



UNIVERSIDAD NACIONAL “PEDRO RUIZ GALLO”
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA



**ENTORNO DE DESARROLLO INTEGRADO Y MEJORA DE LA
INTERACCIÓN DE LOS MICROCONTROLADORES PARA OBTENER UNA
TARJETA DE ADQUISICIÓN Y PROCESAMIENTO DE DATOS PARA USO
GENERAL EN LOS LABORATORIOS DE ELECTRÓNICA.**

TESIS

**PRESENTADA PARA OPTAR EL TÍTULO PROFESIONAL DE:
INGENIERO ELECTRÓNICO**

AUTORES

Bach. MORENO RODRÍGUEZ ALAN ALFONSO
Bach. CHÁVEZ VÁSQUEZ JEYNER

ASESOR

Ing. SEGUNDO FRANCISCO SEGURA ALTAMIRANO.

LAMBAYEQUE – PERÚ
2017

TESIS PROFESIONAL SUSTENTADA POR:

Bach. Moreno Rodríguez Alan Alfonso

Bach. Chávez Vásquez Jeyner

**COMO REQUISITO PARA OBTENER EL TÍTULO PROFESIONAL DE:
INGENIERO ELECTRÓNICO**

**ACEPTADA POR LA ESCUELA PROFESIONAL DE INGENIERÍA
ELECTRÓNICA**

**Ing. Manuel Ramírez Castro
Presidente**

**Ing. Oscar Romero Cortez
Secretario**

**Ing. Chaman Cabrera Lucia
Vocal**

**Ing. Francisco Segura Altamirano
Asesor**

**LAMBAYEQUE – PERÚ
2017**

Dedicatoria

A Dios.

Por haberme permitido llegar hasta este punto y haberme dado salud para lograr mis objetivos, además de su infinita bondad y amor.

A mi madre Martha.

Por haberme apoyado en todo momento, por sus consejos, sus valores, por la motivación constante que me ha permitido ser una persona de bien, pero más que nada, por su amor.

A mi padre Wagner.

Por los ejemplos de perseverancia y constancia que lo caracterizan y que me ha infundado siempre, por el valor mostrado para salir adelante y por su amor.

Alan

Dedico esta tesis.

A mis padres quienes me apoyaron todo el tiempo, a mis maestros que me enseñaron a ponerle esfuerzo en todo.

A los sinodales quienes estudiaron mi tesis y la aprobaron.

A todos los que me apoyaron para escribir y concluir esta tesis. Para ellos
es esta dedicatoria de tesis, pues es a ellos a quienes se las debo por su apoyo incondicional.

Jeyner

Agradecimientos

A la Universidad Nacional Pedro Ruiz Gallo y a sus catedráticos que nos impartieron en las aulas todos sus conocimientos y mostraron también sus valores permitiendo así una formación integral en nosotros y en varios de sus alumnos.

De manera muy especial quiero agradecer al **Ing.Segundo Francisco Segura Altamirano** asesor de tesis y maestro que por varios años impartió sus conocimientos y que por ultimo aportó con criterios, consejos y tiempo valioso para la culminación de este trabajo.

Prefacio

Los Microcontroladores son dispositivos electrónicos que están presentes en nuestro trabajo, en nuestra casa y en nuestras vidas en general. se pueden encontrar controlando el funcionamiento de los ratones y teclados de computadora, en los teléfonos, en hornos microondas y en los televisores de nuestro hogar, por eso están importante el empleo y desarrollo óptimo de sistemas embebidos usando estos dispositivos.

Aprender a programarlos eficientemente estos dispositivos conlleva mucho tiempo y horas de estudio sobre la arquitectura del CPU. ya que el estudiante debe aprender a configurar correctamente los diferentes registros y periféricos en sus diferentes bancos de memoria que cuenta el Microcontrolador.

El lenguaje natural de programación es el ensamblador que es una representación simbólica de los códigos binarios y otras constantes necesarias para programar los microcontroladores. Actualmente este lenguaje se utiliza con frecuencia en ambientes de investigación, especialmente cuando se requiere la manipulación directa del hardware para obtener un alto rendimiento.

Existen lenguajes de programación de alto nivel como el C++, que facilitan programar los microcontroladores, lo que hacen es traducir el programa escrito en C++ a lenguaje ensamblador, este proceso se llama compilación. En este tipo de compiladores también es necesario conocer algo de la arquitectura del CPU para empezar a programar los chips de los micorcontroladores.

Los estudiantes profesionales necesitan un entorno de desarrollo integrado que sea capaz de acceder a la arquitectura del CPU de manera sencilla y eficiente, optimizando el código del programa para no excederse en la capacidad de la memoria flash y así desarrollar sistemas embebidos más complejos a través de una placa de adquisición y procesamiento de datos.

Resumen

Se desarrolló un software en Python llamado Entorno de desarrollo Integrado, quien será el encargado de interpretar el programa escrito en C++, para ser compilado y descargado automáticamente al microcontrolador.

El hardware fue diseñado de manera industrial, esta tarjeta de adquisición y procesamiento de datos cuenta con un microcontrolador de la gama 18 de Microchip. El circuito de la placa cuenta con todas las protecciones contra las corrientes parasitarias y la sobre tensión en cada uno de sus pines de salida o entrada. El usuario podrá programar esta tarjeta de manera fácil y eficiente sin necesidad de un hardware extra para subir el programa en la memoria flash del microcontrolador, el resultado se podrá observar al instante que se descarga el programa a la tarjeta.

El entorno de desarrollo integrado cuenta con una vasta gama de librerías que son interpretadas eficientemente al lenguaje ensamblador haciendo que este ocupe menos espacio en la memoria flash de la tarjeta electrónica y además facilitando el acceso a sus registros y periféricos propia de su arquitectura del CPU.

ABSTRACT

It has developed a software on Python , it called Integrated Development Environment , who will be responsible for interpreting the program written in C ++ , to be compiled and automatically downloaded to the microcontroller.

The hardware was designed industrially , this card acquisition and data processing has a range microcontroller 18 microcontrollers. The circuit board has all protections against parasitic currents and over- voltage in each of its output or input pin . The user can program this card easily and efficiently without the need for extra hardware to upload the program in the flash memory of the microcontroller , the result can be seen immediately that the program is downloaded to the card.

INDICE GENERAL

1	Aspectos Generales	1
1.1	Título del proyecto	1
1.2	Definición del Problema	1
1.3	Formulación de hipótesis	1
1.4	Objetivos del Proyecto	1
1.4.1	Objetivo General	1
1.4.2	Objetivos Específicos	1
1.5	Justificación e Importancia del Proyecto	2
1.6	Justificación Teórica	2
1.7	Justificación Practica	3
1.8	Justificación Metodológica	3
2	ANTECEDENTES	4
2.1	ANTECENDE I	4
2.2	ANTECENDE II	5
2.3	ANTECENDE III	6
3	MARCO TEORICO	7
3.1	INTRODUCCION	7
3.2	MICROCONTROLADOR	8
3.2.1	ARQUITECTURA DE UINA COMPUTADORA	10
3.2.2	DETALLES DE UN MICROCONTROLADOR	11
3.2.3	PERIFERICOS DE UN MICROCONTROLADOR.	15
3.2.4	COMUNICACIÓN SERIAL RS-232	17
3.2.5	I2C (COMUNICACIÓN SERIAL)	18
3.2.6	COMUNICACIÓN POR UNIVERSAL SERIAL BUS	20
3.2.7	FAMILIAS DE MICROCONTROLADORES	22
3.2.8	CONVERTIDORES ANALÓGICOS – DIGITALES	23
3.2.9	OTROS PUERTOS DE COMUNICACIÓN:	23
3.3	MICROCONTROLADORES PIC	24
3.3.1	BREVE RESEÑA HISTORICA	24
3.3.2	CARACTERISTICAS RELEVANTES	25
3.3.3	RECURSOS AUXILIARES.	26
3.3.4	LA FAMILIA DE LOS PIC	27

3.3.5	LOS LENGUAJES USADOS PARA MICROCONTROLADORES	30
3.3.6	PROGRAMACIÓN DEL PIC	32
3.4	PYTHON.....	32
3.4.1	USANDO EL INTÉRPRETE DE PYTHON.....	33
3.4.2	MODO INTERACTIVO	35
3.4.3	EL INTÉRPRETE Y SU ENTORNO	35
3.4.4	PYTHON Y MICROCONTROLADOR.....	38
3.4.5	INTERFACES GRAFICAS (GUI) EN PYTHON	38
3.5	PROTEUS DESING SUITE 8	42
3.6	SDCC (SMALL DEVICE C COMPILER) EN WINDOWS	44
3.6.1	BOOTLOADER EN MICROCONTROLADOR	45
3.7	SISTEMAS DE ADQUISICION DE DATOS	46
4	INTRODUCCIÓN	50
4.1	DISEÑO DE UNA INTERFACE GRAFICA	50
4.2	DIAGRAMA DE FLUJO DE LOS BOTONES DE LA INTERFACE GRAFICA.....	51
4.2.1	BOTON NUEVO	51
4.2.2	BOTON GUARDAR	52
4.2.3	BOTÓN DE COMPILAR CÓDIGO FUENTE	53
4.2.4	BOTON DESCARGAR HADWARE	54
4.3	BOOTLOADER DEL MICROCONTROLADOR.....	55
4.4	DISEÑO DEL CIRCUITO DE ADQUISICION DE DATOS	55
4.4.1	DISEÑO DEL CIRCUITO DE LA TARJETA DE ADQUISICIÓN DE DATOS	56
5	EJECUCIÓN DE PRUEBAS	60
5.1	CREAR UNA APLICACIÓN	61
5.2	ENCUESTA.....	70
5.3	INTERPRETACION DE LOS DATOS.....	70
6	CONCLUSIONES.....	72
6.1	RECOMENDACIONES.....	73
6.2	TRABAJOS FUTUROS	73
7	INSTALACION DE PYTHON	74
7.1	GRABAR EL BOOTLOADER	77
7.2	INSTALAR DRIVER DEL HADWARE.....	79
7.3	DESCRIPCION DE FUNCIONES	83
7.3.1	Funciones generales	83
7.3.2	Funciones de Comunicación.....	84

7.3.3 Algunas funciones propias del compilador SDCC.....93

INDICE DE FIGURAS

FIG 1 ESQUEMA DE UN MICROCONTROLADOR.....	9
FIG 2 MICROCONTROLADOR CONEXIÓN USB.....	20
FIG 3 MICROCONTROLADOR CONEXIÓN USB.....	21
FIG 4 ARQUITECTURA HARVARD DEL MICROCONTROLADOR PIC	25
FIG 5 SEGMENTACIÓN PIPE-LINE	25
FIG 6 PYTHON INICIO	35
FIG 7 EJEMPLO DE PYTHON CONDICIONAL.....	35
FIG 8 CODIGO EJECUTABLE PYTHON	35
FIG 9 PERMISOS A LOS ARCHIVO DE PYTHON	36
FIG 10 0 EJEMPLO DE CALCULADOR DE PYTHON	36
FIG 11 EJEMPLO DE VARIABLES TIPO ENTERO.....	37
FIG 12 EJEMPLO DE VARIABLES TIPO CADENAS.....	37
FIG 13 EJEMPLO DE VARIABLES TIPO LISTA	37
FIG 14 EJEMPLO DE INDICE DE LISTA.....	38
FIG 15INTRODUCCION A PROTEUS.....	42
FIG 16 ETAPAS DE PROTEUS	42
FIG 17FASE DE PROTEUS	43
FIG 18 DIAGRAMA DE FLUJO DE COMPILADO PARA MICROCONTROLADOR PIC.....	44
FIG 19 COMUNICACIÓN CON EL BOOTLOADER.....	46
FIG 20 DIAGRAMA GENERAL DE UN SISTEMA DE ADQUISICIÓN DE DATOS	49
FIG 21 INTERFACE GRÁFICA DEL USUARIO.....	51
FIG 22 DIAGRAMA DE FLUJO DEL BOTÓN NUEVO	51
FIG 23DIAGRAMA DE FLUJO DEL BOTÓN GUARDAR	52
FIG 24 DIAGRAMA DE FLUJO DEL BOTÓN COMPILAR	53
FIG 25 DIAGRAMA DE FLUJO DEL BOTÓN DESCARGAR HARDWARE.....	54
FIG 26 DIAGRAMA DEL MICROCONTROLADOR PIC18F2550	56
FIG 27 CIRCUITO DE LA TARJETA DE ADQUISICIÓN DE DATOS.....	57
FIG 28 ESQUEMA DEL CIRCUITO IMPRESO ARRIBA	58
FIG 29 ESQUEMA DEL CIRCUITO IMPRESO ABAJO	58
FIG 30 CIRCUITO ENSAMBLADO	59
FIG 31 INDICADORES.....	60
FIG 32 MÓDULOS VARIOS	60
FIG 33 IR SENSOR	61
FIG 34 VENTANA NUEVO ARCHIVO	61
FIG 35 ESTRUCTURA BÁSICA DE UNA APLICACIÓN.....	61
FIG 36 DIAGRAMA DE FLUJO PARA EL EJEMPLO 01	62
FIG 37 CÓDIGO FUENTE DEL EJEMPLO 01.	63
FIG 38 CIRCUITO DE CONEXIÓN PARA EL EJEMPLO 01.	63
FIG 39 EJEMPLO 1 CIRCUITO	64
FIG 40 DIAGRAMA DE FLUJO PARA EL EJEMPLO 02	64
FIG 41 CÓDIGO FUENTE DEL EJEMPLO 02	65
FIG 42 CIRCUITO DE CONEXIÓN PARA EL EJEMPLO 02.	65
FIG 43 EJEMPLO 2 CIRCUITO	66

FIG 44 DIAGRAMA DE FLUJO PARA EL EJEMPLO 03.	66
FIG 45 CÓDIGO FUENTE DEL EJEMPLO 02.	67
FIG 46 CIRCUITO DE CONEXION PARA EL EJEMPLO 03	67
FIG 47 CIRCUITO DEL EJEMPLO 03	68

CAPITULO I

Aspectos Generales del Proyecto

1 Aspectos Generales

1.1 Título del proyecto

Entorno de desarrollo Integrado y mejora de la interacción de los microcontroladores para obtener una tarjeta de adquisición y procesamiento de datos para uso general en los laboratorios de Electrónica.

1.2 Definición del Problema

¿Cómo se puede mejorar la interacción con los registros, y periféricos de entrada y salida de los microcontroladores, para obtener una tarjeta de adquisición y procesamiento de datos, veloz y altamente configurable.?

1.3 Formulación de hipótesis

Si se diseña un entorno de desarrollo integrado (IDE), para la programación de un microcontrolador, en donde se pueda interactuar con los registros, y periféricos de entrada y salida de los microcontroladores se obtendrá una tarjeta de adquisición y procesamiento de datos veloz y altamente configurable.

1.4 Objetivos del Proyecto

1.4.1 Objetivo General

Diseñar e implementar un entorno de desarrollo integrado (IDE), para la programación de un microcontrolador, en donde se pueda interactuar con los registros, y periféricos de entrada y salida de los microcontroladores, para obtener una tarjeta de adquisición y procesamiento de datos veloz y altamente configurable.

1.4.2 Objetivos Específicos

- ✓ Estudiar las arquitecturas de los microcontroladores para determinar sus capacidades de adquisición y procesamiento.

- ✓ Diseñar e implementar la placa de adquisición y procesamiento de datos con los microcontroladores seleccionados.
- ✓ Desarrollar un entorno amigable para la programación del microcontrolador que maximice la capacidad de adquisición y procesamiento del microcontrolador.
- ✓ Realizar prueba de desempeño de la tarjeta de adquisición y procesamiento de datos en el desarrollo de los laboratorios del curso de Telecomunicaciones I y Control I.

1.5 Justificación e Importancia del Proyecto

El estudiante de ingeniería electrónica, durante su formación profesional desarrolla diversos prototipos electrónicos y sistemas embebidos, pero para poder desarrollar este tipo de aplicaciones necesitan un elevado conocimiento de programación de microcontroladores, es por eso que muchos de los estudiantes utilizan la tarjeta “arduino” por su flexibilidad tanto en el hardware como en el software, pues esta tarjeta está diseñada para aficionados y no para profesionales de ingeniería electrónica que en muchos casos necesitamos configurar los registros y periféricos a nuestra necesidad para obtener el mayor rendimiento posible del microcontrolador.

Con esta perspectiva he puesto intereses en diseñar que sea capaz de interactuar y programar en el menor tiempo posible sin perder todo el potencial que el microcontrolador nos ofrece, accediendo a los registros y periféricos en forma fácil y en el menor tiempo posible.

Esta tarjeta de adquisición de datos tendrá como núcleo la familia de microcontroladores 18FXX, específicamente como primera versión el uC 18F2550, con el objetivo principal de proponer una nueva técnica digital en el estudiando y así de esta manera lograr que alumnos puedan desarrollar proyectos de mayor proyección en esta área de la MICROELECTRONICA.

1.6 Justificación Teórica

Desde el punto de vista el avance vertiginoso de la tecnología abre un mundo de posibilidades para el desarrollo de prototipos electrónicos,

pues facilita la interacción y programación del microcontrolador en forma eficiente y rápida maximizando la eficiencia que estos dispositivos nos brinda.

1.7 Justificación Practica

Al utilizar esta tarjeta de adquisición de datos los estudiantes podrán desarrollar en menor tiempo las diversas aplicaciones que nuestra formación profesional requiere con buen resultado.

1.8 Justificación Metodológica

Con el uso de esta plataforma tendrán una visión más clara sobre la programación de microcontroladores pues aprenderán a programarlos en un lenguaje de alto nivel, accediendo a los registros y periféricos con funciones programadas.

CAPITULO II

2 ANTECEDENTES

2.1 ANTECEDENTE I

- **TIPO DE ANTECEDENTE**

- Internacional

- **TITULO**

Diseño y Construcción de una Tarjeta Programable de Adquisición, Procesamiento de Datos y Control.

- **AUTOR**

- Feliz Vicente Jiménez
- Joaquín Riveros Juárez

- **LUGAR**

Centro Nacional De Investigación y Desarrollo Tecnológico – Cuernavaca México.

- **AÑO**

- 2006

- **RESUMEN**

La Automatización Industrial se divide en 3 fases: adquisición de datos, procesamiento o tratamiento de la información y control de elementos actuadores, existen alternativas comerciales que permiten realizar una o todas las fases de automatización en un solo componente como tarjetas de adquisición de datos o controladores lógicos programables.

En este trabajo se presenta el diseño, construcción y prueba de una tarjeta programable capaz de realizar las 3 funciones de automatización de un proceso con la capacidad de trabajar de forma autónoma. Se diseñó y fabricó una tarjeta de aplicación para la realización de las pruebas de los módulos de: entradas/salidas

digitales, entradas analógicas, potencia para motores, comunicación con la computadora y memorias EEPROM.

2.2 ANTECEDENTE II

- **TIPO DE ANTECEDENTE**

- Internacional

- **TITULO DEL PROYECTO**

Diseño y construcción de un sistema de adquisición de datos de 4 canales analógicos de entrada basado en un PIC16F877 para uso general en los laboratorios del Centro de investigación avanzada e ingeniería industrial (CIAII)

- **AUTOR**

- Victor Hiram Ibarra Garcia
- Pedro Osorio Osorio

- **LUGAR**

Universidad Autónoma del Estado de Hidalgo.

- **AÑO**

- 2007

- **RESUMEN**

Tesis de grado previa a la obtención del título de Ingeniería Electrónica y Telecomunicaciones. En este trabajo de tesis se desarrolla un sistema de Adquisición de datos, el cual permite la medición de parámetros físicos de Temperatura, Voltaje AC, Voltaje DC y Corriente AC siendo luego visualizados los resultados de manera gráfica y legible en una computadora a través de un software de instrumentación virtual (LABVIEW). El sistema de adquisición de datos propuesto consta de una etapa de acondicionamiento de señal, una etapa de adquisición de datos y una etapa de programación.

2.3 ANTECEDENTE III

- **TIPO DE ANTECEDENTE**

- Nacional

- **TITULO**

Diseño y Construcción de Tarjetas Entrenadoras para Aplicaciones con Microcontroladores Pic, Motorola, Atmel y FPGA para el Entrenamiento En las Diferentes Areas de los Alumnos de la Escuela Profesional De Ingeniería Electrónica.

- **AUTOR**

- Euler Deza Figueroa

- **LUGAR**

Universidad Nacional Pedro Ruiz Gallo

- **AÑO**

- 2009

- **RESUMEN**

En el presente trabajo de investigación plasmo las soluciones más óptimas para el Diseño y Construcción de Tarjetas Entrenadoras para Aplicaciones Con Microcontroladores PIC, Motorola ATMEL y FPGA Para El Entrenamiento en Las Diferentes áreas de Los Alumnos de la Escuela Profesional de Ingeniería Electrónica de la UNPRG, que son en realidad un complemento a las tarjetas de aprendizaje existentes en dicho Laboratorio. El objetivo es hacer de ellas, una herramienta básica del alumnado, y porque no decirlo del ingeniero dedicado a la investigación con estas tecnologías, ya que por medio de sus diferentes interfaces permitirá expandir su uso a áreas como el Control, Electrónica de Potencia, Robótica, Telecomunicaciones, Sistemas Digitales Avanzados, entre otras.

CAPITULO III

3 MARCO TEORICO

3.1 INTRODUCCION

Hoy en día en muchos laboratorios de investigación, al igual que en la industria, se realizan actividades de medición, prueba y automatización de distintos procesos y aplicaciones industriales, Por lo que los sistemas de Adquisición de datos (SAD) basados en computadoras personales (PC) han tomado gran importancia. Por tal motivo grandes empresas como national instrument, Hewlet Packar, Advantech, etc, han incorporado dentro de sus productos diferentes sistemas de adquisición de señales como tarjetas SA-PCI, tarjetas SAD-PCMCIA, módulos SAD RS232 y RS-485.

Todo sistema de adquisición de datos cuenta con tres partes principales las cuales son:

- La etapa de acondicionamiento de señal (Hardware)
- El Dispositivo de adquisición de datos (Hardware)
- La interfaz gráfica para la visualización en la PC de datos adquiridos (Software)

Los sistemas comerciales de adquisición de datos son de gran costo debido a que son dispositivos que garantizan una gran exactitud y precisión en la medición de procesos en los que es necesario contar con un mínimo margen de error. Sin embargo, un sistema de adquisición de datos comercial genera una gran dependencia tecnológica tanto en la etapa de acondicionamiento como en el dispositivo de adquisición ya que se requiere soporte técnico y mantenimiento del sistema en total. También existe dependencia en el aspecto del software en el cual sirve para la manipulación, procesamiento y visualización de los datos adquiridos, debido a que la tecnología ha hecho que el control de las tarjetas sea complicado y no tan transparente para un estudiante.

3.2 MICROCONTROLADOR

Hace unos años, los sistemas de control se implementaban usando exclusivamente lógica de componentes, lo que hacía que fuesen dispositivos de gran tamaño y muy pesados. Para facilitar una velocidad más alta y mejorar la eficiencia de estos dispositivos de control, se trató de reducir su tamaño, apareciendo así los microprocesadores. Siguiendo con el proceso de miniaturización, el siguiente paso consistió en la fabricación de un controlador que integrase todos sus componentes en un sólo chip. A esto es a lo que se le conoce con el nombre de microcontrolador, un computador dentro de un sólo chip. El primer microprocesador fue el Intel 4004, producido en 1971. Se desarrolló originalmente para una calculadora, y resultaba revolucionario para su época. Contenía 2300 transistores en un microprocesador de 4 bits que sólo podía realizar 60000 operaciones por segundo. El primer microprocesador de 8 bits fue el Intel 8008, desarrollado en 1979 para su empleo en terminales informáticos. El Intel 8008 contenía 3300 transistores. El primer microprocesador realmente diseñado para uso general, desarrollado en 1974, fue el Intel 8080 de 8 bits, que contenía 4500 transistores y podía ejecutar 200 000 instrucciones por segundo. Los microprocesadores modernos tienen una capacidad y velocidad mucho mayores. Entre ellos figuran el Intel Pentium Pro, con 5.5 millones de transistores; el UltraSparc-II, de Sun Microsystems, que contiene 5.4 millones de transistores; el PowerPC G4, desarrollado conjuntamente por Apple, IBM y Motorola, con 7 millones de transistores, y el Alpha 21164A, de Digital Equipment Corporation, con 9.3 millones de transistores.

La historia de los microcontroladores surge desde dos vías de desarrollo paralelas; una desde Intel y otra desde Texas Instruments. Los primeros microcontroladores son el 4004 y 4040 de Intel que dieron lugar al 8048, a su vez predecesor del 8051. Aun así, el primer microcontrolador fue el TMS1000 de Texas Instruments. Éste integraba un reloj, procesador, ROM, RAM, y soportes de E/S en un solo chip.

En 1965, la empresa GI creó una división de microelectrónica, GI Microelectronics División, que comenzó su andadura fabricando memorias EPROM y EEPROM, que conformaban las familias AY3-XXXX y AY5-XXXX. A principios de los años 70 diseñó el microprocesador de 16 bits CP1600, razonablemente bueno pero que no manejaba eficazmente las Entradas y Salidas. Para solventar este problema, en 1975 diseñó un chip destinado a controlar E/S: el PIC (Peripheral Interface Controller). Se trataba de un controlador rápido pero limitado y con pocas instrucciones pues iba a trabajar en combinación con el CP1600.

La arquitectura del PIC, que se comercializó en 1975, era sustancialmente la misma que la de los actuales modelos PIC16C5X. En

aquel momento se fabricaba con tecnología NMOS y el producto sólo se ofrecía con memoria ROM y con un pequeño pero robusto microcódigo.

La década de los 80 no fue buena para GI, que tuvo que reestructurar sus negocios, concentrando sus actividades en los semiconductores de potencia. La GI Microelectronics División se convirtió en una empresa subsidiaria, llamada GI Microelectronics Inc. Finalmente, en 1985, la empresa fue vendida a un grupo de inversores de capital de riesgo, los cuales, tras analizar la situación, rebautizaron a la empresa con el nombre de Arizona Microchip Technology y orientaron su negocio a los PIC, las memorias EPROM paralelo y las EEPROM serie. Se comenzó rediseñando los PIC, que pasaron a fabricarse con tecnología CMOS, surgiendo la familia de gama baja PIC16CSX, considerada como la "clásica".

Una de las razones del éxito de los PIC se basa en su utilización. Cuando se aprende a manejar uno de ellos, conociendo su arquitectura y su repertorio de instrucciones, es muy fácil emplear otro modelo. Microchip cuenta con su factoría principal en Chandler, Arizona, en donde se fabrican y prueban los chips con los más avanzados recursos técnicos. En 1993 construyó otra factoría de similares características en Tempe, Arizona. También cuenta con centros de ensamblaje y ensayos en Taiwan y Tailandia. Para tener una idea de su alta producción, hay que tener en cuenta que ha superado el millón de unidades por semana en productos CMOS de la familia PIC16CSX.

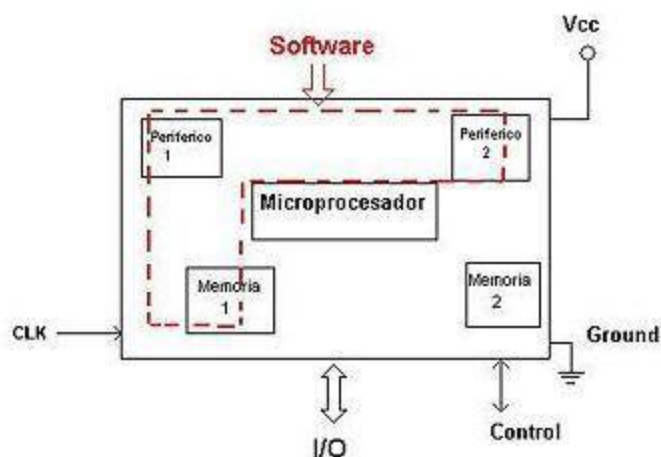


FIG 1 Esquema de un Microcontrolador

3.2.1 ARQUITECTURA DE UNA COMPUTADORA

Básicamente existen dos arquitecturas de computadoras, y por supuesto, están presentes en el mundo de los microcontroladores: Von Neumann y Harvard. Ambas se diferencian en la forma de conexión de la memoria al procesador y en los buses que cada una necesita. La arquitectura Von Neumann es la que se utiliza en las computadoras personales, para ella existe una sola memoria, donde coexisten las instrucciones de programa y los datos, accedidos con un bus de dirección, uno de datos y uno de control.

Debemos comprender que en una PC, cuando se carga un programa en memoria, a éste se le asigna un espacio de direcciones de la memoria que se divide en segmentos, de los cuales típicamente tendremos los siguientes: código (programa), datos y pila. Es por ello que podemos hablar de la memoria como un todo, aunque existan distintos dispositivos físicos en el sistema (HDD, RAM, CD, FLASH).

En el caso de los microcontroladores, existen dos tipos de memoria bien definidas: memoria de datos (típicamente algún tipo de SRAM) y memoria de programas (ROM, PROM, EEPROM, FLASH u de otro tipo no volátil). En este caso la organización es distinta a la de las PC, porque hay circuitos distintos para cada memoria y normalmente no se utilizan los registros de segmentos, sino que la memoria está segregada y el acceso a cada tipo de memoria depende de las instrucciones del procesador.

A pesar de que en los sistemas embebidos con arquitectura Von Neumann la memoria esté segregada, y existan diferencias con respecto a la definición tradicional de esta arquitectura; los buses para acceder a ambos tipos de memoria son los mismos, del procesador solamente salen el bus de datos, el de direcciones, y el de control. Como conclusión, la arquitectura no ha sido alterada, porque la forma en que se conecta la memoria al procesador sigue el mismo principio definido en la arquitectura básica. Esta arquitectura es la variante adecuada para las PC, porque permite ahorrar una buena cantidad de líneas de E/S, que son bastante costosas, sobre todo para aquellos sistemas como las PC, donde el procesador se monta en algún tipo de socket alojado en una placa madre (motherboard). También esta organización les ahorra a los diseñadores de motherboards una buena cantidad de problemas y reduce el costo de este tipo de sistemas.

La otra variante es la arquitectura Harvard, y por excelencia la utilizada en *supercomputadoras*, en los microcontroladores, y sistemas embebidos en general. En este caso, además de la memoria, el procesador tiene los buses segregados, de modo que cada tipo de memoria tiene un bus de datos, uno de direcciones y uno de control.

La ventaja fundamental de esta arquitectura es que permite adecuar el tamaño de los buses a las características de cada tipo de memoria; además, el procesador puede acceder a cada una de ellas de forma simultánea, lo que se traduce en un aumento significativo de la velocidad de procesamiento, típicamente los sistemas con esta arquitectura pueden ser dos veces más rápidos que sistemas similares con arquitectura Von Neumann. La desventaja está en que consume muchas líneas de E/S del procesador; por lo que en sistemas donde el procesador está ubicado en su propio encapsulado, solo se utiliza en supercomputadoras. Sin embargo, en los microcontroladores y otros sistemas embebidos, donde usualmente la memoria de datos y programas comparten el mismo encapsulado que el procesador, este inconveniente deja de ser un problema serio y es por ello que encontramos la arquitectura Harvard en la mayoría de los microcontroladores.

3.2.2 DETALLES DE UN MICROCONTROLADOR

Ahora comenzaremos a ver cómo es que está hecho un MICROCONTROLADOR, no será una explicación demasiado detallada porque desde su invención éste ha tenido importantes revoluciones propias, pero hay aspectos básicos que no han cambiado y que constituyen la base de cualquier microcontrolador.

- ✓ **REGISTROS:** Son un espacio de memoria muy reducido pero necesario para cualquier microprocesador, de aquí se toman los datos para varias operaciones que debe realizar el resto de los circuitos del procesador. Los registros sirven para almacenar los resultados de la ejecución de instrucciones, cargar datos desde la memoria externa o almacenarlos en ella. Mientras mayor sea el número de bits de los registros de datos del procesador, mayores serán sus prestaciones, en cuanto a poder de cómputo y velocidad de ejecución, ya que este parámetro determina la potencia que se puede incorporar al resto de los componentes del sistema, por ejemplo, no tiene sentido tener una ALU de 16 bits en un procesador de 8 bits.
- ✓ **UNIDAD DE CONTROL:** La unidad de control es uno de los elementos fundamentales que determinan las prestaciones del procesador, ya que su tipo y estructura, determina parámetros tales como el tipo de conjunto de instrucciones, velocidad de ejecución, tiempo del ciclo de máquina, tipo de buses que puede tener el sistema, manejo de interrupciones y un buen número de cosas más

que en cualquier procesador van a para a este bloque. Por supuesto, las unidades de control, son el elemento más complejo de un procesador en ella recae la lógica necesaria para la decodificación y ejecución de las instrucciones, el control de los registros, la ALU, los buses y cuanto cosa más se quiera meter dentro del procesador, normalmente están divididas en unidades más pequeñas trabajando de conjunto. La unidad de control agrupa componentes tales como la unidad de decodificación, unidad de ejecución, controladores de memoria cache, controladores de buses, controladores de interrupción, pipelines, entre otros elementos, dependiendo siempre del tipo de procesador.

✓ **CONJUNTO DE INSTRUCCIONES:**

Define las operaciones básicas que puede realizar el procesador, que conjugadas y organizadas forman lo que conocemos como software. El conjunto de instrucciones viene siendo como las letras del alfabeto, el elemento básico del lenguaje, que organizadas adecuadamente permiten escribir palabras, oraciones y cuanto programa se le ocurra. Existen dos tipos básicos de repertorios de instrucciones, que determinan la arquitectura del procesador: CISC y RISC.

CISC (*Complex Instruction Set Computer*): Computador con juego complejo de instrucciones. Un procesador cuyo núcleo está basado en el concepto CISC en su repertorio de instrucciones, no necesita de compiladores costosos ni complejos en mejora de sus prestaciones. El enfoque de este concepto es el desarrollo de lenguajes de alto nivel (HLL: *High Level Language*). Este concepto también configura programas más cortos y de mejor aprovechamiento de la memoria. Se ha visto que algunos algoritmos, de moderada y alta complejidad, se desarrollan mucho más rápido en una máquina CISC que en una RISC.

Dentro de los microcontroladores CISC podemos encontrar a la popular familia **INTEL -51** y la **Z80**, aunque actualmente existen versiones CISC-RISC de estos microcontroladores, que pretenden aprovechar las ventajas de los procesadores RISC a la vez que se mantiene la compatibilidad hacia atrás con las instrucciones de tipo CISC. **RISC (*Reduced Instruction Set Computer*):** Computador con juego reducido de instrucciones. Generalmente este tipo de instrucciones son ejecutadas en un ciclo de la máquina, utilizando modos de direccionamiento simples e instrucciones sencillas. El concepto de segmentación (ejecución de varias instrucciones en el mismo ciclo de máquina) es más fácil aplicarlo a las instrucciones RISC debido a que éstas tienen un ancho en bits constante. Una de

las características más destacables de este tipo de procesadores es que posibilitan el paralelismo en la ejecución, y reducen los accesos a memoria. Dentro de los microcontroladores RISC podemos encontrar a la popular familia **PIC**.

✓ **MEMORIA:**

Anteriormente habíamos visto que la memoria en los microcontroladores debe estar ubicada dentro del mismo encapsulado, esto es así la mayoría de las veces, porque la idea fundamental es mantener el grueso de los circuitos del sistema dentro de un solo integrado. En los microcontroladores la memoria no es abundante, aquí no encontrará Gigabytes de memoria como en las computadoras personales. Típicamente la memoria de programas no excederá de 16 K-localizaciones de memoria no volátil para instrucciones y la memoria RAM ni siquiera llegará a exceder los 5 Kilobytes.

La memoria RAM está destinada al almacenamiento de información temporal que será utilizada por el procesador para realizar cálculos u otro tipo de operaciones lógicas. En el espacio de direcciones de memoria RAM se ubican además los registros de trabajo del procesador y los de configuración y trabajo de los distintos periféricos del microcontrolador. El tipo de memoria utilizada en las memorias RAM de los microcontroladores es SRAM, lo que evita tener que implementar sistemas de refrescamiento como en el caso de las computadoras personales, que utilizan gran cantidad de memoria, típicamente alguna tecnología DRAM

En el caso de la memoria de programas se utilizan diferentes tecnologías, y el uso de una u otra depende de las características de la aplicación a desarrollar, a continuación, se describen las cinco tecnologías existentes, que mayor utilización tienen o han tenido, hasta el momento:

- ✓ **ROM de máscara.** En este caso no se “graba” el programa en memoria, sino que el microcontrolador se fabrica con el programa, es un proceso similar al de producción de los CD comerciales mediante masterización. El costo inicial de producir un circuito de este tipo es alto, porque el diseño y producción de la máscara es un proceso costoso, sin embargo, cuando se necesitan varios miles o incluso cientos de miles de microcontroladores para una aplicación determinada, como por ejemplo, algún electrodoméstico.

- ✓ **OTP One Time Programmable.** Este tipo de memoria, también es conocida como PROM o simplemente ROM. Los microcontroladores con memoria OTP se pueden programar una sola vez, con algún tipo de programador. Se utilizan en sistemas donde el programa no requiera futuras actualizaciones y para series relativamente pequeñas.
- ✓ **PROM Erasable Programmable Read Only Memory.**

Los microcontroladores con este tipo de memoria son muy fáciles de identificar porque su encapsulado es de cerámica y llevan encima una ventanita de vidrio desde la cual puede verse la oblea de silicio del microcontrolador.

- ✓ **EEPROM Electrical Erasable Programmable Read Only Memory.** Fueron el sustituto natural de las memorias EPROM, la diferencia fundamental es que pueden ser borradas eléctricamente, por lo que la ventanilla de cristal de cuarzo y los encapsulados cerámicos no son necesarios. Otra característica destacable de este tipo de microcontrolador es que fue en ellos donde comenzaron a utilizarse los sistemas de programación en circuito o ICSP (In Circuit Serial Programming) que evitan tener que sacar el microcontrolador de la tarjeta que lo aloja para hacer actualizaciones al programa.
- ✓ **Flash.** En el campo de las memorias reprogramables para microcontroladores, son el último avance tecnológico en uso a gran escala, y han sustituido a los microcontroladores con memoria EEPROM. A las ventajas de las memorias FLASH se le adicionan su gran densidad respecto a sus predecesoras lo que permite incrementar la cantidad de memoria de programas a un costo muy bajo. Pueden además ser programadas con las mismas tensiones de alimentación del microcontrolador, el acceso en lectura y la velocidad de programación es superior, disminución de los costos de producción, entre otras. Lo más habitual es encontrar que la memoria de programas y datos está ubicada toda dentro del microcontrolador, de hecho, actualmente son pocos los microcontroladores que permiten conectar memoria de programas en el exterior del encapsulado.

- ✓ **INTERRUPCIONES: Una interrupción** consiste en un mecanismo por el cual un evento interno o externo puede interrumpir la ejecución de

un programa en cualquier momento. A partir de entonces se produce automáticamente un salto a una subrutina de atención a la interrupción, esta atiende inmediatamente el evento y retorna luego a la ejecución del programa exactamente en donde estaba en el momento de ser interrumpido, continuando su tarea justamente donde lo dejo. La interrupción tiene la característica de la inmediatez, nace de la necesidad de ejecutar una subrutina en el instante preciso y, por tanto, se considera su intervención urgente. Este método es más eficaz que la técnica polling ya que el microcontrolador no perderá el tiempo preguntando a la línea de entrada para leer el estado, sino que únicamente el atenderá al periférico cuando este se lo pida mediante la solicitud de interrupción. Las interrupciones constituyen el mecanismo más importante para la conexión del microcontrolador con el exterior ya que sincroniza la ejecución de programas con los acontecimientos externos.

3.2.3 PERIFERICOS DE UN MICROCONTROLADOR.

A continuación, describiremos algunos de los periféricos que con mayor frecuencia encontraremos en los microcontroladores.

- ✓ **ENTRADAS Y SALIDAS DE PROPÓSITO GENERAL:** También conocidos como puertos de E/S, generalmente agrupadas en puertos de 8 bits de longitud, permiten leer datos del exterior o escribir en ellos desde el interior del microcontrolador, el destino habitual es el trabajo con dispositivos simples como relés, LED, o cualquier otra cosa que se le ocurra al programador. Algunos puertos de E/S tienen características especiales que le permiten manejar salidas con determinados requerimientos de corriente, o incorporan mecanismos especiales de interrupción para el procesador. Típicamente cualquier pin de E/S puede ser considerada E/S de propósito general, pero como los microcontroladores no pueden tener infinitos pines, ni siquiera todos los pines que queramos, las E/S de propósito general comparten los pines con otros periféricos. Para usar un pin con cualquiera de las características a él asignadas debemos configurarlo mediante los registros destinados a ello.
- ✓ **TEMPORIZADORES Y CONTADORES:** Son circuitos sincrónicos para el conteo de los pulsos que llegan a su entrada de reloj. Si la fuente de conteo es el oscilador interno del microcontrolador es común que no tengan un pin asociado, y en este caso trabajan como temporizadores. Por otra parte, cuando la fuente de conteo es externa, entonces

tienen asociado un pin configurado como entrada, este es el modo contador. Los temporizadores son uno de los periféricos más habituales en los microcontroladores y se utilizan para muchas tareas, como, por ejemplo, la medición de frecuencia, implementación de relojes, para el trabajo de conjunto con otros periféricos que requieren una base estable de tiempo entre otras funcionalidades. Es frecuente que un microcontrolador típico incorpore más de un temporizador/contador e incluso algunos tienen arreglos de contadores. Como veremos más adelante este periférico es un elemento casi imprescindible y es habitual que tengan asociada alguna interrupción. Los tamaños típicos de los registros de conteo son 8 y 16 bits, pudiendo encontrar dispositivos que solo tienen temporizadores de un tamaño o con más frecuencia con ambos tipos de registro de conteo.

- ✓ **PUERTO SERIE SINCRÓNICO:** Este tipo de periférico se utiliza para comunicar al microcontrolador con otros microcontroladores o con periféricos externos conectados a él, mediante las interfaces SPI (Serial Peripheral Interface) o I2C (Inter-Integrated Circuit). A pesar de que es también un tipo de puerto serie, es costumbre tratarlo de forma diferenciada respecto a la UART/USART porque las interfaces SPI e I2C aparecieron mucho después que la UART/USART, su carácter es únicamente síncronico y no están diseñadas para interconectar al sistema con otros dispositivos independientes como una PC, sino para conectar al microcontrolador dispositivos tales como memorias, pantallas LCD, conversores A/D o D/A.
- ✓ **OTROS PUERTOS DE COMUNICACIÓN:** En un mundo cada vez más orientado a la interconexión de dispositivos, han aparecido muchas interfaces de comunicación y los microcontroladores no se han quedado atrás para incorporarlas, es por ello que podemos encontrar algunos modelos con puertos USB (Universal Serial Bus), CAN (Controller Area Network), Ethernet, puerto paralelo entre otros.
- ✓ **COMPARADORES:** Son circuitos analógicos basados en amplificadores operacionales que tienen la característica de comparar dos señales analógicas y dar como salida los niveles lógicos „0” o „1” en dependencia del resultado de la comparación. Es un periférico muy útil para detectar cambios en señales de entrada de las que solamente nos interesa conocer cuando está en un rango determinado de valores.

- ✓ **MODULADOR DE ANCHO DE PULSOS:** Los PWM (Pulse Width Modulator) son periféricos muy útiles sobre todo para el control de motores, sin embargo hay un grupo de aplicaciones que pueden realizarse con este periférico, dentro de las cuales podemos citar: la conversión digital analógica D/A, el control regulado de luz (dimming) entre otras.
- ✓ **MEMORIA DE DATOS NO VOLÁTIL:** Muchos microcontroladores han incorporado este tipo de memoria como un periférico más, para el almacenamiento de datos de configuración o de los procesos que se controlan. Esta memoria es independiente de la memoria de datos tipo RAM o la memoria de programas, en la que se almacena el código del programa a ejecutar por el procesador del microcontrolador. Muchos de los microcontroladores PIC, incluyen este tipo de memoria, típicamente en forma de memoria EEPROM, incluso algunos de ellos permiten utilizar parte de la memoria de programas como memoria de datos no volátil, por lo que el procesador tiene la capacidad de escribir en la memoria de programas como si ésta fuese un periférico más.

3.2.4 COMUNICACIÓN SERIAL RS-232

El puerto serial, también conocido por el estándar que lo norma, el RS-232, fue creado con el único propósito de contar con una interfaz entre los equipos terminales de datos (Data Terminal Equipment, DTE), y el equipo de comunicación de datos (Data Communications Equipment, DCE) empleando intercambio serial de datos binarios. De esta forma el equipo terminal de datos es el extremo cliente de los datos y el equipo de comunicación de datos es el dispositivo que se encarga de la unión entre los terminales, tal como un módem o algún otro dispositivo de comunicación. [6] El RS-232 fue originalmente adoptado en 1960 por la Asociación de Industrias de la Electrónica, conocida también por sus siglas en inglés EIA, Electronic Industries Association. El estándar evolucionó a través de los años y en 1969 el protocolo RS-232C, fue el estándar elegido por los fabricantes de computadoras personales compatibles con IBM. En 1987 se adoptó la cuarta revisión, el RS-232D, también conocida como EIA-232D. En esta nueva revisión se agregaron 3 líneas de prueba.

En teoría un enlace serial podría requerir de sólo dos cables, una línea de señal y una tierra, para mover la señal serial de una locación a otra. Pero en la práctica esto no funciona correctamente al paso del tiempo ya que algunos bits pueden perder el nivel de la señal, alterando el resultado final. Un bit faltante en la terminal de recepción puede provocar

que todos los bits siguientes sean cambiados o recorridos, resultando en datos incorrectos al convertirlos de regreso a una señal paralela. Por lo tanto, para conseguir una comunicación serial confiable se deben de prevenir estos errores de bit que pueden emerger en varias formas distintas. En la actualidad los microcontroladores comerciales PIC ya proporcionan los circuitos para manejar internamente el protocolo RS-232. Es decir que únicamente por programación y un cable serial entre el PIC y a la computadora es posible realizar la comunicación entre estos dos dispositivos.

3.2.5 I2C (COMUNICACIÓN SERIAL)

I2C PIC es uno de los modos de trabajo del módulo SSP puerto serial síncrono del microcontrolador PIC, en la comunicación I2C se utilizan 2 hilos a lo que se conoce como bus I2C, a estos hilos se conectan los dispositivos que se puedan comunicar mediante el protocolo I2C, por uno de los hilos se enviará una señal de reloj para la sincronización y por el otro hilo se enviarán o recibirán datos, se pueden conectar varios dispositivos de los que uno de ellos será el maestro, es el que generará la señal de reloj además de decidir cuando se inicia o finaliza la comunicación y si la comunicación será de recepción o transmisión de datos, los demás dispositivos conectados al bus I2C se conocen como esclavos.

Cada uno de los dispositivos tiene una dirección, cuando el maestro necesita comunicarse con alguno de los esclavos lo hará enviando la dirección del esclavo a través del bus I2C, cuando el esclavo reciba su dirección podrá comunicarse con el maestro, el maestro además tiene que enviar un bit mediante el cual le indica al esclavo si quiere enviarle un dato o quiere recibir un dato del esclavo.

La velocidad de comunicación puede ser de hasta 100Kbps en el modo estándar o normal, aunque puede llegar en el modo rápido hasta los 400Kbps y en la alta hasta más de 3Mbps, esto dependerá de los dispositivos utilizados, para ello es recomendable revisar sus hojas de datos.

La comunicación I2C se realiza de la siguiente manera

- La señal del pin SDA del maestro pasa de un alto a un bajo mientras la señal su pin SCL esté a un alto, esto es el inicio de la comunicación I2C, al iniciar la comunicación se genera una señal de reloj por el pin SCL del maestro y se empieza a enviar los datos por el pin SDA en grupos de 8 bits o un byte, que es lo mismo.
- El maestro enviará primero la dirección de identificación del esclavo con el que se quiera comunicar junto con el bit de escritura lectura, si el maestro quiere enviar o escribir un dato este bit será 0, si el maestro

quiere leer o recibir un dato este bit será 1, la dirección del esclavo normalmente será de 7 bits aunque también hay direcciones de 10bits.

- Cada vez que el esclavo reciba un byte el esclavo responderá enviando un bit en bajo o 0 al maestro para indicar que se ha establecido la comunicación, este es un bit de confirmación de hay comunicación y se le llama ACK, si la comunicación no se establece el valor del bit ACK será un 1.
- Luego el maestro enviará el dato si lo que se quiere es enviar o escribir un dato en el el registro del esclavo, el esclavo responderá enviando al maestro el bit de confirmación ACK; pero si lo que se quiere es que el maestro recibirá o lea algún dato desde algún registro del esclavo, ocurrirá que si la recepción es correcta es maestro enviará al esclavo el bit de confirmación ACK.
- Luego el maestro enviará el dato si lo que se quiere es enviar o escribir un dato en el registro del esclavo, el esclavo responderá enviando al maestro el bit de confirmación ACK; o de lo contrario se recibirá algún dato si se quiere leer o recibir un dato del registro del esclavo, si la recepción es correcta es maestro enviará al esclavo el bit de confirmación ACK.
- Por último y para finalizar la comunicación la señal del pin SDA pasará de un bajo a un alto mientras la señal de reloj en el pin SCL este en un alto, en ese momento se deja de generar la señal de reloj y la comunicación habrá terminado.

Las conexiones tienen que hacerse de tal manera que los nombres de los pines coincidan, en la siguiente imagen se muestra cómo será la conexión para la comunicación I2C PIC, al ser utilizado el PIC como maestro, con otros dispositivos capaces de comunicarse con el protocolo I2C.

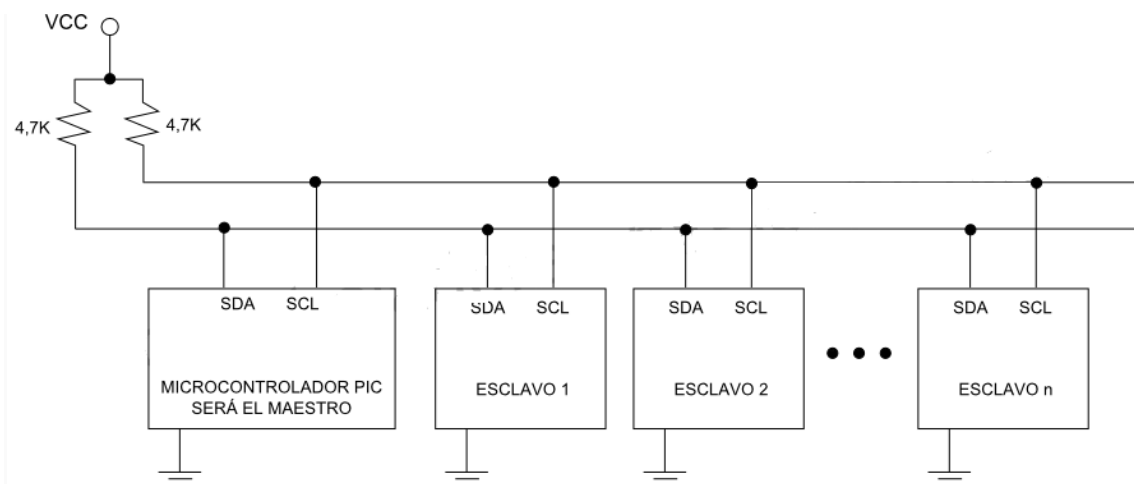


FIG 2 Microcontrolador Conexión USB

La intención de esta sección es ver cómo utilizar el modo de trabajo I2C PIC del módulo SSP para hacerlo trabajar con otros dispositivos, para más información sobre la comunicación I2C se puede visitar este enlace, este otro y muchos más.

3.2.6 COMUNICACIÓN POR UNIVERSAL SERIAL BUS

Tecnología USB: En la última década se ha venido dando un importante giro en la interconectividad de periféricos con los PC, de tal manera que interfases como el puerto paralelo, serial, entre otros, han desaparecido de los computadores. En su reemplazo, la interfase USB (*Universal Serial Bus*) se ha convertido en la manera más popular para conectar dispositivos periféricos a un computador personal o a otros dispositivos periféricos*.

USB aparece en su versión 1.0 en Enero de 1996, después de un gran tiempo en evolución y con un sin número de problemas, que para el año 1998 habían sido solucionados en la versión 1.1. Para el año 2000, se presenta la versión 2.0 libre de errores y con un nuevo conector llamado mini B.

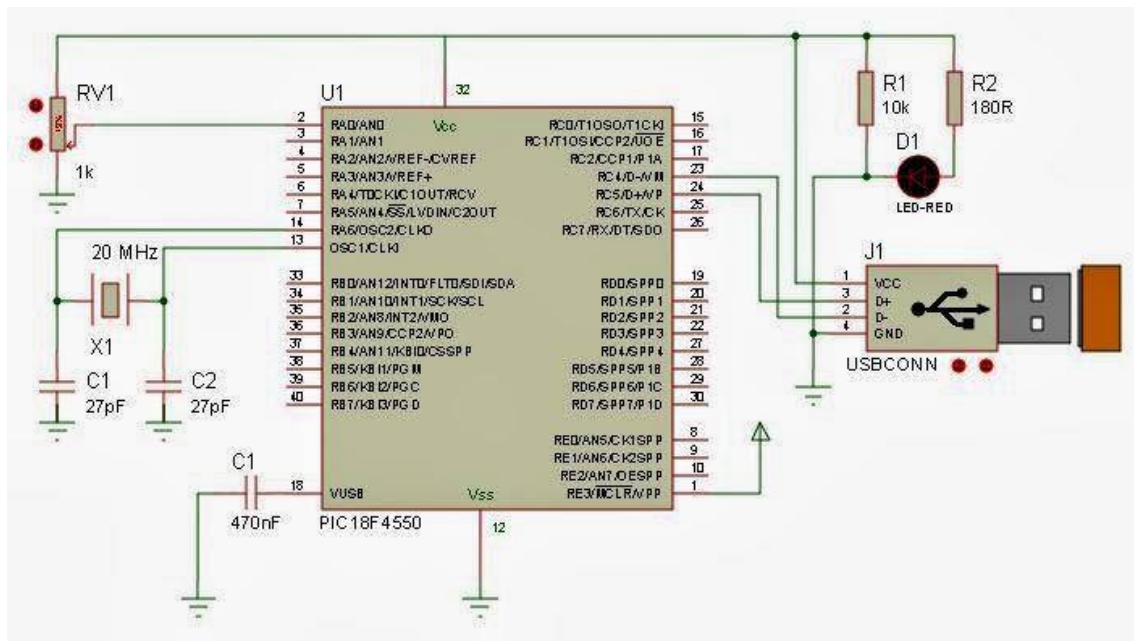


FIG 3 Microcontrolador Conexión USB

La configuración de cada dispositivo en el bus es automática (el sistema operativo lo detecta e instala el *driver* apropiado), sin necesidad de reiniciar el sistema. El usuario no necesita configurar el dispositivo para que este sea reconocido, desde el punto de vista de configuración de IRQ's o direcciones al interior del PC. La conectividad es simple respecto de la expansión del sistema, sin necesidad de abrir el PC e instalar tarjetas adicionales, basta con adicionar HUB's (concentradores). Es posible conectar y desconectar dispositivos sin apagar el sistema para su reconocimiento (*plug and play – hot pluggable*).

➤ **Velocidad:** Se pueden tener tres rangos de velocidad, que superan con creces las velocidades de los buses seriales asíncronos (RS232) e igualan o hasta superan los paralelos. Claro que mientras más dispositivos están colgados al bus, menor será la velocidad promedio de comunicación. Las tres velocidades mencionadas son:

- **HS (High Speed):** Hasta 480Mbps, para cámaras y otros dispositivos que manejan alto tráfico de información a grandes velocidades.
- **FS (Full Speed):** Hasta 12Mbps, para el reemplazo de puertos seriales y paralelos tradicionales, como también para sistemas embebidos para microcontroladores por ejemplo.
- **LS (Low Speed):** Hasta 1.5Mbps, para dispositivos con cables poco apantallados y bastante flexibles, como teclados y ratones.

- **Confiabilidad:** Diseño simple de su hardware, con el fin de minimizar los errores por ruido aditivo y protocolos robustos, que pueden detectar errores y pedir retransmisiones.
- **Bajo costo:** Cables y componentes de muy bajo costo, por su diseño simple y limpio.
- **Bajo consumo:** Con su característica de poder entrar en modos de ahorro de energía (modos de WAIT, SLEEP o STANDBY) y sólo responder cuando sea requerido. Característica bastante importante en sistemas soportados por batería, en donde cada milivatio cuenta.

➤ ***Pero USB también presenta limitaciones y que a continuación se mencionan:***

- **Ausencia en el soporte de los sistemas viejos:** Hoy en día se consiguen interfases entre los viejos sistemas, como RS232, RS485, puerto paralelo, entre otros, y el sistema USB, pero no siempre funcionan bien en las aplicaciones de los usuarios.
- **Límites en la velocidad:** USB no puede competir con buses de velocidades mayores a 400Mbps, como la **IEEE-1394b** que tiene tasas de tranferencia de hasta 3.2Gbps y otras más rápidas de actualidad.
- **Límites de distancia:** USB ha sido diseñado para trabajo en escritorio, en donde distancias de hasta 5mts son muy buenas, pero no podría competir con los viejos sistemas como RS232 y 485, que desarrollan distancias de varios cientos de metros. USB podría desarrollar hasta 30 metros, pero necesita el gasto adicional de los HUB"s.

3.2.7 FAMILIAS DE MICROCONTROLADORES

Los microcontroladores más comunes en uso son:

Empresa	8 bits	16 bits	32 bits
Atmel AVR	ATmega8,89Sx xxx familia similar 8051	ATmega16	

Freescale (antes Motorola)	68HC05, 68HC08, 68HC11, HCS08	68HC12, 68HCS12, 68HCSX12, 68HC16	683xx, PowerPC Architecture,C oldFire
Hitachi, Ltd	H8	X	X
Holtek	HT8		
Intel	MCS-48 (familia 8048) MCS51 (familia 8051) 8xC251	MCS96, MXS296	X
National Semiconductor	COP8	X	X
Microchip	Familia 10f2xx Familia 12Cxx Familia 12Fxx, 16Cxx y 16Fxx 18Cxx y 18Fxx	PIC24F, PIC24H y dsPIC30FXX,dsPIC33F con motor dsp integrado	PIC32
NEC Corporation	78K		
Parallax			
STMicroelectronics	ST 62,ST 7		

TABLA 3.1 Familia de Microcontrolador

3.2.8 CONVERTIDORES ANALÓGICOS – DIGITALES

El desarrollo de la tecnología en los últimos años ha permitido que dentro de los microcontroladores se incluyan convertidores analógicos - digitales que permiten hacer la conversión de una señal analógica en un código binario. Un convertidor analógico – digital (CAD) toma un voltaje analógico de entrada y después de cierto tiempo produce un código digital de salida que representa la entrada analógica. El proceso de conversión analógica – digital por lo general es más complejo y tardado que el proceso digital – analógico. Se han desarrollado y empleado muchos métodos para poder convertir señales analógicas en digitales entre los cuales podemos mencionar el método de conversión de doble pendiente, conversión por red de escaleras y conversión de aproximaciones sucesivas.

3.2.9 OTROS PUERTOS DE COMUNICACIÓN:

En un mundo cada vez más orientado a la interconexión de dispositivos, han aparecido muchas interfaces de comunicación y los microcontroladores no se han quedado atrás para incorporarlas, es por ello

que podemos encontrar algunos modelos con puertos USB (Universal Serial Bus), CAN (Controller Area Network), Ethernet, puerto paralelo entre otros.

3.3 MICROCONTROLADORES PIC

3.3.1 BREVE RESEÑA HISTORICA

En 1965, la empresa **GI** creó una división de microelectrónica, GI Microelectronics División, que comenzó su andadura fabricando memorias EPROM y EEPROM, que conformaban las familias AY3-XXXX y AY5-XXXX. A principios de los años 70 diseñó el microprocesador de 16 bits CP1600, razonablemente bueno pero que no manejaba eficazmente las Entradas y Salidas. Para solventar este problema, en 1975 diseñó un chip destinado a controlar E/S: el PIC (Peripheral Interface Controller). Se trataba de un controlador rápido pero limitado y con pocas instrucciones pues iba a trabajar en combinación con el CP1600. La arquitectura del PIC, que se comercializó en 1975, era sustancialmente la misma que la de los actuales modelos PIC16C5X. En aquel momento se fabricaba con tecnología NMOS y el producto sólo se ofrecía con memoria ROM y con un pequeño pero robusto microcódigo.

La década de los 80 no fue buena para GI, que tuvo que reestructurar sus negocios, concentrando sus actividades en los semiconductores de potencia. La GI Microelectronics División se convirtió en una empresa

subsidiaria, llamada GI Microelectronics Inc. Finalmente, en 1985, la empresa fue vendida a un grupo de inversores de capital de riesgo, los cuales, tras analizar la situación, rebautizaron a la empresa con el nombre de Arizona Microchip Technology y orientaron su negocio a los PIC, las memorias EPROM paralelo y las EEPROM serie. Se comenzó rediseñando los PIC, que pasaron a fabricarse con tecnología CMOS, surgiendo la familia de gama baja PIC16CSX, considerada como la "clásica".

Una de las razones del éxito de los PIC se basa en su utilización. Cuando se aprende a manejar uno de ellos, conociendo su arquitectura y su repertorio de instrucciones, es muy fácil emplear otro modelo. Microchip cuenta con su factoría principal en Chandler, Arizona, en donde se fabrican y prueban los chips con los más avanzados recursos técnicos. En 1993 construyó otra factoría de similares características en Tempe, Arizona. También cuenta con centros de ensamblaje y ensayos en Taiwán y Tailandia. Para tener una idea de su alta producción, hay que tener en

cuenta que ha superado el millón de unidades por semana en productos CMOS de la familia PIC16CSX.

3.3.2 CARACTERISTICAS RELEVANTES

1ª. La arquitectura del procesador sigue el modelo Harvard.* En esta arquitectura, el CPU se conecta de forma independiente y con buses distintos con la memoria de instrucciones y con la de datos.

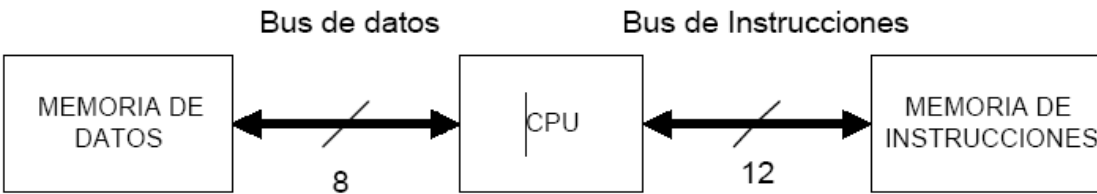


FIG 4 ARQUITECTURA HARVARD DEL MICROCONTROLADOR PIC

La arquitectura Harvard permite al CPU acceder simultáneamente a las dos memorias. Además, propicia numerosas ventajas al funcionamiento del sistema. **2ª. Se aplica la técnica de segmentación (“pipe-line”) en la ejecución de las instrucciones.** La segmentación permite al procesador realizar al mismo tiempo la ejecución de una instrucción y la búsqueda del código de la siguiente. De esta forma se puede ejecutar cada instrucción en un ciclo (un ciclo de instrucción equivale a cuatro ciclos de reloj).

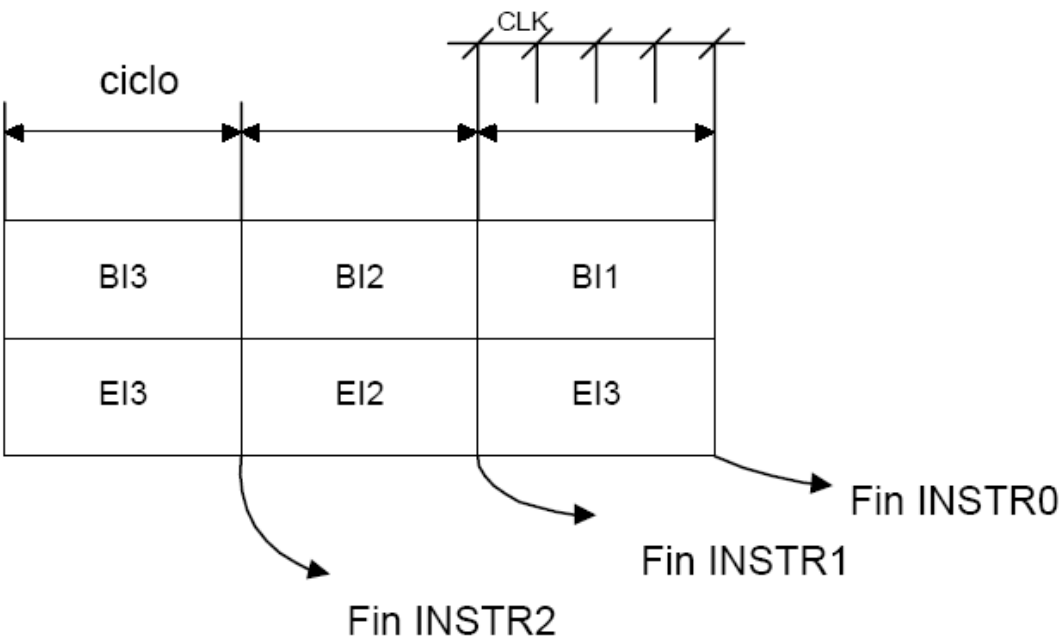


FIG 5 SEGMENTACIÓN PIPE-LINE

La segmentación permite al procesador ejecutar cada instrucción en un ciclo de instrucción equivalente a cuatro ciclos de reloj. En cada ciclo se realiza la búsqueda de una instrucción y la ejecución de la anterior. Las instrucciones de salto ocupan dos ciclos al no conocer la dirección de la siguiente instrucción hasta que no se haya completado la de bifurcación.

3ª. El formato de todas las instrucciones tiene la misma longitud.

Todas las instrucciones de los microcontroladores de la gama baja tienen una longitud de 12 bits. Las de la gama media tienen 14 bits y más las de la gama alta. Esta característica es muy ventajosa en la optimización de la memoria de instrucciones y facilita enormemente la construcción de ensambladores y compiladores.

4ª. Procesador RISC (Computador de Juego de Instrucciones Reducido) Los modelos de la gama baja disponen de un repertorio de 33 instrucciones, 35 los de la gama media y casi 60 los de la alta.

5ª. Todas las instrucciones son ortogonales Cualquier instrucción puede manejar cualquier elemento de la arquitectura como fuente o como destino. **6ª. Arquitectura basada en un banco de registros.** Esto significa que todos los objetos del sistema (puertos de E/S, temporizadores, posiciones de memoria, etc.) están implementados físicamente como registros.

7ª. Diversidad de modelos de microcontroladores con prestaciones y recursos diferentes. La gran variedad de modelos de microcontroladores PIC permite que el usuario pueda seleccionar el más conveniente para su proyecto.

8ª. Herramientas de soporte potentes y económicas. La empresa Microchip y otras que utilizan los PIC ponen a disposición de los usuarios numerosas herramientas para desarrollar hardware y software. Son muy abundantes los programadores, los simuladores software, los emuladores en tiempo real, ensambladores, Compiladores C, Intérpretes y Compiladores BASIC, etc.

3.3.3 RECURSOS AUXILIARES.

Entre los recursos más comunes se citan a los siguientes:

- Núcleos de CPU de 8/16 bits con Arquitectura Harvard modificada
- Memoria Flash y ROM disponible desde 256 bytes a 256 kilobytes
- Puertos de E/S (típicamente 0 a 5,5 voltios)
- Temporizadores de 8/16 bits

- Perro Guardián (“Watchdog”), destinado a provocar una reinicialización cuando el programa queda bloqueado.
- Tecnología Nanowatt para modos de control de energía
- Periféricos serie síncronos y asíncronos: USART, AUSART, EUSART
- Conversores analógicos/digital de 8-10-12 bits
- Comparadores de tensión
- Módulos de captura y comparación PWM
- Controladores LCD
- Periférico MSSP para comunicaciones I²C, SPI, y I²S
- Memoria EEPROM interna con duración de hasta un millón de ciclos de lectura/escritura
- Periféricos de control de motores
- Soporte de interfaz USB
- Soporte de controlador Ethernet
- Soporte de controlador CAN
- Soporte de controlador LIN
- Soporte de controlador Irda

3.3.4 LA FAMILIA DE LOS PIC

Una de las labores más importantes del ingeniero de diseño es la elección del modelo de microcontrolador que mejor satisfaga las necesidades del proyecto con el mínimo presupuesto. Microchip dispone de cuatro gamas de microcontroladores de 8 bits para adaptarse a las necesidades de la mayoría de los clientes potenciales. *

3.3.4.1 GAMA BAJA O BASICA: PIC12C5X CON INSTRUCCIONES DE 12 BITS:

Se trata de una serie de PIC de recursos limitados, pero con una de las mejores relaciones *coste/prestaciones*.

Sus versiones están encapsuladas con 18 y 28 patitas y pueden alimentarse a partir de una tensión de 2,5 V lo que les hace ideales en las aplicaciones que funcionan con pilas. Tienen un repertorio de 33 instrucciones cuyo formato consta de 12 bits. No admiten ningún tipo de interrupción y la pila sólo dispone de dos niveles.

3.3.4.2 GAMA MEDIA: PIC16CXXX CON INSTRUCCIONES DE 14 BITS.

Es la gama más variada y completa de los PIC. Abarca modelos con encapsulado desde 18 patitas hasta 68, cubriendo varias opciones que integran abundantes periféricos. Dentro de esta gama se halla el «fabuloso PIC 16F84». El repertorio de instrucciones es de 35 a 14 bits cada una y compatible con el de la gama baja. Sus distintos modelos contienen todos los recursos que se precisan en las aplicaciones de los microcontroladores de 8 bits. También dispone de interrupciones y una Pila de 8 niveles que permite el anidamiento de subrutinas. La gama media puede clasificarse en las siguientes subfamilias:

- a) Gama media estándar (PIC16C55X)
- b) Gama media con comparador analógico (PIC16C62X/64X/66X)
- c) Gama media con módulo de captura (CCP), modulación de anchura de impulsos (PWM) y puerta serie (PIC16C6X)
- d) Gama media con CAD de 8 bits (PIC16C7X)
- e) Gama media con CAD de precisión (PIC14000)
- f) Gama media con memoria Flash y EEPROM (PIC16F87X y PIC16X8X)
- g) Gama media con Driver LCD (PIC16C92X)

* Una descripción bien detallada se encuentra en el libro Microcontroladores PIC, Diseño Practico de Aplicaciones de Angulo Usategui y Angulo Martinez, Cap 2

3.3.4.3 GAMA ALTA: PIC17CXXX CON INSTRUCCIONES DE 16 BITS

Se alcanzan las 58 instrucciones de 16 bits en el repertorio y sus modelos disponen de un sistema de gestión de interrupciones vectorizadas muy potente. También incluyen variados controladores de periféricos, puertas de comunicación serie y paralelo con elementos externos y un multiplicador hardware de gran velocidad. Quizás la característica más destacable de los componentes de esta gama es su *arquitectura abierta*, que consiste en la posibilidad de ampliación del microcontrolador con elementos externos. Para este fin, las patitas sacan al exterior las líneas de los buses de datos, direcciones y control, a las que se conectan memorias o controladores de periféricos. Esta filosofía de construcción del sistema es la que se empleaba en los microprocesadores y no suele ser una práctica habitual cuando se emplean microcontroladores. Cabe mencionar que estas gamas sólo se utilizan en aplicaciones muy especiales con grandes requerimientos.

3.3.4.4 GAMA MEJORADA: PIC18C(F)XXX CON INSTRUCCIONES DE 16 BITS

En los inicios del tercer milenio de nuestra era Microchip presentó la gama mejorada de los microcontroladores PIC con la finalidad de soportar las aplicaciones avanzadas en las áreas de automoción, comunicaciones, ofimática y control industrial. Sus modelos destacaron por su alta velocidad (40 Mhz) y su gran rendimiento (10 MIPS a 10 Mhz). Entre las aportaciones más representativas de esta serie de modelos que crece cada año, destacan:

- a) Un espacio de direccionamiento para la memoria de programa que permite alcanzar los 2 MB, y 4 KB para la memoria de datos.
- b) Inclusión de la tecnología FLASH para la memoria de código.
- c) Potente juego de 77 instrucciones de 16 bits cada una. Permiten realizar una multiplicación 8 x 8 en un ciclo de instrucción, mover información entre las memorias y modificar el valor de un bit en un registro o en una línea de E/S.
- d) Orientación a la programación en lenguaje C con la incorporación de compiladores muy eficientes para este lenguaje.
- e) Nuevas herramientas para la emulación.

Inicialmente aparecieron cuatro modelos (PIC18C242/252/442/452) con 28 y 40 patitas que tenían hasta 16 KB de memoria de programa y hasta 1.536 bytes de RAM, ambas ampliables. Podían funcionar a 40 MHz, con 16 causas de interrupción, 4 temporizadores, 2 módulos CCP, Conversor A/D de 5 u 8 canales, y comunicación serie y paralelo. Luego aparecieron los PIC18FXXX que incorporaron la memoria FLASH para contener el código. Entre ellos destaca el modelo PIC18F720 con 128 KB de memoria FLASH y 3.840 bytes de RAM, estando encapsulado con 80 patitas.

3.3.4.5 LOS ENANOS DE 8 PATITAS

Se trata de un grupo de PIC (PIC12C (F) XXX) que ha acaparado la atención del mercado. Su principal característica es su reducido tamaño, al disponer un encapsulado de 8 patitas. Se alimentan con un voltaje de corriente continua comprendido entre 2,5 V y 5,5 V y consumen menos de 2 mA cuando trabajan a 5 V y 4 MHz. El formato de sus instrucciones puede ser de 12 o de 14 bits y su repertorio es de 33 o 35 instrucciones, según pertenezcan a la gama baja o media, respectivamente. Los modelos 12C5xx pertenecen a la gama baja, siendo el tamaño de las instrucciones

de 12 bits; mientras que los 12C6xx son de la gama media y sus instrucciones tienen 14 bits. Los modelos 12F6xx poseen memoria FLASH para el programa y EEPROM para los datos. Aunque los PIC enanos sólo tienen 8 patitas, pueden destinar hasta 6 como líneas de E/S para los periféricos porque disponen de un oscilador interno R-C.

3.3.4.6 PIC's WIRELESS

El microcontrolador rfPIC integra todas las prestaciones del PICmicro de Microchip con la capacidad de comunicación wireless UHF para aplicaciones RF de baja potencia. Estos dispositivos ofrecen un diseño muy comprimido para ajustarse a los cada vez más demandados requerimientos de miniaturización en aparatos electrónicos. Aun así, no parecen tener mucha salida en el mercado.

3.3.4.7 PIC's PARA PROCESADO DE SEÑAL (DSPICS)

Los dsPICs son el penúltimo lanzamiento de Microchip, comenzando a producirlos a gran escala a finales de 2004. Son los primeros PICs con bus de datos inherente de 16 bits. Incorporan todas las posibilidades de los anteriores PICs y añaden varias operaciones de **DSP** implementadas en hardware, como multiplicación con suma de acumulador (*multiply-accumulate*, o *MAC*), *barrel shifting*, *bit reversion* o multiplicación 16x16 bits.

3.3.4.8 PICS DE 32 BITS (PIC32)

Microchip Technology lanzo en noviembre de 2007 los nuevos microcontroladores de 32 bits con una velocidad de procesamiento de 1.5 DMIPS/MHz con capacidad HOST USB. Estos MCUs permiten un procesamiento de información increíble con un núcleo de procesador de tipo M4K.

3.3.5 LOS LENGUAJES USADOS PARA MICROCONTROLADORES

Un microcontrolador es un computador metido dentro de un circuito integrado.» Son computadores muy pequeños y baratos por lo que se utilizan para controlar muchos productos comunes en los que se halla incrustado dentro de los mismos, como sucede con el teléfono móvil, el teclado y el ratón del computador, etc. Además, al ser tan pequeños tienen una potencia limitada y sólo sirven para realizar una tarea. En el siglo xxi la realización de proyectos para aplicar los microcontroladores en el gobierno y automatización de multitud de productos y procesos se

presenta como una ingente labor que va a requerir la colaboración de multitud de profesionales y va a ofrecer una oportunidad inigualable para quienes estén preparados adecuadamente.

Los proyectos con microcontroladores exigen un trabajo con hardware consistente en adaptar las patitas del microcontrolador a los periféricos externos que hay que controlar. Además, también requieren la confección de un programa con las instrucciones precisas para que su ejecución origine el procesamiento de la información para obtener los resultados apetecidos. Para construir los programas de los microcontroladores se usan tres lenguajes:

- **Lenguaje ENSAMBLADOR**, de bajo nivel.
- **Lenguaje C**, de nivel medio
- **Lenguaje BASIC**, de alto nivel

El lenguaje Ensamblador se dice que es de bajo nivel porque sus instrucciones son exactamente las que el procesador sabe interpretar y ejecutar. En realidad, el computador digital sólo acepta instrucciones en código binario y el Ensamblador facilita su escritura al programador permitiendo expresarlas mediante «nemónicos», que son tres o cuatro letras significativas que referencian, en inglés, la operación que conlleva la instrucción. Por ejemplo, una instrucción que «mueve» un dato de un sitio (A) a otro (B), en Ensamblador se escribe MOV A,B.

El problema surge en el Ensamblador por la poca potencia de las instrucciones que es capaz de ejecutar el procesador. Normalmente las correspondientes a los microcontroladores de 8 bits que manejamos en este libro, consisten en sumar, restar, hacer operaciones lógicas AND, OR, XOR, también rotar un dato de 8 bits, moverlo de un sitio a otro y muy poquito más. Si deseamos hacer una multiplicación hay que confeccionar un programa que para conseguirlo repita las sumas las veces necesarias. Es un lenguaje de «bajo nivel». El programa Ensamblador lo único que hace es traducir los nemónicos con los que se escriben las instrucciones a código binario para que el procesador sea capaz de interpretarlas y ejecutarlas. Los lenguajes de «alto nivel» tienen instrucciones más potentes: saben multiplicar, sacar la raíz cuadrada y realizar funciones y operaciones mucho más complicadas que las que pueden hacer las instrucciones de la máquina. Pero como la máquina es la misma, la realización de esas instrucciones se tiene que hacer con programas de instrucciones elementales. Cada instrucción de alto nivel se convierte en un programita de instrucciones de bajo nivel. Por eso para que el procesador pueda ejecutar las instrucciones de un lenguaje de alto nivel precisa otro programa que las descomponga en las instrucciones de bajo nivel correspondientes. A estos programas se les llama compiladores. Los compiladores se encargan de traducir un programa confeccionado con

instrucciones de alto nivel a otro equivalente con instrucciones de bajo nivel. Hay una variante de estos programas que reciben el nombre de intérpretes que realizan dicha traducción, pero instrucción por instrucción, o sea, traducen una instrucción de alto nivel en las correspondientes de bajo nivel que ejecuta el procesador y a continuación pasan a la siguiente. Desarrollar programas en lenguaje Ensamblador exige un conocimiento profundo de la arquitectura interna del procesador, lo que requiere una buena base en Electrónica, así como formación en Informática. Las instrucciones de bajo nivel realizan operaciones directamente con los elementos existentes en el interior del procesador. El lenguaje C es de tipo profesional, muy completo y potente, pero su manejo exige una buena formación en Informática. También es muy conveniente conocer la arquitectura interna del procesador y en muchas ocasiones hay que combinarlo con el lenguaje Ensamblador. El lenguaje BASIC tiene potentes instrucciones que se escriben igual que se denominan en inglés y su manejo no requiere conocimientos profundos de arquitectura de procesadores, de Electrónica y tampoco de Informática.

3.3.6 PROGRAMACIÓN DEL PIC

Para transferir el código de un ordenador al PIC normalmente se usa un dispositivo llamado programador. La mayoría de PICs que Microchip distribuye hoy en día incorporan ICSP (*In Circuit Serial Programming*, programación serie incorporada) o LVP (*Low Voltage Programming*, programación a bajo voltaje), lo que permite programar el PIC directamente en el circuito destino. Para la ICSP se usan los pines RB6 y RB7 (En algunos modelos pueden usarse otros pines como el GP0 y GP1 o el RA0 y RA1) como reloj y datos y el MCLR para activar el modo programación aplicando un voltaje de 13 voltios. Existen muchos programadores de PICs, desde los más simples que dejan al software los detalles de comunicaciones, a los más complejos, que pueden verificar el dispositivo a diversas tensiones de alimentación e implementan en hardware casi todas las funcionalidades. Muchos de estos programadores complejos incluyen ellos mismos PICs preprogramados como interfaz para enviar las órdenes al PIC que se desea programar. El software de programación puede ser el ICprog, el WinPic 800, el Terusb1.0, muy comunes entre la gente que utiliza este tipo de microcontroladores.

3.4 PYTHON

Python es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. La elegante sintaxis de Python y su tipado dinámico, junto con su naturaleza

interpretada, hacen de éste un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas. El intérprete de Python y la extensa biblioteca estándar están a libre disposición en forma binaria y de código fuente para las principales plataformas desde el sitio web de Python, <http://www.python.org/>, y puede distribuirse libremente. El mismo sitio contiene también distribuciones y enlaces de muchos módulos libres de Python de terceros, programas y herramientas, y documentación adicional. El intérprete de Python puede extenderse fácilmente con nuevas funcionalidades y tipos de datos implementados en C o C++ (u otros lenguajes accesibles desde C). Python también puede usarse como un lenguaje de extensiones para aplicaciones personalizables. Este tutorial introduce de manera informal al lector a los conceptos y características básicas del lenguaje y el sistema de Python. Es bueno tener un intérprete de Python a mano para experimentar, sin embargo, todos los ejemplos están aislados, por lo tanto, el tutorial puede leerse estando desconectado. Para una descripción de los objetos y módulos estándar, mira la Referencia de la Biblioteca de Python. El Manual de Referencia de Python provee una definición más formal del lenguaje. Para escribir extensiones en C o C++, lee Extendiendo e Integrando el Intérprete de Python y la Referencia de la API Python/C. Hay también numerosos libros que tratan a Python en profundidad. Este tutorial no pretende ser exhaustivo ni tratar cada una de las características, o siquiera las características más usadas. En cambio, introduce la mayoría de las características más notables de Python, y te dará una buena idea del gusto y estilo del lenguaje. Luego de leerlo, serás capaz de leer y escribir módulos y programas en Python, y estarás listo para aprender más de los variados módulos de la biblioteca de Python descriptos en la Referencia de la Biblioteca de Python.

3.4.1 USANDO EL INTÉRPRETE DE PYTHON

Invocando al intérprete Por lo general, el intérprete de Python se instala en *file:/usr/local/bin/python* en las máquinas dónde está disponible; poner */usr/local/bin* en el camino de búsqueda de tu intérprete de comandos Unix hace posible iniciarlo ingresando la orden:

>>Python

...en la terminal. Ya que la elección del directorio dónde vivirá el intérprete es una opción del proceso de instalación, puede estar en otros lugares; consultá a tu Gurú Python local o administrador de sistemas. (Por ejemplo, */usr/local/python* es una alternativa popular). En máquinas con Windows, la instalación de Python por lo general se encuentra en *C:\Python26*, aunque se puede cambiar durante la instalación. Para añadir este directorio al camino, podes ingresar la siguiente orden en el prompt de DOS:


```
set path=%path%;C:\python26
```

Se puede salir del intérprete con estado de salida cero ingresando el carácter de fin de archivo (Control-D en Unix, Control-Z en Windows) en el prompt primario. Si esto no funciona, se puede salir del intérprete ingresando:

```
import sys;  
sys.exit().
```

Las características para editar líneas del intérprete no son muy sofisticadas. En Unix, quien instale el intérprete tendrá habilitado el soporte para la biblioteca GNU readlines, que añade una edición interactiva más elaborada e historia. Tal vez la forma más rápida de detectar si las características de edición están presentes es ingresar Control-P en el primer prompt de Python que aparezca. Si se escucha un beep, las características están presentes; ver Apéndice tut-interacting para una introducción a las teclas. Si no pasa nada, o si aparece, estas características no están disponibles; solo vas a poder usar backspace para borrar los caracteres de la línea actual. La forma de operar del intérprete es parecida a la línea de comandos de Unix: cuando se la llama con la entrada estándar conectada a una terminal lee y ejecuta comandos en forma interactiva; cuando es llamada con un nombre de archivo como argumento o con un archivo como entrada estándar, lee y ejecuta un script del archivo. Una segunda forma de iniciar el intérprete es `python -c comando [arg] ...`, que ejecuta las sentencias en comando, similar a la opción `-c` de la línea de comandos. Ya que las sentencias de Python suelen tener espacios en blanco u otros caracteres que son especiales en la línea de comandos, es normalmente recomendado citar comando entre comillas dobles. Algunos módulos de Python son también útiles como scripts. Pueden invocarse usando `python -m module [arg] ...`, que ejecuta el código de module como si se hubiese ingresado su nombre completo en la línea de comandos.

Notá que existe una diferencia entre `python file` y `python <file`. En el último caso, la entrada solicitada por el programa, como en llamadas a **`input()`** y **`raw_input()`**, son satisfechas desde file. Ya que este archivo ya fue leído hasta el final por el analizador antes de que el programa empiece su ejecución, se encontrará el fin de archivo enseguida. En el primer caso (lo que usualmente vas a querer) son satisfechas por cualquier archivo o dispositivo que esté conectado a la entrada estándar del intérprete de Python.

3.4.2 MODO INTERACTIVO

Se dice que estamos usando el intérprete en modo interactivo, cuando los comandos son leídos desde una terminal. En este modo espera el siguiente comando con el prompt primario, usualmente tres signos mayor-que (>>>); para las líneas de continuación espera con el prompt secundario, por defecto tres puntos (...). Antes de mostrar el prompt primario, el intérprete muestra un mensaje de bienvenida reportando su número de versión y una nota de copyright:

```
python
Python 2.6 (#1, Feb 28 2007, 00:02:06)
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

FIG 6 PYTHON INICIO

Las líneas de continuación son necesarias cuando queremos ingresar un constructor multilínea. Como en el ejemplo, mirá la sentencia if:

```
>>> el_mundo_es_plano = 1
>>> if el_mundo_es_plano:
...     print u";Tené cuidado de no caerte!"
...
;Tené cuidado de no caerte!
```

FIG 7 EJEMPLO DE PYTHON CONDICIONAL

3.4.3 EL INTÉRPRETE Y SU ENTORNO

3.4.3.1 PROGRAMAS EJECUTABLES DE PYTHON

En los sistemas Unix y tipo BSD, los programas Python pueden convertirse directamente en ejecutables, como programas del intérprete de comandos, poniendo la línea:

```
#!/usr/bin/env python
```

FIG 8 CODIGO EJECUTABLE PYTHON

al principio del script y dándole al archivo permisos de ejecución (asumiendo que los intérpretes están en la variable de entorno PATH del

usuario). #! deben ser los primeros dos caracteres del archivo. En algunas plataformas, la primera línea debe terminar al estilo Unix ('\n'), no como en Windows ('\r\n').

Notá que el carácter numeral '#' se usa en Python para comenzar un comentario.

Se le puede dar permisos de ejecución al script usando el comando chmod:

```
$ chmod +x myscript.py
```

FIG 9 PERMISOS A LOS ARCHIVO DE PYTHON

En sistemas Windows, no existe la noción de "modo ejecutable". El instalador de Python asocia automáticamente la extensión .py con python.exe para que al hacerle doble click a un archivo Python se corra el script. La extensión también puede ser .pyw, en este caso se omite la ventana con la consola que normalmente aparece.

3.4.3.2 USAR PYTHON COMO UNA CALCULADORA

Vamos a probar algunos comandos simples en Python. Iniciá un intérprete y esperá por el prompt primario, >>>. (No debería demorar tanto).

a) Números

El intérprete actúa como una simple calculadora; podés ingrsar una expresión y este escribirá los valores. La sintaxis es sencilla: los operadores +, -, * y / funcionan como en la mayoría de los lenguajes (por ejemplo, Pascal o C); los paréntesis pueden ser usados para agrupar. Por ejemplo:

```
>>> 2+2
4
>>> # Este es un comentario
... 2+2
4
>>> 2+2 # y un comentario en la misma línea que el código
4
>>> (50-5*6)/4
5
```

FIG 10 0 EJEMPLO DE CALCULADOR DE PYTHON

El signo igual (=) es usado para asignar un valor a una variable. Luego, ningún resultado es mostrado antes del próximo prompt:

```
>>> ancho = 20
>>> largo = 5*9
>>> ancho * largo
900
```

FIG 11 EJEMPLO DE VARIABLES TIPO ENTERO

b) Cadenas de caracteres

Además de números, Python puede manipular cadenas de texto, las cuales pueden ser expresadas de distintas formas. Pueden estar encerradas en comillas simples o dobles:

```
>>> 'huevos y pan'
'huevos y pan'
>>> 'doesn\'t'
"doesn't"
>>> "doesn't"
"doesn't"
>>> '"Si," le dijo.'
'"Si," le dijo.'
>>> "\"Si,\" le dijo."
'"Si," le dijo.'
>>> '"Isn\'t," she said.'
'"Isn\'t," she said.'
```

FIG 12 EJEMPLO DE VARIABLES TIPO CADENAS

c) Listas

Python tiene varios tipos de datos compuestos, usados para agrupar otros valores. El más versátil es la lista, la cual puede ser escrita como una lista de valores separados por coma (ítems) entre corchetes. No es necesario que los ítems de una lista tengan todos los mismos tipos.

```
>>> a = ['pan', 'huevos', 100, 1234]
>>> a
['pan', 'huevos', 100, 1234]
```

FIG 13 EJEMPLO DE VARIABLES TIPO LISTA

Como los índices de las cadenas de texto, los índices de las listas comienzan en 0, y las listas pueden ser rebanadas, concatenadas y todo lo demás:

```

>>> a[0]
'pan'
>>> a[3]
1234
>>> a[-2]
100
>>> a[1:-1]
['huevos', 100]
>>> a[:2] + ['carne', 2*2]
['pan', 'huevos', 'carne', 4]
>>> 3*a[:3] + ['Boo!']
['pan', 'huevos', 100, 'pan', 'huevos', 100, 'pan', 'huevos', 100, 'Boo!']

```

FIG 14 EJEMPLO DE INDICE DE LISTA

3.4.4 PYTHON Y MICROCONTROLADOR

Python tiene una amplia librería para poder comunicarnos con los microcontroladores y así poder obtener datos de sensores y actuadores para que la computadora pueda procesar la información suministrada.

Ejemplo de comunicación por el puerto serie:

```

import serial

disponibles = []

for i in range(10):
    try:
        s = serial.Serial(i)
        disponibles.append((i, s.portstr))
        s.close()
    except:
        pass

print "Puertos disponibles: "
for numpuerto,nombre in disponibles:
    print "%d -> %s" % (numpuerto, nombre)

```

3.4.5 INTERFACES GRAFICAS (GUI) EN PYTHON

Existen varias librerías que implementan interfaces gráficas de usuario (GUI) en python, las principales son:

- **Tkinter:** Basada en las librerías gráficas TCL/TK, interface "de-facto" (#1) preinstalada con python, es la generalmente recomendada para proyectos triviales y/o de aprendizaje.
- **WxPython:** Basada en WxWidgets (una librería multiplataforma C/C++), "bendecida" (#2) como la "más pitónica" por GvR (creador de Python), y sería la interface por defecto si no hubiese existido TK en primer lugar.
- **PyQT:** basado en la librería C++ QT (KDE)
- **PyGTK:** basado en la librería C GTK (GNOME)

3.4.5.1 Tkinter

- **Ventajas:**

- Preinstalado con python en casi todas las plataformas
- Relativamente simple y fácil de aprender (recomendado para "aprendices")
- Documentación completa

- **Desventajas:**

- Pocos elementos gráficos (sin listados, arboles, etc.)
- Limitado control del comportamiento de la interface (recomendado para proyectos "triviales")
- Lento (dibuja cada botón, etiqueta, menú, etc.) **
- Apariencia "extraña" (no se parece a las aplicaciones nativas)

3.4.5.2 WxPython

- **Ventajas:**

- Completo conjunto de elementos gráficos (listados, arboles, grillas, etc.)
- Flexible control del comportamiento de la interface.
- Rápido y de Apariencia nativa (diseñado para utilizar funciones nativas de cada plataforma).
- "Baterias Incluidas": más de 12 librerías y utilitarios complementarios.
- Independencia: no está orientado a ningún entorno, ni QT ni GTK, hay una capa mas que agrega un grado de libertad adicional.
- No se cierra en el mínimo denominador común; soporta las características comunes de Windows, y las emula en Linux/Mac OS cuando no se pueden hacer nativamente (y viceversa).

- Es mas "pitónico", por ej. espacio de nombres mas claro, sin referencias a C/C++, etc.
- Permite separar completamente el diseño de la interface en XML del código Python.
- Es fácil armar componentes personalizados, tanto que incorpora widgets que no están en wxWidgets mismo, ya que están escritos en Python.
- Documentación completa y ejemplos extensivos.

- **Desventajas:**

- No viene preinstalado con python, se debe instalar un paquete (wxPython en Windows y Mac OS, wxWidgets+wxPython en Linux, aunque en este último caso está generalmente está fácilmente disponible en los repositorios).
- Relativamente mas complejo de aprender
- Al tener un desarrollo bastante rápido y sostenido, se liberan versiones frecuentemente, lo que en la práctica le confiere cierto nivel de "volatilidad" y problemas de compatibilidad si se deben mantener varias versiones de wx para el mismo código.
- Es una capa más sobre el toolkit gráfico que se usa debajo (ej: GTK).
- Las características emuladas de otras plataformas no siempre se ven bien.
- Hacer interfaces multiplataformas que se vean bien requiere conocimiento del toolkit subyacente (win32, gtk).
- En proyectos medianos/grandes, puede ser inestable y difícil de debuggear: en windows es muy facil segfaultear si se pasan parámetros incorrectos.

3.4.5.3 PyQt

- **Ventajas:**

- Completo conjunto de elementos gráficos (listados, arboles, grillas, etc.)
- Flexible y potente control del comportamiento de la interface. Posee un mecanismo de conexión de señales y eventos simple. Se puede definir los eventos más sencillos en diseñador de GUI's (ej: al pulsar este botón,

borrar este campo de texto) y en el código python, definir las acciones más avanzadas.

- Rápido y de Apariencia nativa (las últimas versiones utilizan funciones nativas en windows)
- Se puede separar el diseño de la interface, pero usa un "compilador" pyuic para crear las clases python.
- Arquitectura opcional para Modelo/Vista para las tablas, listas y árboles.

- **Desventajas:**

- No viene preinstalado con python, se debe instalar por separado
- Relativamente mas complejo de aprender
- No del todo "pitónico". En ocasiones emerge la implementación en C++ subyacente, teniendo que hacer casts entre tipos de datos, etc. El prefijo Qt/Q (QtGUI, QWidget, QApplication) hace el código menos "pitónico".
- No hay mucha documentación específica a python, ya que es lenguaje en si no es demasiado considerado

3.4.5.4 PyGTK

- **Ventajas:**

- Completo conjunto de elementos gráficos (listados, arboles, grillas, etc.)
- Flexible y potente control del comportamiento de la interface
- Enlace con PyOrbit para programar aplicaciones en GNOME
- Es estable, y los mensajes de error son correctos.

- **Desventajas:**

- No viene preinstalado con python, se debe instalar por separado
- Relativamente mas complejo de aprender
- Relativamente lento en Windows (dibuja cada botón, etiqueta, menú, etc.) lo que le da una Apariencia "extraña" (aunque es parecido a windows)
- En windows, es la librería que tiene mas dependencias y se instalan por separado.
- Aparentemente tiene la documentación mas precaria de todos

3.5 PROTEUS DESING SUITE 8

Proteus es un entorno integrado diseñado para la realización completa de proyectos de construcción de equipos electrónicos en todas sus etapas: diseño, simulación, depuración y construcción.



FIG 15INTRODUCCION A PROTEUS

Sin la utilización de la suite Proteus, el proceso para construir un equipo electrónico basado en un microprocesador se compone de las siguientes etapas:



FIG 16 ETAPAS DE PROTEUS

El depurado de errores puede convertirse en una labor ardua en tiempo y recursos, lo que conlleva un alto coste económico. Sin embargo con la herramienta Proteus el proceso queda definido con las siguientes etapas:



FIG 17FASE DE PROTEUS

Las ventajas saltan a la vista. Con Proteus las fases de prueba no suponen la necesidad de volver a consruir nuevos prototipos, con el ahorro de costos y tiempo que ello supone.

3.6 SDCC (SMALL DEVICE C COMPILER) EN WINDOWS

SDCC es un compilador Open Source distribuido bajo licencia GPL que compila código en lenguaje C para los siguientes Microcontroladores: Intel 8051, Maxim 80DS390, Zilog Z80, Motorola 68HC08 y los PIC16 y PIC18 de Microchips. Es un compilador multiplataforma, por lo que lo podemos instalar en Windows, Linux y MAC.

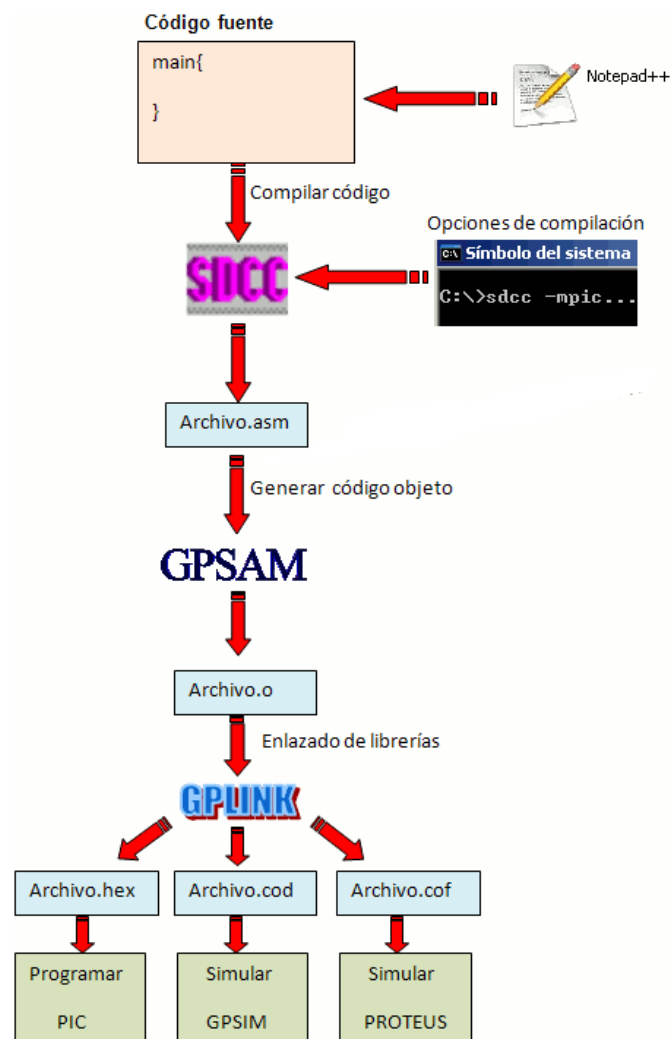


FIG 18 DIAGRAMA DE FLUJO DE COMPILADO PARA MICROCONTROLADOR PIC

3.6.1 BOOTLOADER EN MICROCONTROLADOR

Un BootLoader para Microcontroladores se puede definir como un programa residente en el Microcontrolador (en este caso un PIC) que facilita la carga de los programas del usuario.

El BootLoader hay que programarlo en el PIC de forma convencional a través de un programador externo como el ICD-U64, ICD3, PICkit 3, etc. Una vez programado el BootLoader la carga de los programas de usuario se hacen directamente a través de un canal de comunicación, este canal puede ser el puerto serie, paralelo o USB de nuestro ordenador, en este caso el archivo. HEX se transfiere al PIC a través de una pequeña aplicación de escritorio que hace de interfaz entre el PC y el firmware del Microcontrolador, pero también se suelen utilizar otros buses de comunicación como el bus CAN, SPI, I2C, etc.

3.6.1.1 VENTAJAS DE UTILIZAR UN BOOTLOADER

Los BootLoaders llevan ya tiempo utilizándose en el mundo de los Microcontroladores y su uso ha sido fundamental en el éxito de muchos proyectos populares como: Arduino, Netduino, etc. Estos proyectos basan su éxito en facilitar al usuario una plataforma económica con la que empezar a programar los Microcontroladores y para ello es fundamental el abaratar costes, como el no tener que utilizar un programador externo para cargar las aplicaciones de usuario. Estas placas de desarrollo vienen ya con el Bootloader cargado en la memoria flash del PIC, por lo que no se necesita de ningún Hardware adicional para empezar a programar el Microcontrolador insertado en la placa de desarrollo.

Pero esta no es la única ventaja de utilizar un BootLoader, otra ventaja la tenemos en que podemos actualizar el programa de usuario cargado en el Microcontrolador de manera fácil y sin necesidad de sacar el Micro fuera de la placa donde esté montado. Por ejemplo, un módulo de control instalado en un automóvil que utilice Microcontroladores no se desmonta para actualizar su software, su actualización se realiza a través de puntos de control (SetPoints) accesibles por el técnico de mantenimiento y que se comunican con la unidad de control a través de un canal de comunicación como el bus CAN.

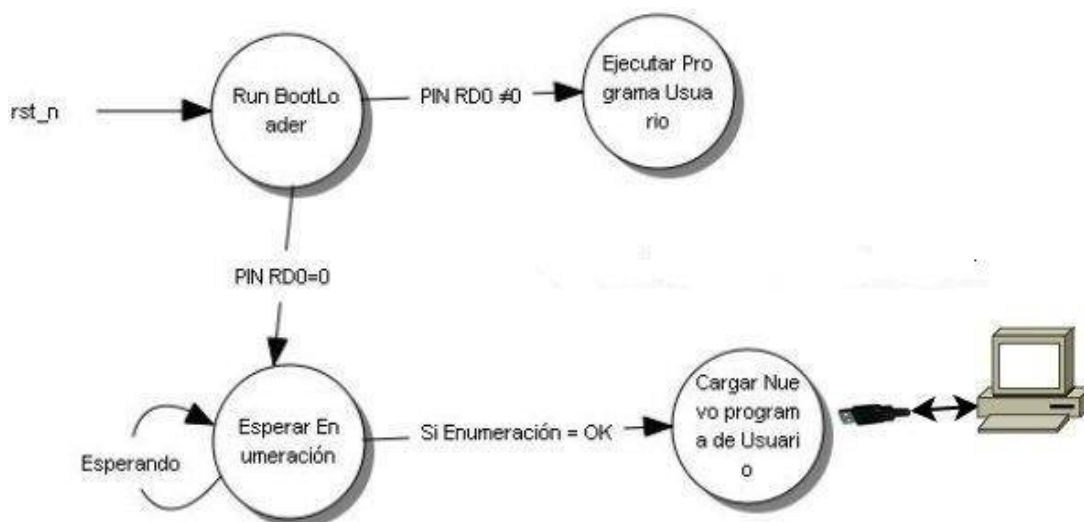


FIG 19 COMUNICACIÓN CON EL BOOTLOADER

3.7 SISTEMAS DE ADQUISICION DE DATOS

Los sistemas de adquisición de datos son sistemas electrónicos completos que contienen diferentes dispositivos electrónicos como convertidores analógicos – digital (CAD), registros de corrimiento, memorias para datos, configuraciones Peripheral Component Interconnect (PCI) para conexión a computadoras o por puerto serie o por puerto Universal Serial Bus (USB). Su principal uso es en la adquisición de señales de interés como voltaje, corriente, temperatura, humedad, peso, etc., para su posterior procesamiento dentro de una computadora o algún sistema de graficado. Podemos considerar como el primer sistema de adquisición de datos al multímetro digital. Los primeros modelos de voltímetros con lecturas digitales presentaban tubos de vacío (bulbos) para los elementos de lectura y tubos para la conversión analógica – digital y la lógica electrónica. Estos instrumentos eran muy pesados, muy grandes y muy caros. Tenían carcasas de metal y eran algo limitados en su rango de funciones. Estos dispositivos requerían una línea de voltaje de 120 volts, lo cual limitaba su portabilidad. Su tiempo de respuesta entre lecturas era lento. Pero si se tenía suficiente dinero para gastar en su construcción, era posible alcanzar una exactitud razonable. En 1960, los modelos transistorizados llegaron a ser muy populares. Con la introducción del Light-Emitting Diode (LED) y las más recientes técnicas de muestreo de ese tiempo, el uso del multímetro digital llegó a ser muy extendido en la industria. El tamaño, costo y requerimientos de alimentación se redujeron de manera que los modelos operados con baterías podían ser obtenidos a precios razonables, permitiendo que los multímetros digitales fueran accesibles para el técnico independiente. [9] Los circuitos integrados hicieron posible desarrollar multímetros con impedancias de entrada altas y con mayor fiabilidad, a un precio más bajo con un rango más amplio de

funciones. La introducción del Liquid Cristal Display (LCD) redujo los requerimientos de alimentación.

Los multímetros digitales de hoy son controlados por chips Large Scale Integrated (LSI) comunes y pueden consistir de un chip LSI especializado para procesar la entrada y un microprocesador para controlar las funciones y la salida junto con algunos componentes externos. La impedancia de entrada puede ser mayor a los 10 000 MΩ para rangos de corriente directa arriba de 20 V. Los instrumentos de control fueron naciendo a medida que las exigencias del proceso lo impusieron. Las necesidades de la industria fueron (y son actualmente) el motor que puso en marcha la inventiva de los fabricantes o de los propios usuarios para idear y llevar a cabo la fabricación de los instrumentos convenientes para los procesos industriales. El desarrollo se inició con los manómetros, termómetros y válvulas manuales localmente montadas. En esta fase eran necesarios muchos operadores para observar los instrumentos y maniobrar las válvulas los procesos industriales eran proyectados empíricamente basándose en la intuición y en la experiencia acumulada y no estaban centralizados para conseguir una mayor eficacia en las funciones del operador.

La siguiente etapa fue la centralización de las funciones de medida y control más importantes, pertenecientes a una operación del proceso en un panel localmente montado. De este modo podía observarse y controlarse el funcionamiento de cada elemento en particular de una manera más coordinada y eficaz. Para hacer esto posible, se desarrollaron instrumentos galvanométricos operados por termopar, termómetros con largos capilares y caudalímetros con largos tubos de conducción de presión diferencial. Sin embargo, los procesos se hicieron más complejos y críticos y llegó hacerse necesario que los operadores observaran el funcionamiento de varias unidades de la instalación simultáneamente. El desarrollo de los transmisores neumáticos permitió la centralización de las funciones de medida y de regulación de toda una unidad del proceso en una sola de control, utilizándose como receptores los instrumentos registradores controladores neumáticos de caja grande que aparecieron hacia el año 1940. Posteriormente, estos instrumentos se perfeccionaron con un diseño modular, conservando la unidad automático-manual de cuatro posiciones en un subpanel aparte.

A medida que paso el tiempo estas salas de control indebidamente grandes, debido al crecimiento de los procesos y al tamaño de los instrumentos convencionales se desarrolló la instrumentación neumática miniatura que apareció en el mercado hacia el año de 1947, dotada ya con conmutación automático-manual e incorporada, pero con el mismo tipo de transferencia.

A principio de los años 50 aparecen los primeros instrumentos electrónicos a válvulas. Más tarde se perfeccionó la unidad automática-manual, consiguiéndose el cambio en un solo paso, sin que se produzcan saltos en la señal de salida a la válvula y aparecen paralelamente los instrumentos electrónicos miniatura alrededor del año 1960. El tamaño de estos instrumentos neumáticos y electrónicos es ya reducido, pero todavía experimentara una normalización posterior.

Los complejos de múltiples procesos empezaron a utilizar salas de control separadas y la coordinación y la comunicación entre los operadores en estas salas de control comenzaron a plantear algunos problemas. Además, se introdujeron equipos centrales de tratamientos de datos que requerían la disponibilidad de diversas señales de medida en un punto central.

Una vez desarrollados los instrumentos miniatura neumáticos y electrónicos los procesos se fueron haciendo poco a poco más complejos y su optimización llego a ser una necesidad. En esta es donde empezaron a utilizarse los primeros computadores. El primer computador electrónico apareció hacia el año 1946, pero los verdaderos computadores de proceso se desarrollaron realmente en los años 1960- 1965 y se aplicaron principalmente en centrales térmicas, industrias metalúrgicas, químicas y petroquímicas.

En 1983 aparece el transmisor digital inteligente con señal de salida analógica de 4 a 20 mA DC y se inicia el desarrollo de las comunicaciones field bus (estándar abierto para entradas, salidas y dispositivos de control de procesos en red que cuando se configuran pueden correr independientemente de una PC). Se elimina las incomodidades y caras calibraciones necesarias en los instrumentos convencionales y se facilita el cambio del campo de medida y auto diagnóstico. En 1986 aparece el primer transmisor enteramente digital con lo que aumenta todavía más las prestaciones, con la única limitación importante en la normalización de las comunicaciones donde todavía no es posible el intercambio de instrumentos de diferentes marcas.

La aplicación de los instrumentos neumáticos y electrónicos analógicos quedara limitada a una pequeña planta, ya que, frente a la instrumentación digital, tiene una peor relación costo/prestaciones, no permite el almacenamiento de volúmenes masivos de información y no disponen de facilidad de comunicación entre instrumentos que posee la digital.

Un SAD no es más que un equipo electrónico cuya función es el control o simplemente el registro de una o varias variables de un proceso

cualquiera, de forma general puede estar compuesto por los siguientes elementos.

1. Sensores.
2. Amplificadores operacionales.
3. Amplificadores de instrumentación.
4. Aisladores.
5. Multiplexores analógicos.
6. Multiplexores digitales.
7. Circuitos Sample and Hold.
8. Conversores Analógico-digital.
9. Conversores Digital-analógico.
10. Microprocesadores.
11. Contadores.
12. Filtros.
13. Comparadores.
14. Fuentes de potencia.

En la siguiente figura se muestra el diagrama general de un SAD.

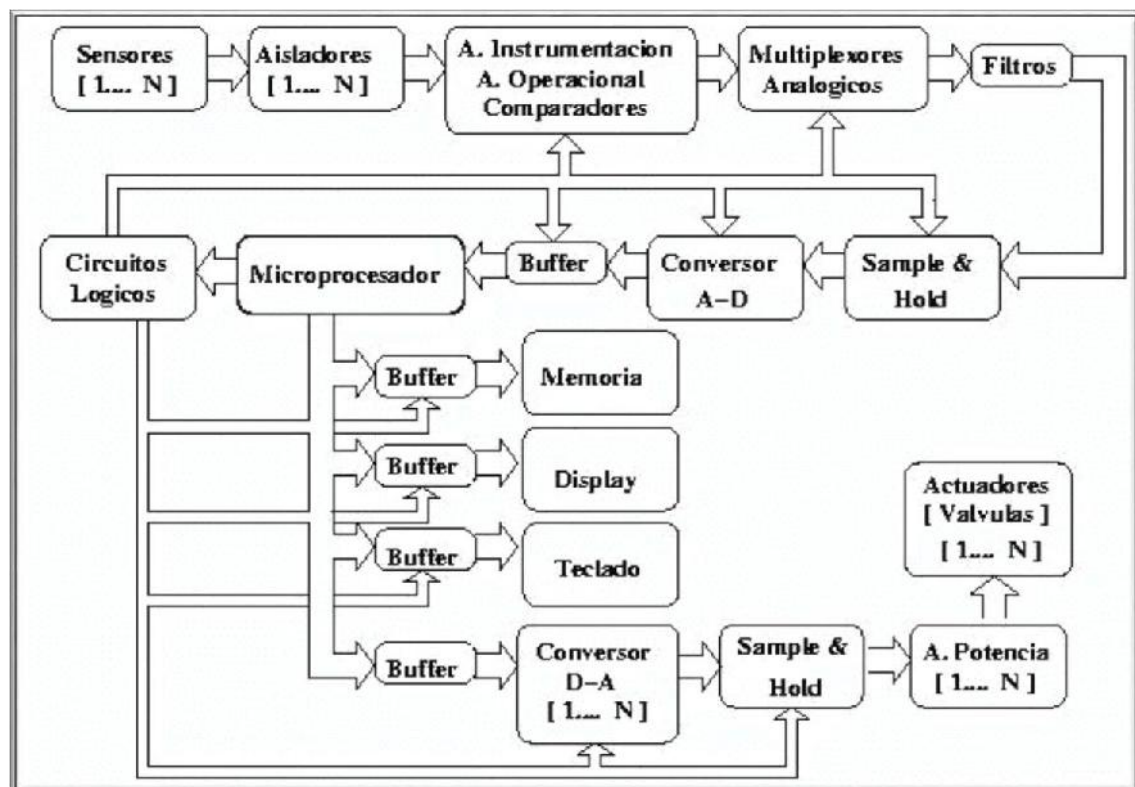


FIG 20 DIAGRAMA GENERAL DE UN SISTEMA DE ADQUISICIÓN DE DATOS

CAPITULO IV

PROGRAMACIÓN DEL ENTORNO GRAFICO INTEGRADO Y DISEÑO DEL CIRCUITO DE ADQUISICION DE DATOS

Para desarrollar un entorno grafico integrado, usaremos el lenguaje de programación Python versión 2.7, que nos servirá para programar la interface gráfica interactiva. Y así poder programar eficientemente nuestra tarjeta de adquisición de datos.

4 INTRODUCCIÓN

El objetivo de esta sección es ofrecer información sobre la programación de la interface interactiva y el diseño del circuito de adquisición de datos, así como su comunicación bidireccional.

4.1 DISEÑO DE UNA INTERFACE GRAFICA

Se diseñó el entorno grafico de acuerdo a las sugerencias y comodidades de los usuarios para facilitar la programación del microcontrolador que contiene la tarjeta de adquisición de datos.

Se optó por el siguiente diseño gráfico:

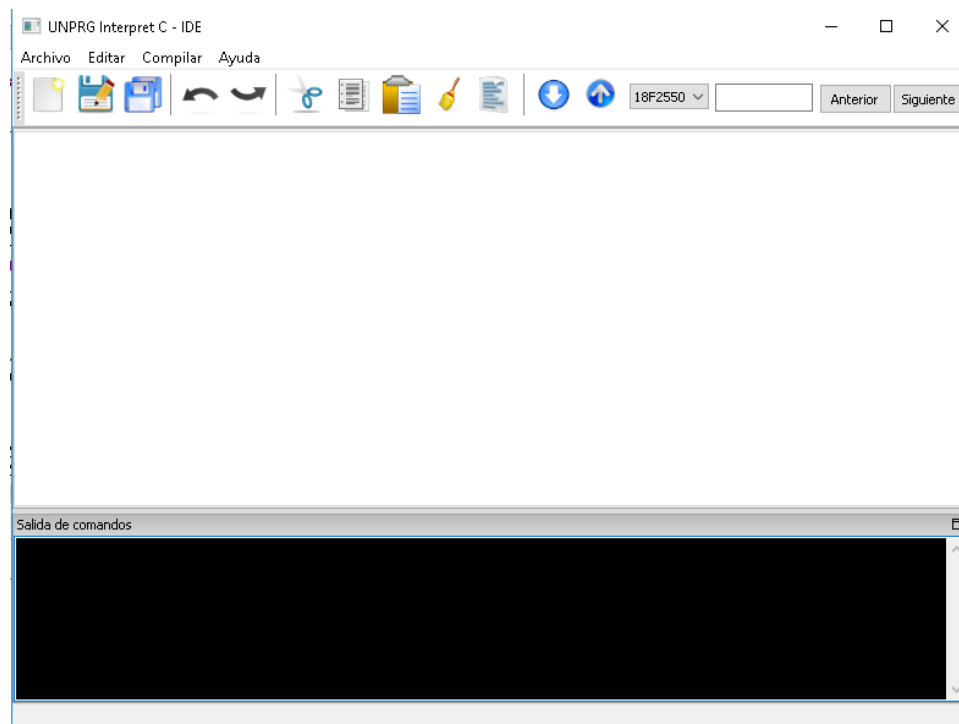


FIG 21 INTERFACE GRÁFICA DEL USUARIO

4.2 DIAGRAMA DE FLUJO DE LOS BOTONES DE LA INTERFACE GRAFICA

4.2.1 BOTON NUEVO



Este botón sirve para crear un nuevo proyecto y su diagrama de flujo es el siguiente:

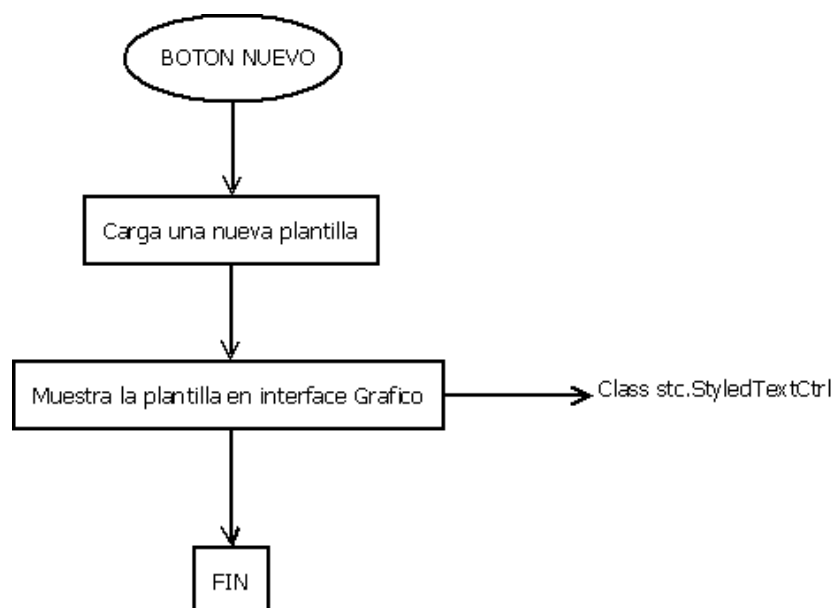


FIG 22 DIAGRAMA DE FLUJO DEL BOTÓN NUEVO

4.2.2 BOTON GUARDAR



Este botón sirve para salvar nuestro proyecto por seguridad, y su diagrama de flujo es el siguiente.

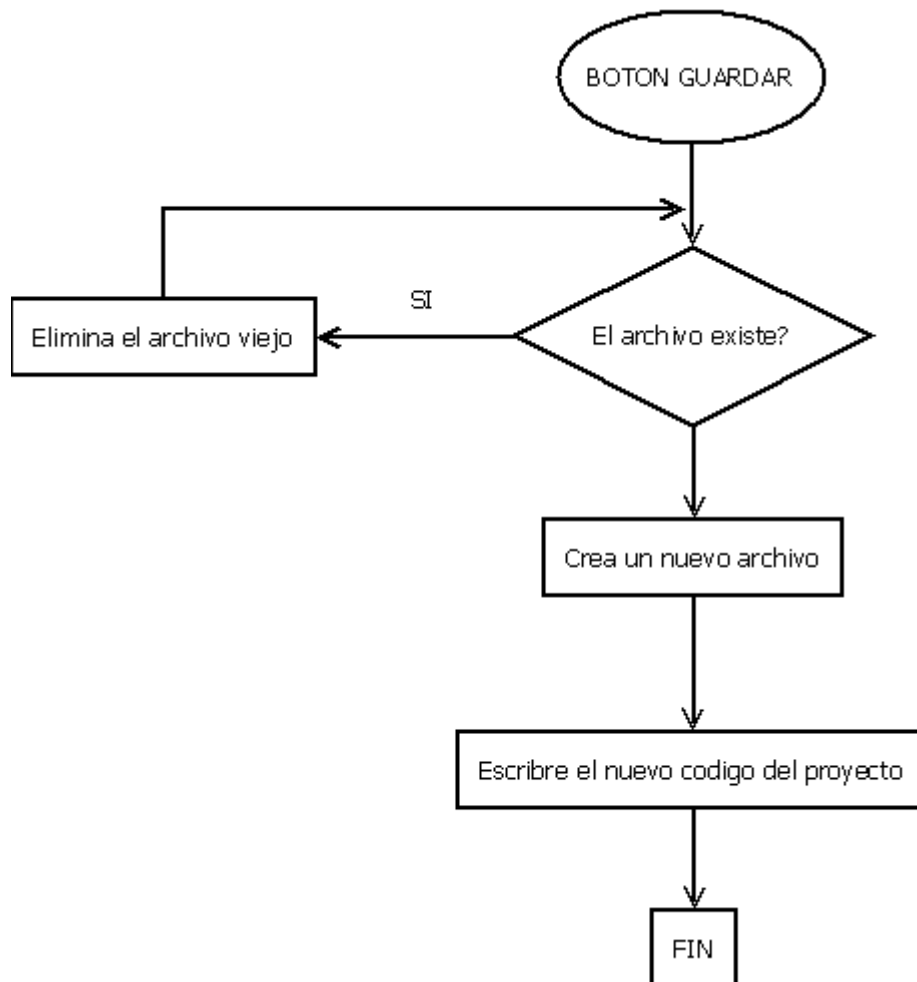


FIG 23 DIAGRAMA DE FLUJO DEL BOTÓN GUARDAR

4.2.3 BOTÓN DE COMPILAR CÓDIGO FUENTE



Este botón sirve para compilar el proyecto con ayuda del SDCC, quien nos genera un archivo con formato. hex

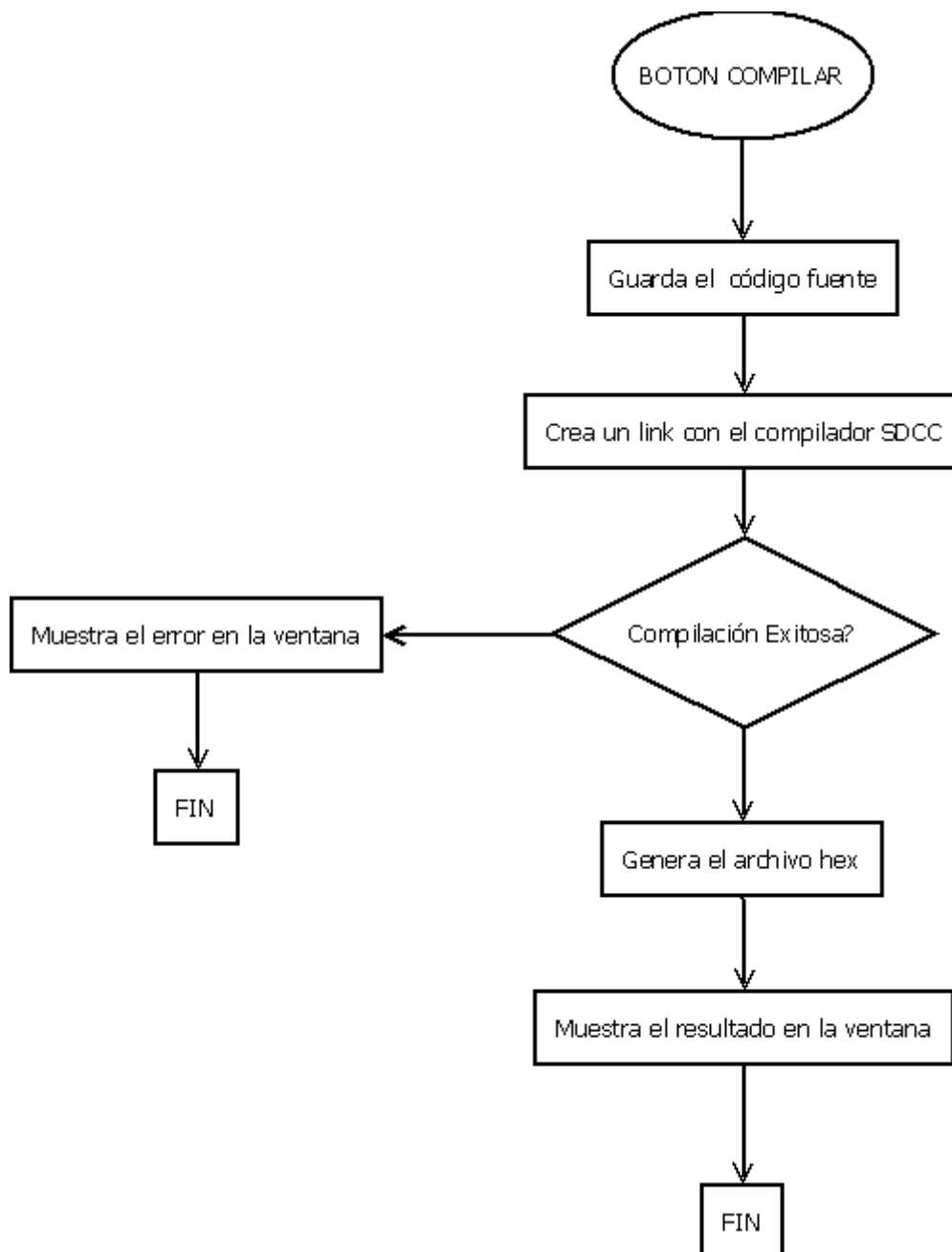


FIG 24 DIAGRAMA DE FLUJO DEL BOTÓN COMPILAR

4.2.4 BOTON DESCARGAR HADWARE



Este botón sirve para descargar el programa generado por el compilador hacia la tarjeta de adquisición de datos, el diagrama de flujo es el siguiente

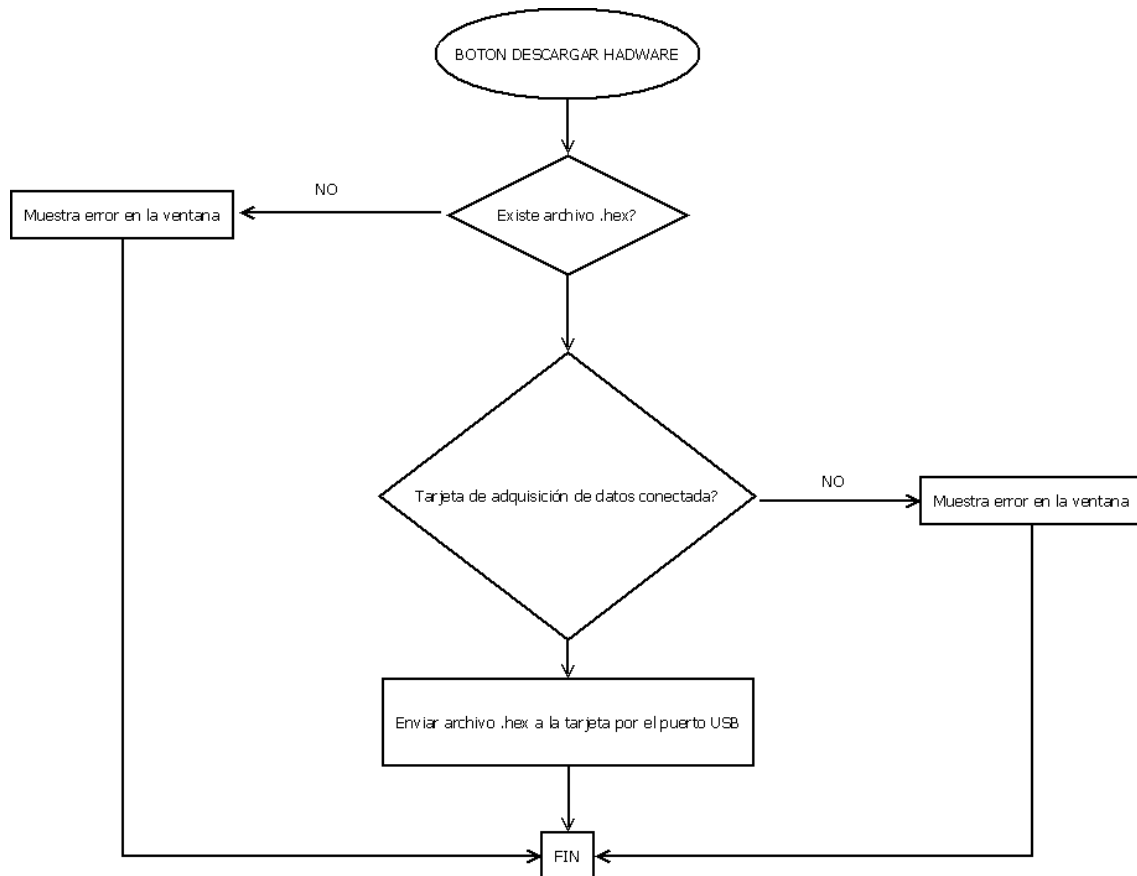
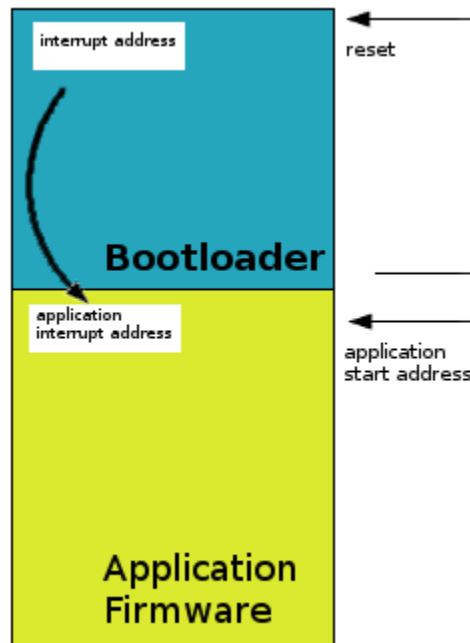


FIG 25 DIAGRAMA DE FLUJO DEL BOTÓN DESCARGAR HARDWARE

4.3 BOOTLOADER DEL MICROCONTROLADOR

Con esta aplicación dejamos de presidir un grabador de microcontrolador, el diagrama de flujo es el siguiente:



4.4 DISEÑO DEL CIRCUITO DE ADQUISICION DE DATOS

Para la presente tesis, se utilizó el PIC18f2550 donde a partir de su estructura se diseñó el circuito de adquisición de datos.

Características del microcontrolador son las siguientes:

- ✓ Voltaje de alimentación 2.7v a 5v
- ✓ Consumo de corriente 25 mA.
- ✓ Bus de 8 bits
- ✓ 48 Mhz de velocidad.
- ✓ 32 kbytes de memoria EEPROM
- ✓ 4 temporizadores
- ✓ 13 Canales analógicos multiplexado.
- ✓ 23 salidas/entradas digitales
- ✓ 2 canales PWM multiplexado.
- ✓ 1 Comunicación UART.
- ✓ 1 Comunicación USB v2.
- ✓ Comunicación I2C.

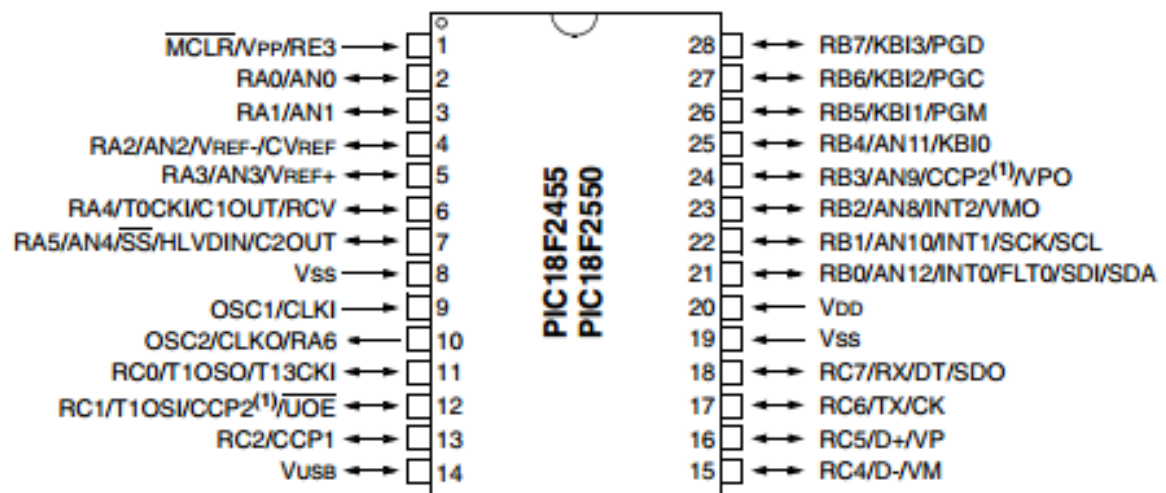


FIG 26 DIAGRAMA DEL MICROCONTROLADOR PIC18F2550

4.4.1 DISEÑO DEL CIRCUITO DE LA TARJETA DE ADQUISICIÓN DE DATOS

Para el diseño del circuito de la tarjeta de adquisición de datos se tomó algunas referencias de módulos arduino, con el fin de hacerlo compatible con todas las versiones, sea añadido una fuente de 3.3v y 5.0v con una intensidad máxima de 500 mA suficiente para embeber diferentes módulos existentes, se utilizó el software PROTEUS para realizar el esquemático.

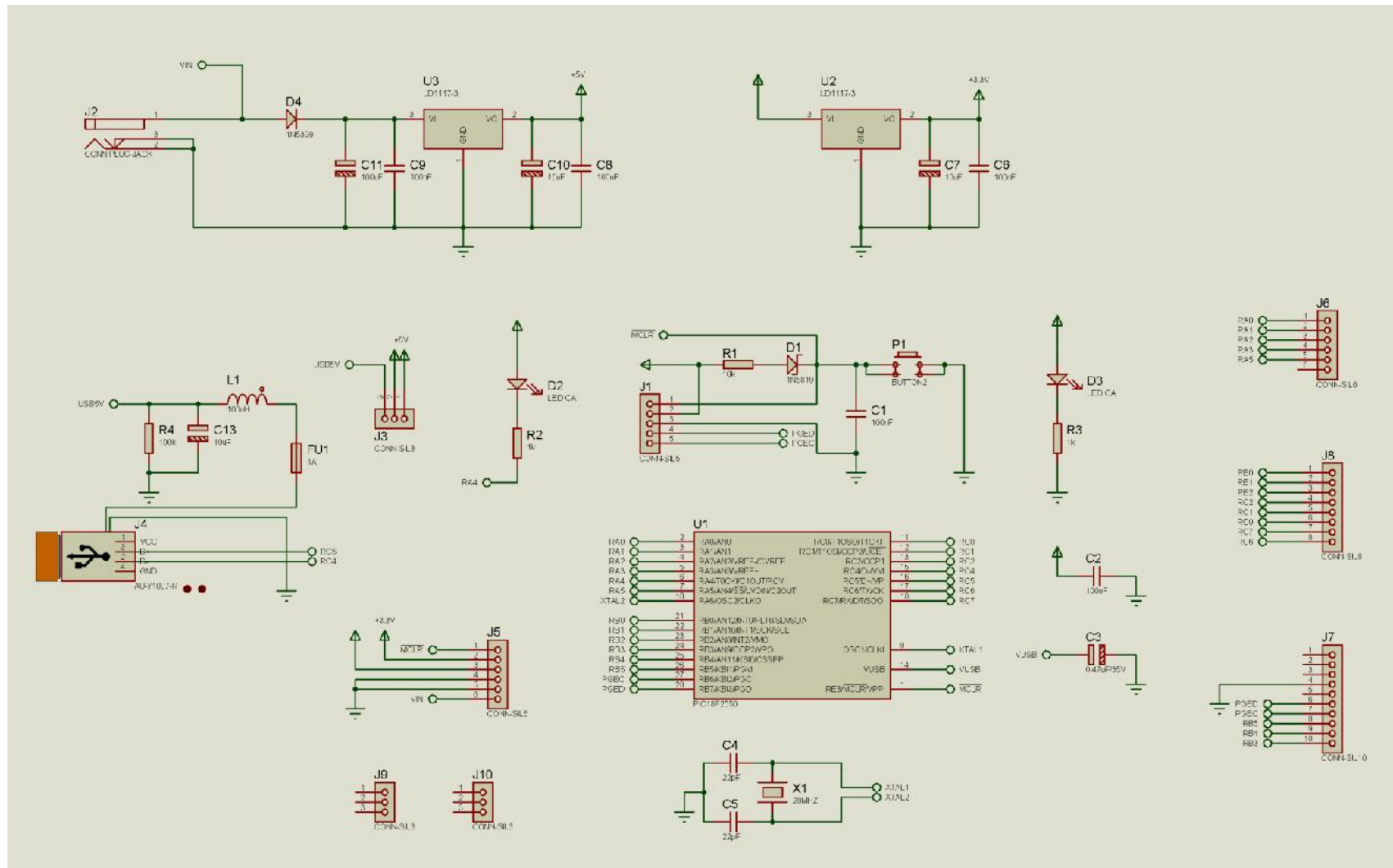


FIG 27 CIRCUITO DE LA TARJETA DE ADQUISICIÓN DE DATOS

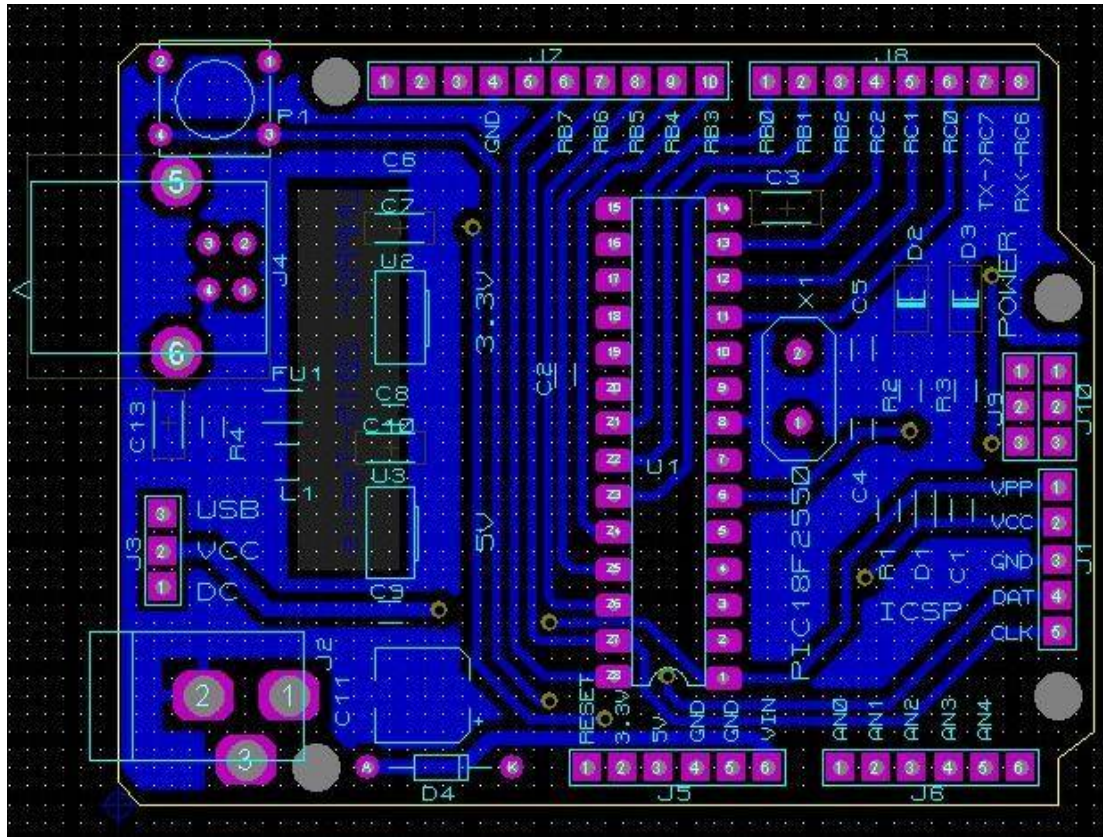


FIG 28 ESQUEMA DEL CIRCUITO IMPRESO ARRIBA

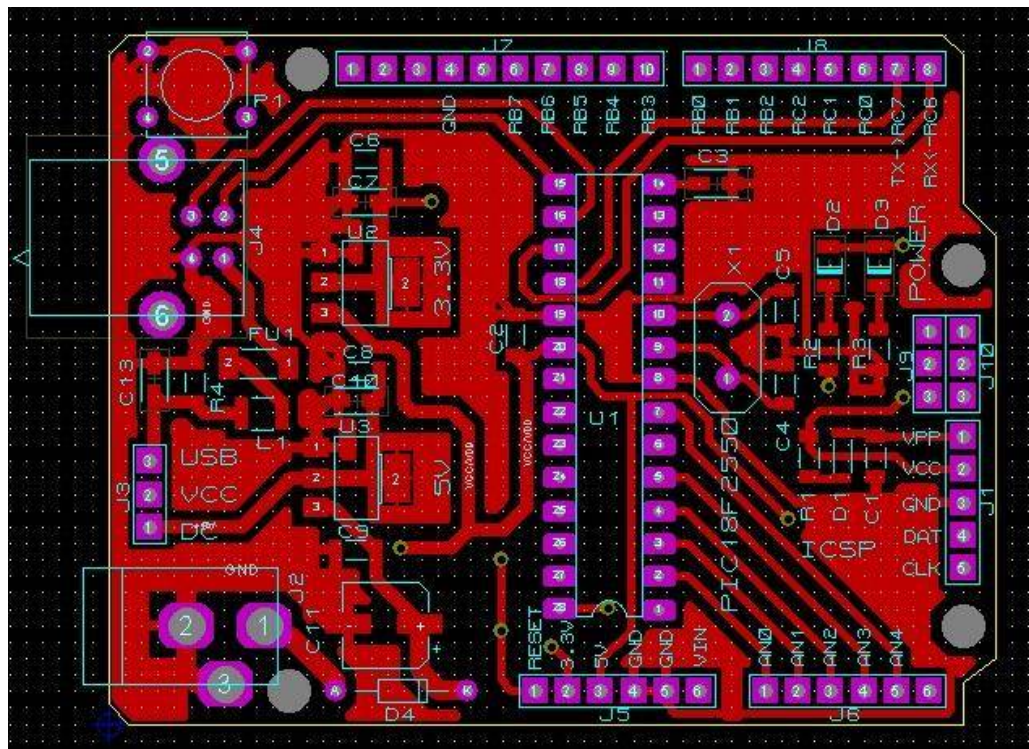


FIG 29 ESQUEMA DEL CIRCUITO IMPRESO ABAJO

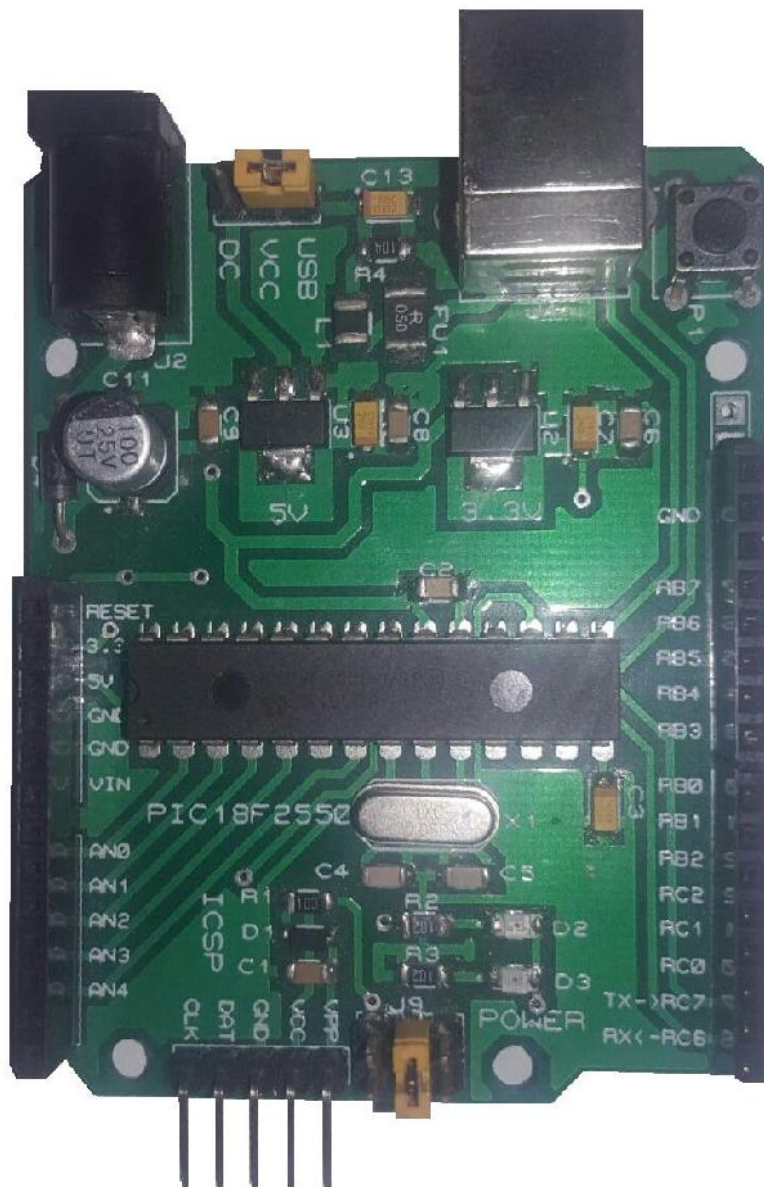


FIG 30 CIRCUITO ENSAMBLADO

CAPITULO V

Ejecución de pruebas e Interpretación de los Datos

5 EJECUCIÓN DE PRUEBAS

Para la realización de nuestras pruebas se ha adquirido sensores e indicadores de entrenamiento, como los siguientes:



FIG 31 INDICADORES



FIG 32 MÓDULOS VARIOS

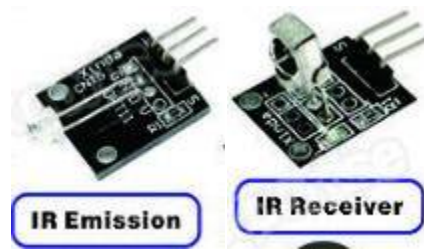


FIG 33 IR SENSOR

5.1 CREAR UNA APLICACIÓN

Para crear una aplicación debes ir al menú del programa *archivo>nuevo* y mostrara la ventana siguiente:

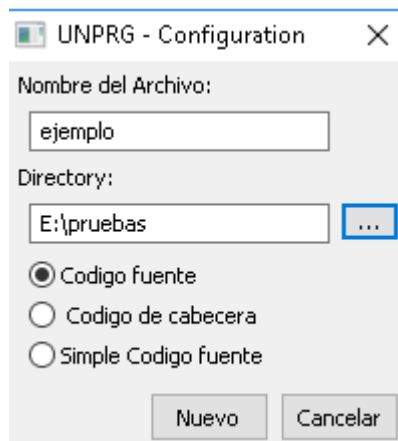


FIG 34 VENTANA NUEVO ARCHIVO

Al presionar el botón Nuevo, de forma inmediata aparece la ventana de edición, como muestra a continuación:



FIG 35 ESTRUCTURA BÁSICA DE UNA APLICACIÓN

A continuación, desarrollaremos ejemplos sencillos utilizando nuestra tarjeta de adquisición de datos.

a) Ejemplo 1 “Hola Mundo”. - Nuestra primera aplicación que desarrollaremos tiene la siguiente lógica:

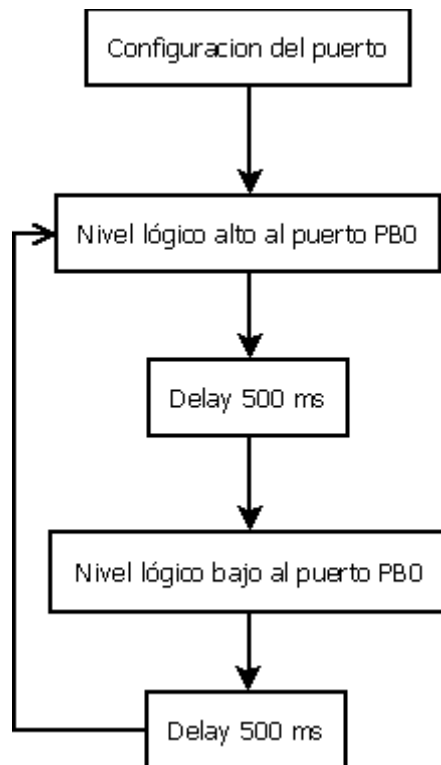


FIG 36 DIAGRAMA DE FLUJO PARA EL EJEMPLO 01

The screenshot shows the UNPRG Interpret C - IDE window. The menu bar includes Archivo, Editar, Compilar, and Ayuda. The toolbar contains icons for file operations and compilation. The file name is 'ejemplo.tki'. The code editor displays the following C code:

```
void setup () {  
    pinMode(0,OUTPUT);  
}  
  
void loop () {  
    digitalWrite(0,HIGH);  
    delay(500);  
    digitalWrite(0,LOW);  
    delay(500);  
}
```

The command prompt at the bottom shows the compilation process:

```
\lib\libpuf.lib e:\todos\tunky\tools\bin\lib\pic16\libio18f2550.lib e:\todos\tunky\tools\bin\lib\pic16\libc18f.lib e:\todos\tunky\tools\bin\lib\pic16\pic18f2550.lib libsdcc.lib -w -se:\todos\tunky\lkr\18f2550.lkr e:\todos\tunky\obj\application_iface.o e:\todos\tunky\obj\usb_descriptors.o e:\todos\tunky\obj\crt0unprg.o e:\todos\tunky\source\main.o  
  
Compilacion exitosa!!  
  
Tama~no: 3028 / 24575 bytes (12% Usado)
```

FIG 37 CÓDIGO FUENTE DEL EJEMPLO 01.

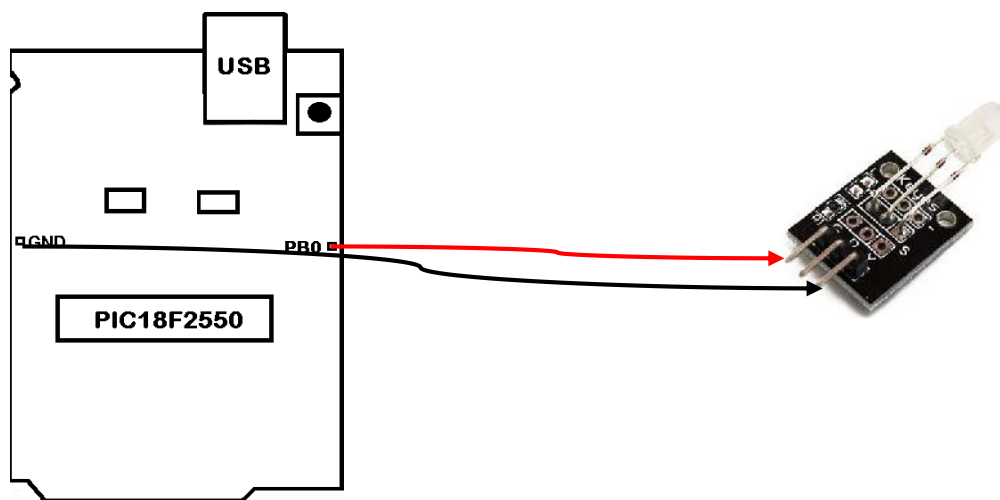


FIG 38 CIRCUITO DE CONEXIÓN PARA EL EJEMPLO 01.



FIG 39 EJEMPLO 1 CIRCUITO

B) Ejemplo 2 “Leer estado lógico”. - Para este ejemplo utilizaremos un pulsador y un led.

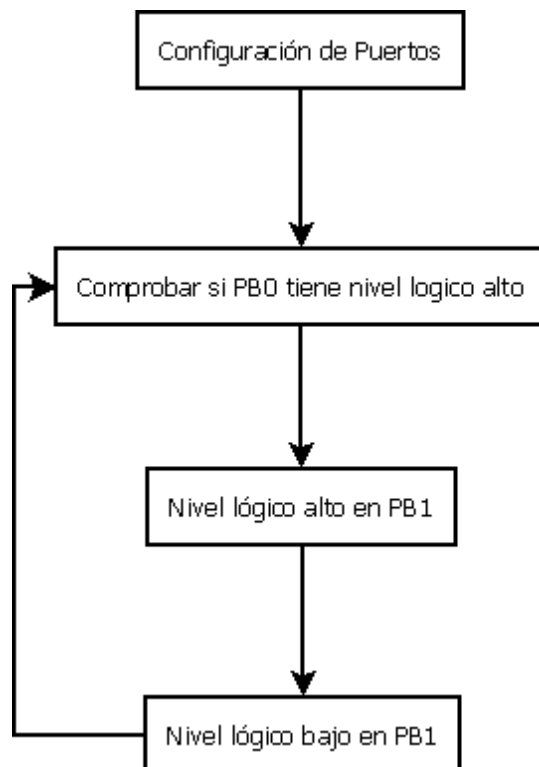


FIG 40 DIAGRAMA DE FLUJO PARA EL EJEMPLO 02

```

*ejemplo.tki
1 void setup()
2 {
3   pinMode(PB1,OUTPUT);
4   pinMode(PB0,INPUT);
5 }
6
7 void loop()
8 {
9   if(digitalRead(PB0)) { |
10    digitalWrite(PB1,HIGH);
11   } else {
12    digitalWrite(PB1,LOW);
13   }
14 }

```

FIG 41 CÓDIGO FUENTE DEL EJEMPLO 02

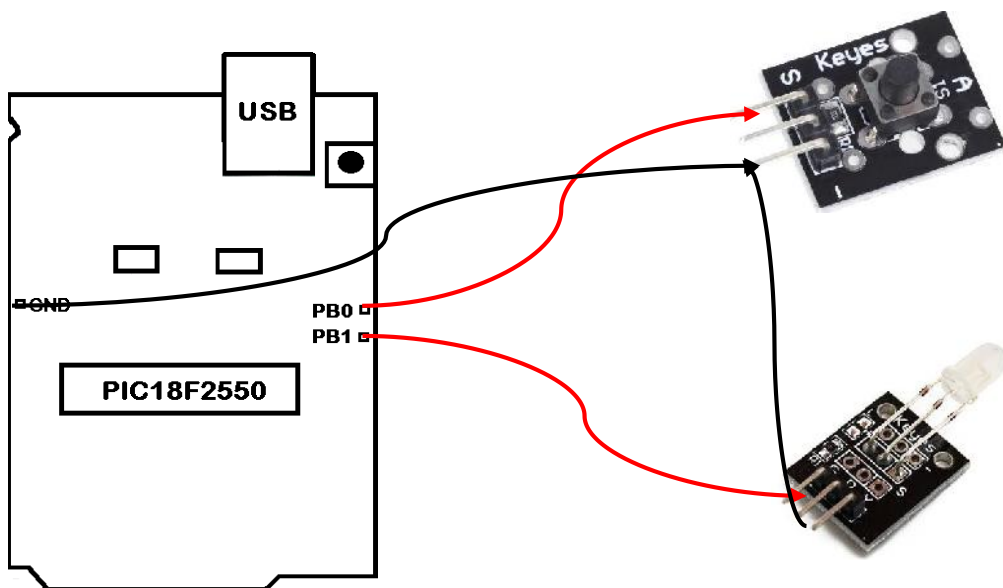


FIG 42 CIRCUITO DE CONEXIÓN PARA EL EJEMPLO 02.

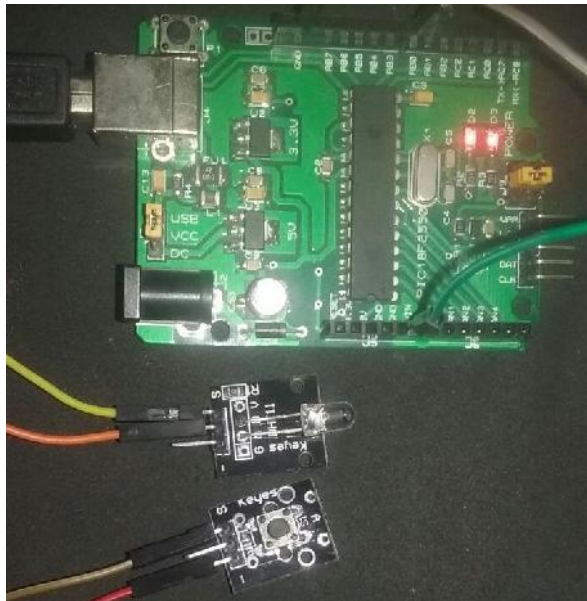


FIG 43 EJEMPLO 2 CIRCUITO

c) Ejemplo 3 “Conversor análogo digital con interface PC”.- Para este ejemplo usaremos el modulo del sensor de temperatura y además se desarrollara una aplicación en la PC donde mostrara los resultados obtenidos por la tarjeta de adquisición de datos.

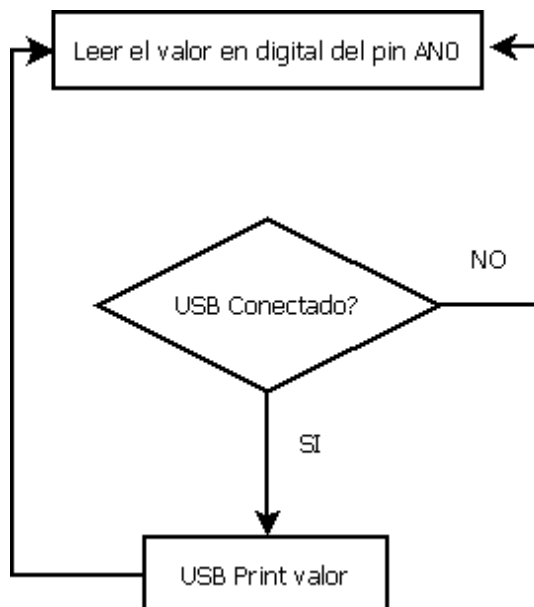


FIG 44 DIAGRAMA DE FLUJO PARA EL EJEMPLO 03.

```

4
5  int temp_an;
6  u8 buffer[24];
7
8  void setup() {
9  }
10
11 void loop() {
12  temp_an = analogRead (AN0);
13  temp_an = (5*temp_an)/255;
14  if (USB.available()) {
15
16  sprintf (buffer, "temperatura = %d \n\r", temp_an);
17  USB.send(buffer,24);
18

```

Output

```

\pic16\libc18f.lib E:\todos\tunky\dist\tools\bin\lib\pic16\pic18f2550.lib libsdcc.lib -w -sE:\todos\tunky\dist\lkr\18f2550.lkr E:\todos\tunky\dist\obj\application_iface.o E:\todos\tunky\dist\obj\usb_descriptors.o E:\todos\tunky\dist\obj\crt0unprg.o E:\todos\tunky\dist\source\main.o

Compilacion exitosa!!

Tama~no: 6560 / 24575 bytes (26% Usado)

```

FIG 45 CÓDIGO FUENTE DEL EJEMPLO 02.

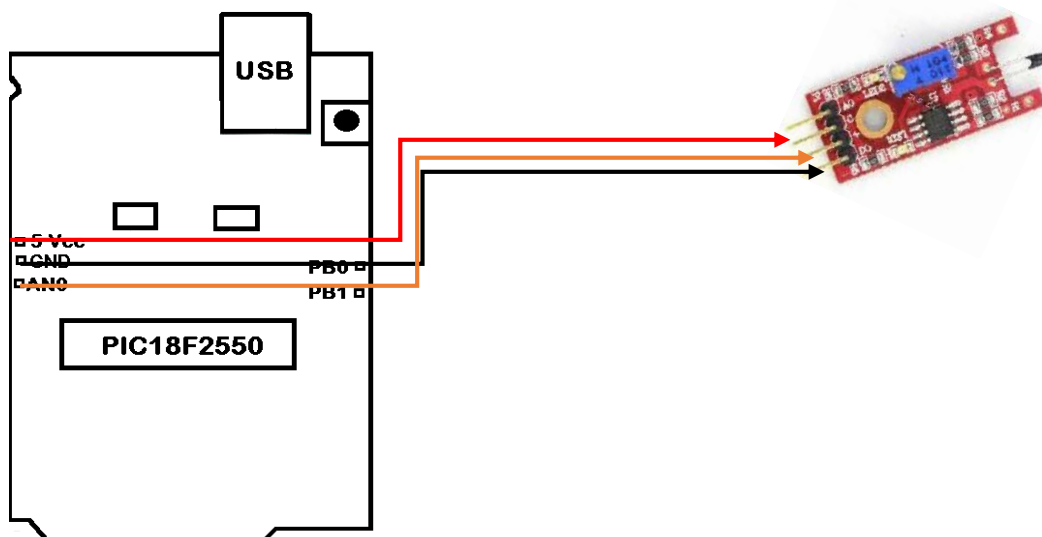


FIG 46 CIRCUITO DE CONEXION PARA EL EJEMPLO 03

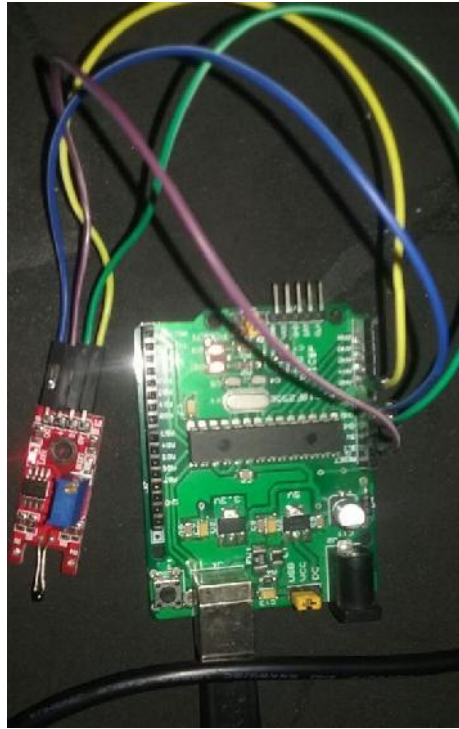


FIG 47 CIRCUITO DEL EJEMPLO 03

- Programa elaborado en Python para la comunicación con la tarjeta:

```
import sys
import usb
import time
class UsbPic:
    def __init__(self, vendor_id, product_id):
        busses = usb.busses()
        self.handle = None
        for bus in busses:
            devices = bus.devices
            for dev in devices:
                if dev.idVendor==vendor_id and dev.idProduct==product_id:
                    self.dev = dev

                    self.conf = self.dev.configurations[0]
                    self.intf = self.conf.interfaces[0][0]
                    self.endpoints = []
                    for endpoint in self.intf.endpoints:
                        self.endpoints.append(endpoint)
                    return
    def open(self):
        if self.handle:
            self.handle = None
```

```

try:
    self.handle = self.dev.open()
    self.handle.detachKernelDriver(0)
    self.handle.detachKernelDriver(1)
    self.handle.setConfiguration(self.conf)
    self.handle.claimInterface(self.intf)
    self.handle.setAltInterface(self.intf)
    return True
except:
    return False

def write(self, ep, buff, timeout = 100):
    try:
        return self.handle.interruptWrite(ep, buff, timeout)
    except:
        return 0
def read(self, ep, size, timeout = 100):
    try:
        return self.handle.interruptRead(ep, size, timeout)

    except:
        return []
def getDeviceName(self):
    return self.handle.getString(2, 40)


def DeviceConnect(self):
    self.device = UsbPic(0x04d8, 0xFEAA)
    if self.device.open():

        print "conectado"
    else:
        print "Dconectado"


def readtodo(self):
    self.device.write(1, [0x81])
    bytread = self.device.read(0x81, 64)
    return bytread
device = UsbPic(0x04d8, 0xFEAA)
device.DeviceConnect()
while True:
    print device.readtodo()
    time.sleep( 0.5)

```

5.2 ENCUESTA

El objetivo de esta encuesta es saber qué tiempo un estudiante de ingeniería electrónica o afines logra crear una aplicación en un corto tiempo.

1.- ¿Usted sabe programar microcontroladores en Asembler?

a. SI b. NO

2.- ¿Usted sabe programar microcontroladores en C++?

a. SI b. NO

3.- ¿Usted tiene algún conocimiento del lenguaje C++?

a. SI b. NO

4- ¿Usted logro programar el microcontrolador PIC en forma rápida utilizando el entorno de desarrollo integrado?

a. Si b. NO

5. Te gustaría seguir usando este entorno de desarrollo integrado para la programación de microcontroladores PIC.

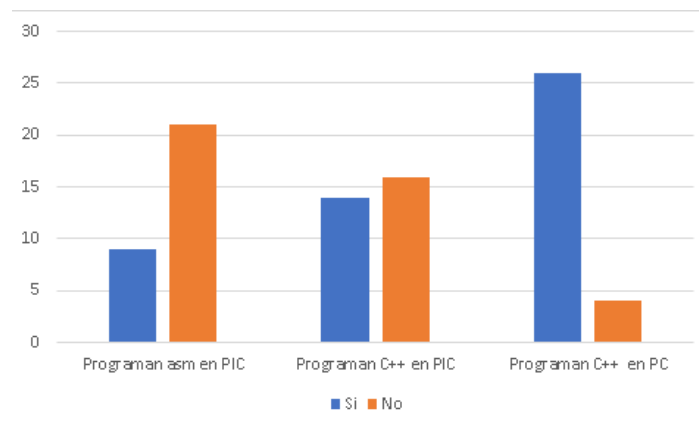
a. SI b. Tal Vez C. NO

¡GRACIAS POR SU APOYO!

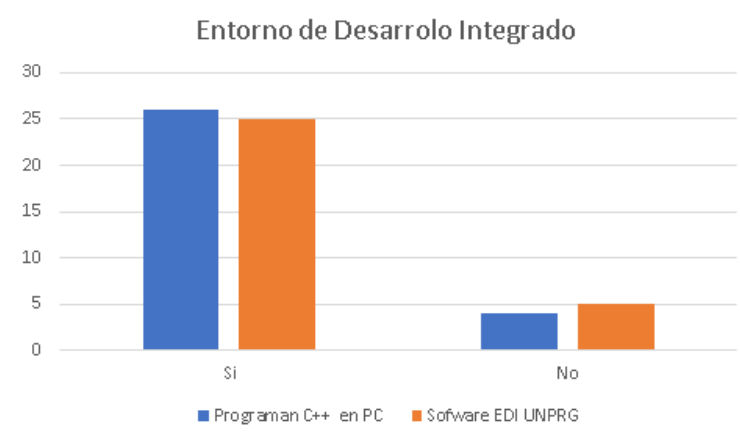
5.3 INTERPRETACION DE LOS DATOS

Las muestras de las encuestas incluyeron a 30 estudiantes de la escuela profesional de ingeniería electrónica del curso de Sistemas Digitales II donde 70% no sabe programar en Asembler para microcontroladores, y 86.6% de los encuestados tienen nociones sobre programación en lenguaje C++ sobre un escritorio o PC, esto debido a que durante su formación profesional aprenden programación I, y II en ese lenguaje, pero desconocen que puede ser usado para programar microcontroladores.

En la siguiente tabla muestra el nivel de conocimiento de programación de los estudiantes.



A continuación, podemos observar la siguiente tabla que representa que casi todos los estudiantes podrán realizar nuestro proyecto de tesis, con solo saber programar C++ en escritorio, pueden adaptarse gracias al entorno de desarrollo integrado rápidamente y así programar los microcontroladores.



CAPITULO VI

Conclusiones y Recomendaciones

6 CONCLUSIONES

- El crear un entorno de desarrollo integrado para microcontroladores, para poder desarrollar una tarjeta de adquisición de datos, rápida y fácil de ensamblar, donde los estudiantes consideran que es más sencillo de programar y además que contribuyendo su formación profesional en el área de sistemas digitales.
- Se utilizó software libre para crear el entorno de desarrollo y programar el microcontrolador.
- Los estudiantes de la escuela de ingeniería electrónica y afines, logran interactuar con la arquitectura del microcontrolador de forma intuitiva donde facilita su aprendizaje.
- Se ha creado funciones compatibles o similares a la plataforma Arduino con la finalidad de utilizar los módulos o sensores comerciales diseñadas para dicha plataforma.
- La plataforma de entorno de desarrollo integrado permite organizar mejor el código del programa para un mejor entendimiento para su posterior análisis.
- El entorno de desarrollo está diseñado para que el estudiante pueda editar, borrar, crear, compilar y subir el firmware de forma rápida y con solo clics.
- El docente con el uso de esta aplicación tendrá las herramientas adecuadas para desarrollar clases teóricas y prácticas,

6.1 RECOMENDACIONES

- Para el correcto uso de este proyecto de tesis, se necesita leer la descripción de cada función que se va a emplear al momento de programar, para que pueda ser parametrizado de forma correcta y funcione de forma óptima en el hardware.
- Evitar errores de conexión del circuito en el hardware como inversión de polos positivo y negativo, sobre tensión, etc., provocando daños en la tarjeta.
- Aprender a programar Python para que puedan crear aplicaciones con entorno grafico para poder mostrar los datos obtenidos de la tarjeta de adquisición de datos en la computadora.
- Crear nuevas librerías dinámicas para optimizar el código del firmware.

6.2 TRABAJOS FUTUROS

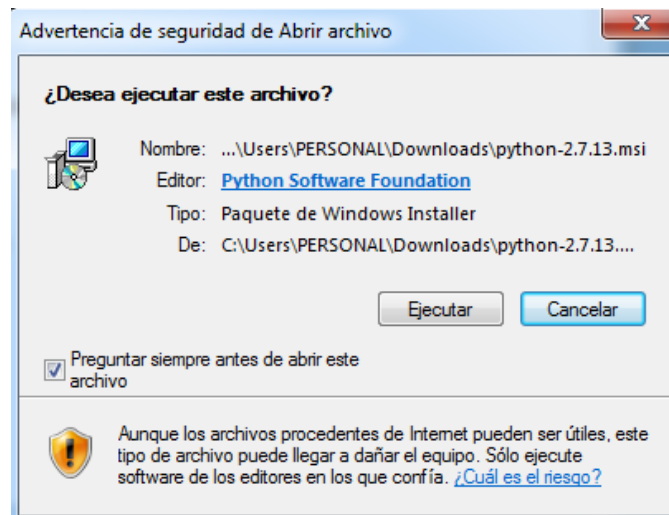
- El proyecto desarrollado en esta tesis funciona con plataforma Windows, una de las mejoras será hacerlo multiplataforma o sea que funcione en Linux y otras distribuciones de la misma.
- Se ampliará las funciones programadas para que pueda ser usado con más módulos o sensores comerciales.

CAPITULO VII

ANEXOS

7 INSTALACION DE PYTHON

1. Descargar Python de la web <https://www.python.org/downloads/> la versión más reciente 2.7.
2. Abrir el archivo python-2.7.13, y seguir los pasos de instalación:



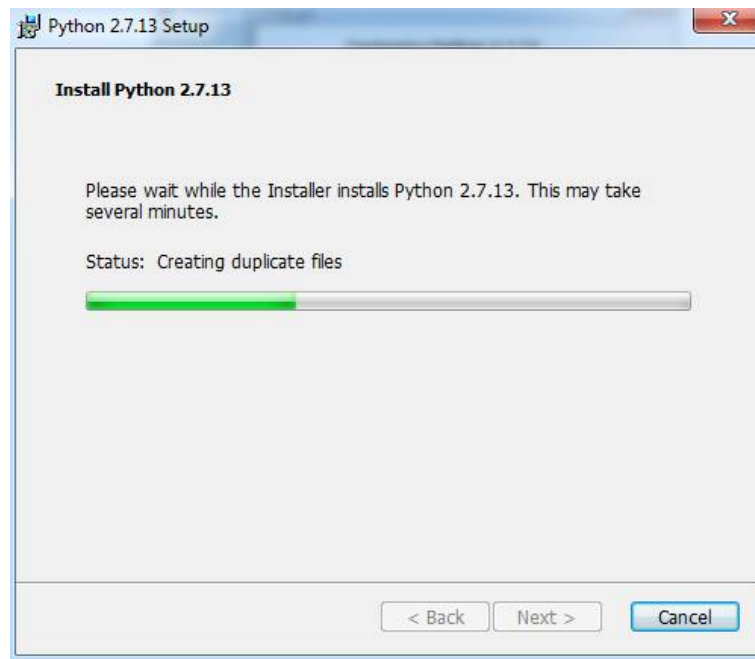
3. Presionamos el botón *Next>* y muestra el siguiente mensaje:



4. Elegimos el directorio de instalación



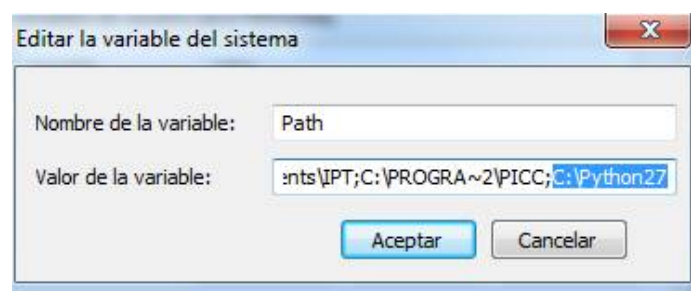
5. Dejamos que la instalación continúe presionando el botón *Next>*:



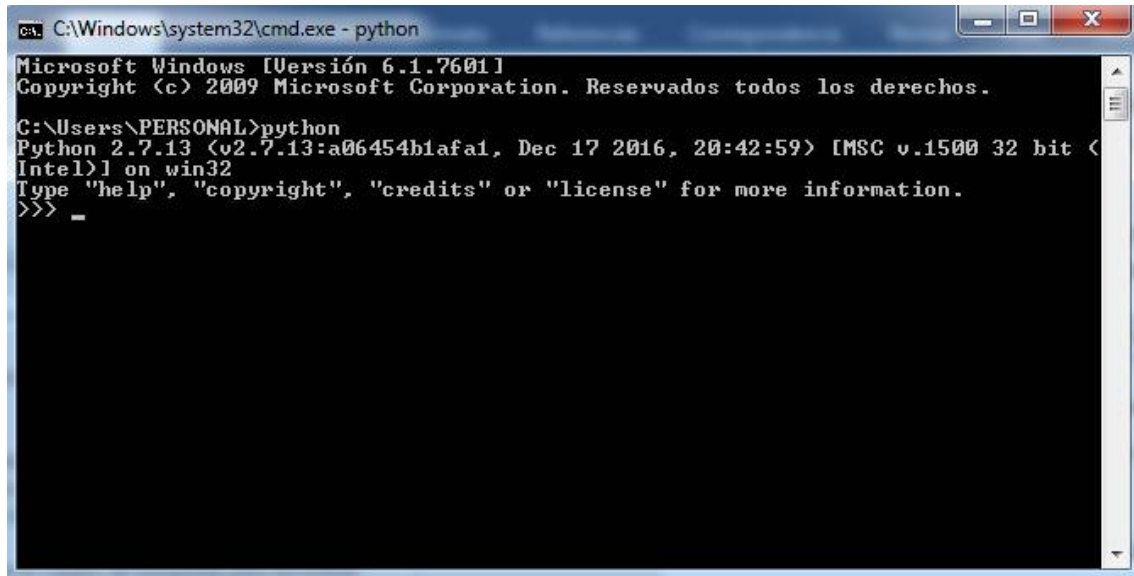
6. Terminamos la instalación



7. Comprobamos si Python está bien instalado verificando la ruta de instalación en variables de entorno del sistema operativo.



8. Nos vamos a símbolo de sistema o prompt para invocar a Python.



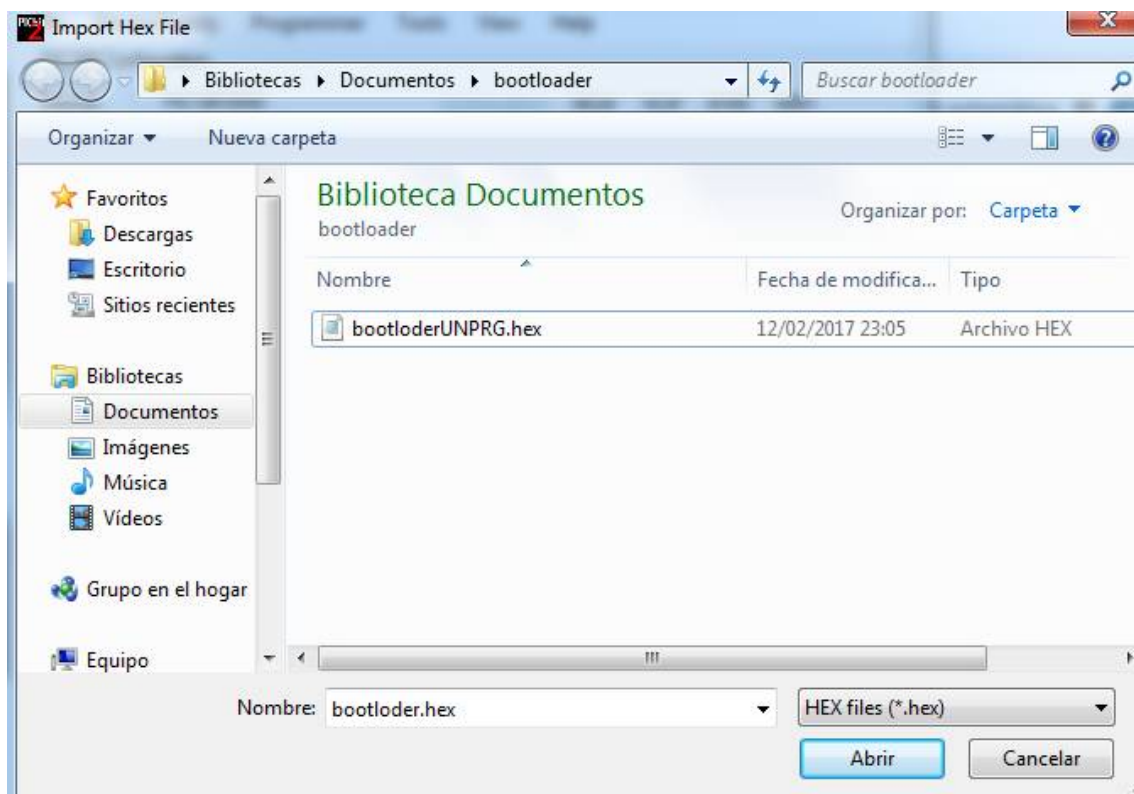
```
C:\Windows\system32\cmd.exe - python
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\PERSONAL>python
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit
Intel] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

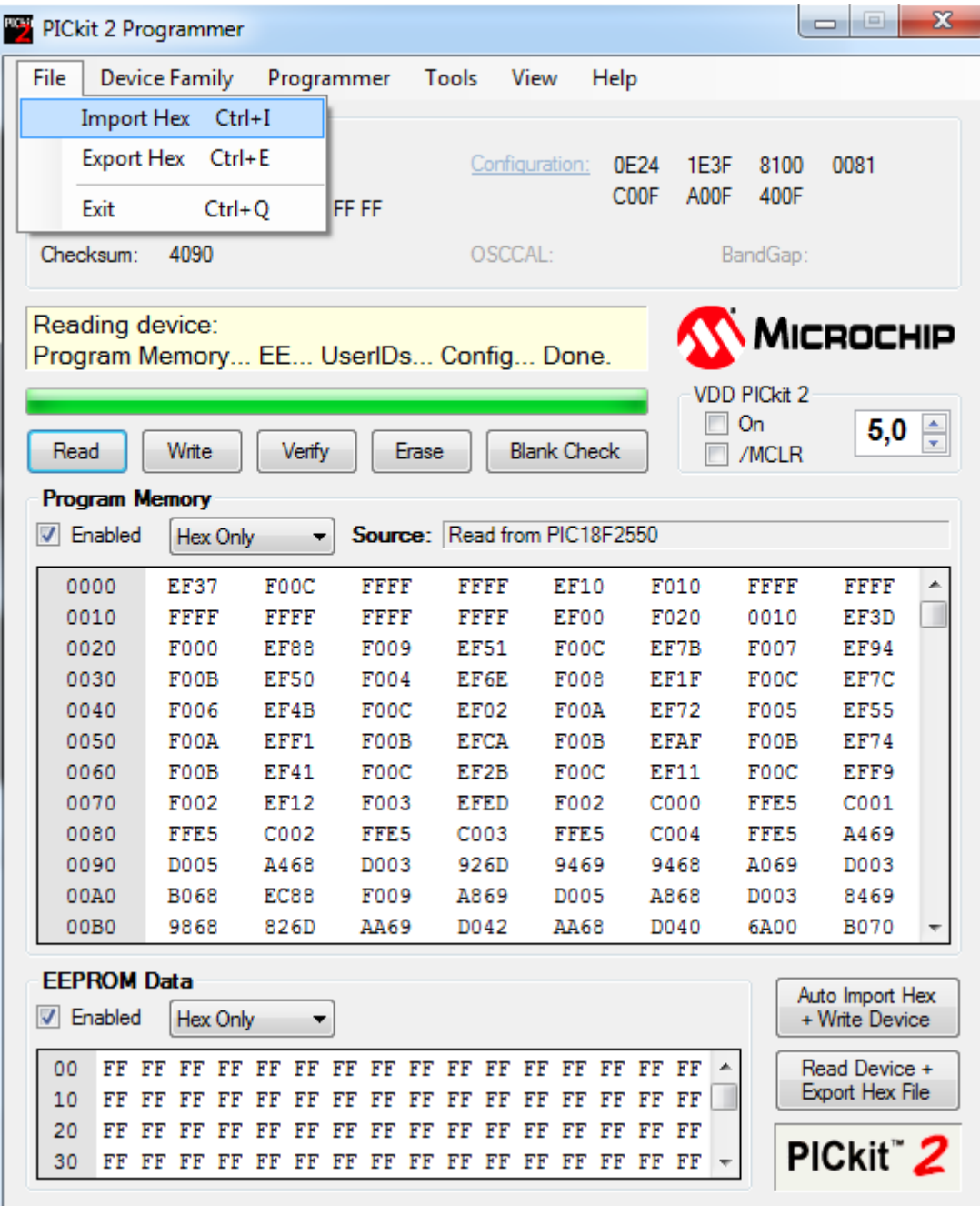
7.1 GRABAR EL BOOTLOADER

Abrimos el programador de microchip llamado PicKit versión 2.78 para grabar el bootloader al microcontrolador.

Clic en el botón abrir y muestra la siguiente ventana:

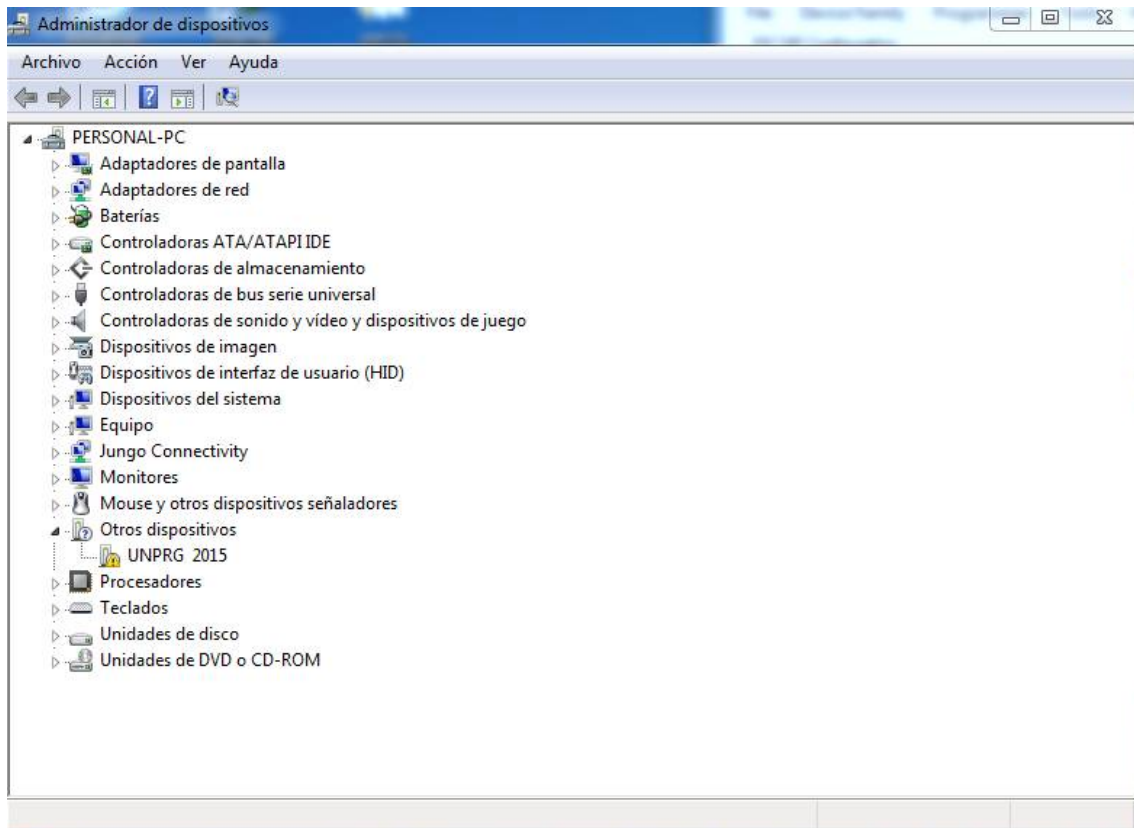


Después de que se halla seccionado el archivo automáticamente se graba en el microcontrolador.

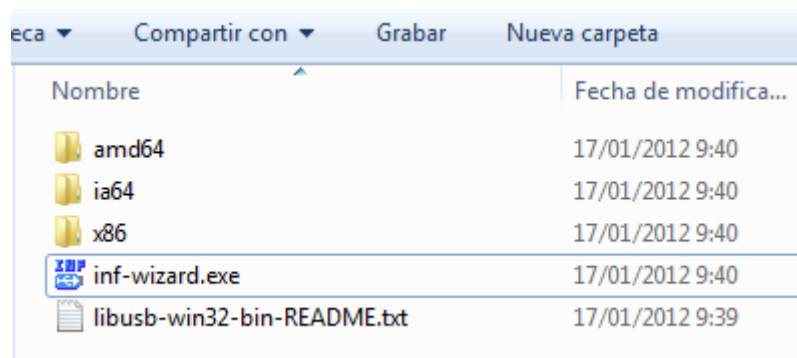


7.2 INSTALAR DRIVER DEL HADWARE

1. La primera vez que se conecta el **hardware** al computador, no es detectado por el sistema porque necesita los drivers de instalación.
2. Nos dirigimos a la ventana de administrador de dispositivos para ver nuestro **hardware** conectado, como muestra la siguiente ventana:



Abrimos la carpeta donde encontramos están los instaladores y abrimos el archivo inf-wizard.



Nos Muestra la siguiente ventana, no cambiamos ningún parámetro y clicamos el botón next> como muestra la siguiente venta:

The screenshot shows the 'libusb-win32 Inf-Wizard' window, specifically the 'Device Configuration' tab. It contains several input fields for device information:

Field	Value
Vendor ID (hex format)	0x04D8
Product ID (hex format)	0xFEAA
MI (hex format)	
Manufacturer Name	Microchip Technology, Inc.
Device Name	UNPRG 2015

At the bottom of the window are three buttons: '< Back', 'Next >', and 'Cancel'.

A continuación, crea el driver y ya tenemos listo para instalar.

The screenshot shows the 'libusb-win32 Inf-Wizard' window, specifically the 'Information' tab. It displays the summary of the driver package created:

Information

A windows driver installation package has been created for the following device:

Vendor ID:	0x04D8
Product ID:	0xFEAA
Device description:	UNPRG 2015
Manufacturer:	Microchip Technology, Inc.

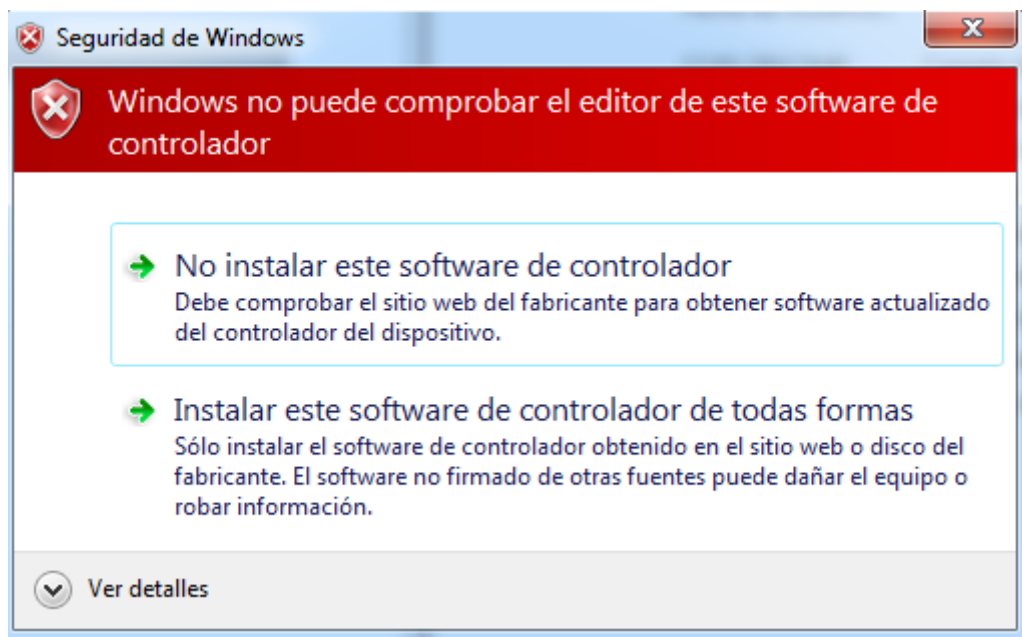
This package contains libusb-win32 v 1.2.6.0 drivers and support for the following platforms: x86, x64, ia64.

At the bottom, there are two buttons: 'Install Now..' and 'Done'.

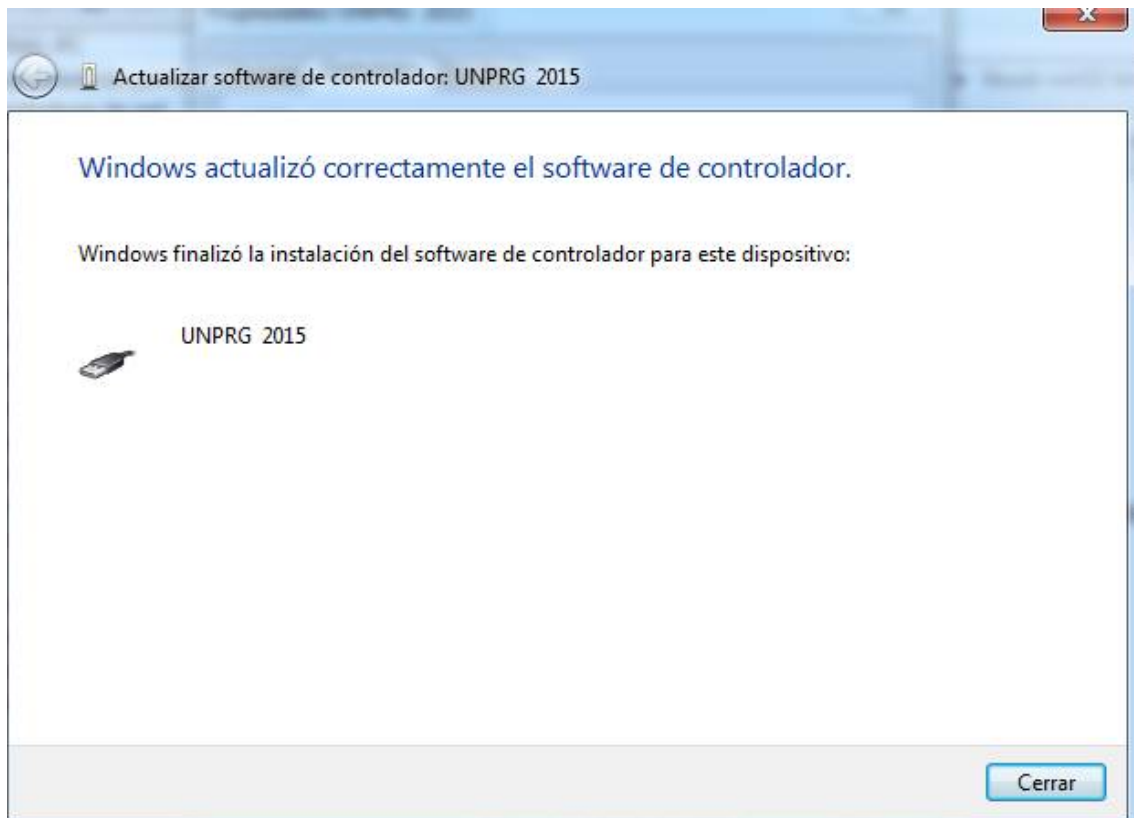
En la carpeta se ha creado un nuevo archivo llamado Installer_64.exe, como muestra la siguiente imagen:

Nombre	Fecha de modifica...	Tipo
amd64	17/01/2012 9:40	Carpeta de ar
ia64	17/01/2012 9:40	Carpeta de ar
license	15/02/2017 23:06	Carpeta de ar
x86	17/01/2012 9:40	Carpeta de ar
inf-wizard.exe	17/01/2012 9:40	Aplicación
installer_x64.exe	15/02/2017 23:06	Aplicación
installer_x86.exe	15/02/2017 23:06	Aplicación
libusb-win32-bin-README.txt	17/01/2012 9:39	Documento c
UNPRG_2015.inf	15/02/2017 23:06	Información:

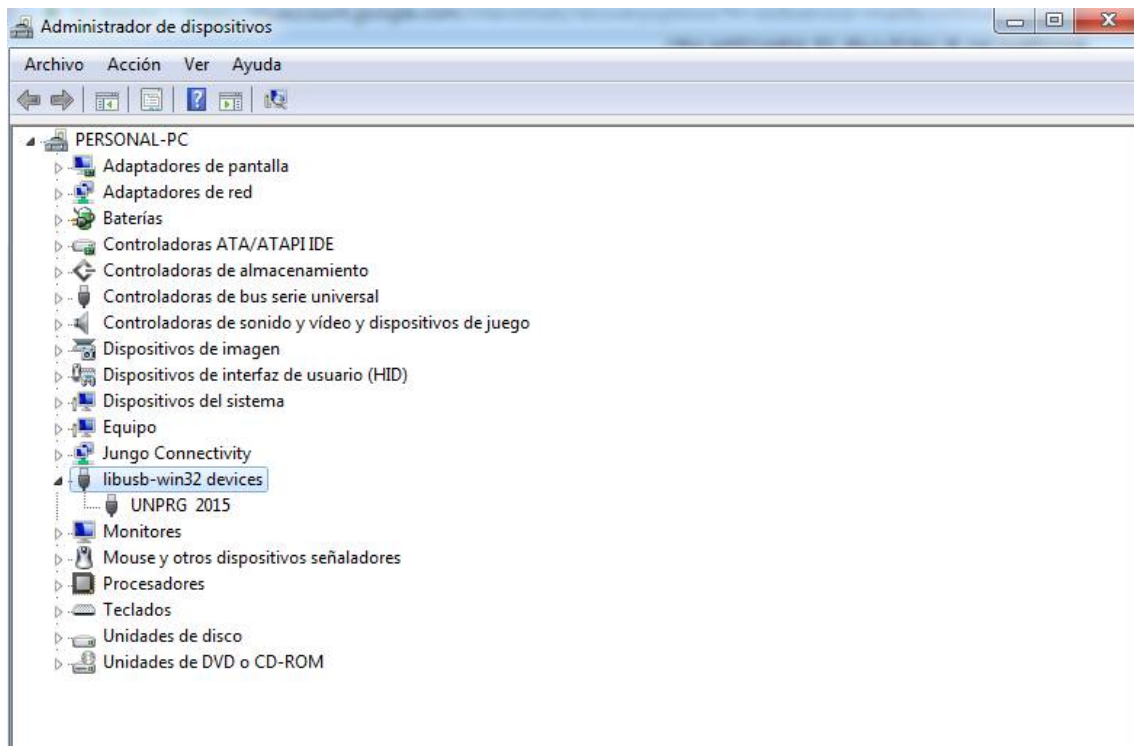
Le damos click al archivo señalado y muestra la siguiente advertencia de seguridad de Windows, donde nosotros debemos elegir la primera opción:



Después muestra la ventana siguiente:



Ya tenemos instalado correctamente el driver, para comprobar que lo que hemos hecho está bien, den la ventana administrador de dispositivos mostrara lo siguiente:



Ya está listo para usar nuestro hardware.

7.3 DESCRIPCION DE FUNCIONES

7.3.1 Funciones generales

- ✓ **AnalogRead.**- Puede obtener un valor digital de una tensión aplicada en su entrada, el valor leído es de 10 bits.
 - Sintaxis: AnalogRead (PIN);
 - PIN = 13 a 17
- ✓ **AnalogWrite** .- Envía una señal modulada por ancho de pulso PWM al PIN indicado, puede ser convertido en una señal analógica usando una pasa bajo.
 - Sintaxis: AnalogWrite (PIN0);
 - PIN = 11 y 12
- ✓ **Delay.**- Retardo de tiempo en milisegundos.
 - Sintaxis: Delay(ms);
 - ms = 1 a 4294967295 milisegundos.
- ✓ **DelayMicrosecond.**- retardo de tiempo en microsegundos.
 - **Sintaxis:** DelayMicrosecond(us);
 - us : 1 a 6553 5 microsegundos
- ✓ **DigitalRead.**- Lee el nivel lógico del pin asignado. Retorna 0 como LOW o 1 como HIGH.
 - **Sintaxis:** DigitalRead (PIN);
 - PIN = 1 al 13
- ✓ **DigitalWrite.**- Escribe un nivel lógico al pin asignado.
 - **Sintaxis:** DigitalWrite (PIN,NIVEL);
 - PIN = 1 al 13
 - NIVEL = 1 como HIGH o 0 como LOW
- ✓ **RandomSeed.**- Genera una semilla para ser utilizado al generar números seudo aleatorio.
 - **Sintaxis:** RadomSeed();
- ✓ **Rand.**- Retorna un valor entero de 16 bit generado seudo-aleatorio.
 - **Sintaxis:** Rand();

- ✓ **Sprintf.-** Escribe cualquier combinación de valores numéricos, caracteres sueltos y cadenas de caracteres.
 - **Sintaxis:** Sprintf(char * buffer, char * format, args, ...);
- ✓ **strcmp .-** Compara dos cadenas de texto, devuelve el valor entero 0 si las dos cadenas de texto son iguales.
 - **Sintaxis :** result = strcmp(*S1, *S2);
 - **S1** = dirección de la cadena 1.
 - **S2** = dirección de la cadena 2.

7.3.2 Funciones de Comunicación

- a) La clase de dispositivo de comunicación (CDC) es una forma de uso general para habilitar todos los tipos de comunicaciones en el bus serie universal (USB):

- **CDC.read** – Lee un carácter desde el dispositivo CDC/USB.
 - **Sintaxis :** CDC.read(Array);
- **CDC.write** – Escribe un caracter en el dispositivo CDC/USB.
 - **Sintaxis :** CDC.write(buffer, int);

- b) El I2C es un bus con múltiples maestros, lo que significa que se pueden conectar varios chips al mismo bus y que todos ellos pueden actuar como maestro, sólo con iniciar la transferencia de datos.

- **I2C.int .-** Inicia el puerto de comunicación I2C
 - **Sintaxis :** I2C.init(I2C_MODE);
 - **I2C_MODE:** se puede utilizar dos velocidades
 - I2C_100KHZ** - 100Khz
 - I2C_400KHZ** - 400Khz
- **I2C.read.-** Lee un carácter desde el dispositivo.
 - **Sintaxis :** I2C.read(Array);
- **I2C.write.-** Escribe un caracter en el dispositivo.
 - **Sintaxis :** I2C.write (buffer, int);

c) Interrupciones

```
void UserInterrupt()
{
    if (PIR1bits.TMR1IF)
    {
        PIR1bits.TMR1IF=0;
        if (digitalRead(0)==0)
            digitalWrite(0,HIGH);
        else
            digitalWrite(0,LOW);
    }
}

void setup()
{
    T1CONbits.RD16=1;
    T1CONbits.T1RUN=0;
    T1CONbits.T1CKPS1=1;
    T1CONbits.T1CKPS0=1;
    T1CONbits.T1OSCEN=0;
    T1CONbits.TMR1CS=0;
    T1CONbits.TMR1ON=1;
    PIE1bits.TMR1IE=1;
    INTCONbits.PEIE=1;
    INTCONbits.GIE=1;
}

void loop()
{
}
```

7.3.3 Algunas funciones propias del compilador SDCC:

ctype.h

```
extern char iscntrl (unsigned char ) ;
extern char isdigit (unsigned char ) ;
extern char isgraph (unsigned char ) ;
extern char islower (unsigned char ) ;
extern char isupper (unsigned char ) ;
extern char isprint (unsigned char ) ;
extern char ispunct (unsigned char ) ;
extern char isspace (unsigned char ) ;
extern char isxdigit (unsigned char ) ;
isalnum(c)
isalpha(c)
tolower(c)
toupper(c)
toascii(c)
```

math.h.-En esta librería se encuentran las funciones relacionadas con las matemáticas por ejemplo:

/ Funciones trigonometricas */*

```
float cotf(const float x) _MATH_REENTRANT;
float asinf(const float x) _MATH_REENTRANT;
float acosf(const float x) _MATH_REENTRANT;
float atanf(const float x) _MATH_REENTRANT;
float atan2f(const float x, const float y);
```

/ funciones hiperbolicas */*

```
float sinhf(const float x) _MATH_REENTRANT;
float coshf(const float x) _MATH_REENTRANT;
float tanhf(const float x) _MATH_REENTRANT;
```

/ funciones logaritmicas y exponenciales */*

```
float expf(const float x);
float logf(const float x) _MATH_REENTRANT;
float log10f(const float x) _MATH_REENTRANT;
```

/ funciones de valor absoluto punto flotante y enteros */*

```
float frexpf(const float x, int *pw2);
float ldexpf(const float x, const int pw2);
float ceilf(float x) _MATH_REENTRANT;
float floorf(float x) _MATH_REENTRANT;
```

```
float modff(float x, float * y);
```

stdlib.h.- En esta librería se encuentran las funciones de conversión como, por ejemplo:

```
/* Convierte una cadena ASCII a Float */
```

```
float atof (char *);
```

```
/* convierte una cadena ASCII a integer */
```

```
int atoi (char *);
```

```
/* convierte una cadena ASCII a un numero entero */
```

```
long atol (char *);
```

```
/* convierte un número entero a una cadena de texto */
```

```
void uitoa(unsigned int, __data char *, unsigned char);
```

```
void itoa(int, __data char*, unsigned char);
```

```
/* convierte un número entero corto a una cadena de texto */
```

```
void ultoa(unsigned long, __data char *, unsigned char);
```

```
void ltoa(long, __data char*, unsigned char);
```

```
/* convierte un número con decimals a una cadena de texto */
```

```
extern char x_ftime(float, __data char *, unsigned char, unsigned char);
```

stdlib.h.- En esta librería se encuentra las funciones para usarlas con las cadenas de texto o ASCII code.

```
char *strcat (char *, char *);
```

```
char *strchr (char *, char);
```

```
int strcmp (char *, char *);
```

```
char *strcpy (char *, char *);
```

```
int strcspn(char *, char *);
```

```
int strlen (char *);
```

```
char *strlwr (char *);
```

```
char *strncat(char *, char *, size_t );
```

```
int strncmp(char *, char *, size_t );
```

```
char *strncpy(char *, char *, size_t );
```

```
char *strpbrk(char *, char *);
```

```
char *strrchr(char *, char);
```

```
int strspn (char *, char *);
```

```
char *strstr (char *, char *);
```

```
char *strtok (char *, char *);  
char *strupr (char *);
```

```
void *memccpy(void *, void *, char, size_t);  
void *memchr(void *, char, size_t);  
int  memcmp (void *, void *, size_t);  
void *memcpy (void *, void *, size_t);  
void *memmove (void *, void *, size_t);  
void *memrchr(void *, char, size_t);  
void *memset (_STRING_SPEC void *, unsigned char, size_t );
```

CAPITULO VII

Referencias Bibliográficas

[Lutz, 2013] Lutz, M. (2013). *Learning python*. . O'Reilly Media, Inc."

[Pérez, 2007] Pérez, E. M. (2007). *Microcontroladores PIC: sistema integrado para el autoaprendizaje*. Marcombo.

[Van Rossum and Drake Jr, 1995] Van Rossum, G. and Drake Jr, F. L. (1995). *Python tutorial*.

[O. Barra Zapata y F. Barra Zapata, 2011] O. Barra Zapata, F. Barra Zapata, (2011). *Microcontroladores PIC con programación PBP*.

[Dogan , 2007] Dogan , I. (2007). *Programación de Microcontroladores PIC: Desarrollo de 30 proyectos con PIC Basic y PIC Basic Professional*.

[Maruch , 2011] Stef Maruch, Aahz Maruch (2011). *Python For Dummies*.

[Allen, 2002] Allen B. (2002). *Think Python: An Introduction to Software Design*.

[Dunn, 2006] Robin Dunn . (2006). *wxPython in Action*.

[Rossano, 2011] Rossano V . (2011). *Proteus VSM: Simulación de Circuitos Electrónicos*.