UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

CENTRO UNIVERSITARIO UAEM ECATEPEC

“Sistema de detección de variables ambientales mediante dsPIC30F4013”

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

P R E S E N T A L A C.

Aketzali Naibi Altamirano Urrutia

ASESOR:

Dr. en C. Rodolfo Zolá García Lozano

REVISORES

M. en C.C. Enrique José Tinajero Pérez

M. en I.S.C. Osdali Bojórquez Islas

ECATEPEC DE MORELOS, ESTADO DE MÉXICO JUNIO, 2017

CARTA DE CESIÓN DE DERECHOS DE AUTOR

El (la) que suscribe ***AKETZALI NAIBI ALTAMIRANO URRUTIA*** Autor del trabajo escrito de evaluación profesional en la opción de **TESIS** con el título **"SISTEMA DE DETECCION DE VARIABLES AMBIENTALES MEDIANTE DSPIC30F4013"** por medio de la presente con fundamento en lo dispuesto en los artículos 5, 18, 24, 25, 27, 30, 32 y 148 de la Ley Federal de Derechos de Autor, así como los artículos 35 y 36 fracción II de la Ley de la Universidad Autónoma del Estado de México; manifiesto mi autoría y originalidad de la obra mencionada que se presentó en el Centro Universitario UAEM Ecatepec para ser evaluada con el fin de obtener el Título Profesional de **INGENIERIA EN COMPUTACION**

Así mismo expreso mi conformidad de ceder los derechos de reproducción, difusión y circulación de esta obra, en forma NO EXCLUSIVA, a la Universidad Autónoma del Estado de México; se podrá realizar a nivel nacional e internacional, de manera parcial o total a través de cualquier medio de información que sea susceptible para ello, en una o varias ocasiones, así como en cualquier soporte documental, todo ello siempre y cuando sus fines sean académicos, humanísticos, tecnológicos, históricos, artísticos, sociales, científicos u otra manifestación de la cultura.

Entendiendo que dicha cesión no genera obligación alguna para la Universidad Autónoma del Estado de México y que podrá o no ejercer los derechos cedidos.

Por lo que el autor da su consentimiento para la publicación de su trabajo escrito de evaluación profesional.

a) Texto completo ■
b) Por capítulo ☐
c) Solamente portada y tabla de contenido ☐

Se firma presente en la ciudad de Ecatepec de Morelos, Estado de México, a los 6 días del mes de Junio de 2017.

AKETZALI NAIBI ALTAMIRANO URRUTIA

UAEM | Universidad Autónoma del Estado de México

Ecatepec de Morelos, Edo. De Méx., a 24 de Mayo de 2017
ASUNTO: VOTO APROBATORIO DE ASESOR

LIA. ADRIANA MORALES LICONA
***JEFA DEL DEPARTAMENTO DE TITULACION* DEL**
CENTRO UNIVERSITARIO U.A.E.M ECATEPEC
P R E S E N T E

Por éste conducto me permito informarle que la pasante **C. AKETZALI NAIBI ALTAMIRANO URRUTIA** con el número de cuenta **1125515**, de la **LICENCIATURA EN INGENIERÍA EN COMPUTACIÓN**, ha concluido el desarrollo de su **TESIS**, con el título:

"Sistema de detección de variables ambientales mediante DSPIC30F4013"

Manifiesto que el borrador del trabajo escrito reúne las características necesarias para ser revisado por la Comisión especial nombrada para tal efecto.

DR. RODOLFO ZOLÁ GARCÍA LOZANO
NO. DE CÉDULA PROFESIONAL: 08708033

PATRIA, CIENCIA Y TRABAJO
"2017, Año del Centenario de la Promulgación de la Constitución Política de los Estados Unidos Mexicanos"

www.uaemex.mx

Av. José Revueltas no. 17 Col. Tierra Blanca, C.P. 55020, Ecatepec, Estado de México.
Tels: 5.7.87.36.26 Fax: 5.7.87.35.10

UAEM | Universidad Autónoma del Estado de México

Ecatepec de Morelos, Edo. De Méx., 07 de Junio de 2017
ASUNTO: VOTO APROBATORIO DE REVISORES

LIA. ADRIANA MORALES LICONA
JEFA DEL DEPARTAMENTO DE TITULACION DEL
CENTRO UNIVERSITARIO UAEM ECATEPEC
P R E S E N T E

Nos es grato comunicarle que el trabajo de **TESIS** titulado:

"SISTEMA DE DETECCION DE VARIABLES AMBIENTALES MEDIANTE DSPIC30F4013"

Que para obtener el título de: **INGENIERIA EN COMPUTACION**

Presenta la pasante: **AKETZALI NAIBI ALTAMIRANO URRUTIA**
Con números de cuenta: **1125515**

Cumplen con los requisitos teóricos-metodológicos suficientes para ser aprobada, pudiendo continuar con los trámites correspondientes para su impresión.

REVISORES

M. en I.S.C. OSDALI BOJORQUEZ ISLAS
CÉDULA PROFESIONAL: 10130353

M. en C.C. ENRIQUE JOSE TINAJERO PEREZ
CÉDULA PROFESIONAL: 9683968

PATRIA, CIENCIA Y TRABAJO
"2017, Año del Centenario de la Promulgación de la Constitución Política de los Estados Unidos Mexicanos"

Av. José Revueltas no. 17 Col. Tierra Blanca, C.P. 55020, Ecatepec, Estado de México.
Tels: 5.7.87.36.26 Fax: 5.7.87.35.10

UAEM | Universidad Autónoma del Estado de México

Ecatepec de Morelos, Edo. De México., a 07 de Junio de 2017

ASUNTO: IMPRESIÓN DE TRABAJO ESCRITO

C. AKETZALI NAIBI ALTAMIRANO URRUTIA
PASANTE DE LA INGENIERIA EN COMPUTACION
P R E S E N T E

Por este medio le comunico a usted que al haber cubierto los trámites correspondientes al desarrollo del trabajo escrito bajo la modalidad **TESIS** con el fin de obtener el Título Profesional, se le aprueba la **IMPRESIÓN DE SU TRABAJO** con el título:

"SISTEMA DE DETECCION DE VARIABLES AMBIENTALES MEDIANTE DSPIC30F4013"

Con el objetivo de establecer la fecha de Evaluación Profesional, le recuerdo que la presentación final del trabajo escrito es de su completa responsabilidad.

Sin más por el momento, reciba un cordial saludo.

PATRIA, CIENCIA Y TRABAJO
"2017, Año del Centenario de la Promulgación de la Constitución Política de los Estados Unidos Mexicanos"

LIA. ADRIANA MORALES LICONA
JEFA DEL DEPARTAMENTO DE TITULACION
DEL CENTRO UNIVERSITARIO UAEM ECATEPEC

CENTRO UNIVERSITARIO U.A.E.M
ECATEPEC
TITULACION

www.uaemex.mx

Av. José Revueltas no. 17 Col. Tierra Blanca, C.P. 55020, Ecatepec, Estado de México.
Tels: 5.7.87.36.26 Fax: 5.7.87.35.10

Carta de cesión de derechos de autor

Voto aprobatorio del asesor.

Voto aprobatorio de los revisores

Oficio de impresión de trabajo escrito

AGRADECIMIENTOS.

A mi brillante y bella madre,
que sin ella yo nada sería.
Ella que nunca pide nada y todo lo da a manos llenas.
Ella que siempre tiene las palabras justas para
aconsejarme, consolarme y confortarme.
Y que además me inspira para escribir esta dedicatoria.

Al Dr. Zolá
El profesor que desde el primer día
que me dio clases, me di cuenta que es una
excelente persona, maestro y amigo.
Gracias por su apoyo y guía para realizar esta tesis.

A mis revisores,
La valiente y gentil M. Osdali
y el atento caballero M. Enrique.
Que simplemente no tengo palabras para agradecer el
tiempo y atención que le dedicaron a éste trabajo.

ÍNDICE

I. Resumen…… 8
II. Introducción…… 8
III. Planteamiento del problema…….. 8
IV. Objetivo general…….. 13
V. Objetivo particular……. 13
Capítulo 1. Marco teórico…… 14
Capítulo 1.1 Dispositivos y referencias de medición…… 14
Capítulo 1.2 Historia…… 31
Capítulo 1.3. Programas de aplicación ……... 33
a) Ejercicio 1.1……... 34
b) Ejercicio 1.2……... 39
c) Ejercicio 1.3……... 42
d) Ejercicio 1.4……... 46
e) Ejercicio 2.1……... 52
f) Ejercicio 2.2……... 57
g) Ejercicio 2.3……... 60
h) Ejercicio 2.4……... 63
i) Ejercicio 3.1……... 67
j) Ejercicio 3.2……... 76
k) Ejercicio 4.1……... 87
l) Ejercicio 4.2……... 92
m) Ejercicio 4.3……... 100
Capítulo 2. Marco de referencia ……. 103
Capítulo 3. Metodología……... 106
Capítulo 3.1 Análisis de requerimientos…….. 107
Capítulo 3.2 Diseño del sistema……... 108
Capítulo 3.3 Diseño del programa…….... 112
Capítulo 3.4 Codificación…….. 116
Capítulo 3.5 Implementación……. 120
Capítulo 3.6 Ensamblaje y pruebas……. 125
a) Resultados……. 135
Capítulo 4. Conclusiones…… 137
VI. Anexos 139
Programación del DSPIC……... 139
Librerías…… 143
VII. Glosario…… 145
VIII. Bibliografía…… 146

I. RESUMEN

Se diseñó y desarrolló un detector de variables ambientales, capaz de realizar la medición de las siguientes variables ambientales a lo largo del día: ruido, luz y temperatura. El proyecto utiliza el convertidor analógico digital (ADC) integrado en el microcontrolador dsPIC30F4013. Los resultados servirán como punto de partida para el desarrollo de proyectos futuros de monitoreo ambiental mediante la utilización de los recursos de procesamiento matemático de señales que ofrece el dsPIC30F4013.

II. INTRODUCCIÓN

La ergonomía tiene su tarea en el análisis de los diversos factores del entorno que influyen sobre el sistema. Trata de prevenir la influencia negativa que las condiciones laborales y la vida cotidiana pueden tener sobre el individuo. En este sentido, la ergonomía busca eliminar los posibles riesgos y condiciones negativas del entorno de desarrollo, para así poder optimizar el rendimiento del individuo dentro del sistema hombre-máquina-entorno [1].

El medio ambiente de trabajo es el resultado de: el clima laboral de la tecnología, de los medios y procedimientos de trabajo así como del entorno del puesto, en el cual confluyen una serie de condiciones invisibles que el trabajador no ve, pero percibe, siente y asimila o rechaza [1].

III. PLANTEAMIENTO DEL PROBLEMA

El medio ambiente es uno de los elementos fundamentales de clara incidencia en el comportamiento, el rendimiento y la motivación del individuo, afectándolo directamente en su salud, su desempeño y su comodidad.

Los efectos de elementos visibles (mobiliario, individuos, herramientas, etc.) e invisibles (viento, ruido, contaminación, etc.), se combinan de tal manera que se constituyen en elementos extremosos y contaminantes que destruyen la integridad del individuo [1].

El ambiente de trabajo tiende a deteriorarse a medida que transcurre el tiempo, unas veces como consecuencia de la fatiga física y otras como resultado del aburrimiento y la

falta de motivación. Por lo que se hace necesario controlar que las condiciones ambientales sean adecuadas para evitar llegar a sobrepasar los límites de resistencia al esfuerzo del individuo.

Las principales condiciones de trabajo que influyen en la salud y rendimiento del individuo son [1]:

- Iluminación
- Temperatura
- Ruido
- Vibración

Iluminación

La iluminación es uno de los elementos de los cuales depende la eficiencia laboral del hombre. De esta manera se incrementa la capacidad de trabajo y del sistema visual del conjunto hombre máquina, evitando además errores e incrementando la productividad y comodidad. El grado de iluminación responde lógicamente al tipo de trabajo que se ejecuta y se mide en función del enceguecimiento y el índice de incomodidad.

Temperatura

La temperatura influye en el bienestar, comodidad, rendimiento y seguridad del trabajador. El excesivo calor produce fatiga, necesitándose más tiempo de recuperación o descanso que si se tratase de temperatura normal. Sus efectos varían de acuerdo con la humedad del ambiente. Por otro lado, el frío también perjudica al individuo, ya que hace perder agilidad, sensibilidad y precisión en las manos y resulta un riesgo en contra de su seguridad.

Ruido

El ruido también es conocido como “sonido no deseado”, consiste en una vibración experimentada a través del aire cuyos parámetros obedecen al de un tono simple (frecuencia e intensidad).

Los sonidos audibles para el ser humano son los que generalmente se encuentran dentro del rango de frecuencias 20 a 20,000 Hz. De acuerdo al informe de la OMS, los sonidos soportables, llamado umbral de audición, son aquellos que no superan los 80 dB. Si se trata de sonido ambiental o permanente, se ha establecido una medida normal que no supere los 55 dB durante el día y 45 dB durante la noche. Los sonidos que generan mayor daño son los que superan los 100 dB, llamado también umbral del dolor acústico [2].

En el primer caso puede escucharse hasta ocho horas seguidas, pero los que utilizan equipos de mayor potencia, con niveles que alcanzan los 130 dB, no deberían escucharse más de 4 minutos sostenidamente.

Uno de los efectos más desfavorables son los trastornos en la audición. Que suceden cuando se sostienen altos sonidos por períodos prolongados, lo que afecta las funciones de los cilios del oído interno. Con el tiempo, el daño de estas células puede llevar a la pérdida gradual de la audición [2].

La aplicación de los conocimientos de la ergonomía ambiental ayuda al diseño y evaluación de puestos y estaciones de trabajo, con el fin de incrementar el desempeño de quienes laboran en ellos.

Aunque los factores anteriores han sido muy estudiados en el ámbito laboral, también aplican en los ambientes académicos e incluso en la vida cotidiana, con el fin de mantener condiciones adecuadas de comodidad, productividad y seguridad.

En lo que respecta al aspecto de seguridad, una de las condiciones que ha adquirido gran interés en los últimos años es la radiación UV debido al crecimiento de padecimientos crónicos relacionados con la salud de la piel e incluso con el sistema inmunitario.

Todos nos encontramos expuestos al Sol que emite luz, calor y radiación UV. Además del Sol, la radiación UV puede ser procedente de numerosas fuentes artificiales utilizadas en la industria, el comercio y durante el tiempo libre [3].

La radiación solar ultravioleta o radiación UV es una parte de la energía radiante del sol que se transmite en forma de ondas electromagnéticas en cantidad constante solar. Su longitud de onda fluctúa entre 100 y 400 nm y constituye la porción más energética del espectro electromagnético que incide sobre la superficie terrestre. Se divide en tres bandas en función de su longitud de onda:

UVA.- Su longitud de onda fluctúa entre 315 y 400 nm. Alcanza totalmente la superficie terrestre, no es retenida por la atmósfera.

UVB.- Su longitud de onda fluctúa entre 280 a 315 nm. El 90% se bloquea por el ozono y el oxígeno de la atmósfera. Es más energética y dañina para la biosfera que la radiación UVA.

UVC.- Su longitud de onda fluctúa entre 100 y 280 nm y constituye la fracción más energética. Este tipo de radiación y otras partículas energéticas (rayos X, rayos gamma y rayos cósmicos) son retenidas totalmente en las regiones externas de la atmósfera y no alcanzan la superficie terrestre [4].

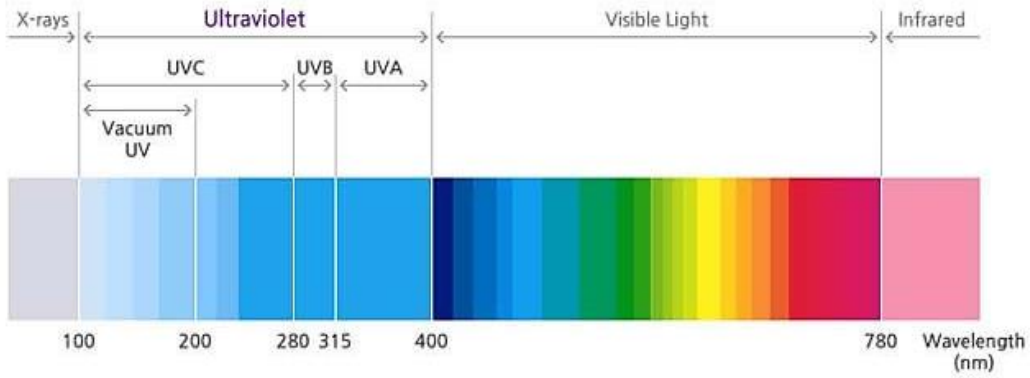


Fig. 1 Longitud de onda de radiación UV

Tanto los rayos UVB como los UVC son altamente nocivos. Los rayos UVA, por su parte, no tienen por qué serlo siempre y cuando se tomen las medidas necesarias al respecto [4].

La exposición a la radiación solar puede producir, en el ser humano, efectos agudos y crónicos en la salud de la piel, los ojos y el sistema inmunitario. La radiación UV acelera

el envejecimiento de la piel y la pérdida gradual de su elasticidad produciendo arrugas y piel seca y áspera. Por otra parte, la radiación ultravioleta (UV) es el factor de riesgo principal para la mayoría de los tipos de cáncer de piel. Las personas que se exponen mucho a los rayos UV procedentes de estas fuentes tienen un mayor riesgo de cáncer de piel [3].

Los rayos UV dañan el ADN de las células de la piel. Los diferentes tipos de cáncer de piel comienzan cuando este daño afecta el ADN de los genes que controlan el crecimiento de las células de la piel [5].

El grado de exposición a esta radiación depende de la intensidad de la luz, del tiempo de exposición y de si la piel ha estado protegida. Las personas que viven en áreas donde están expuestas todo el año a una luz solar intensa tienen mayor riesgo de desarrollar cáncer de piel. Además, estar largo tiempo a la intemperie por motivos de trabajo u ocio sin protegerse con ropas adecuadas y protección solar, incrementa la posibilidad de desarrollarlo.

Con base en lo planteado en esta sección es posible concluir que cada uno de estos aspectos ambientales influye notablemente en la salud y desempeño de los individuos, siendo unos más perjudiciales que otros. Es por tal motivo que se tiene el interés de medir cada una de estas variables. Siendo éste el objetivo del presente trabajo de tesis.

IV. OBJETIVO GENERAL

Diseñar y fabricar sistemas que permitan la medición y procesamiento de diferentes variables ambientales aplicados a la implementación de circuitos de alerta o alarma en función de las condiciones ambientales. Con base en las posibilidades de aplicación de este proyecto, es de particular interés el desarrollo futuro de un sistema de monitoreo y alarma de radiación UV. Debido a la amplitud del proyecto y considerando establecer las bases técnicas necesarias para el desarrollo del mismo, en esta etapa inicial se define el siguiente objetivo:

Diseñar y desarrollar un sistema que permita la captura y procesamiento de las siguientes variables ambientales:

1. Iluminación
2. Temperatura
3. Nivel de ruido

V. OBJETIVOS PARTICULARES

El sistema permitirá realizar la medición de las variables ambientales y desplegará el resultado mediante un display de LCD.

Para cada una de las variables analógicas el sistema permitirá el despliegue de los valores registrados a lo largo del día.

El usuario podrá definir parámetros de alerta para cada una de las variables, de forma que cuando se detecte que alguna de las variables está fuera de rango el sistema emita una señal de alerta.

El sistema será desarrollado con el dsPIC30F4013.

CAPÍTULO 1. MARCO TEÓRICO

En el siguiente capítulo se describen los dispositivos y componentes necesarios que conforman el proyecto a desarrollar. Así como también se hace una breve mención de la historia del DSPIC.

Además se incluyen ejercicios previos al proyecto final, los cuales su nivel de dificultad va en incremento conforme se va avanzando. Donde se hace uso del DSPIC y otros componentes, que son de utilidad para la mejor comprensión del funcionamiento del microcontrolador.

CAPÍTULO 1.1 DISPOSITIVOS Y REFERENCIAS DE MEDICIÓN

1.1.1 AMPLIFICADOR OPERACIONAL

Un amplificador operacional (op-amp) es un dispositivo amplificador electrónico de alta ganancia, acoplado en corriente continua que tiene dos entradas y una salida.

En esta configuración, la salida del dispositivo es de cientos de miles de veces mayor que la diferencia de potencial entre sus entradas. Por lo tanto, el op-amp tiene como objetivo elevar el valor de la tensión, corriente o potencia de una señal variable en el tiempo, procurando mantenerla lo más fiel posible [7].

1.1.1.1 Símbolo electrónico

La Fig. 2 contiene dos características importantes comunes a los amplificadores operacionales típicos: la entrada diferencial y la salida de un terminal. Idealmente este

símbolo significa que el amplificador tiene ganancia de tensión infinita, impedancia de entrada infinita e impedancia de salida cero.

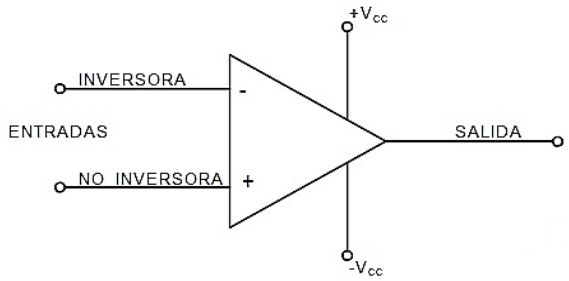


Fig. 2 Símbolo electrónico del amplificador operacional

El amplificador operacional ideal representa un amplificador de tensión perfecto y a menudo se denomina *fuente de tensión controlada por tensión (VCVS – Voltage-Controlled Voltage Source)* [8].

En la Fig. 3 se puede visualizar un VCVS, donde $\boldsymbol{R}_{in}$ es infinita y $\boldsymbol{R}_{out}$ es cero.

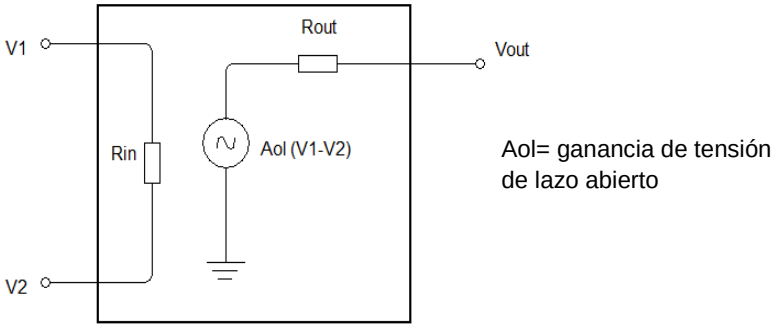


Fig. 3 Circuito equivalente a un op-amp

1.1.1.2 Amplificador inversor

El amplificador inversor utiliza realimentación negativa para estabilizar la ganancia de tensión total. La razón por la que se necesita estabilizar la ganancia de tensión total es porque A_{ol} resulta demasiado grande e inestable para ser útil sin alguna forma de realimentación.

1.1.1.3 Realimentación negativa inversora

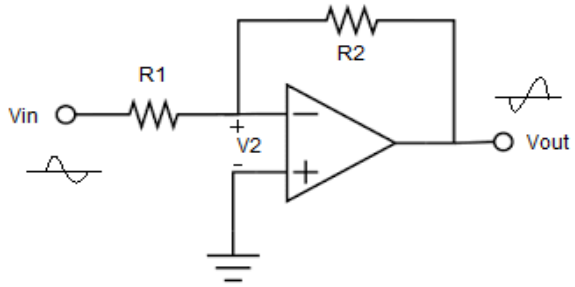


Fig. 4 Amplificador Inversor

En la Fig. 4 una tensión de entrada V_{in} excita la entrada inversora a través de la resistencia R_1, lo que produce una tensión de entrada inversora en V_2. La tensión de entrada se amplifica mediante la ganancia de tensión de lazo abierto para producir una tensión de salida invertida. La tensión de salida se realimenta hacia la entrada a través de la resistencia de realimentación R_2, lo que produce una realimentación negativa porque la salida está desfasada 180° con respecto a la entrada [8].

Así es como la alimentación negativa estabiliza la ganancia total de tensión: Si la ganancia de tensión de lazo abierto A_{ol} crece por alguna razón, la tensión de salida crecerá y realimentará más tensión a la entrada inversora.

1.1.1.4 Amplificador Comparador

Los amplificadores operacionales se utilizan a menudo como comparadores para comparar la amplitud de un voltaje con otro. Un comparador es un circuito que tiene dos terminales de entrada (inversor y no inversor) y una terminal de salida, que siempre está en uno de los dos estados, dependiendo si la tensión es alta o baja.

La manera más simple de construir un comparador consiste en conectar un amplificador operacional sin resistencias de alimentación (Fig. 5). Dada la alta ganancia de tensión de lazo abierto, una tensión de entrada positiva ocasiona una saturación positiva, y una tensión de entrada negativa provocará una saturación negativa. A éste comparador se le conoce como *detector de cruce por cero*.

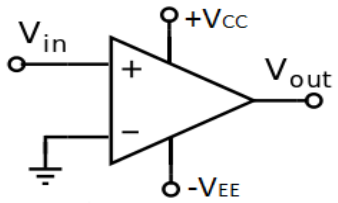


Fig. 5 Comparador detector de cruce cero.

Se le conoce como detector de cruce por cero, ya que idealmente *la tensión de salida conmuta de alta a baja o viceversa cuando la tensión de entrada pasa por el valor cero.* La fig. 6 muestra la respuesta de un detector de cruce por cero.

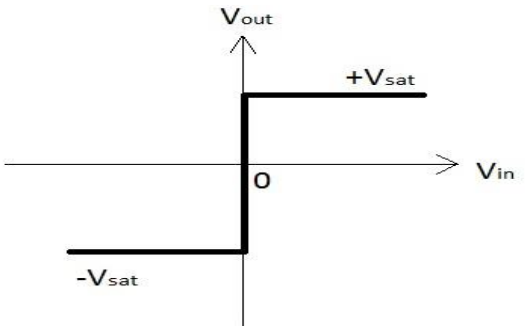


Fig. 6 Respuesta de comparador.

En ciertas aplicaciones es necesario que el punto de conmutación sea diferente de cero. Polarizando alguna de las entradas se consigue desplazar dicho punto donde se desee. Puede ser aplicado un divisor de voltaje para fijar el voltaje de referencia.

Cuando V_{in} es mayor que V_{ref}, la tensión diferencial de entrada es positiva y la tensión de salida está a nivel alto. Si V_{in} es menor que V_{ref}, la tensión referencial de entrada es negativa y la tensión de salida está a nivel bajo.

1.1.2 FOTORRESISTENCIA (LDR)

La fotorresistencia (Fig. 7) es una resistencia cuyo valor dependen de la energía luminosa incidente en ella, específicamente son resistencias cuyo valor de resistividad disminuye a medida que aumenta la energía luminosa incidente sobre ella y viceversa [9].

Una fotorresistencia se compone de un material semiconductor cuya resistencia varía en función de la iluminación. Es por ello por lo que también se le llama resistencia dependiente de luz (light dependent resistors), fotoconductores o células fotoconductoras.

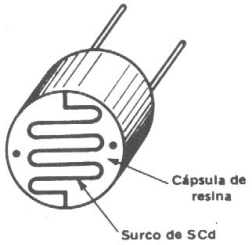


Fig. 7 Fotorresistencia

1.1.2.1 Símbolo electrónico

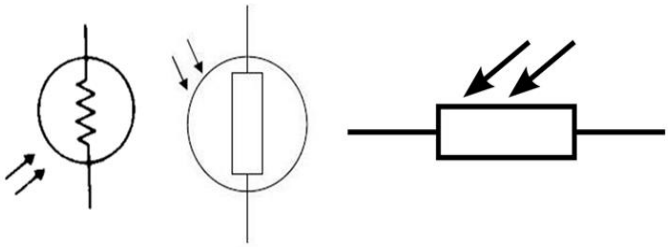


Fig. 8 Símbolo electrónico

1.1.2.2 Materiales de fabricación y características generales

Un fotorresistor está hecho de un semiconductor de alta resistencia. Los materiales fotosensibles más utilizados para la fabricación de las resistencias LDR son: el sulfuro de talio, el sulfuro de cadmio, el sulfuro de plomo, y el seleniuro de cadmio [10].
Las fotorresistencias se caracterizan por la ecuación:

$$R = AE^{-\alpha}$$

Dónde:

R: resistencia de la fotorresistencia.
A, α: constantes que dependen del semiconductor utilizado.
E: densidad superficial de la energía recibida.

Sus valores ohmicos pueden llegar a medir en la oscuridad valores cercanos al MegaOhm (1MΩ) y expuestas a la luz mediremos valores en el entorno de los 100Ω. Y el tiempo de respuesta típico de un LDR está en el orden de la décima de segundo [9].

1.1.2.3 Divisores de Tensión o Voltaje

La fórmula del circuito esencial de un divisor de tensión, también llamado divisor de potencial o divisor de voltaje, es:

$$V_{out} = \frac{R_2}{R_1 + R_2} \cdot V_{in}$$

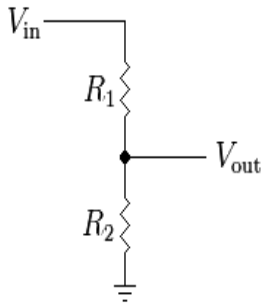


Fig. 9 Divisor resistivo.

Dónde $\boldsymbol{R_1}$ y $\boldsymbol{R_2}$ pueden ser cualquier combinación de resistencias en serie.

Dependiendo de la configuración del divisor de voltaje con la fotorresistencia y sustituyendo a $\boldsymbol{R}_1$ o $\boldsymbol{R}_2$, este trabajará como un sensor de oscuridad o un sensor de luz. El circuito divisor de tensión dará una tensión de la salida que cambia con la iluminación, de forma inversamente proporcional a la cantidad de luz que reciba [7].

Si se toma a $\boldsymbol{R}_2$ como la fotorresistencia (LDR) y ésta se encuentra colocada como se muestra en la figura 10, la salida de voltaje será menor cuando la luz incida sobre la LDR.

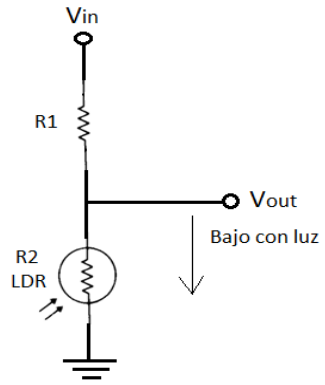


Fig. 10 Divisor de tensión. $\boldsymbol{V_{out}}$ *bajo.*

Por el contrario, si se toma a $\boldsymbol{R_1}$ como LDR y se encuentra colocada como se muestra en la figura 11, la salida de voltaje será mayor cuando la luz incide sobre la LDR.

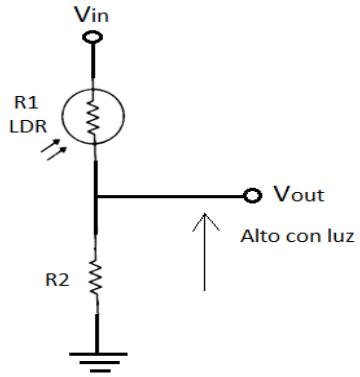


Fig. 11 Divisor de tensión. $\boldsymbol{V_{out}}$ *alto.*

1.1.2.4 Aplicaciones

La mayoría de las fotorresistencias son empleadas en iluminación, apagado y encendido de alumbrado, en alarmas, en cámaras fotográficas, y/o en medidores de luz. Las de la gama infrarroja se encuentran en control de máquinas y detección de objetos [10].

1.1.3. MICRÓFONO ELECTRET

El micrófono de condensador o electret (Fig. 12) es una variante del micrófono de condensador, que utiliza un electrodo, se polarizan las placas durante su fabricación por lo que no necesita alimentación, a excepción del preamplificador interno. [8]

Fig. 12 Micrófono electret

El micrófono electret puede ser omnidireccional o unidireccional, son robustos y de tamaño reducido, lo cual facilita su manipulación para diversas aplicaciones.

1.1.3.1 Símbolo electrónico

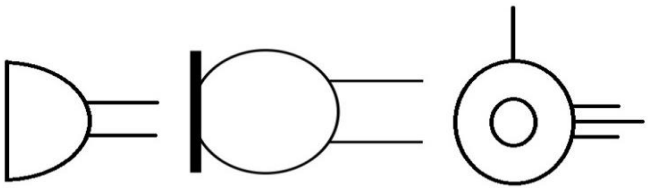


Fig. 13 Símbolo electrónico

1.1.3.2 Materiales de fabricación y características generales

El micro electret es una variante del micrófono de condensador que utiliza un electrodo (fluorocarbonato o policarbonato de fluor), lámina de plástico que al estar polarizado no necesita alimentación. Las placas están polarizadas, es decir, que están cargadas permanentemente desde su fabricación, esto permite extender su durabilidad.

Los micrófonos electret tienen una buena respuesta en frecuencia dentro del rango audible de 50 a 15.000Hz y una sensibilidad entre -50 dB y -70 dB. Son mucho más sensibles en la zona de los agudos. [11]

1.1.3.3 Aplicaciones

La mayoría de micrófonos electret son usados en televisoras como micrófonos de corbata o solapa. También son utilizados como micrófonos para ser pegados a instrumentos específicos o en celulares.

1.1.4 SENSOR DE TEMPERATURA LM35

El LM35 (Fig. 14) es un sensor de temperatura con una precisión calibrada de 1ºC y un rango que abarca desde -55º a +150ºC. [12]

La salida es lineal y equivale a 10 mV por cada grado Celsius, por lo tanto:

- 150 °C = 1500 mV
- 25 °C = 250 mV
- -55 °C = -550 mV

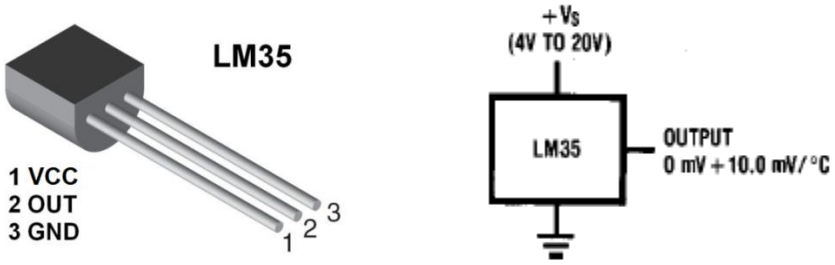


Fig. 14 Sensor LM35

1.1.4.1 Características generales [12]

- Calibrado directamente en grados Celsius.
- Escala de factor lineal.
- Exactitud garantizada 0.5 Cº (a +25 °C).
- Baja impedancia de salida.
- Bajo coste.
- Conveniente para aplicaciones remotas.

1.1.5 PROCESADOR DIGITAL DE SEÑALES (DSP)

El DSP es un procesador monochip diseñado para resolver un conjunto de operaciones matemáticas sobre una señal continua o analógica expresada digitalmente. El diseño de procesadores digitales orientados a soportar los algoritmos matemáticos para el análisis y tratamiento de las señales continuas en tiempo real, son por medio de la utilización de los mismos en sistemas de procesado de señales [6].

Recibe el nombre de DSP un circuito integrado que contiene un procesador digital y un conjunto de recursos complementarios capaces de manejar digitalmente las señales analógicas del mundo real, como los sonidos y las imágenes.

1.1.5.1 Características

a) Los procesadores preferentemente son RISC; con un reducido juego de instrucciones que se ejecutan, generalmente, en un solo ciclo.
b) Utilizan la arquitectura Harvard (Fig. 15) y disponen de dos memorias independientes, una dedicada a contener las instrucciones y otra los datos, posibilitando el acceso simultaneo a ambas informaciones. La memoria de datos suele dividirse en dos espacios independientes que aportan el acceso paralelo.
c) Disponen de recursos físicos complejos para soportar las operaciones específicas de los algoritmos.
d) El repertorio de instrucciones contiene ciertas específicas para resolver los algoritmos matemáticos habituales en el procesamiento de señales.
e) Los modos de direccionamiento son muy sofisticados, ya que localizan los datos y almacenan los resultados de forma óptima para los algoritmos empleados.
f) Disponen de un conjunto de interrupciones muy amplio y veloz, con niveles de prioridad.
g) En el DSP se integran numerosos recursos y periféricos que minimizan el tamaño y simplifican el diseño del sistema.
h) Poseen módulos para el control y optimización del consumo de energía [6].

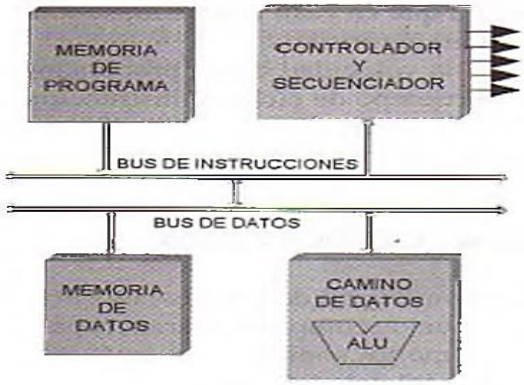


Fig. 15 La arquitectura Harvard dispone de buses independientes para instrucciones y datos.

Actualmente los DSC se comercializan en diferentes dispositivos, agrupados en dos familias: dsPIC30F y dsPIC33F.

Siendo éste proyecto desarrollado con un dispositivo perteneciente a la familia dsPIC30F. En la Tabla 1 se presenta brevemente las características generales de estos microcontroladores [6].

Tabla 1. Características generales del dsPIC30F.

Recurso	Rango de valores
Memoria de programa FLASH	12 Kb – 144 Kb
Memoria de datos RAM	512 Bytes – 8 Kb
Memoria de datos EEPROM	1 Kb – 4 Kb
Patillaje encapsulado	18 – 80 pines
Temporizadores de 16 bits	Hasta 5
Módulo de captura	Hasta 8 entradas
Módulo comparador/PWM	Hasta 8 salidas
Módulo PWM de control de motores	De 6 a 8
Conversor A/D de 10 bits	500 kbps, hasta 16 canales
Conversor A/D de 12 bits	100 kbps, hasta 16 canales
UART	1 – 2
SPI (8-16 bits)	1 – 2
I^2C	1 Módulo
Interfaz CODEC	1
CAN	1 – 2

1.1.5.2 Arquitectura interna del dsPIC30F

- Memoria de programa
- Memoria de datos
- Gestión del sistema y de la energía
- Camino de datos
- Puertas de E/S multifunción
- Periféricos

La memoria de datos RAM (SDRAM) se estructura en dos espacios llamados ‘X’ y ‘Y’, que permiten acceso simultáneo y que pueden alcanzar hasta 8 KB de capacidad en el modelo dsPIC30F, siendo el tamaño de todas sus posiciones de 16 bits. También existe una memoria de datos no volátil de tipo EEPROM. Las instrucciones se alojan en la memoria de programa de tipo FLASH cuyas posiciones tienen un tamaño de 24 bits, igual que el de la mayoría de las instrucciones [6].

El Camino de Datos donde se ejecutan las instrucciones y se procesan los datos se basa en un banco de 16 registros de trabajo (W) de 16 bits de longitud cada uno, que alimentan una ALU típica de MCU, un Motor DSP que sirve para realizar las operaciones DSP de 40 bits y una unidad de división.

Las 7 puertas que agrupan las líneas de E/S son para la comunicación con el exterior (PUERTA A, PUERTA B, PUERTA C, PUERTA D y PUERTA F) se caracterizan por soportar varias funciones multiplexadas. En cuanto a periféricos y recursos auxiliares contenidos en el dsPIC30F4013 abarcan a todos los posibles en la familia: Conversor AD, Módulo de Captura de entrada, Módulo de Comparación de salida, Módulos de comunicación (UART, SPI, I2C y CAN), Osciladores, Perro Guardián, etc. También existen recursos complementarios para el manejo de la energía y la tensión de alimentación.

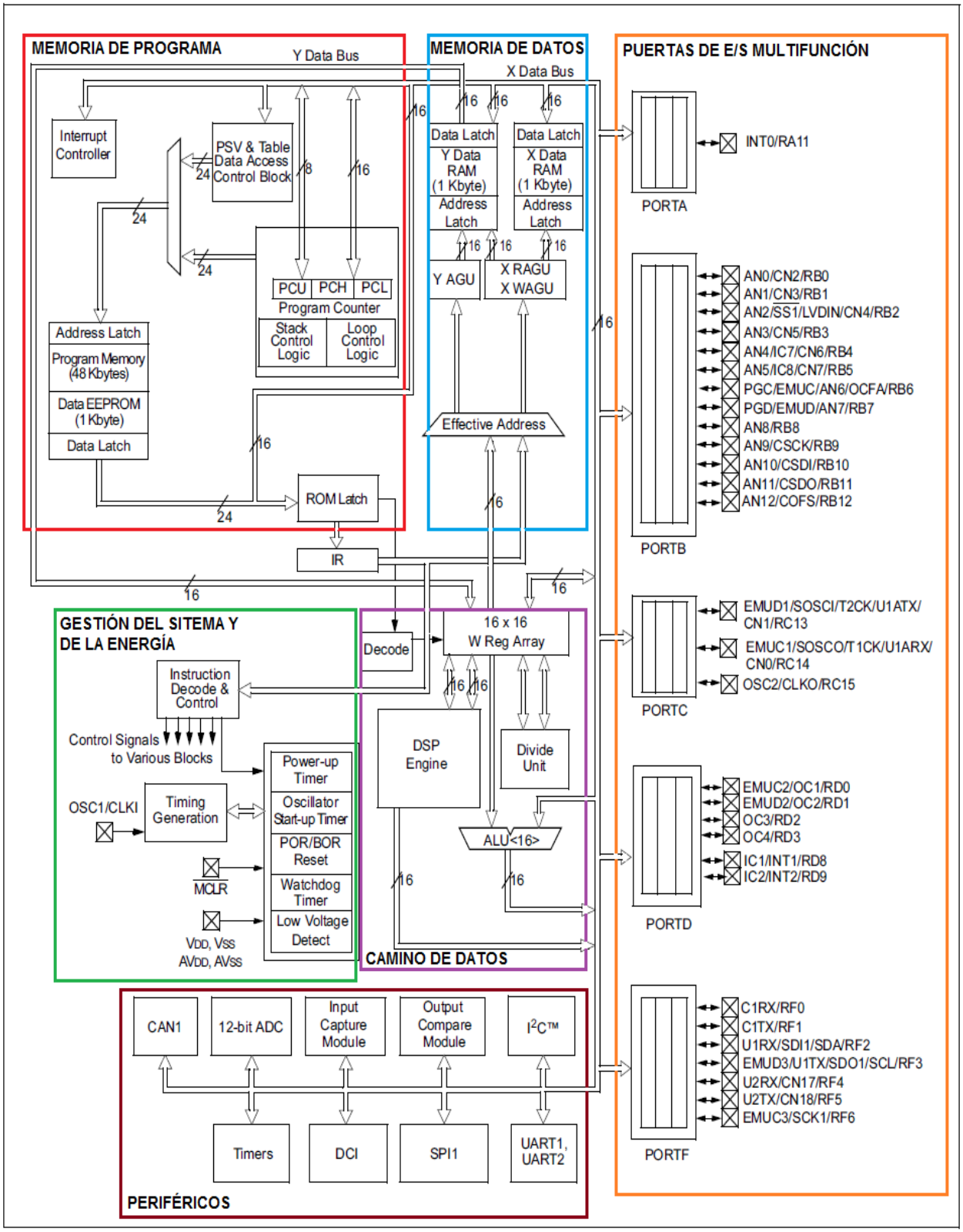


Fig. 16 Diagrama por bloques correspondiente a la arquitectura interna del dsPIC30F4013. Hoja de especificaciones 2004 Microchip Technology Inc.

1.1.6 CONVERSOR ANALÓGICO DIGITAL (ADC)

Tanto las señales analógicas como las digitales son utilizadas para transmitir información, generalmente a través de impulsos eléctricos.

Señal analógica: Una señal analógica (Fig. 17) presenta una variación continua con el tiempo y puede tomar infinitos valores dentro de un rango. Se expanden mediante ondas senoidales y sólo pueden ser leídas por dispositivos capaces de interpretar señales analógicas.

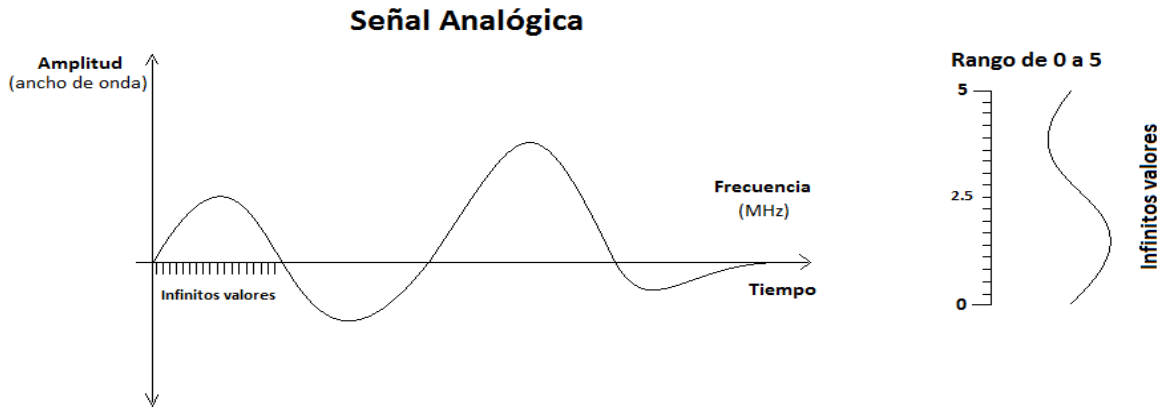


Fig. 17 Señal analógica.

Señal digital: Una señal digital (Fig. 18) tiene variación discontinua con el tiempo. Puede ser discreta y tomar valores en diferentes rangos, o puede tomar dos valores o estados: 0 y 1, conocido como lenguaje máquina. Estos valores son generados por modulación digital. En este caso las ondas no son senoidales, sino cuadradas.

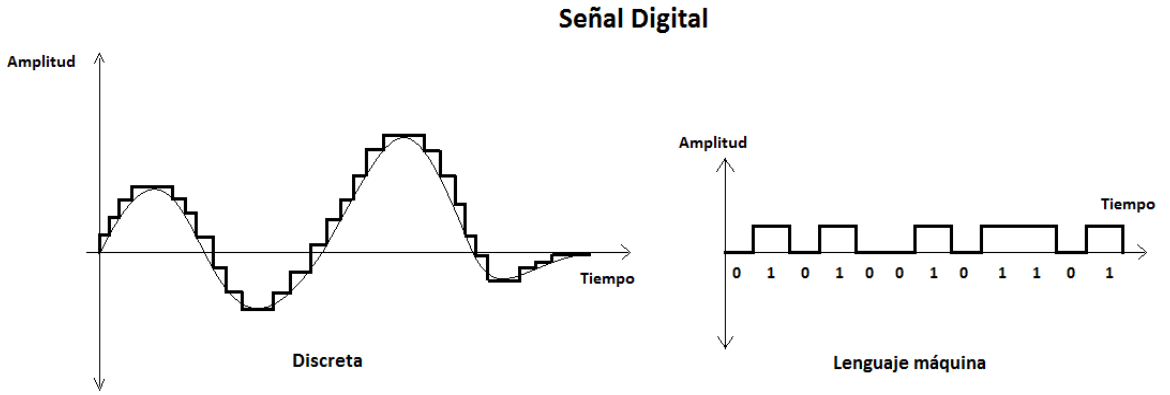


Fig. 18 Señal digital.

En la naturaleza, la mayoría de las señales que se perciben son analógicas, ya que son señales que tienen una variación continua; como el sonido, la energía, la temperatura. Debido a la dualidad onda-partícula de la luz, esta última puede considerarse como una señal analógica (onda) o digital (fotones). Otro ejemplo de señal digital en la naturaleza es la carga eléctrica, debido a que la carga eléctrica de un objeto debe de ser un múltiplo de la carga del electrón, esta es una magnitud discreta.

Todas las señales analógicas reales tienen un ruido que se traduce en un intervalo de incertidumbre. Esto quiere decir que obtenida una muestra de una señal analógica en un instante determinado, es imposible determinar cuál es el valor exacto de la muestra dentro de un intervalo de incertidumbre que introduce el ruido.

Las señales analógicas pueden ser difíciles de manipular, guardar y después recuperar con exactitud. Si esta información analógica se convierte a información digital, se podría manipular y guardar con mayor facilidad.

1.1.6.1 Ventajas de la señal digital.

- Cuando una señal digital es atenuada o experimenta leves perturbaciones, puede ser reconstruida y amplificada mediante sistemas de regeneración de señales.
- Cuenta con sistemas de detección y corrección de errores.
- Facilidad para el procesamiento de la señal. Cualquier operación es realizable a través de cualquier software de edición o procesamiento de señal.
- La señal digital permite la multigeneración infinita sin pérdidas de calidad. (sólo soporta como mucho 4 o 5 generaciones, aplicable sólo a los formatos de disco óptico, la cinta magnética digital)
- Es posible aplicar técnicas de compresión de datos sin pérdidas.

1.1.6.2 Proceso del ADC

La Conversión Analógica-Digital (ADC) consiste en realizar de forma periódica medidas de la amplitud de una señal, redondear sus valores a un conjunto finito de niveles de tensión y registrarlos como números enteros en cualquier tipo de memoria.

El conversor ADC efectúa los siguientes procesos para convertir una señal analógica en digital (Fig.19) [14]:

1.- **Muestreo de la señal analógica (*sampling*).** Toma diferentes muestras de tensiones o voltajes en diferentes puntos de la onda senoidal.

2.- **Cuantización de la propia señal (*quantization*).** Representa los valores de tensiones o voltajes tomados en diferentes puntos de la onda senoidal, que permite medirlos y asignarles sus correspondientes valores en el sistema numérico decimal, antes de convertir esos valores en sistema numérico binario.

3.- **Codificación del resultado de la cuantización, en código binario.** Permite asignarle valores numéricos binarios equivalentes a los valores de tensiones o voltajes que conforman la señal eléctrica analógica original.

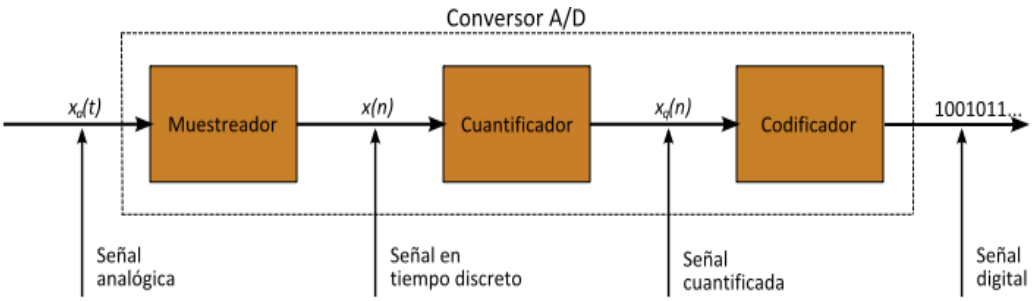


Fig. 19 Conversor ADC.

1.1.6.3 ADC en el dsPIC30F4013.

El dsPIC30F4013 dispone de un conversor analógico digital que permite convertir señales analógicas de entrada en valores digitales de 12 bits con una velocidad de conversión de 10µs. Este módulo está basado en un registro de aproximaciones sucesivas y soporta hasta 16 entradas analógicas multiplexadas sobre un amplificador unipolar de muestreo y retención denominado CH0.

La tensión de referencia puede proceder de la tensión del controlador (AVdd y AVss) o bien de dos pines que pueden recibir dicha tensión de referencia (V_{REF-} y V_{REF+}). Además este conversor puede funcionar mientras el procesador está en modo Sleep con el oscilador RC. Además que el módulo dispone de varios registros para su control:

los registros ADCON1, ADCON2 y ADCON3, controlan el modo de trabajo del conversor analógico digital. El registro ADCHS, por su parte, selecciona los canales de entrada. El registro ADPCFG configura los pines como entradas analógicas o como entradas/salidas digitales. Y el registro ADCSSL selecciona las entradas a escanear o explorar. Los resultados de la conversión se guardan en un buffer de 16 palabras, denominadas ADCBUF0, ADCBUF1,... ADCBUFF [6,15].

En la figura 20 se muestra un esquema del módulo conversor.

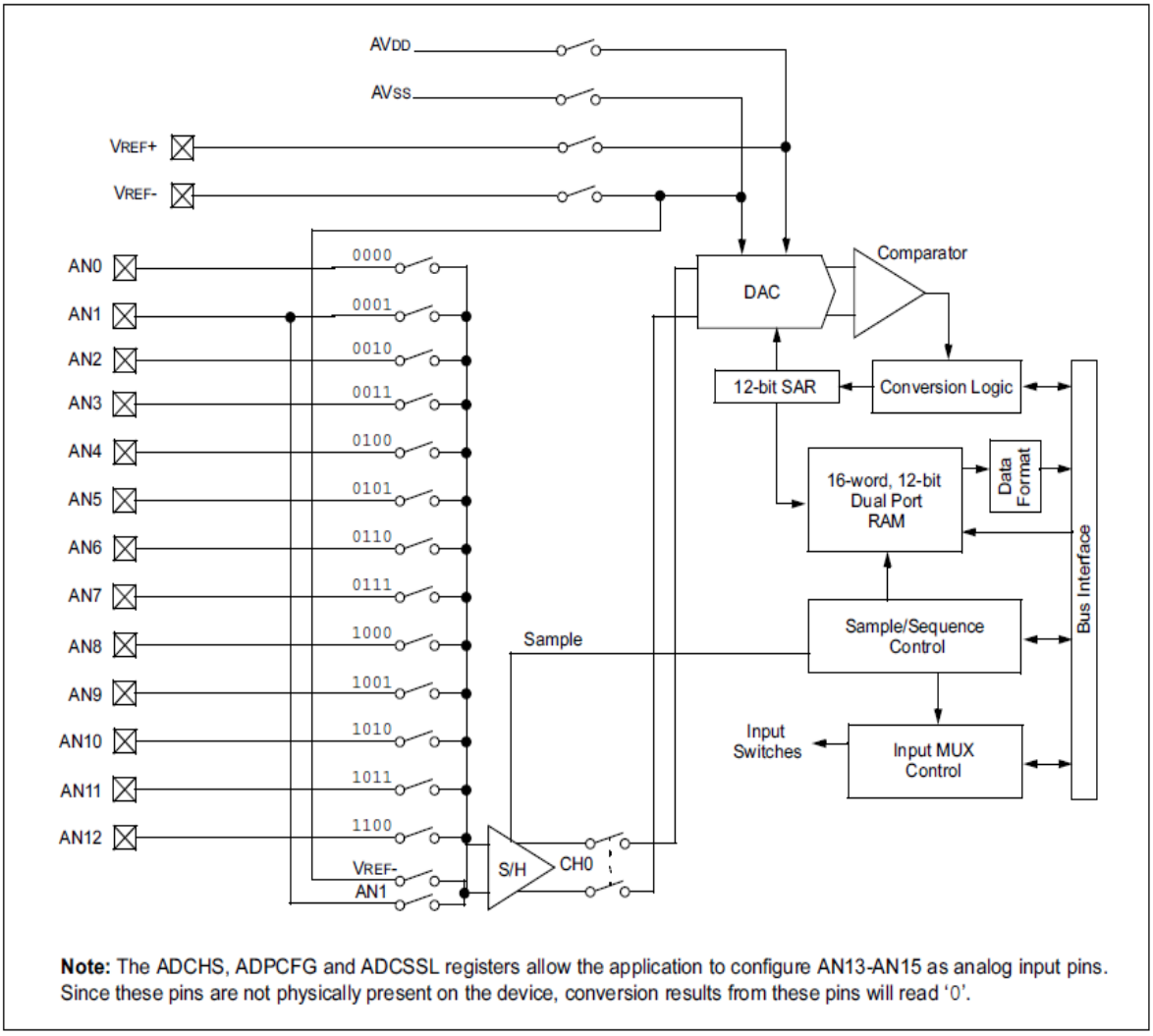


Fig. 20 Diagrama del módulo A/D de 12 bits del dsPIC30F4013. Hoja de especificaciones 2004 Microchip Technology Inc.

CAPÍTULO 1.2 HISTORIA

La empresa que ocupa el primer lugar a nivel mundial en el desarrollo de microcontroladores es Microchip Technology Inc.®. Los PIC (nombre genérico) son sus modelos más populares, los cuales son divididos en familias de 16 bits. Con forme se fueron desarrollando y elevando proyectos hechos con los PIC, los usuarios necesitaban nuevos dispositivos que soportaran funciones de procesamiento digital de señales para atender las nuevas tendencias del mercado orientadas al aumento de la conectividad por Internet, las mejoras relacionadas con la imagen y el sonido, el control de motores, etc. Las nuevas aplicaciones utilizan las funciones típicas de MCU con DSP (procesamiento digital de señales), por lo que Microchip fabricó un circuito hibrido MCU/DSR cuyo manejo es similar a los clásicos microcontroladores pero que incluye las principales prestaciones de los DSR. Siendo así como nació el Controlador Digital de Señales (DSC®), reuniendo las características de un microcontrolador PIC de 16 bits y las de un DSP de gama baja. Así es como surge el comienzo de una nueva era en el mercado de controladores: DSPIC. La primera generación de DSC, conocida como dsPIC30F, pretende facilitar el acercamiento al mundo del procesamiento digital de señales a sus usuarios de MCU de 8 y 16 bits. Posteriormente, la segunda generación, dsPIC33F, incrementa las capacidades, el número de periféricos y el rendimiento, permitiendo acceder a campos más complejos como el procesamiento de imágenes, el sonido, los sensores, entre otros. [6]

1.2.1 Aplicaciones y usos del DSPIC

El área de Procesamiento Digital de Señales (DSP) está muy vigente en el desarrollo tecnológico que vive nuestro país y el mundo entero. Actualmente las ventas del DSPIC en el mercado mundial crecen a un ritmo aproximado al 30% anual, esto es porque se pueden encontrar DSP como dispositivos imprescindibles en algunas aplicaciones concretas. La Tabla 2 menciona algunas de ellas [6].

Tabla 2. Áreas de uso del DSP

Áreas de uso con el DSP	
Medicina	Existencia de aparatos destinados al diagnóstico asistido, las ecografías y la resonancia magnética, que pueden proporcionar información acerca de la fisiología y del flujo de sangre a través de las arterias.
Industria	Innovación en áreas como la exploración petrolera, minera, submarina y espacial.
Control de motores	Uso de los DSP en sistemas como controladores de motores, inversores de potencia, controladores de posición, impresoras y fotocopiadoras, compresores de alta potencia, etc.
Automoción	Incremento de las prestaciones de los automóviles. Por ejemplo, utiliza un DSP en su sistema de rastreo de automóviles aprovechando la red global de satélites GPS para establecer su ubicación precisa.
Militar	Utilizados en el sonar, el radar, el piloto automático y el guiado automático de misiles.
Telecomunicaciones	En un teléfono móvil el DSP emplea complejos algoritmos matemáticos que realzan la diferencia entre el ruido de fondo y la voz del usuario.
Imagen y sonido	Posibilidad de añadir ecos, el reconocimiento de patrones, la compresión/descompresión de imágenes, el reconocimiento y la generación de audio, la cancelación de ruido, la cancelación de eco, el encriptado y la síntesis de voz.

Una de las posibles aplicaciones del sistema es la identificación de factores del medio ambiente e incluso fuentes de radiación, a través de la caracterización de los pulsos eléctricos obtenidos a partir de detectores de radiación.

Con el DSP es posible identificar el rango de operación en aplicaciones analógicas y digitales. De forma autónoma el sistema puede identificar el tipo de señal, la amplitud,

el periodo, la frecuencia y el voltaje de offset de la señal de entrada. A partir de la información anterior el sistema realizará el modelado de las señales y permitirá su comparación con la señal de entrada. En el caso de que la señal sea un tren de pulsos, el sistema puede almacenar los parámetros de cada uno de los pulsos [6].

Con base en las capacidades y aplicaciones del DSPIC anteriores, se considera que el uso de este dispositivo tiene un gran potencial de desarrollo para el procesamiento matemático de estas y otras variables ambientales de interés.

El desarrollo obtenido con el DSPIC fortalece los conocimientos adquiridos de las materias de electrónica impartidas durante la carrera de Ingeniería en Computación. Ya que su arquitectura interna, su programación y modo de conexión resultan similares a los PIC que fueron impartidos en la carrera.

CAPÍTULO 1.3 PROGRAMAS DE APLICACIÓN

En esta sección se presentarán distintos ejemplos de programas desarrollados con la finalidad de conocer las características y comportamiento del dsPIC30F4013. Es importante resaltar que aunque pueden ocuparse diferentes plataformas o aplicaciones, en el presente trabajo se utilizará MPLAB para el desarrollo y prueba de los programas.

La orientación de conectar el programador con el DSPIC para que éste sea reconocido de manera correcta, se encuentran en la sección de Anexos.

Ejemplo 1.

Los semáforos son señales de control de tráfico que se sitúan en intersecciones viales y otros lugares para regular el tráfico vehicular, y por ende, el tránsito peatonal. A continuación se describirán algunos ejemplos en las que se consideran este tipo de aplicaciones.

Ejemplo 1.1

Objetivo.

Diseñar y desarrollar un programa utilizando el dsPIC30F4013 que emule el funcionamiento de los semáforos del crucero vial que se muestra en la Fig. 21.

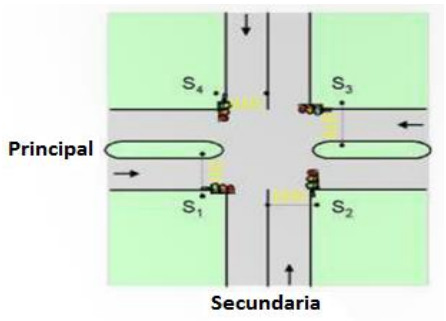


Fig. 21 Crucero vial.

Desarrollo.

En este crucero vial se tienen cuatro semáforos, dos para la avenida principal y dos para la avenida secundaria. Ambas avenidas son de doble sentido, las vueltas son prohibidas por lo que el recorrido solo puede ser recto.

Las siguientes condiciones describen el funcionamiento de los semáforos:

Mientras los semáforos S1 y S3 de la avenida principal, se encuentran en “siga” (verde), los semáforos S2 y S4 de la avenida secundaria, se encontrarán en “alto” (rojo). Los semáforos S2 y S4 se mantienen en rojo aun cuando los semáforos S1 y S3 cambien a “preventivo” (amarillo). Cuando cambien a “alto” los semáforos S1 y S3, los semáforos S2 y S4 estarán en “siga”, manteniéndose así mientras S1 y S3 pasen a “preventivo”. Finalmente se repite el ciclo indefinidamente.

Asumiendo que el flujo vehicular de la avenida principal es mayor que el de la avenida secundaria, se establece que el tiempo en verde de los semáforos S1 y S3 (de la avenida principal) debe ser mayor al tiempo en verde de los semáforos S2 y S4 (de la avenida secundaria).

Del análisis del funcionamiento de los semáforos de un crucero se obtiene la Tabla 3:

Tabla 3. Comportamiento de los semáforos. Análisis de las condiciones de cada semáforo.

S1	**S2**	**S3**	**S4**	**Tiempo**
Verde	Rojo	Verde	Rojo	8 seg
Amarillo	Rojo	Amarillo	Rojo	8 seg
Rojo	Verde	Rojo	Verde	4 seg
Rojo	Amarillo	Rojo	Amarillo	4 seg
Se repite el ciclo…				

Diagrama de flujo.

El diagrama de flujo es una representación gráfica de un proceso. Ofrece una descripción visual de las actividades implicadas en un proceso, mostrando la relación secuencial ente ellas.

En la Fig. 22 se presenta el diagrama de flujo del programa desarrollado para el ejemplo 1.1. Como se puede observar al inicio del diagrama de flujo se configura el programa, mediante el llamado de las librerías y los encabezados. Una de las librerías a utilizar en este programa es la función Delay, a través de la cual se definirá el tiempo de retardo. Posteriormente se habilita el puerto B para conectar cada uno de los LED que emularan el comportamiento del semáforo. En el siguiente paso se definen los bits de salida para cada semáforo según las condiciones descritas en la Tabla 1, agregando el tiempo para cada condición. El tiempo de duración de cada una de las condiciones es implementado mediante la función Delay. El diagrama de flujo de la función se muestra en la Fig. 23. Como se puede observar la función DELAY trabaja a una frecuencia de oscilación de 4MHz porque todo dispositivo Microchip (PIC, DSPIC) se ejecuta en 4 instrucciones de reloj. Esta función es definida en un archivo aparte, ser llamada las veces necesarias, como si fuera una librería.

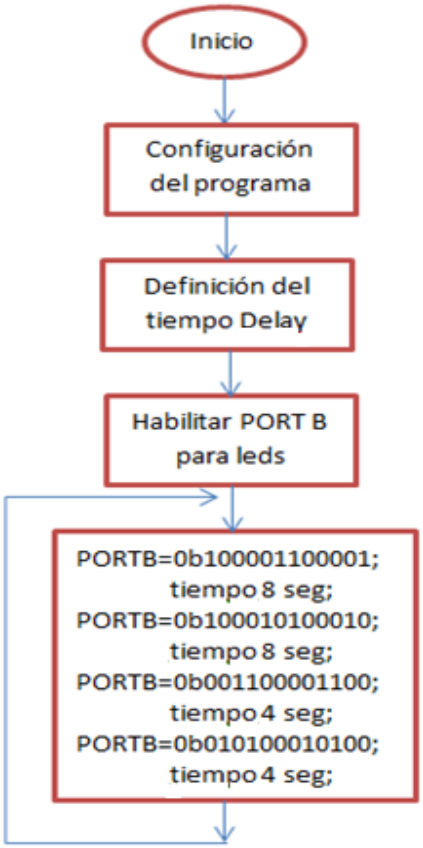


Fig. 22 Diagrama de Flujo del Ejemplo 1.1

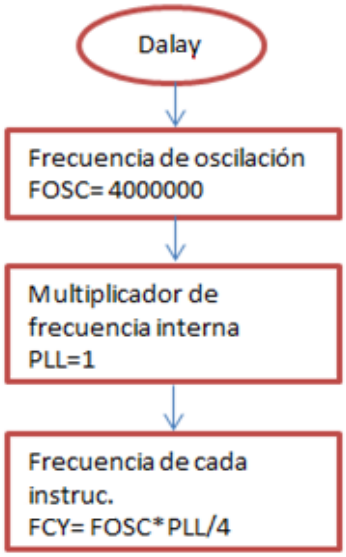


Fig. 23 Diagrama de flujo de función Delay

Código.

A continuación se presenta el código del programa del ejemplo 1.1, donde el color azul representa palabras reservadas, identificadores o tipos de datos. El verde indica que son comentarios que definen el uso de cada sentencia. Y el color negro son los bloques de código.

```
#include "p30f4013.h" //incluye la cabecera del dispositivo a usar
#include "libpic30.h" //librería del pic30 para procesamiento de señales
#include "common.h"; //incluir el nombre del archivo cabecera
//Configuración
_FOSC (CSW_FSCM_OFF & XT);
_FWDT (WDT_OFF);
_FBORPOR(PBOR_OFF & PWRT_16 & MCLR_EN);
_FGS(CODE_PROT_OFF);

#ifndef __DELAY_H
      #define delay_us(x) __delay32(((x*FCY)/100000L)) //delay x us
      #define delay_ms(x) __delay32(((x*FCY)/100L)) //delay x ms
      #define __DELAY_H 1
#endif
```

```
//Función principal
int main()
{
	ADPCFG=0XFFFF; // Todas las entradas del puerto B son digitales.
	TRISB=0;
	PORTB=0;
	while(1) //iniciamos ciclo infinito
	/*b0=verde; b1=amarillo; b2=rojo; b3=verde; b4=amarillo; b5=rojo
	b6=verde; b7=amarillo; b8=rojo; b9=verde; b10=amarillo
	b11=rojo
	Se da la condición de cada puerto
	*/
	{
		//S1
		PORTB=0b100001100001;
		delay_ms(800);

		//S3
		PORTB=0b100010100010;
		delay_ms(800);

		//S2
		PORTB=0b001100001100;
		delay_ms(400);

		//S4
		PORTB=0b010100010100;
		delay_ms(400);
	}
}
```

La simulación se llevó a cabo en MPLAB, donde se observa que el resultado corresponde a lo planteado con anterioridad.

Diagrama circuital.

En la Fig. 24 se muestra el diagrama del circuito a implemntar. En el se puede observar la conexión del dsPIC30F4013 donde el pin 1 está conectado a una resistencia de 10kΩ que va directamente al voltaje. Del pin 2 al 10 son conectados los semáforos S1, S2, S3. Y el semáforo S4 se encuentra conectado en los pines 36,37 y 38.

El oscilador de cristal con sus respetivos capacitores se posicionan en los pines 13 y 14. Además que es importante mantener alimentado todo el DSPIC para su correcto funcionamiento.

Nota: Para los siguientes ejemplos se utiliza el mismo diseño del circuito. Solo cambia el programa.

Diagrama circuital.

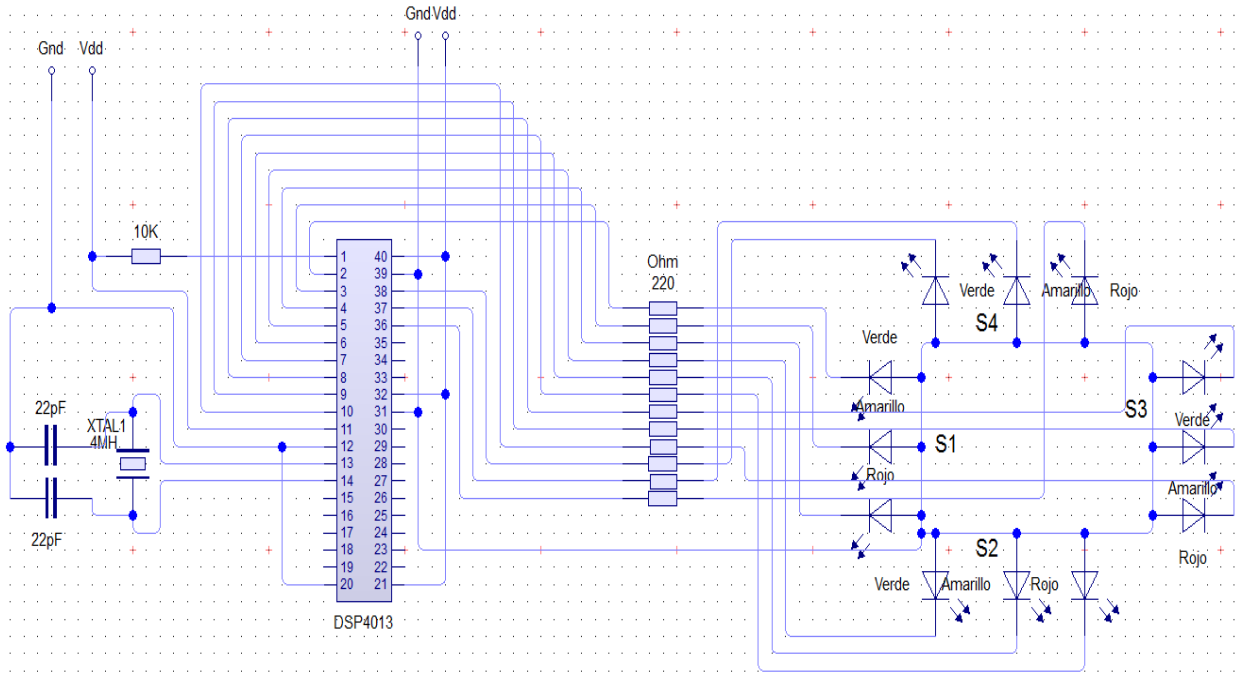


Fig. 24 Diagrama circuital.

Ensamblaje y prueba.

Finalmente se demostró que el programa junto con el diseño del circuito, funciona correctamente.

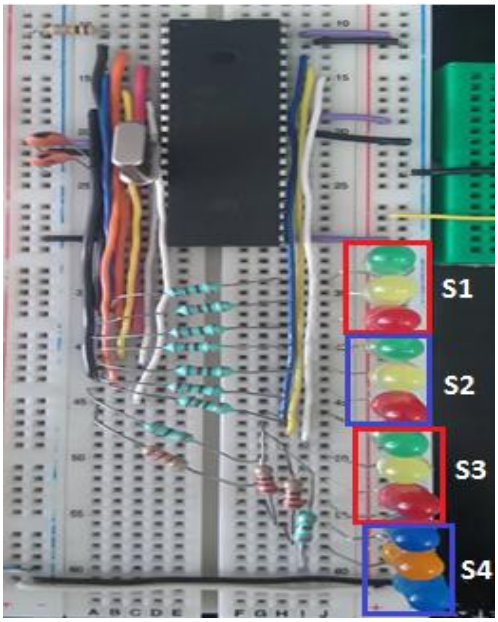


Fig. 25 Ensamblado en el protoboard

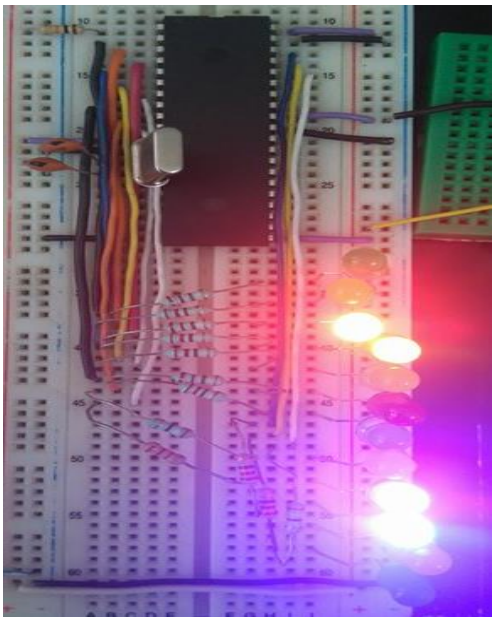


Fig. 26 Prueba del circuito.

Ejemplo 1.2.

Los semáforos tienen distintas funciones dependiendo del horario en el que se encuentren. Se le llama diurno cuando el semáforo realiza su funcionamiento cotidiano, y nocturno cuando el semáforo solo parpadea la señal preventiva.

Objetivo.

Mejorar el programa anterior del crucero vial, agregándole una entrada de control a los semáforos para que funcionen de manera diurno/nocturno.

Desarrollo.

Añadir un control de función que permita dividir el trabajo en dos. Con 1 se activa la función diurno: el semáforo trabaja de forma normal, como ya se había programado en el ejemplo 1.1. Con 0, se activa el modo nocturno: los semáforos se pondrán en forma preventiva, es decir, solo los LED amarillos prenderán y apagaran intermitentemente.

El análisis de lo anterior se reduce en la Tabla 4.

Tabla 4. Control de los semáforos

Control	**Función**
1	Diurno
0	Nocturno

Diagrama de flujo.

En la Fig. 27 se presenta el diagrama de flujo del programa desarrollado para el ejemplo 1.2. Al igual que el primer ejemplo, se observa al inicio del diagrama de flujo que se configura el programa, mediante el llamado de las librerías y los encabezados. También se utiliza la función Delay, a través de la cual se definirá el tiempo de retardo. Posteriormente se habilita el puerto B para conectar cada uno de los LED que emularan el comportamiento del semáforo y se habilita el puerto D para la entrada de datos.

En el siguiente paso se configura la condición que indica si la función es diurna o nocturna. Si el puerto D0 es igual a 1, se activa el funcionamiento diurno. Si no, se activa el funcionamiento nocturno.

Por cada función se definen los bits de salida para cada semáforo según las condiciones descritas en la Tabla 4, agregando el tiempo para cada condición.

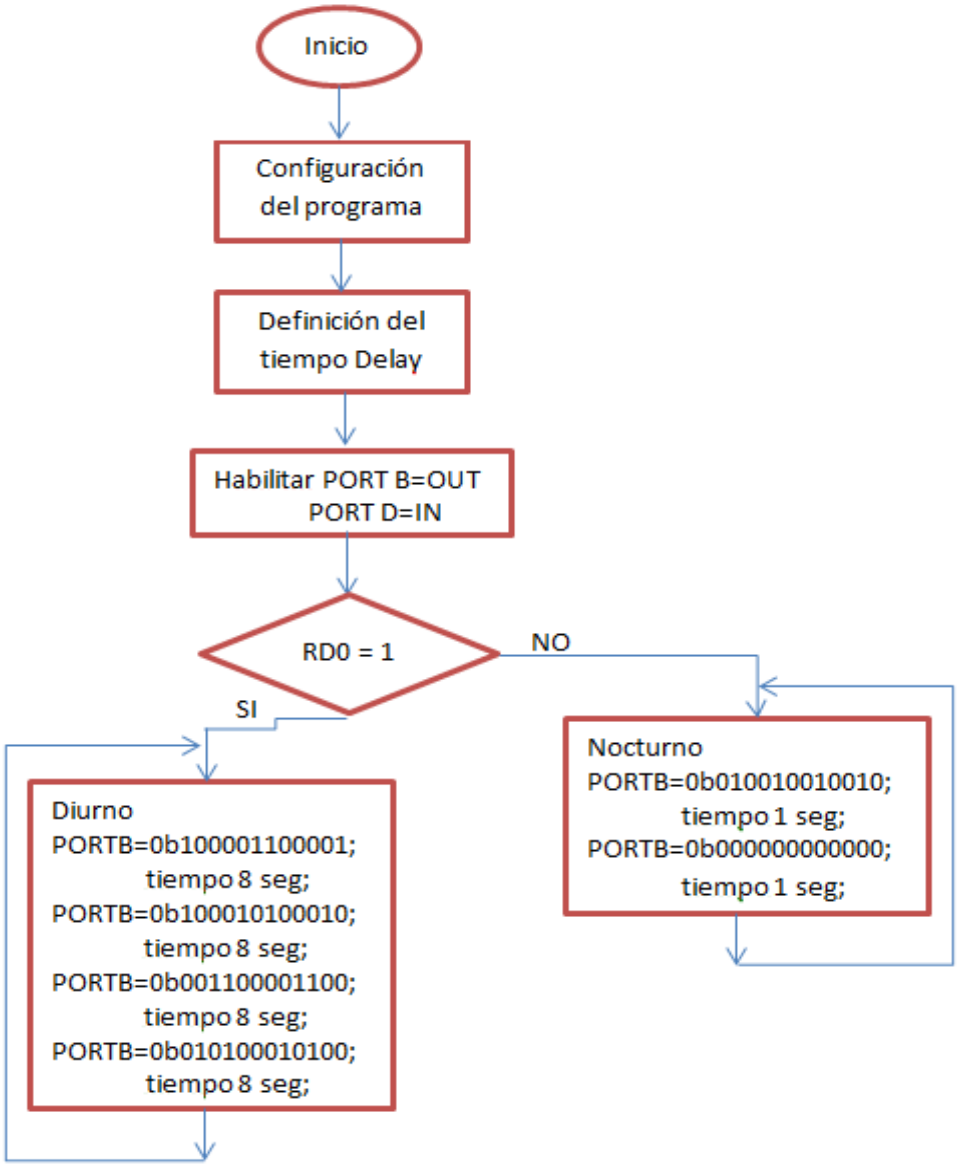


Fig. 27 Diagrama de flujo del Ejemplo 1.2

Código.

A continuación se presenta el código del programa del ejemplo 1.2, donde los colores indican lo mismo que en el ejemplo 1.1.

La condición requerida por el programa y definida en el diagrama de flujo se representa con el *if* y *else* del código.

```
#include "p30f4013.h" //incluye la cabecera del dispositivo a usar
#include "libpic30.h" //librería del pic30 para procesamiento de señales
#include "common.h"; //incluir el nombre del archivo cabecera
//Configuración
_FOSC (CSW_FSCM_OFF & XT);
_FWDT (WDT_OFF);
_FBORPOR(PBOR_OFF & PWRT_16 & MCLR_EN);
_FGS(CODE_PROT_OFF);
#ifndef __DELAY_H
      #define delay_us(x) __delay32(((x*FCY)/100000L)) //delay x us
      #define delay_ms(x) __delay32(((x*FCY)/100L)) //delay x ms
      #define __DELAY_H 1
#endif
//Función principal
int main()
{
      ADPCFG=0XFFFF; // Todas las entradas del puerto B son digitales.
      TRISB=0;
      PORTD=0;
      /*b0=verde; b1=amarillo; b2=rojo; b3=verde; b4=amarillo; b5=rojo
      b6=verde; b7=amarillo; b8=rojo; b9=verde; b10=amarillo
      b11=rojo
      Se da la condición de cada puerto
      */
      if(   PORTDbits.RD0 == 1){
            //Diurno
            PORTB=0b100001100001;
            delay_ms(800);

            PORTB=0b100010100010;
            delay_ms(800);

            PORTB=0b001100001100;
            delay_ms(400);

            PORTB=0b010100010100;
            delay_ms(400);
      }
      else{
            //Nocturno
            PORTB=0b010010010010;
            delay_ms(100);
            PORTB=0b000000000000;
            delay_ms(100);
      }
}
```

Se realizó la isimulación MPLAB y se observa que el resultado corresponde a lo planteado con anterioridad. Solo resta probarlo diseño del circuito anterior.

Ejemplo 1.3

Los semáforos, como otra función, son capaces de cambiar el tiempo de espera o de siga dependiendo del tránsito que se encuentre a determinada hora del día.

Entre más carga de tránsito tenga una avenida, más rápidos deben ser los cambios del semáforo. A este horario se le conoce como *hora pico* (HP).

Entre menor carga de tránsito tenga una avenida, más lentos deben ser los cambios del semáforo. A este horario se le conoce como *hora valle* (HV).

Objetivo.

Agregar un control de tiempo para hora pico y hora valle del tránsito, conservando el control anterior de Diurno/Nocturno.

Desarrollo.

Los controles de cada condición son independientes. Cuando cualquiera de los controles se encuentre en estado de 0, significa que la función que debe realizar estará apagada. En cambio, si están en estado de 1, se realizará su respectiva función programada. El análisis de lo anterior se reduce en la Tabla 5.

Tabla 5 Condición de control de los semáforos.

Estado	**Función**	**Estado**	**Función**	**Tiempo**
0	Apagado	1	Diurno	8 seg
0	Apagado	1	Nocturno	8 seg
0	Apagado	1	HP	5 seg
0	Apagado	1	HV	10 seg

Para las funciones Diurno y Nocturno, se utilizan las mismas características que en el ejemplo anterior. Para las funciones de hora pico (HP) y hora valle (HV) se modificaran sus tiempos de espera para cada semáforo, dependiendo del respectivo horario.

Diagrama de flujo.

En la Fig. 28 se presenta el diagrama de flujo del programa desarrollado para el ejemplo 1.3. Donde se agregan los encabezados y librerías necesarias. Posteriormente se habilita el puerto B para conectar cada uno de los LED que emularan el comportamiento del semáforo y se habilita el puerto D para la entrada de datos.

En el siguiente paso se configuran las condiciones que indican si la función es diurna, nocturna, hora pico y hora valle. Si el puerto D0 es igual a 1, se activa el funcionamiento diurno. Si no, se detiene el proceso. Si el puerto D1 es igual a 1, se activa el funcionamiento nocturno, de lo contrario se detiene el proceso. Si el puerto D2 es igual a 1, se activa el funcionamiento hora pico, de lo contrario se detiene el proceso. Si el puerto D3 es igual a 1, se activa el funcionamiento hora valle. Si no, se detiene el proceso.

Por cada función se definen los bits de salida para cada semáforo según las condiciones descritas en la Tabla 3, agregando el tiempo para cada condición. El tiempo de duración de cada una de las condiciones es implementado mediante la función Delay.

Nota: Las salidas del puerto B fueron cambiadas de binario a hexadecimal para espacio en el diagrama de flujo.

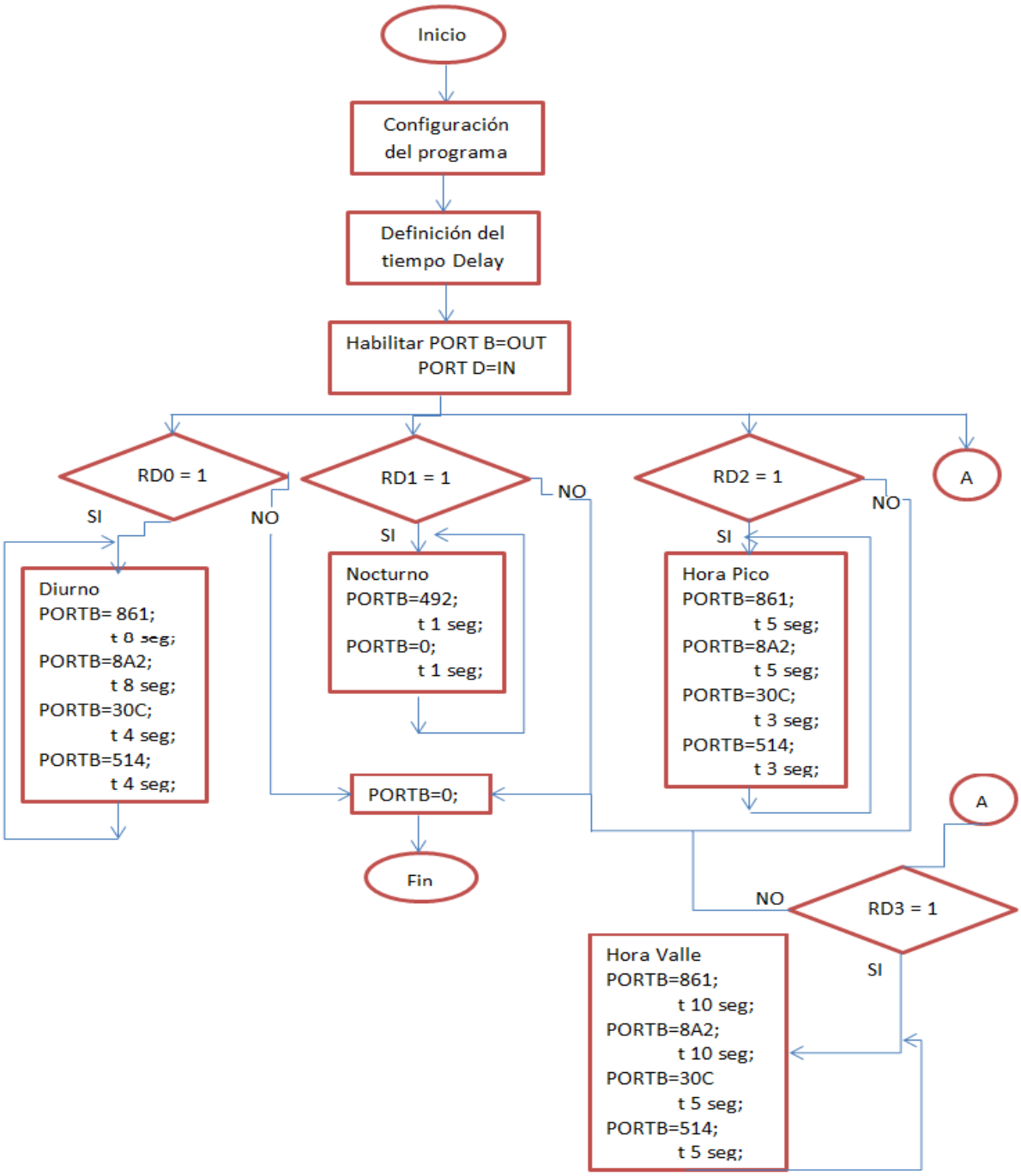


Fig. 28 Diagrama de flujo del Ejercicio 1.3

Código.

A continuación se presenta el código del programa del ejemplo 1.3, donde los colores indican lo mismo que en el ejemplo 1.1.

Las condiciones requeridas por el programa y definidas en el diagrama de flujo se representan con el *if* y *else* del código.

```
#include "p30f4013.h" //incluye la cabecera del dispositivo a usar
#include "libpic30.h" //librería del pic30 para procesamiento de señales
#include "common.h"; //incluir el nombre del archivo cabecera
//Configuración
_FOSC (CSW_FSCM_OFF & XT);
_FWDT (WDT_OFF);
_FBORPOR(PBOR_OFF & PWRT_16 & MCLR_EN);
_FGS(CODE_PROT_OFF);

#ifndef __DELAY_H
      #define delay_us(x) __delay32(((x*FCY)/100000L)) //delay x us
      #define delay_ms(x) __delay32(((x*FCY)/100L)) //delay x ms
      #define __DELAY_H 1
#endif

//Función principal
int main()
{
      ADPCFG=0XFFFF; // Todas las entradas del puerto B son digitales.
      TRISB=0;
      PORTD=0;

      if(   PORTDbits.RD0 == 1){
            //Diurno
            PORTB=0b100001100001;
            delay_ms(800);

            PORTB=0b100010100010;
            delay_ms(800);

            PORTB=0b001100001100;
            delay_ms(400);

            PORTB=0b010100010100;
            delay_ms(400);
      }
      else{
            PORTB=0b000000000000;
      }

      if(   PORTDbits.RD1 == 1){
            //Nocturno
            PORTB=0b010010010010;
            delay_ms(100);
            PORTB=0b000000000000;
            delay_ms(100);
      }
      else{
            PORTB=0b000000000000;
      }
      if(   PORTDbits.RD2 == 1){
```

```
            //HP
            PORTB=0b100001100001;
            delay_ms(500);
            PORTB=0b100010100010;
            delay_ms(500);

            PORTB=0b001100001100;
            delay_ms(300);

            PORTB=0b010100010100;
            delay_ms(300);
      }
      else{
            PORTB=0b000000000000;
      }
      if(   PORTDbits.RD3 == 1){
            //HV
            PORTB=0b100001100001;
            delay_ms(1000);

            PORTB=0b100010100010;
            delay_ms(1000);

            PORTB=0b001100001100;
            delay_ms(500);

            PORTB=0b010100010100;
            delay_ms(500);
      }
      else{
            PORTB=0b000000000000;
      }
}
```

Se realizó la simulación MPLAB y se observa que el resultado corresponde a lo planteado con anterioridad. Solo resta probarlo diseño del circuito anterior.

Ejemplo 1.4

Con el ejemplo anterior cada control de función puede ser realizado correctamente si solo se ocupa un control a la vez, pero ¿sería posible si queremos ocupar solo dos controles para las mismas cuatro funciones?

Objetivo.

Diseñar y desarrollar un programa utilizando el dsPIC30F4013, que sea capaz de realizar las mismas cuatro funciones del ejemplo 1.3 pero solo utilizando 2 controles de entrada.

Desarrollo.

Un control tendrá las funciones Nocturno/Diurno, y con el otro control, las funciones HV/HP. Cuando sean ceros los controles, la función que ejecute será Nocturno, cuando el primer control tenga un 1 la función a realizar será Diurno. Si es el segundo control el que se active y el primero permanezca en cero, la función que ejecute será Hora Pico. Cuando se encuentren en 1 ambos controles, la función que se active será Hora Valle.

Este análisis se puede apreciar en la Tabla 6.

Tabla 6. Condición de control

Control	**Función**
00	Nocturno
01	Diurno
10	HP
11	HV

Diagrama de flujo.

En la Fig. 29 se presenta el diagrama de flujo del programa desarrollado para el ejemplo 1.4. Donde se agregan los encabezados y librerías necesarias. Posteriormente se habilita el puerto B para conectar cada uno de los LED que emularan el comportamiento del semáforo y se habilita el puerto D para la entrada de datos.

En el siguiente paso se configura las condiciones que indican si la función es diurna, nocturna, hora pico y hora valle. Las condiciones son dependientes de una condición principal.

Si el puerto D0 es igual a 0 y el puerto D1 es igual a 0, se activa la función Nocturno. Si el puerto D0 es igual a 0 y el puerto D1 es igual a 1, se activa la función Diurno. Si por el contrario, el puerto D0 es igual a 1 y el puerto D1 es igual a 0, se activa la función HV. Y finalmente, si el puerto D0 es igual a 1 y el puerto D1 es igual a 1, se activa la función HP.

Por cada función se definen los bits de salida para cada semáforo según las condiciones descritas en la Tabla 3, agregando el tiempo para cada condición. El tiempo de duración de cada una de las condiciones es implementado mediante la función Delay.

Nota: Las salidas del puerto B fueron cambiadas de binario a hexadecimal para espacio en el diagrama de flujo.

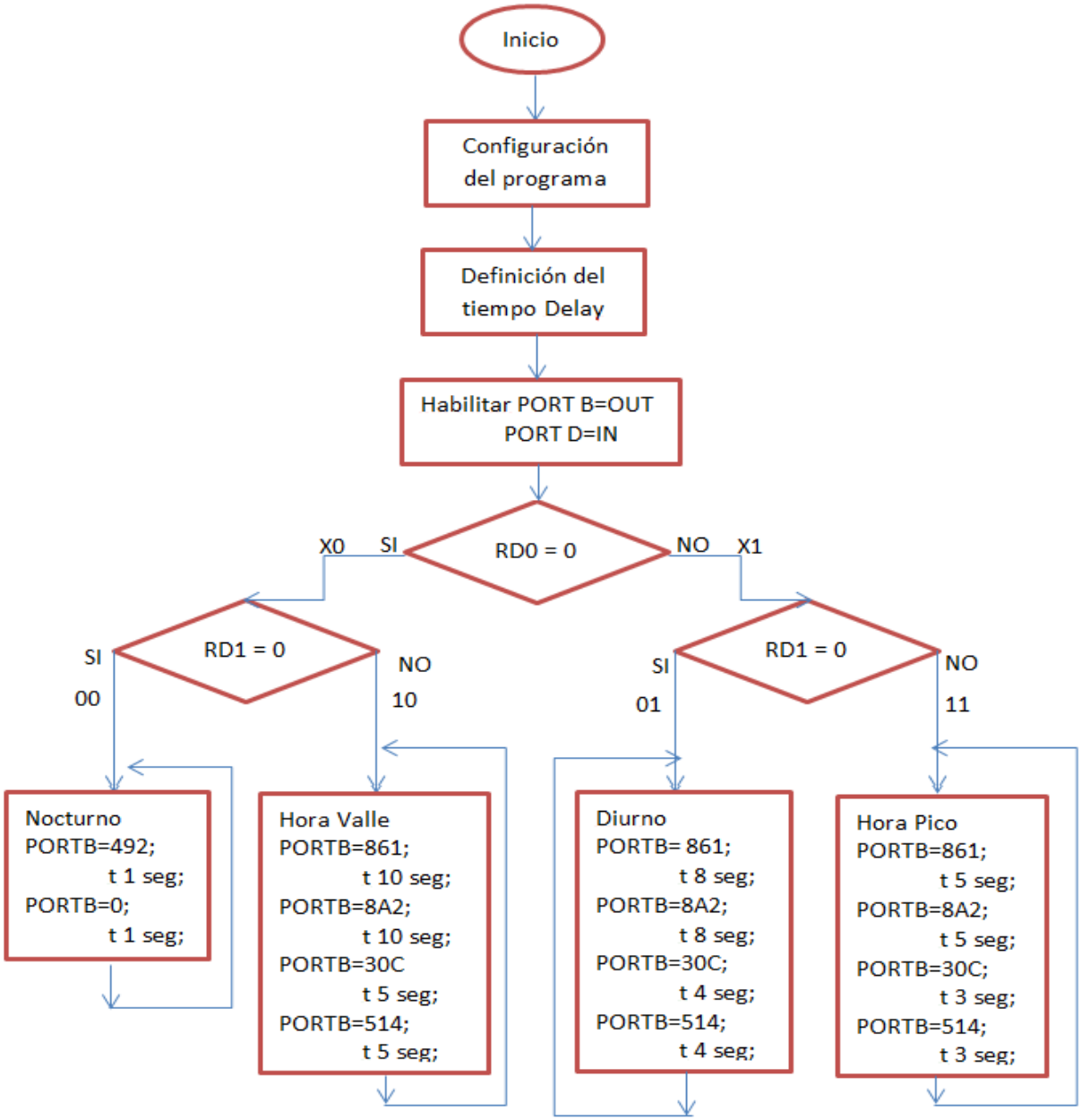


Fig. 29 Diagrama de flujo del Ejercicio 1.4

Código.

A continuación se presenta el código del programa del ejemplo 1.4, donde los colores indican lo mismo que en el ejemplo 1.1.

Las condiciones requeridas por el programa y definidas en el diagrama de flujo se representan con el *if* y *else* anidados del código.

```
#include "p30f4013.h" //incluye la cabecera del dispositivo a usar
#include "libpic30.h" //librería del pic30 para procesamiento de señales
#include "common.h"; //incluir el nombre del archivo cabecera
//Configuración
_FOSC (CSW_FSCM_OFF & XT);
_FWDT (WDT_OFF);
_FBORPOR(PBOR_OFF & PWRT_16 & MCLR_EN);
_FGS(CODE_PROT_OFF);

#ifndef __DELAY_H
      #define delay_us(x) __delay32(((x*FCY)/100000L)) //delay x us
      #define delay_ms(x) __delay32(((x*FCY)/100L)) //delay x ms
      #define __DELAY_H 1
#endif

//Función principal
int main()
{
      ADPCFG=0XFFFF; // Todas las entradas del puerto B son digitales.
      TRISB=0;
      PORTD=0;

      if(   PORTDbits.RD0 == 0){

            if(   PORTDbits.RD1 == 0){
                  //Nocturno
                  PORTB=0b010010010010;
                  delay_ms(100);
                  PORTB=0b000000000000;
                  delay_ms(100);
            }
            else{
                  //Hora Valle
                  PORTB=0b100001100001;
                  delay_ms(1000);

                  PORTB=0b100010100010;
                  delay_ms(1000);

                  PORTB=0b001100001100;
                  delay_ms(500);

                  PORTB=0b010100010100;
                  delay_ms(500);
            }
      }
      else{
            if(   PORTDbits.RD1 == 0){
                  //Diurno
                  PORTB=0b100001100001;
                  delay_ms(800);
```

```
                PORTB=0b100010100010;
                delay_ms(800);

                PORTB=0b001100001100;
                delay_ms(400);

                PORTB=0b010100010100;
                delay_ms(400);
            }
            else{
                //Hora Pico
                PORTB=0b100001100001;
                delay_ms(500);
                PORTB=0b100010100010;
                delay_ms(500);

                PORTB=0b001100001100;
                delay_ms(300);

                PORTB=0b010100010100;
                delay_ms(300);
            }
        }

}
```

La simulación se llevó a cabo en MPLAB y se observa que el resultado corresponde a lo planteado con anterioridad. Solo resta probarlo en el mismo circuito ya diseñando e implementado.

Resultados

Por último se realizó una maqueta (Fig. 30 y 31) para visualizar con mayor precisión el comportamiento de las distintas funciones de los semáforos.

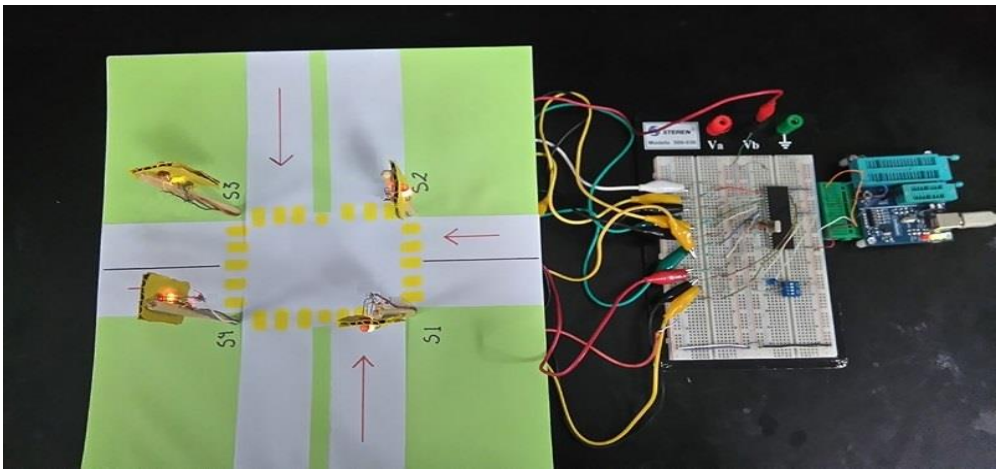


Fig. 30 Maqueta de semáforos, vista superior.

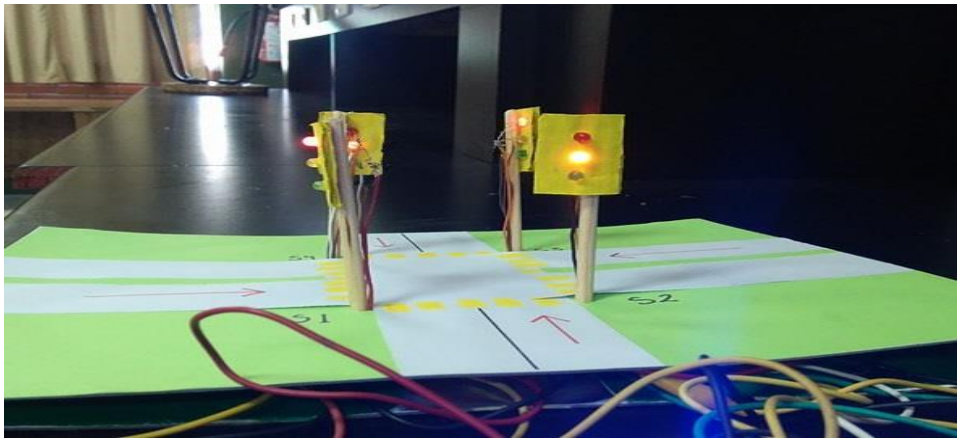


Fig. 31 Maqueta de semáforos, vista lateral.

Ejemplo 2.

La pantalla para desplegar mensajes de texto de LCD (Liquid Cristal Display) tiene la capacidad de mostrar cualquier carácter alfanumérico, lo que permite representar la información que genera cualquier equipo electrónico de una forma fácil y económica. La pantalla consta de una matriz de caracteres de 16x2, es posible desplegar 16 caracteres alfanuméricos en dos líneas de la pantalla.

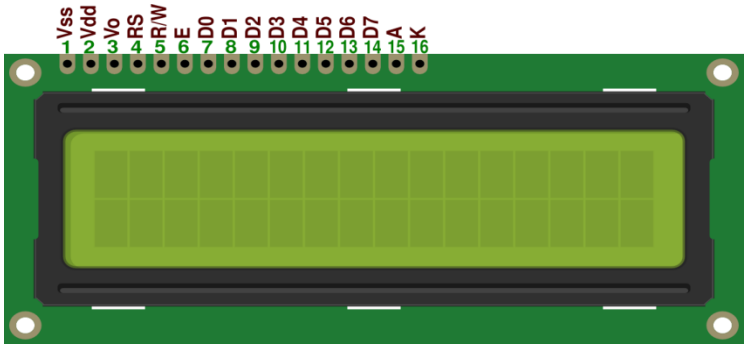


Fig. 32 LCD 2x16: compuesto por 2 líneas de 16 caracteres.

Patillaje

En la Tabla 7 se presenta la descripción de señales empleadas por el módulo LCD así como el número de patilla a la que corresponden.

Tabla 7. Descripción del patillaje del LCD 16x2.

No. Pin	Símbolo	Descripción
1	V_{SS}	Tierra de alimentación (GND).
2	V_{DD}	Alimentación de +5v.
3	V_o	Contraste del LCD. Normalmente se conecta a un potenciómetro a través del cual se aplica una tensión variable entre 0 y +5V que permite regular el contraste.
4	RS	Selección del registro de control/registro de datos RS=0 Selección del registro de control RS=1 Selección del registro de datos
5	R/W	Señal de lectura/escritura R/W=0 El módulo LCD es escrito R/W=1 El módulo LCD es leído
6	E	Señal de activación del módulo LCD E=0 Módulo desconectado E=1 Módulo conectado
7-14	D0-D7	Bus de datos bi-direccional. Transferencia de los datos.
15	A	Ánodo - iluminación del fondo (+5v – 330 Ω).
16	K	Cátodo - iluminación del fondo (GND).

A continuación se describirán algunos ejemplos en los que se utiliza este dispositivo con diferentes aplicaciones.

Ejemplo 2.1

Objetivo

Diseñar y desarrollar un programa utilizando el dsPIC30F4013 que despliegue el siguiente mensaje: primera línea: UAEM CU ECATEPEC, segunda línea: Naibi Altamirano.

Desarrollo.

Para mostrar cualquier mensaje en un LCD es necesario contar con las librerías lcd.h y lcd.c. Estas librerías contienen la configuración de la conexión del LCD con el dsPIC30F4013. Adicionalmente en estas librerías se establece la programación para inicializar el LCD, desplegar un carácter o una cadena de caracteres, la posición de escritura del LCD, y limpiar la pantalla del LCD.

Existen diferentes librerías LCD para diferentes tipos de PICs pero todas cumplen con las mismas funciones ya mencionadas. En el caso del dsPIC30F4013 se tuvo que modificar la librería LCD para que ésta fuera compatible con el DSPIC.

Las librerías se encuentran en la sección de *Anexos*.

Teniendo estas librerías podemos mostrar cualquier mensaje, en la posición deseada con el LCD.

En la Tabla 8 se presenta el mensaje a desplegar.

Tabla 8. Mensaje a mostrar.

Mensaje	**Posición**
UAEM CU ECATEPEC	Línea 1
Naibi Altamirano	Línea 2

Diagrama de flujo.

En la Fig. 33 se presenta el diagrama de flujo del programa desarrollado para el ejemplo 2.1. Se observa al inicio del diagrama de flujo que se configura el programa, mediante el llamado de las librerías y los encabezados, entre ellas se encuentran lcd.h y delay.h

Posteriormente se h|abilita el puerto B en modo salida y el puerto C para entrada de los datos. Hecho esto, se inicializa el LCD y se limpia pantalla. Se indica que posición tomará cada línea del mensaje.

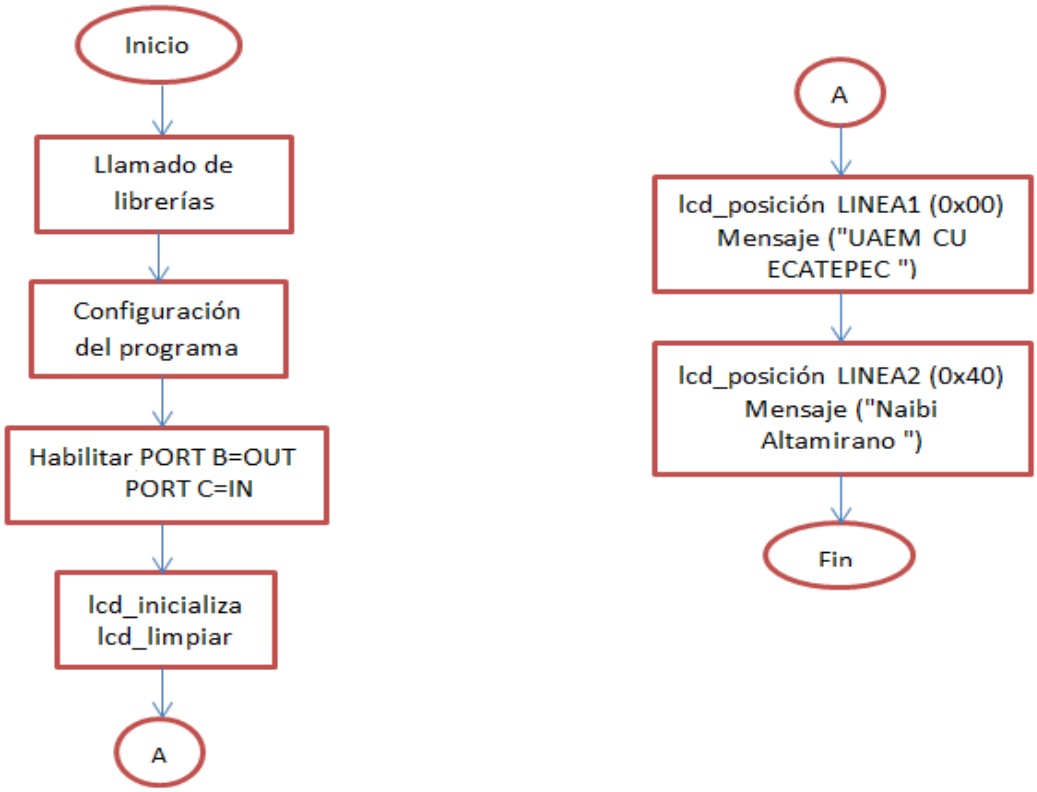


Fig. 33 Diagrama de flujo del Ejemplo 2.1

Código.

A continuación se presenta el código del programa del ejemplo 2.1, donde el color azul representa palabras reservadas, identificadores o tipos de datos. El verde indica que son comentarios que definen el uso de cada sentencia. Y el color negro son los bloques de código.

```
//Librerías necesarias para el DSPIC
#include "p30f4013.h"
#include "libpic30.h"
#include "delay_mio.h"
#include "lcd_mio.h"

//Configuración del programa
_FOSC (CSW_FSCM_OFF & XT);
_FWDT (WDT_OFF);
_FBORPOR (PBOR_ON & MCLR_EN);
_FGS (CODE_PROT_OFF);
int main(){
    ADPCFG=0x0000;      //Todos los pines del puerto B es analógico
    lcd_init();                 //Inicializo LCD
    lcd_clear();              //Limpio el LCD
```

```
    lcd_goto(LINE1);            //Posición Linea1
    lcd_puts("UAEM CU ECATEPEC"); //Cadena de caracteres
    lcd_goto(LINE2)             //Posición Linea2
    lcd_puts("Naibi Altamirano");     //Cadena de caracteres
      while(1){                //El proceso se repite indefinidamente
      }
}
```

La simulación se llevó a cabo en MPLAB, donde observa que el resultado corresponde a lo planteado con anterioridad.

Diagrama circuital.

En la Fig. 34 se muestra el diagrama del circuito a implementar. En el se puede observar la conexión del dsPIC30F4013 donde el pin 1 está conectada una resistencia de 10kΩ que va directamente al voltaje. Del pin 6 al 9 van conectados a los pines D4-D7 del LCD. En los pines 15 y 16 del DSPIC se conectan los pines RS y E del LCD.

Los pines D0 al D3 del LCD deberán estar conectados a GND para no ocasionar conflicto con los pines de datos que si estan siendo utilizados. Las demás terminales del LCD son conectadas como lo indica la Tabla 5.

El oscilador de cristal con sus respetivos capacitores se posicionan en los pines 13 y 14. Además que es importante mantener alimentado todo el DSPIC para su correcto funcionamiento.

Nota: Para los siguientes ejemplos se utiliza el mismo diseño del circuito. Solo cambia el programa.

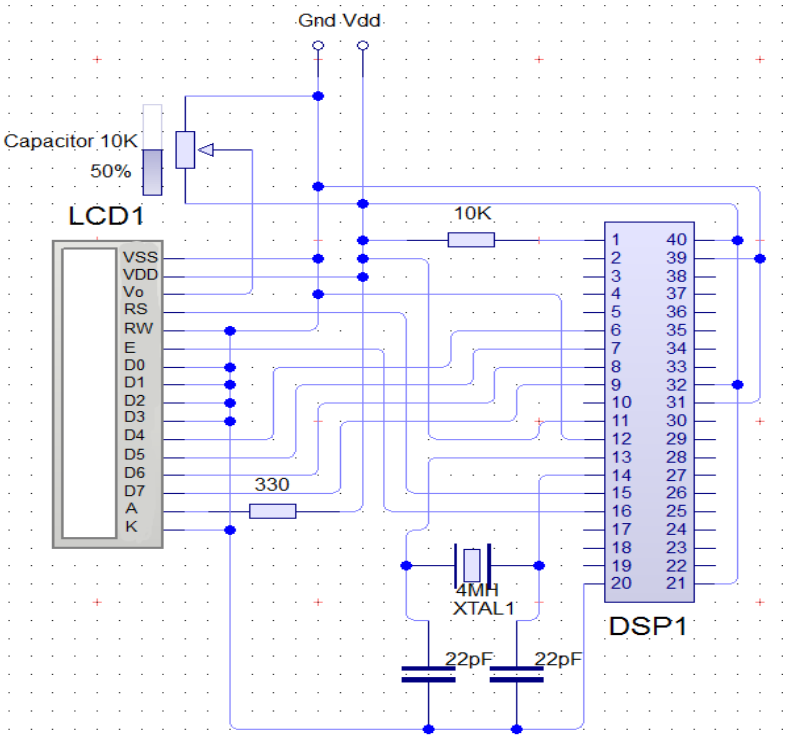


Fig. 34 Diagrama circuital.

Ensamblaje y prueba.

Finalmente se demostró (Fig. 35) que el programa junto con el diseño del circuito, funciona correctamente.

Fig. 35 Ensamblaje.

Ejemplo 2.2

Objetivo

Diseñar y desarrollar un programa utilizando el dsPIC30F4013 que despliegue los siguientes mensajes en movimiento: En la línea 1 "*UAEM CU ECATEPEC*" (mensaje fijo) y en la línea 2 un asterisco en movimiento.

Desarrollo.

El programa se desarrolla utilizando las mismas librerías mencionadas en el ejemplo 2.1. Para implementar el movimiento de la segunda línea se utiliza un ciclo, en este caso implementado con la función FOR. Dentro del ciclo FOR la variable que almacena la posición del cursor se va modificando para dar el efecto de movimiento.

El mensaje a mostrar será como lo muestra la Tabla 9:

Tabla 9. Mensaje a mostrar.

Mensaje	**Posición**	**Estado**
UAEM CU ECATEPEC	Línea 1	Fijo
*****	Línea 2	Movimiento

Diagrama de flujo.

En la Fig. 36 se presenta el diagrama de flujo del programa desarrollado para el ejemplo 2.2. Se observa al inicio del diagrama de flujo que se configura el programa, mediante el llamado de las librerías y los encabezados, entre ellas se encuentran lcd.h y delay.h

Posteriormente se habilita el puerto B en modo salida y el puerto C para entrada de los datos. Hecho esto, se inicializa el LCD y se limpia pantalla. Se indica la posición del primer mensaje y se muestra en pantalla.

Dentro de un ciclo indefinido se desarrolla el ciclo FOR, del cual su valor inicial es 0, la condición es que sea menor o igual a 16, con incrementos de uno en uno. Dentro de

este ciclo FOR se indica la posición que tomará el segundo mensaje. Para que pueda tomar la posición que va indicando el FOR, ésta debe ser sumada con la poción inicial del mansaje.

Finalmente se crea un ciclo parecido al anterior, con la diferencia de que se ingresa un espacio en blanco para suprimir el primer ciclo FOR.

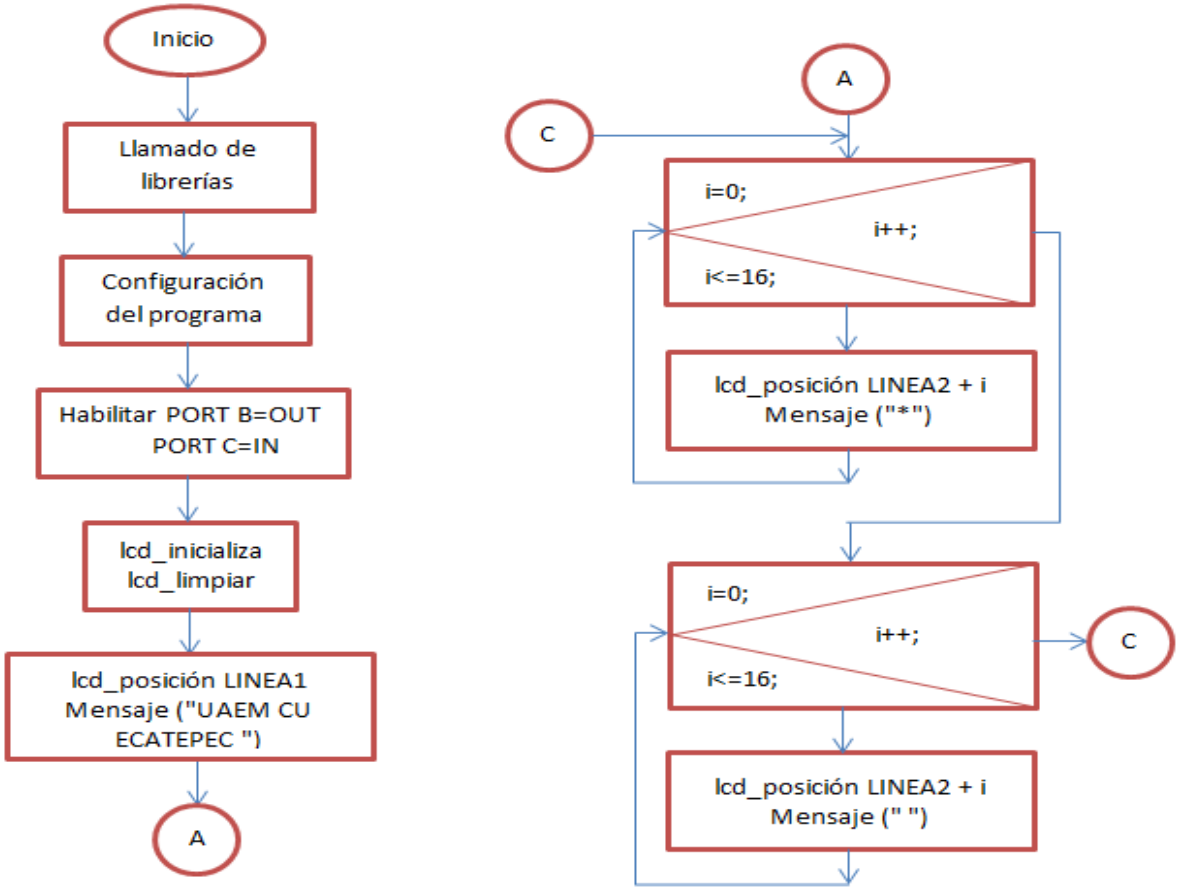


Fig. 36 Diagrama de flujo del Ejemplo 2.2

Código.

A continuación se presenta el código del programa del ejemplo 2.2

```
//Librerías necesarias para el DSPIC
#include "p30f4013.h"
#include "libpic30.h"
#include "delay_mio.h"
#include "lcd_mio.h"

//Configuración del programa
_FOSC (CSW_FSCM_OFF & XT);
```

```
_FWDT (WDT_OFF);
_FBORPOR (PBOR_ON & MCLR_EN);
_FGS (CODE_PROT_OFF);

int main()
{
    ADPCFG=0x0000;      //Todos los pines del puerto B es analógico
    int i;
    lcd_init();                 //Inicializo LCD
    lcd_clear();              //Limpio el LCD
    lcd_goto(LINE1);            //Posición Linea1
    lcd_puts("UAEM CU ECATEPEC"); //Cadena de caracteres
    while(1){
            for (i=0;i<=16;++i)
            {
                  lcd_goto(i+LINE2);  //Linea 2 más el ciclo en que se
encuentre FOR
                  lcd_puts("*");    //Escribe la cadena de caracteres
            }
            for (i=0;i<=16;++i)
            {
                  lcd_goto(i+LINE2);  /  Linea 2 más el ciclo en que se
encuentre FOR
                  lcd_puts(" "); //Escribe la cadena de caracteres
            }
      }
}
```

La simulación se llevó a cabo en MPLAB, donde observa que el resultado corresponde a lo planteado con anterioridad.

Ensamblaje y prueba.

Finalmente se demostró (Fig. 37) que el programa junto con el diseño del circuito, funciona correctamente.

Fig. 37 Ensamblaje

Ejemplo 2.3

Objetivo

Diseñar y desarrollar un programa utilizando el dsPIC30F4013 que despliegue en un conteo decimal que corra de 0 a 15.

Desarrollo.

De manera similar se hace uso de un ciclo FOR para generar el conteo en el proceso. En este caso se utiliza la función *sprintf* para mostrar una cadena de caracteres y especificar otros objetos como el formato y el valor.

Formato de la función **sprintf**

```
sprintf([Objeto],"texto = %a",Valor);
```

Ejemplo.

```
char x[10];
sprintf(x,"El valor es = %i",i);
```

El mensaje a mostrar será como lo muestra la Tabla 10:

Tabla 10. Mensaje a mostrar

Mensaje	**Posición**	**Estado**
Conteo ciclo =	Línea 1	Movimiento

Diagrama de flujo.

En la Fig. 38 se presenta el diagrama de flujo del programa desarrollado para el ejemplo 2.3. Al inicio del diagrama de flujo se configura el programa, mediante el llamado de las librerías y los encabezados, entre ellas se encuentran lcd.h y delay.h

Posteriormente se habilita el puerto B en modo salida y el puerto C para entrada de los datos. Hecho esto, se inicializa el LCD y se limpia pantalla. Se crea un espacio en el buffer para almacenar la cadena de caracteres.

Dentro de un ciclo indefinido se desarrolla el ciclo FOR, el cual su el valor inicial es 0, la condición es que sea menor o igual a 16, con incrementos de uno en uno.

Dentro de este ciclo FOR se indica la posición del primer mensaje, además, se indica que lo almacenado en el buffer será visualizado en el LCD. Llamando la función *sprintf*, se define el primer mensaje junto con el valor que vaya tomando el ciclo.

Finalmente se limpia la pantalla cuando el ciclo FOR haya terminado.

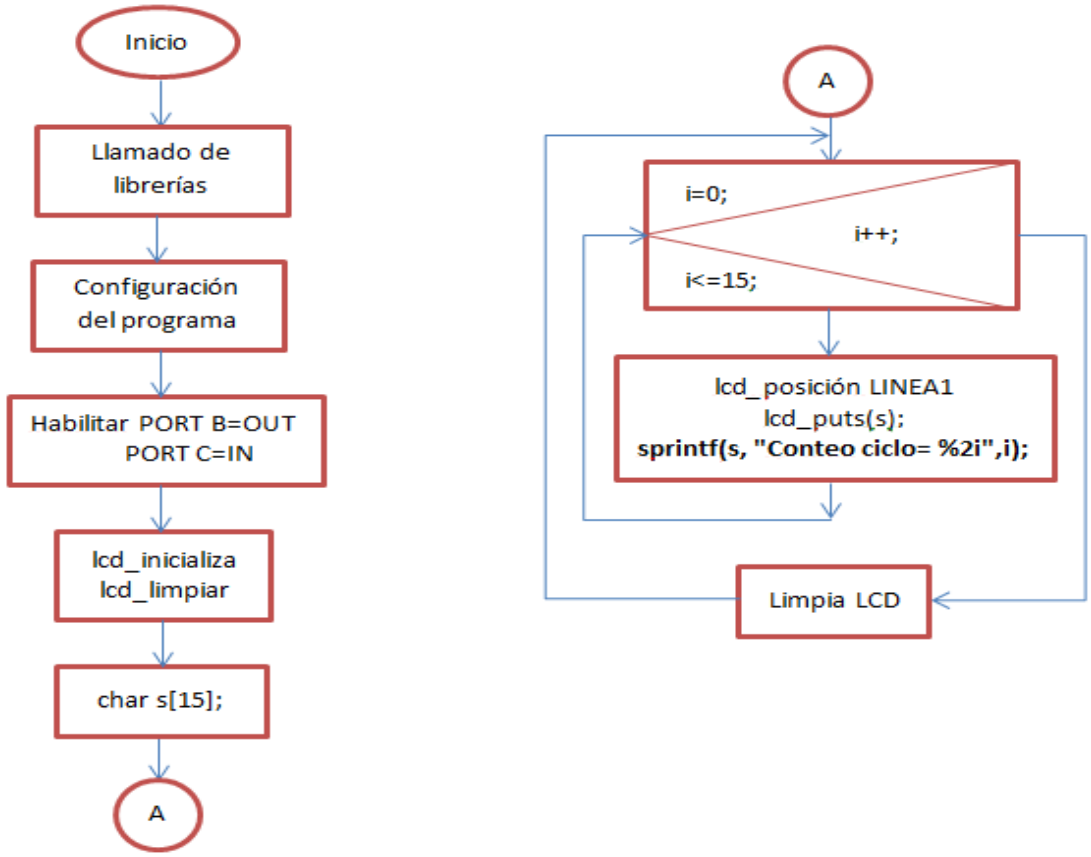


Fig. 38 Diagrama de flujo del Ejercicio 2.3

Código.

A continuación se presenta el código del programa del ejemplo 2.3

```
//Librerías necesarias para el DSPIC
#include "p30f4013.h"
#include "libpic30.h"
#include "delay_mio.h"
#include "lcd_mio.h"
```

```
//Configuración del programa
_FOSC (CSW_FSCM_OFF & XT);
_FWDT (WDT_OFF);
_FBORPOR (PBOR_ON & MCLR_EN);
_FGS (CODE_PROT_OFF);

int main()
{
    ADPCFG=0x0000;      //Todos los pines del puerto B es analógico
    int i;
    lcd_init();                 //Inicializo LCD
    lcd_clear();              //Limpio el LCD
    char s[15];
    while(1){
            for (i=0;i<=15;++i)
            {
                  lcd_goto(LINE1);  //Va a la posición especificada. Linea 1
                  lcd_puts(s);      //La cadena de caracteres se almacena en
la variable s
                  sprintf(s, "Conteo ciclo= %2i",i); //Se especifica el número
de enteros que aparezcan en pantalla (%2)
            }
      }
}
```

La simulación se llevó a cabo en MPLAB y se observa que el resultado corresponde a lo planteado con anterioridad.

Ensamblaje y prueba.

Finalmente se demostró (Fig. 39 y 40) que el programa junto con el diseño del circuito, funciona correctamente.

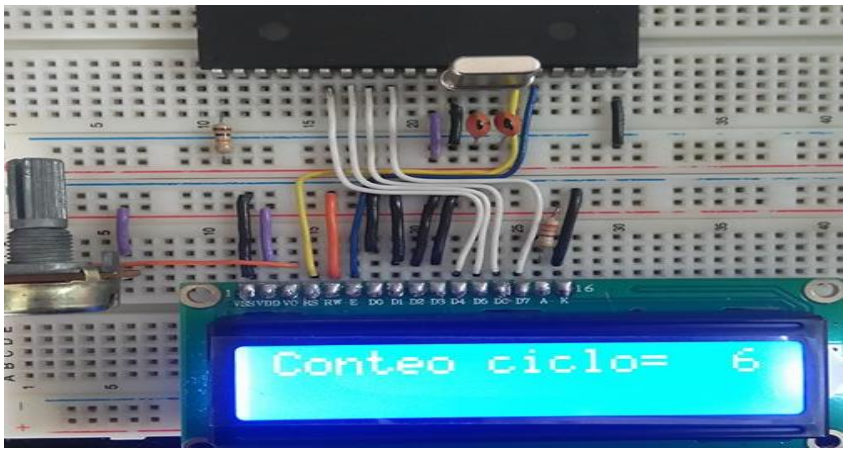


Fig. 39 Ensamblaje

Fig. 40 Ensamblaje

Ejemplo 2.4

Objetivo

Diseñar y desarrollar un programa utilizando el dsPIC30F4013 que despliegue un mensaje de derecha a izquierda.

Desarrollo.

Por medio de un ciclo FOR se puede crear un recorrido para cualquier caracter, tomando en cuenta las dimensiones del LCD para formar los valores y condiciones del ciclo. Si no se toman en cuenta las dimensiones límites del LCD, el mensaje podría ser visualizado en la línea equivocada. Para mostrar el mensaje en movimiento, se utiliza la función *sprintf* para poder visualizar toda la cadena de caracteres.

El mensaje a mostrar será como lo muestra la Tabla 11:

Tabla 11. Mensaje a mostrar

Mensaje	**Posición**	**Estado**
Desplazo del LCD	Línea 1	Fijo
UAEM CU ECATEPEC	Línea 2	Desplazamiento

Diagrama de flujo.

En la Fig. 41 se presenta el diagrama de flujo del programa desarrollado para el ejemplo 2.4. Al inicio del diagrama de flujo se configura el programa, mediante el llamado de las librerías y los encabezados, entre ellas se encuentran lcd.h y delay.h

Posteriormente se habilita el puerto B en modo salida y el puerto C para entrada de los datos. Hecho esto, se inicializa el LCD y se limpia pantalla. Se crea un espacio en el buffer para almacenar la cadena de caracteres.

Dentro de un ciclo indefinido se indica la posición del primer mensaje y se muestra en pantalla. Además, se desarrolla el ciclo FOR, el cual su el valor inicial es 79, porque es un valor menor al del rango final derecho del LCD; la condición es que sea mayor a 21, es cuando termina de mostrarse toda la cadena de texto; con decrementosx de uno en uno.

Dentro de este ciclo FOR se indica la posición del segundo mensaje, el cual debe ser restado con el valor que vaya tomando el ciclo. Llamando la función sprintf, se define el mensaje.

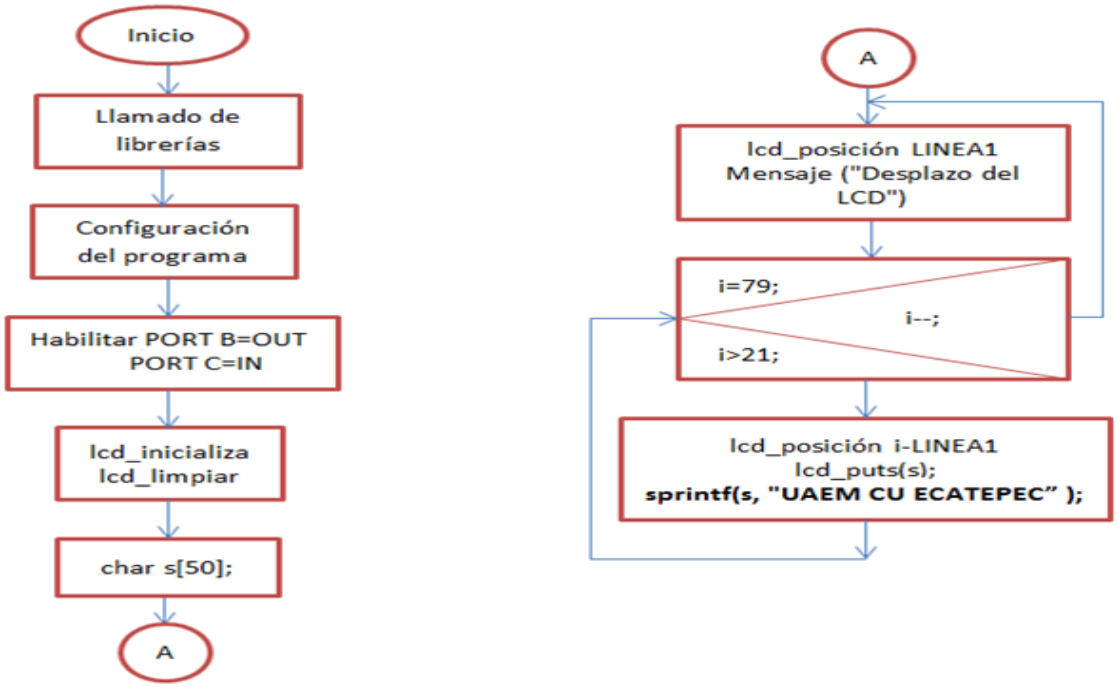


Fig. 41 Diagrama de flujo del Ejercicio 2.4

Código.

A continuación se presenta el código del programa del ejemplo 2.4

```
//Librerías necesarias para el DSPIC
#include "p30f4013.h"
#include "libpic30.h"
#include "delay_mio.h"
#include "lcd_mio.h"

//Configuración del programa
_FOSC (CSW_FSCM_OFF & XT);
_FWDT (WDT_OFF);
_FBORPOR (PBOR_ON & MCLR_EN);
_FGS (CODE_PROT_OFF);

int main()
{
    ADPCFG=0x0000;      //Todos los pines del puerto B es analógico
    int i;
    lcd_init();                 //Inicializo LCD
    lcd_clear();              //Limpio el LCD

    while(1){
            char s[50];
            lcd_goto(LINE1);  //Va a la posición especificada. Linea 1
            lcd_puts("Desplazo del LCD");//Escribe la cadena de caracteres
            for (i=79;i>21;--i)
            {
                  lcd_goto(i-LINE1);
                  lcd_puts(s);      //La cadena de caracteres se almacena en
la variable s
                  sprintf(s,"UAEM   CU   ECATEPEC   ");//Escribe  la  cadena  de
caracteres
            }
      }
}
```

La simulación se llevó a cabo en MPLAB y se observa que el resultado corresponde a lo planteado con anterioridad.

Ensamblaje y prueba.

Finalmente se demostró (Fig. 42 y 43) que el programa junto con el diseño del circuito, funciona correctamente.

Fig. 42 Ensamblaje

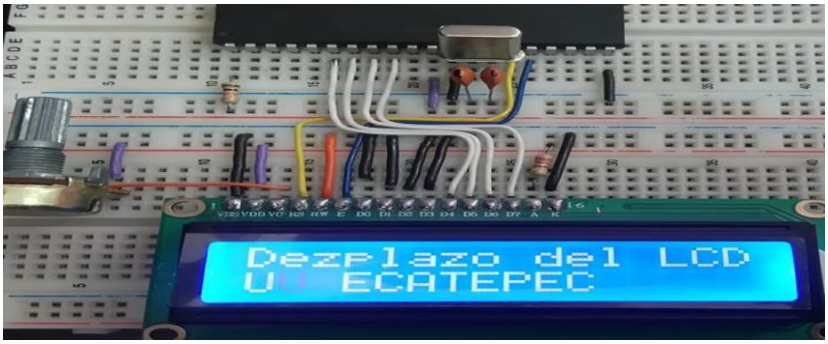


Fig. 43 Ensamblaje

Ejemplo 3.

El teclado matricial, Fig. 44, es un arreglo de botones conectados en filas y columnas, de modo que se pueden leer varios botones con el número mínimo de pines requeridos. Un teclado matricial de 4x4 pueden leer 16 teclas, ocupando solamente 8 líneas de un microcontrolador, 4 líneas de un puerto para las filas y otras 4 líneas para las columnas.

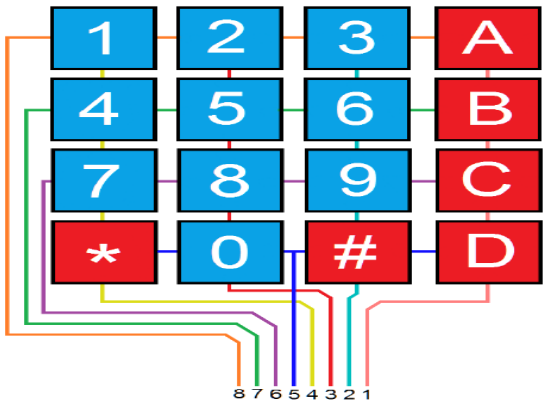


Fig. 44 Conexión del teclado matricial.

Para que el microcontrolador pueda determinar qué botón se pulsa, primero se tiene que cargar cada una de las cuatro filas (pines 5- 8), ya sea un bajo (0) o en alto (1) y luego sondear los estados de las cuatro columnas (pines 1-4). Dependiendo del estado de la fila y columna el microcontrolador puede decir qué botón se presiona.

A continuación se describirán algunos ejemplos en las que se utiliza este dispositivo con diferentes aplicaciones.

Ejemplo 3.1

Objetivo

Diseñar y desarrollar un programa utilizando el dsPIC30F4013 que despliegue en un LCD de 16x2 el valor de cada botón del teclado matricial 4x4.

Desarrollo.

Para trabajar con el teclado matricial se debe especificar las entradas y salidas del puerto del microcontrolador para diferenciar las filas de las columnas. En este caso, se configura el nibble alto (RB4, RB5, RB6, RB7) de PORTB como entrada y el nibble bajo (RB0, RB1, RB2, RB3) de PORTB como salida. En la Tabla 12 y en la Fig. 45 se muestra el resumen de lo anterior.

Tabla 12. Configuración del Puerto B.

Nibble	**PortB**	**Estado**
Bajo	RB0, RB1, RB2, RB3	Salida
Alto	RB4, RB5, RB6, RB7	Entrada

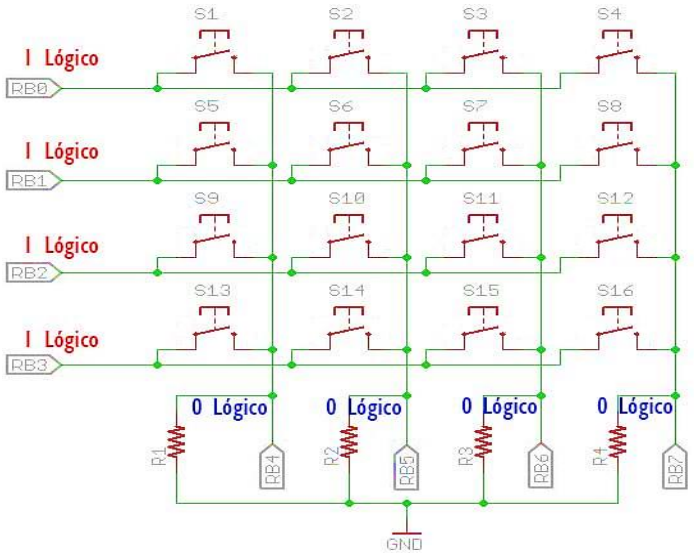


Fig. 45 Conexión de los nibbles de PORTB.

Las salidas están programadas a generar un “1” lógico a cada fila del teclado matricial, y la función de las entradas es detectar que columna se está presionando cuando este cambie de estado de 0 a 1 (Fig. 46).

De esta manera el microcontrolador sabe que se ha oprimido una tecla al detectar un cambio de nivel en los bits más significativos.

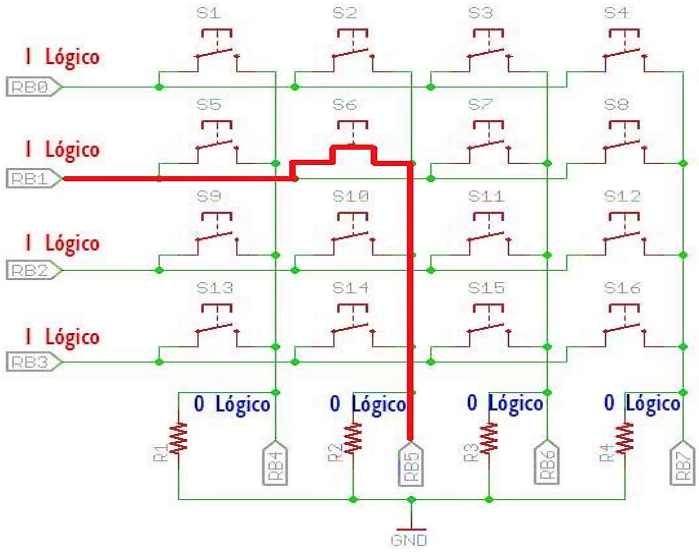


Fig. 46 Cambio de estado de RB5.

Finalmente para representar cada valor en el LCD es necesario retomar la librería utilizada en el ejemplo 2.1. Modificando las salidas de la librería, es decir, para el LCD se utilizará el puerto D para no causar conflicto con el puerto B que es utilizado para el teclado matricial. Con ésta librería solo se imprime la información que se quiera visualizar.

Diagrama de flujo.

En la Fig. 47 se presenta el diagrama de flujo del programa desarrollado para el ejemplo 3.1. Se observa al inicio del diagrama de flujo que se configura el programa, mediante el llamado de las librerías y los encabezados, entre ellas se encuentran lcd.h y delay.h

Posteriormente se definen los puertos de entrada que serán las columnas y los puertos de salida que serán las filas del teclado matricial. Hecho esto, se inicializa el LCD y se limpia pantalla. Se manda un mensaje inicial en la segunda línea del LCD.

Se genera un ciclo constante de encendido y apagado de los nibbles menos significativos.

Si en RB0 está en estado alto, se activa la fila1, si en la fila 1 está en estado alto RB4, en el LCD aparecerá 1. Si por el contrario, está activado RB5 se imprime 2. Si es RB6 que se encuentra activado, su valor será 3. Si RB7 está en estado alto, el valor que imprima será la letra A.

Si en RB1 está en estado alto, se activa la fila2, realizando las mismas condiciones que en el bit anterior, con la diferencia de mostrar datos distintos en el LCD. La fila3 se activa cuando RB2 se encuentre en estado alto y la fila4 con RB3.

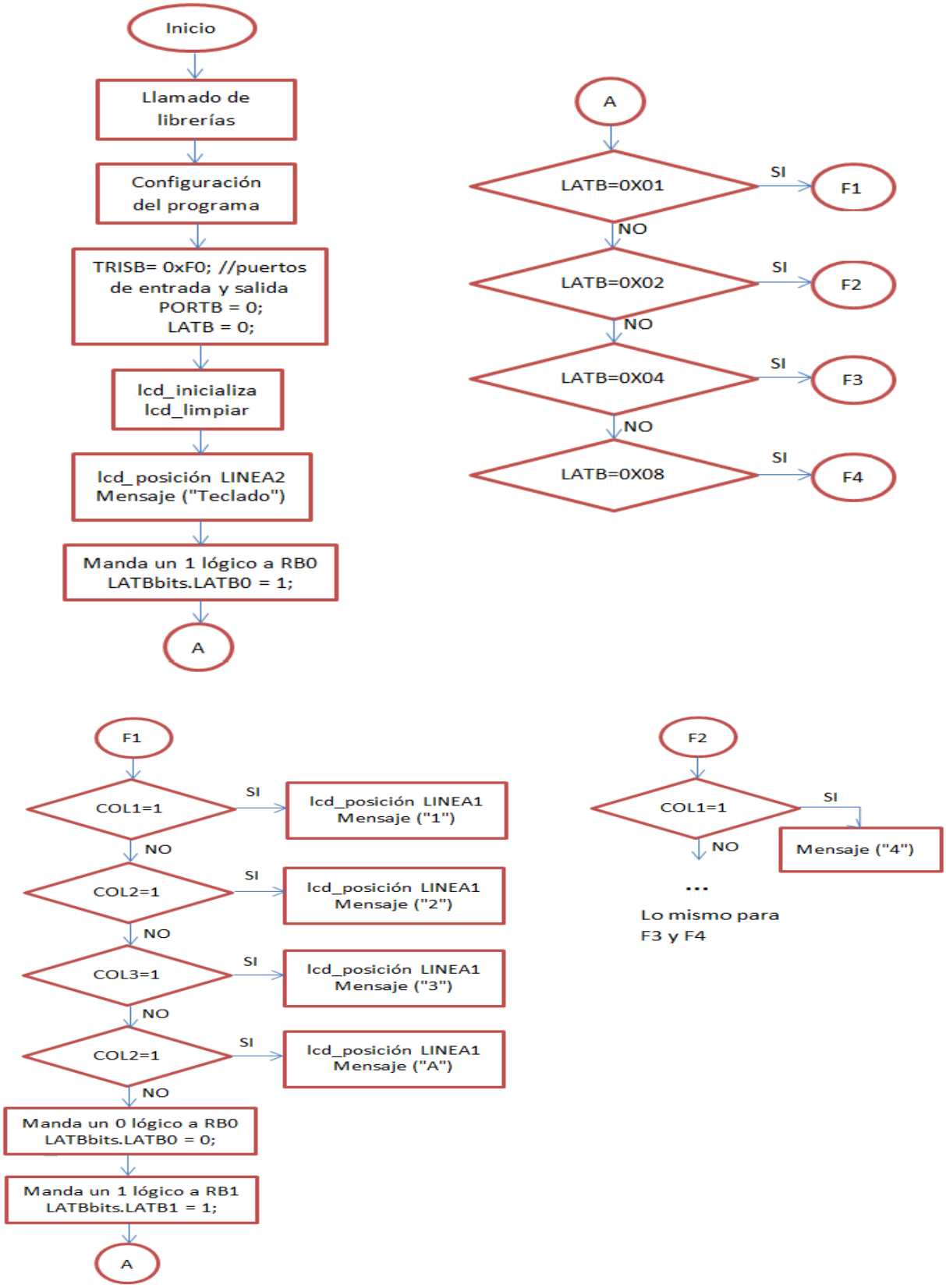


Fig. 47 Diagrama de flujo del Ejercicio 3.1

Código.

A continuación se presenta el código del programa del ejemplo 3.1, donde el color azul representa palabras reservadas, identificadores o tipos de datos. El verde indica que son comentarios que definen el uso de cada sentencia. Y el color negro son los bloques de código.

```
//Librerías necesarias para el DSPIC
#include "p30f4013.h"
#include "libpic30.h"
#include "delay_mio.h"
#include "lcd_mio.h"

//Configuración del programa
_FOSC (CSW_FSCM_OFF & XT);
_FWDT (WDT_OFF);
_FBORPOR (PBOR_ON & MCLR_EN);
_FGS (CODE_PROT_OFF);
#define COL1      PORTBbits.RB4
#define COL2      PORTBbits.RB5
#define COL3      PORTBbits.RB6
#define COL4      PORTBbits.RB7
int main()
{
     TRISB= 0xF0; // Configura el nibble alto de PORTB como entrada y el
     nibble bajo como salida
    PORTB = 0;                  //entradas
    LATB = 0;                   //salidas
    lcd_init();               //Inicializo LCD
    lcd_clear();                //Limpio el LCD
    while(1){
           lcd_goto(LINE2); //posición Línea 2
            lcd_puts("Teclado");//mensaje fijo
            bits:               //etiqueta
            LATBbits.LATB0 = 1; //manda 1 a RB0
            goto rb0;           //se dirige a rbo
            bits1:              //etiqueta
            LATBbits.LATB1 = 1; // manda 1 a RB1
            goto rb0;
            bits2:              //etiqueta
            LATBbits.LATB2 = 1; manda 1 a RB2
            goto rb0;
            bits3:              //etiqueta
            LATBbits.LATB3 = 1; manda 1 a RB3
            goto rb0;
      }
      rb0:
            if(LATB==0X01){ //SI en puerto B hay 00000001
                  goto fila1; //se dirige a fila1
            }
            else                //SI NO
                  if(LATB==0X02){ //SI en puerto B hay 00000010
                        goto fila2; //se dirige a fila2
                  }
            else                //SI NO
                  if(LATB==0X04){ //SI en puerto B hay 00000100
                        goto fila3; //se dirige a fila3
                  }
            else                //SI NO
```

```
            if(LATB==0X08){ //SI en puerto B hay 00001000
                  goto fila4; //se dirige a fila4
            }
fila1:
      if(COL1 == 1){    //SI columna 1 = 1
            lcd_goto(LINE1);
            lcd_puts("1");
      }
      else if(COL2 == 1){ //SI columna 2 = 1
            lcd_goto(LINE1);
            lcd_puts("2");
            }
      else if(COL3 == 1){ //SI columna 3 = 1
            lcd_goto(LINE1);
            lcd_puts("3");
            }
      else if(COL4 == 1){ //SI columna 4 = 1
            lcd_goto(LINE1);
            lcd_puts("A");
            }
      LATBbits.LATB0 = 0;
      goto bits1;
fila2:
      if(COL1 == 1){
            lcd_goto(LINE1);
            lcd_puts("4");
            }
      else if(COL2 == 1){
            lcd_goto(LINE1);
            lcd_puts("5");
            }
      else if(COL3 == 1){
            lcd_goto(LINE1);
            lcd_puts("6");
            }
      else if(COL4 == 1){
            lcd_goto(LINE1);
            lcd_puts("B");
            }
      LATBbits.LATB1 = 0;
      goto bits2;
fila3:
      if(COL1 == 1){
            lcd_goto(LINE1);
            lcd_puts("7");
            }
      else if(COL2 == 1){
            lcd_goto(LINE1);
            lcd_puts("8");
            }
      else if(COL3 == 1){
            lcd_goto(LINE1);
            lcd_puts("9");
            }
      else if(COL4 == 1){
            lcd_goto(LINE1);
            lcd_puts("C");
            }
      LATBbits.LATB2 = 0;
      goto bits3;
fila4:
      if(COL1 == 1){
            lcd_goto(LINE1);
            lcd_puts("*");
            }
      else if(COL2 == 1){
            lcd_goto(LINE1);
            lcd_puts("0");
            }
      else if(COL3 == 1){
            lcd_goto(LINE1);
            lcd_puts("#");
            }
      else if(COL4 == 1){
            lcd_goto(LINE1);
            lcd_puts("D");
            }
      LATBbits.LATB3 = 0;
      goto bits;
}//Fin de main
```

La simulación se llevó a cabo en MPLAB y se observa que el resultado corresponde a lo planteado con anterioridad.

Diagrama circuital.

En la Fig. 49 se muestra el diagrama del circuito a implementar. En el se puede observar la conexión del dsPIC30F4013 donde el pin 1 está conectado a una resistencia de 10kΩ que va directamente al voltaje. En los pines 15 y 16 del DSPIC se conectan los pines RS y E del LCD. Los pines D0 al D3 del LCD deberan estar conectados a GND para no ocacionar conflicto con los pines de datos utilizados. Los pines de datos están colocados en el puerto D del DSPIC; D4 y D5 en el pin 34 y 33, respectivamente. D6 en pin 22 y D7 en pin 19.

Las demás terminales del LCD son conectadas como lo indica la Tabla 12.

Para el teclado matricial se utilizan ocho pines del puerto B, las filas se conectan de RB0 a RB3 (patillaje: 2-5) y las columnas de RB4 a RB7 (patillaje: 6-9).

Cada columna del teclado matricial tienen que ir conectada a una resistencia de valor minimo de 330Ω para hacer el efecto de Pull-Down (Fig. 48), de esta manera cuando el interruptor este abierto la corriente se dirija hacia la resistencia dejando un valor 0 (LOW) y si el interruptor esta cerrado la corriente se moverá a Vout dejando un valor lógico 1 (HIGH), manteniendo visualizado el valor del teclado.

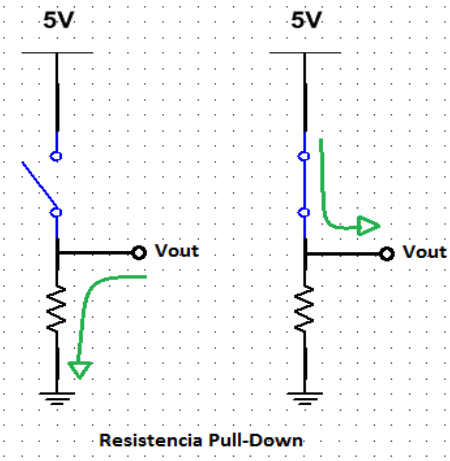


Fig. 48 Diagrama Resistencia PULL-DOWN

Además, la configuración Pull-Down, a diferncia del Pull-Up, resulta conveniente porque no todo el tiempo se encuentra consumiendo portencia del circuito.

El oscilador de cristal con sus respetivos capacitores se posicionan en los pines 13 y 14. Además que es importante mantener alimentado todo el DSPIC para su correcto funcionamiento.

Nota: Para los siguientes ejemplos se utiliza el mismo diseño del circuito. Solo cambia el programa.

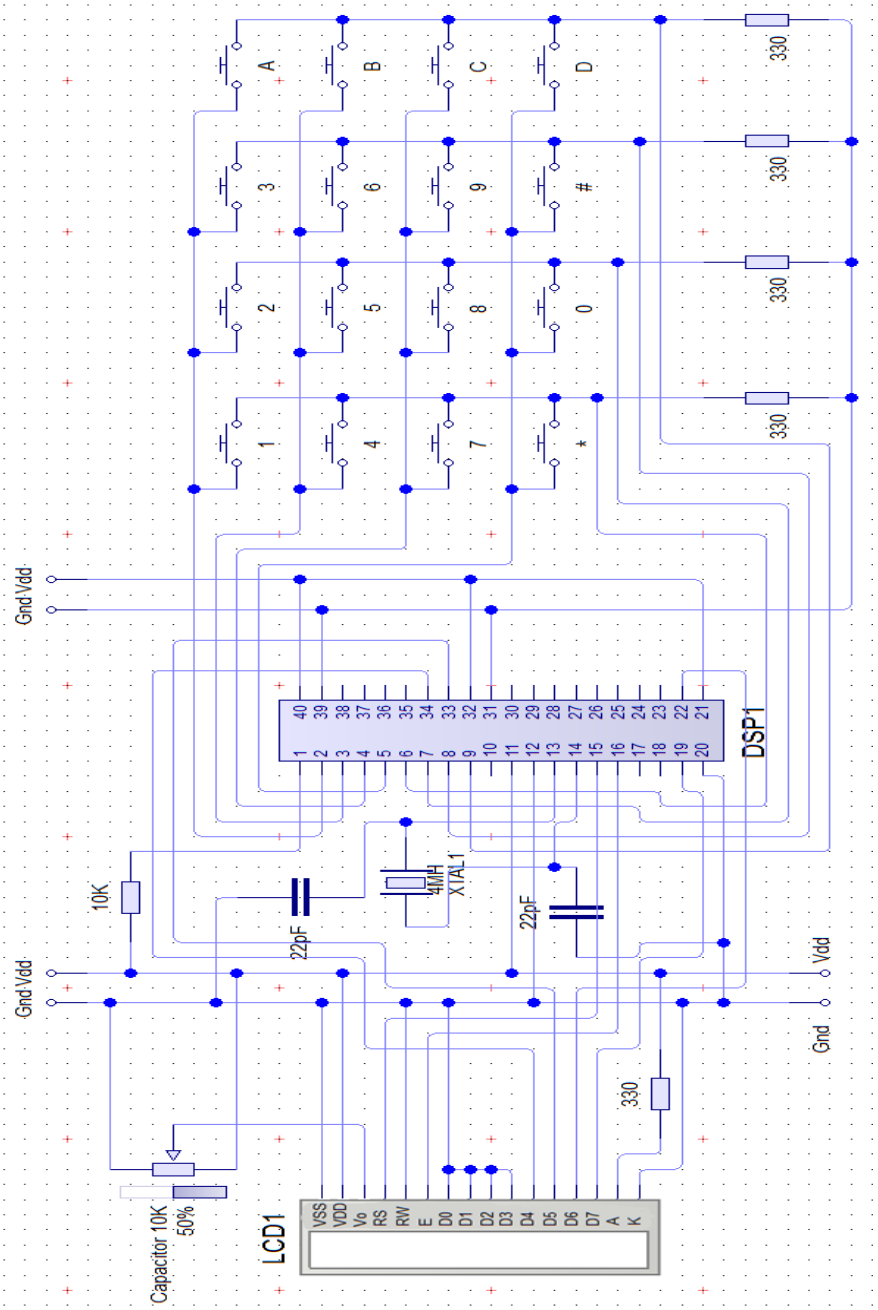


Fig. 49 Diagrama circuital.

Ensamblaje y prueba.

Finalmente se demostró (Fig. 50 y 51) que el programa junto con el diseño del circuito, funciona correctamente.

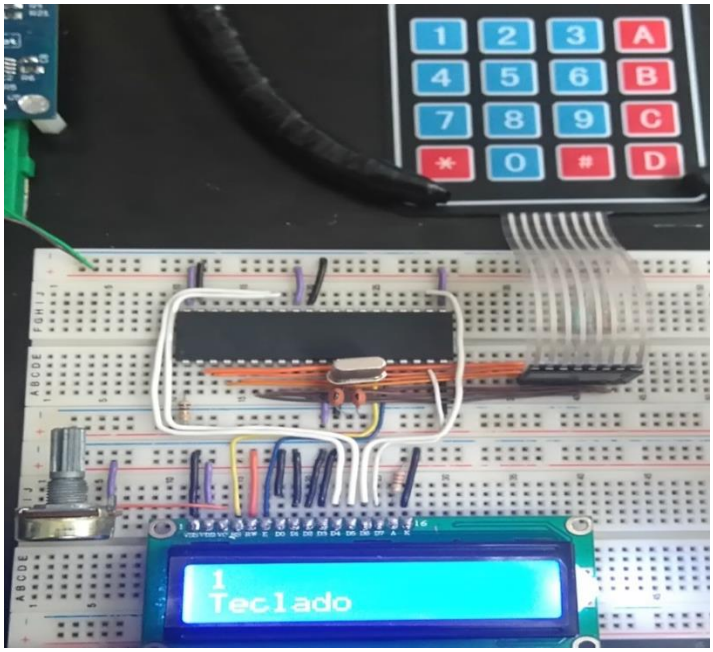


Fig. 50 Ensamblaje

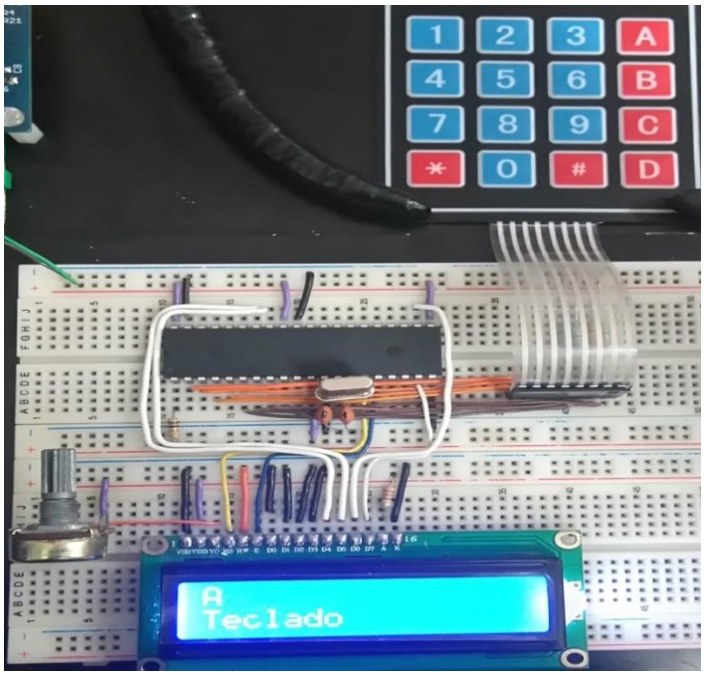


Fig. 51 Ensamblaje

Ejemplo 3.2

Objetivo

Diseñar y desarrollar un programa utilizando el dsPIC30F4013 que emule el funcionamiento de una calculadora básica.

Desarrollo

Para el desarrollo de esta práctica, el valor de cada botón del teclado matricial debe ser de tipo entero para que pueda ser almacenado y calculado con la operación deseada. Es por esta razón que se ha creado una librería para identificar el valor de cada botón del teclado, siendo totalmente diferente con el ejemplo 3.1, donde cada valor era mostrado en una cadena de caracteres. La práctica 3.2 constará de dos partes, en la primera se describe el funcionamiento de la librería del teclado matricial, y la segunda parte describe la realización de la calculadora básica.

Primera parte

El análisis de cómo funciona el teclado matricial como dispositivo está descrito en el ejemplo 3.1. De igual manera se debe especificar las entradas y salidas del puerto del microcontrolador para diferenciar las filas de las columnas. El nibble alto (RB4, RB5, RB6, RB7) de PORTB se define como entrada y el nibble bajo (RB0, RB1, RB2, RB3) de PORTB como salida. En la Tabla 12 se muestra el resumen de lo anterior.

Nota: Como se realizará una librería, se deben crear dos archivos, uno de ellos debe ser guardado con extensión **.c** donde se encuentre todos los procesos necesarios, y otro archivo con extensión **.h**, donde su función principal es mandar a llamar cada método utilizado en el archivo **.c**. Es necesario contar con el archivo *.h* ya que éste es colocado en el encabezado del programa principal del ejemplo 3.2.

Diagrama de flujo.

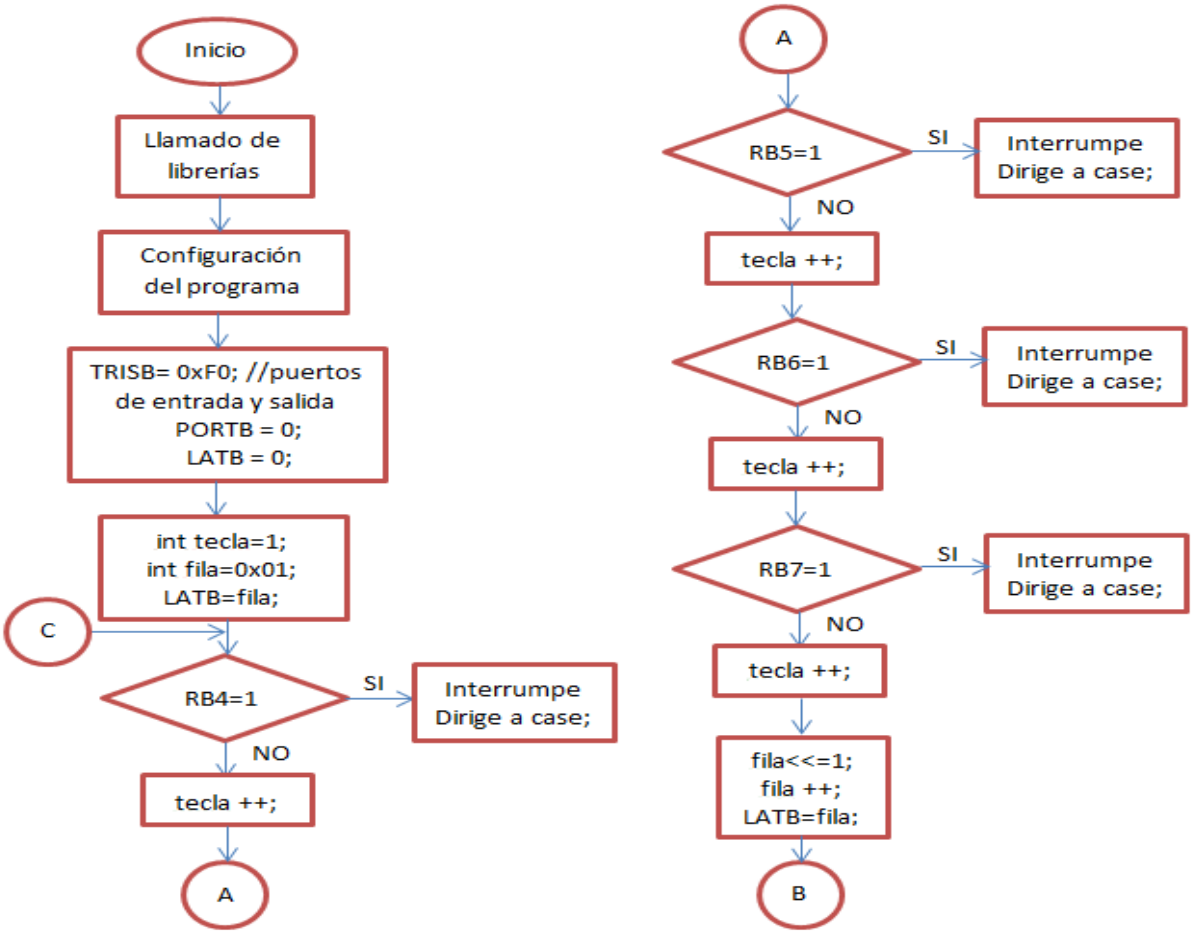
En la Fig. 52 se presenta el diagrama de flujo del programa desarrollado para crear la librería del teclado matricial 4x4. Se observa al inicio del diagrama de flujo que se configura el programa, mediante el llamado de las librerías y los encabezados.

Posteriormente se definen los puertos de entrada que serán las columnas y los puertos de salida que serán las filas del teclado matricial. Se declara “tecla”, una variable de tipo entero que almacena la posición de la tecla presionada. Así como una variable “fila” que indique la fila en la que se encuentra presionada la tecla, creando con esta un ciclo constante de encendido y apagado para cada fila.

Se genera un ciclo constante de encendido o apagado de los nibbles más significativos:

Si RB4 está en estado alto, interrumpe el proceso para indicar que botón se está presionando con relación a la fila activada. Retornando un valor definido para cada botón.

Si fuera RB5 quien estuviera activado, le suma un uno a la variable *tecla* para que interrumpa el proceso en la posición definida junto con la fila activada, retornando el valor correspondiente. Lo mismo pasa para RB6 y RB7.



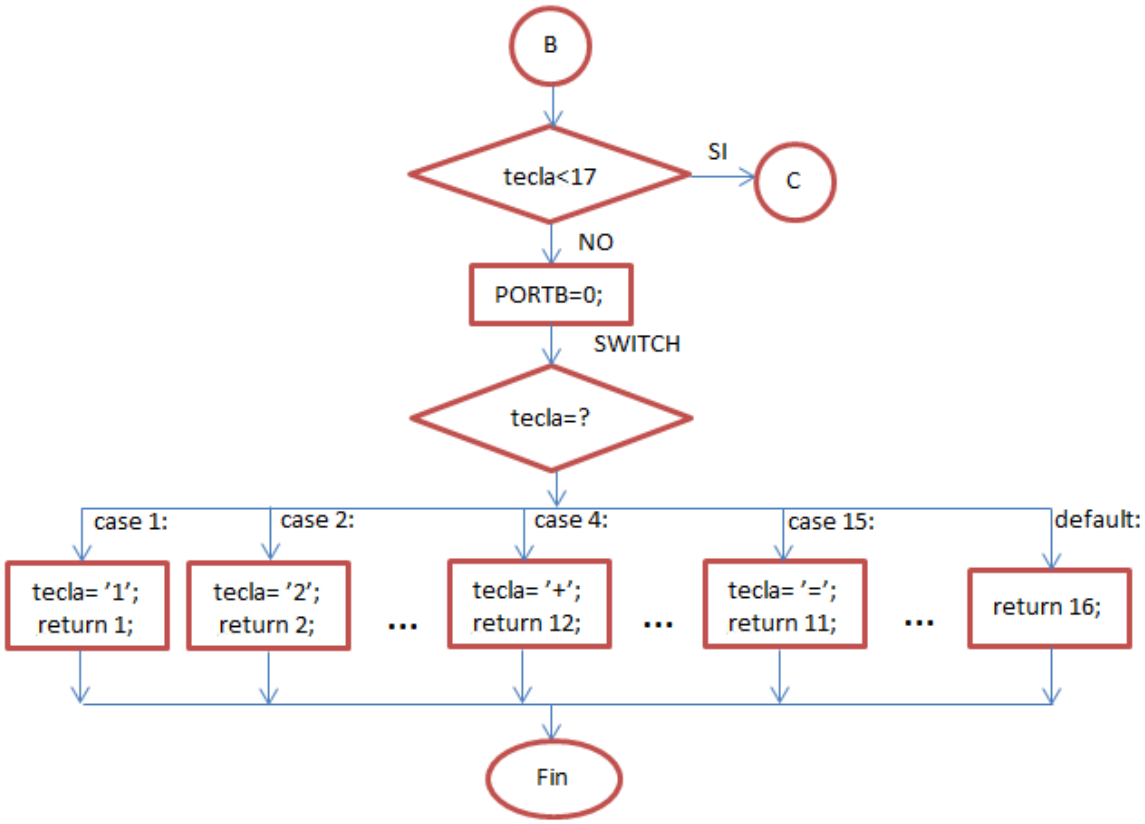


Fig. 52 Diagrama de flujo del Ejercicio 3.2

Teclado.c

Código.

A continuación se presenta el código del programa del ejemplo 3.2 parte 1, donde el color azul representa palabras reservadas, identificadores o tipos de datos. El verde indica que son comentarios que definen el uso de cada sentencia. Y el color negro son los bloques de código.

TECLADO.C

```
// Este programa obtiene la tecla pulsada del teclado matricial y la decodifica.

//Librerías necesarias para el DSPIC
#include "p30f4013.h"
#include "libpic30.h"
#include "teclado.h"

int leeTecla(void)
{
    int tecla = 0;      //Tecla = 0
    int fila = 0x01; // Fila1 = 1, demás filas = 0
```

```
    LATB = fila;
    do
    {
      //Columna1 = 1?
      if(PORTBbits.RB4 == 1)
      break;      //si, interrumpe y se dirige al switch
      tecla++;  //no, suma uno a tecla

      //Columna2 = 0?
      if(PORTBbits.RB5 == 1)
      break;
      tecla++;

      //Columna3 = 0?
      if(PORTBbits.RB6 == 1)
      break;
      tecla++;

      //Columna4 = 0?
      if(PORTBbits.RB7 == 1)
      break;
      tecla++;a
      //Siguiente Fila=1
      fila <<= 1; fila++;
      LATB  = fila;
    }while(tecla < 16); //Última tecla?
    PORTB = 0;
    // Decodifica la tecla
    switch(tecla)
    {
      case 0: tecla='1'; return 1;
      case 1: tecla='2'; return 2;
      case 2: tecla='3'; return 3;
      case 3: tecla='+'; return 12; //SUMA
      case 4: tecla='4'; return 4;
      case 5: tecla='5'; return 5;
      case 6: tecla='6'; return 6;
      case 7: tecla='-'; return 13;//RESTA
      case 8: tecla='7'; return 7;
      case 9: tecla='8'; return 8;
      case 10: tecla='9'; return 9;
      case 11: tecla='*'; return 14;      //MULTIPLICACIÓN
      case 12: tecla='CE'; return 10;     //BORRAR
      case 13: tecla='0'; return 0;
      case 14: tecla='='; return 11;      //IGUAL
      case 15: tecla='/'; return 15;      //DIVISIÓN
      default: return 16;
    }
}//fin de función leeTecla
void iniciaTeclado(void)
{
     ADPCFG = 0xFFFF; // Todas las entradas del puerto B son digitales.
// Configura el nibble alto de PORTB como entrada y el nibble bajo como salida
     TRISB= 0xF0;
     PORTB = 0;
     LATB = 0;
}
void setTeclado(void)
{
      TRISB = 0xF0;
//Deja listo PORTB para producir interrupción (salidas enviando 0)
     PORTB = 0;
     LATB = 0;
}
```

TECLADO.H

```
//Este programa obtiene el valor de los métodos creados en TECLADO.C

//directivas que permiten comprobar si un identificador está o no actualmente
definido
#ifndef TECLADO_H
#define TECLADO_H

void iniciaTeclado(void);
void setTeclado(void);
int leeTecla(void);

#endif
```

La simulación se llevó a cabo en MPLAB y se observa que el resultado corresponde a lo planteado con anterioridad.

Segunda parte

Obtenidos los valores de cada tecla se pueden crear las condiciones para cada operación de la calculadora básica. Para ello se necesita almacenar 2 números cualesquiera y el resultado se obtendrá dependiendo de la operación que se haya realizado.

Tabla 13. Operaciones básicas.

Número A	Número B	Operación	Resultado
Valor1	Valor2	+	Resul=Valor1+Valor2
Valor1	Valor2	-	Resul=Valor1-Valor2
Valor1	Valor2	*	Resul=Valor1*Valor2
Valor1	Valor2	/	Resul=Valor1/Valor2

Diagrama de flujo.

En la Fig. 53 se presenta el diagrama de flujo del programa desarrollado para el ejemplo 3.2. Se observa al inicio del diagrama de flujo que se configura el programa, mediante el llamado de las librerías y los encabezados, entre estos se encuentra la librería teclado.h y lcd.h (ubicado en anexos).

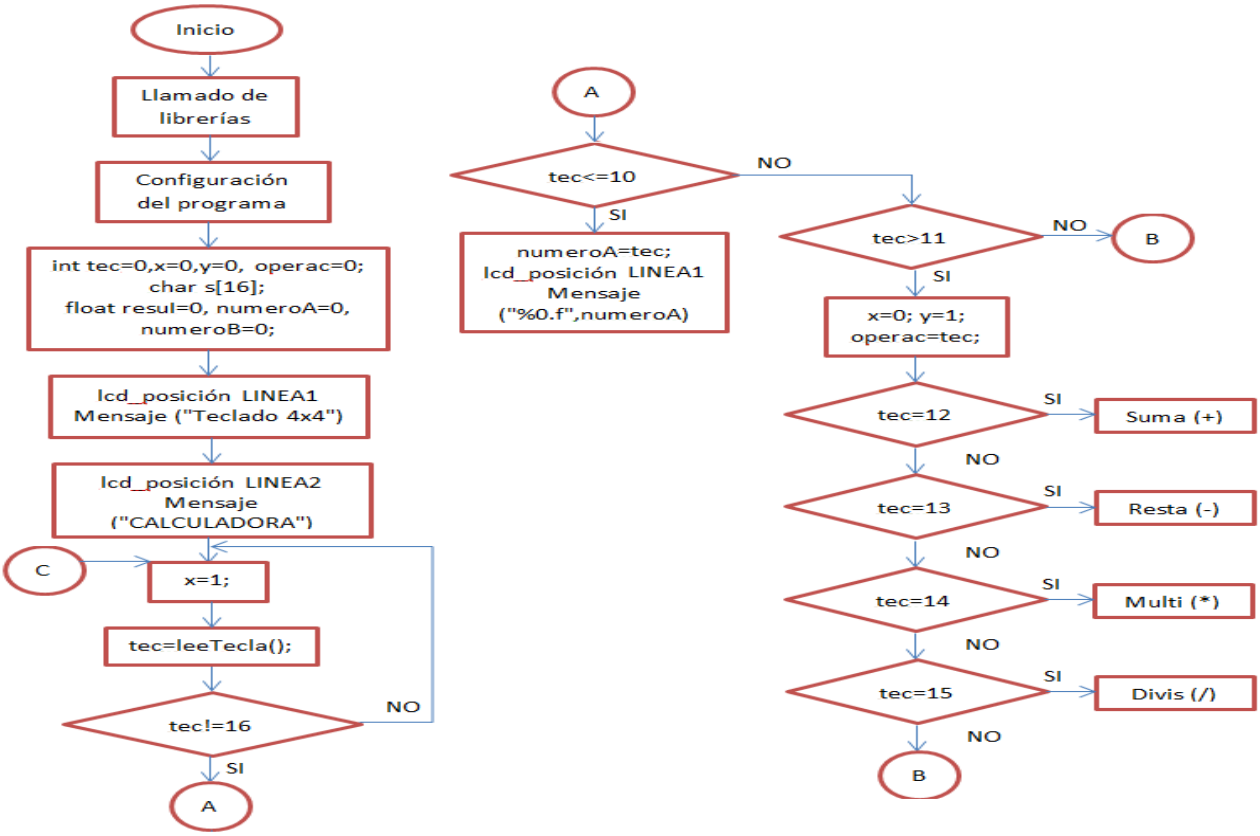
Posteriormente se definen las variables que serán utilizadas para almacenar los valores ingresados, estos valores deben ser de tipo flotante ya que se tendrá que mostrar decimales en el caso de que se realice una división no exacta.

Para inicializar el programa, se configura un mensaje a modo de título de la práctica: “Teclado 4x4” en la primer línea del LCD y “CALCULADORA” en la segunda línea del LCD, seguido de esto, se crea el proceso para el cálculo de cada operación.

El primer paso es crear una condición que identifique que el número ingresado sea menor al valor límite creado en la librería teclado.c, así como almacenar toda tecla menor a 10 en la variable *numeroA,* mostrada en el LCD.

Si se ingresa una tecla mayor a 11, ese valor será almacenado una variable llamada *operac*, es decir, si *operac* es igual a 12, significa que la operación que se requiere es una suma. Si *operac* es igual a 13, será resta la operación a realizar. De esta manera se programan las demás operaciones (multiplicación y división).

Seleccionada la operación, se ingresa el *numeroB,* también visualizado en el LCD. Contando con estos datos, se crea la condición principal: el resultado, la cual de esta se derivan las condiciones dependientes de cada operación. El resultado es mostrado en la segunda línea del LCD.



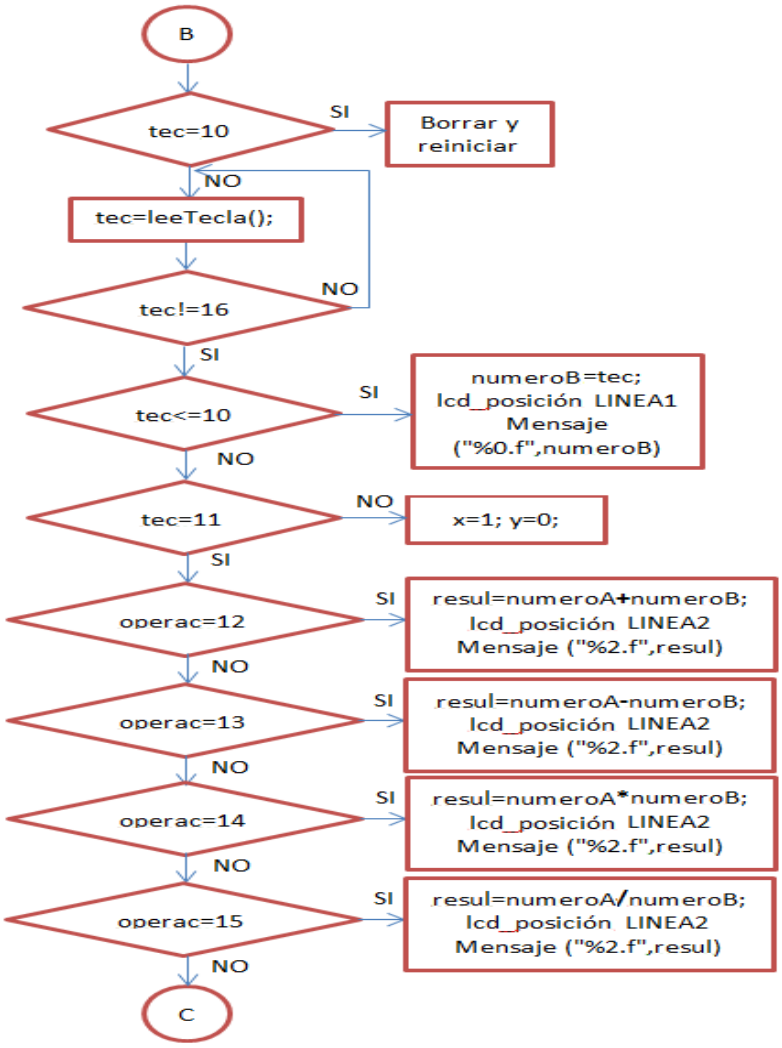


Fig. 53 Diagrama de flujo del Ejemplo 3.2

Calculadora.c

Código.

A continuación se presenta el código del programa del ejemplo 3.2 parte 2.

CALCULADORA.C

```
// Este programa realiza las operaciones básicas de una calculadora.

//Librerías necesarias para el DSPIC
#include "p30f4013.h"
#include "libpic30.h"
#include "teclado.h"

int tec=0,x=0,y=0,operac=0;
char s[16];
float resul=0,numeroA=0,numeroB=0;
```

```
int main()
{
      lcd_init();             //Inicializo LCD
      lcd_clear();                  //Limpio el LCD
      iniciaTeclado();              //Inicializa teclado

      lcd_goto(LINE1);              //posiciona en línea 1
      lcd_puts("Teclado 4x4");      //muestra mensaje
      lcd_goto(LINE2);              //posiciona en línea 2
      lcd_puts("CALCULADORA   ");//muestra mensaje
      lcd_clear();                  //limpia pantalla
      proceso();                    //comienza el método proceso
      setTeclado();                 //reinicia valores del puerto
}
void proceso(){
      while(1){
      x=1;              //estando X en 1 activa el ciclo while
            while(x){
                  tec=leeTecla();//iguala  en  tec  el  número  obtenido  del
teclado
                  if(tec!=16){      //si tec es diferente a 16
                        if(tec<=10){ //si tec es menor o igual a 10
                              numeroA=tec;//iguala en numeroA el valor de tec
                              lcd_goto(LINE1+0);//se muestra en LCD
                              lcd_puts(s);
                              sprintf(s,"%.0f ",numeroA);
                        }
                        if(tec>11){//si tec es mayor a 11
                              x=0; //x se desactiva
                              y=1; //y se activa para el siguiente while
                              operac=tec;//el valor de tecla
                              if (operac==12){//si es igual a 12
                                                //es suma
                                    lcd_goto(LINE1+1);
                                    lcd_puts("+");//imprime el signo
                              }
                              if (operac==13){ //si es igual a 13
                                                 //es resta
                                    lcd_goto(LINE1+1);
                                    lcd_puts("-");
                              }
                              if (operac==14){ //si es igual a 14
                                                 //es multiplicación
                                    lcd_goto(LINE1+1);
                                    lcd_puts("*");
                              }
                              if (operac==15){ //si es igual a 15
                                                 //es división
                                    lcd_goto(LINE1+1);
                                    lcd_puts("/");
                              }
                        }
                        if(tec==11){ //si es igual a 11 (=)
                              if(operac==12){//si es suma
                              resul=numeroA+numeroB;//operación
                              lcd_goto(LINE2);
                              lcd_puts(s);
                              sprintf(s,"=%.0f ",resul);
                              }
                              if(operac==13){//si es resta
                              resul=numeroA-numeroB;//operación
                              lcd_goto(LINE2);
                              lcd_puts(s);
                              sprintf(s,"=%.0f ",resul);}
```

```
                        if(operac==14){//si es multiplicación
                        resul=numeroA*numeroB;//operación
                        lcd_goto(LINE2);
                        lcd_puts(s);
                        sprintf(s,"=%.0f ",resul);
                        }
                        if(operac==15){//si es división
                              if(numeroB!=0){//si    el    numeroB    es
diferente a 0 puede realizar la operación
                                    resul=numeroA/numeroB;
                                    lcd_goto(LINE2);
                                    lcd_puts(s);
                                    sprintf(s,"=%.2f ",resul);
                              }if(numeroB==0){   ){//si   el   numeroB   es
igual a 0 mostrará “Error”
                                    lcd_goto(LINE2);
                                    lcd_puts("Error");
                              }
                        }
                  }
                  if(tec==10){ //si es igual a 10
                        setTeclado();//reinicia valores
                        lcd_clear();//limpia LCD
                  }
            }
      }while(y){
            tec=leeTecla();
            if(tec!=16){
                  if(tec<=10){ //si tec es menor o igual a 10
                        numeroB=tec;//iguala en numeroB el valor de tec
                        lcd_goto(LINE1+2);
                        lcd_puts(s);
                        sprintf(s,"%.0f ",numeroB);
                  }
                  if(tec==11){
                        if(operac==12){
                        resul=numeroA+numeroB;
                        lcd_goto(LINE2);
                        lcd_puts(s);
                        sprintf(s,"=%.0f ",resul);
                        }
                        if(operac==13){
                        resul=numeroA-numeroB;
                        lcd_goto(LINE2);
                        lcd_puts(s);
                        sprintf(s,"=%.0f ",resul);
                        }
                        if(operac==14){
                        resul=numeroA*numeroB;
                        lcd_goto(LINE2);
                        lcd_puts(s);
                        sprintf(s,"=%.0f ",resul);
                        }
                        if(operac==15){
                              if(numeroB!=0){
                                    resul=numeroA/numeroB;
                                    lcd_goto(LINE2);
                                    lcd_puts(s);
                                    sprintf(s,"=%.2f ",resul);
                              }if(numeroB==0){
                                    lcd_goto(LINE2);
                                    lcd_puts("Error");
                              }
                        }
```

```
                                x=1;
                                y=0;
                        }
                        if(tec==10){
                                setTeclado();
                                lcd_clear();
                        }
                }
        }
}
```

La simulación se llevó a cabo en MPLAB, donde se observa que el resultado corresponde a lo planteado con anterioridad.

Ensamblaje y prueba.

Finalmente se demostró (Fig. 54-58) que el programa junto con el diseño del circuito, funciona correctamente.

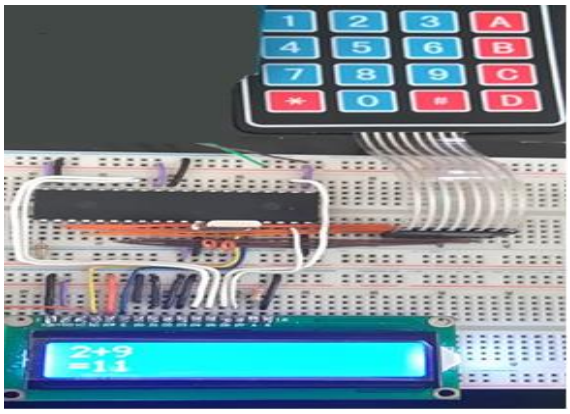


Fig. 54 Suma.

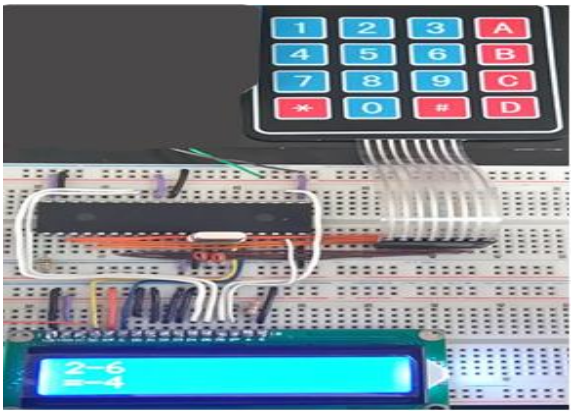


Fig. 55 Resta.

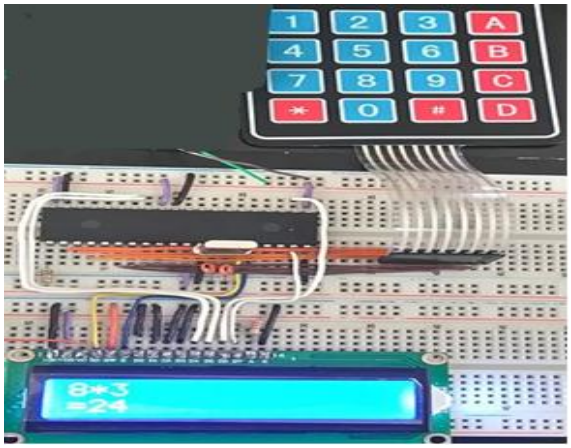


Fig. 56 Multiplicación.

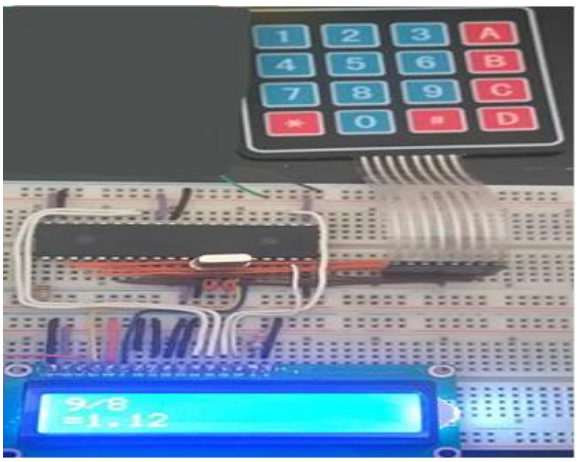


Fig. 57 División.

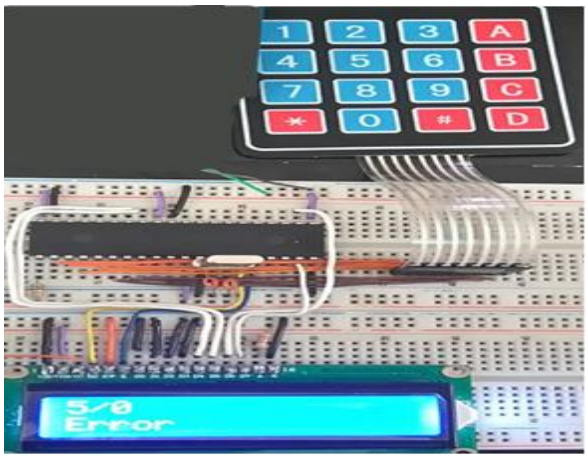


Fig. 58 División entre cero.

4. A continuación se realizarán algunos ejemplos con el ADC.

Ejemplo 4.1

Objetivo

Diseñar y desarrollar un programa utilizando el ADC interno del dsPIC30F4013 que realice la conversión de señal analógica en señal digital. Donde el voltaje ingresado por el puerto B defina el tiempo de visualización de la salida por puerto F.

Desarrollo.

Para realizar una conversión, se deben seguir los siguientes pasos:

1. Configurar las patitas analógicas, las tensiones de referencia y las entradas y salidas digitales.
2. Seleccionar los canales de entrada A/D.
3. Seleccionar el reloj para la conversión.
4. Seleccionar el disparo (trigger) de la conversión.
5. Poner en marcha el módulo A/D.
6. Configurar la interrupción (si fuera necesario).
    a. Limpiar el bit ADIF.
    b. Seleccionar la prioridad de la interrupción.
    c. Activar el bit ADIE.
7. Comenzar el muestreo.
8. Comenzar la conversión.
9. Esperar a que se complete la conversión, lo que se detecta por una interrupción o porque el bit DONE del registro ADCON1 se pone a 1.
10. Leer el buffer A/D con el resultado (ADCBUFO, ADCBUF1,...) y limpiar el bit ADIF.

Diagrama de flujo.

En la Fig. 59 se presenta el diagrama de flujo del programa desarrollado para el ejemplo 4.1. Como se puede observar al inicio del diagrama de flujo se configura el

programa, mediante el llamado de las librerías y los encabezados. Una de las librerías a utilizar en este programa es la función Delay, a través de la cual se definirá el tiempo de retardo del voltaje ingresado. Posteriormente se configura el puerto B como entrada y el puerto F como salida. Hecho lo anterior se especifican los puertos analógicos a utilizar. Se desactiva el ADC, así como los canales A/D entrada, el tiempo del reloj y buffer 2. El ADC del bit 1 permanecerá activado.

Configurada la conversión, se procede a realizar un ciclo indefinido donde se realiza el muestreo del ADC activado por espacios de tiempos, deteniendo el muestreo para realizar la conversión. Si la conversión es correcta, se almacena lo que se encuentra en el buffer en una variable de tipo entero. Esta variable será lo que defina el tiempo de retardo cada que se modifique en voltaje de entrada. Todo esto visualizado en LED's por el puerto F.

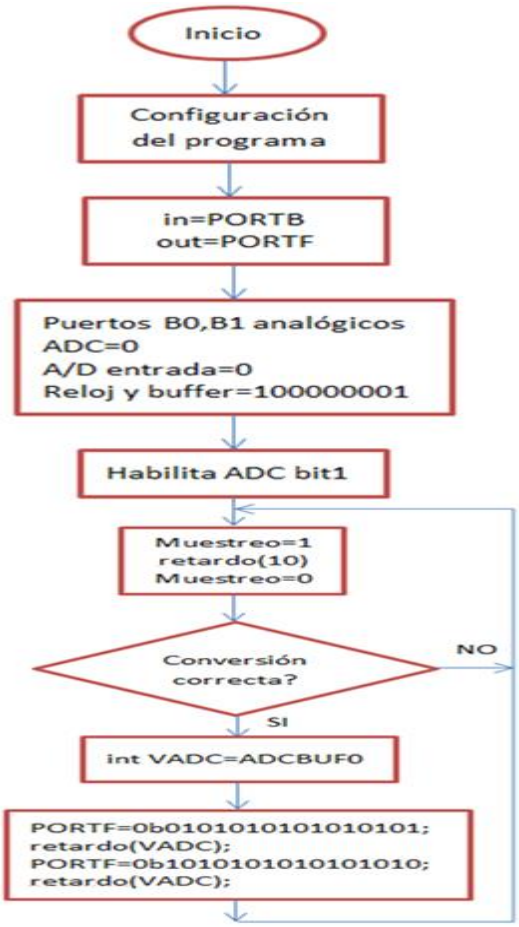


Fig. 59 Diagrama de flujo del Ejemplo 4.1

Código.

A continuación se presenta el código del programa del ejemplo 4.1, donde el color azul representa palabras reservadas, identificadores o tipos de datos. El verde indica que son comentarios que definen el uso de cada sentencia. Y el color negro son los bloques de código.

```
#include "p30f4013.h" //incluye la cabecera del dispositivo a usar
#include "libpic30.h" //librería del pic30 para procesamiento de señales
#include "delay_mio.h"//librería Delay

//Configuración
_FOSC (CSW_FSCM_OFF & XT);
_FWDT (WDT_OFF);
_FBORPOR (PBOR_ON & MCLR_EN);

#ifndef __DELAY_H
      #define delay_us(x) __delay32(((x*FCY)/100000L))//delay x us
      #define delay_ms(x) __delay32(((x*FCY)/10L)) //delay x ms
      #define __DELAY_H 1
#endif

int main()
{
      TRISB =0XFFFF;          //Entrada PORTB
      TRISF = 0;              //Salida PORTF
      ADPCFG=0XFFFC;          //  Puertos  B0,  B1  como  analógicos,  lo  demás
digital
      //configuración general del ADC
      ADCON1=0;               //ADC desactivado
      //configuración canales A/D entrada
      ADCHS=0;                //CH0NA=0
      ADCSSL=0;               //SCAN desactivado para todos los canales
      //Tiempos
      ADCON3=0x0101;          //1 TAD y reloj desactivado del sistema
      //Buffers
      ADCON2=0x0101;

      ADCON1bits.ADON = 1;    //habilita ADC

      while(1){
            ADCON1bits.SAMP = 1; //Iniciar muestreo
            delay_ms(10);           // Tiempo para realizar el muestreo
            ADCON1bits.SAMP = 0; //Detener muestreo, iniciar conversión
            while(!ADCON1bits.DONE); //Se ha realizado la conversión?
                  int VADC=ADCBUF0; //Variable  que  almacena  el  valor  de  la
conversión
                  PORTF=0b0101010101010101; //LED encendidos
                  delay_ms(VADC);   //Tiempo de retardo
                  PORTF=0b1010101010101010;
                  delay_ms(VADC);
      }
      return 0;
}
```

La simulación por compilación se llevó a cabo en MPLAB y se observa que el resultado corresponde a lo planteado con anterioridad.

Diagrama circuital.

En la Fig. 60 se muestra el diagrama del circuito a implemntar. En el se puede observar la conexión del dsPIC30F4013 donde el pin 1 está conectado a una resistencia de 10kΩ que va directamente al voltaje. En los pines 2 y 3, como lo indica la hoja de especificaciones del DSPIC, son los voltajes de referencia, es decir, los valores de voltaje máximo (V_{REF+}) y mínimo (V_{REF-}) que se medirán. Es decir, a través del pin 2 entra la señal analógica a convertir. Y el pin 3 se conecta a tierra. En este caso, se agrega un potenciómetro para tener mayor estabilidad en la variación de voltaje.

El oscilador de cristal con sus respetivos capacitores se posicionan en los pines 13 y 14. Además que es importante mantener alimentado todo el DSPIC para su correcto funcionamiento.

Finalmente por el puerto F (del pin 25 al pin 30), se conectan resistencias de 330Ω que se conectan a LED's para visualizar el resultado.

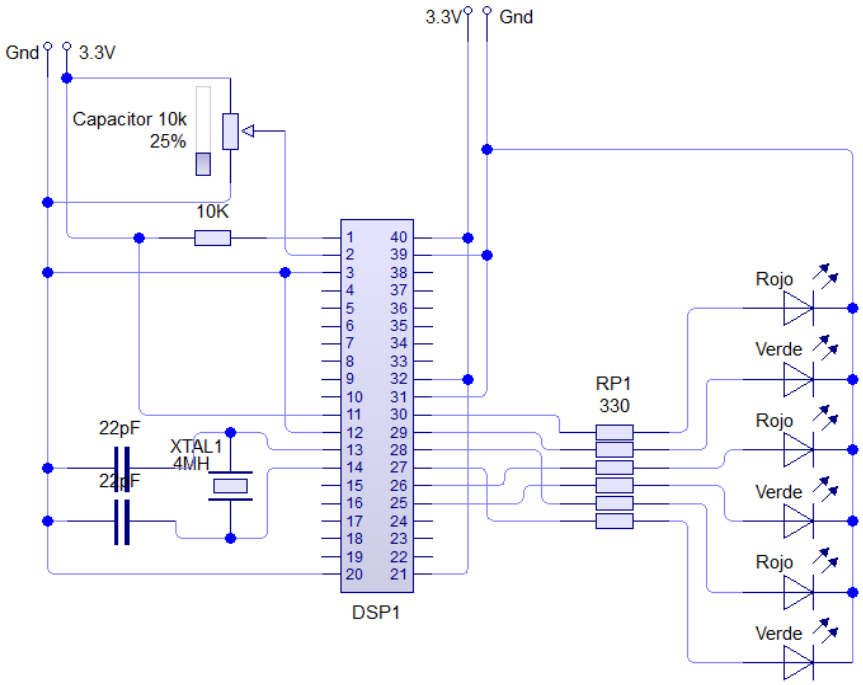


Fig. 60 Diagrama circuital.

Ensamblaje y prueba.

Finalmente se demostró (Fig. 61 y 62) que el programa junto con el diseño del circuito, funciona correctamente.

En la figura 61 se observa que a mayor voltaje, el retardo de encendido de los LED verdes es mayor.

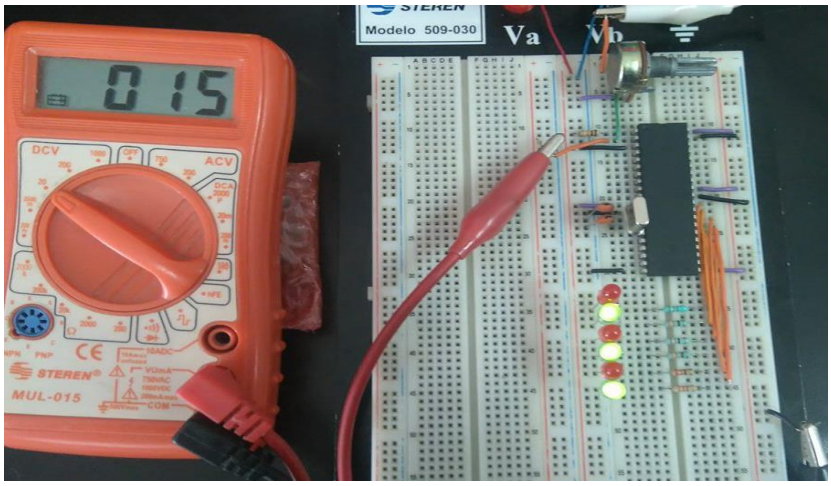


Fig. 61 Ensamblaje y comprobación.

Así como entre menor voltaje reciba el DSPIC, el cambio entre los LED es más rápido.

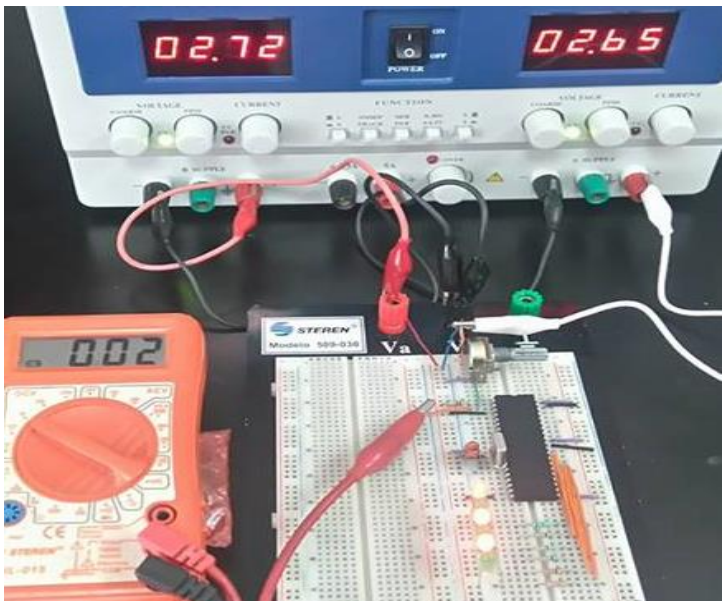


Fig. 62 Medición del voltaje ingresado.

Ejemplo 4.2

Objetivo

Diseñar y desarrollar un voltímetro digital, utilizando el ADC interno del dsPIC30F4013 y display de siete segmentos.

Desarrollo.

El primer paso es realizar la conversión analógico-digital como se muestra en el ejemplo 4.1. Además, es necesario crear dos arreglos, uno para cada display a utilizar. Los arreglos permiten convertir del número binario obtenido del ADC a un código BCD de siete segmentos para su despliegue. Se utilizan dos arreglos porque los display serán colocados en diferente puerto, uno de ellos utilizará una parte del puerto B y el otro empleará el puerto F. Mediante el primer y segundo display se presentarán los valores enteros y decimales, respectivamente.

El valor obtenido de la conversión analógico-digital, es almacenado en una variable llamada *i*. Este valor es utilizado para calcular el voltaje ingresado en el dsPIC y mostrarlo en los displays ya configurados.

Operaciones.

Un ADC de n bits puede representar hasta 2^n valores digitales, de modo que para la realización de éste ejemplo, a la entrada analógica se le asigna un valor en V_{REF-} de 0 v y en V_{REF+} 3.5 v con resolución de 10 bits, es decir:

$$2^{10} = 1024$$

Siendo así, se puede obtener un valor aproximado de *i*, dónde el voltaje ingresado lo llamaremos *j*.

Ahora, suponiendo que si *j =1*, ¿cuál será el valor de *i* ?

$$\mathbf{1} = (\mathbf{10} * \boldsymbol{i})/\mathbf{1024}$$

$$1024 = (10 * i)$$

$$\frac{1024}{10} = \boldsymbol{i}$$

$$\boldsymbol{i} = 102.4$$

Con **n** bits, solo se pueden formar 2^n valores discretos diferentes. Por lo tanto habrá valores analógicos que no podrán ser representados con exactitud (error de cuantización). Por lo tanto, para obtener un valor exacto, se modifica levemente 1024, cambiándolo por 1050. Éste nuevo valor se considera el más preciso, ya que se obtuvo a base de las diversas pruebas realizadas. Se repite la operación anterior para comprobar que el valor de *i* sea exacto.

$$\mathbf{1} = (\mathbf{10} * \boldsymbol{i})/\mathbf{1050}$$

$$1050 = (10 * i)$$

$$\frac{1050}{10} = \boldsymbol{i}$$

$$\boldsymbol{i} = \mathbf{105}$$

Con este nuevo valor, podemos declarar la siguiente fórmula para calcular los primeros decimales del voltaje.

$$j = (10 * i)/1050$$

Dónde *j* será el número que represente el voltaje ingresado. Hecho este proceso, se asegura mayor exactitud en la medición.

Para calcular los enteros se requiere del último valor en *j*, para que cuando termine de medirse los decimales, entre el primer valor entero. Entonces:

$$k = j/10$$

Dónde *k* representa el voltaje ingresado (enteros). Y *j* es dividido entre 10 para obtener sólo los decimales.

Realizado el análisis anterior se procede con el desarrollo del diagrama de flujo.

Diagrama de flujo.

En la Fig. 63 se presenta el diagrama de flujo del programa desarrollado para el ejemplo 4.2. Como se puede observar al inicio del diagrama de flujo se configura el programa, mediante el llamado de las librerías y los encabezados. Posteriormente se declaran los arreglos para cada display, se configuran los puertos B y F como salida. Así mismo, se definen las variables *i*, *j* y *k*. Hecho lo anterior se especifican los puertos analógicos a utilizar. En el caso del puerto B, las entradas serán solamente B0, B1. Se desactiva el ADC, así como los canales A/D entrada, el tiempo del reloj y buffer 2. El ADC del bit 1 permanecerá activado.

Configurada la conversión, se procede a realizar un ciclo indefinido donde se realiza el muestreo del ADC activado por espacios de tiempos, deteniendo el muestreo para realizar la conversión. Si la conversión es correcta, se almacena lo que se encuentra en el buffer en una variable de tipo entero llamada *i*. Esta variable es utilizada para almacenar el voltaje ingresado en el DSPIC y además, es utilizada para realizar los cálculos descritos anteriormente.

Se realiza la primera operación para obtener los decimales (del 0 al 9). Si *j* es menor o igual a 9 se muestra el valor por el display del puerto F, manteniendo en 0 el display del puerto B. Si *j* es mayor a 9, se realiza la segunda operación para obtener a los enteros (*k*), mostrando el valor en el display del puerto B. En caso de que *k* sea menor a dos, se resta el valor de *j* menos diez; si el caso es que *k* es menor a tres, se resta *j* menos veinte y finalmente, si el caso es que *k* es menor a cuatro, se resta *j* menos treinta.

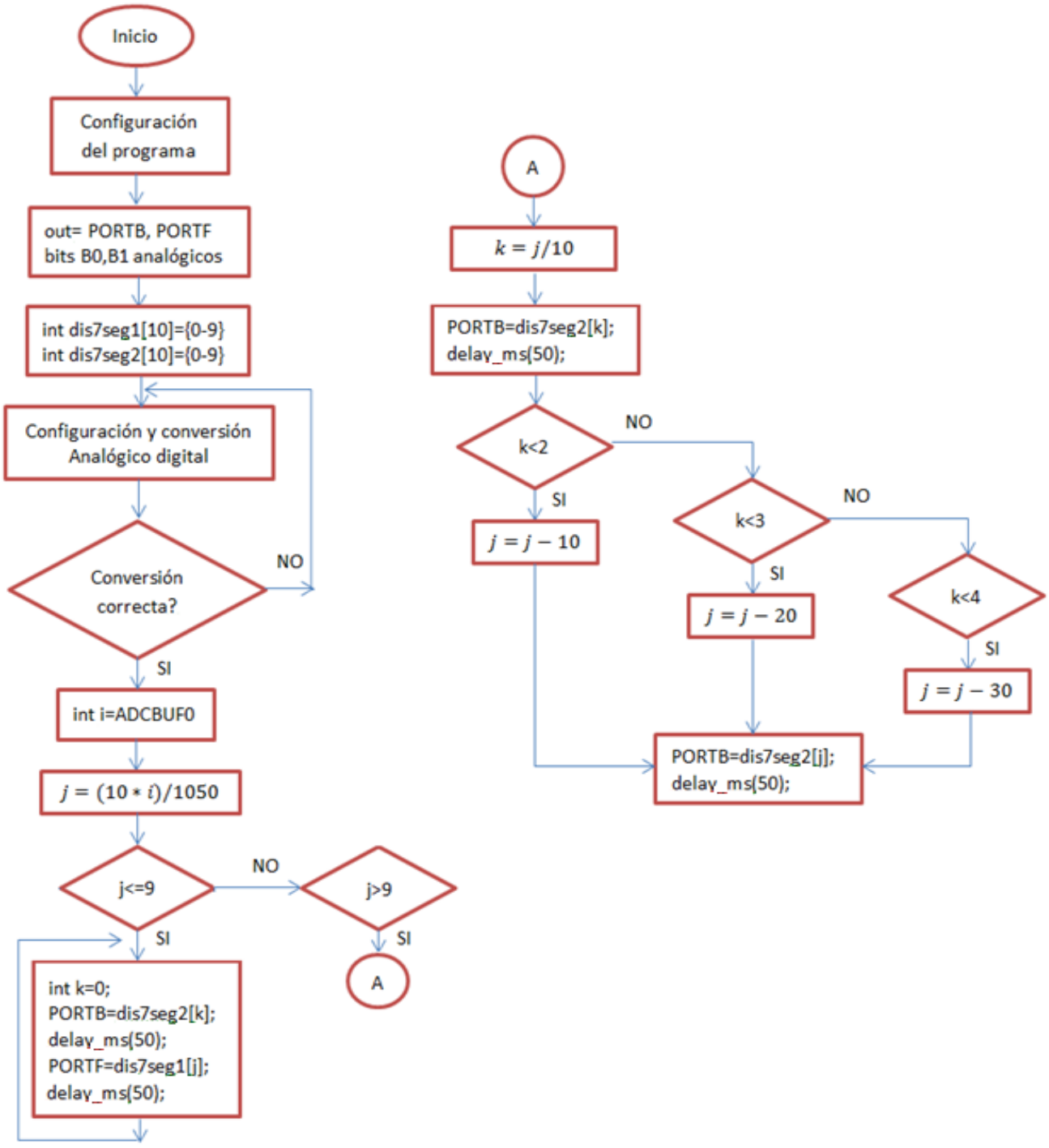


Fig. 63 Diagrama de flujo del Ejemplo 4.2

Código.

A continuación se presenta el código del programa del ejemplo 4.2, donde los colores indican lo mismo que en el ejemplo 4.1.

```
#include "p30f4013.h" //incluye la cabecera del dispositivo a usar
#include "libpic30.h" //librería del pic30 para procesamiento de señales
#include "delay_mio.h"//librería Delay

//Configuración
_FOSC (CSW_FSCM_OFF & XT);
_FWDT (WDT_OFF);
_FBORPOR (PBOR_ON & MCLR_EN);

#ifndef __DELAY_H
      #define delay_us(x) __delay32(((x*FCY)/100000L))//delay x us
      #define delay_ms(x) __delay32(((x*FCY)/10L)) //delay x ms
      #define __DELAY_H 1
#endif

int dis7seg1[10]= {0x7E,0x30,0x6D,0x79,0x33,0x5B,0x5F,0x70,0x7F,0x73}; //Tabla
con los datos correspondientes al número a visualizar. Decimales
int dis7seg2[10]= {0x1F8,0xC0,0x1B4,0x1E4,0xCC,0x16C,0x17C,0x1C0,0x1FC,0x1CC};
//Tabla con los datos correspondientes al número a visualizar. Enteros


int main()
{
      PORTB=0;                  //Puerto B salida
      PORTF=0;                  //Puerto F salida
      int i=0,j=0,k=0;          //Declaración de variables

      TRISB =0X3;         //Entrada B0, B1, los demás como salida
      TRISF = 0;                //Salida
      ADPCFG=0XFFFC;            //Puertos B0, B1 como analógicos, lo demás en
digital
      //configuración general del ADC
      ADCON1=0;                 //ADC desactivado
      //configuración canales A/D entrada
      ADCHS=0;                  //CH0NA=0
      ADCSSL=0;                 //SCAN desactivado para todos los canales
      //Tiempos
      ADCON3=0x0101;            //1 TAD y reloj desactivado del sistema
      //Buffers
      ADCON2=0x0101;

      ADCON1bits.ADON = 1;      //habilita ADC

      while(1){
            ADCON1bits.SAMP = 1; //Iniciar muestreo
            delay_ms(10);       // Tiempo para realizar el muestreo
            ADCON1bits.SAMP = 0; //Detener muestreo, iniciar conversión
            while(!ADCON1bits.DONE); //Se ha realizado la conversión?
                  i=ADCBUF0;          //almacena el valor convertido
                  j=(10*i)/1050;      //Operación  para  obtener  los  primeros
decimales

                  if(j<=9){   //condición para primeros 9 decimales
                        k=0;  //el valor de enteros se mantiene…
                        PORTB=dis7seg2[k]; //…en ceros
                        delay_ms(50);
```

```
                        PORTF=dis7seg1[j]; //muestra el valor de cada decimal
por el puerto F
                        delay_ms(50); //mantiene el valor visualizado
                  }

                  if(j>9){    //condición para enteros
                        k=j/10; //operación para obtener los enteros
                        PORTB=dis7seg2[k]; //muestra el valor de cada decimal
por el puerto B
                        delay_ms(50); //mantiene el valor visualizado
                        //decimales
                        if(k<2){ //si K es menor a dos
                              j=j-10; //resta el valor de j con 10
                              PORTF=dis7seg1[j];//resultado por puerto F
                              delay_ms(50); //mantiene el valor visualizado
                        }else //si no,
                        if(k<3){ //si K es menor a tres
                              j=j-20; //resta el valor de j con 20
                              PORTF=dis7seg1[j];
                              delay_ms(50);
                        }else //si no,
                        if(k<4){ //si K es menor a cuatro
                              j=j-30; //resta el valor de j con 30
                              PORTF=dis7seg1[j];
                              delay_ms(50);
                        }
                  }
      } }
```

La simulación por compilación se llevó a cabo en MPLAB, donde se observa que el resultado corresponde a lo planteado con anterioridad.

Diagrama circuital.

En la Fig. 64 se muestra el diagrama del circuito a implemntar. En el se puede observar la conexión del dsPIC30F4013 donde el pin 1 está conectado a una resistencia de 10kΩ que va directamente al voltaje. En los pines 2 y 3, como lo indica la hoja de especificaciones del DSPIC, son los voltajes de referencia, es decir, los valores de voltaje máximo (V_{REF+}) y mínimo (V_{REF-}) que se medirán. Es decir, a través del pin 2 entra la señal analógica a convertir. Y el pin 3 se conecta a tierra. En este caso, se agrega un potenciómetro para tener mayor estabilidad en la variación de voltaje.

El oscilador de cristal con sus respetivos capacitores se posicionan en los pines 13 y 14. Además que es importante mantener alimentado todo el DSPIC para su correcto funcionamiento.

El primer display se conecta a partir de RB2 (pin 4), de manera descendente, es decir, en RB2 está el segmento G del display, hasta llegar al segmento A en el pin 10. El

segundo display se conecta en el puerto F, a partir del pin 30, dónde el segmento G está conectado en RF0 y el segmento A en RF6 (pin 24).

Ambos displays son cátodo común, así que cada uno lleva una resistencia 330Ω como minimo valor, que a su vez son conectados a tierra. El punto decimal del primer display permanece encendido conectando su segmento con una resistencia al voltaje.

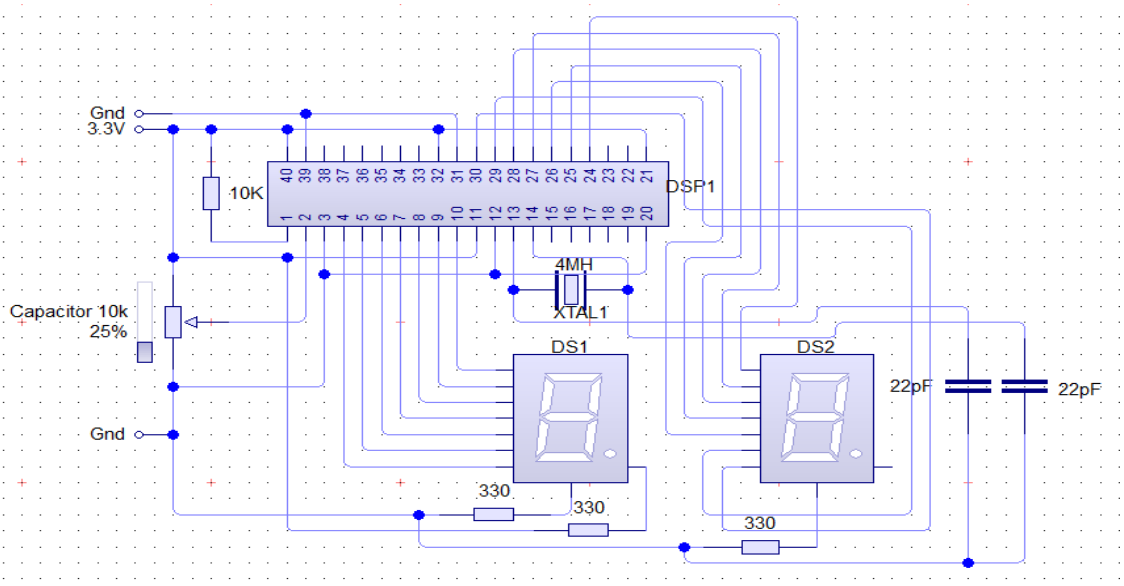


Fig. 64 Diagrama circuital.

Ensamblaje y prueba.

Finalmente se demostró (Fig.65-68) que el programa junto con el diseño del circuito, funciona correctamente.

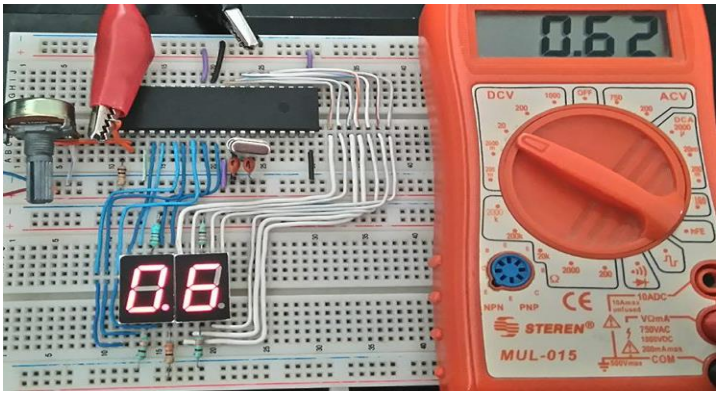


Fig. 65 Prueba de medición (1)

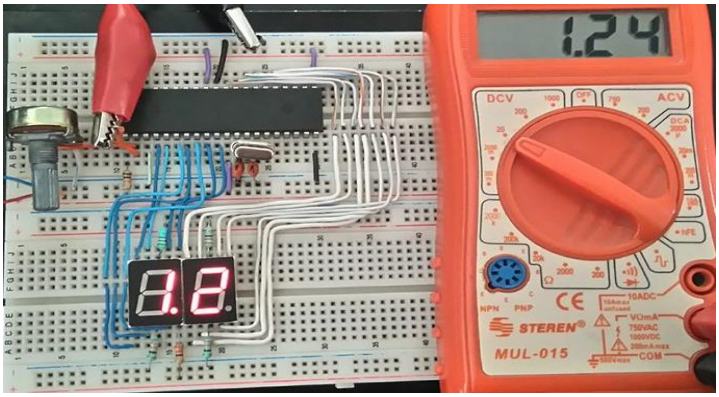


Fig. 66 Prueba de medición (2)

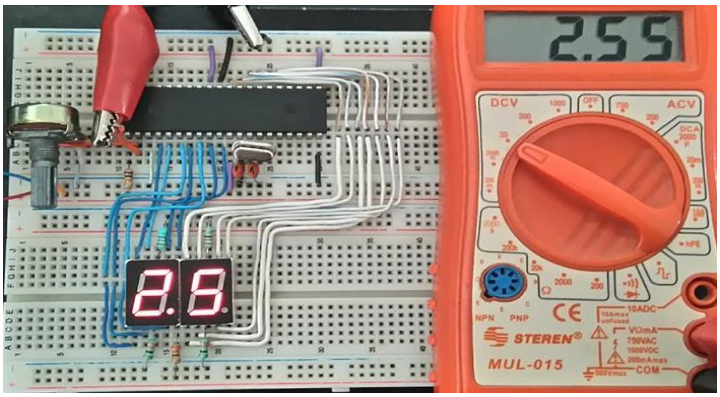


Fig. 67 Prueba de medición (3)

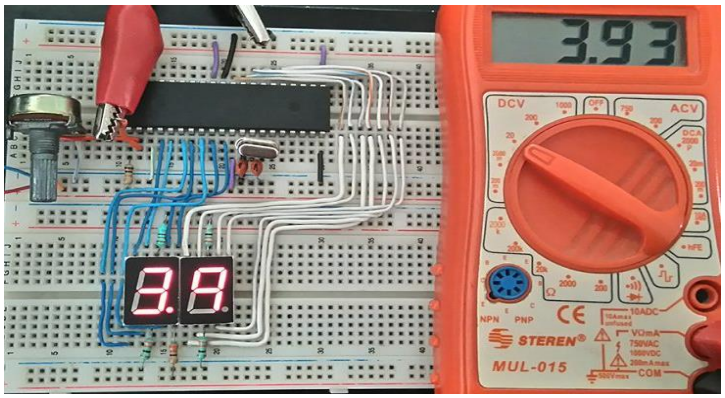


Fig. 68 Prueba de medición (4)

Ejemplo 4.3

Objetivo

Utilizando el voltímetro digital de la práctica anterior, medir la iluminación ambiental.

Desarrollo.

Para obtener el voltaje de la iluminación ambiental es necesario hacer uso de una fotorresistencia combinada con un amplificador operacional que trabaje de manera inversa.

El DSPIC trabaja con el mismo programa de la práctica anterior, es decir que el diagrama de flujo como el código, se pueden encontrar en el Ejemplo 4.2

Diagrama circuital.

En la Fig. 69 se muestra el diagrama del circuito a implemntar. En el se puede observar la conexión del op-amp, donde es alimentado con +9v en la terminal 7 y -9v en la 4. La entrada no inversora (pin 3) es conectada a tierra. En la entrada inversora (pin 2) se encuentra conectada la fotoresistencia de hasta 2.0MΩ, siendo alimentada por una resistencia de 2.2kΩ que va directamente a la fuente. En esa misma entrada del pin 2 se conecta con el mismo valor de 2.2kΩ la resistencia de entrada (Ri) y la resistencia de retroalimentación (Rf) para obtener una ganancia de 1.

A la salida del op-amp (pin 6) se interconecta la resistencia Rf. Además se conecta un LED para visualizar el cambio de la iluminación. De la misma salida se conecta la misma configuración para un segundo op-amp, utilizando resistencias de 1kΩ para Ri y Rf. Finalmente, a la salida del segundo amplificador se conecta un LED para visualizar el cambio, además que ésta es la señal que será introducida en el pin 2 del dsPIC30F4013.

La interconexión del DSPIC es la misma utilizada en el Ejemplo 4.2

Nota: Tanto los amplificadores operacionales como la conexión del dsPIC30F4013, deben estar conectados a una tierra común, ya que éste se toma como el punto de referencia para ambas conexiones.

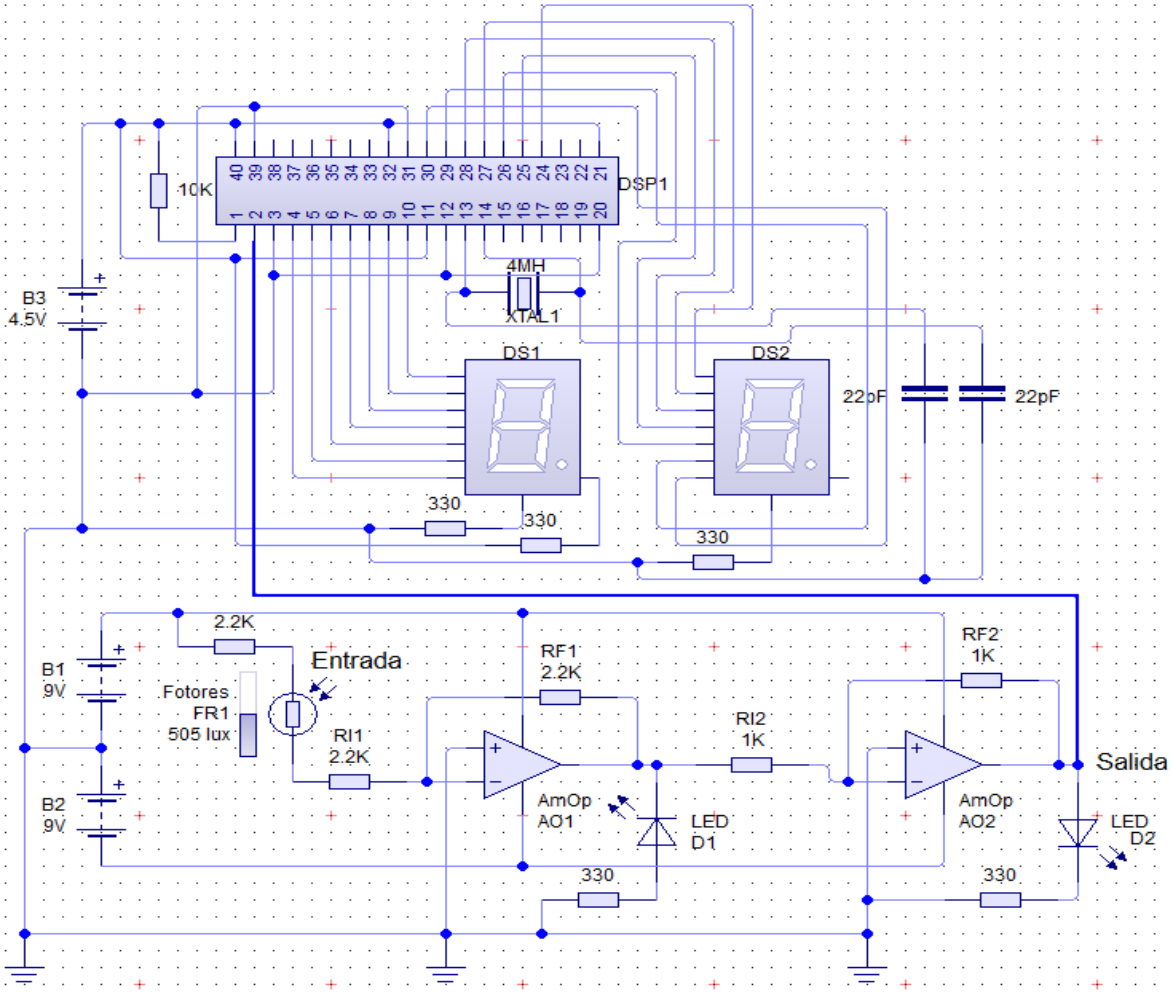


Fig. 69 Diagrama circuital

Ensamblaje y prueba.

Finalmente se demostró (Fig. 70-72) que el programa junto con el diseño del circuito, funciona correctamente.

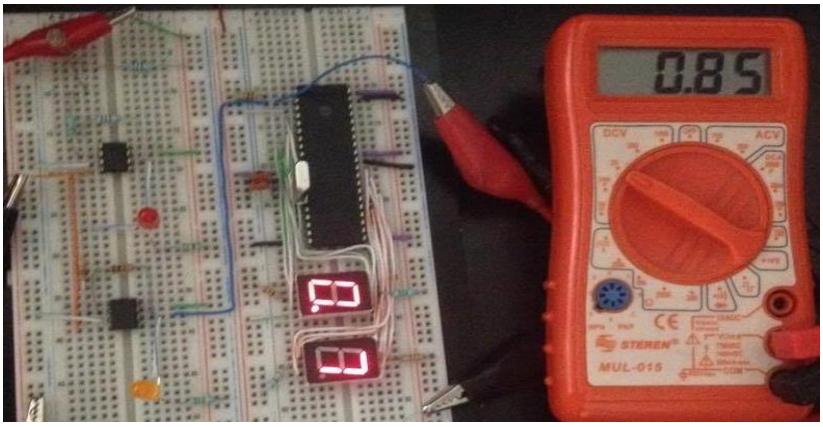


Fig. 70 Prueba 1

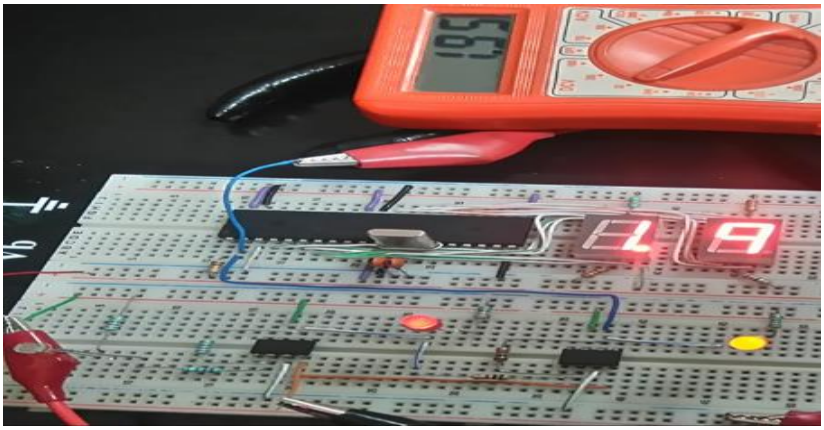


Fig. 71 Prueba 2

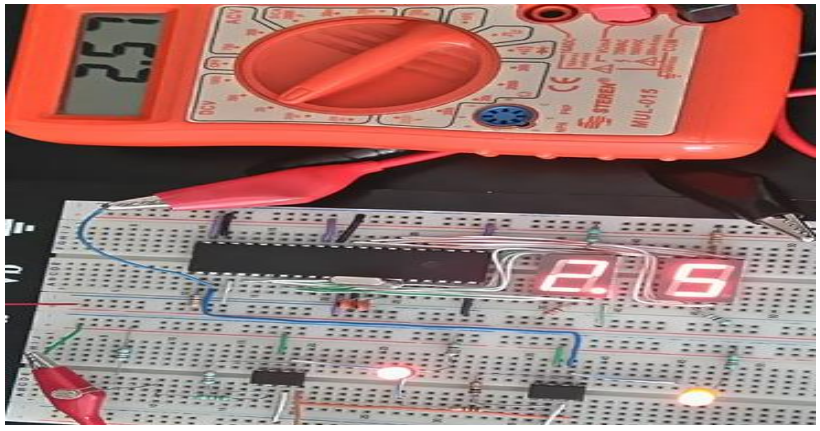


Fig. 72 Prueba 3

CAPÍTULO 2. MARCO DE REFERENCIA

El cáncer de piel ocupa el segundo lugar más frecuente en México, hoy, gracias a la mayor exposición que se tiene al sol y la costumbre de asistir a camas de bronceado, es posible observar la enfermedad desde los 18 años de edad, con los grandes costos que esto implica.

Datos de la Organización Mundial de la Salud (OMS) señalan que el cáncer de piel es el más frecuente en los seres humanos, pues cada año se diagnostican dos millones de nuevos casos en todo el mundo. En México ocupa el primer lugar de incidencia entre hombres y el tercero entre mujeres.

En el sector privado el costo de tratamiento para cáncer de piel tipo Melanoma, puede costar hasta 500 mil pesos al año, pero si se diagnostica a tiempo, el costo se reduce entre 10 y 50 mil pesos [13].

Actualmente existen diversos instrumentos de detección y medición UV en el marcado, la mayoría conocidos como radiómetros.

Los radiómetros se han diseñado para mediciones precisas de la radiación ultravioleta de la atmósfera para tres diferentes rangos espectrales. Todos los modelos miden la radiación UV global [16,17].

PCE-EMF 823

El medidor de radiación PCE-EMF 823 (Fig. 73) está enfocado en la radiación electromagnética para determinar la radiación en Tesla o micro Gauss. El medidor de radiación electromagnetica PCE-EMF823 ha sido especialmente concebido para medir radiaciones electromagnéticas emitidas por aparatos eléctricos como televisores, lámparas, ordenadores, conductores de corriente, pantallas e instalaciones eléctricas industriales. Su costo aproximado es de $ 2,134.65 MXN.

Fig. 73 Radiómetro PCE-EMF823

Radiómetro UVA - UVB PCE-UV34

El medidor de radiación UVA - UVB PCE-UV34 (Fig. 74) es un aparato para la medición de la radiación ultravioleta. Este medidor detecta longitudes de onda de entre 290 a 390 nm. Entre las aplicaciones de este tipo de sensores se encuentran las cabinas de bronceado para detectar cuándo los niveles de radiación UVA pueden comenzar a deteriorar la salud de los usuarios. Este instrumento es de fácil uso debido a que cuenta con detector externo. Su costo aproximado es de $3,571.40 MXN.

Fig. 74 Radiómetro PCE-UV34

TPM-SPO3414F

Monitorea radiación crítica de UV. Este medidor (Fig. 75) portátil mide el flujo de fotones en el UV en un rango de longitud de onda de 250 a 400 nm en unidades micromoles de fotones por metro cuadrado por segundo. Ideal para las plantas. Determina la capacidad filtrante de UV de la cubierta de los invernaderos o de barreras de cristal. Incluye una batería de 9V. Su costo aproximado es de $4999.00 MXN.

Fig. 75 Radiómetro TPM-SPO3414F

TPM-UV-340B

El Radiómetro TP-UV-340B (Fig. 76) utiliza un sensor exclusivo que filtra la radiación ultravioleta mostrando solamente la UV-A y UV-B. Puede ser utilizado para monitoreo de radiación en diferentes procesos tales como: esterilización por UV, borrado de UV EPROM, curación de enlaces, adhesivos y recubrimientos, etc. Su costo aproximado es de $6,080.94 MXN.

Fig. 76 Radiómetro TPM-UV-340B

Radiómetro UV Magnaflux

El radiómetro UV Magnaflux (Figura 77) es capaz de medir rayos UV-A con un diseño compacto y robusto, fácil de operar. El medidor de UV multifuncional ha demostrado ser muy eficaz para medir y calibrar las fuentes de luz ultravioleta utilizada en partículas magnéticas fluorescente y líquidos penetrantes. El sistema está integrado por un sensor remoto desmontable, pantalla digital LCD de alto contraste, batería de 9V y estuche protector. Su costo aproximado es de $7,587.90 MXN.

Fig. 77 Radiómetro UV Magnaflux

CAPÍTULO 3. METODOLOGÍA

Para el avance del proyecto de investigación, se utilizó como estructura principal la metodología en cascada para el desarrollo del prototipo. Como el proyecto tiene componentes importantes de desarrollo de hardware y software, fue necesario hacer algunas ligeras adecuaciones enfocadas a contemplar estas características.

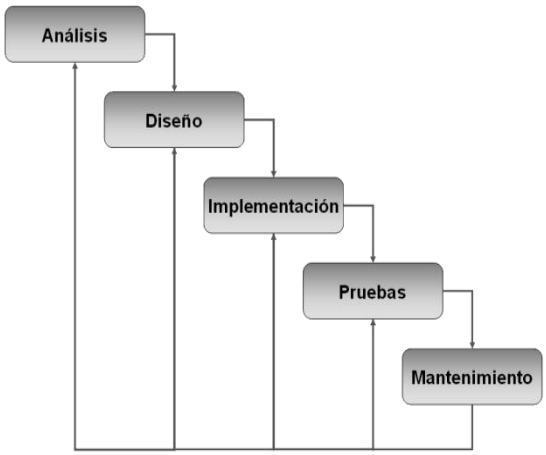


Fig. 78 Metodología en cascada.

En las siguientes secciones se describen los pasos realizados en el desarrollo del proyecto.

CAPÍTULO 3.1 ANÁLISIS DE REQUERIMIENTOS

Los requerimientos establecidos se describen a continuación:

Requerimientos funcionales.

Las señales analógicas a medir son capturadas por medio de un:

1. Detector de:
    a. Iluminación; medido a través de voltaje de manera concisa la luminosidad general del ambiente.
    b. Temperatura; con un rango que abarca desde los 0° hasta 150°C.
    c. Ruido; evaluado por voltaje, capturando el ruido ambiental por medio de un micrófono de uso genérico.

Despliegue de información

2. El Display LCD muestra el valor de las variables.

Señales de alerta, que indican por medio de colores el nivel en que se encuentra la señal detectada.

3. Variables fuera de rango
    a. Verde
    b. Amarillo
    c. Rojo

Requerimientos no funcionales.

Conversión ADC

4. DSPIC
    a. Entradas analógicas
    b. Salidas digitales

Lenguaje de programación

5. Desarrollado en lenguaje C
    a. Herramienta MPLAB

CAPÍTULO 3.2 DISEÑO DEL SISTEMA

En la Fig. 79 se presenta el diagrama de bloques del sistema a desarrollar. Con base en la función de cada uno de los módulos, estos se agruparon en cinco bloques como se muestran en el diagrama. En las secciones posteriores se explicará más a detalle la funcionabilidad de cada bloque.

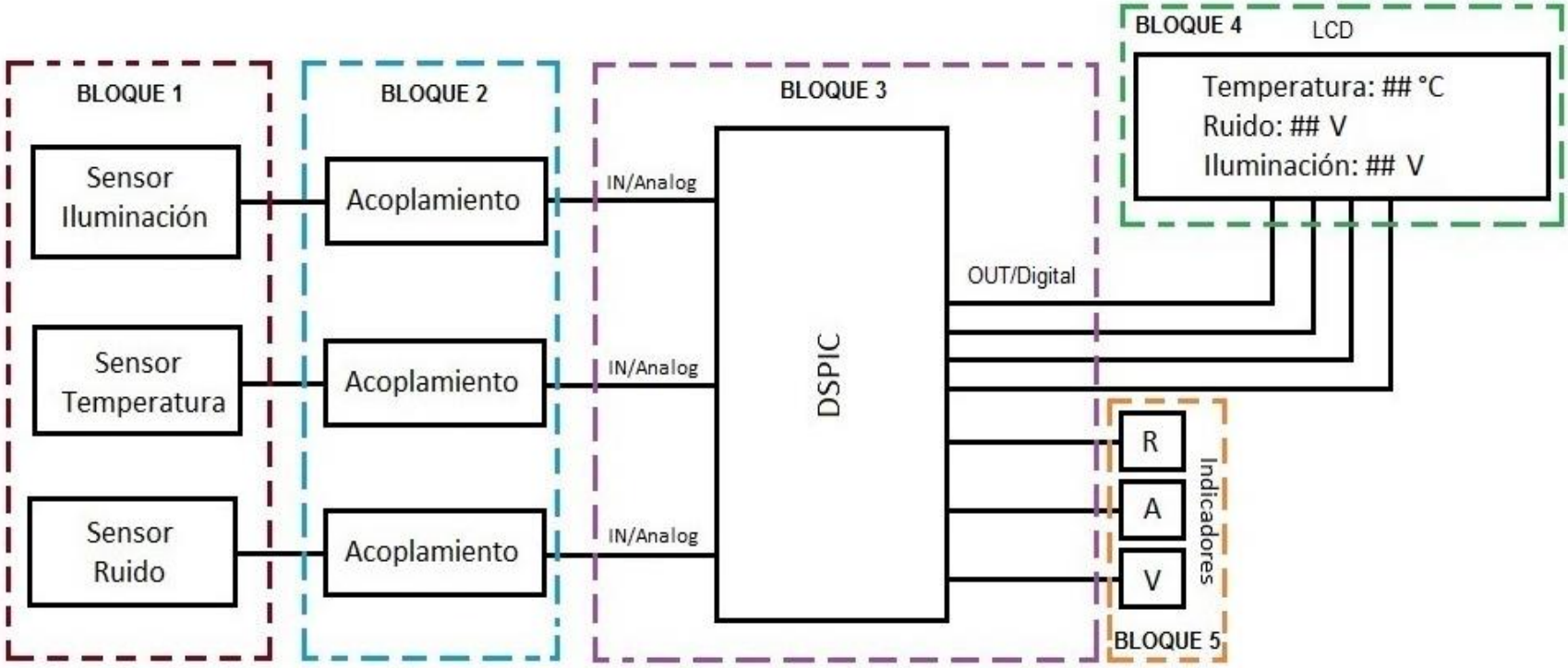


Fig. 79 Diseño general del proyecto.

BLOQUE 1: Sensores

El bloque 1 está compuesto por los sensores utilizados para la detección de cada factor.

a) Sensor de Iluminación: está conformado principalmente por una fotorresistencia LDR.
b) Sensor de temperatura: conformado principalmente por un sensor LM35.
c) Sensor de ruido: éste se encuentra compuesto por un micrófono Electret.

BLOQUE 2: Acoplamiento

La función que realiza el bloque 2 es obtener la señal analógica de cualquiera de los factores ambientales ya mencionados, por medio de un amplificador operacional que trabaje de manera inversa.

El amplificador operacional (op-amp) inversor toma muestras de corriente o tensión de una señal variable en el tiempo, elevando su valor procurando mantenerla lo más fiel posible de manera inversa.

Obteniendo a la salida una señal negativa, lo que requiere volver a invertir la señal utilizando un segundo op-amp con el fin de mandar ésta señal positiva al tercer bloque.

La figura 80 muestra la explicación anterior.

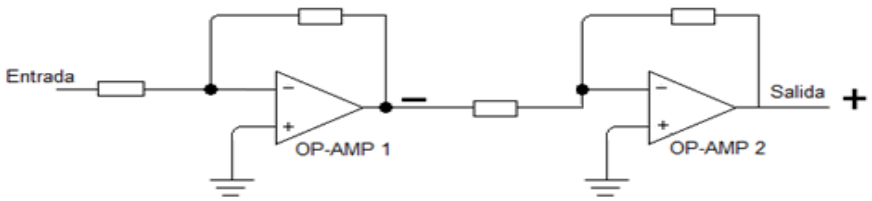


Fig. 80 Acoplamiento: doble amplificador inversor.

BLOQUE 3: DSPIC

El bloque 3 se considera el corazón del diseño, porque su trabajo consiste en la conversión ADC, es decir, que realiza el muestreo, la cuantización y codificación de la

señal analógica en digital. Esto es posible gracias a la capacidad que ofrece el DSPIC, tanto para convertir señales como para programar funciones específicas.

BLOQUE 4: LCD

El display LCD tiene la misión de mostrar los valores de los factores medidos en tiempo real.

BLOQUE 5: Indicadores

Por medio de LED's de diferente color se indicarán los valores fuera de rango.

Utilizando el ADC interno que dispone el dsPIC30F4013 se realiza la conversión de señal analógica en señal digital para crear un medidor ambiental. Donde el rango de la señal de entrada es de 0v a 4v ingresado por el puerto B. La señal de entrada es obtenida por medio de switches (conectados en el puerto F) que, dependiendo su posición, los sensores (ruido, iluminación y temperatura) serán seleccionados para trabajar como señal de entrada al DSPIC. La salida es configurada por el puerto D y visualizada por el LCD de 16x2.

Pasos de conversión A/D del dsPIC30F4013.

Para realizar una conversión, se deben seguir los 10 pasos que se mencionan en el ejemplo 4.1.

Para representar los valores en el LCD es necesario hacer uso de la librería LCD.c que se encuentra en anexos, con la modificación de que el puerto de los datos del LCD sea por PORTD en vez de PORTB, ya que el puerto B es utilizado para recibir las señales mencionadas y para mostrar el nivel en que se encuentra la señal por medio de los LED.

El valor obtenido de la conversión analógico-digital, es almacenado en una variable llamada *i*. Este valor es utilizado para calcular el voltaje ingresado en el DSPIC y mostrarlo en el LCD.

Operaciones.

Un ADC de n bits puede representar hasta 2^n valores digitales, de modo que para la realización de éste caso, la entrada analógica variará dependiendo la señal que sea seleccionada, pero cada una trabaja con resolución de 10 bits, es decir:

$$2^{10} = 1024$$

Siendo así, se puede obtener un valor aproximado de *i*, dónde el voltaje ingresado lo llamaremos *j*.

Suponiendo que *j* sea igual a *1*, ¿cuál será el valor de i?

$$\mathbf{1} = (\mathbf{10} * \boldsymbol{i})/\mathbf{1024}$$

$$1024 = (10 * i)$$

$$\frac{1024}{10} = \boldsymbol{i}$$

$$\boldsymbol{i} = 102.4$$

Con **n** bits, solo se pueden formar 2^n valores discretos diferentes. Por lo tanto habrá valores analógicos que no podrán ser representados con exactitud (error de cuantización). Por lo tanto, para obtener un valor exacto, se modifica levemente 1024, cambiándolo por 1050. Éste nuevo valor se considera el más preciso, ya que se obtuvo a base de las diversas pruebas realizadas. Se repite la operación anterior para comprobar que el valor de *i* sea exacto.

$$\mathbf{1} = (\mathbf{10} * \boldsymbol{i})/\mathbf{1050}$$

$$1050 = (10 * i)$$

$$\frac{1050}{10} = \boldsymbol{i}$$

$$\boldsymbol{i} = \mathbf{105}$$

Con este nuevo valor, podemos declarar la siguiente fórmula para calcular los primeros valores exactos del voltaje.

$$j = (10 * i)/1050$$

Dónde *j* será el número que represente el voltaje ingresado. Hecho este proceso, se asegura mayor exactitud en la medición.

Para calcular un valor aproximado a lo real se requiere que *j* sea dividido entre 10. Entonces se propone la siguiente fórmula:

$$j = j/10$$

Con las operaciones anteriores se procede a crear el diagrama de flujo para el proyecto.

CAPÍTULO 3.3 DISEÑO DEL PROGRAMA

Diagrama de flujo.

En la Fig. 81 se presenta el diagrama de flujo del programa desarrollado para el proyecto. Como se puede observar al inicio del diagrama de flujo se configura el programa, mediante el llamado de las librerías (entre ellas, la librería Delay) y los encabezados. Posteriormente se declaran las variables *i*, *j* que se utilizarán para el cálculo de los valores correspondientes. También se configuran los puertos F y B como entrada y salida, respectivamente. Lo siguiente es inicializar y limpiar la pantalla LCD. Se muestra el mensaje principal “MEDIDOR AMBIENTAL”, abarcando ambas líneas del LCD y centrado.

Hecho lo anterior, se inicia el proceso condicional para cada uno de las entradas analógicas. Cada una es controlada por pulsos que son ingresados por el puerto F. Si el switch 1 está en 1 lógico, la condición a realizar es la de “Ruido”, se especifican los puertos analógicos a utilizar. En el caso del puerto B, la entrada analógica será

solamente B0. Para todos los casos se desactiva el ADC, así como los canales A/D entrada, el tiempo del reloj y buffer 2. El ADC del bit 1 permanecerá activado.

Configurada la conversión, se procede a realizar un ciclo indefinido donde se realiza el muestreo del ADC activado por espacios de tiempos, deteniendo el muestreo para realizar la conversión. Si la conversión es correcta, se almacena lo que se encuentra en el buffer en una variable de tipo entero llamada *i*. Esta variable es utilizada para almacenar el voltaje ingresado en el dsPIC. Y además, es utilizada para realizar los cálculos descritos anteriormente.

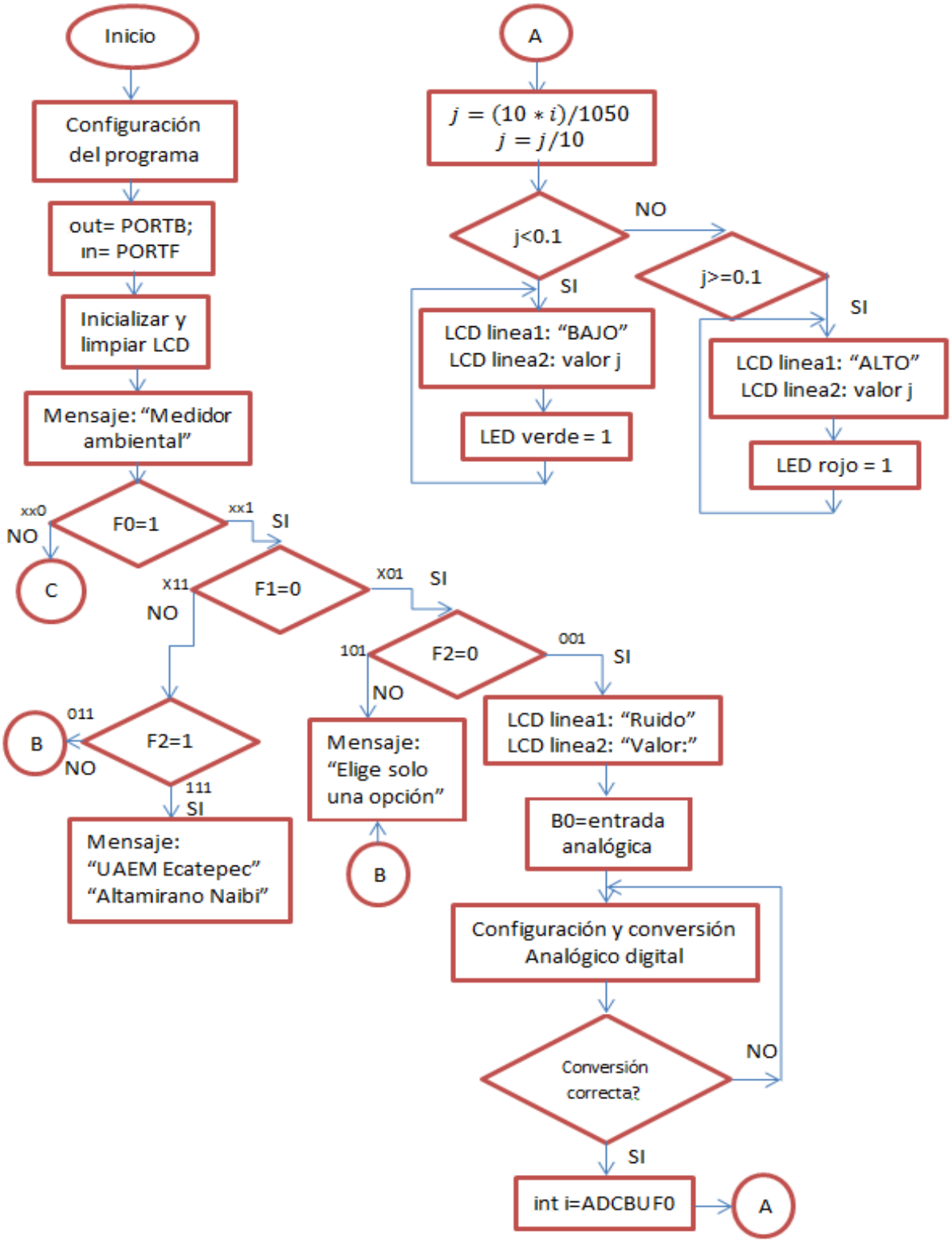
Se realiza la primera y segunda operación para obtener el valor medido. Si *j* es menor a 0.1 significa que el ruido ambiental es bajo mostrándolo por el LCD y encendiendo un LED indicador. Si *j* es mayor o igual a 0.1, indica que el ruido ambiental es alto, y de igual manera se visualiza por el LCD y por un LED indicador.

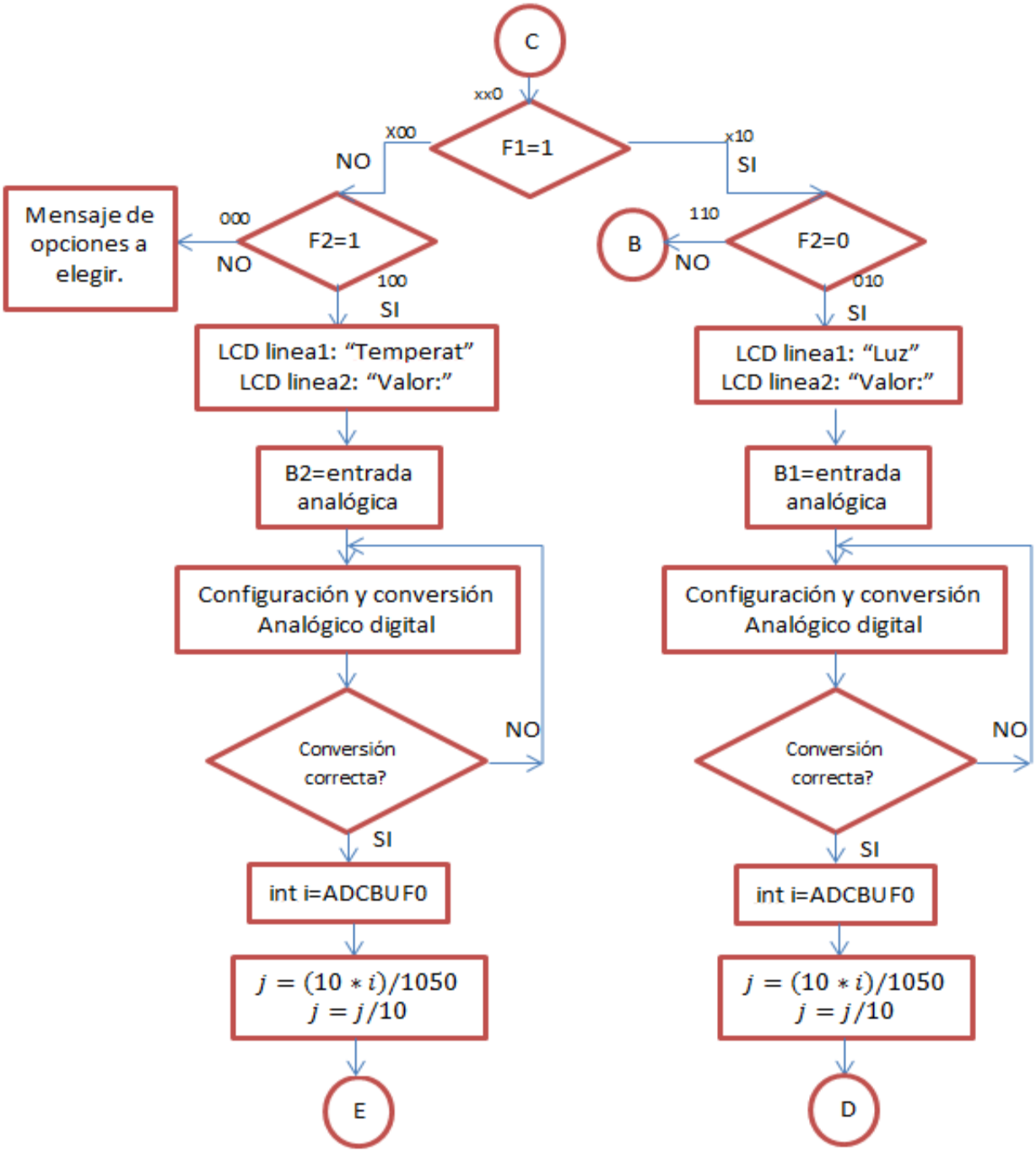
Si el switch 2 está en 1 lógico, la condición a realizar es la de “Luz”, donde se especifican los puertos analógicos a utilizar. En el caso del puerto B, la entrada analógica será solamente B1. Se realiza la misma conversión ADC y ciclo de muestreo. Lo que difiere es el modo de clasificar el nivel de luz ambiental. Si *j* es menor o igual a 0.99 la luz es baja. Si *j* es mayor o igual a 1 pero menor o igual a 2, la luz es media. O si *j* es mayor a 2, quiere decir que hay una alta luz ambiental. Para cada condición se muestra el estado actual en LCD y LED indicador.

Si el switch 3 está en 1 lógico, la condición a realizar es la de “Temperatura”, especificando el puerto B2 como entrada analógica. Se realiza la misma conversión ADC y ciclo de muestreo. La condición para temperatura es: Si *j* es menor o igual a 10 la temperatura es baja. Si *j* es mayor o igual a 11 pero menor o igual a 20, la luz temperatura es media. O si *j* es mayor a 21, quiere decir que la temperatura ambiental es alta. Para cada condición se muestra el estado actual en LCD y LED indicador.

Si dos switches llegasen a estar en 1 lógico al mismo tiempo se le indica al usuario por medio de un mensaje que sólo puede elegir una opción.

Si los 3 switches están en 1 lógico se muestra en el LCD el nombre de la escuela y de la creadora del proyecto.




C
xx0
F1=1
x00
NO
x10
SI
Mensaje de opciones a elegir.
000
F2=1
NO
100
SI
B
110
F2=0
NO
010
SI
LCD linea1: "Temperat"
LCD linea2: "Valor:"
LCD linea1: "Luz"
LCD linea2: "Valor:"
B2=entrada analógica
B1=entrada analógica
Configuración y conversión Analógico digital
Configuración y conversión Analógico digital
Conversión correcta?
NO
SI
Conversión correcta?
NO
SI
int i=ADCBUF0
int i=ADCBUF0
$j = (10 * i)/1050$
$j = j/10$
$j = (10 * i)/1050$
$j = j/10$
E
D

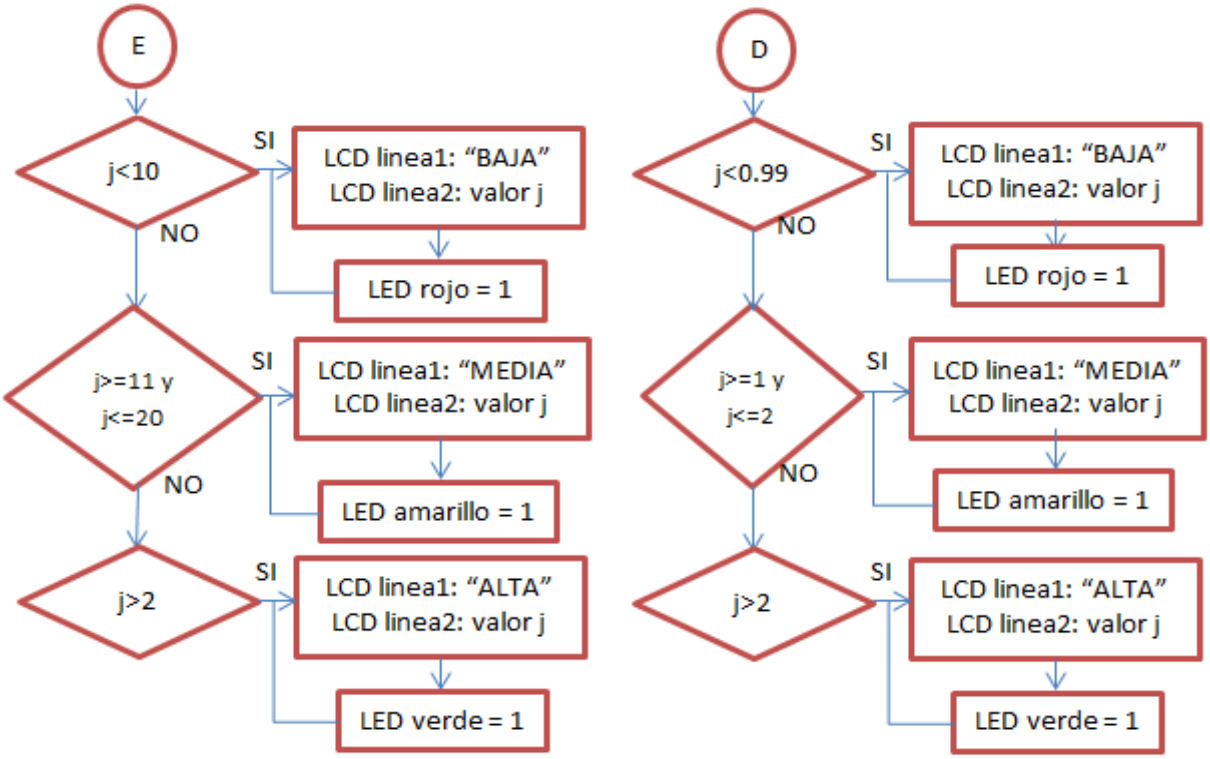


Fig. 81 Diagrama de flujo del Proyecto

CAPÍTULO 3.4 CODIFICACIÓN

A continuación se presenta el código del programa desarrollado en el apartado anterior, donde los colores indican lo mismo que en los ejemplos mencionados con anterioridad.

```
#include "p30f4013.h" //incluye la cabecera del dispositivo a usar
#include "libpic30.h" //librería del pic30 para procesamiento de señales
#include "delay_mio.h"//librería Delay
#include "lcd_mio.h"  //librería de LCD, configurada para puerto D

//Configuración
_FOSC (CSW_FSCM_OFF & XT);
_FWDT (WDT_OFF);
_FBORPOR (PBOR_ON & MCLR_EN);

#ifndef __DELAY_H
      #define delay_us(x) __delay32(((x*FCY)/100000L))//delay x us
      #define delay_ms(x) __delay32(((x*FCY)/10L)) //delay x ms
      #define __DELAY_H 1
#endif

char s[16];                    //número de caracteres necesarios del LCD
float i=0,j=0;                 //Variables a utilizar
```

```
int main()
{
	PORTF=0;		//Declaración de los puertos F y B
	TRISF =0;
	PORTB=0;

	lcd_init();		//Inicializa LCD
	lcd_clear();		//Limpia el LCD
	lcd_goto(LINE1+4);	//Posición del cursor del LCD
	lcd_puts("MEDIDOR");	//Mensaje
	lcd_goto(LINE2+3);
	lcd_puts("AMBIENTAL");
	lcd_clear();		//Limpiar pantalla
	proceso();		//Llama a y ejecuta proceso()
	lcd_clear();
}
void proceso(){			//Parámetro proceso()
	if(PORTFbits.RF0 == 1){ //xx1
		if(PORTFbits.RF1 == 0){ //x01
			if(PORTFbits.RF2 == 0){ //001
				//RUIDO
				lcd_goto(LINE1+3);
				lcd_puts("Ruido");
				lcd_goto(LINE2);
				lcd_puts("Valor:");

				TRISB =0X1;		//Entrada B0, los demás salida
				ADPCFG=0XFFFE;	//  Puerto  B0  como  analógico,  los
demás digital
				//configuración general del ADC
				ADCON1=0;		//ADC desactivado
			//configuración canales A/D entrada
				ADCHS=0;		//CH0NA0
				ADCSSL=0;		//SCAN para el canal 0 NA 0
				//Tiempos
				ADCON3=0x0101;	//1  TAD  y  reloj  desactivado  del
sistema
				//Buffers
				ADCON2=0x0101;
				ADCON1bits.ADON = 1;	//Habilita ADC
				while(1){
					ADCON1bits.SAMP = 1; //Iniciar muestreo
					delay_ms(1);	//  Tiempo  para  realizar  el
muestreo
					ADCON1bits.SAMP = 0; //Detener muestreo, iniciar
conversión
					while(!ADCON1bits.DONE);//Se  ha  realizado  la
conversión?
					i=ADCBUF0;  //variable  i  almacena  el  dato
obtenido en la conversión
					j=(10*i)/1070;	//Operaciones	para
representar en el LCD
					j=j/10;
					if(j<0.1){ //Condición para ruido bajo
						lcd_goto(LINE2+7);
						lcd_puts(s);
						sprintf(s,"%.2f v ",j);
						lcd_goto(LINE1+9);
						lcd_puts("BAJO");
						PORTB=0x8; //led verde
						delay_ms(100);
					}
					if(j>=0.1){//Condición para ruido alto
							lcd_goto(LINE2+7);
```

```
                                                lcd_puts(s);
                                                sprintf(s,"%.2f v ",j);
                                                lcd_goto(LINE1+9);
                                                lcd_puts("ALTO");
                                                PORTB=0x20; //led rojo
                                                delay_ms(100);
                                          }
                              }
                        }//fin 001
                        else{//101
                              lcd_goto(LINE1);
                              lcd_puts("Elige solo una");
                              lcd_goto(LINE2);
                              lcd_puts("opcion.");
                        }
                  }//fin x01
                  else{//x11
                        if(PORTFbits.RF2 == 1){//111
                              lcd_goto(LINE1);
                              lcd_puts("CU UAEM ECATEPEC");
                              lcd_goto(LINE2);
                              lcd_puts("Naibi Altamirano");
                        }//fin 111
                        else{//011
                              lcd_goto(LINE1);
                              lcd_puts("Elige solo una");
                              lcd_goto(LINE2);
                              lcd_puts("opcion.");
                        }
                  }
            }//fin xx1
            else{//xx0
                  if(PORTFbits.RF1 == 1){//x10
                        if(PORTFbits.RF2 == 0){//010
                              //LUZ
                              lcd_goto(LINE1+3);
                              lcd_puts("LUZ");
                              lcd_goto(LINE2);
                              lcd_puts("Valor:");

                              TRISB =0X2;       //Entrada B1, los demás salida
                              ADPCFG=0XFFFD;    //  Puerto  B1  como  analógico,  los
demás digital
                              ADCON1=0;
                              ADCHS=0x1;        //CH0NA1
                              ADCSSL=0;
                              //Tiempos
                              ADCON3=0x0101;
                              ADCON2=0x0101;
                              ADCON1bits.ADON = 1;
                              while(1){
                                    ADCON1bits.SAMP = 1;
                                    delay_ms(1);
                                    ADCON1bits.SAMP = 0;
                                    while(!ADCON1bits.DONE);

                                          i=ADCBUF0;
                                          j=(10*i)/1023;
                                          j=j/10;

                                          if(j<=0.99){ //Condición para luz baja
                                                lcd_goto(LINE2+7);
                                                lcd_puts(s);
                                                sprintf(s,"%.2f v ",j);
```

```
                                    lcd_goto(LINE1+7);
                                    lcd_puts("BAJA ");
                                    PORTB=0x20; //led rojo
                                    delay_ms(100);
                              }
                              else if(j>=1 && j<=2){//Condición para luz
media
                                    lcd_goto(LINE2+7);
                                    lcd_puts(s);
                                    sprintf(s,"%.2f v ",j);
                                    lcd_goto(LINE1+7);
                                    lcd_puts("MEDIA ");
                                    PORTB=0x10; //led amarillo
                                    delay_ms(100);
                               }
                              else if(j>2){//Condición para luz alta
                                    lcd_goto(LINE2+7);
                                    lcd_puts(s);
                                    sprintf(s,"%.2f v ",j);
                                    lcd_goto(LINE1+7);
                                    lcd_puts("ALTA ");
                                    PORTB=0x8; //led verde
                                    delay_ms(100);
                               }
                        }
                  }//fin 010
                  else{//110
                        lcd_goto(LINE1);
                        lcd_puts("Elige solo una");
                        lcd_goto(LINE2);
                        lcd_puts("opcion.");
                  }
            }//fin x10
            else{//x00
                  if(PORTFbits.RF2 == 1){//100
                        //TEMPERATURA
                        lcd_goto(LINE1);
                        lcd_puts("Temperat");
                        lcd_goto(LINE2);
                        lcd_puts("Valor:");

                        TRISB =0X4;       //Entrada B2, los demás salida
                        ADPCFG=0XFFFB;    //  Puerto  B2,  como  analógico,  los
demás digital
                        ADCON1=0;
                        ADCHS=0x2;        //CH0NA2
                        ADCSSL=0;
                        ADCON3=0x0101;
                        ADCON2=0;
                        ADCON1bits.ADON = 1;
                        while(1){
                              ADCON1bits.SAMP = 1;
                              delay_ms(.2);
                              ADCON1bits.SAMP = 0;
                              while(!ADCON1bits.DONE);

                              i=ADCBUF0;
                              j=(10*i)/1030;

                              if(j<=10){   //Condición   para   temp
baja
                              lcd_goto(LINE1+9);
                              lcd_puts("BAJA ");
                              lcd_goto(LINE2+7);
```

```
                                        lcd_puts(s);
                                        sprintf(s,"%.2f 'C ",j);
                                        PORTB=0x20; //led rojo
                                        delay_ms(100);
                                    }
                                    else  if(j>=11  &&  j<=20){//Condición  para
temp media
                                            lcd_goto(LINE2+7);
                                            lcd_puts(s);
                                            sprintf(s,"%.2f 'C ",j);
                                            lcd_goto(LINE1+9);
                                            lcd_puts("MEDIA ");
                                            PORTB=0x10; //led amarillo
                                            delay_ms(100);
                                        }
                                    else if(j>=21){ //Condición para temp alta
                                            lcd_goto(LINE1+9);
                                            lcd_puts("ALTA");
                                            lcd_goto(LINE2+7);
                                            lcd_puts(s);
                                            sprintf(s,"%.2f 'C ",j);
                                            PORTB=0x8; //led verde
                                            delay_ms(100);
                                        }
                            }
                        }//fin 111
                        else{//000
                            lcd_goto(LINE1);
                            lcd_puts("1.Ruido   2.Luz");
                            lcd_goto(LINE2);
                            lcd_puts("3.Temperatura");
                        }
                }
            }//fin xx0
}
```

La simulación por compilación se llevó a cabo en MPLAB, donde se observa que el resultado corresponde a lo planteado con anterioridad.

CAPÍTULO 3.5 IMPLEMENTACIÓN

Diagrama circuital.

La explicación del diagrama circuital será divido en cuatro partes, donde las tres primeras partes es dedicado a la explicacion de cada circuito ambiental: ruido, luz y temperatura. Y la cuarta parte es la conexión del dsPIC30F4013 en conjunto con el LCD.

Parte 1. Ruido

En el diagrama del circuito detector de ruido (Fig. 82), está conformado principalmente por dos op-amp inversores y un comparador, que tienen la función de amplificar y

comparar la señal recibida por el micrófono electret. La conexión de alimentación de los tres op-amp, es con +9v en la terminal 7 y -9v en la 4. Para los inversores, la entrada no inversora (pin 3) es conectada a tierra. Para el comparador, la entrada inversora está directamente conectada a un voltaje de referencia. En la entrada inversora (pin 2) del primer inversor, se encuentra conectada una resistencia de 3.3kΩ que interconecta con el microfono electret, el cual es alimentado a través de una resistencia de 2.2kΩ a los 9v. En esa misma entrada del pin 2 del inversor, se conecta otra resistencia con valor de 150kΩ. La resistencia de entrada (Ri = 3.3kΩ) y la resistencia de retroalimentación (Rf = 150kΩ) son para obtener una ganancia de 45.5.

$$Ganancia \quad \rightarrow \quad \Delta_v = \frac{-R_f}{R_i}$$

Una gancia más alta puede provocar que el op-amp se queme y ya no sea util para el circuito.

A la salida del op-amp (pin 6) se interconecta la resistencia Rf, además, de la misma salida se conecta la misma configuración para un segundo inversor, utilizando resistencias de 1kΩ para Ri y Rf. En la salida del segundo amplificador se conecta el comparador a traves de la entrada no inversora, con el fin de comparar la señal que recibe del segundo inversor con el voltaje de referencia. Finalmente, en la salida del comparador se conecta un LED para visualizar el correcto funcionamiento del circuito.

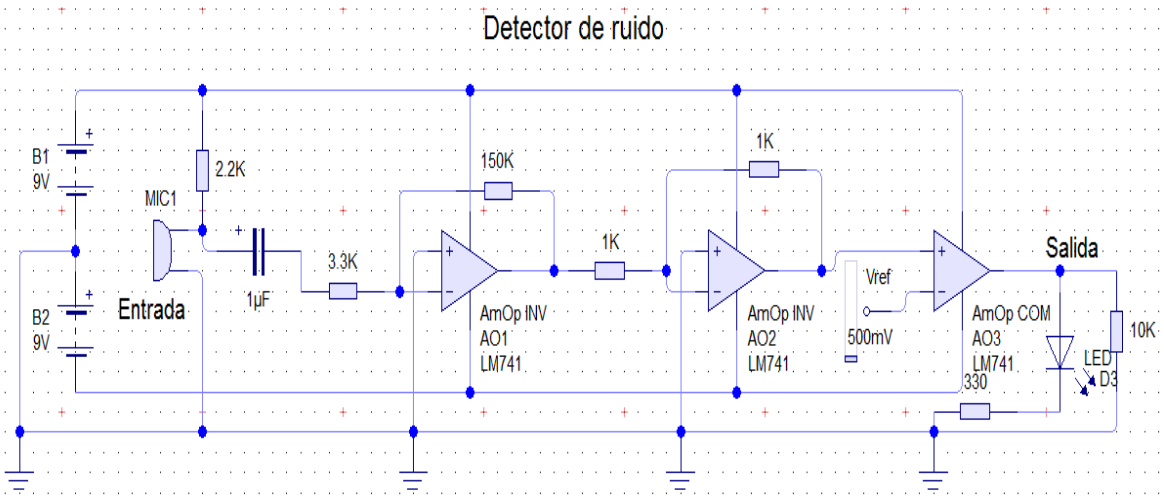


Fig. 82 Detector de Ruido.

Parte 2. Luz

En la Fig. 83 se muestra el diagrama del circuito detector de luz. Compuesto principalmente por dos op-amp inversores y una fotorresistencia de 1kΩ a 1MΩ. La conexión de alimentación de los op-amp inversores es la misma que la del circuito detector de ruido. La entrada no inversora del primer (pin 3) es conectada a tierra. En la entrada inversora (pin 2) se encuentra conectada la fotoresistencia, siendo alimentada por una resistencia de 2.2kΩ que va directamente a la fuente. En esa misma entrada del pin 2 se conecta con el mismo valor de 2.2kΩ la resistencia de entrada (Ri) y la resistencia de retroalimentación (Rf) para obtener una ganancia de 1.

A la salida del op-amp (pin 6) se interconecta la resistencia Rf. Y se conecta la misma configuración para el segundo inversor, utilizando resistencias de 1kΩ para Ri y Rf. Finalmente, a la salida del segundo amplificador se conecta un LED para visualizar el cambio de iluminación.

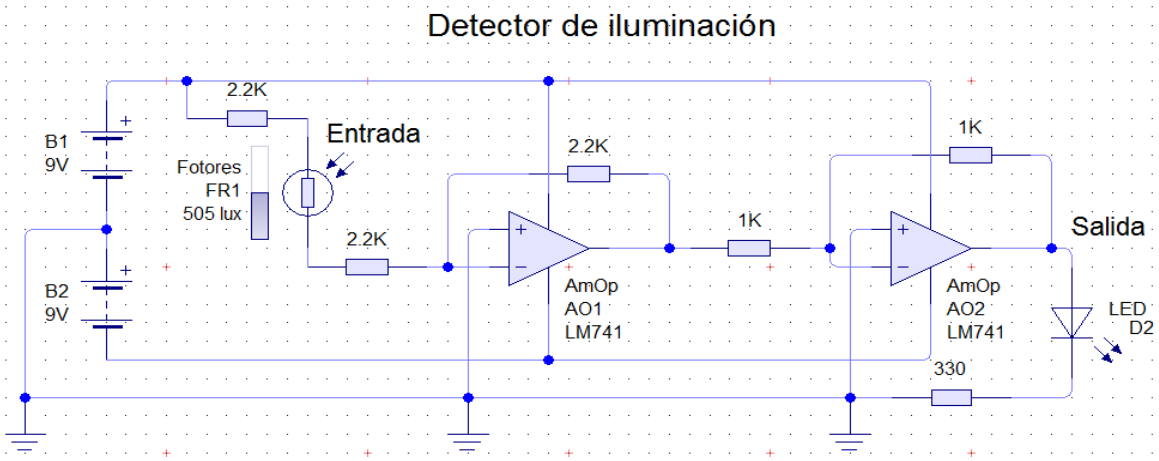


Fig. 83 Detector de Luz.

Parte 3.

El diagrama del circuito detector de temperatura (Fig. 84), etá compuesto principalmente por dos op-amp inversores y un sensor de temperatura LM35. La conexión de alimentación de los op-amp inversores es la misma que la del circuito detector de ruido. La entrada no inversora del primer (pin 3) es conectada a tierra. En la entrada inversora (pin 2) se encuentra conectada a una resistencia de 1kΩ (Ri), que

interconecta con la salida del sensor LM35, el cual es directamente alimentado a los 9v. En esa misma entrada del pin 2 del inversor, se conecta una segunda resistencia de 10kΩ (Rf), para obtener una ganancia de 10.

A la salida del op-amp (pin 6) se interconecta la resistencia Rf. Además, se conecta la misma configuración para el segundo inversor, utilizando resistencias de 1kΩ para Ri y Rf.

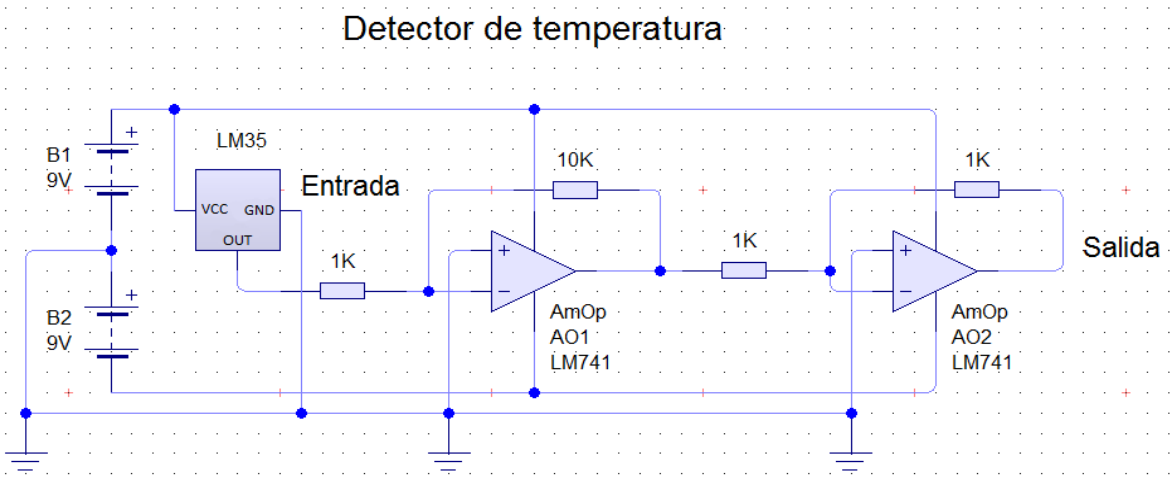


Fig. 84 Detector de Temperatura.

Parte 4.

En la Fig. 85 se muestra el diagrama del circuito de la conexión del dsPIC30F4013, donde el pin 1 está conectado a una resistencia de 10kΩ que va directamente al voltaje. Los pines 2, 3, 4, estan recervados como entradas para conectar las salidas de cada detector ambiental. En los pines siguientes (5, 6 y 7), se encuentran conectados LEDs indicadores de estado.

El oscilador de cristal con sus respetivos capacitores se posicionan en los pines 13 y 14. Además es importante mantener alimentado todo el DSPIC para su correcto funcionamiento.

Por el puerto F (pin 26, 29 y 30) se conecta el control de entrada, el cual es por medio de un dipswitch de tres posiciones, donde el primer switch 1 (sw1) es conectado al pin

30, el sw2 al 29 y el sw3 al 26. El dipswitch está conectado a la fuente de alimentación y con resistencias de 1k a tierra para evitar corto circuito.

En los pines 15 y 16 del DSPIC se conectan los pines RS y E del LCD respectivamente. En el puerto D (pin 19, 20, 33 y 34)se encuentran conectados los pines de datos D4-D7 del LCD. Los pines D0 al D3 del LCD deberan estar conectados a GND para no ocasionar conflicto con los pines de datos que si son utilizados. Las demás terminales del LCD son conectadas como lo indica la tabla 7 del LCD.

Nota: Tanto los detectores ambientales como la conexión del dsPIC30F4013, deben estar conectados a una tierra común, ya que éste se toma como el punto de referencia para ambas conexiones.

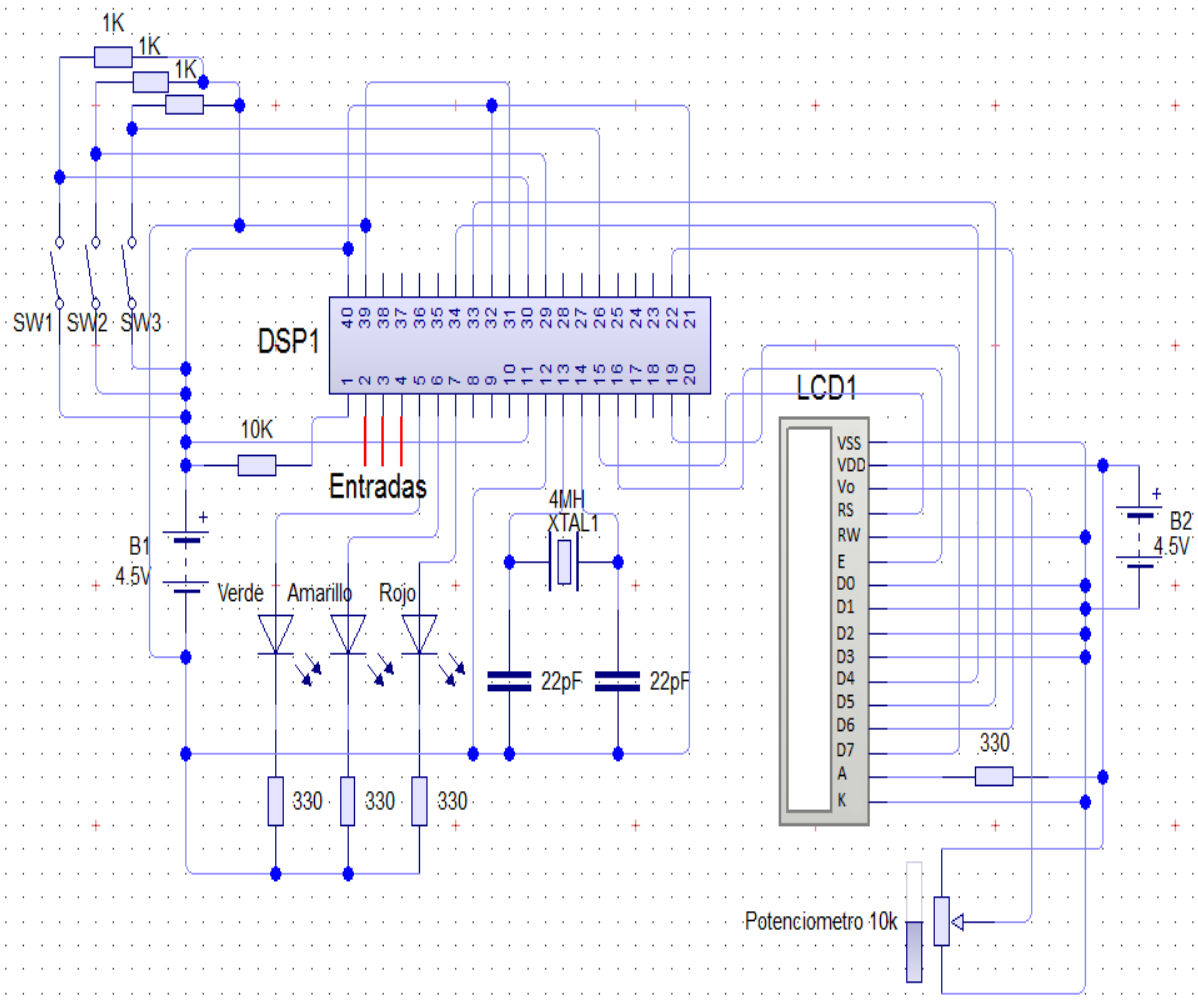


Fig. 85 Circuito principal: dsPIC30F4013 con LCD.

CAPÍTULO 3.6 ENSAMBLAJE Y PRUEBAS

Siguiendo cada diagrama circuital, se montó físicamente en protoboard para realizar pruebas y obtener resultados.

Ensamblaje del circuito detector de ruido.

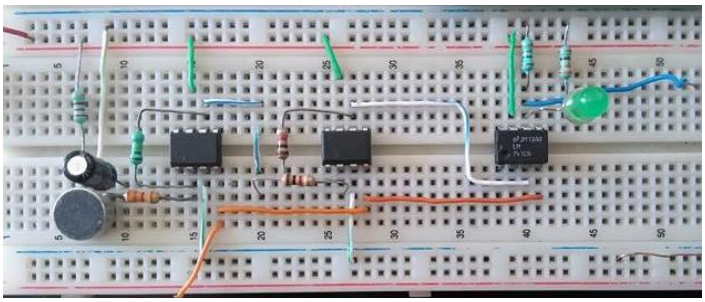


Fig. 86 Circuito detector de ruido.

Prueba del circuito de ruido.

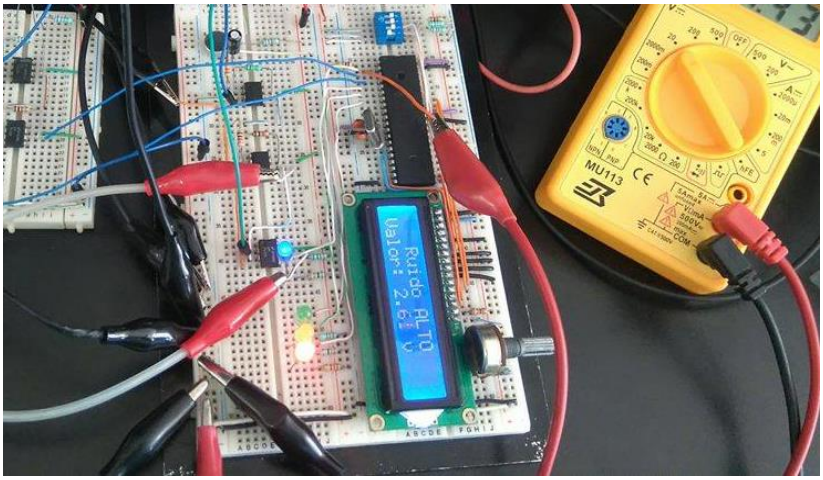


Fig. 87 Prueba: ruido.

Por medio de un osciloscopio puede ser visible la cantidad de ruido que capta el micrófono.

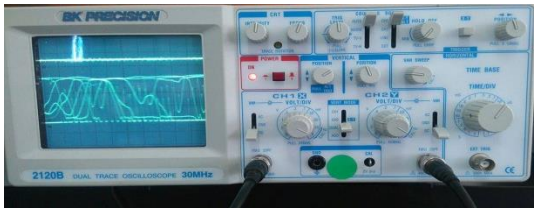


Fig. 88 Visualización de ruido.

Ensamblaje del circuito detector de iluminación.

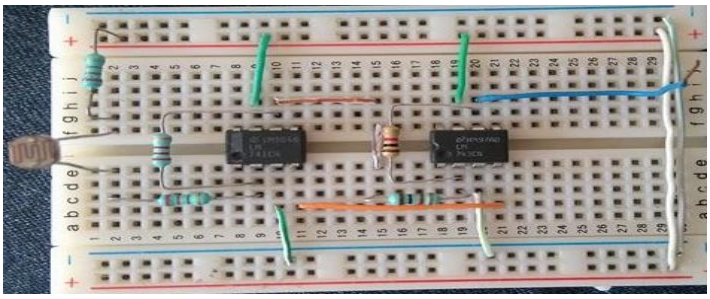


Fig. 89 Circuito detector de iluminación.

Prueba del circuito de iluminación.

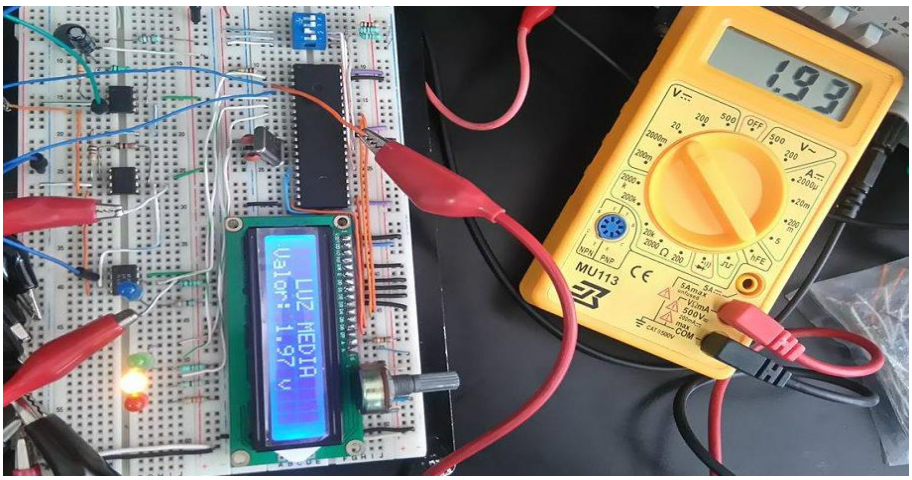


Fig. 90 Prueba: Iluminación.

Ensamblaje del circuito detector de temperatura.

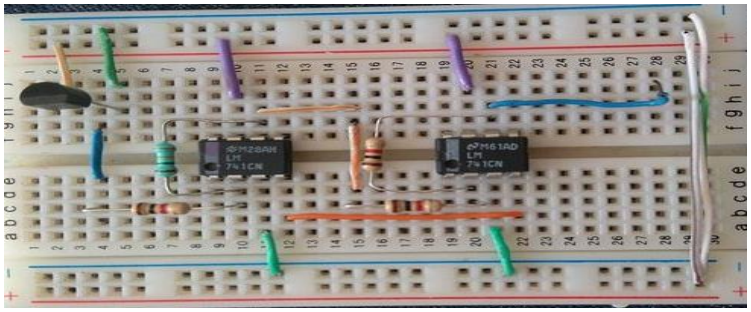


Fig. 91 Circuito detector de temperatura.

Prueba del circuito de temperatura.

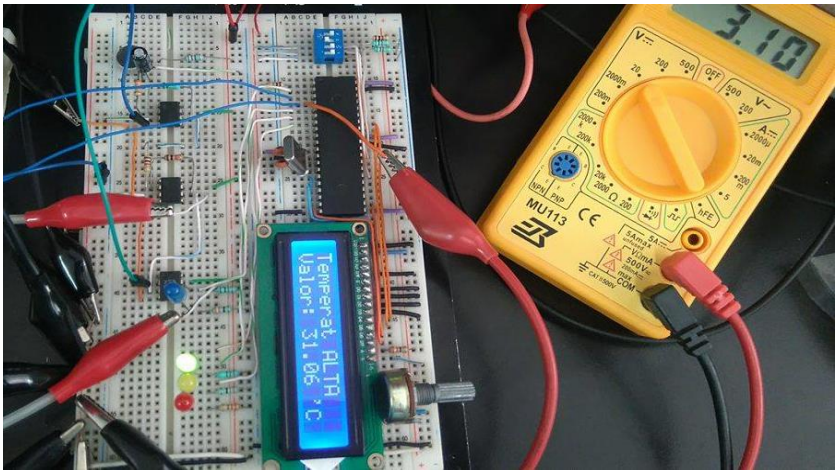


Fig. 92 Prueba: temperatura.

Ensamblaje del circuito principal dsPIC30F4013.

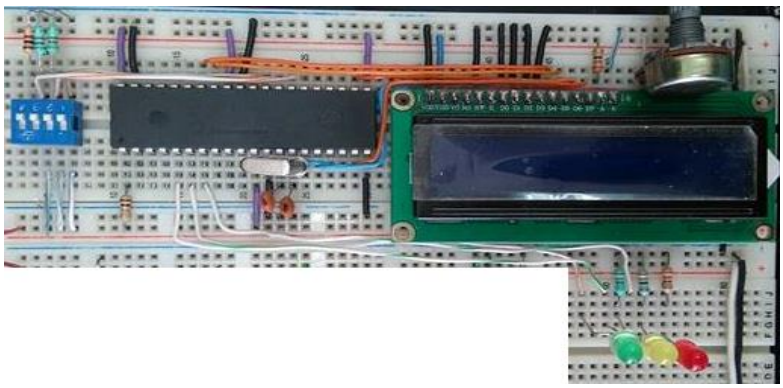


Fig. 93 Circuito principal.

Cada una de las funciones programadas en el DSPIC trabaja como lo planeado. En la fig. 94 se muestra el mensaje que aparece cuando los todos los switches se encuentran en 1 lógico.

Fig. 94 Mensaje estando en 1 el controlador.

Finalmente se demostró (Fig. 95) que el programa junto con el diseño de los circuitos, funciona correctamente.

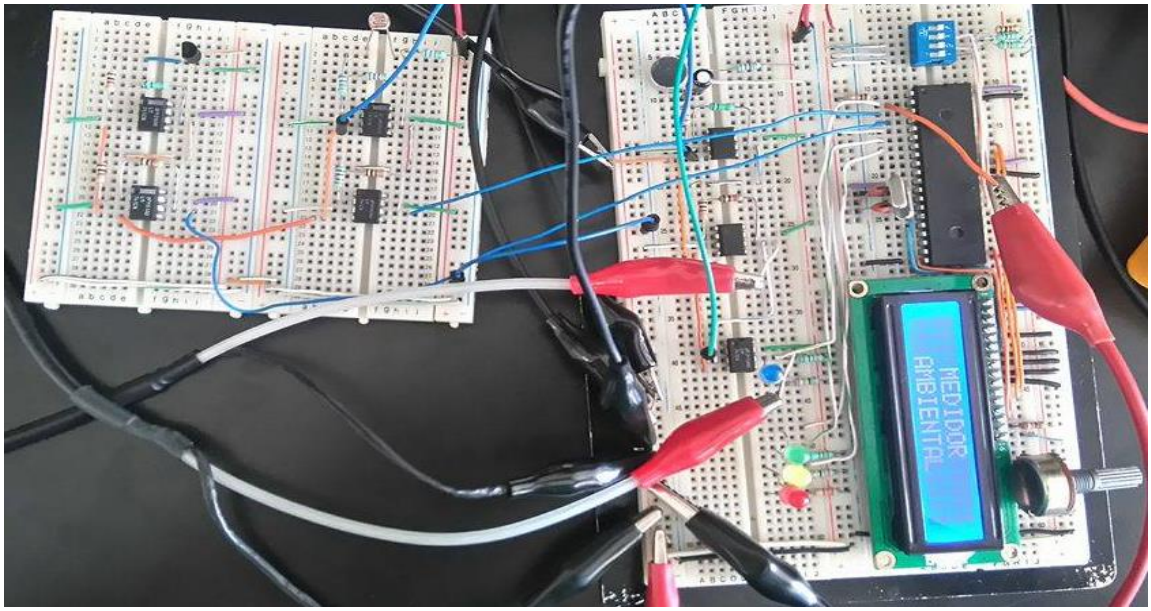


Fig. 95 Circuito completo.

Proceso de creación de placas

Una vez comprobado que los resultados son los deseados, se hace el proceso de creación de placas, con el fin de tener cada componente de los circuitos mencionados, fijos y sobre todo, para una mejor presentación.

El primer paso para realizar este proceso es, por medio de un software especial, diseñar las pistas, posición de cada componente y tamaño de la placa fenólica.

Existen infinidad de software que pueden realizar el mismo trabajo pero por comodidad de instalación y experiencia en trabajos pasados, el software que fue utilizado se llama PCB Wizard.

PCB Wizard es un programa diseñado para el ámbito educativo que permite crear esquemas de circuitos electrónicos y a partir de estos, obtener de una manera sencilla el diseño del circuito impreso a una o dos caras. Trabaja en conjunto con Live Wire (programa utilizado para diseñar todos los circuitos de este proyecto).

Cada circuito ambiental, así como el principal, fue diseñado de manera que se apegue al circuito presentado en el protoboard. A continuación se presenta de la fig. 96 a 99, las pistas de cada circuito.

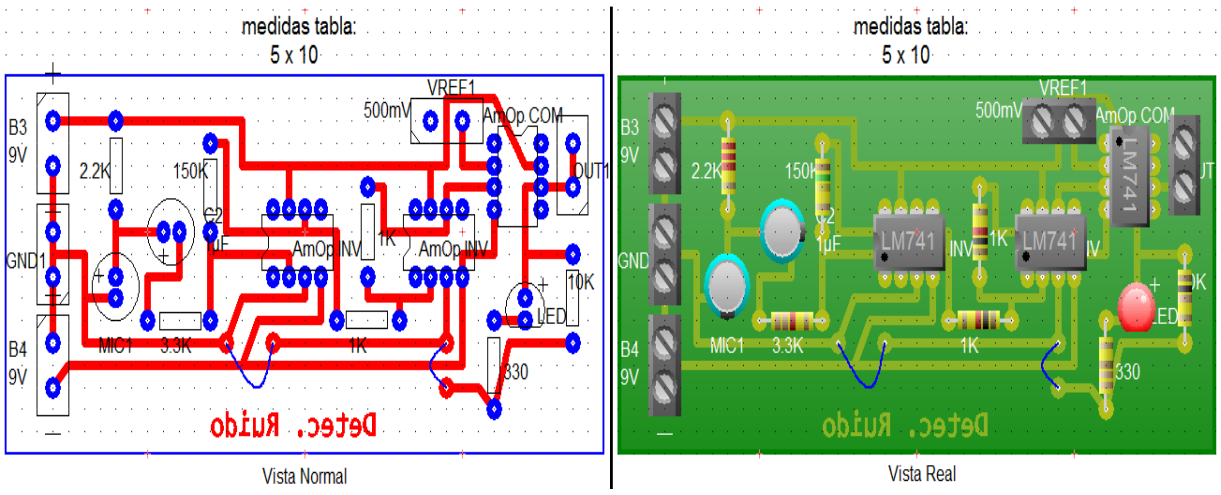


Fig. 96 Pistas: Detector de ruido.

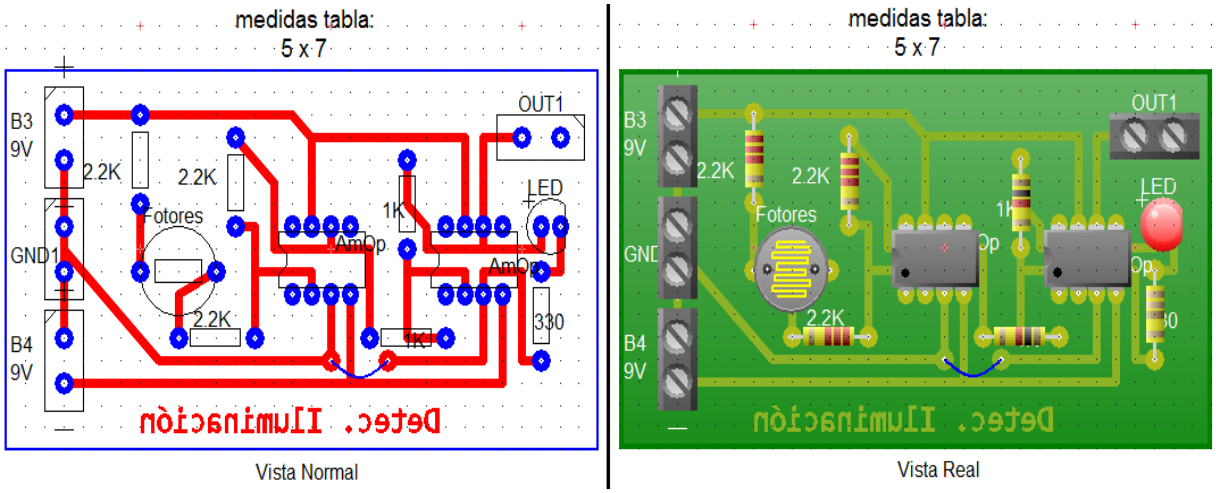


Fig. 97 Pistas: Detector de luz.

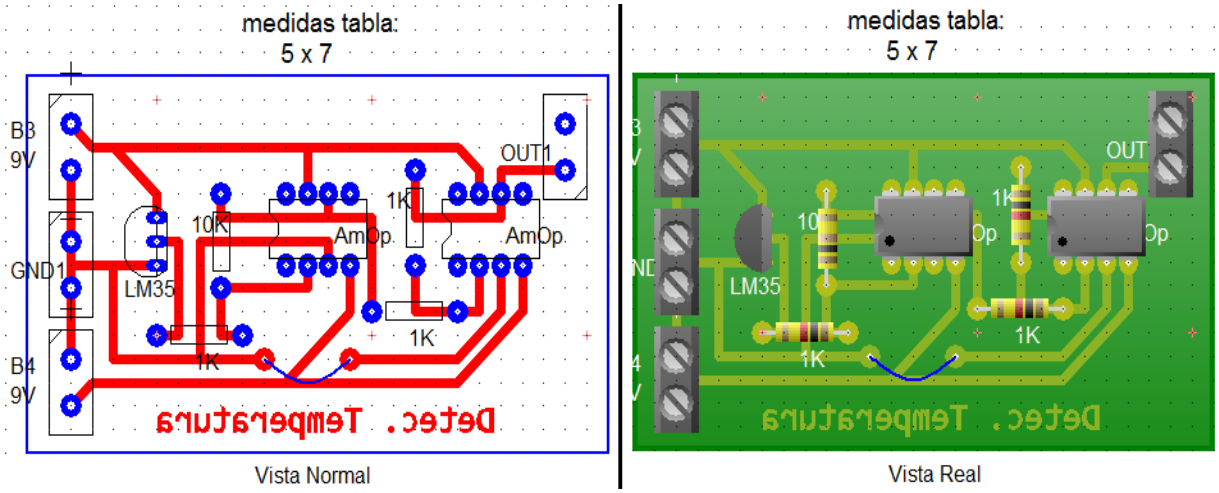


Fig. 98 Pistas: Detector de temperatura.

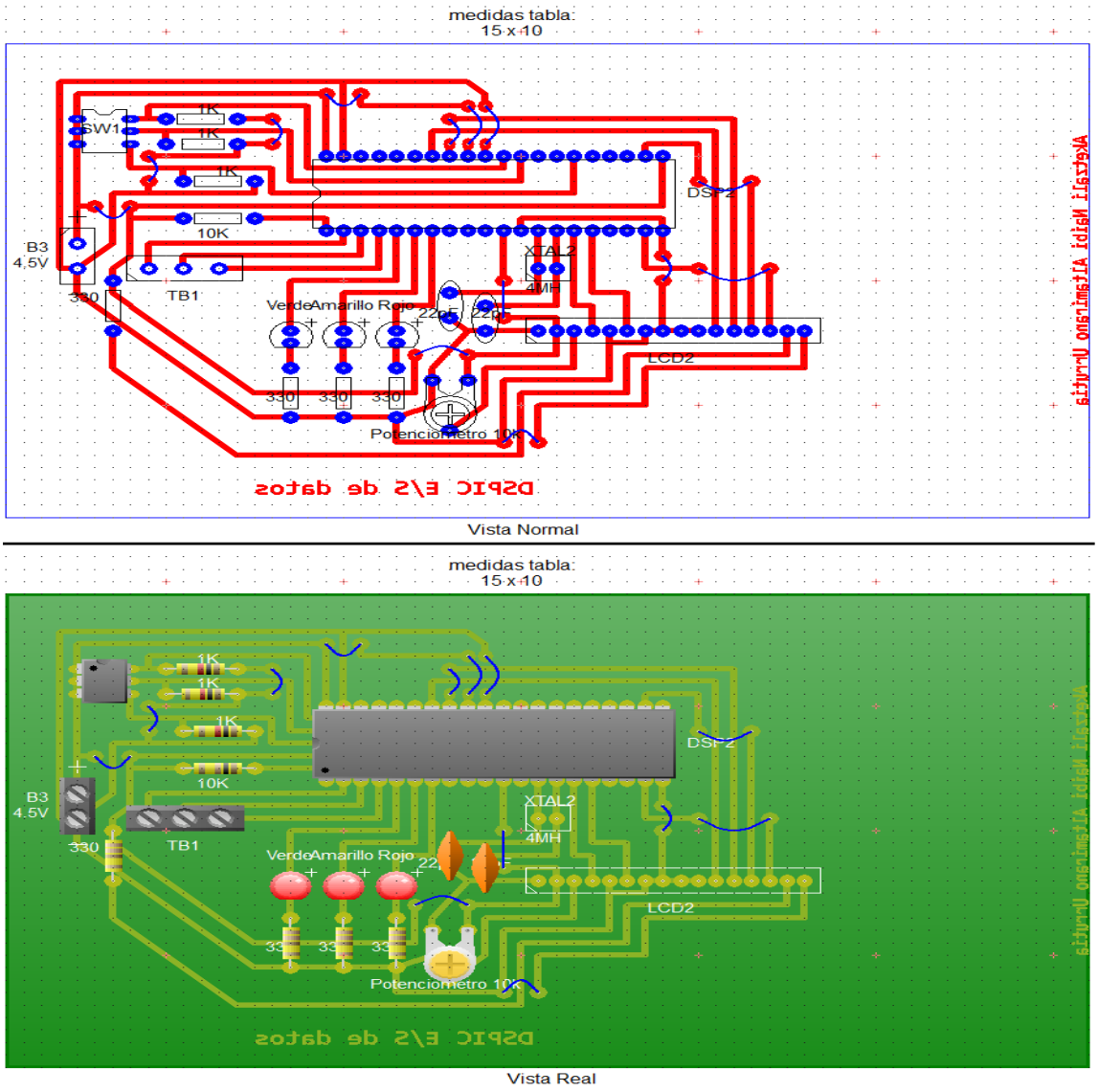


Fig. 99 Pistas: Circuito Principal.

Hecho lo anterior, lo siguiente es imprimir los circuitos en papel euculene de dos caras, para poder adherir la tinta en las placas fenólicas.

Nota: el papel mencionado es termo transferible y la impresión es láser.

Teniendo la impresión de cada circuito, se ajusta ésta misma en cada placa, de modo que la tinta quede del lado del cobre de la placa.

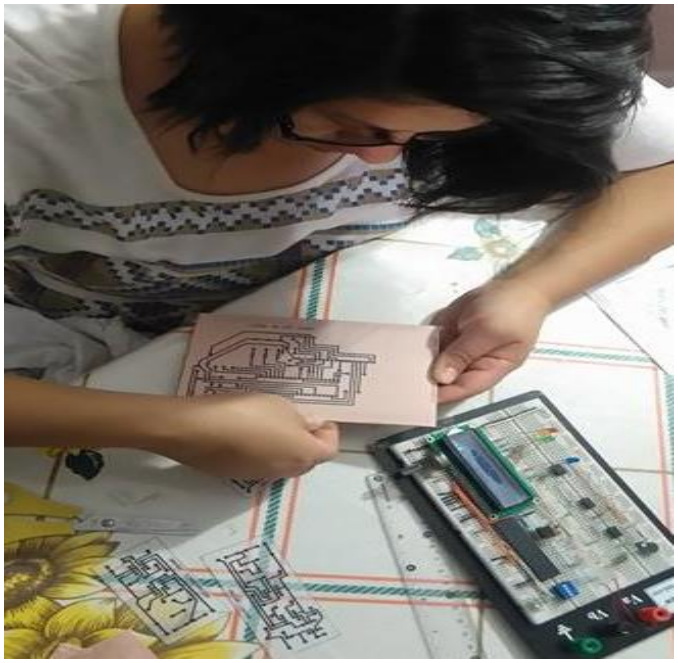


Fig. 100 Proceso de ajuste de impresión.

Una vez colocadas las impresiones, se planchan las placas. Con el elevado calor, la tinta se adhiere a la placa (Fig. 101), las pistas que no se llegan a pegar completamente en la placa, se repasan con plumón de tinta indeleble.

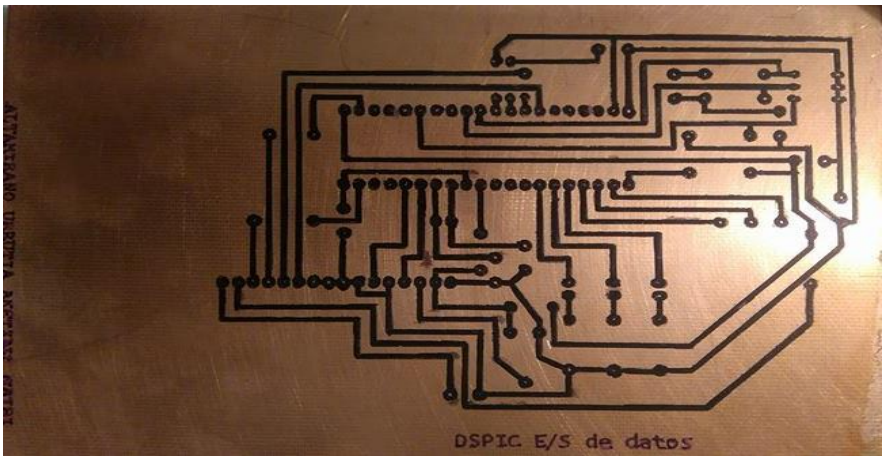


Fig. 101 Placa con pista.

El paso siguiente es retirar el cobre sobrante de la placa. Para disolver el cobre se utiliza una solución corrosiva llamada cloruro férrico ($FeCl_3$). Las placas se sumergen dentro de ésta solución (Fig. 102), mezclada con un poco de agua, durante unos

minutos para que finalmente salgan las placas sin cobre en los espacios sin utilizar, como se muestra en la fig. 103.

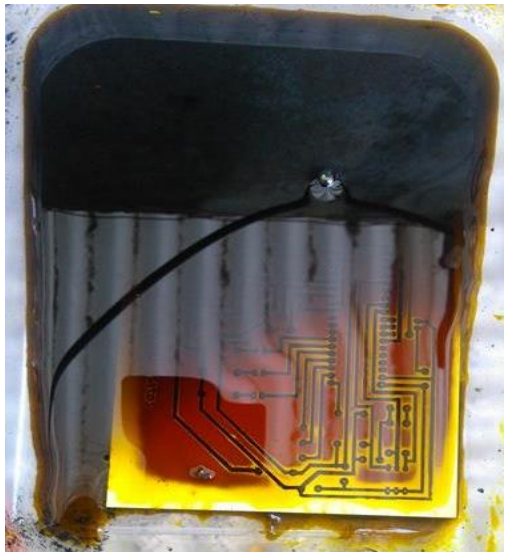


Fig. 102. Placa sumergida en $\boldsymbol{FeCl_3}$

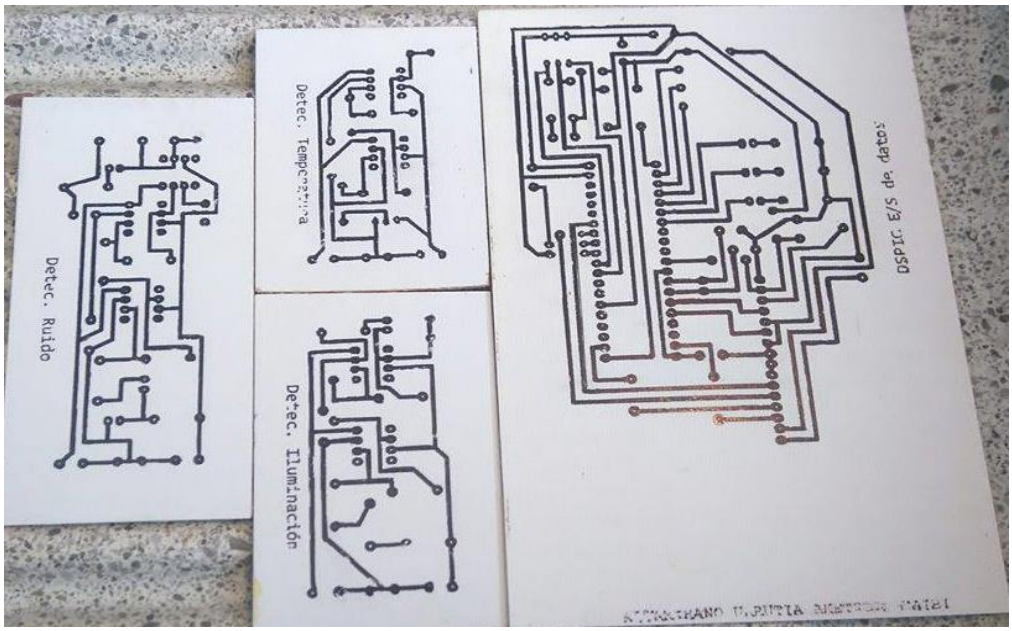


Fig. 103 Placa libre de cobre sobrante.

Para retirar la tinta de las placas, éstas se sumergen en alcohol etílico (etanol) y con ayuda de una lija tipo agua, se tallan para facilitar el desprendimiento de la tinta. Es

necesario eliminar la tinta para que la soldadura se adhiera y haga contacto directo con las pistas.

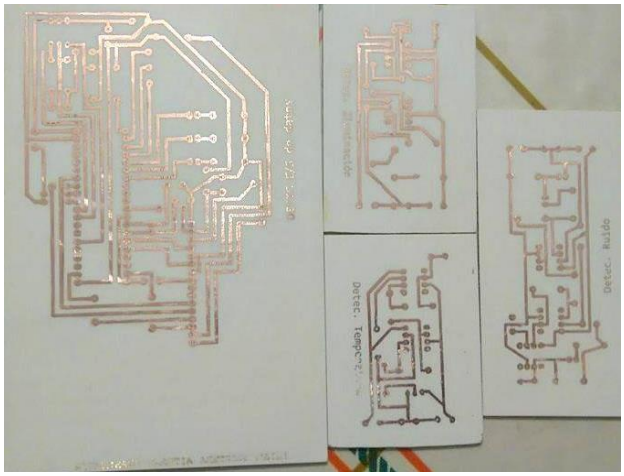


Fig. 104 Placas sin tinta.

Lo siguiente es perforar todas las placas para poder introducir los componentes necesarios. El tamaño de broca que se utilizó es de 1/16”, la cual resulta tener mayor facilidad de adquisición que una de 1/32”.

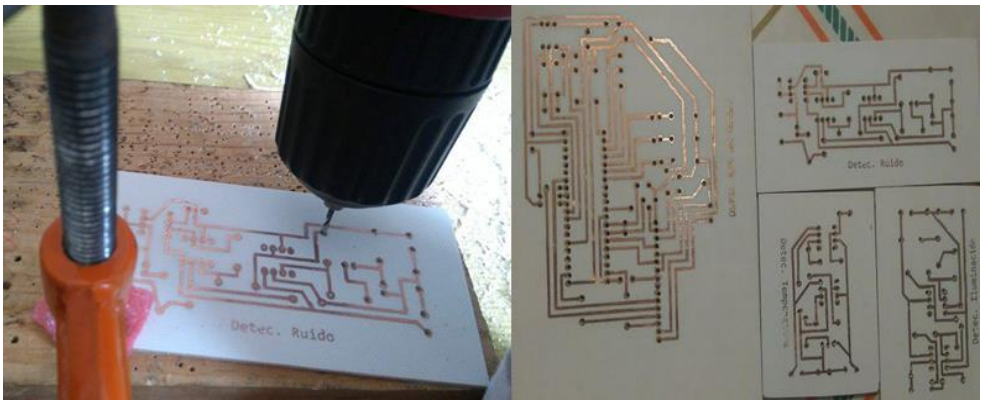


Fig. 105 Perforación de placas.

Una vez perforadas todas las placas, se soldan de manera correcta todos los componentes.

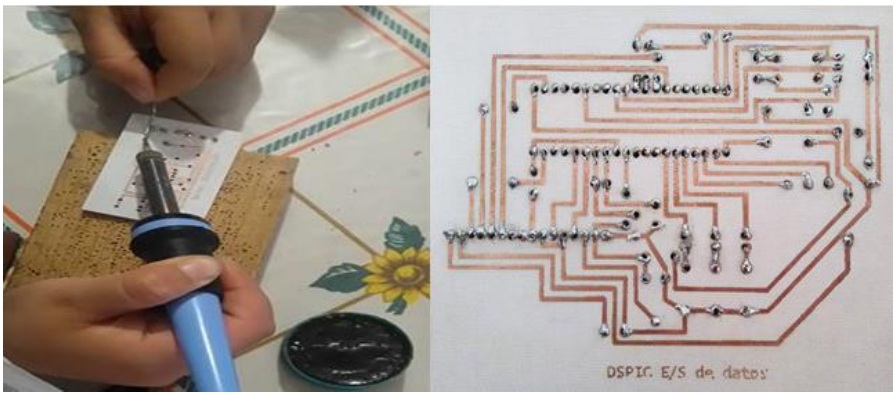


Fig. 106 Placa soldada.

Resultados.

Finalmente, sólo resta probar cada placa y en conjunto, para revisar que su funcionamiento sea correcto.

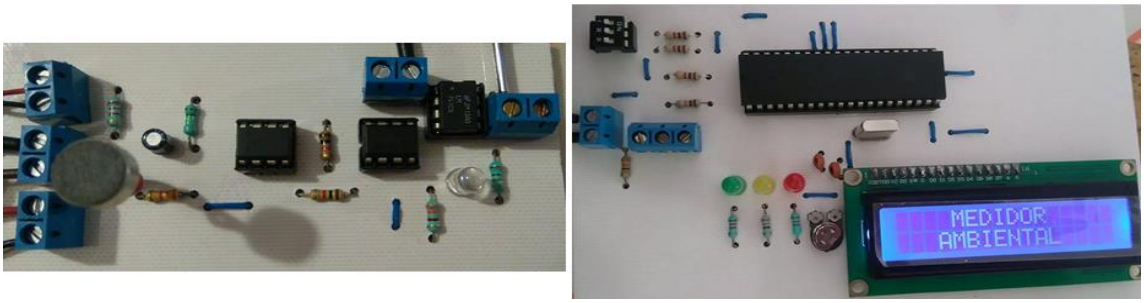


Fig. 107 Prueba de cada placa.

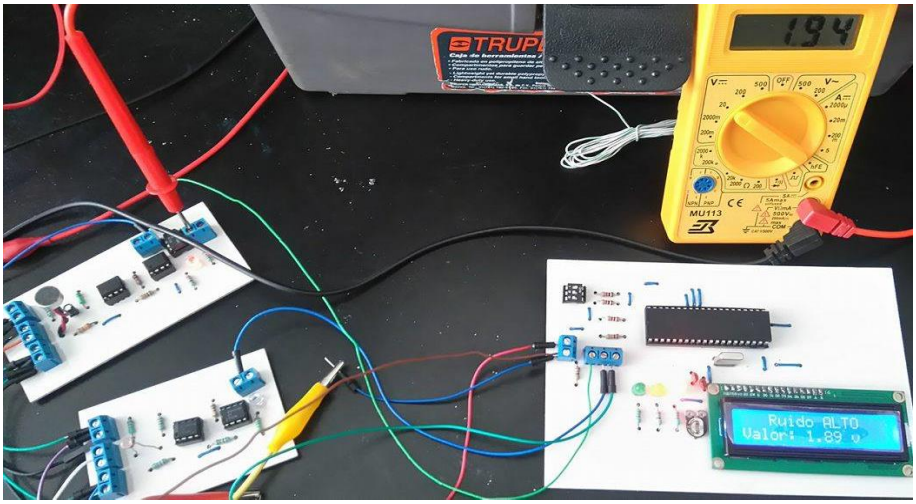


Fig. 108 Prueba en conjunto: Ruido

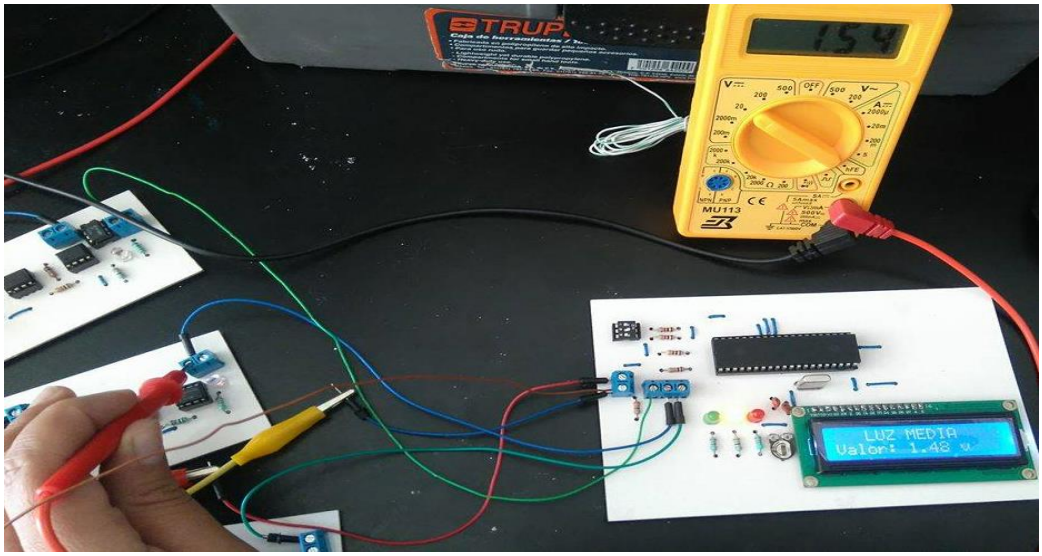


Fig. 109 Prueba en conjunto: Luz

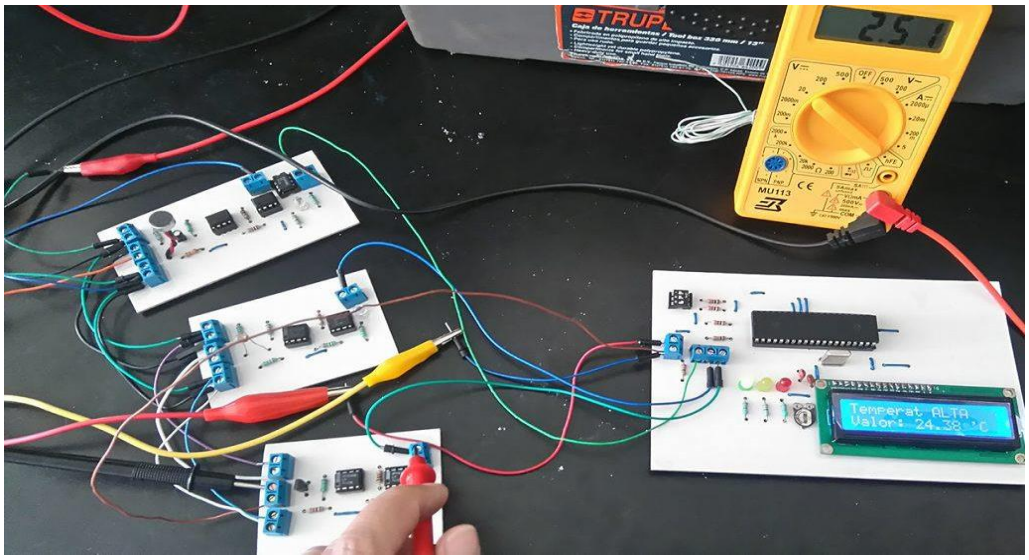


Fig. 110 Prueba en conjunto: Temperatura

Por último, todas las placas son montadas sobre una base de acrílico para una mejor presentación.

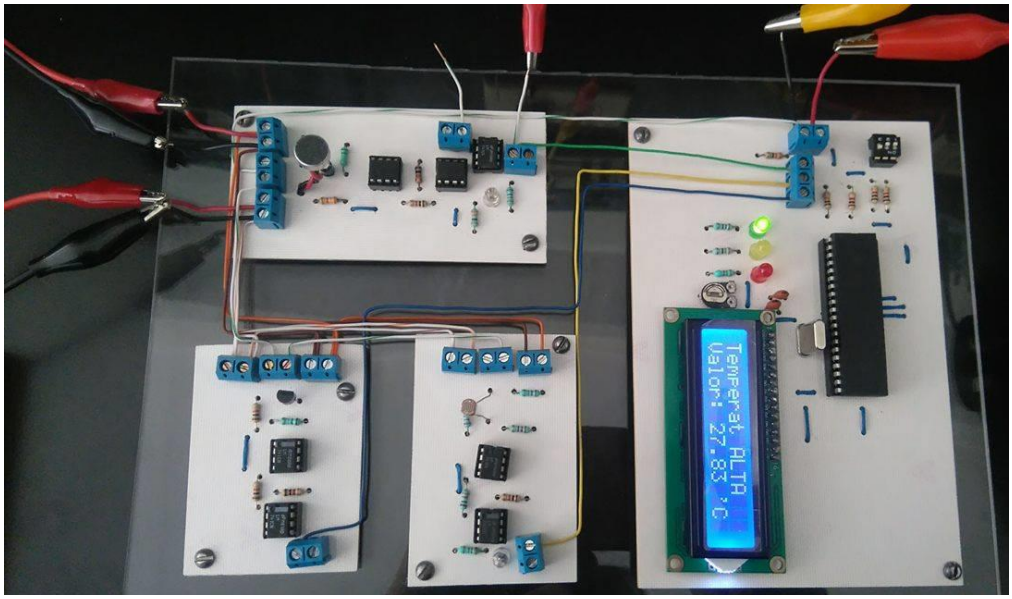


Fig. 111 Resultado final.

CAPÍTULO 4. CONCLUSIONES

Como resultado del trabajo de investigación y desarrollo del presente proyecto se llegaron a las siguientes conclusiones:

- Estar enterados de los efectos que causan en los seres humanos los factores ambientales ayuda en la prevención y toma de medidas necesarias para tener salud y un desempeño de calidad.
- Se diseñó y desarrolló un sistema que permite la medición y procesamiento (básico) de las siguientes variables ambientales: ruido, luz y temperatura.
- El sistema permite realizar la medición de las variables ambientales a lo largo del día y el resultado de la medición es visualizado mediante un display de LCD.
- El sistema cuenta con valores de referencia, introducidos por el usuario. Estos valores de referencia le permiten al sistema desplegar mensajes sobre el nivel de

la variable que está siendo medida y despliega en el display un mensaje de alerta para el usuario.

- Con el uso de sencillos circuitos basados en amplificadores operacionales y sus diferentes funciones, se pueden realizar diversos diseños que ayudan a solucionar de una manera efectiva y económica problemas cotidianos como los que se hian presentado en este proyecto.
- El circuito y programa propuestos tienen la capacidad de medir otro tipo de entradas analógicas. En caso necesario, prácticamente sólo se conectaría el sensor a la entrada analógica del sistema y ajustar los estándares de medición.
- Con base en lo anterior, existe la posibilidad de una ampliación futura en el proyecto, desarrollando un sistema de monitoreo y alarma de radiación UV.
- Es necesario profundizar en el uso del DSPIC para el procesamiento matemático de las señales analógicas con el objetivo de mejorar las capacidades del sistema.
- Estudiar y practicar con el microcontrolador dsPIC30F4013 abre un panorama amplio para la comparación de otros dispositivos y tomar las mejores opciones para con éste solucionar posibles problemas que se presentan en nuestro entorno.

X. ANEXOS

Programación del DSPIC

Para la programación de cualquier microcontrolador es indispensable utilizar una herramienta que permita transferir el programa desde el ordenador al microcontrolador. En el presente trabajo se utiliza MPLAB, el cual es una Plataforma de Desarrollo Integrada para Windows, que permite escribir el programa para los PIC (en lenguaje ensamblador o en C), crear proyectos, ensamblar o compilar, simular el programa o programar el componente.

De manera general el procedimiento de desarrollo de un proyecto con microcontroladores requiere de seguir los siguientes pasos:

1. Análisis de requerimientos. En esta etapa el diseñador debe entender las características del problema a resolver.
2. Diseño.
    a. Diagrama a bloques. En donde se identifiquen los módulos de hardware a desarrollar.
    b. Diagrama de flujo. Diagrama en donde se representa la secuencia lógica del programa a desarrollar.
3. Codificación. El diseñador realizará la codificación del programa en lenguaje ensamblador (o en lenguaje C) a través de un editor de textos.
4. Compilación. Fase en que el programa es traducido del código fuente al código máquina para que pueda ejecutarse
5. Simulación. En esta etapa se utiliza un simulador para verificar que el comportamiento funcional del programa es adecuado.
6. Programación del PIC. Se transfiere el programa al microcontrolador mediante el uso de un programador. Cuando se compila un programa en MPLAB se generan varios archivos, entre ellos se crea un archivo en lenguaje máquina con extensión *.hex*. En el presente trabajo se utiliza el programador Mini-Pic V2.2 (Fig. 112) para este proceso.

De manera general, para programar un PIC, éste es insertado en un zócalo ZIF del programador. Los datos son transferidos de distinta forma según la interfaz, ya sea serial o paralela.

En lo que se refiere a la programación del dsPIC30F4013 mediante el programador Mini-Pic V2.2 la programación se debe de hacer mediante interconexión externa de los componentes. Esta situación se presenta debido a que el Mini-Pic V2.2 no cuenta con todas las terminales de alimentación necesarias para la programación de este dispositivo en particular. De esta forma cuando se intentaba programar el DSPIC, el programador no lo reconocía.

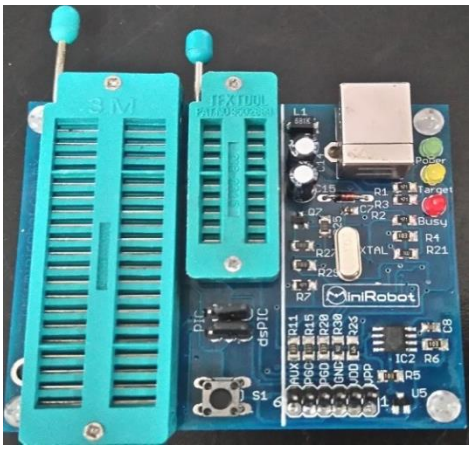


Fig. 112 Programador Mini-Pic V2.2, MiniRoobot de Microchip.

Para programar PICs o DSPICs con el Mini-Pic V2.2, se configura el dispositivo de programación mediante los conectores que se muestran con líneas rojas en la Fig. 113.

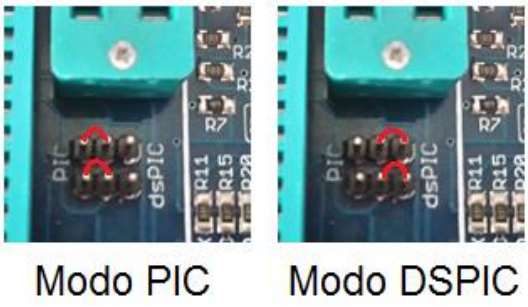


Fig. 113 Configuración del programador Mini-Pic V2.2

Una vez configurado el programador para DSPIC, se utilizan los pines de salida del programador (Fig. 114):

- VPP: MCLR, voltaje de programación (12v o 14v)
- VDD: Voltaje positivo (2.5v – 5v)
- VSS: Gnd (tierra)
- PGD: Datos (ICSPDAT)
- PGC: Clock, pulso de reloj (ICSPCLK)

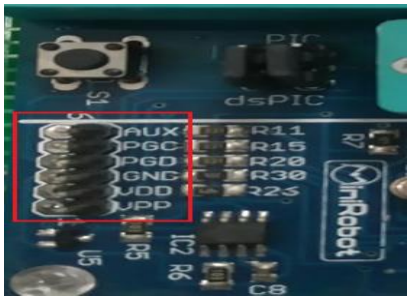


Fig. 114 Pines de salida del programador Mini-Pic V2.2

Se procede a alimentar todo el DSPIC para que el software con el que trabaja el programador reconozca completamente el microcontrolador. Para alimentarlo correctamente se debe contar con la hoja de especificaciones del dispositivo a trabajar, en este caso con el dsPIC30F4013.

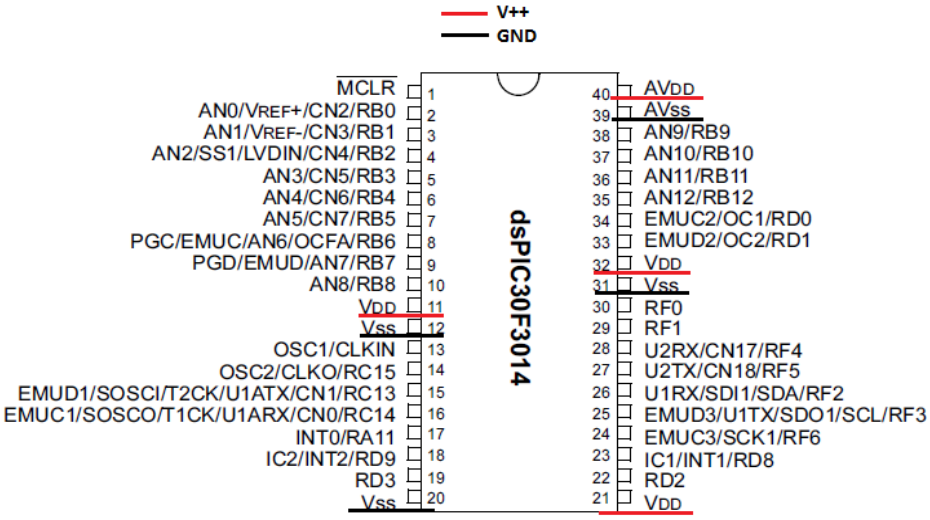


Fig. 115 Conexiones de alimentación del dsPIC30F4013.

Para realizar la conexión de los pines de salida del programador con el dispositivo se debe observar la tabla 14 donde describe la conexión del diagrama de la fig. 116 del dsPIC30F4013.

Tabla 14 Conexión de los pines del dsPIC30F4013 al programador.

No. Pin	Nombre	Conecta con:
1	MCLR	VPP
1	MCLR	Resistencia 10kΩ a VDD
8	PGC/EMUC/…	PGC
9	PGD/EMUD/…	PGD
11,21,32,40	VDD	VDD
12,20,31,39	VSS	GND

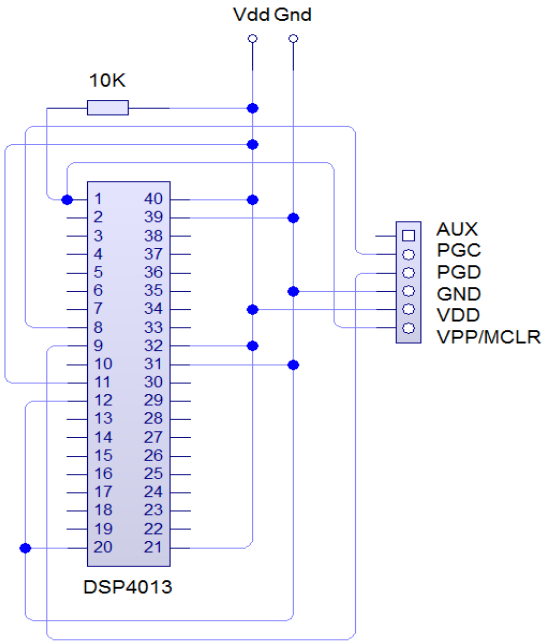


Fig. 116 Diagrama de conexión del dsPIC30F4013 con el programador.

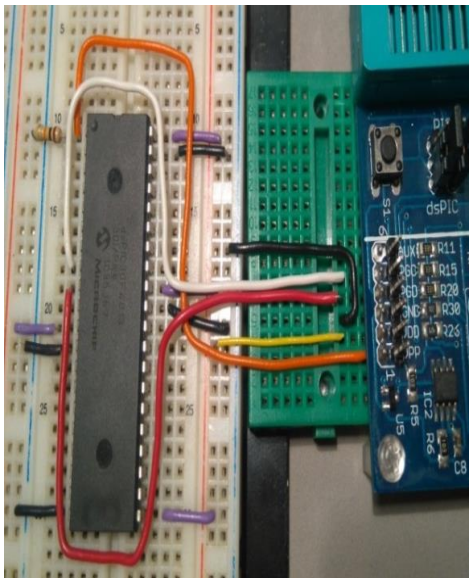


Fig. 117 Diagrama físico de la conexión del dsPIC30F4013 con el programador.

El software que utiliza el Mini-Pic V2.2 es el PICkit2 Programmer, el cual debe reconocer el dispositivo una vez que se encuentre conectado correctamente.

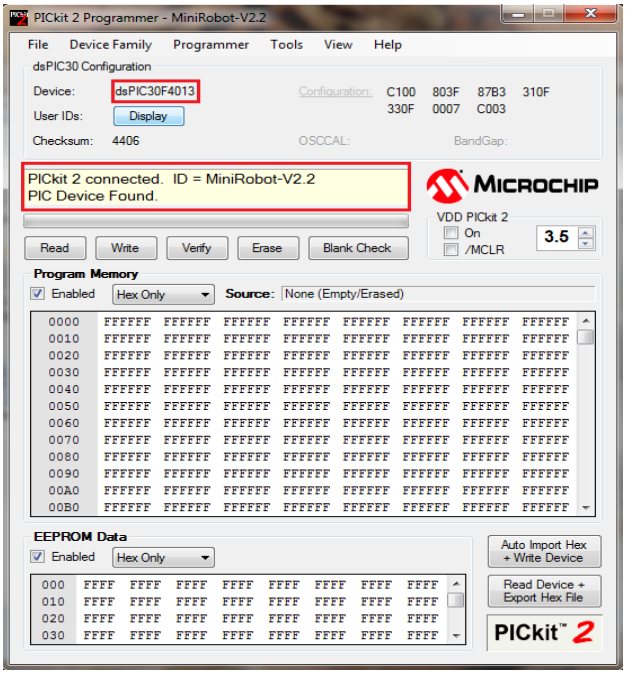


Fig. 118 Ventana principal del software PICkit2 Programmer

Librerías

Lcd.c

```
#include "p30f4013.h"   //incluye la librería del dsPIC30F4013
#include "libpic30.h" //librería del pic30 para procesamiento de señales
#include "lcd.h" //librería .h donde llama las funciones del LCD
#include "delay.h"//librería del tiempo de retardo
//Configuración de la frecuencia del tiempo
#ifndef __DELAY_H
      #define delay_us(x) __delay32(((x*FCY)/100000L)) //delay x us
      #define delay_ms(x) __delay32(((x*FCY)/10L)) //delay x ms
      #define __DELAY_H 1
#endif
//Configuración de los puertos de entrada y salida del DSPIC al LCD
#define LCD_RS    LATCbits.LATC13   // Selección del registro (RS)
#define LCD_EN    LATCbits.LATC14   // Señal de activación (E)
#define LCD_D4    LATBbits.LATB4    // Bits de Datos
#define LCD_D5    LATBbits.LATB5    // “”
#define LCD_D6    LATBbits.LATB6    // “”
#define LCD_D7    LATBbits.LATB7    // “”

#define     LCD_STROBE  ((LCD_EN = 1),(LCD_EN=0)) //Habilitar-deshabilitar E
/*Escribe un byte en el LCD en 4 bits*/
/*****************************************/
void lcd_write(unsigned char c)
{
      if(c & 0x80) LCD_D7=1; else LCD_D7=0;
      if(c & 0x40) LCD_D6=1; else LCD_D6=0;
```

```
      if(c & 0x20) LCD_D5=1; else LCD_D5=0;
      if(c & 0x10) LCD_D4=1; else LCD_D4=0;
      LCD_STROBE;
      if(c & 0x08) LCD_D7=1; else LCD_D7=0;
      if(c & 0x04) LCD_D6=1; else LCD_D6=0;
      if(c & 0x02) LCD_D5=1; else LCD_D5=0;
      if(c & 0x01) LCD_D4=1; else LCD_D4=0;
      LCD_STROBE;
      delay_ms(40);
}
/*Limpia el LCD*/
/*****************************************/
void lcd_clear(void)
{
      LCD_RS = 0;
      lcd_write(0x1);
      delay_ms(20);
}
/*Escribe una cadena de caracteres en el LCD*/
/*****************************************/
void lcd_puts(const char * s)
{
      LCD_RS = 1; // escribe los caracteres
      while(*s) lcd_write(*s++);
}
/*Escribe un solo caracter en el LCD*/
/*****************************************/
void lcd_putch(unsigned char c)
{
      LCD_RS = 1; // escribe el caracteres
      lcd_write(c);
}
/*Va a la posición específica*/
/*****************************************/
void lcd_goto(unsigned char pos)
{
      LCD_RS = 0;
      lcd_write(0x80 + pos);
}
/*Inicializa el LCD*/
/*****************************************/
void lcd_init(void)
{
   ADPCFG = 0xFFFF;     //Todos los pines del puerto B es analógico

   TRISC = 0;           //configuración de los puertos B y C
   TRISB = 0;
   PORTB = 0;
   LATB = 0;
      LCD_RS = 0;       // Escribe el control de los bytes
      delay_ms(15);          // Retraso al encender
      LCD_D4 = 1;       // Activa puertos de datos
      LCD_D5 = 1;
      LCD_STROBE;       //Enable
      delay_ms(50);
      LCD_STROBE;
      delay_ms(100);
      LCD_STROBE;
      delay_ms(50);
      LCD_D4 = 0;       //Establecer en modo de 4 bits
      LCD_STROBE;
      delay_ms(40);
      lcd_write(0x28);  //modo 4 bits, fuente 16x2, 2 lineas
      lcd_write(0x0C);  //display encendido
```

```
      lcd_write(0x06);  //modo cursor
      lcd_write(0x01);  //limpia display y resetea el cursor
}
```

Lcd.h

```
/*Funciones de la librería lcd.c*/
/*************************************/
extern void lcd_write(unsigned char);   // Escribe un byte en el LCD en 4 bits
extern void lcd_clear(void);            // Limpia el LCD
extern void lcd_puts(const char * s);   // Escribe una cadena de caracteres en
el LCD
extern void lcd_goto(unsigned char pos);// Va a la posición específica
extern void lcd_putch(unsigned char c); // Escribe un solo caracter en el LCD
extern void lcd_init(void);             // Inicializa el LCD


/*Ajusta la posición del cursor*/
/*******************************/
#define     lcd_cursor(x)   lcd_write(((x)&0x7F)|0x80)
#define LINE1           0x00  //posición de línea 1
#define LINE2           0x40  //posición de línea 2
```

Delay.h

```
#define     FOSC  400000 //Frecuencia de oscilación
#define     PLL   1       //Multiplicador   de   frecuencia   interno   en   el
dispositivo
#define     FCY   FOSC*PLL/4
//Frecuencia de Cada Instrucción, 4000000*1/4, es entre 4 porque
//todo dispositivo Microchip(pic,dspic)se ejecuta en
//4 instrucciones de reloj
```

IX. GLOSARIO

CAN: (*Controller Area Network*) es un protocolo de comunicaciones desarrollado por la firma alemana Robert Bosch GmbH, basado en una topología bus.

Cilios (oído interno): Estructuras celulares que se caracterizan por presentarse como apéndices con aspecto de pelo que contienen una estructura central altamente ordenada.

Melanoma: Tipo más serio de cáncer de piel. Con frecuencia el primer signo de un melanoma es un cambio de tamaño, forma, color o textura de un lunar.

PWM: Modulación por ancho de pulso.

RISC: *Computador con Conjunto de Instrucciones Reducidas.* Tipo de diseño de CPU generalmente utilizado en microprocesadores o microcontroladores.

VIII. Bibliografía

1. La ergonomía y el medio ambiente del trabajo. 2016, (s.f.). Recuperado de PDF Sitio web: http://tesis.uson.mx/digital/tesis/docs/4431/Capitulo6.pdf

2. Nuno J. (14 de Junio 2006). El oído humano tiene un límite para tolerar el ruido. 2017, de La Gaceta Sitio web: http://www.lagaceta.com.ar/nota/212481/salud/oido-humano-tiene-limite-para-tolerar-ruido.html

3. La Organización Mundial de la Salud. (2003). Índice UV solar mundial, Guía práctica. Biblioteca de la OMS-PDF Sitio web: http://www.who.int/uv/publications/en/uvispa.pdf

4. Dr. Méndez Flores, A. (2011). Rayos ultravioleta (UV). 2016, de Ciencias Médicas Sitio web: http://blog.ciencias-medicas.com/archives/1423

5. La Sociedad Americana Contra El Cáncer. (19 de Marzo 2015). ¿Qué es la radiación ultravioleta (UV)?. 26 de Julio 2016, de American Cancer Society Sitio web: https://www.cancer.org/es/cancer/cancer-de-piel/prevencion-y-deteccion-temprana/que-es-la-radiacion-de-luz-ultravioleta.html

6. Angulo Usategui, J. M., Etxebarría Ruiz, A., Angulo Martínez, I., Trueba Parra, I.. (2008). El mundo de los DSP. En dsPIC Diseño Práctico de Aplicaciones (4,7,9,10,18,39). México: Mc Graw Hill.

7. Floyd, Thomas L. (2008). Dispositivos Electrónicos, 8va. Edición. (69,593) México: PEARSON EDUCACIÓN.

8. Malvino, Albert P. (2000). Principios de electrónica. 6ta. Edición. (668,669,682,683, 880, 888) España: Mc Graw Hill.

9. Moreno, G. y Martínez, F. (2007). Mediciones Industriales. 2016, de blogspot Sitio web: http://martinezmorenomedicionesind.blogspot.mx/2007/06/fotoresistencia-ldr_16.html

10. (s.f.). (2009). LDR - Resistencia dependiente de la luz - Fotorresistencia. 2017, de Electrónica-electronics Sitio web: http://electronica-electronics.com/info/LDR-fotoresistencia.html

11. Robótica y Mecatrónica. (s.f.). Micrófonos Electret. 2016, de electronicaestudio.com Sitio web: http://www.electronicaestudio.com/docs/SHT-014_info.pdf

12. (s.a). (s.f). LM35. Sensor de precisión de temperatura centígrada. 2016, de PDF Sitio web: http://blog.utp.edu.co/jnsanchez/files/2011/03/LM351.pdf

13. Servín Magaña, R. (18 de Febrero 2014). Cáncer de piel, segundo lugar de incidencia en México: FMD. 2016, de El Financiero Sitio web: http://www.elfinanciero.com.mx/sociedad/cancer-de-piel-segundo-lugar-de-incidencia-en-mexico-fmd.html

14. García Álvarez J. A.. (2015). Así funciona la conversión analógico digital. 2016, de Así Funciona Sitio web: http://www.asifunciona.com/electronica/af_conv_ad/conv_ad_5.htm

15. Domene Figuerola S. (2007). Conversor A/D del dsPIC30F4013 . 2016, de PDF Sitio web: http://server-die.alc.upv.es/asignaturas/PAEEES/2006-07/Conversor%20AD%20del%20dsPIC30F4013.pdf

16. PCE Ibérica S.L. Instrumentación. (2000). Medidores de radiación. 2017, de PCE Instruments Sitio web: https://www.pce-instruments.com/espanol/instrumento-medida/medidor/medidor-de-radiacion-kat_70064_1.htm

17. TMP Equipos. (2007). Radiómetros Catálogo. 2016, de Tpm Equipos México Sitio web: http://tpmequipos.com/62740_98-medidor-de-conductividad-eco.html