UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

TESIS DE GRADO

PREVIA A LA OBTENCIÓN DEL TÍTULO

INGENIERO EN TELECOMUNICACIONES CON MENCIÓN EN GESTIÓN EMPRESARIAL

TEMA:

"DISEÑO E IMPLEMENTACIÓN DE TARJETAS ESCUDOS PARA UNA PLATAFORMA HARDWARE CON PIC18F4550 PARA EL APRENDIZAJE Y LA ELABORACIÓN DE PROYECTOS CON MICROCONTROLADORES"

ELABORADO POR:

WILSON ALEJANDRO MADRID PACHECO

Guayaquil, 08 de Mayo del 2014

TESIS DE GRADO

TÍTULO

“DISEÑO E IMPLEMENTACIÓN DE TARJETAS ESCUDOS PARA UNA PLATAFORMA HARDWARE CON PIC18F4550 PARA EL APRENDIZAJE Y LA ELABORACIÓN DE PROYECTOS CON MICROCONTROLADORES"

Presentada a la Facultad de Educación Técnica para el Desarrollo, Carrera de Ingeniería en Telecomunicaciones de la Universidad Católica de Santiago de Guayaquil.

POR:

WILSON ALEJANDRO MADRID PACHECO

Para dar cumplimiento con uno de los requisitos para optar por el título de:

INGENIERO EN TELECOMUNICACIONES

MENCIÓN EN GESTIÓN EMPRESARIAL

Miembros del Tribunal

_____	_____
Ing. Manuel Romero Paz **Decano de la Facultad**	**Ing. Eduardo Mendoza, MGS** **Docente Tutor**
_____	_____
Ing. Carlos Romero Rosero **Revisor de Tesis**	**Ing. Pedro Tutiven López** **Revisor de Tesis**
_____	_____
Eco. Gladys Contreras **Coordinadora Administrativa**	**Ing. Luis Vallejo** **Coordinador Académico**

_____

Ing. Armando Heras Sánchez
Director de Carrera

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

INGENIERÍA EN TELECOMUNICACIONES

DECLARACIÓN DE RESPONSABILIDAD

WILSON ALEJANDRO MADRID PACHECO

DECLARO QUE:

El proyecto de tesis denominado **"DISEÑO E IMPLEMENTACIÓN DE TARJETAS ESCUDOS PARA UNA PLATAFORMA HARDWARE CON PIC18F4550 PARA EL APRENDIZAJE Y LA ELABORACIÓN DE PROYECTOS CON MICROCONTROLADORES"** ha sido desarrollado con base a una investigación exhaustiva, respetando derechos intelectuales de terceros conforme las citas que constan en las páginas correspondientes, cuyas fuentes se incorporan en la bibliografía.

Consecuentemente este trabajo es de mi autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance científico del proyecto de grado en mención.

Guayaquil, 08 de Mayo del 2014

_____

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

INGENIERÍA EN TELECOMUNICACIONES

AUTORIZACIÓN

Yo, WILSON ALEJANDRO MADRID PACHECO

Autorizo a la Universidad Católica de Santiago de Guayaquil, la publicación, en la biblioteca de la institución del proyecto titulado: **"DISEÑO E IMPLEMENTACIÓN DE TARJETAS ESCUDOS PARA UNA PLATAFORMA HARDWARE CON PIC18F4550 PARA EL APRENDIZAJE Y LA ELABORACIÓN DE PROYECTOS CON MICROCONTROLADORES"**, cuyo contenido, ideas y criterios son de nuestra exclusiva responsabilidad y autoría.

Guayaquil, 08 de Mayo 2014

_____

DEDICATORIA

La elaboración de este proyecto está dedicada a mis Padres y Hermanos, pilares fundamentales en mi vida. Sin ellos, jamás hubiese podido conseguir lo que tengo hasta hora. Su tenacidad y lucha insaciable han hecho de ellos el gran ejemplo a seguir y destacar, no solo para mí, sino para mis hermanos y familia en general. También dedico este proyecto a mi esposa, compañera inseparable de cada jornada de mi vida. Ella representó gran esfuerzo y pilar fundamental en momentos de decline y cansancio con su apoyo sentimental.

A mis suegros y cuñados, por haberme dado su fuerza, su apoyo incondicional, por creer en mí y no dudar a lo largo de todo este tiempo.

A ellos este proyecto, que sin ellos, no hubiese podido ser.

Wilson Alejandro Madrid Pacheco

AGRADECIMIENTO

Este proyecto es el resultado del esfuerzo y apoyo de muchas personas que han estado pendientes de mí durante el transcurso de mis estudios y de mi vida.

En primer lugar quiero agradecer a DIOS por ayudarme a superar los diversos obstáculos que me presenta la vida y por bendecirme para llegar hasta donde he llegado el día de hoy.

A mis padres Alejando Madrid, Cumandá Pacheco y a mis hermanos Carlos, Christian, Dhiego y Daniel quienes a lo largo de toda mi vida han apoyado y motivado mi formación académica, creyeron en mí en todo momento y no dudaron de mis habilidades.

A mi Esposa Fernanda y su familia Valdiviezo Heredia, quienes en todo momento me ayudaron a seguir con mis estudios, brindándome su confianza, apoyo y aprecio.

A los Docentes a quienes les debo gran parte de mis conocimientos, gracias a su paciencia y enseñanza, brindándome siempre su orientación con profesionalismo ético durante este largo camino.

A mi director de tesis, Msc. Eduardo Mendoza, a mis revisores de tesis Ing. Carlos Romero e Ing. Pedro Tutiven y finalmente a esta prestigiosa Universidad la cual abre sus puertas a jóvenes como yo, preparándonos para un futuro competitivo y formándonos como personas de bien.

¡Muchas Gracias!

INDICE DE CONTENIDO

RESUMEN ..... **XIV**

ABSTRACT ..... **XV**

INTRODUCCIÓN **1**

CAPITULO I **2**

ASPECTOS GENERALES **2**

1.1 PLANTEAMIENTO DEL PROBLEMA.- 2
1.2 JUSTIFICACIÓN.- 2
1.2.1 TECNOLÓGICAS.- 3
1.2.2 ECONÓMICAS.- 3
1.2.3 PRÁCTICAS.- 4
1.3 ALCANCE.- 4
1.4 OBJETIVOS.- 5
1.4.1 OBJETIVO GENERAL.- 5
1.4.2 OBJETIVOS ESPECIFICOS.- 5
1.5 HIPÓTESIS.- 6
1.6 METODOLOGÍA.- 6

CAPÍTULO II **7**

MARCO TEÓRICO **7**

2.1 ANTECEDENTES.- 7
2.2 EL MICROCONTROLADOR PIC18F4550.- 8
2.2.1 CARACTERISTICAS DEL PIC18F4550.- 9
2.2.1.1 RESISTENCIA DE LA MEMORIA.- 11
2.2.2.2 AUTO-PROGRAMACION.- 12
2.2.2.3 INSTRUCCIÓN DE SET EXTENDIDO.- 12
2.2.2.4 MÓDULO CCP.- 12
2.2.2.3 USART DIRECCIONABLE.- 12
2.2.2.4 CONVERTIDOR A/D DE 10-BIT.- 13
2.2.2.5 WATCHDOG (WDT).- 13
2.3 INTRODUCCION A LAS TARJETAS ESCUDOS.- 13

2.4 ESTRUCTURA DE UNA TARJETA ESCUDO.- 15
2.4.1 COMPONENTES.- 15
2.4.1.1 ENCAPSULADOS DE TECNOLOGÍA DE AGUJEROS PASANTES (THT).- 15
2.4.1.2 ENCAPSULADOS DE MONTAJE SUPERFICIAL (SMD/SMT).- 17
2.4.2 PISTAS O CAMINOS DE UN CIRCUITO IMPRESO.- 20
2.4.2.1 CAPAS EN UN CIRCUITO IMPRESO (PCB).- 21
2.4.3 APILAMIENTOS DE CABECERAS.- 23
2.4.4 LIQUIDACION FÍSICA.- 24
2.4.5 CONTENCION DE PATILLAS.- 25
2.4.6 INTERACCION SOFTWARE.- 26
2.4.7 REQUERIMIENTO DE ENERGÍA.- 27
2.5 ARQUITECTURA DE DISEÑO.- 27
2.5.1 DISEÑO MONO-BLOQUE.- 28
2.5.2 DISEÑO MONO-BLOQUE SIN CONEXIONES.- 28
2.5.3 DISEÑO MODULAR.- 29
2.6 VENTAJAS DE UNA TARJETA ESCUDO.- 30
2.7 DESVENTAJAS DE UNA TARJETA ESCUDO.- 31
2.8 TARJETAS ESCUDOS COMO PLATAFORMAS DOCENTES PARA LA ENSEÑANZA Y APRENDIZAJE DE MICROCONTROLADORES.- 32
2.9 COMPILADORPIC PIC-C CCS.- 33
2.9.1 PRINCIPALES VENTAJAS.- 35
2.10 PROGRAMACION EN PIC-C CCS.- 35
2.10.1 ESTRUCTURA DE UN PROGRAMA EN CCS.- 36
2.10.2 TIPOS DE DATOS.- 37
2.10.3 LAS CONSTANTES.- 38
2.9.4 VARIABLES.- 40
2.11 EAGLE: SOFTWARE PARA DISEÑOS DE PCB´S.- 42
2.11.1 INTERFAZ DE USUARIO DE EAGLE.- 42

CAPITULO III 46

DISEÑO DE TARJETAS ESCUDOS 46

3.1 TARJETA ESCUDO XBEE-USB CON MÓDULO XBEE DE 1MW A 100MW.- 47
3.1.1 ESPECIFICACIONES DE DISEÑO.- 48

3.1.2 APLICACIONES.- 51
3.2 TARJETA ESCUDO XBEE-TTL CON MÓDULO XBEE DE 1MW -100MW.- 52
3.2.1 ESPECIFICACIONES DE DISEÑO.- 53
3.2.2 APLICACIONES.- 56
3.3 TARJETA ESCUDO CON SENSOR INFRARROJO QRD1114.- 57
3.3.1 ESPECIFICACIONES.- 57
3.3.2 APLICACIONES.- 60
3.4 TARJETA ESCUDO PARA CONTROL DE SERVO-MOTORES Y MÓDULO SERVO-MOTOR HS311.- 60
3.4.1 ESPECIFICACIONES.- 61
3.5 TARJETA ESCUDO ETHERNET CON CONECTOR RJ45.- 65
3.6 TARJETA ESCUDO CON SENSOR ULTRASONICO HY-SFR05.- 69
3.3.1 ESPECIFICACIONES.- 69
3.6.2 APLICACIONES.- 70

CAPÍTULO IV 71

DISEÑO E IMPLEMENTACIÓN DE SOFTWARE DE LAS TARJETAS ESCUDOS 71

4.1 PROGRAMACIÓN DE LA TARJETA ESCUDO CON MÓDULO XBEE-USB.- 72
4.1.1 CODIGO DE PROGRAMACIÓN PARA PUERTOS DE CONEXIÓN CON EL MICROCONTROLADOR.- 72
4.2 PROGRAMACIÓN DE LA TARJETA ESCUDO CON MÓDULO XBEE-TTL.- 73
4.2.1 CÓDIGO DE PROGRAMACIÓN PARA PUERTOS DE CONEXIÓN CON EL MICROCONTROLADOR.- 74
4.3 TARJETA ESCUDO CON SENSOR INFRARROJO QRD1114.- 75
4.3.1 CÓDIGO DE PROGRAMACIÓN PARA PUERTOS DE CONEXIÓN CON EL MICROCONTROLADOR.- 76
4.4 TARJETA ESCUDO CONTROL DE SERVO-MOTORES CON MÓDULO SERVOMOTOR HS311.- 77
4.4.1 CÓDIGO DE PROGRAMACIÓN PARA PUERTOS DE CONEXIÓN CON EL MICROCONTROLADOR.- 77
4.5 TARJETA ESCUDO ETHERNET CON CONECTOR RJ45.- 78
4.5.1 CÓDIGO DE PROGRAMACIÓN PARA PUERTOS DE CONEXIÓN CON EL MICROCONTROLADOR.- 79
4.6 TARJETA ESCUDO CON SENSOR ULTRASÓNICO HY-SFR05.- 80

4.6.1 CÓDIGO DE PROGRAMACIÓN PARA PUERTOS DE CONEXIÓN CON EL MICROCONTROLADOR.- 81

CAPÍTULO V 82

COSTOS 82

GLOSARIO TÉCNICO 85

CONCLUSIONES 88

RECOMENDACIONES 90

BIBLIOGRAFIA 91

ANEXOS 95

MANUAL PARA CREAR PROYECTOS EN PIC C COMPILER 96
MANUAL PARA EL USO DE SOFTWARE PICKIT2 103
MANUAL PARA USO DE SOFTWARE ACCESSPORT 108
PRÁCTICA # 1 114
TEMA: COMUNICACIÓN UART 114
PRÁCTICA # 2 118
TEMA: RECEPCIÓN DE DATOS POR ANTENA XBEE 118
PRÁCTICA # 3 124
TEMA: SENSOR DE COLOR 124
PRÁCTICA # 4 129
TEMA: SENSOR DE LUZ 129
PRÁCTICA # 5 134
TEMA: BUZZER 134
PRÁCTICA # 6 139
TEMA: SERVOMOTOR 139
PRÁCTICA # 7 144
TEMA: CONTROL DE CARGA 144
PRÁCTICA # 8 149
TEMA: LUMINARIA INTELIGENTE 149

INDICE DE FIGURAS

Figura 2.1: Distribución de patillas del PIC18F4550.……9
Figura 2.2: Ejemplos de componentes de Tecnología de Agujeros Pasantes …..……..…..…….……..……16
Figura 2.3: Componentes THT en una tarjeta escudo ……..…..…..……..…16
Figura 2.4: Ejemplos de componentes de Montaje Superficial…….……18
Figura 2.5: Componentes de Montaje Superficial en una tarjeta escudo …….…….……18
Figura 2.6: Caminos de circuito impreso en una tarjeta escudo.…….……..…21
Figura 2.7: Ejemplos de 2 capas (a), 4 capas (b) y 6 capas……..23
Figura 2.8: Apilamientos de cabeceras de una tarjeta escudo ……...…….24
Figura 2.9: Liquidación física entre dos tarjetas escudos……...……...25
Figura 2.10: Patillas que contienen sus respectivas pistas de conexión……..…..26
Figura 2.11: Requerimiento de energía externo de una tarjeta escudo…...……27
Figura 2.12: Micro-Entrenador con Diseño Mono-Bloque…….……28
Figura 2.13: Micro-Entrenador con Diseño Mono-Bloque sin conexiones…….…29
Figura 2.14: Diseño Modular con diferentes bloques de conexiones…….……..30
Figura 2.15: Estructura básica de un programa en CCS…….……..37
Figura 2.16: Ejemplo de variables en CCS C…….…….40
Figura 2.17: Ventana de trabajo del editor de PCB de EAGLE……43
Figura 2.18: Ventana de trabajo en EAGLE.…….…….……44

Figura 3.1: Tarjeta Escudo XBEE-USB y el módulo Xbee de 1mw-100m..……...47
Figura 3.2: Especificaciones de diseño de la tarjeta escudo XBEE-USB……..……..48
Figura 3.3: Esquemático de la Tarjeta Escudo XBEE-USB……..50
Figura 3.4: Esquema PCB de la tarjeta escudo XBEE-USB……51
Figura 3.5: Tarjeta Escudo XBEE-TTL y el módulo Xbee de 1mw-100m..……....53
Figura 3.6: Especificaciones de diseño de la tarjeta escudo XBEE-TTL…….…….……..54
Figura 3.7: Esquemático de la Tarjeta Escudo XBEE-TTL ……..…..55
Figura 3.8: Esquema PCB de la tarjeta escudo XBEE-TTL……...56
Figura 3.9: Tarjeta Escudo con Sensor Infrarrojo QRD1114…….57

Figura 3.10: Especificaciones de diseño de la tarjeta escudo con sensor infrarrojo QRD1114……58
Figura 3.11: Esquemático de la Tarjeta Escudo Infrarrojo QRD1114……..59
Figura 3.12: Tarjeta Escudo para control de Servos-Motores y módulo servo-motor HS311……..…….61
Figura 3.13: Especificaciones de diseño de la tarjeta escudo para control de servo-motores…….62
Figura 3.14: Esquemático de la Tarjeta Escudo Controladora de Servos-Motores…….....63
Figura 3.15: Esquema PCB de la tarjeta escudo para servo-motores…….64
Figura 3.16 Tarjeta Escudo Ethernet con conector RJ45…….…..65
Figura 3.17 Especificaciones de diseño de Tarjeta Escudo Ethernet con conector RJ45……..66
Figura 3.18: Esquemático de la Tarjeta Escudo Ethernet con conector RJ45…...67
Figura 3.19: Esquema PCB de la tarjeta escudo Ethernet con conector RJ45…..68
Figura 3.20: Tarjeta Escudo con sensor Ultrasónico HY-SFR05……..69
Figura 3.21: Especificaciones de diseño de la tarjeta escudo con sensor Ultrasónico HY-SFR05……..70
Figura 4.1: Conexiones de patillas de la tarjeta Ethernet…….……..77

INDICE DE TABLAS

Tabla 2.1: Características del PIC18F4550……..10
Tabla 2.2: Número de capas en un PCB ……..……..…….22
Tabla 2.3: Tipos de Datos…….…….38
Tabla 2.4: Tipos de Constantes en CCS C ……..…….……...39
Tabla 2.5: Definición de Constantes con un sufijo en CCS C……39
Tabla 2.6: Caracteres Especiales en CCS C…….……...40
Tabla 2.7: Tipos de archivos en EAGLE…….……...40

Tabla 4.1: Conexiones de puertos……..…70
Tabla 4.2: Conexiones de puertos……..72
Tabla 4.3: Conexiones de puertos……..73

Tabla 4.4: Conexiones de puertos……..75
Tabla 4.5: Conexiones de puertos……..78

RESUMEN

El presente trabajo de tesis abarca aspectos metodológicos conocidos en su mayoría por los estudiantes y docentes de la Facultad Técnica, que permitirán desarrollar una gran variedad de aplicaciones tecnológicas para comprender y familiarizarse con dispositivos electrónicos encabezados por microcontroladores, desarrollados de tal manera que sean compatibles con tecnologías que en la actualidad son muy comunes, y están presentes en ambientes domésticos, industriales, mecánicos, etc.

El capítulo 1, describe las generalidades del proyecto de graduación donde se justifica la propuesta ante un problema palpable dentro de la Facultad Técnica.

El capítulo 2, describe un marco teórico amplio y muy investigativo correspondiente a las tarjetas escudos, sirviendo para el aprendizaje académico.

El capítulo 3, describe la parte física y el diseño de las tarjetas escudos, considerando la máxima cantidad de funcionalidades basadas en micro-controladores y algunas aplicaciones que pueden ayudar a entender su funcionamiento y la forma de como interactuar entre medios tangibles.

El capítulo 4, describe la programación de las tarjetas escudos realizadas en el capítulo anterior y así poder adquirir y mostrar datos de acuerdo a la funcionalidad que se requiere, aprovechando al máximo su potencial.

El capítulo 5, se refiere al costo económico para la implementación de cada tarjeta escudo.

ABSTRACT

This thesis covers methodological aspects known mostly by students and teachers that will develop a variety of applications to understand and become familiar with electronic devices, developed in a way that is compatible with technologies that currently are common, and are present in domestic, industrial environments, mechanics, etc.

Chapter 1 describes the general graduation project where the proposal before a palpable problem within the Technical College is warranted.

Chapter 2 describes a very broad theoretical framework and research relevant to the shields cards, serving for academic learning.

Chapter 3 describes the physical part and the design of the shield cards, considering the maximum number of features based on micro - controllers and some applications that can help you understand how it works and how to interact with tangible media.

Chapter 4 describes the programming of cards shields made in the previous chapter so you can acquire and display data according to the functionality that is required to maximize its potential.

Chapter 5 refers to the economic cost to implement each shield card.

INTRODUCCIÓN

En la actualidad el uso cotidiano de dispositivos electrónicos es muy frecuente, ya que se los utiliza en un sin número de aplicaciones que podemos encontrar tanto en nuestro hogar en aplicaciones como juguetes, pantallas led, teléfonos móviles, etc. En cualquier lugar público o privado que visitemos en aplicaciones como detector de huellas, sensores de movimiento, sensores de luz, etc. En fin se los puede encontrar en cualquier equipo electrónico que realice alguna función de acción inteligente.

Entonces, el uso de plataformas hardware con tarjetas escudos, que son dispositivos electrónicos que integran microcontroladores, sensores digitales y componentes electrónicos que proporcionan diversas capacidades básicas y comunicación de datos; ayudan a que su funcionalidad, versatilidad y sus prestaciones se amplíen adecuadamente de acuerdo al microcontrolador que se use. En este sentido, una de las grandes ventajas de trabajar con tarjetas escudos es la facilidad de integración de tarjetas y módulos.

A las tarjetas escudos se les pueden acoplar módulos, que son componentes que se conectan encima de ellas para extender sus capacidades significativamente, como conexión Ethernet, XBEE o controladores de Servos-Motores, etc. A su vez estas tarjetas permiten conectarse a otras para seguir expandiendo las capacidades de la plataforma hardware principal.

Esta investigación aplicada y experimental, pretende proporcionar herramientas hardware que faciliten para los estudiantes el aprendizaje, reduzcan tiempo y recursos económicos en el desarrollo de proyectos y prototipos basados en microcontroladores.

CAPITULO I

ASPECTOS GENERALES

1.1 PLANTEAMIENTO DEL PROBLEMA.-

La necesidad de los estudiantes de la Facultad Técnica para desarrollar proyectos basados en microcontroladores aplicables a las diferentes asignaturas existentes en la malla curricular, hacen que sea necesaria la implementación y el diseño de tarjetas escudos, debido a su versatilidad para adaptarse a multitud de usos, comunicación a PC y a otros dispositivos electrónicos.

Además, la necesidad de disponer de plataformas electrónicas amigables y económicas que permitan el acopio de material y librerías orientadas a facilitar el desarrollo de prototipos basados en microcontroladores de acuerdo a las necesidades académicas del estudiante.

Fundamentalmente estos aspectos, llevan a plantearse el diseño e implementación de tarjetas escudos, que incrementen la versatilidad de las plataformas hardware y ayuden al aprendizaje de los estudiantes de la Facultad Técnica.

1.2 JUSTIFICACIÓN.-

Las razones por las que este proyecto se desea llevar a cabo son de carácter tecnológico, económico y práctico. Con estas tarjetas escudos, se pretende generar aplicaciones y soluciones orientadas a mejorar la calidad y disminuir el tiempo de elaboración de proyectos con microcontroladores por parte de

estudiantes y docentes, y tratar de cubrir un cierto número de necesidades que se nombran a continuación:

1.2.1 TECNOLÓGICAS.-

- ✓ La necesidad de probar circuitos integrados y módulos digitales, aplicando los conocimientos adquiridos antes y después de cada práctica.

- ✓ La necesidad de disponer de tarjetas escudos que faciliten el uso y la programación de sensores, dispositivos digitales y analógicos.

- ✓ La necesidad de disponer de librerías y más repertorios digitales orientados a fortalecer la programación de sistemas electrónicos embebidos.

1.2.2 ECONÓMICAS.-

- ✓ La necesidad de disponer de una plataforma sencilla, versátil, amigable y económica, que permita adquirir conocimientos con microcontroladores.

- ✓ La necesidad de reducir la dependencia tecnológica, a través de la compra de tarjetas de desarrollo, que muchas de las veces no se ajustan a las necesidades de los estudiantes, docentes e investigadores de la Facultad.

1.2.3 PRÁCTICAS.-

- ✓ La necesidad de los estudiantes de la Facultad Técnica para desarrollar proyectos basados en microcontroladores aplicables a las diferentes asignaturas existentes, hacen que sea necesario el diseño e implementación de tarjetas escudos, debido a su versatilidad para adaptarse a multitud de usos, comunicación a PC y a otros dispositivos.

- ✓ La necesidad constante de los estudiantes para desarrollar cada semestre proyectos de clases, especialmente de tutorías.

- ✓ La necesidad de disponer de plataformas amigables que permitan y favorezcan el acoplo de material y librerías orientadas a facilitar y acelerar el desarrollo de proyectos y prototipos.

Fundamentalmente estos aspectos, llevan a plantearse el diseño e implementación de tarjetas escudos, que incrementen la versatilidad de la plataforma hardware.

Además, se tendría la oportunidad más eficiente y sencilla de montar, desmontar y programar directamente las tarjetas escudos que se usan cotidianamente en nuestro ámbito, lo cual permitiría familiarizarse de manera más directa con los distintos periféricos que se adaptan a ellas, oportunidad que no se da al hacer prácticas y proyectos en los simuladores disponibles.

1.3 ALCANCE.-

El presente anteproyecto de tesis tiene como finalidad realizar el diseño e implementación de tarjetas escudos adaptables que complementarán a una plataforma de hardware con microcontrolador PIC 18F4550 elaborada en

otra tesis, y así poder facilitar el estudio práctico y la elaboración de proyectos con microcontroladores en la Facultad Técnica de la Universidad Católica Santiago de Guayaquil , teniendo como objetivo principal su aplicación directa en un hardware que permita realizar prácticas programables, ayudando de esta manera el aprendizaje de los estudiantes.

Las tarjetas escudos y los módulos que se implementarán son:

- Tarjeta escudo Xbee-USB con módulo Xbee de 1mw-100m.
- Tarjeta escudo XBEE-TTL con módulo Xbee de 1mw-100m.
- Tarjeta escudo con Sensor Infrarrojo QRD1114.
- Tarjeta escudo Control de Servo-Motores con módulo servomotor HS311.
- Tarjeta escudo Ethernet con conector RJ45.
- Tarjeta escudo con Sensor Ultrasónico HY-SFR05.

1.4 OBJETIVOS.-

1.4.1 OBJETIVO GENERAL.-

- Diseñar tarjetas escudos como soporte de sensores digitales y actuadores eléctricos que sean compatibles con la plataforma hardware basada en el microcontrolador PIC18F4550, mejorando y ampliando sus funcionalidades.

1.4.2 OBJETIVOS ESPECIFICOS.-

- Desarrollar herramientas que aporten a los estudiantes de la Facultad Técnica un hardware probado y confiable que les permita realizar una gran variedad de aplicaciones tecnológicas, sin necesidad de construir un circuito en especial para cada práctica.

- Contribuir al desarrollo de proyectos de tutoría e investigación con microcontroladores a las diferentes asignaturas, proporcionando tarjetas económicas que se ajusten a las necesidades académicas y económicas del estudiante.

- Desarrollar librerías, controladores (drivers) en lenguaje C y manuales de prácticas, para los diferentes módulos y dispositivos digitales que sirvan como repertorio digital para los estudiantes de la Facultad Técnica.

1.5 HIPÓTESIS.-

El uso y la implementación de las tarjetas escudos es una opción sustentable para potencializar la plataforma hardware con PIC18f4550 y la adquisición de conocimientos con microcontroladores por parte de Estudiantes de la Facultad Técnica de la Universidad Católica de Santiago de Guayaquil.

Estas tarjetas garantizarán un mejor desempeño y desenvolvimiento académico por parte de los estudiantes durante las prácticas y desarrollo de proyectos de tutorías de las diferentes asignaturas de la Facultad Técnica.

1.6 METODOLOGÍA.-

Este trabajo de tesis se enmarca dentro del tipo de investigación descriptiva, exploratoria e implementaria, utiliza un enfoque metodológico cuantitativo porque se debe crear un cierto número de tarjetas escudos y diseñar sus respectivos esquemáticos con el fin de que sean lo más económicas y didácticas posibles.

CAPÍTULO II

MARCO TEÓRICO

2.1 ANTECEDENTES.-

E. Mendoza (2007) "El desarrollo constante de la electrónica digital ha dado lugar a dispositivos cada vez más complejos, entre ellos los microcontroladores", que son circuitos integrados que incorporan todos los bloques funcionales de un sistema en un único encapsulado, los cuales permiten la comunicación mediante combinaciones de bits y generan señales digitales internas y externas, para ejecutar de manera continua una secuencia de instrucciones que permitan controlar un sistema o subsistema electrónico".

Siendo el microcontrolador el núcleo de un sistema electrónico versátil de bajo costo y reducido tamaño, llamado tarjetas escudos, que son capaces de detectar señales de entrada y generar señales de salida de un equipo, sistema o instrumento; características que permiten la fácil implementación y comunicación de sistemas de inteligencia distribuida a lo largo de sistemas más complejos, donde los microcontroladores son los semiconductores más abundantes de todos en la actualidad.

Tales atributos hacen que los microcontroladores implementados con las tarjetas escudos sean dispositivos importantes de conocer y dominar, lo cual crea la necesidad de lograr un entendimiento más a fondo y obtener los conocimientos necesarios en la programación de estos dispositivos para el diseño de sistemas digitales/analógicos o analógicos/digitales con la adaptación a aplicaciones de educación y aprendizaje.

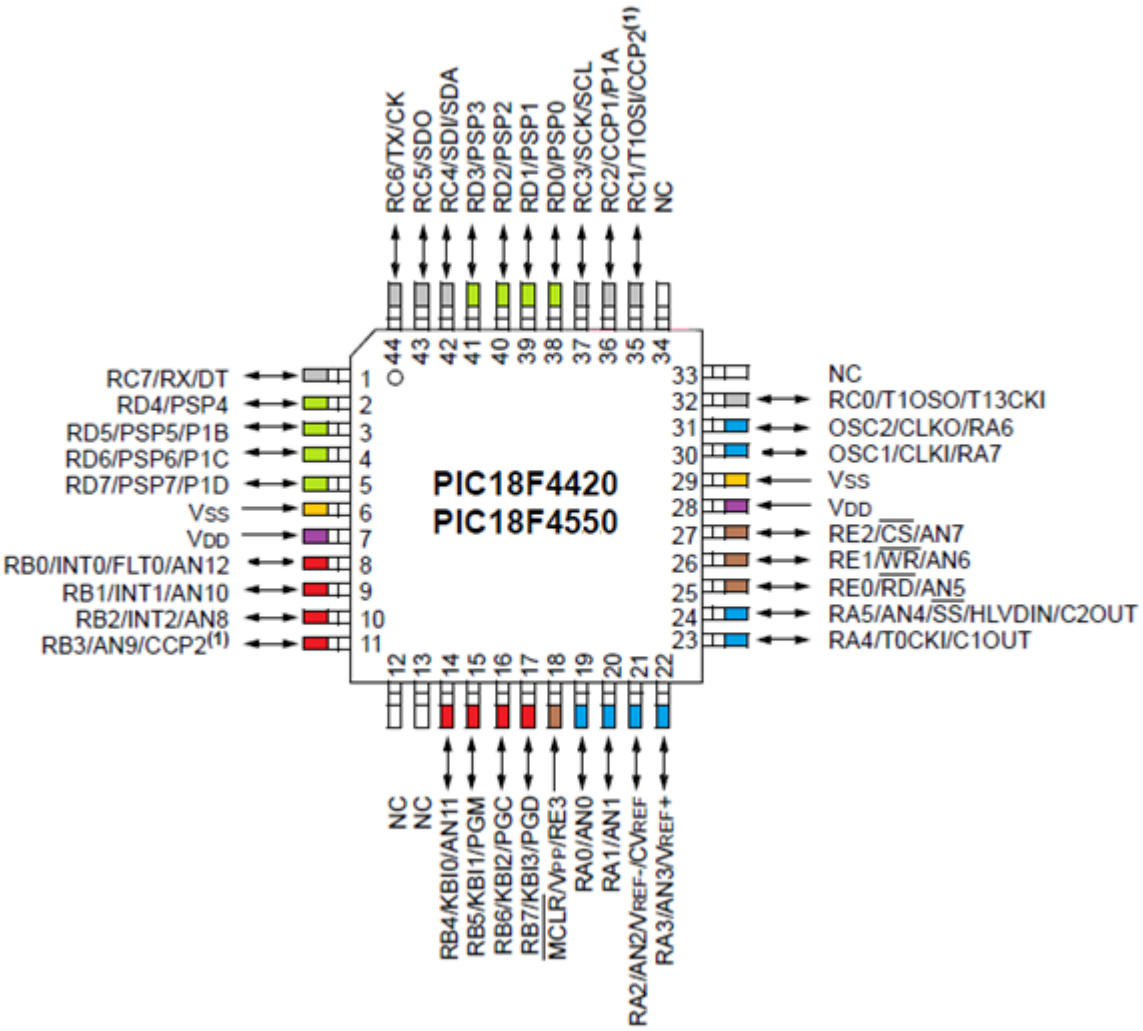
La adaptación de las tarjetas escudos para el funcionamiento con microcontroladores, apuntan siempre hacia la actualización tecnológica que hoy en día presenta una alta tendencia hacia la automatización y una gran variedad de aplicaciones que permiten entender las comunicaciones digitales y ampliar el conocimiento y el desarrollo de proyectos con microcontroladores por su gran versatilidad.

Por ello la implementación de tarjetas escudos que complementan una plataforma hardware con microcontroladores, ayudarán efectivamente a la consecución de objetivos tecnológicos y prácticos muy fundamentales en el aprendizaje del estudiante.

2.2 EL MICROCONTROLADOR PIC18F4550.-

J. Angulo (2010) "Los microcontroladores PIC existen en gamas de 8bit, 16bit y 32bit. Dentro de la gama más simple de 8bit se encuentra el microcontrolador PIC18F4550, el cual pertenece a la familia PIC18 MCU". Sus características de memoria de programa, memoria RAM, número de Entradas/Salidas, número de canales analógicos y tipos de puertos de comunicación, han hecho de este PIC uno de los más utilizados para diversas aplicaciones. En la figura 2.1 se muestra el esquema del PIC.

Figura 2.1: Distribución de patillas del PIC18F4550.



.

Fuente: Hoja de datos MICROCHIP-DS39631A.

Elaborado: Autor

2.2.1 CARACTERISTICAS DEL PIC18F4550.-

A continuación en la tabla 2.1 se detallan las principales características del PIC18F4550.

Tabla 2.1: Características del PIC18F4550.

	PIC18F4550
Frecuencia de operación	DC - 40MHZ
Memoria de programa (BYTES)	32768
Memoria del programa (INSTRUCCIONES)	16384
Memoria de datos (BYTES)	1536
Memoria EEPROM de datos (BYTES)	256
Fuentes de interrupción	20
Puertos de E/S	Ports A, B, C, D, E
Temporizadores	4
Módulos de captura (PWM)	1
Comunicaciones serie	MSSP, Enhanced USART
Comunicaciones paralelas	Si
Módulo A/D 10-BIT	13 Canales de entrada
Restablecimiento	POR, BOR, Reset, Stack Full, Stack Underflow (PWRT, OST), MCLR (opcional), WDT
Detector programable de alta/baja tensión	Si
Detector de restablecimiento de programación	Si
Conjunto de instrucciones	75 y 83 con set extendido
Paquete	40 pin / 44 pin QFN y TQFP

Fuente: Hoja de datos MICROCHIP.

Elaborado: Autor.

En su estructura interna soporta:

- Tecnología FLASH/EEPROM de baja potencia y alta velocidad.
- 1 conversor A/D de 10 bits.
- 3 sincronizadores de tiempo (Timers).
- 2 módulos para captura/comparación/PWM.
- Módulo Serial Maestro Sincrónico (MSSP) con dos modos de operación.
- Módulo USART direccionable, soporta RS485 y RS232.
- Detector de bajo voltaje programable.
- POR (Power On Reset), PWRT (Power Up Timer), OST (Oscillator Start-up Timer).
- WDR (Watchdog Timer) con oscilador RC incorporado y período programable desde 4 hasta 131 ms.
- Usa protección de código programable.
- Modo de ahorro de energía.
- Bajo consumo de potencia (menos de 1.6 mA con 5V y 4MHz).
- 3 fuentes externas de interrupción.
- 4 fuentes de interrupción por cambios de entradas.
- Acepta 4 tipos de osciladores de cristal (hasta 40MHz).
- Acepta 2 tipos de osciladores RC externos (hasta 4MHz).
- Acepta 2 tipos de relojes externos (hasta 40MHz).
- Amplio rango de voltaje de operación (2V a 5.5V).

2.2.1.1 RESISTENCIA DE LA MEMORIA.-

J. Angulo (2005) "Las celdas de memoria temporal (memoria flash) tanto para la memoria de programa y los datos EEPROM permiten ciclos de escritura hasta 100.000 para la memoria de programa y 1.000.000 para la memoria EEPROM".

2.2.2.2 AUTO-PROGRAMACION.-

A. Mazarredo (2007) "Estos dispositivos pueden escribir en sus propios espacios de memoria del programa en virtud del control de software". Mediante el uso de una rutina de gestor de arranque situado en el bloque de arranque, protegida en la parte superior de la memoria de programa, se hace posible crear una aplicación que permita actualizarse a sí mismo.

2.2.2.3 INSTRUCCIÓN DE SET EXTENDIDO.-

M. Etxebarria (2007) "La familia PIC18F4550 introduce una extensión opcional para el conjunto de instrucciones PIC18, que añade 8 nuevas instrucciones y un modo de direccionamiento indexado". Esta extensión, habilitada como opción en la configuración del dispositivo, ha sido diseñado específicamente para optimizar el código de aplicación reentrante desarrollado originalmente en lenguajes de alto nivel, tales como el C.

2.2.2.4 MÓDULO CCP.-

En el modo PWM, este módulo ofrece 1, 2 o 4 salidas moduladas para el control de medio puente y los controladores de puente completo. Otras características incluyen auto-apagado, para desactivar salidas PWM de las condiciones de selección de interrupción o de otro tipo y el pre-arranque automático, para reactivar las salidas una vez que la condición ha despejado.

2.2.2.3 USART DIRECCIONABLE.-

F. Valdés (2007) "Este módulo de comunicación serie es capaz de funcionar con el estándar RS-232 y proporciona soporte para el protocolo de bus LIN". Además, posee la detección automática de velocidad y una velocidad de

transmisión generador de 16 bits para una mejor resolución. Cuando el microcontrolador utiliza el bloque oscilador interno, el USART proporciona un funcionamiento estable para aplicaciones sin necesidad de utilizar un cristal externo (o la necesidad de añadir potencia de acompañamiento).

2.2.2.4 CONVERTIDOR A/D DE 10-BIT.-

A. Hambley (2001) "Este módulo incorpora el tiempo de adquisición programable, lo que permite un canal que se seleccionará y una conversión" para ser iniciado sin esperar un período de muestreo y, por tanto, reducir el código de inicio.

2.2.2.5 WATCHDOG (WDT).-

Esta versión mejorada incluye una de 16 bits, lo que permite un rango de tiempo de espera prolongado que es estable a través de la tensión de funcionamiento y la temperatura. Su función es contar cada cierto número de pulsos de reloj en un determinado tiempo, esperando algún evento generado por el programa, si no llega, el watchdog se activa y hace que todo empiece de nuevo y si llega el evento, entonces no hace nada.

2.3 INTRODUCCION A LAS TARJETAS ESCUDOS.-

Una tarjeta escudo es una placa impresa conformada por una variedad de componentes electrónicos, que se pueden conectar en la parte superior de una placa base o plataforma hardware basada en microcontroladores PIC para ampliar generalmente sus capacidades, con la opción ser conectadas una encima de otra.

Las tarjetas escudos suelen tener diseños bastante simples y en muchas ocasiones son de código abierto. Se pueden diseñar tarjetas escudos con los componentes electrónicos más básicos hasta con los más sofisticados. Se pueden usar para crear objetos interactivos, leyendo datos de una gran variedad de interruptores, sensores, y controlar multitud de tipos de luces, motores y otros actuadores físicos.

Las diferentes tarjetas escudos que existen siguen la misma filosofía que su plataforma hardware base, que es en donde van a ser conectadas.

A. Saravia (2010). "Al igual que las computadoras de escritorio tienen ranuras de expansión llamados zócalos, en los que se pueden agregar tarjetas de expansión de vídeo, sonido, red, y muchas otras cosas", las tarjetas escudos tienen encabezados de apilamientos o patillas, ya sea en la parte superior o inferior, donde se puede adaptar o conectar más tarjetas para incrementar las funcionalidades.

La gran diferencia, sin embargo, A. Coria (2010) es que "las ranuras de expansión de un ordenador son finitas e independientes", esto quiere decir si el equipo tiene 6 ranuras, se puede poner en 6 tarjetas. Y debido a que (con algunas excepciones), cada tarjeta es totalmente independiente, no importa lo que se pueda utilizar. Podrían ser de 6 tarjetas de vídeo, o 1 tarjeta de video y 5 tarjetas de red, o lo que sea. El equipo considera cada tarjeta de forma aislada e independiente, de manera que no entren en conflicto entre sí.

En cambio, con las tarjetas escudos no existe un número específico de zócalos (slots), y las tarjetas se pueden apilar una encima de la otra para combinar sus características y funcionalidades. La mayoría de las veces sólo se tendrá un escudo a la vez montado en la plataforma hardware base, pero dependiendo de la aplicación a realizar se pueden montar más de una a la

vez y para ello debe de existir una estructura de compatibilidad para que puedan trabajar juntos.

2.4 ESTRUCTURA DE UNA TARJETA ESCUDO.-

La estructura de una tarjeta escudo se basa en los siguientes parámetros que son:

2.4.1 COMPONENTES.-

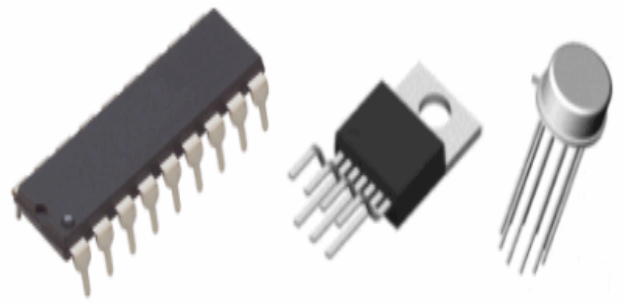
Las tarjetas escudos pueden tener en su estructura un sin número de encapsulados de componentes electrónicos para un mismo dispositivo. Básicamente existen 3 tipos de familias de encapsulados electrónicos, los cuales son:

2.4.1.1 ENCAPSULADOS DE TECNOLOGÍA DE AGUJEROS PASANTES (THT).-

A. Curtidor (2012) "Son todos aquellos componentes que poseen pequeñas patillas para ser instalados en perforaciones metalizadas (llamadas Thru-Hole)". Este tipo de componentes se sueldan por la capa opuesta de la tarjeta escudo. Generalmente son montados por un solo lado de la tarjeta y soldados del otro lado.

A continuación en la figura 2.2 se muestran algunos componentes THT.

Figura 2.2: Ejemplos de componentes de Tecnología de Agujeros Pasantes.

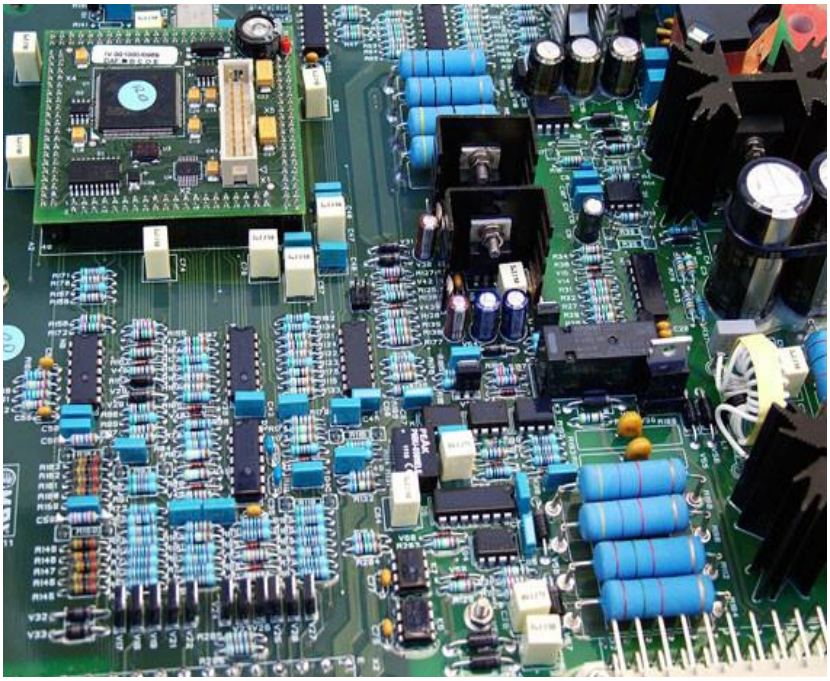


Fuente: www.tme.eu

Elaborado: www.tme.eu

En la figura 2.3 se puede observar los componentes de tecnología de agujeros pasantes montados en una tarjeta escudo.

Figura 2.3: Componentes THT en una tarjeta escudo.



Fuente: www.prodisa.es

Elaborado: www.prodisa.es

VENTAJAS.-

- Proporcionan uniones mecánicas fuertes, en comparación con las técnicas de los componentes de montaje superficial (SMT).

- Son económicos y asequibles.

- Son componentes que tienen mayores resistencias térmicas entre el interior y el ambiente.

DESVENTAJAS.-

- Limitan el área de encaminamiento disponible para las trazas de señal entre los demás componentes electrónicos.

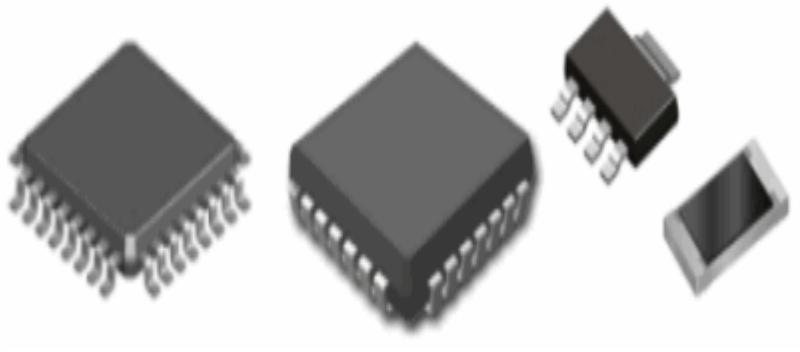
- La perforación adicional requerida hace que las tarjetas escudos sean más caras de producir.

2.4.1.2 ENCAPSULADOS DE MONTAJE SUPERFICIAL (SMD/SMT).-

C. Herrera (2012) "Son todos aquellos componentes que se montan en forma superficial, es decir sin necesidad de hacer agujeros en la base" de la tarjeta. Tienen la ventaja de que pueden montarse por ambos lados de la tarjeta escudo, además de ser más pequeños que los Thru-Hole, lo que permite hacer circuitos más pequeños y más densos.

Se los utiliza en la mayoría de las veces para diseños en alta frecuencia debido a su pequeño tamaño, en la figura 2.4 se muestra algunos tipos de componentes SMD/SMT.

Figura 2.4: Ejemplos de componentes de Montaje Superficial.



Fuente: www.tme.eu

Elaborado: www.tme.eu

La figura 2.5 muestra los componentes SMT/SMD montados en una tarjeta escudo.

Figura 2.5: Componentes de Montaje Superficial en una tarjeta escudo.

Fuente: www.asembli.com

Elaborado: www.asembli.com

VENTAJAS.

- Son componentes electrónicos muy pequeños.

- Proporcionan un menor costo y menor tiempo de producción para cada tarjeta escudo.

- Mejor comportamiento mecánico y mayor resistencia a vibraciones y golpes.

- Son más fáciles de ensamblar en la tarjeta.

- Generan menos cantidad de agujeros en la base de la tarjeta.

- Ahorran espacio y longitud en las pistas de cobre, porque se puede hacer que las tarjetas sean más pequeñas, y así reducir la longitud de las pistas.

- Son componentes electrónicos que están preparados para las últimas tecnologías.

- Minimizan la inductancia y la capacitancia parásita de cables conductores, que pueden menoscabar la función del circuito.

DESVENTAJAS.-

- El reducido tamaño, implica que la superficie de disipación también es menor, y normalmente la resistencia térmica entre el interior del componente y el exterior es más grande.

- Al ser sus conexiones más pequeñas, requieren de técnicas especializadas para su ensamblaje en la tarjeta.

- La reparación de este tipo de componentes electrónicos es más difícil y costosa.

- No están diseñados para soportar altas potencias o voltajes, por lo que en ciertas ocasiones se combinan con los componentes Thru-Hole.

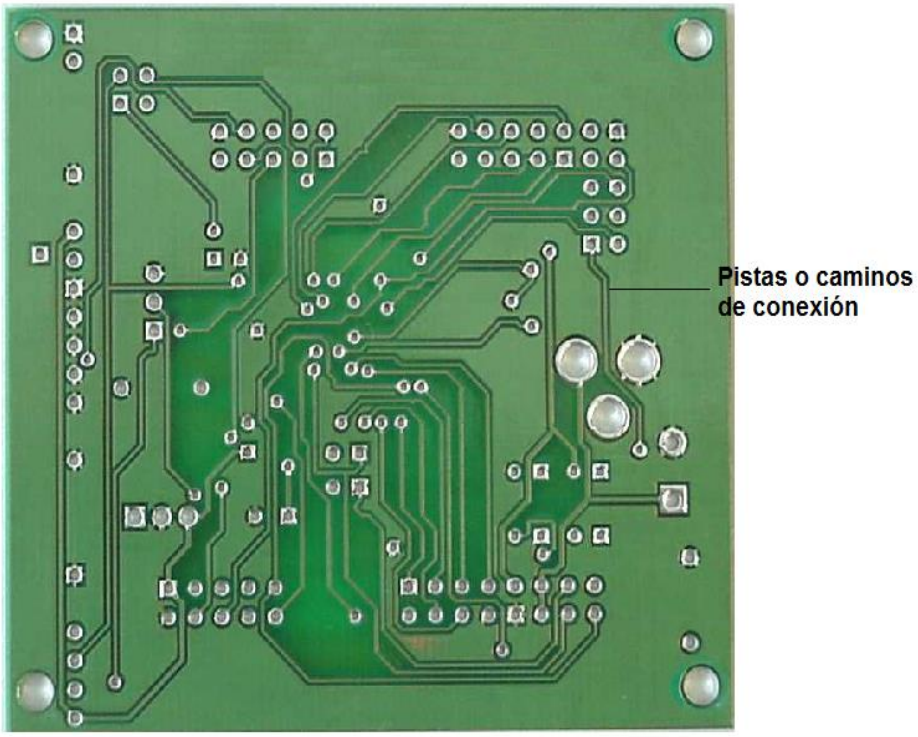
- Requieren un mayor control de la temperatura al ser ensamblados en la placa de la tarjeta escudo.

2.4.2 PISTAS O CAMINOS DE UN CIRCUITO IMPRESO.-

M. Sierra (2003) "Son líneas continuas y laminadas que están constituidas por un material conductor, para poder conectar eléctricamente a los componentes de una tarjeta escudo y sostenerlos mecánicamente en su superficie".

Los caminos generalmente son de cobre por ser un excelente conductor, y van impresos sobre una base que se fabrica con resinas de fibra de vidrio reforzada, como se muestra en la figura 2.6.

Figura 2.6: Caminos de circuito impreso en una tarjeta escudo.



Fuente: www.dynamoelectronics.com

Elaborado: Autor.

2.4.2.1 CAPAS EN UN CIRCUITO IMPRESO (PCB).-

J. Angulo (2009) "Los circuitos impresos o PCB pueden fabricarse y diseñarse de varias capas (layers) conductoras" para la comunicación entre los componentes electrónicos que lo conforman, en un software llamado EAGLE.

En la siguiente tabla 2.2 se muestra el orden de capas en un circuito impreso.

Tabla 2.2: Número de capas en un PCB.

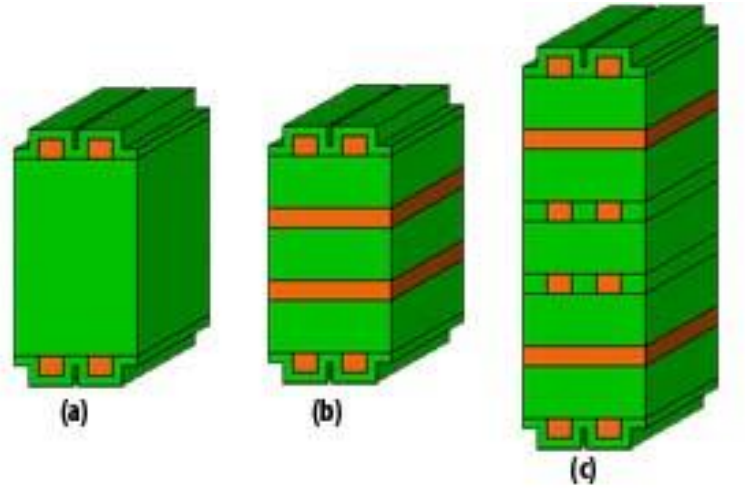
CAPAS	Descripción Capa CAD
1	serigrafía superior
2	máscara de soldado superior
3	Botón de máscara
4	Capa 1
5	Sustrato
6	Capa 2
...	...
n-1	Sustrato
n	Capa n (conductora)
n+1	Botón de máscara
n+2	máscara de soldado inferior
n+3	serigrafía inferior

Fuente: www.pcb.electrosoft.cl

Elaborado: Autor.

En la figura 2.7 se pueden apreciar 3 capas. Lo que se destaca de color naranja corresponden a las capas de cada uno de los conductores. El espesor del circuito impreso puede variar dependiendo de la aplicación, sin embargo la más utilizada es 1.6 [mm].

Figura 2.7: Ejemplos de 2 capas (a), 4 capas (b) y 6 capas (c).



Fuente: www.pcb.electrosoft.cl

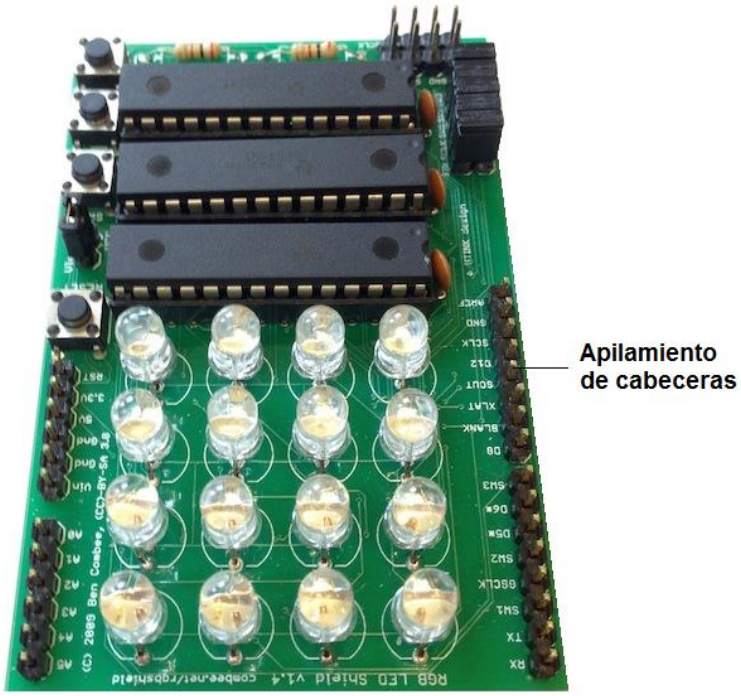
Elaborado: www.pcb.electrosoft.cl

2.4.3 APILAMIENTOS DE CABECERAS.-

H. Singh (2008) "Todas las tarjetas deben de tener apilamientos de cabeceras". Muchos escudos están equipados con encabezados regulares separatistas que sobresalen por debajo de la tarjeta, pero no proporcionan un lugar en la parte superior donde otro escudo puede ser conectado.

Por eso esta característica se debe utilizar siempre en la estructura de diseño de una tarjeta escudo, como se muestra en la figura 2.8.

Figura 2.8: Apilamientos de cabeceras de una tarjeta escudo.



Fuente: www.dynamoelectronics.com

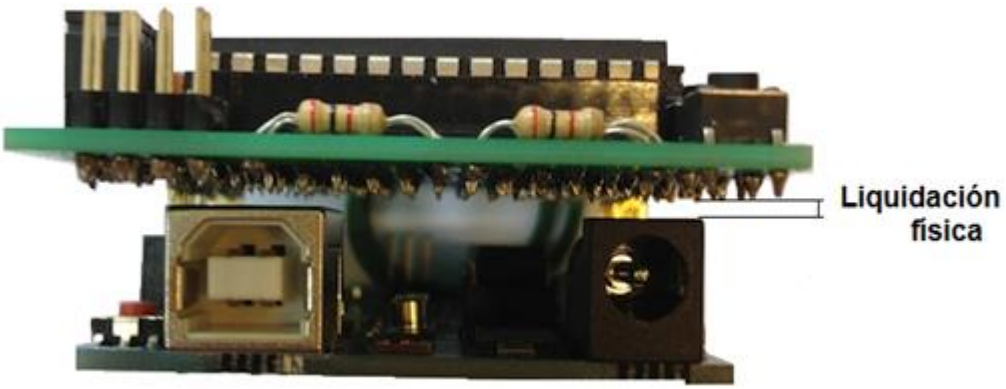
Elaborado: Autor

.

2.4.4 LIQUIDACION FÍSICA.-

La liquidación física se basa en la conexión total e íntegra de cada apilamiento de cabecera sin interrumpir otro tipo de conexión dentro de la misma tarjeta escudo, o el roce con los componentes de otra tarjeta escudo, como se muestra en la figura 2.9

Figura 2.9: Liquidación física entre dos tarjetas escudos.



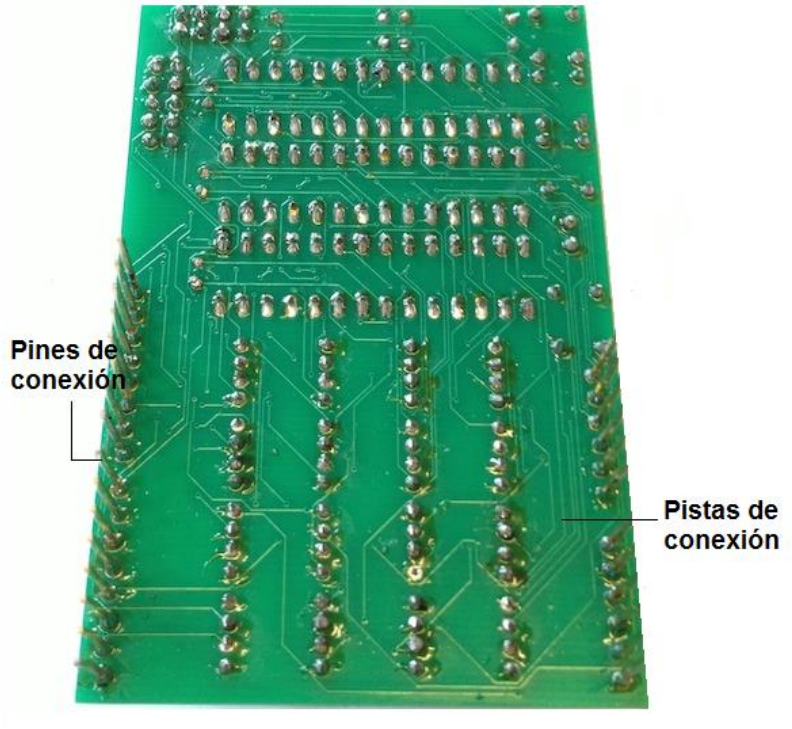
Fuente: www.impronic.com
Elaborado: Autor.

Incluso si las tarjetas escudos han apilado encabezados, todavía pueden tener partes físicamente altas y pueden interferir a otra tarjeta colocada en la parte superior; lo que impediría una liquidación física correcta.

2.4.5 CONTENCION DE PATILLAS.-

Es comprobar la asignación de patillas de la tarjeta escudo que se desea utilizar, y asegurar de que estas tarjetas no van a estar conectadas por las mismas patillas y causar algún tipo interrupción, como se muestra en la figura 2.10.

Figura 2.10: Patillas que contienen sus respectivas pistas de conexión.



Fuente: www.dynamoelectronics.com

Elaborado: Autor.

T. Wilmshurst (2007) "Solo se pueden compartir las patillas que contengan una tensión o voltaje de referencia"; ya que las tarjetas escudos no pueden compartir las mismas patillas de datos, porque no tendrían comunicación con el microcontrolador.

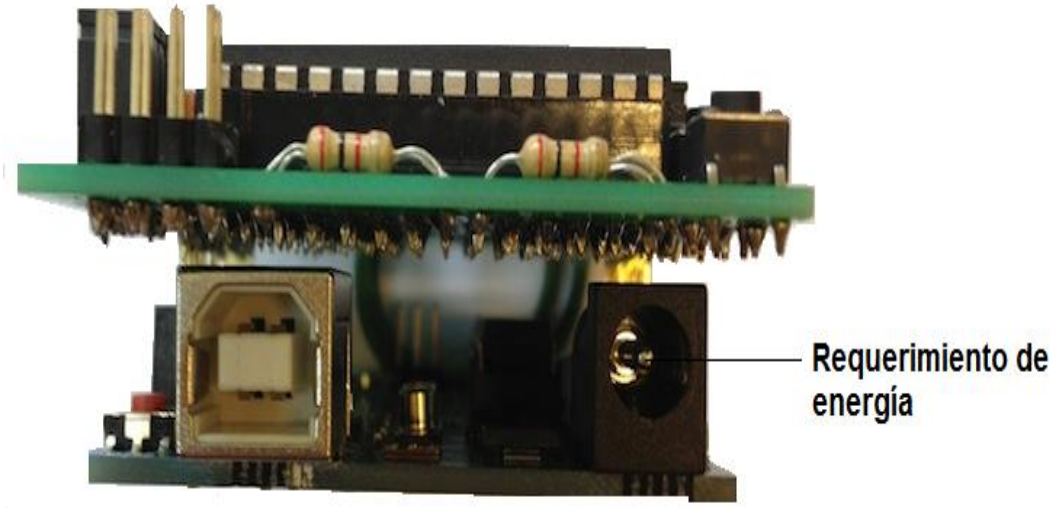
2.4.6 INTERACCION SOFTWARE.-

G. Tojeiro (2009) "Una tarjeta con microcontroladores debe tener una buena cantidad de sobrecarga de software", como bibliotecas de gran tamaño. A veces se puede tornar un poco complicado el combinar tarjetas escudos, solo si se utiliza un número alto de memorias; ya que se genera conflictos de interrupciones o requisitos de tiempo ajustados.

2.4.7 REQUERIMIENTO DE ENERGÍA.-

Las tarjetas escudos J. González (2001) "requieren de una cantidad de energía mínima o máxima de alimentación", que se puede disponer desde la plataforma hardware base para su correcto funcionamiento, y en caso de no obtenerla desde la plataforma se la debe alimentar externamente a través de conectores como el USB o de fuentes de voltaje, como se muestra en la figura 2.11.

Figura 2.11: Requerimiento de energía externo de una tarjeta escudo.



Fuente: www.impronic.com
Elaborado: Autor.

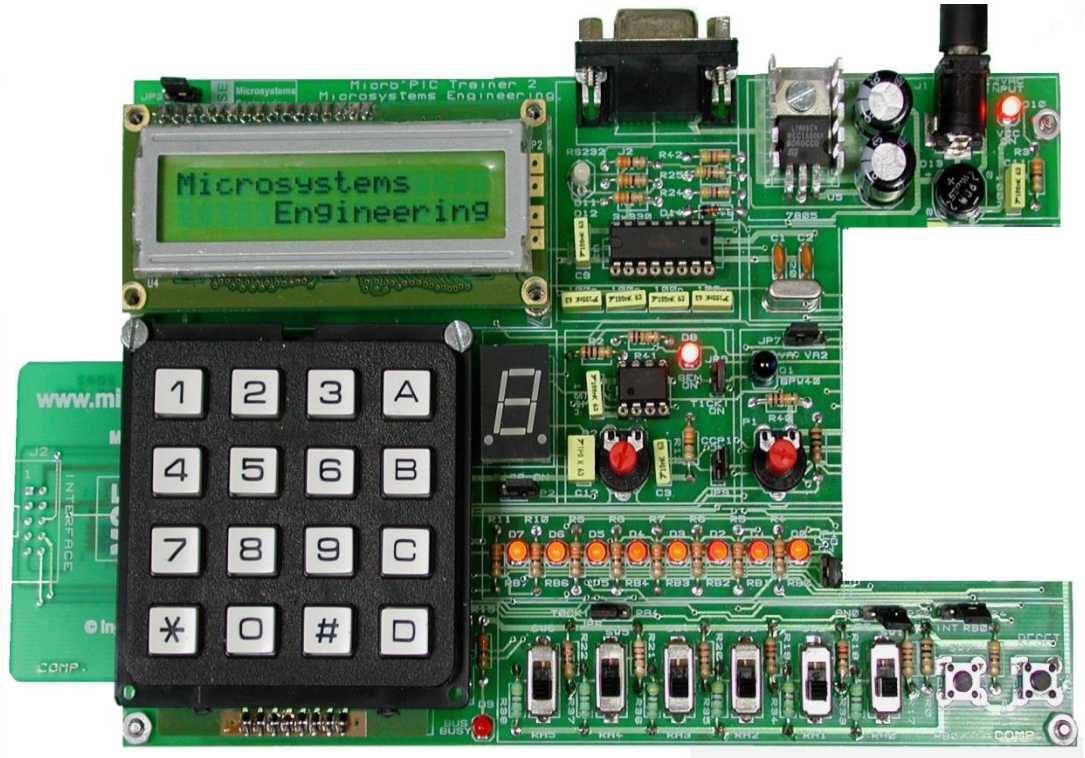
2.5 ARQUITECTURA DE DISEÑO.-

Según H. Martínez "Existen distintas arquitecturas de diseño para una tarjeta escudo que se pueden clasificar según tres modelos":

2.5.1 DISEÑO MONO-BLOQUE.-

H. Martínez (2012) "Está formado por una tarjeta electrónica en donde se encuentran todos los componentes y todas las conexiones ya establecidas entre ellos y el microcontrolador", como se muestra a continuación en la figura 2.12.

Figura 2.12: Micro-Entrenador con Diseño Mono-Bloque.



Fuente: www.mycrosystem.com

Elaborado: Autor.

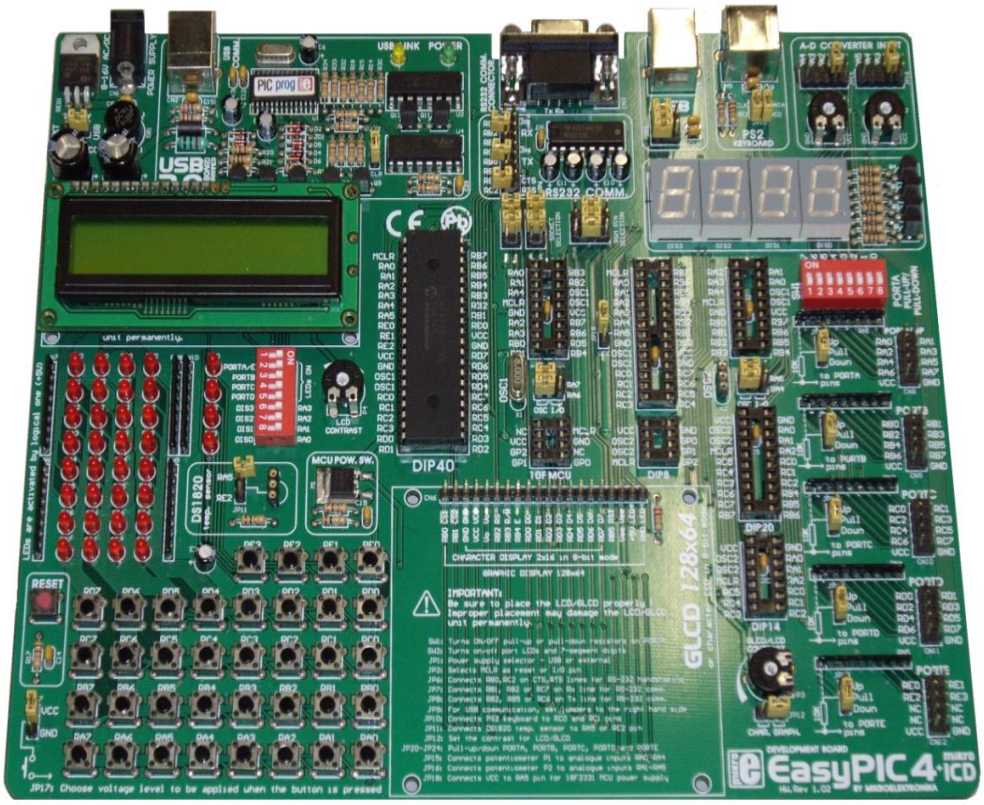
2.5.2 DISEÑO MONO-BLOQUE SIN CONEXIONES.-

P. Fernández (2012) "Está formado por una tarjeta electrónica en donde se encuentran todos los componentes y sus circuitos auxiliares, pero con la única diferencia que no están realizadas las conexiones con el

microcontrolador". Estas tarjetas pueden incluir un módulo base de conexión rápida, para facilitar las conexiones de diferentes bloques.

A continuación en la figura 2.13 se muestra una tarjeta con diseño mono-bloque sin conexiones.

Figura 2.13: Micro-Entrenador con Diseño Mono-Bloque sin conexiones.



Fuente: www.arvc.umh.es

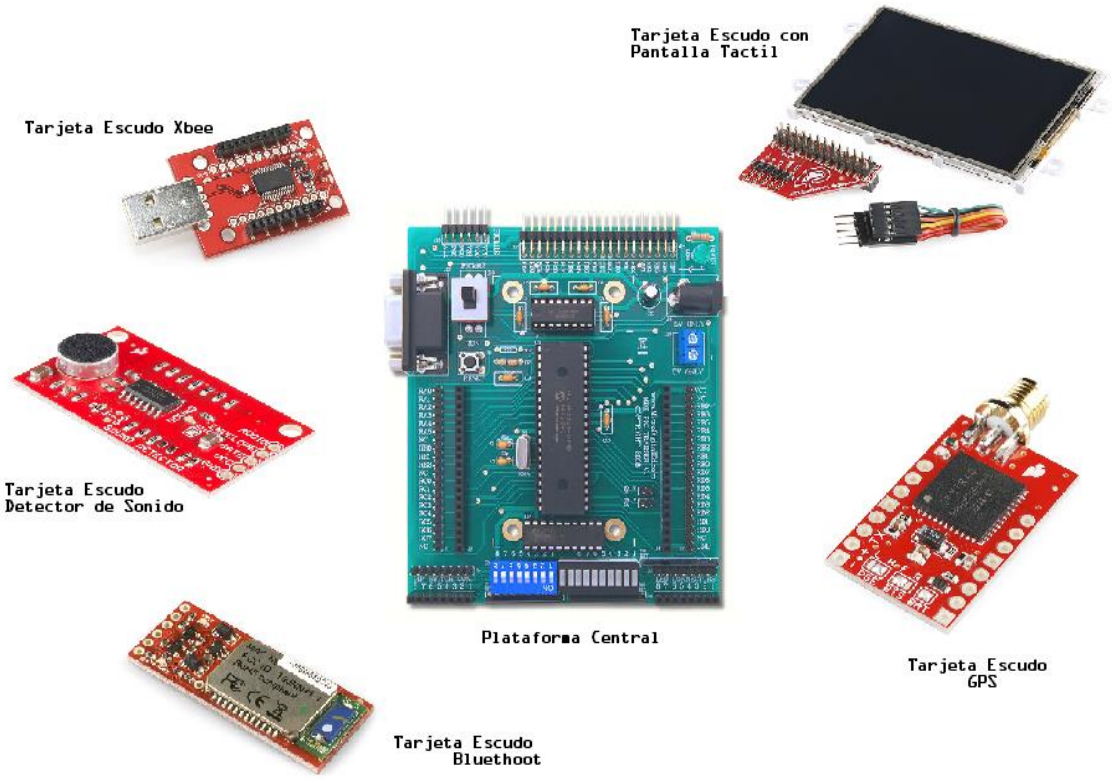
Elaborado: www.arvc.umh.es.

2.5.3 DISEÑO MODULAR.-

H. Martínez (2012) "Se compone de una tarjeta central en donde está el microcontrolador y todos los circuitos auxiliares, además de una serie de

conectores que permiten extraer absolutamente todos los puertos del microcontrolador" y conectarse a diferentes módulos o tarjetas escudos que amplían su capacidad funcional, como se muestra en la figura 2.14.

Figura 2.14: Diseño Modular con diferentes bloques de conexiones.



Fuente: www.sparkfun.com

Elaborado: Autor.

2.6 VENTAJAS DE UNA TARJETA ESCUDO.-

Algunas de las ventajas que presentan son las siguientes:

- **ASEQUIBLES.-** Los componentes electrónicos para fabricar y desarrollar una tarjeta escudo son de fácil adquisición, o se pueden adquirir tarjetas escudos ya hechas por fabricantes.

- **MULTIPLATAFORMA.-** El programa (software) que se usa para la programación de las tarjetas puede tener interacción con sistemas operativos como Windows, Macintosh y Linux. Cabe recalcar que la mayoría de los entornos para microcontroladores existentes están limitados a Windows.

- **PROGRAMACION SIMPLE Y DIRECTA.-** La programación es fácil de usar para estudiantes principiantes y lo suficientemente flexible para estudiantes avanzados, por la utilización del lenguaje C en el compilador CCS para la programación respectiva de cada módulo de las tarjetas escudos.

- **SOFTWARE AMPLIABLE.-** La programación para las tarjetas escudos puede ampliarse a través de librerías en lenguaje C del compilador CCS, y así añadir directamente los códigos de programación a las aplicaciones que se desean realizar.

- **HARDWARE ESCALABLE.-** Las tarjetas escudo están basadas en el microcontrolador PIC18F4550, lo cual facilita la ampliación, optimización, construcción y diseño de las aplicaciones para los módulos que se puedan desarrollar.

2.7 DESVENTAJAS DE UNA TARJETA ESCUDO.-

La única desventaja que pueden presentar las tarjetas escudos es que son implementaciones muy pequeñas y con un mínimo descuido las podemos hacer caer o aplastarlas y dañarlas.

2.8 TARJETAS ESCUDOS COMO PLATAFORMAS DOCENTES PARA LA ENSEÑANZA Y APRENDIZAJE DE MICROCONTROLADORES.-

Las tarjetas escudos para microcontroladores son plataformas electrónicas que permiten comprobar funcionalmente los programas diseñados previamente de una forma teórica en clase, con un método mucho más directo y confiable tanto para el docente como para el alumno.

En efecto, los docentes y alumnos podrán diseñar sus programas probándolos directamente a la tarjeta escudo que estará montada en una plataforma de hardware base que contiene al microcontrolador PIC18F4550.

En el momento de diseñar una tarjeta escudo como plataforma de aprendizaje para estudiantes, lo primero que se debe hacer es observar las que ya existen actualmente, para poder adaptarlas o ampliarlas para satisfacer los requisitos académicos específicos.

En fin, con las tarjetas escudos el ambiente académico podrá:

- Disponer de un equipo flexible, capaz de soportar la mayoría de los proyectos y montajes que se requieren, ya que contienen la mayoría de los recursos que se usan normalmente. Además, facilita los montajes rápidos y seguros.

- Servir de plataforma de aprendizaje de las tecnologías clásicas y modernas, que aparecen continuamente en la Electrónica y en los Microcontroladores.

2.9 COMPILADORPIC PIC-C CCS.-

G. Breijo (2008) "El compilador PIC-C CCS, denominado PCWH, utiliza un lenguaje de códigos muy eficiente, fácil y manejable, pero sobre todo de alto nivel, basado en C ANSI que contiene las funciones y librerías necesarias para el diseño de cualquier aplicación basada en microcontroladores" PIC: matemáticas, control de protocolos serial, I2C, etc. Además, suministra los controladores (drivers) para diversos dispositivos como LCD, convertidores AD, relojes en tiempo real, EEPROM serie, etc.

Este compilador soporta dispositivos de todas las familias de los microcontroladores PIC y en su última versión (5.025) se incluyen librerías para poder controlar más de 300 modelos de microcontroladores PIC.

El compilador incluye funciones para acceder al hardware de los procesadores PIC, tal como ***read_adc ()*** para leer el valor de un conversor A/D. La entrada y salida de la comunicación se maneja describiendo las características de los puertos en un PRAGMA. Funciones tales como ***input ()*** y ***output_high ()*** mantienen apropiadamente los registros tri-Estado. Las variables, incluyendo estructuras pueden ser directamente mapeadas a la memoria tal como los puertos de entrada y salida para representar mejor la estructura del Hardware en C.

La velocidad del reloj del microcontrolador se puede especificar en un PRAGMA para permitir que las funciones incorporadas retrasen un número dado de microsegundos o milisegundos. Las funciones de E/S serie permiten que funciones estándar como ***get ()*** y ***printf ()*** sean usadas para RS-232.

El transceptor serie del Hardware se usa en las partes que aplican cuando es posible. Para otros casos el compilador genera un transceptor serie de software. Los operadores estándar de C y las funciones estándar

incorporadas se optimizan para producir código muy eficiente para funciones de bits y de E/S.

Durante el proceso de enlazado analiza la estructura del programa. Las funciones que se llaman unas a otras con frecuencia se agrupan juntas en el mismo segmento de página. La herramienta transparente al usuario maneja funciones a través de las páginas automáticamente. Las funciones se pueden implementar in-line o separadas. Según H. Vallejo (1998) "la RAM se reserva eficientemente para determinar cuántas ubicaciones pueden ser reutilizadas". Las cadenas constantes y tablas se almacenan en la ROM del dispositivo.

El compilador produce principalmente tres tipos de archivos:

- Archivos con extensión ***.hex*** que permitirá grabar el programa ejecutable en el PIC por medio del uso de un programador.

- El archivo ***.asm*** contendrá un listado en ensamblador del programa compilado con la información del mapeo de memoria. Estos archivos son muy útiles para el *debugging* de los programas y para determinar la cantidad de pasos de programas (ciclos de ejecución) contiene la aplicación.

- Los archivos con extensiones ***.pre*** contienen la información procesada del programa, ***#defines, #includes***, etc. La cual es expandida y guardada en el archivo.

O. Zapata (2001) "La salida en ***.hex*** y los archivos de depuración son seleccionables y compatibles con emuladores y programadores populares incluyendo MPLAB IDE para depuración a nivel de fuente". PCW incluye un poderoso IDE bajo Windows.

2.9.1 PRINCIPALES VENTAJAS.-

Según G. Breijo (2008) "Las principales ventajas son":

- ✓ Está basado en el ANSI C.
- ✓ Soporte completo de las familias de Microcontroladores PIC.
- ✓ Salida Assembly.
- ✓ Biblioteca de funciones pre-compiladas y directivas de las que disponen.
- ✓ Funciones listas para usarse, ahorra mucho trabajo al programador.
- ✓ Industria estándar INTEL Hex 8 bit Mergerd format (INHX8M).
- ✓ Soporta interrupciones internas y externas.
- ✓ Tipos de datos 8, 6, 16 bit int, char, long, punteros, unsigned, etc.
- ✓ Inserción de código asamblea asm().
- ✓ Operadores aritméticos, incluyendo multiplicación, división, modulo y otros.
- ✓ Las variables y funciones no utilizadas son borradas automáticamente.
- ✓ Reutilización de RAM.
- ✓ Instrucciones simples.
- ✓ Se tiene un aprovechamiento eficiente de los recursos del PIC.

2.10 PROGRAMACION EN PIC-C CCS.-

M. Bates (2006) "La programación en lenguaje C, para PIC, es bastante similar en su estructura y sintaxis a la programación tradicional para ordenadores, una de las diferencias está en las librerías creadas

específicamente para los microcontroladores PIC" como por ejemplo para pantallas, teclados matriciales, bus I2C, etc.

La función ***main ()*** es imprescindible en todo programa escrito en lenguaje C, pues es la función principal, desde aquí se puede hacer el llamado a otras funciones.

Para crear un programa en lenguaje C, hay que seguir los pasos siguientes:

- Especificaciones del programa.
- Hacer organigrama.
- Escribir el código fuente.
- Compilar + enlazar.
- Depurar errores, si es que los hay.

2.10.1 ESTRUCTURA DE UN PROGRAMA EN CCS.-

Para escribir un programa en C con el CCS, según G. Breijo (2008) "se deben tener en cuenta una serie de elementos básicos de su estructura", que se detallan a continuación y se muestran en la figura 2.15.

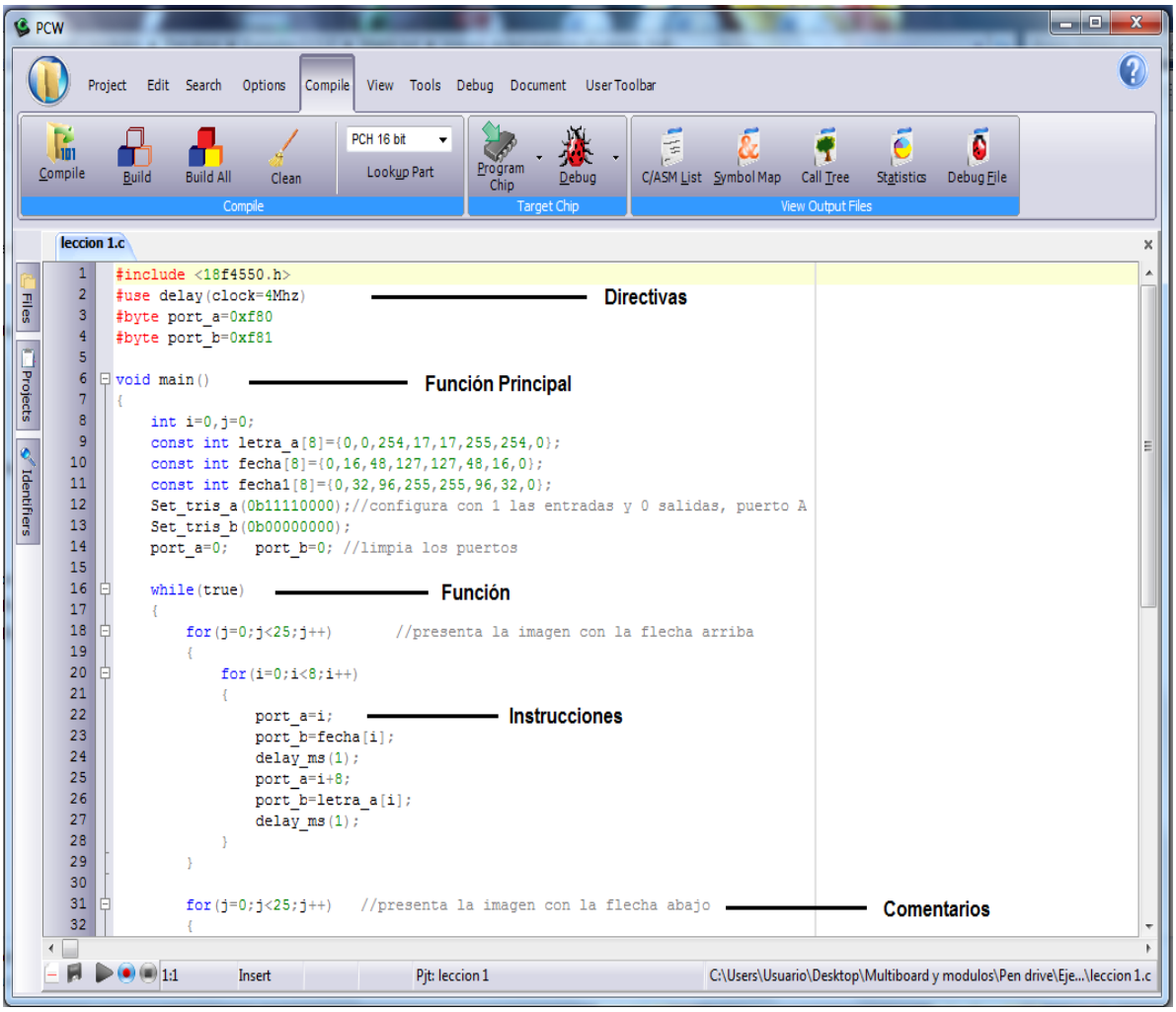
- **DIRECTIVAS DE PROCESADO.-** Controlan la conversión del programa a código maquina por parte del compilador.

- **PROGRAMAS O FUNCIONES.-** Conjunto de instrucciones. Puede haber uno o varios; en cualquier caso siempre debe haber uno definido como principal mediante la inclusión de la llamada ***main ()***.

- **INSTRUCCIONES.-** Indican cómo se debe comportar el PIC en todo momento.

- **COMENTARIOS.-** Permiten describir lo que significa cada línea del programa.

Figura 2.15: Estructura básica de un programa en CCS.



Fuente: Compilador CCS.
Elaborado: Autor.

2.10.2 TIPOS DE DATOS.-

G. Breijo (2008) "CCS C acepta los tipos de datos que se detallan en las siguientes tablas".

Tabla 2.3: Tipos de Datos.

TIPO	TAMAÑO	RANGO	DESCRIPCION
Int1 Short	1 bit	0 a 1	Entero de 1 bit
Int Int8	8 bit	0 a 255	Entero
Int16 Long	16 bit	0 a 65.535	Entero de 16 bit
Int 32	32 bit	0 a 4.294.967.295	Entero de 32 bit
Float	32 bit	±1.175x10^-38 a ±3.402x10^+38	Coma flotante
Char	8 bit	0 a 255	Carácter
Void	-	-	Sin valor
Signed Int8	8 bit	-128 +127	Entero con signo
Signed Int16	16 bit	-32768 a 32767	Entero largo con signo
Signed Int 32	32 bit	-2^31 a +(2^31 + 1)	Entero 32 bit con signo

Fuente: Libro Compilador CCS C – G. Breijo (2008).
Elaborado: Autor.

2.10.3 LAS CONSTANTES.-

G. Breijo (2008) "Las constantes se pueden especificar en decimal, octal, hexadecimal o en binario", como se detalla en la tabla 2.4.

Tabla 2.4: Tipos de Constantes en CCS C.

123	Decimal
0123	Octal (0)
0x123	Hexadecimal (0x)
0b010010	Binario (0b)
´/x´	Carácter
´/010´	Carácter octal
´/xA5´	Carácter hexadecimal

Fuente: Libro Compilador CCS C – G. Breijo (2008).
Elaborado: Autor.

Además, se pueden definir constantes con un sufijo, como se detalla en la tabla 2.5.

Tabla 2.5: Definición de Constantes con un sufijo en CCS C.

Int8	127U
Long	80UL
Signed INT16	80L
Float	3.14F
Char	Con comillas simples ´C´

Fuente: Libro Compilador CCS C – G. Breijo (2008).
Elaborado: Autor.

También se definen caracteres especiales, algunos como se muestran en la tabla 2.6.

Tabla 2.6: Caracteres Especiales en CCS C.

/n	Cambio de línea.
/r	Retorno de carro.
/T	Tabulación.
/b	Backspace.

Fuente: Libro Compilador CCS C – G. Breijo (2008)
Elaborado: Autor.

2.9.4 VARIABLES.-

G. Breijo (2008) "Las variables se utilizan para nombrar posiciones de memoria RAM; se deben declarar obligatoriamente, antes de ser utilizadas"; para eso se debe indicar el nombre y el tipo de dato que se utilizará. Se definen de la siguiente forma:

Tipo Nombre_Variable = [Valor Inicial]

Tipo: Hace referencia a cualquiera de los tipos de datos visto en el punto 2.9.2.

Nombre_Variable: Puede ser cualquiera y el valor inicial es opcional; como se muestra en el siguiente ejemplo:

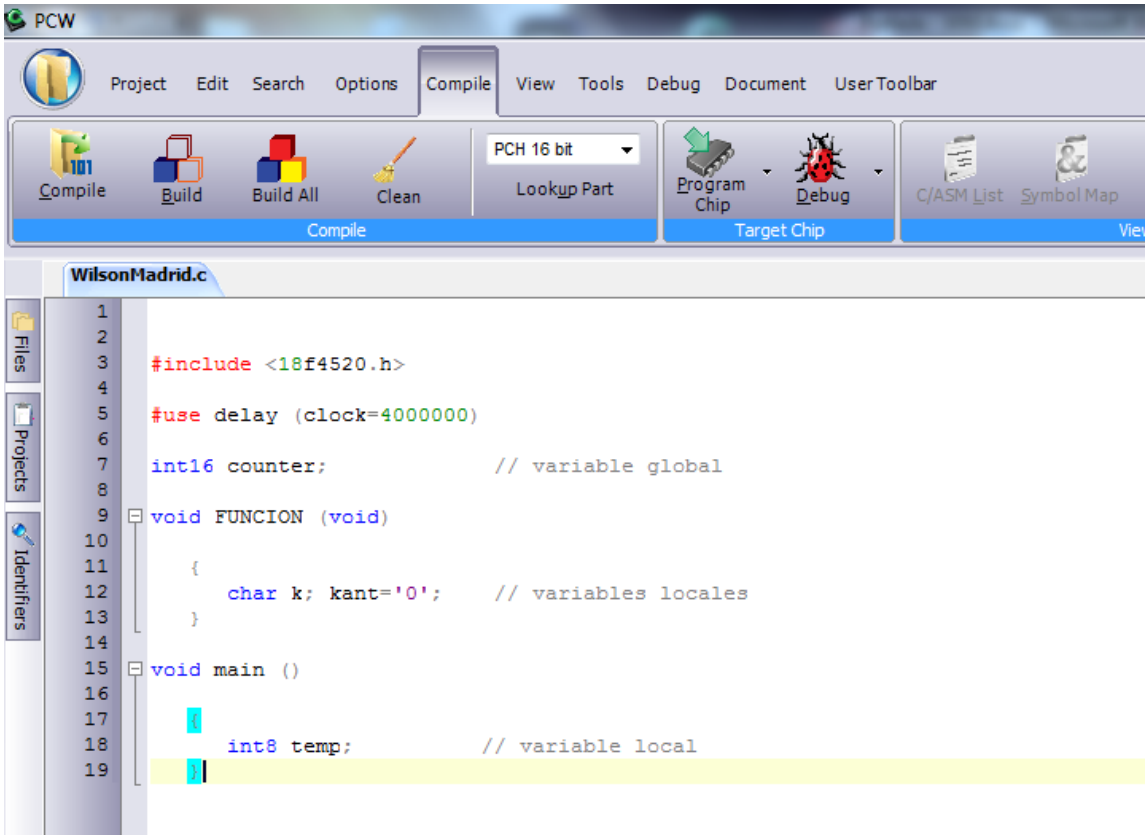
Float temp_limit=500.0;

Las variables definidas en un programa pueden ser de tipo LOCAL o GLOBAL. Las variables locales sólo se utilizan en la función donde se encuentran declaradas; las variables globales se pueden utilizar en todas las funciones del programa. Ambas deben declararse antes de ser utilizadas y

las globales deben declararse antes de cualquier función y fuera de ellas. Las variables globales son puestas a cero cuando se inicia la función principal ***main ().***

En la figura 2.16 se muestra el entorno de trabajo del programa CCS y algunas variables que se pueden utilizar.

Figura 2.16: Ejemplo de variables en CCS C.



Fuente: Software CCS.

Elaborado: Autor.

Las variables pueden ser definidas con:

- **AUTO.-** Es usada por defecto, no hace falta que se indique; donde la variable existe mientras la función esta activa. Estas variables no se inicializan a cero. Su valor se pierde cuando se sale de la función.
- **STATIC.-** Una variable local se activa como global, se inicializa a cero y mantiene su valor al entrar y salir de la función.

- **EXTERN.-** Permite el uso de variables en compilaciones múltiples.

2.11 EAGLE: SOFTWARE PARA DISEÑOS DE PCB´S.-

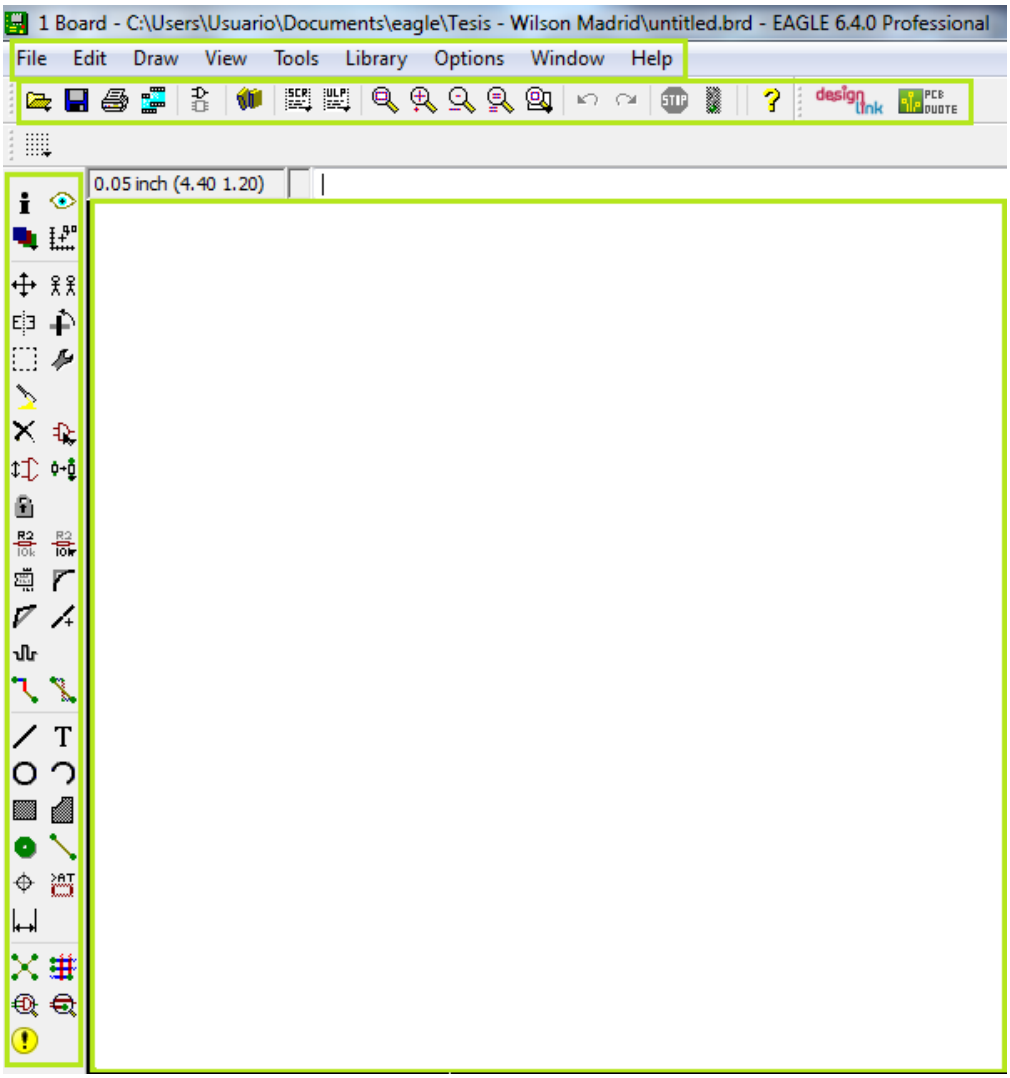
Todas las tarjetas escudos, incluida la plataforma hardware que contiene al microcontrolador están diseñadas en un software de dibujo asistido llamado Eagle PCB. Usando este programa se puede crear circuitos impresos partiendo de un esquemático previamente diseñado, utilizando herramientas de fácil uso.

2.11.1 INTERFAZ DE USUARIO DE EAGLE.-

M. Guadilla (2000) "Eagle se ha diseñado de tal modo que cualquier acción se realiza mediante un comando". Normalmente el usuario selecciona los comandos de la barra de herramientas que se encuentra a la izquierda de la ventana de trabajo, haciendo clic sobre los iconos. Pero existe otra opción, que es introducir los comandos mediante texto en la barra superior de la ventana de trabajo.

La figura 2.17 contiene la ventana de trabajo del editor de PCB. Como se puede apreciar de arriba hacia abajo, se tiene la barra de menú, barra de herramientas activas, barra de parámetros y coordenada de pantalla con línea de comando. A la izquierda se encuentra la barra de herramientas de comando.

Figura 2.17: Ventana de trabajo del editor de PCB de EAGLE.

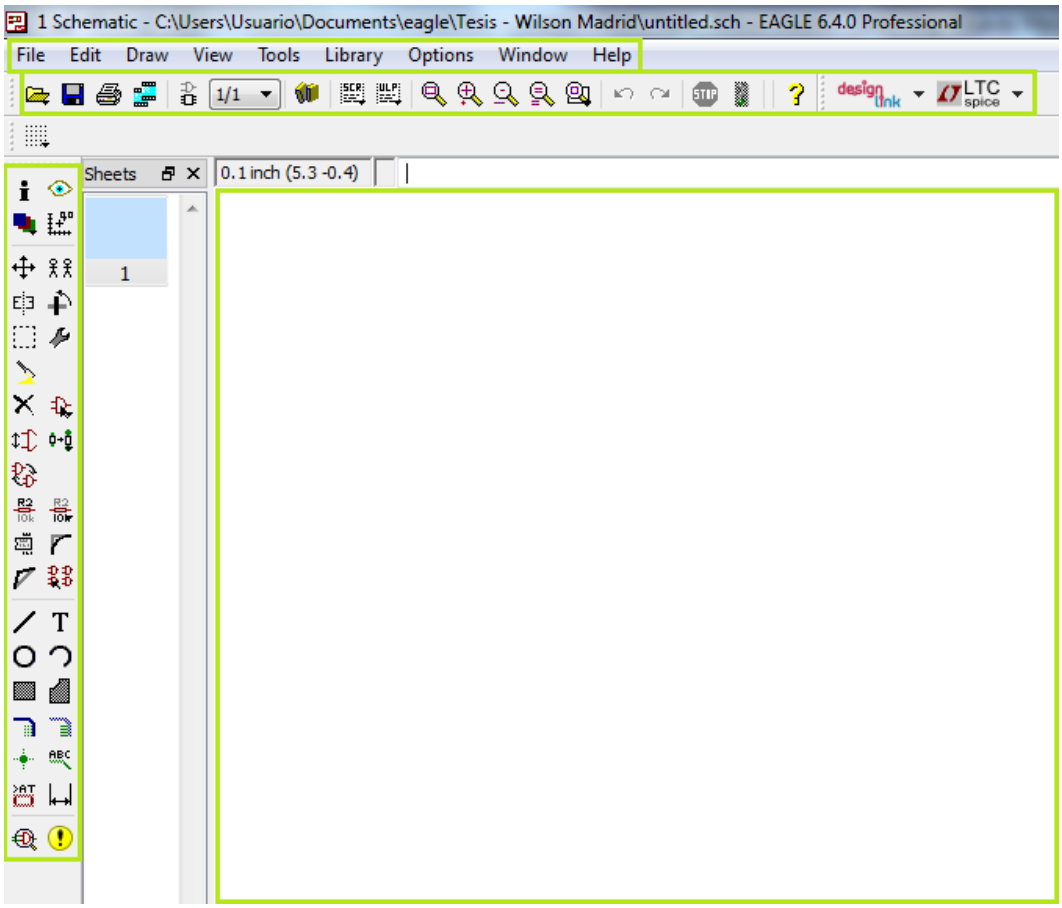


Fuente: Software Eagle.

Elaborado: Autor.

En la figura 2.18 se muestra la ventana de trabajo del editor de esquemas que tiene la misma distribución que la ventana del editor de PCB.

Figura 2.18: Ventana de trabajo en EAGLE.



Fuente: Software EAGLE.

Elaborado: Autor.

2.11.2 TIPOS DE ARCHIVOS.

En la tabla 2.7 se muestra un listado con los tipos de archivo más importantes que pueden ser editados con EAGLE:

Tabla 2.7: Tipos de Archivos en Eagle.

TIPO	VENTANA	NOMBRE
Placa	Editor de Líneas de conexión	*.brd
Esquema	Editor de Esquemas	*.sch
Librería	Editor de Librerías	*.lbr
Fichero Script	Editor de Textos	*.scr
Programa en Lenguaje Usuario	Editor de Textos	*.ulp

Fuente: Manual de Software Eagle.

Elaborado: Autor.

CAPITULO III

DISEÑO DE TARJETAS ESCUDOS

Las tarjetas escudos se realizaron por método impreso y de 2 capas, debido a que este método es un poco económico y muy confiable para la elaboración de tarjetas con circuitos impresos.

Se comprobó la continuidad de las pistas y de las conexiones entre las caras de las tarjetas escudos a lo largo del proceso de montaje de los componentes para constatar las conexiones eléctricas de la plataforma programable.

Las tarjetas escudos y los módulos que se implementaron son:

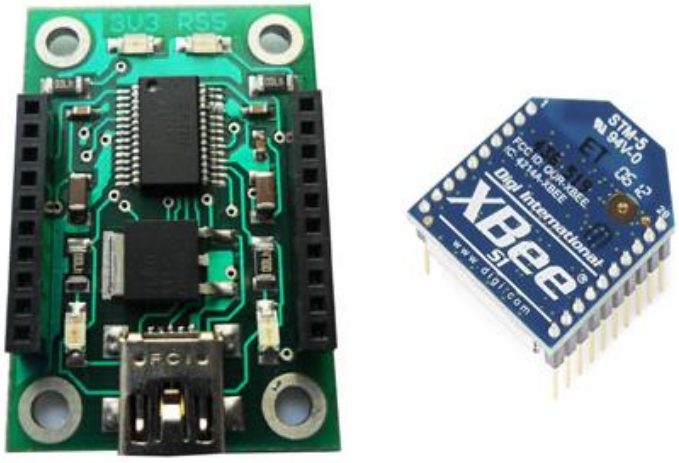
1. Tarjeta escudo Xbee-USB con módulo Xbee de 1mw-100m.
2. Tarjeta escudo XBEE-TTL con módulo Xbee de 1mw-100m.
3. Tarjeta escudo con Sensor Infrarrojo QRD1114.
4. Tarjeta escudo Control de Servo-Motores con módulo servomotor HS311.
5. Tarjeta escudo Ethernet con conector RJ45.
6. Tarjeta escudo con Sensor Ultrasónico HY-SFR05.

Finalmente se integrarán a la plataforma principal con sus respectivos puertos de conexión.

3.1 TARJETA ESCUDO XBEE-USB CON MÓDULO XBEE DE 1mw a 100mw.-

La tarjeta escudo XBEE - USB es un módulo de comunicación serial por puerto USB que se puede usar para la interface entre cualquier módulo XBEE y un PC, como se muestra en la figura 3.1.

Figura 3.1: Tarjeta Escudo XBEE-USB y el módulo Xbee de 1mw-100m.



Fuente: Autor.
Elaborado: Autor.

Esta tarjeta escudo permite conectar y utilizar cualquier módulo XBEE directamente mediante un puerto USB. Su diseño es muy útil y tan sencillo que usa el interfaz USB del PC para la alimentación de voltaje, además con el cable USB se tendrá acceso a las patillas de TX/RX del módulo y estará listo para funcionar y empezar a programarlo.

Donde;

TX: Pin de transmisión.

RX: Pin de Recepción.

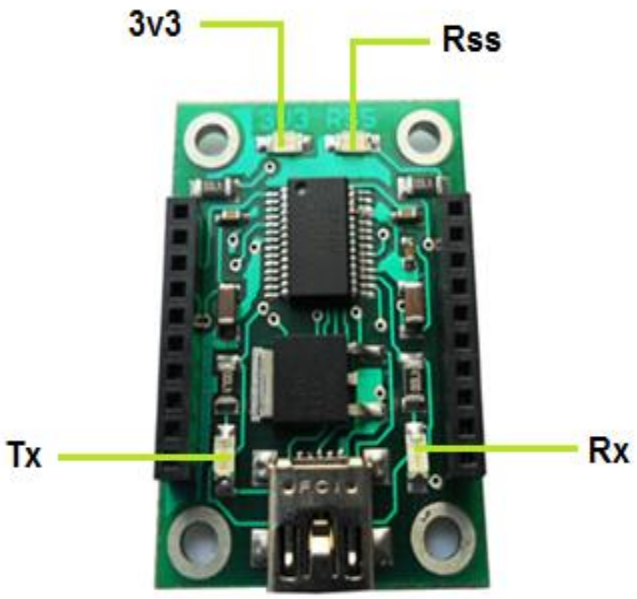
Es muy ideal para establecer una base inalámbrica desde un PC y así poder conectar sin cables a una placa que utilice un módulo XBEE. Esto aumentará el número de aplicaciones en robótica, o incluso en proyectos de telemetría.

3.1.1 ESPECIFICACIONES DE DISEÑO.-

La figura 3.2 muestra las especificaciones de diseño de la tarjeta y la figura 3.3 el esquemático de diseño.

- Un Led indicador de encendido (3V3), RSS, TX y RX.
- Velocidad de transferencia de datos desde 300 Baud hasta 3MBaud.
- Soporta 7 a 8 bits de datos, 1 o 2 bits stop.
- Alimentación de voltaje a través del interfaz USB del PC.

Figura 3.2: Especificaciones de diseño de la tarjeta escudo XBEE-USB.



Fuente: Autor.
Elaborado: Autor.

Dónde:

- **3V3**: Indica el voltaje XBEE.
- **Tx:** Indica la transmisión de datos.
- **Rx:** Indica la recepción de datos.
- **Rss:** Indica la actividad XBEE (RSSI).

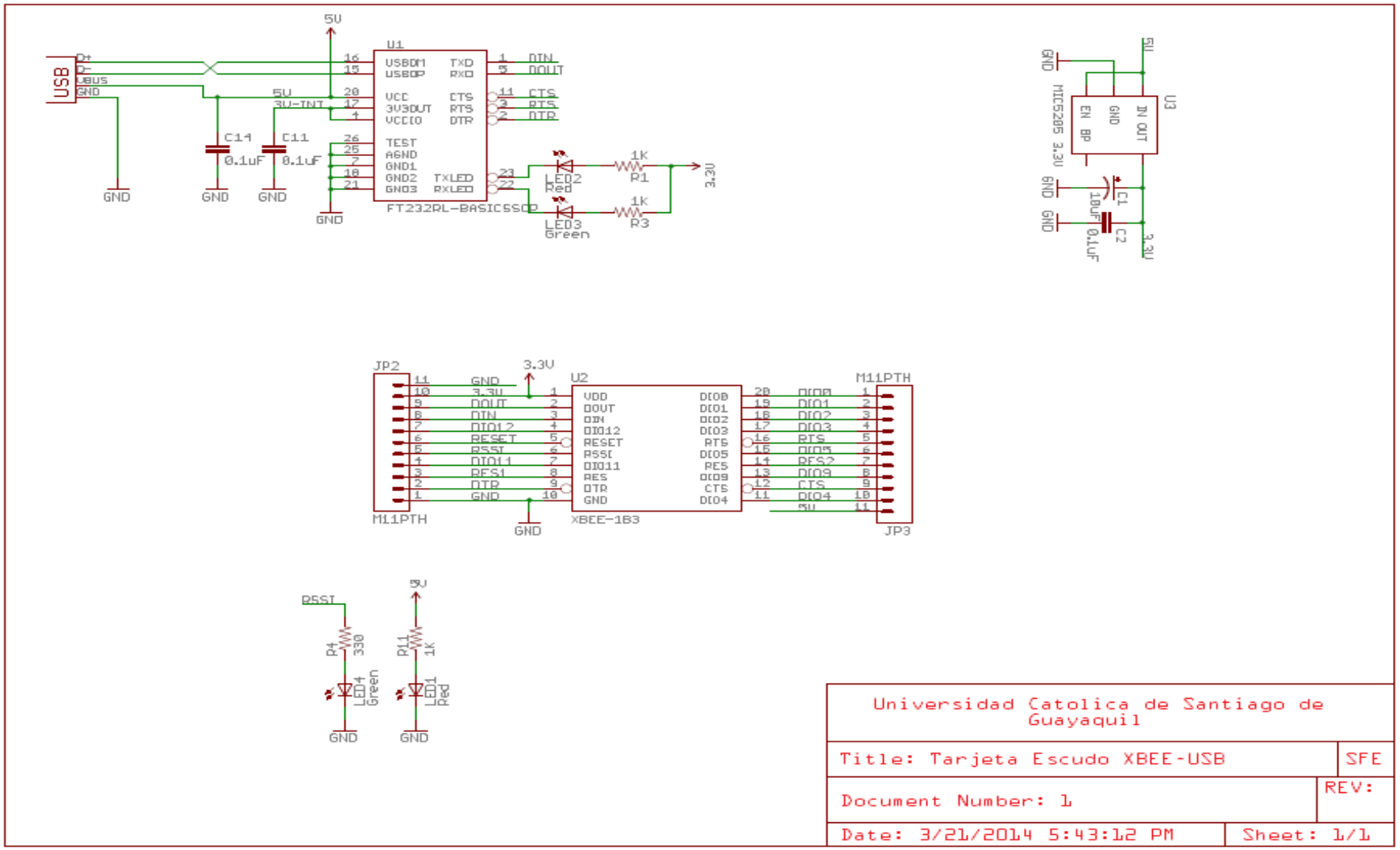
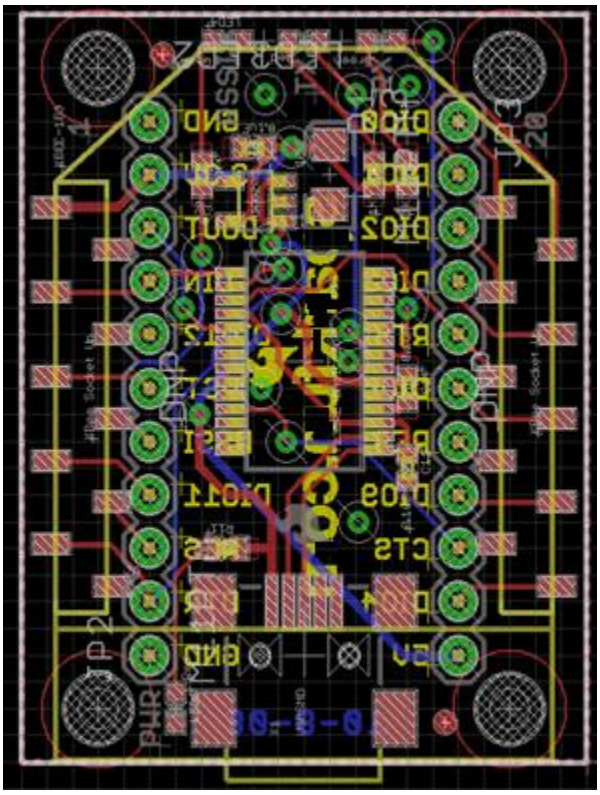


Figura 3.3: Esquemático de la Tarjeta Escudo XBEE-USB.

Fuente: Autor

La figura 3.4 muestra el esquema PCB de la tarjeta, con la ubicación de cada uno de los componentes y sus respectivas conexiones sobre la placa de circuitos impresos.

Figura 3.4: Esquema PCB de la tarjeta escudo XBEE-USB.



Fuente: Autor.

Elaborado: Autor.

3.1.2 APLICACIONES.-

Algunas aplicaciones en las que se puede usar la tarjeta son:

- ✓ Puede usarse en una interface de comunicación serial con el computador y el módulo XBEE.
- ✓ En la adquisición y envío de datos hacia el computador.

- ✓ Configuración inalámbrica de Módulos XBEE.
- ✓ Telemetría, para el uso de mediciones.
- ✓ Comunicaciones inalámbricas.
- ✓ Radio control.

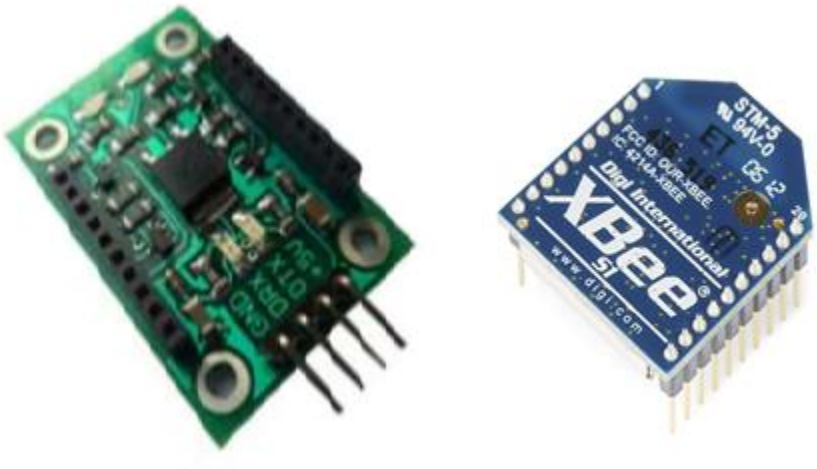
3.2 TARJETA ESCUDO XBEE-TTL CON MÓDULO XBEE DE 1mw - 100mw.-

Esta tarjeta escudo XBEE - TTL es un módulo de comunicación serial que sirve para ajustar los niveles de voltajes de cualquier módulo XBEE a niveles TTL.

Esta tarjeta está desarrollada para que de una forma muy simple se puedan adaptar los transmisores Inalámbricos Xbee a una computadora por medio de la conexión USB que se conecta directamente al PC (a diferencia de la Explorer mini USB), esta tarjeta puede soportar módulos otras series Xbee a parte la implementada. La tarjeta no necesita alimentación de voltaje adicional, ya que esta se toma directamente del puerto USB, adicionalmente cuenta con un botón de reset que se usa en algunas programaciones de los módulos Xbee. Además se la puede utilizar como un conversor de USB a serie y a este puerto se puede acceder mediante 4 pines que se encuentran en la parte frontal.

La figura 3.5 muestra la tarjeta escudo con su respectivo módulo XBEE.

Figura 3.5: Tarjeta Escudo XBEE-TTL y módulo Xbee de 1mw-100m.



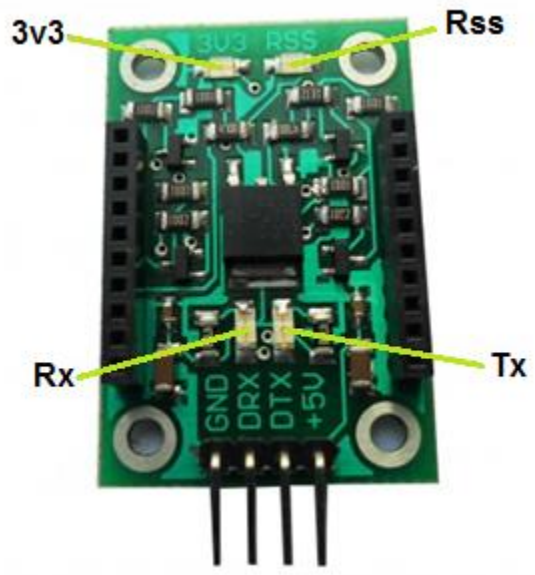
Fuente: Autor.
Elaborado: Autor.

3.2.1 ESPECIFICACIONES DE DISEÑO.-

La figura 3.5 muestra las especificaciones de diseño de la tarjeta y la figura 3.6 el esquemático de diseño.

- Un led indicador de encendido (3V3), RSS, TX y RX.
- 4 pines para alimentación de voltaje y comunicaciones serial.
- Transferencia de velocidad de datos de 300 Baud hasta 3MBaud.
- Soporta 7 a 8 bits datos y 1 o 2 bits stop.

Figura 3.6: Especificaciones de diseño de la tarjeta escudo XBEE-TTL.



Fuente: Autor.

Elaborado: Autor.

Dónde:

- **3v3:** Indica el voltaje del módulo XBEE.
- **Tx:** Indica la transmisión de datos.
- **Rx:** Indica la recepción de datos.
- **DRX:** Es el pin de recepción de datos en niveles TTL.
- **DTX:** Es el pin de transmisión de datos en niveles TTL.
- **+5V:** Es la salida de voltaje 5VDC.
- **GND:** Es la referencia 0V.
- **RSS:** Indica la actividad del módulo XBEE (RSSI).

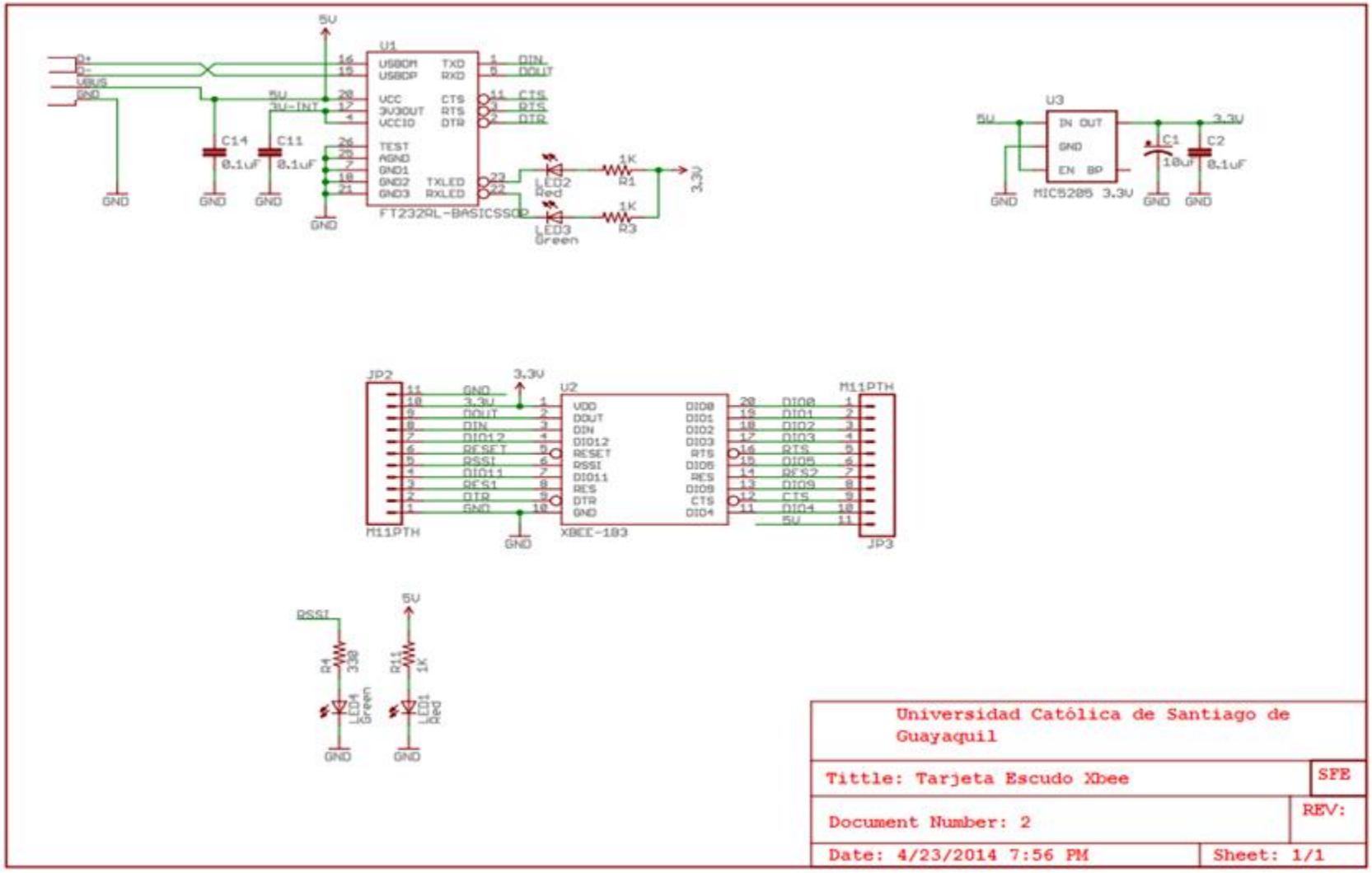
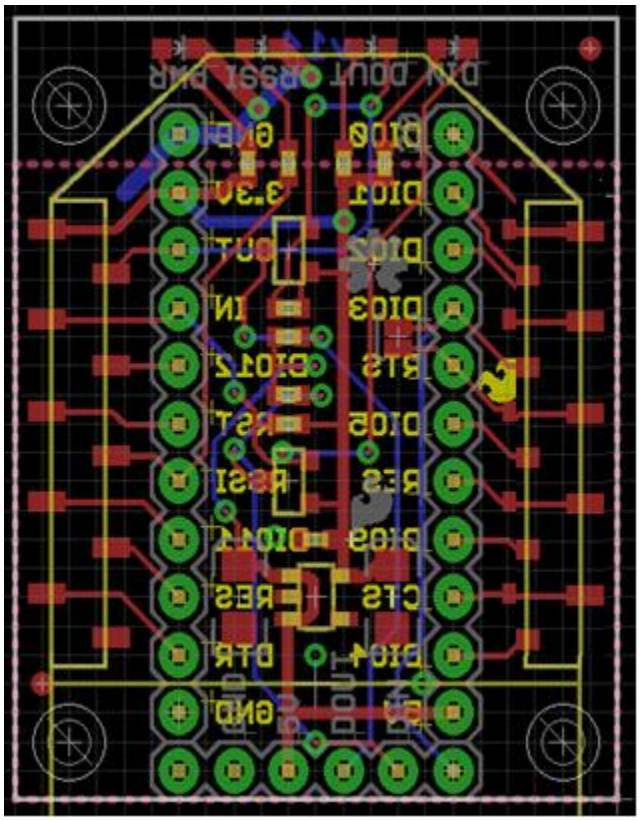


Figura 3.7: Esquemático de la Tarjeta Escudo XBEE-TTL.

Fuente: Autor

La figura 3.8 muestra el esquema PCB de la tarjeta, con la ubicación de cada uno de los componentes y sus respectivas conexiones sobre la placa de circuitos impresos.

Figura 3.8: Esquema PCB de la tarjeta escudo XBEE-TTL.



Fuente: Autor.

Elaborado: Autor.

3.2.2 APLICACIONES.-

Algunas aplicaciones en las que se puede usar la tarjeta son:

- ✓ Interface de comunicación serial para niveles TTL.
- ✓ Adquisición y envío de datos seriales.

3.3 TARJETA ESCUDO CON SENSOR INFRARROJO QRD1114.-

Esta tarjeta escudo cuenta con un sensor infrarrojo QRD1114 con conector de 3 patillas, es diseñada para el uso de robots seguidores de línea con una superficie reflectora. Este sensor en la tarjeta integra el acondicionamiento electrónico necesario para tener una señal de salida digital, y así poder diferenciar el contraste entre color negro y blanco, además la tarjeta tiene un led en la parte superior para verificar la lectura del sensor.

Para su uso solo se tiene que conectar alimentación, tierra y la salida a la plataforma base.

Figura 3.9: Tarjeta Escudo con Sensor Infrarrojo QRD1114.

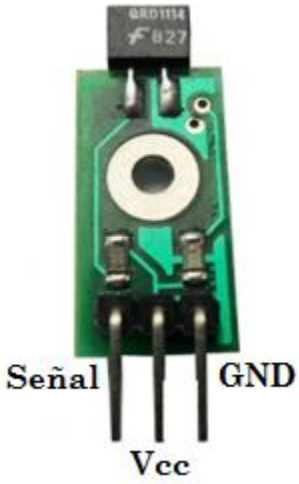
Fuente: Autor.
Elaborado: Autor.

3.3.1 ESPECIFICACIONES.-

La figura 3.8 muestra las especificaciones de diseño de la tarjeta y la figura 3.9 el esquemático de diseño.

- 1 sensores infrarrojo QRD1114.
- Conector de 3 patillas tipo extensión servomotor.
- La distancia máxima de la superficie reflectora es de 8mm.
- Posee una alimentación de voltaje de 5VDC.

Figura 3.10: Especificaciones de diseño de la tarjeta escudo con sensor infrarrojo QRD1114.



Fuente: Autor.

Elaborado: Autor.

.

Donde;

- **VCC:** Es la entrada de voltaje +5VDC.
- **S:** Es la señal del sensor QRD1114.
- **GND:** Es el voltaje a tierra.

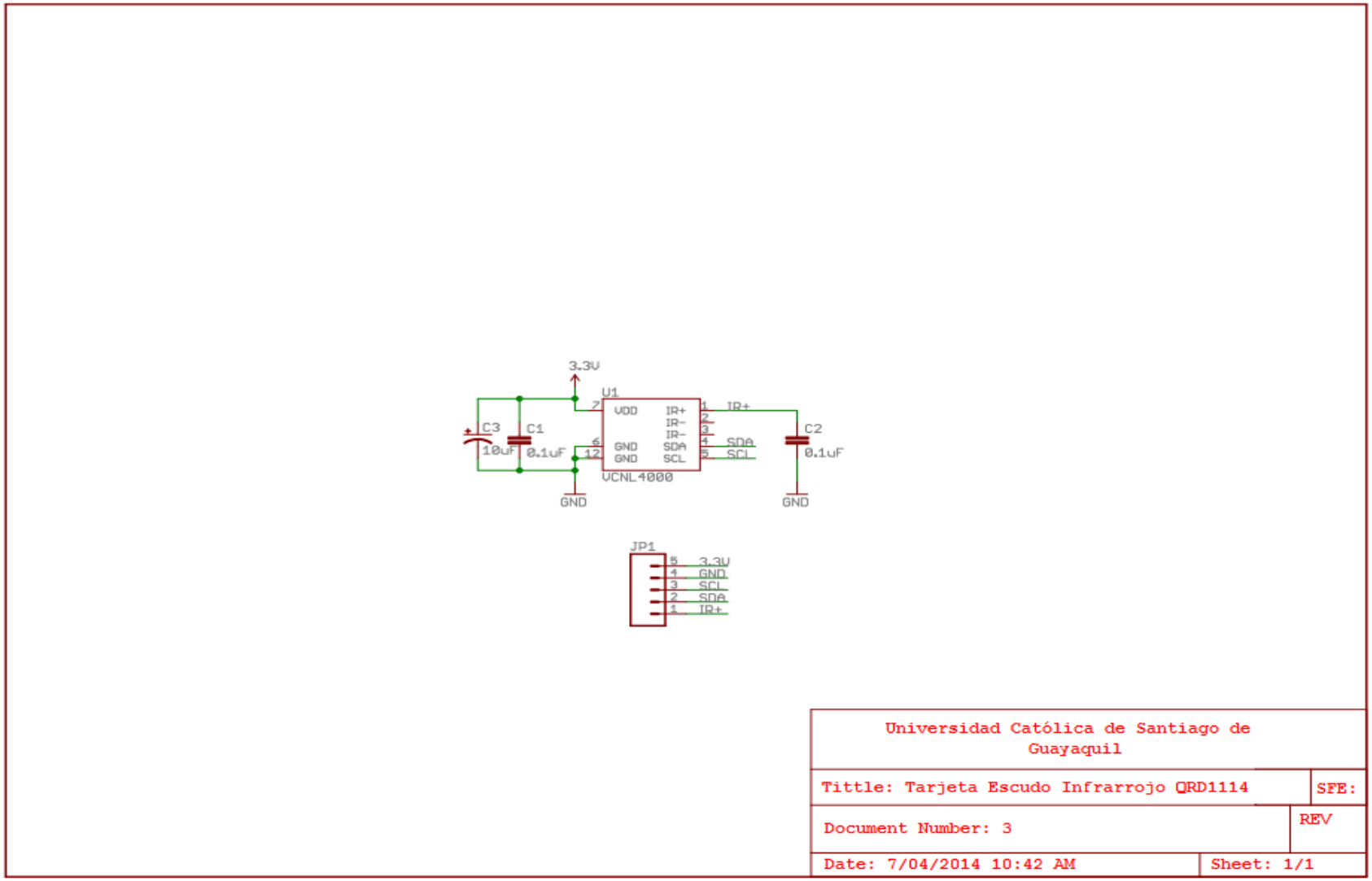


Figura 3.11: Esquemático de la Tarjeta Escudo Infrarrojo QRD1114.

Fuente: Autor

3.3.2 APLICACIONES.-

Algunas aplicaciones en las que se puede usar la tarjeta son:

- ✓ Detección de líneas en robot seguidores de líneas.
- ✓ Identificación de colores blanco y negro.

3.4 TARJETA ESCUDO PARA CONTROL DE SERVO-MOTORES Y MÓDULO SERVO-MOTOR HS311.-

Esta tarjeta escudo permite controlar hasta 8 servos-motores simultáneamente tanto desde el microcontrolador ubicado en la plataforma base como desde un PC y capacidad para trabar con el bus de comunicación serial I2C, el cual utiliza dos líneas para transmitir la información.

Tiene un controlador inteligente para este tipo de servomotores y además permite controlar tanto el rango como la posición mediante sencillos códigos de programas enviados por su puerto con detección automática.

Se utiliza el servomotor Hitec HS-311, porque es el servo más usado para el aprendizaje académico, esto debido a su bajo precio y alta calidad, como se muestra en la figura 3.12.

Figura 3.12: Tarjeta Escudo para control de Servos-Motores y módulo servo-motor HS311.

Fuente: Autor.

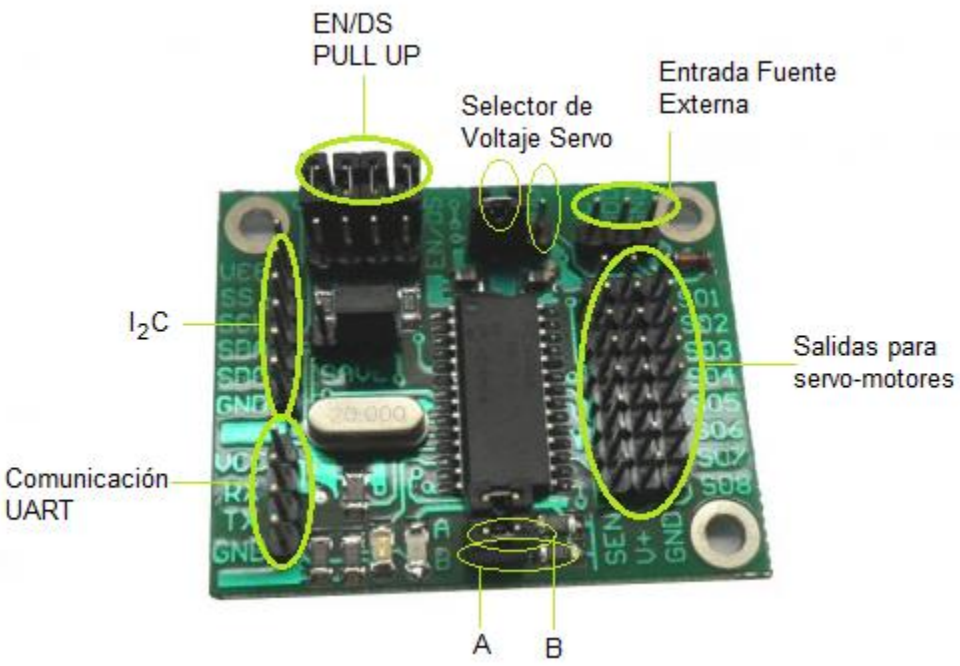
Elaborado: Autor.

3.4.1 ESPECIFICACIONES.-

La figura 3.13 muestra las especificaciones de diseño de la tarjeta y la figura 3.14 el esquemático de diseño.

- 8 puertos para el control de servos-motores.
- Voltaje de E/S de 0V y 5V.
- Voltaje de alimentación de 5-16V.
- Consumo de corriente de 5mA.

Figura 3.13: Especificaciones de diseño de la tarjeta escudo para control de servo-motores.



Fuente: Autor.

Elaborado: Autor.

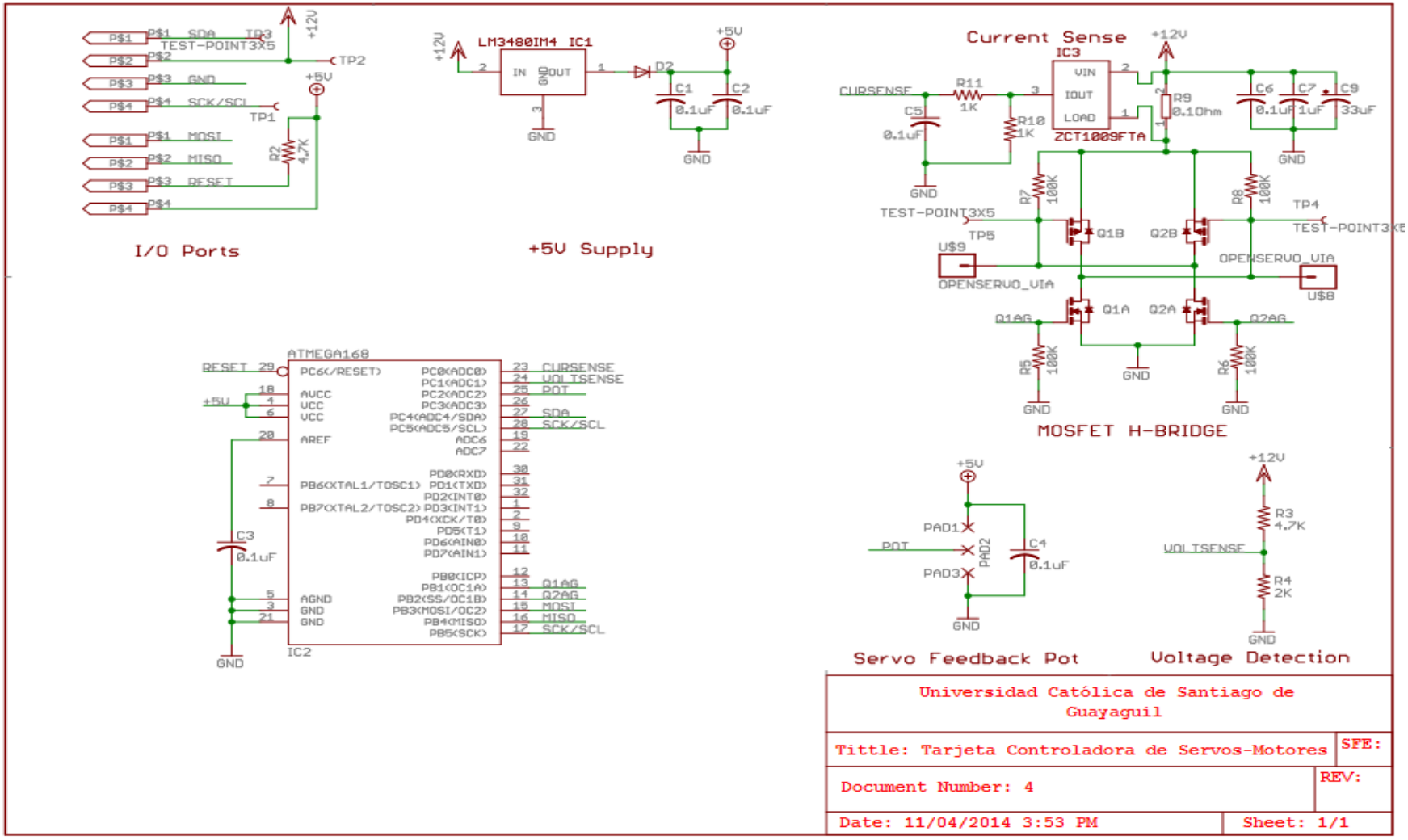
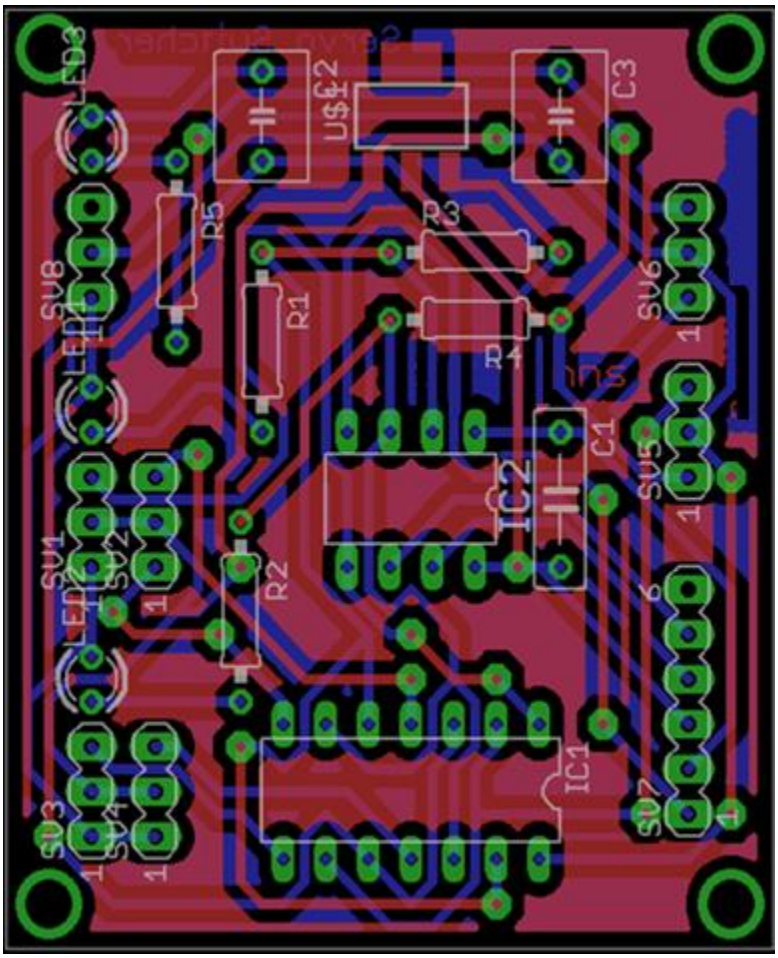


Figura 3.14: Esquemático de la Tarjeta Escudo Controladora de Servos-Motores.

Fuente: Autor.

La figura 3.15 muestra el esquema PCB de la tarjeta, con la ubicación de cada uno de los componentes y sus respectivas conexiones sobre la placa de circuitos impresos.

Figura 3.15: Esquema PCB de la tarjeta escudo para servo-motores.



Fuente: Autor.

Elaborado: Autor.

3.4.2 APLICACIONES.

Algunas aplicaciones en las que se puede usar la tarjeta son:

- ✓ Implementación en robótica con múltiples servo-motores.
- ✓ Aeromodelismo.
- ✓ Control de servo-motores para brazos robóticos, bípedos, etc.

3.5 TARJETA ESCUDO ETHERNET CON CONECTOR RJ45.-

Esta tarjeta escudo permite disponer de todo lo necesario para una interfaz de red por Ethernet y conectar a diferentes sistemas como el internet y otros por medio de una red LAN de comunicación, para cualquier microcontrolador, en este caso el 18F4550, como se muestra en la figura 3.16

Figura 3.16 Tarjeta Escudo Ethernet con conector RJ45.

Fuente: Autor.
Elaborado: Autor.

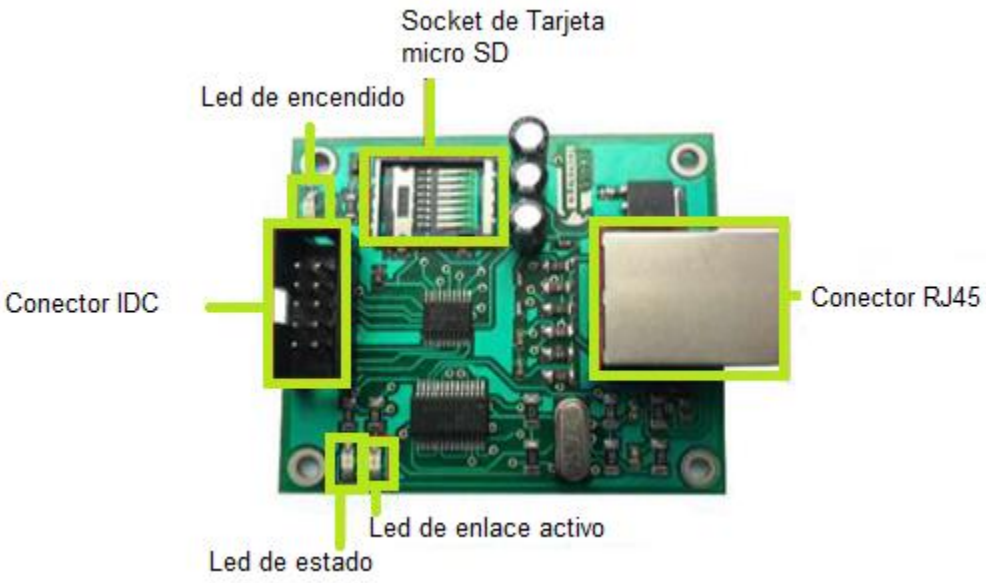
Además, permite enviar y recibir datos por Ethernet mediante paquetes UDP.

3.5.1 ESPECIFICACIONES.-

Entre las especificaciones de su diseño consta de:

- 1 LED de estado.
- 1 conector IDC.
- 1 conector RJ45 que posee bobinado interno.
- Interfaz SPI.
- Buffer de memoria de 8Kb.

Figura 3.17 Especificaciones de diseño de Tarjeta Escudo Ethernet con conector RJ45.



Fuente: Autor.

Elaborado: Autor.

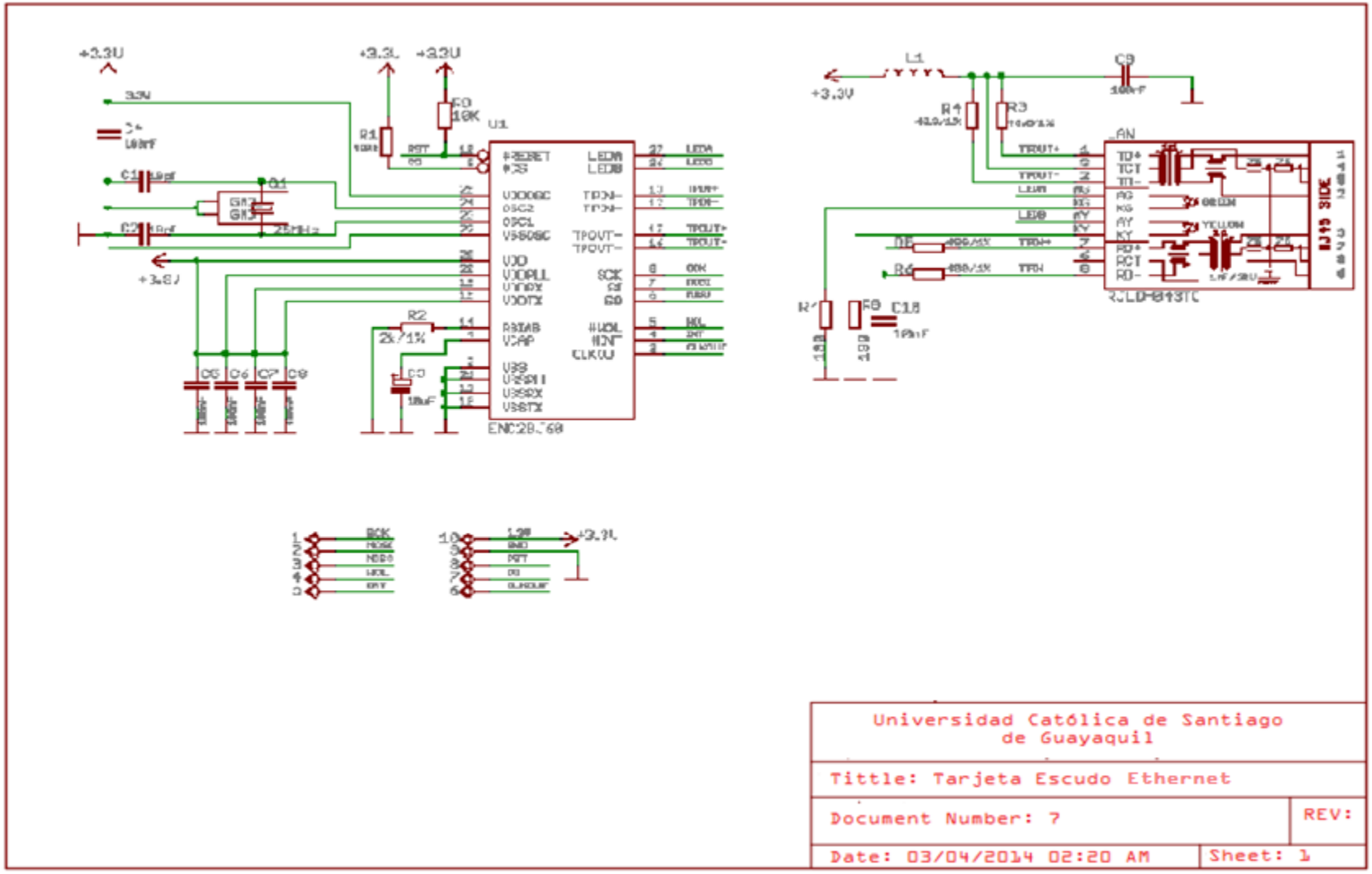
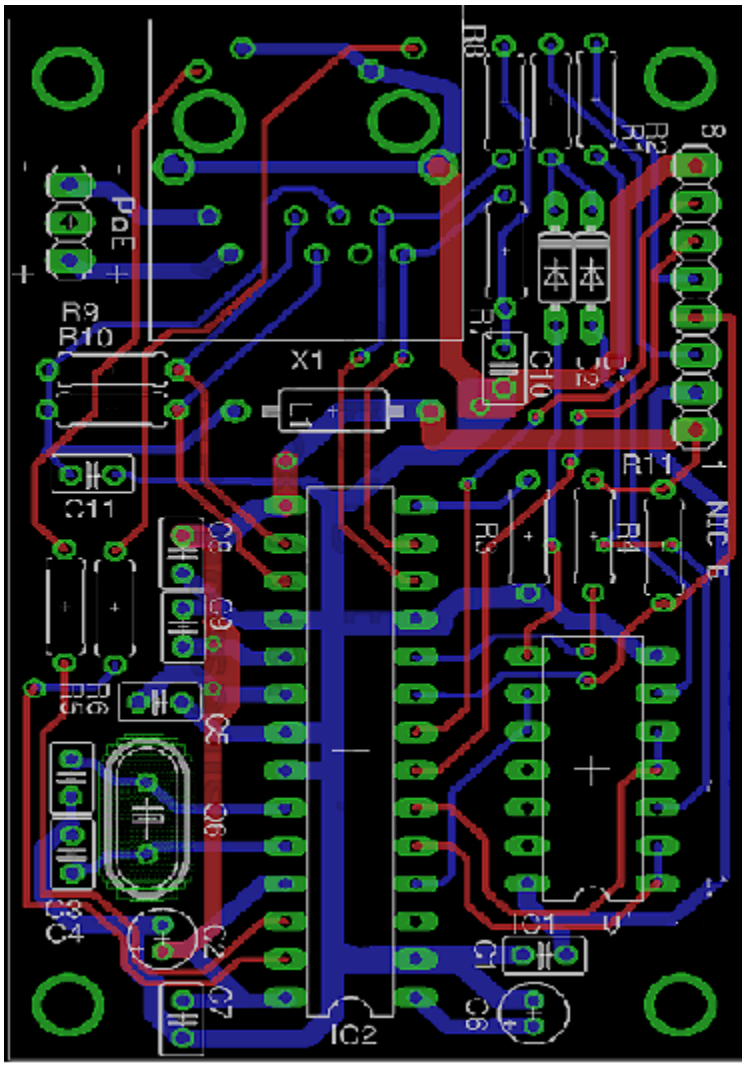


Figura 3.18: Esquemático de la Tarjeta Escudo Ethernet con conector RJ45.

Fuente: Autor

Figura 3.19: Esquema PCB de la tarjeta escudo Ethernet con conector RJ45.



Fuente: Autor.

Elaborado: Autor.

3.5.2 APLICACIONES.-

Algunas aplicaciones en las que se puede usar la tarjeta son:

- ✓ Adquisición y envío de datos vía Ethernet por paquetes UDP para monitoreo de temperatura y control de eventos.

- ✓ Conexión del microcontrolador a Internet por medio del puerto TCP/IP.

3.6 TARJETA ESCUDO CON SENSOR ULTRASONICO HY-SFR05.-

Esta tarjeta escudo con modulo ultrasónico o de ultrasonido integrado y con receptor y emisor en un solo dispositivo, el cual permite hacer mediciones de distancia, control de nivel, detector de objetos, etc.; como se muestra en la figura 3.20

Figura 3.20: Tarjeta Escudo con sensor Ultrasónico HY-SFR05.

Fuente: Autor.
Elaborado: Autor.

3.3.1 ESPECIFICACIONES.-

La figura 3.20 muestra las especificaciones de diseño de la tarjeta y la figura 3.21 el esquemático de diseño.

Figura 3.21: Especificaciones de diseño de la tarjeta escudo con sensor Ultrasónico HY-SFR05.

Fuente: Autor.

Elaborado: Autor.

3.6.2 APLICACIONES.-

Algunas aplicaciones en las que se puede usar la tarjeta son:

- ✓ Medición de distancia.
- ✓ Control de nivel.
- ✓ Detector de objetos.

CAPÍTULO IV

DISEÑO E IMPLEMENTACIÓN DE SOFTWARE DE LAS TARJETAS ESCUDOS

Si se desea realizar la programación de los microcontroladores PIC en un lenguaje como el C, es preciso utilizar un compilador de C. Dicho compilador nos generara ficheros en formato Intel-hexadecimal, que es el necesario para programar (utilizando un programador de PIC) un microcontrolador de 6, 8, 18 o 44 pines.

El compilador de C que se utiliza para la programación de las tarjetas escudos es el PCW de la casa CCS Inc. A su vez, el compilador lo integrara en un entorno de desarrollo (IDE) que nos va a permitir desarrollar todas y cada una de las fases que se compone un proyecto, desde la edición hasta la compilación pasando por la depuración de errores. La última fase, a excepción de la depuración y retoques hardware finales, será programar el PIC.

Este compilador traduce el código C del archivo fuente a lenguaje máquina para los microcontroladores PIC, generando así un archivo en formato hexadecimal ***.hex*****.** Además genera otros seis ficheros.

Para la grabación del código de programación en el microcontrolador PIC18F4550 se utiliza el PicKit2.

4.1 PROGRAMACIÓN DE LA TARJETA ESCUDO CON MÓDULO XBEE-USB.-

Para la programación de la tarjeta escudo, se siguió la siguiente tabla con las indicaciones de las conexiones al microcontrolador PIC18F4550.

Tabla 4.1: Conexiones de puertos.

PINES DE CONEXIÓN	XBEE-USB
PUERTO A	NC
PUERTO B	NC
PUERTO C	RC7
PUERTO D	NC
PUERTO E	NC

Fuente: Autor.
Elaborado: Autor.

NOTA.- Las siglas NC significan No conectado.

4.1.1 CODIGO DE PROGRAMACIÓN PARA PUERTOS DE CONEXIÓN CON EL MICROCONTROLADOR.-

Para la asignación del encabezado de códigos para la programación de las tarjetas utilizamos los siguientes pasos y comandos:

1. Usamos el protocolo de comunicación serial RS232 a través del siguiente comando

```
#use rs232
```

2. Asignamos al protocolo la velocidad de comunicación en baudios (baud) en que vamos a transmitir los datos.

   Baud = 9600

3. Agregamos el bit de paridad que nos permite comprobar los errores.

   Parity = N

4. Asignamos la patilla (PIN) de la tarjeta que tiene comunicación con el puerto del microcontrolador para la salida de datos.

   Xmit = PIN_C6

5. Asignamos a que patilla (PIN) del microcontrolador por donde va a recibir los datos.

   Rcv = PIN_C7

6. Por último el tamaño de la cantidad de datos que recibe el microcontrolador que es de 8 bits.

   Bits = 8

El código completo de la programación de la tarjeta se lo encontrara en la sección de ANEXOS.

4.2 PROGRAMACIÓN DE LA TARJETA ESCUDO CON MÓDULO XBEE-TTL.-

Para la programación de la tarjeta escudo, se siguió la siguiente tabla con las indicaciones de las conexiones al microcontrolador PIC18F4550.

Tabla 4.2: Conexiones de puertos.

PINES DE CONEXIÓN	XBEE-TTL
PUERTO A	NC
PUERTO B	NC
PUERTO C	RC6
PUERTO D	NC
PUERTO E	NC

Fuente: Autor.
Elaborado: Autor.

NOTA.- Las siglas NC significan No conectado.

4.2.1 CÓDIGO DE PROGRAMACIÓN PARA PUERTOS DE CONEXIÓN CON EL MICROCONTROLADOR.-

Para la asignación del encabezado de códigos para la programación de las tarjetas utilizamos los siguientes pasos y comandos:

1. Usamos el protocolo de comunicación serial RS232 a través del siguiente comando

   #use rs232

2. Asignamos al protocolo la velocidad de comunicación en baudios (baud) en que vamos a transmitir los datos.

   Baud = 9600

3. Agregamos el bit de paridad que nos permite comprobar los errores.

   Parity = N

4. Asignamos la patilla (PIN) de la tarjeta que tiene comunicación con el puerto del microcontrolador para la salida de datos.

   Xmit = PIN_C6

5. Asignamos a que patilla (PIN) del microcontrolador por donde va a recibir los datos.

   Rcv = PIN_C7

6. Por último el tamaño de la cantidad de datos que recibe el microcontrolador que es de 8 bits.

   Bits = 8

El código completo de la programación de la tarjeta se lo encontrara en la sección de ANEXOS.

4.3 TARJETA ESCUDO CON SENSOR INFRARROJO QRD1114.-

Para la programación de la tarjeta escudo, se siguió la siguiente tabla con las indicaciones de las conexiones al microcontrolador PIC18F4550.

Tabla 4.3: Conexiones de puertos.

PINES DE CONEXIÓN	TARJETA SENSOR	LED
PUERTO A	NC	NC
PUERTO B	NC	NC
PUERTO C	RC0	NC
PUERTO D	NC	ENCENDIDO
PUERTO E	NC	NC

Fuente: Autor.
Elaborado: Autor.

NOTA.- Las siglas NC significan No conectado.

4.3.1 CÓDIGO DE PROGRAMACIÓN PARA PUERTOS DE CONEXIÓN CON EL MICROCONTROLADOR.-

Para la asignación del encabezado de códigos para la programación de las tarjetas utilizamos los siguientes pasos y comandos:

1. Usamos el protocolo de comunicación serial RS232 a través del siguiente comando

   #use rs232

2. Asignamos al protocolo la velocidad de comunicación en baudios (baud) en que vamos a transmitir los datos.

   Baud = 9600

3. Agregamos el bit de paridad que nos permite comprobar los errores.

   Parity = N

4. Asignamos la patilla (PIN) de la tarjeta que tiene comunicación con el puerto del microcontrolador para la salida de datos.

   Xmit = PIN_C6

5. Asignamos a que patilla (PIN) del microcontrolador por donde va a recibir los datos.

   Rcv = PIN_C7

6. Por último el tamaño de la cantidad de datos que recibe el microcontrolador que es de 8 bits.

   Bits = 8

El código completo de la programación de la tarjeta se lo encontrara en la sección de ANEXOS.

4.4 TARJETA ESCUDO CONTROL DE SERVO-MOTORES CON MÓDULO SERVOMOTOR HS311.-

Para la programación de la tarjeta escudo, se siguió la siguiente tabla con las indicaciones de las conexiones al microcontrolador PIC18F4550.

Tabla 4.4: Conexiones de puertos.

PUERTOS DE CONEXIÓN	SERVOMOTOR
PUERTO A	NC
PUERTO B	NC
PUERTO C	NC
PUERTO D	RD0 a RD7
PUERTO E	NC

Fuente: Autor.
Elaborado: Autor.

NOTA.- Las siglas NC significan No conectado.

4.4.1 CÓDIGO DE PROGRAMACIÓN PARA PUERTOS DE CONEXIÓN CON EL MICROCONTROLADOR.-

Para la asignación del encabezado de códigos para la programación de las tarjetas utilizamos los siguientes pasos y comandos:

1. Usamos el protocolo de comunicación serial RS232 a través del siguiente comando

#use rs232

2. Asignamos al protocolo la velocidad de comunicación en baudios (baud) en que vamos a transmitir los datos.

   Baud = 9600

3. Agregamos el bit de paridad que nos permite comprobar los errores.

   Parity = N

4. Asignamos la patilla (PIN) de la tarjeta que tiene comunicación con el puerto del microcontrolador para la salida de datos.

   Xmit = PIN_C6

5. Asignamos a que patilla (PIN) del microcontrolador por donde va a recibir los datos.

   Rcv = PIN_C7

6. Por último el tamaño de la cantidad de datos que recibe el microcontrolador que es de 8 bits.

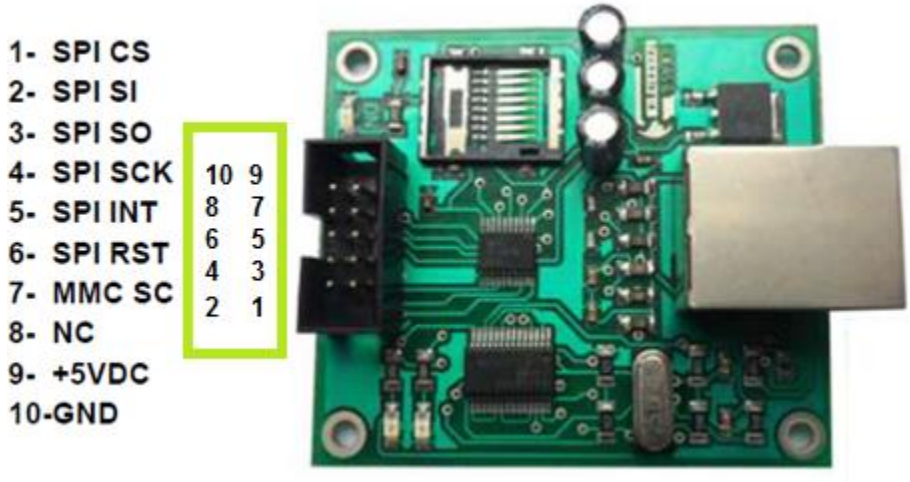
   Bits = 8

El código completo de la programación de la tarjeta se lo encontrara en la sección de ANEXOS.

4.5 TARJETA ESCUDO ETHERNET CON CONECTOR RJ45.-

Para la programación de la tarjeta escudo, se siguió la siguiente figura con las indicaciones de las conexiones al microcontrolador PIC18F4550.

Figura 4.1: Conexiones de patillas de la tarjeta Ethernet.



Fuente: Autor.

Elaborado: Autor.

4.5.1 CÓDIGO DE PROGRAMACIÓN PARA PUERTOS DE CONEXIÓN CON EL MICROCONTROLADOR.-

Para la asignación del encabezado de códigos para la programación de las tarjetas utilizamos los siguientes pasos y comandos:

1. Usamos el protocolo de comunicación serial RS232 a través del siguiente comando

   #use rs232

2. Asignamos al protocolo la velocidad de comunicación en baudios (baud) en que vamos a transmitir los datos.

   Baud = 9600

3. Agregamos el bit de paridad que nos permite comprobar los errores.

   Parity = N

4. Asignamos la patilla (PIN) de la tarjeta que tiene comunicación con el puerto del microcontrolador para la salida de datos.

   Xmit = PIN_C6

5. Asignamos a que patilla (PIN) del microcontrolador por donde va a recibir los datos.

   Rcv = PIN_C7

6. Por último el tamaño de la cantidad de datos que recibe el microcontrolador que es de 8 bits.

   Bits = 8

El código completo de la programación de la tarjeta se lo encontrara en la sección de ANEXOS.

4.6 TARJETA ESCUDO CON SENSOR ULTRASÓNICO HY-SFR05.-

Para la programación de la tarjeta escudo, se siguió la siguiente tabla con las indicaciones de las conexiones al microcontrolador PIC18F4550.

Tabla 4.5: Conexiones de puertos.

PUERTOS DE CONEXIÓN	SERVOMOTOR
PUERTO A	NC
PUERTO B	RB0 a RB4
PUERTO C	NC
PUERTO D	NC
PUERTO E	NC

Fuente: Autor.

Elaborado: Autor.

4.6.1 CÓDIGO DE PROGRAMACIÓN PARA PUERTOS DE CONEXIÓN CON EL MICROCONTROLADOR.-

Para la asignación del encabezado de códigos para la programación de las tarjetas utilizamos los siguientes pasos y comandos:

1. Usamos el protocolo de comunicación serial RS232 a través del siguiente comando

   #use rs232

2. Asignamos al protocolo la velocidad de comunicación en baudios (baud) en que vamos a transmitir los datos.

   Baud = 9600

3. Agregamos el bit de paridad que nos permite comprobar los errores.

   Parity = N

4. Asignamos la patilla (PIN) de la tarjeta que tiene comunicación con el puerto del microcontrolador para la salida de datos.

   Xmit = PIN_C6

5. Asignamos a que patilla (PIN) del microcontrolador por donde va a recibir los datos.

   Rcv = PIN_C7

6. Por último el tamaño de la cantidad de datos que recibe el microcontrolador que es de 8 bits.

   Bits = 8

El código completo de la programación de la tarjeta se lo encontrara en la sección de ANEXOS.

CAPÍTULO V

COSTOS

5.1 Tarjeta escudo Xbee-USB con módulo Xbee de 1mw-100m.

	COSTO
Módulo Xbee de 1mw-100m	14,95
Componentes Electrónicos	8,85
Circuito Impreso	5
SUBTOTAL	28,8
IVA 12%	3,46
TOTAL	**32,26**

5.2 Tarjeta escudo XBEE-TTL con módulo Xbee de 1mw-100m.

	COSTO
Módulo Xbee de 1mw-100m	14,95
Componentes Electrónicos	7,56
Circuito Impreso	5
SUBTOTAL	27,51
IVA 12%	3,30
TOTAL	**30,81**

5.3 Tarjeta escudo con Sensor Infrarrojo QRD1114.

	COSTO
Sensor QRD1114	1,95
Componentes Electrónicos	0,93
Circuito Impreso	4,5
SUBTOTAL	7,38
IVA 12%	0,89
TOTAL	**8,27**

5.4 Tarjeta escudo Control de Servo-Motores con módulo servomotor HS311.

	COSTO
Servomotor HS311	10,64
Componentes Electrónicos	4,85
Circuito Impreso	15,7
SUBTOTAL	31,19
IVA 12%	3,74
TOTAL	**34,93**

5.5 Tarjeta escudo Ethernet con conector RJ45.

	COSTO
Conector Ethernet RJ45	2,95
Componentes Electrónicos	6,85
Circuito Impreso	8,5
SUBTOTAL	18,3
IVA 12%	2,19
TOTAL	**20,49**

5.6 Tarjeta escudo con Sensor Ultrasónico HY-SFR05.

	COSTO
Sensor Ultrasónico HY-SFR05	16,98
SUBTOTAL	16,98
IVA 12%	2,04
TOTAL	**19,02**

El gasto total es de:

SUBTOTAL	130,16
IVA 12%	15,62
TOTAL	**145,78**

GLOSARIO TÉCNICO

A/D	Analógico/Digital.
ADC	*Analog Digital Converter* / Convertidor Analógico Digital.
ADCON0	Registro 0 de control del convertidor analógico/digital.
ADCON1	Registro 1 de control del convertidor analógico/digital.
ADFM	Bit selector de formato de resultado del convertidor analógico/digital.
ADRESH	Byte alto del registro de resultado del convertidor analógico/digital.
ADRESL	Byte bajo del registro de resultado del convertidor analógico/digital.
AC	*Alternating Current* / Corriente Alterna.
Bms	Bit menos significativo.
CAN	Controller Area Network / Red de Area de Controlador.
DC	Direct Current / Corriente Directa.
CMOS	Complementary Metal Oxide / Semiconductor de Oxido de Metal Complementario.
COP8	Control Oriented Processor / Procesador Orientado a Control.
CPU	Central Processing Unit / Unidad Central de Procesamiento.
D/A	Digital/Analógico.
DAC	*Digital* Analog Converter / Convertidor Digital Analógico.
DIP	Dual Inline Package / Encapsulado Doble en línea.
DSC	Digital Signal Controller / Controlador Digital de Señales.
DSP	Digital Signal Processor / Procesador Digital de Señales.
E/S	Entradas y Salidas.
EEPROM	Electrically Erasable Programmable Read-Only Memory / Memoria de solo lectura programable eléctricamente borrable.
EIA	Electronic Industries Association / Asociación de Industrias de Electronica.
FFT	Fast Fourier Transform / Transformada Rápida de Fourier.

FUSES	Fusibles.
HSPLL	Oscilador del microcontrolador.
ICD	In-Circuit Debugger / Depurador en Circuito.
ICE	In-Circuit Emulator / Emulador en Circuito.
IDE	Integrated Development Environment / Ambiente Integrado de Desarrollo.
IEEE	Institute for Electrical and Electronics Engineers / Instituto para Ingenieros Electricos y Electronicos.
IIC	Inter-Integrated Circuit / Inter-conexión de Circuitos Integrados.
ISP	In System Programming / Programacion en Sistema.
KHz	Kilo-Hertz.
GLCD	Graphic Liquid Crystal Display / Pantalla Grafica de Cristal Líquido.
LED	Light Emitting Diode / Diodo Emisor de Luz.
LPT	Line Print Terminal / Terminal de Línea de Impresora.
mA	Mili-Ampere.
MHz.	Mega-Hertz.
MIDI	Musical Instrument Digital Interface / Interfaz Digital de Instrumento Musical.
MIMO	Multiple Imput - Multiple Output / Múltiples Entradas - Múltiples Salidas.
MIPS	Million of Instructions per Second / Millones de Instrucciones por Segundo.
MOSFET	Metal Oxide Semiconductor Field Effect Transistor / Transistor de efecto de campo de Semiconductor de Oxido de Metal.
nF	Nano-Faradio.
pF	Pico-Faradio.
PIC	Controlador de Interfaz Periférico.
PWM	Pulse Width Modulation / Modulación por Ancho de Pulso.
RAM	Random Access Memory / Memoria de Acceso Aleatorio.
RC	Resistencia Capacitor.

RISC	Reduced Instruction Set Computer / Computadora con Conjunto Reducido de Instrucciones.
ROM	Read-Only Memory / Memoria de Solo Lectura.
RS232	Protocolo de comunicación serial.
SCI	Serial Comunication Interface / Interfaz de Comunicación Serial.
SCL	Serial Clock / Reloj Serial.
SD	Secure Digital / Seguro Digital.
SDA	Serial Data / Dato Serial.
SISO	Single Input - Single Output / Una Entrada – Una Salida.
Spbrg	Registro Generador de Baudios.
TOSC	Tiempo de Oscilación.
TTL	Transistor-Transistor Logic / Lógica Transistor-Transistor.
µC	Microcontrolador.
µF	Micro-Faradio.
µs	Micro-segundo.
USART	Universal Synchronous – Asynchronous Receiver Transmitter / Transmisor-Receptor Universal Síncrono-Asíncrono.
USB	Universal Serial Bus / Bus Serie Universal.
VDC	Voltage Direct Current / Voltaje de Corriente Directa.
Wi-Fi	Wireless Fidelity / Fidelidad Inalambrica.

CONCLUSIONES

- El resultado final del desarrollo de este trabajo de tesis, son varias tarjetas escudos programables, que permiten controlar una amplia variedad de procesos de acuerdo al algoritmo de control en el software programado en el microcontrolador.

- Las tarjetas escudos se diseñaron y se fabricaron cumpliendo los objetivos específicos, además para que tengan una coherencia entre los módulos aplicados y la estética, para reducir al menor tamaño posible cada tarjeta; dando a relucir su versatilidad cuando sean empleadas.

- La tarjeta escudo da al alumno o docente un instrumento poderoso para el aprendizaje y la enseñanza en la programación e implementación de aplicaciones basadas en microcontroladores PIC, de tal forma que una vez completado su aprendizaje a través de aplicaciones propuestas, puedan a su vez implementar una serie de aplicaciones más específicas de acuerdo a las necesidades requeridas.

- El desarrollo de las tarjetas escudos ayudan a que el número de aplicaciones inicialmente desarrolladas para la plataforma hardware con PIC18F4550, se vayan enriqueciendo sin necesidad de realizar cambios en la tarjeta principal, eso es posible lograrlo gracias a la gran modularidad que presenta el sistema de cada tarjeta escudo.

- Todos los programas desarrollados para las tarjetas escudos fueron elaborados en el compilador CCS C COMPILER; donde el propio compilador tiene un gran número de librerías para el manejo de un sin fin de dispositivos externos como internos del microcontrolador PIC.

- Se crearon librerías que servirán de apoyo complementario en el manejo de los módulos implementados para el manejo de las tarjetas escudos.

- Utilizar el lenguaje ensamblador para los rangos de gama baja, media y alta por versatilidad y mejor aprovechamiento de recursos del microcontrolador PIC.

- El lenguaje C es un lenguaje de alto nivel y puede resultar muy útil al combinarlo con el lenguaje ensamblador del CCS C COMPILER, ya que nos puede ahorrar tiempo de programación, depuración y simulación sobre todo en microcontroladores de gama alta y en los dsPIC.

- Las tarjetas escudos diseñadas en el presente proyecto son programables y pueden interactuar con cualquier programa de instrumentación; como por ejemplo LABVIEW, comparadas con otras tarjetas escudos de adquisición de datos que funcionan solamente con el programa del fabricante.

- Los resultados de costos en la elaboración y diseño de las tarjetas, fueron muy económicos, como se lo planeo antes de comenzar el proyecto de tesis.

RECOMENDACIONES

- Evite trabajar en ambientes muy húmedos o con peligro que las tarjetas entren en contacto con el agua, y tener en cuenta los límites de temperatura de operación y almacenamiento de cada tarjeta para evitar daños.

- Revise los esquemáticos y demás ayudas que identifique las funciones y conexiones de cada tarjeta escudo, para poder evitar cortocircuitos que puedan dañar los módulos de las tarjetas u otros dispositivos.

- Programar siguiendo las instrucciones y lineamientos establecidos en el documento base, ya que en él se encuentran los parámetros necesarios para una correcta programación de las tarjetas.

- Antes de realizar la conexión en tiempo real de las prácticas en las tarjetas escudos, es preferible simular previamente en el computador.

- Respetar los rangos máximos de corriente y voltaje para cada tarjeta escudo.

- Retirar con cuidado las conexiones realizadas entre las tarjetas escudos y la plataforma base, con el fin de evitar daños en los pines.

BIBLIOGRAFIA

[1] José María Angulo, Susana Romero, Ignacio Angulo. (2010) **MICROCONTROLADORES PIC / PIC16F87X – PIC18FXXX**. (2da. Ed.) Madrid, España. McGraw-Hill.

[2] Omar Barra Zapata, Franklin Barra Zapata. (2011) **MICROCONTROLADORES PIC CON PROGRAMACION Pic Basic Pro**. (8va. Ed.). Barcelona, España. McGraw-Hill.

[3] José María Angulo, Iván Trueba, Ignacio Angulo. (2009). **MICROCONTROLADORES DSPIC / DISEÑO PRACTICO DE APLICACIONES**. (4ta. Ed.). Madrid, España. McGraw-Hill.

[4] Eduardo García Breijo. (2008). **COMPILADOR C CCS Y SIMULADOR PROTEUS PARA MICROCONTROLADORES PIC.** (1ra. Ed.) México DF, México. Algaomega.

[5] Martin Bates. (2006). **INTERFACING PIC MICROCONTROLLERS, EMBEDDED DESIGN BY INTERACTIVE SIMULATION.** (1era. Ed.) Gran Bretaña. Newnes.

[6] Tim Wilmshurst. (2007). **DESIGNING EMBEDDED SYSTEMS WITCH PIC MICROCONTROLLERS, PRINCIPLES AND APPLICATIONS.** (1ra. Ed.) Gran Bretaña. Newnes.

[7] Horacio Vallejo. (1998). **MICROCONTROLADORES PIC – APRENDA UNA PROFESION**. (1ra. Ed.) Madrid, España. Quarks S.R.L Editorial.

[8] Mikel Etxebarria Isuskiza. (2011). **MICROCONTROLADORES PIC – TEORIA Y PRACTICA.** (1ra. Ed.) Madrid, España. Creaciones Copyright.

[9] Andrés Curtidor, Camilo Herrera. (2012). **DISEÑO Y CONSTRUCCION DE MODULOS ENTRENADORES PARA MICROCONTROLADORES**. (1ra. Ed.) Madrid, España. Editorial Académica Española.

[10] F. Valdés, R. Pallas. (2007). **MICROCONTROLADORES: FUNDAMENTOS Y APLICACIONES CON PIC**. (1ra. Ed.) Barcelona, España. Marcombo S.A.

[11] G. Tojeiro Calaza. (2009). **SIMULACION DE CIRCUITOS ELECTRÓNICOS Y MICROCONTROLADORES A TRAVÉS DE EJEMPLOS.** (1ra. Ed.) Barcelona, España. Editorial Marcombo.

[12] Alda Mazarredo (2007). **INGENIERÍA DE MICROSISTEMAS PROGRAMADOS.** (Vol. 1) Bilbao, España.

[13] José María Angulo, Ignacio Angulo. (2005). **MICROCONTROLADORES PIC / DISEÑO PRACTICO DE APLICACIONES.** (1ra. Ed.). Madrid, España. McGraw-Hill.

[14] Andrés Saravia, Ariel Coria. (2010). **ARQUITECTURA Y PROGRAMACIÓN DE MICROCONTROLADORES PIC**. (2da. Ed.). Buenos Aires, Argentina. El Autor.

[15] Juan González de la Rosa. (2001). **CIRCUITOS ELECTRÓNICOS CON AMPLIFICADORES OPERACIONALES**. (1ra. Ed.). Barcelona, España. Marcombo S.A

[16] Manuel Sierra Pérez. (2003). **ELECTRÓNICA DE COMUNICACIONES** (1ra. Ed.) Madrid, España. Pearson Alhambra.

[17] R. de Castro Fernández. (2008). **ELECTRÓNICA FORMULARIOS TÉCNICOS Y CIENTÍFICOS**. (1ra. Ed.). Madrid, España. Bellisco.

[18] Allan Hambley. (2001). **ELECTRÓNICA.** (1ra. Ed.). Madrid, España. Pearson Alhambra.

[19] Harprit Singh Sandhu. (2008). **MAKING PIC MICROCONTROLLER INSTRUMENTES & CONTROLLERS**. (1ra. Ed.). Estados Unidos. McGraw-Hill.

[20] Jhon Lovine. (2004). **PIC MICROCONTROLLER PROJECT BOOK FOR PIC**. (2da. Ed.). Estados Unidos. McGraw-Hill.

[21] Eduardo Mendoza. (2007). **TESIS DE APLICACIÓN DE LENGUAJE PARA MICROCONTROLADORES PIC Y VISUAL C# EN EL DESARROLLO DE UN SISTEMA DE CONTROL DE ACCESO PEATONAL.** Guayaquil. UPS.

BIBLIOGRAFÍA ELECTRÓNICA:

- www.dontronics.com
- www.ccsinfo.com
- www.dynamoelectronics.com
- www.arv.umh.es
- www.impronic.com
- www.digitentric.com
- www.dspace.espoch.edu.ec
- www.dspace.espol.edu.ec
- www.pcb.electrosoft.cl
- www.ccsinfo.com/forum/viewtopic.php?t=39388
- www.robots101.com/wp-content/uploads/2011/09/kontrol.c

- www.ccsinfo.com/forum/viewtopic.php?p=131084
- www.bricogeek.com/shop/148-controlador-de-servomotores-micro-serial.html
- www.es.scribd.com/doc/206918459/Articulo-SAAEI-Plataforma-uC
- www.upcommons.upc.edu/eprints/bitstream/2117/14420/1/Art%C3%ADculo%20SAAEI%20-%20Plataforma%20uC.pdf

ANEXOS

MANUAL PARA CREAR PROYECTOS EN PIC C COMPILER

Se presentan a continuación los pasos para crear un proyecto en PIC C CCS.

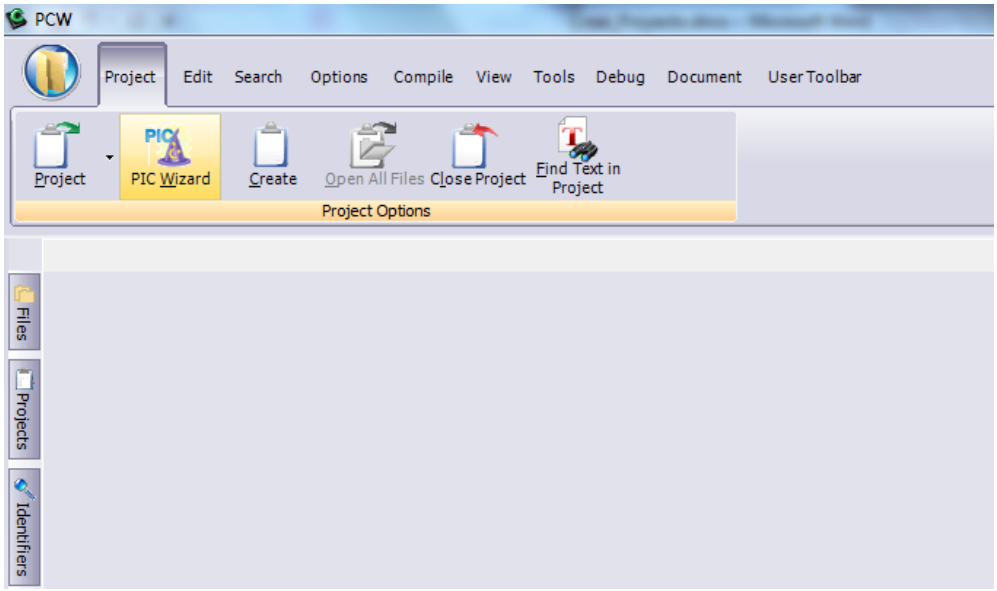
OBJETIVOS.-

- Crear un proyecto para el desarrollo de código embebido orientado a microcontroladores, para el desarrollo de prácticas y analizar su comportamiento.

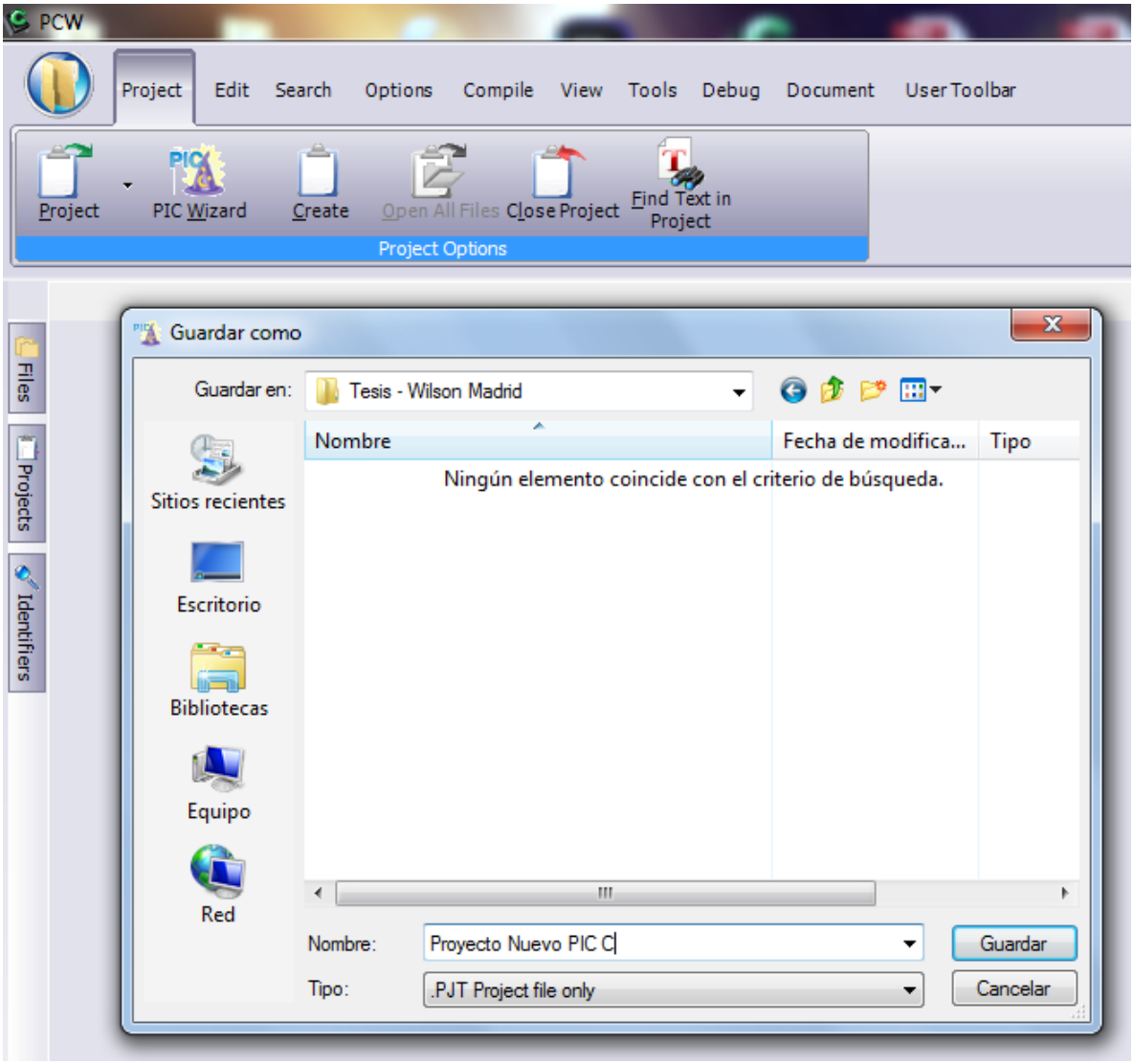
MODO DE USO.-

1. Ejecutar el ícono de PIC C CCS.

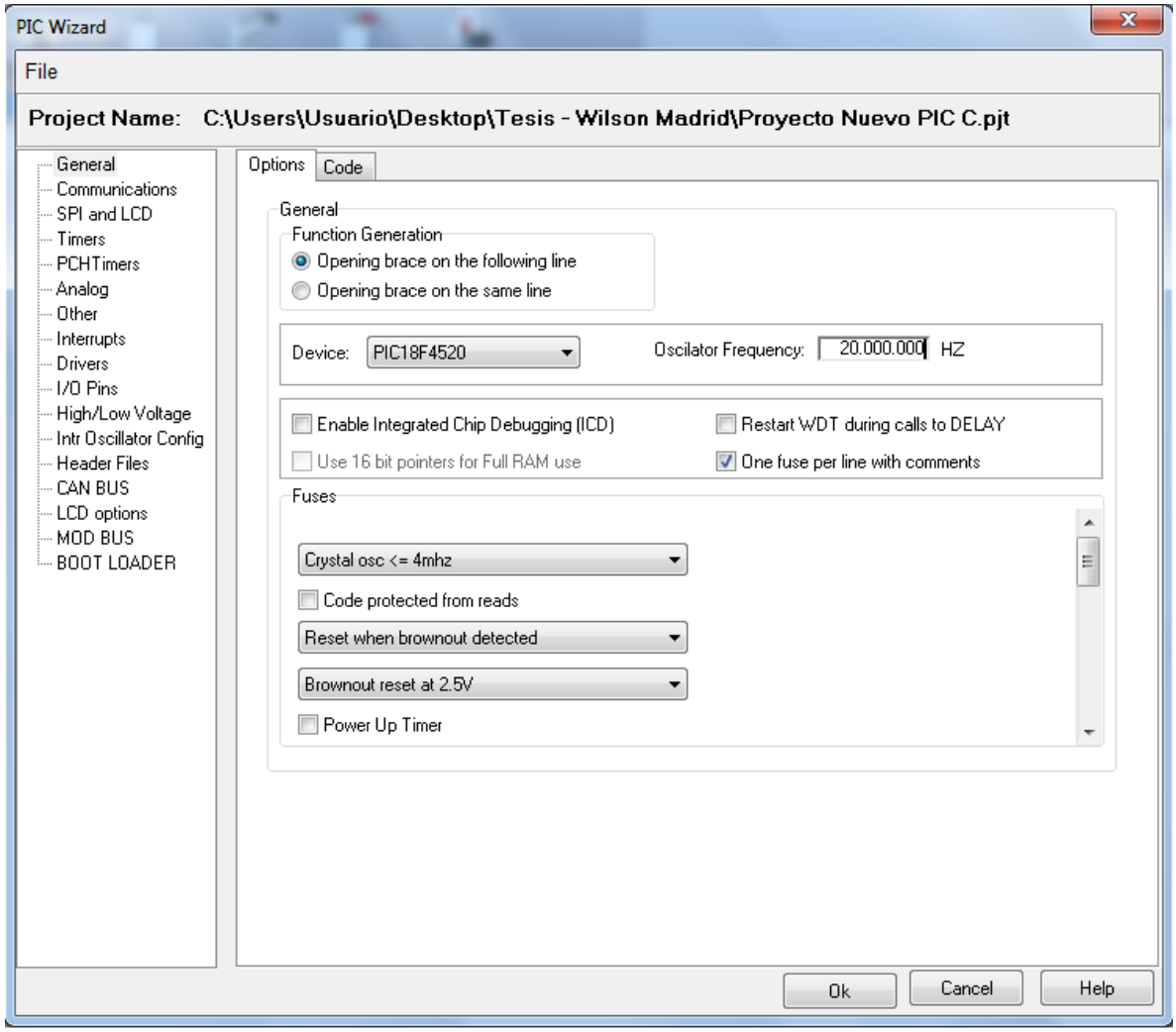
2. Seleccionar ***Project.***
3. Seleccionar ***Pic Wizard.***



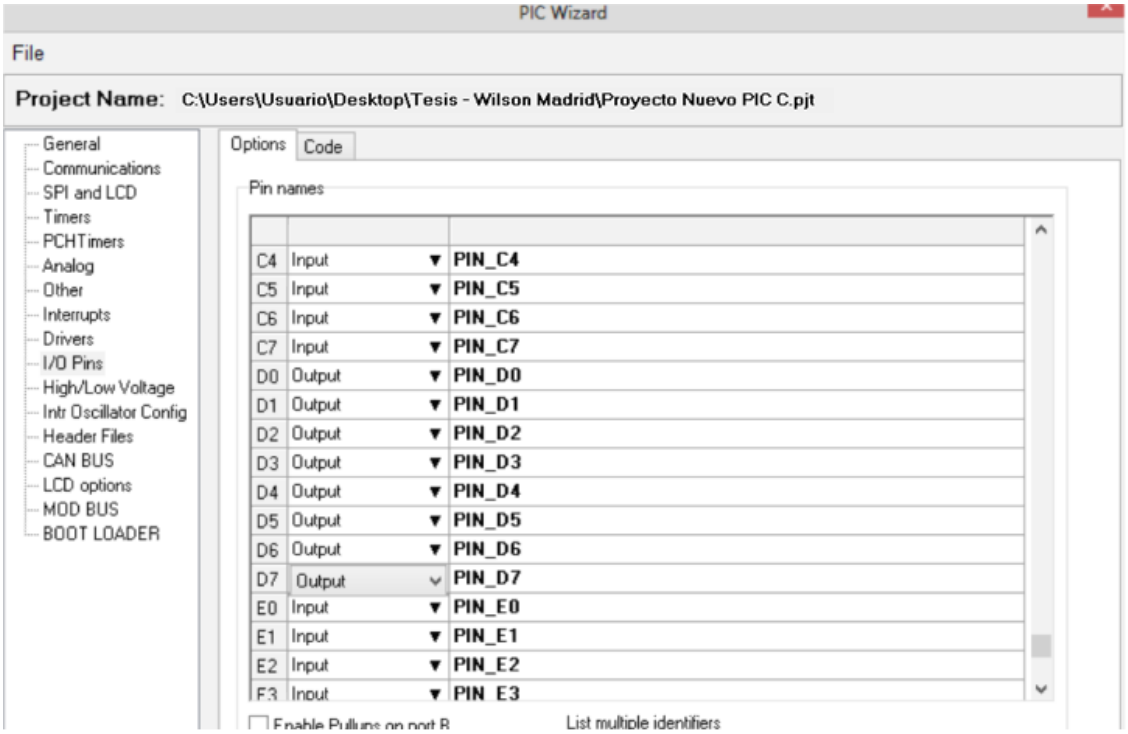
4. Crear una carpeta donde se van a guardar los códigos.
5. Escribir el nombre que se desea para el proyecto.



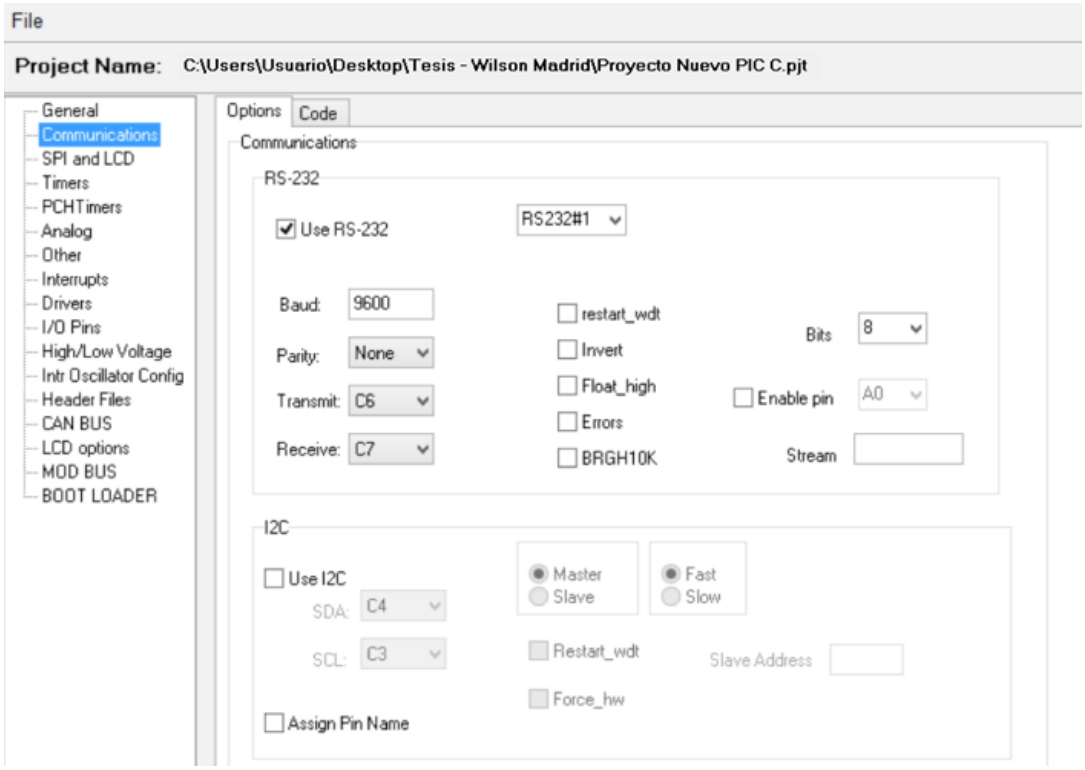
6. Ventana de configuración inicial.
7. Configurar el PIC a utilizar y la frecuencia de trabajo.



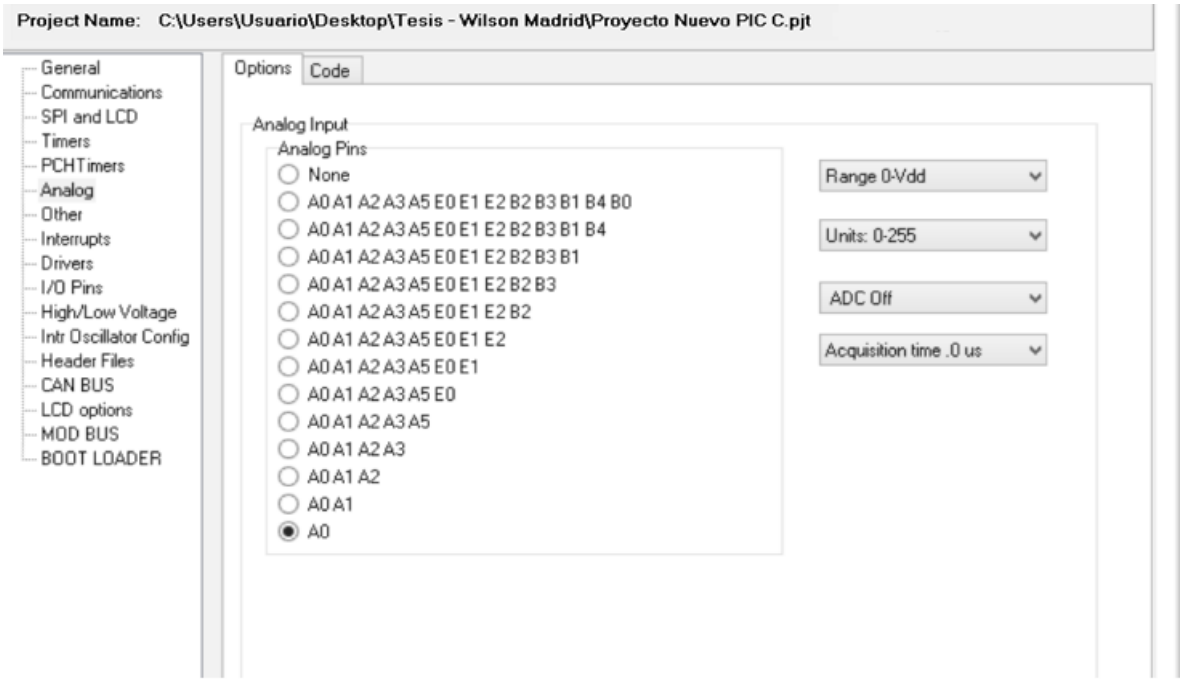
8. Seleccionar en el menú de la izquierda la opción ***I/O Pins.***
9. Configurar como salida los pines desde ***D0*** hasta ***D7*** para el encendido de LED´s.



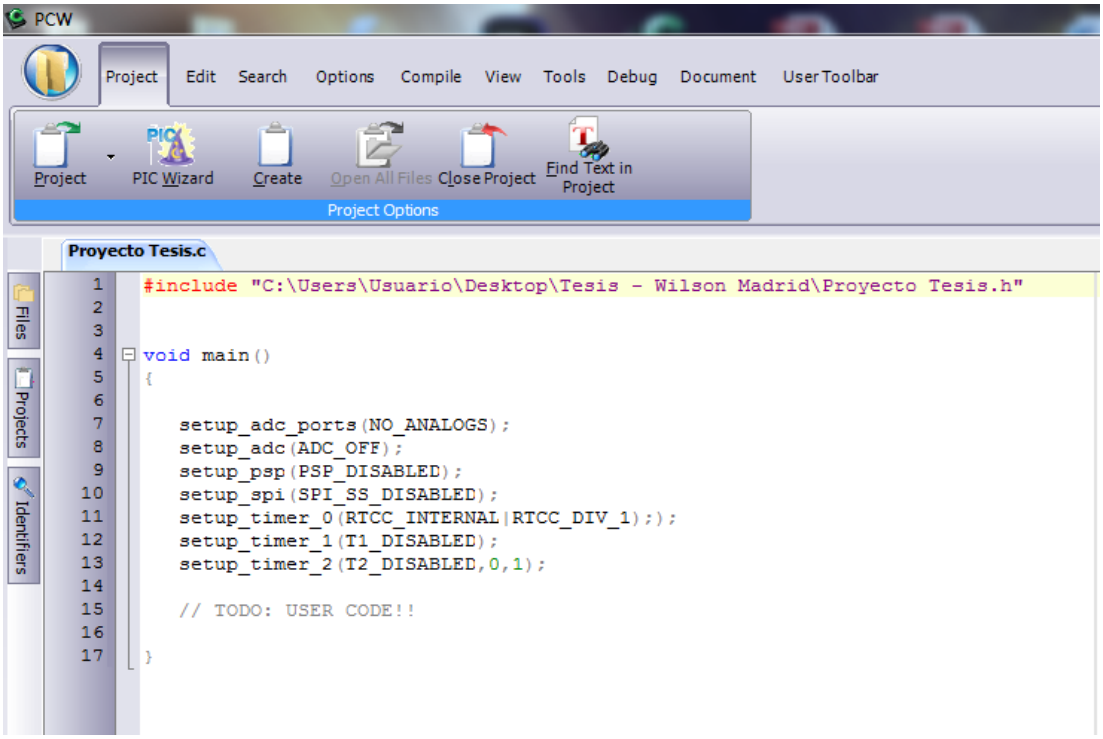
10. Seleccionar en el menú de la izquierda la opción ***Comunications.***
11. Configurar el PIC para comunicación ***Serial UART.***



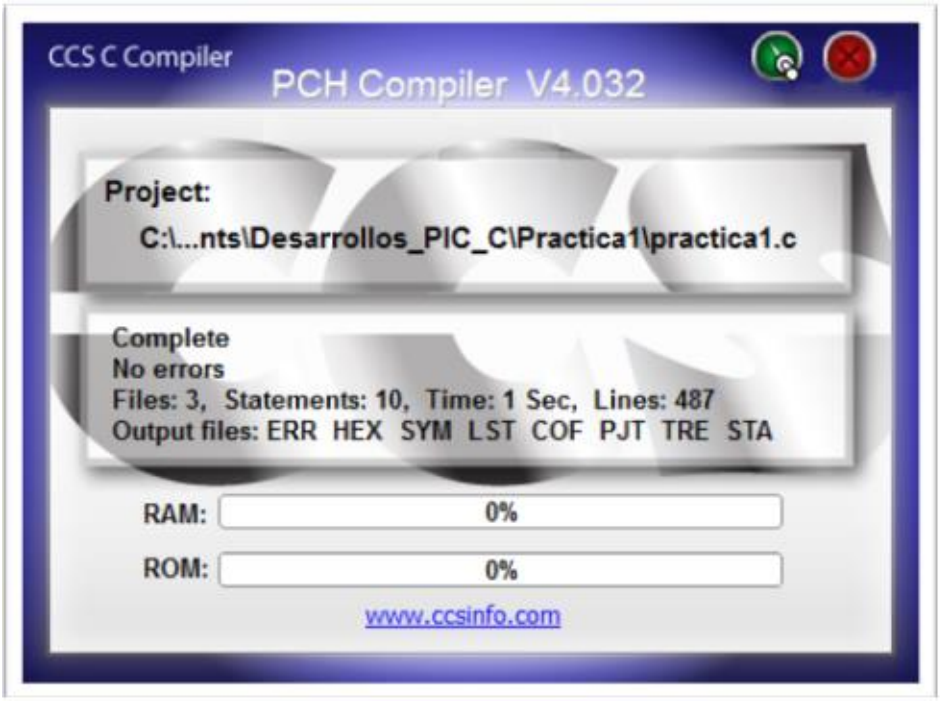
12. Seleccionar en el menú de la izquierda la opción ***Analog.***
13. Configurar la conversión ***Analógica/Digital A0*.**



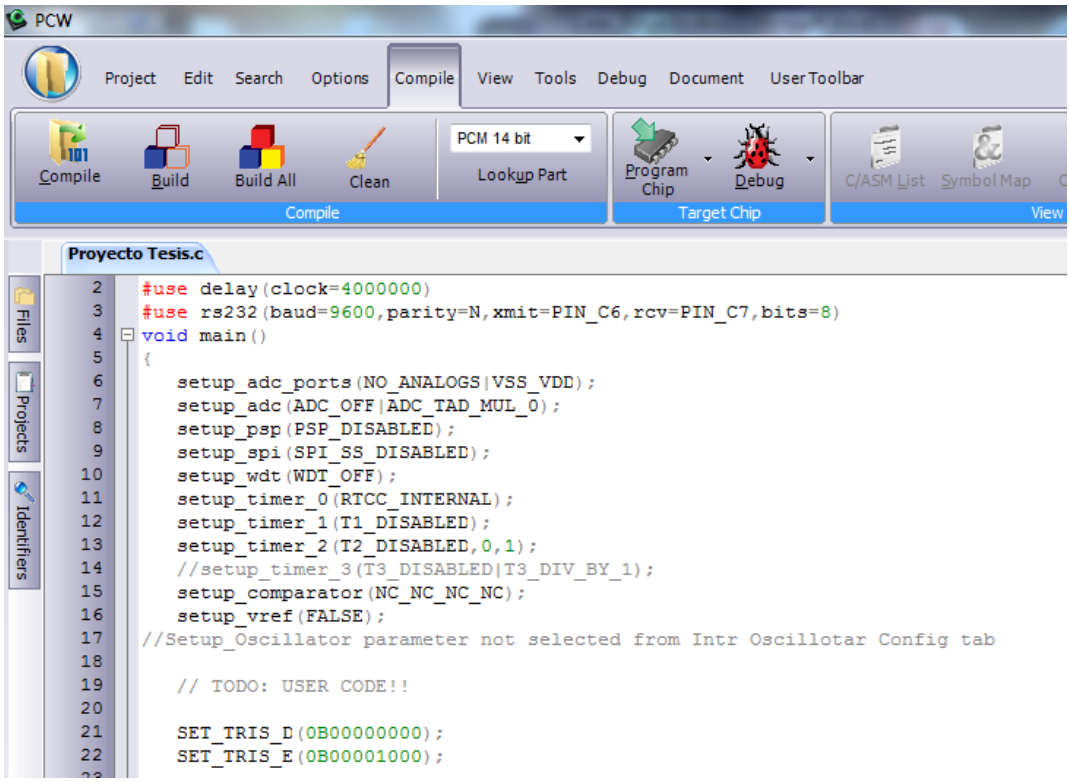
14. Ventana Principal del Archivo extensión ***.C.***



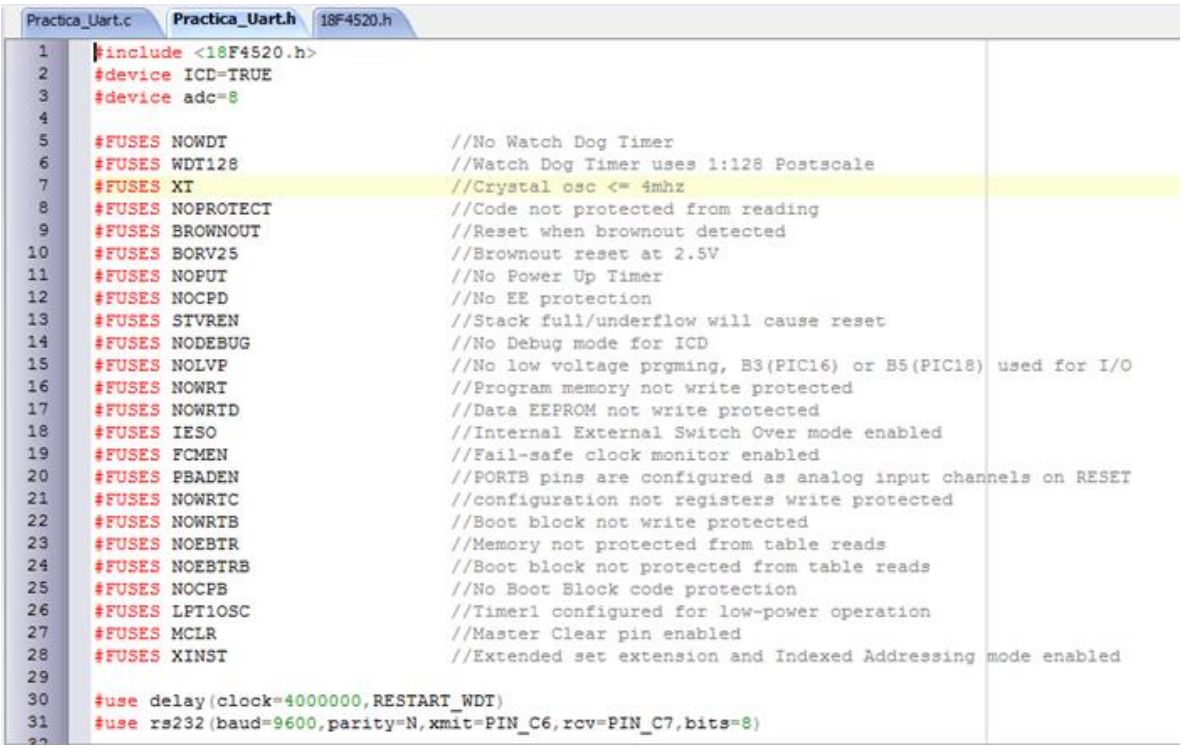
15. Seleccionar en el menú principal la opción ***Compile.***



16. Ventana lista para el inicio de desarrollo de código.



17. Ventana Principal del archivo con extensión ***.H.***

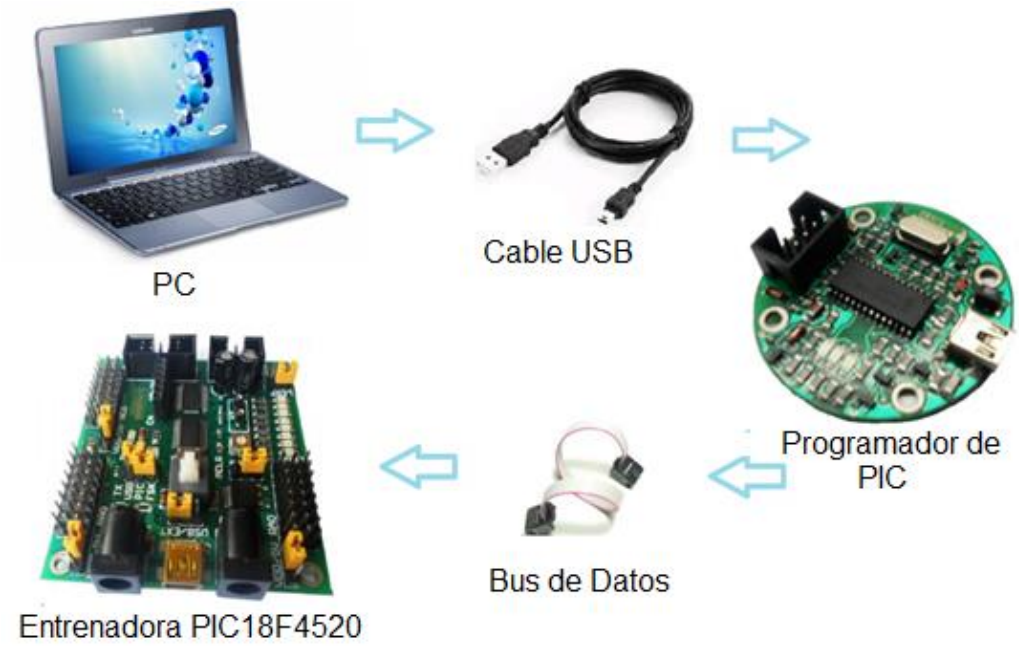



```
#include <18F4520.h>
#device ICD=TRUE
#device adc=8

#FUSES NOWDT                    //No Watch Dog Timer
#FUSES WDT128                   //Watch Dog Timer uses 1:128 Postscale
#FUSES XT                       //Crystal osc <= 4mhz
#FUSES NOPROTECT                //Code not protected from reading
#FUSES BROWNOUT                 //Reset when brownout detected
#FUSES BORV25                   //Brownout reset at 2.5V
#FUSES NOPUT                    //No Power Up Timer
#FUSES NOCPD                    //No EE protection
#FUSES STVREN                   //Stack full/underflow will cause reset
#FUSES NODEBUG                  //No Debug mode for ICD
#FUSES NOLVP                    //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES NOWRT                    //Program memory not write protected
#FUSES NOWRTD                   //Data EEPROM not write protected
#FUSES IESO                     //Internal External Switch Over mode enabled
#FUSES FCMEN                    //Fail-safe clock monitor enabled
#FUSES PBADEN                   //PORTB pins are configured as analog input channels on RESET
#FUSES NOWRTC                   //configuration not registers write protected
#FUSES NOWRTB                   //Boot block not write protected
#FUSES NOEBTR                   //Memory not protected from table reads
#FUSES NOEBTRB                  //Boot block not protected from table reads
#FUSES NOCPB                    //No Boot Block code protection
#FUSES LPT1OSC                  //Timer1 configured for low-power operation
#FUSES MCLR                     //Master Clear pin enabled
#FUSES XINST                    //Extended set extension and Indexed Addressing mode enabled

#use delay(clock=4000000,RESTART_WDT)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)
```

MANUAL PARA EL USO DE SOFTWARE PICKIT2

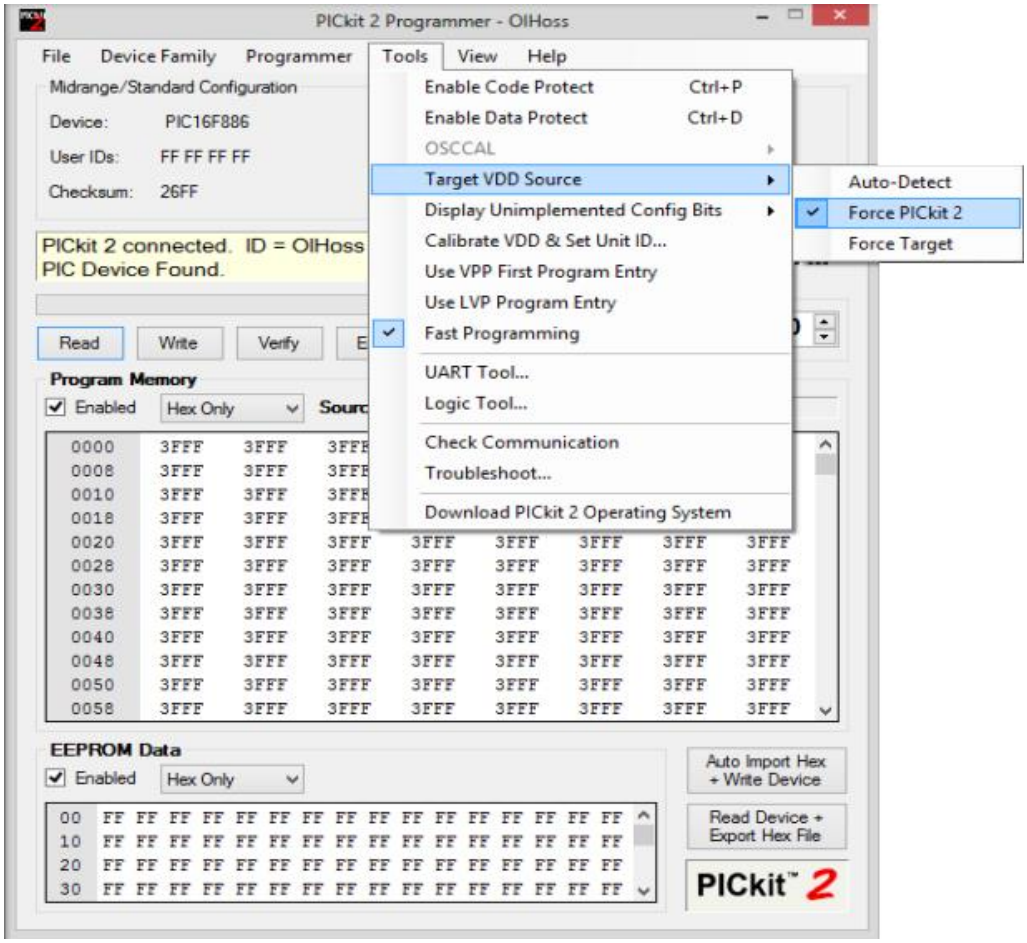
Se presentan a continuación los pasos para poder usar el PICKIT2.

1. Conectar el hardware, como se detalla en el siguiente gráfico.

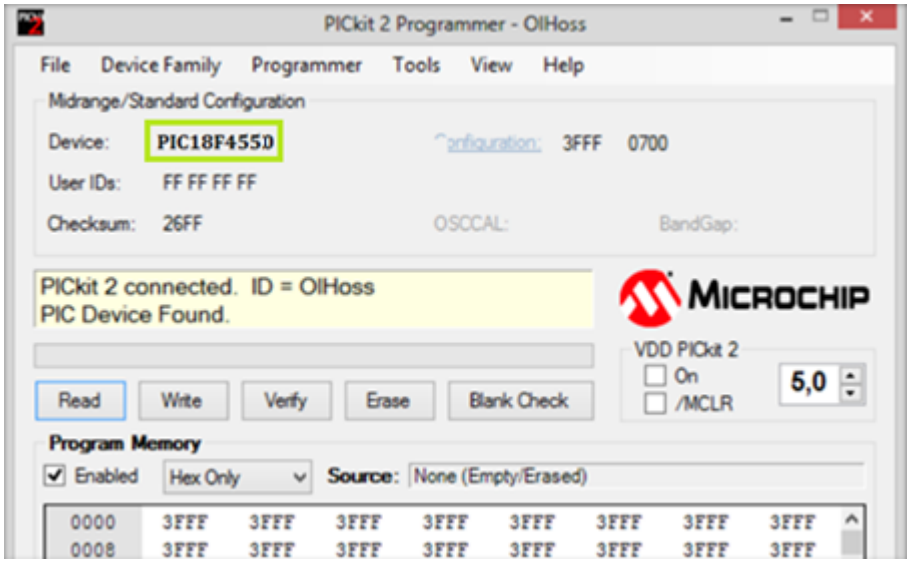


2. Ejecutar el ícono de Pickit2.

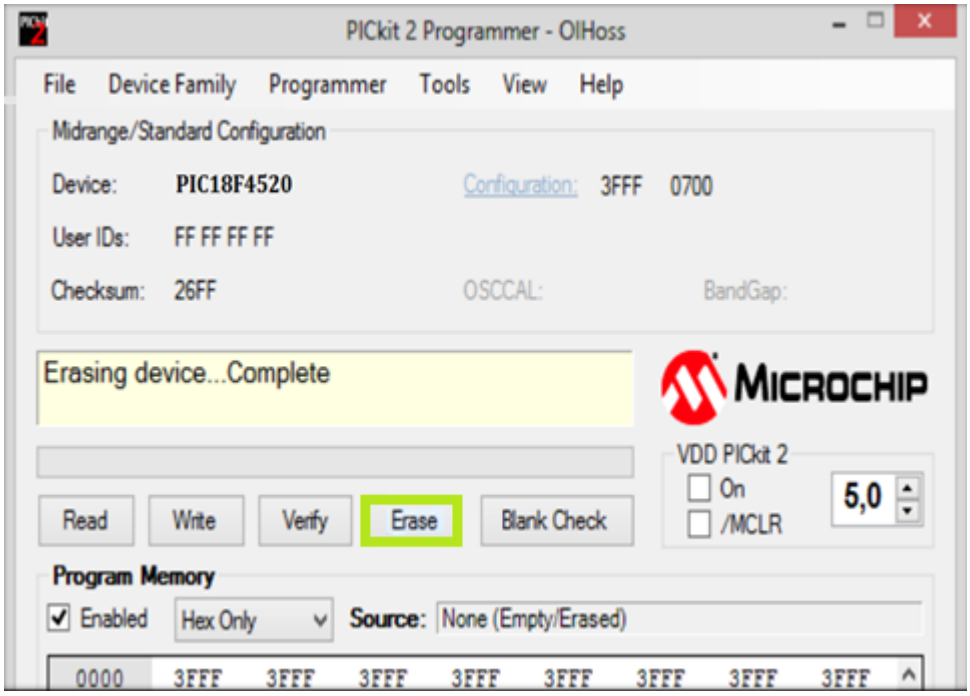
3. Seleccionar en el menú principal la opción ***Tools.***
4. Seleccionar ***Target VDD Source.***
5. Seleccionar ***Force PICkit2.***



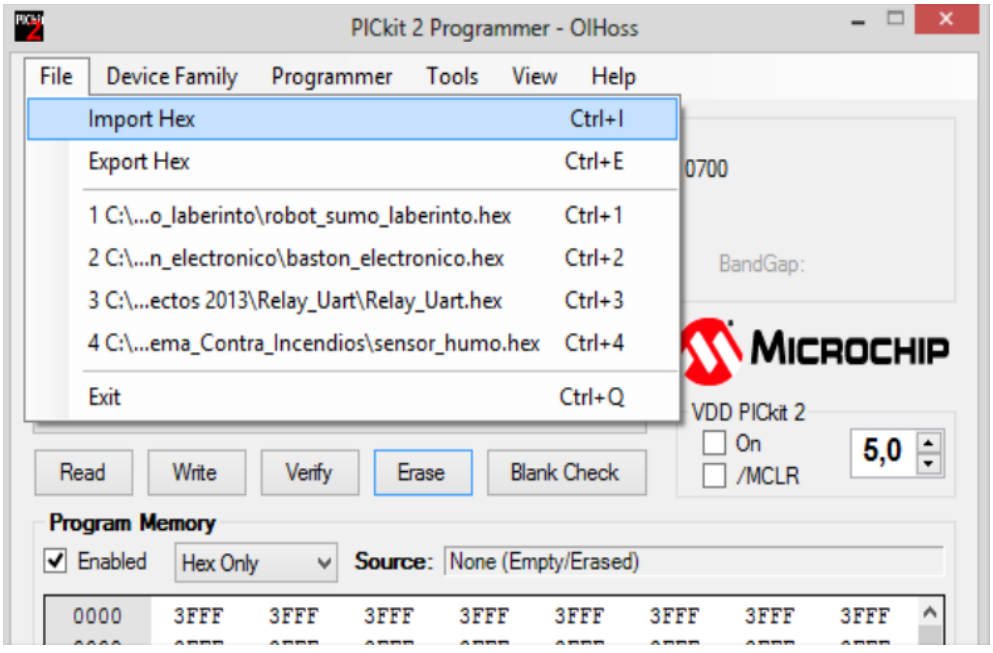
6. Observar que el dispositivo haya sido detectado, en este caso es el PIC18F4550.



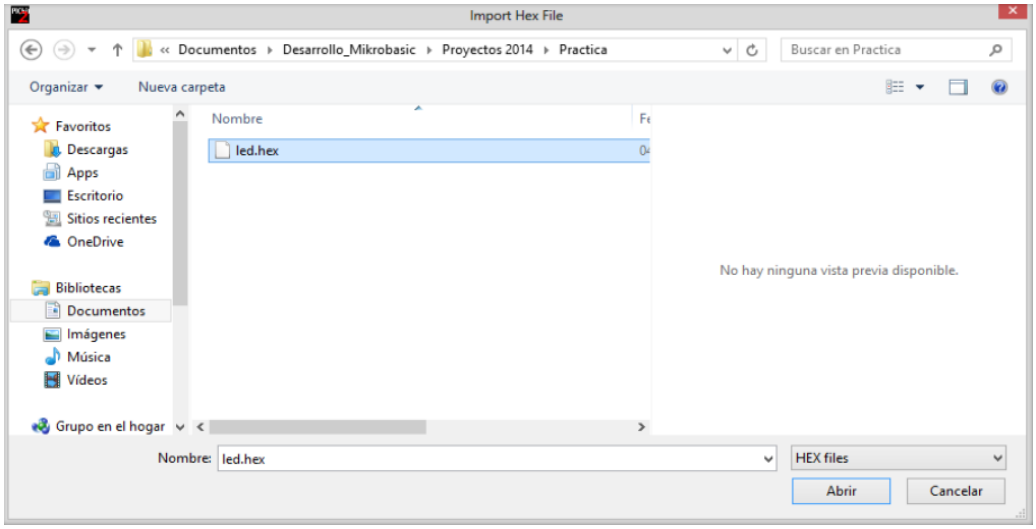
7. Seleccionar la opción ***Erase.***

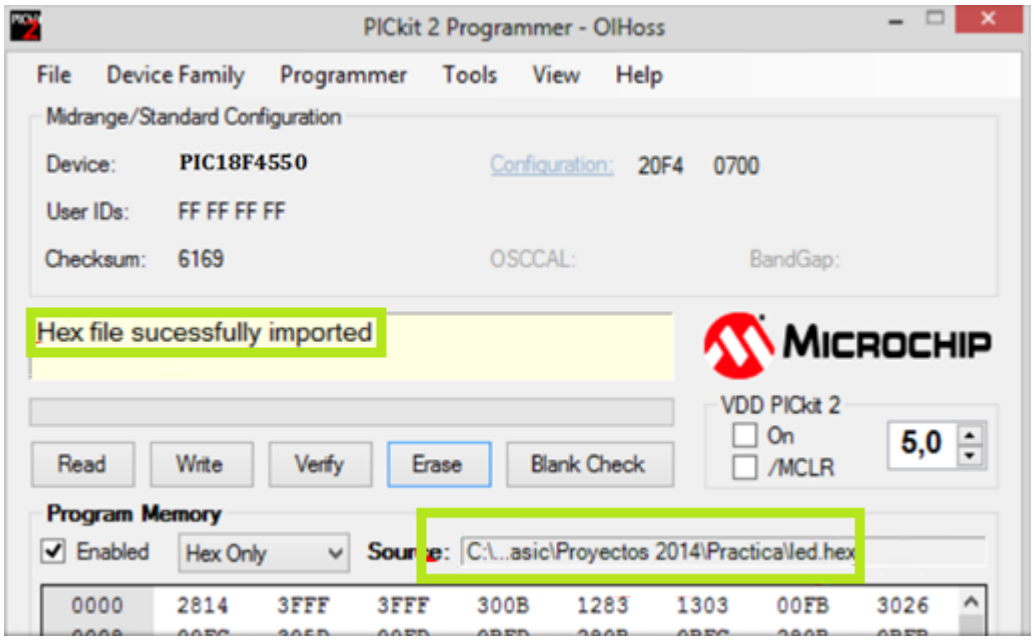


8. Seleccionar en el menú la opción ***File.***
9. Seleccionar la opción ***Import Hex.***

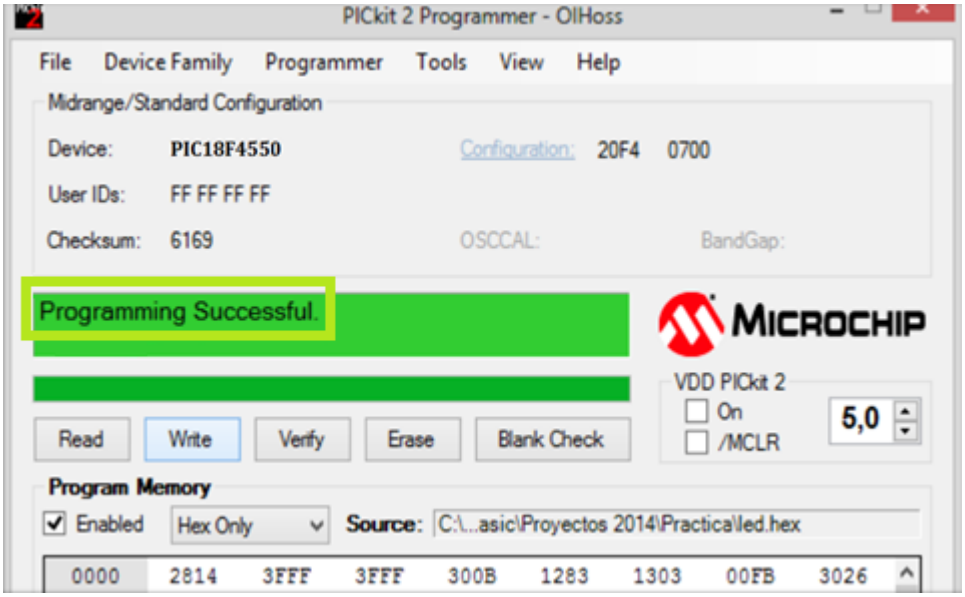


10. Buscar el archivo .HEX en la carpeta donde se creo el proyecto cuando se utilizó el programa PIC C CCS, para que se puede realizar la importación del archivo.





11. Seleccionar la opción ***Write*** para que el código de programa sea escrito en el microcontrolador.


PICkit 2 Programmer - OlHoss
File
Device Family
Programmer
Tools
View
Help
Midrange/Standard Configuration
Device: PIC18F4550
Configuration: 20F4 0700
User IDs: FF FF FF FF
Checksum: 6169
OSCCAL:
BandGap:
Programming Successful.
MICROCHIP
VDD PICkit 2
On
/MCLR
5,0
Read
Write
Verify
Erase
Blank Check
Program Memory
Enabled
Hex Only
Source: C:\...asic\Proyectos 2014\Practica\led.hex
0000 2814 3FFF 3FFF 300B 1283 1303 00FB 3026

MANUAL PARA USO DE SOFTWARE ACCESSPORT

AccessPort es un software que permite comunicación con dispositivos por medio del protocolo serial UART.

OBJETIVOS.-

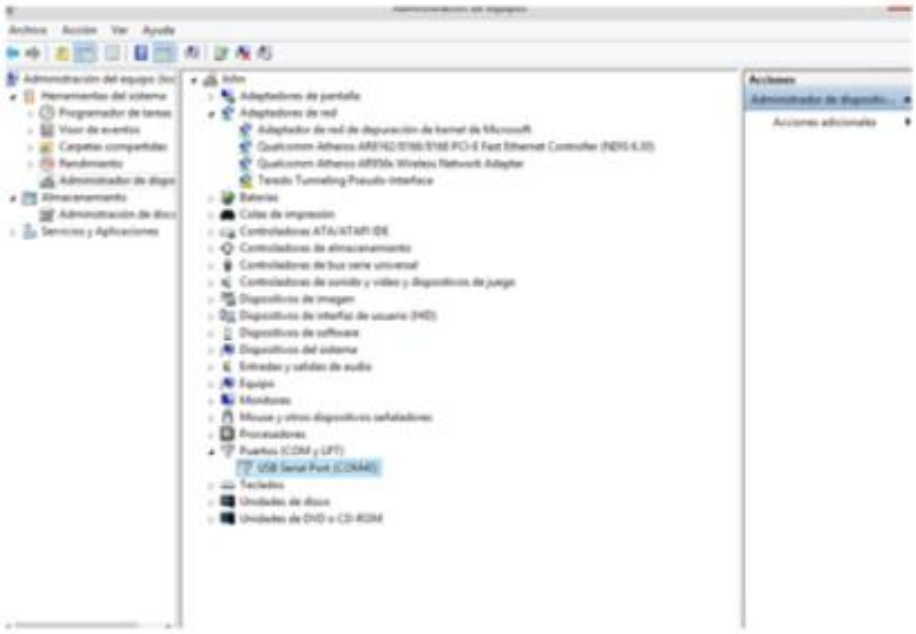
- Manipular la ventana de AccessPort para la comunicación Serial UART.
- Identificar la ventana de envío de datos.
- Identificar la ventana de recepción de datos.
- Identificar el menú de configuración de Puerto COM.
- Identificar el menú de configuración BAUDRATE.

DIAGRAMA ESQUEMÁTICO.-

MODO DE USO.-

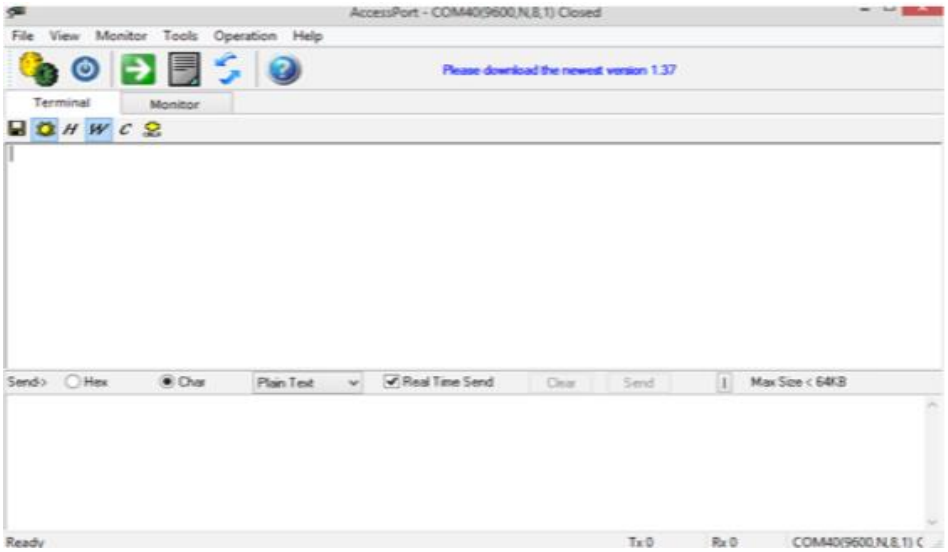
1. Conectar la tarjeta de control al Computador por el puerto USB.

2. Ingresar al Administrador de Dispositivos y verificar el Puerto COM asignado.

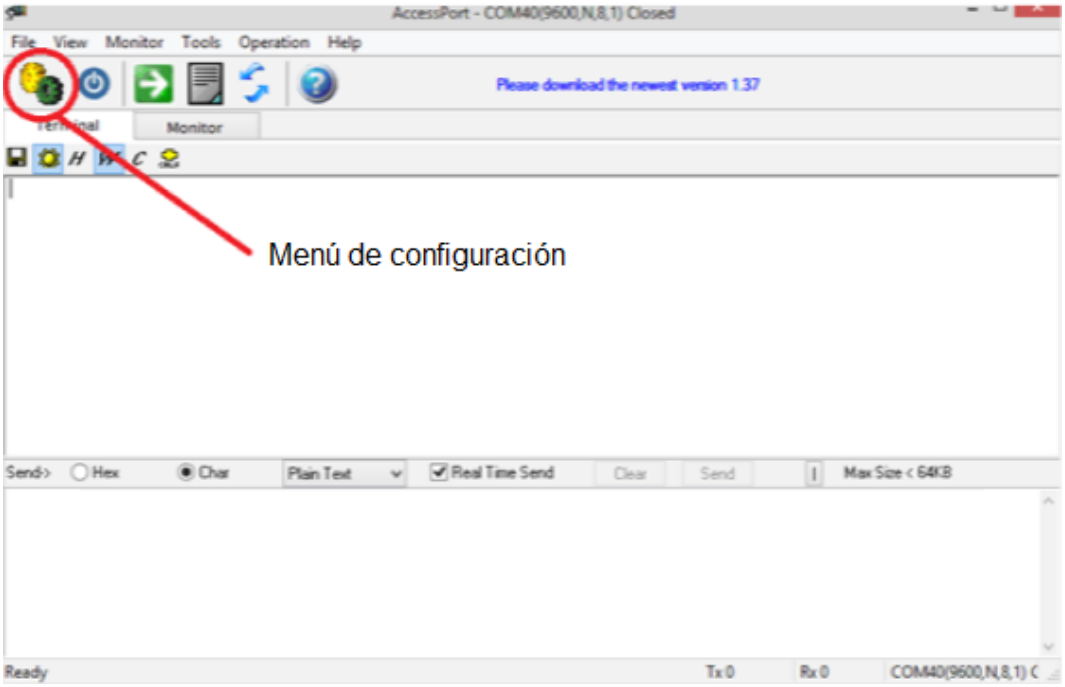


3. Ejecutar el ícono de AccessPort.

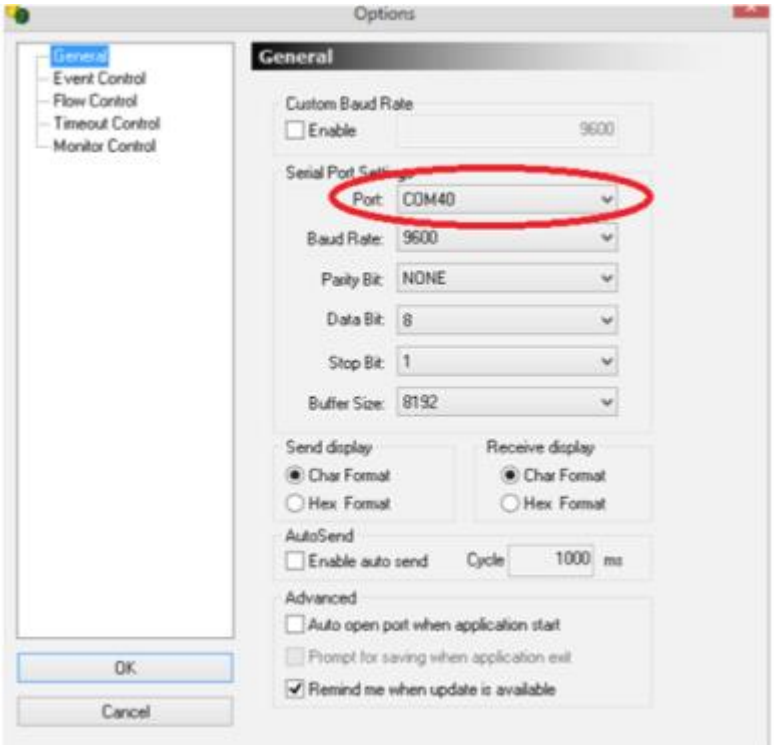
4. Ventana principal de configuración.



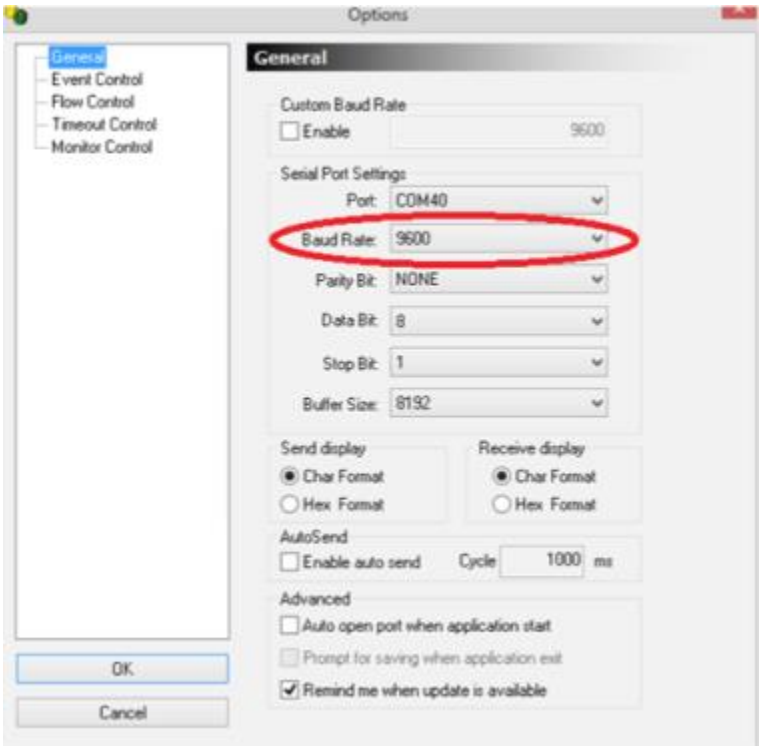
5. Ingresar al menú de configuración.



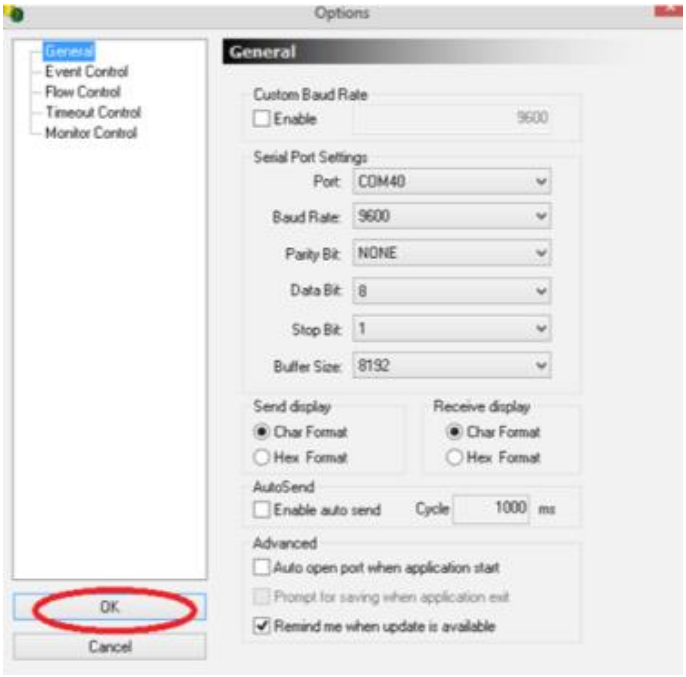
6. Configurar el Puerto COM asignado en el Administrador de Dispositivos



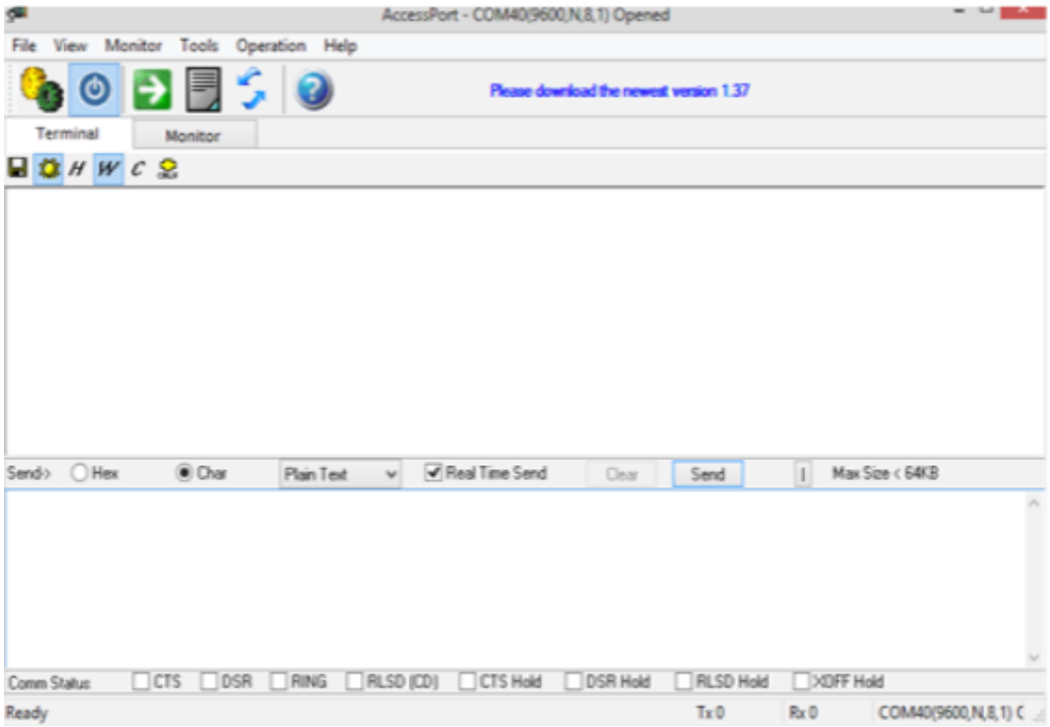
7. Configurar la tasa de transmisión de BITS según lo programado “9600”.



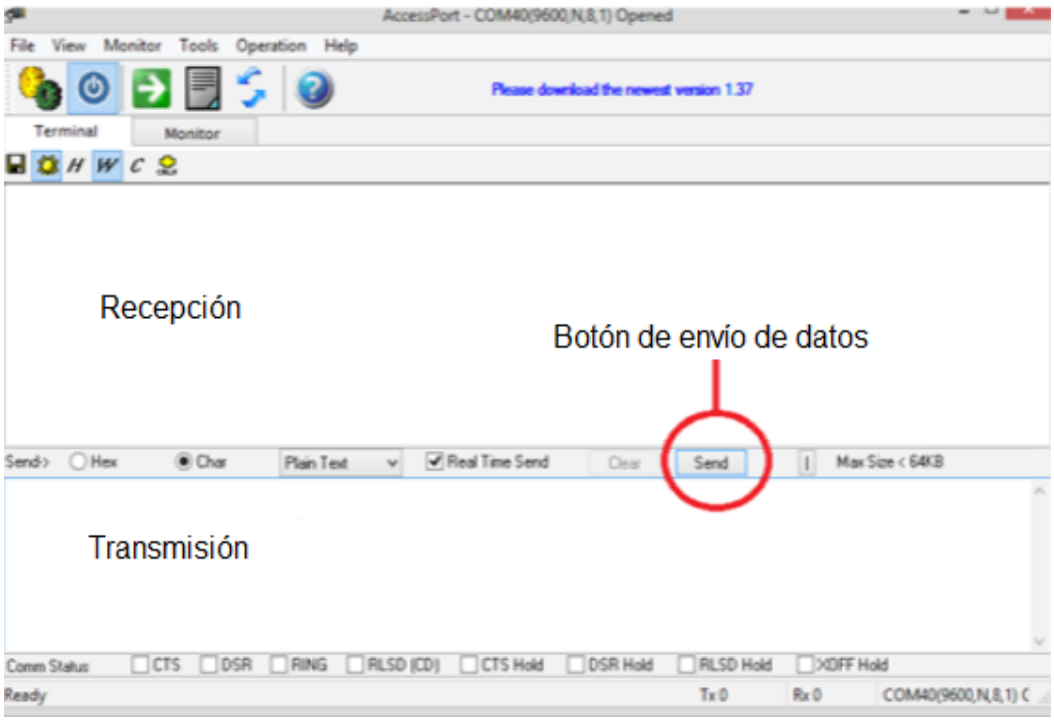
8. Seleccionar la opción ***OK.***



9. Cuando el enlace se realiza con éxito, es indicado por ***BOTON*** *Azul.*



NOTA: No olvidar las Partes importantes de la ventana principal.



PRÁCTICA # 1

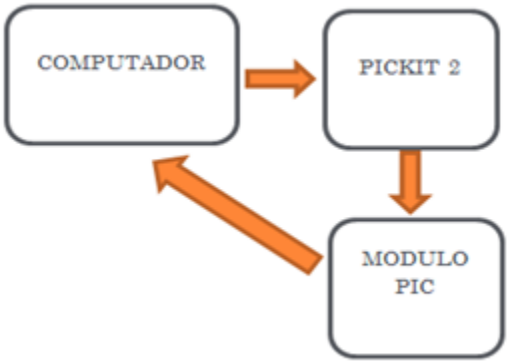
TEMA: COMUNICACIÓN UART

Esta práctica se dedica a dar una introducción para el uso del Protocolo de Comunicación UART de la Tarjeta de Entrenamiento con PIC 18F4550. Una vez desarrollado el código de ejemplo, el alumno debe realizar cambios en el software de control, para verificar lo aprendido.

OBJETIVOS.-

- Crear un proyecto con el software de control PIC C COMPILER.
- Crear un código para el control del Módulo de Entrenamiento con PIC 18F4550.
- Programar la tarjeta de control con el software PICKIT 2.
- Programar la tarjeta con el Software de control inicial de la Practica 2.
- Realizar pruebas de laboratorio, mediante cambios en el algoritmo de control para consolidar los conocimientos adquiridos.
- Instalar en DRIVER USB-UART para comunicaciones serial Uart entre PIC y PC.
- Programar el Módulo con PIC 18f4550 para enviar mensajes al computador desde el PIC.

DIAGRAMA ESQUEMÁTICO.-



CÓDIGO DE PROGRAMACIÓN.-

```
#include
void main ()
{
    setup_adc_ports (NO_ANALOGS|VSS_VDD);
   setup_adc (ADC_OFF|ADC_TAD_MUL_0);
   setup_psp (PSP_DISABLED);
   setup_spi (SPI_SS_DISABLED);
   setup_wdt (WDT_OFF);
   setup_timer_0 (RTCC_INTERNAL);
   setup_timer_1 (T1_DISABLED);
   setup_timer_2 (T2_DISABLED,0,1);
   setup_comparator (NC_NC_NC_NC);
   setup_vref (FALSE);

while (1)
{
        Printf ("HOLA...");
        OUTPUT_D (0XFF);
        Delay_ms (500);
        OUTPUT_D (0X00);
        Delay_ms (500);
```

}

}

SOFTWARE DE CONTROL ARCHIVO .C.-

Practica_Uart.c | Practica_Uart.h | 18F4520.h

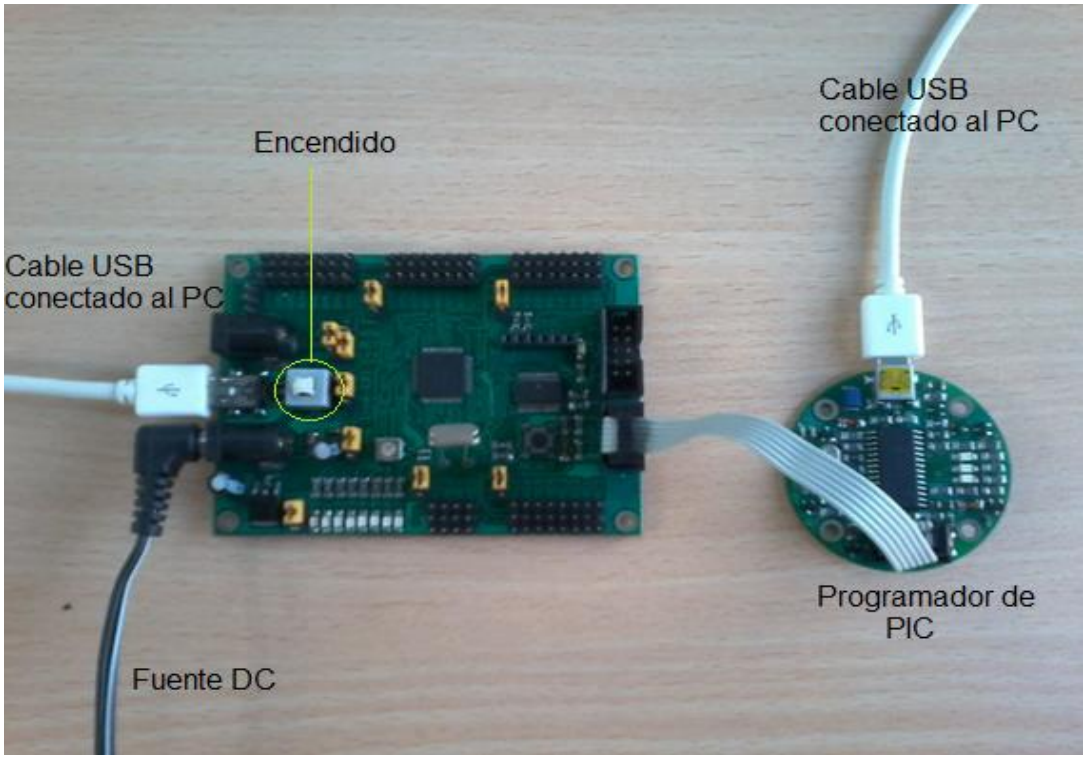
```
#include            "C:\Users\ Documents\Desarrollos_PIC_C\Practica_2\Practica_Uart\Practica_Uart.h"


void main()
{

   setup_adc_ports(NO_ANALOGS|VSS_VDD);
   setup_adc(ADC_OFF|ADC_TAD_MUL_0);
   setup_psp(PSP_DISABLED);
   setup_spi(SPI_SS_DISABLED);
   setup_wdt(WDT_OFF);
   setup_timer_0(RTCC_INTERNAL);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DISABLED,0,1);
   setup_comparator(NC_NC_NC_NC);
   setup_vref(FALSE);
//Setup_Oscillator parameter not selected from Intr Oscillotar Config tab

   // TODO: USER CODE!!
 while(1)
{
         printf("HOLA...");
         OUTPUT_D(0XFF);
         Delay_ms(500);
         OUTPUT_D(0X00);
         Delay_ms(500);
}

}
```

SOFTWARE DE CONTROL ARCHIVO .H.-

Practica_Uart.c | Practica_Uart.h | 18F4520.h

```
#include <18F4520.h>
#device ICD=TRUE
#device adc=8

#FUSES NOWDT                    //No Watch Dog Timer
#FUSES WDT128                   //Watch Dog Timer uses 1:128 Postscale
#FUSES XT                       //Crystal osc <= 4mhz
#FUSES NOPROTECT                //Code not protected from reading
#FUSES BROWNOUT                 //Reset when brownout detected
#FUSES BORV25                   //Brownout reset at 2.5V
#FUSES NOPUT                    //No Power Up Timer
#FUSES NOCPD                    //No EE protection
#FUSES STVREN                   //Stack full/underflow will cause reset
#FUSES NODEBUG                  //No Debug mode for ICD
#FUSES NOLVP                    //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES NOWRT                    //Program memory not write protected
#FUSES NOWRTD                   //Data EEPROM not write protected
#FUSES IESO                     //Internal External Switch Over mode enabled
#FUSES FCMEN                    //Fail-safe clock monitor enabled
#FUSES PBADEN                   //PORTB pins are configured as analog input channels on RESET
#FUSES NOWRTC                   //configuration not registers write protected
#FUSES NOWRTB                   //Boot block not write protected
#FUSES NOEBTR                   //Memory not protected from table reads
#FUSES NOEBTRB                  //Boot block not protected from table reads
#FUSES NOCPB                    //No Boot Block code protection
#FUSES LPT1OSC                  //Timer1 configured for low-power operation
#FUSES MCLR                     //Master Clear pin enabled
#FUSES XINST                    //Extended set extension and Indexed Addressing mode enabled

#use delay(clock=4000000,RESTART_WDT)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)
```

IMPLEMENTACIÓN.-



PRÁCTICA # 2

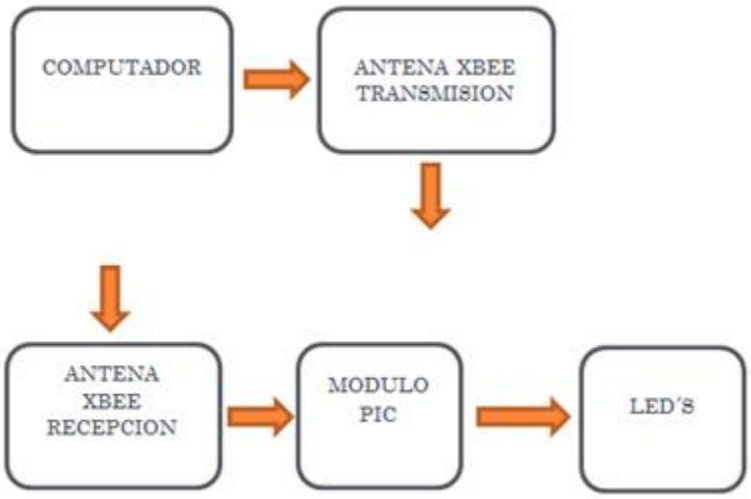
TEMA: RECEPCIÓN DE DATOS POR ANTENA XBEE

Esta práctica se dedica a dar una introducción para el encendido de LED´s de la Tarjeta de Entrenamiento con PIC 18F4520 por envío de datos de forma inalámbrica utilizando antena X-BEE. Una vez desarrollado el código de ejemplo, el alumno debe realizar cambios en el software de control, para verificar lo aprendido.

OBJETIVOS.-

- Crear un proyecto con el software de control PIC C COMPILER.
- Crear un código para el control del Módulo de Entrenamiento con PIC 18F4550.
- Programar la tarjeta de control con el software PICKIT 2.
- Programar la tarjeta con el Software de control inicial de la Practica 3.
- Realizar pruebas de laboratorio, mediante cambios en el algoritmo de control para consolidar los conocimientos adquiridos.
- Programar el Módulo con PIC 18f4550 para la recepción de datos de control de encendido.
- Programar el Módulo con PIC 18f4520 para el encendido de LED´s vía inalámbrica utilizando antena X_BEE por medio del software de Comunicación Serial AccessPort.

DIAGRAMA ESQUEMÁTICO.-

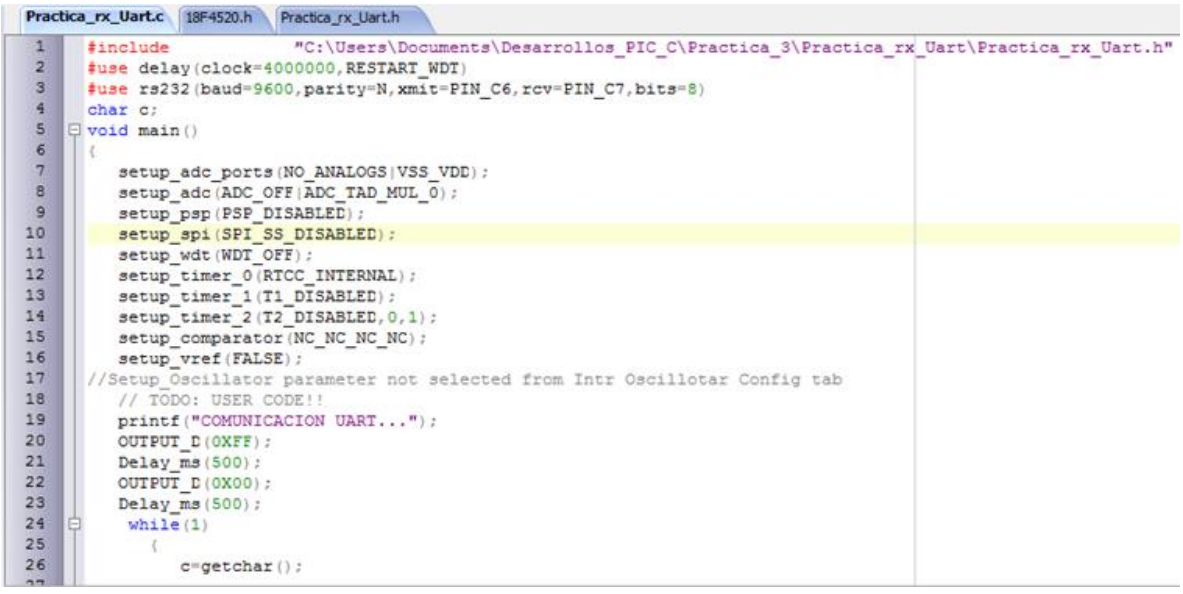


CÓDIGO DE PROGRAMACIÓN.-

```
#include
#use delay (clock=4000000, RESTART_WDT)
#use rs232 (baud=9600, parity=N, xmit=PIN_C6, rcv=PIN_C7, bits=8)
char c;
void main()
{
  setup_adc_ports (NO_ANALOGS|VSS_VDD);
  setup_adc (ADC_OFF|ADC_TAD_MUL_0);
  setup_psp (PSP_DISABLED);
  setup_spi (SPI_SS_DISABLED);
  setup_wdt (WDT_OFF);
  setup_timer_0 (RTCC_INTERNAL);
  setup_timer_1 (T1_DISABLED);
  setup_timer_2 (T2_DISABLED,0,1);
  setup_comparator (NC_NC_NC_NC);
  setup_vref (FALSE);

  printf ("COMUNICACION UART...");
  OUTPUT_D (0XFF);
```

```
   Delay_ms (500);
   OUTPUT_D (0X00);
   Delay_ms (500);
    while (1)
     {
        c = getchar ();
        if (c=='1')
        {
           OUTPUT_D (0XFF);
           Delay_ms (500);
           OUTPUT_D (0X00);
           Delay_ms (500);
        }
     }
}
```

SOFTWARE DE CONTROL ARCHIVO .C.-



SOFTWARE DE CONTROL ARCHIVO .H.-

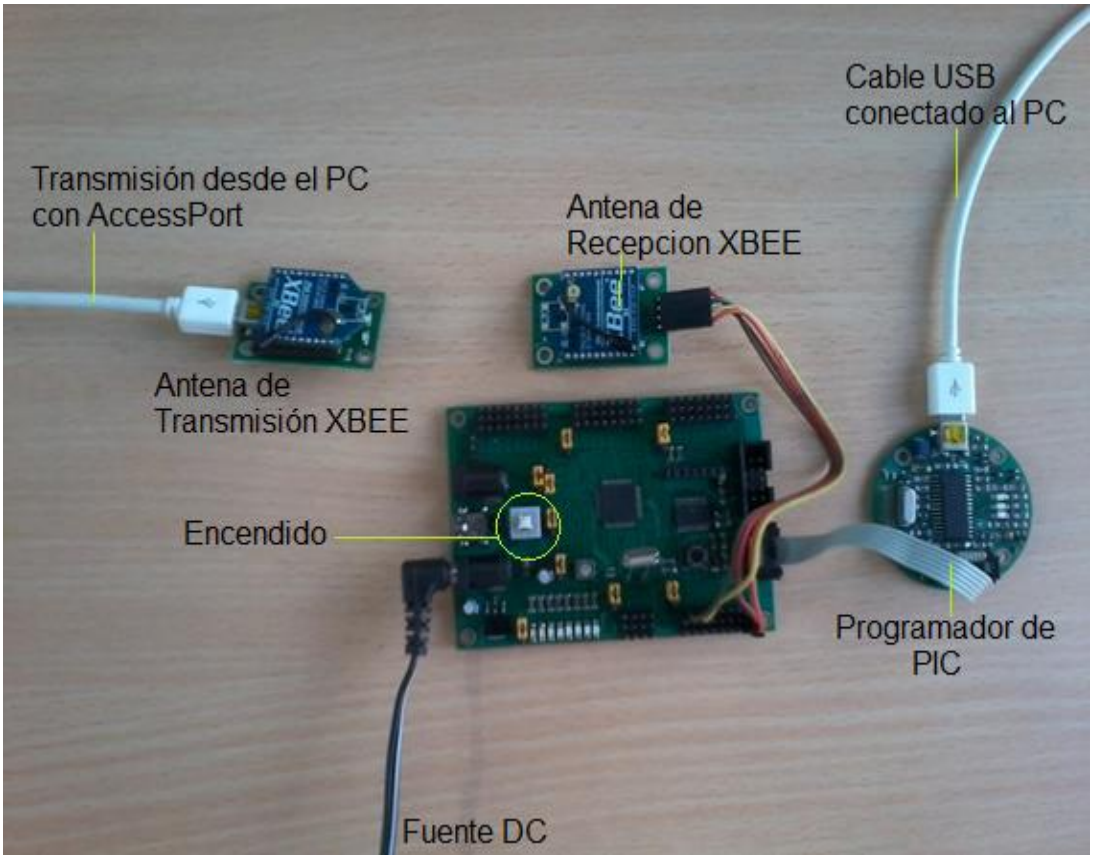
Practica_rx_Uart.c | 18F4520.h | Practica_rx_Uart.h *

```
#include <18F4520.h>
#device ICD=TRUE
#device adc=8
#FUSES NOWDT                    //No Watch Dog Timer
#FUSES WDT128                   //Watch Dog Timer uses 1:128 Postscale
#FUSES XT                       //Crystal osc <= 4mhz
#FUSES NOPROTECT                //Code not protected from reading
#FUSES NOBROWNOUT                 //Reset when brownout detected
#FUSES BORV25                   //Brownout reset at 2.5V
#FUSES NOPUT                    //No Power Up Timer
#FUSES NOCPD                    //No EE protection
#FUSES STVREN                   //Stack full/underflow will cause reset
#FUSES NODEBUG                  //No Debug mode for ICD
#FUSES NOLVP                    //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES NOWRT                    //Program memory not write protected
#FUSES NOWRTD                   //Data EEPROM not write protected
#FUSES IESO                     //Internal External Switch Over mode enabled
#FUSES FCMEN                    //Fail-safe clock monitor enabled
#FUSES NOPBADEN                 //PORTB pins are configured as digital I/O on RESET
#FUSES NOWRTC                   //configuration not registers write protected
#FUSES NOWRTB                   //Boot block not write protected
#FUSES NOEBTR                   //Memory not protected from table reads
#FUSES NOEBTRB                  //Boot block not protected from table reads
#FUSES NOCPB                    //No Boot Block code protection
#FUSES LPT1OSC                  //Timer1 configured for low-power operation
#FUSES MCLR                     //Master Clear pin enabled
#FUSES XINST                    //Extended set extension and Indexed Addressing mode enabled

#use delay(clock=4000000,RESTART_WDT)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)
```

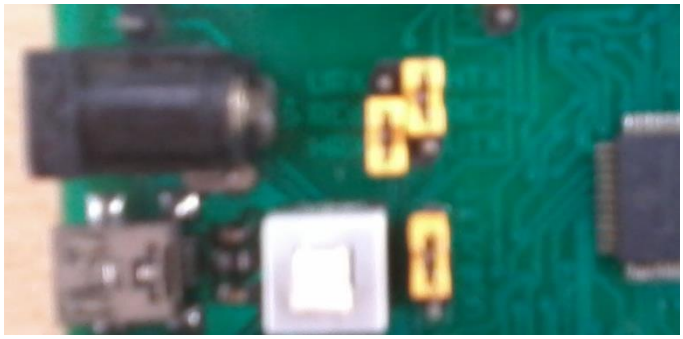
IMPLEMENTACIÓN.-

IMPLEMENTACIÓN DE TRANSMISION.-

IMPLEMENTACIÓN DE RECEPCIÓN.-



NOTA.- Para comunicación por antena XBEE colocar el jumper como se muestra en la siguiente figura.



CONEXIONES DE PUERTOS.-

PINES DE CONEXIÓN	XBEE-TX	XBEE-RX
PUERTO A	NC	NC
PUERTO B	NC	NC
PUERTO C	RC7	RC6
PUERTO D	NC	NC
PUERTO E	NC	NC

NOTA.- Las siglas NC significan No conectado.

PRÁCTICA # 3

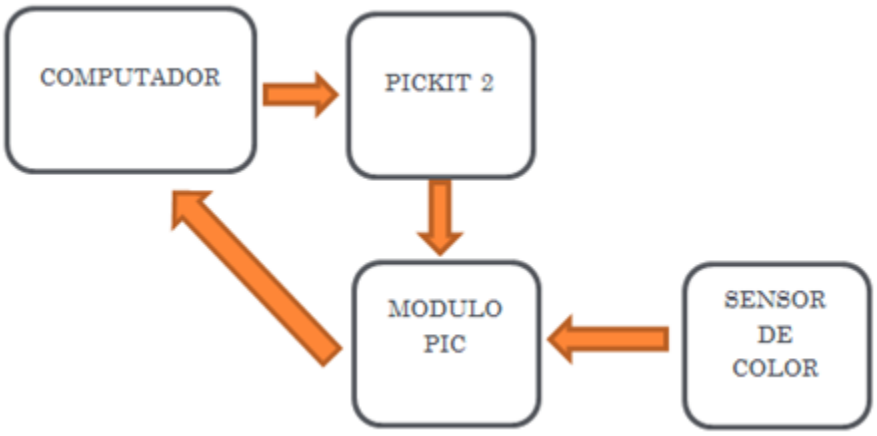
TEMA: SENSOR DE COLOR

Esta práctica se dedica a dar una introducción para hacer lecturas del estado de un Sensor de Color Blanco-Negro, con la Tarjeta de Entrenamiento con PIC 18F4520. Una vez desarrollado el código de ejemplo, el alumno debe realizar cambios en el software de control, para verificar lo aprendido.

OBJETIVOS.-

- Crear un proyecto con el software de control PIC C COMPILER.
- Crear un código para el control del Módulo de Entrenamiento con PIC 18F4550.
- Programar la tarjeta de control con el software PICKIT 2.
- Programar la tarjeta con el Software de control inicial de la Practica 4.
- Realizar pruebas de laboratorio, mediante cambios en el algoritmo de control para consolidar los conocimientos adquiridos.
- Programar el Módulo con PIC 18f4550 para la lectura del estado del Sensor de Color y enviar los datos por UART hacia el computador.

DIAGRAMA ESQUEMÁTICO.-



CÓDIGO DE PROGRAMACIÓN.-

```
#include #use delay (clock=4000000, RESTART_WDT)
#use rs232 (baud=9600, parity=N, xmit=PIN_C6, rcv=PIN_C7, bits=8)

void main()
{
  setup_adc_ports (NO_ANALOGS|VSS_VDD);
  setup_adc (ADC_OFF|ADC_TAD_MUL_0);
  setup_psp (PSP_DISABLED);
  setup_spi (SPI_SS_DISABLED);
  setup_wdt (WDT_OFF);
  setup_timer_0 (RTCC_INTERNAL);
  setup_timer_1 (T1_DISABLED);
  setup_timer_2 (T2_DISABLED,0,1);
  setup_comparator (NC_NC_NC_NC);
  setup_vref (FALSE);
  while (1)
  {
    IF (INPUT (PIN_C0)==0)
    {
      PUTS ("COLOR BLANCO");
      OUTPUT_D (0X0F);
      DELAY_MS (1000);
    }
    IF (INPUT (PIN_C0)==1)
    {
      PUTS ("COLOR NEGRO");
      OUTPUT_D (0XF0);
      DELAY_MS (1000);
    }
```

```
  }
}
```

SOFTWARE DE CONTROL ARCHIVO .C.-

Sensor_color.c * | Sensor_color.h | 18F4520.h

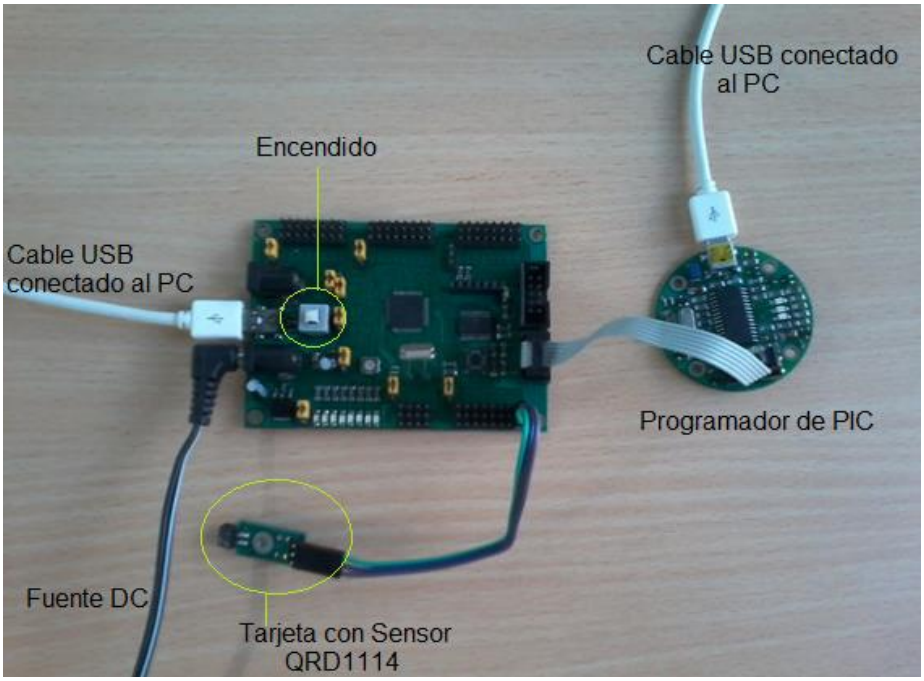
```
#include            "C:\Users\Documents\Desarrollos_PIC_C\Practica_4\Sensor_color.h"
#use delay(clock=4000000,RESTART_WDT)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)
void main()
{
   setup_adc_ports(NO_ANALOGS|VSS_VDD);
   setup_adc(ADC_OFF|ADC_TAD_MUL_0);
   setup_psp(PSP_DISABLED);
   setup_spi(SPI_SS_DISABLED);
   setup_wdt(WDT_OFF);
   setup_timer_0(RTCC_INTERNAL);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DISABLED,0,1);
   setup_comparator(NC_NC_NC_NC);
   setup_vref(FALSE);
   while (1)
   {
      IF (INPUT(PIN_C0)==0)
      {
         PUTS("COLOR BLANCO");
         OUTPUT_D(0X0F);
         DELAY_MS(1000);
      }
      IF (INPUT(PIN_C0)==1)
      {
         PUTS("COLOR NEGRO");
         OUTPUT_D(0XF0);
         DELAY_MS(1000);
      }
   }
}
```

SOFTWARE DE CONTROL ARCHIVO .H.-

Sensor_color.c * | Sensor_color.h | 18F4520.h

```
#include <18F4520.h>
#device adc=8
#device ICD=TRUE
#FUSES NOWDT                    //No Watch Dog Timer
#FUSES WDT128                   //Watch Dog Timer uses 1:128 Postscale
#FUSES XT                      //High speed Osc (> 4mhz)
#FUSES NOPROTECT                //Code not protected from reading
#FUSES BROWNOUT                 //Reset when brownout detected
#FUSES BORV25                   //Brownout reset at 2.5V
#FUSES NOPUT                    //No Power Up Timer
#FUSES NOCPD                    //No EE protection
#FUSES STVREN                   //Stack full/underflow will cause reset
#FUSES NODEBUG                  //No Debug mode for ICD
#FUSES NOLVP                     //Low Voltage Programming on B3(PIC16) or B5(PIC18)
#FUSES NOWRT                    //Program memory not write protected
#FUSES NOWRTD                   //Data EEPROM not write protected
#FUSES IESO                     //Internal External Switch Over mode enabled
#FUSES FCMEN                    //Fail-safe clock monitor enabled
#FUSES NOPBADEN                 //PORTB pins are configured as digital I/O on RESET
#FUSES NOWRTC                   //configuration not registers write protected
#FUSES NOWRTB                   //Boot block not write protected
#FUSES NOEBTR                   //Memory not protected from table reads
#FUSES NOEBTRB                  //Boot block not protected from table reads
#FUSES NOCPB                    //No Boot Block code protection
#FUSES LPT1OSC                  //Timer1 configured for low-power operation
#FUSES MCLR                     //Master Clear pin enabled
#FUSES XINST                    //Extended set extension and Indexed Addressing mode enabled

#use delay(clock=4000000,RESTART_WDT)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)
```

IMPLEMENTACIÓN.-



CONEXIONES DE PUERTOS.-

PINES DE CONEXIÓN	TARJETA SENSOR	LED
PUERTO A	NC	NC
PUERTO B	NC	NC
PUERTO C	RC0	NC
PUERTO D	NC	ENCENDIDO
PUERTO E	NC	NC

NOTA.- Las siglas NC significan No conectado.

PRÁCTICA # 4

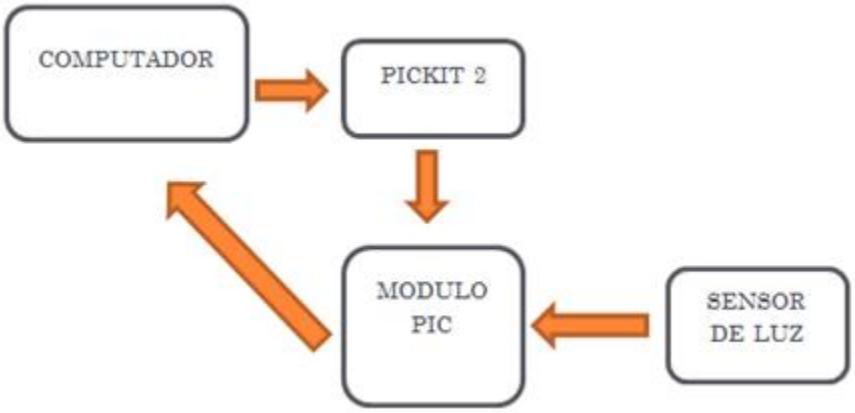
TEMA: SENSOR DE LUZ

Esta práctica se dedica a dar una introducción para hacer lecturas del estado de un Sensor de Luz, con el Módulo de Entrenamiento con PIC 18F4520. Una vez desarrollado el código de ejemplo, el alumno debe realizar cambios en el software de control, para verificar lo aprendido.

OBJETIVOS.-

- Crear un proyecto con el software de control PIC C COMPILER.
- Crear un código para el control del Módulo de Entrenamiento con PIC 18F4550.
- Programar la tarjeta de control con el software PICKIT 2.
- Programar la tarjeta con el Software de control inicial de la Practica 5.
- Realizar pruebas de laboratorio, mediante cambios en el algoritmo de control para consolidar los conocimientos adquiridos.
- Programar el Módulo con PIC 18f4550 para la lectura del estado del Sensor de Luz y enviar los datos por UART hacia el computador.

DIAGRAMA ESQUEMÁTICO.-



CÓDIGO DE PROGRAMACIÓN.-

```
#include
#use delay (clock=4000000, RESTART_WDT)
#use rs232 (baud=9600, parity=N, xmit=PIN_C6,rcv=PIN_C7,bits=8)
int lectura=0;
CHAR info [10];
void main()
{
  setup_adc_ports (AN0|VSS_VDD);
  //setup_adc (ADC_OFF|ADC_TAD_MUL_0);
  setup_adc (ADC_CLOCK_INTERNAL);
  setup_psp (PSP_DISABLED);
  setup_spi (SPI_SS_DISABLED);
  setup_wdt (WDT_OFF);
  setup_timer_0 (RTCC_INTERNAL);
  setup_timer_1 (T1_DISABLED);
  setup_timer_2 (T2_DISABLED,0,1);
  setup_comparator (NC_NC_NC_NC);
  setup_vref (FALSE);

  set_tris_d (0x00);
  set_tris_a (0x01);
  set_tris_E (0x08);
  set_tris_C (0x80);
while (1)
{
      set_adc_channel (0);
      delay_ms (200);
      lectura = read_adc();
      OUTPUT_D (Lectura);
```

```
        //printf("A/D value = %2x\n\r", lectura);  // valor Hexadecimal
        Printf ("A/D value = %U\n\r", lectura);     // valor en números enteros
        delay_ms (1000);
}
}
```

SOFTWARE DE CONTROL ARCHIVO .C.-

Sensor_de_luz.c * | Sensor_de_luz.h | 18F4520.h

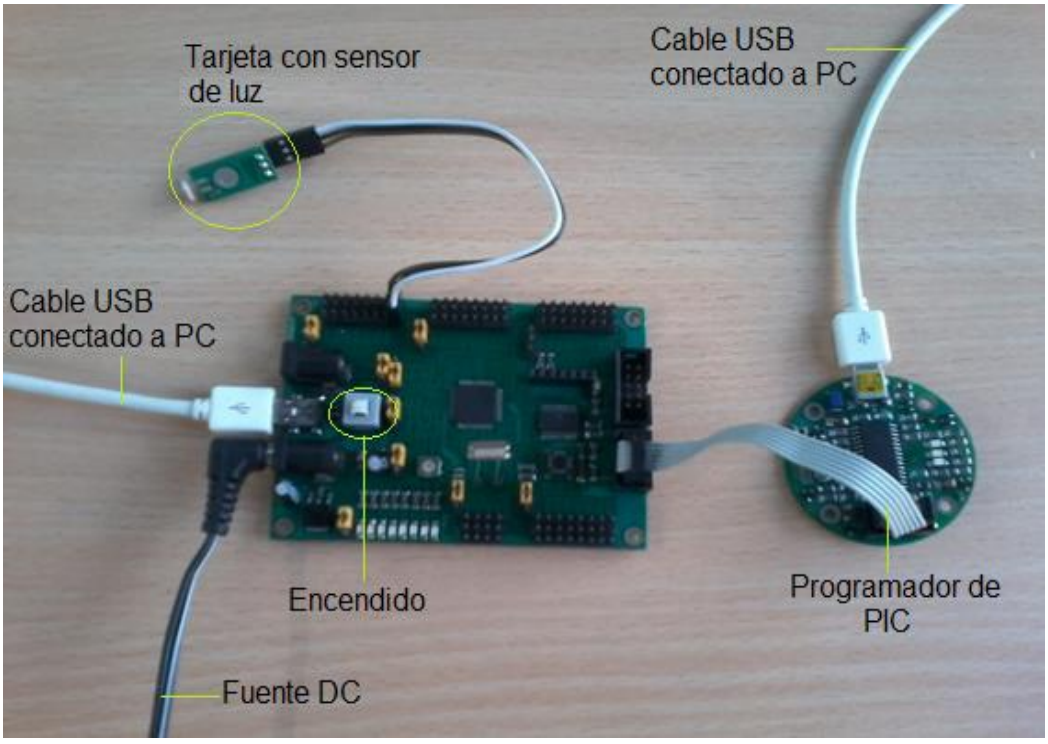
```
1  #include          "C:\Users\Documents\Desarrollos_PIC_C\Practica_5\Sensor_de_luz.h"
2  #use delay(clock=4000000,RESTART_WDT)
3  #use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)
4  int lectura=0;
5  CHAR info [10];
6  void main()
7  {
8     setup_adc_ports(AN0|VSS_VDD);
9     //setup_adc(ADC_OFF|ADC_TAD_MUL_0);
10    setup_adc(ADC_CLOCK_INTERNAL);
11    setup_psp(PSP_DISABLED);
12    setup_spi(SPI_SS_DISABLED);
13    setup_wdt(WDT_OFF);
14    setup_timer_0(RTCC_INTERNAL);
15    setup_timer_1(T1_DISABLED);
16    setup_timer_2(T2_DISABLED,0,1);
17    setup_comparator(NC_NC_NC_NC);
18    setup_vref(FALSE);
19 //Setup_Oscillator parameter not selected from Intr Oscillotar Config tab
20    // TODO: USER CODE!!
21    set_tris_d(0x00);
22    set_tris_a(0x01);
23    set_tris_E(0x08);
24    set_tris_C(0x80);
25 while(1)
26 {
27        set_adc_channel(0);
28        delay_ms(200);
29        lectura = read_adc();
30        OUTPUT_D(Lectura);
31        //printf("A/D value = %2x\n\r",lectura);  //valor Hexadecimal
```

SOFTWARE DE CONTROL ARCHIVO .H.-

Sensor_de_luz.c * | Sensor_de_luz.h * | 18F4520.h

```
#include <18F4520.h>
#device adc=8
#device ICD=TRUE
#FUSES NOWDT                    //No Watch Dog Timer
#FUSES WDT128                   //Watch Dog Timer uses 1:128 Postscale
#FUSES XT                       //High speed Osc (> 4mhz)
#FUSES NOPROTECT                //Code not protected from reading
#FUSES NOBROWNOUT                 //Reset when brownout detected
#FUSES BORV25                   //Brownout reset at 2.5V
#FUSES NOPUT                    //No Power Up Timer
#FUSES NOCPD                    //No EE protection
#FUSES STVREN                   //Stack full/underflow will cause reset
#FUSES NODEBUG                  //No Debug mode for ICD
#FUSES NOLVP                      //Low Voltage Programming on B3(PIC16) or B5(PIC18)
#FUSES NOWRT                    //Program memory not write protected
#FUSES NOWRTD                   //Data EEPROM not write protected
#FUSES IESO                     //Internal External Switch Over mode enabled
#FUSES FCMEN                    //Fail-safe clock monitor enabled
#FUSES PBADEN                   //PORTB pins are configured as analog input channels on RESET
#FUSES NOWRTC                   //configuration not registers write protected
#FUSES NOWRTB                   //Boot block not write protected
#FUSES NOEBTR                   //Memory not protected from table reads
#FUSES NOEBTRB                  //Boot block not protected from table reads
#FUSES NOCPB                    //No Boot Block code protection
#FUSES LPT1OSC                  //Timer1 configured for low-power operation
#FUSES MCLR                     //Master Clear pin enabled
#FUSES XINST                    //Extended set extension and Indexed Addressing mode enabled
#use delay(clock=4000000,RESTART_WDT)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)
```

IMPLEMENTACIÓN.-



CONEXIONES DE PUERTOS.-

PINES DE CONEXIÓN	TARJETA SENSOR	LED
PUERTO A	RA0	NC
PUERTO B	NC	NC
PUERTO C	NC	NC
PUERTO D	NC	ENCENDIDO
PUERTO E	NC	NC

NOTA.- Las siglas NC significan No conectado.

PRÁCTICA # 5

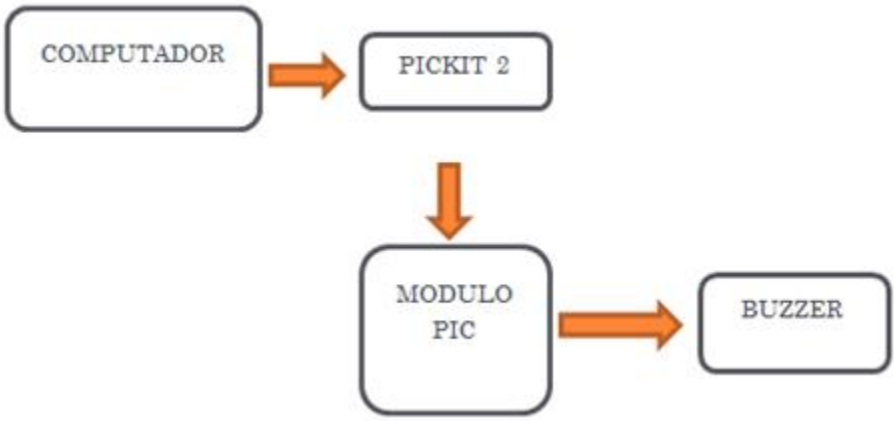
TEMA: BUZZER

Esta práctica se dedica a dar una introducción para generar un sonido con BUZZER utilizado para alertas, con la Tarjeta de Entrenamiento con PIC 18F4520. Una vez desarrollado el código de ejemplo, el alumno debe realizar cambios en el software de control, para verificar lo aprendido.

OBJETIVOS.-

- Crear un proyecto con el software de control PIC C COMPILER.
- Crear un código para el control del Módulo de Entrenamiento con PIC 18F4550.
- Programar la tarjeta de control con el software PICKIT 2.
- Programar la tarjeta con el Software de control inicial de la Practica 6.
- Realizar pruebas de laboratorio, mediante cambios en el algoritmo de control para consolidar los conocimientos adquiridos.
- Programar el Módulo con PIC 18f4550 para generar un sonido periódico usualmente utilizado en sistemas de alerta.

DIAGRAMA ESQUEMÁTICO.-



CÓDIGO DE PROGRAMACIÓN.-

```
#include
void main()
{
   setup_adc_ports (NO_ANALOGS|VSS_VDD);
   setup_adc (ADC_OFF|ADC_TAD_MUL_0);
   setup_psp (PSP_DISABLED);
   setup_spi (SPI_SS_DISABLED);
   setup_wdt (WDT_OFF);
   setup_timer_0 (RTCC_INTERNAL);
   setup_timer_1 (T1_DISABLED);
   setup_timer_2 (T2_DISABLED,0,1);
   setup_comparator (NC_NC_NC_NC);
   setup_vref (FALSE);
   setup_oscillator
(OSC_8MHZ|OSC_INTRC|OSC_31250|OSC_PLL_OFF);

   SET_TRIS_D (0X00);
   OUTPUT_D (0X00);
        while (1)
        {
          OUTPUT_HIGH (PIN_D0);
          DELAY_MS (300);
          OUTPUT_LOW (PIN_D0);
          DELAY_MS (300);
          OUTPUT_HIGH (PIN_D0);
          DELAY_MS (300);
          OUTPUT_LOW (PIN_D0);
          DELAY_MS (1000);
        }
```

}

SOFTWARE DE CONTROL ARCHIVO .C.-

Buzzer.c * | servo_motor.c | servo_motor.h

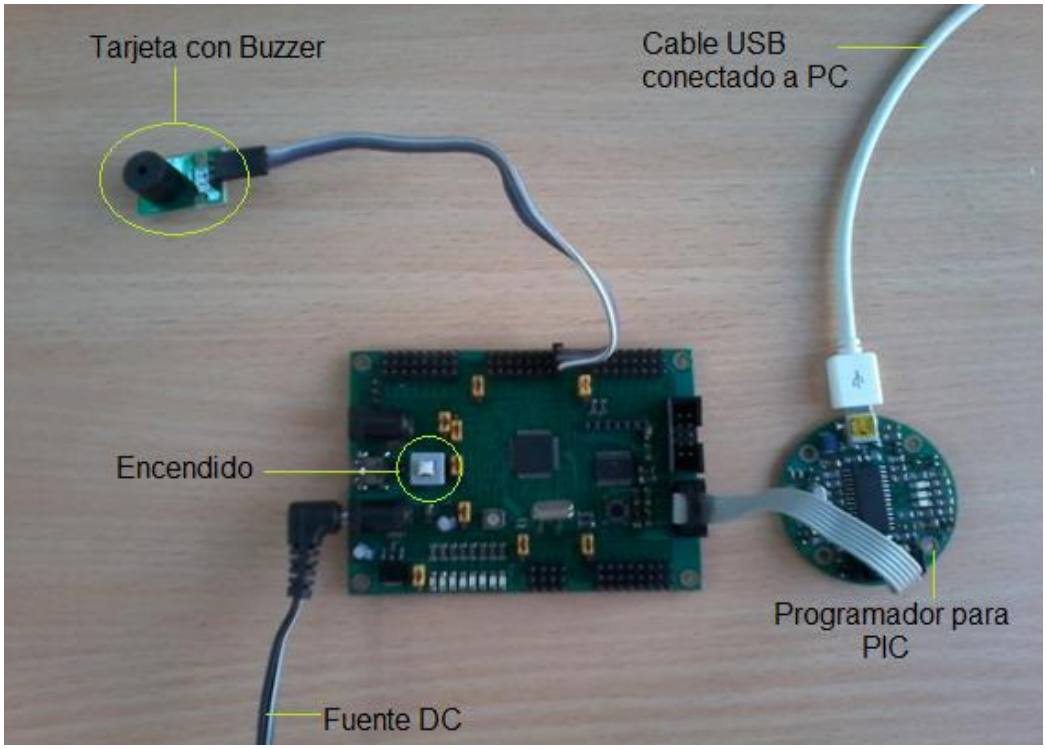
```
#include            "C:\Users\Documents\Desarrollos_PIC_C\Practica_6\Buzzer.h"
void main()
{
   setup_adc_ports(NO_ANALOGS|VSS_VDD);
   setup_adc(ADC_OFF|ADC_TAD_MUL_0);
   setup_psp(PSP_DISABLED);
   setup_spi(SPI_SS_DISABLED);
   setup_wdt(WDT_OFF);
   setup_timer_0(RTCC_INTERNAL);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DISABLED,0,1);
   setup_comparator(NC_NC_NC_NC);
   setup_vref(FALSE);
   setup_oscillator(OSC_8MHZ|OSC_INTRC|OSC_31250|OSC_PLL_OFF);
   // TODO: USER CODE!!
   SET_TRIS_D(0X00);
   OUTPUT_D(0X00);
         while(1)
         {
            OUTPUT_HIGH(PIN_D0);
            DELAY_MS(300);
            OUTPUT_LOW(PIN_D0);
            DELAY_MS(300);
            OUTPUT_HIGH(PIN_D0);
            DELAY_MS(300);
            OUTPUT_LOW(PIN_D0);
            DELAY_MS(1000);
         }
}
```

SOFTWARE DE CONTROL ARCHIVO .H.-

Buzzer.c * | Buzzer.h | 18F4520.h

```
1  #include <18F4520.h>
2  #device adc=8
3  #device ICD=TRUE
4  #FUSES NOWDT                    //No Watch Dog Timer
5  #FUSES WDT128                   //Watch Dog Timer uses 1:128 Postscale
6  #FUSES XT                       //High speed Osc (> 4mhz)
7  #FUSES NOPROTECT                //Code not protected from reading
8  #FUSES NOBROWNOUT                 //Reset when brownout detected
9  #FUSES BORV25                   //Brownout reset at 2.5V
10 #FUSES NOPUT                    //No Power Up Timer
11 #FUSES NOCPD                    //No EE protection
12 #FUSES STVREN                   //Stack full/underflow will cause reset
13 #FUSES NODEBUG                  //No Debug mode for ICD
14 #FUSES NOLVP                      //Low Voltage Programming on B3(PIC16) or B5(PIC18)
15 #FUSES NOWRT                    //Program memory not write protected
16 #FUSES NOWRTD                   //Data EEPROM not write protected
17 #FUSES IESO                     //Internal External Switch Over mode enabled
18 #FUSES FCMEN                    //Fail-safe clock monitor enabled
19 #FUSES PBADEN                   //PORTB pins are configured as analog input channels on RESET
20 #FUSES NOWRTC                   //configuration not registers write protected
21 #FUSES NOWRTB                   //Boot block not write protected
22 #FUSES NOEBTR                   //Memory not protected from table reads
23 #FUSES NOEBTRB                  //Boot block not protected from table reads
24 #FUSES NOCPB                    //No Boot Block code protection
25 #FUSES LPT1OSC                  //Timer1 configured for low-power operation
26 #FUSES MCLR                     //Master Clear pin enabled
27 #FUSES XINST                    //Extended set extension and Indexed Addressing mode enabled
28
29 #use delay(clock=4000000)
30 #use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)
31
```

IMPLEMENTACIÓN.-



CONEXIÓN DE PUERTOS.-

PINES DE CONEXIÓN	TARJETA BUZZER
PUERTO A	NC
PUERTO B	NC
PUERTO C	NC
PUERTO D	RD0
PUERTO E	NC

NOTA.- Las siglas NC significan No conectado.

PRÁCTICA # 6

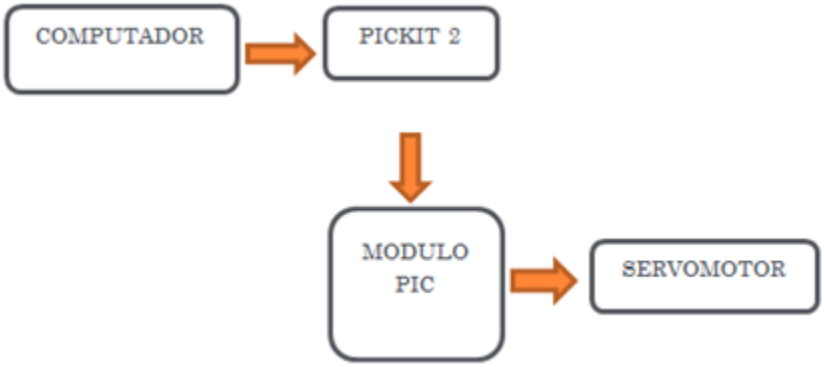
TEMA: SERVOMOTOR

Esta práctica se dedica a dar una introducción para controlar el giro de un Servomotor utilizado en robótica, con la Tarjeta de Entrenamiento con PIC 18F4550. Una vez desarrollado el código de ejemplo, el alumno debe realizar cambios en el software de control, para verificar lo aprendido.

OBJETIVOS.-

- Crear un proyecto con el software de control PIC C COMPILER.
- Crear un código para el control del Módulo de Entrenamiento con PIC 18F4550.
- Programar la tarjeta de control con el software PICKIT 2.
- Programar la tarjeta con el Software de control inicial de la Practica 7.
- Realizar pruebas de laboratorio, mediante cambios en el algoritmo de control para consolidar los conocimientos adquiridos.
- Programar el Módulo con PIC 18f4550 para hacer girar el servomotor en 3 posiciones diferentes.

DIAGRAMA ESQUEMÁTICO.-

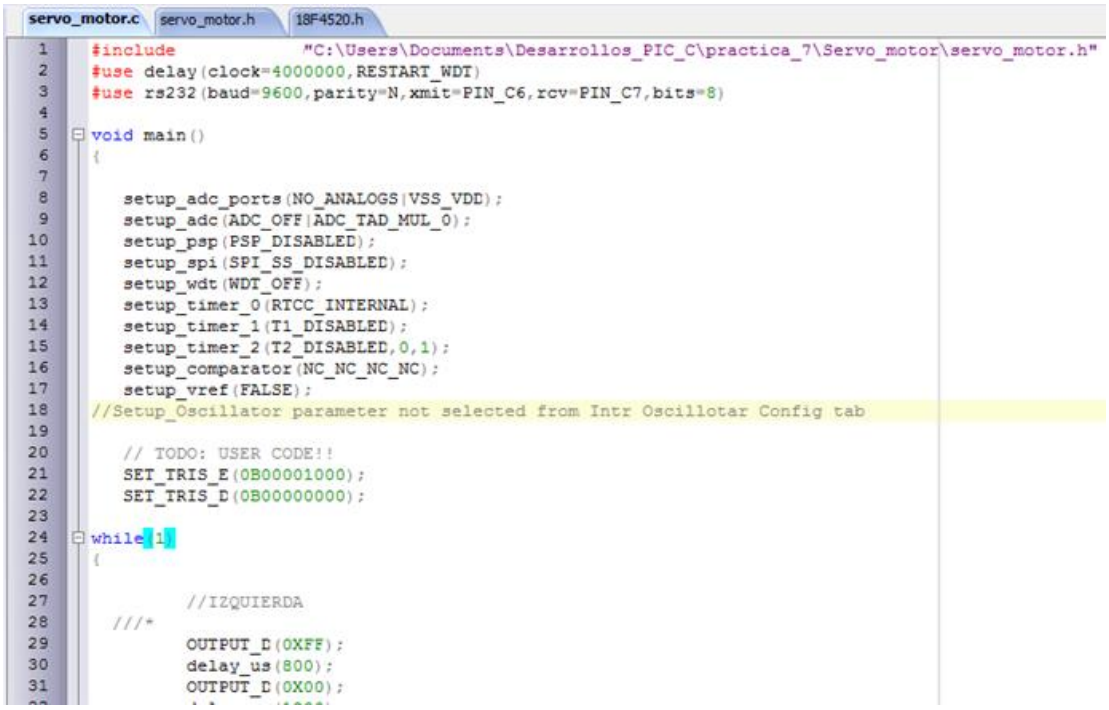


CÓDIGO DE PROGRAMACIÓN.-

```
#include
#use delay (clock=4000000, RESTART_WDT)
#use rs232 (baud=9600, parity=N, xmit=PIN_C6,rcv=PIN_C7,bits=8)
void main()
{
 setup_adc_ports (NO_ANALOGS|VSS_VDD);
 setup_adc (ADC_OFF|ADC_TAD_MUL_0);
 setup_psp (PSP_DISABLED);
 setup_spi (SPI_SS_DISABLED);
 setup_wdt (WDT_OFF);
 setup_timer_0 (RTCC_INTERNAL);
 setup_timer_1 (T1_DISABLED);
 setup_timer_2 (T2_DISABLED,0,1);
 setup_comparator (NC_NC_NC_NC);
 setup_vref (FALSE);
//Setup_Oscillator parameter not selected from Intr Oscillotar Config tab
 // TODO: USER CODE!!
 SET_TRIS_E (0B00001000);
 SET_TRIS_D (0B00000000);
while (1)
{
      //IZQUIERDA
 ///*
      OUTPUT_D (0XFF);
      delay_us (800);
      OUTPUT_D(0X00);
      delay_us (1200);
 //*/
      //CENTRO
```

```
 /*
      OUTPUT_D (0XFF);
      delay_us (1500);
      OUTPUT_D (0X00);
      delay_us (500);
*/
      //DERECHA
 /*
      OUTPUT_D (0XFF);
      delay_us (2400);
      OUTPUT_D (0X00);
      delay_us (300);
 */
}
}
```

SOFTWARE DE CONTROL ARCHIVO .C.-

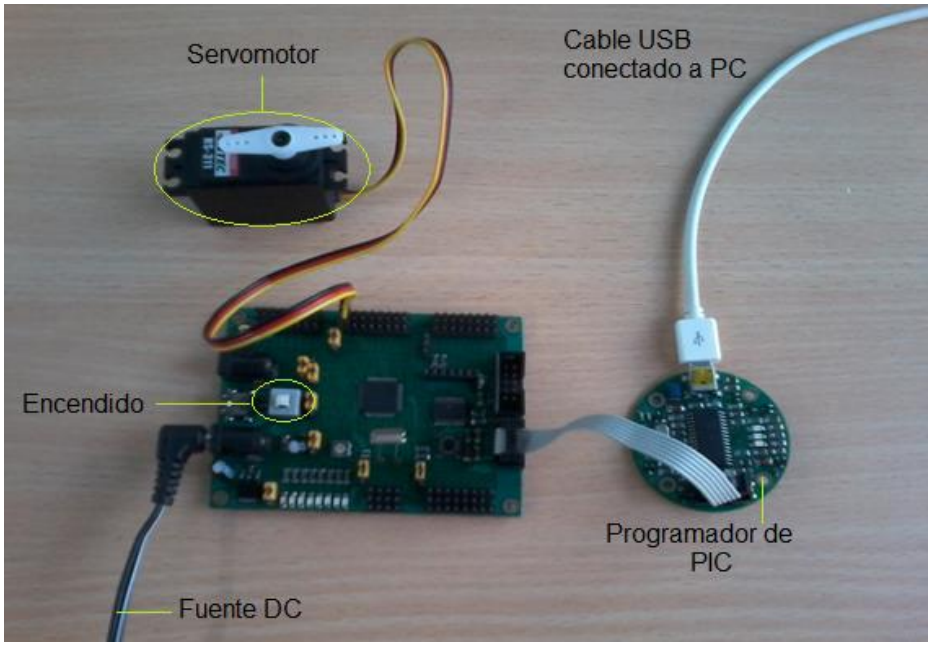


SOFTWARE DE CONTROL ARCHIVO .H.-

servo_motor.c | servo_motor.h | 18F4520.h

```
#include <18F4520.h>
#device ICD=TRUE
#device adc=8

#FUSES NOWDT                    //No Watch Dog Timer
#FUSES WDT128                   //Watch Dog Timer uses 1:128 Postscale
#FUSES XT                       //Crystal osc <= 4mhz
#FUSES NOPROTECT                //Code not protected from reading
#FUSES NOBROWNOUT                 //Reset when brownout detected
#FUSES BORV25                   //Brownout reset at 2.5V
#FUSES NOPUT                    //No Power Up Timer
#FUSES NOCPD                    //No EE protection
#FUSES STVREN                   //Stack full/underflow will cause reset
#FUSES NODEBUG                  //No Debug mode for ICD
#FUSES NOLVP                      //Low Voltage Programming on B3(PIC16) or B5(PIC18)
#FUSES NOWRT                    //Program memory not write protected
#FUSES NOWRTD                   //Data EEPROM not write protected
#FUSES IESO                     //Internal External Switch Over mode enabled
#FUSES FCMEN                    //Fail-safe clock monitor enabled
#FUSES PBADEN                  //PORTB pins are configured as digital I/O on RESET
#FUSES NOWRTC                    //configuration not registers write protected
#FUSES NOWRTB                   //Boot block not write protected
#FUSES NOEBTR                   //Memory not protected from table reads
#FUSES NOEBTRB                  //Boot block not protected from table reads
#FUSES NOCPB                    //No Boot Block code protection
#FUSES LPT1OSC                  //Timer1 configured for low-power operation
#FUSES NOMCLR                   //Master Clear pin used for I/O
#FUSES XINST                    //Extended set extension and Indexed Addressing mode enabled

#use delay(clock=4000000,RESTART_WDT)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)
```

IMPLEMENTACIÓN.-



CONEXIONES DE PUERTOS.-

PUERTOS DE CONEXIÓN	SERVOMOTOR
PUERTO A	NC
PUERTO B	NC
PUERTO C	NC
PUERTO D	RD0 a RD7
PUERTO E	NC

NOTA.- Las siglas NC significan No conectado.

PRÁCTICA # 7

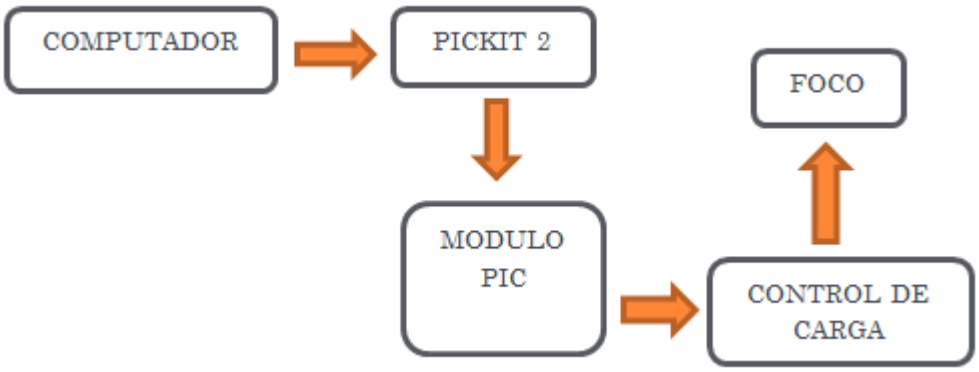
TEMA: CONTROL DE CARGA

Esta práctica se dedica a dar una introducción para controlar el encendido y apagado de un foco utilizado en Sistemas Inteligentes, con el Módulo de Entrenamiento con PIC 18F4550. Una vez desarrollado el código de ejemplo, el alumno debe realizar cambios en el software de control, para verificar lo aprendido.

OBJETIVOS.-

- Crear un proyecto con el software de control PIC C COMPILER.
- Crear un código para el control del Módulo de Entrenamiento con PIC 18F4550.
- Programar la tarjeta de control con el software PICKIT 2.
- Programar la tarjeta con el Software de control inicial de la Practica 8.
- Realizar pruebas de laboratorio, mediante cambios en el algoritmo de control para consolidar los conocimientos adquiridos.
- Programar el Módulo con PIC 18F4550 para hacer encender y apagar un foco automáticamente.

DIAGRAMA ESQUEMÁTICO.-



CÓDIGO DE PROGRAMACIÓN.-

```
#include
#use delay (clock=4000000, RESTART_WDT)
#use rs232 (baud=9600, parity=N, xmit=PIN_C6,rcv=PIN_C7,bits=8)

void main()
        {
          setup_adc_ports (NO_ANALOGS|VSS_VDD);
          setup_adc (ADC_OFF|ADC_TAD_MUL_0);
          setup_psp (PSP_DISABLED);
          setup_spi (SPI_SS_DISABLED);
          setup_wdt (WDT_OFF);
          setup_timer_0 (RTCC_INTERNAL);
          setup_timer_1 (T1_DISABLED);
          setup_timer_2 (T2_DISABLED,0,1);
          setup_timer_3 (T3_DISABLED|T3_DIV_BY_1);
          setup_comparator (NC_NC_NC_NC);
          setup_vref (FALSE);

         while (1)
          {
            OUTPUT_D (0XFF);
            DELAY_MS (300);
            OUTPUT_D (0X00);
            DELAY_MS (300);
          }
        }
```

SOFTWARE DE CONTROL ARCHIVO .C.-

Practica_8.c * | Practica_8.h | 18F4520.h

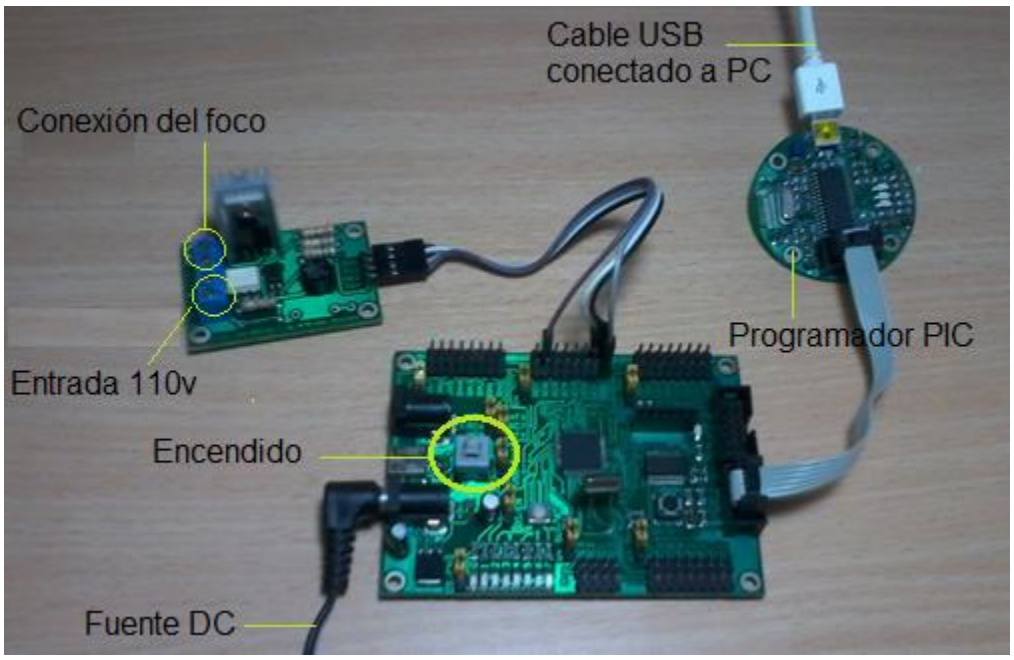
```
#include "C:\Users\Documents\Desarrollos_PIC_C\Practica_8.h"
#use delay(clock=4000000,RESTART_WDT)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)
void main()
{
   setup_adc_ports(NO_ANALOGS|VSS_VDD);
   setup_adc(ADC_OFF|ADC_TAD_MUL_0);
   setup_psp(PSP_DISABLED);
   setup_spi(SPI_SS_DISABLED);
   setup_wdt(WDT_OFF);
   setup_timer_0(RTCC_INTERNAL);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DISABLED,0,1);
   setup_timer_3(T3_DISABLED|T3_DIV_BY_1);
   setup_comparator(NC_NC_NC_NC);
   setup_vref(FALSE);
//Setup_Oscillator parameter not selected from Intr Oscillotar Config tab
   // TODO: USER CODE!!
   while(1)
   {
      OUTPUT_D(0XFF);
      DELAY_MS(300);
      OUTPUT_D(0X00);
      DELAY_MS(300);
   }
}
```

SOFTWARE DE CONTROL ARCHIVO .H.-

Practica_8.c * | Practica_8.h * | 18F4520.h

```
#include <18F4520.h>
#device ICD=TRUE
#device adc=8
#FUSES NOWDT                    //No Watch Dog Timer
#FUSES WDT128                   //Watch Dog Timer uses 1:128 Postscale
#FUSES XT                       //Crystal osc <= 4mhz
#FUSES NOPROTECT                //Code not protected from reading
#FUSES NOBROWNOUT                 //Reset when brownout detected
#FUSES BORV25                   //Brownout reset at 2.5V
#FUSES NOPUT                    //No Power Up Timer
#FUSES NOCPD                    //No EE protection
#FUSES STVREN                   //Stack full/underflow will cause reset
#FUSES NODEBUG                  //No Debug mode for ICD
#FUSES NOLVP                      //Low Voltage Programming on B3(PIC16) or B5(PIC18)
#FUSES NOWRT                    //Program memory not write protected
#FUSES NOWRTD                   //Data EEPROM not write protected
#FUSES IESO                     //Internal External Switch Over mode enabled
#FUSES FCMEN                    //Fail-safe clock monitor enabled
#FUSES PBADEN                   //PORTB pins are configured as analog input channels on RESET
#FUSES NOWRTC                   //configuration not registers write protected
#FUSES NOWRTB                   //Boot block not write protected
#FUSES NOEBTR                   //Memory not protected from table reads
#FUSES NOEBTRB                  //Boot block not protected from table reads
#FUSES NOCPB                    //No Boot Block code protection
#FUSES LPT1OSC                  //Timer1 configured for low-power operation
#FUSES NOMCLR                     //Master Clear pin enabled
#FUSES XINST                    //Extended set extension and Indexed Addressing mode enabled
#use delay(clock=4000000,RESTART_WDT)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)
```

IMPLEMENTACIÓN.-



CONEXIONES DE PUERTOS.-

PINES DE CONEXIÓN	CONTROL DE CARGA
PUERTO A	NC
PUERTO B	NC
PUERTO C	NC
PUERTO D	CAR
PUERTO E	NC
GND	GND
VCC	+5V
GND	CCE

NOTA.- Las siglas NC significan No conectado.

PRÁCTICA # 8

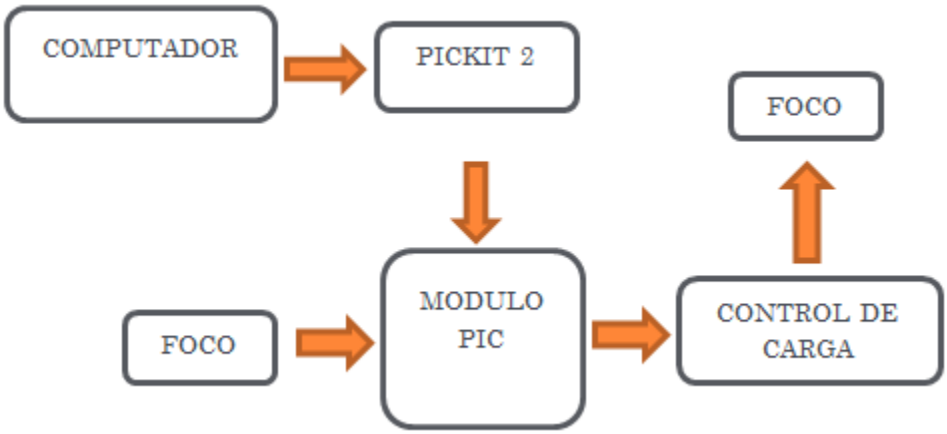
TEMA: LUMINARIA INTELIGENTE

Esta práctica se dedica a dar una introducción para controlar el encendido y apagado de un foco verificando el estado de un Sensor de Luz utilizado en Sistemas Inteligentes, el sistema enciende el foco durante la obscuridad y apaga el foco durante el día, desarrollado con el Módulo de Entrenamiento con PIC 18F4550. Una vez desarrollado el código de ejemplo, el alumno debe realizar cambios en el software de control, para verificar lo aprendido.

OBJETIVOS.-

- Crear un proyecto con el software de control PIC C COMPILER.
- Crear un código para el control del Módulo de Entrenamiento con PIC 18F4550.
- Programar la tarjeta de control con el software PICKIT 2.
- Programar la tarjeta con el Software de control inicial de la Practica 9.
- Realizar pruebas de laboratorio, mediante cambios en el algoritmo de control para consolidar los conocimientos adquiridos.
- Programar el Módulo con PIC 18f4550 para hacer encender y apagar un foco automáticamente dependiendo del estado del Sensor de Luz.

DIAGRAMA ESQUEMÁTICO.-



CÓDIGO DE PROGRAMACIÓN.-

```
#include
#use delay (clock=4000000, RESTART_WDT)
#use rs232 (baud=9600, parity=N, xmit=PIN_C6,rcv=PIN_C7,bits=8)
int lectura=0;

void main()
{
  setup_adc_ports (AN0|VSS_VDD);
  //setup_adc (ADC_OFF|ADC_TAD_MUL_0);
  setup_adc (ADC_CLOCK_INTERNAL);
  setup_psp (PSP_DISABLED);
  setup_spi (SPI_SS_DISABLED);
  setup_wdt (WDT_OFF);
  setup_timer_0 (RTCC_INTERNAL);
  setup_timer_1 (T1_DISABLED);
  setup_timer_2 (T2_DISABLED,0,1);
  setup_comparator (NC_NC_NC_NC);
  setup_vref (FALSE);
  set_tris_d (0x00);
  set_tris_a (0x01);
```

```
   set_tris_E (0x08);
   set_tris_C (0x80);

while(1)
{
       set_adc_channel (0);
       delay_ms (200);
       lectura = read_adc();
       printf("A/D value = %U\n\r",lectura);     //valor en numeros enteros
       if (lectura<250)
       {
         OUTPUT_D(0x00);  //APAGAR FOCO
       }
       else
       {
         OUTPUT_D(0xFF); //ENCENDER FOCO
       }
       delay_ms(1000);
  }
  }
```

SOFTWARE DE CONTROL ARCHIVO .C.-

Practica_9.c * | Practica_9.h | 18F4520.h

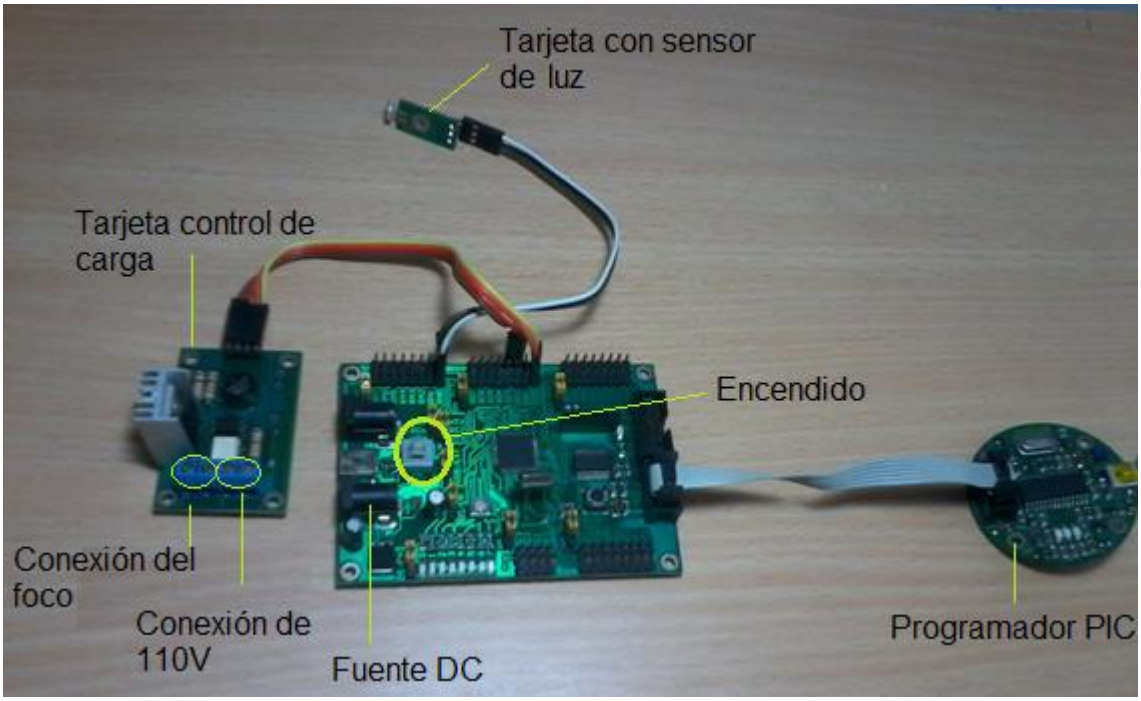
```
10      setup_psp(PSP_DISABLED);
11      setup_spi(SPI_SS_DISABLED);
12      setup_wdt(WDT_OFF);
13      setup_timer_0(RTCC_INTERNAL);
14      setup_timer_1(T1_DISABLED);
15      setup_timer_2(T2_DISABLED,0,1);
16      setup_comparator(NC_NC_NC_NC);
17      setup_vref(FALSE);
18      set_tris_d(0x00);
19      set_tris_a(0x01);
20      set_tris_E(0x08);
21      set_tris_C(0x80);
22  while(1)
23  {
24          set_adc_channel(0);
25          delay_ms(200);
26          lectura = read_adc();
27          printf("A/D value = %U\n\r",lectura);      //valor en numeros enteros
28          if (lectura<250)
29          {
30             OUTPUT_D(0x00);  //APAGAR FOCO
31          }
32          else
33          {
34             OUTPUT_D(0xFF); //ENCENDER FOCO
35          }
36          delay_ms(1000);
37  }
```

SOFTWARE DE CONTROL ARCHIVO .H.-

Practica_9.c * | Practica_9.h * | 18F4520.h

```
1   #include <18F4520.h>
2   #device ICD=TRUE
3   #device adc=8
4   #FUSES NOWDT                    //No Watch Dog Timer
5   #FUSES WDT128                   //Watch Dog Timer uses 1:128 Postscale
6   #FUSES XT                       //Crystal osc <= 4mhz
7   #FUSES NOPROTECT                //Code not protected from reading
8   #FUSES NOBROWNOUT                 //Reset when brownout detected
9   #FUSES BORV25                   //Brownout reset at 2.5V
10  #FUSES NOPUT                    //No Power Up Timer
11  #FUSES NOCPD                    //No EE protection
12  #FUSES STVREN                   //Stack full/underflow will cause reset
13  #FUSES NODEBUG                  //No Debug mode for ICD
14  #FUSES NOLVP                      //Low Voltage Programming on B3(PIC16) or B5(PIC18)
15  #FUSES NOWRT                    //Program memory not write protected
16  #FUSES NOWRTD                   //Data EEPROM not write protected
17  #FUSES IESO                     //Internal External Switch Over mode enabled
18  #FUSES FCMEN                    //Fail-safe clock monitor enabled
19  #FUSES PBADEN                   //PORTB pins are configured as analog input channels on RESET
20  #FUSES NOWRTC                   //configuration not registers write protected
21  #FUSES NOWRTB                   //Boot block not write protected
22  #FUSES NOEBTR                   //Memory not protected from table reads
23  #FUSES NOEBTRB                  //Boot block not protected from table reads
24  #FUSES NOCPB                    //No Boot Block code protection
25  #FUSES LPT1OSC                  //Timer1 configured for low-power operation
26  #FUSES MCLR                     //Master Clear pin enabled
27  #FUSES XINST                    //Extended set extension and Indexed Addressing mode enabled
28  #use delay(clock=4000000,RESTART_WDT)
```

IMPLEMENTACIÓN.-



CONEXIONES DE PUERTOS.-

PINES DE CONEXIÓN	CONTROL DE CARGA	SENSOR DE LUZ
PUERTO A	NC	RA0
PUERTO B	NC	NC
PUERTO C	NC	NC
PUERTO D	CAR	NC
PUERTO E	NC	NC
GND	GND	NC
VCC	+5V	NC
GND	CCE	NC

NOTA.- Las siglas NC significan No conectado.