósito general o entrada/salida de algún periférico interno del microcontrolador.

PERIFÉRICOS INTERNOS O RECURSOS ESPECIALES: Además de la variación de cantidad de memoria, la incorporación o no de estos recursos, así como el número de ellos se traduce en las diferentes versiones de un microcontrolador. Estos recursos pueden ser temporizadores o contadores, convertidor A/D, comparador analógico, modulador PWM, interfaces de comunicación USART, I2C, USB, CAN, etc.

1.2. REVISIÓN DE ALGUNOS MODELOS

Existen en el mercado una gran variedad de microcontroladores, ya que es uno de los principales componentes de la electrónica digital. La clasificación más recurrente es atendiendo

al número de bits del bus de datos, en la cual están clasificados en 8, 16 y 32 bits. Cuanto mayor es el número de bits, mayor es la eficiencia en operaciones con datos grandes, por ejemplo, en procesado de audio o video, pero esto repercute en una mayor complejidad del microcontrolador y en un aumento del costo. Para aplicaciones que no requieran el uso de datos grandes, es suficiente con un microcontrolador de bus de 8 o 16 bits. Por este motivo y porque el microcontrolador que se va a utilizar en este proyecto es de 8 bits, en este apartado, se van a revisar solo las familias de 8 bits de las marcas más relevantes.

AVR. Son microcontroladores de 8 bits y arquitectura RISC de la marca Atmel (adquirida por Microchip en 2016). Son muy rápidos y eficientes, ya que ejecutan la mayoría de las instrucciones en un solo ciclo de reloj y disponen de varios modos de ahorro de energía [3]. Se dividen en tres familias:

- TinyAVR: son microcontroladores de pequeño tamaño, optimizados para una alta eficiencia energética y facilidad de uso. Aunque no disponen de mucha memoria, la integración de varios periféricos permite su utilización sin agregar componentes externos. Frecuencia máxima de funcionamiento de 20 MHz.
- MegaAVR: tienen un tamaño mayor y disponen de más memoria que los anteriores. Se utilizan en diseños que requieren mayor potencia y gran cantidad de código de programa. Disponen de ocho canales ADC, dos temporizadores de 8 bits y hasta tres de 16 bits, hasta diez salidas PWM, hasta tres puertos UARTA y dos SPI. Soportan una frecuencia de hasta 20 MHz.
- XMegaAVR: esta familia es una mejora de la anterior, dispone de doce canales ADC, hasta cinco temporizadores de 16 bits, hasta dieciséis salidas PWM, interfaz USB, hasta cinco puertos UART y siete SPI. Soportan una frecuencia máxima de 32 MHz.

PIC. Nombre procedente de “Programmable Interface Controller” de la marca Microchip. Son microcontroladores de fácil uso, de bajo costo y muy fáciles de programar en lenguaje C. Utilizan arquitectura RISC con un conjunto de hasta 35 instrucciones. Necesitan de cuatro ciclos de reloj para ejecutar una instrucción que cambie el contador de programa y ocho ciclos para instrucciones de ramificación. Se dispone de una gran variedad de versiones en distintos encapsulados, desde 8 hasta 84 pines, además de varias herramientas de desarrollo software y hardware de bajo coste [4].

- PIC10 (10FXXX): Los dispositivos PIC10FXXX de 8 bits son los más pequeños, de menor costo y rendimiento. Disponen de un máximo de 4 entradas/salidas en un encapsulado de 8 pines.
- PIC12 (PIC12FXXX): Son los dispositivos de rango medio-bajo, cuentan con un máximo de 8 entradas/salidas, una pila de hardware de 8 niveles y memoria de datos EEPROM. También disponen de varios periféricos analógicos y digitales en serie, como: SPI, I2C, USART y convertidor A / D.
- PIC16 (16FXXX): Aumentan el número de periféricos de la familia anterior además de permitir aplicaciones que necesitan más espacio de código con una memoria Flash de entre 7 y 14 Kbyte y un mayor número de entradas/salidas.
- PIC18 (18FXXX): La familia PIC18 utiliza una arquitectura de palabra de programa de 16 bits, un multiplicador de hardware 8x8 y múltiples interrupciones internas y externas, con el rendimiento más alto en la familia de 8 bits a una frecuencia máxima de funcionamiento de 64 MHz.

STM8. De la marca STMicroelectronics, son microcontroladores robustos que ofrecen un rendimiento de hasta 20 MIPS (millones de instrucciones por segundo) a una frecuencia de 24 MHz, además de una gran cantidad de entradas/salidas y periféricos integrados [5].

- STM8S: Se suelen utilizar en control de motores, administración de baterías, fuentes de alimentación y administración de energía. Son microcontroladores especializados en rendimiento analógico y aplicaciones DiSEqC (Digital Satellite Equipment Control), que es un protocolo de comunicación especial utilizado entre un receptor satelital y un computador que controla el posicionamiento de dicho receptor.
- STM8L: Son dispositivos de consumo ultra bajo. Se utilizan en aplicaciones donde el consumo de energía es crítico, por ejemplo, dispositivos portátiles a batería. En el modo de menor potencia consumen un mínimo de 0.3 µA y 180 µA en modo normal.
- STM8AF: Se utilizan en aplicaciones automotrices en las que no es posible comprometer los parámetros. Son capaces de soportar temperaturas ambientales de hasta 150 °C.
- STM8AL: Se utilizan en aplicaciones de consumo de energía crítico, que funcionan a batería, como el uso de sensores, RTC, …

NXP. Los microcontroladores que se describen a continuación, de la marca NXP Semiconductor [6], fueron adquiridos por esta cuando eran propiedad de Freescale, que, a su vez, era la división de microcontroladores de Motorola después de independizarse de esta.

- S08P: son dispositivos fiables para entornos industriales y en interfaces expuestas al medio exterior, con buen rendimiento frente a descargas electrostáticas y transitorios eléctricos. Entre sus características más importantes destacan una interfaz de detección táctil (TSI) y un temporizador flexible para control de motores, lo que reduce el costo del sistema.
- S08S: son una solución de alto rendimiento para aplicaciones de control de motores de baja potencia, y para ello poseen una unidad pre-drivers MOSFET trifásico, amplificadores de corriente, protección contra sobretensión y sobrecorriente. Cuenta con un controlador de administración de energía, convertidor A/D de 12 bits, …
- S08Q: proporcionan una eficiencia energética extrema con un rango de tensión de 1.8 a 3.6 V para equipos portátiles con batería. Posee unos registros que controlan la activación de reloj para los periféricos no utilizados y así optimizar el consumo de energía. Entre los periféricos que incorpora destacan un RTC, convertidor A/D de 12 bits, UART, comparador analógico… La frecuencia máxima de funcionamiento es 20 MHz.
- RS08: son dispositivos de bajo costo que incorpora un controlador LCD. Están indicados para aparatos pequeños, equipos médicos u otras aplicaciones industriales. Frecuencia de funcionamiento de hasta 10 MHz.

1.3. SELECCIÓN DEL MICROCONTROLADOR

El objetivo de la utilización de un microcontrolador es controlar y realizar una serie de funciones de forma eficiente, con un consumo de energía mínimo y con un reducido costo económico. Por tanto, para la elección de un microcontrolador se deben tener en cuenta aspectos como los que se mencionan a continuación para seleccionar el dispositivo mínimo que cumpla los requerimientos de la aplicación:

- Cantidad de memoria.
- Frecuencia máxima y mínima de funcionamiento.
- Longitud de los datos a manejar, 8, 16, 32 bits, …

- Tipo y cantidad de periféricos, ADC, PWM, Temporizadores…
- Protocolos de comunicación USART, I2C, SPI, …
- Tensión de alimentación.

En el caso del presente trabajo, la gama del microcontrolador estaba impuesta, y era ATmega de Atmel. Aparte de esto, las características imprescindibles con las que ha de contar el microcontrolador a elegir son las siguientes:

Protocolos de comunicación JTAG, I2C, UART o SPI. Una de las especificaciones de este proyecto es el diseño de una tarjeta de desarrollo que permita In-System-Programming y emulación/depuración de programa. El protocolo SPI permite In-System-Programming pero no emulación/depuración, característica que sí incorpora el protocolo JTAG y motivo por el que, inevitablemente, el microcontrolador seleccionado debe ser compatible con dicho protocolo.

Tipo de encapsulado PDIP. Debido a que la tarjeta será ensamblada de forma manual, la técnica de soldadura más apropiada para este caso se da para encapsulados tipo PDIP.

Utilizando el buscador paramétrico disponible en la página web de la marca Atmel, y estableciendo los parámetros imprescindibles antes mencionados, protocolo JTAG y encapsulado PDIP, se obtiene el resultado mostrado en la Imagen 2.

New/Popular 8-bit AVR Products

Reset All Filters | Show/Hide Columns | Show ALL Products | Download | Switch Views: Summary / Show All Specs

To SORT a column click the column header. Use CTRL key to select multiple values.

Product	Program Memory (KB)	Program Memory Size (KWords)	UART	SPI	I2C	JTAG	Packages
ATmega324PA	32	16	2	3	1	Debug/Program/Boundary Scan	40/PDIP 44/TQFP 44/VQFN 49/VFBGA
ATmega32A	32	16	1	1	1	Debug/Program/Boundary Scan	40/PDIP 44/TQFP 44/VQFN
ATmega644PA	64	32	2	3	1	Debug/Program/Boundary Scan	40/PDIP 44/TQFP 44/VQFN
ATmega1284P	128	64	2	3	1	Debug/Program/Boundary Scan	40/PDIP 44/TQFP 44/VQFN
ATmega164PA	16	8	2	3	1	Debug/Program/Boundary Scan	40/PDIP 44/TQFP 44/VQFN 49/VFBGA
ATmega16A	16	8	1	1	1	Debug/Program/Boundary Scan	40/PDIP 44/TQFP 44/VQFN

Imagen 2. Búsqueda parametrizada del microcontrolador.

Se ha elegido el ATmega16 que es el más simple y económico que cumple las especificaciones. Los demás dispositivos tienen dos módulos PWM más y dos canales de comunicación UART y SPI más, características que no son necesarias en este proyecto. El ATmega32 es igual que el ATmega16, pero con el doble de memoria, por lo que también se podrá utilizar en la tarjeta de desarrollo que se va a diseñar en este proyecto. La Tabla 1 muestra las características principales del microcontrolador seleccionado.

Program Memory Type	Flash
Program Memory Size (KB)	16
CPU Speed (MIPS/DMIPS)	16
SRAM Bytes	1,024
Data EEPROM/HEF (bytes)	512
Digital Communication Peripherals	1-UART, 1-SPI, 1-I2C
Capture/Compare/PWM Peripherals	1 Input Capture, 1 CCP, 4PWM
Timers	2 x 8-bit, 1 x 16-bit
Number of Comparators	1
Temperature Range (C)	-40 to 85
Operating Voltage Range (V)	2.7 to 5.5
Pin Count	44

Tabla 1. Características básicas del microcontrolador ATmega16.

2. PROGRAMACIÓN DE UN MICROCONTROLADOR

Las aplicaciones que realiza un microcontrolador son diseñadas previamente en un PC utilizando, generalmente, un lenguaje de alto nivel y posteriormente un compilador transforma el código de la aplicación en un archivo hexadecimal. Este archivo hexadecimal debe cargarse en la memoria flash del microcontrolador para que este pueda desempeñar su función. El proceso de carga en la memoria suele denominarse "flasheado", y para ello, los microcontroladores deben disponer de una interfaz de programación.

Existen varios métodos de programación, pero los más extendidos son SPI y JTAG. Además de estos métodos, existen otros que se utilizan en menor medida, pero también son importantes y se describirán en este capítulo.

Para que se pueda llevar a cabo el proceso de programación debe haber una conexión entre el microcontrolador y el PC, y de esto se encarga el programador o adaptador de programación.

La interfaz de comunicación estándar de los ordenadores actuales es el puerto USB, pero la mayoría de los adaptadores de programación para microcontroladores no utilizan USB debido a que fueron diseñados con anterioridad a la implantación de USB, cuando una de las interfaces estándar era RS-232. Para solucionar este problema existen en el mercado adaptadores como el FTDI FT232 o el CH340G cuya misión es adaptar la interfaz RS232 a UART para comunicarse con el PC mediante el puerto serie COM, que comparte el puerto físico con USB. De esta forma se evita adoptar la interfaz USB y las complicaciones que ello conlleva.

2.1. IN-SYSTEM PROGRAMMING

También denominado ICP (In-Circuit-Programming) o ICSP (In-Circuit Serial Programming), es un método que permite la programación y reprogramación en serie de memorias no volátiles (flash y EEPROM) de microcontroladores mientras están instalados en un sistema completo [7].

Previamente a este sistema, los microcontroladores debían ser programados en dispositivos externos de programación.

VENTAJAS

Con este método se evita incluir en la PCB un módulo programador, lo que reduce el costo económico y tiempo dedicados a la programación, durante el desarrollo en laboratorio y en actualizaciones de software.

Se evita la programación de microcontroladores antes de ser soldados en la placa destino y se elimina la posibilidad de que los dispositivos reciban algún daño.

Se permite a los fabricantes comprar mayores cantidades de chips a un mejor precio sin el riesgo que implica adquirir productos preprogramados.

Se reduce el tiempo de fabricación de placas al integrar la programación y las pruebas en una sola fase de producción. Además, permite la actualización de códigos durante la fase de producción.

INCONVENIENTES

Requiere de la adaptación de la línea de montaje para incluir los sistemas de programación y control, aumentando el coste de producción.

No todos los métodos de programación permiten In-System-Programming.

Los microcontroladores AVR permiten la programación serie mediante los protocolos JTAG y SPI, además de otros métodos menos usuales, por ejemplo, programación de alto voltaje [8].

2.2. PROTOCOLO SPI

SPI (Serial Peripherical Interface) es un protocolo de comunicación síncrona que trabaja en modo full dúplex para recibir y transmitir información simultáneamente, permitiendo que los dispositivos puedan comunicarse entre sí utilizando dos líneas de datos diferentes y una línea de reloj que proporciona el sincronismo.

Dentro de este protocolo, uno de los dispositivos se define como maestro y los demás como esclavos. Cuando existe más de un esclavo puede resultar necesaria la incorporación de una nueva línea de selección para cada esclavo, resultando en un mínimo de cuatro líneas a partir de una misma referencia de tensión.

- MOSI (Master Out Slave In): Línea que transmite los datos del maestro hacia los esclavos.
- MISO (Master In Slave Out): Línea que transmite los datos de los esclavos hacia el maestro.
- SCK (Clock): Línea que proviene del maestro y se encarga de sincronizar los dispositivos.
- SS (Slave Select): Línea encargada de seleccionar y a su vez, habilitar un esclavo.

Este protocolo permite la conexión de dispositivos en paralelo.

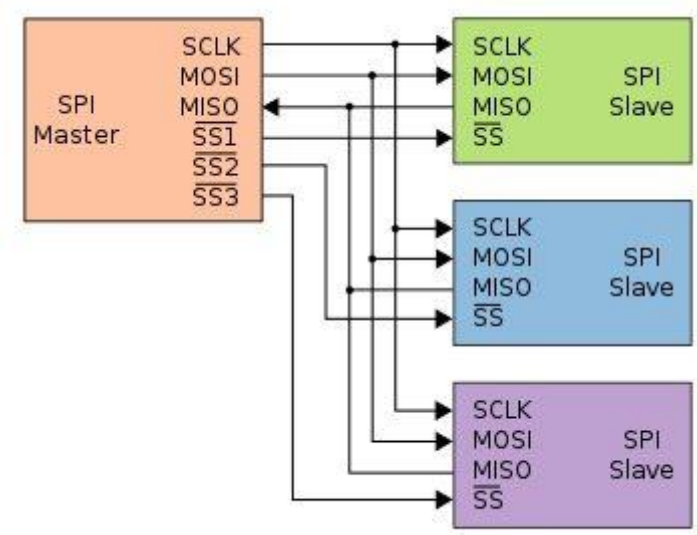


Imagen 3. Protocolo SPI. Conexión de varios dispositivos en paralelo.

Se permite una alta velocidad de transmisión, configurable a través de software, dependiendo también de los dispositivos utilizados en el sistema. Los datos transmitidos no están limitados a una anchura de ocho bits. Solo es adecuado a cortas distancias, inferiores a 30 cm. No se incluye un mecanismo de control para saber si un dato ha sido transmitido correctamente.

El método de programación ISP es, si no el más común, de los más utilizados para programar memorias flash o EEPROM, así como los bits de configuración y bloqueo en el caso de microcontroladores Atmel AVR. Esto se debe, en parte, a que existen en el mercado una gran variedad de programadores clonados a un precio muy reducido, y, por otra parte, a la variedad de programas que se encuentran en Internet para el uso de dichos programadores, lo que, en conjunto, ofrece una gran facilidad de utilización y eficacia de este método [9].

2.3. PROTOCOLO JTAG

El protocolo JTAG es un “In-System Debugging” o depuración en sistema para microcontroladores, aunque esta interfaz también permite la programación.

JTAG es el acrónimo de Join Test Action Group, convertido en el estándar IEEE 1149.1 “Standard Test Access Port and Boundary-Scan Architecture” [10] y utilizado para la comprobación de circuitos impresos.

JTAG es una herramienta de depuración en sistema, lo que facilita el desarrollo de aplicaciones complejas, permitiendo la emulación del circuito y simultáneamente controlar el estado del programa del microcontrolador mientras este se ejecuta en el circuito.

Un entorno de desarrollo de la interfaz JTAG en un PC permite al programador controlar la ejecución del programa en el microcontrolador en tiempo real, detener o pausar la ejecución, establecer puntos de ruptura en el código, así como, manipular los registros internos del microcontrolador, avanzar o retroceder paso a paso en el código, además de otras funciones. Todo ello con el fin de corregir errores de código y lógica de los sistemas utilizados. Todas estas funcionalidades no son posibles con el protocolo SPI.

La conexión entre el entorno de desarrollo y el microcontrolador se lleva a cabo mediante el depurador, que es el hardware que implementa el protocolo JTAG. Los depuradores más conocidos para los microcontroladores AVR son el JTAG-ICE mk1, mk2 o mk3.

Imagen 4. Fotografías de programadores/depuradores JTAG-ICE.

Una característica del protocolo JTAG que tiene un gran potencial es el “Boundary Scan” o testeo de límites, característica que tiene la capacidad de controlar y monitorizar los niveles lógicos de los pines de entradas y salidas digitales del microcontrolador, permitiendo configurar los dispositivos para controlar los valores en sus pines de salida y observar los valores de

entrada recibidos para, de esta manera, proporcionar un mecanismo de puesta a prueba y verificación. Este modo es capaz de detectar pistas abiertas o cortocircuitos, verificación de la integridad de los periféricos del microcontrolador, de las memorias flash, RAM, etc., reduciendo significativamente el acceso físico requerido para probar una placa [11].

Uno de los beneficios de este método de testeo es que no se necesitan sistemas de inspección especializados, como inspección de rayos X, que no están disponibles en un laboratorio de diseño.

Además de las funciones de testeo y depuración, el protocolo JTAG permite la programación y verificación de las memorias flash y EEPROM, y en el caso de microcontroladores AVR, también de los bits de configuración y bloqueo.

Está compuesta de cinco líneas conocidas como puertos de acceso de prueba o TAP:

- TCK: es la señal de reloj, proporcionada por el depurador para sincronizar al microcontrolador.
- TDO: señal de salida de datos.
- TDI: señal de entrada de datos.
- TMS: esta señal está gobernada por el controlador de la máquina de estados para seleccionar entre los diferentes modos de funcionamiento.
- RESET: señal de control de reset del microcontrolador. Esta señal no es requerida por la interfaz, pero se utiliza para resetear el microcontrolador y de esta manera asegurarse de que este no actúa sobre ninguno de los pines entradas o salidas utilizados por la interfaz durante la programación.

En el caso de Atmel, el soporte para depuración se considera privado, por lo que solo se distribuye a proveedores seleccionados la herramienta de depuración, depurador o debugger.

2.4. DEBUGWIRE

Es un método de depuración desarrollado por Atmel para sus microcontroladores AVR que no implementan el protocolo JTAG. Fue diseñado especialmente para los dispositivos cuyo número de pines no permitía la dedicación de cuatro de ellos para utilizar JTAG y mantener un número considerable para las posibles aplicaciones.

El protocolo debugWire está basado en un bus de una sola línea que permite una transmisión de datos bidireccional. La línea utilizada para este bus es el pin RESET de los dispositivos, lo que permite ser implementado en los microcontroladores más pequeños de la familia ATtiny, como es el caso de los ATtiny25/45/85 disponibles en encapsulados de 8 pines, de los cuales, seis son entradas/salidas y dos para alimentación y masa [12].

Este método de depuración es menos potente que JTAG y no tiene acceso a los bits de configuración y bloqueo.

El firmware del protocolo, propiedad de Atmel, es considerado como privado y la única forma de utilizarlo es mediante los depuradores JTAG ICE mk2 y mk3, Atmel ICE y AVR Dragon.

2.5. BOOTLOADER

No es un método de programación, sino un gestor de arranque que se encuentra en una sección reservada, configurable por el programador, de la memoria flash. El bootloader hace uso de las funciones de auto-modificación de flash, disponible en los microcontroladores AVR compatibles, para permitir la programación a través de los datos de programa cargados desde una fuente externa, como puede ser una memoria de datos externa o mediante comunicación serie con un PC [13].

El bootloader debe ser programado en el microcontrolador mediante un programador externo, ya que por sí mismo no tiene tal función. Igualmente, el bootloader no es compatible con la modificación de los bits de configuración y necesita del programador externo para tal función.

La limitación de este método de programación es el hecho de que, por sí mismo, ocupa cierto espacio de memoria, lo que limita la cantidad de memoria flash disponible para la aplicación.

3. ENTORNO DE PROGRAMACIÓN Y DEPURACIÓN

Un entorno de programación se conoce como IDE (Entorno de Desarrollo Integrado) y es un programa, o un conjunto de programas, que engloban todas las tareas necesarias para el desarrollo de un programa o aplicación [14]. Las tareas que debe llevar a cabo este programa son:

- Edición del código fuente de la aplicación.
- Compilación y enlazado.
- Ejecución y depuración de la aplicación.

Un IDE puede incorporar una serie de herramientas, utilidades, ayudas o asistentes que faciliten el desarrollo de aplicaciones, así como reducir considerablemente el tiempo empleado en esta fase de un proyecto. Además, debe dar soporte a los microcontroladores, programadores y depuradores propios de la marca, y ser capaz de comunicarse con el microcontrolador mediante un programador o depurador.

Los componentes esenciales que debe incorporar un entorno de programación y depuración son el editor de texto, el compilador, el enlazador y el depurador.

EDITOR DE TEXTO

Es un programa que permite escribir o editar las instrucciones del programa, así como los comentarios para un fácil entendimiento del código. Además, debe generar los archivos de texto que contienen dichas instrucciones de forma que sea compatible con el siguiente componente, el compilador.

COMPILADOR

Es un programa cuya función es convertir, interpretar y traducir los archivos generados en el editor de texto en lenguaje máquina que pueda ser ejecutado en el microcontrolador.

ENLAZADOR

Cuando se desarrollan aplicaciones complejas, el código debe estar organizado en subprogramas. Cada subprograma forma un archivo, al igual que el programa principal. La función del enlazador es unir en un único archivo el resultado de la compilación de cada uno de los subprogramas, así como de las bibliotecas utilizadas. El archivo generado contiene el código fuente de la aplicación, generalmente de tipo hexadecimal, el cual será cargado en la memoria de programa del microcontrolador.

DEPURADOR

Una vez programado el microcontrolador, lo habitual es que no funcione todo de la forma esperada, y sea necesario solucionar los errores. Hay errores evidentes que son fácil de localizar, pero otros no se encuentran tan fácil. La función del depurador es ayudar a solucionar todos estos errores. Para ello dispone de una serie de utilidades que permiten controlar el curso del programa mientras este se ejecuta en el microcontrolador, así como editar los registros, monitorizar variables, etc. Para llevar a cabo la depuración es necesario que haya comunicación entre el IDE y el microcontrolador a través del dispositivo hardware programador/depurador.

3.1. EJEMPLOS DE IDEs

La marca STMicroelectronic dispone de una gran variedad IDEs, por ejemplo, el STVD-STM8 para microcontroladores de 8 bits. Es un entorno básico que proporciona las utilidades básicas e imprescindibles que cualquier IDE debe poseer.

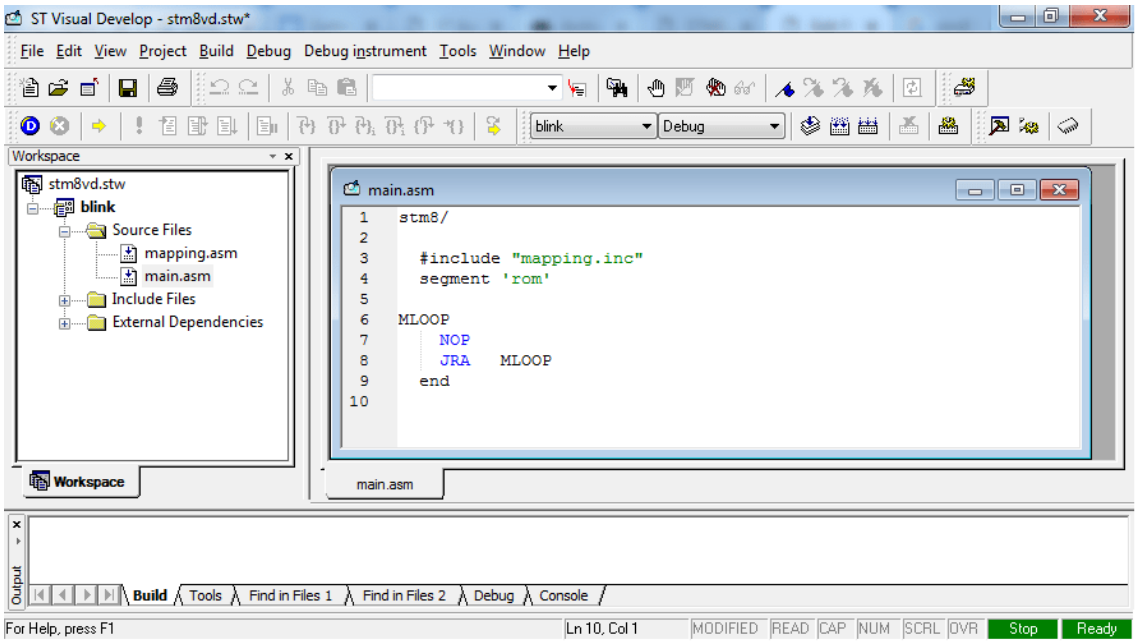


Imagen 5. Entorno de desarrollo STVD-STM8.

Para los microcontroladores de 32 bits, STMicroelectronic ofrece STM32CubeIDE (ver Imagen 6).

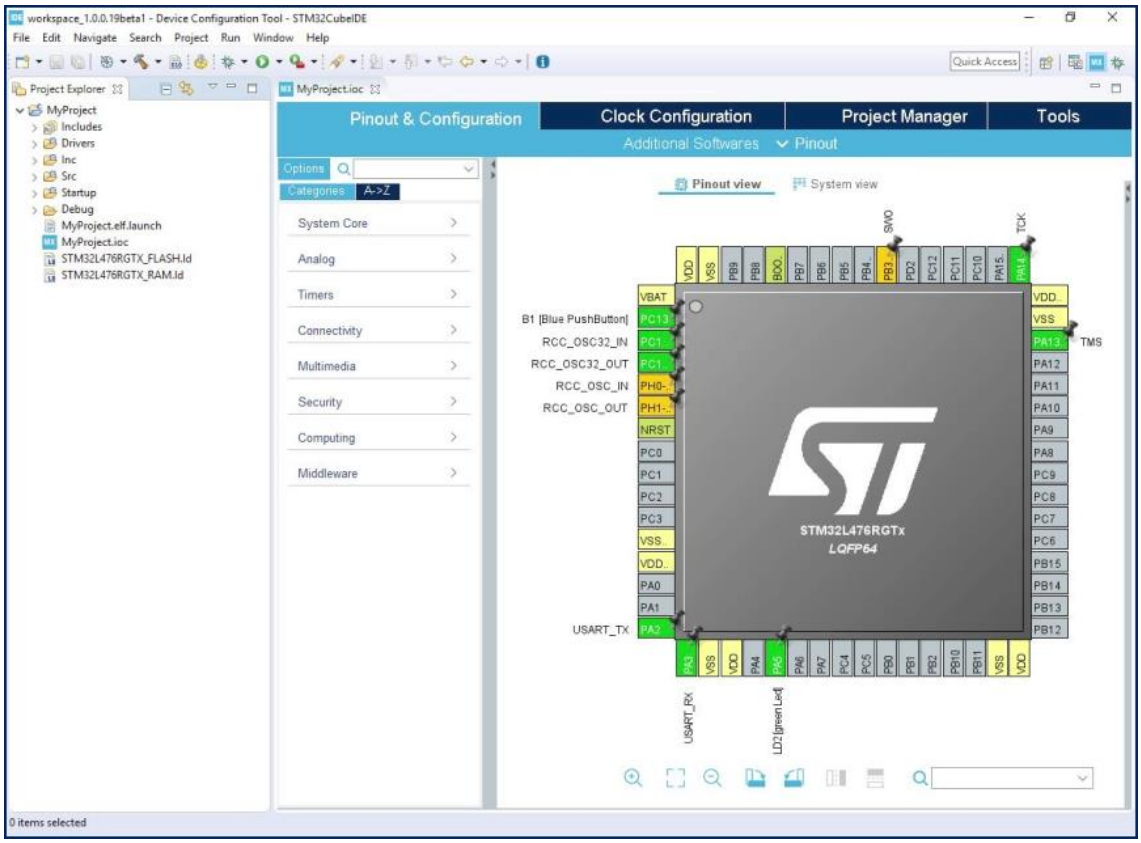


Imagen 6. Entorno de desarrollo STM32Cube.

Este es un entorno más avanzado que permite configurar las funciones de cada pin directamente sobre una vista del microcontrolador y generar automáticamente el código asociado a dichas funciones.

La marca Atmel ofrece el Atmel Studio (ver Imagen 7) para los microcontroladores AVR y SAM. Es un entorno básico, ya que no dispone de ningún asistente que facilite el desarrollo de las aplicaciones.

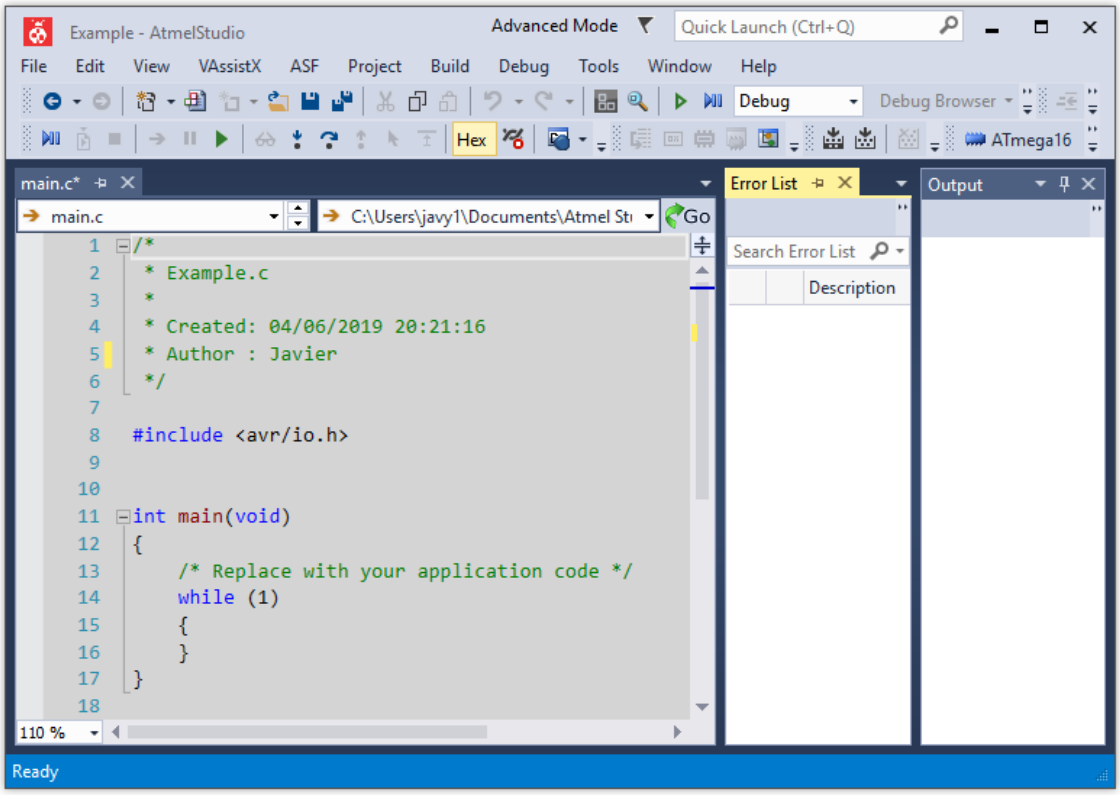


Imagen 7. Entorno de desarrollo Atmel Studio 7.

La marca Microchip ofrece el entorno MPLAB X IDE para sus microcontroladores PIC, aunque en su última versión, también es compatible con AVR y SAM, ya que Microchip es propietaria de Atmel. MPLAB X es un entorno avanzado, permite la instalación de extensiones como MPLAB Code Configurator, que es un asistente de configuración para los dispositivos.

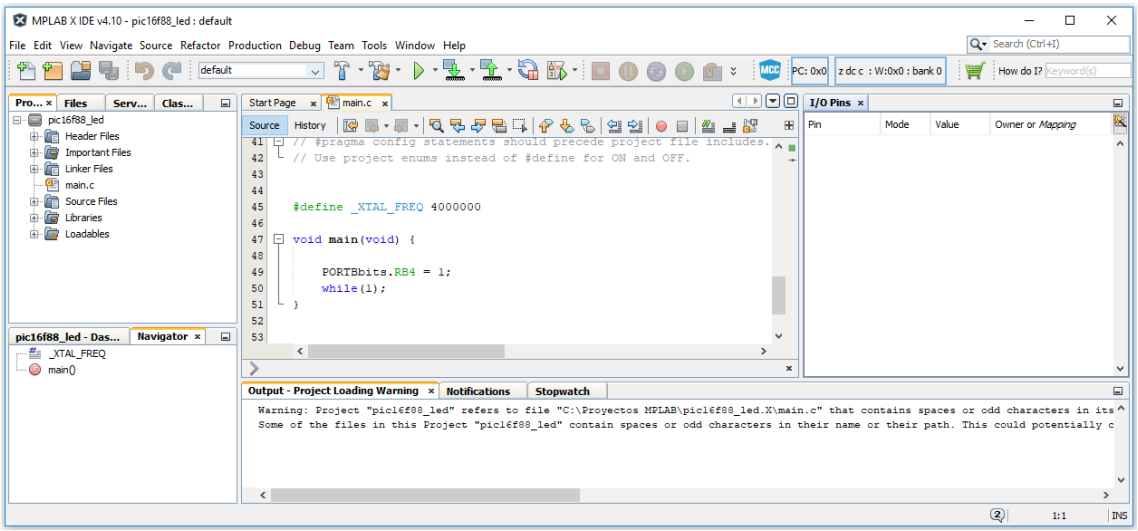


Imagen 8. Entorno de desarrollo MPLAB X.

Cada marca de microcontroladores proporciona su IDE, que debe ser flexible en la organización de los accesos a las utilidades y herramientas, en la organización de las diferentes ventanas y pestañas, aspecto del editor, y otras muchas funcionalidades, con el fin de minimizar el tiempo de desarrollo de las aplicaciones, y que este sea lo más cómodo posible para el programador. De esta forma se reducirá el número de errores durante la programación.

Algunos IDEs incorporan asistentes que permite configurar la frecuencia de reloj o los periféricos, entre otros, de forma fácil y generar el código automáticamente, lo que facilita en buena medida la tarea de programación. Algunos de estos asistentes, al igual que ciertas utilidades, solo están disponibles bajo licencia.

Por tanto, las versiones básicas de los IDEs suelen ser gratuitas, pero permiten la ampliación mediante extensiones o nuevas herramientas, algunas gratuitas y otras bajo licencia.

4. DEPURADOR JTAG ICE

La programación y depuración mediante JTAG de los microcontroladores ATmega compatibles con este protocolo se realiza a través de un programador/depurador JTAG. Como se ha mencionado en un apartado anterior, existen diferentes dispositivos de depuración JTAG que suelen tener precios relativamente elevados, excepto el JTAG ICE mk1, que fue liberado por Atmel, haciendo posible su implementación sin necesidad de licencia. Por este motivo, el depurador que se va a implementar en la tarjeta de desarrollo objetivo de este proyecto será el JTAG ICE.

Imagen 9. Programador/depurador JTAG-ICE mk1.

JTAG ICE

Es una herramienta “In-System Debug” o depuración en sistema, que permite la depuración de los microcontroladores ATmega compatibles con la interfaz JTAG instalados en la placa destino. Para el manejo de esta herramienta se utiliza el entorno de depuración AVR Studio en sus versiones 3.52 hasta 4.18.

La interfaz JTAG es un controlador de puerto de acceso de prueba o TAP, de cuatro líneas, y cumple con el estándar IEEE 1149.1 [10] que fue desarrollado como un estándar para probar de forma eficiente la conectividad de una placa de circuitos. Los dispositivos AVR de Atmel, compatibles con esta interfaz, tienen soporte completo de programación y depuración en chip.

JTAG ICE dispone de un chip interno que contiene la lógica de depuración encargada de controlar y monitorear la ejecución del microcontrolador conectado al depurador. La depuración se realiza con un método OCD (On Chip Debug) en el microcontrolador destino, al

contrario que los métodos de emulación, cuya función es imitar el comportamiento exacto del microcontrolador utilizando el hardware propio del emulador.

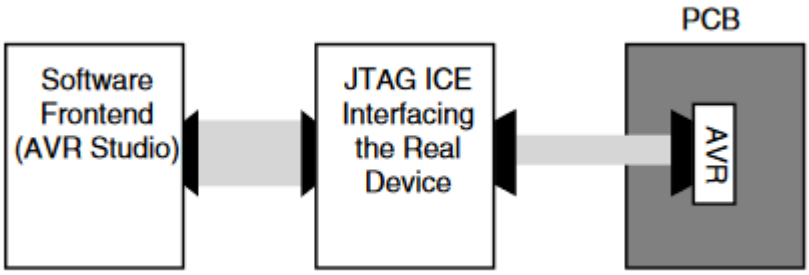


Imagen 10. Conexión interna del sistema OCD JTAG-ICE.

El dispositivo JTAG ICE tiene varios métodos y características de funcionamiento [15].

- Modo Run: La ejecución del código en el microcontrolador se realiza de forma independiente. Cuando se pausa la ejecución, el sistema OCD lee todos los registros y memorias y se transmite la información al entorno AVR Studio. Este modo no permite rastrear el código ejecutado con anterioridad a una pausa.
- Modo Stopped: Se activa al alcanzar un punto de interrupción y se detiene la ejecución del programa. Las E/S siguen ejecutándose, por ejemplo, se puede recibir un dato USART mientras el programa se encuentra detenido.
- Puntos de interrupción de software: Es una función que se coloca en el entorno de depuración y se traduce en una instrucción de interrupción en la memoria flash que, cuando se ejecuta, produce una detención de la ejecución en el microcontrolador. Estos puntos de interrupción requieren la reprogramación del microcontrolador y se utilizan como puntos fijos.
- Puntos de interrupción de hardware: El JTAG ICE permite tres puntos de interrupción hardware y se diferencian de los anteriores en que pueden ser modificados durante la ejecución, sin necesidad de reprogramar la memoria flash.
- Registros de E/S: Cuando se ejecuta un punto de interrupción o una pausa, la ejecución del programa se detiene, el programador lee todos los registros de E/S del microcontrolador y los presenta en el entorno de depuración.
- Simple paso a paso: Son funciones del entorno de depuración que permiten controlar la ejecución del código paso a paso, en dos opciones: paso a paso por

procedimientos (el contador de programa no pasa al interior de los procedimientos) y paso a paso por instrucciones

La conexión entre JTAG ICE y la placa de destino requiere un mínimo de seis líneas, TCK, TDO, TDI, TMS, Vcc y GND. La línea Reset es opcional y se utiliza para poner al microcontrolador en modo reset y así evitar cambios en las líneas JTAG durante la programación. El depurador no admite la conexión de varios microcontroladores en cadena.

Aunque el dispositivo no fuese desarrollado por completo, se puede utilizar si se aceptan algunas limitaciones como:

- Entorno de programación y depuración antiguo, sin asistentes que faciliten el desarrollo de aplicaciones y sin la posibilidad de actualizaciones por incompatibilidad con el depurador.
- Solo se dispone de tres puntos de ruptura por hardware, modificables durante la depuración.
- Los putos de ruptura por software no estaban implementados cuando se canceló el desarrollo del dispositivo, por lo que no están disponibles.

A cambio de estas limitaciones se obtiene la ventaja de un funcionamiento aceptable a un costo muy económico.

Para un uso profesional esta no es la herramienta indicada por las limitaciones mencionadas y por las bajas prestaciones que ofrece. Para este tipo de usos existen en el mercado otras herramientas con altas prestaciones y mayor soporte, pero también a un costo considerablemente superior.

5. EL MICROCONTROLADOR ATMEGA16

El ATmega16 es un microcontrolador de 8 bits de marca Atmel, perteneciente a la familia MEGA de la gama AVR.

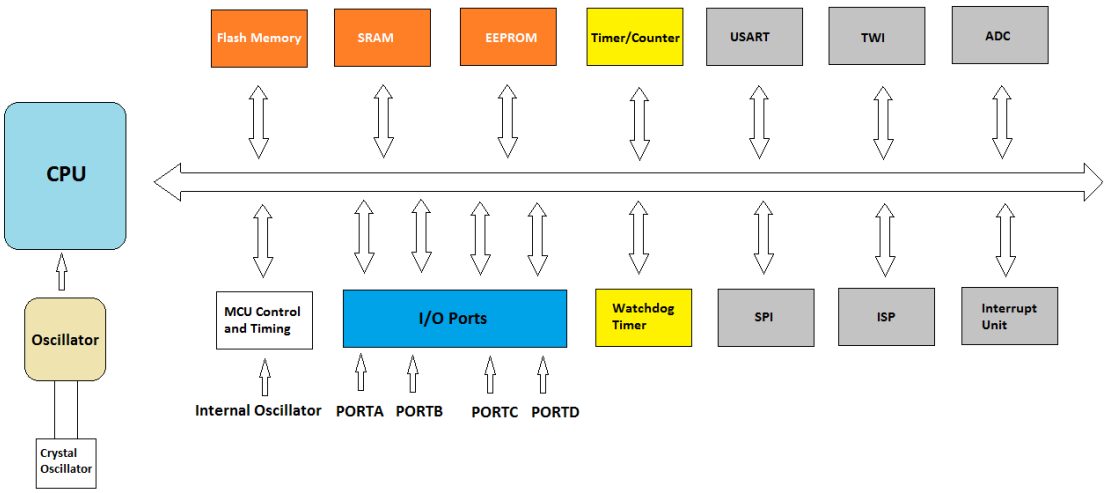


Imagen 11. Estructura básica del microcontrolador ATmega16.

CPU

Para maximizar el rendimiento y el paralelismo, el AVR utiliza una arquitectura RISC con un set de 131 instrucciones. Las instrucciones en la memoria de programa se ejecutan con una segmentación de dos etapas, es decir, mientras una instrucción se está ejecutando, la siguiente instrucción se está capturando de la memoria de programa simultáneamente, lo que permite ejecutar instrucciones en cada ciclo de reloj.

Consta de un archivo de 32 registros de 8 bits de propósito general conectados directamente a la unidad aritmético lógica, ALU, permitiendo a esta realizar operaciones en un único ciclo de reloj y logrando un rendimiento aproximado de 1 MIPS (Millones de Instrucciones Por Segundo) por cada MHz de frecuencia.

MEMORIA FLASH

El ATmega16A dispone de 16 Kbytes de memoria flash reprogramable para el almacenamiento del programa. Debido a que el contador de programa es de 13 bits y a que todas las instrucciones del AVR tienen 16 o 32 bits de anchura, la memoria está organizada como 8k x 16. 8k se debe a que los 13 bits del contador de programa permiten direccionar 2^{13} = 8192 espacios de memoria, y 16 se debe a la anchura mínima de instrucción. La memoria se

divide en sección inferior de arranque (bootloader) y sección superior de aplicación, cuyos tamaños se pueden modificar mediante los *fuse bits*, o bits de configuración. Esta memoria tiene una duración de al menos 10.000 ciclos de escritura/borrado.

MEMORIA SRAM

Es una memoria tipo RAM de 1 Kbyte que se utiliza para almacenar datos temporales durante la ejecución, por ejemplo, almacenar el contador de programa mientras se ejecutan interrupciones o llamadas a subrutinas.

MEMORIA EEPROM

El ATmega16A contiene 512 bytes de memoria EEPROM que se utiliza para almacenamiento de datos de larga duración. Esta memoria tiene una duración de al menos 100.000 ciclos de escritura/borrado.

OSCILADOR

El ATmega16A incorpora un oscilador RC interno capaz de generar un reloj interno que puede funcionar a 1 MHz, 2 MHz, 4 MHz y 8 MHz. Además, todos los microcontroladores AVR pueden funcionar con un oscilador externo. Este microcontrolador en concreto permite una frecuencia de reloj de hasta 16 MHz.

PROTOCOLOS DE COMUNICACIÓN

Prácticamente todos los microcontroladores AVR disponen de los protocolos de comunicación SPI, I2C y USART, pero existen algunas versiones que, además de los anteriores, incluyen JTAG, como es el caso del ATmega16, o USB, disponible en los ATmega8U2/16U2/16U4 y en la familia AVR XMEGA.

TEMPORIZADORES/CONTADORES

El ATmega16 dispone de dos temporizadores/contadores de 8 bits, el Timer/Counter0 y el Timer/Counter2, además de otro de 16 bits, el Timer/Counter1. Estos disponen de varios modos de funcionamiento como contador, comparador, temporizador con prescalers independientes y cuatro canales PWM.

CONVERTIDOR ANALÓGICO DIGITAL

El convertidor analógico digital del que dispone el este microcontrolador es de 10 bits y tipo aproximación sucesiva. Está conectado al puerto A mediante un multiplexor de ocho

entradas analógicas. El convertidor permite el funcionamiento de los canales de forma independiente o en pares diferenciales, dos de los cuales disponen de una etapa de ganancia programable.

FUSE Y LOCK BYTES

Los “fuse bits” o bits fusible sirven para poder configurar parámetros o características importantes del microcontrolador, como su memoria, frecuencia de trabajo, interfaz de programación, bootloader, entre otros. Este microcontrolador cuenta con dos fuse bytes (ver Tabla 2 y Tabla 3). Para programar cualquiera de estos bits, se establece en estado lógico bajo. Debido a la importancia que tiene la utilización correcta de estos bytes de configuración, se explica brevemente la función de cada uno de ellos:

- OCDEN habilita la depuración en chip.
- JTAGEN habilita la comunicación JTAG.
- SPIEN habilita la programación y descarga de datos mediante SPI.
- CKOPT habilita un condensador interno de 36 pF entre el pin XTAL1 y GND para configuración de la fuente de reloj.
- EESAVE evita que la memoria EEPROM sea preservada durante el borrado del chip.
- BOOTSZ1:0 establece el tamaño de la sección de memoria dedicada al bootloader.
- BOOTRST configura el vector de reset para saltar a la sección del bootloader.

Fuse High Byte	Bit No.	Description	Default Value
OCDEN[4]	7	Enable OCD	1 (unprogrammed, OCD disabled)
JTAGEN[5]	6	Enable JTAG	0 (programmed, JTAG enabled)
SPIEN[1]	5	Enable SPI Serial Program and Data Downloading	0 (programmed, SPI prog. enabled)
CKOPT[2]	4	Oscillator options	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed, EEPROM not preserved)
BOOTSZ1	2	Select Boot Size (see Table 100 for details)	0 (programmed)[3]
BOOTSZ0	1	Select Boot Size (see Table 100 for details)	0 (programmed)[3]
BOOTRST	0	Select reset vector	1 (unprogrammed)

Tabla 2. Byte fusible alto del microcontrolador ATmega16.

- BODLEVEL configura la tensión mínima para reinicio del microcontrolador en 4 V o 2,7 V.
- BODEN habilita el reinicio del microcontrolador si la tensión de alimentación disminuye de la tensión mínima establecida por BODLEVEL.
- SUT1:0 selecciona el tiempo de inicio del microcontrolador.
- CKSEL3:0 se utilizan para seleccionar la fuente de reloj y el rango de frecuencias que permite cada configuración de estos bits.

Fuse Low Byte	Bit No.	Description	Default Value
BODLEVEL	7	Brown-out Detector trigger level	1 (unprogrammed)
BODEN	6	Brown-out Detector enable	1 (unprogrammed, BOD disabled)
SUT1	5	Select start-up time	1 (unprogrammed)(1)
SUT0	4	Select start-up time	0 (programmed)(1)
CKSEL3	3	Select Clock source	0 (programmed)(2)
CKSEL2	2	Select Clock source	0 (programmed)(2)
CKSEL1	1	Select Clock source	0 (programmed)(2)
CKSEL0	0	Select Clock source	1 (unprogrammed)(2)

Tabla 3. Byte fusible bajo del microcontrolador ATmega16.

Los “lock bits” o bits de bloqueo (ver Tabla 4) sirven para restringir el acceso de lectura o programación de las memorias flash y EEPROM del microcontrolador. Existen dos tipos, bits LB, que se utilizan para bloquear el acceso a la memoria mediante un programador externo y bloquear los fuse bits, y bits BLB, que bloquean el acceso a la misma memoria desde el bootloader o desde el código de la aplicación.

Table 103. Lock Bit Byte(1)

Lock Bit Byte	Bit No.	Description	Default Value
	7	–	1 (unprogrammed)
	6	–	1 (unprogrammed)
BLB12	5	Boot Lock bit	1 (unprogrammed)
BLB11	4	Boot Lock bit	1 (unprogrammed)
BLB02	3	Boot Lock bit	1 (unprogrammed)
BLB01	2	Boot Lock bit	1 (unprogrammed)
LB2	1	Lock bit	1 (unprogrammed)
LB1	0	Lock bit	1 (unprogrammed)

Tabla 4. Bits de bloqueo del microcontrolador ATmega16.

La Imagen 12 muestra la arquitectura del microcontrolador ATmega16 con todos sus periféricos.

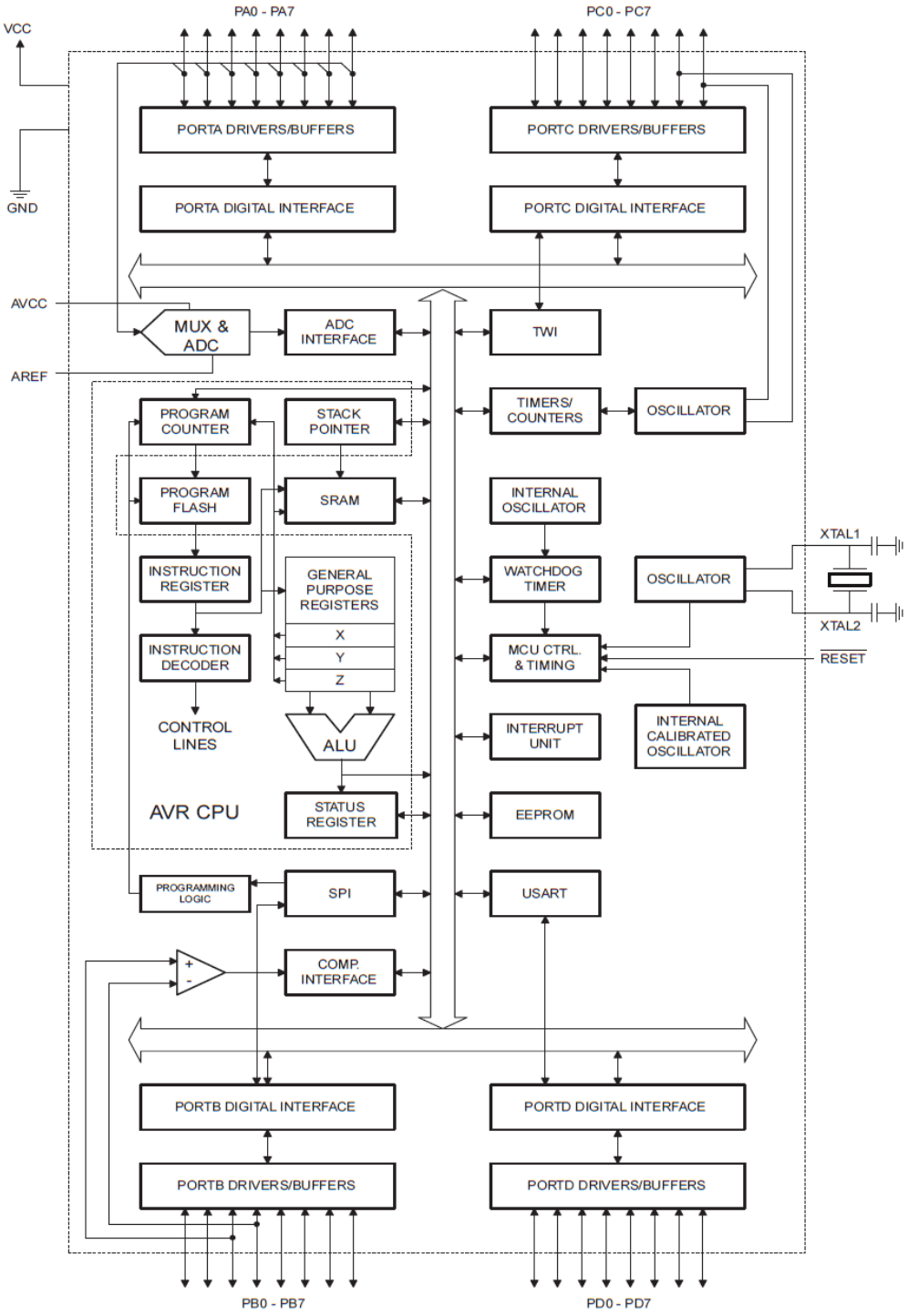


Imagen 12. Arquitectura del microcontrolador ATmega16

Para más información, en los archivos adjuntos de este proyecto se encuentra la hoja de datos del microcontrolador ATmega

6. DISEÑO DEL ESQUEMÁTICO DE LA PLACA DE DESARROLLO

Antes de comenzar con el diseño de la placa es importante plantearse el objetivo o ámbito de uso de la placa, si está destinada al aprendizaje o, por el contrario, para el desarrollo de aplicaciones más complejas.

Además de lo anterior, se debe hacer una relación de características que se desean implementar en la placa, así como, encontrar un programa de diseño adecuado a nuestras necesidades.

6.1. CONSIDERACIONES, CARACTERÍSTICAS Y PROGRAMAS

Si el objetivo es el aprendizaje de la programación de microcontroladores, sería suficiente con algunos interruptores o pulsadores y algunos leds, aunque también se puede incluir, por ejemplo, algún display, entre otros.

Si el objetivo es desarrollar aplicaciones más complejas, se puede incluir lo anterior y seleccionar algunos sensores, pantallas… entre una gran variedad de opciones.

Otra opción de ampliación de una placa de desarrollo es incluir puertos de conexión para distintos protocolos de comunicación, por ejemplo, I2C, USART, etc. además de puertos del microcontrolador para uso genérico. La utilización de puertos es muy interesante debido a que es imposible que una placa de desarrollo sea útil para cualquier aplicación, y por tanto, sea necesario ayudarse de una placa de pruebas o protoboard para aumentar las posibilidades de aplicación.

Cuando se tiene claro el objetivo de la placa, se hace el recuento de las características que se desean implementar, teniendo en cuenta que cuantos más periféricos se incluyan, mayor será el tamaño de la placa y menor su manejabilidad. Por tanto, se debe buscar un equilibrio entre un número aceptable de posibilidades en un tamaño que sea manejable.

Además de los periféricos, la placa debe permitir la comunicación con un PC donde se desarrollarán los programas que se ejecutarán en el microcontrolador. El tipo de comunicación depende del programador que se vaya a utilizar, por lo que este aspecto también es necesario estudiarlo antes de comenzar el diseño.

Igualmente se debe seleccionar la fuente de alimentación, elegir el microcontrolador, y el tipo de encapsulado, como se comentó en una sección anterior.

La función de la placa que se va a diseñar en este proyecto será el de aprendizaje o entrenamiento de programación para microcontroladores AVR ATmega y por ello no se va a incluir ningún tipo de sensor o componentes específicos, sino periféricos polivalentes. Las características a implementar en la placa son:

- Microcontrolador AVR ATmega16
- Programador/depurador JTAG ICE mk1 integrado en la placa.
- La fuente de alimentación será el PC, que proporciona una tensión de 5 V a través del puerto USB.
- Interfaces de comunicación JTAG, SPI, I2C y USART.
- Puerto B del microcontrolador de propósito general y para programación mediante protocolo SPI.
- Control PWM para led y control en frecuencia para zumbador.
- LCD 16 x 2, display de 7 segmentos, tres leds y un zumbador como periféricos de salida.
- Teclado numérico, dos interruptores, dos pulsadores y un potenciómetro como periféricos de entrada.

Una vez que se tienen claras todas las características que se desean incluir en la placa de desarrollo, el siguiente paso es realizar un esquemático. Existen muchos programas que permiten realizar esquemáticos y diseñar la PCB. A continuación, se mencionan algunos de ellos:

- ALTIUM: es un software que cubre todas las etapas de desarrollo, desde el diseño del esquemático, simulación de circuitos, diseño de PCB, análisis de integridad de señales, … hasta el prototipo final. Es un software bastante costoso y generalmente lo utilizan empresas dedicadas al diseño electrónico.
- EASYEDA: es una herramienta de simulación de circuitos y diseño de PCB, de uso completamente online y gratuito, sin necesidad de instalación de cualquier software.
- KICAD: un software libre multiplataforma para diseño esquemáticos y PCB.
- D S PCB: software libre creado por RS que cuenta con utilidades para recuperar símbolos de esquemáticos y huellas para diversos componentes que RS distribuye.
- AUTODESK EAGLE: software multiplataforma, fácil de usar, con una amplia variedad de bibliotecas de componentes y piezas lista para usar.

Para el diseño de la placa de este proyecto se ha utilizado Autodesk Eagle.

6.2. MÓDULO PROGRAMADOR / DEPURADOR JTAG ICE mk1

JTAG ICE mk1 es un dispositivo de hardware libre, basado en un microcontrolador ATmega16. El hardware de este dispositivo se puede dividir en cinco módulos:

- Módulo de comunicación: utiliza un puerto RS232 para la comunicación con el PC.
- Módulo de control: se trata de un microcontrolador ATmega16 que se encarga de convertir las instrucciones del entorno de desarrollo en instrucciones válidas para la interfaz AVR JTAG, además de supervisar y establecer una serie de señales de control.
- Modulo convertidor de nivel: garantiza una comunicación exitosa con placas que funcionan a voltajes diferentes, adaptando los niveles de tensión al rango permitido por el controlador.
- Adaptador JTAG: es el puerto de conexión, basado en un conector IDC de 10 pines.
- Módulo de potencia: se encarga de adaptar el nivel de tensión de una fuente de alimentación externa al rango de tensión admitido por el dispositivo.

En la Imagen 13 se observa un diagrama de bloques de este dispositivo, obtenido del manual de usuario [15].

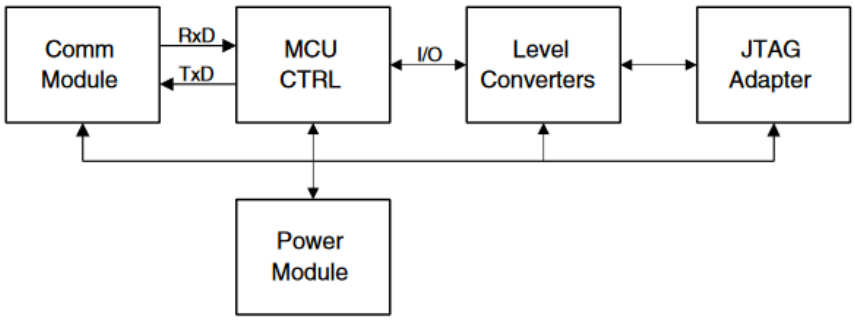


Imagen 13. Esquema de bloques del JTAG-ICE mk1.

Para integrar este dispositivo en la placa que se va a diseñar, es necesario realizar las siguientes modificaciones respecto al hardware de serie:

- El módulo de comunicación será sustituido por un módulo FTDI FT232 que se analizará en detalle en el siguiente apartado.
- El módulo de potencia será eliminado porque el dispositivo se alimentará desde el PC a través del módulo FTDI. Este módulo se explicará en el siguiente apartado.
- El módulo convertidor de nivel será eliminado porque este dispositivo utilizará el mismo nivel de tensión que el microcontrolador objetivo de programación.
- Además del microcontrolador, se necesita una serie de componentes externos, como son:
- Dos condensadores de 100 nF y uno de 4,7 µF para una tensión de alimentación estable, resistencia de 10 KΩ Pull-Up para el pin Reset.
- Cristal de cuarzo de 7,3728 MHz con dos condensadores de 22 pF.
- Led blanco de 3 mm con resistencia limitadora de 10 KΩ para visualizar el estado de la comunicación.
- Interfaz JTAG de 10 pines (ver Imagen 14) para programador JTAG externo.
- Jumper para la alimentación del microcontrolador. Cuando se utilice un programador JTAG externo se desactivará la alimentación mediante el jumper para evitar el conflicto entre ambos programadores.

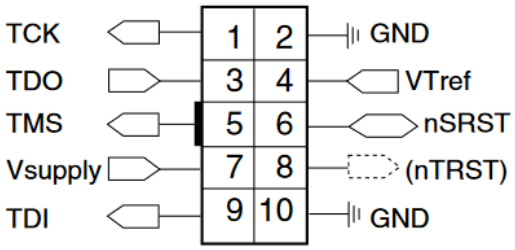


Imagen 14. Interfaz JTAG.

El esquemático de módulo programador y el módulo FTDI realizado con el programa Autodesk Eagle se observa en la Imagen 15.

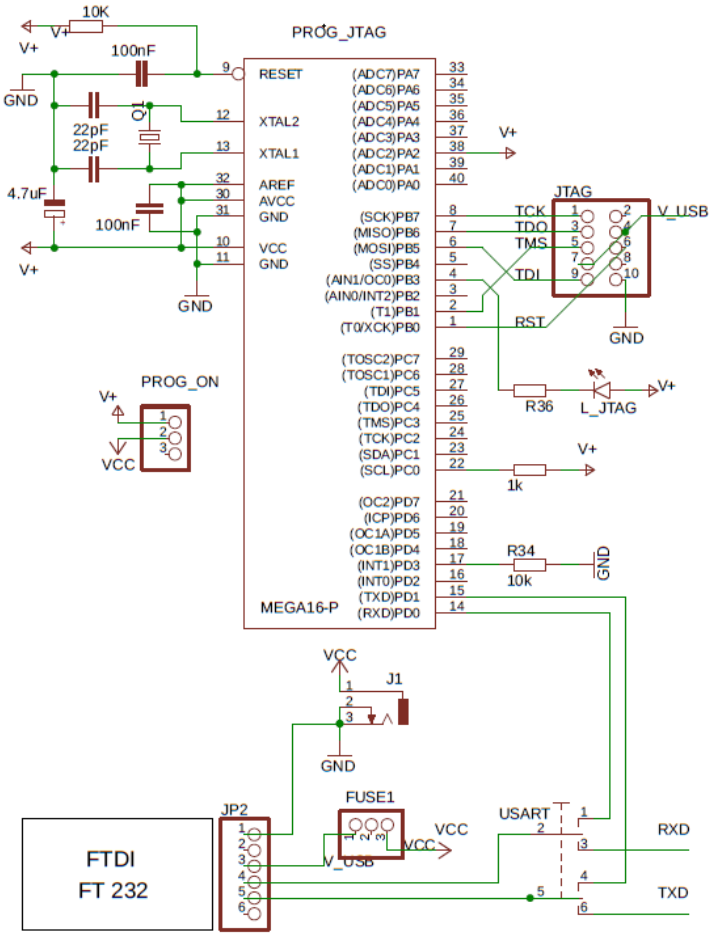


Imagen 15. Esquemático del programador/depurador JTAG-ICE mk1.

6.3. MÓDULO FTDI FT232

El FT232 es un conversor USB a serial UART que permite la comunicación entre un PC, a través del puerto USB, y un dispositivo externo mediante el protocolo UART [16].

Se necesita la instalación de un controlador en el PC para que este reconozca al conversor. El controlador es proporcionado por la marca FTDI, y se puede descargar desde la página web oficial.

Se puede adquirir en el mercado ya montado en una PCB (ver Imagen 16) con un conector mini USB, pines para la comunicación con el dispositivo externo, alimentación y masa. Además, dispone de un jumper para selección del nivel de voltaje de comunicación que requiera el dispositivo externo, 3,3 V o 5 V.

La placa incluye un fusible de 500 mA para protección del puerto USB del PC. Teniendo en cuenta que el consumo típico del chip FT232 es de 15 mA, esta placa es capaz de alimentar a dispositivos externos con un consumo inferior a 485 mA.

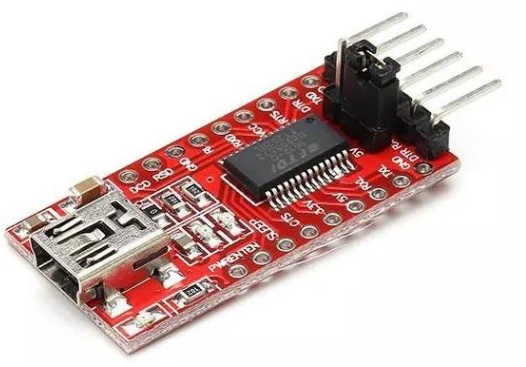


Imagen 16. Módulo conversor USB UART basado en chip FTDI FT232RL.

El módulo FTDI se comunicará con el programador mediante la interfaz USART, situada en los pines 14 y 15, RXD y TXD respectivamente (ver Imagen 17). Además, el módulo FTDI, también podrá comunicarse con el microcontrolador principal de la misma forma que lo hace con el programador, ya que ambos son microcontroladores ATmega16. Para ello es necesario añadir un interruptor de dos vías que selecciona entre el programador y el microcontrolador principal, como se puede observar en la parte inferior de la Imagen 15.

6.4. MÓDULO MICROCONTROLADOR ATMEGA16

Este módulo estará formado por el microcontrolador principal y los siguientes periféricos y componentes discretos:

- Display LCD de 2 filas y 16 caracteres por fila con un interruptor para alimentación y un potenciómetro para regulación del contraste.
- Entrada analógica mediante un potenciómetro.
- Dos pulsadores para interrupciones externas y dos interruptores. Tanto los pulsadores como los interruptores contarán con una red RC para reducir el efecto rebote. Este aspecto se explicará más adelante.
- Tres leds blancos de 5 mm con resistencias limitadoras de 1 KΩ
- Un piezoeléctrico a modo de zumbador para generar sonidos, con resistencia limitadora de 120 Ω.
- Pulsador para resetear el microcontrolador.

- Un condensador de 100 nF y otro de 4,7 µF para una tensión de alimentación estable y una resistencia de 10 KΩ Pull-Up para el pin Reset.
- Pulsador para resetear el microcontrolador.
- Cuatro condensadores de 2,2 µF y tres resistencias de 390 Ω formarán las redes RC de los pulsadores e interruptores para eliminar el efecto rebote.
- Tira de 11 pines tipo hembra para uso de propósito general del puerto B, pin reset, alimentación y masa. Esto, además, permitirá la programación del microcontrolador mediante protocolo SPI.

Los componentes principales de entre los arriba indicados serán tratados más detalladamente en el capítulo 11, en el que se ilustra su funcionamiento con un fragmento de código que muestra las funcionalidades más importantes de dichos dispositivos.

La distribución de los periféricos en los distintos puertos del microcontrolador se hará según la

Tabla 5, que se ha elaborado teniendo en cuenta el pinout del microcontrolador (ver Imagen 17).

Puerto	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
A	Display LCD							Entrada analógica
B	Interfaz SPI					INT2, pulsador	Interruptores	
	Puerto de propósito general							
C			Interfaz JTAG				Interfaz I2C	
D	Led	Led	Zumbador	Led	INT1, pulsador	INT0, Teclado	Interfaz USART	

Tabla 5. Distribución de las funciones en los puertos del microcontrolador ATmega16.

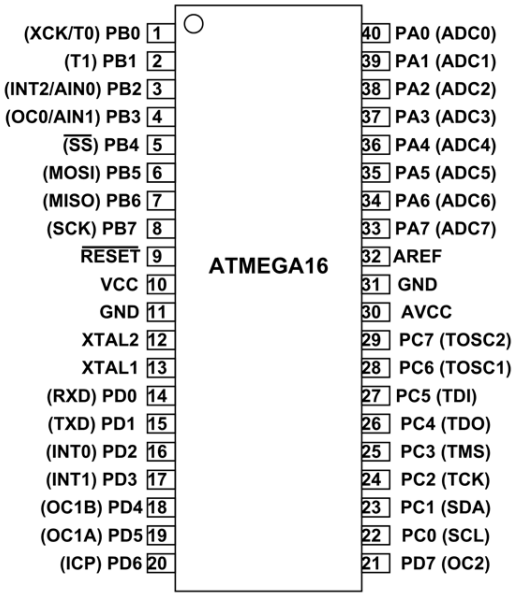


Imagen 17. Pinout del microcontrolador ATmega16.

El microcontrolador ATmega16 estará montado sobre un zócalo ZIF (Zero Insertion Force) que permitirá una colocación y extracción fácil y rápida del microcontrolador en la placa de desarrollo.

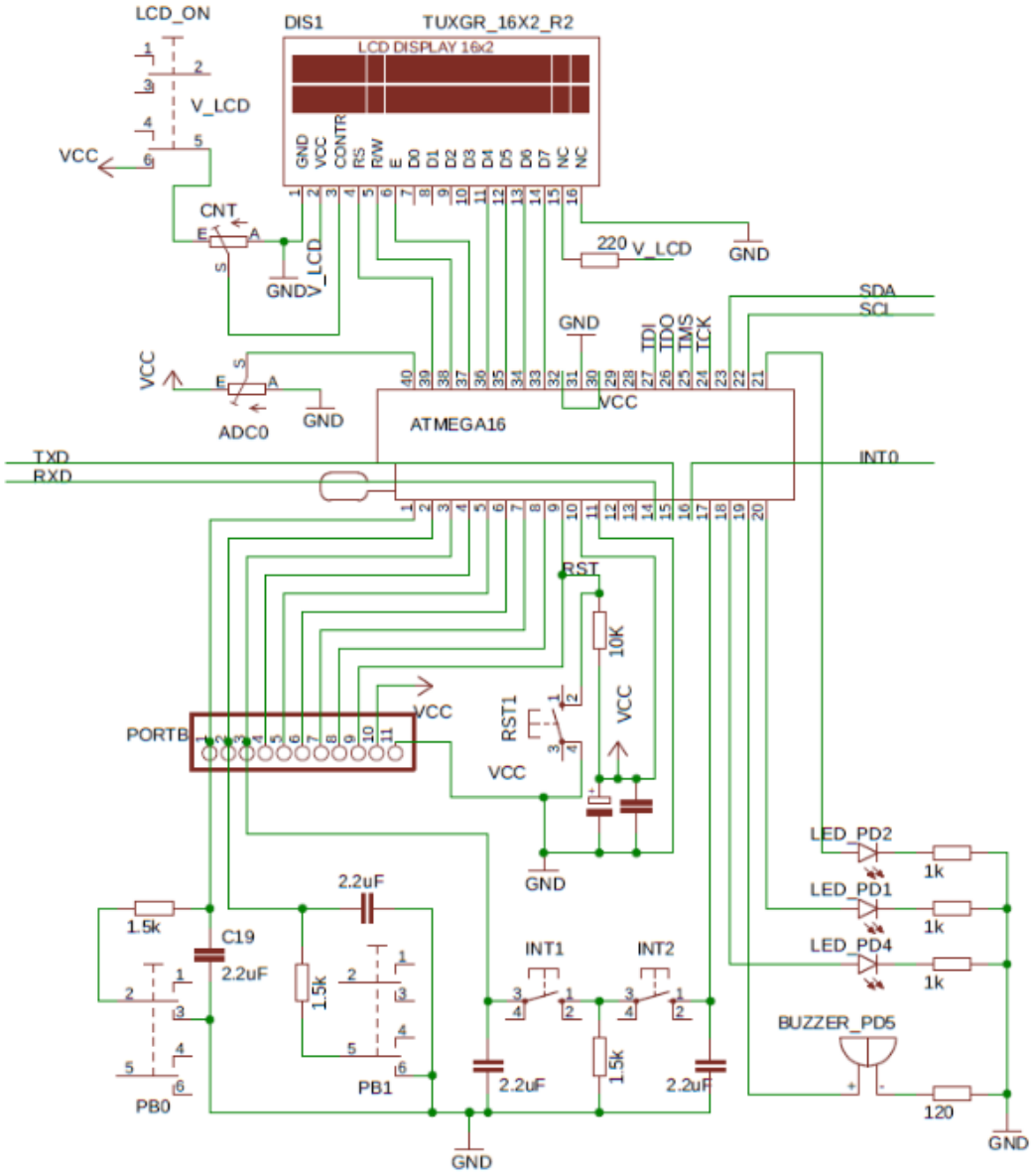


Imagen 18. Esquemático del módulo microcontrolador y algunos periféricos.

6.4.1. EFECTO REBOTE DE UN PULSADOR O INTERRUPTOR

Cuando se acciona o deja de accionarse un interruptor o un pulsador se producen unos rebotes. Estos rebotes provocan un efecto transitorio compuesto por una secuencia de activaciones y desactivaciones que pueden ser detectadas por el microcontrolador y provocar un funcionamiento aleatorio de las acciones programadas por dichos interruptores o pulsadores. Este efecto se puede observar en la Imagen 19, capturada con un osciloscopio.

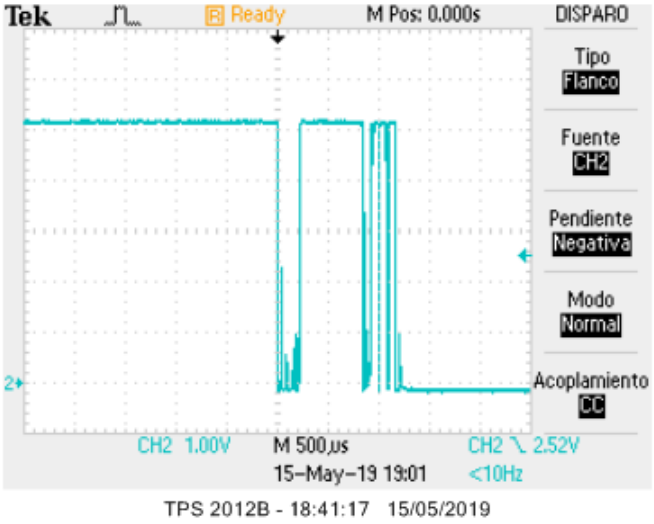


Imagen 19. Captura de los rebotes de un pulsador con osciloscopio.

El efecto rebote se puede eliminar mediante una red RC, compuesta por un condensador y una resistencia. Los valores de estos componentes se calcularán teniendo en cuenta la duración de los rebotes más un tiempo prudencial para asegurar el fin del transitorio. En este caso se calculará la red RC para un tiempo de 4 ms.

La constante de tiempo de una red RC es el producto de los valores de la resistencia y el condensador:

$$\tau = R \cdot C$$

Teóricamente, el tiempo de carga del condensador es infinito, pero en la práctica se considera que alcanza una carga superior al 99 % transcurrido un tiempo aproximado de cinco veces la constante de tiempo. Si se fija un valor de 2,2 µF para el condensador, se puede calcular fácilmente el valor de la resistencia:

$$5 \cdot \tau = 5 \cdot R \cdot C = 4\,ms \quad \rightarrow \quad R = \frac{4\,ms}{5 \cdot 2{,}2\,\mu F} \approx 364\,\Omega$$

Como el valor calculado para la resistencia no lo tenemos disponible, se utilizará el valor inmediatamente superior, 390 Ω.

En la Imagen 20 se muestra el transitorio del mismo pulsador, pero esta vez, incluyendo la red RC calculada.

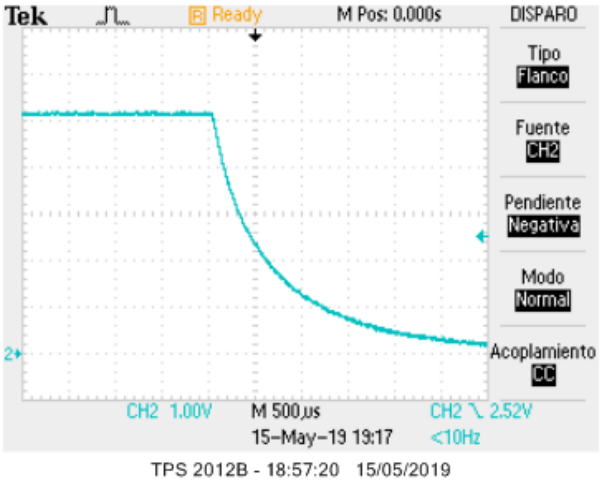


Imagen 20.Captura del contacto de un pulsador con red RC con osciloscopio.

6.5. MÓDULO EXPANSOR DE ENTRADAS / SALIDAS MEDIANTE I2C

Este módulo estará compuesto por dos módulos de expansión de puertos, un teclado, un display de 7 segmentos y un conector de 4 pines de interfaz I2C para dispositivos externos.

Se utilizarán dos circuitos integrados PCF8574P, con zócalos y encapsulados tipo PDIP, para aumentar en dos el número de puertos del microcontrolador. Estos dos puertos añadidos se controlarán mediante protocolo I2C y se utilizarán como puerto de entrada para el teclado y puerto de salida para display de 7 segmentos.

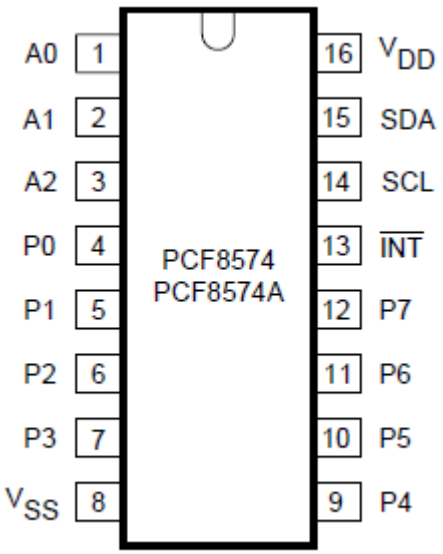


Imagen 21. Pinout del circuito integrado PCF8574.

Este tipo de circuitos integrados disponen de un pin de interrupción que se activa cuando se detecta un cambio en el puerto configurado como entrada. Esta opción se utilizará en el módulo del teclado y se conectará a la interrupción externa INT0 del microcontrolador para detectar las pulsaciones en las teclas.

Para evitar que los módulos I2C permanezcan encendidos cuando no se estén utilizando se incluirá un interruptor para la alimentación de dichos módulos y un led blanco de 3 mm como indicador de alimentación.

El display de 7 segmentos irá acompañado de las resistencias limitadoras de 1 KΩ correspondientes a cada segmento.

Además, se añadirán dos resistencias Pull-Up de 4,7 KΩ a las líneas SDA y SCL de la interfaz I2C.

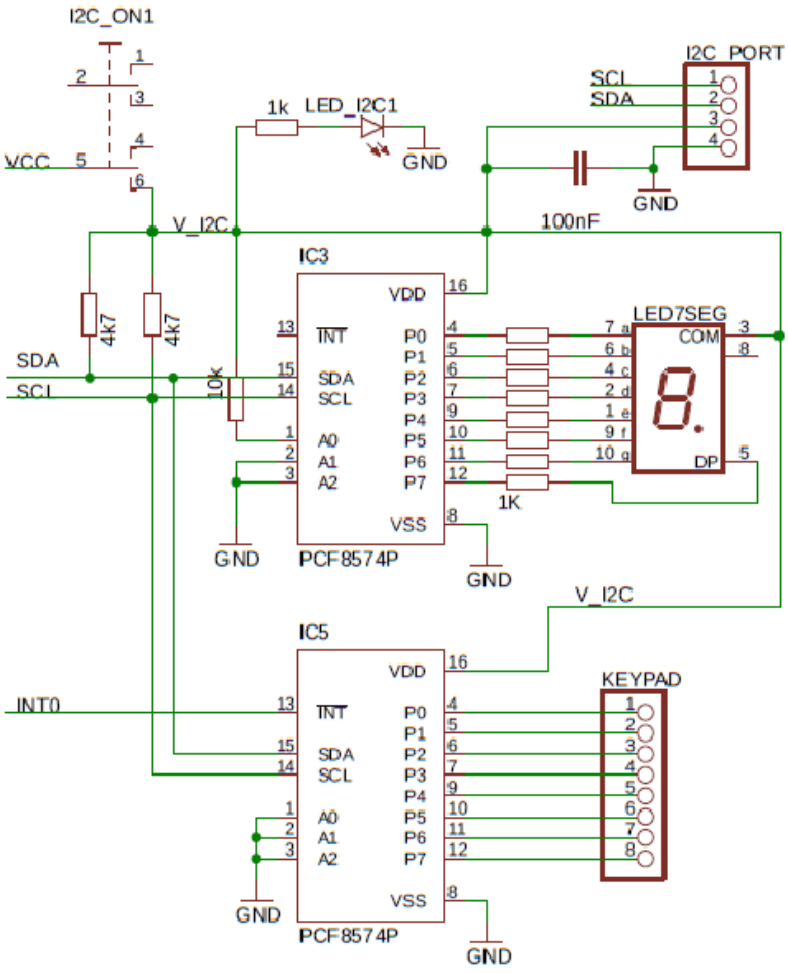


Imagen 22. Esquemático de módulo I2C con periféricos

7. DISEÑO DE PCB

Después de realizar el esquemático es conveniente montar cada una de las partes del esquema por separado en una placa de pruebas para comprobar el funcionamiento de cada uno de los módulos.

En Autodesk Eagle, el esquemático se realiza en un subprograma, al igual que el diseño de la PCB se realiza en otro subprograma. Ambos subprogramas están enlazados, de forma que los cambios en los componentes realizados en un subprograma, se actualizan automáticamente en el otro.

Cuando se inicia un nuevo diseño de PCB a partir del esquemático, aparecen todos los componentes unidos por las conexiones eléctricas, como se observa en la Imagen 23.

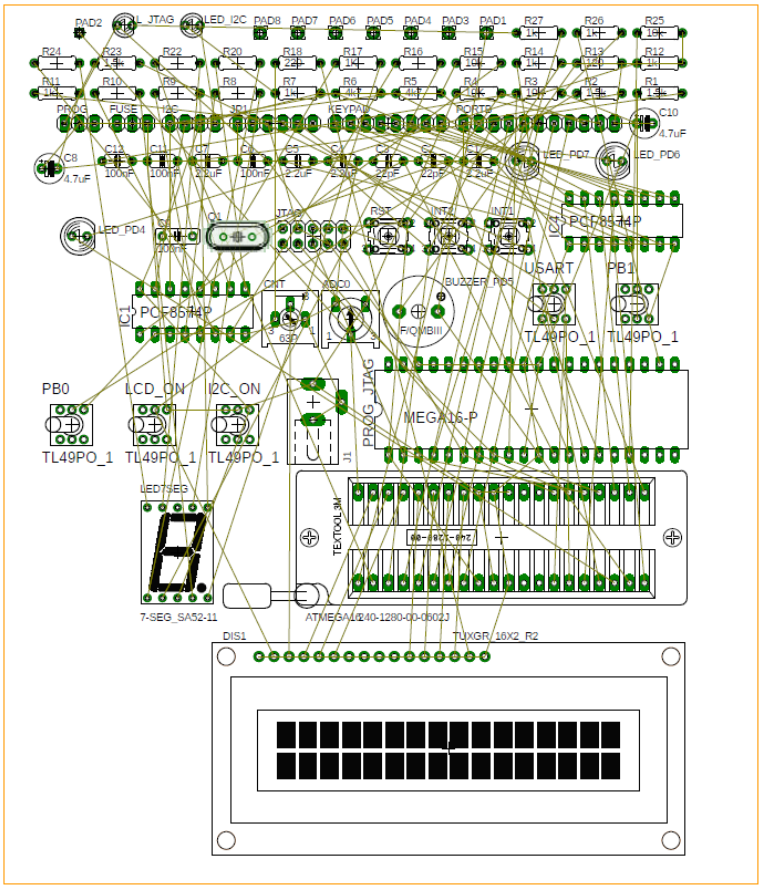


Imagen 23. Captura del inicio del diseño de la PCB.

A continuación, se estudia la colocación aproximada de los componentes teniendo en cuenta aspectos como:

- La posición de los pulsadores o interruptores debe tener una cierta separación entre ellos para un accionamiento cómodo.

- La posición del puerto de alimentación y conexión con el PC debe estar en un extremo de la placa.
- La posición de los leds y los displays debe tener buena visibilidad, sin riesgo de ser cubiertos mientras se accionan las entradas.
- Colocar el microcontrolador con una orientación que permita el acceso fácil a la mayor parte de los periféricos para facilitar el posterior ruteado de la PCB.

Antes de comenzar con trazado de las pistas, es necesario tener en cuenta las limitaciones de fabricación. Cada fábrica de PCBs impone estas limitaciones que se deben a la precisión de la maquinaria, entre otros. Por ejemplo, anchura y separación mínimas de pistas, diámetro mínimo y máximo de vías, etc. Un ejemplo de algunas de estas restricciones se muestra en la Imagen 25, que corresponde a la fábrica a la cual se encargará la fabricación de la placa diseñada en este proyecto.

Los programas para diseño de PCBs deben tener una herramienta para introducir estos parámetros. En el programa que se ha utilizado para diseñar la placa de este proyecto, Autodesk Eagle 9, esta herramienta se denomina DRC (ver Imagen 24). Si se configura antes de comenzar el enrutado, la propia herramienta impide el trazado de pistas que incumplan los parámetros, y si se configura con posterioridad, indicará un error por cada zona de conflicto.

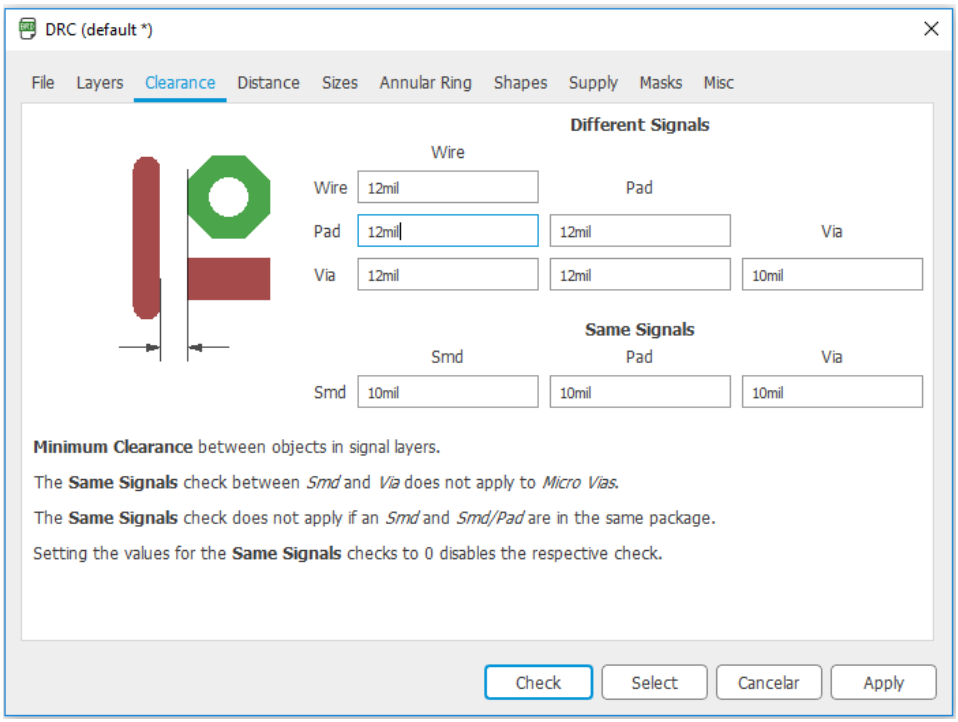


Imagen 24. Establecimiento de reglas de diseño para PCB en Autodesk Eagle.

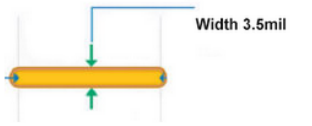
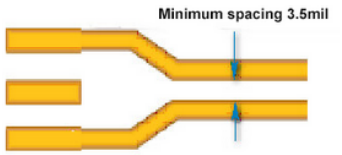
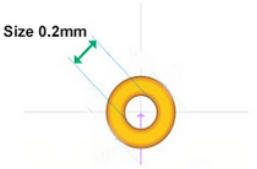
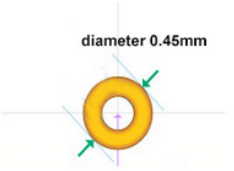
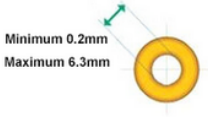
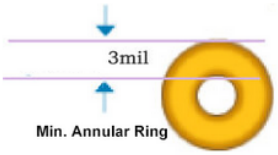
Min. Trace	3.5mil	For Single&Double Layer PCB, the minimum trace width is 5 mil; For Multi Layer PCB, the minimum trace width is 3.5mil.	
Min. Spacing	3.5mil	For Single&Double Layer PCB, the minimum spacing is 5 mil; For Multi Layer PCB, the minimum spacing is 3.5mil.	
Min. Via hole size	0.2mm	For Single&Double Layer PCB, the minimum via hole size is 0.3mm; For Multi Layer PCB, the minimum via hole size is 0.2mm.	
Min. Via diameter	0.45mm	For Single&Double Layer PCB, the minimum Via diameter is 0.6mm; For Multi Layer PCB, the minimum via diameter is 0.45mm.	
Via To Trace	≥5mil	Minimum distance between via(plated holes) and trace is 5mil.	
Drill Hole Size	0.2--6.3mm	Min. drill size is 0.2mm, Max. drill size is 6.3mm.	
Hole Size Tolerance	±0.08mm	e.g. For the 0.6mm hole, the finished hole size between 0.52mm to 0.68mm is acceptable.	
Annular Ring	≥3mil	Annular ring surrounded by traces should be equal to or larger than 3mil	

Imagen 25. Capacidades de fabricación de PCBs de JLCPCB.

Para llevar a cabo el ruteado de la placa, se trazarán las pistas intentando seguir un orden de menor a mayor longitud. En placas en las que se utilizarán componentes de orificio pasante, el trazado de pistas se realiza en la capa “botom” (cara inferior de la placa), excepto en las situaciones en que es inevitable el cruce de pistas, que también se pueden trazar en la capa “top” (cara superior de la placa), teniendo en cuenta que los componentes se soldarán en la cara

inferior. Cuando se utilizan componentes de montaje superficial, las pistas se trazan en la capa “top”, misma cara de montaje, aunque cuando se trata de diseños complejos, se pueden montar componentes en ambas caras.

En la Imagen 26 se muestra una captura del diseño de la placa con el trazado de pistas completado

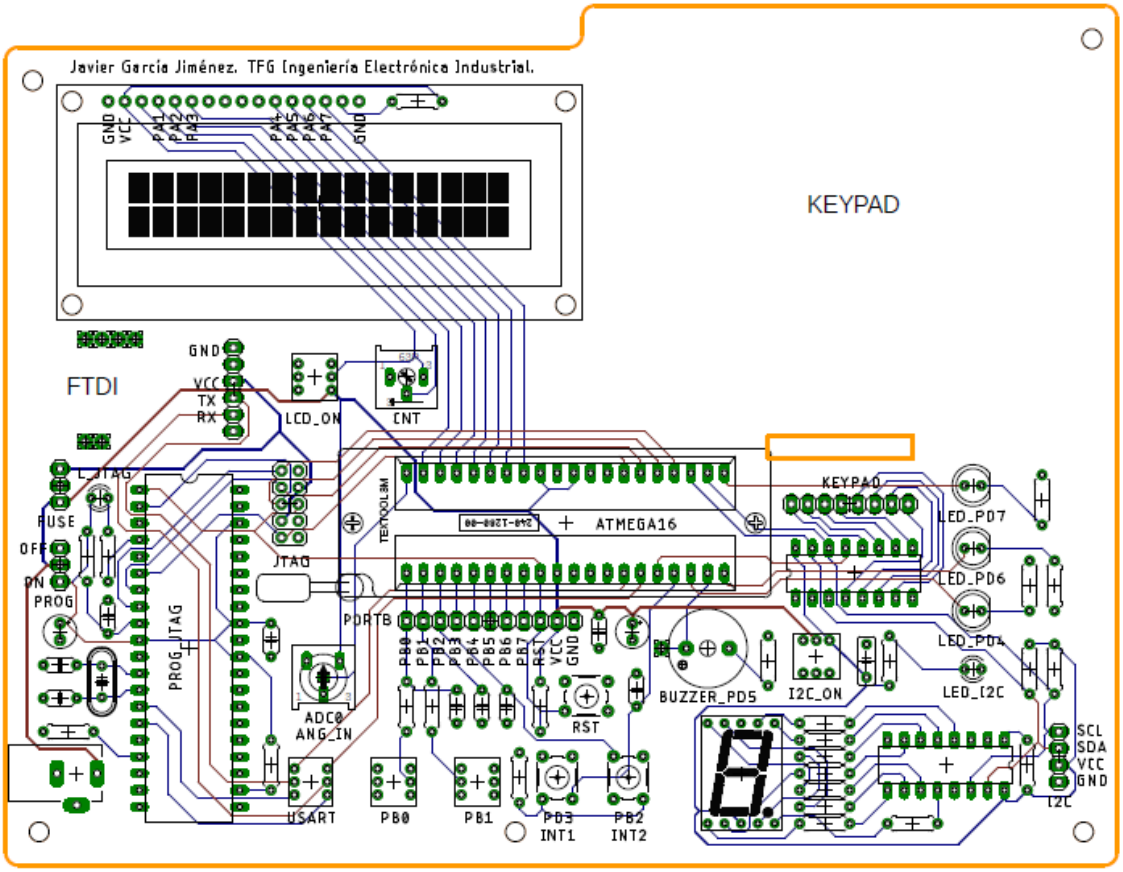


Imagen 26. Captura del diseño de PCBs con trazado de pistas completado.

Una vez finalizado el diseño, el programa incorpora una herramienta que genera unos archivos con las distintas capas del diseño. Estos se denominan archivos “Gerber” y son los que necesita la fábrica de PCBs para la fabricación de la placa.

Algunos programas disponen de un visualizador que muestra una imagen de la placa para ofrecer una idea del aspecto de esta una vez fabricada, ver Imagen 27.

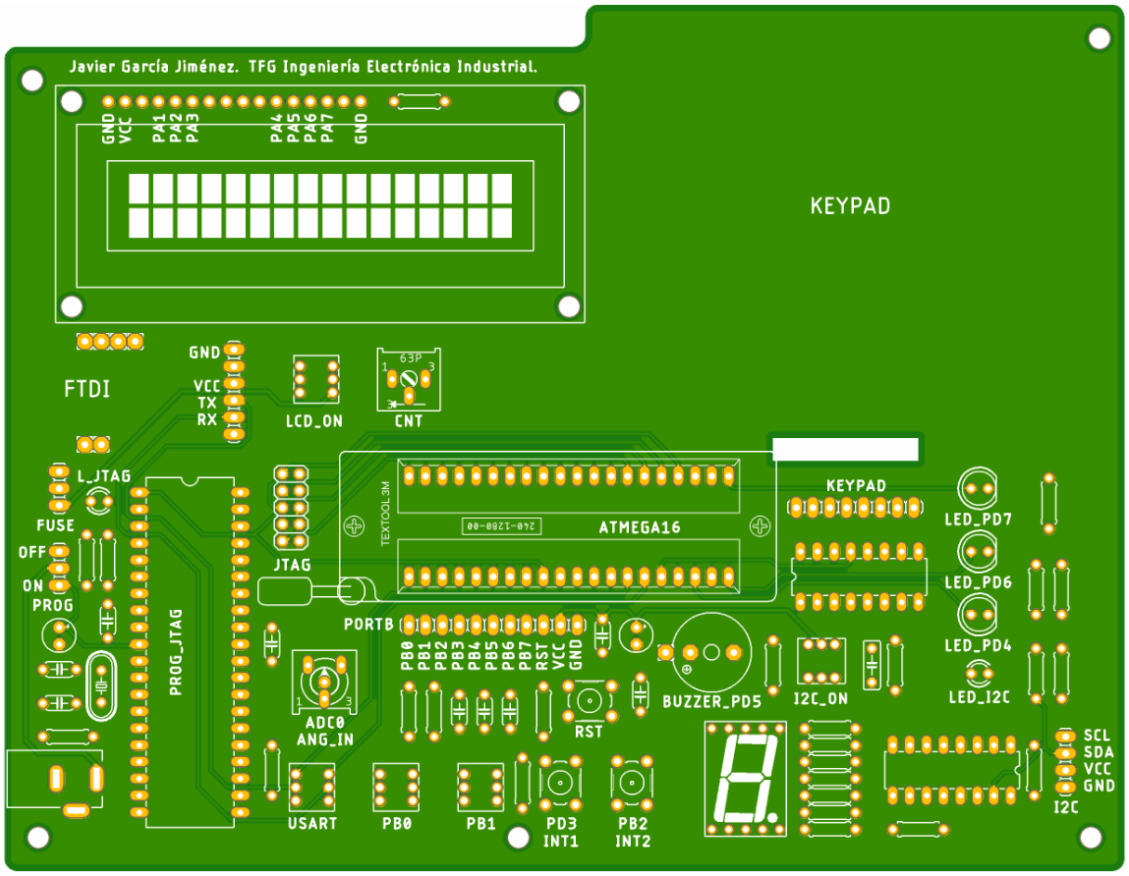


Imagen 27. Vista del diseño generado por el visualizador de Autodesk Eagle.

8. ENCAPSULADOS DE COMPONENTES ELECTRÓNICOS

El encapsulado de un componente electrónicos tiene la misión de proteger al componente contra golpes, vibraciones, polvo, humedad… que en resumen se trata de aislar al componente respecto a su entorno, y además, permitir su instalación, ya sea de forma manual o automatizada.

En este capítulo se hará una introducción a los componentes electrónicos que se utilizarán en la fabricación de la placa objetivo de diseño de este proyecto, los diferentes tipos de montaje y algunos tipos de encapsulado de circuitos integrados utilizados en microcontroladores, así como sus posibilidades de soldadura en laboratorio.

8.1. COMPONENTES DE TIPO DE MONTAJE EN ORIFICIO PASANTE

La tecnología de montaje de este tipo de componentes se conoce como THT del inglés *Through-Hole Technology* y a los componentes como THD del inglés *Through-Hole Device*. Estos componentes se caracterizan por la forma de sus contactos, denominados “pines” o “patillas”, de sección circular, cuadrada o rectangular, generalmente inferior a 1 mm de lado o diámetro, y cuya longitud varía aproximadamente entre 3 y 30 mm. Están fabricados de aleación dúctil, que permite a estos adaptarse a la separación de los orificios de montaje en la PCB.

Los componentes más representativos de este tipo de montaje son las resistencias y los condensadores electrolíticos y cerámicos. Normalmente tienen forma cilíndrica alargada, en cuyos extremos se disponen los pines de conexión. Existen dos tipos, forma axial, con un pin en cada extremo del cilindro, y forma radial, con los dos pines en el mismo extremo del cilindro.

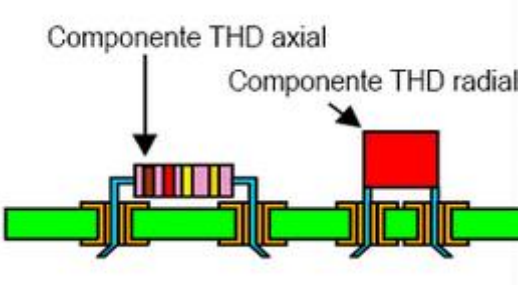


Imagen 28. Tecnología THT. Montaje de componentes THD.

Esta técnica de montaje suele reservarse para componentes voluminosos o pesados, como condensadores electrolíticos, inductores, o semiconductores de potencia que debido al tamaño son capaces de disipar mayor energía.

Este tipo de componentes están en desuso para diseños de producción en masa debido a la dificultad en el uso con maquinaria de colocación automatizada, pero en diseño de prototipos de laboratorio que no requieran alta frecuencia son ideales por la facilidad de colocación y soldadura manual. También se utilizan en placas de potencia, en las que el mayor tamaño de los componentes es un problema menor respecto a las ventajas de una mayor disipación térmica, permitiendo, además, la colocación de disipadores para controlar mayores potencias.

Otro inconveniente de este tipo de componentes es el encarecimiento de la fabricación de placas debido a un mayor tamaño de estas y a la realización de los orificios.

8.2. COMPONENTES DE TIPO DE MONTAJE SUPERFICIAL

La tecnología de montaje de este tipo de componentes se conoce como SMT del inglés *Surface-Mount Technology* y a los componentes como SMD del inglés *Surface-Mount Device.*

Se trata de componentes miniaturizados y rediseñados de sus equivalentes de montaje en orificio pasante para ser manejados por máquinas en lugar de hacerlo manualmente.

Los pines o patillas disminuyen en longitud en el caso de circuitos integrados, o se ven sustituidos por placas de conexión en el caso de resistencias, condensadores, diodos... ya que serán soldados en la misma cara de la placa donde se sitúa el componente.

El proceso de soldadura de los componentes requiere que la PCB disponga de una almohadilla de soldadura, de cobre o estaño sin orificio, para cada conexión del componente, y sobre la que se aplica una capa de pasta de soldadura mediante una plantilla de acero o níquel y un proceso de serigrafía o mediante un mecanismo de impresión a chorro, seguido de la colocación de los componentes sobre la PCB en una máquina de colocación. A continuación, las placas se transportan al horno de soldadura por reflujo, el cual dispone de una primera zona de precalentamiento gradual y uniforme, y una segunda zona de mayor temperatura que funde la pasta de soldadura. Finalmente, la placa puede ser sometida a un proceso de limpieza de residuos.

Una vez concluida la etapa de soldadura, es necesario la inspección visual para detectar componentes faltantes o desalineados y puentes de soldadura. Además de esta inspección, la placa puede someterse a un proceso de prueba.

Este tipo de montaje permite la instalación de estos componentes en las dos caras de una PCB, reduciendo el tamaño de esta y logrando una mayor densidad de circuitos. Además, se puede combinar con el montaje en orificio pasante, resultando las denominadas placas híbridas que aprovechan las ventajas de ambos tipos de montaje.

Debido a la complicación del proceso de colocación y soldadura de los componentes, este tipo de montaje solo se utiliza para producción en masa o laboratorios avanzados que dispongan las máquinas necesarias, como es el caso del diseño de prototipos que requieren alta frecuencia de funcionamiento, ya que el montaje superficial es más inmune a las capacidades o inducciones parásitas que el montaje en orificio pasante.

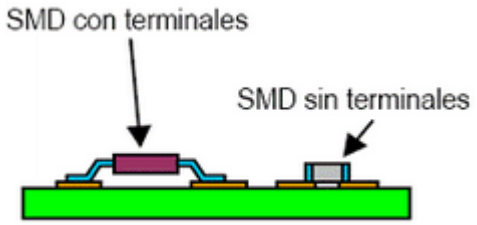


Imagen 29. Tecnología SMT. Montaje de componentes SMD.

8.3. ENCAPSULADOS BÁSICOS DE CIRCUITOS INTEGRADOS

Un encapsulado es un recubrimiento de material aislante que protege al circuito integrado y que permite la comunicación con el exterior mediante unos contactos metálicos. El material puede ser cerámico (carburo de silicio, berilia y óxido de aluminio) o plástico (dióxido de silicio, poliamidas y epoxi).

El encapsulado es el resultado de un proceso con tres partes fundamentales:

Circuito integrado o pastilla de silicio.

Hilos conductores: son las uniones entre las pistas del circuito integrado y los contactos metálicos del encapsulado final. Estos hilos han de tener ciertas características, como baja resistencia eléctrica, baja inducción, fáciles de soldar sobre la pastilla de silicio, adecuada dilatación térmica, etc.

Base: es el recubrimiento de las dos partes anteriores, cuya función es dar soporte al circuito, protección frente al entorno y disipar el calor generado en el circuito integrado, por lo que ha de tener una alta conductividad térmica y un gran poder de aislamiento.

Existe una gran variedad de tipos de encapsulado. A continuación, se explicarán brevemente las características de los tipos de encapsulado disponibles para el microcontrolador que se utilizará en este proyecto:

PDIP (Plactic Dual In-line Package): El encapsulado de doble línea de circuitos integrados es el más común. Está formado por dos filas paralelas de pines de orificio pasante. Cada uno de los pines de una misma fila están separados por una distancia de 0.1” que corresponde a 2.54 mm y es la separación estándar en el montaje de circuitos en placas protoboard. Cuando se montan en placas PCB, se insertan por un lado de esta y son soldados por el otro lado, lo que imposibilita la eliminación del componente para sustitución. Para solucionar este inconveniente se recurre a la instalación de un zócalo del mismo tamaño y distribución de patillaje que el circuito integrado. Las ventajas de este tipo de encapsulado son la facilidad de soldadura de forma manual o una buena disipación térmica debido al tamaño, lo que repercute en unas mayores dimensiones de la placa PCB. Otro inconveniente es la limitación de frecuencia de reloj, ya que esta señal se ve afectada por la inductancia que se crea en los pines debido a la longitud de estos, pudiendo verse agravado este problema con la utilización de un zócalo.

Imagen 30. Encapsulado PDIP 40.

SOIC (Small Outline Integrated Circuit): Este encapsulado es similar al PDIP en la distribución de los pines, pero estos son de tipo ala de gaviota y, por tanto, de montaje superficial. La separación entre los pines es de 0,05” que corresponde a 1,27 mm. Con esta

separación entre pines, el tamaño del componente es la mitad que el equivalente con encapsulado DIP. El número de pines suele estar entre ocho y treinta y dos. Este tipo de encapsulado

Imagen 31. Encapsulado SOIC 20.

QFP (Quad Flat Package): Es un encapsulado de montaje superficial normalmente cuadrado, aunque existen variantes rectangulares. Los pines de conexión son de tipo ala de gaviota. La separación entre los pines puede variar entre 0,4 y 1 mm. Existen algunas variantes de este encapsulado cuya diferencia principal es el espesor, variando entre 1 mm para el TQFP, que es la versión más delgada, y 3.8 mm para el PQFP. El tamaño común varía entre 7x7 mm y 40x40 mm dependiendo del número total de pines, que suele oscilar entre 8 y 304. Este tipo de encapsulado se puede soldar manualmente, aunque es complicado y se necesita experiencia.

Imagen 32. Encapsulado TQFP 44.

QFN (Quad Flat No Leads): Este tipo de encapsulado de montaje superficial generalmente tiene forma cuadrada, aunque también puede ser rectangular. En lugar de pines utiliza almohadillas metálicas en la periferia de la cara inferior. Estas almohadillas pueden ocupar solo el borde de la cara, o, además, una parte del borde lateral. En el área central de la cara inferior suele haber una almohadilla térmica para mejorar la disipación de calor. El número de almohadillas varía de 8 a 68, con una separación entre ellas de 0,4 o 0,5 mm. Las dimensiones exteriores típicas varían entre 3 y 10 mm de lado y un espesor máximo de 1 mm, aunque existe la variante TQFN con un espesor máximo de 0,8 mm. Debido a la disposición de las almohadillas, la labor de soldadura manual o inspección, es muy complicada o imposible, por lo que este tipo de encapsulados se utilizan en fabricación industrial. La ventaja de este tipo se debe a que la ausencia de pines en las conexiones, lo que proporciona un mayor rendimiento eléctrico y permite trabajar con frecuencias de reloj más altas que en los anteriores encapsulados.

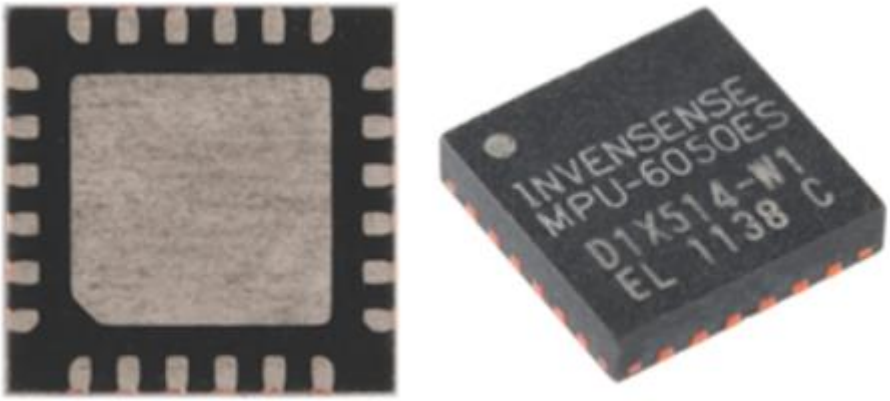


Imagen 33. Encapsulado QFN 24.

BGA (Ball Grid Array): Son encapsulados muy pequeños generalmente cuadrados. Tienen las conexiones en la cara inferior y estas constan de pequeñas bolitas de estaño dispuestas en forma de matriz, con una separación entre ellas de 1 y 1,5 mm. Debido a esta distribución de las conexiones se reduce el tamaño del componente. El inconveniente principal es el acceso a las conexiones para soldadura o inspección de esta, que se debe realizar mediante rayos X. Debido a estos inconvenientes este tipo de encapsulado solo se utiliza en fabricación industrial.

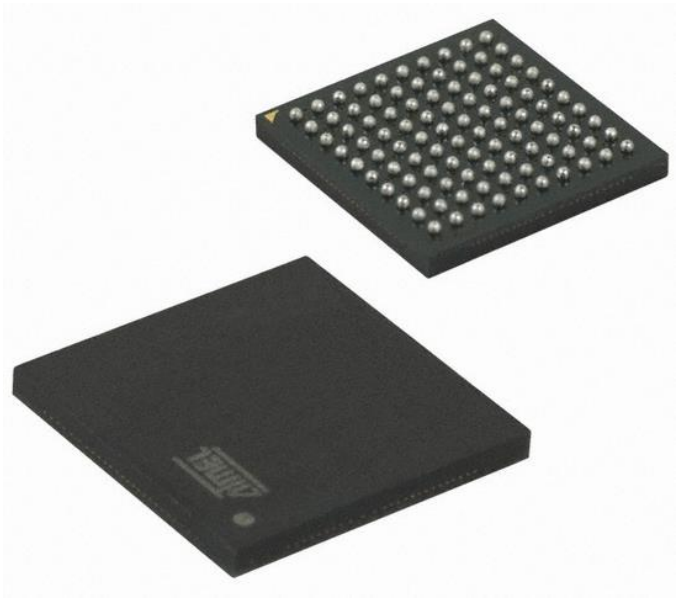


Imagen 34. Encapsulado BGA 100.

9. DISEÑO DE PLACA DE ADAPTACIÓN DE ENCAPSULADO TQFP A PDIP

En la fase de elección de microcontrolador durante el desarrollo de un prototipo de aplicación se suele elegir una versión con encapsulado tipo PDIP por la posibilidad de montaje en placas de pruebas o para facilitar la tarea de soldadura. A veces, el microcontrolador seleccionado no dispone de versión PDIP, y es habitual solucionar este problema mediante el empleo de placas de adaptación de encapsulados de montaje superficial a orificio pasante.

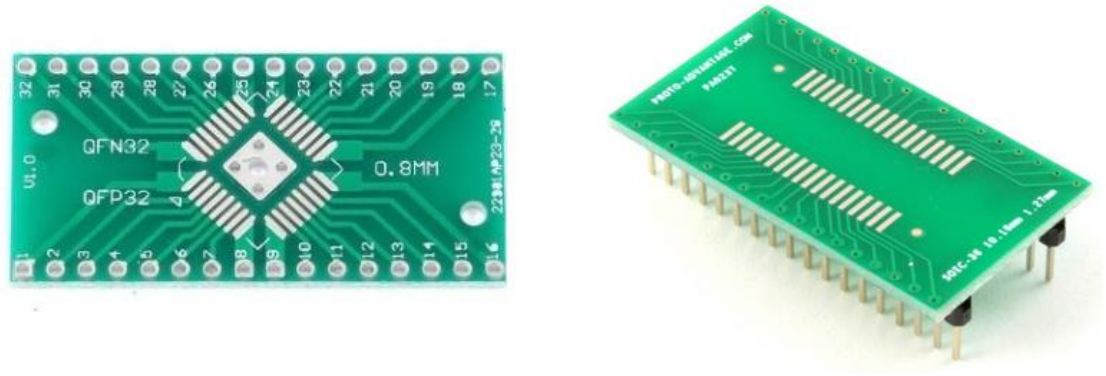


Imagen 35. Placa de adaptación de encapsulado QFN32 a DIP32 (Izq.) y SOIC36 a DIP36 (Dcha.).

Estas placas pueden tener los pads de una forma alargada, lo que permite soldar encapsulados de dos tipos, QFP y QFN.

La soldadura de los dispositivos en las placas de adaptación es relativamente compleja debido a que son componentes de pequeño tamaño y es necesario que la persona encarda de la soldadura tenga suficiente experiencia, ya que la separación entre los pines es muy pequeña, entre 0.4 y 1 mm. Cuando la distancia entre pines es tan pequeña, es aconsejable la utilización de una lupa para observar con mayor detalle el desarrollo de la soldadura, además de realizar una inspección de todos los pines para asegurarse de que no hay ningún cortocircuito y que todas las soldaduras son correctas.

En estos casos, el riesgo de dañar la placa de adaptación es menor que cuando se trata de una placa completa. En desarrollo de prototipos, cuando se encarga la fabricación de las placas a un proveedor, se proporcionan varias unidades, de forma que se dispone de varias oportunidades para conseguir un resultado aceptable y de esta forma se tiene un margen de maniobra cuando se comete algún error de soldadura que implique desechar una placa.

Una característica de este proyecto es el diseño de placas de adaptación de encapsulados de montaje superficial a orificio pasante. Esta placa está destinada a adaptar el encapsulado TQFP del microcontrolador AVR ATmega16 a tipo PDIP.

En primer lugar se ha consultado la disposición de pines de los tipos de encapsulados PDIP y TQFP (ver **¡Error! No se encuentra el origen de la referencia.**) en la hoja de c aracterísticas del microcontrolador.

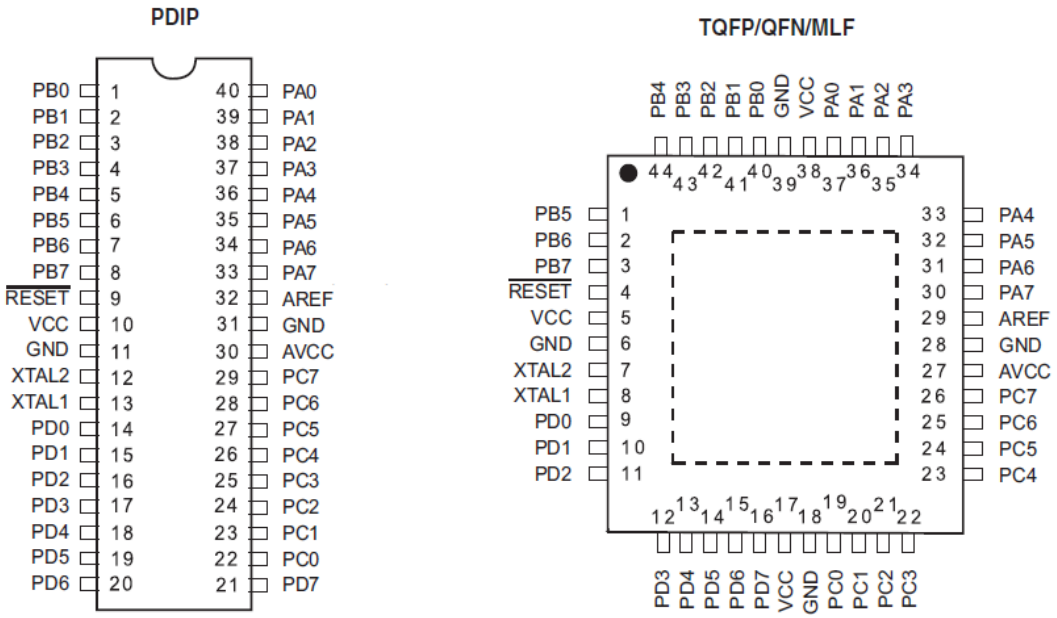


Imagen 36. Pinout del microcontrolador ATmega16.

Analizando ambas disposiciones y teniendo en cuenta que el resultado final de la placa de adaptación debe ser compatible con la disposición del encapsulado PDIP, se ha realizado un esquemático y posteriormente el diseño de la PCB (ver **¡Error! No se encuentra el origen de l a referencia.**).

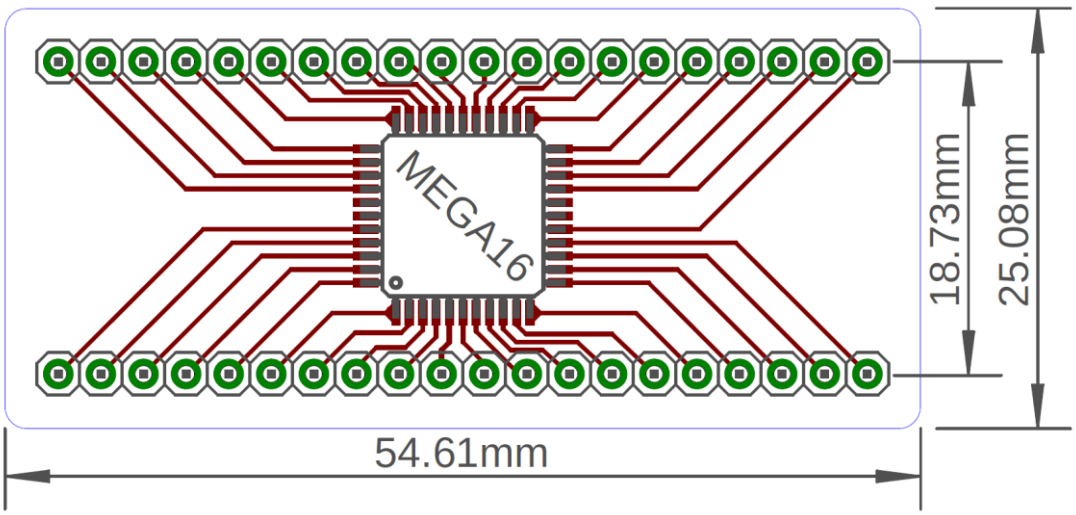


Imagen 37. Esquemático de placa de adaptación de encapsulado TQFN44 a DIP40.

10. DESCRIPCIÓN Y PROGRAMACIÓN DE MÓDULOS I2C

En este capítulo se describen los periféricos controlados por protocolo I2C, indicando las características principales de cada uno de ellos y un pequeño código ejemplo de su funcionamiento.

10.1. DESCRIPCIÓN DE MÓDULO EXPANSOR DE E/S MEDIANTE I2C

Este módulo está basado en el chip PCF8574P, que es un expansor de entradas y salidas digitales de 8 bits. Se controla a través del protocolo I2C que utiliza solo dos líneas para su comunicación, una de reloj para sincronización denominada SCL y otra para datos, denominada SDA [17]. También cuenta con una salida de interrupción que se habilita cuando se detecta un cambio de estado en cualquier pin configurado como entrada. Dispone de tres pines para configurar la dirección del dispositivo, conectando cada uno de ellos a alimentación o masa, lo que permite conectar hasta ocho dispositivos de este tipo a un mismo bus I2C.

Las características eléctricas del puerto de este integrado son una corriente máxima de salida por pin de 300 µA en estado alto y una corriente de entrada de 25 mA en estado bajo.

La corriente típica de un led suele ser de varios miliamperios, por lo que, para utilizar leds con este integrado, será necesario activarlos a nivel bajo.

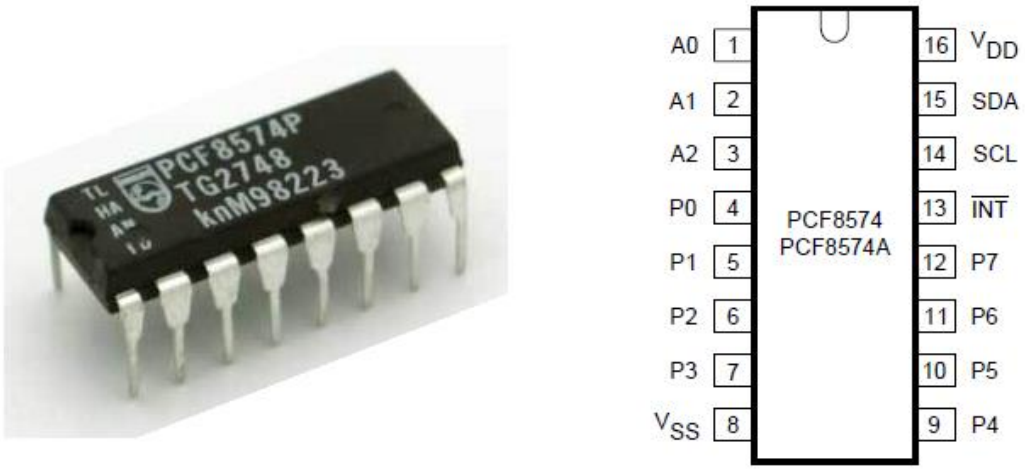


Imagen 38. Fotografía y pinout del circuito integrado PCF8574.

La comunicación con este integrado es muy sencilla. En primer lugar, el dispositivo maestro envía al bus I2C una trama con la dirección del esclavo a quien va dirigida la comunicación, ver imagen.

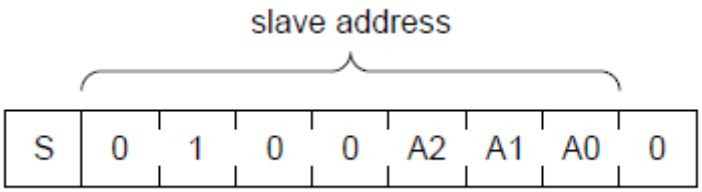


Imagen 39. Trama enviada por el dispositivo maestro con la dirección del dispositivo esclavo.

Donde S es la condición de inicio del protocolo I2C, los bits A2:0 determinan la dirección del dispositivo. El estado de estos bits depende de la conexión eléctrica de los pines correspondientes a alimentación o masa. La posición del "0" menos significativo indica al dispositivo el modo de escritura para la siguiente trama recibida. Si fuese un 1, indicaría el modo lectura, y sería el dispositivo quien enviaría la trama al bus con el estado del puerto.

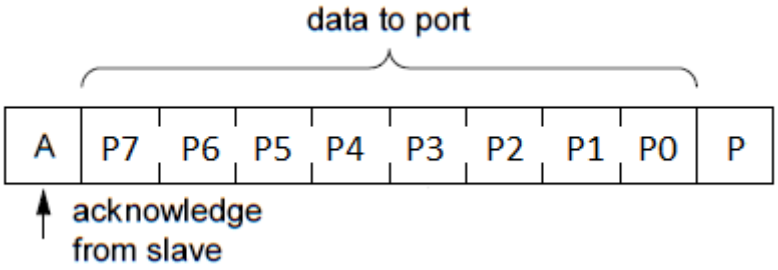


Imagen 40. Trama enviada por el dispositivo esclavo con el estado del puerto.

La Imagen 40 muestra la trama del mensaje enviado por el dispositivo, en modo lectura de entradas digitales, al bus I2C. Donde A es un bit de reconocimiento que es puesto en estado bajo por el dispositivo receptor cada vez que ha recibido un byte. P7:0 es el dato que el dispositivo esclavo debe escribir en el puerto cuando está en modo escritura, o el dato que el dispositivo esclavo envía al bus, procedente de la lectura del puerto, cuando está en modo lectura. P es el bit de parada enviado por el dispositivo maestro para terminar la comunicación.

Para más información, en los documentos adjuntos de este proyecto se incluirá la hoja de datos de este dispositivo.

10.2. PROGRAMACIÓN DE CONTROL PARA EL MÓDULO PCF8574P

Para controlar el funcionamiento del chip PCF8574P se utilizará una librería I2C con las funciones necesarias para manejar el protocolo. Estas funciones configuran los registros del

microcontrolador asociados al protocolo I2C, que en la hoja de datos del microcontrolador se denomina “Two-wire Serial Interface”. La librería I2C estará disponible en los archivos adjuntos de este proyecto.

A continuación, se muestra un código ejemplo para leer o escribir datos en el dispositivo PDF8574P:

```
#include “i2c.h”

uint8_t readPCF(void)
{
	uint8_t dato;
	i2c_iniciar();
	i2c_inicia_com();
	i2c_envia_dato(0x41);   // 0x41 es la dirección de lectura del dispositivo
	dato = i2c_recibe_dato_nack();   // Lectura de un solo byte
	i2c_detener();
	return dato;
}

void writePCF(uint8_t data, uint8_t address)
{
	i2c_iniciar();
	i2c_inicia_com();
	i2c_envia_dato(address);
	i2c_envia_dato(data);
	i2c_detener();
}
```

10.3. PROGRAMACIÓN DEL TECLADO MATRICIAL

El teclado dispone de una conexión para cada fila y otra para cada columna, con un total de 8 conexiones para controlar las 16 teclas (ver Imagen 41). Cuando se pulsa una tecla, las conexiones correspondientes a la fila y a la columna de la tecla quedarán cortocircuitadas mientras la tecla permanezca pulsada.

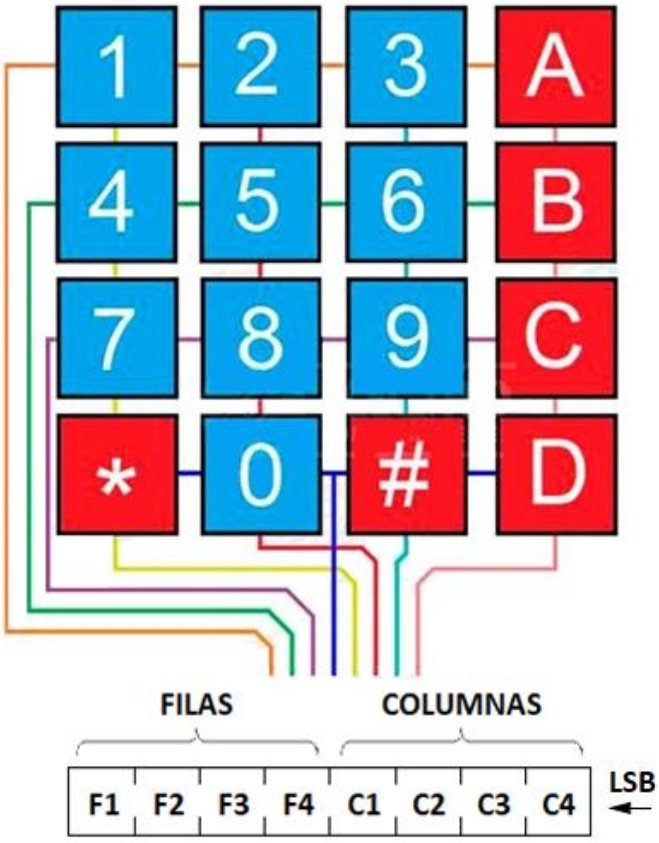


Imagen 41. Interfaz de conexión de teclado matricial.

La idea para saber qué tecla ha sido pulsada es poner las columnas en estado alto, configuradas como entradas, mediante las resistencias de Pull-Up internas del integrado PCF8574P. Las filas se mantienen en estado bajo y configuradas como salidas. Por tanto, el estado del puerto en forma hexadecimal será: 0x0F

En el momento que se pulse una tecla se activará la interrupción y será detectada por el microcontrolador; además, la columna correspondiente cambiará a estado bajo.

A continuación, se leerán las columnas (entradas), que tendrán estados altos excepto la entrada correspondiente a la columna de la tecla pulsada, que tendrá un estado bajo. En este momento solo se conoce la columna de la tecla pulsada.

Después se pone la salida correspondiente a la primera fila en estado alto y se lee el estado de la columna que anteriormente tenía un estado bajo. Si el estado ha cambiado, la tecla pulsada pertenece a la primera fila. Si el estado de la columna no ha cambiado, se restaura a estado bajo la salida de la primera fila y se repite el mismo proceso con la segunda fila y así sucesivamente hasta la cuarta fila.

La programación del proceso descrito sería la siguiente:

```
char getTeclaKeypad(void)
{
        char key = ' ';
        uint8_t row, column;
        uint8_t rowKey = 0;
        uint8_t columnKey = 0;
        char matrix[4][4] = {{'1','2','3','A'},
                             {'4','5','6','B'},
                             {'7','8','9','C'},
                             {'*','0','#','D'}};
        resetINT0();

        for (column=0; column<4; column++)
        {
                if ((~(readPCF())) & (1<<column))
                {
                        rowKey = (row*(-1)) + 7; // Columna encontrada
                for (row=4; row<8; row++)
                        {
                                writePCF((0x0F | (1<<row)), 0x40);
                                if (readPCF() & (1<<column))
                                {
                                        columnKey = (column*(-1)) + 3;
                                        // Tecla encontrada
                                        break;
                                }
                                writePCF(0x0F, 0x40);
                        }
                        writePCF(0x0F, 0x40);
                }
        }
        while(readPCF() != 0x0F); // Espera mientras hay tecla pulsada.

        setINT0();
        return matrix[rowKey][columnKey];
}
```

10.4. PROGRAMACIÓN DEL DISPLAY DE 7 SEGMENTOS

Este display está formado por siete leds o segmentos dispuestos en forma de “8” (ver Imagen 42) para representar caracteres cuya forma sea representable de acuerdo a las posibilidades que ofrecen los siete leds. Cada uno de los leds se puede apagar o encender de forma independiente.

Existen dos tipos: de ánodo común y de cátodo común. Esto hace referencia a la conexión interna de los leds, el tipo ánodo común tiene los ánodos de todos los leds unidos en un solo

pin y se activan a nivel bajo. Igualmente, los de tipo cátodo común, tienen todos los cátodos unidos en un solo pin y se activan a nivel alto.

Además de los siete leds, algunos displays incorporan un octavo led para utilizarlo como punto decimal.

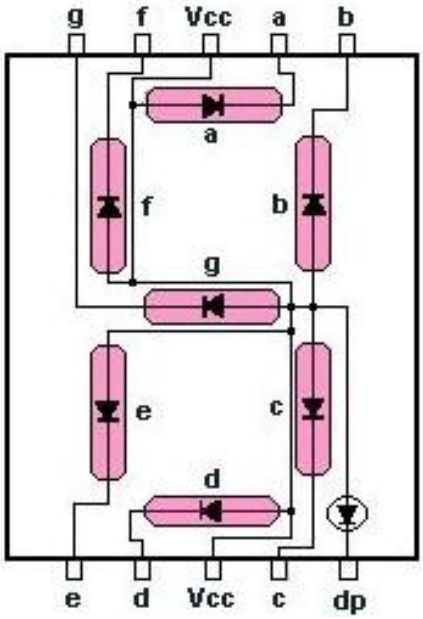


Imagen 42. Pinout de un display de 7 segmentos de tipo ánodo común.

Para controlar este display hay que iniciar la comunicación I2C, enviar la dirección del dispositivo para modo escritura, que en formato hexadecimal es 0x42, enviar el dato correspondiente al dígito o carácter hexadecimal que se desee mostrar y detener la comunicación. Este proceso lo realiza la función “writePCF”, con los parámetros de entrada dato y dirección. La programación sería la siguiente:

```
uint8_t getDisp7(uint8_t caracter){
  if (caracter == '0') {return 0x3F;}
  if (caracter == '1') {return 0x06;}
  if (caracter == '2') {return 0x5B;}
  if (caracter == '3') {return 0x4F;}
  if (caracter == '4') {return 0x66;}
  if (caracter == '5') {return 0x6D;}
  if (caracter == '6') {return 0x7D;}
  if (caracter == '7') {return 0x07;}
  if (caracter == '8') {return 0x7F;}
  if (caracter == '9') {return 0x6F;}
  if (caracter == 'A') {return 0x77;}
  if (caracter == 'B') {return 0x7C;}
  if (caracter == 'C') {return 0x39;}
  if (caracter == 'D') {return 0x5E;}
  if (caracter == 'E') {return 0x79;}
  if (caracter == 'F') {return 0x71;}
  return 0x00; // Si el carácter de entrada no coincide con ninguno de los anteriores,
               // se retorna este valor que corresponde con el display apagado.
}
writePCF(~(getDisp7(tecla)), 0x42);
```

La función “getDisp7” recibe como parámetro de entrada el carácter que se desea mostrar y devuelve un dato con los segmentos que se deben activar para representar dicho carácter.

Como el display utilizado en la placa de desarrollo es de tipo ánodo común, el led de cada segmento se enciende cuando el bit correspondiente del integrado PCF8574P tiene un estado bajo, configurado como salida y actuando como sumidero de corriente. Por este motivo, el dato se envía negado.

11. DESCRIPCIÓN Y PROGRAMACIÓN DEL RESTO DE DISPOSITIVOS

En este capítulo se describen el resto de periféricos que constituyen la placa diseñada, indicando las características principales de cada uno de ellos y un pequeño código ejemplo de su funcionamiento.

11.1. DISPLAY LCD 16x2

Es un dispositivo empleado para visualización de contenidos o información de forma gráfica y, por tanto, se trata de un periférico de salida (ver Imagen 43). Consta de dos filas con dieciséis caracteres por fila y un microcontrolador Hitachi HD44780 que dirige todo el funcionamiento.

La descripción del patillaje del LCD es:

- VSS y VDD: pines de alimentación y masa.
- Vo: pin de contraste. Regulación con un potenciómetro.
- RS: selección de registro para control en estado bajo y registro de datos en estado alto.
- RW: pin de escritura en estado bajo y lectura en estado alto.
- E: pin de habilitación. En estado alto está habilitada la recepción de datos.
- D0:7 bus de datos. D0 es el LSB.
- A y K: ánodo y cátodo del led de retroiluminación.

Imagen 43. Display LCD de 2 filas y 16 columnas.

Para controlar el funcionamiento de esta LCD se utilizará una librería con las funciones necesarias. Las funciones más utilizadas se muestran en un pequeño código ejemplo a continuación:

```
#include "lcd.h"

lcd_init(LCD_DISP_ON);     // Inicialización de LCD.
lcd_clrscr();         // Limpia el LCD y sitúa el cursor en fila 0 columna 0.
lcd_gotoxy(0,1);         // Sitúa el cursor en: fila 1 y columna 0.
lcd_putc('1');       // Escribe un carácter en la posición del cursor.
lcd_puts("Hola mundo!!");         // Escribe una cadena de caracteres.
```

Las funciones “lcd_putc()” o “lcd_puts()” solo admiten caracteres, no dígitos, y escriben el carácter o caracteres en la posición que tuviera el cursor, desplazándose este automáticamente hasta la siguiente columna de la misma fila.

Los valores de las columnas y filas comienzan en 0, y si se utilizan valores superiores a 1 para la fila y 15 para la columna, el carácter o caracteres no se escribirán en el LCD

11.2. CONVERTIDOR A/D. ENTRADA ANALÓGICA CON POTENCIÓMETRO

En la placa se incluye un potenciómetro de 10 KΩ (ver Imagen 44) conectado a alimentación, masa, y el pin ajustable conectado al pin PA0 del microcontrolador, que también es la entrada ADC0 del convertidor A/D, como se puede observar en el pinout del microcontrolador, Imagen 17.

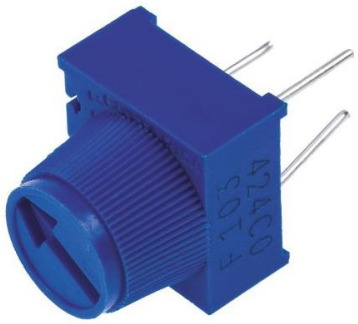


Imagen 44. Potenciómetro de 10 KΩ.

Este dispositivo interno del microcontrolador tiene una resolución de 10 bits, es decir, convierte una entrada analógica en un valor digital de 10 bits. El valor mínimo representa la tensión de masa o GND, y el valor máximo, la tensión de referencia.

El funcionamiento del convertidor A/D se configura en los registros asociados a este, los cuales son ADMUX, ADCSRA, ADC y SFIOR.

ADC Multiplexer Selection Register – ADMUX	Bit	7	6	5	4	3	2	1	0	ADMUX
		REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	
	Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	Initial Value	0	0	0	0	0	0	0	0	
ADC Control and Status Register A – ADCSRA	Bit	7	6	5	4	3	2	1	0	ADCSRA
		ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	
	Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	Initial Value	0	0	0	0	0	0	0	0	
The ADC Data Register – ADCL and ADCH	Bit	15	14	13	12	11	10	9	8	ADCH
		–	–	–	–	–	–	ADC9	ADC8	
		ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
ADLAR = 0		7	6	5	4	3	2	1	0	
	Read/Write	R	R	R	R	R	R	R	R	
		R	R	R	R	R	R	R	R	
	Initial Value	0	0	0	0	0	0	0	0	
		0	0	0	0	0	0	0	0	
Special FunctionIO Register – SFIOR	Bit	7	6	5	4	3	2	1	0	SFIOR
		ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	
	Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
	Initial Value	0	0	0	0	0	0	0	0	

Tabla 6. Registros de configuración para convertidor A/D.

En los bits MUX4:0 del registro ADMUX se seleccionan las entradas del convertidor A/D en modo de entrada individual, diferencial positiva o negativa, con o sin ganancia (Tabla 84 de la hoja de datos del ATmega16, disponible en archivos adjuntos del proyecto), el bit ADLAR selecciona la alineación a izquierda o derecha del valor de la conversión en el registro ADC y los bits REF1:0 seleccionan la tensión de referencia entre tres opciones (ver Tabla 7):

- Tensión externa en el pin AREF. El máximo valor de esta referencia es la tensión de alimentación.
- Tensión de alimentación en el pin AVCC.
- Tensión de referencia interna de 2,56 V.

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

Tabla 7. Referencias de tensión seleccionadas en los bits REFS1:0 del registro ADMUX.

En el registro ADCSRA se configura el control y el estado del convertidor A/D:

- ADEN: en estado alto habilita el convertidor.
- ADSC: al activar este bit se inicia la conversión.
- ADATE: habilita el disparador automático. Se iniciará la conversión en cada flanco positivo de la señal de disparo seleccionada en el registro SFIOR.
- ADIF: flag de interrupción. Se activa al completarse una conversión y produce la ejecución de la interrupción si esta se encuentra habilitada.
- ADIE: en estado alto, la interrupción por conversión completada está activada. Requiere la activación del bit I del registro SREG, interrupciones globales activadas.
- ADPS2:0 selección del prescaler para el reloj del convertidor A/D. (Tabla 85 de la hoja de datos del ATmega16).

El registro ADC es un registro compuesto por dos bytes, ADCH y ADCL, que almacenan el resultado de la conversión, un valor de 10 bits.

En el registro SFIOR se selecciona el modo de funcionamiento del convertidor (Ver Tabla 8).

ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free Running mode
0	0	1	Analog Comparator
0	1	0	External Interrupt Request 0
0	1	1	Timer/Counter0 Compare Match
1	0	0	Timer/Counter0 Overflow
1	0	1	Timer/Counter1 Compare Match B
1	1	0	Timer/Counter1 Overflow
1	1	1	Timer/Counter1 Capture Event

Tabla 8. Modos de funcionamiento del convertidor A/D.

A continuación, se muestra un pequeño código ejemplo de funcionamiento del convertidor A/D en modo funcionamiento libre, con referencia de tensión de alimentación, y con interrupción por conversión completada habilitada.

```
#include <avr/io.h>
#include <avr/interrupt.h>

void adcInit(void);      // Declaración
volatile uint16_t dato;

int main(void)
{
        adcInit(void);
        while (1)
        {

        }
}
ISR(ADC_vect)
{
  dato = ADC;
}
void adcInit(void)
{
        ADMUX |= (1<<REFS0);      // Referencia de tensión AVCC y canal ADC0
        ADCSRA |= (1<<ADEN)|(1<<ADSC)|(1<<ADATE)|(1<<ADIE)|(1<<ADPS2)|(1<<ADPS1);
        SFIOR = 0;      // Modo funcionamiento libre.
        sei();      // Habilitación de interrupciones globales.
}
```

11.3. INTERRUPTORES Y PULSADORES COMO ENTRADAS DE PROPÓSITO GENERAL

En la placa se incluyen dos interruptores, conectados a los pines PB0 y PB1, que funcionarán como simples entradas digitales y dos pulsadores, conectados a los pines PB2 y PD3, que podrán funcionar como simples entradas digitales o como interrupciones externas.

Imagen 45. Tipos de interruptores y pulsadores que se utilizarán en la placa.

Los interruptores, aunque son de dos vías, solo se utilizará una de ellas para conectar el pin del microcontrolador a masa cuando el interruptor se encuentre accionado. Los pulsadores funcionarán de la misma forma, con la única diferencia de que no disponen de enclavamiento, es decir, sus contactos se mantienen unidos mientras permanecen pulsados.

Los registros de configuración para los puertos del microcontrolador son:

- Registro DDRx: x se sustituirá por el puerto, A, B, C o D. Este registro de 8 bits se utiliza para configurar cada pin de un puerto como entrada, estableciendo el bit correspondiente en estado bajo.
- Registro PORTx: igualmente, x se sustituirá por el puerto, A, B, C o D. Este registro, también de 8 bits, se utiliza para activar la resistencia de Pull-Up interna asociada a cada pin si este está configurado como entrada en el registro DDRx, y el bit correspondiente del registro PORTx se establece en estado alto.
- Registro PINx: este es el registro para lectura digital del estado de cada bit, independientemente de que esté configurado como entrada o como salida.

Un ejemplo del funcionamiento de estos registros como entradas se mostrará más adelante en este mismo apartado, y del funcionamiento como salidas en el apartado posterior.

Las interrupciones externas se configuran en los registros MCUCR, MCUCSR y GICR.

MCU Control Register – MCUCR	Bit	7	6	5	4	3	2	1	0	MCUCR
		SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	
	Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	Initial Value	0	0	0	0	0	0	0	0	
MCU Control and Status Register – MCUCSR	Bit	7	6	5	4	3	2	1	0	MCUCSR
		JTD	ISC2	–	JTRF	WDRF	BORF	EXTRF	PORF	
	Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
	Initial Value	0	0	0						
General Interrupt Control Register – GICR	Bit	7	6	5	4	3	2	1	0	GICR
		INT1	INT0	INT2	–	–	–	IVSEL	IVCE	
	Read/Write	R/W	R/W	R/W	R	R	R	R/W	R/W	
	Initial Value	0	0	0	0	0	0	0	0	

Tabla 9. Registros de configuración para interrupciones externas.

El registro MCUCR configura el modo de disparo de las interrupciones INT0 e INT1 por nivel bajo, cambio lógico y flanco de subida o bajada en el pin asociado a cada interrupción. En la Tabla 10 se muestran los modos para la interrupción INT1, y de la misma forma sería para la interrupción INT0 en los bits ISC01 e ISC00.

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

Tabla 10. Selección del modo de disparo de la interrupción externa INT1.

En MCUCSR (ver Tabla 9) se configura el disparo de la interrupción externa INT2 por flanco de bajada si el bit ISC2 está escrito a cero, y por flanco de subida en caso contrario.

En el registro GICR (ver Tabla 9) se habilitan las interrupciones externas INT0, INT1 e INT2, activando los bits correspondientes.

A continuación, se muestra un pequeño código ejemplo para la configuración de las entradas conectadas a los interruptores y pulsadores y para las interrupciones externas INT1 e INT2:

```
#include <avr/io.h>
#include <avr/interrupt.h>

int main(void)
{
	DDRB &= ~((1<<DDD0)|(1<<DDD1)|(1<<DDD2));   // PB0, PB1 y PB2 Configurados
                                                    // como entradas digitales
	PORTB = (1<<PORTB0)|(1<<PORTB1)|(1<<PORTB2);   // Activación de resistencias
                                                  // internas de PULL-UP para PB0, PB1 y PB2
	DDRD &= ~(1<<DDD3);    // PD3 Configurado como entrada digital
	PORTD = (1<<PORTD3);   // Activación de resistencia interna de PULL-UP para PD3

	MCUCR |= (1<<ISC11);      // Flanco de bajada en INT1 genera la interrupción
	MCUCSR &= ~(1<<ISC2);    // Flanco de bajada en INT2 genera la interrupción
	GICR |= (1<<INT1)|(1<<INT2);    // Habilitación de INT1 e INT2

	sei();   // Habilitación de interrupciones globales en registro SREG.

	while (1)
	{

	}
}
 ISR(INT1_vect)
{
		// Código a ejecutar en la interrupción INT1
}
```

```
ISR(INT2_vect)
{
        // Código a ejecutar en la interrupción INT2
}
```

11.4. LEDS Y ZUMBADOR PARA PROPÓSITO GENERAL. SALIDAS EN MODO PWM

En la placa se incluyen tres leds, conectados a los pines PD4, PD6 y PD7 del microcontrolador, que podrán funcionar como salidas digitales o salidas en modo PWM. También se incluye un zumbador, conectado al pin PD5, que funcionará en modo PWM de Fase y Frecuencia correcta del Timer1

Para configurar los leds como salidas de propósito general, se utilizan los siguientes registros de los puertos del microcontrolador:

Registro DDRx: x se sustituirá por el puerto, A, B, C o D. Este registro de 8 bits se utiliza para configurar cada pin de un puerto como salida, estableciendo el bit correspondiente en estado alto.

Registro PORTx: igualmente, x se sustituirá por el puerto, A, B, C o D. Este registro, también de 8 bits, se utiliza para establecer el estado digital de cada pin configurado como salida en el registro DDRx correspondiente.

El led conectado al pin PD4, además de utilizarse como salida de propósito general, también puede utilizarse como salida en modo PWM, controlada por el timer1. Este temporizador también puede controlar el pin PD5 en modo salida PWM, y por este motivo, el zumbador se ha conectado en dicho pin.

En el microcontrolador ATmega16 dispone de tres modalidades de PWM, modo rápido, fase correcta y frecuencia y fase correcta. El timer0 y timer2 disponen de los modos rápido y fase correcta, mientras que el timer1 dispone de los tres modos. A continuación, se explicarán brevemente las tres modalidades nombrando los registros del timer1, aunque para los demás temporizadores el funcionamiento es prácticamente igual, a diferencia de los registros asociados a dichos temporizadores.

El timer1 controla dos salidas en modo PWM, PD4 como salida OC1B y PD5 como salida OC1A. Para hacer referencia a ambas salidas, se utilizará OC1x, sustituyendo x por A o B según la salida utilizada.

PWM modo rápido: el contador comienza en 0, incrementando su valor en cada ciclo, hasta que termina en el valor máximo, establecido por la resolución configurada entre 8, 9 o 10 bits. Cuando se alcanza el valor máximo, el conteo se reinicia. El valor del conteo se actualiza en el registro TCNT1. Dicho valor se compara, en cada incremento, con el valor almacenado en el registro OCR1x, para, cuando ambos valores coincidan, invertir el estado de la salida (ver Imagen 46), y, de esta manera, controlar el ciclo de trabajo de la señal de salida, pin OC1x, mediante la modificación del valor de OCR1x. La frecuencia se establece mediante el prescaler, seleccionado en el registro TCCR1B, y el número de bits de la resolución, seleccionado en los registros TCCR1A y TCCR1B según la Tabla 12.

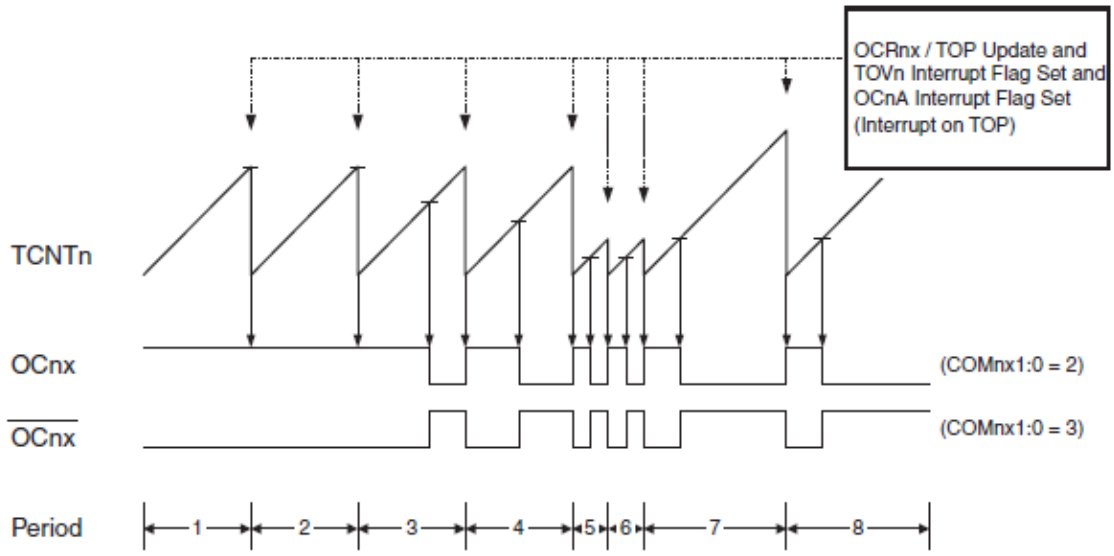


Imagen 46. Ejemplo de PWM modo rápido.

PWM modo fase correcta: este modo es muy similar al anterior, con la diferencia de que, una vez que el conteo ascendente alcanza el valor máximo, comienza el conteo descendente hasta el valor 0. Cuando el valor del conteo coincide con el valor de OCR1x, se invierte la señal de salida, sucediendo esto durante las dos fases de conteo, ascendente y descendente. La señal de salida, OC1x, se observa en la Imagen 47. La frecuencia en este modo es la mitad que en el modo rápido.

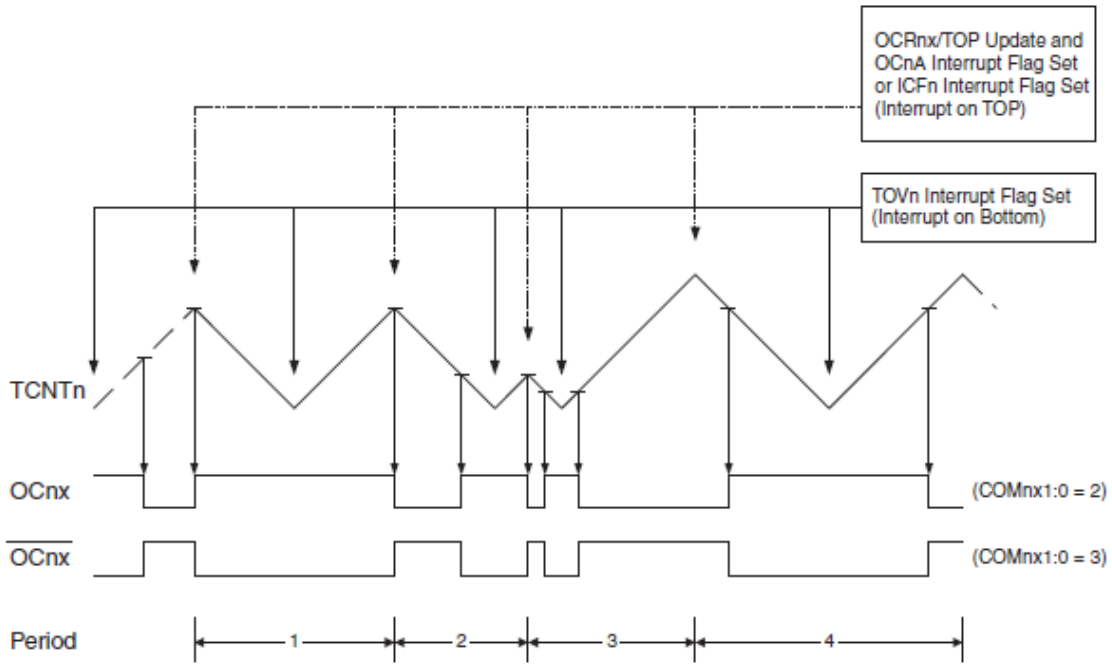


Imagen 47. Ejemplo de PWM modo fase correcta.

PWM modo fase y frecuencia correcta: está basado, al igual que el modo de fase correcta, en un sistema de doble pendiente, es decir, conteo ascendente y descendente. El valor máximo de conteo se establece en el registro OCR1x y se actualiza cada vez que el contador pasa por 0. La señal de salida, OC1x, se invierte cada vez que el contador alcanza la mitad del valor de OCR1x (ver Imagen 48). De esta forma, el ciclo de trabajo queda fijado al 50%, siendo esta la principal diferencia con los modos anteriores. Por tanto, la modificación del valor máximo de conteo afecta a la frecuencia.

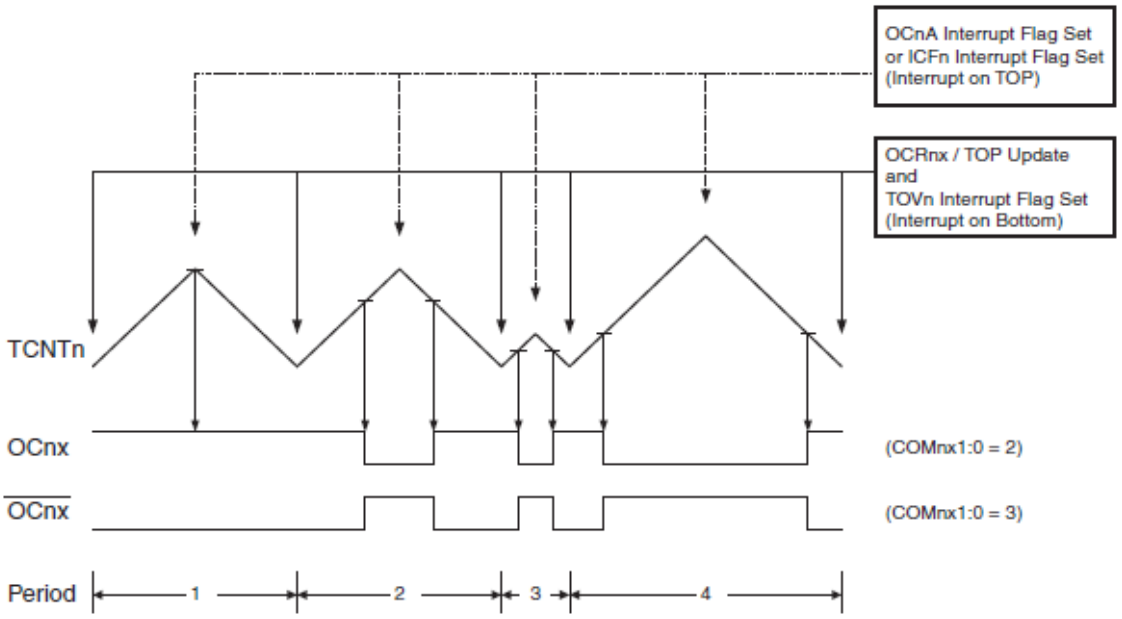


Imagen 48.. Ejemplo de PWM modo fase y frecuencia correcta.

Los registros asociados al funcionamiento del módulo PWM se muestran en la Tabla 11. En ellos se configuran el modo de cambio del nivel lógico de salida, selección de la modalidad PWM (ver Tabla 12) y de prescaler, además de otras funciones específicas.

Timer/Counter1 Control Register A – TCCR1A	Bit	7	6	5	4	3	2	1	0	TCCR1A
		COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	
	Read/Write	R/W	R/W	R/W	R/W	W	W	R/W	R/W	
	Initial Value	0	0	0	0	0	0	0	0	
Timer/Counter1 Control Register B – TCCR1B	Bit	7	6	5	4	3	2	1	0	TCCR1B
		ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	
	Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
	Initial Value	0	0	0	0	0	0	0	0	

Tabla 11. Registros de configuración para modo PWM del timer1.

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1X	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	Reserved	–	–	–
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Tabla 12. Modalidades PWM del timer1.

En la placa diseñada se podrá comprobar el funcionamiento de las modalidades rápida y fase correcta en los leds conectados a los pines PD4, salida OC1B del timer1, y PD7, salida OC2 del timer2, y la modalidad fase y frecuencia correcta en el zumbador, conectado al pin PD5, salida OC1A del timer1.

A continuación, se mostrará un pequeño código ejemplo donde se configuran los registros para utilizar dos salidas PWM. En este caso se podrá modificar el brillo del led PD4 (salida OC1B) mediante PWM modo rápido y emitir sonido en el zumbador PD5 (salida OC1A) mediante PWM modo fase y frecuencia correcta, utilizando, en ambos casos, la entrada analógica explicada en un apartado anterior:

```
#include <avr/io.h>
#include <avr/interrupt.h>

void pwmInit(void); // Declaración de la función.
uint8_t modo = 1;
int main(void)
{
        adcInit(); // Inicialización del Convertidor A/D para entrada analógica.
        pwmInit();
        sei(); // Habilitación de las interrupciones globales.
```

```
        while (1)
        {
                // Código para el bucle infinito
        }
}
ISR(ADC_vect){
        if (modo == 1)
        {
                OCR1B = ADC/4; // Ciclo de trabajo en LED_PD4
        }
        if (modo == 3)
        {
                OCR1A = 2000 - ADC; // Frecuencia en BUZZER_PD5
        }
}
void pwmInit(void)
{
        if (modo == 1){ // PWM MODO RÁPIDO
                DDRD |= (1<<DDD4); // Configuración del pin PD4 (OC1B) como salida
                TCCR1A = (1<<COM1B1) | (1<<WGM10); // WGM MODO 5 (8 bits)
                TCCR1B = (1<<WGM12) | (1<<CS10); // Sin prescaler
        }
        if (modo == 3){ // PWM MODO FASE Y FRECUENCIA CORRECTA
                DDRD |= (1<<DDD5); // Configuración del pin PD5 (OC1A) como salida
                TCCR1A = (1<<COM1A0) | (1<<WGM10); // WGM MODO 9
                TCCR1B = (1<<WGM13) | (1<<CS10); // Sin prescaler
        }
}
```

En la rutina de atención a la interrupción del convertidor A/D se actualizan los valores de los registros OCR1A y OCR1B. Esta rutina se ejecuta cuando se completa una conversión en el convertidor A/D.

En el ejemplo anterior, la instrucción que actualiza el valor del registro OCR1B, "OCR1B=ADC/4;", la división por 4 se debe a que se está utilizando un modo PWM rápido de 8 bits, mientras que el convertidor A/D proporciona valores de 10 bits. De esta manera, el valor máximo proporcionado por el convertidor será también el valor máximo permitido por el modo PWM rápido de 8 bits.

En el caso del zumbador, la instrucción que actualiza el valor del registro OCR1A es "OCR1A=2000-ADC;". Esto se debe a que el zumbador emite sonido en el rango aproximado de frecuencias entre 2 y 5 KHz. El proceso de conversión para adaptar los valores que proporciona el convertidor A/D (registro ADC) a los valores de frecuencia permitidos por el zumbador se muestra en la Imagen 49.

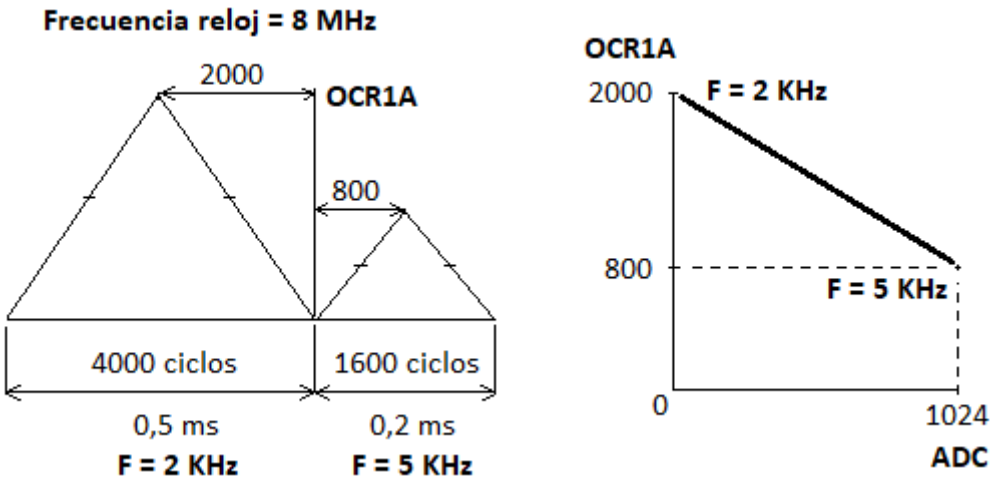


Imagen 49. Proceso para adaptación de valores ADC a valores de OCR1A.

Para encontrar la relación entre OCR1A y ADC se aplica la ecuación de una recta en su forma pendiente-ordenada al origen, obteniéndose:

$$OCR1A = 2000 - 1{,}17 \cdot ADC$$

Como las frecuencias 2 KHz y 5 KHz del zumbador son aproximadas, la ecuación anterior se simplificará, utilizando finalmente la ecuación:

$$OCR1A = 2000 - ADC$$

En los archivos adjuntos de este proyecto se encuentra la hoja de datos del microcontrolador, donde se podrá encontrar toda la información para el uso de las diferentes modalidades PWM.

11.5. COMUNICACIÓN MEDIANTE PROTOCOLO USART

Como se ha mencionado en un capítulo anterior, en la placa se incluye un interruptor de dos vías y dos posiciones, que permite seleccionar la comunicación del PC, mediante protocolo USART y el módulo FTDI, con el programador/depurador o con el microcontrolador.

Los registros que controlan el protocolo USART en el microcontrolador se muestran en la Tabla 13.

USART I/O Data Register – UDR	Bit	7	6	5	4	3	2	1	0	
		RXB[7:0]								UDR (Read)
		TXB[7:0]								UDR (Write)
	Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	Initial Value	0	0	0	0	0	0	0	0	
USART Control and Status Register A – UCSRA	Bit	7	6	5	4	3	2	1	0	
		RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	UCSRA
	Read/Write	R	R/W	R	R	R	R	R/W	R/W	
	Initial Value	0	0	1	0	0	0	0	0	
USART Control and Status Register B – UCSRB	Bit	7	6	5	4	3	2	1	0	
		RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
	Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
	Initial Value	0	0	0	0	0	0	0	0	
USART Control and Status Register C – UCSRC	Bit	7	6	5	4	3	2	1	0	
		URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
	Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	Initial Value	1	0	0	0	0	1	1	0	
USART Baud Rate Registers – UBRRL and UBRRH	Bit	15	14	13	12	11	10	9	8	
		URSEL	–	–	–	UBRR[11:8]				UBRRH
		UBRR[7:0]								UBRRL
		7	6	5	4	3	2	1	0	
	Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	Initial Value	0	0	0	0	0	0	0	0	
		0	0	0	0	0	0	0	0	

Tabla 13. Registros de configuración para protocolo USART.

El registro UDR es un buffer que almacena el dato que se va a transmitir o que se ha recibido.

UCSRA contiene los flags para datos enviados o recibidos, bit de estado del buffer, bits de errores de trama, sobre escritura o paridad, bit para doble velocidad en modo asíncrono y bit para comunicación multiprocesador.

UCSRB contiene los bits de habilitación de interrupción por dato recibido, enviado o registro de dato vacío, bits de habilitación de receptor y transmisor USART y el noveno bit para datos recibidos de 9 bits.

UCSRC contiene un bit para selección del acceso al registro UCSRC o al UBRRH, bit de selección para modo síncrono o asíncrono, bits de selección para el modo de paridad, bit de selección para número de bits de stop, bits para selección del número de bits de los datos de transmisión o recepción.

UBRR es un registro de 12 bits que contiene la velocidad de transmisión en baudios.

A continuación, se muestra un pequeño código ejemplo de la configuración de los registros y funciones para enviar datos en una comunicación USART en modo asíncrono, con ocho bits de datos, sin bit de paridad, un bit de stop y una velocidad de 19200 bps a 8 MHz de frecuencia de reloj.

```
volatile char datoUsart;

ISR(USART_RXC_vect)
{
        datoUsart = UDR; // Dato recibido y guardado en la variable datoUsart.
}
void usartInit(void)
{
        cli();
        UCSRA = 0;
        UCSRB |= (1<<RXCIE)|(1<<RXEN)|(1<<TXEN);
        UCSRC |= ((1<<URSEL) | (1<<UCSZ1) | (1<<UCSZ0));
        UBRRL = 0x19; // Baud rate para F_CPU = 8 MHz y 19200 bps
        sei();
}
void envia_caracter_usart(char caracter)
{
        while(!(UCSRA&(1<<5)));    // Espera mientras el buffer no está vacío.
        UDR = caracter; // Volcado del carácter a enviar en el buffer.
}
void envia_cadena_usart(char* cadena) // Envío de cadena de caracteres.
{
        while(*cadena !=0x00) // Espera mientras el último valor de la cadena
        {                              // sea diferente a carácter nulo.
                envia_caracter_usart(*cadena);
                cadena++;
        }
}
```

En los archivos adjuntos de este proyecto se encuentra la hoja de datos del microcontrolador, donde se podrá encontrar toda la información para el uso de este protocolo.

12. CONCLUSIONES

Este proyecto intenta ayudar a cualquier estudiante o aficionado a la electrónica a percibir de forma más sencilla el diseño, realización y montaje de placas de desarrollo y prototipos, así como presentar los microcontroladores como una opción para realizar cualquier tipo de aplicación. También se hace una introducción al software para programación y depuración de estas aplicaciones, herramientas de ayuda, asistentes para facilitar la programación, etc.

La posibilidad de implementación del programador y depurador JTAG ICE mk1 con licencia libre proporciona una herramienta muy útil para el desarrollo de aplicaciones basadas en microcontroladores ATmega a un precio reducido, debido solo al coste de los componentes, aunque teniendo en cuenta sus limitaciones.

La placa de desarrollo realizada ofrece un número aceptable de posibilidades para el aprendizaje de la programación de microcontroladores, con un tamaño que permite su utilización de forma cómoda y manejable. Además, ofrece la posibilidad de ampliación mediante el protocolo I2C y el puerto B del microcontrolador ayudándose de una placa de pruebas.

El puerto B también permite la programación del microcontrolador mediante un método alternativo al protocolo JTAG, como es el protocolo SPI, aunque este no permite depuración. También se incluye un puerto para un programador/depurador externo mediante protocolo JTAG.

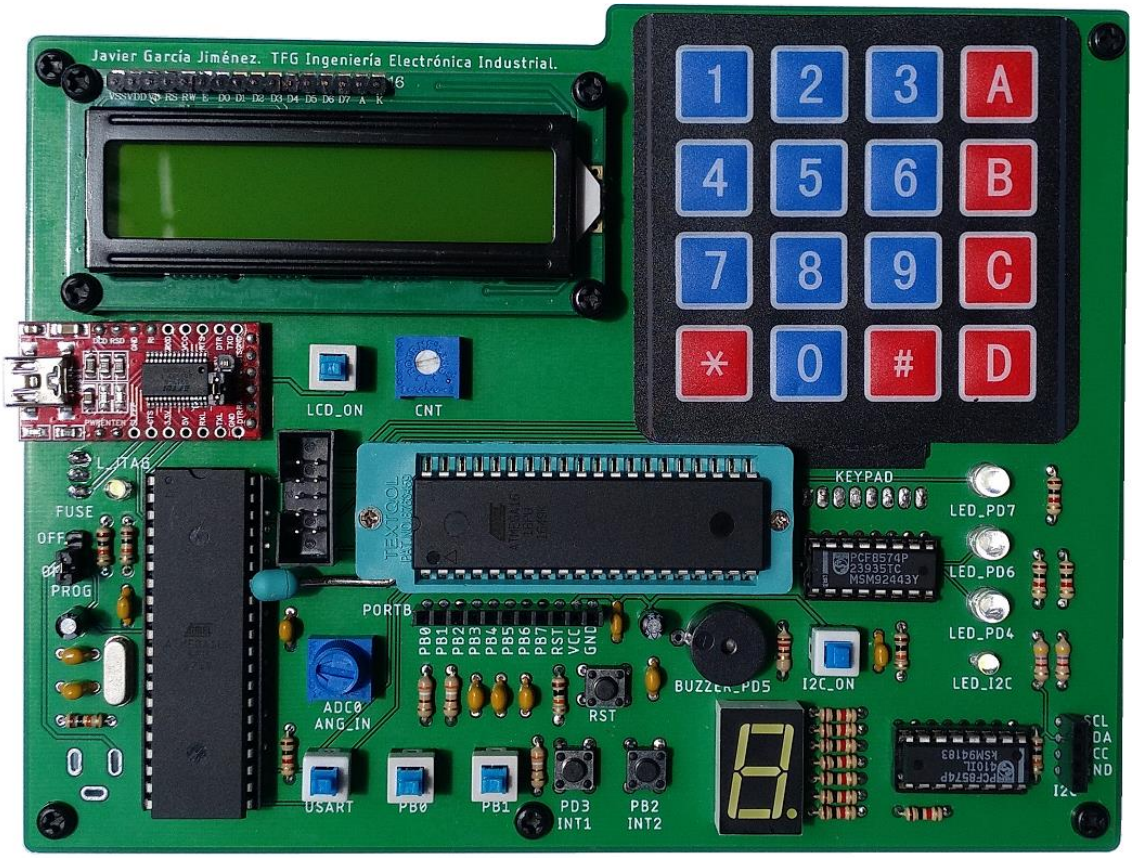


Imagen 50. Fotografía de la placa terminada.

13. BIBLIOGRAFÍA

[1] A. Daga, «Engineersgarage,» [En línea]. Available: https://www.engineersgarage.com/articles/avr-microcontroller. [Último acceso: Junio 2019].

[2] P. Espeso, «Xataka,» Diciembre 2012. [En línea]. Available: https://www.xataka.com/componentes/cisc-frente-a-risc-una-batalla-en-blanco-y-negro. [Último acceso: Junio 2019].

[3] Microchip, «AVR MCU Family,» [En línea]. Available: https://www.microchip.com/design-centers/8-bit/avr-mcus/device-selection. [Último acceso: Junio 2019].

[4] Microchip, «PIC MCU Families,» [En línea]. Available: https://www.microchip.com/design-centers/8-bit/pic-mcus/device-selection. [Último acceso: Junio 2019].

[5] STMicroclectronics, «STM8 8-bit MCUs,» [En línea]. Available: https://www.st.com/content/st_com/en/products/microcontrollers-microprocessors/stm8-8-bit-mcus.html. [Último acceso: Junio 2019].

[6] NXP, «NXP 8/16 bit MCUs,» [En línea]. Available: https://www.nxp.com/products/processors-and-microcontrollers/additional-processors-and-mcus/8-16-bit-mcus:8-16-BIT-MICROCONTROLLERS. [Último acceso: Junio 2019].

[7] A. C. «AVR910: In-System Programming,» 11 2016. [En línea]. Available: http://ww1.microchip.com/downloads/en/appnotes/atmel-0943-in-system-programming_applicationnote_avr910.pdf. [Último acceso: Junio 2019].

[8] M. «ISP Programming,» [En línea]. Available: https://www.microchip.com/webdoc/stk500/stk500.ispprogramming.html. [Último acceso: Junio 2019].

[9] K. Navarro, «Panamahitek,» Octubre 2014. [En línea]. Available: http://panamahitek.com/como-funciona-el-protocolo-spi/. [Último acceso: Junio 2019].

[10] I. o. E. a. E. E. «IEEE Standard for Test Access Port and Boundary-Scan Architecture,» New York, 2013.

[11] Psykhon, «Todopic,» Febrero 2007. [En línea]. Available: https://www.todopic.com.ar/foros/index.php?topic=16129.0. [Último acceso: Junio 2019].

[12] Mikrocontroller.net, «DebugWIRE,» [En línea]. Available: https://www.mikrocontroller.net/articles/DebugWIRE. [Último acceso: Junio 2019].

[13] Abcminiuser, «AVRFREAKS,» AVR Programming Methods, Mayo 2006. [En línea]. Available: https://www.avrfreaks.net/forum/tut-hard-avr-programming-methods?name=PNphpBB2&file=viewtopic&t=38691. [Último acceso: Junio 2019].

[14] C. Walls, Embedded Software. The Works, 2012.

[15] A. C. «AVR JTAG-ICE User Guide,» 2001. [En línea]. Available: http://ww1.microchip.com/downloads/en/devicedoc/doc2475.pdf. [Último acceso: Junio 2019].

[16] F. C. «FT232R USB UART IC Datasheet,» 2019. [En línea]. Available: https://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf. [Último acceso: 26 06 2019].

[17] ELPROCUS, «I2C bus protocol Tutorial, Interface with applications,» [En línea]. Available: https://www.elprocus.com/i2c-bus-protocol-tutorial-interface-applications/. [Último acceso: Junio 2019].

[18] M. Ali Mazidi, S. Naimi y S. Naimi, The AVR microcontroller and embedded system, Pearson, 2010.

ANEXO I

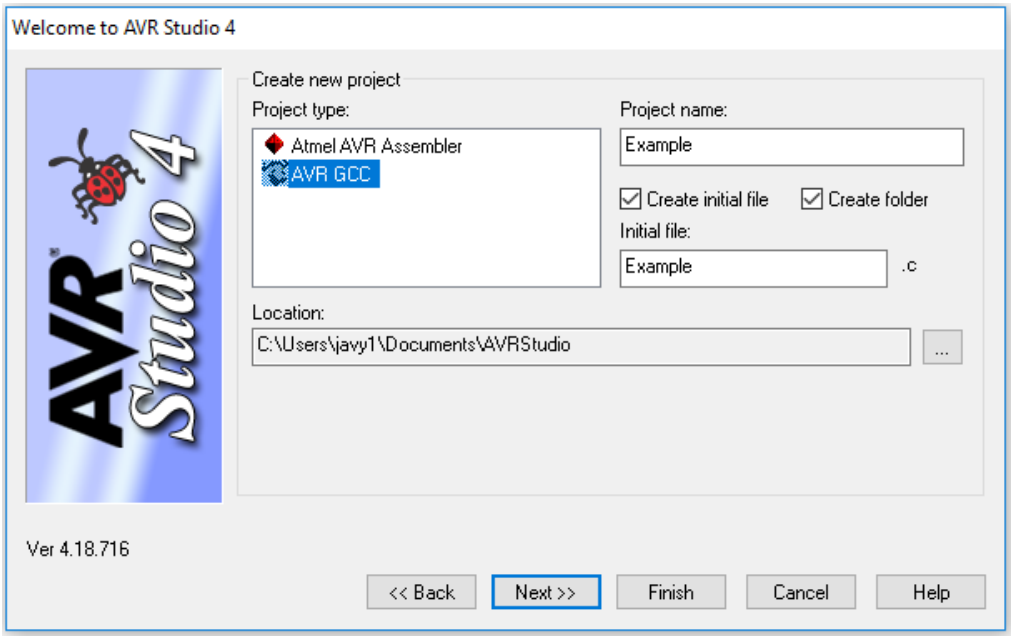
ENTORNO DE PROGRAMACIÓN Y DEPURACIÓN AVR STUDIO 4.18

CONTENIDO

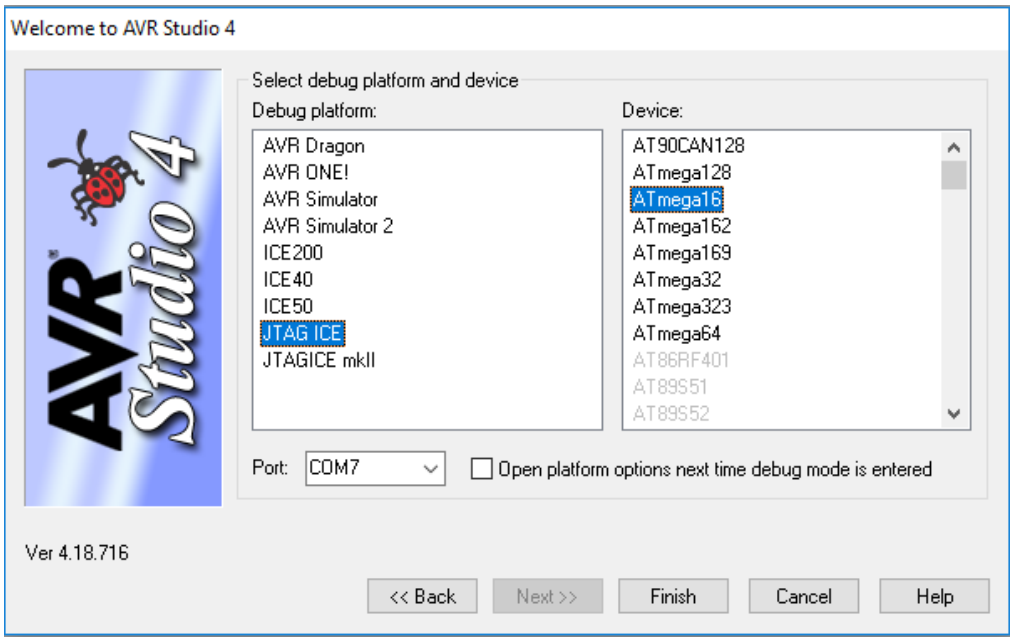
ÍNDICE DE CAPTURAS

1. ASISTENTE PARA CREACIÓN DE UN PROYECTO

AVR Studio 4 es un entorno de programación y depuración lanzado en el año 2002 y, por tanto, se trata de un entorno muy básico. Incluye un asistente para la creación de proyectos en el cual se elige el nombre del proyecto, dirección, compilador GCC (para programación en C o C++) o Ensamblador, plataforma de depuración, por ejemplo, JTAG ICE o AVR Simulator, y el microcontrolador a utilizar.



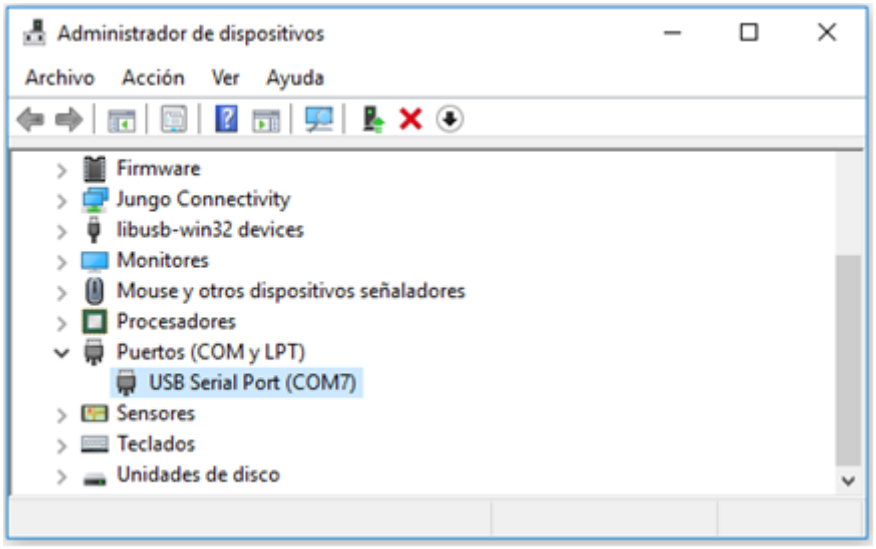
Captura 1. Elección del compilador, nombre de proyecto y ubicación.



Captura 2. Selección del dispositivo programador, microcontrolador y puerto COM.

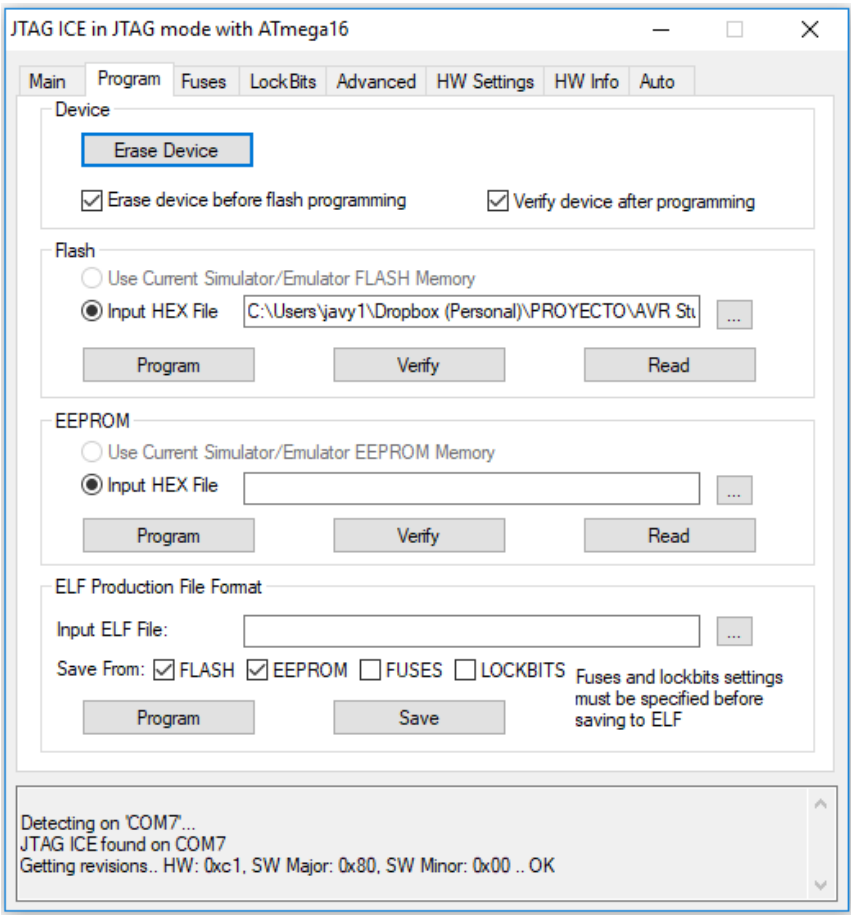
2. CONEXIÓN DEL PROGRAMADOR AL PC. MENÚS DE PROGRAMACIÓN

Cuando se conecta el módulo FTDI del programador al PC en un puerto USB, se detecta como Puerto Serie (COM). Normalmente un PC tiene varios de estos puertos y el número asignado depende del puerto USB al cual se conecte el módulo FTDI, en nuestro caso ha resultado ser el COM7.

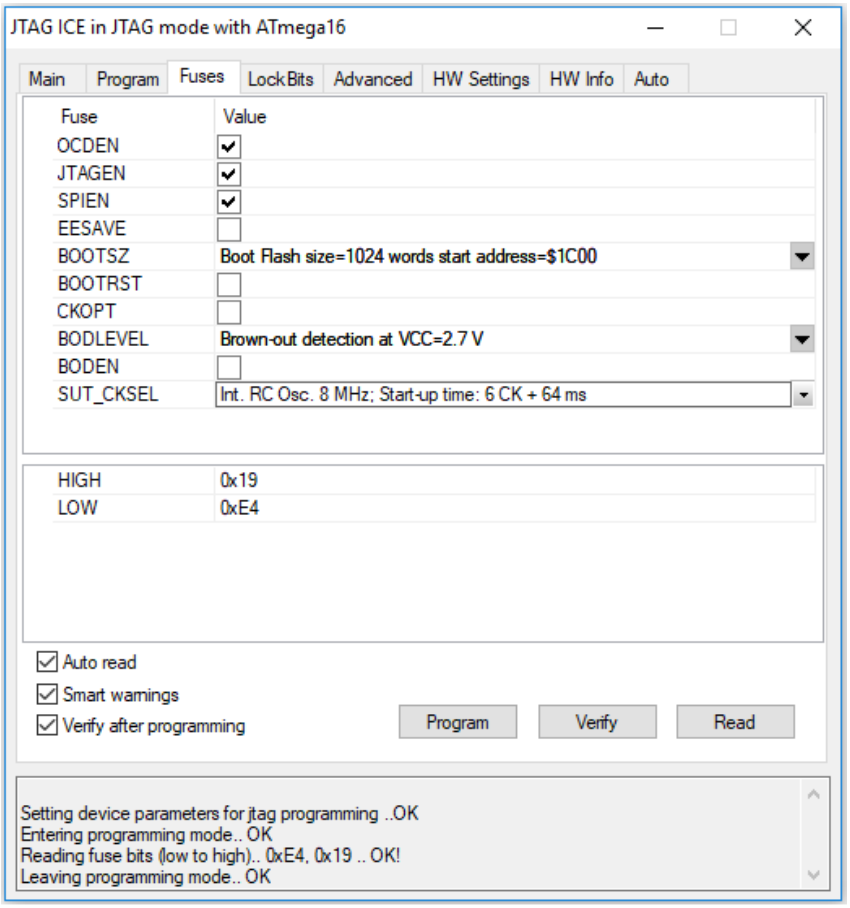


Captura 3. Comprobación del Puerto Serie asignado.

Cuando el programa detecta un programador o depurador compatible conectado a un puerto serie del PC, permite el acceso al menú de programación, en el que se puede programar la memoria flash o EEPROM, los “fuse bits” y “lock bits”. Estos últimos se pueden consultar en el apartado FUSE Y LOCK BYTES



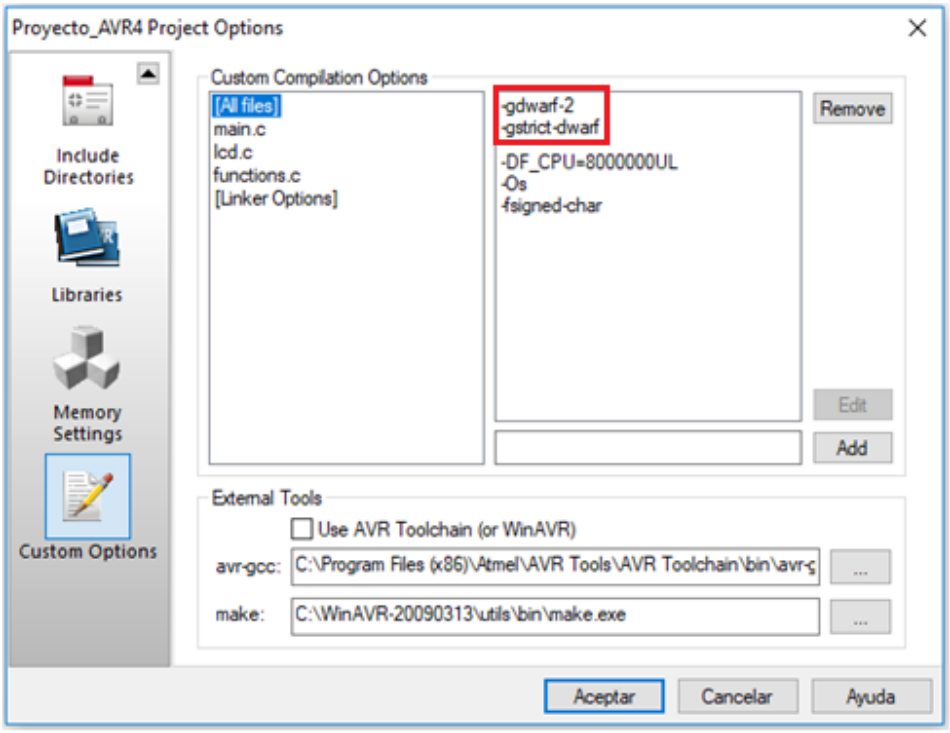
Captura 4. Menú de programación de memoria.



Captura 5. Menú de programación de fuses.

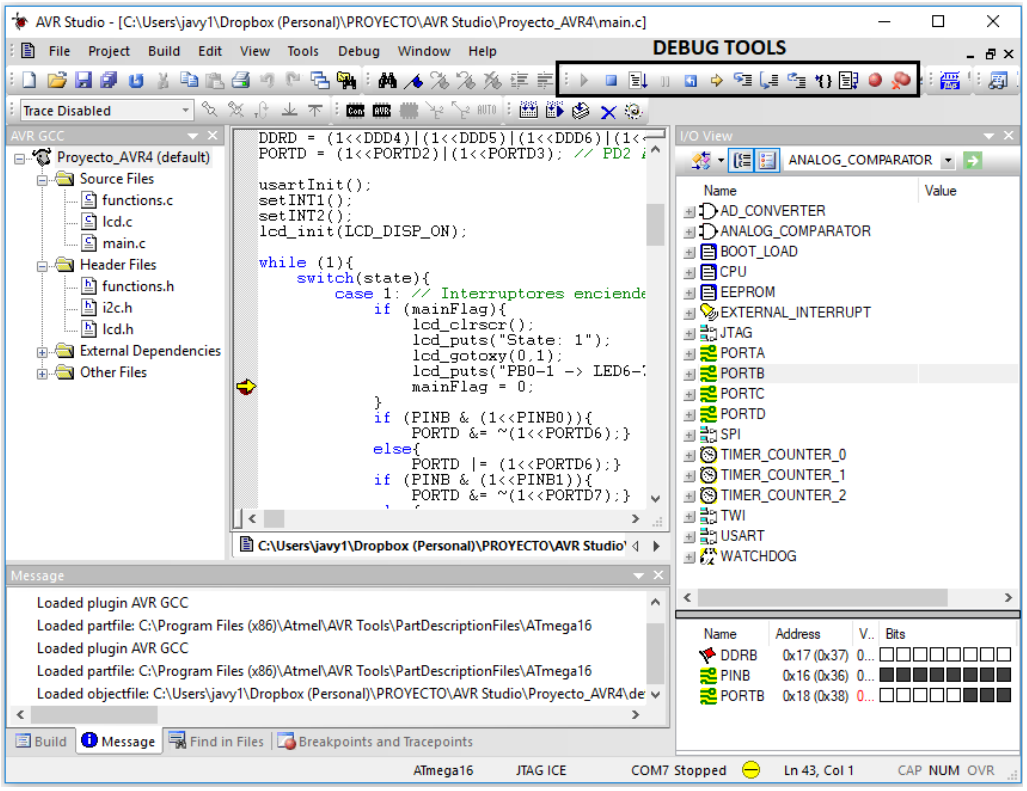
3. DEPURACIÓN DE CÓDIGO

Para que el programa permita iniciar la depuración de cualquier código, es necesario seleccionar una serie de comandos en la ventana de configuración del proyecto, pestaña "Custom Options". Si los comandos "-gdwarf-2" y "gstrict-dwarf" no aparecen, estos se pueden añadir tecleándolos en el espacio y pulsando el botón "Add". En esta misma ventana se deben seleccionar las "External tools". Toda esta configuración se muestra en la Captura 6.



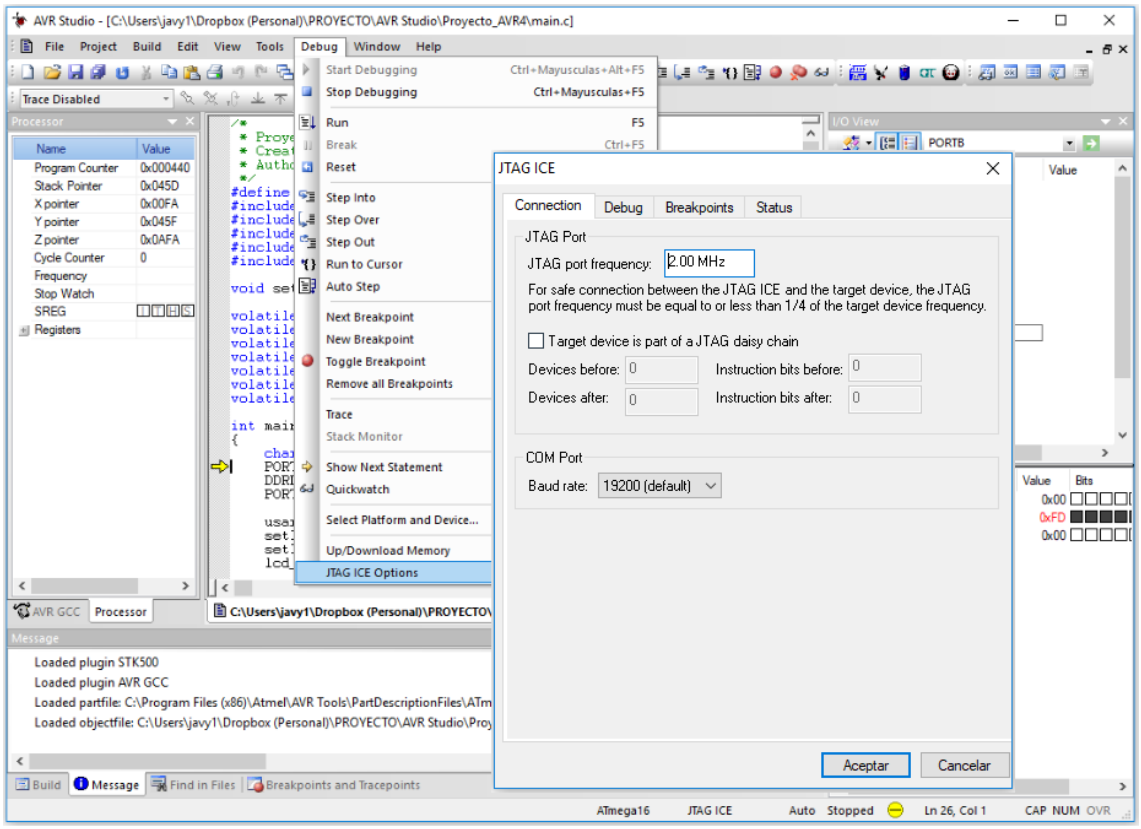
Captura 6. Opciones de proyecto. Añadir comandos.

El entorno consta de las barras de herramientas genéricas en la parte superior, la ventana central del editor de texto y ventanas laterales configurables para depuración, vista de registros del microcontrolador, variables del programa, etc.



Captura 7. Modo depuración. Herramientas y visualización de registros.

Una vez iniciada la depuración, se puede configurar la frecuencia de comunicación con el depurador, teniendo en cuenta que debe ser igual o inferior a una cuarta parte de la frecuencia a la que esté funcionando el microcontrolador, configurada en el “low fuse”.



Captura 8. Modificación de frecuencia del puerto JTAG.

4. VERSIONES DE PROGRAMAS Y ENLACES DE DESCARGA.

AVR Studio 4.18 es la última versión que soporta el depurador JTAG ICE mk1. Esto se debe a que el dispositivo fue declarado obsoleto por Atmel antes de completar su desarrollo para centrarse en una versión mejorada, el JTAG ICE mk2.

Por motivos de compatibilidad entre el programa y el programador, las librerías y el compilador utilizados, así como el programa AVR Studio, deben ser unas versiones concretas y que están disponibles en los siguientes enlaces:

- El software AVR STUDIO 4.18 se puede descargar de forma gratuita en la siguiente dirección web: https://www.microchip.com/mplab/avr-support/avr-and-sam-downloads-archive

- El compilador AVR-GCC, disponible en el programa WinAVR versión 20090313. Como se trata de un software libre, está disponible de forma gratuita en el enlace: https://sourceforge.net/projects/winavr/files/WinAVR/20090313/
- AVR Toolchain versión 3.4.3.1573. Es una colección de herramientas y librerías necesarias para crear aplicaciones en AVR Studio. Se puede descargar en el enlace: https://www.mikrocontroller.net/articles/Atmel_Studio

ANEXO II

PROGRAMA EJEMPLO PARA REVISIÓN DE TODOS LOS MÓDULOS

El objetivo de este anexo es mostrar la parte principal de un programa que haga una demostración del funcionamiento básico de todas las posibilidades que ofrece la placa de desarrollo. El resto de funciones se han definido en los capítulos DESCRIPCIÓN Y PROGRAMACIÓN DE MÓDULOS I2CI2C y DESCRIPCIÓN Y PROGRAMACIÓN DEL RESTO DE DISPOSITIVOS.

El programa principal dispone de cuatro estados:

- Estado 1: Se lee el estado de los interruptores PB0 y PB1 y se muestra dicho estado en los leds PD6 y PD7 respectivamente. Demuestra el uso de entradas, salidas e interrupción externa para propósitos generales.
- Estado 2: La entrada analógica ADC0 controla el brillo del led PD4 utilizando PWM modo rápido.
- Estado 3: Cada tecla del teclado tiene asociada una frecuencia que actúa sobre el zumbador como notas musicales utilizando PWM en modo fase y frecuencia correcta. Demuestra la utilización de entradas mediante I2C y funcionamiento del modo PWM modo fase y frecuencia correcta.
- Estado 4: Se muestra en el display de 7 segmentos la tecla pulsada en el teclado. Demuestra la utilización de entradas y salidas mediante protocolo I2C.

El display LCD muestra en cada momento el estado activo y una breve información sobre dicho estado.

Independientemente del estado activo, el pulsador PD3 asociado a la interrupción externa INT1 cambia al siguiente estado. El pulsador PB2, asociado a la interrupción externa INT2, invierte el estado de los leds PD6 y PD7.

Si se activa el interruptor USART, la comunicación con el PC pasa del programador al microcontrolador, pudiendo cambiar el estado del programa principal desde el PC tecleando el número del estado que se desee, o las teclas "+" o "-", para pasar al siguiente o anterior estado respectivamente. En este modo, las funciones de depuración o programación no están disponibles, y para restablecerlas es necesario desactivar el interruptor USART para cambiar la comunicación al programador.

```
volatile char datoUsart = '0';
volatile char nextState = 1;
volatile char state = 1;
volatile char mainFlag = 1;  // Este flag activa la configuración inicial de cada estado.
int main(void)
{
	char tecla = ' ';
	// Configuración de registros para interruptores PB0 y PB1,
	// leds PD6 y PD7 e interrupciones externas INT0, INT1 e INT2
	PORTB = (1<<PORTB0)|(1<<PORTB1)|(1<<PORTB2); // INPUT PULL-UP
	DDRD = (1<<DDD4)|(1<<DDD5)|(1<<DDD6)|(1<<DDD7); // OUTPUT
	PORTD = (1<<PORTD2)|(1<<PORTD3); // INPUT PULL-UP
	setINT1();
	setINT2();
	usartInit();
	lcd_init(LCD_DISP_ON);
	while (1) {
		switch(state) {
			case 1: // Interruptores PB0 y PB1 actúan sobre los
				// leds PD6 y PD7 respectivamente.
				if (mainFlag){
					lcd_clrscr();
					lcd_puts("State: 1");
					lcd_gotoxy(0,1);
					lcd_puts("PB0-1 -> LED6-7");
					mainFlag = 0;
				}
				if (PINB & (1<<PINB0)){
					PORTD &= ~(1<<PORTD6);}
				else{
					PORTD |= (1<<PORTD6);}
				if (PINB & (1<<PINB1)){
					PORTD &= ~(1<<PORTD7);}
```

```
                else{
                        PORTD |= (1<<PORTD7);}
                break;
        case 2: // PWM led_PD4 con entrada analógica
                if (mainFlag){
                        lcd_clrscr();
                        lcd_puts("State: 2");
                        lcd_gotoxy(0,1);
                        lcd_puts("PWM LED_PD4");
                        pwmInit(1); // PWM LED_PD4
                        adcInit();
                        mainFlag = 0;
                }
                break;
        case 3: // Keypad modo notas musicales con zumbador.
                if (mainFlag){
                        lcd_clrscr();
                        lcd_puts("State: 3");
                        lcd_gotoxy(0,1);
                        lcd_puts("KEYPAD - BUZZER");
                        writePCF(0x0F, 0x40);
                        setINT0();
                        mainFlag = 0;
                }
                if (flagKeypad){
                        getTeclaKeypad();
                        pwmInit(2);
                        setTimer0(500);
                        flagKeypad = 0;
                }
                break;
        case 4: // Display 7 seg. indica tecla pulsada en el teclado.
                if (mainFlag){
                        lcd_clrscr();
                        lcd_puts("State: 4");
                        lcd_gotoxy(0,1);
                        lcd_puts("KEYPAD - DISP7");
                        writePCF(0x0F, 0x40);
                        setINT0();
                        mainFlag = 0;
                }
                if (flagKeypad){
                        tecla = getTeclaKeypad();
                        writePCF(~(getDisp7(tecla)), 0x42);
                        flagKeypad = 0;
                }
                break;
} // Switch
if ((state == 4) && (nextState !=4)){
        writePCF(0xFF, 0x42); // Apaga el display de 7 segmentos.
}
```

```
                state = nextState;
   } // while(1)
} // main()

ISR(INT1_vect)
{
        nextState++;
        if (nextState == 5){nextState = 1;}
        mainFlag = 1;
}
ISR(INT2_vect){
        PORTD &= 0x3F; // Invierte el estado de las salidas PD6 y PD7.
}
ISR(USART_RXC_vect)
{
        datoUsart = UDR;
        if (datoUsart == '1'){nextState = 1;}
        if (datoUsart == '2'){nextState = 2;}
        if (datoUsart == '3'){nextState = 3;}
        if (datoUsart == '4'){nextState = 4;}
        if (datoUsart == '+'){nextState++;}
        if (datoUsart == '-'){nextState--;}
        if (nextState == 5){nextState = 1;}
        if (nextState == 0){nextState = 4;}
        if ((nextState<1)||(nextState>4)){
              envia_cadena_usart("Estado no valido\n");
        }
        if (state != nextState){
              mainFlag = 1;
        }
}
```

PLANOS

PLANO 1. Programador/Depurador JTAG ICE mk1.

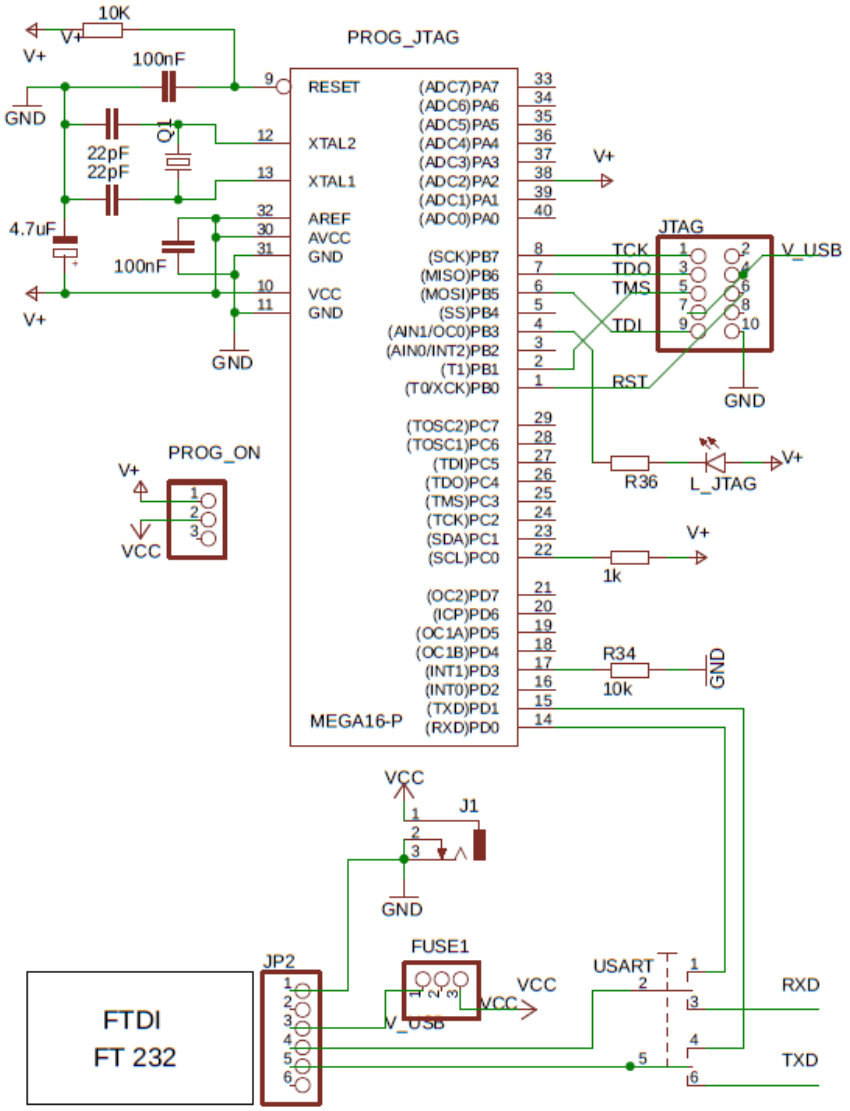
PLANO 2. Microcontrolador ATmega16 con algunos periféricos.

PLANO 3. Expansor de entradas / salidas mediante I2C.

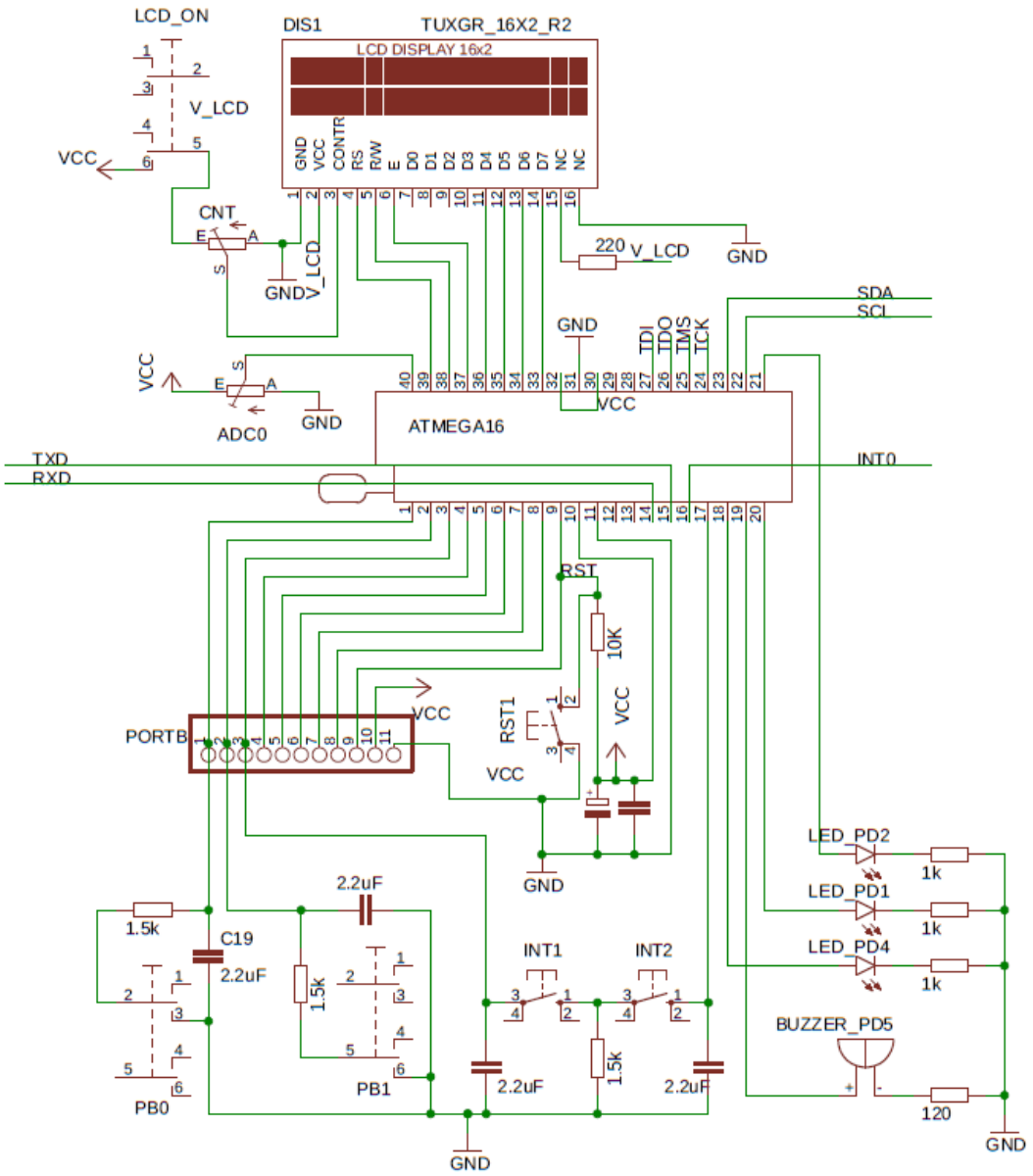
PLANO 4. Ruteado de cara superior de placa de desarrollo.

PLANO 5. Ruteado de cara inferior de placa de desarrollo.

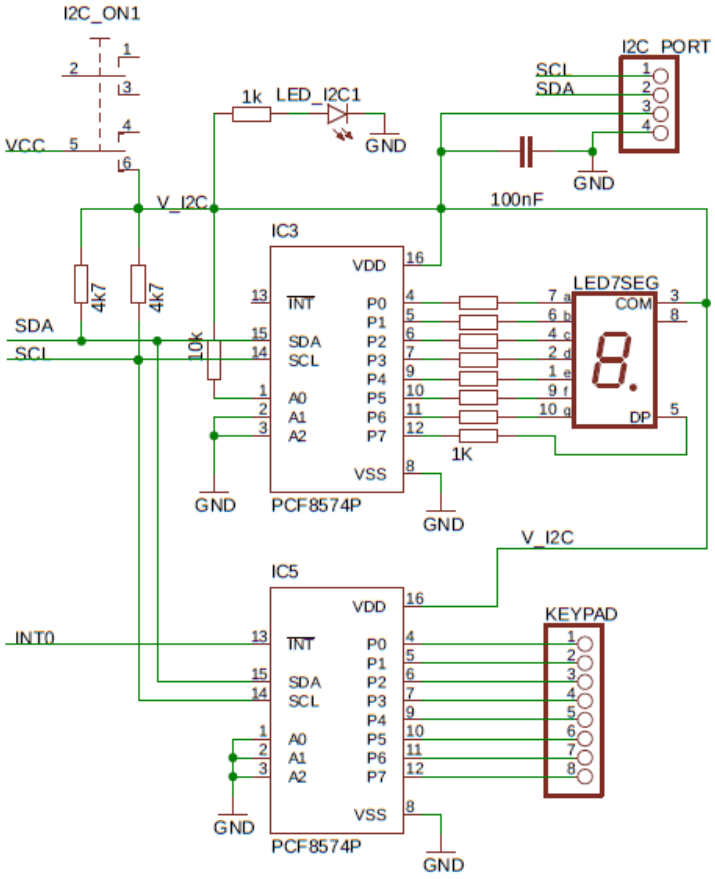
PLANO 6. Ruteado de placa de adaptación TQFP a PDIP.



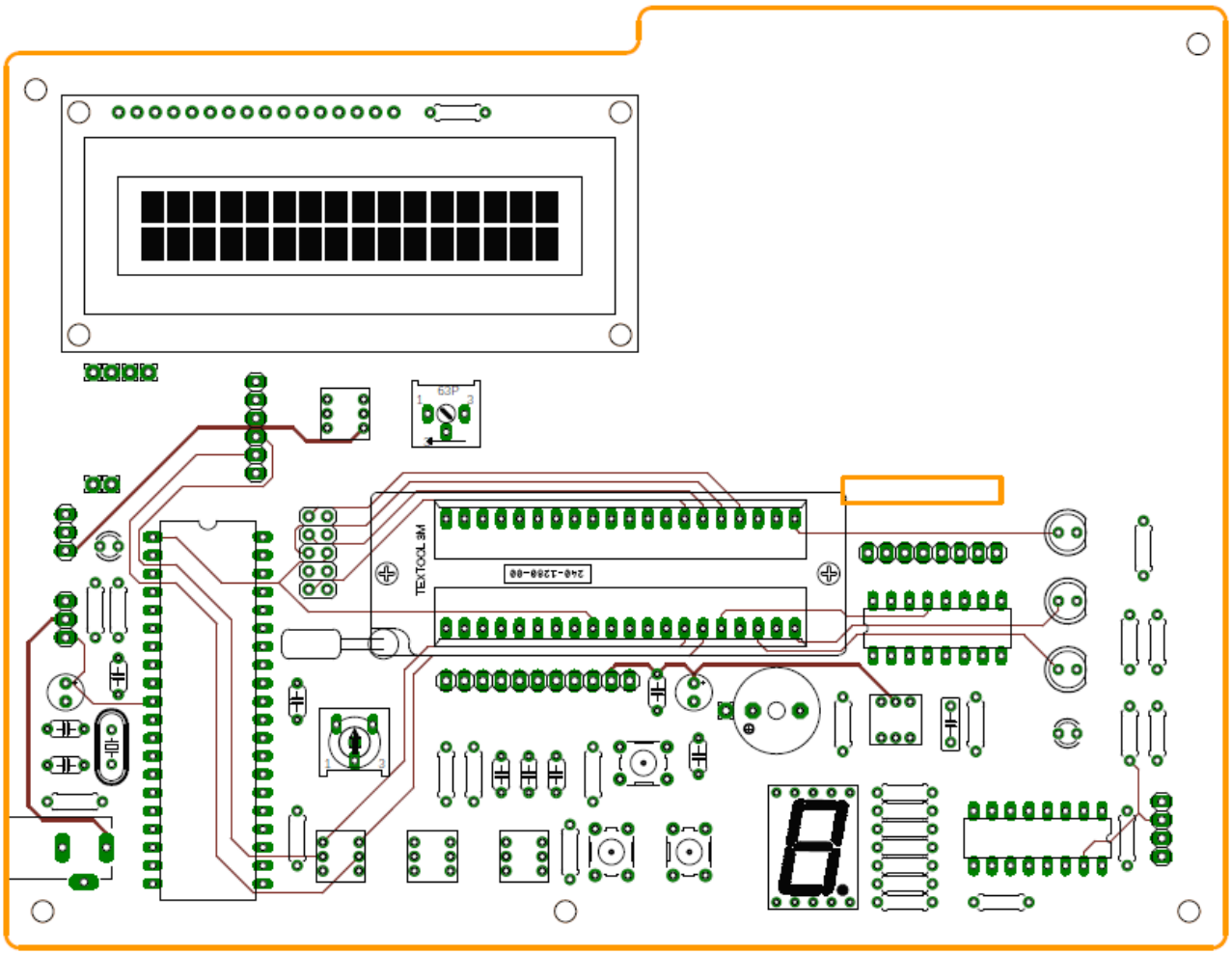
Título Proyecto: Sistema básico de programación y desarrollo de microcontrolador ATMEGA	
Descripción plano: Programador / depurador JTAG ICE mk1 con módulo FTDI para comunicación y alimentación.	
Autor: Javier García Jiménez	Plano Nº 1
Fecha: 15 de junio de 2019	Escala: 1 / 1



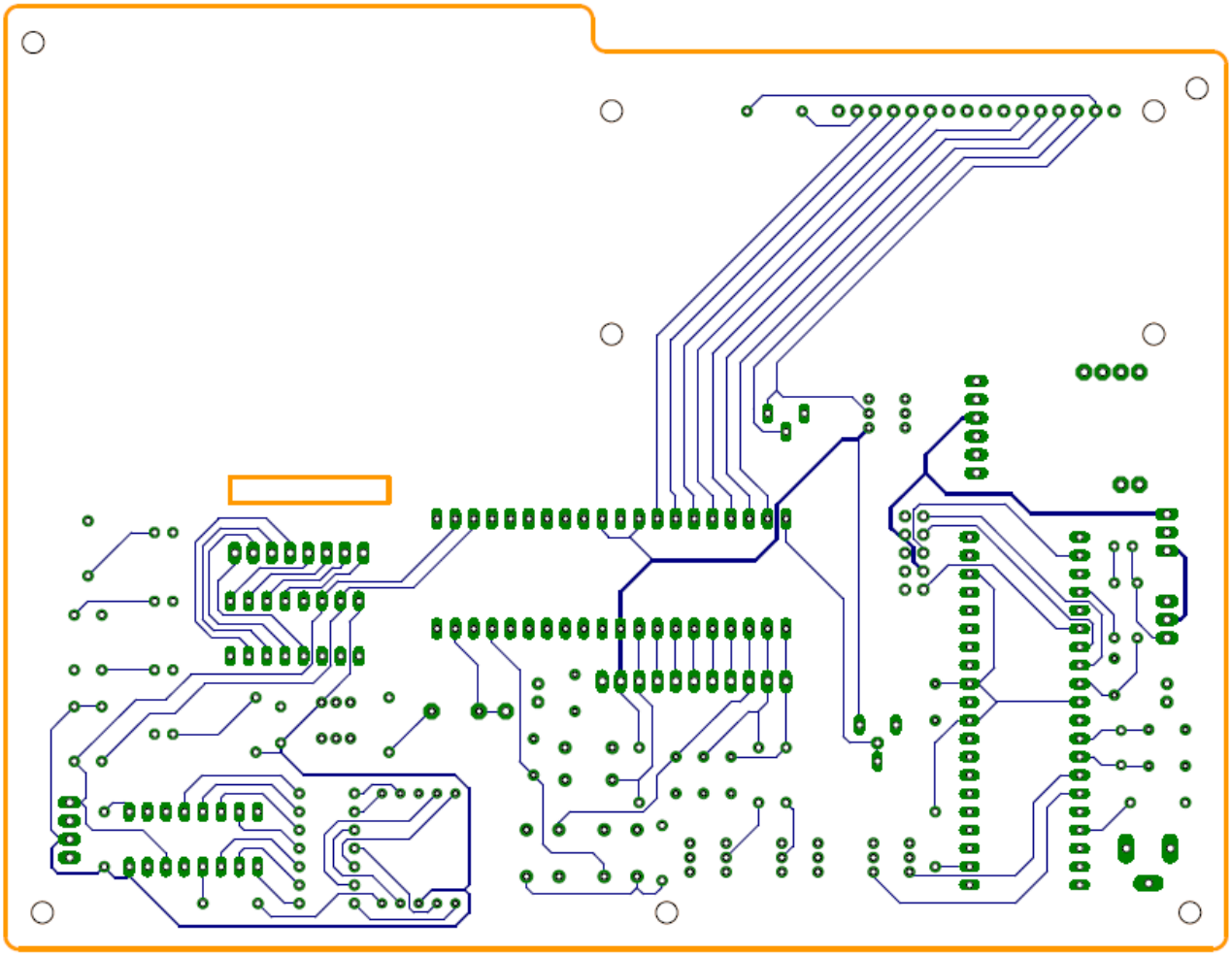
Título Proyecto: Sistema básico de programación y desarrollo de microcontrolador ATMEGA	
Descripción plano: Microcontrolador ATmega16 con algunos periféricos	
Autor: Javier García Jiménez	Plano Nº 2
Fecha: 15 de junio de 2019	Escala: 1 / 1



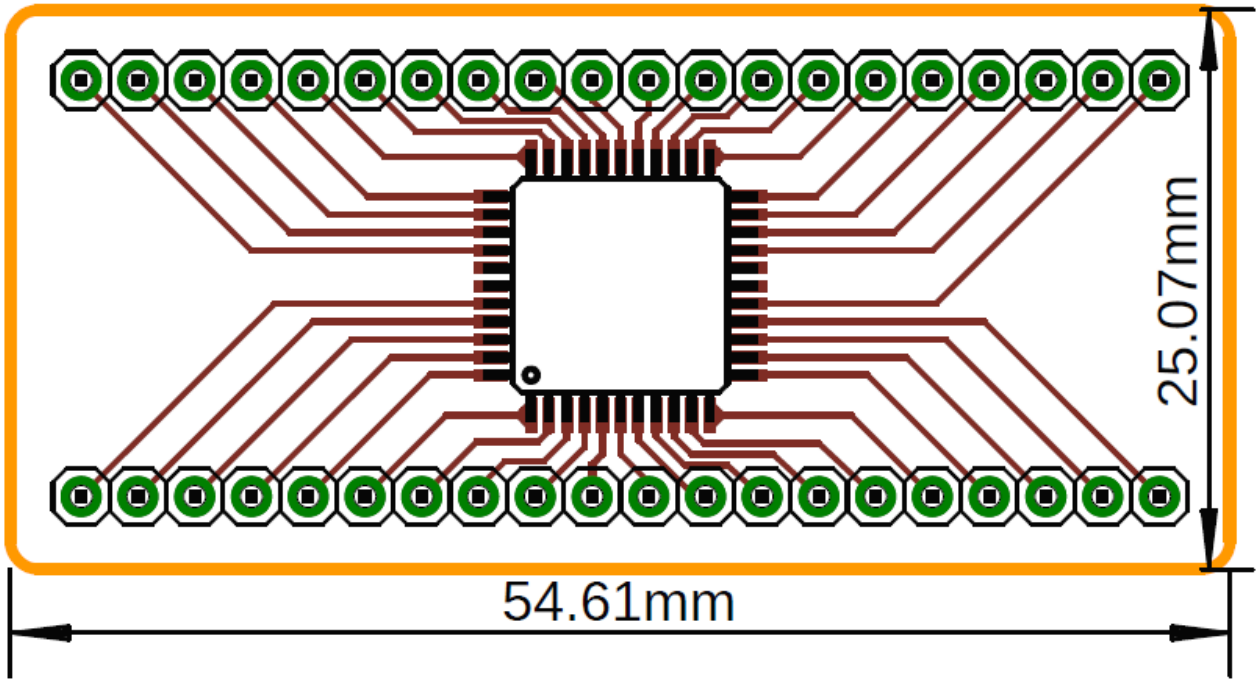
Título Proyecto:	Sistema básico de programación y desarrollo de microcontrolador ATMEGA
Descripción plano:	Expansores de E / S con PCF8574P mediante protocolo I2C
Autor: Javier García Jiménez	Plano Nº 3
Fecha: 15 de junio de 2019	Escala: 1 / 1



Título Proyecto: Sistema básico de programación y desarrollo de microcontrolador ATMEGA	
Descripción plano: Ruteado de cara superior de placa de desarrollo	
Autor: Javier García Jiménez	Plano Nº **4**
Fecha: 15 de junio de 2019	Escala: 1 / 1



Título Proyecto: Sistema básico de programación y desarrollo de microcontrolador ATMEGA	
Descripción plano: Ruteado de cara inferior de planca de desarrollo	
Autor: Javier García Jiménez	Plano Nº **5**
Fecha: 15 de junio de 2019	Escala: 1 / 1



Título Proyecto: Sistema básico de programación y desarrollo de microcontrolador ATMEGA	
Descripción plano: Ruteado de placa de adaptación de encapsulado TQFP a PDIP	
Autor: Javier García Jiménez	Plano Nº **6**
Fecha: 15 de junio de 2019	Escala: **3 / 1**

PRESUPUESTO

Como el objetivo de este proyecto no es la producción en masa de la placa, sino de diseño y montaje propio, no se van a incluir en el presupuesto costes de diseño o de mano de obra, solamente el coste material.

Código	Descripción	Cantidad	Precio (€)	Subtotal (€)
C01	PCB 169 x 131 mm + gastos de envío	1	18,49	18,49
C02	Display LCD 16 x 2	1	1,99	1,99
C03	Teclado matricial 4 x 4	1	1,77	1,77
C04	Módulo FTDI conversor USB-UART	1	3,28	3,28
C05	MCU ATmega16	2	4,18	8,36
C06	Circuito Integrado PCF8574P	2	1,16	2,32
C07	Zócalo ZIF 40 pines	1	2,37	2,37
C08	Zócalo DIP 40 pines	1	0,94	0,94
C09	Zócalo DIP 16 pines	2	0,19	0,38
C10	Crystal 7.3728 MHz	1	1,66	1,66
C11	Conector IDC 10 pines hembra	1	0,40	0,40
C12	Trimmer 10 KΩ	2	1,35	2,70
C13	Interruptor pcb 2 vías	5	0,21	1,05
C14	Pulsador pcb	3	0,11	0,33
C15	Buzzer	1	0,60	0,60
C16	Display 7 segmentos	1	1,70	1,70
C17	Tira 40 pines hembra 2,54 mm	1	0,38	0,38
C18	Tira 40 pines macho 2.54 mm	1	0,33	0,33
C20	Fusible 400 mA	1	0,18	0,18

C21	Resistencia 1 KΩ	12	0,14	1,68
C22	Resistencia 10 KΩ	6	0,14	0,84
C23	Resistencia 4,7 KΩ	2	0,14	0,28
C24	Resistencia 220 Ω	5	0,14	0,70
C25	Condensador electrolítico 4,7 µF	2	0,13	0,26
C26	Condensador cerámico 2,2 µF	4	0,21	0,84
C27	Condensador cerámico 100 nF	4	0,19	0,76
C28	Condensador cerámico 22 pF	2	0,21	0,42
C29	Led blanco 5 mm	3	0,20	0,60
C30	Led blanco 3 mm	2	0,38	0,76
C31	Espaciadores PCB	11	0,10	1,10
C32	Cable USB – mini USB	1	1,39	1,39

Total	58,86
21% I.V.A.	12,36
Total PPTO.	71,22