

**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN**

“Desarrollo de plataforma interactiva para la demostración de instrucciones en Lenguaje

Ensamblador utilizado con los microcontroladores de Atmel.”

**TESINA DE SEMINARIO**

Previa la obtención del Título de:

**Ingeniero en Electricidad Especialización en Electrónica y Automatización**

**Industrial**

Presentado por:

Ronald Romulo Rivera Murillo

Nelson Hernan Castro Reyes

GUAYAQUIL – ECUADOR AÑO 2012

I

**AGRADECIMIENTO**

A Dios.

A nuestros Padres.

A nuestro Profesor Ing. Carlos Valdivieso.

Y todas las personas que de alguna u otra manera nos brindaron su apoyo para la implementación de este proyecto.

II

**DEDICATORIA**

A Dios por darme las fuerza para seguir adelante y la oportunidad de finalizar esta etapa de mi vida de estudios.

A mis padres por el apoyo incondicional que me brindaron en esta etapa de mi vida y a todas las personas que de alguna manera me han brindado su apoyo por sobre todas las cosas, su comprensión, a mis profesores que han compartido sus conocimientos.

***Ronald Rómulo Rivera M.***

III

**DEDICATORIA**

A toda mi familia en especial a mis padres y al Profesor Ing. Carlos Valdivieso por el apoyo incondicional que me brindaron

***Nelson Hernán Castro Reyes.***

IV

**TRIBUNAL DE SUSTENTACIÓN**

Ing. Carlos Valdivieso

Profesor de Seminario de Graduación

Ing. Hugo Villavicencio V. Profesor Delegado del Decano

V

**DECLARACIÓN EXPRESA**

“La responsabilidad del contenido de esta tesina, nos corresponde exclusivamente; y el patrimonio intelectual del mismo a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL”.

(Reglamento de exámenes y títulos profesionales de la ESPOL)

Ronald Rómulo Rivera Murillo

Nelson Hernán Castro Reyes

VI

**RESUMEN**

La finalidad del presente proyecto es el desarrollo de una plataforma interactiva para la demostración de instrucciones en Lenguaje Ensamblador utilizado con los microcontroladores de Atmel para ello se utilizarán varias herramientas de Software tales como un programador para microcontroladores Atmel como lo es el AVR Studio

4 y un simulador de circuitos electrónicos como lo es el Proteus y para la implementación física se utilizará el **Kit de desarrollo AVR butterfly**, que contiene el siguiente hardware: un microcontrolador ATmega169V, LCD, Joystick, altavoz, cristal de 32 KHz, DataFlash de 4 Mbit, convertidor de nivel RS-232, interfaz USART, interfaz USI, sensor de temperatura, sensor de luz, ADC, conectores para acceso a periféricos, y Batería de 3 V.

Para el desarrollo del tema proyecto se ha escogido comenzar con el tratamiento del set de instrucciones. Para realizar dicho estudio se basa en el microcontrolador ATMEGA 169 analizando cada una de sus instrucciones aritméticas, lógicas, transferencia de datos, control y de bit realizando un resumen completo de todas las instrucciones de dicho microcontrolador.

En el capítulo uno se describe una breve comparación de los microcontroladores de atmel con los microcontroladores de microchip e Intel basado en el cada set de instrucciones que utiliza cada microcontrolador.

VII

El capítulo dos trata sobre las Herramientas software y hardware que afianzan la

implementación y optimización para desarrollo del Proyecto.

En el capítulo tres se describe cada aplicación desarrollada en lenguaje assembler donde se bosqueja cada aplicación a nuestra plataforma interactiva en el cual demostraremos las instrucciones del lenguaje ensamblador para los microcontroladores de la familia ATMEL.

En el capitulo 4 se mostrará los respectivos diagramas de bloques, diagramas de flujo

código de programación junto con las simulaciones y pruebas del proyecto.

VIII

**ÍNDICE GENERAL**

**Contenido**

**AGRADECIMIENTO .............................................................................................. I DEDICATORIA. .................................................................................................... II TRIBUNAL DE SUSTENTACIÓN .....................................................................IIV DECLARACIÓN EXPRESA ..................................................................................V RESUMEN ...........................................................................................................VII ÍNDICE GENERAL ........................................................................................... VIII ÍNDICE DE GRAFICOS .................................................................................... XII INTRODUCCIÓN ............................................................................................. XXV CAPÍTULO 1........................................................................................................... 1**

**SET DE INSTRUCCIONES ................................................................................... 1**

**1.1 El conjunto de instrucciones AVR y su importancia ..................................... 2**

**1.2 Programa y modos de direccionamiento de datos ......................................... 2**

**1.2.1 Direccionamiento de Registro Directo (Registro Único) .............................. 3**

**1.2.2 Direccionamiento de Registro directo (dos registros) ................................... 4**

**1.2.3 Direccionamiento de E / S directo .................................................................. 5**

**1.2.4 Modos de direccionamiento directos ............................................................. 6**

IX

**1.2.5 Modos de direccionamiento indirectos........................................................... 7**

**1.2.6 Direccionamiento de programas directo ........................................................ 8**

**1.2.7 Direccionamiento de programas relativos .................................................... 9**

**1.3 Tipos de Instrucciones del Atmega169 ....................................................... 10**

**1.3.1 Instrucciones aritméticas y lógicas………………………………………….11**

**1.3.2 Instrucciones de control del programa ....................................................... 11**

**1.3.3 Instrucciones de transferencia de datos ...................................................... 12**

**1.3.4 Instrucciones de bits .................................................................................... 13**

**1.4 Breve comparación entre las instrucciones de los microcontroladores:**

**ATMEL (ATMEGA 169), MICROCHIP (PIC16F877) e INTEL (8051) ......... 13**

**CAPÍTULO 2......................................................................................................... 15**

**FUNDAMENTACION TEÓRICA ....................................................................... 15**

**2.1 Herramientas de software para el desarrollo del Proyecto……………….15**

**2.1.1 AVRstudio4………………………..………………………………………….15**

**2.1.2 Descripcion General del Ide en AVR STUDIO..…………………………...17**

**2.1.3 Simulador en AVR STUDIO 4………….…………………………………...18**

**2.1.4 Proteus versión 7.4 Portable……….………………………………………...19**

**2.2 Herramientas de Hardware para la Implementación del Proyecto……...20**

**2.2.1 Kit De Desarrollo Avr Butterfly……………………………………………..20**

**2.2.2 Hardware disponible en el Kit Avr Butterfly………………………………22**

**2.2.3 Firmware incluido en el Kit Avr Butterfly ……………………...…………23**

**2.2.4 Programacion mediante conexión serial(uart) con la PC…..…...…………26**

**2.2.5 El lcd del Kit Avr Butterfly……………….……………………...………….27**

**2.2.6 Conexiones entre el lcd y el microntrolador atmega169 en el Kit Avr**

**Butterfly27………………………………………………………………………......27**

X

**2.2.7 Fuente de Alimentacion Externa……………..…………………...…………29**

**CAPÍTULO 3......................................................................................................... 30**

**DISEÑO E IMPLEMENTACIÓN DEL PROYECTO ........................................ 30**

**3.1 Generalidades……………………………….………………………………...30**

**3.2 Implementación………………………………………….……………………30**

**3.3 Plataforma interactiva………………………………………………………..31**

**3.4 Materiales Utilizados………………………………………………………….34**

**3.5 Breve bosquejo de los Ejercicios………………………………………..........34**

**CAPÍTULO 4......................................................................................................... 38**

**DESARROLLO Y SIMULACIÓN DEL PROYECTO ....................................... 38**

**4 Desarrollo de los Ejercicios …………………………………………………38**

**4.1 Dados Electrónicos…………………………………………………………...38**

**4.1.1 Diagrama de Bloque Dados electrónicos……………………………………38**

**4.1.2 Algoritmo Dados Electrónicos……………………………………………….39**

**4.1.3 Programación Dados electrónicos…………………………………………...42**

**4.1.4 Simulación Dados electrónicos………………………………………………50**

**4.2 Semáforos Vehículos – peatones…………………………………………….50**

**4.2.1 Diagrama de Bloque Circuito Semáforos Vehículos - peatones………….51**

**4.2.2 Algoritmo Semáforos Vehículos - peatones………………………………...52**

**4.2.3 Programación Semáforos Vehículos - peatones……………………………55**

**4.2.4 Simulación del Semáforos vehículos-peatones……………………………..67**

**4.3 Cerradura electrónica………………………………………………………..67**

**4.3.1 Diagrama de Bloque Cerradura Electronica……………………………….68**

**4.3.2 Algoritmo de Cerradura Electronica…………….…………………………68**

XI

**4.3.3 Programación Cerradura electrónica………………………………………72**

**4.3.4 Simulacion en Proteus Cerradura Electronica…………………………….88**

**4.4 Maquina Electronica de bebidas…………………………………………….89**

**4.4.1 Diagrama de Bloques Maquinas Electronica de bebidas……………..…...89**

**4.4.2 Algoritmo Maquina Electronica de bebidas………………………………..90**

**4.4.3 Programacion Maquina Electronica de bebidas...…………………………94**

**4.4.4 Simulacion en Proteus Maquinas Electronica de bebidas..………………111**

**4.5 Calculadora Binaria……..………………………………………..…………112**

**4.5.1 Diagrama de Bloques Calculadora Binaria..………………………………112**

**4.5.2 Algoritmo Calculadora Binaria………...………..…………………………112**

**4.5.3 Programacion Calculadora Binaria...………………………………………117**

**4.5.4 Simulacion en Proteus Calculadora Binaria……………………………….128**

**CONCLUSIONES ...............................................................................................129**

**RECOMENDACIONES………………………………………………………..…131**

**ANEXOS...............................................................................................................132**

**BIBLIOGRAFÍA ..................................................................................................137**

XII

**ÍNDICE DE GRÁFICOS**

Figura 1.1.: Direccionamiento de registro directo (Registro Único)…..………………4

Figura 1.2.: Direccionamiento De Registro Directo (Dos Registros)...……………….5

Figura 1.3.: Direccionamiento De Registro I/O Directo……………..……………......6

Figura 1.4.: Acceso Directo De Memoria De Datos…………………..………………7

Figura 1.5.: Acceso indirecto memoria de datos……………………...……………….8

Figura 1.6.: Direccionamiento De Programas Directo………………...……………....9

Figura 1.7.: Direccionamiento De Programa Relativo………………...……………..10

Figura 2.1.: AVR Studio 4……………………………………………...…………....17

Figura 2.2.: Entorno de Desarrollo Integrado AVR Studio4…………...…………….18

Figura 2.3.: AVR Studio, selección del Dispositivo a Simular ………...……………19

Figura 2.4.: Interfaz gráfica de Proteus………………………………...…………….20

Figura 2.5.: Kit AVR Butterfly………………………………………...…………….21

Figura 2.6.: Hardware Disponible (Parte Frontal)……………………………………23

Figura 2.7.: Hardware Disponible (Parte Posterior)………………………………….23

Figura 2.8.: Firmware Incluido en el AVR Butterfly (en español)…………………..25

Figura 2.9.: Conexiones para interfaz USART del AVR Butterfly………….………27

Figura 2.10.: Vidrio LCD…………………………………………………………….28

Figura 2.11.: Segmentos y Letras de Referencia de los Dígitos LCD……….……….28

Figura2.12.: AVR Butterfly, PORTB (a) y PORTD (b)……………………………..29

Figura 2.13.: Baterías AA de 1.5 V y porta-baterías……………………….………...29

Figura 3.1.: Prototipo de plataforma interactiva……………….………………….....32

XIII

Figura 3.2.: Esquema de conexiones de la plataforma interactiva……..…………….33

Figura 4.1.: Diagrama de Bloque Dados Electrónicos……………………..……..….39

Figura 4.2 .: Diagrama ASM Dados electrónicos…………………………..……......39

Figura 4.3.: Diagrama ASM subrutina para dados1………………………...………..40

Figura 4.4.: Diagrama ASM subrutina para dados2………………………...………. 41

Figura 4.5.: Circuito en proteus Dados electrónicos…………..……………………..50

Figura 4.6.: Diagrama de Bloque Semáforo………………………………………….51

Figura 4.7.: Diagrama ASM Semáforo……………………………………...……….52

Figura 4.8.: Diagrama ASM Semáforo calle 1……………………………………….53

Figura 4.9.: Diagrama ASM Semáforo calle 2……………………………………….54

Figura 4.10.: Circuito de proteus Semáforo…………………..……………………...67

Figura 4.11.: Diagrama de Bloque Cerradura electrónica…….……………………..68

Figura 4.12.: Diagrama ASM Cerradura electrónica………………………………...69

Figura 4.13.: (continuación) Diagrama ASM Cerradura electrónica……………….70

Figura 4.14.: Diagrama ASM cambio de clave……………………………………...71

Figura 4.15.: Circuito de proteus Cerradura electrónica…………………….……..88

Figura 4.16.: Diagrama de Bloque Maquina electrónica de bebidas………….……89

Figura 4.17.: Diagrama ASM Maquinas electrónica de bebidas………………..…...90

Figura 4.18.: (continuación) Diagrama ASM Maquina electrónica de bebidas…….91

Figura 4.19.: Diagrama ASM pedir dinero Maquina electrónica de bebidas……...…92

Figura 4.20.: Diagrama ASM rotar segmentos en display………………………...…93

Figura 4.21.: Circuito de proteus Maquinas electrónica de bebidas…………….....111

Figura 4.22.: Diagrama de Bloque Calculadora Binaria…………………………....112

Figura 4.23.: Diagrama ASM Calculadora Binaria. ……………………………….113

Figura 4.24.: (continuación) Diagrama ASM Calculadora Binaria……………...…114

XIV

Figura 4.25.: (continuación) Diagrama ASM Calculadora Binaria ………...….…..115

Figura 4.26.: (continuación) Diagrama ASM Calculadora Binaria ………………..116

Figura 4.27.: Circuito de proteus Calculadora Binaria………………….…………..128

XV

**INTRODUCCIÓN**

El objetivo del proyecto es desarrollar e implementar una serie de ejercicios claves que permitan comprender toda la teoría y el funcionamiento referente a las diversas instrucciones de los Microcontroladores atmel de ocho bit. Tratando de darle un uso las diferentes herramientas que dispone el AVR Butterfly y demás elementos; para de este modo facilitar la comprensión y entendimiento de Desarrollo de plataforma interactiva para la demostración de instrucciones en Lenguaje Ensamblador utilizado con los microcontroladores de Atmel.

En el primer capítulo, se menciona una descripción general del proyecto, sus partes, la teoría detrás del funcionamiento del protocolo y los diferentes elementos que deben configurarse en el microcontrolador principal que contiene el AVR Butterfly.

En el segundo capítulo, se da un detalle sobre las herramientas de hardware: AVR Butterfly y de las herramientas de software: AVR STUDIO4 con su compilador AVR GCC que permite usar archivos en C y la herramienta de simulación PROTEUS.

El tercer capítulo, trata del diseño e implementación del proyect o, una breve descripción y una lista de materiales de los elementos que van a contener cada uno de los ejercicios planteados en el proyecto.

XVI

En el cuarto y último capítulo se detalla, desarrolla e implementa de manera independiente cada ejercicio propuesto en el proyecto; incluyendo el diagrama de bloques, diagrama de flujo, descripción del algoritmo y simulaciones.

**Capítulo 1**

**SET DE INSTRUCCIONES**

Para la ejecución de este proyecto se ha creado una plataforma interactiva que permitirá la demostración de las diversas instrucciones de los Microcontroladores atmel de ocho bit. Para el desarrollo del tema propuesto se ha escogido comenzar con el tratamiento del set de instrucciones. para realizar dicho estudio nos basamos en el microcontrolador ATMEGA 169 analizando cada una de sus instrucciones aritméticas, lógicas, transferencia de datos, control y de bit realizando un resumen completo de todas las instrucciones de dicho microcontrolador también analizaremos los diversos modos de direccionamiento de los datos en esta arquitectura y finalizaremos con una breve comparación entre tres microcontroladores de 8 bits y poseen arquitectura Harvard (es decir, existen espacios de direcciones separados para código y datos). Y para realizar esta comparación utilizaremos los microcontroladores:

16F877 (MICROCHIP)

8051 (INTEL)h

ATMEGA169 (ATMEL)

2

**1.1 EL CONJUNTO DE INSTRUCCIONES AVR Y SU**

**IMPORTANCIA.**

El conjunto de instrucciones de un procesador o un controlador es como el vocabulario del procesador. Cada instrucción controla alguna parte del procesador y permite a los programadores manipular los datos en la memoria, así como también a los dispositivos de entrada y de salida. Las instrucciones del procesador se pueden clasificar de muchas maneras atendiendo a la forma como las instrucciones operan y acceden sobre los datos. Esto se llama el programa y los modos de direccionamiento de datos del procesador.

**1.2 PROGRAMA Y MODOS DE DIRECCIONAMIENTO DE**

**DATOS**

Las instrucciones AVR también se pueden clasificar por sus diferentes modos de direccionamiento. Se puede decir que un modo de direccionamiento es un mecanismo que permite conocer la ubicación de un dato o instrucción. Ya que los operandos y resultados de una instrucción son accedidos a través de los modos de direccionamientos. Los microcontroladores AVR RISC soportan modos de direccionamiento poderosos y eficaces para el acceso a la memoria de programa

(FLASH) y memoria de datos (SRAM, archivo de Registro y memoria de I/O). Esta sección describe los diferentes modos de direccionamiento soportados por la arquitectura AVR. En las figuras, OP es el código de funcionamiento de la palabra

(Word) de la instrucción. Para simplificar, no todas las figuras muestran la situación

exacta de los bits de direccionamiento.

3

**1.2.1 DIRECCIONAMIENTO DE REGISTRO DIRECTO**

**(REGISTRO ÚNICO).**

Las instrucciones de registro directo pueden operar en cualquiera de los 32 registros del archivo de registro. Se lee el contenido de un registro, opera en el contenido del registro, y después, almacena el resultado de la operación de nuevo en el mismo registro. La figura 1.1 ilustra el origen y el destino de este tipo de instrucciones. Ejemplos de algunas de este tipo de instrucciones son los siguientes:

Rd es cualquier registro del archivo de registro y es el registro de destino para la

operación.

COM Rd: complemento a 1 (invertir todos los bits) del registro Rd. se vuelve a

almacenar en el registro Rd.

INC Rd: Incrementa el contenido de Rd en uno.

CLR Rd: Cargas $ 00 en el registro Rd.

LSL Rd: Cambia el contenido del registro Rd un lugar a la izquierda. Un "0" se

desplaza en posición de bit 0, y el contenido de bit7 se copian en la bandera de acarreo.

4

FIGURA 1-1 direccionamiento de registro directo (registro único).

**1.2.2 DIRECCIONAMIENTO DE REGISTRO DIRECTO (dos**

**registros).**

En este tipo de instrucciones dos registros están involucrados. Los dos registros se nombran como el registro de origen (Rs), y el registro de destino (Rd). La instrucción lee los dos registros y opera en su contenido y almacena el resultado de vuelta en el registro de destino. La Figura 1.2 ilustra el origen y el destino de este tipo de instrucciones. Los ejemplos de estas instrucciones son:

ADD Rd, Rs: suma dos registros (Rd + Rs) y el resultado lo almacena en el registro de

destino (Rd).

SUB Rd, Rs: resta dos registros (Rd- Rs) y el resultado lo almacena en el registro de

destino (Rd).

AND Rd, Rs: realiza la operación lógica AND entre Rd y Rs y el resultado lo almacena

en el respectivo registro de destino Rd.

MOV Rd, Rs: copia o mueve el contenido del registro Rs al registro de destino Rd.

5

FIGURA 1.2 direccionamiento de registro directo (dos registros).

**1.2.3 DIRECCIONAMIENTO DE REGISTRO E/S DIRECTO.**

Estas instrucciones se utilizan para acceder al espacio de E/S. Los registros de E/S sólo

se pueden acceder mediante estas instrucciones:

IN Rd, PORTADDRESS: carga el contenido del registro de E/S al registro de destino

(Rd)

OUT PORTADDRESS, Rs: carga el contenido del registro Rs al registro E/S.

Rd, Rs puede ser cualquiera de los 32 registros del archivo de registro y los registros de E/S puede ser cualquier registro de toda la gama de $00 a $3F (un total de 64 E/S de registros). La figura 1.3 ilustra este tipo de direccionamiento.

6

FIGURA 1.3 direccionamiento de registro I/O directo.

**1.2.4 MODOS DE DIRECCIONAMIENTO DIRECTOS.**

La instrucción contiene la dirección de memoria además de un campo Rd que contiene el registro destino o fuente. Así que un máximo de 64 Kb de memoria de datos se puede acceder mediante este tipo de instrucciones. Los ejemplos de estas instrucciones son las siguientes:

K es una dirección de 16 bits.

LDS Rd, K: almacena y copia el valor contenido en la dirección de memoria K en el

registro de destino (Rd).

STS K, Rs: almacena y copia el valor contenido en el registro (Rs) en la dirección de

memoria K. La figura 1.4 ilustra cómo las instrucciones de los datos directos operan.

7

FIGURA 1.4 Acceso directo de memoria de datos.

**1.2.5 MODOS DE DIRECCIONAMIENTO INDIRECTOS.**

Estos son similares al tipo de datos directos de las instrucciones, excepto que estas instrucciones son una sola palabra y un registro del puntero (X, Y o Z) que tiene la dirección base de la memoria de datos. A la dirección base en el registro del puntero, un desplazamiento se pueden agregar, así como algunas operaciones de incremento/decremento en el contenido del puntero. Los Ejemplos de estas instrucciones son:

X es el registro del puntero (un par de registros R26, R27).

Rd es el registro de destino.

LD Rd, X: el registro de destino (Rd) se carga con el contenido de la memoria de datos

al que apunta el registro X

LD Rd, X+: Rd es el registro de destino y se carga con el contenido de la memoria de datos que apunta el registro X, y después que la memoria se accede, en el registro X es incrementado.

8

La Figura 1.5 ilustra cómo una variante de las instrucciones de datos indirectos opera.

FIGURA 1.5 Acceso indirecto memoria de datos.

**1.2.6 DIRECCIONAMIENTO DE PROGRAMAS DIRECTO.**

La ejecución del programa salta a la posición de memoria indicada por el registro Z.los

Ejemplos de este tipo de instrucciones son:

ICALL: llamada indirecta de subrutina.

IJMP: salto indirecto a un lugar en la memoria del programa indicado por el registro Z.

9

La Figura 1.6 ilustra cómo las instrucciones indirectas del programa operan.

FIGURA 1.6 direccionamiento de programas directo

**1.2.7 DIRECCIONAMIENTO DE PROGRAMAS RELATIVOS.**

La instrucción contiene un desplazamiento K que se suma al contenido del PC para encontrar la siguiente instrucción a ejecutar. (-2048 <K<2047).Estas instrucciones son del tipo:

RJMP label: salto relativo hacia la subrutina (label).

RCALL label: llamado relativo a la subrutina (label).

10

La Figura 1.7 ilustra cómo las instrucciones relativas programa operan.

FIGURA 1-7 direccionamiento de programa relativo.

**1.3 TIPOS DE INSTRUCCIONES DEL ATMEGA169**

El microcontrolador atmega169 es uno de los tantos microcontroladores que poseen la arquitectura (RISC), la arquitectura RISC nos dice que el microcontrolador tiene un conjunto de instrucciones reducido, todas las instrucciones hacen un total de ciento treinta, las cuales están divididas en cuatro grandes grupos como lo son:

Instrucciones aritméticas y lógicas.

Instrucciones de control del programa.

Instrucciones de transferencia de datos.

Instrucciones de bits.

11

**1.3.1 INSTRUCCIONES ARITMÉTICAS Y LÓGICAS .**

Las instrucciones aritméticas y lógicas realizan las operaciones indicadas (suma, resta con/sin acarreo, multiplicación con/sin signo, operaciones lógicas como la AND, OR, OR EXCLUSIVO, etc.) con los contenidos de los registros involucrados o a su vez también se pueden efectuar entre un registro y una constante. Este grupo de instrucciones se característica por modificar las banderas, de acuerdo al resultado. Algunos de los ejemplos de este tipo de instrucciones son:

ADD Rd, Rr realiza la suma entre los registros Rd y Rr, el resultado lo almacena en

Rd en pocas palabras: Rd ← Rd + Rr

SBC Rd, Rr realiza la substracción con acarreo entre el registro Rd y Rr, el resultado

lo almacena en Rd en pocas palabras: Rd ← Rd – Rr – C

OR Rd, Rr realiza la operación lógica OR entre los registros Rd y Rr, el resultado

es almacenado en Rd es decir: Rd ← Rd v Rr.

En el anexo se encuentra la tabla 1.1 donde podemos ver un resumen completo de todas

las instrucciones aritméticas y lógicas

**1.3.2 INSTRUCCIONES DE CONTROL DEL PROGRAMA**

Permiten modificar la secuencia normal de ejecución de un programa, puede hacerse por salto condicional relativo o absoluto, motivo por el cual algunos suelen llamarle instrucciones de salto también algunos programadores las clasifican en grupos dependiendo si el salto es condicional o incondicional es llamada de subrutina o regreso de interrupciones o subrutina, etc. Ejemplos de este tipo de instrucciones son:

12

RJMP K salto relativo a una ubicación en la memoria del programa en pocas palabras:

PC ← PC + k + 1

RETI retorno de interrupción es decir realiza la operación PC ← STACK SBIC P, b salta si el bit b del registro de E/S es seteado es decir if (P(b)=1) PC ← PC + 2 or 3

En el anexo se encuentra la tabla 1.2 donde podemos ver un resumen completo de

todas las instrucciones de control del programa.

**1.3.3 INSTRUCCIONES DE TRANSFERENCIA DE DATOS**

Copian datos de un origen (Rs o Rr, puntero X, Y o Z) a un destino (Rd, puntero X, Y o Z), sin modificar el origen y normalmente sin afectar a las banderas indicadoras de condición. Pueden transferir palabras, fracciones de palabras (bytes, media palabra) o bloques completos de n bytes o palabras. Algunos de los ejemplos de este tipo de instrucciones son:

MOV Rd, Rs realiza la copia del Rs en Rd es decir Rd ← Rs

ST X, Rr almacena registro en memoria apuntada por el puntero X

es decir (X) ← Rr

En el anexo se encuentra la tabla 1.3 donde podemos ver un resumen completo de todas

las instrucciones de transferencia de datos

13

**1.3.4 INSTRUCCIONES DE BITS.**

La característica principal es la de actuar sobre un bit del registro sin alterar a los demás. Los bits se numeran desde cero (0) hasta siete (7), de acuerdo a la posición que tienen dentro del byte: 0 el menos significativo (extremo derecho). 7 el más significativo (extremo izquierdo).algunos ejemplos de estos tipos de instrucciones son:

CBI A, b setea el bit b del puerto E/S es decir E/S (P, b) ← 1

BCLR s borrar la bandera de la posición s del registro estado. En resumen SREG(s)

← 0

SES setea la bandera de signo es decir S ← 1. En el anexo se encuentra la tabla 1.4

donde podemos ver un resumen completo de todas las instrucciones de bits.

**1.4 BREVE COMPARACIÓN ENTRE LAS INSTRUCCIONES DE LOS MICROCONTROLADORES: ATMEL (ATMEGA 169), MICROCHIP (16F877) E INTEL (8051)**

Los tres microcontroladores pertenecen a la familia de microcontroladores de 8 bits y poseen arquitectura Harvard (es decir la memoria de instrucciones y la de datos son independientes). Los microcontroladores ATMEGA169 (ATMEL) y PIC16F877

(MICROCHIP) poseen un conjunto reducido de instrucciones o también llamada tecnología RISC avanzada mientras que el 8051 (INTEL) no la posee ya que este posee la tecnología CISC (tienen un conjunto de instrucciones que se caracteriza por ser muy amplio y permitir operaciones complejas entre operandos situados en la memoria o en los registros internos).

14

Refiriéndonos al número de instrucciones en lenguaje ensamblador tenemos al microcontrolador ATMEGA169 con un repertorio de 130 instrucciones, el 8051 de INTEL tiene 50 instrucciones, mientras que el microcontrolador de microchip PIC16F877 solo posee 35 instrucciones siendo este el microcontrolador con menor número de instrucciones entre los tres esto no quiere decir que sea el peor.

En lo que respecta a rendimiento tenemos que el microcontrolador ATMEGA169 posee un rendimiento de hasta 16MIPS a 16Mhz, El PIC16f877 5MIPS a 20MHz mientras que el 8051 de INTEL se estima un rendimiento aproximado de 600.000 IPS a 12Mhz aunque en la realidad es exactamente 11.059Mhz. Como nos podemos dar cuenta el microcontrolador ATMEGA 169 de ATMEL tiene un mayor rendimiento con respecto al PIC16F877 de MICRCHIP y al 8051 de INTEL.

Los microcontroladores ATMEGA 169 y PIC 16F877 tienen un modo de vigilancia WATCH DOG o perro guardián Este módulo permite inicializar la CPU cuando se rebasa el contador, y permite recupera el sistema cuando se pierde el control del programa. El PÎC16F877 y el ATMEGA 169 se los puede llevar a un modo de stand by

(SLEEP) Deteniendo la CPU. La Principal ventaja es el consumo de la CPU baja a niveles muy bajo. Además el ATMEGA 169 posee 5 diferentes modos de SLEEP dependiendo de la aplicación se activa uno de estos modos (Idle, ADC Noise Reduction, Power-Down, Power-Save ó Stand by), además en el ATMEGA 169 SLEEP tiene su propio registro de control SMCR, el microcontrolador 8051 de INTEL carece de estos recursos.

**Capítulo 2**

**FUNDAMENTACION TEÓRICA**

En el presente capítulo se describe los fundamentos teóricos básicos de las herramientas de software y hardware utilizados para la implementación de este proyecto.

**2.1 HERRAMIENTAS DE SOFTWARE PARA EL DESARROLLO**

**DEL PROYECTO.**

Los Software para el desarrollo del proyecto serán descritos a continuación:

AVR Studio 4 de Atmel. Entorno de programación para el microcontrolador

ATmega169 del Kit de desarrollo avrbutterfly.

Proteus 7.7 SP2 Portable. Software para la simulación del proyecto para interactuar el

hardware y software de manera virtual.

**2.1.1 AVRSTUDIO 4**

AVR Studio 4 es el programa propio de ATMEL Corp. utilizado para compilar el

bootloader (en ensamblador).[2]

16

AVR Studio 4 proporciona herramientas para la administración de proyectos, edición de archivo fuente, simulación del chip e interfaz para emulación In-circuit para la poderosa familia RISC de microcontroladores AVR de 8 bits.

AVR es una familia de microcontroladores de 8 bits con una amplia gama de variantes

diferentes en:

Tamaño del programa de la memoria (flash)

Tamaño de la memoria EEPROM

Número de pines I / O

Número de servicios tales como características de los chips UART y ADC

Paquete de formularios

AVR Studio apoya al diseñador en el diseño, desarrollo, depuración y parte de la

comprobación del proceso.

AVR Studio 4 consiste de muchas ventanas y sub-módulos.

17

A continuación se puede apreciar la interfaz del AVR STUDIO4 como se ilustra en la figura 2.1

Figura 2.1 AVR Studio 4

**2.1.2 DESCRIPCIÓN GENERAL DEL IDE EN AVR STUDIO 4.[2]**

Como se dijo anteriormente, el AVR Studio es un Entorno de Desarrollo Integrado

(IDE). Éste tiene una arquitectura modular completamente nueva, que incluso permite interactuar con software de otros fabricantes. AVR Studio 4 proporciona herramientas para la administración de proyectos, edición de archivo fuente, simulación del chip e interfaz para emulación In-circuit para la poderosa familia RISC de microcontroladores AVR de 8 bits.

AVR Studio 4 consiste de muchas ventanas y sub-módulos. Cada ventana apoya a las

partes del trabajo que se intenta emprender.

18

En la Figura 2.2 se puede apreciar las ventanas principales del IDE.

Figura. 2.2 Entorno de Desarrollo Integrado AVR Studio4

El AVR Studio 4 soporta un amplio rango de herramientas para emulación y depuración. Por conveniencia el usuario debe seleccionar la funcionalidad de simulación incluida, el AVR Simulator; ya que, el resto de opciones requieren de H/W especial. Se debe seleccionar el microcontrolador que utilizará en su aplicación, en este caso ATmega169. El AVR Studio iniciará y abrirá un archivo .asm.

**2.1.3 SIMULADOR EN AVR STUDIO 4.[5]**

El AVR Simulator es un simulador para la arquitectura y dispositivos AVR. Este simula la CPU, incluyendo todas las instrucciones, interrupciones y la mayoría de los módulos de I/O del chip. [5]

19

A continuación en la figura 2.3 se muestra el avr simulador.

Figura 2.3 AVR Studio, selección del Dispositivo a Simular

El AVR Simulator permite al usuario usar los comandos normales de depuración tal como Run, Break, Reset, Single step, set breakpoints y watch variables. Las vistas I/O, Register y Memory son totalmente funcionales usando el AVR Simulator.

**2.1.4 PROTEUS VERSIÓN 7.4 PORTABLE**

Nos permite elaborar diseños de circuitos electrónicos complejos, donde la cual se incluye la captura de los esquemas y simulación realizados con microcontroladores de varios tipos.

El Programa PROTEUS es una aplicación CAD que se compone de tres módulos

básicos:

20

**ISIS** (“Esquema Inteligente de Sistema de Entrada”) que es el módulo de captura de

esquemas.

**VSM** (“Modelamiento de Sistema Virtual”) es el módulo de simulación, incluyendo

PROSPICE.

**ARES** (“Modelamiento Avanzado de Enrutamiento”) es el módulo para realización de

circuitos impresos (PCB).

Figura 2.4 Interfaz gráfica de Proteus

**2.2 HERRAMIENTAS DE HARDWARE PARA LA**

**IMPLEMENTACIÓN DEL PROYECTO.**

El Hardware para el desarrollo del proyecto se describe a continuación como lo es el **KIT DE DESARROLLO AVR BUTTERFLY t**tarjeta utilizada para la implementación de nuestra plataforma interactiva [1]

**2.2.1 KIT DE DESARROLLO AVR BUTTERFLY**

El Kit AVR Butterfly se diseñó para demostrar los beneficios y las características

importantes de los microcontroladores ATMEL. [3]

21

El AVR Butterfly utiliza el microcontrolador AVR ATmega169V, que combina la Tecnología Flash con el más avanzado y versátil microcontrolador de 8 bits disponible. En la Figura 2.5 se puede apreciar el Kit AVR Butterfly.

Figura 2.5 Kit AVR Butterfly

El Kit AVR Butterfly contiene características principales:

La arquitectura AVR en general y la ATmega169 en particular.

Diseño de bajo consumo de energía. El encapsulado tipo MLF. Periféricos:

Controlador LCD.

Memorias:

Flash, EEPROM, SRAM.

Data Flash externa.

Interfaces de comunicación:

UART, SPI, USI.

Métodos de programación.

Self-Programming/Bootloader, SPI, Paralelo, JTAG.

22

Convertidor Analógico Digital (ADC).

Timers/Counters:

Contador de Tiempo Real (RTC).

Modulación de Ancho de Pulso (PWM).

**2.2.2 HARDWARE DISPONIBLE EN EL KIT AVR BUTTERFLY.[1]** Los siguientes recursos están disponibles en el Kit AVR Butterfly: Microcontrolador ATmega169V (en encapsulado tipo MLF).

Pantalla tipo vidrio LCD de 120 segmentos, para demostrar las capacidades del

controlador de LCD incluido dentro del ATmega169.

Joystick de cinco direcciones, incluida la presión en el centro.

Altavoz piezoeléctrico, para reproducir sonidos.

Cristal de 32 KHz para el RTC.

Memoria DataFlash de 4 Mbit, para el almacenar datos.

Convertidor de nivel RS-232 e interfaz USART, para comunicarse con unidades fuera

del Kit sin la necesidad de hardware adicional.

Termistor de Coeficiente de Temperatura Negativo (NTC), para sensar y medir

temperatura.

Resistencia Dependiente de Luz (LDR), para sensar y medir intensidad luminosa. Acceso externo al canal 1 del ADC del ATmega169, para lectura de voltaje en el rango de 0 a 5 V.

Emulación JTAG, para depuración.

Interfaz USI, para una interfaz adicional de comunicación.

Terminales externas para conectores tipo Header, para el acceso a periféricos.

Batería de 3 V tipo botón (600mAh), para proveer de energía y permitir el

funcionamiento del AVR Butterfly.

23

Bootloader, para programación mediante la PC sin hardware especial.

Aplicación demostrativa preprogramada.

Compatibilidad con el Entorno de Desarrollo AVR Studio 4.

En las Figuras 2.6 y 2.7 se observa el Hardware disponible en el AVR Butterfly.

Figura 2.6 Hardware Disponible (Parte Frontal)

Figura 2.7 Hardware Disponible (Parte Posterior)

**2.2.3 FIRMWARE INCLUIDO EN EL KIT AVR BUTTERFLY[3]**

El AVR Butterfly viene con una aplicación pre programada. Esta sección presentará una

revisión de los elementos de esta aplicación.

Los siguientes bloques vienen preprogramados en el AVR Butterfly:

Código Cargador de Arranque (Bootloader Code).

Código de la Aplicación.

24

Máquina de Estados. Funciones incluidas: Nombre-etiqueta. Reloj (fecha).

Mediciones de temperatura.

Mediciones de luz.

Lecturas de voltaje.

Reproducción de tonadas/melodías. Ahorro de energía automático. Ajuste de contraste del LCD.

Más funciones podrán ser agregadas después, como por ejemplo:

Calculadora.

Función de recordatorio.

Alarma (alarmas diarias, temporizadores para la cocina, etc.).

Reproducción de melodías y visualización del texto (función de Karaoke).

Con la DataFlash de 4 Mb el usuario podrá almacenar una cantidad grande de datos.

25

La Figura 2.8 muestra el menú de la aplicación que viene con el AVR Butterfly. La

columna a la izquierda muestra el menú principal: “AVR Butterfly”, “Tiempo”,

“Música”, etc.

Figura. 2.8 Firmware Incluido en el AVR Butterfly (en español)

26

**2.2.4 PROGRAMACIÓN MEDIANTE CONEXIÓN SERIAL (UART) CON LA**

**PC [4]**

El AVR Butterfly tiene incluido un convertidor de nivel para la interfaz RS-232. Esto significa que no se necesita de hardware especial para reprogramar al AVR Butterfly utilizando la característica self-programming del ATmega169. A continuación se explica brevemente la distribución de los pines y como se debe realizar el cableado para la comunicación serial entre el AVR Butterfly y la PC.

La comunicación con la PC requiere de tres líneas: TXD, RXD y GND. TXD es la línea para transmitir datos desde la PC hacia el AVR Butterfly, RXD es la línea para recepción de datos enviados desde el AVR Butterfly hacia la PC y GND es la tierra común. En el anexo se encuentra la Tabla 2.1 donde se observa la distribución de los pines para la comunicación serial, a la izquierda los pines del AVR Butterfly y a la derecha los pines del conector DB9 de la PC. [8]

27

En la Figura 2.9 se observa como se debe hacer el cableado para la comunicación, a través de la interfaz serial RS-232, entre el AVR Butterfly y la PC. A la izquierda se aprecia un conector DB9 hembra soldado a los cables que se conectan a la interfaz USART del AVR Butterfly (derecha).

Figura. 2.9 Conexiones para interfaz USART del AVR Butterfly

**2.2.5 EL LCD DEL KIT AVR BUTTERFLY.**

Interfaz muy simple para mostrar información podría ser el estado de unos LEDs. El microcontrolador atmega169 tiene un controlador LCD (LCD Driver) integrado capaz de controlar hasta 100 segmentos. El núcleo altamente eficiente y el consumo de corriente muy bajo de este dispositivo lo hace ideal para aplicaciones energizadas por batería que requieren de una interfaz humana.

**2.2.6 CONEXIONES ENTRE EL LCD Y EL MICROCONTROLADOR**

**ATMEGA169 EN EL KIT AVR BUTTERFLY.[6]**

Los pines para el LCD en el AVR se sitúan en PORTA, PORTC, PORTD y PORTG. El vidrio LCD incluido en el AVR Butterfly tiene un total de 120 segmentos controlados a través de cuatro terminales comunes y 30 líneas de segmento. Puesto que el Atmega169

28

es capaz de manejar 100 segmentos, algunos de los segmentos del vidrio LCD no están

conectados en el AVR Butterfly.

En la Figura 2.10 se muestra el vidrio LCD montado en el AVR Butterfly. Este consiste de siete símbolos alfanuméricos y varios símbolos fijos: números de cero a nueve, una campana, un indicador de batería descargada (low-battery) y flechas de navegación

Figura. 2.10 Vidrio LCD

El AVR Butterfly tiene seis grupos similares de segmentos, donde cada grupo de segmentos es capaz de desplegar un carácter alfanumérico. Un cierto grupo de segmentos capaz de desplegar un carácter alfanumérico es llamado un dígito LCD. Este consiste de 14 segmentos separados. La Figura 2.11 muestra un dígito LCD y la letra usada para referirse a cada uno de los segmentos dentro del dígito LCD.

Figura. 2.11 Segmentos y Letras de Referencia de los Dígitos LCD

29

El LCD debe funcionar a ¼ Duty y 1/3 Bias (ver Capítulo 2, Controlador de LCD) para poder controlar los cuatro terminales comunes. Es muy importante notar que la energía para el LCD es suministrada desde el microcontrolador ATmega169 y no tienen líneas de alimentación separadas.

**2.2.7 FUENTE DE ALIMENTACIÓN EXTERNA.**

El avrbutterfly puede ser alimentado/energizado por una fuente de voltaje externa de 3 v como se muestra en la figura 2.12 a través de los pines vcc\_ext y gnd de los conectores externos portb y portd.

Figura. 2.12 AVR Butterfly, PORTB (a) y PORTD (b)

Se recomienda utilizar dos baterías tipo AA de 1.5 V y un porta-baterías para las

mismas, tal como se muestra en la Figura 2.13

Figura 2.13 Baterías AA de 1.5 V y porta-baterías

**Capítulo 3**

**DISEÑO E IMPLEMENTACIÓN DEL PROYECTO**

**3.1 GENERALIDADES**

En el presente capitulo tendremos una visión global de nuestro proyecto el cual se trata de una práctica plataforma interactiva en la cual demostraremos las instrucciones del lenguaje ensamblador para los microcontroladores de la familia ATMEL y lo haremos por medio del AVR - BUTTERFLY el cual contiene al microcontrolador Atmega 169 y una serie de elementos (resistencias, Leds , displays, etc.). La demostración de las instrucciones se la hará por medio de cinco sencillos pero prácticos ejercicios en los cuales hemos hecho lo posible para utilizar los diferentes tipos de instrucciones de dicho microcontrolador ya sean este tipo de instrucciones de tipo aritméticas, lógicas, transferencia de datos, control del programa y bits.

**3.2 IMPLEMENTACIÓN**

Uno de los principales problemas en al realización de este proyecto fue ¿Cómo Desarrollar una plataforma interactiva para demostrar las instrucciones en Lenguaje Ensamblador utilizando microcontroladores ATMEL? Nosotros pensamos que la

31

mejor manera para la demostración de las instrucciones de los microcontroladores ATMEL es desarrollando un conjunto de ejercicios. donde cada uno estos ejercicios esta destinado a resolver un problema especifico, los cuales están integrados en una plataforma compacta, plataforma que probablemente para los expertos en este tipo de microcontroladores podría ser cuestionados como una plataforma demasiado sencilla o básica. Pero es de gran utilidad para las personas que estén dando sus primeros pasos con este tipo de microcontroladores ya que por medio de este conjunto de ejercicios podrán observar y comprobar a través de software (AVR Studio 4, Proteus) y hardware

(AVR Butterfly, Leds, pulsadores, displays, etc ) las diversas aplicaciones de las instrucciones en lenguaje Ensamblador de los microcontroladores de la familia ATMEL.

**3.3 PLATAFORMA INTERACTIVA**

En la presente plataforma la cual se la puede apreciar en la figura 3.1, como podemos observar consta de un teclado matricial sencillo formado por resistencias y botoneras, juego de leds que hacen las veces de indicadores, y un display en el cual se muestra valores. En la plataforma interactiva se desarrollaran problemas como:

Dados electrónicos

Semáforos vehículos-peatones Cerradura electrónica Calculadora Binaria

Maquinas de bebidas electrónica

32

Las aplicaciones que se los ha desarrollado de la forma más didáctica posible para ver

de una manera clara el comportamiento del lenguaje Ensamblador.

A continuación en la siguiente figura 3.1 se muestra la plataforma interactiva.

Figura 3.1 Prototipo de plataforma interactiva.

33

A continuación se muestra el esquema de nuestra plataforma interactiva donde se aprecia el bosquejo de cada ejercicio aplicado a dicha plataforma junto con las conexiones definidas en la tarjeta de desarrollo avr butterfly.

A V R B U T T E R F L Y

**UART**

**P U E R T O E**

L C D

4

5

6

**J O Y S T IC K**

RR R

R R R R

RR R R

RR R R

1 3 5 7 + +

0 2 4 6 \_ \_

**P U E R T O B**

**R E S E T**

R R R R

R R R R

**D a d o s E le c t r o n ic o s**

P B 7 P E 6

RR R

RR R R

**1**

PB 0

**2 3 / +**

**M a q u in a v e n d e d o r a d e B e b id a s**

PB 1

**4 / E X O R 5**

**6 / -**

**+ VCC**

P E 4

**7 /OR 8**

**9 / x**

PB 2

**AND 0 =**

**+ VCC**

P E 5

PB 3

PB 4 R 1

PB 5 R 1

PB 6 R 1

**S E M A F O R O S**

**C A L L E - A V E N I D A - P E A T O N E S**

**C e r r a d u r a e le c t r ó n ic a / C a lc u la d o r a B in a r ia**

Figura 3.2 esquema de conexiones de la plataforma interactiva

34

**3.4 MATERIALES UTILIZADOS**

Los materiales utilizados para montar esta plataforma interactiva son los siguientes:

Kit Avr butterfly (Atmega 169, LCD)

Protoboard Resistencias pull- up Resistencias

Leds

Displays de 7 segmentos cátodo común

Pulsadores

Baterías alcalinas AA porta baterías Jumpers

Cables utp

**3.5 BREVE BOSQUEJO DE LOS EJERCICIOS.**

Como ya lo sabíamos esta plataforma consta de cinco ejercicios orientados

a resolver un problema específico los cuales serán descritos a continuación:

35

**Ejercicio 1: Dados electrónicos.**

En este ejercicio se genera unos dados pseudo-aleatorios presionando un boton para el dado 1 y un segundo botón para el dado 2 representando estos los dado 1 y dado 2 respectivamente, mostrándonos dichos dados en un arreglo estratégico de leds(colocados en el puerto B del Atmega 169) dándonos la apariencia de dados reales y mostrando cual dado es generado en la LCD del AVR Butterfly.

Materiales usados en este ejercicio AVR Butterfly (Atmega 169, LCD) Leds, botoneras

Resistencias (330 ohm)

**Ejercicio 2: Semáforos Vehículos – peatones.**

En el presente ejercicio se desarrolla una versión completa a escala de un semáforo vehículos – peatones de calle y avenida en el cual si un peatón desea cruzar deberá presionar un botón para calle y avenida respectiva y esperar la orden de cruzar (AVR Butterfly LCD), desde luego que los vehículos de la calle o avenida estén detenidos

(leds rojoson). Luego se hace el conteo regresivo del tiempo estimado para que una

persona cruce la calle (AVR Butterfly LCD), y finalmente restablecer el flujo vehicular

(leds verdeson).

Materiales usados en este ejercicio AVR Butterfly (Atmega 169, LCD) Leds

36

Botoneras.

Resistencias (330 ohm)

**Ejercicio 3: Cerradura electrónica.**

En este ejercicio vamos a desarrollar una cerradura electrónica la cual posee sencillo teclado matricial (pulsadores y resistencias) la cual tiene la opción de ingresar o cambiar clave donde tenemos un botón para seleccionar, y otro botón para aceptar. Para poder abrir la puerta se procederá a ingresar los cuatros números que conforman la clave

(1234), si la clave fue correcta se abrirá la puerta y se presentara un mensaje en el LCD del AVR Butterfly .Y si falla el ingreso de la clave por tres ocasiones esta activara una alarma audible (AVR Butterfly speaker) y la señal de alarma en el LCD. Si se selecciona la opción cambiar clave primero deberemos ingresar la clave antigua.

Materiales usados en este ejercicio.

AVR Butterfly (Atmega 169, LCD, speaker)

Botoneras.

Teclado matricial (resistencias 1K ohm)

**Ejercicio 4: Maquinas Electrónica de bebidas.**

En el presente ejercicio se desarrolla una simulación de una maquina electrónica de bebidas la cual ofrece a la clientela cuatro tipos de bebidas mostradas en la LCD del AVR Butterfly (pepsi 50, profit 1, v220 15, club 2), para observar los tipos de bebidas presionamos el botón para elegir la bebida. Y tenemos un segundo botón para aceptar. Al observar los diferentes tipos de bebidas también podremos apreciar su precio en un display de 7 segmentos catodo común cuyos valores están desde uno a

37

cuatro dólares americanos dependiendo del tipo de bebida. Al elegir la bebida se procede a ingresar las monedas las cuales pueden ser de 25 ctvs, 50 ctvs, 1 usd con el botón de elegir puede seleccionar el dinero , se acepta las monedas con un segundo botón y se visualizara el número de monedas que se ha ingresado en el display de 7 segmentos. Y después de haber ingresado el valor correspondiente en monedas se acepta y se simula una secuencia de segmentos que hace las veces de entrega de la bebida.

Materiales usados en este ejercicio. AVR Butterfly (Atmega 169, LCD) Display 7 segmentos (Cátodo Común ) Botoneras.

**Ejercicio 5: Calculadora Básica.**

En este ejercicio se realiza la simulación de una calculadora binaria para resolver operaciones binarias como la AND, OR, EXOR, SUMA, RESTA, MULTIPLICACION. Para desarrollar esta calculadora haremos uso de un teclado matricial y de la LCD del AVR Butterfly

Materiales usados en este ejercicio

AVR Butterfly (Atmega 169, LCD)

Teclado matricial (resistencias 1K ohm)

38

**Capítulo 4**

**Desarrollo y Simulación del Proyecto**

**4 DESARROLLO DE LOS EJERCICIOS**

En este capitulo se describen los modos de operación de los elementos que conforman los ejercicios y su funcionamiento en conjunto para la aplicación implementada usando la tarjeta de desarrollo AVR BUTTERFLY.

**4.1 DADOS ELECTRÓNICOS.**

En este ejercicio generamos unos dados pseudo-aleatorios presionando los botones izquierda/derecha del joystick del AVR Butterfly representando estos los dado 1 y dado 2 respectivamente, mostrándonos dichos dados en un arreglo estratégico de leds(colocados en el puerto B del Atmega 169) dándonos la apariencia de dados reales y mostrando cual dado es generado en la LCD del AVR Butterfly.

**4.1.1 DIAGRAMA DE BLOQUE DADOS ELECTRÓNICOS**

Para comprender esta aplicación se realizó el diagrama de bloque dados electrónicos como se indica en la figura 4.1, donde vemos que se realiza una comunicación con el AVR BUTTERFLY cual muestra sus salidas conectadas a los LEDs.

39

BOTONERAS

BUTTERFLY

2 7

Figura 4.1 Diagrama de Bloque dados electrónicos

**4.1.2 ALGORITMO DADOS ELECTRÓNICOS.**

Aquí observamos el algoritmo Dados electrónicos del proyecto donde describe el

funcionamiento optimizado presionando un botón

INICIO

INC R18

INC R19

DELAY

INC R18

0 0

PINE,4 PINE, 5

1 1

DADO 1 DADO 2

2 3

Figura 4.2 Diagrama ASM Dados electrónicos.

40

A continuación se muestra las subrutinas para el dado1 donde se muestra el numero del dado 1, esta salida son usando el puerto B de salida del Kit avr butterfly como se muestra en la figura 4.3, dicho numero aleatorio es mostrado en la plataforma interactiva mediante el encendido de leds.

DADO1

2 3

TEMP←R11

PORTB←00000000

TEMP==1

PORTB←00001000 DELAY

TEMP==2

PORTB←00010100

DELAY

TEMP==3

PORTB←01001001

DELAY

TEMP==4

PORTB←01100011

DELAY

TEMP==5

PORTB←01101011

DELAY

Figura 4.3 Diagrama ASM subrutina para dados1

41

A continuación se muestra las subrutinas para el dado2 donde se muestra el numero del dado 2, esta salida son usando el puerto B de salida del Kit avr butterfly como se muestra en la figura 4.4, dicho numero aleatorio es mostrado en la plataforma interactiva mediante el encendido de leds.

DADO2

2 3

TEMP←R10

TEMP==1

PORTB←00001000 DELAY

TEMP==2

PORTB←00010100

DELAY

TEMP==3

PORTB←01001001

DELAY

TEMP==4

PORTB←01100011

DELAY

TEMP==5

PORTB←01101011

DELAY

Figura 4.4 Diagrama ASM subrutina para dados2

42

**4.1.3 PROGRAMACIÓN DADOS ELECTRÓNICOS.**

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* DADOS ELECTRONICOS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; GRUPO # 10

; INTEGRANTES:

; Ronald Rivera

; Nelson Castro

; NOMBRE DEL PROGRAMA:

; Dados Electrónicos.

; DESCRIPCIÓN DEL PROYECTO

; SE GENERAN NUMEROS PSEUDOALEATORIOS Y SON MOSTRADOS EN EL PUERTO B

; CADA VEZ QUE SE PRESIONE UNA BOTONERA DADO1 (PINE4) O DADO2 (PINE5)

; ESTOS VALORES SE REPRESENTARAN EN UN ARREGLO DE LEDS

(DADOS)

; NOMBRE DEL ARCHIVO:

; dados.asm

.include "m169def.inc"

; DECLARACION DE VARIABLES

.def char = r21

.def digit = r22

.def nibble = r23

.def A = r19

.def REG\_TEMP = R20

.def TEMP =R18

;DECLARACION DEL VECTOR DE RESET

.ORG $0000

RJMP reset reset:

/\*CONFIGURACION DE PUNTERO DE PILA PARA QUE LAS

LLAMADAS A SUBRUTINAS PUEDAN TRABAJAR CORRECTAMENTE \*/ LDI A,LOW(RAMEND)

OUT SPL,A

LDI A,HIGH(RAMEND) OUT SPH,A

; DECLARACION DE PUERTOS DE ENTRADA Y SALIDA LDI A,0XFF

OUT DDRB,A

LDI REG\_TEMP,0B00000000

LDI TEMP,0B00000000

OUT PORTE,REG\_TEMP OUT DDRE,TEMP

NOP

IN TEMP,PINE

rcall lcdInit ; INICIALIZA LA LCD

RCALL inicio\_LCD ; MUESTRA MENSAJE DE INICIO "DADOS ELECTRONICOS"

43

RCALL MENSAJE\_DE\_ESPERA ; MUESTRA EL MENSAJE DE ESPERA

"WAIT"

/\*======================================

====PROGRAMA PRINCIPAL CICLO INFINITO===

=======================================\*/ MAIN:

; NUMEROS PSEUDOALEATORIOS DE LOS DADOS

inc r10

inc r11

INC R10

IN TEMP,PINE ; LEEMOS EL PUERTO E PARA VER SI SE HA PRESIONADO UNA BOTON

ANDI TEMP,0B00110000 ; FILTRA LOS DOS UNICOS PINES QUE NECESITAMOS SABER SU ESTADO PINE4, PIE5

CPI TEMP,0X10 ; COMPARA A VER SI SE HA PRESIONADO EL BOTON PB4

BREQ dadomos1\_

CPI TEMP,0X20 ; COMPARA A VER SI SE HA PRESIONADO EL BOTON PB5

BREQ dadomos2\_ RJMP MAIN DADOMOS1\_:

RCALL dadomos1

RJMP MAIN DADOMOS2\_:

RCALL dadomos2

RJMP MAIN

/\*===============================

=====MESAJE INICIAL DEL LCD======

====="DADOS ELECTR...============

===============================\*/

inicio\_LCD:

ldi char,'D' ldi digit, 0 rcall lcd\_out ldi char,'A' inc digit rcall lcd\_out

ldi char,'D' inc digit rcall lcd\_out

ldi char,'O' inc digit rcall lcd\_out

ldi char,'S' inc digit rcall lcd\_out

44

rcall DELAY

ldi char,'E' ldi digit, 0 rcall lcd\_out

ldi char,'L' inc digit rcall lcd\_out

ldi char,'E' inc digit rcall lcd\_out

ldi char,'C'

inc digit rcall lcd\_out ldi char,'T' inc digit rcall lcd\_out ldi char,'R' inc digit rcall lcd\_out

rcall DELAY ldi char,'O' ldi digit, 0 rcall lcd\_out ldi char,'N' inc digit

rcall lcd\_out ldi char,'I' inc digit rcall lcd\_out ldi char,'C' inc digit rcall lcd\_out ldi char,'O' inc digit rcall lcd\_out

45

ldi char,'S' inc digit rcall lcd\_out

rcall DELAY rcall lcd\_clear ldi r18,00 ldi r19,76

RET

/\*===============================

========MESAJE DE ESPERA========

=========="WAIT===============\*/ MENSAJE\_DE\_ESPERA:

ldi char,'W' ldi digit, 0 rcall lcd\_out

ldi char,'A' inc digit rcall lcd\_out

ldi char,'I' inc digit rcall lcd\_out

ldi char,'T'

inc digit

RET

rcall lcd\_out

finlaz:

RCALL DELAY RCALL DELAY rcall lcd\_clear

rjmp MENSAJE\_DE\_ESPERA RJMP MAIN

dadomos1:

rcall lcd\_clear rcall dado1 rjmp finlaz

dadomos2:

rcall lcd\_clear rcall dado2

46

rjmp finlaz

/\*===========================================================

======

====== MUESTRA EL MENSAJE "DADO1" Y COMPARA EL VALOR DEL REGISTRO==

=QUE HACE LAS VECES DE NUMERO ALEATORIO Y LO CARGA EN EL PUERTO B==

=======EN UN ARREGLO DE LEDS QUE HACEN LAS VECES DE DADO===========

=============================================================

=====\*/

dado1:

ldi char,'D' ldi digit, 0 rcall lcd\_out

ldi char,'A' inc digit rcall lcd\_out

ldi char,'D' inc digit rcall lcd\_out

ldi char,'O' inc digit rcall lcd\_out

ldi char,'1' inc digit rcall lcd\_out

mov TEMP,r11 ; GUARDA EL VALOR DEL REGISTRO R11 DADO PSEUDOLEATORIO 1 EN UNA VARIABLE TEMPORAL

andi TEMP,$07 ; FILTRA LOS 3 PRIMEROS NUMEROS PARA COMPARAR LOS NUMEROS DE LOS DADOS DEL 1 - 6

cpi TEMP,$01 ; COMPARA SI ESE NUMERO ES = A 1,2,3,...6 Y SI LO ES SE LO CARGA EN EL ARREGLO DE LEDS

breq uno

cpi TEMP,$02 ; DE TAL MANERA QUE TOME LA FORMA DE DICHO NUMERO EN EL DADO

breq dos

cpi TEMP,$03

breq tres

cpi TEMP,$04

breq cuatro

cpi TEMP,$05

47

uno:

dos:

breq cinco

cpi TEMP,$06

breq seis

ldi char,'1'

ldi A,0b00001000 out PORTB,A rjmp mostrar

ldi char,'2'

ldi A,0b01000001

out PORTB,A

rjmp mostrar

tres:

cuatro:

cinco:

seis:

ldi char,'3'

ldi A,0b01001001 out PORTB,A rjmp mostrar

ldi char,'4'

ldi A,0b01100011 out PORTB,A rjmp mostrar

ldi char,'5'

ldi A,0b01101011 out PORTB,A rjmp mostrar

ldi char,'6'

ldi A,0b01110111 out PORTB,A rjmp mostrar

/\*\*=================================

===TIEMPO QUE SE MUESTRA LOS LEDS====

===ENCENDIDOS EN EL ARREGLO DE LEDS==

====================================\*/

mostrar:

rcall DELAY rcall DELAY ldi A,0x00

out PORTB,A

ret

/\*===========================================================

====== MUESTRA EL MENSAJE "DADO2" Y COMPARA VALOR DEL REGISTRO QUE HACE LAS VECES DE NUMERO ALEATORIO Y LO CARGA EN EL PUERTO B EN UN ARREGLO DE LEDS QUE HACEN LAS VECES DE DADO======================================================\*/

48

dado2:

uno1:

dos1:

ldi char,'D' ldi digit, 0 rcall lcd\_out

ldi char,'A' inc digit rcall lcd\_out

ldi char,'D' inc digit rcall lcd\_out

ldi char,'O' inc digit rcall lcd\_out

ldi char,'2' inc digit rcall lcd\_out

mov TEMP,r10 andi TEMP,$07 cpi TEMP,$01 breq uno1

cpi TEMP,$02

breq dos1

cpi TEMP,$03

breq tres1

cpi TEMP,$04 breq cuatro1 cpi TEMP,$05 breq cinco1 cpi TEMP,$06 breq seis1

ldi char,'1'

ldi A,0b00001000 out PORTB,A rjmp mostrar

ldi char,'2'

ldi A,0b01000001

out PORTB,A

rjmp mostrar

tres1:

ldi char,'3'

ldi A,0b01001001

49

out PORTB,A

rjmp mostrar cuatro1:

ldi char,'4'

ldi A,0b01100011 out PORTB,A rjmp mostrar

cinco1:

seis1:

ldi char,'5'

ldi A,0b01101011 out PORTB,A rjmp mostrar

ldi char,'6'

ldi A,0b01110111 out PORTB,A rjmp mostrar

/\*===============================

===========RETARDOS============

=================================\*/ DELAY:

ldi R25, $06

WGLOOP0: ldi R26, $FF WGLOOP1: ldi R27, $FF WGLOOP2: dec R27

brne WGLOOP2

dec R26

brne WGLOOP1

dec R25

brne WGLOOP0

RET

DELAY1:

ldi R25, $01

WGLOOP01: ldi R26, $0F WGLOOP11: ldi R27, $0F WGLOOP21: dec R27

brne WGLOOP21

dec R26

brne WGLOOP11

dec R25

brne WGLOOP01

RET

DELAY2:

ldi R25, $05

WGLOOP02: ldi R26, $0F WGLOOP12: ldi R27, $1F WGLOOP22: dec R27

brne WGLOOP22

dec R26

brne WGLOOP12

50

dec R25

brne WGLOOP02

RET

/\*======================================== INCLUYENDO LIBRERIA ENCARGADA DE LA LCD

=======================================\*/

.include "lcd\_driver.asm"

**4.1.4 SIMULACIÓN DADOS ELECTRÓNICOS.**

El objetivo de esta simulación es observar generamos dados pseudo-aleatorios presionando un botón (izquierda/derecha del joystick del AVR Butterfly) y mostrándonos dichos dados en un arreglo estratégico de leds (colocados en el puerto B del Atmega169 ) como se muestra en la figura 4.5

Figura 4.5 Circuito de proteus Dados electrónicos.

**4.2 SEMÁFOROS VEHÍCULOS – PEATONES.**

En el presente ejercicio se desarrolla una versión completa a escala de un semáforo vehículos – peatones de calle y avenida en el cual si un peatón desea cruzar deberá presionar un botón para calle y avenida respectiva y esperar la orden de cruzar (AVR

51

Butterfly LCD), desde luego que los vehículos de la calle o avenida estén detenidos

(leds rojoson). Luego se hace el conteo regresivo del tiempo estimado para que una persona cruce la calle (AVR Butterfly LCD), y finalmente restablecer el flujo vehicular (leds verdeson).

**4.2.1 DIAGRAMA DE BLOQUE CIRCUITO SEMÁFOROS**

**VEHÍCULOS - PEATONES.**

Para un mejor entendimiento se realizó el diagrama de bloque como se indica en la figura siguiente, donde vemos que se realiza una comunicación con el AVR BUTTERFLY cual muestra sus salidas conectadas a los LEDs El diagrama correspondiente se muestra en la figura 4.6

BOTONERAS

BUTTERFLY

2

Figura 4.6 Diagrama de Bloque Semáforo.

52

**4.2.2 ALGORITMO SEMÁFOROS VEHÍCULOS - PEATONES.**

En el siguiente diagrama ASM se muestra la lógica utilizada para realizar la programación en el Circuito Semáforo. En la figura 4.7 se muestra su correspondiente diagrama ASM.

Figura 4.7 Diagrama ASM Semáforo**.**

53

En esta figura se muestra el diagrama de flujo grama de la calle1 donde mediante retardos (delay) se muestra el tiempo que tiene el peaton para cruzar dicha calle, asignamos como salida el puerto B en este caso es ña calle 1.

Figura 4.8 Diagrama ASM Semáforo calle 1

54

En esta figura se muestra el diagrama de flujo grama de la calle2 walk donde mediante retardos (delay) se muestra el tiempo que tiene el peaton para cruzar dicha calle, asignamos como salida el puerto B

Figura 4.9 Diagrama ASM Semáforo calle 2

55

**4.2.3 PROGRAMACIÓN SEMÁFOROS VEHÍCULOS -**

**PEATONES.**

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* SEMAFORO \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; GRUPO # 10

; INTEGRANTES:

; Ronald Rivera

; Nelson Castro

; NOMBRE DEL PROGRAMA:

; Semaforo.

;

; DESCRIPCIÓN DEL PROYECTO

; EN ESTE EJERCICIO DESARROLLAMOS LA SECUENCIA DE LUCES DE UN SEMAFORO

; CALLE - AVENIDA (SALIDAS PUERTO B) DONDE TAMBIEN SE DA LA OPCION DE

; UN SEMAFORO PARA PEATONES (LCD BUTTERFLY) EL CUAL SI DESEA CRUZAR LA CALLE

;O AVENIDA DEBERA PRESIONAR UN BOTON (PINE4, PINE5) Y ESPERAR LA ORDEN DE CAMINAR

; NOMBRE DEL ARCHIVO:

; Semaforo.asm

.include "m169def.inc"

; DECLARACION DE VARIABLES

.def char = r21

.def digit = r22

.def nibble = r23

.def A = r16

.def semafo = r20

.def calle1 = r27

.def calle2 = r28

.def digito = r19

;DECLARACION DEL VECTOR DE RESET

.ORG $0000

RJMP reset reset:

/\*CONFIGURACION DE PUNTERO DE PILA PARA QUE LAS

LLAMADAS A SUBRUTINAS PUEDAN TRABAJAR CORRECTAMENTE \*/ LDI A,LOW(RAMEND)

OUT SPL,A

LDI A,HIGH(RAMEND) OUT SPH,A

56

; DECLARACION DE PUERTOS DE ENTRADAS Y SALIDAS LDI A,0XFF

OUT DDRB,A

LDI A,0B00000000

OUT PORTE,A OUT DDRE,A NOP

rcall lcdInit inicio:

RCALL MENSAJE\_INICIAL

rcall DELAY

rcall lcd\_clear

start: ; CALLE 1 SEMAFORO EN VERDE, CALLE 2 SEMAFORO EN ROJO INICIOCALLE1:

rcall STOP1MESS ;MENSAJE EL CUAL INDICA AL PEATON QUE NO DEBE CRUZAR LA CALLE 1 (NO CRUZE LA CALLE)

ldi semafo,0b00001100

out PORTB,semafo ; CARGA EL VALOR DEL ESTADO DE LAS LUCES EN EL PUERTO B

rcall DELAY ; RETARDOS (TIEMPO QUE PERMANECE LA LUZ VERDE Y ROJA ENCENDIDAS)

rcall PREGUNTAR ; EL PEATON PRESIONO EL BOTON PARA PASAR LA CALLE 1 O 2

rcall DELAY

rcall PREGUNTAR

rcall DELAY

rcall PREGUNTAR

rcall DELAY

rcall PREGUNTAR

rcall DELAY

rcall PREGUNTAR

rcall DELAY

rcall PREGUNTAR

rcall DELAY

rcall PREGUNTAR

rcall CONFIRMAR ; CONFIRMA SI HA PRESIONADO EL BOTON PARA CRUZAR LA CALLE 1 O 2

; CALLE 1 SEMAFORO AMARILLO , CALLE 2 SEMAFORO AUN EN ROJO CONTCALLE1:

ldi semafo,0b00001010

out PORTB,semafo ; CARGA EL VALOR DEL ESTADO DE LAS LUCES EN EL PUERTO B

rcall DELAY ; RETARDOS (TIEMPO QUE PERMANECE LA LUZ AMARILLA Y ROJA ENCENDIDAS)

rcall PREGUNTAR ; EL PEATON PRESIONO EL BOTON PARA PASAR LA CALLE 1 O 2

rcall DELAY

rcall PREGUNTAR

rcall DELAY

57

rcall PREGUNTAR

rcall DELAY

rcall PREGUNTAR

rcall CONFIRMAR ; CONFIRMA SI HA PRESIONADO EL BOTON PARA CRUZAR LA CALLE 1 O 2

; CALLE 1 SEMAFORO EN ROJO , CALLE 2 SEMAFORO EN VERDE INICIOCALLE2:

rcall STOP2MESS ;MENSAJE EL CUAL INDICA AL PEATON QUE NO DEBE CRUZAR LA CALLE 2

ldi semafo,0b00100001

out PORTB,semafo ; CARGA EL VALOR DEL ESTADO DE LAS LUCES EN EL PUERTO B

rcall DELAY ; RETARDOS (TIEMPO QUE PERMANECE LA LUZ ROJA Y VERDE ENCENDIDAS)

rcall PREGUNTAR ; EL PEATON PRESIONO EL BOTON PARA PASAR LA CALLE 1 O 2

rcall DELAY

rcall PREGUNTAR

rcall DELAY

rcall PREGUNTAR

rcall DELAY

rcall PREGUNTAR

rcall DELAY

rcall PREGUNTAR

rcall DELAY

rcall PREGUNTAR

rcall DELAY

rcall PREGUNTAR

rcall CONFIRMAR ; CONFIRMA SI HA PRESIONADO EL BOTON PARA CRUZAR LA CALLE 1 O 2

; CALLE 1 SEMAFORO AUN EN ROJO , CALLE 2 SEMAFORO EN AMARILLO CONTCALLE2:

ldi semafo,0b00010001

out PORTB,semafo ; CARGA EL VALOR DEL ESTADO DE LAS LUCES EN EL PUERTO B

rcall DELAY ; RETARDOS (TIEMPO QUE PERMANECE LA LUZ ROJA Y AMARILLA ENCENDIDAS)

rcall PREGUNTAR ; EL PEATON PRESIONO EL BOTON PARA PASAR LA CALLE 1 O 2

rcall DELAY

rcall PREGUNTAR

rcall DELAY

rcall PREGUNTAR

rcall DELAY

rcall PREGUNTAR

rcall CONFIRMAR ; CONFIRMA SI HA PRESIONADO EL BOTON PARA CRUZAR LA CALLE 1 O 2

rjmp start ; SE REPITE DE NUEVO LA SECUENCIA DE LUCES

58

/\*=====================================================

===SI EL PEATON HA PRESIONADO UN BOTON YA SEA ESTE=====

===PARA CRUZAR CALLE 1 O 2 SE INCREMENTA UN CONTADOR===

======================================================\*/ PREGUNTAR:

sbis PINE,4

RJMP SALTAR

inc calle1

SALTAR:

sbis PINE,5

RJMP SALIR

inc calle2

SALIR:

ret

/\*===========================================================

==SI EL CONTADOR RESPECTIVO ES >= 1 QUIERE DECIR QUE SE ======

==DESEA CRUZAR LA CALLE POR Q SE HA PRESIONADO MAS D UNA VEZ EL BOTON RESPECTIVO========================================\*/ CONFIRMAR:

cpi calle1,1 brge calle1walk cpi calle2,1

brge calle2walk

ret calle2walk:

rjmp calle2walkex calle1walk:

rcall WAIT1MESS ; MENSAJE EL CUAL INDICA AL PEATON QUE DESEA CRUZAR LA CALLE 1 QUE ESPERE UNOS SEGUNDOS (WAIT1)

ldi digito,6 ; VALOR QUE SE SETEARA EL CONTEO DE ESPERA EL CUAL SE DECREMENTARA

rcall MOS\_NUM ; MUESTRA EL VALOR (WAIT16)----> OJO AQUI SOLO MUESTRA EL 6 YA QUE EL WAIT1 SE LO HACE DOS LINEAS MAS ARRIBA

sbi PINB,6 ; ENCIENDE UN LED INDICADOR (SE HA PRESIONADO EL BOTON CORRECTAMENTE)

clr calle1 ; BORRA EL CONTADOR DE CALLE 1

rcall DELAY ; RETARDOS TIEMPOS DE ESPERA Y TIEMPO QUE ESTA HABILITADO EL PASO DE PEATONES POR LA CALLE 1

dec digito ; DECREMENTA EN CONTEO DE ESPERA OSEA SALDRA EN EL LCD (WAIT15)

rcall MOS\_NUM ; MUESTRA EL VALOR DEL DECREMENTO

(WAIT15)----> OJO AQUI SOLO MUESTRA EL 5 YA QUE EL WAIT2 SE LO HIZO AL INICIO

cbi PINB,6 ; APAGA EL LED INDICADOR (SE HA PRESIONADO EL BOTON CORRECTAMENTE) ENCERA EL BIT 6 DEL PUERTO B

rcall DELAY

dec digito

rcall MOS\_NUM

sbi PINB,6

59

rcall DELAY

dec digito

rcall MOS\_NUM

ldi semafo,0b00001010 ; SEMAFORO CALLE 1 NARANJA SEMAFORO CALLE 2 ROJO

out PORTB,semafo ; CARGA EL VALOR DE LOS ESTADOS DE LAS LUCES EN EL PUERTO B

rcall DELAY

dec digito

rcall MOS\_NUM

sbi PINB,6 rcall DELAY dec digito

rcall MOS\_NUM cbi PORTB,6 rcall DELAY

dec digito

rcall MOS\_NUM sbi PORTB,6 rcall DELAY

ldi semafo,0b00100001 ; SEMAFORO CALLE 1 EN ROJO , SEMAFORO CALLE 2 EN VERDE

out PORTB,semafo ; CARGA EL VALOR DE LOS ESTADOS DE LAS LUCES EN EL PUERTO B

rcall WALK1MESS ; MENSAJE EL CUAL INDICA AL PEATON QUE DESEA CRUZAR LA CALLE 1 QUE CAMINE (WALK1)

ldi digito,7 ; VALOR QUE SE SETEARA EL TIEMPO PARA CRUZAR LA CALLE EL CUAL SE DECREMENTARA

rcall MOS\_NUM ; MUESTRA EN LA LCD (WALK17) ESTA LINEA DE CODIGO SOLO MUESTRA EL 7 YA Q EL WALK1 SE LO HIZO 2

LINEAS MAS ARRIBA

rcall DELAY ; RETARDO

rcall STOP2MESS ; MENSAJE QUE INDICA DETENERSE A LOS PEATONES QUE DESEAN CRUZAR LA CALLE 2 SALDRA EN EL LCD

(STOP2)

dec digito ; DECREMENTA EL TIEMPO PARA CRUZAR LA CALLE 1

rcall MOS\_NUM ; SE MUESTRA EL DECREMENTO OJO PERO COMO NOS DIMOS CUENTA SALDRA JUNTO A STOP2 ES DECIR (STOP26)

rcall DELAY

rcall WALK1MESS ; APARECE EL MENSAJE DE CRUZAR CALLE

1 Y ASI SE ALTERNARAN CAMINE X LA CALLE 1 Y NO PASE POR LA CALLE 2

dec digito ; ES DECIR WALK1(X), STOP2(X) DONDE X SERA EL CONTEO REGRESIVO DESDE 7 A 0

rcall MOS\_NUM

rcall DELAY

rcall STOP2MESS

dec digito rcall MOS\_NUM

60

rcall DELAY

rcall WALK1MESS

dec digito rcall MOS\_NUM rcall DELAY

rcall STOP2MESS

dec digito rcall MOS\_NUM rcall DELAY

rcall WALK1MESS

dec digito rcall MOS\_NUM rcall DELAY

rcall STOP2MESS

dec digito rcall MOS\_NUM rcall DELAY

rcall lcd\_clear ; LIMPIAMOS LA LCD

rcall STOP2MESS ; MUESTRA SOLO EL MENSAJE DETENERSE PARA LA CALLE 2

rjmp CONTCALLE2 ; REGRESA A SU ESTADO RESPECTIVO CALLE

1 SEMAFORO AMARILLO , CALLE 2 SEMAFORO AUN EN ROJO

calle2walkex:

rcall WAIT2MESS ; MENSAJE EL CUAL INDICA AL PEATON QUE DESEA CRUZAR LA CALLE 2 QUE ESPERE UNOS SEGUNDOS (WAIT2)

ldi digito,6 ; VALOR QUE SE SETEARA EL CONTEO DE ESPERA EL CUAL SE DECREMENTARA

rcall MOS\_NUM ; MUESTRA EL VALOR (WAIT26)----> OJO AQUI SOLO MUESTRA EL 6 YA QUE EL WAIT2 SE LO HACE DOS LINEAS MAS ARRIBA

sbi PINB,6 ; ENCIENDE UN LED INDICADOR (SE HA PRESIONADO EL BOTON CORRECTAMENTE)

clr calle2 ; BORRA EL CONTADOR DE CALLE 2

rcall DELAY ; RETARDOS TIEMPOS DE ESPERA Y TIEMPO QUE ESTA HABILITADO EL PASO DE PEATONES POR LA CALLE 2

dec digito ; DECREMENTA EN CONTEO DE ESPERA OSEA SALDRA EN EL LCD (WAIT25)

rcall MOS\_NUM ; MUESTRA EL VALOR DEL DECREMENTO

(WAIT25)----> OJO AQUI SOLO MUESTRA EL 5 YA QUE EL WAIT2 SE LO HIZO AL INICIO

cbi PINB,6 ; APAGA LED INDICADOR (SE HA PRESIONADO EL BOTON CORRECTAMENTE) ENCERA EL BIT 6 DEL PUERTO B

rcall DELAY

dec digito

rcall MOS\_NUM

sbi PINB,6

rcall delay

61

ldi semafo,0b00010001 ; SEMAFORO CALLE 1 ROJO SEMAFORO CALLE 2 NARANJA

out PORTB,semafo ; CARGA EL VALOR DE LOS ESTADOS DE LAS LUCES EN EL PUERTO B

dec digito

rcall MOS\_NUM

rcall DELAY

dec digito

rcall MOS\_NUM

sbi PINB,6 rcall DELAY dec digito

rcall MOS\_NUM cbi PORTB,6 rcall DELAY

dec digito

rcall MOS\_NUM sbi PORTB,6 rcall DELAY

ldi semafo,0b00001100 ; SEMAFORO CALLE 1 EN VERDE , SEMAFORO CALLE 2 EN ROJO

out PORTB,semafo ; CARGA EL VALOR DE LOS ESTADOS DE LAS LUCES EN EL PUERTO B

rcall WALK2MESS ; MENSAJE EL CUAL INDICA AL PEATON QUE DESEA CRUZAR LA CALLE 2 QUE CAMINE (WALK2)

ldi digito,7 ; VALOR QUE SE SETEARA EL TIEMPO PARA CRUZAR LA CALLE EL CUAL SE DECREMENTARA

rcall MOS\_NUM ; MUESTRA EN LA LCD (WALK27) ESTA LINEA DE CODIGO SOLO MUESTRA EL 7 YA Q EL WALK2 SE LO HIZO 2 LINEAS MAS ARRIBA

rcall DELAY ; RETARDO

rcall STOP1MESS ; MENSAJE QUE INDICA DETENERSE A LOS PEATONES QUE DESEAN CRUZAR LA CALLE 1 SALDRA EN EL LCD

(STOP1)

dec digito ; DECREMENTA EL TIEMPO PARA CRUZAR LA CALLE 2

rcall MOS\_NUM ; SE MUESTRA EL DECREMENTO OJO PERO COMO NOS DIMOS CUENTA SALDRA JUNTO A STOP1 ES DECIR (STOP16)

rcall DELAY

rcall WALK2MESS ; APARECE EL MENSAJE DE CRUZAR CALLE

2 Y ASI SE ALTERNARAN CAMINE X LA CALLE 2 Y NO PASE POR LA CALLE 1

dec digito ; ES DECIR WALK2(X), STOP1(X) DONDE X SERA EL CONTEO REGRESIVO DESDE 7 A 0

rcall MOS\_NUM

rcall DELAY

rcall STOP1MESS

dec digito rcall MOS\_NUM rcall DELAY

62

rcall WALK2MESS

dec digito rcall MOS\_NUM rcall DELAY

rcall STOP1MESS

dec digito rcall MOS\_NUM rcall DELAY

rcall WALK2MESS

dec digito rcall MOS\_NUM rcall DELAY

rcall STOP1MESS

dec digito rcall MOS\_NUM rcall DELAY

rcall lcd\_clear ; LIMPIAMOS LA LCD

rcall STOP1MESS ; MUESTRA SOLO EL MENSAJE DETENERSE PARA LA CALLE 1

rjmp CONTCALLE1 ; REGRESA A SU ESTADO RESPECTIVO CALLE

1 SEMAFORO AUN EN ROJO , CALLE 2 SEMAFORO EN AMARILLO

/\*=====================================================

=MUESTRA EL NUMERO RESPECTIVO DECREMNTANDOSE EL LA LCD=

=====EN LA ULTIMA POSICION DE IZQUIERDA A DERECHA======

======================================================\*/ MOS\_NUM:

RET

ldi r16,0x30 add digito,r16 mov char,digito ldi digit,7

rcall lcd\_out ldi r16,0x30 sub digito,r16

/\*====================================

=====MENSAJES MOSTRADOS EN LA LCD=====

=====================================\*/ MENSAJE\_INICIAL:

ldi char,'S' ldi digit, 0 rcall lcd\_out

ldi char,'E' inc digit rcall lcd\_out

63

ldi char,'M' inc digit rcall lcd\_out

ldi char,'A' inc digit rcall lcd\_out

ldi char,'F' inc digit rcall lcd\_out

rcall DELAY rcall lcd\_clear ldi char,'O'

ldi digit, 0

rcall lcd\_out ldi char,'R' inc digit rcall lcd\_out

RET

ldi char,'O' inc digit rcall lcd\_out

WAIT1MESS:

ldi char,'W' ldi digit, 0 rcall lcd\_out

ldi char,'A' inc digit rcall lcd\_out

ldi char,'I' inc digit rcall lcd\_out

ldi char,'T' inc digit rcall lcd\_out

RET

ldi char,'1' inc digit rcall lcd\_out

64

WAIT2MESS:

ldi char,'W' ldi digit, 0 rcall lcd\_out

ldi char,'A' inc digit rcall lcd\_out

ldi char,'I' inc digit rcall lcd\_out

ldi char,'T' inc digit rcall lcd\_out

RET

ldi char,'2' inc digit rcall lcd\_out

WALK1MESS:

ldi char,'W' ldi digit, 0 rcall lcd\_out

ldi char,'A' inc digit rcall lcd\_out

ldi char,'L' inc digit rcall lcd\_out

ldi char,'K' inc digit rcall lcd\_out

RET

ldi char,'1' inc digit rcall lcd\_out

WALK2MESS:

ldi char,'W' ldi digit, 0 rcall lcd\_out

ldi char,'A'

65

inc digit rcall lcd\_out

ldi char,'L' inc digit rcall lcd\_out

ldi char,'K' inc digit rcall lcd\_out

RET

ldi char,'2' inc digit rcall lcd\_out

STOP1MESS:

ldi char,'S' ldi digit, 0 rcall lcd\_out

ldi char,'T' inc digit rcall lcd\_out

ldi char,'O' inc digit rcall lcd\_out

ldi char,'P' inc digit rcall lcd\_out

ldi char,'1' inc digit rcall lcd\_out

RET

STOP2MESS:

ldi char,'S' ldi digit, 0 rcall lcd\_out

ldi char,'T' inc digit rcall lcd\_out

ldi char,'O'

inc digit

66

rcall lcd\_out ldi char,'P' inc digit rcall lcd\_out

RET

ldi char,'2' inc digit rcall lcd\_out

/\*===========RETARDOS==============\*/ DELAY:

ldi R25, $06

WGLOOP0: ldi R26, $FF WGLOOP1: ldi R27, $FF WGLOOP2: dec R27

brne WGLOOP2

dec R26

brne WGLOOP1

dec R25

brne WGLOOP0

RET DELAY1:

ldi R25, $01

WGLOOP01: ldi R26, $0F WGLOOP11: ldi R27, $0F WGLOOP21: dec R27

brne WGLOOP21

dec R26

brne WGLOOP11

dec R25

brne WGLOOP01

RET DELAY2:

ldi R25, $05

WGLOOP02: ldi R26, $0F WGLOOP12: ldi R27, $1F WGLOOP22: dec R27

brne WGLOOP22

dec R26

brne WGLOOP12

dec R25

brne WGLOOP02

RET

/\*====================================== INCLUYENDO LIBRERIA ENCARGADA DE LA LCD

=======================================\*/

.include "lcd\_driver.asm"

67

**4.2.4 SIMULACIÓN DEL SEMÁFOROS VEHÍCULOS-**

**PEATONES.**

En este ejercicio se simula en la herramienta Proteus a escala de un semáforo vehículos- peatones el cual si un peatón desea cruzar la calle deberá presionar un botón para el semáforo1 y un segundo botón para el semáforo 2 y esperar la orden de cruzar (AVR Butterfly LCD) como se muestra en la figura 4.10

Figura 4.10 Circuito de proteus Semáforo

**4.3 CERRADURA ELECTRÓNICA.**

En este ejercicio se desarrolla una cerradura electrónica la cual posee sencillo teclado matricial (pulsadores y resistencias) la cual tiene la opción de ingresar o cambiar clave donde tenemos un botón para seleccionar, y otro botón para aceptar. Para poder abrir la puerta se procederá a ingresar los cuatros números que conforman la clave

(1234), si la clave fue correcta se abrirá la puerta y se presentara un mensaje en el

LCD del AVR Butterfly .Y si falla el ingreso de la clave por tres ocasiones esta

68

activara una alarma audible (AVR Butterfly speaker) y la señal de alarma en el LCD.

Si se selecciona la opción cambiar clave primero deberemos ingresar la clave antigua.

**4.3.1 DIAGRAMA DE BLOQUE CERRADURA**

**ELECTRONICA.**

Se realizó el diagrama de bloque Cerradura electrónica como se indica en la figura

4.11, donde vemos que se realiza una comunicación con el AVR BUTTERFLY y un

teclado matricial (pulsadores y resistencias) la cual tiene la opción de cambiar la clave

(botón derecho del joystick del AVR Butterfly).

BOTON

2

TECLADO

BUTTERFLY LCD

10

Figura 4.11 Diagrama de Bloque Cerradura electrónica.

**4.3.2 ALGORITMO DE CERRADURA ELECTRONICA.**

Aquí observamos el algoritmo cerradura electrónica del proyecto donde describe el funcionamiento optimizado presionando un botón, con el uso de un teclado para poder digitar la clave de la cerradura.

69

INICIO

CAJA FUERTE 2

0 0

PINE,4 PINE.5

1

OPCION←1

1

0

OPCION==1 OPCION==2

OPCION==1 INGRESAR CLAVE

INGRESAR CLAVE

CAMBIAR CLAVE

2

OPCION==2 CAMBIAR CLAVE

2

3

TECLA

TECLA

TECLA

TECLA

A==0XFF

1

A==0XFF

1

A==0XFF

1

A==0XFF

00 0 0

DIGITO>>10

1

DIGITO>>10

1

DIGITO>>10

1

DIGITO>>10

00 0 0

0 0 0

DIGITO>=1

DIGITO>=1

DIGITO>=1

DIGITO>=1

11 1 1

DIGITO1←DIGITO

DIGITO2←DIGITO

DIGITO3←DIGITO

DIGITO4←DIGITO

4

Figura 4.12 Diagrama ASM Cerradura electrónica.

70

A continuación se muestra la continuación del diagrama asm cerradura electrónica donde se puede apreciar que en cada bloque valida cada clave ingresada, si el ingreso es fallido por quinta vez se activa una alarma.

4

DIGITO1==CLA1

FALLO

DIGITO2==CLA2

FALLA++

DIGITO3==CLA3

0

FALLA==5 3

1

DIGITO4==CLA4

ALARMA

ACEPTAR

0

PINE,2

1 1

2

PUERTA ABIERTA

PINE,2

1

2

Figura 4.13 (continuación) Diagrama ASM Cerradura electrónica

71

A continuación se muestra la una subrutina para poder cambiar la clave de la cerradura electrónica dicha clave es ingresada mediante un teclado donde se valida cada digito ingresado si es mayor a cero o si el digito es mayor igual a uno.

PROCESO CAMBIAR CLAVE

1

TECLA

E S T E

P A==0XFF

R O C E S O

DIGITO>>0

S E

R E P I

T DIGITO >=1

E

4

V E

C CLA1←DIGITO

E S

**.**

**.**

**.**

CLA4←DIGITO

MOSTRAR CAMBIO ACEPTAR

2

Figura 4.14 Diagrama ASM cambio de clave.

72

**4.3.3 PROGRAMACIÓN CERRADURA ELECTRÓNICA.**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// MICROCONTROLADORES AVANZADOS

//

// GRUPO: # 7

//

// INTEGRANTES: RONALD RIVERA - NELSON CASTRO

//

// NOMBRE DEL PROYECTO: CERRADURA ELECTRONICA.

//

// DESCRIPCION: CONSISTE EN UNA CERRADURA ELECTRONICA

// LA CUAL POSEE UN SENCILLO TECLADO

// MATRICIAL, LA CUAL TIENE LA OPCION DE

// INGRESARR O CAMBIAR LA CLAVE.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.include "m169def.inc"

//============DECLARACION DE VARIABLES======================

.def char = r21

.def digit = r22

.def nibble = r23

.def A = r16

.def digito = r19

.def opcion = r24

.def digito1= r10

.def digito2= r11

.def digito3= r12

.def digito4= r13

.def fallas = r28

//===========DECLARACION DEL VECTOR DE RESET===============

.ORG $0000

RJMP reset reset:

.ESEG

eeconst:.DB 0xff

.CSEG

//====CONFIGURACION DE PUNTERO DE PILA PARA QUE LAS==========

//====LLAMADAS A SUBRUTINAS PUEDAN TRABAJAR===============

//====CORRECTAMENTE======================================= \*/ LDI A,LOW(RAMEND)

OUT SPL,A

LDI A,HIGH(RAMEND) OUT SPH,A

73

//====DECLARACION DE PUERTOS DE ENTRADA Y SALIDA============

ldi A,0x00

OUT DDRE,A

ldi A,0x00

OUT PORTE,A

ldi A,0xF0

OUT DDRB,A ldi A,0x0F OUT PORTB,A

//===EVALUACION SI EXISTE CLAVE PREVIAMENTE=================

//===INGRESADA==============================================

//===DIRECCION 0: VALE 0 SI NO SE HA INGRESADO CLAVE

//===ANTERIOR(CLAVE DE FABRICA), VALE 1 SI=====================

//===SI TIENE NUEVA CLAVE=====================================

//===DIRECCION 1..4: CLAVE DE 4 DIGITOS==========================

LDI r20,0

OUT EEARL,r20 rcall EEPROMRD cpi r16,0xff

breq CARGA\_CLAVE

rjmp cont

//==========PROCESO PARA PODER CARGAR LA CLAVE=============

//============================================================ CARGA\_CLAVE:

ldi r20,1

out EEARL,r20

ldi r20,5

out EEDR,r20

rcall EEPROMWR

ldi r20,2

out EEARL,r20

ldi r20,5

out EEDR,r20

rcall EEPROMWR

ldi r20,3

out EEARL,r20

ldi r20,5

out EEDR,r20

rcall EEPROMWR

ldi r20,4

out EEARL,r20

ldi r20,5

out EEDR,r20

rcall EEPROMWR

74

//============================================================

//=======RUTINA PRINCIPAL=====================================

cont:

rcall lcdInit ldi fallas,0

//============================================================

//===========MESAJE INICIAL DEL LCD===========================

//===========CERRADURA ELECTRONICA=========================

inicio:

ldi char,'C' ldi digit, 0 rcall lcd\_out ldi char,'E' inc digit rcall lcd\_out ldi char,'R' inc digit rcall lcd\_out ldi char,'R' inc digit rcall lcd\_out ldi char,'A' inc digit rcall lcd\_out

rcall DELAY rcall lcd\_clear ldi char,'D'

ldi digit, 0 rcall lcd\_out ldi char,'U' inc digit rcall lcd\_out ldi char,'R' inc digit rcall lcd\_out ldi char,'A' inc digit rcall lcd\_out

rcall DELAY

ldi char,'E' ldi digit, 0 rcall lcd\_out ldi char,'L' inc digit rcall lcd\_out ldi char,'E' inc digit rcall lcd\_out

75

va:

ldi char,'C' inc digit rcall lcd\_out ldi char,'T' inc digit rcall lcd\_out ldi char,'R' inc digit rcall lcd\_out

rcall DELAY rcall lcd\_clear ldi char,'O'

ldi digit, 0 rcall lcd\_out ldi char,'N' inc digit rcall lcd\_out ldi char,'I' inc digit rcall lcd\_out ldi char,'C' inc digit rcall lcd\_out ldi char,'A' inc digit rcall lcd\_out

rcall DELAY

rcall lcd\_clear

rcall ING\_CLAVE //INGRESAR CLAVE

ldi opcion,1 //CARGAR OPCION 1

comienza:

clr A

rcall DELAY //LLAMADO A RETARDO

in A,PINE

ANDI A,0B00110000 //FILTRO

cpi A,0x10 breq presre cpi A,0x20

breq ejecutar //EJECUTA EL PROCESO DE INGRESO DE CLAVE

rjmp comienza presre:

dec opcion //COMPARACION DE LA OPCION INGRESADA

cpi opcion,0 breq camcla brne ingcla presfo:

inc opcion cpi opcion,3 breq ingcla

76

brne camcla ingcla:

ldi opcion,1

rcall ING\_CLAVE rjmp comienza camcla:

ldi opcion,2

rcall CAM\_CLAVE rjmp comienza ejecutar:

cpi opcion,1 breq opcion1 brne opcion2 opcion1:

rjmp ingclaop opcion2:

rjmp camclaop ingclaop:

rcall lcd\_clear ldi digit,0

ldi digito,0xFF

dig1:

rcall TECLA

cpi digito,0

brge filtroingreso1 rjmp dig1 filtroingreso1:

cpi digito,10 brlo pasofiltro1 rjmp dig1 pasofiltro1:

mov digito1,digito rcall mos\_num

rcall DELAY

inc digit

ldi digito,0xFF

dig2:

rcall TECLA

cpi digito,0

brge filtroingreso2 rjmp dig2 filtroingreso2:

cpi digito,10 brlo pasofiltro2 rjmp dig2 pasofiltro2:

mov digito2,digito rcall mos\_num

rcall DELAY

inc digit

77

ldi digito,0xFF

dig3:

rcall TECLA

cpi digito,0

brge filtroingreso3 rjmp dig3 filtroingreso3:

cpi digito,10 brlo pasofiltro3 rjmp dig3 pasofiltro3:

mov digito3,digito rcall mos\_num

rcall DELAY

inc digit

ldi digito,0xFF

dig4:

rcall TECLA

cpi digito,0

brge filtroingreso4 rjmp dig4 filtroingreso4:

cpi digito,10 brlo pasofiltro4 rjmp dig4 pasofiltro4:

mov digito4,digito rcall mos\_num

rcall DELAY

inc digit

ldi digito,0xFF

evaluar:

//============================================================

//==EVALUANDO EL PRIMER DIGITO INGRESADO CON EL PRIMER======

//==DIGITO GRABADO===========================================

//============================================================ LDI r20,1

OUT EEARL,r20 rcall EEPROMRD ldi A,5

cp digito1,A

brne fallo

//============================================================

//=======EVALUANDO EL SEGUNDO DIGITO INGRESADO CON EL======

//=======SEGUNDO DIGITO GRABADO============================

//============================================================ LDI r20,2

OUT EEARL,r20 rcall EEPROMRD ldi A,5

78

cp digito2,A

brne fallo

//============================================================

//====EVALUANDO EL SEGUNDO DIGITO INGRESADO CON EL SEGUNDO

//====DIGITO GRABADO========================================

//============================================================ LDI r20,3

OUT EEARL,r20 rcall EEPROMRD ldi A,5

cp digito3,A

brne fallo

//============================================================

//=====EVALUANDO EL SEGUNDO DIGITO INGRESADO CON EL========

//=====SEGUNDO DIGITO GRABADO==============================

//============================================================ LDI r20,4

OUT EEARL,r20 rcall EEPROMRD ldi A,5

cp digito4,A

brne fallo

//============================================================

//==CONTINUA SI LOS 4 DIGITOS INGRESADOS FUERON CORRECTOS===

//============================================================

rcall ACEPTO

ingresobien:

in A,PINE

ANDI A,0B00110000

cpi A,0x10

breq va1\_

rjmp ingresobien va1\_:

rjmp va fallo:

inc fallas cpi fallas,3

breq ALARM brne FALLA ALARM:

rcall ALARMA

ciclo:

cbi PORTB,5 rcall DELAY1 sbi PORTB,5 in A,PINE

ANDI A,0B00110000

cpi A,0x10

breq va\_

79

rjmp ciclo va\_:

rjmp va

FALLA:

rcall FALLADO rjmp ingclaop camclaop:

rcall lcd\_clear ldi digit,0

ldi digito,0xFF

dig12:

rcall TECLA

cpi digito,0

brge filtroingreso12 rjmp dig12 filtroingreso12:

cpi digito,10 brlo pasofiltro12 rjmp dig12 pasofiltro12:

mov digito1,digito rcall mos\_num

rcall DELAY

inc digit

ldi digito,0xFF

dig22:

rcall TECLA

cpi digito,0

brge filtroingreso22 rjmp dig22 filtroingreso22:

cpi digito,10 brlo pasofiltro22 rjmp dig22 pasofiltro22:

mov digito2,digito rcall mos\_num

rcall DELAY

inc digit

ldi digito,0xFF

dig32:

rcall TECLA

cpi digito,0

brge filtroingreso32 rjmp dig32 filtroingreso32:

cpi digito,10 brlo pasofiltro32 rjmp dig32 pasofiltro32:

80

mov digito3,digito rcall mos\_num

rcall DELAY

inc digit

ldi digito,0xFF

dig42:

rcall TECLA

cpi digito,0

brge filtroingreso42 rjmp dig42 filtroingreso42:

cpi digito,10 brlo pasofiltro42 rjmp dig42 pasofiltro42:

mov digito4,digito rcall mos\_num

rcall DELAY

inc digit

ldi digito,0xFF

evaluar2:

//============================================================

//===EVALUANDO EL PRIMER DIGITO INGRESADO CON EL PRIMER====

//===DIGITO GRABADO=========================================

//============================================================ LDI r20,1

OUT EEARL,r20 rcall EEPROMRD ldi A,5

cp digito1,A

brne fallo1

//============================================================

//=========EVALUANDO EL SEGUNDO DIGITO INGRESADO CON EL====

//=========SEGUNDO DIGITO GRABADO==========================

//============================================================ LDI r20,2

OUT EEARL,r20

rcall EEPROMRD

ldi A,5

cp digito2,A

brne fallo1

//============================================================

//======EVALUANDO EL SEGUNDO DIGITO INGRESADO CON EL=======

//======SEGUNDO DIGITO=======================================

//======GRABADO=============================================

//============================================================ LDI r20,3

OUT EEARL,r20

rcall EEPROMRD

81

ldi A,5

cp digito3,A

brne fallo1

//============================================================

//=========EVALUANDO EL SEGUNDO DIGITO INGRESADO CON EL====

//=========SEGUNDO DIGITO GRABADO==========================

//============================================================ LDI r20,4

OUT EEARL,r20 rcall EEPROMRD ldi A,5

cp digito4,A

brne fallo1

rjmp CAMBIANDO

fallo1:

rjmp fallo

//============================================================

//=========CONTINUA SI LOS 4 DIGITOS INGRESADOS FUERON=======

//=========CORRECTOS========================================

//============================================================ CAMBIANDO:

rcall CAM\_CLAVE

rcall lcd\_clear ldi digit,0

ldi digito,0xFF

dig13:

rcall TECLA

cpi digito,0

brge filtroingreso13 rjmp dig13 filtroingreso13:

cpi digito,10 brlo pasofiltro13 rjmp dig13 pasofiltro13:

mov digito1,digito rcall mos\_num

rcall DELAY

inc digit

ldi digito,0xFF

dig23:

rcall TECLA

cpi digito,0

brge filtroingreso23 rjmp dig23 filtroingreso23:

cpi digito,10 brlo pasofiltro23 rjmp dig23

82

pasofiltro23:

mov digito2,digito rcall mos\_num

rcall DELAY

inc digit

ldi digito,0xFF

dig33:

rcall TECLA

cpi digito,0

brge filtroingreso33 rjmp dig33 filtroingreso33:

cpi digito,10 brlo pasofiltro33 rjmp dig32 pasofiltro33:

mov digito3,digito rcall mos\_num

rcall DELAY

inc digit

ldi digito,0xFF

dig43:

rcall TECLA

cpi digito,0

brge filtroingreso43 rjmp dig43 filtroingreso43:

cpi digito,10 brlo pasofiltro43 rjmp dig43 pasofiltro43:

mov digito4,digito rcall mos\_num

rcall DELAY

inc digit

ldi digito,0xFF

guardar:

//============================================================

//=====EVALUANDO EL PRIMER DIGITO INGRESADO CON EL==========

//=====PRIMER DIGITO GRABADO================================

//============================================================

ldi r20,1

out EEARL,r20

out EEDR,digito1

rcall EEPROMWR

//============================================================

//=====EVALUANDO EL SEGUNDO DIGITO INGRESADO CON EL========

//=====SEGUNDO DIGITO GRABADO==============================

//============================================================

ldi r20,2

83

out EEARL,r20

out EEDR,digito2

rcall EEPROMWR

//============================================================

//=========EVALUANDO EL SEGUNDO DIGITO INGRESADO CON EL==== SEGUNDO DIGITO GRABADO====================================

//============================================================

ldi r20,3

out EEARL,r20

out EEDR,digito3

rcall EEPROMWR

//============================================================

//=====EVALUANDO EL SEGUNDO DIGITO INGRESADO CON EL========

//=====SEGUNDO DIGITO GRABADO==============================

//============================================================

ldi r20,4

out EEARL,r20

out EEDR,digito4 rcall EEPROMWR rjmp va

//============================================================

//=========MENSAJE EN LA LCD DE INGRESO DE CLAVE=============

//============================================================ ING\_CLAVE:

ldi char,'I'

ldi digit, 0 rcall lcd\_out ldi char,'N' inc digit rcall lcd\_out ldi char,'G' inc digit rcall lcd\_out ldi char,'C' inc digit rcall lcd\_out ldi char,'L' inc digit rcall lcd\_out ldi char,'A' inc digit rcall lcd\_out

rcall DELAY RET

//============================================================

//=========MENSAJE EN LA LCD DE CAMBIO DE CLAVE=============

//============================================================ CAM\_CLAVE:

ldi char,'C'

84

ldi digit, 0 rcall lcd\_out ldi char,'A' inc digit rcall lcd\_out ldi char,'M' inc digit rcall lcd\_out ldi char,'C' inc digit rcall lcd\_out ldi char,'L' inc digit rcall lcd\_out ldi char,'A' inc digit rcall lcd\_out

rcall DELAY RET

MOS\_NUM:

ldi r16,0x30 add digito,r16 mov char,digito rcall lcd\_out

ldi r16,0x30 sub digito,r16 rcall DELAY RET

//============================================================

//=========MENSAJE EN LA LCD CEROPE=========================

============================================================ ACEPTO:

ldi char,'C' ldi digit, 0 rcall lcd\_out ldi char,'E' inc digit rcall lcd\_out ldi char,'R' inc digit rcall lcd\_out ldi char,'O' inc digit rcall lcd\_out ldi char,'P' inc digit rcall lcd\_out ldi char,'E' inc digit rcall lcd\_out

85

rcall DELAY RET

//============================================================

//=========MENSAJE EN LA LCD DE FALLA========================

//============================================================ FALLADO:

ldi char,'F' ldi digit, 0 rcall lcd\_out ldi char,'A' inc digit rcall lcd\_out ldi char,'L' inc digit rcall lcd\_out ldi char,'L' inc digit rcall lcd\_out ldi char,'O' inc digit rcall lcd\_out

mov char,fallas ldi A,0x30

add char,A inc digit rcall lcd\_out

rcall DELAY RET

//============================================================

//=========MENSAJE EN LA LCD DE ALARMA======================

//============================================================ ALARMA:

ldi char,'A' ldi digit, 0 rcall lcd\_out

ldi char,'L' inc digit rcall lcd\_out ldi char,'A' inc digit rcall lcd\_out ldi char,'R' inc digit rcall lcd\_out ldi char,'M' inc digit rcall lcd\_out ldi char,'A' inc digit

86

rcall lcd\_out rcall DELAY RET

//============================================================

//=========MENSAJE EN LA LCD DE NUEVA CLAVE=================

//============================================================ NEWCLAVE:

ldi char,'N' ldi digit, 0 rcall lcd\_out ldi char,'U' inc digit rcall lcd\_out ldi char,'V' inc digit rcall lcd\_out ldi char,'C' inc digit rcall lcd\_out ldi char,'L' inc digit rcall lcd\_out ldi char,'A' inc digit rcall lcd\_out

rcall DELAY RET

//============================================================

//=========RETARDO===========================================

============================================================= DELAY:

ldi R25, $06

WGLOOP0: ldi R26, $FF WGLOOP1: ldi R27, $FF WGLOOP2: dec R27

brne WGLOOP2

dec R26

brne WGLOOP1

dec R25

brne WGLOOP0

RET

//============================================================

//====RETARDO 1==============================================

DELAY1:

ldi R25, $01

WGLOOP01: ldi R26, $0F WGLOOP11: ldi R27, $0F WGLOOP21: dec R27

87

brne WGLOOP21

dec R26

brne WGLOOP11

dec R25

brne WGLOOP01

RET

//===========================================================

//=========RETARDO 2=========================================

DELAY2:

ldi R25, $05

WGLOOP02: ldi R26, $0F WGLOOP12: ldi R27, $1F WGLOOP22: dec R27

brne WGLOOP22

dec R26

brne WGLOOP12

dec R25

brne WGLOOP02

RET

EEPROMWR:

sbic EECR,EEWE

rjmp EEPROMWR

in r16, SREG ; ALAMACENAMIENTO DEL VALOR EN SREG

cli ; DESABILITAR INTERRUPCIONES DURANTE EL TIEMPO DE SECUENCIA

sbi EECR, EEMWE ; INICIO DE LA ESCRITURA EN LA EEPROM

sbi EECR, EEWE

out SREG, r16

RET

EEPROMRD:

sbic EECR,EEWE

rjmp EEPROMRD

in r16, SREG ; ALAMACENAR EL VALOR DE SREG

cli

sbi EECR,EERE

; LECTURA DE LOS DATOS DESDE EL REGISTRO

in r16,EEDR

out SREG,r16

RET

/\*===========================================================

//=====INCLUYENDO LIBRERIA ENCARGADA DE LA LCD============\*/

.include "lcd\_driver.asm"

88

**4.3.4 SIMULACION EN PROTEUS CERRADURA**

**ELECTRONICA.**

Simulación de una cerradura electrónica la cual posee sencillo teclado matricial

(pulsadores y resistencias) la cual tiene la opción de cambiar la clave. Para poder abrir la puerta se procederá a ingresar los cuatros números que conforman la clave seguido del botón aceptar como se muestra en la figura 4.15

Figura 4.15 Circuito de proteus Cerradura electrónica.

89

**4.4 MAQUINAS ELECTRONICA DE BEBIDAS.**

En el presente ejercicio se desarrolla una simulación de una maquina electrónica de bebidas la cual ofrece a la clientela cuatro tipos de bebidas mostradas en la LCD del AVR Butterfly (pepsi, profit, v220, club), para observar los tipos de bebidas presionamos el botón para elegir la bebida. Y tenemos un segundo botón para aceptar. Al observar los diferentes tipos de bebidas también podremos apreciar su precio en un display de 7 segmentos catodo común cuyos valores están desde uno a cuatro dólares americanos dependiendo del tipo de bebida. Al elegir la bebida se procede a ingresar las monedas las cuales pueden ser de 25 ctvs, 50 ctvs, 1 usd con el botón de elegir puede seleccionar el dinero , se acepta las monedas con un segundo botón y se visualizara el número de monedas que se ha ingresado en el display de 7 segmentos. Y después de haber ingresado el valor correspondiente en monedas se acepta y se simula una secuencia de segmentos que hace las veces de entrega de la bebida.

**4.4.1 DIAGRAMA DE BLOQUES MAQUINAS ELECTRONICA**

**DE BEBIDAS.**

Se realizó el diagrama de bloque maquina electrónica de bebidas como se indica en la figura 4.16, donde se observa que se realiza una comunicación con el AVR BUTTERFLY que ofrece a la clientela cuatro tipos de bebidas mostradas en la LCD del AVR Butterfly (cokeco, vino, vodka, whisky).

BOTON

2

BUTTERFLY

LCD

Figura 4.16 Diagrama de Bloque Maquina electrónica de bebidas

90

**4.4.2 ALGORITMO MAQUINA ELECTRONICA DE BEBIDAS.**

Aquí se observa el algoritmo maquina electrónica de bebidas del proyecto donde describe el funcionamiento optimizado donde se presiona botones para dar inicio y elegir el tipo de producto que ofrece la licorera y su valor.

INICIO

7

MENSAJE DE 4

LICORERA ELECTRONICA

1

PINE,4 0

OPCION ++

0

PINE,5

5

1

OPCION 1 0 3

MENSAJE COCA COLA

0

OPCION 2

1

4

MENSAJE

VINO OPCION 3

0

1

4 MENSAJE WHISKI

OPCION 4

1

0

0

OPCION 5

4

MENSAJE VODKA

5

4

4

Figura 4.17 Diagrama ASM Maquina electrónica de bebidas.

91

A continuación se muestra una subrutina donde se valida si el cliente elige la cantidad de dinero y donde valida la respectiva bebida elegida anteriormente por el cliente, la selección del dinero con la bebida se muestra mediante la rotación de los segment os de un display de 7 segmentos al finalizar retorna y le ofrece al usuario la oportunidad de volver a elegir la bebida con el respectivo dinero elegido.

8

VALOR

3 DINERO

1

0

OPCION 1

1

COCA COLA

OPCION 2

0

1

1 VINO OPCION 3 0

1

OPCION 4

WISKY

VODKA

ROTAR EN DISPLAY

7

Figura 4.18 (continuación) Diagrama ASM Maquina electrónica de bebidas.

92

A continuación se muestra la subrutina de elección de dinero la cual se la hace mediante una botonera la cual tiene asignado un PIN,4 del puerto E de acuerdo a la elección del usuario se acepta mediante la segunda botonera conectada al PIN,5 del puerto E de nuestro Kit avr butterfly.

3

INGRESAR DINERO

1

PINE,4 0

1

SELECCIÓN

1 =1

0

PINE,5

MENSAJE

25C

8

SELECCIÓN

=2

0

1

6

MENSAJE

50C

SELECCIÓN

=3

0

1

MENSAJE

SELECCIÓN

D11 =4

6

1

6 MENU

SELECCIÓN

=5 6

ACEPTAR

6

Figura 4.19 Diagrama ASM pedir dinero maquina electrónica de bebidas.

93

A continuación se muestra un el diagrama asm para rotar los segmentos de un display de 7 segmentos donde asignamos como salida el puerto B de nuestro Kit de desarrollo avr butterfly

ROTAR EN DISPLAY

INICIO

PORTB ←00000001

CF ←1

RLC ← A

PORTB ← A

0

A==255

1

FIN

Figura 4.20 Diagrama ASM rotar segmentos en display

94

**4.4.3 PROGRAMACION MAQUINA ELECTRONICA DE**

**BEBIDAS.**

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MAQUINA DE BEBIDAS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; GRUPO # 10

; INTEGRANTES:

; Ronald Rivera

; Nelson Castro

; NOMBRE DEL PROGRAMA:

; BEBIDAS.

; DESCRIPCIÓN DEL PROYECTO

;EN ESTE PROYECTO SIMULAREMOS UNA MAQUINA BENEFICA VENDEDORA DE BEBIDAS

;YA QUE NO ENTREGA VUELTO DEBIDO QUE ES DISEÑADA PARA RECAUDAR FONDOS

;SI LA PERSONA INGRESA UNA CANTIDAD MAYOR AL PRECIO DE LA BEBIDA ESCOJIDA

;LA MAQUINA LE PRESENTARA UN MENSAJE DE AGRADECIMIENTO CASO CONTRARIO NO MOSTRARA NADA

; ESTA MAQUINA SOLO ADMITE MONEDAS DE 25 CTVS, 50 CTVS Y MONEDAS DE DOLLAR SIMULARA LA

; ENTREGA ROTANDO UN DISPLAY DE 7 SEGMENTOS

; NOMBRE DEL ARCHIVO:

; bebidas.asm

.include "m169def.inc"

; DECLARACION DE VARIABLES

.def char = r21

.def digit = r22

.def nibble = r23

.def A = r16

.def dineroing= r10

.def seleccion= r11

.def opcion = r28

.def mon25 = r13

.def mon50 = r14

.def dolar = r15

.def a1 = r29

;DECLARACION DEL VECTOR DE RESET

.ORG $0000

RJMP reset

RESET:

/\*CONFIGURACION DE PUNTERO DE PILA PARA QUE LAS

LLAMADAS A SUBRUTINAS PUEDAN TRABAJAR CORRECTAMENTE\*/ LDI A,LOW(RAMEND)

OUT SPL,A

LDI A,HIGH(RAMEND)

95

OUT SPH,A

; DECLARACION DE PUERTOS DE ENTRADA Y SALIDA LDI A,0XFF

OUT DDRB,A

LDI A, 0B01000000

OUT PORTE,A OUT DDRE,A NOP

rcall lcdInit

//se inicializa algunas variables inicio:

ldi A,0

mov dineroing,A mov seleccion,A mov opcion,A mov mon25,A mov mon50,A mov dolar,A

rcall mensaje\_inicial rcall mensaje\_espera ldi A,0

mov opcion,A

//selecciono el tipo de bebida y la elijo con los botones

//del puerto E 4 y 5

comienza:

clr A

rcall DELAY

in A,PINE

ANDI A,0B00110000

cpi A,0x10 breq selopc cpi A,0x20 breq pedir1 rjmp comienza PEDIR1:

RJMP PEDIR

// selecciono los diversos tipos de bebidas

// y muestro en la lcd el nombre de las bebidas selopc:

inc opcion

CPI OPCION,5

BREQ ENCERAOPCION

mov A1,opcion cpi A1,1

breq muestrapepsi cpi A1,2

breq muestraprofit1

cpi A1,3

breq muestra220V1

cpi A1,4

96

breq muestraclub1

ENCERAOPCION: LDI OPCION,0 muestraprofit1: RJMP muestraprofit muestra220V1: RJMP muestra220V muestraclub1:

RJMP muestraclub muestrapepsi:

rcall lcd\_clear rcall MPEPSI

rcall VALORPEPSI rjmp comienza muestraprofit:

rcall lcd\_clear rcall MPROFIT

rcall VALORPROFIT rjmp comienza muestra220V:

rcall lcd\_clear

. rcall M220V

rcall VALOR220V rjmp comienza muestraclub:

rcall lcd\_clear rcall MCLUB

rcall VALORCLUB

rjmp comienza

/\*===========================================================

=============MUESTRO LOS PRECIOS DE LAS BEBIDAS =============

=EN LOS DISPLAYS DE 7 SEGMENTOS UTILIZANDO LA MULTIPLEXACION============================================\*/

VALORPEPSI: CLR R24

CLR A LAZO1:

INC R24

sbi portb,7

// sbi portf,7

ldi A,0B10111111 out PORTB,A rcall delay1

cbi portb,7

// cbi portf,7

sbi porte,6

ldi A,0B01101101

out PORTB,A

97

rcall delay1

cbi porte,6

CPI R24,250

BREQ FIN1

RJMP LAZO1

FIN1: RET

VALORPROFIT: CLR R24

CLR A LAZO2: INC R24 sbi portb,7

// sbi portf,7

ldi A,0B10000110 out PORTB,A rcall delay1

cbi portb,7

//cbi portf,7

sbi porte,6

ldi A,0B00111111 out PORTB,A rcall delay1

cbi porte,6

CPI R24,250

BREQ FIN2

RJMP LAZO2

FIN2: RET

VALOR220V: CLR R24

CLR A LAZO3: INC R24 sbi portb,7

//sbi portf,7

ldi A,0B10000110 out PORTB,A rcall delay1

cbi portb,7

// cbi portf,7

sbi porte,6

ldi A,0B01101101 out PORTB,A rcall delay1

cbi porte,6

CPI R24,250

BREQ FIN3

RJMP LAZO3

FIN3:

98

RET VALORCLUB: CLR R24

CLR A LAZO4: INC R24 sbi portb,7

// sbi portf,7

ldi A,0B11011011 out PORTB,A rcall delay1

cbi portb,7

//cbi portf,7

sbi porte,6

ldi A,0B00111111 out PORTB,A rcall delay1

cbi porte,6

CPI R24,250

BREQ FIN4

RJMP LAZO4

FIN4: RET pedir:

rcall lcd\_clear rcall ING\_DIN ldi A1,0

mov seleccion,A1

ldi A1,0x00

out PORTB,A1

comienzapedir:

in A,PINE

ANDI A,0B00110000 //FILTRO

cpi A,0x10

breq selopcdin

rjmp comienzapedir

/\*SELECCIONO EL TIPO DE MONEDA QUE DESEO INGRESAR\*/

selopcdin:

inc seleccion mov A,seleccion cpi A,1

breq ingreso25

cpi A,2

breq ingreso50

cpi A,3

breq ingreso1

cpi A,5

breq ingreso25

rjmp selopcdin

99

//INGRESANDO UNA MONEDA DE 25 CTVS

ingreso25:

rcall lcd\_clear ldi A,1

mov seleccion,A

rcall C25

mov A,mon25

rcall MOSTRARENDISPLAY

wait25:

in A,PINE

ANDI A,0B00110000

cpi A,0x10

breq comienzapedir cpi A,0x20

breq incen25 rjmp wait25 incen25:

rcall DELAY

ldi A,25

add dineroing,A

inc mon25

mov A,mon25

rcall MOSTRARENDISPLAY RCALL ejecutar

rjmp wait25

//INGRESANDO UNA MONEDA DE 50 CTVS

ingreso50:

rcall lcd\_clear rcall C50

mov A,mon50

rcall MOSTRARENDISPLAY

wait50:

in A,PINE

ANDI A,0B00110000

cpi A,0x10

breq comienzapedir cpi A,0x20

breq incen50 rjmp wait50 incen50:

rcall DELAY

ldi A,50

add dineroing,A

inc mon50

mov A,mon50

rcall MOSTRARENDISPLAY RCALL ejecutar

rjmp wait50

//INGRESANDO UNA MONEDA DE 1 DOLLAR

ingreso1:

100

rcall lcd\_clear rcall D1

mov A,dolar

rcall MOSTRARENDISPLAY

wait1:

in A,PINE

ANDI A,0B00110000

cpi A,0x10

breq comienzapedir1

cpi A,0x20 breq indol1 rjmp wait1 comienzapedir1:

RJMP comienzapedir indol1:

rcall DELAY

ldi A,100

add dineroing,A

inc dolar mov A,dolar

rcall MOSTRARENDISPLAY RCALL ejecutar

rjmp wait1

waitac:

in A,PINE

ANDI A,0B00110000 //FILTRO

cpi A,0x10

breq comienzapedir1 // comienzapedir1

rjmp waitac ejecutar:

ldi A,1

cp opcion,A breq evalPEPSI ldi A,2

cp opcion,A breq evalPROFIT ldi A,3

cp opcion,A breq eval220V ldi A,4

cp opcion,A

breq evalCLUB

/\*==================================

==EVALUA SI EL DINERO INGRESADO ES==

==SUFICIENTE PARA COMPRAR LA BEBIDA==

===================================\*/

evalPEPSI:

ldi A,50

cp dineroing,A

brsh entregaPEPSI

101

rjmp pedir evalPROFIT: ldi A,100

cp dineroing,A brsh entregaPROFIT rjmp pedir

eval220V:

ldi A,150

cp dineroing,A brsh entrega220V rjmp pedir evalCLUB:

ldi A,200

cp dineroing,A brsh entregaCLUB rjmp pedir

/\*===========================================================

==========

UNA VEZ EVALUADO Y SI ES MAYOR O IGUAL AL PRECIO SE PROCEDE A SIMULAR

LA ENTREGA Y SI ES MAYOR SE AGRADECE POR LA DONACION LO QUE NO SUCEDERA

CUANDO SE INGRESA EL VALOR EXACTO

===========================================================\*/

entregaPEPSI: rcall lcd\_clear rcall MPEPSI

rcall ROTARENTREGA

rcall DELAY

ldi A,0

out PORTB,A

ldi A,50

SUB dineroing,A LDI A,0

CPSE dineroing,A RCALL GRACIAS RCALL DELAY rjmp INICIO entregaPROFIT: rcall lcd\_clear

rcall MPROFIT

rcall ROTARENTREGA

rcall DELAY

ldi A,0

out PORTB,A

ldi A,50

SUB dineroing,A LDI A,0

CPSE dineroing,A

102

RCALL GRACIAS RCALL DELAY rjmp INICIO entrega220V:

rcall lcd\_clear rcall M220V

rcall ROTARENTREGA

rcall DELAY

ldi A,0

out PORTB,A

ldi A,50

SUB dineroing,A LDI A,0

CPSE dineroing,A RCALL GRACIAS RCALL DELAY rjmp INICIO entregaCLUB:

rcall lcd\_clear rcall MCLUB

rcall ROTARENTREGA

rcall DELAY

ldi A,0

out PORTB,A

ldi A,50

SUB dineroing,A LDI A,0

CPSE dineroing,A RCALL GRACIAS RCALL DELAY rjmp INICIO

/\*======================================

==SE MUESTRA LA CANTIDAD DE MONEDAS EN==

======UN DISPLAY DE 7 SEGMENTOS=======

=======================================\*/

MOSTRARENDISPLAY:

sbi porte,6

cpi A,0

breq carga0 cpi A,1 breq carga1

cpi A,2

breq carga2

cpi A,3

breq carga3

cpi A,4

breq carga4

cpi A,5

breq carga5

103

cpi A,6

breq carga6

cpi A,7

breq carga7

cpi A,8

breq carga8

cpi A,9

breq carga9

RET

// SE CARGAN LOS RESPECTIVOS NUMEROS

carga0:

ldi A,0x3F out PORTB,A rjmp salirmostrar

carga1:

ldi A,0x06 out PORTB,A rjmp salirmostrar

carga2:

ldi A,0x5B out PORTB,A rjmp salirmostrar

carga3:

ldi A,0x4F out PORTB,A rjmp salirmostrar

carga4:

ldi A,0x66 out PORTB,A rjmp salirmostrar

carga5:

ldi A,0x6D out PORTB,A rjmp salirmostrar

carga6:

ldi A,0x7D out PORTB,A rjmp salirmostrar

carga7:

ldi A,0x07 out PORTB,A rjmp salirmostrar

carga8:

ldi A,0x7F out PORTB,A rjmp salirmostrar

carga9:

ldi A,0x6F out PORTB,A rjmp salirmostrar

104

salirmostrar: RET

/\*=================================== HACE ROTAR LOS SEMENTOS DEL DISPLAY PARA SIMULAR LA ENTREGA DE LA BEBIDA

======================================\*/ ROTARENTREGA:

// sbi portb,7

ldi A,0x01

out PORTB,A rcall DELAY sec

rol A

out PORTB,A rcall DELAY sec

rol A

out PORTB,A rcall DELAY sec

rol A

out PORTB,A rcall DELAY sec

rol A

out PORTB,A rcall DELAY sec

rol A

out PORTB,A rcall DELAY sec

rol A

out PORTB,A

rcall DELAY

// cbi portb,7

RET

/\*====================================

=====MENSAJES MOSTRADOS EN LA LCD=====

=====================================\*/ ING\_DIN:

ldi char,'I'

ldi digit, 0

rcall lcd\_out ldi char,'N' inc digit rcall lcd\_out

105

ldi char,'G' inc digit rcall lcd\_out

ldi char,'D'

inc digit rcall lcd\_out ldi char,'I' inc digit rcall lcd\_out ldi char,'N' inc digit rcall lcd\_out

RET

rcall DELAY

mensaje\_inicial:

ldi char,'B' ldi digit, 0 rcall lcd\_out

ldi char,'E' inc digit rcall lcd\_out

ldi char,'B' inc digit rcall lcd\_out

ldi char,'I' inc digit rcall lcd\_out

ldi char,'D' inc digit rcall lcd\_out

rcall DELAY rcall lcd\_clear ldi char,'A'

ldi digit, 0

rcall lcd\_out ldi char,'S' inc digit

106

rcall lcd\_out

ret

rcall DELAY

MPEPSI:

ldi char,'P' ldi digit, 0 rcall lcd\_out

ldi char,'E' inc digit rcall lcd\_out

ldi char,'P' inc digit rcall lcd\_out

ldi char,'S'

inc digit rcall lcd\_out ldi char,'I' inc digit rcall lcd\_out

RET

rcall DELAY

MPROFIT:

ldi char,'P' ldi digit, 0 rcall lcd\_out

ldi char,'R' inc digit rcall lcd\_out

ldi char,'O' inc digit rcall lcd\_out

ldi char,'F'

inc digit rcall lcd\_out ldi char,'I' inc digit

107

rcall lcd\_out ldi char,'T' inc digit

rcall lcd\_out

RET

rcall DELAY

M220V:

ldi char,'2' ldi digit, 0 rcall lcd\_out

ldi char,'2' inc digit rcall lcd\_out

ldi char,'0' inc digit rcall lcd\_out

ldi char,'V'

inc digit

rcall lcd\_out

RET

rcall DELAY

MCLUB:

ldi char,'C' ldi digit, 0 rcall lcd\_out

ldi char,'L' inc digit rcall lcd\_out

ldi char,'U' inc digit rcall lcd\_out

ldi char,'B'

inc digit

RET

rcall lcd\_out rcall DELAY

108

mensaje\_espera: ldi char,'E' ldi digit, 0

rcall lcd\_out ldi char,'S' inc digit rcall lcd\_out ldi char,'P' inc digit rcall lcd\_out ldi char,'E' inc digit rcall lcd\_out ldi char,'R' inc digit rcall lcd\_out

ret

C25:

RET

C50:

ldi char,'A' inc digit rcall lcd\_out

ldi char,'C' ldi digit, 0 rcall lcd\_out

ldi char,'2' inc digit rcall lcd\_out

ldi char,'5' inc digit rcall lcd\_out

rcall DELAY

ldi char,'C' ldi digit, 0 rcall lcd\_out

ldi char,'5' inc digit rcall lcd\_out

109

ldi char,'0' inc digit rcall lcd\_out

RET

D1:

RET

rcall DELAY

ldi char,'1' ldi digit, 0 rcall lcd\_out

ldi char,'U' inc digit rcall lcd\_out

ldi char,'S' inc digit rcall lcd\_out

ldi char,'D' inc digit rcall lcd\_out

rcall DELAY

GRACIAS:

ldi char,'G' ldi digit, 0 rcall lcd\_out

ldi char,'R' inc digit rcall lcd\_out

ldi char,'A' inc digit rcall lcd\_out

ldi char,'C'

inc digit rcall lcd\_out ldi char,'I' inc digit rcall lcd\_out

110

rcall DELAY

rcall lcd\_clear ldi char,'A' LDI digit,0 rcall lcd\_out

RET

ldi char,'S' inc digit rcall lcd\_out

/\*===============================

===========RETARDOS==============

=================================\*/

DELAY:

ldi R25, $06

WGLOOP0: ldi R26, $FF WGLOOP1: ldi R27, $FF WGLOOP2: dec R27

brne WGLOOP2

dec R26

brne WGLOOP1

dec R25

brne WGLOOP0

RET

DELAY1:

ldi R25, $01

WGLOOP01: ldi R26, $0F WGLOOP11: ldi R27, $0F WGLOOP21: dec R27

brne WGLOOP21

dec R26

brne WGLOOP11

dec R25

brne WGLOOP01

RET

DELAY2:

ldi R25, $05

WGLOOP02: ldi R26, $0F WGLOOP12: ldi R27, $1F WGLOOP22: dec R27

brne WGLOOP22

dec R26

brne WGLOOP12

dec R25

brne WGLOOP02

RET

111

; delaying 3999 cycles:

delay1m:

ldi R25, $1F WGLOOP00: ldi R26, $2A WGLOOP111: dec R26

brne WGLOOP111

dec R25

brne WGLOOP00

; -----------------------------

; delaying 1 cycle:

nop

ret

; =============================

/\*====================================== INCLUYENDO LIBRERIA ENCARGADA DE LA LCD

=======================================\*/

.include "lcd\_driver.asm"

**4.4.4 SIMULACION EN PROTEUS MAQUINA ELECTRONICA**

**DE BEBIDAS.**

Simulación de una licorera electrónica donde mediante botones se al ofrece a la clientela cuatro tipos de bebidas mostradas en la LCD del AVR Butterfly (pepsi, profit, v220, club) como se muestra en la figura 4.21

Figura 4.21 Circuito de proteus Maquina de bebidas electrónica.

112

**4.5 CALCULADORA BINARIA.**

En este ejercicio se realiza la simulación de una calculadora binaria para resolver operaciones binarias como la AND, OR, EXOR, SUMA, RESTA, MULTIPLICACION. Para desarrollar esta calculadora haremos uso de un teclado matricial y de la LCD del AVR Butterfly

**4.5.1 DIAGRAMA DE BLOQUES CALCULADORA BINARIA.**

Se realizó el diagrama de bloque de una Calculadora Binaria como se indica en la figura 4.22, donde vemos que se realiza la simulación de una calculadora binaria para resolver operaciones binarias como la AND, OR, EXOR, SUMA, RESTA, MULTIPLICACION.

TECLADO

BUTTERFLY LCD

9

Figura 4.22 Diagrama de Bloque Calculadora Binaria.

**4.5.2 ALGORITMO CALCULADORA BINARIA .**

Se observa el siguiente diagrama de flujo de la calculadora binaria de nuestro proyecto donde describe el funcionamiento de una manera que se pueda entender cada fase de dicho proyecto inicialmente presionamos una tecla y validamos cada digito ingresado ya que por ser una calculadora binaria solo acepta cero o uno.

113

A continuacion el diagrama de flujo de la calculadora binaria donde hace la validacion de cada digito ingresado donde es comparado si es mayor a 10 si es mayor se compara el numero1 si es mayor a 0 si se cumple esto se ejecuta la operación elegida.

Caso contrario si el digito es menor que 10, se valida si es igual a cero luego de esto podría cumplir dos opciones de digito igual a 0 y digito igual a 1 para esta opciones si son ejecutadas se cumple un clc, se hace un rlc,A y se le asigna el valor de A a numero1.

Figura 4.23 Diagrama ASM Calculadora Binaria.

114

A continuación se muestra la continuación del diagrama de flujo de la calculadora binaria donde se valida cada operación para que esta pueda ser presentada por medio de la LCD de la AVR BUTTERFLY si el operador es igual a cerosi se cumple se realiza la suma de dígitos y se muestra el resultado , también se muestra el algoritmo para el exceso de numero

Figura 4.24 (continuación) Diagrama ASM Calculadora Binaria.

115

Figura 4.25 (continuación) Diagrama ASM Calculadora Binaria.

116

Figura 4.26 (continuación) Diagrama ASM Calculadora Binaria

117

**4.5.3 PROGRAMACION CALCULADORA BASICA…**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// MICROCONTROLADORES AVANZADOS

// GRUPO: # 7

// INTEGRANTES: RONALD RIVERA - NELSON CASTRO

// NOMBRE DEL PROYECTO: CALCULADORA BINARIA.

// DESCRIPCION: SE REALIZA LA SIMULACION DE UNA CALCULADORA BINARIA PARA RESOLVER OPERACIONES BINARIAS COMO AND, OR, EXOR,SUMA, RESTA, MULTIPLICACION.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.include "m169def.inc"

//==========DECLARACION DE VARIABLES========================

.def char = r21

.def digit = r22

.def nibble = r23

.def A = r16

.def digito = r19

.def opcion = r24

.def numero1 = r10

.def numero2 = r11

.def operacion= r12

.def resultado= r13

.def ingresado= r28

//===========DECLARACION DEL VECTOR DE RESET================

.ORG $0000

RJMP reset reset:

.ESEG

eeconst:.DB 0xff

.CSEG

//=====CONFIGURACION DE PUNTERO DE PILA PARA QUE LAS=========

//LLAMADAS A SUBRUTINAS PUEDAN TRABAJAR CORRECTAMENTE \*/

LDI A,LOW(RAMEND) OUT SPL,A

LDI A,HIGH(RAMEND); OUT SPH,A

//=====DECLARACION DE PUERTOS DE ENTRADA Y SALIDA=========== SER A

OUT PORTE,A

ldi A,0xF0

OUT DDRB,A ldi A,0x0F OUT PORTB,A

118

//============================================================

//======RUTINA PRINCIPAL======================================

//============================================================

cont:

rcall lcdInit //INICIALIZA LCD

//============================================================

//===========CODIGO INICIAL===================================

//===========CALCULADORA BINARIA===========================

inicio:

ldi char,'C' ldi digit, 0 rcall lcd\_out ldi char,'A' inc digit rcall lcd\_out ldi char,'L' inc digit rcall lcd\_out ldi char,'C' inc digit rcall lcd\_out ldi char,'U' inc digit rcall lcd\_out

rcall DELAY rcall lcd\_clear ldi char,'L'

ldi digit, 0 rcall lcd\_out ldi char,'A' inc digit rcall lcd\_out ldi char,'D' inc digit rcall lcd\_out ldi char,'O' inc digit rcall lcd\_out ldi char,'R' inc digit rcall lcd\_out ldi char,'A' inc digit rcall lcd\_out

rcall DELAY

ldi char,'B' ldi digit, 0 rcall lcd\_out ldi char,'I'

119

inc digit rcall lcd\_out ldi char,'N' inc digit rcall lcd\_out ldi char,'A' inc digit rcall lcd\_out ldi char,'R' inc digit rcall lcd\_out ldi char,'I' inc digit rcall lcd\_out

rcall DELAY rcall lcd\_clear ldi char,'A'

ldi digit, 0 rcall lcd\_out rcall DELAY rcall lcd\_clear

//============================================================

//=======PROCESOS CALCULADORA BINARIA======================

va:

ldi digito,0xFF

//============================================================

//====PROCESO DE INGRESO DE DIGITO============================

ingreso1:

ldi digito,0xFF

rcall TECLA

cpi digito,0xFF breq ingreso1 cpi digito,10

breq saltacomando1 brlo ingresandodigito saltacomando1:

rjmp comando

//============================================================

//===========PROCESO DE INGRESO DE DIGITO====================

//============================================================

ingresandodigito:

cpi digito,0

brge filtroingreso1

rjmp ingreso1

120

//============================================================

//===========PROCESO DE FILTRO DE DIGITO======================

//============================================================

filtroingreso1:

cpi digito,2 brlo pasofiltro1 rjmp ingreso1 pasofiltro1:

cpi digito,0 breq desplaza01 cpi digito,1 breq desplaza11

//============================================================

//=======PROCESO DE DESPLAZAMIENTO 01=======================

//============================================================

desplaza01:

clc

rol numero1 rcall mos\_num rcall DELAY inc digit

inc ingresado rjmp ingreso1

//============================================================

//===========PROCESO DE DESPLAZAMIENTO 11===================

//============================================================

desplaza11:

sec

rol numero1 rcall mos\_num rcall DELAY inc digit

inc ingresado rjmp ingreso1 comando:

ldi A,1

cp numero1,A brlo ingreso1 ldi A,10

cp digito,A breq cargasuma ldi A,11

cp digito,A breq cargaresta ldi A,12

cp digito,A breq cargamult ldi A,13

121

cp digito,A breq cargaor ldi A,14

cp digito,A breq cargaand ldi A,15

cp digito,A breq cargaexor rjmp ingreso1

//============================================================

//===========PROCESO DE INGRESO2=============================

//============================================================

ingreso2:

ldi digito,0xFF

rcall TECLA

cpi digito,0xFF breq ingreso2 cpi digito,20 breq finishop

brlo ingresandodigito1

ingresandodigito1:

cpi digito,0

brge filtroingreso2 rjmp ingreso2 filtroingreso2:

cpi digito,2 brlo pasofiltro2 rjmp ingreso2 pasofiltro2:

cpi digito,0 breq desplaza02 cpi digito,1 breq desplaza12 desplaza02:

clc

rol numero2 rcall mos\_num rcall DELAY inc digit

inc ingresado rjmp ingreso2 desplaza12:

sec

rol numero2 rcall mos\_num rcall DELAY inc digit

inc ingresado rjmp ingreso2

122

//===========================================================

//===========PROCESO DE CARGAR LA OPERACION SUMA==========

//============================================================

cargasuma:

mov operacion,digito rcall lcd\_clear

clr digit

clr ingresado rjmp ingreso2

//============================================================

//=======PROCESO DE CARGAR LA OPERACION RESTA==============

//============================================================

cargaresta:

mov operacion,digito rcall lcd\_clear

clr digit

clr ingresado rjmp ingreso2

//============================================================

//====PROCESO DE CARGAR LA OPERACION MULTIPLICACION========

//============================================================

cargamult:

mov operacion,digito rcall lcd\_clear

clr digit

clr ingresado rjmp ingreso2

//============================================================

//====PROCESO DE CARGAR LA OPERACION AND===================

//============================================================

cargaand:

mov operacion,digito rcall lcd\_clear

clr digit

clr ingresado rjmp ingreso2

//============================================================

//===========PROCESO DE CARGAR LA OPERACION OR=============

//============================================================

cargaor:

mov operacion,digito rcall lcd\_clear

clr digit

clr ingresado rjmp ingreso2

123

//============================================================

//===========PROCESO DE CARGAR LA OPERACION XOR============

//============================================================

cargaexor:

mov operacion,digito rcall lcd\_clear

clr digit

clr ingresado rjmp ingreso2 finishop:

ldi A,10

cp operacion,A

breq suma ldi A,11

cp operacion,A

breq resta ldi A,12

cp operacion,A

breq mult ldi A,13

cp operacion,A

breq orop ldi A,14

cp operacion,A

breq andop ldi A,15

cp operacion,A breq exoop suma:

rcall lcd\_clear

add numero1,numero2 mov resultado,numero1 ldi A,0b00011111

cp resultado,A brlo visualizar brge excesosum rjmp visualizar

//============================================================

//===========MENSAJE DE EXCESO DE DIGITOS====================

//============================================================

excesosum:

rjmp excedio

124

//============================================================

//===========MUESTRA EN PANTALLA LA RESTA ==================

//============================================================

resta:

rcall lcd\_clear

sub numero1,numero2 mov resultado,numero1 ldi A,0b11100000

cp resultado,A brlo visualizar brge complemento rjmp visualizar

//============================================================

//===========MUESTRA EN PANTALLA EL COMPLEMENTO===========

//============================================================

complemento: rcall lcd\_clear com resultado inc resultado ldi digit,0

ldi digito,1

rcall MOS\_NUM

rjmp visualizar

//============================================================

//===========MUESTRA EN PANTALLA LA MULTIOLICACION=========

//============================================================

mult:

rcall lcd\_clear

mul numero1,numero2

mov resultado,r0 ldi A,0b00011111 cp resultado,A brlo visualizar

brge excedio rjmp visualizar excedio:

rcall lcd\_clear rcall EXCESO rjmp finalizando

//============================================================

//===========MUESTRA EN PANTALLA LA OPERACION OR===========

//============================================================

orop:

rcall lcd\_clear

or numero1,numero2 mov resultado,numero1 rjmp visualizar

125

//============================================================

//===========MUESTRA EN PANTALLA LA OPERACION AND==========

//============================================================

andop:

rcall lcd\_clear

and numero1,numero2 mov resultado,numero1 rjmp visualizar

//============================================================

//===========MUESTRA EN PANTALLA LA OPERACION XOR==========

//============================================================

exoop:

rcall lcd\_clear

eor numero1,numero2 mov resultado,numero1 rjmp visualizar visualizar:

ldi digit,5

//============================================================

//===========MUESTRA LOS RESULTADOS DE LAS OPERACIONES=====

//============================================================

mostrando:

sbrc resultado,0 rjmp mostrar0 rjmp mostrar1 mostrar1:

ldi digito,0

rcall MOS\_NUM

clc

ror resultado dec digit

cpi digit,0

breq finalizando rjmp mostrando mostrar0:

ldi digito,1

rcall MOS\_NUM

clc

ror resultado dec digit

cpi digit,0

breq finalizando rjmp mostrando finalizando:

rcall DELAY rcall lcd\_clear clr digito

126

clr digit

clr numero1 clr numero2 clr resultado rjmp ingreso1

MOS\_NUM:

ldi r16,0x30 add digito,r16 mov char,digito rcall lcd\_out

ldi r16,0x30 sub digito,r16 rcall DELAY RET

EXCESO:

ldi char,'E' ldi digit, 0 rcall lcd\_out ldi char,'X' inc digit rcall lcd\_out ldi char,'C' inc digit rcall lcd\_out ldi char,'E' inc digit

rcall lcd\_out ldi char,'S' inc digit rcall lcd\_out

ldi char,'O' inc digit rcall lcd\_out

rcall DELAY

RET

127

//============================================================

//===========RETARDOS=======================================

//===========================================================

DELAY:

ldi R25, $06

WGLOOP0: ldi R26, $FF WGLOOP1: ldi R27, $FF WGLOOP2: dec R27

brne WGLOOP2

dec R26

brne WGLOOP1

dec R25

brne WGLOOP0

RET

DELAY1:

ldi R25, $01

WGLOOP01: ldi R26, $0F WGLOOP11: ldi R27, $0F WGLOOP21: dec R27

brne WGLOOP21

dec R26

brne WGLOOP11

dec R25

brne WGLOOP01

RET

DELAY2:

ldi R25, $05

WGLOOP02: ldi R26, $0F WGLOOP12: ldi R27, $1F WGLOOP22: dec R27

brne WGLOOP22

dec R26

brne WGLOOP12

dec R25

brne WGLOOP02

RET

/\*===========================================================

====INCLUYENDO LIBRERIA ENCARGADA DE LA LCD===============

===========================================================\*/

.include "lcd\_driver.asm"

128

**4.5.4 SIMULACION EN PROTEUS CALCULADORA BINARIA.**

Simulación de una calculadora binaria la cual posee sencillo teclado matricial

(pulsadores y resistencias) como se indica en la figura 4.25, para resolver operaciones binarias como la AND, OR, EXOR, SUMA, RESTA, MULTIPLICACION.

Figura 4.27 Circuito de proteus Calculadora Binaria.

**Conclusiones**

1) La plataforma de trabajo implementada incluye el Kit de Desarrollo AvrButterfly es una herramienta de aprendizaje práctica, eficaz y muy confiable, ya que con el desarrollo de aplicaciones o ejercicios va mostrando progresivamente las características básicas del microcontrolador así como también las facilidades que nos ofrece el software que se utiliza para la elaboración de proyectos.

2) Se realizó el diseño y la implementación de la plataforma interactiva y se comprobó el funcionamiento de la aplicacio nes y ejercicios, con sendas prácticas de fácil entendimiento, quedando así demostrado el funcionamiento del Kit AvrButterfly.

3) El software de programación Avr Studio 4, es una herramienta poderosa que permite depurar y simular un proyecto antes de ser probado con el hardware además con el Kit AvrButterfly se mejora el área del aprendizaje que brinda al estudiante acceso a tecnologías innovadoras y recientes para mejorar su capacitación.

4) Se concluye que el Programa PROTEUS nos permite una excelente visualización dentro del circuito utilizado; a través de la simulación que se puede observar, analizar y mejorar y así obtener una mejor visualización del proyecto además se realiza un exhaustivo análisis del circuito implementado.

5) Se concluye que el diseño de este kit de desarrollo avrbutterfly con el uso de microcontroladores nos ha facilitado la vida; puesto que estos dispositivos son muy eficientes y se rigen con respecto a sus especificaciones o características de fabricación, gracias a esto hoy en día se han logrado grandes aplicaciones como: la electrónica, robótica, domótica, etc.

**Recomendaciones**

1) Se recomienda siempre realizar un Diagrama de flujo que nos permita tener un bosquejo general de la solución, en éste diagrama se deben colocar todos los pasos que vamos a realizar en la ejecución del programa. Este Diagrama debe ser escrito de una manera que el usuario lo pueda comprenderlo con un lenguaje claro, preciso y conciso para que cualquier persona que no esté familiarizada con los microcontroladores lo pueda comprender de una manera muy fácil.

2) Se recomienda estar informado con los datos teóricos de los diferentes dispositivos que se van a utilizar en este proyecto ya que de esta manera se sabrá más de cada elemento y se estará menos propensos a errores en el momento de implementarlo.

3) Se recomienda modularizar la programación de cada ejercicio o aplicación en diferentes etapas ya que de esta manera se podrá comprender la lógica de cada uno y así se podrá obtener un mejor beneficio para aplicarlo en diferentes campos de estudio.

4) Se recomienda leer el datasheet y tutoriales del Kit AVR Butterfly, toda la

configuración de los componentes de hardware y de sus pines ya que así se

facilitaría alguna la corrección de alguna conexión errónea puede generar

daños en algún componentes.

**ANEXOS**

Tabla 1.1 resumen de todas las instrucciones aritméticas y lógicas

Tabla 1.2 resumen de todas las instrucciones de control del programa

Tabla 1.3 resumen de todas las instrucciones de transferencia de datos

Tabla 1.4 resumen de todas las instrucciones de bits

Tabla. 2.1 Distribución de pines, AVR Butterfly Vs. PC

**Bibliografía**

[1] PARDUE, Joe, *C ProgrammingforMicrocontrollers*, tomo 1, 1raEdición,

Editorial Smiley Micros, Knoxville-Tennessee noviembre del 2011.

[2] SNILSBERG, Introduction to the Atmel AVR Butterfly,

http://www.atmel.com, 28 de noviembre del 2011.

[3] SNILSBERG, AVR Butterfly Evaluation Kit User,

http://www.atmel.com/products/avr/, 15 de octubre 2011

[4] SNILSBERG, Rolf Kristian , AVRProg User Guide, http://

www.atmel.com, 16 octubre del 2011.

[5] BARNETTE, M.D., AVR Simulation with the ATMEL AVR Studio 4,

http://www.avr.freaks.com, 27 de noviembre del 2011.

[6] SNILSBERG, Rolf Kristian, AVR065: LCD Driver for the STK502 and

AVR Butterfly, http://www.atmel.com, 17 de noviembre del 2011.

[7] AVR306: Using the AVR UART in C, http://www.atmel.com, 28 de

noviembre del 2011.

[8] Wikipedia, Información general RS232: http://es.wikipedia.org/wiki/RS-

232, 20 noviembre 2011.