UNIVERSIDAD MAYOR DE SAN ANDRES

FACULTAD DE TECNOLOGIA

CARRERA: ELECTRONICA Y TELECOMUNICACIONES

NIVEL: LICENCIATURA

EXAMEN DE GRADO

TRABAJO DE APLICACION

“MAQUINA EXPENDEDORA AUTOMATIZADA DE BEBIDAS NATURALES PARA EL GIMNASIO DEL DEPARTAMENTO POLICIAL NUMERO 2, DE LA CIUDAD DE LA PAZ "

Postulante: Univ. Aduviri Plata Humberto

La Paz- Bolivia

2016

AGRADECIMIENTOS

Agradezco a Dios por cuidarme, darme salud, y pedirle que siempre me de sabiduría a lo largo de mi vida. A mis padres Sr. Cirilo Aduviri Huaychu y Sra. Damiana Plata que siempre confían en mí y me dan su apoyo incondicional, gracias de corazón.

Gracias a la carrera de Electrónica y telecomunicaciones por darme experiencias únicas e inigualables dentro sus aulas y fuera de ellas.

RESUMEN

En nuestro contexto no existe una máquina expendedora automatizada de bebidas frías naturales accesible a toda la sociedad boliviana, además que responda a nuestras necesidades e interés en lugares de entrenamiento como gimnasios, por ello con este proyecto se quiere subsanar esa necesidad boliviana.

De esta manera impulsamos la generación de productos tecnológicos electrónicos que se adecuen a nuestro contexto; es decir, producción tecnológico boliviano.

El presente proyecto de aplicación tiene por finalidad Implementar un prototipo en base al microcontrolador AVR ATMEGA16, de una máquina expendedora automatizada de bebidas naturales, que se active al introducir monedas y sea de fácil manejo para el usuario final.

Asimismo es importante señalar que para la planificación, organización y ejecución de los circuitos se reutilizaron diversos materiales para reducir el costo de la máquina expendedora automatizada de bebidas naturales.

De esta manera se está generando espacios tecnológicos bolivianos que respondan a nuestras necesidades e intereses, cuidando nuestra madre tierra.

Índice **pagina**

Indice de figuras **Pagína**

Introducción

Gracias al avance de la tecnología hoy se puede realizar una variedad de productos electrónicos que estén diseñados para nuestro entorno y necesidades locales.

En este proyecto de aplicación se implementará un prototipo de una máquina expendedora de bebidas naturales no envasadas, que sea de fácil manejo para el cliente.

Antecedentes

En Bolivia existe máquinas expendedoras de bebidas frías de operación mecánica teniendo que estar asistido por una persona, también existen máquinas expendedoras electrónicas que funcionan con energía eléctrica, que ofrecen una variedad de bebidas envasadas en botella o lata. Vendiendo sin la presencia de un dependiente para cobrar. Periódicamente un empleado repone el producto y recoge el dinero.

En nuestro contexto no existe una máquina expendedora de bebidas naturales, que ofrezca similar servicio de auto venta.

CAPITULO I

1 Planteamiento del problema

En la ciudad de La Paz existe máquinas expendedoras de distintos productos en base a la autogestión monetaria realizada por el cliente brindando un servicio permanente, estos están ubicados en diferentes lugares de esparcimiento, entrenamiento, diversión y tránsito. Cabe resaltar que los "gimnasios" no cuentan con una máquina de bebidas naturales que funcione al introducir monedas, para mantener el equilibrio e hidratación corporal; porque no existe un personal para dicho fin.

No existe una máquina que se adecue a nuestro contexto en cuanto al producto que se quiere ofrecer y los beneficios que se quieren lograr como ser:

- La disponibilidad de un vaso con bebida de cocción natural en el momento oportuno y sitio ideal, el gimnasio del Departamento Policial número dos de la cuidad de La Paz. A través de una máquina expendedora de bebidas que funcione con monedas.
- La falta de ayuda con proyectos tecnológicos que impulsen a mantener nuestras bebidas naturales y tradicionales refrescos de orejón, cereza, cebada coca, y otros.
- Generación de productos tecnológicos electrónicos que se adecuen a nuestro contexto; es decir producción tecnológico boliviano.

1.2 Justificación del trabajo

1.2.1 Justificación económica

La máquina expendedora de bebidas naturales tiene un costo reducido frente a las maquinas existentes para este propósito, siendo así más accesible para nuestro medio.

1.2.2 Justificación Social

La máquina expendedora de bebidas quiere promover el consumo de refrescos tradicionales de cereza, trigo, manzana, moko chinchi, en lugares de entrenamiento.

Bajo el concepto de cuidado al medio ambiente reciclando, este proyecto utiliza bombas eléctricas de agua del sistema limpiaparabrisas de automóviles que están fuera de circulación.

1.2.3 Justificación tecnológica

Para la implementación de la máquina expendedora de bebidas naturales se utilizara como componente principal el microcontrolador AVR ATMEGA16 por la facilidad en el set de instrucciones, programación en su memoria, bajo consumo de energía y su costo reducido frente a otro microcontrolador como es el PIC.

1.3 Objetivos

1.3.1 Objetivo general

Implementar un prototipo en base al microcontrolador AVR ATMEGA16, de una máquina expendedora automatizada de bebidas naturales, que se active al introducir monedas y sea de fácil manejo para el usuario final.

1.3.2 Objetivos específicos

- Realizar el diagrama de flujo que guie el proceso para realizar el código fuente.
- Realizar el código de programación para el microcontrolador AVR ATMEGA16 capaz de interactuar con el circuito actuador y los periféricos de comunicación.
- Validar el código fuente y realizar el grabado en la memoria del microcontrolador AVR ATMEGA16
- Aplicar el circuito de funcionamiento para el microcontrolador AVR ATMEGA16
- Ejecutar el circuito actuador el cual se encarga de despachar la bebida a través de bombas eléctricas de automóviles.
- Proporcionar al usuario periféricos de sencilla operación de la máquina expendedora de bebidas naturales.

1.4 Delimitación

El prototipo de máquina expendedora automatizada de bebidas será diseñado para ambientes cerrados no así a la intemperie ni expuesto a situaciones

climatológicas extremas. El mismo esta ubicado en el gimnasio del Departamento Policial número dos de la cuidad de La Paz.

El prototipo de máquina expendedora automatizada de bebidas estará orientado a la distribución de bebidas naturales de cereza, trigo, manzana, moko chinchi; sin elementos sólidos, acuosos, gelatinosos, rugosos, flemosos.

La máquina expendedora automatizada de bebidas trabaja exclusivamente con la introducción de monedas sin valor legal.

CAPITULO II

2 Fundamentación teórica

2.1 introducción

La incursión de la tecnología electrónica a nuestro modo de vida, costumbres, necesidades y tradiciones ha dado lugar al surgimiento de proyectos como la máquina expendedora automatizada de bebidas naturales, para este objetivo se realiza un estudio del sistema de procesamiento central, sistema actuador y periféricos de comunicación.

2.2 Sistema de procesamiento central

2.2.1 Microcontrolador AVR ATMEGA16

El AVR es una CPU de arquitectura Harvard el, ATMEGA16 es un microcontrolador de 8 bits de la familia MEGA de la gama AVR de Atmel con bajo consumo de energía. Está basado en la arquitectura RISC con 131 instrucciones en ensamblador (disponibles para toda la familia MEGA) de las cuales la mayoría se ejecuta en un solo ciclo de instrucción. El ATMEGA16 puede funcionar a una frecuencia máxima de 16 MHz.

Este microcontrolador cuenta con 16 KB de memoria FLASH (de programa), 1 KB de memoria RAM estática y 513 bytes de memoria EEPROM. Según características, la memoria FLASH puede ser reprogramada 10,000 veces y la EEPROM 100,000, aproximadamente.

2.2.2 Características del microcontrolador ATMEGA16

- CPU de 131 instrucciones.
- 32 registros de propósito general (cada uno de 8 bits)
- Memoria de programa flash de 16KBytes

- Memoria SRAM de 1KByte

- 32 líneas de entrada/salida

- 3 timers (se puede generar hasta 4 PWM)

- ADC de 10 bits

- Comparador Analógico

- Comunicación serial USART, SPI, TWI

- Timer Watchdog

- Contador de tiempo real

El ATmega16 cuenta con 40 pines en empaquetado DIP (figura 1), los cuales 32 son de entrada/salida de propósito general divididos en 4 puertos de 8 bits cada uno, designados PORTA, PORTB, PORTC y PORTD.

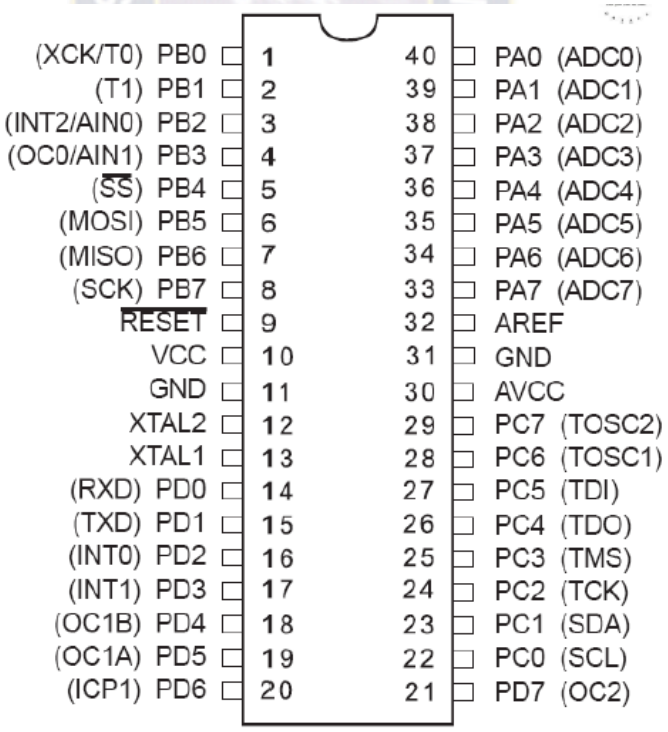


Figura 1
Empaquetado DIP microcontrolador ATMEGA16
Fuente: organización interna de los AVRs. Página seis

2.2.3 Circuito auxiliar externo

Para que todo microcontrolador sea capaz de funcionar en cualquier proyecto es necesario realizar las siguientes conexiones:

2.2.3.1 La alimentación

La tensión necesaria para el funcionamiento del AVR según especificaciones del fabricante está en el rango de 2,7 a 5,5 voltios continuos. Suministrados en los pines Vcc y Gnd

2.2.3.2 El reloj (oscilador)

El controlador ejecuta el programa a la frecuencia del reloj, el reloj puede ser interno o externo usando un cristal de cuarzo o, un circuito resonante LC, o incluso un circuito RC. Al alimentar el microcontrolador el reloj comienza a operar. Estos se aplican atreves de los pines XTAL1, XTAL2 con reloj externo puede trabajar hasta 20 MHz.

Se puede configurar su frecuencia de trabajo a través de su oscilador interno a 1, 2, 4 y 8 MHz.

El tipo de reloj con el que trabaja el microcontrolador se configura con los bits CKCEL3…0 (figura 2)

CKSEL3 ... 0	OPCIÓN
0000	Generador externo
0001	Reservado
0010	Oscilador RC interno calibrado
0011	Oscilador RC interno de 128 KHz
0100 – 0101	Oscilador a Cristal de baja frecuencia
0110 – 0111	Oscilador a Cristal de pleno funcionamiento
1000 ... 1111	Oscilador a Cristal de baja potencia

Figura 2
Opciones de configuración de reloj
Fuente: http://es.slideshare.net

2.2.3.3 El circuito externo de reinicio (opcional)

Es el circuito por el que podemos reiniciar el chip en cualquier momento para que vuelva al inicio del programa, se implementa en el pin de RESET introduciendo un nivel bajo de voltaje.

Lo más práctico, para facilitar un reinicio manual, es utilizar un pulsador, similar al que se encuentra en los ordenadores, el fabricante recomienda que se intercale una resistencia de 50 a 100 ohmios entre el pulsador y el pin de RESET, para evitar posibles corrientes inducidas de más de 80mA que podrían bloquear al chip cuando este se lleve a masa (reinicio).

Debido a que el pulsador no produce una respuesta instantánea, producto de los rebotes de este (transitorio), se generan una serie de pulsos hasta quedar estabilizado en un estado permanente. Para evitar esto se puede usar un condensador instalado en paralelo con la entrada RESET.

2.2.4 CPU

La función principal de la CPU es asegurar la correcta ejecución de programas. La CPU debe tener acceso a las memorias, realizar cálculos, controlar periféricos y manejar interrupciones.

Para maximizar el rendimiento y paralelismo, el AVR usa una arquitectura Harvard con memorias y buses separados para instrucciones y datos. La ALU soporta operaciones aritméticas y lógicas entre registros o entre un registro y una constante. Aunque también hay operaciones con un solo registro.

2.2.5 Ejecución de instrucciones

El flujo del programa por naturaleza es secuencial. Puede ser modificado por instrucciones de saltos condicionales e incondicionales y llamadas a rutinas, que pueden abarcar completamente el espacio de direcciones.

Las instrucciones en la memoria de Programa son ejecutadas con una segmentación de dos etapas. Mientras una instrucción está siendo ejecutada, la siguiente es capturada de la memoria de programa. Este concepto hace que se produzca una instrucción por cada ciclo de reloj.

Para la ejecución de instrucciones aritméticas y lógicas, la duración del ciclo es suficiente para permitir la lectura de registros, la operación de la ALU y la escritura en el registro destino. Figura 3

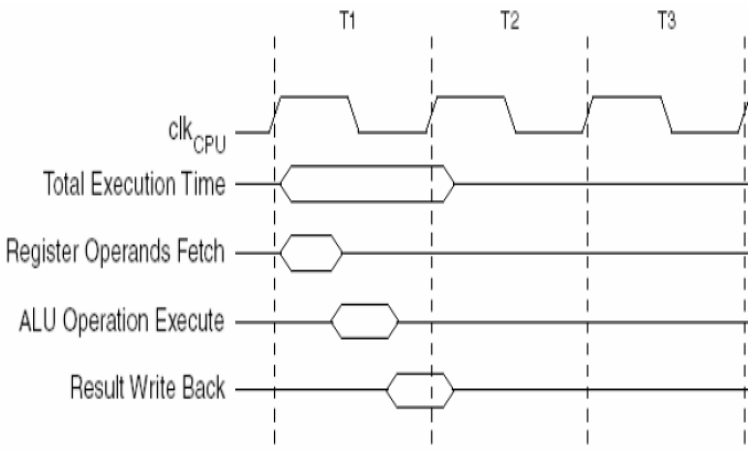


Figura 3
Ciclo de duración en la ejecución de instrucciones
Fuente:
www.exa.unicen.edu.ar/catedras/.../3_Overview_Microcontroladores_ATMEL.pdf

2.2.6 Mapa de memoria

el mapa de memoria esta compuesto por las siguientes etapas. Figura 4

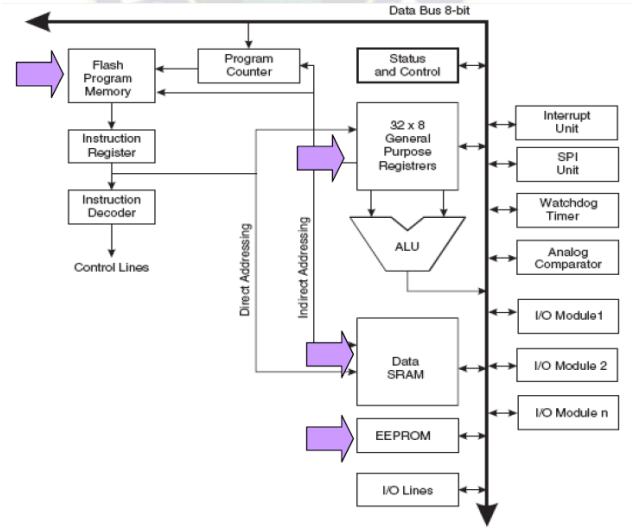


Figura 4
Mapa de memoria avr
Fuente:
www.exa.unicen.edu.ar/catedras/.../3_Overview_Microcontroladores_ATMEL.pdf

2.2.6.1 Memoria de programa (flas program memory)

La memoria se puede dividir en una sección para aplicación y una sección de arranque, donde podría manejarse un cargador para auto programación. En el espacio de almacenamiento se incluyen los Vectores de Interrupción, iniciando en la dirección 0x000.

Espacio continuo de memoria Flash para el ATMega16 es de 16 KBytes (8 K x 16 bits)

2.2.6.2 Memoria de propósito general (general propuse registrers)

La arquitectura AVR es del tipo Registro – Registro. Existen diversas instrucciones para cargas o almacenamientos, ya sean de manera directa o indirecta.

La etapa de ejecución de un acceso a memoria, ya sea carga o almacenamiento, requiere de dos ciclos de reloj.

2.2.6.3 Memoria de datos

Es un espacio de 1120 localidades de 8 bits e incluyen:

- Un conjunto de 32 localidades (Registros).
- 64 Registros I / O (Puertos, configuración de recursos, etc.).
- 1024 localidades de propósito general. (RAM)

2.2.6.4 Archivo de Registros.- tienen acceso a todos ellos. Las que operan un registro con una constante sólo trabajan con los registros de R17 a R31, cada registro tienen una dirección que le permite ser tratado como cualquier otra localidad de RAM (0x000 – 0x01F), utilizando instrucciones de Carga (LD) y almacenamiento (ST). De R26 a R31 pueden usarse como apuntadores para direccionamiento indirecto.

2.2.6.5 Registros I/O.- Son 64 Registros e incluyen a los Puertos de Entrada/Salida, así como registros para la configuración, el control y el estado de los recursos internos, se tiene acceso a los registros I/O con las instrucciones IN y OUT, intercambiando datos con el Archivo de Registros. Con estas instrucciones deben usarse las direcciones 0x00 - 0x3F. Los registros I/O pueden ser tratados como memoria.

2.2.6.6 Registro de estado

El registro de estado proporciona información acerca del resultado de operaciones aritméticas, operaciones lógicas, estado de las interrupciones.

2.2.6.7 Pila de programa

La pila es implementada en el espacio de propósito general (RAM), es usada para almacenamiento temporal de variables o durante la llamada de subrutinas o el manejo de interrupciones. Realmente se compone de 2 registros, para la parte alta (SPH) y para la parte baja (SPL), esto para direccionar al espacio completo de memoria.

2.2.6.8 Memoria EEPROM

Este es un espacio no volátil para el almacenamiento de datos, en el ATMEGA16 es de 512 bytes.

2.2.6.9 Puertos de entrada y salida del ATMEGA16

Por cada puerto de E/S existen 3 registros que lo controlan: DDRx, PORTx y PINx (la x hace referencia al nombre del puerto seleccionado: A, B, C.. etc.). Cada bit de estos registros hace referencia a un pin físico del puerto en el chip.

Escribiendo un 1 en la posición del pin en el DDRx configurará ese pin como un pin de salida y escribiendo un 0 configurará el pin como un pin de entrada.

El registro PORTx sirve para enviar datos a los pines de salida o activar PULL-UPS en pines de entrada

El registro PINx sirve para leer puertos de entrada en los terminales

2.2.7 Sistema actuador

Este sistema está compuesto por los siguientes elementos principales.

2.2.7.1 Relés electromecánicos

Están formados por una bobina y unos contactos los cuales pueden conmutar corriente continua o bien corriente alterna.

Relés de tipo armadura.

Son los más antiguos y también los más utilizados. El electroimán hace vascular la armadura al ser excitada, cerrando los contactos dependiendo de si es N.O con alta impedancia o N.C baja impedancia (normalmente abierto o normalmente cerrado).

Las características generales de cualquier relé son:

- El aislamiento entre los terminales de entrada y de salida.
- Adaptación sencilla a la fuente de control.
- Posibilidad de soportar sobrecargas, tanto en el circuito de entrada como en el de salida.

2.2.7.2 Bombas eléctricas de automóviles

Las bombas eléctricas trabajan normalmente con un voltaje que varía entre 12 y 13 voltios, suministrados al momento de pasar el interruptor, En ese momento

comienza a girar el motor eléctrico, suministrando una presión que puede variar desde 5 hasta 15 libras por pulgada cuadrada

Estos bombas son sometidas a rigurosas pruebas de funcionamiento, como por ejemplo, operar en forma continua 500 horas a 4.000 rpm con salida total o ensayos de duración en condiciones extremas, ofreciendo así, garantía de calidad y larga vida.

2.2.7.3 Transistor bipolar

Es un transistor de unión bipolar de mediana potencia, destinado para propósito general en amplificación y conmutación, construido con semiconductor silicio

Para lograr que el transistor entre en corte, el valor de la corriente de base debe ser bajo o mejor aún, cero (figura 5)

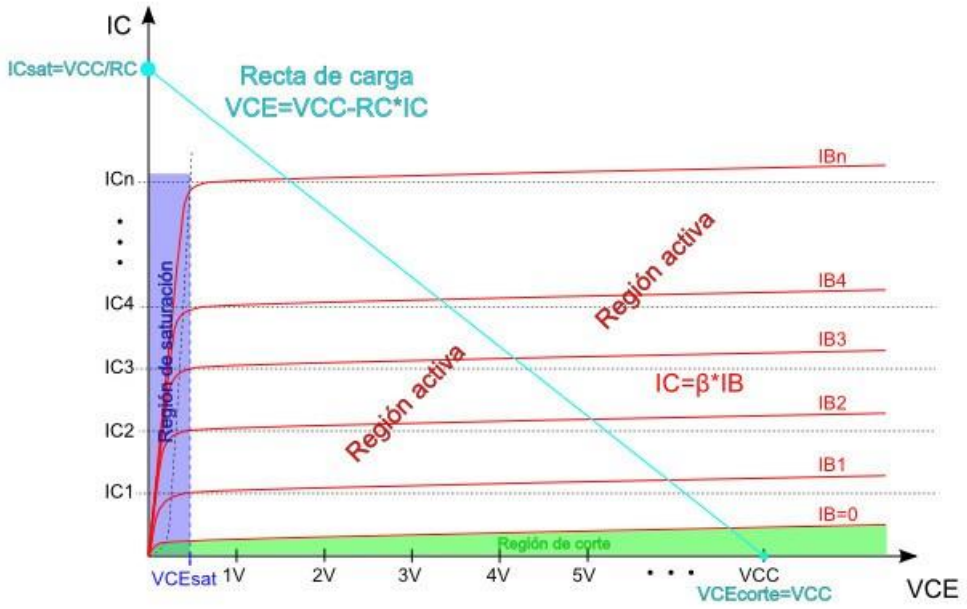


Figura 5
Diagrama de regiones corte y saturación del transistor
Fuente: http://mrelbernitutoriales.com

Para lograr que el transistor entre en saturación, el valor de la corriente de base debe calcularse dependiendo de la carga que se esté operando entre encendido y apagado (funcionamiento de interruptor)

Si se conoce cuál es la corriente que necesita la carga para activarse (bobina de relé electromecánico), se tiene el valor de corriente que habrá de conducir el transistor cuando este en saturación y con el valor de la fuente de alimentación del circuito, se puede su funcionamiento.

2.3 Periféricos de comunicación

En este proyecto los periféricos de comunicación dispuestos son:

2.3.1 LCD

LCD significa: “Liquid Cristal Display” o en español “Pantalla de cristal líquido“, y es una pantalla delgada y plana, formada por un número de píxeles en color o monocromos colocados delante de una fuente de luz o reflectora.

Estos se caracterizan principalmente por el número de caracteres que son capaces de representar, en este caso un display LCD 16×2, es capaz de representar 2 filas de 16 caracteres.

Los displays LCD diseñados para interactuar con circuitos integrados, de entrada de 4/8 bits en paralelo, para luego mostrar en su pantalla caracteres, letras y números conocidos y entendibles para el ser humano.

La tensión nominal de alimentación es de 5V, con un consumo menor de 5mA.

El LCD dispone de dos tipos de memorias independientes: la DD RAM En esta memoria se almacenan los caracteres que están siendo visualizados o que se encuentran en posiciones no visibles. El display almacena en esta memoria dos líneas de 40 caracteres pero sólo se visualizan 2 líneas de 16 caracteres. Por ello

la DD RAM tiene un tamaño de 2x40=80 bytes. Para localizar los elementos dentro del display virtual se va a utilizar un par de coordenadas (x, y) donde x representa la posición horizontal (comprendida entre 1-40) e y representa la línea (1-2).

La memoria CG RAM está dividida en 8 bloques, correspondiendo cada bloque a un carácter definible por el usuario. Por ello el usuario puede definir como máximo 8 caracteres, cuyos códigos van del 0 al 7, cada carácter está constituido por una matriz de 5 columnas x 8 filas (la asignación de pines se muestra en anexo).

2.3.2 Pulsador y tragamonedas

Este elemente permite el paso o interrupción de la corriente mientras es accionado. Cuando ya no se actúa sobre él vuelve a su posición de reposo. Puede ser el contacto normalmente cerrado en reposo NC, o con el contacto normalmente abierto NC. Consta del botón pulsador; una lámina conductora que establece contacto con los dos terminales al oprimir el botón y un muelle que hace recobrar a la lámina su posición inicial al cesar la presión sobre el botón pulsador.

CAPITULO III

3 Desarrollo del trabajo

3.1 Introducción

Con el desarrollo de este proyecto se realiza una máquina expendedora de bebidas naturales, que ofrece un servicio con disponibilidad en cualquier momento del día.

La máquina estará conectada a la energía eléctrica para su funcionamiento, para su operación con el usuario estará disponible un tragamonedas (sin valor legal), dos pulsadores y un display LCD. El cliente deberá colocar un vaso para recibir su bebida e introducir una moneda en el tragamonedas, el display mostrara un mensaje indicando escoger un sabor el cual se debe escoger presionando uno de los pulsadores designado para este fin, una vez realizada la selección la maquina despachara la opción elegida en el vaso, finalizando así este proceso y volviendo al estado inicial a la espera de otro usuario.

El proyecto maneja como principal componente al microcontrolador ATMEGA16

3.2 Descripción general del sistema

El sistema general está divido de la siguiente manera, sistema de procesamiento central, sistema de actuador y los periféricos de comunicación (Figura 6).

Los periféricos de comunicación en este caso el tragamonedas y los pulsadores al activarse enviaran un pulso al microcontrolador por los puertos PA0, PA1 y PA2 respectivamente a la vez el LCD estará recibiendo mensajes desde el micro-controlador para ser observados en su pantalla durante todo el proceso. Una vez que el micro-contralor recibe los pulsos de entrada ejecuta el programa para activar los bits del puerto B (PB0 y PB1) configurados como salida enviando una

señal constante durante un tiempo programado y activar las bombas eléctricas despachando la bebida.

Figura 6
Sistema general de la máquina expendedora de bebidas
Fuente: elaboración propia

3.4 Diseño

3.4.1 Sistema de procesamiento central

3.3.1.2 Diagrama de flujo

Un diagrama de flujo es una representación gráfica de un proceso. Cada paso del proceso es representado por un símbolo diferente que contiene una breve descripción de la etapa de proceso, unidos entre sí con flechas que indican la dirección de flujo del proceso (figura 7,8).

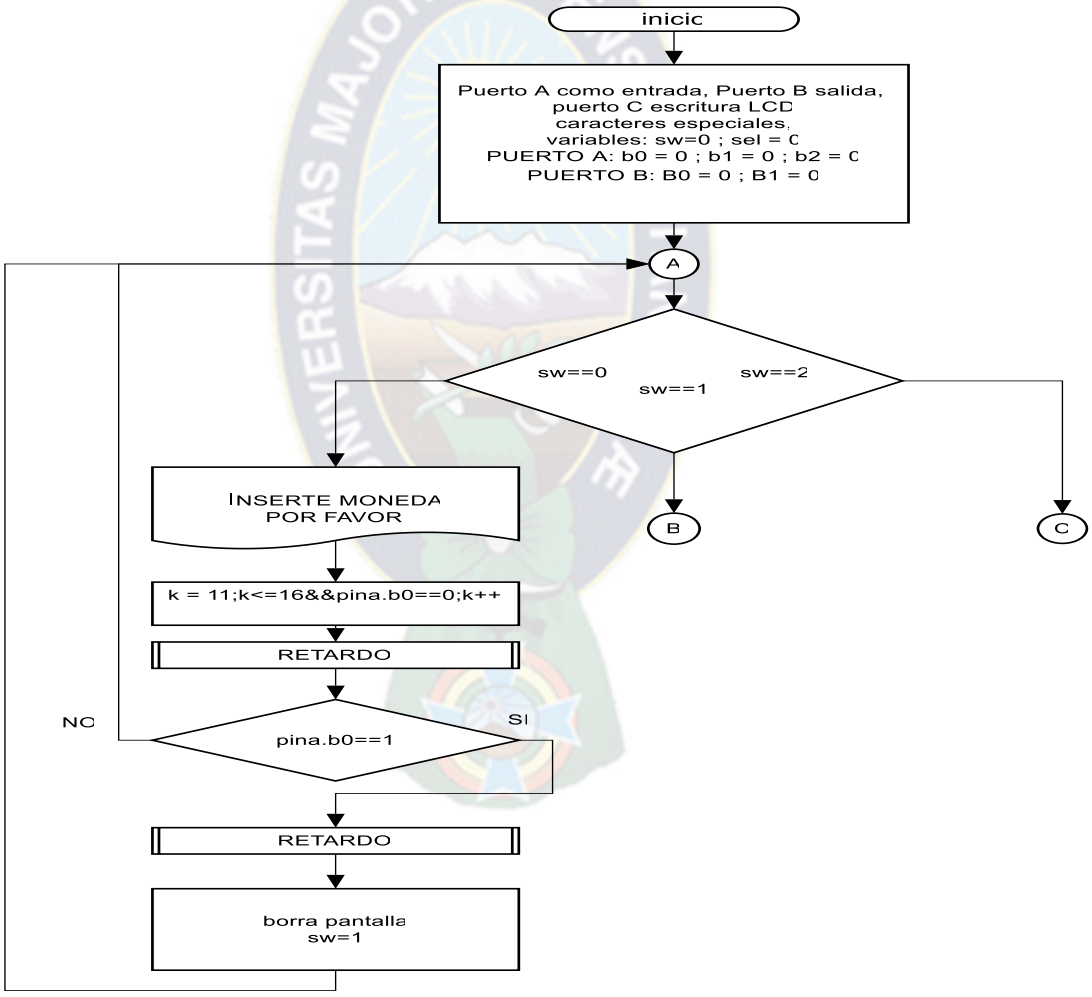


Figura 7
Diagrama de flujo para programa máquina expendedora de bebidas
Fuente: elaboración propia

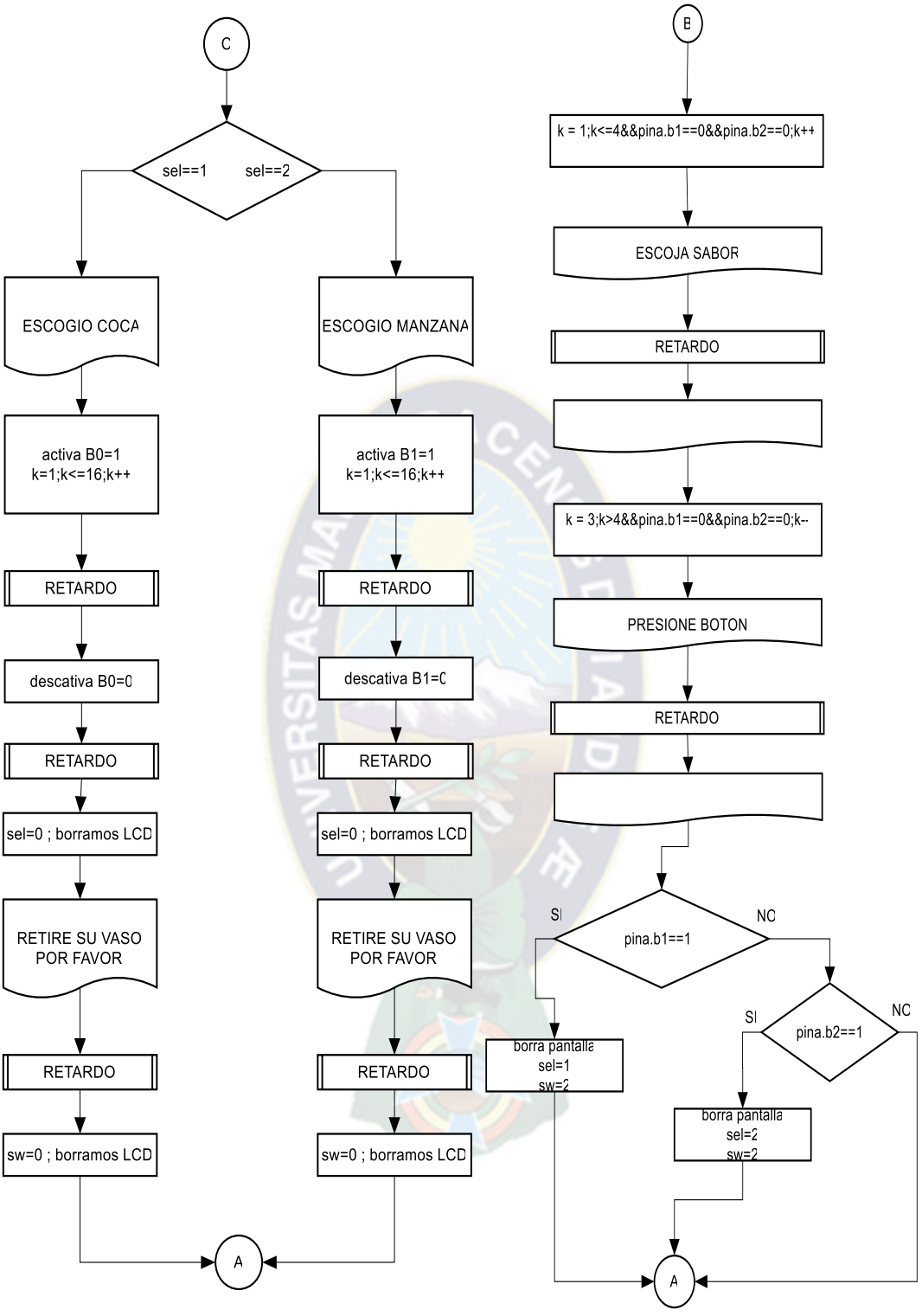


Figura 8
Diagrama de flujo para programa máquina expendedora de bebidas
Fuente: elaboración propia

3.3.1.3 Código de programación

En base al diagrama de flujo generado anteriormente se realiza el código fuente, capaz de recibir los pulsos de activación en los de los bits del puerto A configurados como entradas, provenientes del tragamonedas y de los pulsadores, además de enviar una señal constante a sus bits del puerto B configurados como salidas durante un tiempo.

```
// Lcd pin salida
sbit LCD_RS at PORTC0_bit;
sbit LCD_EN at PORTC1_bit;
sbit LCD_D7 at PORTC7_bit;
sbit LCD_D6 at PORTC6_bit;
sbit LCD_D5 at PORTC5_bit;
sbit LCD_D4 at PORTC4_bit;

// Pin dirección
sbit LCD_RS_Direction at DDC0_bit;
sbit LCD_EN_Direction at DDC1_bit;
sbit LCD_D7_Direction at DDC7_bit;
sbit LCD_D6_Direction at DDC6_bit;
sbit LCD_D5_Direction at DDC5_bit;
sbit LCD_D4_Direction at DDC4_bit;
void crear(char aux[],char pos);
signed int k;
short mon=0b11110010;
char sw=0,sel=0;
char nar[]={0,14,31,21,17,17,14,0};
char vaso[]={17,17,17,17,31,31,14,0};
char man[]={3,12,31,17,17,17,31,0};
void main()
{
    LCD_INIT();
    LCD_CMD(_LCD_CURSOR_OFF);
    DDRA.b0=0;
```

```
porta.b0=0;
DDRA.B1=0;
porta.b1=0;
DDRA.B2=0;
porta.b2=0;
DDRB.B0=1;
PORTB.B0=0;
DDRB.B1=1;
PORTB.B1=0;
crear(nar,64);
crear(vaso,72);
crear(man,80);
while (1)
{
    while (sw==0)
    {
        lcd_out(1,1,"INSERTE MONEDA");
        LCD_OUT(2,1,"POR FAVOR");
        lcd_chr(2,16,0b10111010);
        for (k=11;k<=16 && pina.b0==0;k++)
        {
           lcd_chr(2,k,mon);
           lcd_chr(2,k-1,' ');
           delay_ms(200);
        }
        if (pina.b0==1)
           {
             delay_ms(200);
             lcd_cmd(_lcd_clear);
             sw=1;
           }
    }
    while (sw==1)
    {
        for (k=1;k<=4 && pina.b1==0 && pina.b2==0;k++)
        {
           lcd_out(1,k,"ESCOJA SABOR");
           lcd_chr(1,k-1,' ');
           delay_ms(200);
        }
        lcd_out(1,1,"               ");
```

```
      for (k=3;k>0 && pina.b1==0 && pina.b2==0;k--)
      {
         lcd_out(2,k,"PRESIONE BOTON");
         lcd_chr(2,k+14,' ');
         delay_ms(200);
      }
      lcd_out(2,1,"               ");
      if (pina.b1==1)
      {
        delay_ms(200);
        sel=1;
        lcd_cmd(_lcd_clear);
        sw=2;
      }
      if (pina.b2==1)
      {
        delay_ms(200);
        sel=2;
        lcd_cmd(_lcd_clear);
        sw=2;
      }
}
while (sw==2)
{
      if (sel==1)
      {
        lcd_out(1,1,"ESCOGIO COCA");
        PORTB.B0=1;
        for (k=1;k<=16 ;k++)
        {
           lcd_chr(2,k,1);
           lcd_chr(2,k-1,0);
           lcd_chr(2,k-2,' ');
           delay_ms(50);
        }
        PORTB.B0=0;
        delay_ms(1000);
        sel=0;
        lcd_cmd(_lcd_clear);
        lcd_out(1,1,"RETIRE SU VASO");
        lcd_out(2,5,"POR FAVOR");
```

```
                DELAY_MS(3000);
                lcd_cmd(_lcd_clear);
                sw=0;
              }
              if (sel==2)
              {
                lcd_out(1,1,"ESCOGIO MANZANA");
                PORTB.B1=1;
                for (k=1;k<=16 ;k++)
                {
                   lcd_chr(2,k,1);
                   lcd_chr(2,k-1,2);
                   lcd_chr(2,k-2,' ');
                   delay_ms(50);
                }
                PORTB.B1=0;
                delay_ms(1000);
                sel=0;
                lcd_cmd(_lcd_clear);
                lcd_out(1,1,"RETIRE SU VASO");
                lcd_out(2,5,"POR FAVOR");
                DELAY_MS(300);
                lcd_cmd(_lcd_clear);
                sw=0;
              }
          }
      }
}
void crear(char aux[],char pos)
{
     LCD_Cmd(pos);
     for(k=0;k<=6;k++)
     {
        LCD_Chr_Cp(aux[k]);
     }
}
```

3.3.1.4 Validación del código fuente y quemado en la memoria del ATMEGA16

para el ATMEGA16 se utiliza el compilador Micro C AVR en lenguaje C que ofrece un entorno manejable y librerías para el manejo del LCD (figura 9), este programa verificara que no exista errores (figura 10) en el código fuente a la vez generara un archivo para realizar el grabado en el microcontrolador

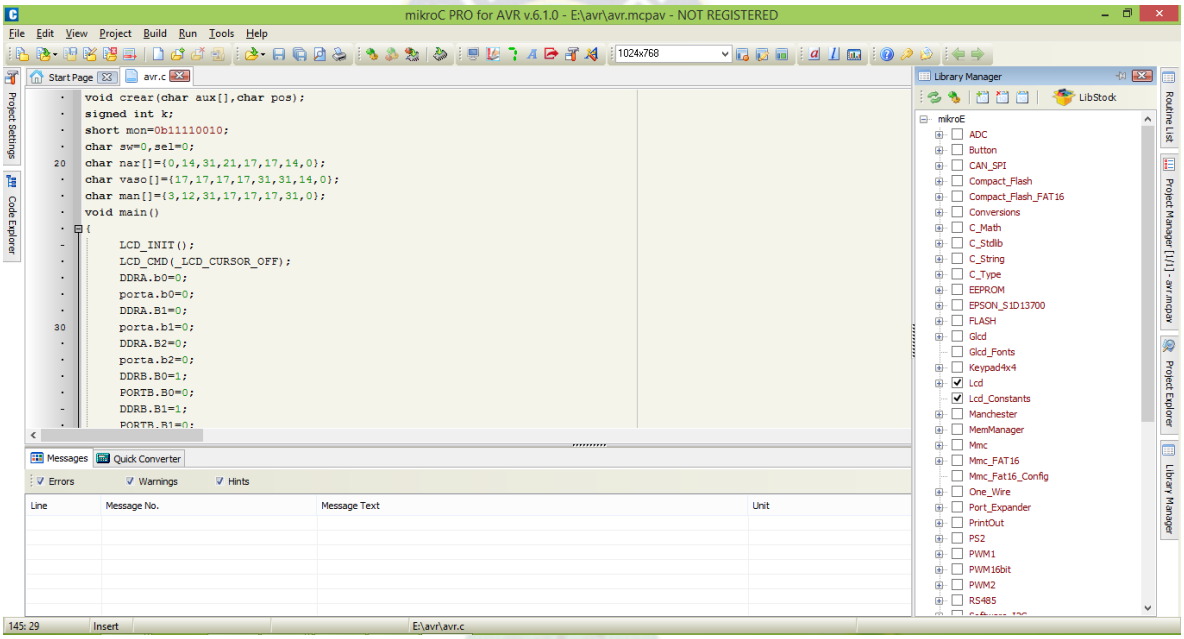


Figura 9
Entorno de programación del Micro C AVR
Fuente: elaboración propia

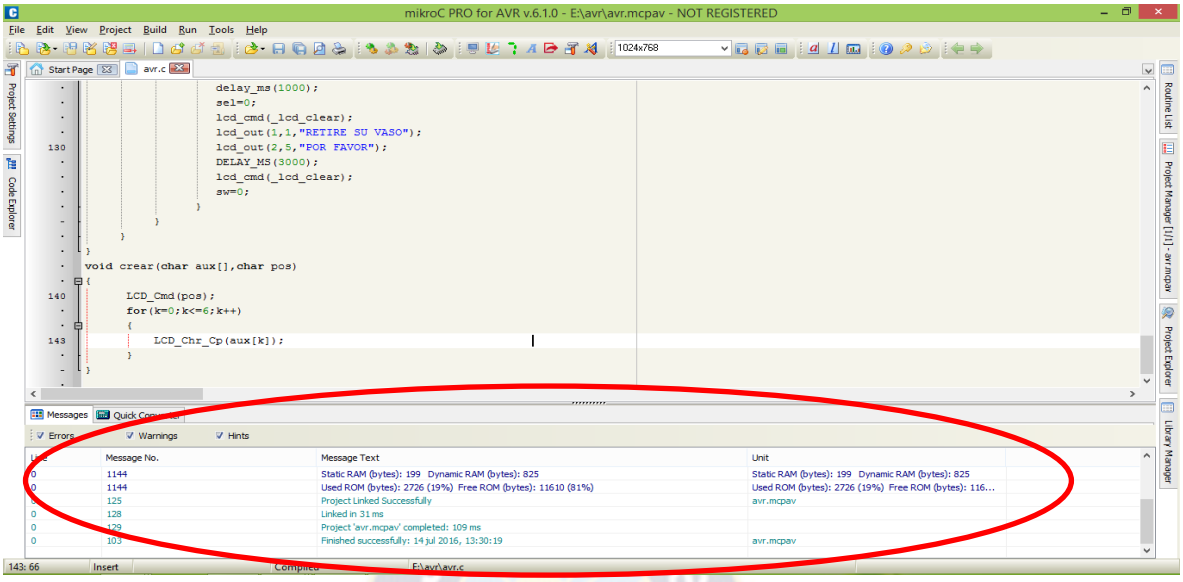


Figura 10
Código validado sin errores para su compilación
Fuente: elaboración propia

Mediante el programa PonyProg (figura 11). Se realizara el grabado del programa en la memoria del ATMEGA16, para esto se utiliza un cable serial con el siguiente circuito (figura 12)

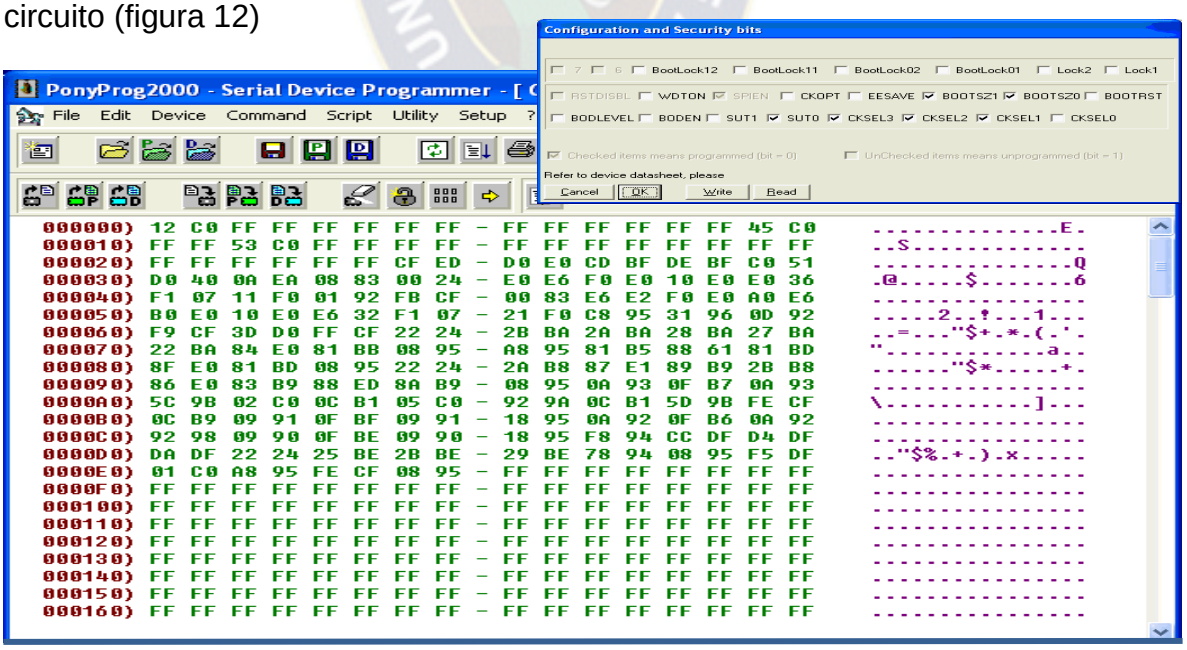


Figura 11
Entorno de grabación en PonyProg
Fuente: elaboración propia

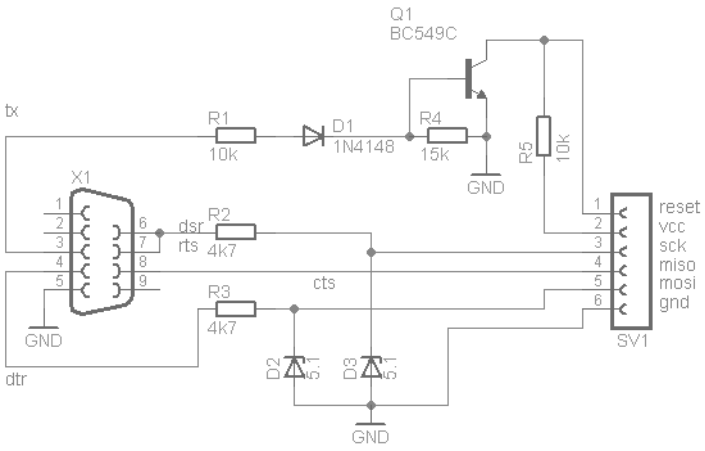


Figura 12
Cable serial para programador AVR
Fuente: www.electronics-diy.com

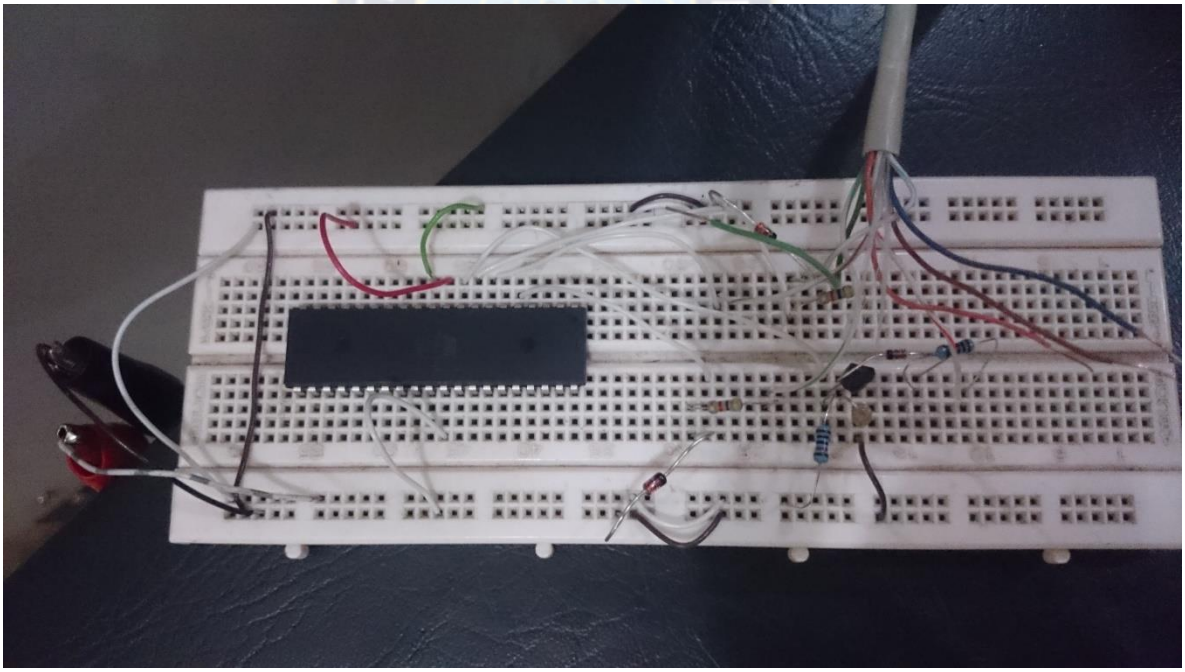


Figura 13
Circuito quemador para AVR armado en prothoboard
Fuente: elaboración propia

3.3.1.5 Circuito de funcionamiento para el microcontrolador ATMEGA16

el circuito necesario para que el microcontrolador funcione de forma fisica , con las siguientes conexiones:

- un cristal externo de 4 Mhz en los pins 12 y 13
- pin 10 a Vcc y 11 a Gnd, alimentado a travez de un relgulador de tension el LM7805 (fugura 14)

El LM7805 un dispositivo electrónico que tiene la capacidad de regular voltaje positivo de 5V a 1A de corriente, estamos obligados a garantizar una fuente de tensión constante, ya que nuesto circuito general se alimentara con 12 voltios, entoces aplicaremos 12 voltios a la entrada del regulador y obtendremos 5 voltios a la salida con C2 de 100 microfaradios y C1 10 microfaradios. Ademas tendremos un led piloto conectado en serie a una resistencia que nos indicara que nuestro microcontrolador se esta alimentando.

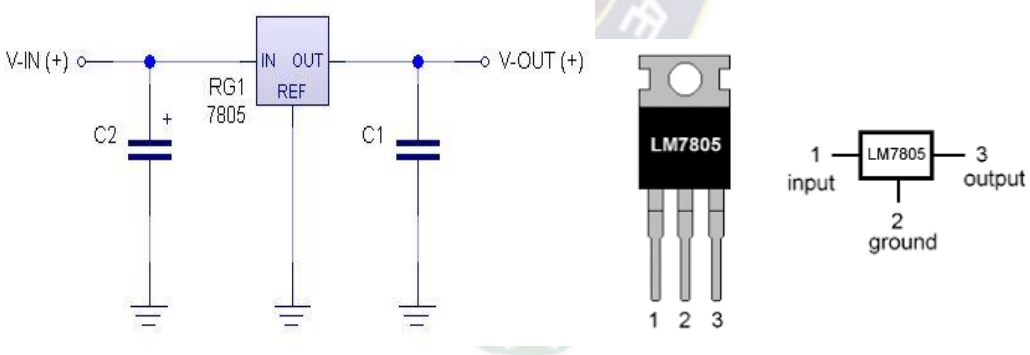


Figura 14
Regulador de voltaje LM7805
Fuente: www.electrontools.com

3.2 Sistema actuador

El sistema actuador se activa una vez que el microcontrolador envía un voltaje continuo entre 3 a 5 voltios con una corriente aproximada de 3 mA a través de los

bits de salida del puerto B; esta voltaje llega al base del transistor BC548 el cual está cumpliendo la función de entrar en saturación de esa manera se induce una corriente en la bobina del relé logrando que sus contactos que pasen de estar abiertos a estar cerrados de esta manera puede circular el voltaje necesario para entrar en funcionamiento las bombas eléctricas, que suministraran el líquido hacia el vaso.

El circuito es el de la figura 15, donde:

RC = represente la bobina del relé

VCC = VBB = voltaje de circuito (5V)

IB = corriente de beta

RB = resistencia que se calcula para que el circuito entre en saturación

β = 100 (se obtiene de la hoja técnica del transistor)

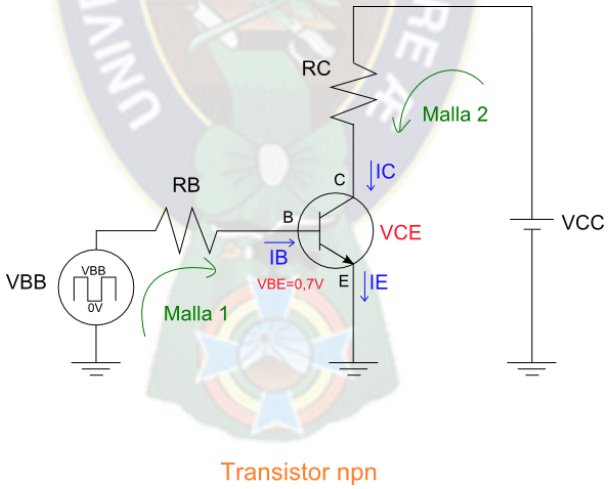


Figura 15
Circuito de corte y saturación
Fuente: http://mrelbernitutoriales.com

Para calcular RB se utiliza las siguientes ecuaciones en base a análisis de mallas.

ICsat=VCC/RC

IB=ICsat/β

VBB=IB*RB+VBE, de donde

RB=(VBB-0,7)/IB, luego:

RB=(VBB-0,7)/(5*ICsat/β)

3.3 Periféricos de comunicación

Los periféricos de comunicación son los encargados de realizar la comunicación entre la máquina y el usuario, el tragamonedas al recibir una moneda activa un pulso positivo (5v) que envía al micro-controlador este empezará a ejecutar el programa habilitando los dos pulsadores para que nuevamente se envié otro pulso positivo (5v) que continúe con el programa enviando esas vez un voltaje continuo a través de los pines habilitados como salidas durante un tiempo controlado en el programa por medio de retardos, durante todo el proceso el display está enviando mensajes guía para la operación de la máquina expendedora.

Los pulsadores están conectados a resistencias de 1kohmios para evitar el efecto rebote. El circuito completo con los periféricos de comunicación y demás se muestra en el simulador ISIS PROTEUS (figura 16).

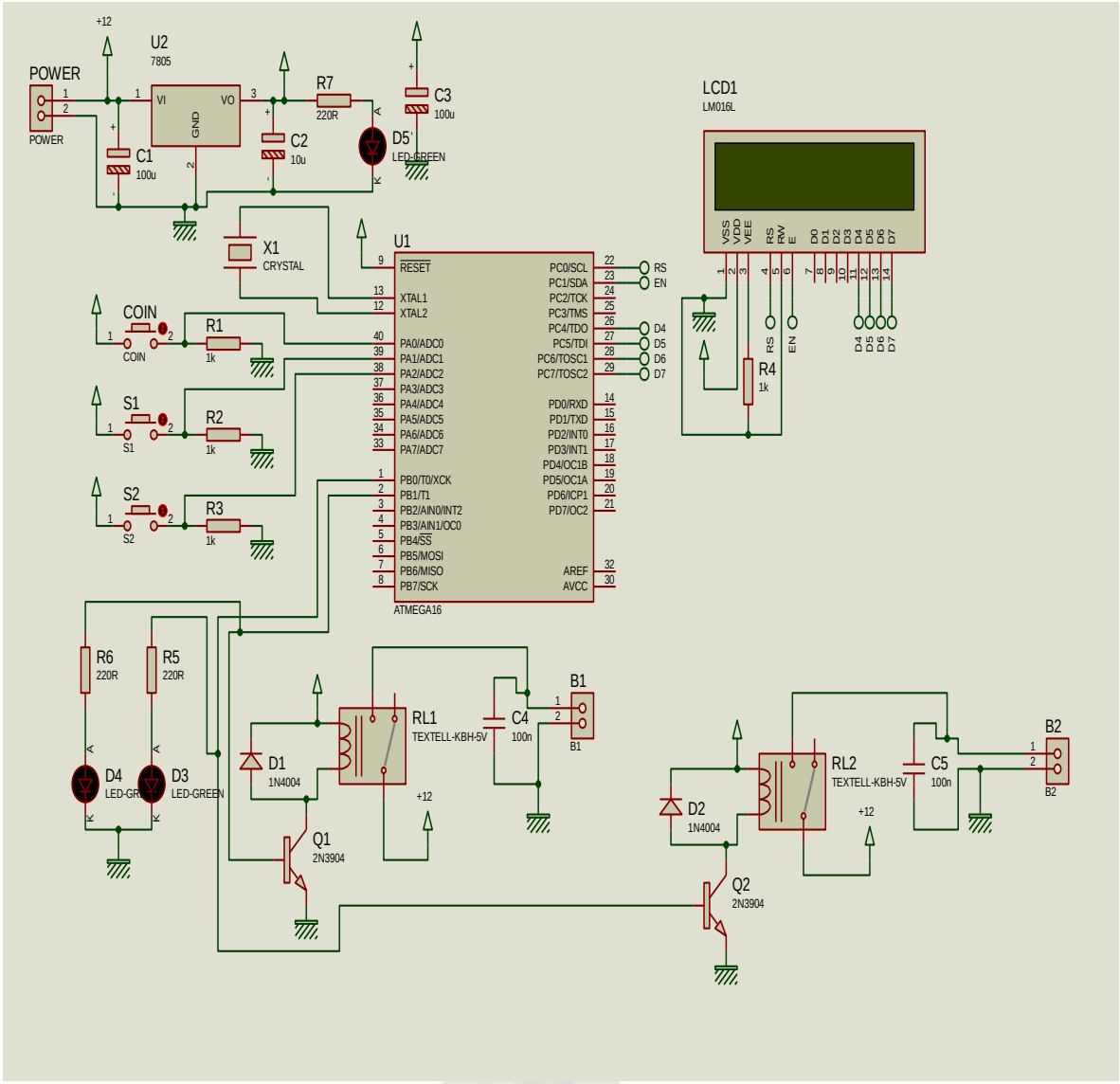


Figura 16
Circuito completo simulado en ISIS PROTEUS
Fuente: elaboración propia

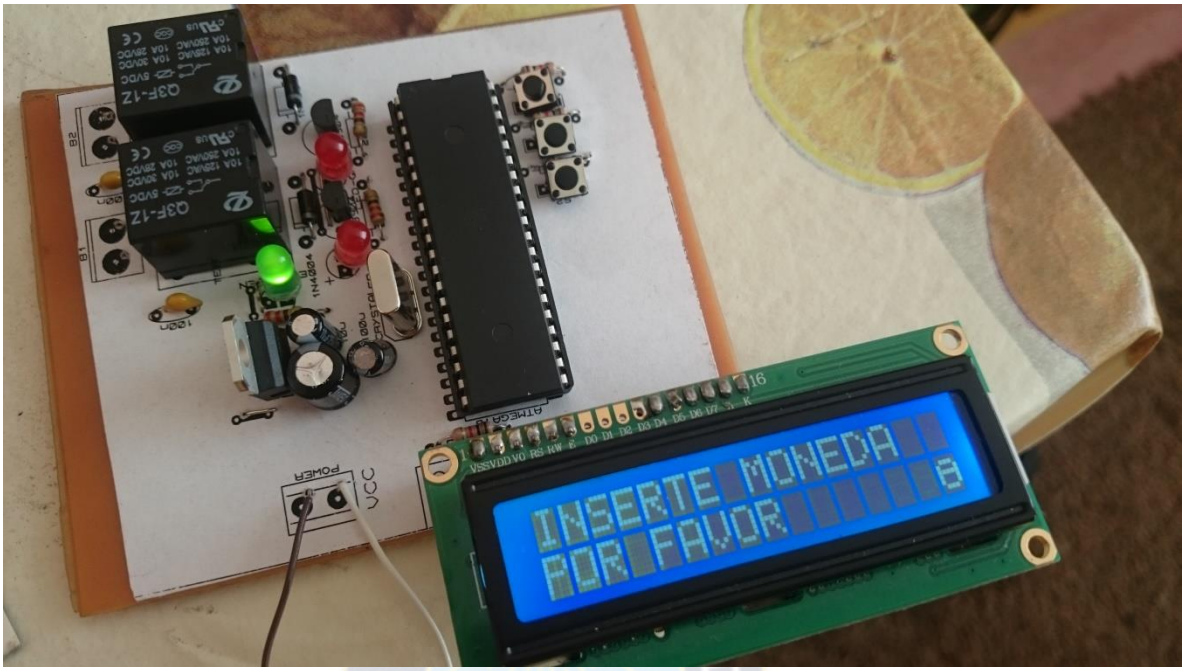


Figura 17
Circuito completo armado en placa
Fuente: elaboración propia

3.4 Análisis de costos

Para el presente sistema se realiza una estimación de costos en bolivianos de los materiales y componentes utilizados en el diseño, estos precios se obtuvieron por medio de cotizaciones en tiendas electrónicas cabe aclarar que en la siguiente tabla no está incluido la mano de obra tampoco los derechos de autoría.

TABLA DE PRECIOS DE COMPONENTES

ITEM	CANTIDAD	DESCRIPCION	PRECIO UNITARIO (Bs.)	TOTAL
1	8	RESISTOR DE ¼ WATT	0.50	4.00
2	4	CAPACITOR	2.00	8.00
3	1	REGULADOR DE VOLTAJE	7.00	7.00
4	2	TRANSISTOR	2.00	4.00
5	1	CRISTAL	5.00	5.00
6	2	DIODO	2.00	4.00

7	2	RELE	7.00	14.00
8	3	PUSADOR	5.00	15.00
9	1	FUENTE AC/DC	50.00	50.00
10	2	BOMBAS DE AGUA	75.00	150.00
11	1	AVR ATMEGA16, SOCKET	38.00	38.00
12	1	LCD	60.00	60.00
			PRECIO TOTAL	**359.00**

TABLA DE PRECIOS DE MATERIALES ADICIONALES

ITEM	**CANTIDAD**	**DESCRIPCION**	**PRECIO UNITARIO (Bs.)**	**TOTAL**
1	1	QUEMADO DE PLACA IMPRESA	30.00	30.00
2	1	CAJA CONTENEDORA DEL CIRCUITO	50.00	50.00
3	2	MANGUERA	3.00	6.00
			PRECIO TOTAL	**86.00**

Considerando los gastos de en componentes y gastos en materiales adicionales se logra un total de:

Costo de componentes	359.00 Bolivianos
Costo de materiales adicionales	86.00 Bolivianos
TOTAL	**445.00 Bolivianos**

CAPITULO IV

4.1 Conclusiones y recomendaciones

El proyecto de máquina expendedora automatizada de bebidas naturales, es un éxito ya que se logró que funcione dentro los límites expuestos en un inicio.

- Ofrece bebidas naturales por medio de la autogestión del usuario.
- El microcontrolador ATMEGA16 trabaja como elemento principal.
- Ofrece interfaces sencillas de operación para el cliente.
- Es un producto tecnológico de iniciativa boliviana.

Para el uso de monedas con valor legal se recomienda reemplazar el tragamonedas actual, teniendo en cuenta que este cambio incrementara el precio de la máquina expendedora de bebidas naturales.

Con el fin de lograr un control en la cantidad de bebida disponible y monedas insertadas se recomienda adicionar software en el código fuente o instalar sensores, circuitos, de medición y conteo.

4.2 Bibliografía

www.netzek.com/2013/12/atmega16-introduccion.html

www.vidaembebida.wordpress.com/2014/07/09/el-atmega16/#more-172

www.utm.mx/~fsantiag/Micros/2_Organizacion_AVRs.pdf

www.dte.us.es/personal/pparra/EdC-T4-AVR-v0-1a-parte

www.exa.unicen.edu.ar/catedras/.../3_Overview_Microcontroladores_ATMEL.pdf

www.electrontools.com

http://mrelbernitutoriales.com

ANEXOS

RESUMEN DE LOS COMANDOS LCD

	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
Borrar Display	0	0	0	0	0	0	0	0	0	1
Cursor a Home	0	0	0	0	0	0	0	0	1	*
Establecer modo de funcionamiento	0	0	0	0	0	0	0	1	I/D	S
Control ON/OFF	0	0	0	0	0	0	1	D	C	B
Desplazamiento del cursos/display	0	0	0	0	0	1	S/C	R/L	*	*
Modo de transferencia	0	0	0	0	1	DL	1	0	*	*
Acceso a memoria CG RAM	0	0	0	1	Dirección de la CG RAM					
Acceso a memoria DD RAM	0	0	1	Dirección de la DD RAM						
Lectura de dirección y del flag de ocupado	0	1	BF	Contador de dirección						
Escritura de datos en CG RAM /DD RAM	1	0	Dato a escribir							
Lectura de datos en CG RAM/DD RAM	1	1	Dato leído							

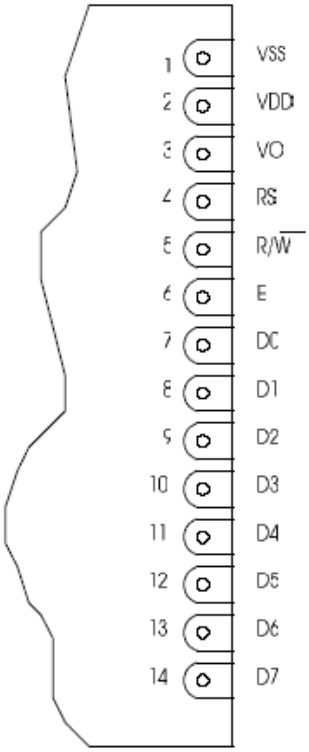
I/D = 1: Incrementar contador direcciones
S=1: Desplazamiento del display
D=1: Display ON
C=1: Cursor ON
B=1: Parpadeo del caracter en la posición del cursor
S/C=1: Desplazar el display.
R/L=1: Desplazamiento a la derecha
DL=1: Configurar display a 8 bits
BF=1: Display ocupado

I/D = 0: Decrementar contador direccione:
S=0: Display 'quieto
D=0: Display OFI
C=2: Cursor OFI
B=0: No hay parpadec
S/C=0: Desplazar el curso
R/L=0: Desplazamiento a la izquierdc
DL=0: Configurar display a 4 bit:
BF=0: Display listo para ejecutar otra operaciór

CODIGO ASOCIADO A CADA CARÁCTER IMPRIMIBLE EN EL DISPLAY

Código	Car.	Código	Car.	Código	Car.	Código	Car.	Código	Car.	Código	Car.
$20	espacio	$30	0	$40		$50	P	$60	‘	$70	p
$21	!	$31	1	$41	A	$51	Q	$61	a	$71	q
$22	“	$32	2	$42	B	$52	R	$62	b	$72	r
$23	#	$33	3	$43	C	$53	S	$63	c	$73	s
$24	$	$34	4	$44	D	$54	T	$64	d	$74	t
$25	%	$35	5	$45	E	$55	U	$65	e	$75	u
$26	&	$36	6	$46	F	$56	V	$66	f	$76	v
$27	‘	$37	7	$47	G	$57	W	$67	g	$77	w
$28	(	$38	8	$48	H	$58	X	$68	h	$78	x
$29	)	$39	9	$49	I	$59	Y	$69	i	$79	y
$2A	*	$3A	:	$4A	J	$5A	Z	$6A	j	$7A	z
$2B	+	$3B	;	$4B	K	$5B	[	$6B	k	$7B	{
$2C	,	$3C	<	$4C	L	$5C		$6C	l	$7C	|
$2D	-	$3D	=	$4D	M	$5D	]	$6D	m	$7D	}
$2E	.	$3E	>	$4E	N	$5E	^	$6E	n	$7E	→
$2F	/	$3F	?	$4F	O	$5F	_	$6F	o	$7F	←

ASIGNACION DE PINES LCD



Nº de PIN	Simbolo	Descripción
1	VSS	Masc
2	VDD	Alimentación
3	VO	Voltaje de ajuste del contraste
4	RS	Selección de registro
5	R/W	Lectura/escritura
6	E	Enable
7	D0	Bit de datos menos significativo
8	D1	Bit de datos
9	D2	Bit de datos
10	D3	Bit de datos
11	D4	Bit de datos
12	D5	Bit de datos
13	D6	Bit de datos
14	D7	Bit de datos mas significativo

SET INSTRUCCIONES AVR

Conditional Branch Summary

Test	Boolean	Mnemonic	Complementary	Boolean	Mnemonic	Comment
Rd > Rr	Z•(N ⊕ V) = 0	BRLT(1)	Rd ≤ Rr	Z+(N ⊕ V) = 1	BRGE*	Signed
Rd □ Rr	(N ⊕ V) = 0	BRGE	Rd < Rr	(N ⊕ V) = 1	BRLT	Signed
Rd = Rr	Z = 1	BREQ	Rd ≠ Rr	Z = 0	BRNE	Signed
Rd ≤ Rr	Z+(N ⊕ V) = 1	BRGE(1)	Rd > Rr	Z•(N ⊕ V) = 0	BRLT*	Signed
Rd < Rr	(N ⊕ V) = 1	BRLT	Rd ≥ Rr	(N ⊕ V) = 0	BRGE	Signed
Rd > Rr	C + Z = 0	BRLO(1)	Rd ≤ Rr	C + Z = 1	BRSH*	Unsigned
Rd □ Rr	C = 0	BRSH/BRCC	Rd < Rr	C = 1	BRLO/BRCS	Unsigned
Rd = Rr	Z = 1	BREQ	Rd ≠ Rr	Z = 0	BRNE	Unsigned
Rd ≤ Rr	C + Z = 1	BRSH(1)	Rd > Rr	C + Z = 0	BRLO*	Unsigned
Rd < Rr	C = 1	BRLO/BRCS	Rd ≥ Rr	C = 0	BRSH/BRCC	Unsigned
Carry	C = 1	BRCS	No carry	C = 0	BRCC	Simple
Negative	N = 1	BRMI	Positive	N = 0	BRPL	Simple
Overflow	V = 1	BRVS	No overflow	V = 0	BRVC	Simple
Zero	Z = 1	BREQ	Not zero	Z = 0	BRNE	Simple

Note: 1. Interchange Rd and Rr in the operation before the test, i.e., CP Rd,Rr → CP Rr,Rd

Complete Instruction Set Summary

Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks	#Clocks XMEGA
Arithmetic and Logic Instructions						
ADD	Rd, Rr	Add without Carry	Rd ← Rd + Rr	Z,C,N,V,S,H	1	
ADC	Rd, Rr	Add with Carry	Rd ← Rd + Rr + C	Z,C,N,V,S,H	1	
ADIW[1]	Rd, K	Add Immediate to Word	Rd ← Rd + 1:Rd + K	Z,C,N,V,S	2	
SUB	Rd, Rr	Subtract without Carry	Rd ← Rd - Rr	Z,C,N,V,S,H	1	
SUBI	Rd, K	Subtract Immediate	Rd ← Rd - K	Z,C,N,V,S,H	1	
SBC	Rd, Rr	Subtract with Carry	Rd ← Rd - Rr - C	Z,C,N,V,S,H	1	
SBCI	Rd, K	Subtract Immediate with Carry	Rd ← Rd - K - C	Z,C,N,V,S,H	1	
SBIW[1]	Rd, K	Subtract Immediate from Word	Rd + 1:Rd ← Rd + 1:Rd - K	Z,C,N,V,S	2	
AND	Rd, Rr	Logical AND	Rd ← Rd • Rr	Z,N,V,S	1	
ANDI	Rd, K	Logical AND with Immediate	Rd ← Rd • K	Z,N,V,S	1	
OR	Rd, Rr	Logical OR	Rd ← Rd v Rr	Z,N,V,S	1	
ORI	Rd, K	Logical OR with Immediate	Rd ← Rd v K	Z,N,V,S	1	
EOR	Rd, Rr	Exclusive OR	Rd ← Rd ⊕ Rr	Z,N,V,S	1	
COM	Rd	One's Complement	Rd ← $FF - Rd	Z,C,N,V,S	1	
NEG	Rd	Two's Complement	Rd ← $00 - Rd	Z,C,N,V,S,H	1	
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V,S	1	
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • ($FFh - K)	Z,N,V,S	1	
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V,S	1	
DEC	Rd	Decrement	Rd ← Rd - 1	Z,N,V,S	1	
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V,S	1	
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V,S	1	
SER	Rd	Set Register	Rd ← $FF	None	1	
MUL[1]	Rd,Rr	Multiply Unsigned	R1:R0 ← Rd x Rr (UU)	Z,C	2	
MULS[1]	Rd,Rr	Multiply Signed	R1:R0 ← Rd x Rr (SS)	Z,C	2	
MULSU[1]	Rd,Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr (SU)	Z,C	2	
FMUL[1]	Rd,Rr	Fractional Multiply Unsigned	R1:R0 ← Rd x Rr<<1 (UU)	Z,C	2	
FMULS[1]	Rd,Rr	Fractional Multiply Signed	R1:R0 ← Rd x Rr<<1 (SS)	Z,C	2	
FMULSU[1]	Rd,Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← Rd x Rr<<1 (SU)	Z,C	2	
DES	K	Data Encryption	if (H = 0) then R15:R0 ← Encrypt(R15:R0, K) else if (H = 1) then R15:R0 ← Decrypt(R15:R0, K)			1/2
Branch Instructions						
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2	
IJMP[1]		Indirect Jump to (Z)	PC(15:0) ← Z, PC(21:16) ← 0	None	2	
EIJMP[1]		Extended Indirect Jump to (Z)	PC(15:0) ← Z, PC(21:16) ← EIND	None	2	
JMP[1]	k	Jump	PC ← k	None	3	

Mnemonics	Operands	Description	Operation	Flags	#Clocks	#Clocks XMEGA
RCALL	k	Relative Call Subroutine	PC ← PC + k + 1	None	3 / 4[3][5]	2 / 3[3]
ICALL[1]		Indirect Call to (Z)	PC(15:0) ← Z, PC(21:16) ← 0	None	3 / 4[3]	2 / 3[3]
EICALL[1]		Extended Indirect Call to (Z)	PC(15:0) ← Z, PC(21:16) ← EIND	None	4 [3]	3 [3]
CALL[1]	k	call Subroutine	PC ← k	None	4 / 5[3]	3 / 4[3]
RET		Subroutine Return	PC ← STACK	None	4 / 5[3]	
RETI		Interrupt Return	PC ← STACK	I	4 / 5[3]	
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1 / 2 / 3	
CP	Rd,Rr	Compare	Rd - Rr	Z,C,N,V,S,H	1	
CPC	Rd,Rr	Compare with Carry	Rd - Rr - C	Z,C,N,V,S,H	1	
CPI	Rd,K	Compare with Immediate	Rd - K	Z,C,N,V,S,H	1	
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b) = 0) PC ← PC + 2 or 3	None	1 / 2 / 3	
SBRS	Rr, b	Skip if Bit in Register Set	if (Rr(b) = 1) PC ← PC + 2 or 3	None	1 / 2 / 3	
SBIC	A, b	Skip if Bit in I/O Register Cleared	if (I/O(A,b) = 0) PC ← PC + 2 or 3	None	1 / 2 / 3	2 / 3 / 4
SBIS	A, b	Skip if Bit in I/O Register Set	If (I/O(A,b) =1) PC ← PC + 2 or 3	None	1 / 2 / 3	2 / 3 / 4
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC ← PC + k + 1	None	1 / 2	
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC ← PC + k + 1	None	1 / 2	
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1 / 2	
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1 / 2	
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1 / 2	
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1 / 2	
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1 / 2	
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1 / 2	
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1 / 2	
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1 / 2	
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1 / 2	
BRLT	k	Branch if Less Than, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1 / 2	
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1 / 2	
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1 / 2	
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1 / 2	
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1 / 2	
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1 / 2	
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1 / 2	
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1 / 2	
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1 / 2	
Data Transfer Instructions						
MOV	Rd, Rr	Copy Register	Rd ← Rr	None	1	
MOVW[1]	Rd, Rr	Copy Register Pair	Rd+1:Rd ← Rr+1:Rr	None	1	
LDI	Rd, K	Load Immediate	Rd ← K	None	1	
LDS[1]	Rd, k	Load Direct from data space	Rd ← (k)	None	1[5]/2[3]	2[3][4]
LD[2]	Rd, X	Load Indirect	Rd ← (X)	None	1[5]2[3]	1[3][4]

Mnemonics	Operands	Description	Operation	Flags	#Clocks	#Clocks XMEGA
LD[2]	Rd, X+	Load Indirect and Post-Increment	Rd ← (X) X ← X + 1	None	2[3]	1[3][4]
LD[2]	Rd, -X	Load Indirect and Pre-Decrement	X ← X - 1, ← X - 1 Rd ← (X) ← (X)	None	2[3]/3[5]	2[3][4]
LD[2]	Rd, Y	Load Indirect	Rd ← (Y) ← (Y)	None	1[5]/2[3]	1[3][4]
LD[2]	Rd, Y+	Load Indirect and Post-Increment	Rd ← (Y) Y ← Y + 1	None	2[3]	1[3][4]
LD[2]	Rd, -Y	Load Indirect and Pre-Decrement	Y ← Y - 1 Rd ← (Y)	None	2[3]/3[5]	2[3][4]
LDD[1]	Rd, Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2[3]	2[3][4]
LD[2]	Rd, Z	Load Indirect	Rd ← (Z)	None	1[5]/2[3]	1[3][4]
LD[2]	Rd, Z+	Load Indirect and Post-Increment	Rd ← (Z), Z ← Z+1	None	2[3]	1[3][4]
LD[2]	Rd, -Z	Load Indirect and Pre-Decrement	Z ← Z - 1, Rd ← (Z)	None	2[3]/3[5]	2[3][4]
LDD[1]	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2[3]	2[3][4]
STS[1]	k, Rr	Store Direct to Data Space	(k) ← Rd	None	1[5]/2[3]	2[3]
ST[2]	X, Rr	Store Indirect	(X) ← Rr	None	1[5]/2[3]	1[3]
ST[2]	X+, Rr	Store Indirect and Post-Increment	(X) ← Rr, X ← X + 1	None	1[5]/2[3]	1[3]
ST[2]	-X, Rr	Store Indirect and Pre-Decrement	X ← X - 1, (X) ← Rr	None	2[3]	2[3]
ST[2]	Y, Rr	Store Indirect	(Y) ← Rr	None	1[5]/2[3]	1[3]
ST[2]	Y+, Rr	Store Indirect and Post-Increment	(Y) ← Rr, Y ← Y + 1	None	1[5]/2[3]	1[3]
ST[2]	-Y, Rr	Store Indirect and Pre-Decrement	Y ← Y - 1, (Y) ← Rr	None	2[3]	2[3]
STD[1]	Y+q, Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2[3]	2[3]
ST[2]	Z, Rr	Store Indirect	(Z) ← Rr	None	1[5]/2[3]	1[3]
ST[2]	Z+, Rr	Store Indirect and Post-Increment	(Z) ← Rr Z ← Z + 1	None	1[5]/2[3]	1[3]
ST[2]	-Z, Rr	Store Indirect and Pre-Decrement	Z ← Z - 1	None	2[3]	2[3]
STD[1]	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2[3]	2[3]
LPM[1][2]		Load Program Memory	R0 ← (Z)	None	3	3
LPM[1][2]	Rd, Z	Load Program Memory	Rd ← (Z)	None	3	3
LPM[1][2]	Rd, Z+	Load Program Memory and Post-Increment	Rd ← (Z), Z ← Z + 1	None	3	3
ELPM[1]		Extended Load Program Memory	R0 ← (RAMPZ:Z)	None	3	
ELPM[1]	Rd, Z	Extended Load Program Memory	Rd ← (RAMPZ:Z)	None	3	
ELPM[1]	Rd, Z+	Extended Load Program Memory and Post-Increment	Rd ← (RAMPZ:Z), Z ← Z + 1	None	3	
SPM[1]		Store Program Memory	(RAMPZ:Z) ← R1:R0	None	-	-
SPM[1]	Z+	Store Program Memory and Post-Increment by 2	(RAMPZ:Z) ← R1:R0, Z ← Z + 2	None	-	-
IN	Rd, A	In From I/O Location	Rd ← I/O(A)	None	1	
OUT	A, Rr	Out To I/O Location	I/O(A) ← Rr	None	1	
PUSH[1]	Rr	Push Register on Stack	STACK ← Rr	None	2	1[3]
POP[1]	Rd	Pop Register from Stack	Rd ← STACK	None	2	2[3]

Mnemonics	Operands	Description	Operation	Flags	#Clocks	#Clocks XMEGA
Bit and Bit-test Instructions						
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0, C ← Rd(7)	Z,C,N,V,H	1	
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0, C ← Rd(0)	Z,C,N,V	1	
ROL	Rd	Rotate Left Through Carry	Rd(0) ← C, Rd(n+1) ← Rd(n), C ← Rd(7)	Z,C,N,V,H	1	
ROR	Rd	Rotate Right Through Carry	Rd(7) ← C, Rd(n) ← Rd(n+1), C ← Rd(0)	Z,C,N,V	1	
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1	
SWAP	Rd	Swap Nibbles	Rd(3..0) ↔ Rd(7..4)	None	1	
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1	
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1	
SBI	A, b	Set Bit in I/O Register	I/O(A, b) ← 1	None	1[5]/2	1
CBI	A, b	Clear Bit in I/O Register	I/O(A, b) ← 0	None	1[5]/2	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1	
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1	
SEC		Set Carry	C ← 1	C	1	
CLC		Clear Carry	C ← 0	C	1	
SEN		Set Negative Flag	N ← 1	N	1	
CLN		Clear Negative Flag	N ← 0	N	1	
SEZ		Set Zero Flag	Z ← 1	Z	1	
CLZ		Clear Zero Flag	Z ← 0	Z	1	
SEI		Global Interrupt Enable	I ← 1	I	1	
CLI		Global Interrupt Disable	I ← 0	I	1	
SES		Set Signed Test Flag	S ← 1	S	1	
CLS		Clear Signed Test Flag	S ← 0	S	1	
SEV		Set Two's Complement Overflow	V ← 1	V	1	
CLV		Clear Two's Complement Overflow	V ← 0	V	1	
SET		Set T in SREG	T ← 1	T	1	
CLT		Clear T in SREG	T ← 0	T	1	
SEH		Set Half Carry Flag in SREG	H ← 1	H	1	
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1	
MCU Control Instructions						
BREAK[1]		Break	(See specific descr. for BREAK)	None	1	
NOP		No Operation		None	1	
SLEEP		Sleep	(see specific descr. for Sleep)	None	1	
WDR		Watchdog Reset	(see specific descr. for WDR)	None	1	

Notes: 1. This instruction is not available in all devices. Refer to the device specific instruction set summary.
2. Not all variants of this instruction are available in all devices. Refer to the device specific instruction set summary.
3. Cycle times for Data memory accesses assume internal memory accesses, and are not valid for accesses via the external RAM interface.

Memoria de Programa
Vectores de Interrupción en el ATmega16

Vector No.	Program Address[2]	Source	Interrupt Definition
1	$000[1]	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$002	INT0	External Interrupt Request 0
3	$004	INT1	External Interrupt Request 1
4	$006	TIMER2 COMP	Timer/Counter2 Compare Match
5	$008	TIMER2 OVF	Timer/Counter2 Overflow
6	$00A	TIMER1 CAPT	Timer/Counter1 Capture Event
7	$00C	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	$00E	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	$010	TIMER1 OVF	Timer/Counter1 Overflow
10	$012	TIMER0 OVF	Timer/Counter0 Overflow
11	$014	SPI, STC	Serial Transfer Complete
12	$016	USART, RXC	USART, Rx Complete
13	$018	USART, UDRE	USART Data Register Empty
14	$01A	USART, TXC	USART, Tx Complete
15	$01C	ADC	ADC Conversion Complete
16	$01E	EE_RDY	EEPROM Ready
17	$020	ANA_COMP	Analog Comparator
18	$022	TWI	Two-wire Serial Interface
19	$024	INT2	External Interrupt Request 2
20	$026	TIMER0 COMP	Timer/Counter0 Compare Match
21	$028	SPM_RDY	Store Program Memory Ready

CARACTERÍSTICAS DEL REGULADOR LM7805

Parameter	Symbol	Conditions		MC7805/LM7805 Min.	Typ.	Max.	Unit
Output Voltage	V_O	T_J =+25 °C		4.8	5.0	5.2	V
		5.0mA ≤ I_O ≤ 1.0A, P_O ≤ 15W V_I = 7V to 20V		4.75	5.0	5.25	
Line Regulation (Note1)	Regline	T_J=+25 °C	V_O = 7V to 25V	-	4.0	100	mV
			V_I = 8V to 12V	-	1.6	50	
Load Regulation (Note1)	Regload	T_J=+25 °C	I_O = 5.0mA to1.5A	-	9	100	mV
			I_O =250mA to 750mA	-	4	50	
Quiescent Current	I_Q	T_J =+25 °C		-	5.0	8.0	mA
Quiescent Current Change	ΔI_Q	I_O = 5mA to 1.0A		-	0.03	0.5	mA
		V_I= 7V to 25V		-	0.3	1.3	
Output Voltage Drift	$\Delta V_O/\Delta T$	I_O= 5mA		-	-0.8	-	mV/ °C
Output Noise Voltage	V_N	f = 10Hz to 100KHz, T_A=+25 °C		-	42	-	µV/Vo
Ripple Rejection	RR	f = 120Hz V_O = 8V to 18V		62	73	-	dB
Dropout Voltage	V_{Drop}	I_O = 1A, T_J =+25 °C		-	2	-	V
Output Resistance	r_O	f = 1KHz		-	15	-	mΩ
Short Circuit Current	I_{SC}	V_I = 35V, T_A =+25 °C		-	230	-	mA
Peak Current	I_{PK}	T_J =+25 °C		-	2.2	-	A