Proyecto 1. Microcontrolador con CPU AVR

Atuesta Vargas, Julio Cesar. López Sánchez, Luisa Fernanda

(Jcatuestav, lflopezs) @uqvirtual.edu.co

Universidad del Quindío

[[1]](#footnote-1) ***Resumen*—**  En el presente informe se encuentra el procedimiento realizado para la implementación de un microcontrolador con CPU AVR en la herramienta de simulación Proteus, donde se analizó el funcionamiento y la importancia de los microcontroladores

*Palabras Clave*— *AVR, Proteus, resistencia pull-up, tri-estado, puertos, registros, modfile.*

1. INTRODUCCIÓN

Desde la invención de los semiconductores, el desarrollo de la tecnología digital ha dado lugar a dispositivos cada vez más complejos y rápidos, entre ellos los microcontroladores, los cuales se encuentran en nuestro trabajo, nuestra casa, nuestra vida. Se dice que un controlador es un dispositivo que se emplea para manejar uno o varios procesos, hasta hace unos 35 años, los controladores se construían con componentes electrónicos de lógica discreta; posteriormente se emplearon los microprocesadores, apoyados con chips de memoria y dispositivos de E/S sobre una tarjeta de circuito impreso. Desde comienzos de los 90 todos los elementos del controlador se han podido incluir en un solo circuito integrado, el cual recibe el nombre de microcontrolador. Es decir, un microcontrolador es un chip que posee en su interior un microprocesador, memoria de programa, memoria de datos y puertos para comunicarse con el exterior.

1. MARCO TEÓRICO

**Microcontrolador**

Es un [circuito integrado](https://es.wikipedia.org/wiki/Circuito_integrado) programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica. Un microcontrolador incluye en su interior las tres principales unidades funcionales de una [computadora](https://es.wikipedia.org/wiki/Computadora):  [unidad central de procesamiento](https://es.wikipedia.org/wiki/Unidad_central_de_procesamiento), [memoria](https://es.wikipedia.org/wiki/Memoria_(inform%C3%A1tica)) y [periféricos](https://es.wikipedia.org/wiki/Perif%C3%A9rico_(inform%C3%A1tica)) [entrada/salida](https://es.wikipedia.org/wiki/Entrada/salida). Los microcontroladores están diseñados para reducir el costo económico y el consumo de energía de un sistema en particular. Por eso el tamaño de la [unidad central de procesamiento](https://es.wikipedia.org/wiki/Unidad_central_de_procesamiento), la cantidad de memoria y los periféricos incluidos dependerán de la aplicación. El control de un electrodoméstico sencillo como una batidora utilizará un procesador muy pequeño (4 u 8 bits) porque sustituirá a un autómata finito. En cambio, un reproductor de música y/o vídeo digital ([MP3](https://es.wikipedia.org/wiki/MP3) o [MP4](https://es.wikipedia.org/wiki/MP4)) requerirá de un procesador de [32 bits](https://es.wikipedia.org/wiki/32_bits) o de [64 bits](https://es.wikipedia.org/wiki/64_bits) y de uno o más [códec](https://es.wikipedia.org/wiki/C%C3%B3dec) de [señal digital](https://es.wikipedia.org/wiki/Se%C3%B1al_digital) (audio y/o vídeo). El control de un sistema de frenos ABS ([Antilock Brake System](https://es.wikipedia.org/wiki/Antilock_Brake_System" \o "Antilock Brake System)) se basa normalmente en un microcontrolador de [16 bits](https://es.wikipedia.org/wiki/16_bits), al igual que el sistema de control electrónico del motor en un [automóvil](https://es.wikipedia.org/wiki/Autom%C3%B3vil). [1]

**Memorias**

Se ha visto que la memoria en los microcontroladores debe estar ubicada dentro del mismo encapsulado, esto es así la mayoría de las veces, porque la idea fundamental es mantener el grueso de los circuitos del sistema dentro de un solo integrado. En los microcontroladores la memoria no es abundante, aquí no encontrará Gigabytes de memoria como en las computadoras personales. Típicamente la memoria de programas no excederá de 16 K-localizaciones de memoria no volátil (flash o eprom) para contener los programas. La memoria RAM está destinada al almacenamiento de información temporal que será utilizada por el procesador para realizar cálculos u otro tipo de operaciones lógicas. En el espacio de direcciones de memoria RAM se ubican además los registros de trabajo del procesador y los de configuración y trabajo de los distintos periféricos del microcontrolador. Es por ello que en la mayoría de los casos, aunque se tenga un espacio de direcciones de un tamaño determinado, la cantidad de memoria RAM de que dispone el programador para almacenar sus datos es menor que la que puede direccionar el procesador. El tipo de memoria utilizada en las memorias RAM de los microcontroladores es SRAM, lo que evita tener que implementar sistemas de [refrescamiento](https://es.wikipedia.org/wiki/Refrescamiento_de_memoria) como en el caso de las computadoras personales, que utilizan gran cantidad de memoria, típicamente alguna tecnología DRAM. A pesar de que la memoria SRAM es más costosa que la DRAM, es el tipo adecuado para los microcontroladores porque éstos poseen pequeñas cantidades de memoria RAM. [1]

**La Unidad Central de Proceso o CPU**

Es el elemento más importante del microcontrolador y determina sus principales características, tanto a nivel de hardware como de software. Se encarga de direccionar la memoria de instrucciones, recibir el código OP de la instrucción en curso, su decodificación y la ejecución de la operación que implica la instrucción, así como la búsqueda de los operandos y el almacenamiento del resultado.  
Existen tres tipos de CPU en cuanto a la forma de “procesar” las instrucciones:

* CISC: Un gran número de procesadores usados en los microcontroladores están basados en la filosofía CISC (Computadores de Juego de Instrucciones Complejo). Disponen de más de 80 instrucciones de máquina en su repertorio, algunas de las cuales son muy sofisticadas y potentes, requiriendo muchos ciclos para su ejecución. Una ventaja de los procesadores CISC es que ofrecen al programador instrucciones complejas que actúan como macros. [2]
* RISC: Tanto la industria de los computadores comerciales como la de los microcontroladores, están decantándose hacia la filosofía RISC (Computadores de Juego de Instrucciones Reducido). En estos procesadores el repertorio de instrucciones de máquina es muy reducido y las instrucciones son simples y, generalmente, se ejecutan en un ciclo. La sencillez y rapidez de las instrucciones permiten optimizar el hardware y el software del procesador. [2]
* SISC: En los microcontroladores destinados a aplicaciones muy concretas, el juego de instrucciones, además de ser reducido, es "específico"; o sea, las instrucciones se adaptan a las necesidades de la aplicación prevista. Esta filosofía se ha bautizado con el nombre de SISC (Computadores de Juego de Instrucciones Específico). [2]

**Puertos de Entrada y Salida**

La principal utilidad de las patas que posee el chip que contiene un microcontrolador es soportar las líneas de E/S que comunican al computador interno con los periféricos exteriores y, según los controladores de periféricos que posea cada modelo de microcontrolador, se destinan a proporcionar el soporte a las señales de entrada, salida y control. Todos los microcontroladores destinan algunas de sus patillas a soportar líneas de E/S de tipo digital, esto es, todo o nada. Por lo general, estas líneas se agrupan de ocho en ocho formando Puertos. Las líneas digitales de los Puertos pueden configurarse como Entrada o como Salida, cargando un 1 ó un 0 en el bit correspondiente de un registro destinado a su configuración. [2]

1. ANÁLISIS Y PROCEDIMIENTO

Inicialmente se debe conocer la arquitectura de un microcontrolador basado en una CPU de 8 bits (Figura 1).

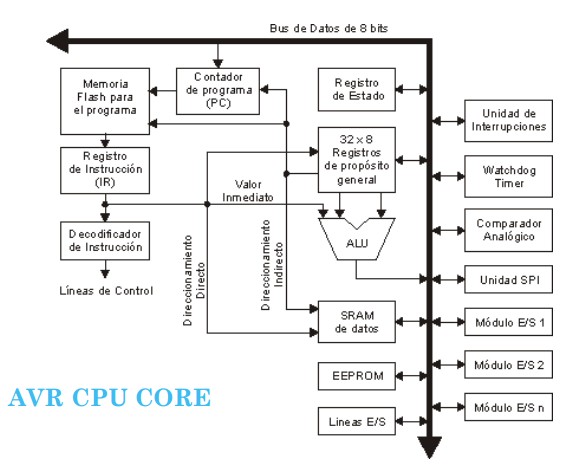


Fig.1 Arquitectura CPU Core [1]

Teniendo la arquitectura del microcontrolador se procede a implementar el circuito interno de los pines que componen uno de los puertos de entrada/salida del microcontrolador, para esto se utiliza la herramienta de simulación Proteus. En la figura 2 se muestra el circuito de uno de estos pines.

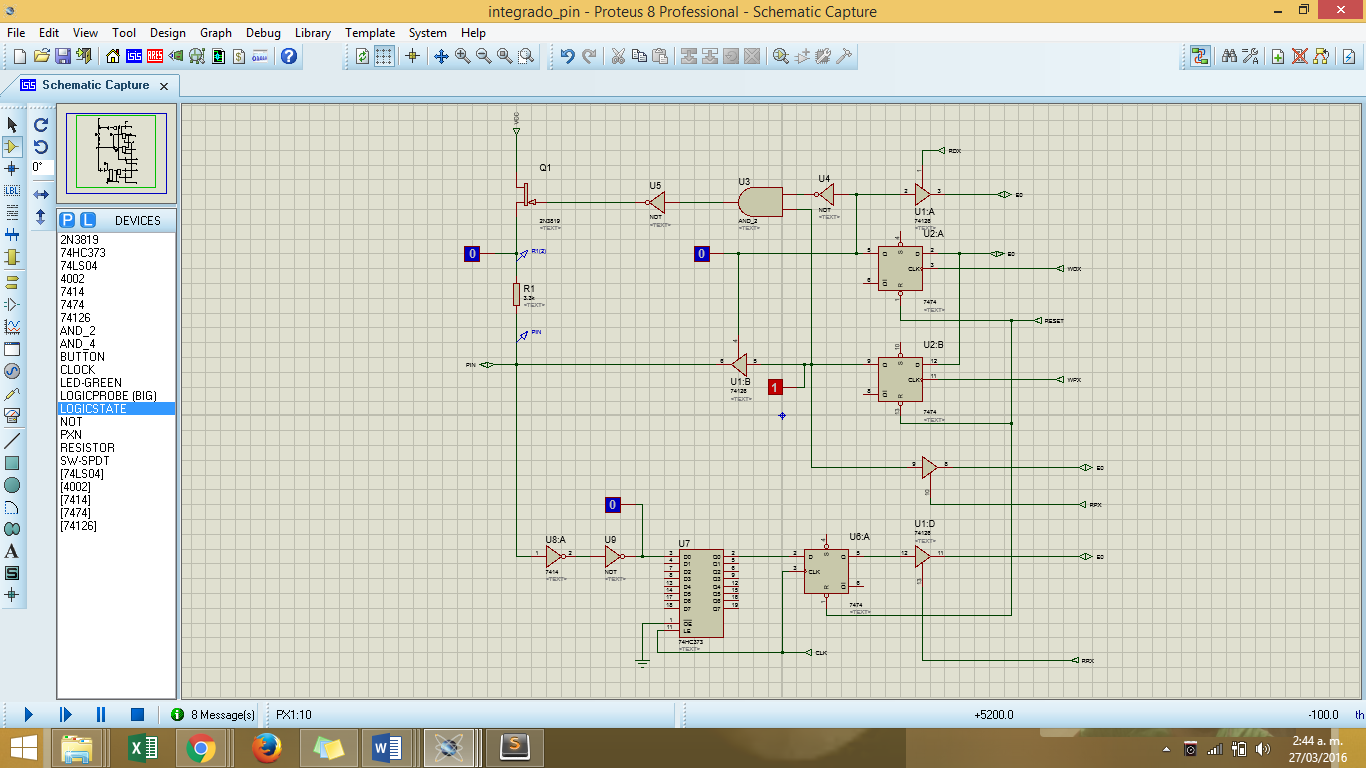


Fig.2 Circuito interno del pin

Este circuito se utiliza para configurar el pin como entrada o salida (Figura 3), además de esto la salida o configuración Pull-up (Figura 4), y por último la sincronización y lectura del puerto (Figura 5).

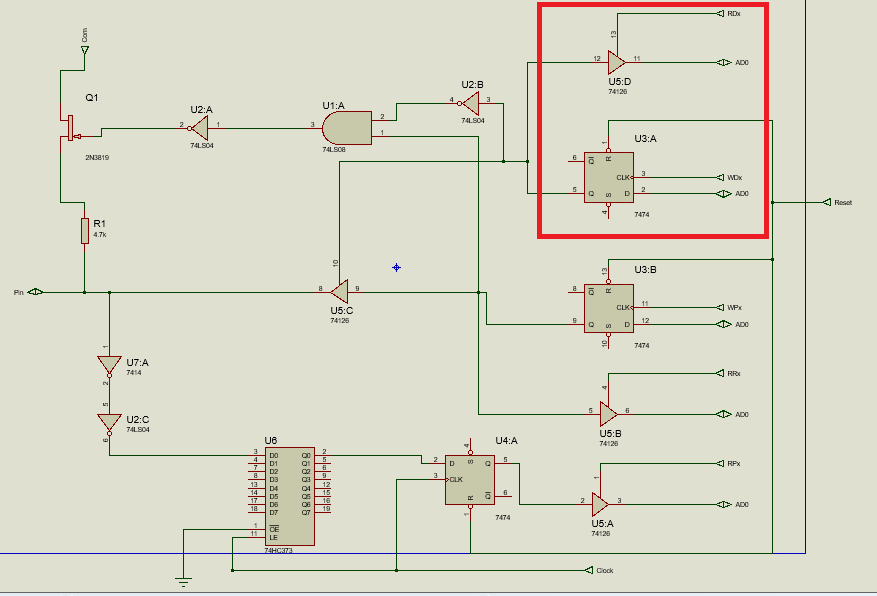


Fig.3 Registro de configuración: Entrada o salida.

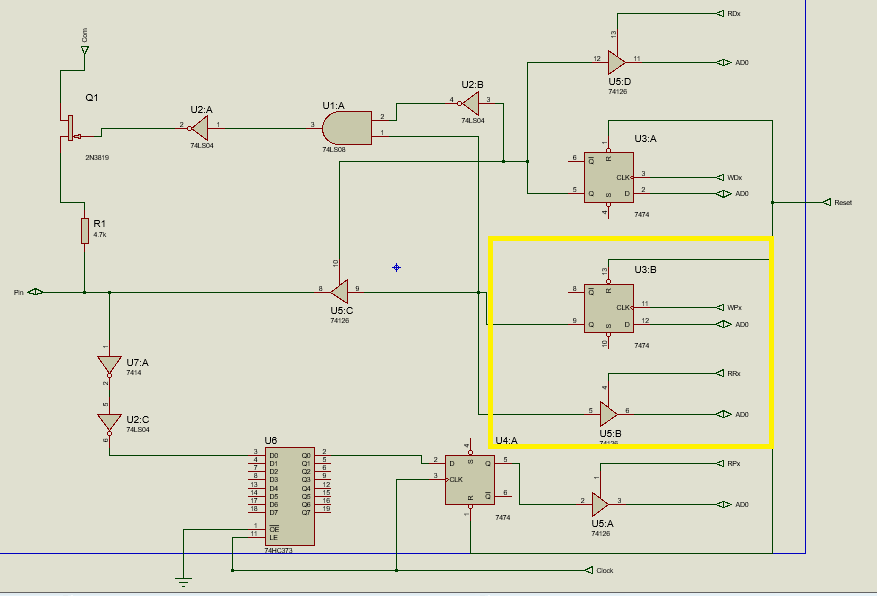


Fig.4 Salida o configuración Pull-up.

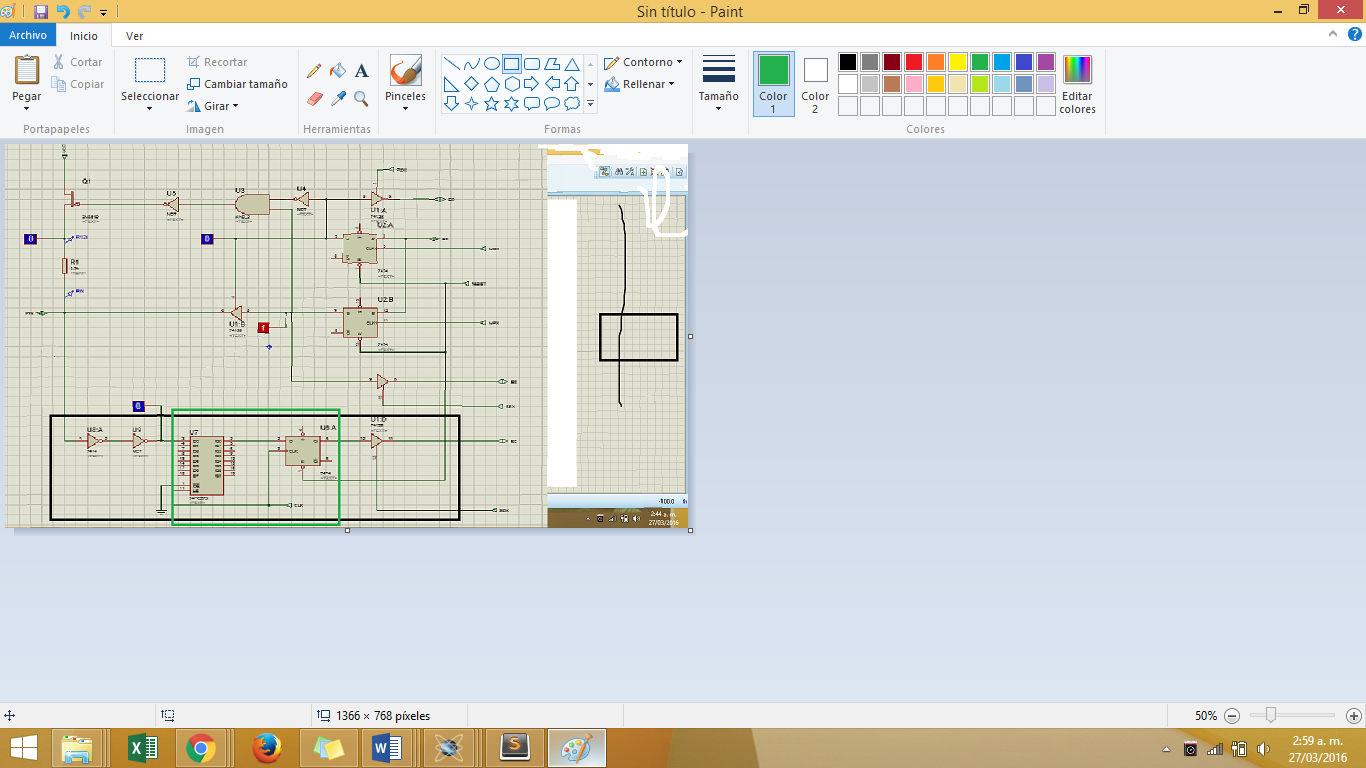


Fig.5. Sincronización (verde) y lectura del puerto.

Se realizan pruebas sobre el funcionamiento del esquemático del pin, al variar las posibles combinaciones de los registros WDX y WPX se obtienen los resultados de la tabla 1.

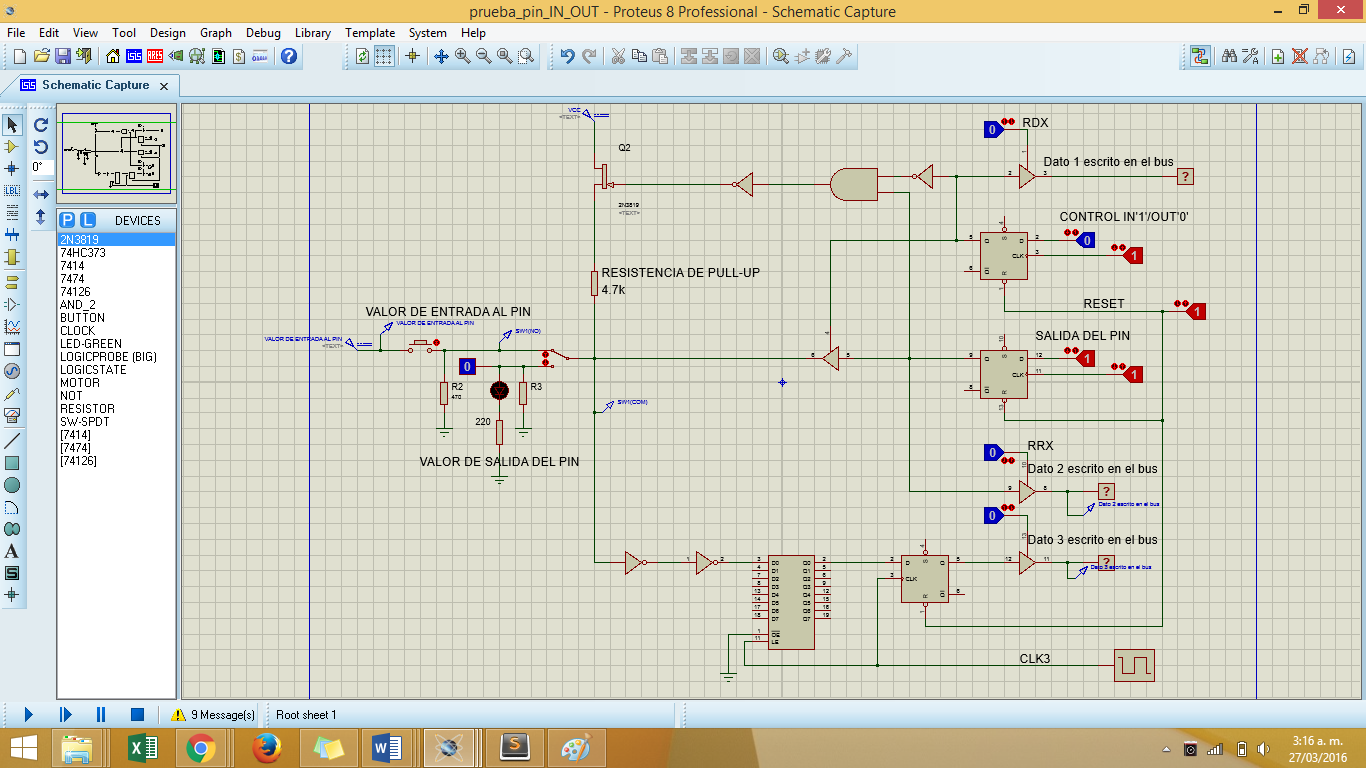


Fig. 6 Prueba del circuito interno del Pin.

Tabla 1. Valores obtenidos en el pin

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| WDX | 0 | 0 | 1 | 1 |
| WPX | 0 | 1 | 0 | 1 |
| Pin | 1 | 0 | 0 | 1 |

Los valores obtenidos en el pin corresponden en las dos primeras combinaciones a una ata impedancia en entre el pin y los registros WDX y WPX, el valor obtenido en estas se consigue al activar o desactivar la resistencia de pull-up por medio del mosfet, en las dos combinaciones siguientes el puerto está configurado como salida, y el valor obtenido en el pin es el dato cargado en el registro WPX.

En los resultados obtenidos en la tabla 1 se deben considerar dos condiciones.

Primero, para obtener estos valores se debe configurar siempre en primer lugar el registro WDX (en código esto es lo mismo que declarar en primer lugar el DDRX y luego el PORTX), y preferiblemente haber activado el reset de los registros para evitar que un estado anterior pueda influir en el funcionamiento del puerto.

Segundo, al conectar una carga al pin se puede producir el efecto de una resistencia de PullDown, esto cambiaría el valor del primer estado, de forma que solo se obtendría un uno siempre que ambos registros (WDX y WPX) tengan cargado en su salida un valor alto, Además los valores resistivos de la carga pueden hacer que se obtenga un valor lógico indeterminado, este efecto se observó en un diodo led con una resistencia a tierra de un valor superior a los 420Ω.

Una vez configurado el pin, y probado su correcto funcionamiento se procede a realizar el puerto completo, el cual se conforma de ocho pines idénticos donde cada pin se conecta a una línea independiente del bus de datos proveniente del microprocesador, además de los bits de control de los registros, éstos últimos son comunes para todo el puerto.

En la figura 7, se muestra la implementación, donde se tienen los ocho pines configurados como se mostró anteriormente, además de estos las compuertas lógicas de control de los registros.

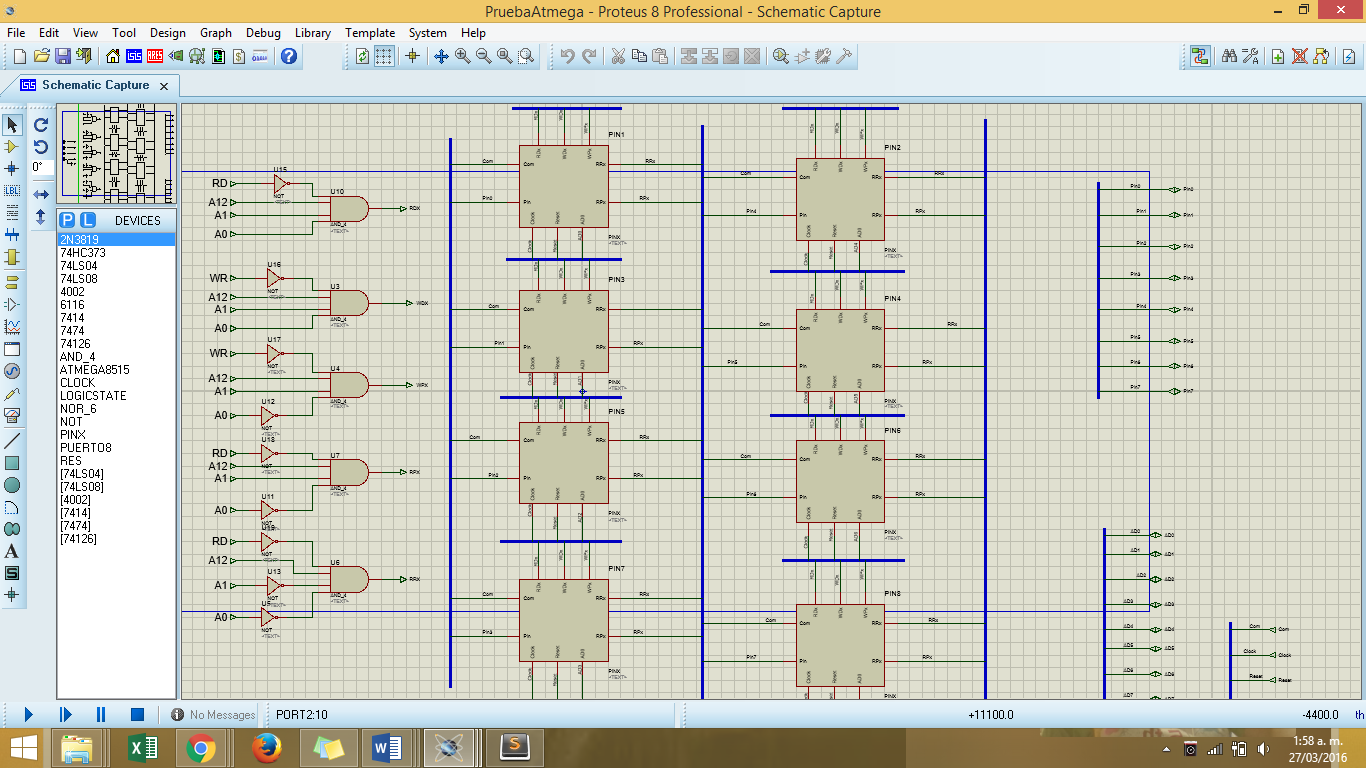


Fig.7 Implementación del puerto con las compuertas lógicas.

De la figura anterior se puede inferir que:

Es posible configurar de forma independiente cada pin como entrada o salida ya que el bit que indica este estado se extrae de la línea del bus de datos, la entrada “D” de ambos registros (WDX y WPX) también se conecta a la misma línea, por lo tanto cuando se desea obtener una con las configuraciones uno-cero o cero-uno de la tabla 1 es necesario tener en cuenta el orden con el que se cargan los registros.

Debido a la conexión física que presenta el puerto no es posible leer el valor que entra en un solo pin a la vez, ya que la opción de tomar el dato se activa con el bit RRX el cual es común en todos los pines, si se desea el valor de un solo pin debe extraerse del bus de datos por medio de código, se implementa la conexión física de esta forma ya que es más fácil obtener el dato en particular en la programación, además serían necesarias más líneas de dirección para controlar la lectura de cada pin de forma independiente.

En la tabla 2, se muestra las direcciones enviadas a al circuito lógico combinacional, así mismo se envían los registros RD y WR, tenga en cuenta que éstos últimos se activan con un nivel bajo.

Tabla 2. Combinaciones que activan los bits de control

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  | 1 | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 0 | 1 |
|  | 1 | 1 | 1 | 1 | 0 |
|  | 1 | 1 | 0 | 1 | 0 |
|  | 1 | 0 | 0 | 1 | 0 |

Donde se tiene que:

* : Write DDRx El estado E/S del pin
* : Write PORTx. El cual carga el dato que se va poner a la salida cuando el pin escribe.
* : Read PORTx PIN Recupera el valor enviado al pin en un estado anterior
* : Read DDRx Recupera el estado anterior E/S del pin
* : Read PORTx REGISTER Lee el valor a la entrada del Pin

Después de tener listo el puerto, se procede a realizar las pruebas con el ATMEGA 8515, cargando en este un archivo .hex la conexión y los resultados obtenidos se muestran en la figura 8.

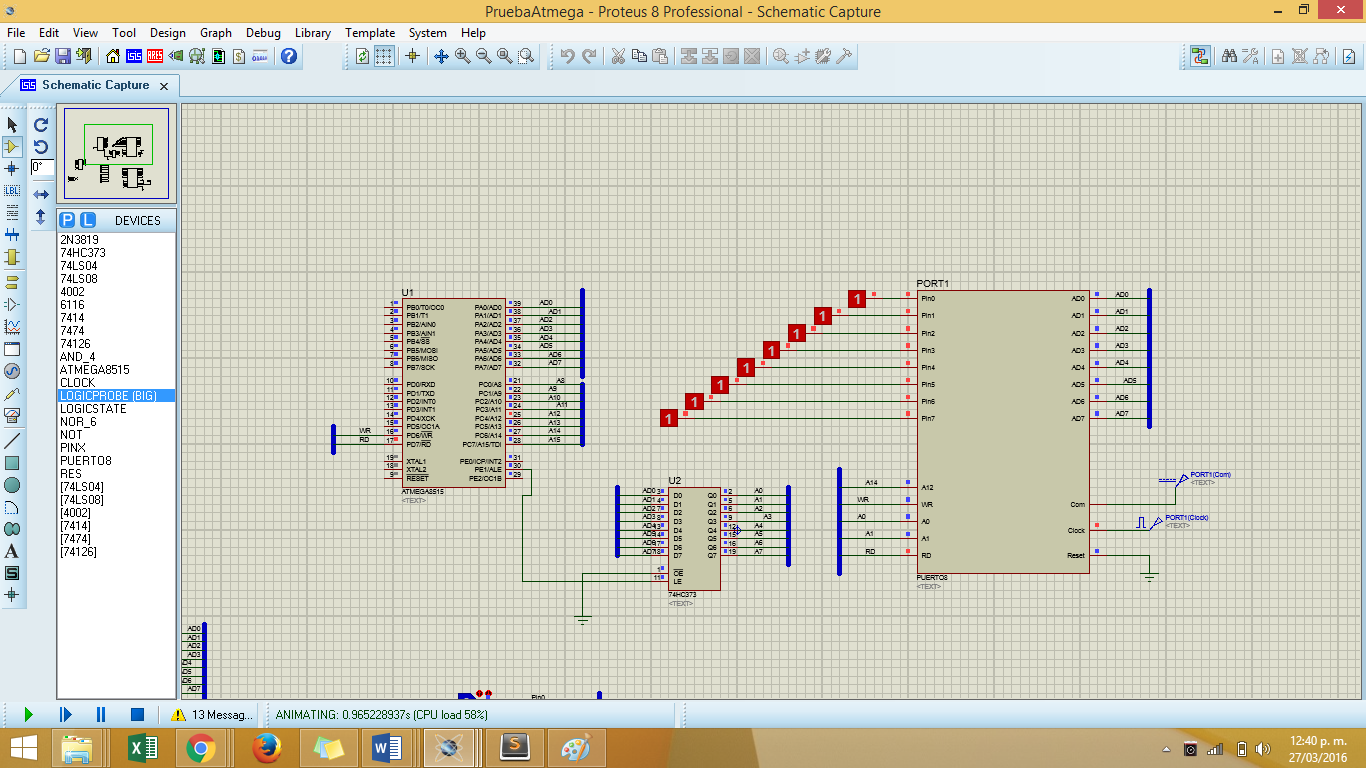


Fig.8 Prueba del microcontrolador con el atmega 8515.

De igual manera para tener una noción más clara del funcionamiento de este se usa un código que incluye un puerto configurado como entrada y otro como salida.

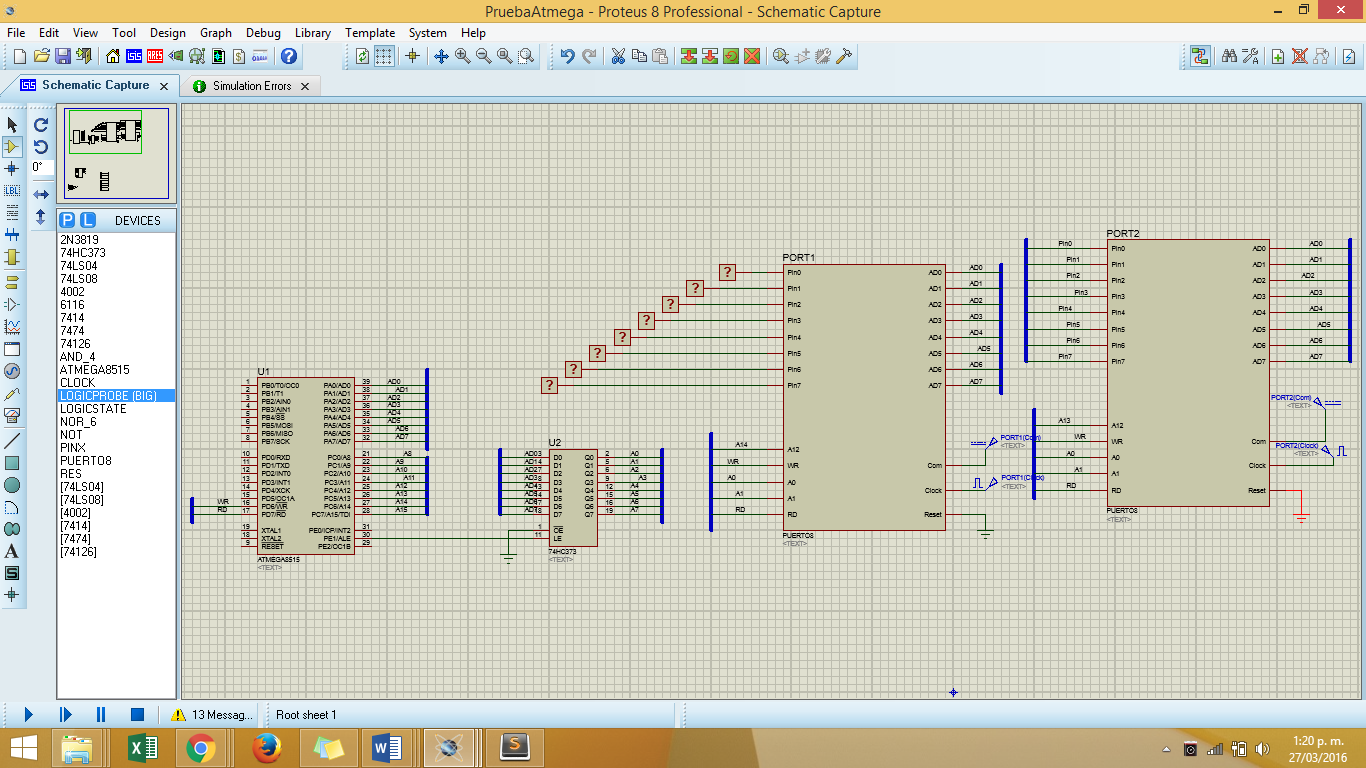


Fig.9. Prueba de dos puertos

Se usa una memoria de las librerías de proteus como la RAM del sistema, de esta se debe controlar el valor de las líneas de dirección ya que el número de estas líneas determinan el tamaño de la memoria RAM y cada combinación que se da en las líneas representa un espacio en la memoria, por eso se ha reservado un espacio para las combinaciones de las líneas de control.

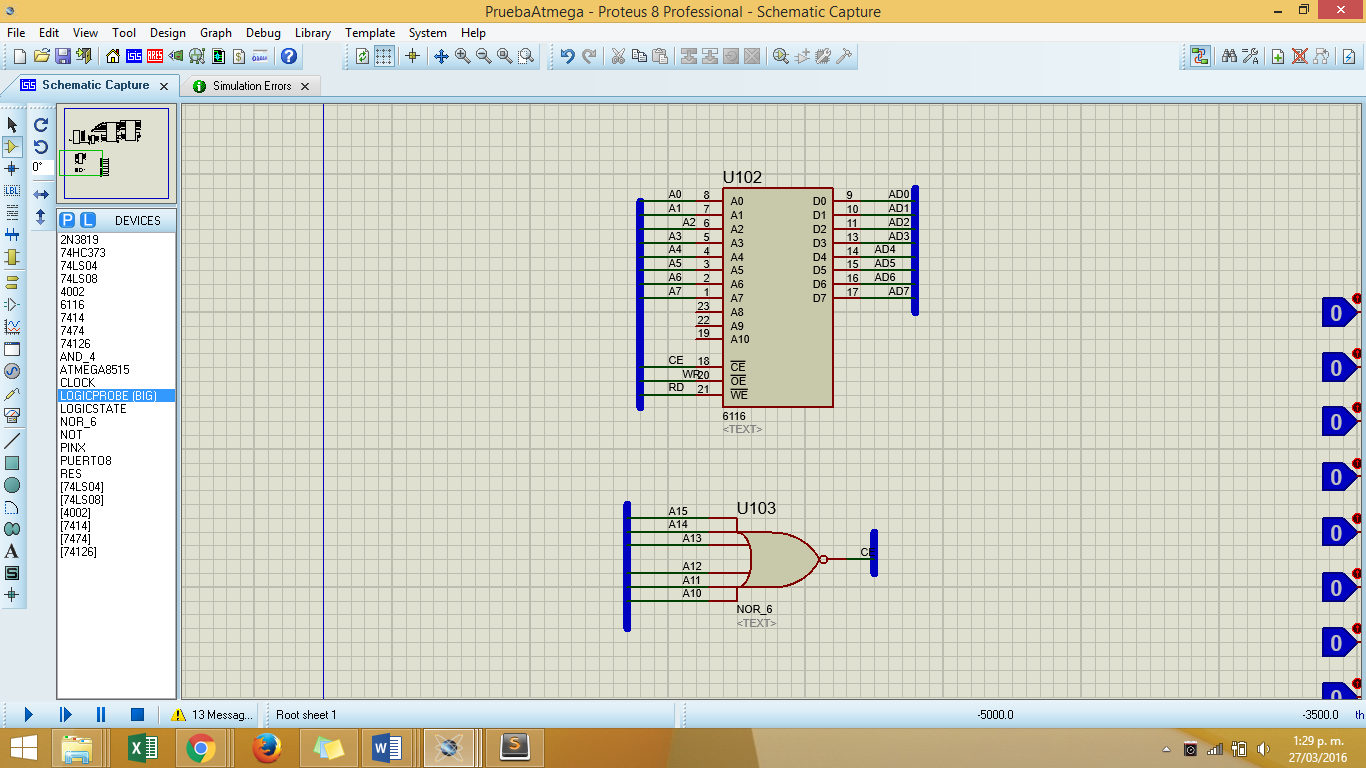


Fig.10. Memoria RAM con compuerta de control de líneas de dirección.

Finalmente se configura una salida tipo pwm configurado el registro TCCR0, este es un registro de 8 bits en el cual cada bit configura una parte del funcionamiento de la señal del pwm. La toma de decisiones del registro se basa en comparaciones dependiendo del modo de funcionamiento en el que se encuentre el registro, con un valor determinado por el usuario, el OCRX, un registro de 8 bits.

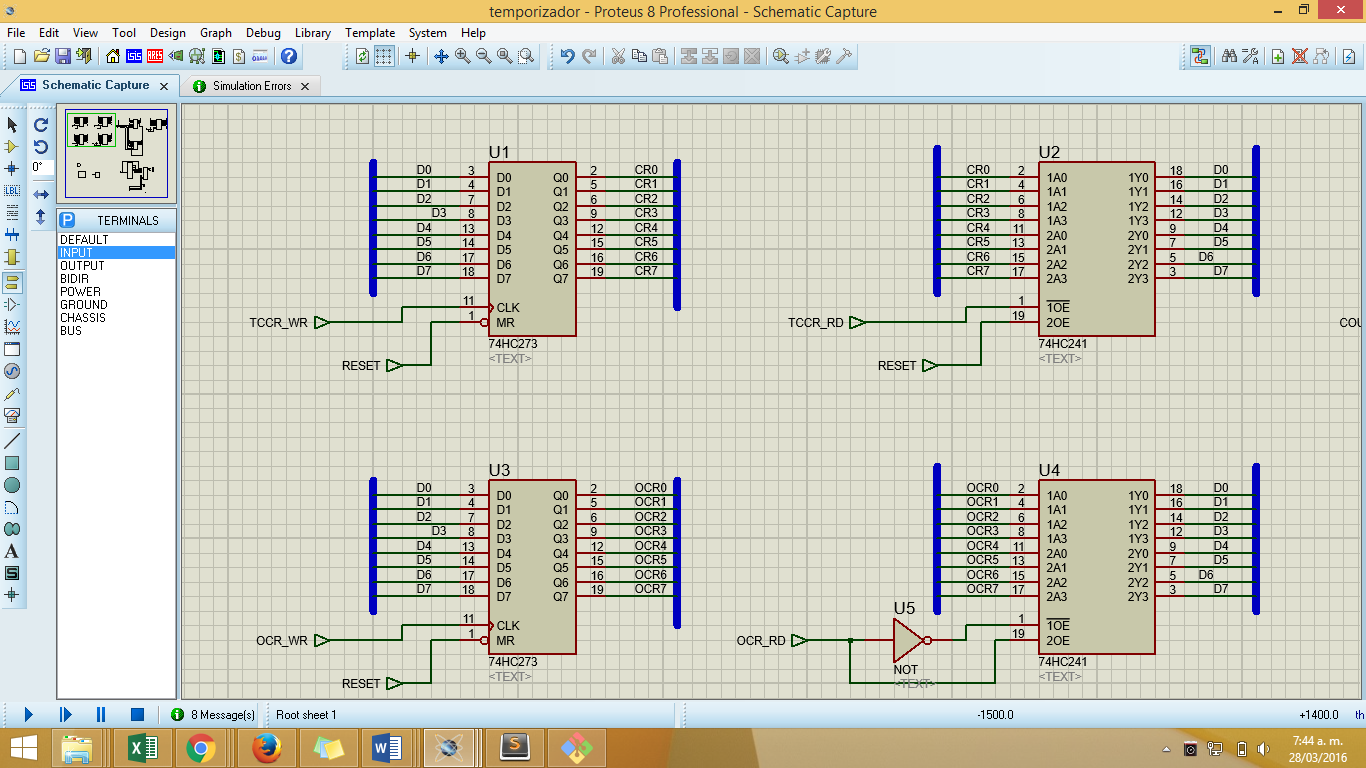


Fig. 11. Configuración del TCCRX o el OCR

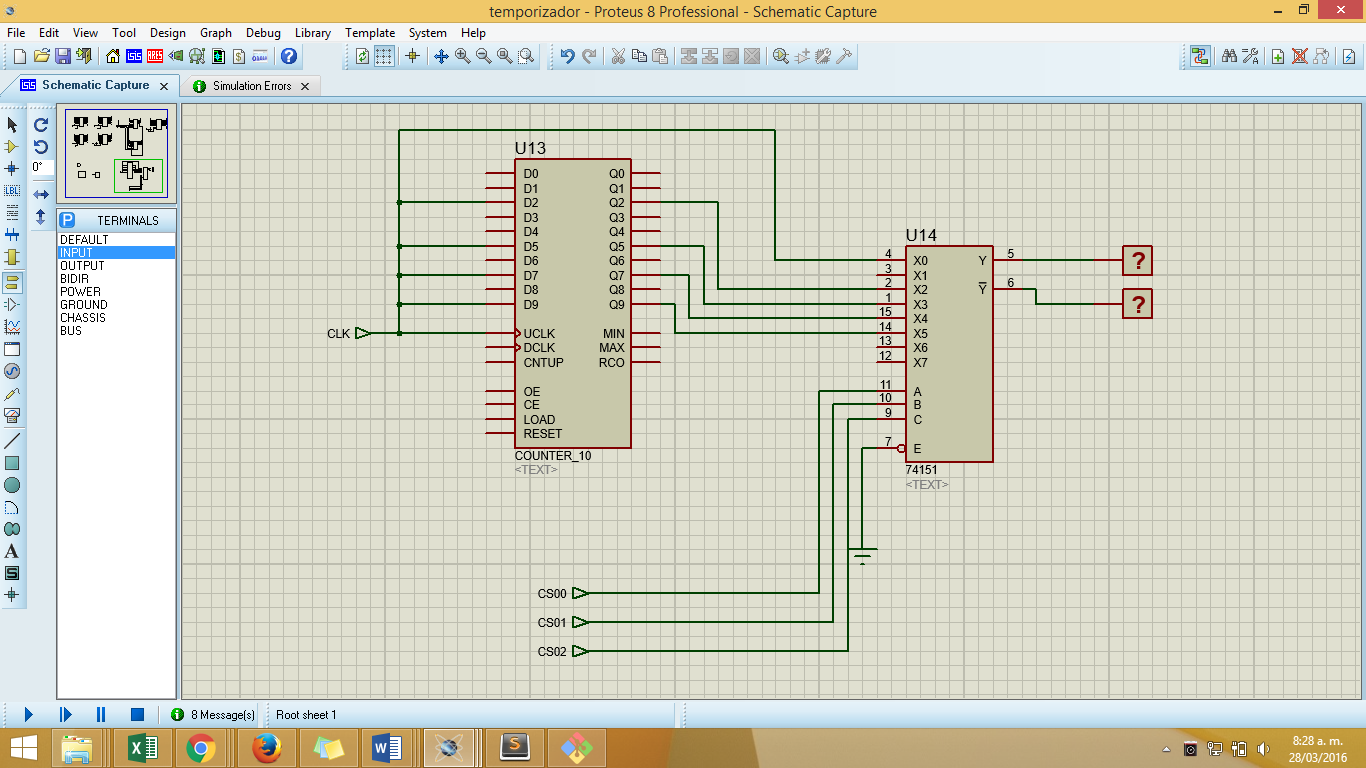


Fig. 12. Pre escalado

El pre escalado permite el cambio de la frecuencia de trabajo del pwm a partir de la frecuencia de reloj del microcontrolador permitiendo tener una señal con menor variación en el tiempo, muy útil para circuitos lentos donde se manejen motores o sistemas de frecuencia baja.

1. CONCLUSIONES

* Los software de programación o de simulación grafica como el Proteus, utilizado en este proyecto, son de vital importancia, ya que permiten simular problemas y sistemas presentados en la vida cotidiana.
* La arquitectura presente en el microcontrolador busca la optimización de tareas específicas, que aunque son esenciales para el funcionamiento de cualquier sistema se realizan consumiendo pocos recursos computacionales, de esta forma el dispositivo crea un equilibrio entre compatibilidad y funcionalidad.
* El tamaño físico del dispositivo se condiciona por la capacidad de éste, principalmente por sus memorias ya que un aumento en el tamaño de la memoria RAM representa una conexión física extra y una cantidad de 2^n registros (flip-flops) para poder almacenar la nueva cantidad de datos disponibles, de forma que se ocupara el espacio necesario por los transistores para crear cada flipflop, es por esto que al crear junturas más estrechas pero igual de eficientes en los semiconductores se permite crear dispositivos digitales más pequeños y potentes.
* La implementación del Pwm permite al microcontrolador integrarse en tareas más avanzadas y especializadas, aunque comunes en el oficio de la ingeniería, por medio del manejo de motores, iluminación, telecomunicaciones, todos los sistemas que requieran conmutación, en la construcción e un conversor ADC y sistemas de control On-Off.

1. REFERENCIAS

[1] **Wikipedia, Colaboradores de. 2016.** Wikipedia. La enciclopedia libre. [En línea] 11 de 02 de 2016. [Citado el: 22 de 03 de 2016.] https://es.wikipedia.org/w/index.php?title=Microcontrolador&oldid=89078691.

[2] **2015.** Aprendiendo-microcontroladores. *wikispaces.* [En línea] 15 de 04 de 2015. [Citado el: 22 de 03 de 2016.] https://aprendiendo-microcontroladores.wikispaces.com/ATMEL+AVR.

[3] Fundamentos y Aplicaciones de lo Microcontroladores AVR ed1, A. Vera Tansama ,W.D. Jimenez. J.E. Cardona. Armenia Quindio 2008,

1. [↑](#footnote-ref-1)