Instrumentación de un sistema de paneles solares para el diseño de un controlador basado en el efecto Peltier-Seebeck

C. José Juan Atanacio Ángeles

2017

Universidad Autónoma de Querétaro
Facultad de Ingeniería

Instrumentación de un sistema de paneles solares para el diseño de un controlador de temperatura basado en el efecto Peltier-Seebeck

Tesis

Que como parte de los requisitos para obtener el título de

Ingeniero en Automatización

(Electrónica Industrial)

Presenta

C. José Juan Atanacio Ángeles

Santiago de Querétaro, Querétaro.

Enero 2017

- Escudo y letras doradas

- Pastas duras color negro, tamaño carta

Universidad Autónoma de Querétaro
Facultad de Ingeniería
Ingeniería en Automatización
Electrónica Industrial

"Instrumentación de un sistema de paneles solares para el diseño de un controlador de temperatura basado en el efecto Peltier-Seebeck"

TESIS

Que como parte de los requisitos para obtener el título de:

Ingeniero en Automatización
Electrónica Industrial

Presenta:
José Juan Atanacio Ángeles

Dirigido por:

M.C. José Luis Avendaño Juárez

SINODALES

M.C. José Luis Avendaño Juárez
Asesor — Firma

Dra. Lucero Gómez Herrera
Sinodal — Firma

Dr. Gonzalo Macías Bobadilla
Sinodal — Firma

Dr. José Gabriel Ríos Moreno
Sinodal — Firma

Centro Universitario
Santiago de Querétaro, Qro.
Septiembre 2016
México

RESUMEN

La temperatura y la irradiación son dos variables cruciales en la generación de energía eléctrica mediante sistemas fotovoltaicos, por medio del control de éstas se obtienen valores de corriente y voltaje cercanos a los valores proporcionados por los fabricantes. La temperatura en el panel fotovoltaico para su optima operación debe estar entre los 24.5 y 25.5 °C. Lo que se describe en esta tesis es un proyecto que se realizó en la Facultad de Ingeniería de la Universidad Autónoma de Querétaro, en el cual se implementó un sistema embebido, por medio de tarjetas controladas por microcontroladores PIC, para el monitoreo de temperatura e irradiación en un panel solar. Dichas variables se transmiten por medio de un sistema de comunicación inalámbrica mediante módulos Xbee a una computadora que hace las funciones de un servidor; tomando los datos de la tarjeta mediante USB. Se usa el control NETCommOCX de Microsoft Excel con lo cual se obtiene un registro y una representación gráfica de dichas variables; estos datos son usados por un controlador diseñado para mantener la temperatura de los paneles en el valor óptimo de operación (esto es un trabajo adicional que se desarrolló en una tesis de maestría). Para pruebas de funcionalidad se usa un control On/Off con histéresis de 2 °C con lo cual se obtiene un control cercano al deseado. La señal de control se envía desde la computadora hasta el panel para activar las celdas de Peltier-Seebeck por medio de un puente H para con ello mantener la temperatura del panel en el rango de operación óptima, ya sea, calentándolo o enfriándolo. Finalmente, es importante mencionar que en el presente documento se explica la metodología realizada para la adquisición, procesamiento y almacenamiento de los datos generados en el sistema fotovoltaico.

(**Palabras clave:** Energía solar, sistemas fotovoltaicos, celdas Peltier-Seebeck, Xbee, puente H**,** NETCommOCX)

DEDICATORIAS

A mis padres por ser el pilar fundamental en todo lo que soy, en toda mi educación, tanto académica, como de la vida, por su incondicional apoyo a lo largo de todo este tiempo.

A mi esposa e hija por apoyarme e impulsarme aún en los momentos y situaciones más difíciles, motivándome para poder culminar éste, el principio del proyecto de nuestras vidas.

Muchas gracias mis amores.

AGRADECIMIENTOS

Este trabajo puedo llegar a su fin, gracias a todos y cada uno de mis profesores que, durante mis años de estudio, me enseñaron, me aconsejaron y me guiaron para poder hoy ser lo que soy como profesional.

Quiero agradecer en especial al M. C. José Luis Avendaño Juárez, ya que fue un profesor que además de darme su conocimiento, me brindo un apoyo, respaldo y amistad, lo cual me ayudo para poder desarrollarme durante la carrera y en este último escalón, me apoyo hasta poder lograr este trabajo.

En particular quiero dar las gracias el apoyo otorgado por los proyectos FOFI de la UAQ ya que sin su apoyo de investigación y monetario, este trabajo no habrías llegado a su culminación.

GRACIAS.

ÍNDICE TEMÁTICO

RESUMEN i

DEDICATORIAS ii

AGRADECIMIENTOS iii

ÍNDICE TEMÁTICO iv

ÍNDICE DE FIGURAS vi

ÍNDICE DE TABLAS x

1. INTRODUCCIÓN 1

1.1. Antecedentes 3

1.2. Hipótesis 7

1.3. Objetivos 7

1.3.1. Objetivo General: 7

1.3.2. Objetivo Particular: 8

1.4. Justificación 8

2. FUNDAMENTACIÓN TEÓRICA 10

2.1 Efecto de la Irradiación en los sistemas FV 10

2.2 Efecto de la temperatura en los paneles solares 11

2.3 El efecto Peltier-Seebeck 13

3. METOLOGÍA 18

3.1 Selección de Sensores 18

3.1.1 Sensor de Temperatura 18

3.1.2 Sensor de Irradiación 22

3.2 Prueba de sensores 23

3.3 Montaje de los sensores en el Sistema FV. 28

3.4 Adquirir datos en el SATD 30

3.5 Realizar comunicación Wireless hacia el servidor. 31

3.6 Adquirir datos en la PC. 35

3.7 Desarrollar SCADA apropiado. 35

3.8 Convertir a señal de potencia 35

4. RESULTADOS 37
4.1. Resultados Finales 37
4.1.1. Pruebas de Comunicación bidireccional entre el PIC y la PC. 37
4.1.2. Pruebas de Sensor de Temperatura y envió de dato a PC. 55
4.1.3. Instrumentación de sensor de Irradiación, lectura y envió desde el PIC hacia la PC. 60
4.1.4. Celdas Peltier prueba de funcionamiento. 66
4.1.5. Fabricación de PCBs y montajes de sensores en el Panel FV. 68
4.1.6. Desarrollo del SCADA en PC. 75
4.1.7. Diagrama de flujo general y pruebas finales. 77
5. CONCLUSIONES 82
6. TRABAJOS A FUTURO 83
LITERATURA CITADA 84
APÉNDICE 87
Apéndice A 87
A1 Programa VB Habilitar y Deshabilitar puerto COM 87
A2 Programa VB envió de datos por USB 87
A3 Programa VB Recepción de datos 87
A4 Programa VB Recepción Varias Temperaturas 88
Apéndice B 89
B1 Programa comunicación PIC vs Hyperterminal 89
B2 Drive CDC 90
B3 Librería DS1820 91
B4 Sub Librería DS1820 101
B5 Programa Envío de Temperatura 1 sensor 102
B6 Programa PIC Main para envió de varias Temperaturas 102
B7 Programa PIC lectura de Irradiación 103

ÍNDICE DE FIGURAS

Figura **Página**

2.1 (a) Característica I-V de un panel comercial modelo BPSX60 para una temperatura e irradiación dadas. (b) Característica P-V para las mismas condiciones. 10

2.2 Efecto de la irradiación sobre la característica I-V de un panel FV. .. 11

2.3 Ejemplo de variación de la característica I-V de un panel FV al variar la temperatura, manteniendo la irradiación (G) constante (1000 W/m2) 12

2.4 Estructura de una celda Peltier. 14

3.1 Diagrama del sensor DS18B20.. 19

3.2 Memoria del DS18B20.. 21

3.3 Registro de temperatura.. 21

3.4 Tiempo de inicialización. 24

3.5 Simulación en ISIS Profesional del sensor DS18B20..... 28

3.6 Asignación de pines del sensor DS18B20.. 28

3.7 Sección transversal del sensor diseñado para su utilización en el exterior.. 29

3.8 Sensor encapsulado para aplicaciones en exteriores. 30

3.9 Diagrama a bloques el SATD. 31

3.10 Canales para el protocolo de modelos Xbee. 32

3.11 Modulo XBee S1 de Digi.. 33

3.12 Conexiones mínimas del módulo XBee. 34
3.13 Esquemático módulo XBee y PIC. 34
4.1 Pagina de Drive NETCommOCX. 38
4.2 Abrir VB desde Excel. 38
4.3 Abrir UserForm desde el VB de Excel. 39
4.4 Activar nuevo Drive. 39
4.5 Habilitar nuevo Drive desde pestaña Programador de Excel. 40
4.6 Habilitar nuevo Drive y colocarlo en la hoja de Excel deseada. 40
4.7 Pantalla de Pruebas Excel para recepción y envió de datos. 41
4.8 Prueba conexión Excel vs Hyperterminal. 42
4.9 Abrir puerto COM colocando el puerto COM y presionando Abrir puerto. 42
4.10 Envió de A hacia hyperterminal colocando la letra en Dato a Enviar y presionar Enviar Dato. 43
4.11 Envió de B hacia Excel, colocar la letra en ASCII y presionar SEND. 43
4.12 Configuración básica de PIC para uso de puerto USB. 44
4.13 Descriptores de PIC vs PC en librería usb_desc_cdc.h. 45
4.14 String a mostrar en PIC al ser conectado a modificarse en librería usb_desc_cdc.h. 46
4.15 Presionar Actualizar Software de controlador para poder cargar el drive 46
4.16 Buscar Software en el Equipo. 47
4.17 Elegir en lista en el Equipo. 48
4.18 Presionar Mostrar todos los dispositivos. 48
4.19 Presionar en Usar disco para poder cargar el drive. 49
4.20 Buscar Ruta de Drive. 49

4.21 Se Actualizo Correctamente. 50
4.22 Nuevo Puerto Serial. 51
4.23 LEDs de Estado, Rojo PIC no configurado por la PC y Verde cuando sea configurado 52
4.24 Colocar el número de puerto COM y presionar Abrir Puerto. 52
4.25 Se observará un cero en Dato Recibido enviado por el PIC. 53
4.26 Al enviar un 2 al PIC este Encender LED en B7. 53
4.27 Al enviar un 3 en PIC apagara el LED colocado en B7. 54
4.28 LED encendido en B7 del PIC. 54
4.29 LED apagado en B7 en el PIC. 55
4.30 Diagrama de conexión externa e interna de sensor DS18B20. 545
4.31 Recepción de Temperatura al presionar abrir puerto 56
4.32 Termómetro IR modelo 568 marca Fluke. 56
4.33 Comparación grafica entre el termómetro patrón contra los datos obtenidos en la PC. 56
4.34 Diferencia obtenida entre termómetro patrón y sistema diseñado. ... 56
4.35 Recepción varias temperaturas enviadas por el PIC a la PC. 59
4.32 Datos técnicos Sensor irradiació. 60
4.33 Medición de mV de Sensor de irradiación a diferentes horas de día cuando el sol es más intenso. 62
4.34 Circuito simulado en ISI Profesional de Amplificador no inversor G=50. 63
4.35 Circuito Real de Amplificador no inversor G=50. 64
4.36 Grafica de Irradiación real. 64
4.37 Recepción de irradiación ya estandarizada por parte del PIC. 66

4.38 Puente H, ejemplo para nuestro caso se usaron TIPs 35 y 36 de mayor potencia. 67
4.39 Puente H real ya implementado con los TIPs 25 y 26. 68
4.40 PCBs de PIC PC y PIC panel desarrollados en Ares. 68
4.41 Placas y Circuitos Impresos. 69
4.42 Placas terminadas. 69
4.43 Base para colocación de Panel. 70
4.44 Celda Peltier con grasa silica. 70
4.45 Celdas Peltier colocadas en panel. 71
4.46 Celda Peltier fijadas con Silicón Caliente. 71
4.47 Celda Peltier distribución. 72
4.48 Sensor de Temperatura DS1820 con recubrimiento de aluminio y silicón. 72
4.49 Sensor de Temperatura DS1820 fijado con silicón Caliente. 73
4.50 Distribución de sensores de temperatura y celdas. 73
4.51 Tarjeta de PIC panel y periféricos conectados. 74
4.52 Caja de PIC PC con USB. 74
4.53 Pantalla Principal ya con todos los datos de temperatura e irradiación. 75
4.54 Pantalla del histórico obtenida en el Excel. 76

ÍNDICE DE TABLAS

Tabla **Página**

3.1 Pinout DS18B20. 20

3.2 Relación de datos de temperatura..... 22

3.3 Comandos ROM.…..... 25

3.4 Comando de función... 26

4.1 Valores Irradiación VS mV 61

4.2 Datos sin controlador por la mañana..... 78

4.3 Datos con controlador por la mañana..... 79

4.4 Datos sin controlador por la tarde..... 80

4.5 Datos con controlador por la tarde. 80

1. INTRODUCCIÓN

Uno de los principales retos que se presentan hoy en día es el suministro de energía eléctrica convencional a nuestros hogares, lo cual implica grandes gastos para entregar la energía eléctrica desde su punto de generación (hidroeléctrica, termoeléctrica, etc.) hasta el interruptor de una lámpara en nuestros hogares. Estos gastos van desde los enormes transformadores que aumentan el voltaje y reducen la corriente para transportar la energía largas distancias por delgados cables, la infraestructura de torres y postes, los cables que se deben tender a lo largo del trayecto de la energía y hasta las horas hombre que se gastan en ajustes, calibraciones y demás acciones que se realizan para tener la energía eléctrica al alcance. El otro problema que trae consigo la generación de energía de la forma convencional es el daño ambiental que se presenta debido a la quema de combustibles fósiles, los cuales afectan directamente en el calentamiento global, lluvias ácidas, smog, contaminación del agua, vertederos residuales, destrucción de hábitats y pérdida de recursos naturales. Los problemas antes descritos sumados con algunos otros inconvenientes que presenta la generación de energía eléctrica de la forma convencional nos hace cuestionarnos, ¿Existe alguna forma más limpia y que ahorre recursos económicos?

La respuesta a la pregunta anterior son las energías alternativas o renovables las cuales son energías que se obtienen de medios naturales en teoría inagotables, ya sea por la inmensa cantidad de energía que contienen o porque son capaces de regenerarse por medios naturales y que proveen de electricidad sin contaminar el planeta; sin embargo, no son empleadas con frecuencia debido a su alto costo de implementación y mantenimiento. Entre las más importantes destacan: Biomasa: que utiliza la descomposición de residuos orgánicos para crear combustible y calentar un fluido mediante calderas y equipamientos homologados, Hidráulica: es aquella que se obtiene del aprovechamiento de las energías cinética y potencial de la corriente del agua, saltos de agua o mareas, Eólica: es la energía obtenida del viento, es decir, la energía cinética generada por efecto de las

corrientes de aire, Geotérmica: es aquella energía que puede obtenerse mediante el aprovechamiento del calor del interior de la Tierra, Mareomotriz: es la que se obtiene aprovechando las mareas, mediante su acoplamiento a un alternador se puede utilizar el sistema para la generación de electricidad; y por último tenemos, la llamada Energía Solar que es la energía obtenida mediante la captación de la luz y el calor emitidos por el Sol [1], que es en resumidas cuentas la obtención de energía eléctrica a través de Sistemas Fotovoltaicos (FV).

Los sistemas FV, están constituidos por celdas fotovoltaicas que son dispositivos semiconductores tipo diodo que, al recibir irradiación solar, se excitan y provocan saltos en los electrones, generando una pequeña diferencia de potencial en sus extremos. El acoplamiento en serie o en paralelo de varios de estos fotodiodos permite la obtención de voltajes y corrientes mayores para alimentar pequeños dispositivos electrónicos, uno de los problemas que se tienen con este tipo de generación de energía, es el alto costo de los materiales que se requieren para la instalación del sistema FV, aunque con el tiempo dicha inversión monetaria se va amortiguando hasta el punto en el cual el usuario ve un ahorro total del gasto por la energía que consume.

Otro inconveniente es que, las celdas están construidas con materiales semiconductores, que al igual que la mayoría de los dispositivos electrónicos son gravemente afectados por los cambios de irradiación y de temperatura, esto produce el deterioro de las celdas, calentamientos, fallas estructurales y demás fenómenos que ocurren cuando entran en juego estas variables. En el caso de los sistemas FV ocasiona que la energía que se nos entrega no sea constante durante las horas de operación del sistema. Este es un punto importante, dado que, para desarrollar un algoritmo de control apropiado, se deben de tomar en cuenta estas variables para así obtener siempre un buen desempeño del sistema FV.

Por lo anterior en esta tesis se desarrolló una instrumentación y adquisición de datos de un panel solar para poder crear históricos del comportamiento tanto de la

irradiación como de la temperatura; y con ello, que el usuario de estos datos pueda por medio su control desarrollado darnos una señal de accionamiento de las celdas montadas en el panel ya sea para calentar o para enfriar el panel solar y así obtener la mayor eficiencia del sistema FV.

Finalmente, el presente trabajo está distribuido en 8 secciones: 1. Introducción, en la cual se revisan los trabajos relacionados con el aquí mostrado, se plantea la hipótesis sobre la tesis y con base en lo anterior se proponen los objetivos generales y específicos para el desarrollo del trabajo con sus respectivas justificaciones. 2. Revisión Literaria, Se revisa básicamente los efectos de la irradiación y temperatura sobre el panel solar y el efecto denominado Peltier-Seebeck. 3. Metodología, Se muestra de forma organizada el desarrollo y los pasos que se siguieron para el desarrollo del proyecto de tesis. 4. Resultados, se muestran los resultados que se obtuvieron como resultado del desarrollado. 5. Conclusiones, se presenta el análisis final de los objetivos planteados en un inicio. 7. Literatura Citada, se presentan las fuentes de los documentos tomados para refutar los análisis presentados dentro de la tesis, y 8. Apéndice, se colocan los programas, controladores y librearías de los dispositivos y aplicaciones aquí desarrollados.

1.1. Antecedentes

Atendiendo al problema presentado en párrafos anteriores donde se presentó el efecto de la temperatura y la irradiación sobre los sistemas FV; en la Facultad de Ingeniería (FI) de Universidad Autónoma de Querétaro (UAQ), se trabajó en un proyecto encaminado a resolver esta problemática; se presentó una propuesta a los proyectos FOFI de la UAQ y fue aprobado con el número de registro FIN201214. El objetivo de este proyecto fue estudiar y aplicar el efecto Peltier-Seebeck para controlar la temperatura de un sistema FV; esto debido a que la temperatura es un factor influyente en la eficiencia de la conversión de la energía solar en energía eléctrica. Además, se trabajó en conjunto con Samuel Viñez

Romero en su trabajo de maestría para facilitar los datos de temperatura necesarios para el diseño, validación e implementación del control de temperatura para un sistema FV.

Para implementar la propuesta de este proyecto se hizo una investigación de los trabajos relacionados con la instrumentación de sistemas FV para medir la temperatura y/o la irradiación. El primer lugar donde se realizó esta búsqueda fue en la biblioteca de la FI de la UAQ revisando algunos libros y tesis de los compañeros en los cuales se notó que todos hablan de cómo implementar o colocar un sistema FV pero ninguno guarda una relación estrecha con lo que se pretende en este trabajo de tesis; por lo cual la búsqueda se extendió a revistas internacionales en las que se encontraron algunos artículos relacionados con el tema, de los cuales a continuación se presenta un breve resumen sobre su relación con el trabajo aquí presentado en un orden del menos a más relacionado con el proyecto.

El primer trabajo habla acerca de un estudio realizado en el área de Tokio de agosto de 1999 a julio de 2000 sobre la red de conexión de los sistemas FV para casas residenciales; centrándose en la temperatura del módulo. Se demostró que los sistemas FV operan en un amplio rango de temperaturas, y que el rango de temperatura del módulo, en el que los sistemas FV operan difieren mucho incluso para el mismo tipo de módulo debido principalmente al entorno ambiental de instalación. Además, se discute la influencia de la temperatura anual del módulo en producción de energía cuantitativamente. Estos resultados indicaron que es muy importante tener en cuenta las características de temperatura tanto de las celdas solares como la del entorno [2]. En este primer artículo se explica y aclara que es importante tomar en cuenta la temperatura del entorno que rodea al panel dado que esto influye en demasía en la producción de energía del sistema FV.

En un estudio realizado en Malasia en el 2006, en el que la investigación trató de averiguar la relación entre la eficiencia de las celdas solares y el clima de Malasia. En este experimento la corriente de salida y el voltaje del panel solar se

recaudaron durante 24 horas durante una semana. Estos datos se utilizaron para obtener la eficiencia de panel solar. Y al mismo tiempo, se obtuvieron los parámetros climáticos utilizando una estación meteorológica. La prueba de correlación y regresión se hicieron para obtener la relación matemática entre los parámetros climáticos y la eficiencia de las celdas solares. Los resultados de las pruebas mostraron que la irradiación solar y la temperatura habían influido mucho, mientras que la humedad y el viento habían influido muy poco en la eficiencia de las celdas solares policristalinas [3]. Lo más importante a lo que hace referencia este artículo, es que en su estudio la temperatura ambiental y la irradiación que inciden en el sistema FV tienen gran impacto en eficiencia de las celdas.

Otro artículo importante realizado también de Malasia, pero del año 2011, presenta un control fuzzy basado en el método de perturbación y observación del MPPT (seguidor de punto de máxima potencia) de un sistema FV. El sistema FV se modela y analiza en MATLAB / SIMULINK. Con el análisis completo se deduce que el sistema FV tiene una tensión óptima donde la celda FV produce la máxima potencia en un punto particular. Debido a la no linealidad de las características de voltaje-corriente en el sistema FV, es difícil determinar analíticamente la máxima potencia para operar la tensión de salida, esto debido a que varía con el cambio de la irradiación solar y la temperatura de la celda. El MPPT se implementa para identificar el punto de potencia máxima de operación, para luego regular la operación del panel solar en un voltaje particular y así obtener la máxima potencia de salida [4]. En este artículo se presenta algo mucho más cercano a lo que se desarrolla en este trabajo de tesis, pero con la deficiencia de que sólo es simulado y de que el MPPT sólo lleva el voltaje a un punto en el cual nos puede entregar la máxima potencia de la celda, pero esto sin controlar algún parámetro de la celda FV, como podría ser la irradiación que recibe o la temperatura del panel FV.

También dentro de la investigación se encontró el desarrollo de un control estratégico con limitación de producción, para un sistema FV. Primeramente, explica que la mayoría de los sistemas FV trabajan generalmente con algoritmos de MPPT para producir energía tanto como sea posible y que su potencia de

salida cambia principalmente en función de la irradiación solar, por lo que este tipo de controles son impredecibles para la generación eficiente de energía. Para los sistemas FV que están conectados a la red eléctrica, la producción con un algoritmo MPPT podría ser muy fluctuante debido a posibles variaciones significativas de la irradiación solar, lo cual podría ser un problema para la calidad de potencia de la red eléctrica. Por otra parte, para sistemas FV autónomos, que suelen trabajar con el almacenamiento de energía, la producción de MPPT podría causar problemas adicionales en el equilibrio de energía, cuando el almacenamiento alcanza un estado superior del límite de carga, es en estos casos cuando la producción del sistema FV debe usar el control estratégico para limitar la producción. Lo que lleva a controlar la energía FV en cualquier nivel dentro de la capacidad de producción. Si disminuye la irradiación solar y la energía FV el MPPT no es capaz de dar salida a la potencia deseada, la estrategia de control continúa operando el sistema FV con un algoritmo MPPT. La transición entre MPPT y el modo restringido o limitado debe ser continua; los dos modos deben operar por separado [5]. En este caso se observa que, a diferencia del artículo de Malasia, en este se hace hincapié en que es muy importante la irradiación recibida para realizar el algoritmo MPPT, pero a la vez en este trabajo no habla de la importancia de la temperatura de la celda FV que también es un punto muy interesante para ser analizado.

Por último, se analizó el artículo: “un enfoque basado en mediciones de temperatura por medio de un algoritmo MPPT aplicado en los sistemas FV”; el cual es lo más parecido a lo realizado en el proyecto aquí presentado, pero con unas sensibles e importantes diferencias. Se expone que se aplica una variación del algoritmo MPPT a un sistema FV denominado MPPT-TEMP, esté emplea sensores de temperatura LM35 con el fin de obtener una muestra de la temperatura de la superficie del módulo FV y, de ella, calcula la tensión óptima que debe imponerse a través del dispositivo FV. A diferencia del método de subida de pendiente (Hill Climbing), que emplean sensores de tensión y corriente para estimar el máximo punto de potencia (MPP), en el MPPT-TEMP el sensor de

corriente es sustituido por uno de temperatura, lo que implica algunas ventajas, como la reducción de costos, la obtención del MPP (punto de máxima potencia) incluso durante súbitas variaciones del medio ambiente (irradiación y/o temperatura) y sin oscilaciones alrededor del MPP en estado estacionario [6]. En este artículo se habla más de cerca de la importancia de la medición de las variables que afectan directamente sobre la eficiencia del sistema FV.

Como podemos observar del análisis de los artículos especializados en el tema es de suma importancia para el diseño de controladores de MPPT, el conocimiento de la temperatura y/o la irradiación del medio en el cual se implementa el sistema FV. Con la investigación presentada ahora se puede dar pie a la presentación de la hipótesis de trabajo.

1.2. Hipótesis

Mediante la implementación de un sistema para la adquisición de las variables temperatura e irradiación, y la transmisión de dicha información de forma inalámbrica hacia un concentrador de información; se puede proveer de los datos necesarios para la operación de un controlador con el cual se puede obtener el máximo punto de operación de un panel solar, esto variando la temperatura del panel por medio de las celdas Peltier-Seebeck.

1.3. Objetivos

1.3.1. Objetivo General:

Medir la temperatura e irradiación de un panel solar y transmitir dichas señales por medio de comunicación inalámbrica a un servidor para obtener el perfil de temperatura e irradiación del panel solar y con esto alimentar de datos al controlador desarrollado en la FI, y con la señal de control poder calentar o enfriar el panel según sea el caso.

1.3.2. Objetivo Particular:

- Implementar la electrónica necesaria para realizar la comunicación inalámbrica para la transmisión de datos.
- Implementar en una computadora que funcione como servidor, una aplicación para graficar los datos de temperatura e irradiación del panel solar.
- Crear una base de datos para el modelado del panel solar que servirá para alimentar de datos al controlador de temperatura desarrollado en la FI.
- Transmitir al panel solar las señales de mando producida por el controlador de temperatura para mantener dicha variable en el panel solar en el rango requerido.

1.4. Justificación

Como se ha presentado en las líneas anteriores es muy importante en un sistema FV la adquisición de datos tanto de la temperatura como de la irradiación que recibe dado que son las dos variables que afectan directamente en la eficiencia del sistema, además de tener un control con el cual garantizar que siempre podamos tener el mejor desempeño de nuestro sistema, lo cual ya no sería una simple simulación. Otro aspecto importante a tomar en cuenta es que la colocación de los sensores tanto de temperatura como de irradiación, no es algo sencillo y mucho menos tomando en cuenta que la mayoría de las veces los sistemas FV están colocados en las azoteas de los hogares o en este caso en la azotea del edificio de la dirección de la FI, por lo cual, sí se desea realizar un control eficiente lo primero que se debe de hacer es determinar una ecuación la cual caracterice el comportamiento del sistema FV por algunos meses, mediante el que se obtendrán los datos provenientes de los sensores en el llamado Sistema de Adquisición y Transmisión de Datos (SATD), el cual a través de señales inalámbricas envía los datos hacia una servidor en el cual se tendrá montado un

sistema SCADA en donde se podrá observar la evolución de las variables y además se tendrá el histórico de los datos para así desarrollar el modelado del sistema FV y también crear un algoritmo de control más eficiente.

Otro aspecto a resaltar es que en los cinco artículos antes mencionados se repite constantemente que la temperatura y la irradiación afectan en gran medida en el rendimiento del sistema FV, pero en ninguno de ellos se habla a ciencia cierta cómo poder resolver este problema. La solución aquí presentada, es que mediante actuadores basados en el efecto Peltier-Seebeck y de la señal de control mandada por el servidor una vez que la información fue analizada, se pueda controlar la temperatura de los paneles para así tener siempre la temperatura en el rango para el cual el sistema FV nos entregue su máxima potencia de salida.

2. FUNDAMENTACIÓN TEÓRICA

La curva característica I-V del panel FV como se observa en la figura 2.1, se ve afectada por factores ambientales tales como la intensidad de iluminación o Irradiación (G), la temperatura (T) y la distribución espectral de la luz solar.

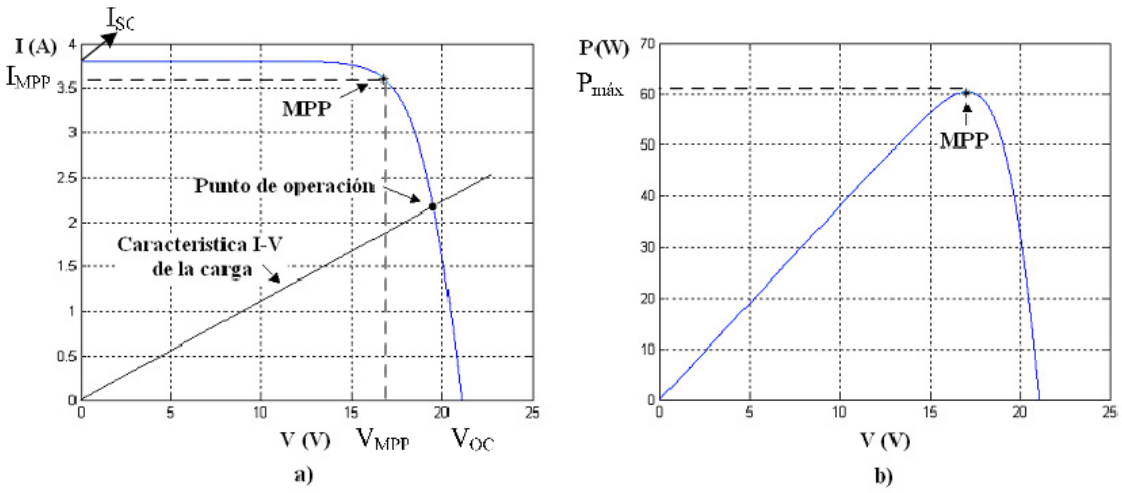


Figura. 2.1. (a) Característica I-V de un panel comercial modelo BPSX60 para una temperatura e irradiación dadas. (b) Característica P-V para las mismas condiciones [imágenes tomadas de [7]]**.**

2.1 Efecto de la Irradiación en los sistemas FV

En general la irradiación afecta principalmente a la corriente, de forma que se puede considerar que la corriente de cortocircuito del generador fotovoltaico es proporcional a la irradiación [7]:

$$I_{SC}(G_2) = I_{SC}(G_2)\frac{G_2}{G_1} \quad (1)$$

En esta expresión, $I_{SC}(G_2)$ es la corriente de cortocircuito para un nivel de irradiación G_2 e I_{SC} (G_1) es la corriente de cortocircuito para un nivel de irradiación G_1.

La ecuación (1) es válida para variaciones de irradiación a temperatura constante, y resulta una aproximación cuando ésta varía, ya que supone despreciar los efectos que la temperatura tiene sobre la corriente de cortocircuito. Sin embargo, puede considerarse una expresión adecuada para un cálculo aproximado de los valores de I_{SC} a diferentes irradiaciones, ya que el error que se comete es inferior al 0.5%.

La figura 2.2 muestra un ejemplo de la influencia de la intensidad de iluminación sobre una curva I-V para un panel FV a distintos niveles de irradiación y temperatura constante (50° C).

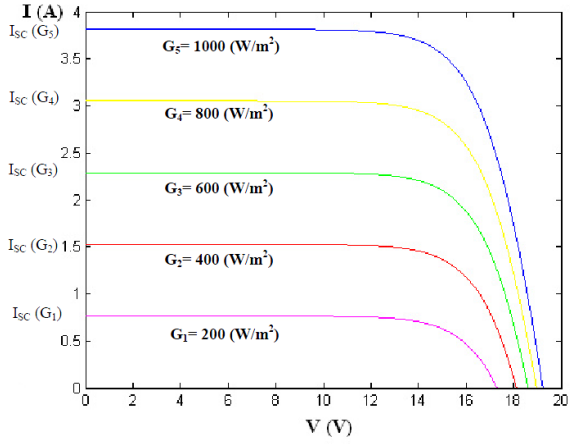


Figura. 2.2. Efecto de la irradiación sobre la característica I-V de un panel FV [imagen tomada de [7]]**.**

2.2 Efecto de la temperatura en los paneles solares

La temperatura afecta principalmente a los valores de voltaje de la característica I-V y tiene su mayor influencia en el voltaje de circuito abierto (V_{OC}), aunque también modifica los valores del punto de máxima potencia y el valor de I_{SC} (éste muy ligeramente). En la figura 2.3 puede observarse un ejemplo de la variación de la curva característica I-V de un panel o generador fotovoltaico al variar la temperatura, manteniendo la irradiación constante [7].

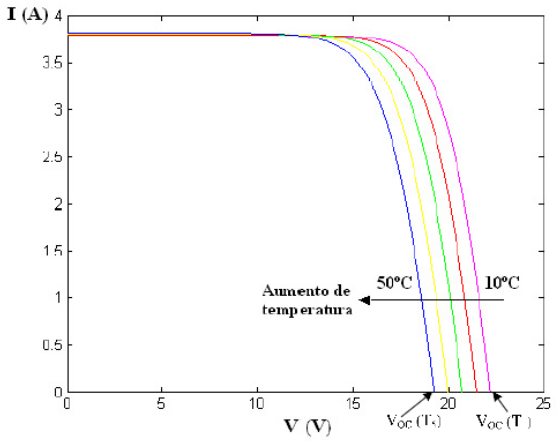


Figura. 2.3. Ejemplo de variación de la característica I-V de un panel FV al variar la temperatura, manteniendo la irradiación (G) constante (1000 W/m2) [imagen tomada de [7]]**.**

Existen tres coeficientes α, β y γ que representan la variación de los parámetros fundamentales de la característica I-V del generador fotovoltaico con la temperatura. Así, α expresa la variación de la corriente de cortocircuito con la temperatura, β la variación del voltaje de circuito abierto y γ la variación del punto de máxima potencia.

En general los fabricantes de módulos o paneles FV incluyen en sus hojas de características técnicas los valores de estos tres coeficientes, cuyos valores más comunes pueden ser:

$$\alpha = \frac{\delta I_{SC}}{\delta T} \approx -0.04\% \quad (2)$$

$$\beta = \frac{\delta V_{OC}}{\delta T} \approx -0.37\% \quad (3)$$

$$\gamma = \frac{\delta P_{MPP}}{\delta T} \approx -0.44\% \quad (4)$$

Los valores presentados en las ecuaciones (2), (3) y (4) son valores típicos para dispositivos de silicio monocristalino expresados en porcentaje, sin embargo, es más común encontrarse estos valores como referencia a los de una célula constituyente del sistema FV con valores típicos. Para el caso de una celda de aproximadamente 100 cm^2:

$$\alpha \approx -1.5 \ {}^{mA}/_{\circ C} \quad (5)$$

$$\beta \approx -2.3 \ {}^{mV}/_{\circ C} \quad (6)$$

$$\left[\frac{1}{p_{MPP}}\gamma \approx {}^{0.00441}/_{\circ C}\right]\alpha \approx -1.5 \ {}^{mA}/_{\circ C} \quad (7)$$

2.3 El efecto Peltier-Seebeck

Las celdas Peltier, son dispositivos termoeléctricos que se caracterizan por la aparición de una diferencia de temperaturas entre las dos caras de un semiconductor cuando una corriente eléctrica lo atraviesa [7].

Estructuralmente la celda Peltier, está formada por un conjunto de elementos semiconductores p-n formados por telurio-bismuto agrupados de tal forma que consolidan una estructura de celda, figura 2.4.

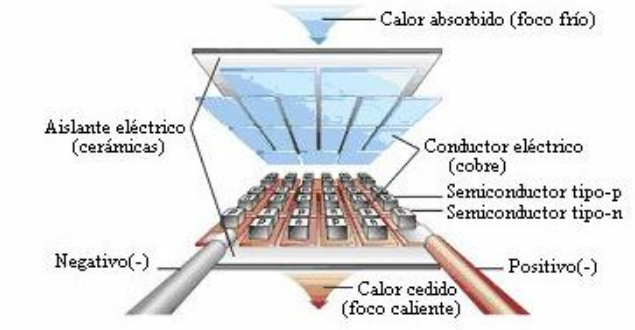


Figura. 2.4. Estructura de una celda Peltier [imagen tomada de [7]]**.**

Al aplicar una diferencia de potencial sobre la celda, se generará un flujo de calor por unidad de tiempo o potencia calorífica en la cara caliente, esto se puede determinar mediante la ecuación 8.

$$Q_{pc} = \alpha T_C I \tag{8}$$

Donde Tc (Kelvin) es la temperatura de la cara caliente, α es el coeficiente Seebeck e I (Amperes) la corriente que atraviesa al circuito.

Por el mismo efecto, la absorción de calor por unidad de tiempo o potencia calorífica en la cara fría está dada por la ecuación 9.

$$Q_{pf} = \alpha T_f I \tag{9}$$

Siendo T_f (Kelvin) la temperatura de la cara fría. De otro lado, si se consideran las pérdidas por unidad de tiempo por efecto Joule, las cuales se supone que se reparten a la mitad para cada cara, éstas quedarán expresadas por la ecuación 10.

$$Q_j = \frac{1}{2} I^2 R \tag{10}$$

Donde I (amperes) es la corriente eléctrica y R (ohms) es la resistencia eléctrica de la celda Peltier. La diferencia de temperaturas entre ambas caras producirá un efecto de conducción térmica entre la cara caliente y la cara fría, cuantificable mediante la ecuación 11.

$$Q_{CT} = \frac{T_C - T_F}{R_{TH}} \tag{11}$$

En donde R_{TH} representa la resistencia térmica entre la cara caliente y la fría. El flujo neto calorífico absorbido por la cara fría, será haciendo el balance energético por medio de la ecuación 12.

$$Q_F = Q_{PF} - Q_J - Q_{CT} = \alpha T_F I - \frac{1}{2} I^2 R - \frac{T_C - T_F}{R_{TH}} \tag{12}$$

Mientras que el calor cedido y que debe ser disipado a través de la cara caliente será como se expresa en la ecuación 13.

$$Q_C = Q_{PC} - Q_J - Q_{CT} = \alpha T_C I - \frac{1}{2} I^2 R - \frac{T_C - T_F}{R_{TH}} \tag{13}$$

Utilizando el primer principio de la termodinámica, la potencia eléctrica suministrada será la diferencia de los dos flujos caloríficos de disipación y absorción concluyendo esto en la ecuación 14.

$$P_e = Q_C - Q_F = \alpha (T_C - T_F) I + I^2 R = \alpha \Delta T I + I^2 R \tag{14}$$

Para poder entender el comportamiento de una celda Peltier, es necesario tomar en cuenta que en un dispositivo de esta naturaleza son varios los fenómenos que acontecen. Se debe mencionar que los efectos presentes son el efecto Joule, efecto Thompson y efecto Peltier, además de las propias características de la transmisión de calor, sin embargo, no todos son de igual magnitud e importancia.

El efecto Joule es la más conocida interacción entre un fenómeno eléctrico, la conducción de corriente eléctrica, y su fenómeno térmico asociado, el calentamiento del conductor por el que circula la corriente. La materia ofrece cierta

"resistencia" al movimiento de los electrones, los cuales ceden energía cinética al entorno en los sucesivos choques. Esta energía proporcionada por los electrones se disipa en forma de calor y se define como se expresa en la ecuación 15:

$$Q = I^2 R t \tag{15}$$

El efecto Peltier fue descubierto en el año 1834 por el físico francés Peltier J. C. A. surgió sobre la base del descubrimiento del físico alemán Seebeck T.J. en 1821, quien observó que en un circuito formado por dos conductores distintos A y B, cuyas uniones soldadas se encuentran en medios con temperaturas distintas, aparece entre ambos una diferencia de potencial. Esta diferencia de potencial depende de la naturaleza de los conductores y de la diferencia de temperaturas. Este dispositivo se conoce como termopar. La esencia del efecto Peltier, que básicamente es lo contrario del efecto Seebeck, consiste en hacer pasar una corriente procedente de una fuente de energía eléctrica continua, a través de un circuito formado por dos conductores de distinta naturaleza, obteniéndose que una de sus uniones absorbe calor y la otra lo cede. El calor que cede el foco caliente será la suma de la energía eléctrica aportada al termoelemento y el calor que absorbe el foco frío. Estos termoelementos, configurados de este modo, constituyen una máquina frigorífica.

El efecto Thomson, descubierto en 1857 por Thompson W, consiste en la absorción o liberación de calor por parte de un conductor eléctrico, con un gradiente o diferencia de temperaturas por el cual circula una corriente eléctrica. Este efecto relaciona el efecto Seebeck y el efecto Peltier de la siguiente forma:

$$\dot{Q} = \beta I \Delta T \tag{16}$$

Donde β es el coeficiente de Thomson, ΔT la diferencia de temperatura e I la corriente eléctrica [7]

Una vez revisado los conceptos básicos que fundamentan a este trabajo de tesis se procede a presentar la metodología con la cual se pretende cumplir el objetivo del trabajo y verificar la hipótesis planteada.

3. METOLOGÍA

3.1 Selección de Sensores

Una vez planteado el qué y el para qué, ahora tenemos que definir el cómo. Lo primero que se debe de hacer es obtener los datos que se necesitan analizar y esto se logra gracias a los sensores, los cuales son dispositivos que cuentan con un elemento sensible tanto a magnitudes físicas como químicas, están diseñados para transformar estas variables de instrumentación comúnmente a variables eléctricas o digitales.

3.1.1 Sensor de Temperatura

Como se analizó en párrafos anteriores, las variables físicas que más nos interesan son la irradiación y la temperatura, por lo cual se comenzó la búsqueda de sensores para dichas variables. Primero se buscó el sensor de temperatura con el cual se pudieran obtener los datos de una forma rápida, sencilla y de bajo costo. Se evaluó la posibilidad de usar sensores como termopares, termistores y RTD's que por más lineales que sean la mayoría de las veces necesitan una etapa de acondicionamiento de señal lo cual resultaría un poco más complicado y caro, estas ideas se desecharon; otro sensor de temperatura con el cual se ha tenido experiencia es el LM35, el cual si bien entrega una señal de voltaje, lo cual fácilmente se puede introducir al ADC de un microcontrolador y hacer las operaciones necesarias para tener en éste, el valor correcto de la variable; nos resulta otro problema que para este proyecto es importante, ¿Cómo hacer para tener más de 9 sensores trabajando al mismo tiempo? La respuesta más sencilla sería comprar un microcontrolador con 9 canales para el ADC, lo cual sería buena opción, pero se gastaría un poco más para llevar la señal desde la posición del sensor hasta el lugar donde se encuentra el microcontrolador, por lo cual esta no sería una buena opción.

Así que continuando con la búsqueda del sensor indicado se encontró con el sensor DS18B20 de Dallas Semicondutor™ [8] el cual es un termómetro digital

diseñado por la empresa Dallas Semiconductor que provee medidas de temperatura en grados Celsius, en códigos digitales de 9 a 12-bits. El DS18B20 tiene un rango de operación de -10°C a +85°C con una precisión de 0.5°C, asimismo el rango de temperatura de operación es de - 55°C a +125°C. Además, el DS18B20 puede alimentarse directamente de la línea de datos (parasite power), eliminando la necesidad de fuentes de voltaje externo. Otra característica del DS18B20 es que tiene un código de serie único de 64 bits llamado CODE ROM, que permite el funcionamiento de múltiples DS18B20s a través del mismo bus, con un máximo de treinta y dos dispositivos, así pues, es sencillo usar un microcontrolador para manejar muchos termómetros digitales distribuidos sobre un área grande.

Con este sensor se resuelven los problemas antes mencionados de controlar varios sensores a la vez, algunas otras características importantes del DS18B20 es que usa el protocolo 1-Wire exclusivo de Dallas Semiconductor que implementa la interface de comunicación usando una señal de control, la figura 3.1 muestra las funciones del diagrama a bloques del DS18B20. En este sistema de comunicación, el microcontrolador (dispositivo maestro) identifica y direcciona los dispositivos en el bus usando el código único de 64-bits de cada dispositivo.

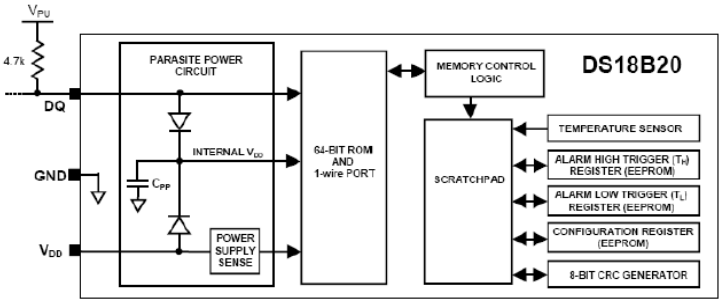


Figura. 3.1. Diagrama del sensor DS18B20 [imagen tomada de [8]].

La línea de control requiere una resistencia de polarización (pull-up) de todos los dispositivos que sean asociados al bus (DQ en caso del DS18B20) la descripción

de los pines se muestra en la Tabla 3.1. El fabricante recomienda una resistencia de polarización de 4.7KΩ.

Tabla 3.1. Pinout DS18B20 [tomada de [8]].

SO	µSOP	TO-92	Símbolo	Descripción
5	4	1	GND	**Tierra.**
4	1	2	DQ	**Entrada y salida de datos:** Interfase 1-wire. También provee alimentación al dispositivo cuando este es usado en modo de alimentación parasita.
3	8	3	V_{DD}	**Voltaje (opcional):** PIN V_{DD} debe ser puerta a tierra para operar en modo de alimentación parasita.

La memoria del DS18B20 está organizada como se muestra en la figura 3.2, esta es una memoria interna SRAM (Static Random Access Memory) con un almacenamiento no volátil EEPROM (Electrically-Erasable Programmable Read-Only Memory) de alta velocidad usada para retener datos temporalmente (scratchpad memory) lo cual se referirá de aquí en adelante con el nombre de memoria inmediata. La memoria de sólo lectura (ROM read-only memory) almacena el código de serie único, de 64 bits de los dispositivos. La memoria inmediata contiene el registro de temperatura de dos bytes que almacena la salida digital del sensor. Además, la memoria inmediata provee el acceso a los registros de disparo de alarma superiores e inferiores de un byte (TH y TL temperature high and temperature low) y, el registro de configuración de un byte.

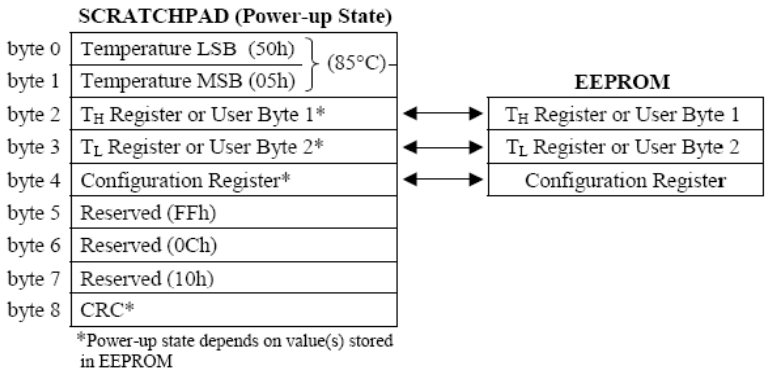


Figura. 3.2. Memoria del DS18B20 [imagen tomada de [8]].

El registro de configuración permite al usuario establecer la resolución de la temperatura para la conversión digital de 9, 10, 11, o 12 bits correspondiendo a los incrementos de 0.5°C, 0.25°C, 0.125°C, y 0.0625°C respectivamente, la resolución predefinida es de 12 bits. El dato de temperatura es calibrado en grados centígrados y es almacenado como un número de 16 bits en complemento a dos (signo extendido) en el registro de temperatura (figura 3.3). Los bits asignados con la letra S indican si la temperatura es positiva o negativa, cuando la temperatura es positiva S=0 y si la temperatura es negativa S=1. Si el DS18B20 se configura para la resolución de 12 bits, todos los bits en el registro de temperatura contendrán datos válidos. Los registros TH, TL y de configuración son no volátiles (EEPROM), así es que retendrán datos cuando el dispositivo este sin energía.

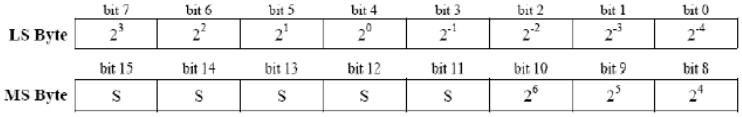


Figura. 3.3. Registro de temperatura [imagen tomada de [8]].

La Tabla 3.2 muestra ejemplos de datos de salida digital y la correspondiente temperatura leída para una resolución de conversión de 12 bits.

Tabla 3.2. Relación de datos de temperatura [tomada de [8]].

Temperatura	Salida Digital (Binario)	Salida Digital (Hexadecimal)
+125 °C	0000 0111 1101 000	07D0h
+85 °C	0000 0101 0101 0000	0550h
+25.0625 °C	0000 0001 1001 0001	0191h
+10.125 °C	0000 0000 1010 0010	00A2h
+0.5 °C	0000 0000 0000 1000	0008h
0 °C	0000 0000 0000 0000	0000h
-0.5 °C	1111 1111 1111 1000	FFF8h
-10.125 °C	11111 1111 0101 1110	FF5Eh
-25.0625 °C	1111 1110 0110 1111	FE6Fh
-55 °C	1111 1100 1001 000	FC90h

3.1.2 Sensor de Irradiación

Una vez con el sensor de temperatura se procedió a realizar la búsqueda de sensores un poco menos comunes como lo son los de irradiación, conocidos en general como piranómetro procedente de las palabras griegas fuego, hacia arriba y medida. Existen 3 tipos de piranómetros el denominado Eppley y el B/N los cuales usan una termopila que en su parte caliente están pintadas de un color absorbente de radiación y en la parte fría de un color menos absorbente; la característica principal de estos piranómetros es que pueden o no contar con un estabilizador de temperatura interna lo cual los hace más estables y de una mejor respuesta, pero esto los hace de un costo más elevado, con un alto mantenimiento y calibración. El tercer tipo de piranómetro es el que está basado en el uso de una célula fotovoltaica como detector. La respuesta espectral de estos sensores no incluye todo el espectro de interés, por lo que la calidad de la medida depende de las condiciones atmosféricas. Sin embargo, su rápida respuesta, ligereza y, sobre todo, menor costo en comparación con los piranómetros de alta calidad, hacen

que este tipo de piranómetros vayan ganando terreno en algunos campos de aplicación, como los relacionados con la agricultura o con la evaluación de plantas fotovoltaica [13]; como lo es el presente trabajo en el cual solo queremos obtener una representación de cómo es la evolución de la irradiación durante las horas de trabajo del panel solar.

Se realizó la búsqueda de este tipo de sensores en varias tiendas de electrónica digital y el que más se adecua a nuestra aplicación es el Spektron 210, el cual nos menciona en su hoja de datos que tiene un rango de 0 a 1000 W/m2 el cual es un sensor de silicio para la medición de la irradiación solar. El Spektron 210 entrega una tensión proporcional a la intensidad de la irradiación solar la cual es aproximadamente 75mV a 1000 W/m2.

3.2 Prueba de sensores

Una vez que los sensores fueron escogidos el siguiente paso fue probar el funcionamiento de estos, por lo cual se comenzó con el sensor de temperatura para el cual se debe de seguir una secuencia especifica de pasos para obtener los datos del sensor y trabajar con ellos en el microcontrolador.

La secuencia que se debe realizar para la comunicación entre el microcontrolador y el sensor DS18B20 es la siguiente: inicialización, comandos ROM y comandos de función del DS18B20. Es de gran importancia resaltar que se debe seguir la secuencia antes mencionada de forma correcta para acceder al DS18B20, de lo contrario el dispositivo no responderá si algún paso se perdió o está fuera de orden, solo se realiza una excepción de esta regla si se utilizan los comandos Search Rom [F0h] y Alarm Search [Ech], después de enviar cualquiera de estos comandos ROM, el maestro deberá retornar al primer paso de la secuencia.

La inicialización consiste en un pulso de restablecimiento transmitido por el dispositivo maestro seguido por un pulso de presencia transmitido por el esclavo. El pulso de presencia enviado en el bus, alerta al maestro que el circuito esclavo

está listo para operar. Los pulsos de inicialización y presencia se muestran en la figura 3.4.

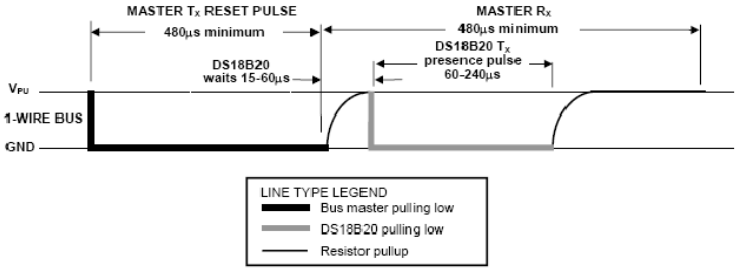


Figura. 3.4. Tiempo de inicialización [imagen tomada de [8]]**.**

El maestro transmite el pulso de inicialización colocando el bus 1-Wire en nivel bajo (0L) durante un tiempo de 480µs mínimo, a continuación, el maestro libera el bus y entra en modo de recepción (RX). Cuando el bus es liberado, la resistencia de polarización de 4.7kΩ pone el bus a nivel alto, si el DS18B20 detecta este borde creciente, espera de 15 a 60µs y entonces transmite un pulso de presencia, colocando un nivel bajo en el bus durante 240µs.

Enseguida que el maestro detecta el pulso de presencia puede enviar un comando ROM, estos comandos operan en CODE ROM del esclavo (cada DS18B20 tiene un código serial único de 64 bits almacenado en la ROM) con esto se permite al maestro seleccionar un dispositivo en específico si es que hay varios presentes en el bus 1-wire. Estos comandos también permiten al dispositivo maestro determinar cuántos y que tipos de circuitos están o si algún circuito ha experimentado una condición de alarma. El DS18B20 cuenta con cinco comandos ROM, cada comando tiene una longitud de 8 bits, el dispositivo maestro debe emitir un comando ROM adecuado antes de emitir un comando de función al DS18B20, los comando ROM son los mostrados en la tabla 3.3.

Tabla 3.3. Comandos ROM [tomada de [8]].

Comando	Hex
Search ROM	F0h
Read ROM Comand	33h
Macht ROM Comand	55h
Skip ROM Comand	CCh
Alarm Search Comand	ECh

El comando Search ROM (F0H) inicializa el circuito, el maestro realiza una búsqueda para identificar los CODE ROM de todos los dispositivos esclavos que se encuentran en el bus, con esto el maestro identifica el número de esclavos y sus tipos de circuitos. Mientras que el comando Read ROM (33H) se utiliza cuando existe un esclavo en el bus, con esto se permite al maestro leer el CODE ROM de 64 bits del esclavo sin necesidad de realizar el comando Search ROM.

Igualmente, el comando Macht ROM (55H) seguido del CODE ROM permite al maestro dirigirse a un dispositivo esclavo en un bus con varios dispositivos o un solo dispositivo (bus multidrop o single-drop). Solamente el circuito que contenga exactamente los 64 bits del CODE ROM responderá al comando de función emitido por el maestro, los demás esclavos en el bus esperarán por un pulso de inicialización.

Del mismo modo el comando Skip ROM (CCH) permite al maestro dirigirse simultáneamente a todos los circuitos, sin la necesidad de enviar algún CODE ROM de información; por ejemplo, el maestro puede hacer que todos los DS18B20 presentes en el bus realicen conversiones simultáneas de temperatura. Al enviar un comando Skip ROM seguido de un comando Convert T (44H). Solo si existe un dispositivo esclavo en el bus se utiliza este comando seguido del Read Scratchpad (BEH), permitiendo ahorrar tiempo al maestro que lea el dispositivo sin necesidad de enviar el CODE ROM, cuando hay más de un dispositivo puede causar una colisión de datos.

Enseguida que el maestro ha usado un comando ROM para comunicarse con el DS18B20, el maestro puede emitir uno de los comandos de función mostrados en la Tabla 3.4. Estos comandos permiten escribir y leer en la memoria inmediata del DS18B20, inicializar conversiones de temperatura y determinar el modo de polarización del dispositivo.

Tabla 3.4 Comando de función [tomada de [8]]**.**

Comando	Hex
Convert T	44h
Write Scratchpad	4Eh
Read Scratchpad	BEh
Copy Scratchpad	48h
Recall E2	B8h
Read Power Supply	B4h

El comando Convert T (44H) inicia una conversión, el resultado es almacenado en dos bytes en el registro de temperatura en la memoria inmediata, si el DS18B20 es alimentado por una fuente externa, el maestro puede emitir un intervalo de tiempo después del comando Convert T, enseguida el esclavo responde transmitiendo un 0 mientras está realizando la conversión y un 1 cuando ha finalizado la conversión.

Del mismo modo el comando Write Scratchpad (4EH) permite al maestro escribir tres bytes de datos en la memoria inmediata del DS18B20, el primer byte de datos es escrito dentro del registro TH, el segundo en el registro TL y el tercero en el registro de configuración. Los tres bytes se deben escribir antes que el maestro emita el pulso de inicialización, caso contrario los datos pueden ser dañados.

El comando Read Scratchpad (BEH) permite al maestro leer los datos de la memoria inmediata, puede emitir un pulso de inicialización al terminar la lectura en cualquier momento si sólo parte de la memoria inmediata es necesaria. Mientras que el comando Copy Scratchpad (48H) realiza una copia del contenido de la memoria inmediata (registros TH, TL y configuración) a la EEPROM.

Igualmente, el comando Recall E2 (B8H) recuerda los valores de alarma (TH, TL) y datos de configuración de la EEPROM y coloca los datos en bytes de dos, tres y cuatro respectivamente en la memoria inmediata. De igual forma el comando Read Power Supply (B4H) se envía seguido por un intervalo de tiempo para determinar, si en el bus algún DS18B20 está usando alimentación parasita (parasite power). [8]

Para facilitar el uso de estos sensores se consiguió una librería la cual solo nos pide el pin donde se conectarán los sensores y con ello nos entrega por medio de funciones la lectura para poder nosotros usarla como nos convenga ya sea entero o float. Una vez con la librería se procedió a probar la funcionalidad de dichos sensores DS18B20 y el microcontrolador que se uso fue el PIC18f2550 de Microchip el cual cuenta con características útiles para el desarrollo del proyecto como lo son el uso del puerto USB [15].

Como primera prueba de que el sensor está funcionando bien, se hizo una simulación en ISIS Professional, como se puede observar en la figura 3.5, en la cual con el uso de una LCD se muestra en la temperatura que el PIC18f2550 adquiere del sensor DS18B20.

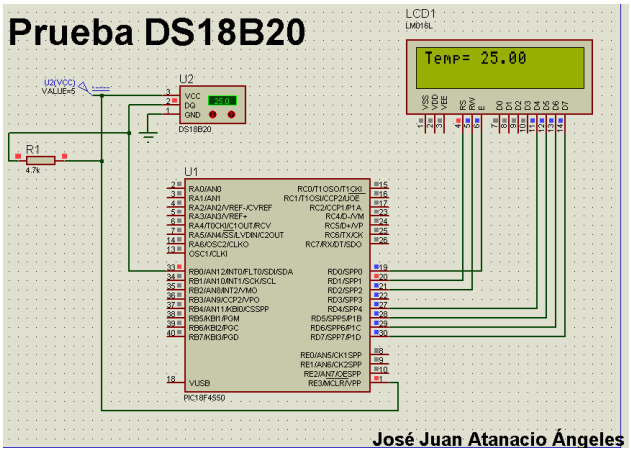


Figura. 3.5. Simulación en ISIS Profesional del sensor DS18B20 [creación propia]**.**

Ya con la simulación funcionando se realizaron pruebas en físico, en las cuales se armó el mismo circuito en un protoboard obteniendo los mismos resultados.

3.3 Montaje de los sensores en el Sistema FV.

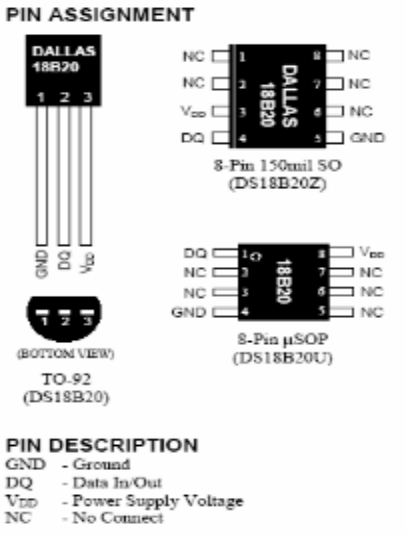


Figura. 3.6 Asignación de pines del sensor DS18B20 [imagen tomada de [8]].

Una vez que tenemos los sensores funcionando podemos realizar el montaje de los sensores DS18B20 en cada uno de los paneles del sistema FV. Pero dado que el sensor DS18B20 consta de tres pines en encapsulado TO-92 (ver figura 3.6). Éste no está diseñado específicamente para funcionar a la intemperie, de ahí que se haya tenido que diseñar y construir un encapsulado específico (ver figura 3.7) para: 1) albergar al sensor protegiéndolo de los agentes atmosféricos externos y 2) permitir su perfecta adherencia a la superficie de un panel fotovoltaico. Por supuesto, este encapsulado adicional, aunque debe proteger al sensor de la intemperie, debe permitir a la vez una excelente transmisión térmica, de modo que las medidas de temperatura no queden falseadas por el encapsulado.

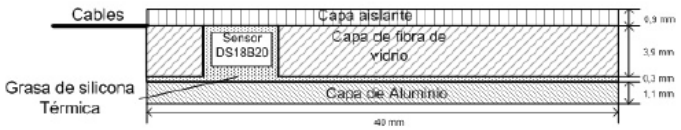


Figura. 3.7 Sección transversal del sensor diseñado para su utilización en el exterior [imagen tomada de [12]].

En la figura 3.7 se observa el diseño y montaje embebido del sensor en una cápsula rectangular de varias capas. La capa superior es de material aislante para proteger las conexiones eléctricas de los agentes meteorológicos externos. La capa inferior que está en contacto con la superficie del panel fotovoltaico es metálica, de aluminio, lo cual evita la corrosión y facilita una excelente transmisión térmica desde la superficie del panel fotovoltaico al interior del encapsulado del sensor. A continuación de esta capa metálica hay otra de grasa de silicona térmica que por un lado hace hermético el encapsulado y, por otro, conecta térmicamente el sensor con la superficie del panel fotovoltaico.

Por último, para terminar de conformar el encapsulado y darle la rigidez y tamaño adecuados, se utiliza una capa de relleno de fibra de vidrio. En la figura 3.8 se muestra el dispositivo real construido.

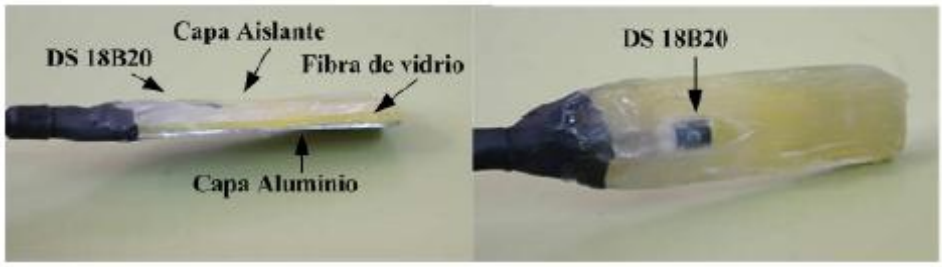


Figura. 3.8 Sensor encapsulado para aplicaciones en exteriores [imagen tomada de [12]]

Ahora bien, con el encapsulado listo se procedió a colocar los sensores en los paneles, la distribución que se realizara es para abarcar la mayor área posible y así tener datos más confiables sobre las lecturas de temperatura.

Para el caso del sensor de irradiación lo único que se tuvo que realizar fue crear una etapa de amplificación ya que la señal de salida del sensor es de 75mV en su máximo valor, lo cual haría que tuviéramos muy poco rango de lectura, por lo que se decidió colocar un amplificador operacional en configuración no inversora [20] y con ello lograr amplificar la señal de 75mV hasta obtener una salida aproximada de 5V.

3.4 Adquirir datos en el SATD

En los días posteriores al acondicionamiento de los sensores, se llevó a cabo el SATD el cual tiene el diagrama a bloques de la figura 3.9, el cual su objetivo básico es la integración de los diferentes recursos que lo componen: Transductores de diferentes tipos y naturaleza, multiplexores, amplificadores, conversores ADC y DAC, además el uso del microcontrolador como CPU del SATD diseñado, utilizando de este microcontrolador todas sus prestaciones:

interrupciones, temporizadores, comunicación serie así como hacer uso de memorias y puertos externos y creando con todo ello un sistema que se encargue de una aplicación específica como es obtener una variable para una posterior control, transmisión o simplemente registro [22].

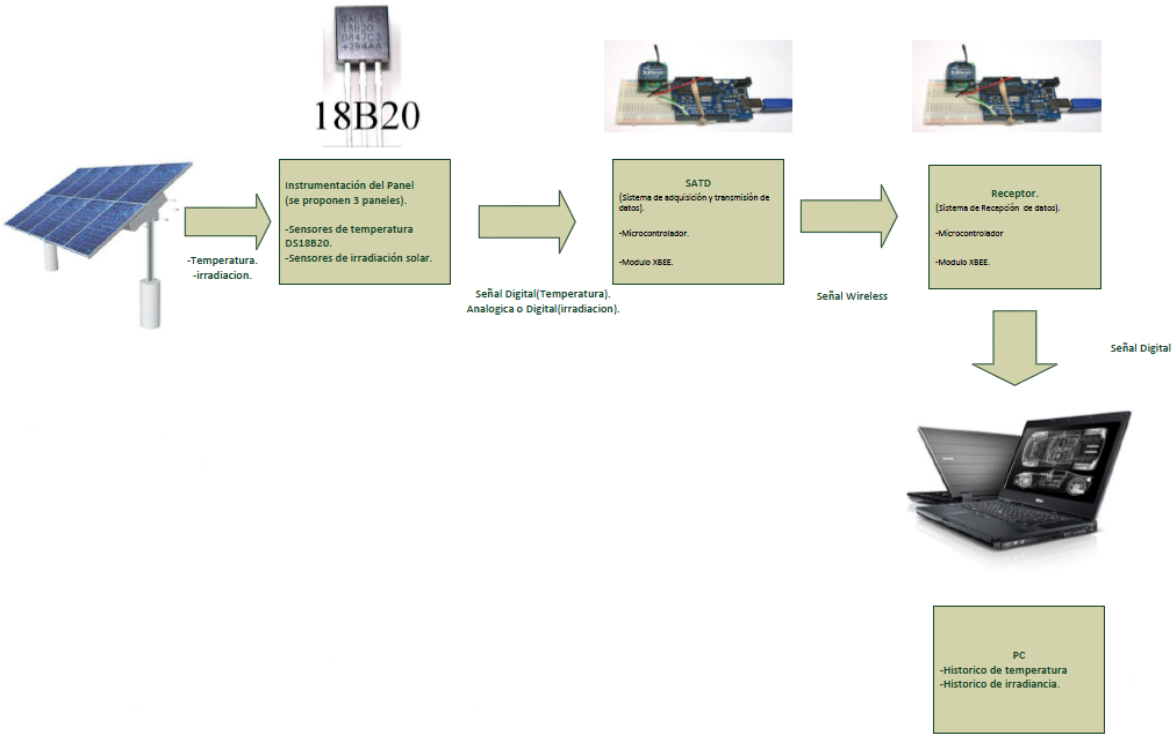


Figura. 3.9. Diagrama a bloques el SATD [creación propia]**.**

Con las pruebas realizadas anteriormente del correcto funcionamiento del DS18B20 y el sensor de irradiación, la única tarea que faltaría sería la de mandar estos datos recibidos de los sensores vía Wireless hacia el servidor.

3.5 Realizar comunicación Wireless hacia el servidor.

Para realizar la comunicación Wireless entre el SATD y el servidor, se tiene que usar algún tipo de comunicación inalámbrica. Las comunicaciones inalámbricas son un conjunto de sistemas de comunicación y tecnologías asociadas que

emplean el espectro radioeléctrico como vehículo de comunicación. Debido a esto, los extremos de comunicación no se encuentran unidos por ningún medio físico, estos dispositivos solo se encuentran en el emisor y receptor, los cuales pueden ser: antenas, teléfonos celulares, computadoras personales, PDA (Personal Digital Assistant), etc. Por lo que se optó por usar las comunicaciones ZigBee las cuales están basadas en el estándar IEEE 802.15.4 de redes inalámbricas de área personal WPAN (Wireless Personal Area Network) y su objetivo principal son las aplicaciones que requieren comunicaciones seguras con baja tasa de envío de datos y el bajo consumo de energía y por ende larga vida útil de las baterías, como lo es nuestro proyecto.

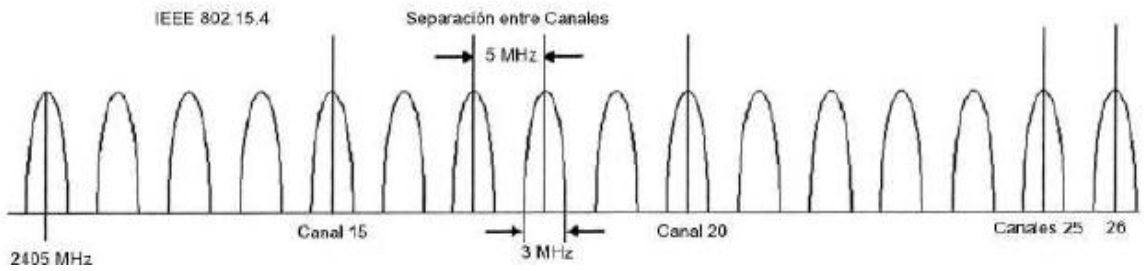


Figura. 3.10 Canales para el protocolo de modelos XBee [imagen tomada de [11]]

ZigBee es un protocolo de comunicaciones inalámbricas con velocidades comprendidas entre 20 Kb/s y 250 Kb/s. El alcance depende de la potencia de transmisión del módulo, tomando como ejemplo los módulos fabricados por la empresa Maxtream de 1mW de potencia, los rangos de alcance son de 30 metros en interiores y 100 metros en exteriores. A diferencia de Bluetooth, este protocolo no utiliza espectro extendido por salto de frecuencia FHSS (Frecuency Hooping Spread Spectrum), sino que realiza las comunicaciones a través de una única frecuencia, es decir de un canal. Normalmente puede escogerse un canal entre 16 posibles, sin embargo, los valores se asignan desde el 11 al 26 tal como se muestra en la figura 3.10. Una red ZigBee puede estar formada por hasta 255

nodos, los cuales la mayor parte del tiempo se encuentran en modo de bajo consumo (Sleep Mode), permitiendo consumir menos energía que las otras tecnologías inalámbricas. ZigBee puede usar las bandas libres de 2.4 GHz a nivel mundial, 868 MHz en Europa y 915 MHz en Estados Unidos. Un sensor equipado con un receptor/transmisor (transceiver) ZigBee puede ser alimentado con dos pilas AA durante al menos seis meses y hasta dos años. Las topologías que se pueden configurar serían las siguientes: estrella, punto a punto, malla y árbol.

Ahora bien, los módulos que se usaran para el desarrollo de este proyecto son los denominados XBee S1 fabricado por Digi mostrado en la figura 3.11, puede ser ajustado para usarse en redes de configuración: punto a punto, punto a multipunto o red de pares (peer to peer).

Figura. 3.11 Modulo XBee S1 de Digi [imagen tomada de [11]].

En la figura 3.12 se muestran las conexiones externas mínimas para el funcionamiento del módulo XBeePro, específicamente este dispositivo requiere una alimentación de 2.8 a 3.4 volts, la conexión a tierra y las líneas de transmisión de datos (TXD y RXD). La comunicación con un micro controlador se realiza mediante el protocolo UART (Universal Asynchronous Receiver-Transmitter). El control de flujo (RTS y CTS) se utiliza cuando existe el envío de una gran cantidad de información utilizando más pines del módulo XBeePro, para propósitos de este proyecto no se permite el uso de control de flujo y es necesario desactivar esta opción en el programa de configuración del módulo XBeePro.

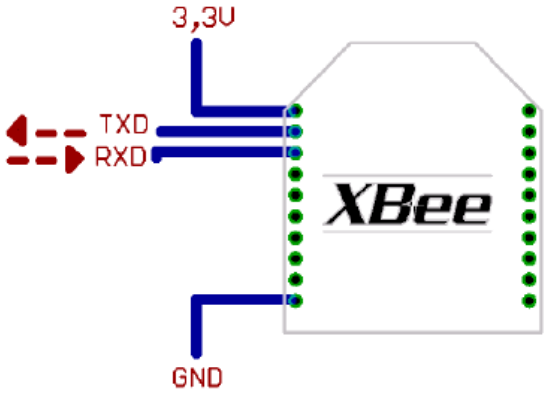


Figura. 3.12 Conexiones mínimas del módulo XBee [imagen tomada de [10]]**.**

Como se observa en la figura 3.13 el circuito de acoplamiento con el PIC18f2550 es muy sencillo, por lo cual no tiene dificultad alguna, es importante destacar que el módulo XBee entrará en modo recibir/transmitir cuando le llega algún paquete RF (Radio Frequency) a través de la antena (Receive Mode), o cuando se envía información serial a través del pin tres (UART Data in) la cual será transmitida (Transmit Mode). Por otro lado, el PIC18f250 entrara en modo de bajo consumo (Sleep Mode) cuando el módulo no se encuentra en uso, es decir entra en un gasto mínimo de energía [9].

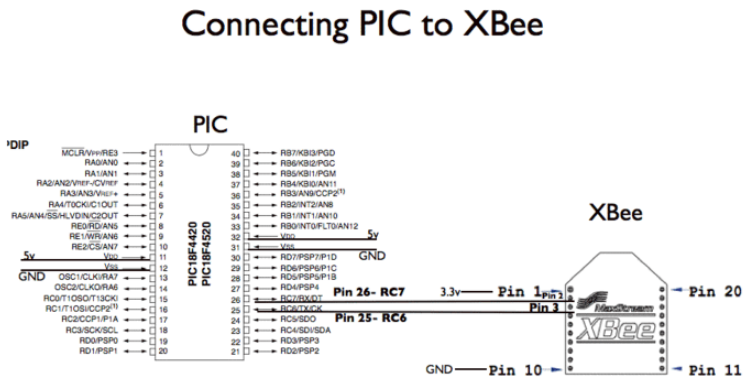


Figura. 3.13 Esquemático módulo XBee y PIC [imagen tomada de [19]]**.**

3.6 Adquirir datos en la PC.

El último paso para tener una comunicación completa es tener dos módulos XBee uno en el sistema FV y otro en el la PC. El circuito del módulo XBee para el sistema FV es sencillo dado que solo sería un acople directo entre el XBee y el PIC; el circuito para la PC también es algo sencillo dado que con el uso de otro módulo XBee y con el PIC18F2550 se puede tener acceso al uso del USB, lo cual nos facilitaría la tarea del adquirir los datos en la PC; dado que no se tendría que desarrollar otro circuito para pasar los datos del PIC18f2550 a la PC como sería el caso de usar la comunicación RS232, así que los módulos XBee junto con los PICs solo serían un intermediario entre la sistema FV y la PC.

3.7 Desarrollar SCADA apropiado.

Ya con los datos de temperatura e irradiación en la PC ahora la etapa siguiente es tomar esos datos y usarlos. Se desarrolló una aplicación en Visual Basic de Excel, en la cual podemos observar una gráfica con la respuesta de la temperatura de cada sensor instalado y otra con la irradiación, y además, llevar una base de datos de dichas variables con su determinada fecha y hora en una hoja aparte; a su vez estos datos se mandan al controlador el cual devuelve la señal de control la cual debe de ser regresada por medio nuevamente del uso de la red: PC, USB, PIC1, Módulo XBee1, Módulo XBee2, PIC2 y por último llegar a la etapa de potencia para activar el actuador de efecto Peltier-Seebeck y por ende controlar la temperatura del sistema FV.

3.8 Convertir a señal de potencia

En esta parte del proyecto la señal enviada por el control de la PC llegara al PIC del Panel FV activara un puente H [23] el cual normalmente se usa para controlar el giro de un motor de DC, lo cual es cambiar la polaridad de dicho motor; para nuestro caso como ya se describió lo que requerimos es precisamente eso,

cambiar la polaridad de la alimentación de las celdas Peltier-Seebeck y con ello se puede calentar o enfriar el panel solar.

4. RESULTADOS

Al finalizar este proyecto lo que se esperaba era tener implementada la electrónica necesaria para realizar la comunicación inalámbrica y transmitir los datos de las variables desde el panel FV hasta el servidor; éste debía tener la aplicación en la cual los datos de temperatura e irradiación del panel FV se podrían observar gráficamente, además esta aplicación debería crear una base de datos para el modelado del panel FV que servirá para alimentar de datos al controlador de temperatura del panel FV desarrollado en la FI. Finalmente, se esperaba tener la transmisión de la señal de control desde el servidor hasta el panel FV para poder mantener la temperatura en un rango de temperatura requerido en la superficie del panel.

4.1. Resultados Finales

4.1.1. Pruebas de Comunicación bidireccional entre el PIC y la PC.

Lo primero que se buscó antes de comenzar con las pruebas de los sensores fue probar la comunicación entre el PIC18F2550 y la PC específicamente a la plataforma de Excel por medio de Visual Basic for Aplications (VBA), lo cual sirvió de mucho para ver y analizar los datos obtenidos tanto de temperatura como de irradiación procedentes de los sensores colocados en el panel FV y con ello poder alimentar el controlador el cual nos daría la señal de control para las celdas Peltier-Seebeck.

Lo primero que se hizo fue buscar e instalar el drive para usar el control NETCommOCX en VBA con el cual podemos obtener los datos que nos envía el PIC18F2550 por medio del puerto USB y manejarlos como si fuesen datos seriales. La instalación se muestra a continuación.

Primero que nada, debemos descarga el drive [17]:

Figura. 4.1 Pagina de Drive NETCommOCX [captura de pantalla]**.**

Para el proyecto se descargó el Instalador para Windows 7, una vez descargado solo se descomprime, se corre el Setup, se asigna la siguiente dirección C:\Program Files\Hard & Software y listo tendremos el control NETCommOCX en la librería completamente dentro de Excel.

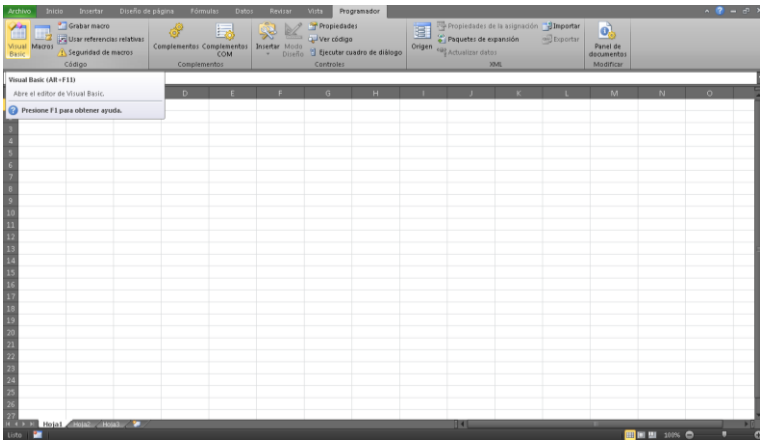


Figura. 4.2 Abrir VB desde Excel [creación propia]**.**

Para habilitar este nuevo control en Excel lo que se debe hacer es abrir el Excel, después el VBA (ver figura 4.2).

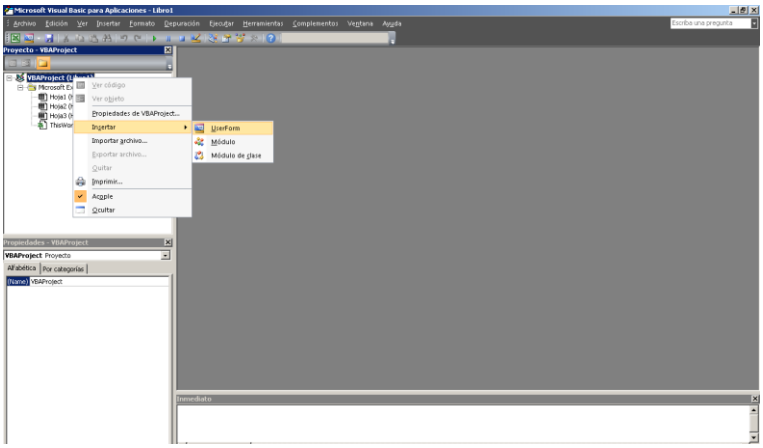


Figura. 4.3 Abrir UserForm desde el VB de Excel [creación propia]**.**

Una vez dentro de VBA nos posicionamos sobre el nombre del Libro damos clic derecho, después Insertar, y clic en UserForm como se muestra en la figura 4.3. Ya con el UserForm creado, seleccionamos el menú Herramientas y después en Controles adicionales, ahí buscamos el control NETCommOCX.NETComm lo seleccionamos y damos en Aceptar.

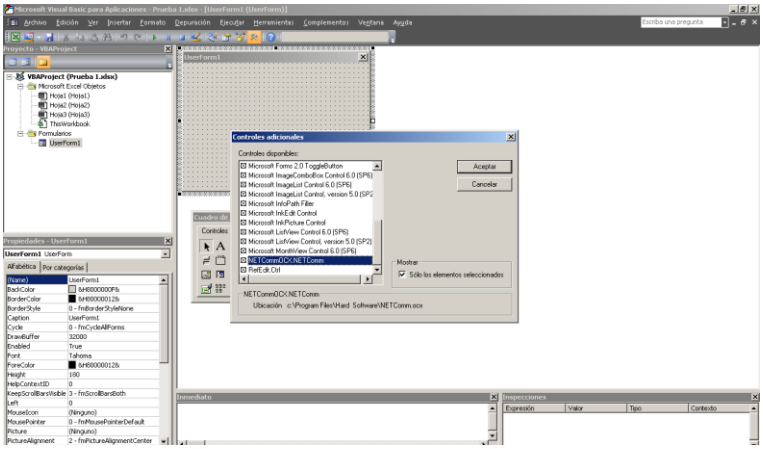


Figura. 4.4 Activar nuevo Drive [creación propia]**.**

Con la aplicación de los pasos anteriores, en nuestro Cuadro de Herramientas se observará un nuevo control llamado NETComm (ver figura 4.4). Ahora podemos cerrar y eliminar el UserForm, con ello podemos regresar a la Hoja1, buscamos el

menú Programador, después Insertar y ahí buscamos dentro de los Controles ActiveX el submenú Más Controles y seleccionamos como se muestra en la figura 4.5.

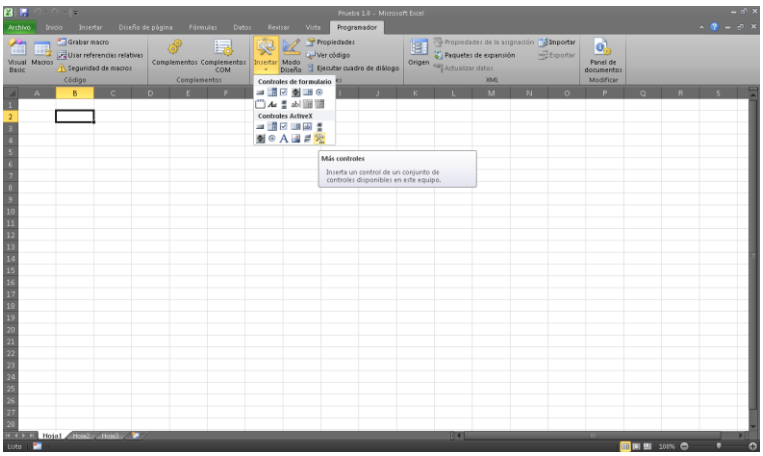


Figura. 4.5 Habilitar nuevo Drive desde pestaña Programador de Excel [creación propia]**.**

En la nueva ventana buscamos el NETCommOCX.NETComm y damos Aceptar, con lo cual podremos dibujar un cuadrado en cualquier parte la hoja, con lo que activaremos las propiedades de este control dentro de esta hoja (ver figura 4.6).

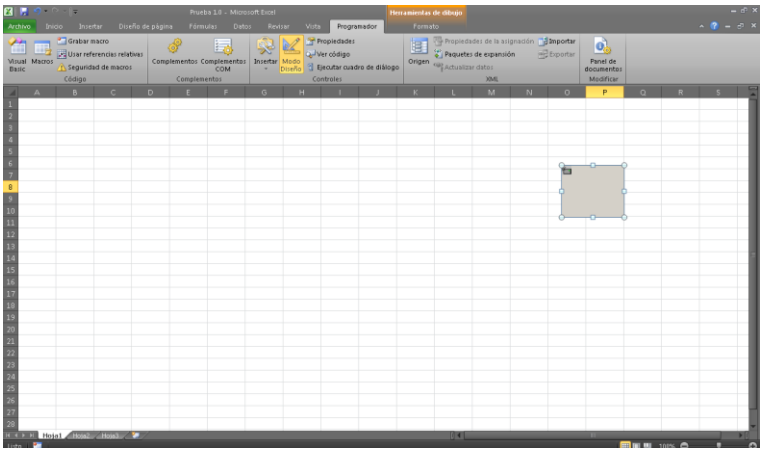


Figura. 4.6 Habilitar nuevo Drive y colocarlo en la hoja de Excel deseada [creación propia]**.**

Para crear el programa, lo primero que se realizó fue la creación de los botones y las celdas de entrada por llenar, como se muestra en la figura 4.7.

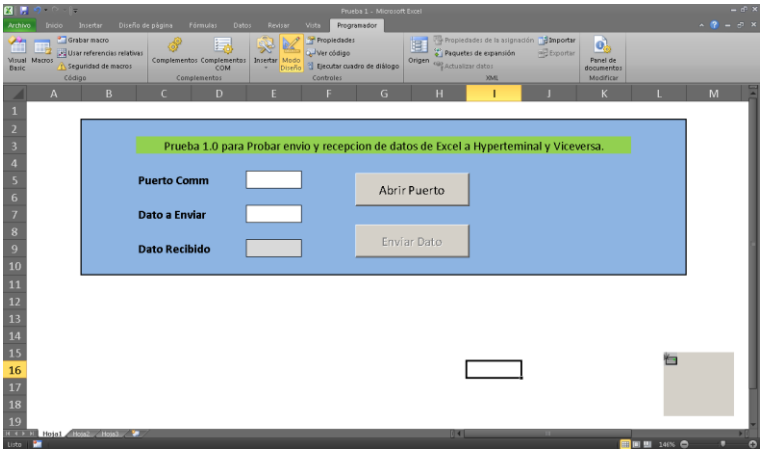


Figura. 4.7 Pantalla de Pruebas Excel para recepción y envió de datos [creación propia]**.**

Ya con las entradas y salidas definidas se creó el programa para poder enviar un dato a la hyperterminal y viceversa. La lógica del programa es la siguiente: primero se debe de abrir el puerto para lo cual se usa uno de los botones el cual al ser presionado abrirá, configurara el puerto con el COM elegido y habilitara el siguiente botón que es el de envió; y en el caso cuando se desea cerrar el puerto terminara la conexión y deshabilitara el botón 2 y cambiara el letrero del botón 1.

Ahora que tenemos conexión con el puerto COM elegido podemos comenzar a enviar datos con el botón 2 el cual fue habilitado al presionar el botón 1. Por último, si el control NETCommOCX detecta una llegada de dato se activa la rutina que se presenta en el Apéndice A.

Ahora ya con la aplicación desarrollada se procedió a probar la conexión con el uso la hyperterminal que nos proporciona el programa PIC C Compiler (Siow) (ver figura 4.8).

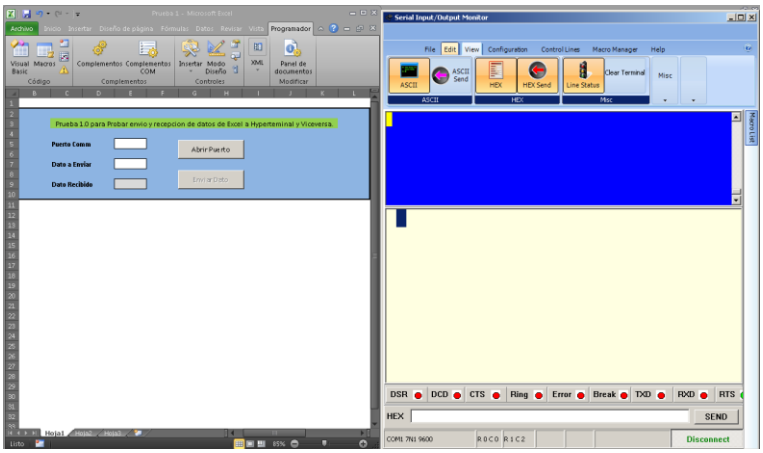


Figura. 4.8 Prueba conexión Excel vs Hyperterminal [creación propia]**.**

Lo primero que se debe de hacer es definir el COM deseado y a continuación se presiona el botón “Abrir Puerto”, con lo cual se activara el botón “Enviar Dato” y en la hyperterminal se observa que se encienden los botones DSR y DCD como se observa en la figura 4.9.

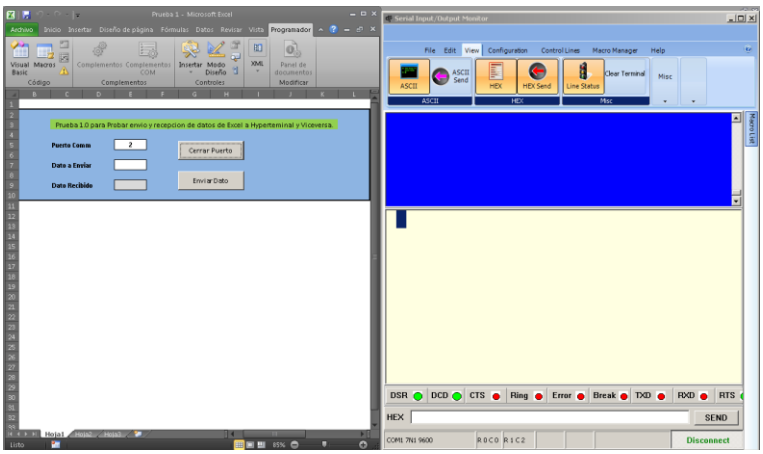


Figura. 4.9 Abrir puerto COM colocando el puerto COM y presionando Abrir puerto [creación propia]**.**

Ahora, colocamos el dato a enviar por el puerto para este caso es una “A” y presionamos “Enviar Dato”, y al observar la Hyperterminal se ve la “A”. Con lo cual queda comprobado el envío de datos (ver figura 4.10).

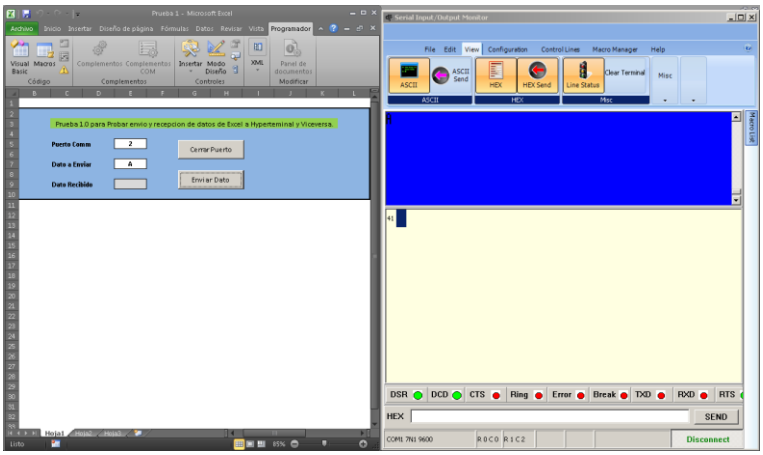


Figura. 4.10 Envió de A hacia hyperterminal colocando la letra en Dato a Enviar y presionar Enviar Dato [creación propia]**.**

A continuación, se prueba la recepción de datos en el programa de VBA, solo se coloca el dato en este caso una “B” en el puerto de salida de la hyperterminal y presionamos Send, al observar la celda de “Dato Recibido”, observamos la “B” con lo que comprobamos también la recepción de datos del programa (ver figura 4.11).

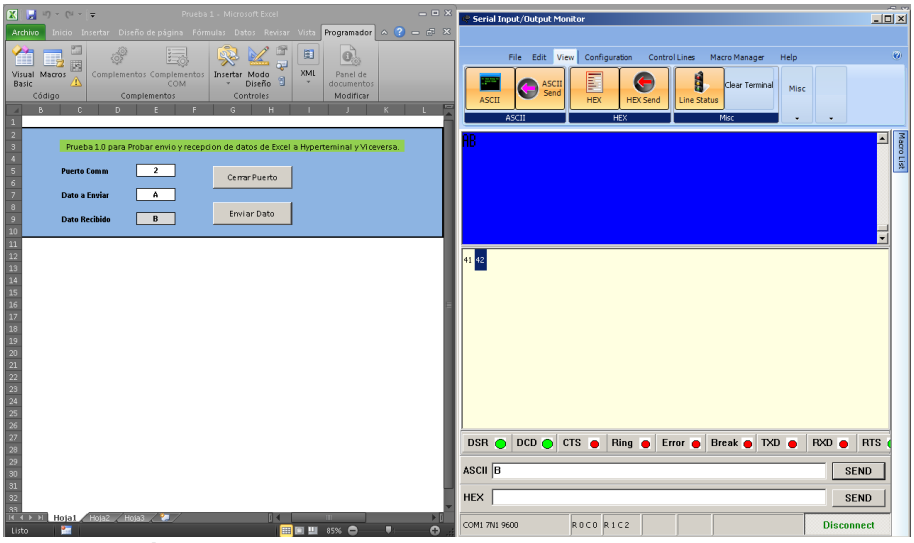


Figura. 4.11 Envió de B hacia Excel, colocar la letra en ASCII y presionar SEND [creación propia]**.**

Ahora se continuó con las pruebas de comunicación desde el PIC hacia la PC y viceversa, para lo cual se requirió primero que nada tener instalado en nuestro PC el software PIC C Compiler en el cual crearemos el programa a instalar en el PIC, para nuestro caso se usó el PIC18F250 por ser compacto y tener la opción de USB, se usó la conexión básica que describe su hoja técnica como se muestra en figura 4.12.

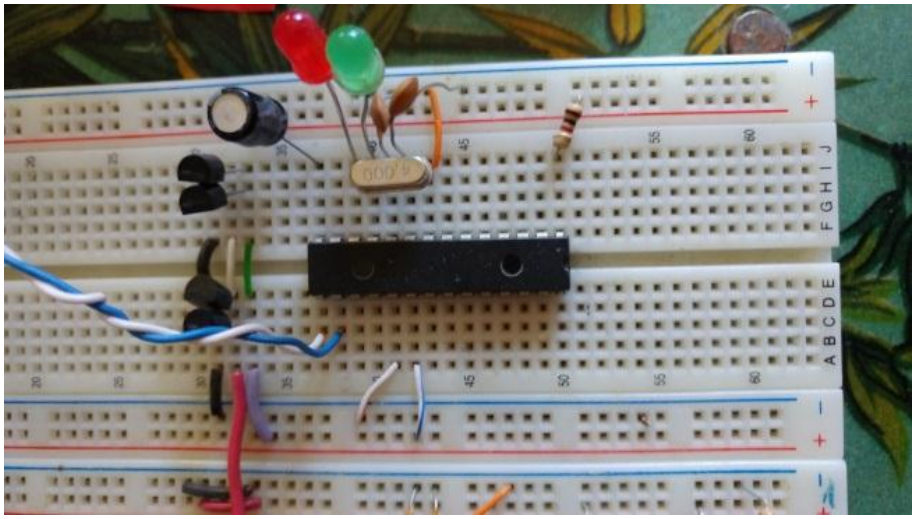


Figura. 4.12 Configuración básica de PIC para uso de puerto USB, capacitor en C3, LEDs de estado en C0 y C1; cristal de 4MHz, capacitores de 33pF, resistor de habilitación y cables de comunicación en D0 y D1 [creación propia]**.**

Para realizar la comunicación entre el PIC y la PC se usó una opción de comunicación que nos proporcionan los PICs en la cual podemos emular un puerto serie virtual mediante el uso de la librería usb_cdc.h, con lo cual al conectar nuestro PIC a la PC está lo identificara como un puerto serie (puerto COM).

Primero que nada, se realizó un programa fácil en el cual solo enviaremos de la PC un “0” mientras no se reciba alguna otra opción cada 2 segundos. Si se recibe un “1” se para el envío de datos, y se recibe “2” se enviará el “0” nuevamente. Finalmente, si se recibe un “3” se encenderá el LED que está conectado en B7 y si llega un “4” se apagará el LED.

Una cuestión muy importante es el PLL del PIC. Necesitamos que a nuestro micro le lleguen sólo 4MHz por lo que hay que usar un prescaler. Para ello en CCS se usa la sentencia PLLX, donde X significa la división de nuestro clock. Así pues, si tenemos un cristal de 20MHz, el prescaler tendrá que ser 20 / 4 = 5 entonces es PLL5. Si por el contrario nuestro cristal es de 12MHz sería PLL3. Para nuestro caso el cristal es de 4MHz por lo cual el PLL es debe de ser PLL1. En el Apéndice B se muestra el programa para el PIC.

Es importante resaltar que para que el PIC sea reconocido por la PC, se debe de tener hermanados los descriptores VID y PIC tanto en la librería usb_desc_cdc.h como el drive mchpcdc.inf. El primero se encuentra en los drives de PIC C y debe de tener los valores que se muestran en la figura 4.13.

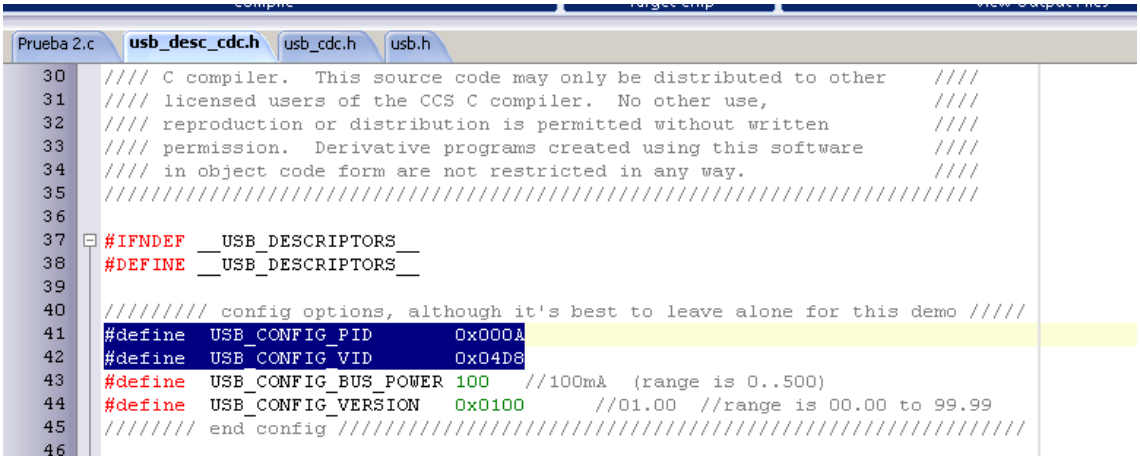


Figura. 4.13 Descriptores de PIC vs PC en librería usb_desc_cdc.h [creación propia].

Además, al final de estas librerías también se puede cambiar el nombre del producto para este caso quedo como Serial J.J (ver figura 4.14).

```
  //string 2 - product
        24, //length of string index
        USB_DESC_STRING_TYPE, //descriptor type 0x03 (STRING)
        'S',0,
        'E',0,
        'R',0,
        'I',0,
        'A',0,
        'L',0,
        ' ',0,
        'J',0,
        '.',0,
        'J',0,
        '.',0
};
#endif    //!defined(USB_STRINGS_OVERWRITTEN)
```

Define
USB_DESC_STRING_TYPE 0x03

Figura. 4.14 String a mostrar en PIC al ser conectado a modificarse en librería usb_desc_cdc.h [creación propia]**.**

Para el drive mchpcdc.inf lo podemos crear en un bloc de notas y llenarlo con la información que se encuentra en el Apéndice B y modificar las letras en negrita de los nombres, descriptores VID y PIC; y a continuación guardarlo con el nombre mchpcdc.inf en el siguiente link: C:\Windows\System32\Drive CDC. Ahora podemos programar el PIC y conectarlo. Al conectarlo abrimos el administrador de dispositivos y nos aparecerá en otros dispositivos, ya que no está reconocido, ahora daremos clic derecho sobre él y clic en actualizar software de controlador como se muestra en la figura 4.15

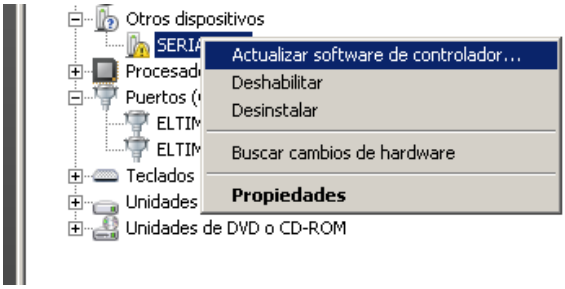


Figura. 4.15 Presionar Actualizar Software de controlador para poder cargar el drive [creación propia]**.**

A continuación, buscar software en equipo (ver figura 4.16):

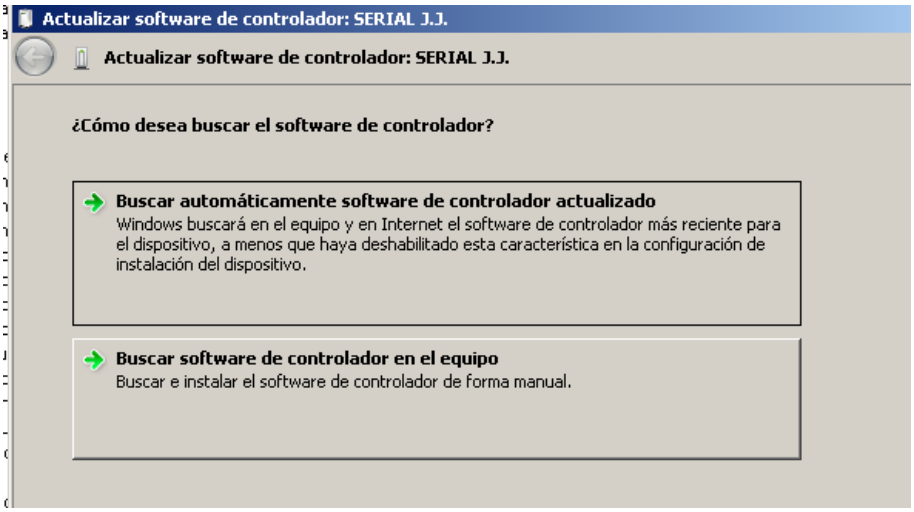


Figura. 4.16 Buscar Software en el Equipo [creación propia]**.**

Ahora damos clic en Elegir en una lista de controladores (ver figura 4.17).

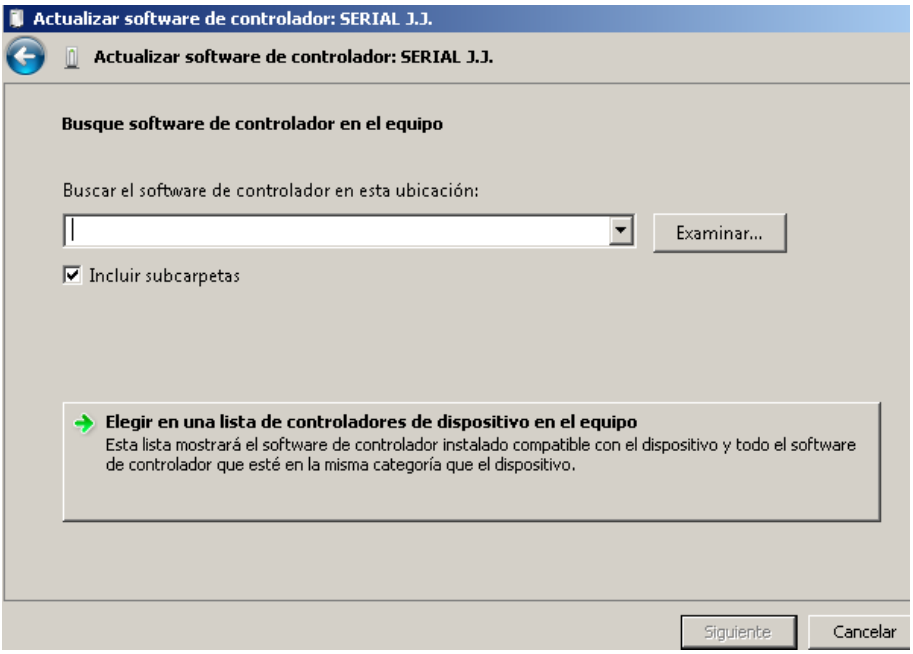


Figura. 4.17 Elegir en lista en el Equipo [creación propia]**.**

Ahora como se muestra en la figura 4.18 seleccionamos Mostrar todos y siguiente.

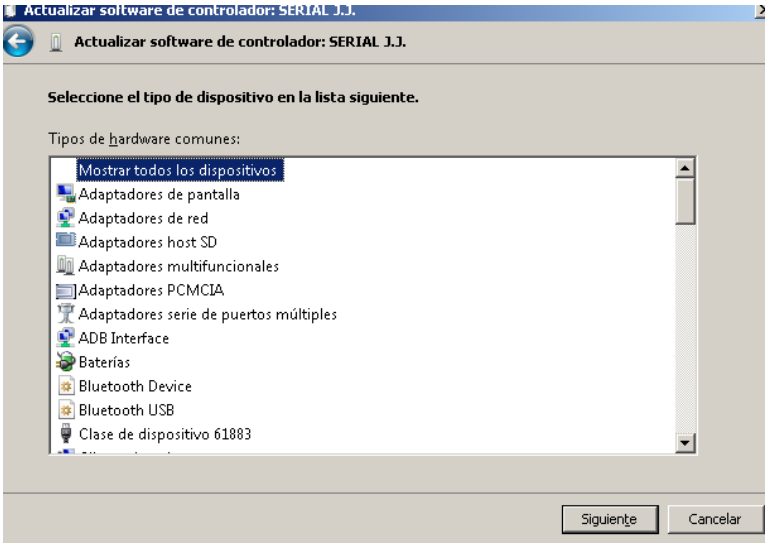


Figura. 4.18 Presionar Mostrar todos los dispositivos [creación propia]**.**

Ahora en presionamos en Usar disco (ver figura 4.19).

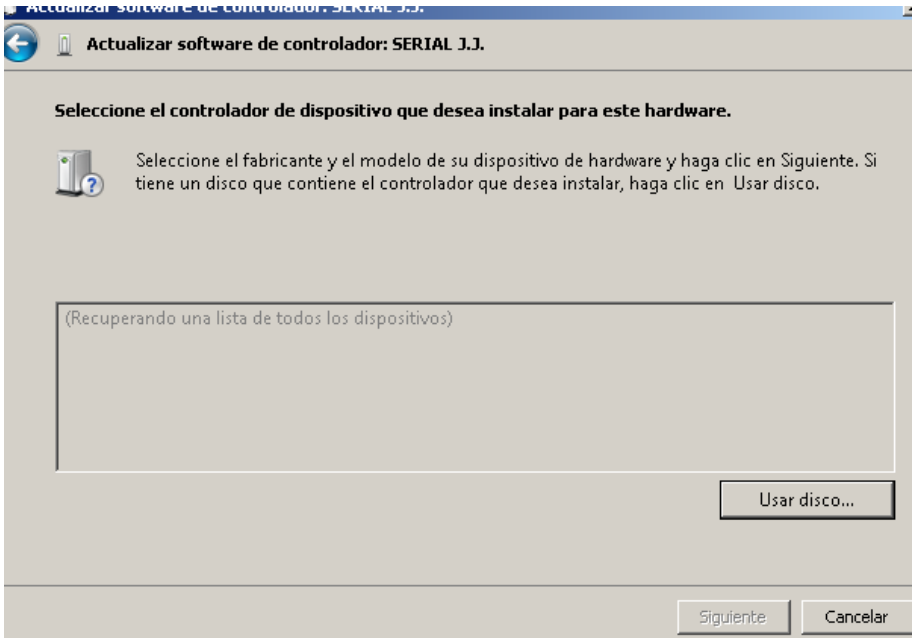


Figura. 4.19 Presionar en Usar disco para poder cargar el drive [creación propia]**.**

Y como se muestra en la figura 4.20 buscamos en el link donde los guardamos:

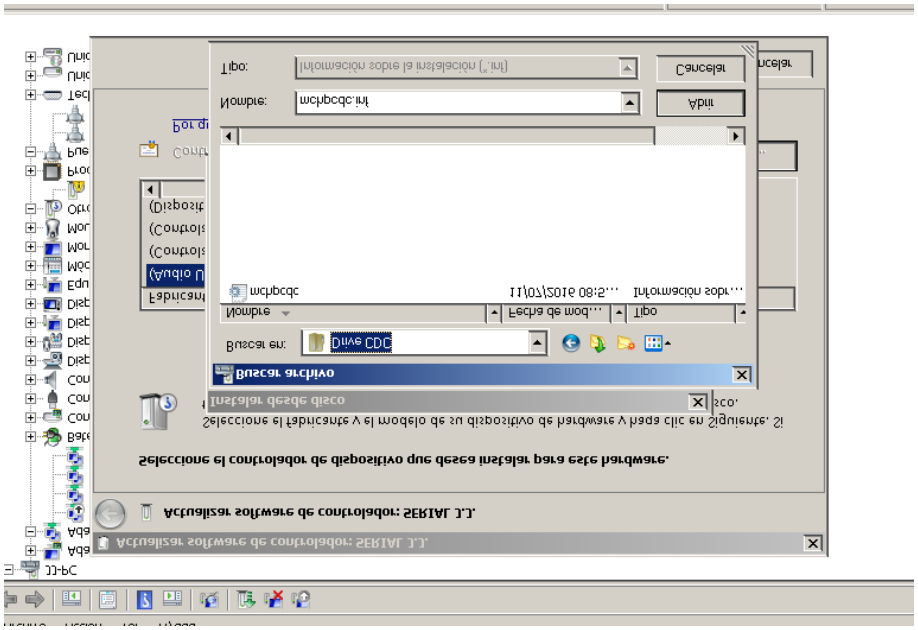


Figura. 4.20 Buscar Ruta de Drive [creación propia]**.**

Abrir, aceptar y siguiente, y esperamos hasta que se instale dando los permisos necesarios como se muestra en la figura 4.21.

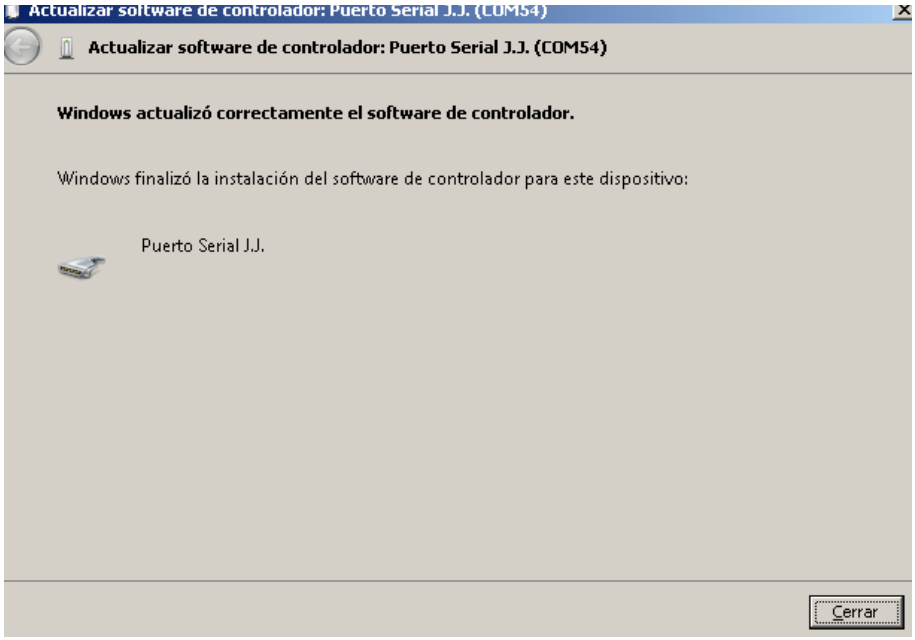


Figura. 4.21 Se Actualizo Correctamente [creación propia]**.**

Ahora en administrador de dispositivos lo encontraremos como un puerto COM (ver figura 4.22):

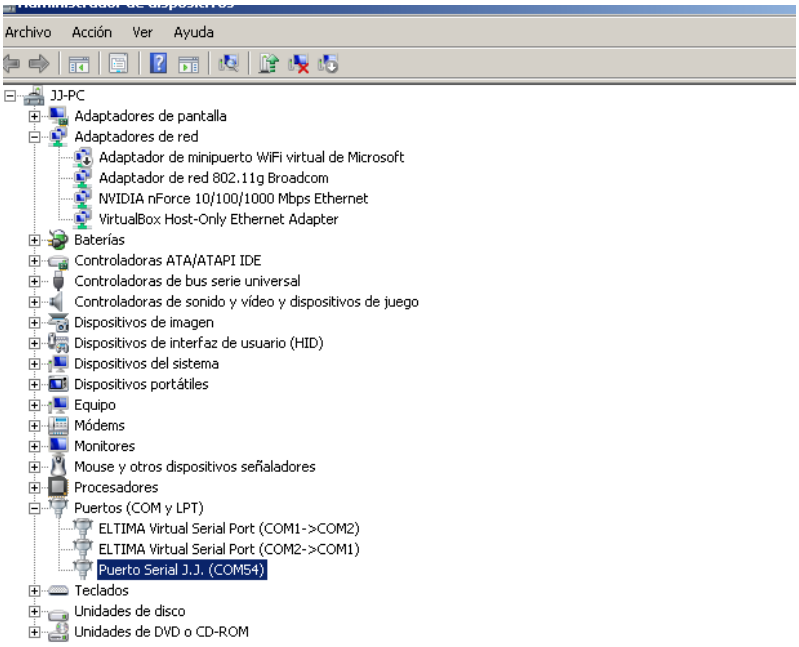


Figura. 4.22 Nuevo Puerto Serial [creación propia]**.**

Con lo cual si ahora conectamos el PIC a la PC se identificará como un puerto COM y podremos proceder a realizar la prueba. Cabe resaltar que esta operación solo se debe de realizar una vez por cada puerto.

Ya que se realizó lo anterior se procedió a conectar el PIC para realizar las pruebas de comunicación con la aplicación en Excel. Al conectar el PIC en la PC podemos observar en el PIC como primeramente se enciende el LED rojo y al ser enumerado (configurado por la PC) se apaga este LED y se enciende el LED verde (ver figura 4.23).

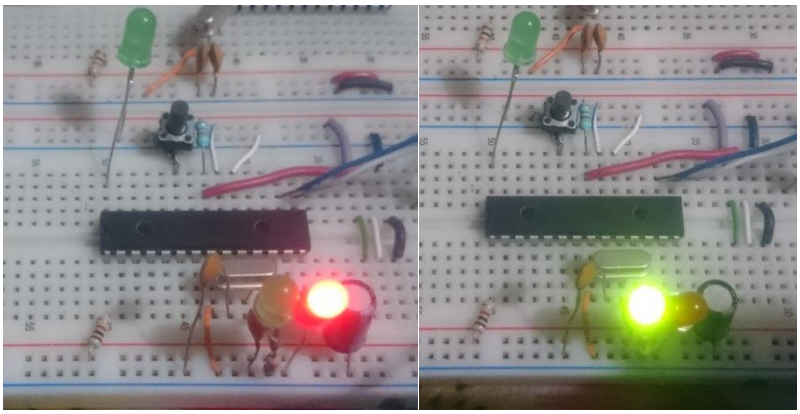


Figura. 4.23 LEDs de Estado, Rojo PIC no configurado por la PC y Verde cuando se ha configurado, [creación propia]**.**

Abrimos el programa que teníamos para las pruebas internas con la hyperterminal, colocamos el puerto COM con el cual fue identificado el PIC y damos en Abrir Puerto como se muestra en figura 4.24.

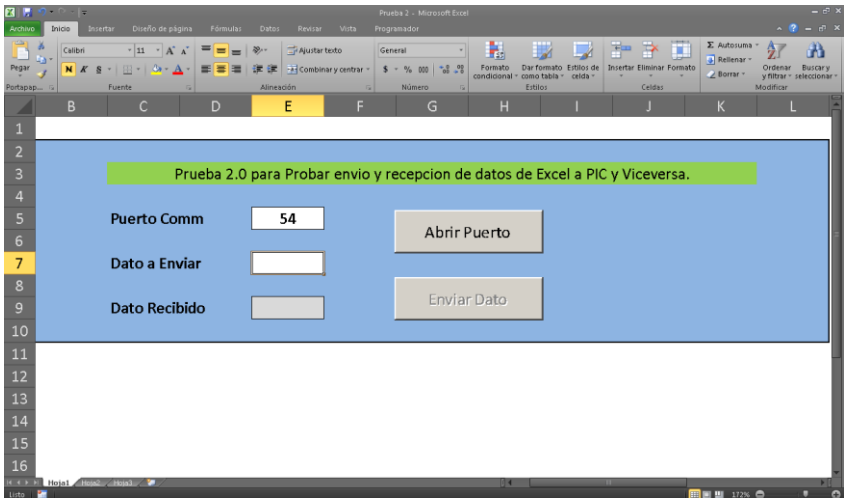


Figura. 4.24 Colocar el número de puerto COM y presionar Abrir Puerto [creación propia]**.**

Al presionar el botón inmediatamente recibimos un cero en Dato Recibido como se muestra en la figura 4.25, ya que el PIC está ciclado en enviar un “0” hasta que detecte alguna otra opción, por lo cual, si borramos este cero, volverá aparecer.

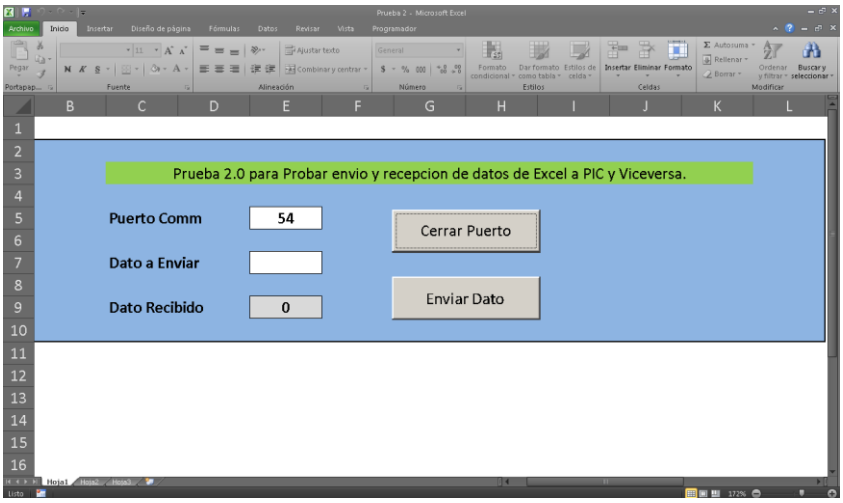


Figura. 4.25 Se observará un cero en Dato Recibido enviado por el PIC [creación propia]**.**

Ahora bien, si escribimos un “1” en Dato a Enviar, presionamos Enviar Dato y borramos el cero de Dato Recibido, notaremos que ya no llega más el cero, ya que la opción 1 bloquea el envío de dato del PIC hacia la PC.

Ahora si escribimos un “2” en Dato a Enviar, presionamos Enviar Dato, se vuelve a activar la opción de envío de datos del PIC hacia la PC (ver figura 4.26).

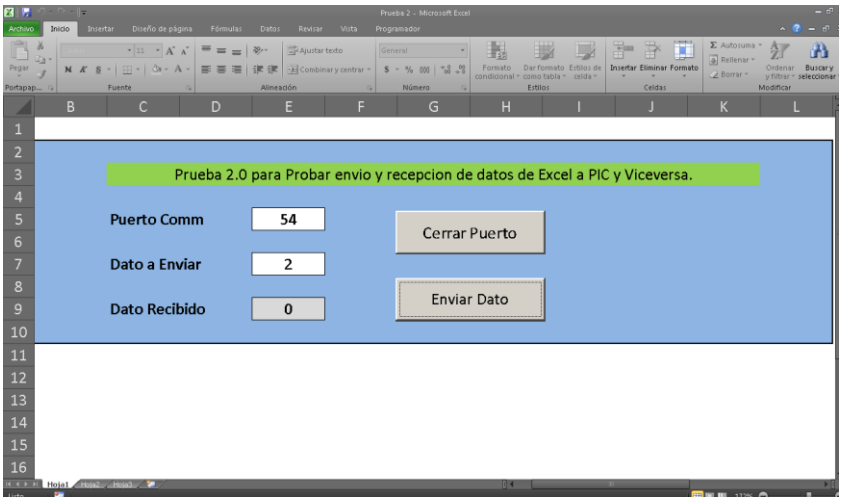


Figura. 4.26 Al enviar un 2 al PIC este Encender LED en B7 [creación propia]**.**

La siguiente opción es enviar un 3 como se muestra en figura 4.27, con lo cual se enciende el LED que tenemos conectado en B7 (ver figura 4.28).

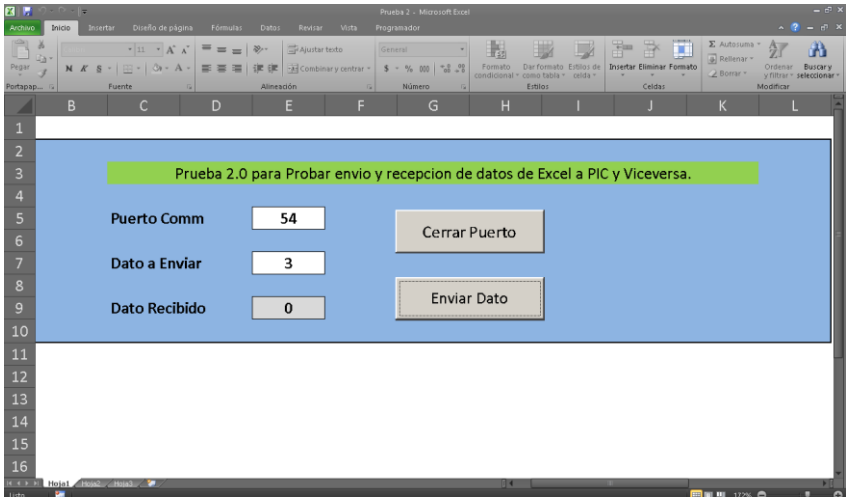


Figura. 4.27 Al enviar un 3 en PIC apagara el LED colocado en B7 [creación propia]**.**

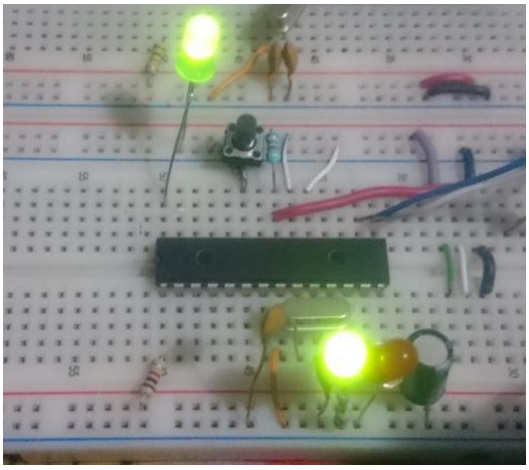


Figura. 4.28 LED encendido en B7 del PIC [creación propia]**.**

Y finalmente si se envía el 4 de la PC al PIC se apaga el LED de B7 (ver figura 4.29).

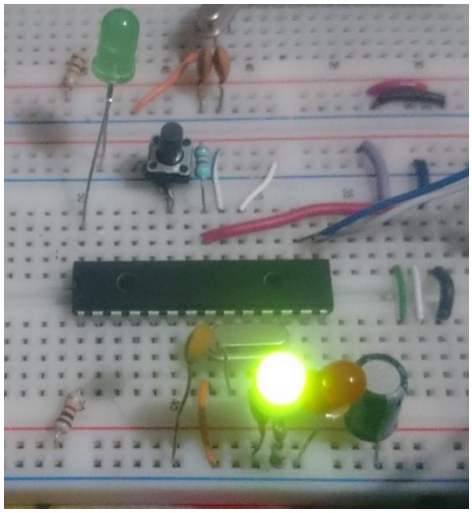


Figura. 4.29 LED apagado en B7 en el PIC [creación propia]**.**

Con esto se comprueba la comunicación entre la PC (Excel) y el PIC por medio de una emulación de una comunicación serial por medio del puerto USB.

4.1.2. Pruebas de Sensor de Temperatura y envió de dato a PC.

Ahora que ya se tenía la comunicación entre el PIC y la PC se procedió a enviar el dato de temperatura. Para esto se usaron los sensores de temperatura elegidos y simulados (los Dallas DS18B20), de los cuales se mencionó tienen la ventaja de ser digitales y one-wire por lo cual cuenta con solo tres terminales dos de alimentación y el pin denominado “data” observar figura 4.30. Eso ayuda en que por una entrada del PIC se pueden conectar varios sensores en paralelo.

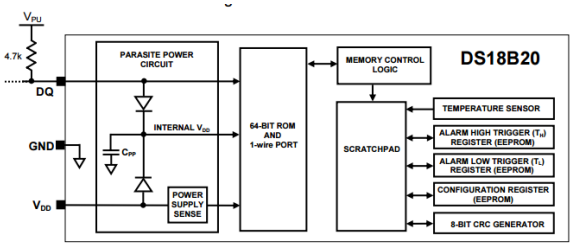


Figura. 4.30 Diagrama de conexión externa e interna de sensor DS18B20 [8]**.**

Ya con la conexión armada se procedió a realizar el programa para lo cual se buscó la librería ds1820.h para poder obtener los datos en el PIC la cual se encuentra en el Apéndice B. Además de la librería general del sensor se usó una sub-librería typesds1820.h la cual también se encuentra en el Apéndice B. A continuación, se creó el programa para la prueba descrita en la cual simplemente cada 2 segundos se toma el dato de un sensor de temperatura, se envía vía USB hacia la PC y se muestra en la celda de Dato Recibido, como se muestra a en el Apéndice B. En la figura 4.31 se muestra el resultado en la pantalla con solo abrir el puerto COM.

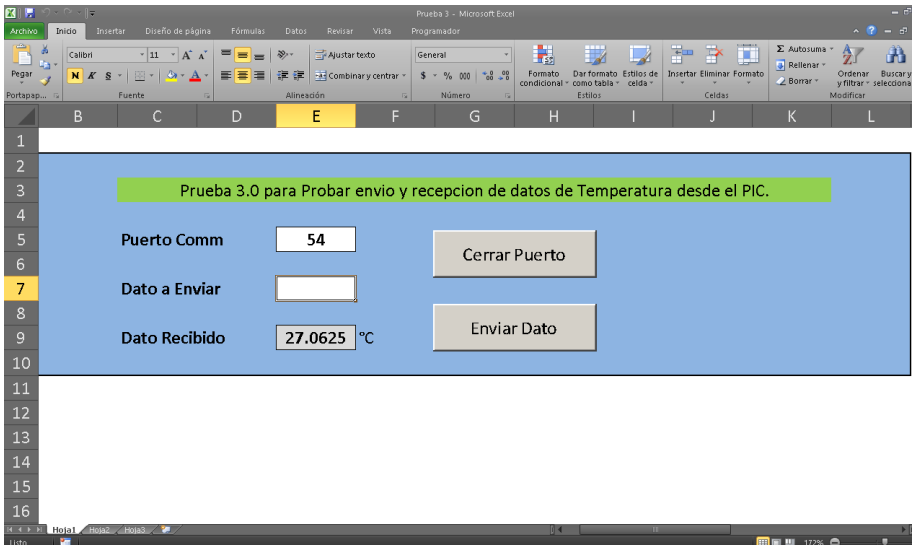


Figura. 4.31 Recepción de Temperatura al presionar abrir puerto [creación propia]**.**

Ya con la comunicación del sensor hacia la computadora ahora podemos proceder a tomar datos e ir comparándolos con el sensor de laboratorio figura 4.32.

Figura. 4.32 Termómetro IR modelo 568 marca Fluke [creación propia].

Para tener una representación gráfica del comportamiento del sensor DS18B20 se realizó un muestreo de datos cada cinco minutos por una hora con quince minutos para obtener un total de 15 muestras. Con lo anterior se obtuvo la gráfica mostrada en la figura 4.33.

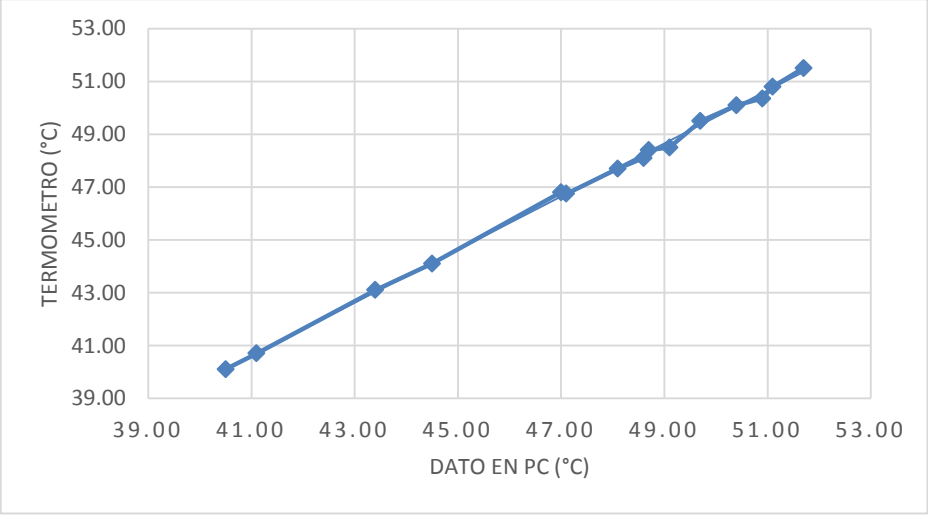


Figura. 4.33 Comparación grafica entre el termómetro patrón contra los datos obtenidos en la PC [creación propia]**.**

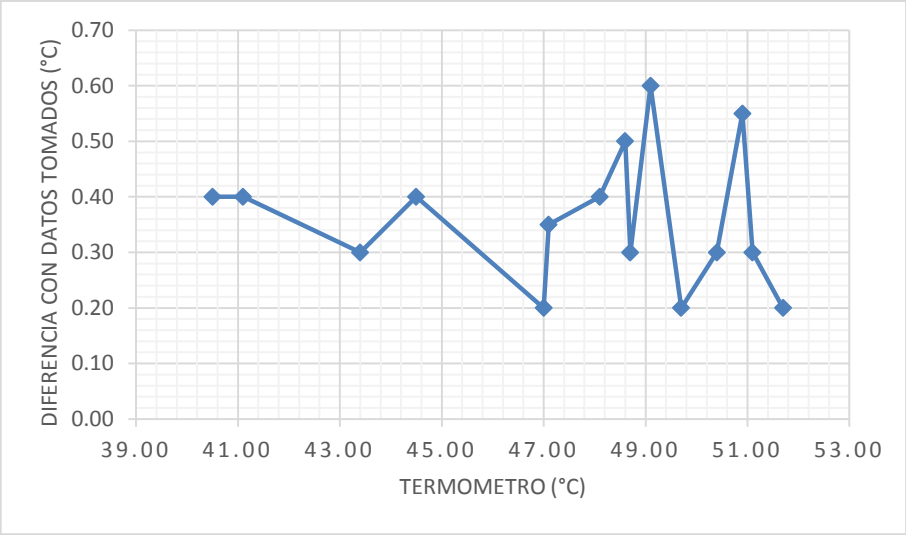


Figura. 4.34 Diferencia obtenida entre termómetro patrón y sistema diseñado. [creación propia]**.**

La variación máxima obtenida fue de 0.60 °C lo cual representa menos del 1.23% lo cual se considera que para el tipo de aplicación (sistemas fotovoltaicos) en el cual se usaran los sensores es más que buena ya que se realizó a un bajo costo y con buena precisión.

Ya con la comunicación del sensor con la PC y con la validación de los datos; el siguiente paso fue enviar los datos de los 4 sensores con los cuales se contó. Al programa anteriormente realizado solo se le agregaron algunas mejoras, primero que nada, se tenía que enviar cada temperatura junto con un identificador para poder colocar cada temperatura en el lugar definido para cada sensor en la hoja de Excel, por lo cual del PIC mandamos 6 caracteres, el primer carácter es el identificador para saber de cual sensor proviene el dato: 1, 2, 3 o 4 según sea el caso y después los 5 caracteres de la lectura de temperatura, por ejemplo: 34.27; los cuatro dígitos y el punto. Además, se agregó un activador de envió de datos que simplemente es una bandera que cambia cuando en la PC se presiona el botón de Pedir Temperaturas o Parar Envío, y con ello se activa la opción para

enviar las temperaturas. El programa con las modificaciones se muestras en el Apéndice B.

Para el caso del programa en VB también se hicieron algunas mejoras principalmente en la función NetComm1_OnComm la cual se encuentra en el Apéndice A.

Cómo se observa en el programa del Apéndice A, solo se lee el primer carácter para saber de qué sensor proviene la temperatura y en base a eso se decide a que opción de los if´s puede entrar para leer los 5 caracteres restantes y se coloca en la celda de Excel designada. El resultado de esta operación se ve en la figura 4.35. Con lo que se comprobó el envío de los cuatro sensores de temperatura y la correcta recepción en la PC.

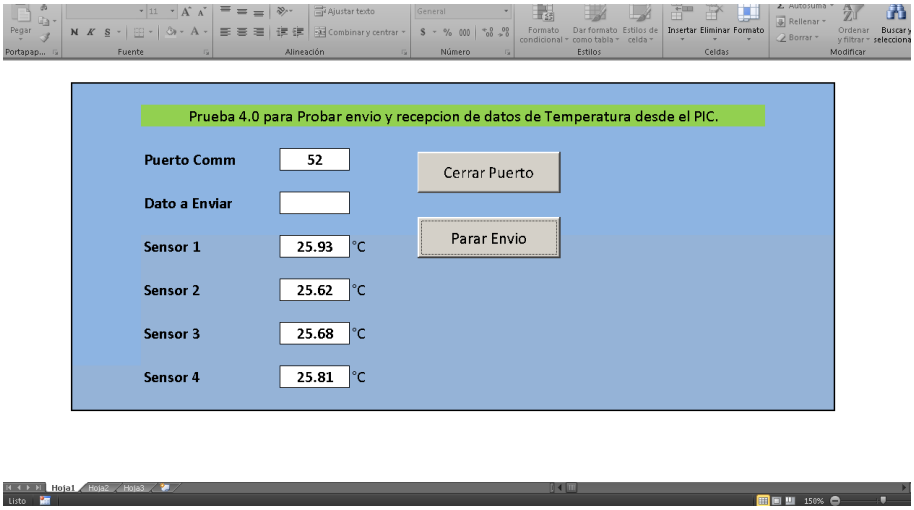
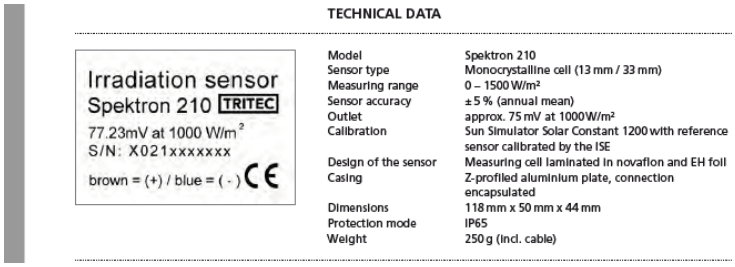


Figura. 4.35 Recepción varias temperaturas enviadas por el PIC a la PC [creación propia]**.**

4.1.3. Instrumentación de sensor de Irradiación, lectura y envió desde el PIC hacia la PC.

Como ya se mencionó se usó el sensor de irradiación el Spektron 210, el cual nos menciona en su hoja de datos que a 1000 W/m2 entrega 75mV (ver figura 4.36).



TECHNICAL DATA

Model	Spektron 210
Sensor type	Monocrystalline cell (13 mm / 33 mm)
Measuring range	0 – 1500 W/m²
Sensor accuracy	± 5 % (annual mean)
Outlet	approx. 75 mV at 1000 W/m²
Calibration	Sun Simulator Solar Constant 1200 with reference sensor calibrated by the ISE
Design of the sensor	Measuring cell laminated in novaflon and EH foil
Casing	Z-profiled aluminium plate, connection encapsulated
Dimensions	118 mm x 50 mm x 44 mm
Protection mode	IP65
Weight	250 g (incl. cable)

Figura. 4.36 Datos técnicos Sensor irradiación [imagen tomada de [14]]**.**

Para comprobar lo anterior se colocó el sensor junto con un medidor de irradiación y un multímetro para poder leer lo que manda el sensor de mV (ver figura 4.37). Los resultados de esta prueba se presentan en la tabla 4.1. En la tabla 4.1, se muestra la relación entre las lecturas obtenidas por el multímetro y piranómetro, a lo largo de una hora de medición; se realizaron otras medidas en los siguientes días, pero los resultados fueron similares por lo cual se toma la anterior como la mejor muestra y se determinó que 100mV sería el valor máximo de detección que nos entregaría el sensor de irradiación. Por lo cual 100mV del sensor serian 5V de entrada al ADC del PIC.

Tabla 4.1. Valores Irradiación VS mV [creación propia]**.**

Irradiación (W/m2)	mV	Hora
0	0.0	0
890	54.9	10:50
937	58.3	10:55
970	60.4	11:00
1026	64.0	11:05
1048	65.5	11:10
1049	65.6	11:15
1062	66.4	11:20
1070	67.5	11:25
1095	68.6	11:30
1100	68.7	11:35
1128	69.8	11:40
1145	70.6	11:45
1153	72.3	11:50
1168	73.7	11:55
1178	75.5	12:00
1199	76.1	12:05
1205	77.0	12:10
1224	77.7	12:20

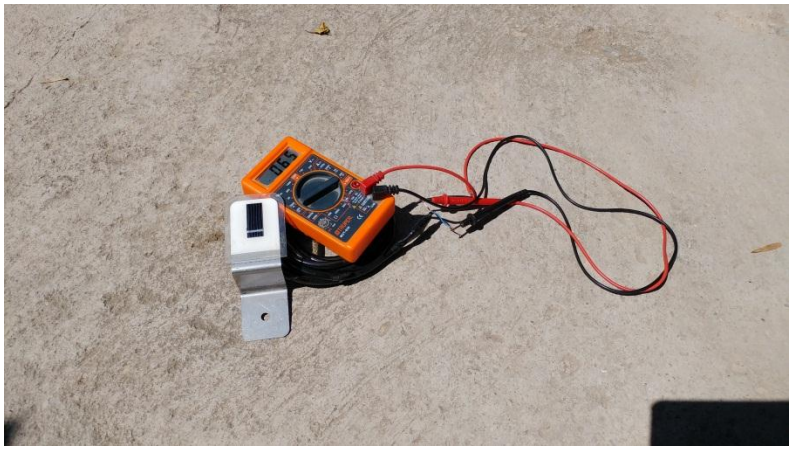


Figura. 4.37 Medición de mV de Sensor de irradiación a diferentes horas de día cuando el sol es más intenso [creación propia]**.**

Para lograr lo anterior se tuvo que diseñar un amplificador con el cual se pudieran pasar los 100mV a 5V, por lo cual se decidió usar un amplificador operación en configuración no inversor [20], para ello la ganancia requerida en esta configuración está dada en función del voltaje máximo que se desea aplicar al convertidor ADC. Para este caso, se fija su valor en Vmax = 5 Vdc y el valor máximo que nos puede entregar el sensor de irradiación. De esta forma, la ganancia está dada por:

$$G = \frac{V_{max}}{V_{sensor}} = \frac{5V}{0.100V} = 50 \tag{17}$$

Con este valor de ganancia y fijando uno de los valores de las resistencias se diseña la configuración no inversora: La ganancia (G) en una configuración no inversora está dada por:

$$G = \left(1 + \frac{R_f}{R_i}\right) \tag{18}$$

Si hacemos las sustituciones y fijamos $R_i = 1000\Omega$ se obtiene de la ecuación 18:

$$50 = \left(1 + \frac{R_f}{1000}\right) \tag{19}$$

Y despejando $R_f = 49{,}999\Omega$, este valor no es comercial, pero el más cercano es $50k\Omega$. Ahora simulando el amplificador no inversor con una entrada de 100mV y usando un amplificador LM358N [21], se decidió usar este tipo de amplificador ya que tiene una configuración interna la cual nos evita usar una fuente de voltaje dual de ±12V y lo podemos usar simplemente con una fuente de alimentación de 5VDC. La simulación se muestra en la figura 4.38 y en físico se muestra en la figura 4.39.

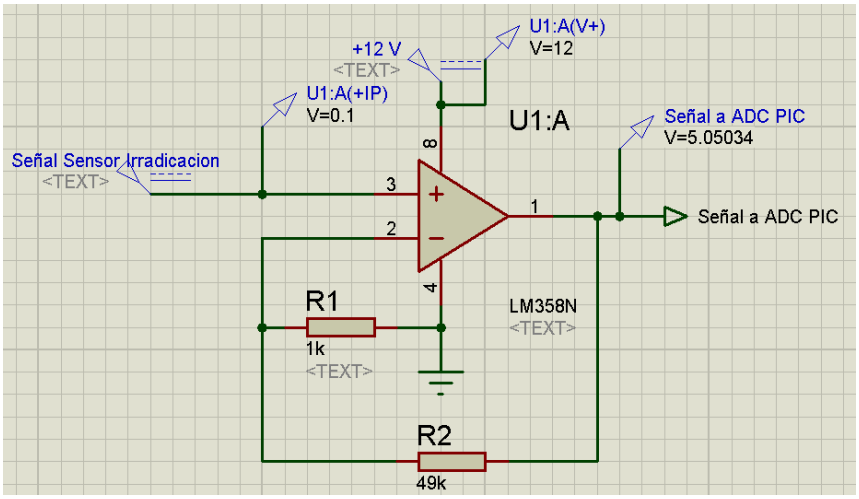


Figura. 4.38 Circuito simulado en ISI Profesional de Amplificador no inversor G=50 [creación propia]**.**

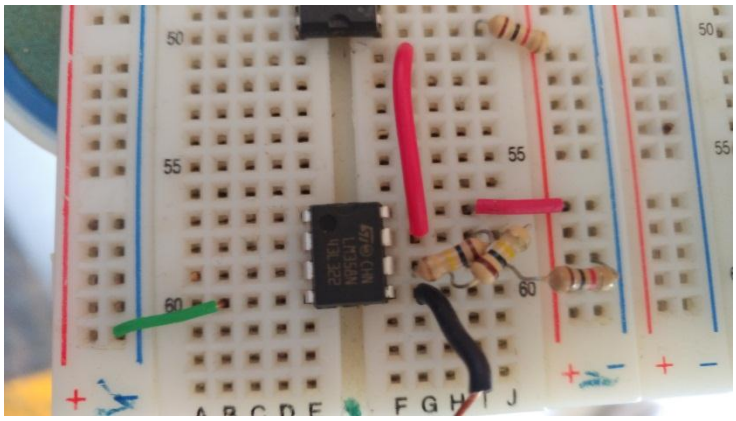


Figura. 4.39 Circuito Real de Amplificador no inversor G=50 [creación propia]**.**

Para corroborar los datos obtenidos se realizó la conexión al PIC y este a la PC para poder observar la lectura de irradiación en la pantalla. Por lo cual se realizó un pequeño programa para el PIC para poder leer la señal de amplificador por el ADC, este valor convertirlo a W/m2 y enviárselo a la PC. Para conseguir lo anterior se graficaron los datos (ver figura 4.40) y se obtuvo la ecuación 20.

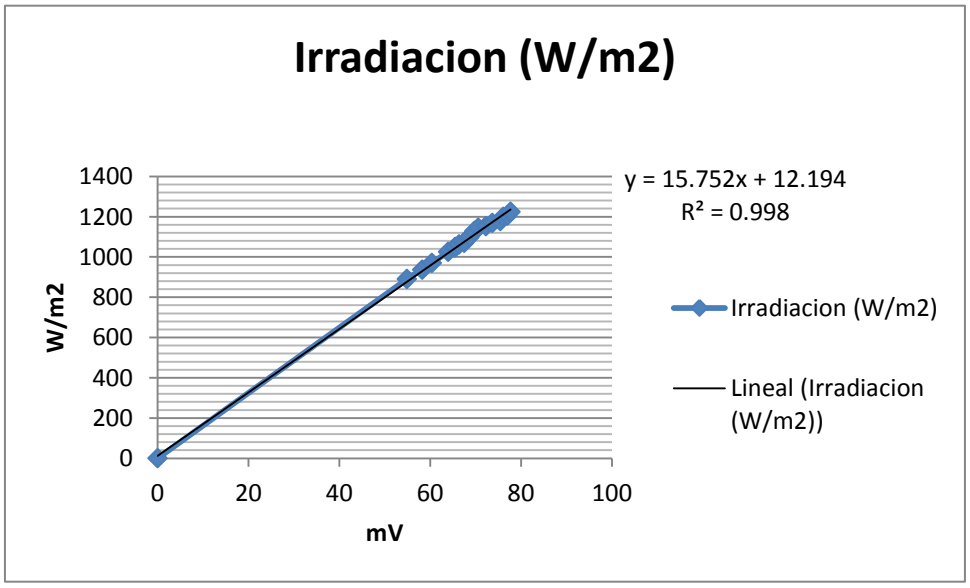


Figura. 4.40 Grafica de Irradiación real.

Con la ecuación que se obtiene:

$$Irradiacion = 15.752x + 12.194 \quad (20)$$

Con ella obtenemos el valor máximo de irradiación que fue: $Irradiacion = 15.752(100) + 12.194 = 1587.394\ \frac{W}{m^2}$. Por otro lado, revisando el histórico de radiación del 2007-2010 presentado por Centro de Geociencias de la UNAM campus Juriquilla [16], nos muestra que el máximo histórico registrado es de $1536\ \frac{W}{m^2}$ en la ciudad de Querétaro, por lo cual el valor obtenido es una buena aproximación.

Ahora bien, para hacer la conversión dentro del PIC18f2550 se usó el ADC a 10bits que representan $2^{10} = 1024\ niveles$ que para nuestro caso se podrá usar desde 0 – 1023 niveles, por lo cual para esta resolución se tendrá una lectura por cada nivel de:

$$Resolución = \frac{v_{reef}}{Niveles} = \frac{5\ V}{1023} = 0.004887586\ V \quad (21)$$

Para para hacer la conversión directa en el PIC ya sabemos que a 5V en radiación serán $1587.4\ \frac{W}{m^2}$ entonces se tiene que:

$$Irradiación = \frac{Irradiacion_{MAX}}{Niveles} = \frac{1587.4\ \frac{W}{m^2}}{1023} = 1.551710655\ \frac{W}{m^2} \quad (22)$$

Por lo que para cada nivel se tendrá $1.551710655\ \frac{W}{m^2}$ .Entonces para realizar la conversión en el PIC se tendrá la línea de código:

$$irradiacion\_float = (float)(irradiacion\_float * (1.551710655)); \quad (23)$$

Para comprobar la funcionalidad se realizó la misma prueba que para los sensores de temperatura, la cual consiste en leer el ADC hacer la conversión y enviarlo por USB hacia la PC y mostrarlo por medio de VB, lo anterior se presenta en la figura 4.41.

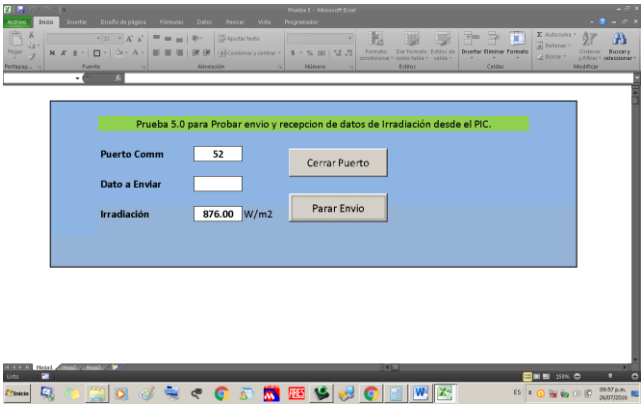


Figura. 4.41 Recepción de irradiación ya estandarizada por parte del PIC [creación propia]**.**

Ya con lo anterior se terminó de probar todos los sensores que se ocuparon y un poco de la interfaz que se usó. Solo faltaría la etapa de potencia de las celdas Peltier y los montajes.

4.1.4. Celdas Peltier prueba de funcionamiento.

El paso siguiente fue hacer las pruebas necesarias de control e implementación del circuito de potencia para controlar las celdas Peltier, para esto se desarrolló un puente H de potencia con transistores TIP35 y 36 [18], los cuales soportan hasta 25 A, el circuito que se probó y se muestra su diagrama eléctrico en la figura 4.42.

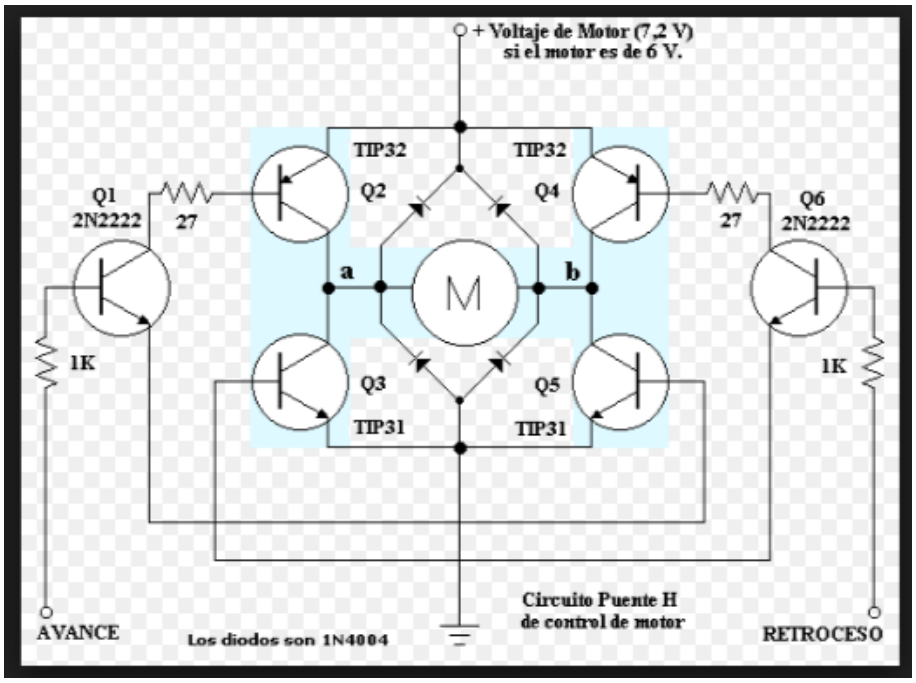


Figura. 4.42 Puente H, ejemplo para nuestro caso se usaron TIPs 35 y 36 de mayor potencia [imagen tomada de [18]]**.**

Una vez armado el circuito se realizaron pruebas con una sola celda por el tipo de fuente con la que se contaba en ese momento. Ya en otras pruebas se habilito una fuente de PC de 21 A que fue más que suficiente para hacer las pruebas necesarias. En la figura 4.43 se presenta el circuito del puente H implementado. Se realizaron pruebas alimentando directamente los transistores de control para determinar con que voltaje calentaba y con cual enfriaba.

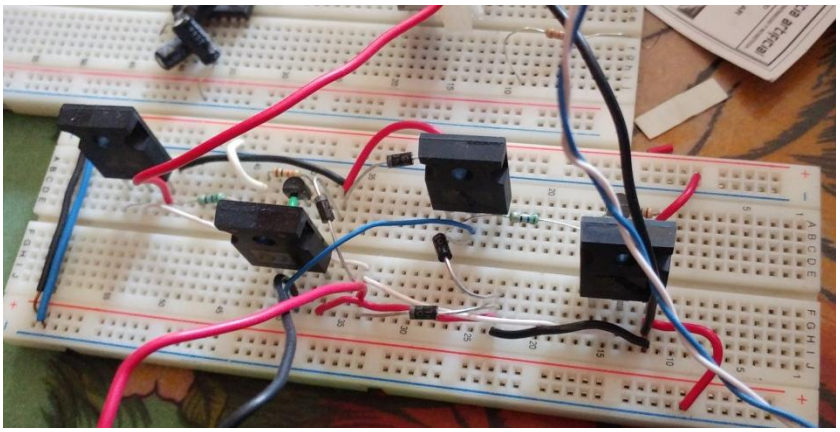


Figura. 4.43 Puente H real ya implementado con los TIPs 25 y 26 [creación propia]**.**

4.1.5. Fabricación de PCBs y montajes de sensores en el Panel FV.

Una vez terminadas las pruebas, ya solo faltaba la presentación, por lo cual una de las primeras acciones que se tomaron fue la fabricación de los PCBs para los dos PIC que se usaran uno en PC y el otro en el Panel. Primero se crearon los diseños en Ares (ver figura 4.44).

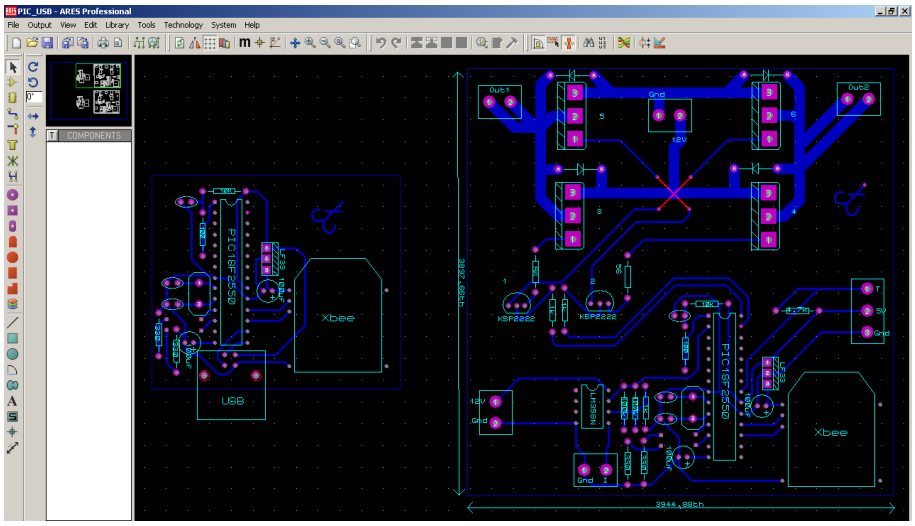


Figura. 4.44 PCBs de PIC PC y PIC panel desarrollados en Ares [creación propia]**.**

Una vez ya hechos los diseños del PCB, se procedió a imprimirlo, plancharlos y meterlos en acido para poder dejar solo las pistas (ver figura 4.45).

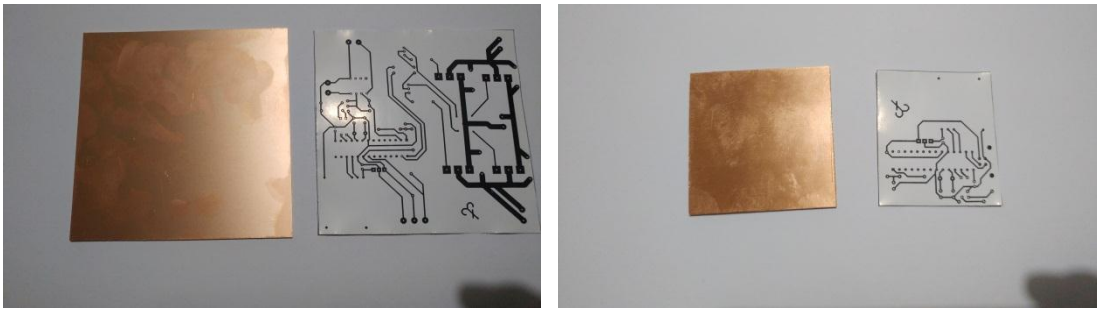


Figura. 4.45 Placas y Circuitos Impresos [creación propia]**.**

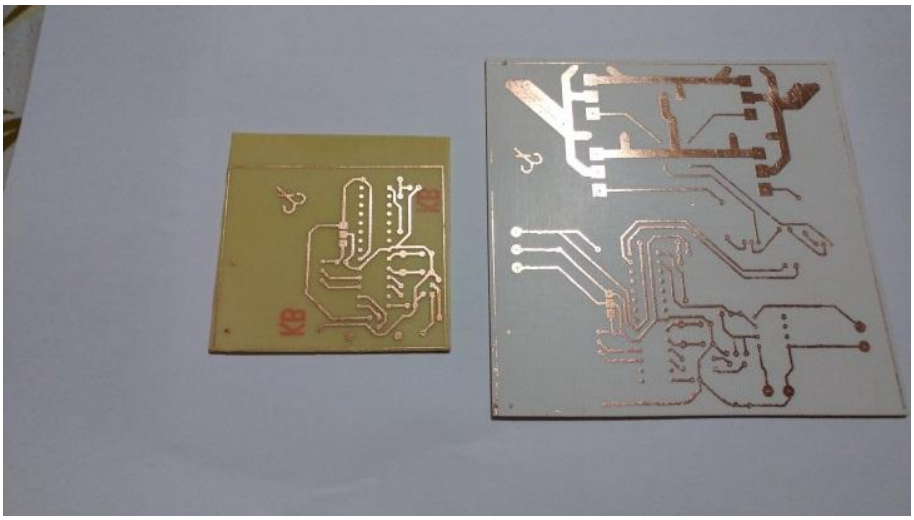


Figura. 4.46 Placas terminadas [creación propia]**.**

Una vez terminadas las placas (ver figura 4.46) solo se perforaron y se soldaron los componentes. A continuación, se procedió con la instalación de los sensores. Primero que nada, se fabricó una base para el Panel FV, en la cual se montará el panel junto con sus sensores y actuadores (ver figura 4.47). A continuación, se colocaron las celdas Peltier para lo cual se usó un poco de grasa de silicón térmica (ver figura 4.48, 4.49, 4.50 y 4.51) para tener una mejor transferencia de energía y un poco de silicón caliente para fijarlos.

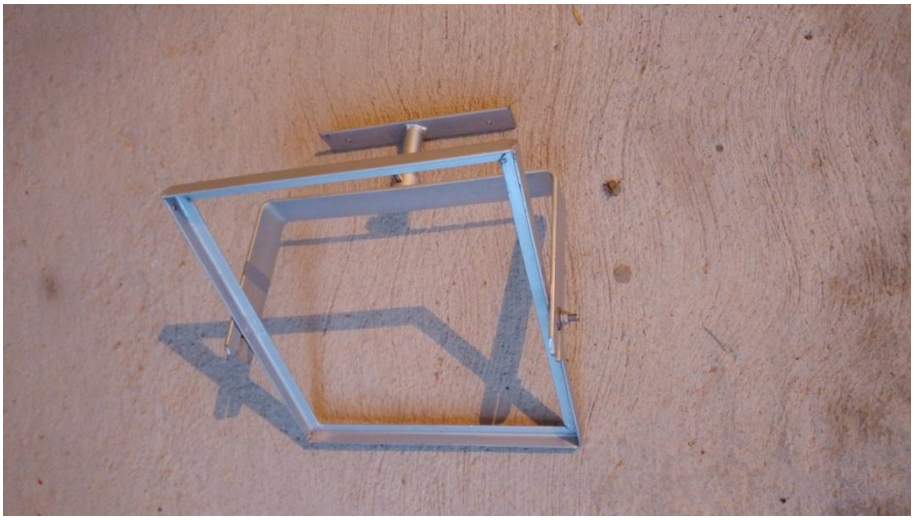


Figura. 4.47 Base para colocación de Panel [creación propia]**.**

Figura. 4.48 Celda Peltier con grasa silica [creación propia]**.**

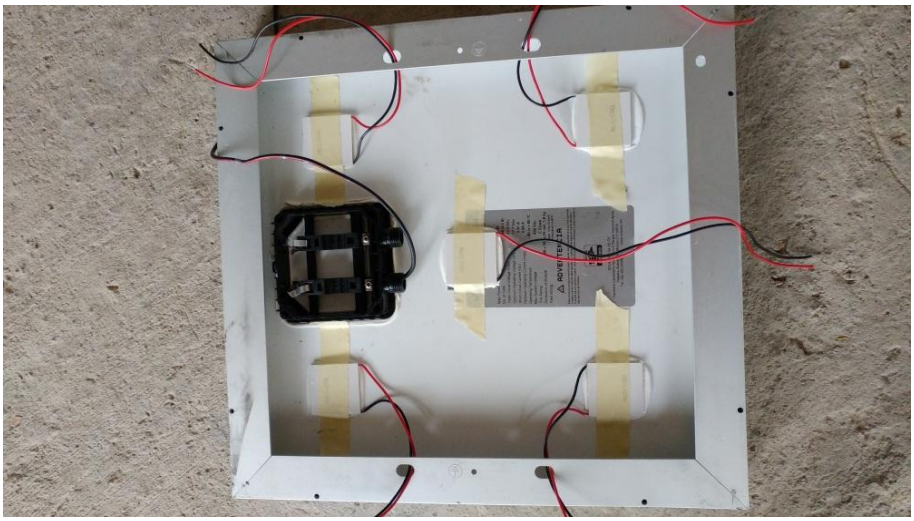


Figura. 4.49 Celdas Peltier colocadas en panel [creación propia]**.**

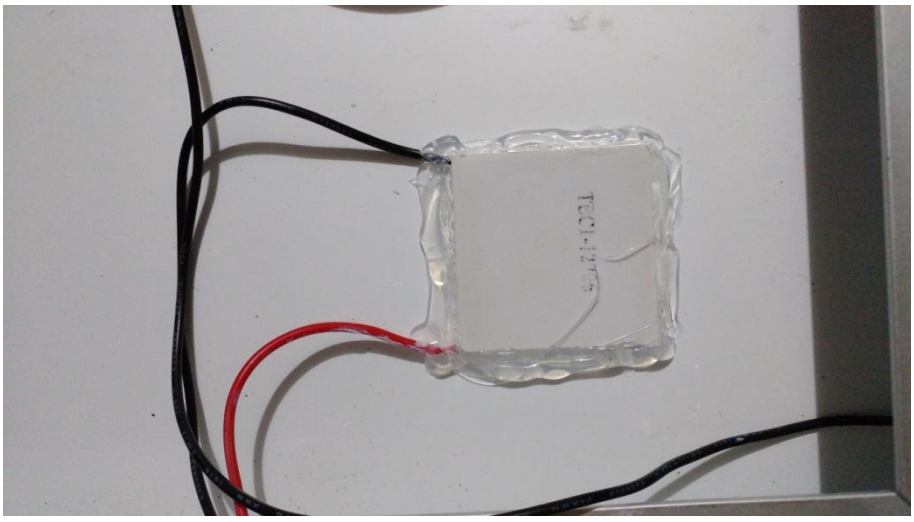


Figura. 4.50 Celda Peltier fijadas con Silicón Caliente [creación propia]**.**

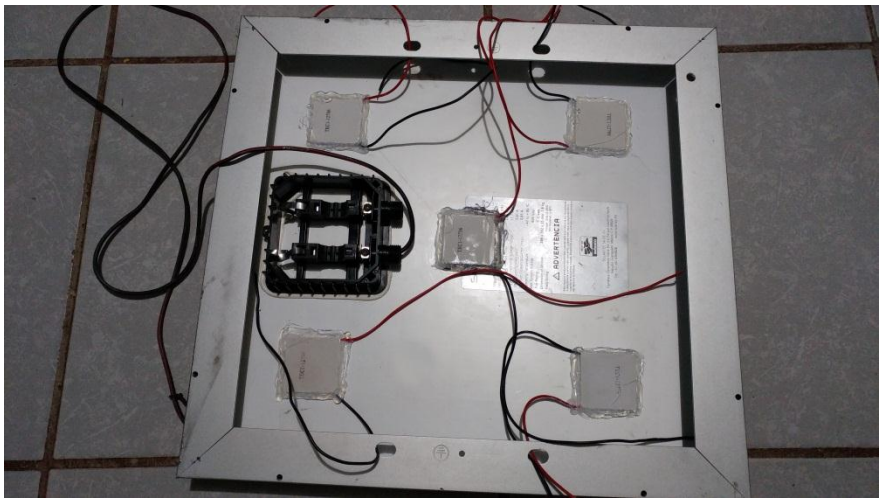


Figura. 4.51 Celda Peltier distribución [creación propia]**.**

Por último, se fijaron los sensores de temperatura, se aplicó el mismo principio que con las celdas, se les aplicó una delgada capa de grasa de silicón (ver figura 4.52) y después se fijaron con silicón caliente, para ello antes se realizó un forro con una hoja de aluminio, grasa de silicón y al final una capa de silicón caliente (ver figura 4.53). Se realizó el cableado necesario para unir los 4 sensores y se procedió a su instalación.

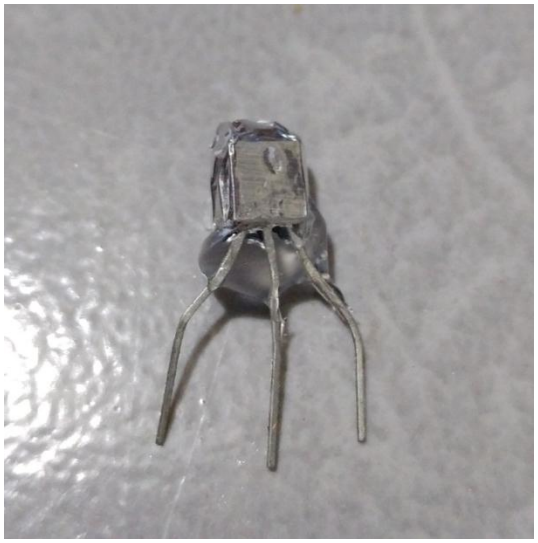


Figura. 4.52 Sensor de Temperatura DS1820 con recubrimiento de aluminio y silicón [creación propia]**.**

Figura. 4.53 Sensor de Temperatura DS1820 fijado con silicón Caliente [creación propia]**.**

Los sensores de temperatura se colocaron en las intersecciones en las cuales la temperatura es similar para los cuatro (ver figura 4.54).

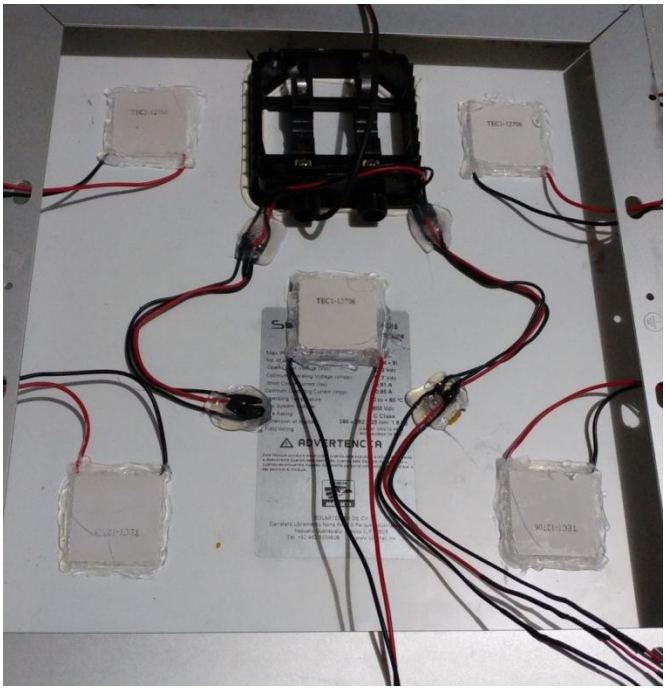


Figura. 4.54 Distribución de sensores de temperatura y celdas [creación propia]**.**

Finalmente, una vez colocados los sensores y actuadores, se realizó las conexiones de todos los componentes a las tarjetas y se realizaron las pruebas de funcionamiento (ver figura 4.55 y 4.56).

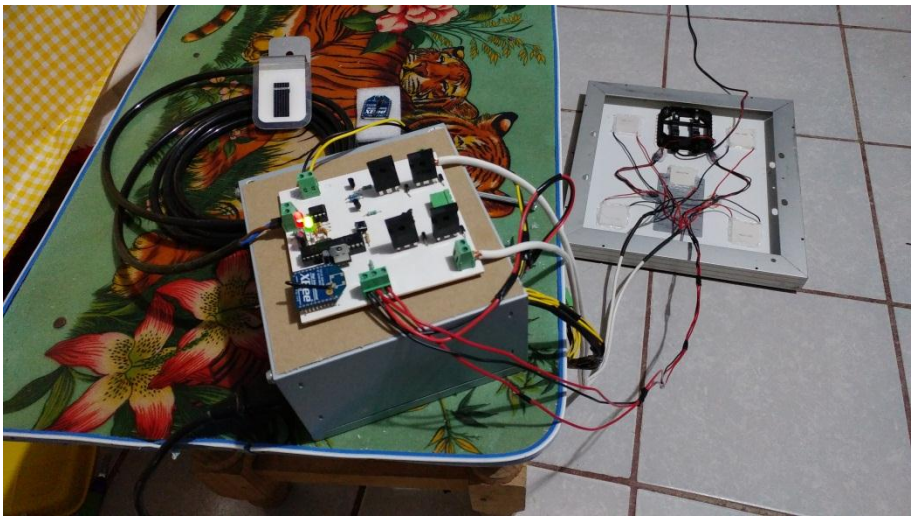


Figura. 4.55 Tarjeta de PIC panel y periféricos conectados [creación propia]**.**

Figura. 4.56 Caja de PIC PC con USB [creación propia]**.**

4.1.6. Desarrollo del SCADA en PC.

Para lo anterior, se modificó el programa que se venía manejando. Al programa que se tenía de las temperaturas se le agregó que leyera también al sensor de irradiación, y que en base al promedio de las temperaturas mandará la señal de calentamiento o enfriamiento de las celdas y guardara cada minuto los valores presentes en ese momento para dejarlos como históricos (ver figura 4.57).

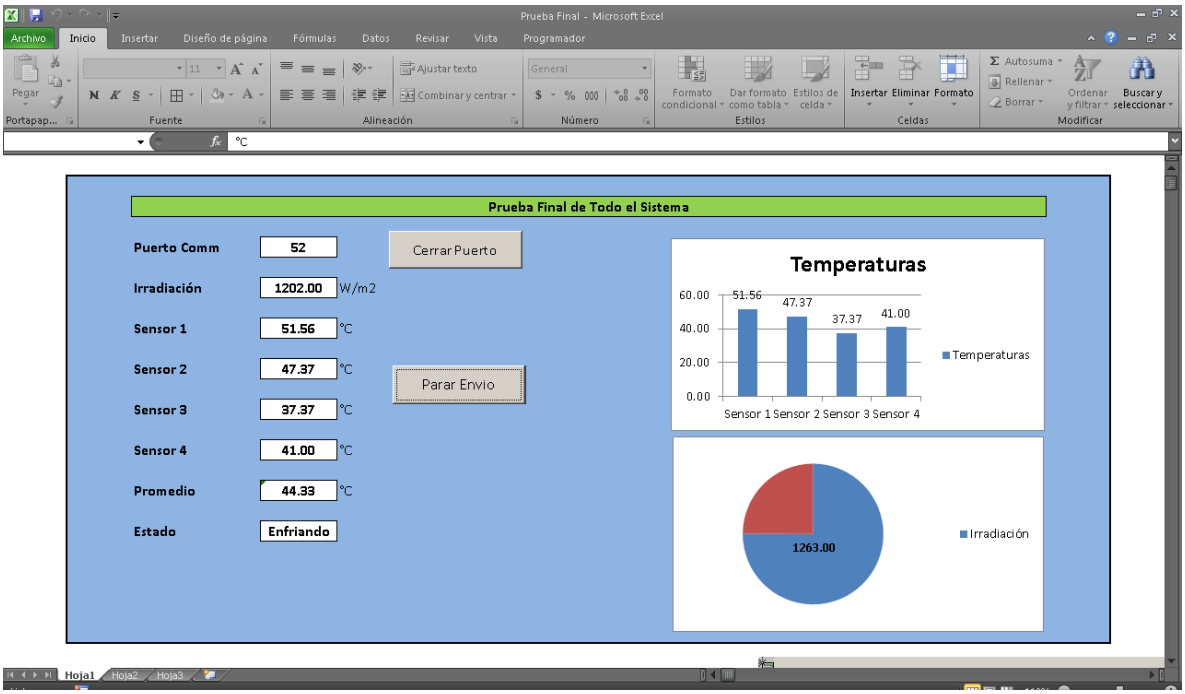
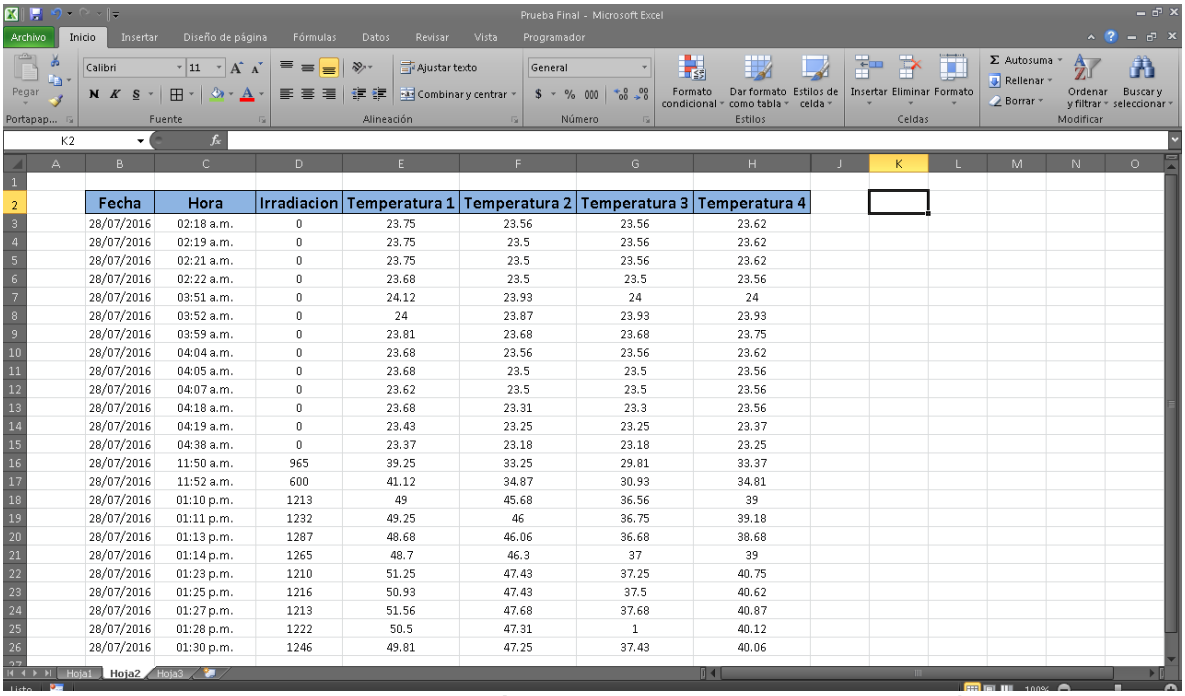


Figura. 4.57 Pantalla Principal ya con todos los datos de temperatura e irradiación [creación propia]**.**

En la figura 4.57 se muestra como quedo la interfaz dentro de la primera hoja del Excel, como se puede observar se muestran los 5 sensores, dos graficas donde se puede observar fácilmente el comportamiento de ellos y se tiene una etiqueta donde se muestra el estado que tienen las celdas Peltier el cual puede ser Off,

Calentando o Enfriando, y por último en la hoja 2 del Excel se muestra una tabla en la cual se guardan los datos presentes cada minuto para poder hacer el análisis que se requiera (ver figura 4.58).



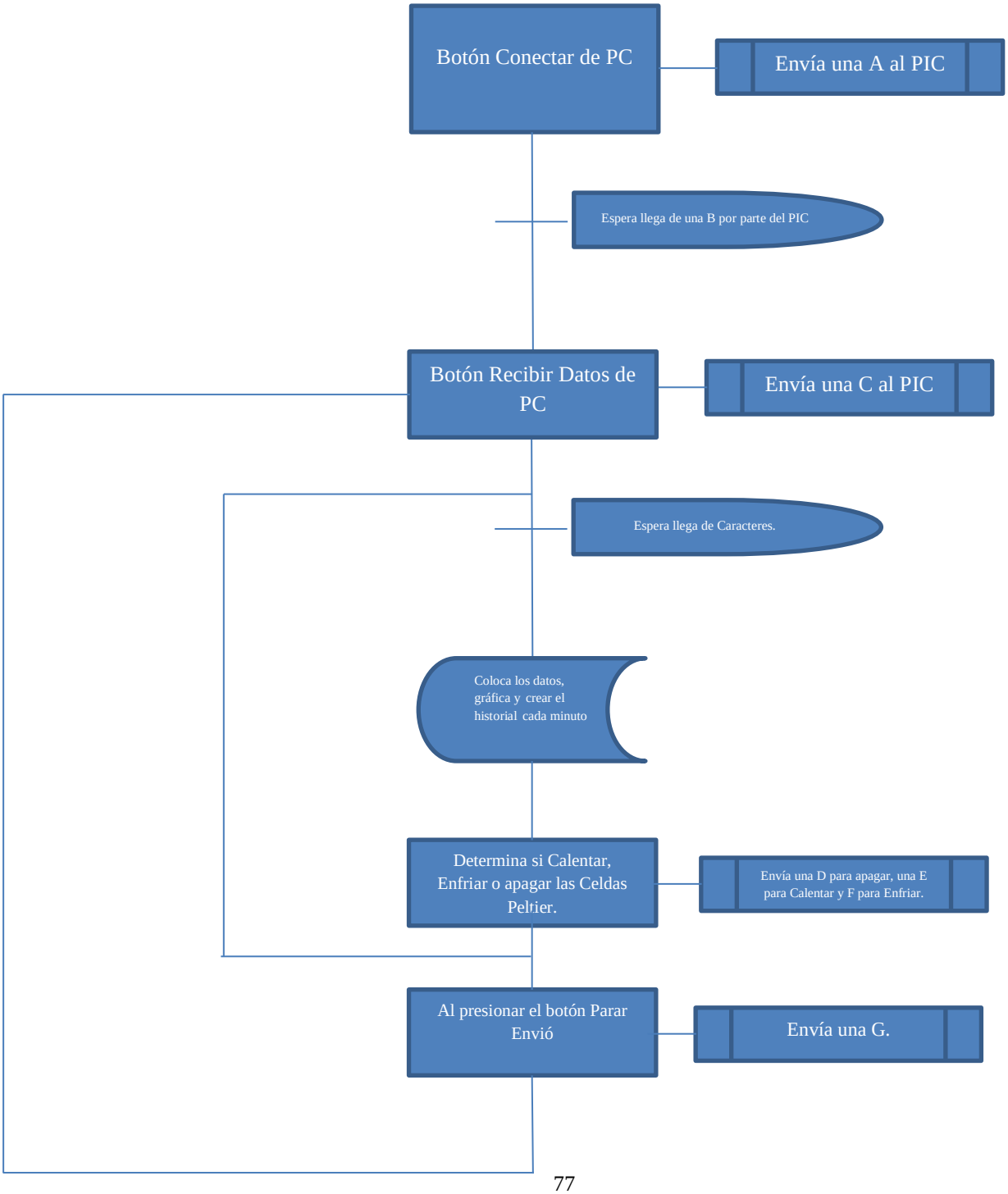
Fecha	Hora	Irradiacion	Temperatura 1	Temperatura 2	Temperatura 3	Temperatura 4
28/07/2016	02:18 a.m.	0	23.75	23.56	23.56	23.62
28/07/2016	02:19 a.m.	0	23.75	23.5	23.56	23.62
28/07/2016	02:21 a.m.	0	23.75	23.5	23.56	23.62
28/07/2016	02:22 a.m.	0	23.68	23.5	23.5	23.56
28/07/2016	03:51 a.m.	0	24.12	23.93	24	24
28/07/2016	03:52 a.m.	0	24	23.87	23.93	23.93
28/07/2016	03:59 a.m.	0	23.81	23.68	23.68	23.75
28/07/2016	04:04 a.m.	0	23.68	23.56	23.56	23.62
28/07/2016	04:05 a.m.	0	23.68	23.5	23.5	23.56
28/07/2016	04:07 a.m.	0	23.62	23.5	23.5	23.56
28/07/2016	04:18 a.m.	0	23.68	23.31	23.3	23.56
28/07/2016	04:19 a.m.	0	23.43	23.25	23.25	23.37
28/07/2016	04:38 a.m.	0	23.37	23.18	23.18	23.25
28/07/2016	11:50 a.m.	965	39.25	33.25	29.81	33.37
28/07/2016	11:52 a.m.	600	41.12	34.87	30.93	34.81
28/07/2016	01:10 p.m.	1213	49	45.68	36.56	39
28/07/2016	01:11 p.m.	1232	49.25	46	36.75	39.18
28/07/2016	01:13 p.m.	1287	48.68	46.06	36.68	38.68
28/07/2016	01:14 p.m.	1265	48.7	46.3	37	39
28/07/2016	01:23 p.m.	1210	51.25	47.43	37.25	40.75
28/07/2016	01:25 p.m.	1216	50.93	47.43	37.5	40.62
28/07/2016	01:27 p.m.	1213	51.56	47.68	37.68	40.87
28/07/2016	01:28 p.m.	1222	50.5	47.31	1	40.12
28/07/2016	01:30 p.m.	1246	49.81	47.25	37.43	40.06

Figura. 4.58 Pantalla del histórico obtenida en el Excel [creación propia]**.**

A los datos de imagen 4.58 se le quitaron varias horas para observar de manera clara el comportamiento durante diferentes horas del día.

4.1.7. Diagrama de flujo general y pruebas finales.

A continuación, se presenta el diagrama de flujo general del sistema.



Cabe mencionar que el PIC de la PC solo se usa para enlazar las comunicaciones entre la PC y el PIC del Panel FV, por medio de comunicación serial mediante los módulos XBee. Así mismo, el controlador que se utilizó para realizar las pruebas fue un On/Off. En las siguientes tablas 4.2 y 4.4, se muestran los datos obtenidos sin la aplicación del control, por lo cual la temperatura se mantiene por la mañana en valores entre 23 y 24 °C; y por la tarde los valores se elevan hasta 52 °C. En el caso de las tablas 7 y 9, se muestra como el control funciona ya que, aunque la Irradiación está arriba de 1000 W/m2 (por la tarde) la temperatura se mueve entre 25 y 26 °C o cuando la irradiación es 0 W/m2 (por la mañana) la temperatura igual se mantiene dentro del rango propuesto.

Tabla 4.2. Datos sin controlador por la mañana [creación propia]**.**

Fecha	Hora	Irradiación	Temperatura 1	Temperatura 2	Temperatura 3	Temperatura 4
26/07/2016	07:00 a.m.	0	23.75	23.56	23.56	23.62
26/07/2016	07:01 a.m.	0	23.75	23.5	23.56	23.62
26/07/2016	07:03 a.m.	0	23.75	23.5	23.56	23.62
26/07/2016	07:04 a.m.	0	23.68	23.5	23.5	23.56
26/07/2016	08:06 a.m.	100	24.12	23.93	24	24
26/07/2016	08:07 a.m.	121	24	23.87	23.93	23.93
26/07/2016	08:08 a.m.	115	23.81	23.68	23.68	23.75
26/07/2016	08:09 a.m.	120	23.68	23.56	23.56	23.62
26/07/2016	08:10 a.m.	145	23.68	23.5	23.5	23.56

Tabla 4.3. Datos con controlador por la mañana [creación propia]**.**

Fecha	Hora	Irradiación	Temperatura 1	Temperatura 2	Temperatura 3	Temperatura 4
26/07/2016	08:30 a.m.	124	25.71	25.69	25.68	25.71
26/07/2016	08:31 a.m.	126	25.7	25.73	25.72	25.69
26/07/2016	08:33 a.m.	130	26.55	26.5	26.2	26
26/07/2016	08:34 a.m.	128	265.9	25.85	25.83	25.9
26/07/2016	08:35 a.m.	120	24.94	24.9	24.93	24.99
26/07/2016	09:01 a.m.	100	24.93	25.12	25.1	25.14
26/07/2016	09:02 a.m.	99	25.23	25.34	25.48	25.37
26/07/2016	09:04 a.m.	40	25.52	25.49	25.51	25.56
26/07/2016	09:05 a.m.	100	25.67	25.6	25.65	25.59
26/07/2016	09:07 a.m.	121	25.69	25.58	25.59	25.61
26/07/2016	09:08 a.m.	128	25.65	25.62	25.69	25.62

Como se observa en la tabla 4.3 en este primer caso por la mañana se ve una mejoría en la temperatura manteniéndose en los valores recomendados por el fabricante (25°C), esto nos es de suma ayuda ya que uno de los principales objetivos de este trabajo es mantener la temperatura dentro de este rango ya que es en estos valores en los cuales el voltaje se compartan más cercano a su valor de voltaje en circuito abierto (valor proporcional a la irradiación y el máximo valor de voltaje del panel), por el contrario si la temperatura aumenta demasiado como es el caso que se muestra en la tabla 8 esto nos afecta directamente en el voltaje entregado por el panel, aunque la teoría nos dice que aumenta un poco la corriente esta es casi insignificante ya que es mayor la caída de voltaje, por lo cual la potencia entregada también disminuye.

Tabla 4.4. Datos sin controlador por la tarde [creación propia]**.**

Fecha	Hora	Irradiación	Temperatura 1	Temperatura 2	Temperatura 3	Temperatura 4
26/07/2016	01:13 p.m.	1287	48.68	46.06	36.68	38.68
26/07/2016	01:14 p.m.	1265	48.7	46.3	37	39
26/07/2016	01:23 p.m.	1210	51.25	47.43	37.25	40.75
26/07/2016	01:25 p.m.	1216	50.93	47.43	37.5	40.62
26/07/2016	01:27 p.m.	1213	51.56	47.68	37.68	40.87
26/07/2016	01:28 p.m.	1222	50.5	47.31	37.65	40.12
26/07/2016	01:30 p.m.	1246	49.81	47.25	37.43	40.06
26/07/2016	01:36 p.m.	1211	50.1	47.56	37.87	40.43
26/07/2016	01:37 p.m.	1211	50.81	47.93	38.06	40.68
26/07/2016	01:39 p.m.	1244	51.87	48.5	38.2	41

Tabla 4.5. Datos con controlador por la tarde [creación propia]**.**

Fecha	Hora	Irradiación	Temperatura 1	Temperatura 2	Temperatura 3	Temperatura 4
26/07/2016	01:50 p.m.	1247	25.34	25.13	25.24	25.56
26/07/2016	01:52 p.m.	1291	26.8	25.66	25.27	25.58
26/07/2016	01:53 p.m.	1245	28.1	28.89	28.67	28.59
26/07/2016	01:55 p.m.	1270	25.9	25.76	25.69	24.98
26/07/2016	01:56 p.m.	1268	25.76	25.67	25.72	25.45
26/07/2016	01:58 p.m.	450	25.34	25.14	25.78	25.52

26/07/2016	01:59 p.m.	1232	27.34	26.5	26.72	26.6
26/07/2016	02:01 p.m.	1154	25.46	25.78	25.69	25.63
26/07/2016	02:02 p.m.	1140	25.3	25.75	25.65	25.64
26/07/2016	02:04 p.m.	1112	25.89	25.76	25.7	25.63
26/07/2016	02:05 p.m.	1196	26.98	26.67	26.68	26.67

Ahora bien, si consideramos los datos obtenidos en la tabla 4.5, podemos decir que controlamos la temperatura del panel y podemos mantenerlo en el rango propuesto de 25 a 26 °C aun cuando la irradiación sea muy elevada (valor máximo obtenido 1291 W/m2) lo cual implicaría que sin control la temperatura se elevaría aproximadamente hasta 50°C como se muestra en la tabla 4.4. ¿Esto en que nos favorece?, pues en primer lugar en que estamos obtenido el valor máximo de voltaje recomendado (cercando al Voc) al poder controlar la temperatura, y en segunda (y quizás la más importante en muchos casos) que obtenemos al mismo tiempo la mayor cantidad de corriente ya que la corriente es proporcional a la irradiación.

Y con lo anterior podemos afirmar que, al obtener los valores máximos de corriente y voltaje, podemos obtener el máximo punto de potencia (MPP) lo cual nos ayuda en demasía a mejorar la eficiencia del panel y tener el mínimo de pérdidas, lo cual se comprobara con la implementación del controlador creado en la FI.

5. CONCLUSIONES

Finalmente podemos decir que los objetivos principales se cumplieron, se creó un sistema de comunicación inalámbrica por medio de los módulos de comunicación inalámbrica Xbee, se creó una comunicación directa con la PC, lo que facilitó la captación de los datos proporcionados por los sensores de temperatura e irradiación. Además de que con la construcción del sistema de adquisición de datos se puede analizar mejor los datos y presentar graficas del comportamiento del sistema fotovoltaico.

Otro punto que se alcanzo fue la creación del control de potencia para accionar y cambiar la polaridad a las celdas Peltier-Seebeck, donde el punto más crítico es la corriente que manejan, lo cual se ve reflejado en el ancho de las pistas para el PCB, los cables a usar y los dispositivos que funcionen con dicha corriente.

También se creó una aplicación sobre Visual Basic en conjunto con Microsoft Excel en la cual se puede gozar de las bondades que nos da Microsoft, ya que fácilmente se pueden manejar los datos, guardarlos y graficarlos sin necesidad de manejar algún otro lenguaje.

Y por último se pudo hacer un acercamiento al control de MPPT para el cual se desean usar los datos obtenidos ya que, por medio de un control On/Off la temperatura del panel FV se pudo controlar desde la interfaz creada en Excel. Y para la mayoría de los casos se mantuvo la temperatura dentro de los rangos especificados en los objetivos.

6. TRABAJOS A FUTURO

Aunque el sistema de monitoreo es funcional, la implementación del mismo va a depender del tamaño de los paneles solares o en general sistema FV, para tal caso se proponen los siguientes puntos a mejorar:

- Colocar un número de celdas Peltier adecuado al tamaño del panel fotovoltaico para el control de temperatura.
- Colocar disipadores a los TIPs que conforman al puente H para manejar corrientes de mayor valor o cambiar los TIPs por otro tipo dispositivos de mayor corriente.
- Comprar módulos Xbee de mayor alcance para tener mayor área de cobertura.
- Probar el sistema en paneles más grandes para ver si es factible ya que se requiere de una fuente de poder de gran tamaño para alimentar a las celdas Peltier.
- Usar la instrumentación y adquisición da datos aquí presentada para continuar con la investigación del proyecto FOFI y determinar si con el control creado por la FI y el usuario se puede controlar la temperatura del panel FV de una mejor manera.

LITERATURA CITADA

[1].http://www.capitaldelabiodiversidad.es/2012/02/energias-renovables-definicion-y.html (Energias Renovables). Fecha de acceso: 25 de Septiembre 2013.

[2]. Kensuke Nishioka, Ryuzou Hagihara (2003), “Analysis of grid-connected PV systems for residential houses in the Tokyo area focusing on module temperature”, 3rd World Conference on Photovoltaic Energy Conversion, pp. 2306-2309.

[3]. W. M. W. Mariam, and S. Husni (2006), “Influence of Malaysian Climate on the Efficiency of Polycrystalline Solar Cells”, First International Power and Energy Coference PECon 2006 November 28-29, pp. 54-57.

[4]. C. S. Chin, P. Neelakantan, H. P. Yoong, K. T. K. Teo (2011), “Fuzzy Logic Based MPPT for Photovoltaic Modules Influenced by Solar Irradiation and Cell Temperature”, UKSim 13th International Conference on Modelling and Simulation, pp. 376-381.

[5]. Baochao Wang, Issam Houssamo (2011), “A simple PV constrained production control strategy”, AVENUES-GSU, pp. 969-974.

[6]. Roberto F. Coelho, Filipe M. Conce (2010), “A MPPT Approach Based on Temperature Measurements Applied in PV Systems”, 9th IEEE/IAS International Conference on Industry Applications.

[7]. Barrera Navarro, Agustin (2012), “Controlador de temperatura PID, Neuronal y fussy para condensar agua en una celda Peltier”, Centro de Investigación en Ciencia Aplicada y Tecnología Avanzada, Instituto Nacional Politécnico, Unidad Querétaro.

[8].http://neutrino.phys.ksu.edu/~gahs/doublechooz/DC_SlowMRS/DS/DS18B20.pdf (hoja de datos del sensor DS18B20). Fecha de acceso: 8 de Octubre 2013.

[9]. http://www.xbee.cl/descargas.html (Aplicaciones de comunicación inalámbrica con módulo XBEE). Fecha de acceso: 8 de Octubre 2013.

[10]. https://noescomolocuentan.wordpress.com/2014/11/21/primeros-pasos-con-xbee-y-nueva-version-xctu/ (Configuración mínima de Xbee). Fecha de acceso: 8 de Octubre 2013.

[11]. http://dev4an.blogspot.mx/2013/09/introduccion-xbee.html (Protocolo de comunicación Xbee). Fecha de acceso: 8 de Octubre 2013.

[12]. Martínez Bohórquez, Miguel Ángel (2009), "Aportaciones a la instrumentación electrónica en la optimización de sistemas basados en energía solar". Departamento de Ingeniería Electrónica, de Sistemas Informáticos y Automática.

[13]. Pérez Carrasco, Daniel (2010), "Procedimiento de mantenimiento y calibración de estación radiométrica". Universidad de Sevilla, España.

[14].http://www.tritec-energy.com/common/pdf/tritec/Spektron-210_en.pdf (Datasheet de Sensor de Irradiación 210). Fecha de acceso: 25 de Septiembre 2013.

[15]. http://ww1.microchip.com/downloads/en/devicedoc/39632c.pdf (Datasheet de PIC18f2550). Fecha de acceso: 25 de Septiembre 2013.

[16]. http://www.geociencias.unam.mx/~bole/eboletin/reporte_meteo-2010-OK.pdf

(Datos de Irradiación en Querétaro en 2010). Fecha de acceso: 24 de Septiembre 2013.

[17]. http://www.hardandsoftware.net/NETCommOCX.htm (Drive para usar NETCommOCX en Excel). Fecha de acceso: 3 de Diciembre 2015.

[18]. http://blutintegrado.blogspot.mx/2012/05/puente-h.html (Esquemático de puente H). Fecha de acceso: 11 de Enero 2016.

[19]. http://hades.mech.northwestern.edu/index.php/File:Pictopic.jpg (Esquemático de Xbee). Fecha de acceso: 8 de Octubre 2013.

[20]. Karki, Jim (1998), “Understanding Operational Amplifier Specifications”. Digital Signal Processing Solutions, Texas Instruments. pp. 13-14.

[21].http://pdf.datasheetcatalog.com/datasheet2/d/0krxqjyrr14uhy7yk1if661zzqwy.pdf (Datasheet de Amplificador operacional LM358N). Fecha de acceso: 24 de Septiembre 2013.

[22]. http://www.ni.com/data-acquisition/what-is/esa/ (Descripción de un SATD). Fecha de acceso: 30 de Septiembre 2013.

APÉNDICE

Apéndice A

A1 Programa VB Habilitar y Deshabilitar puerto COM

```
'-------------------------------------------------------------------------------------
' Proposito     : Habilitar/Deshabilitar el puerto Com
'-------------------------------------------------------------------------------------
Private Sub CommandButton1_Click()

   If NETComm1.PortOpen = False Then
      NETComm1.CommPort = Hoja1.Range("E5").FormulaR1C1       'Tomar numero del puerto.
      NETComm1.DTREnable = True
      NETComm1.Handshaking = comNone
      NETComm1.InBufferSize = 244                    'Definir tamaño del buffer de entrada.
      NETComm1.InputLen = 6                         'Definir longitud de entrada.
      NETComm1.InputMode = comInputModeText                'Definir Modo de entrada.
      NETComm1.OutBufferSize = 244                  'Definir tamaño del buffer de salida.
      NETComm1.NullDiscard = False
      NETComm1.RThreshold = 1
      NETComm1.RTSEnable = False
      NETComm1.SThreshold = False
      NETComm1.Settings = "9600, N, 8, 1"            'Definir caracteristicas de conexion.
      NETComm1.PortOpen = True                      'Abrir el puerto.
      CommandButton1.Caption = "Cerrar Puerto"          'Cambiar texto del boton 1.
      CommandButton2.EnabLED = True                 'Habilitar boton 2.
   Else
      NETComm1.PortOpen = False                     'Cerrar puerto.
      CommandButton1.Caption = "Abrir Puerto"           'Cambiar texto del boton 1.
      CommandButton2.EnabLED = False                'Deshabilitar boton 2.
   End If
End Sub
```

A2 Programa VB envió de datos por USB

```
'-------------------------------------------------------------------------------------
' Proposito     :Enviar dato por Usb
'-------------------------------------------------------------------------------------
Private Sub CommandButton2_Click()

   NETComm1.Output = Hoja1.Range("E7").FormulaR1C1 '  enviar dato a la salida del puerto COM.

End Sub
```

A3 Programa VB Recepción de datos

```
'-------------------------------------------------------------------------------------
' Proposito     :Recibir datos a excel
'-------------------------------------------------------------------------------------

Private Sub NETComm1_OnComm()

   Hoja1.Range("E9") = NETComm1.InputData
```

```
End Sub
```

A4 Programa VB Recepción Varias Temperaturas

```
Private Sub NETComm1_OnComm()

    NETComm1.InputLen = 1                       'Recibir el primer caracter.
    sensor = NETComm1.InputData

    If sensor = 1 Then                      'Sensor 1 de Temperatura
       NETComm1.InputLen = 5                    'Recibir los caracteres restantes.
       Hoja1.Range("E9") = NETComm1.InputData
    End If
    If sensor = 2 Then                      'Sensor 2 de Temperatura
       NETComm1.InputLen = 5
       Hoja1.Range("E11") = NETComm1.InputData
    End If
    If sensor = 3 Then                      'Sensor 3 de Temperatura
       NETComm1.InputLen = 5
       Hoja1.Range("E13") = NETComm1.InputData
    End If
    If sensor = 4 Then                      'Sensor 4 de Temperatura
       NETComm1.InputLen = 5
       Hoja1.Range("E15") = NETComm1.InputData
    End If

End Sub
```

Apéndice B

B1 Programa comunicación PIC vs Hyperterminal

```
#include <18f2550.h>
#fuses HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL1,CPUDIV1,VREGEN          //PLL1 usando un
cristal de 4MHz.

#use delay(clock=48M)
#include <usb_cdc.h>                                    //libreria de emulacion de puerto serie.
#include <usb_desc_cdc.h>                  //libreria para configurar el VID (0x04D8) y el PID (0x000A)
#define LEDR PIN_c1                                     //LED rojo para la espera de la conexion USB.
#define LEDV PIN_c0                                   //LED  verde, se enciende cuando el USB esta conectado.
#define LED_OFF   output_low
#define LED_ON    output_high

int estado;
int dato;

void main (){
  usb_cdc_init();                                     //Inicializacion del modo CDC.
  usb_init();                                        //Inicializacion del puerto USB.
   LED_ON(LEDR);
   LED_OFF(LEDV);
   delay_ms(1000);
   usb_enumerated();                          //Sentencia para esperar hasta que la PC configure el PIC.
   LED_ON(LEDV);
   LED_OFF(LEDR);
   LED_OFF(PIN_b7);
   estado=0;
   dato=0;
  do{
    usb_task();                                       //comprueba la conexion..
    if(usb_enumerated()){                                 //corroborar que el usb esta conectado
      if (estado==0){
      usb_cdc_putc("0");                                 //si continua en el estado 0 manda un cero a la PC.
      delay_ms(2000);
      }
      if (usb_cdc_kbhit()==1){                            //detecta algo en el bus de entrada.
        dato=(int)(usb_cdc_getc()-48);                  //recibe el dato, lo hace entero y lo guarda en dato.

        if(dato==1){
          estado=1;                        //si dato recibido es 1 el estado pasa a 1, ya no envia datos al PC.
        }
        if(dato==2){
          estado=0;                              //si dato recibido es 0 el estado pasa a 0, envia dato "0" a PC.
        }
        if(dato==3){
          LED_ON(PIN_b7);                                 //si dato es 3 enciende LED
        }
        if(dato==4){
          LED_OFF(PIN_b7);                                //si dato es 4 enciende LED
        }
      }
    }
  }while(true);
}
```

B2 Drive CDC

```
; Windows USB CDC ACM Setup File
; Copyright (c) 2000 Microsoft Corporation
; Copyright (C) 2004 Microchip Technology Inc.

[Version]
Signature="$Windows NT$"
Class=Ports
ClassGuid={4D36E978-E325-11CE-BFC1-08002BE10318}
Provider=%MCHP%
LayoutFile=layout.inf
DriverVer=08/17/2001,5.1.2600.0

[Manufacturer]
%MFGNAME%=DeviceList

[DestinationDirs]
DefaultDestDir=12

[SourceDisksFiles]

[SourceDisksNames]

[DeviceList]
;Aquí deberemos de poner el mismo VID e ID //que hemos programado en el micro de lo contrario los drivers no ;serán
detectados. VID = 04D8, ID=000A
%DESCRIPTION%=DriverInstall, USB\VID_04D8&PID_000A

;------------------------------------------------------------------------------
; Windows 2000/XP Sections
;------------------------------------------------------------------------------

[DriverInstall.nt]
CopyFiles=DriverCopyFiles
AddReg=DriverInstall.nt.AddReg

[DriverCopyFiles]
usbser.sys,,,0x20

[DriverInstall.nt.AddReg]
HKR,,DevLoader,,*ntkern
HKR,,NTMPDriver,,usbser.sys
HKR,,EnumPropPages32,,"MsPorts.dll,SerialPortPropPageProvider"

[DriverInstall.nt.Services]
AddService=usbser, 0x00000002, DriverService

[DriverService]
DisplayName=%SERVICE%
ServiceType=1
StartType=3
ErrorControl=1
ServiceBinary=%12%\usbser.sys

;------------------------------------------------------------------------------
; String Definitions
;------------------------------------------------------------------------------

[Strings]
;Aquí pondremos la información de nuestro dispositivo
MCHP="Jose Atanacio"
MFGNAME="Juan Angeles"
DESCRIPTION="Puerto Serial J.J."
SERVICE="USB RS-232 Emulation Driver"
```

B3 Librería DS1820

```
  /*** FILEHEADER ********************************************************************
 *
 *    FILENAME:    ds1820.h
 *    DATE:        25.02.2005
 *    AUTHOR:      Christian Stadler
 *
 *    DESCRIPTION: Driver for DS1820 1-Wire Temperature sensor (Dallas)
 *
 ************************************************************************************/

/*** HISTORY OF CHANGE ************************************************************
 *
 *    $Log: /pic/_drv/ds1820.h $
 *
 * 9     13.11.10 20:02 Stadler
 * - changed interrupt lock and delay functions to #defines to remove
 * function call overhead
 *
 * 8     6.11.10 10:24 Stadler
 * - adjusted 1-wire timing
 *
 * 7     5.11.10 22:55 Stadler
 * - changed driver API
 *
 * 6     5.11.10 21:59 Stadler
 * - added DS18B20 support
 * - fixed problem with ROM search algorithm
 *
 * 5     2.11.10 20:25 Stadler
 * - changed function DS1820_FindFirstDev to DS1820_FindFirstDevice
 * - changed function DS1820_FindNextDev to DS1820_FindNextDevice
 * - updated code style
 *
 * 4     31.10.10 17:12 Stadler
 * - introduced DS1820_DelayUs
 *
 * 3     25.10.10 13:09 Stadler
 * - added interrupt lock
 *
 * 2     12.03.05 11:24 Stadler
 * - added EEPROM write function
 * - added "Search ROM Algorithm" to control multiple devices
 *
 * 1     26.02.05 18:18 Stadler
 * Driver for DS1820 1-Wire Temperature sensor (Dallas)
 *
 ************************************************************************************/

#ifndef _DS1820_H
#define _DS1820_H


/* check configuration of driver */
#ifndef DS1820_DATAPIN
#error DS1820 data pin not defined!
#endif


#define TEMP_RES            0x100 /* temperature resolution => 1/256°C = 0.0039°C */


/* ------------------------------------------------------------------------ */
/*                      DS1820 Timing Parameters                            */
/* ------------------------------------------------------------------------ */
```

```
#define DS1820_RST_PULSE       480   /* master reset pulse time in [us] */
#define DS1820_MSTR_BITSTART   2     /* delay time for bit start by master */
#define DS1820_PRESENCE_WAIT   40    /* delay after master reset pulse in [us] */
#define DS1820_PRESENCE_FIN    480   /* dealy after reading of presence pulse [us] */
#define DS1820_BITREAD_DLY     5     /* bit read delay */
#define DS1820_BITWRITE_DLY    100   /* bit write delay */


/* ---------------------------------------------------------------------------- */
/*                        DS1820 Registers                                      */
/* ---------------------------------------------------------------------------- */

#define DS1820_REG_TEMPLSB    0
#define DS1820_REG_TEMPMSB    1
#define DS1820_REG_CNTREMAIN  6
#define DS1820_REG_CNTPERSEC  7
#define DS1820_SCRPADMEM_LEN  9     /* length of scratchpad memory */

#define DS1820_ADDR_LEN       8


/* ---------------------------------------------------------------------------- */
/*                        DS1820 Commands                                       */
/* ---------------------------------------------------------------------------- */

#define DS1820_CMD_SEARCHROM     0xF0
#define DS1820_CMD_READROM       0x33
#define DS1820_CMD_MATCHROM      0x55
#define DS1820_CMD_SKIPROM       0xCC
#define DS1820_CMD_ALARMSEARCH   0xEC
#define DS1820_CMD_CONVERTTEMP   0x44
#define DS1820_CMD_WRITESCRPAD   0x4E
#define DS1820_CMD_READSCRPAD    0xBE
#define DS1820_CMD_COPYSCRPAD    0x48
#define DS1820_CMD_RECALLEE      0xB8


#define DS1820_FAMILY_CODE_DS18B20      0x28
#define DS1820_FAMILY_CODE_DS18S20      0x10


/* ---------------------------------------------------------------------------- */
/*                        static variables                                      */
/* ---------------------------------------------------------------------------- */

static bool bDoneFlag;
static uint8 nLastDiscrepancy_u8;
static uint8 nRomAddr_au8[DS1820_ADDR_LEN];



/* ---------------------------------------------------------------------------- */
/*                       Low-Level Functions                                    */
/* ---------------------------------------------------------------------------- */


/*******************************************************************************
 * FUNCTION:   DS1820_DelayUs
 * PURPOSE:    Delay for the given number of micro seconds.
 *
 * INPUT:      dly_us      number of micro seconds to delay
 * OUTPUT:     -
 * RETURN:     -
 *******************************************************************************/
#define DS1820_DelayUs(dly_us)       delay_us(dly_us)


/*******************************************************************************
```

```
 * FUNCTION:   DS1820_DelayMs
 * PURPOSE:    Delay for the given number of milliseconds.
 *
 * INPUT:      dly_ms      number of milliseconds to delay
 * OUTPUT:     -
 * RETURN:     -
 ******************************************************************************/
#define DS1820_DelayMs(dly_ms)   delay_ms(dly_ms)


/******************************************************************************
 * FUNCTION:   DS1820_DisableInterrupts
 * PURPOSE:    Disable interrupts
 *
 * INPUT:      -
 * OUTPUT:     -
 * RETURN:     -
 ******************************************************************************/
#ifdef DS1820_INTERRUPT_LOCK
#define DS1820_DisableInterrupts()  disable_interrupts(GLOBAL)
#else
#define DS1820_DisableInterrupts()
#endif



/******************************************************************************
 * FUNCTION:   DS1820_EnableInterrupts
 * PURPOSE:    Enable interrupts
 *
 * INPUT:      -
 * OUTPUT:     -
 * RETURN:     -
 ******************************************************************************/
#ifdef DS1820_INTERRUPT_LOCK
#define DS1820_EnableInterrupts()   enable_interrupts(GLOBAL)
#else
#define DS1820_EnableInterrupts()
#endif


/******************************************************************************
 * FUNCTION:   DS1820_Reset
 * PURPOSE:    Initializes the DS1820 device.
 *
 * INPUT:      -
 * OUTPUT:     -
 * RETURN:     FALSE if at least one device is on the 1-wire bus, TRUE otherwise
 ******************************************************************************/
bool DS1820_Reset(void)
{
   bool bPresPulse;

   DS1820_DisableInterrupts();

   /* reset pulse */
   output_low(DS1820_DATAPIN);
   DS1820_DelayUs(DS1820_RST_PULSE);
   output_high(DS1820_DATAPIN);

   /* wait until pullup pull 1-wire bus to high */
   DS1820_DelayUs(DS1820_PRESENCE_WAIT);

   /* get presence pulse */
   bPresPulse = input(DS1820_DATAPIN);

   DS1820_DelayUs(424);

   DS1820_EnableInterrupts();
```

```
   return bPresPulse;
}


/*****************************************************************************
 * FUNCTION:   DS1820_ReadBit
 * PURPOSE:    Reads a single bit from the DS1820 device.
 *
 * INPUT:      -
 * OUTPUT:     -
 * RETURN:     bool        value of the bit which as been read form the DS1820
 *****************************************************************************/
bool DS1820_ReadBit(void)
{
   bool bBit;

   DS1820_DisableInterrupts();

   output_low(DS1820_DATAPIN);
   DS1820_DelayUs(DS1820_MSTR_BITSTART);
   input(DS1820_DATAPIN);
   DS1820_DelayUs(DS1820_BITREAD_DLY);

   bBit = input(DS1820_DATAPIN);

   DS1820_EnableInterrupts();

   return (bBit);
}


/*****************************************************************************
 * FUNCTION:   DS1820_WriteBit
 * PURPOSE:    Writes a single bit to the DS1820 device.
 *
 * INPUT:      bBit        value of bit to be written
 * OUTPUT:     -
 * RETURN:     -
 *****************************************************************************/
void DS1820_WriteBit(bool bBit)
{
   DS1820_DisableInterrupts();

   output_low(DS1820_DATAPIN);
   DS1820_DelayUs(DS1820_MSTR_BITSTART);

   if (bBit != FALSE)
   {
      output_high(DS1820_DATAPIN);
   }

   DS1820_DelayUs(DS1820_BITWRITE_DLY);
   output_high(DS1820_DATAPIN);

   DS1820_EnableInterrupts();
}


/*****************************************************************************
 * FUNCTION:   DS1820_ReadByte
 * PURPOSE:    Reads a single byte from the DS1820 device.
 *
 * INPUT:      -
 * OUTPUT:     -
 * RETURN:     uint8          byte which has been read from the DS1820
 *****************************************************************************/
uint8 DS1820_ReadByte(void)
{
```

```
  uint8 i;
  uint8 value = 0;

  for (i=0 ; i < 8; i++)
  {
    if ( DS1820_ReadBit() )
    {
      value |= (1 << i);
    }
    DS1820_DelayUs(120);
  }
  return(value);
}


/*****************************************************************************
 * FUNCTION:   DS1820_WriteByte
 * PURPOSE:    Writes a single byte to the DS1820 device.
 *
 * INPUT:      val_u8         byte to be written
 * OUTPUT:     -
 * RETURN:     -
 *****************************************************************************/
void DS1820_WriteByte(uint8 val_u8)
{
  uint8 i;
  uint8 temp;

  for (i=0; i < 8; i++)      /* writes byte, one bit at a time */
  {
    temp = val_u8 >> i;     /* shifts val right 'i' spaces */
    temp &= 0x01;           /* copy that bit to temp */
    DS1820_WriteBit(temp);  /* write bit in temp into */
  }

  DS1820_DelayUs(105);
}



/* ------------------------------------------------------------------------ */
/*                          API Interface                                   */
/* ------------------------------------------------------------------------ */


/*****************************************************************************
 * FUNCTION:   DS1820_AddrDevice
 * PURPOSE:    Addresses a single or all devices on the 1-wire bus.
 *
 * INPUT:      nAddrMethod       use DS1820_CMD_MATCHROM to select a single
 *                               device or DS1820_CMD_SKIPROM to select all
 * OUTPUT:     -
 * RETURN:     -
 *****************************************************************************/
void DS1820_AddrDevice(uint8 nAddrMethod)
{
  uint8 i;

  if (nAddrMethod == DS1820_CMD_MATCHROM)
  {
    DS1820_WriteByte(DS1820_CMD_MATCHROM);     /* address single devices on bus */
    for (i = 0; i < DS1820_ADDR_LEN; i ++)
    {
      DS1820_WriteByte(nRomAddr_au8[i]);
    }
  }
  else
  {
    DS1820_WriteByte(DS1820_CMD_SKIPROM);     /* address all devices on bus */
```

```
  }
}


/*****************************************************************************
 * FUNCTION:   DS1820_FindNextDevice
 * PURPOSE:    Finds next device connected to the 1-wire bus.
 *
 * INPUT:      -
 * OUTPUT:     nRomAddr_au8[]       ROM code of the next device
 * RETURN:     bool                 TRUE if there are more devices on the 1-wire
 *                                  bus, FALSE otherwise
 *****************************************************************************/
bool DS1820_FindNextDevice(void)
{
   uint8 state_u8;
   uint8 byteidx_u8;
   uint8 mask_u8 = 1;
   uint8 bitpos_u8 = 1;
   uint8 nDiscrepancyMarker_u8 = 0;
   bool bit_b;
   bool bStatus;
   bool next_b = FALSE;

   /* init ROM address */
   for (byteidx_u8=0; byteidx_u8 < 8; byteidx_u8 ++)
   {
      nRomAddr_au8[byteidx_u8] = 0x00;
   }

   bStatus = DS1820_Reset();        /* reset the 1-wire */

   if (bStatus || bDoneFlag)        /* no device found */
   {
      nLastDiscrepancy_u8 = 0;     /* reset the search */
      return FALSE;
   }

   /* send search rom command */
   DS1820_WriteByte(DS1820_CMD_SEARCHROM);

   byteidx_u8 = 0;
   do
   {
      state_u8 = 0;

      /* read bit */
      if ( DS1820_ReadBit() != 0 )
      {
         state_u8 = 2;
      }
      DS1820_DelayUs(120);

      /* read bit complement */
      if ( DS1820_ReadBit() != 0 )
      {
         state_u8 |= 1;
      }
      DS1820_DelayUs(120);

      /* description for values of state_u8: */
      /* 00    There are devices connected to the bus which have conflicting */
      /*       bits in the current ROM code bit position. */
      /* 01    All devices connected to the bus have a 0 in this bit position. */
      /* 10    All devices connected to the bus have a 1 in this bit position. */
      /* 11    There are no devices connected to the 1-wire bus. */

      /* if there are no devices on the bus */
      if (state_u8 == 3)
```

```
    {
        break;
    }
    else
    {
        /* devices have the same logical value at this position */
        if (state_u8 > 0)
        {
            /* get bit value */
            bit_b = (bool)(state_u8 >> 1);
        }
        /* devices have confilcting bits in the current ROM code */
        else
        {
            /* if there was a conflict on the last iteration */
            if (bitpos_u8 < nLastDiscrepancy_u8)
            {
                /* take same bit as in last iteration */
                bit_b = ( (nRomAddr_au8[byteidx_u8] & mask_u8) > 0 );
            }
            else
            {
                bit_b = (bitpos_u8 == nLastDiscrepancy_u8);
            }

            if (bit_b == 0)
            {
                nDiscrepancyMarker_u8 = bitpos_u8;
            }
        }

        /* store bit in ROM address */
        if (bit_b != 0)
        {
            nRomAddr_au8[byteidx_u8] |= mask_u8;
        }
        else
        {
            nRomAddr_au8[byteidx_u8] &= ~mask_u8;
        }

        DS1820_WriteBit(bit_b);

        /* increment bit position */
        bitpos_u8 ++;

        /* calculate next mask value */
        mask_u8 = mask_u8 << 1;

        /* check if this byte has finished */
        if (mask_u8 == 0)
        {
            byteidx_u8 ++;  /* advance to next byte of ROM mask */
            mask_u8 = 1;    /* update mask */
        }
    }
} while (byteidx_u8 < DS1820_ADDR_LEN);


/* if search was unsuccessful then */
if (bitpos_u8 < 65)
{
    /* reset the last discrepancy to 0 */
    nLastDiscrepancy_u8 = 0;
}
else
{
    /* search was successful */
    nLastDiscrepancy_u8 = nDiscrepancyMarker_u8;
```

```
      bDoneFlag = (nLastDiscrepancy_u8 == 0);

      /* indicates search is not complete yet, more parts remain */
      next_b = TRUE;
   }

   return next_b;
}


/*****************************************************************************
 * FUNCTION:   DS1820_FindFirstDevice
 * PURPOSE:    Starts the device search on the 1-wire bus.
 *
 * INPUT:      -
 * OUTPUT:     nRomAddr_au8[]       ROM code of the first device
 * RETURN:     bool                 TRUE if there are more devices on the 1-wire
 *                                  bus, FALSE otherwise
 *****************************************************************************/
bool DS1820_FindFirstDevice(void)
{
   nLastDiscrepancy_u8 = 0;
   bDoneFlag = FALSE;

   return ( DS1820_FindNextDevice() );
}


/*****************************************************************************
 * FUNCTION:   DS1820_WriteEEPROM
 * PURPOSE:    Writes to the DS1820 EEPROM memory (2 bytes available).
 *
 * INPUT:      nTHigh         high byte of EEPROM
 *             nTLow          low byte of EEPROM
 * OUTPUT:     -
 * RETURN:     -
 *****************************************************************************/
void DS1820_WriteEEPROM(uint8 nTHigh, uint8 nTLow)
{
   /* --- write to scratchpad ---------------------------------------------- */
   DS1820_Reset();
   DS1820_AddrDevice(DS1820_CMD_MATCHROM);
   DS1820_WriteByte(DS1820_CMD_WRITESCRPAD); /* start conversion */
   DS1820_WriteByte(nTHigh);
   DS1820_WriteByte(nTLow);

   DS1820_DelayUs(10);

   DS1820_Reset();
   DS1820_AddrDevice(DS1820_CMD_MATCHROM);
   DS1820_WriteByte(DS1820_CMD_COPYSCRPAD); /* start conversion */

   delay_ms(10);
}


/*****************************************************************************
 * FUNCTION:   DS1820_GetTempRaw
 * PURPOSE:    Get temperature raw value from single DS1820 device.
 *
 *             Scratchpad Memory Layout
 *             Byte  Register
 *             0     Temperature_LSB
 *             1     Temperature_MSB
 *             2     Temp Alarm High / User Byte 1
 *             3     Temp Alarm Low / User Byte 2
 *             4     Reserved
 *             5     Reserved
 *             6     Count_Remain
```

```
 *          7    Count_per_C
 *          8    CRC
 *
 *          Temperature calculation for DS18S20 (Family Code 0x10):
 *          ======================================================
 *                                    (Count_per_C - Count_Remain)
 *          Temperature = temp_raw - 0.25 + ----------------------------
 *                                          Count_per_C
 *
 *          Where temp_raw is the value from the temp_MSB and temp_LSB with
 *          the least significant bit removed (the 0.5C bit).
 *
 *
 *          Temperature calculation for DS18B20 (Family Code 0x28):
 *          ======================================================
 *                 bit7   bit6   bit5   bit4   bit3   bit2   bit1   bit0
 *          LSB      2^3    2^2    2^1    2^0    2^-1   2^-2   2^-3   2^-4
 *                 bit15  bit14  bit13  bit12  bit3   bit2   bit1   bit0
 *          MSB      S      S      S      S      S      2^6    2^5    2^4
 *
 *          The temperature data is stored as a 16-bit sign-extended two's
 *          complement number in the temperature register. The sign bits (S)
 *          indicate if the temperature is positive or negative: for
 *          positive numbers S = 0 and for negative numbers S = 1.
 *
 * RETURN:     sint16         raw temperature value with a resolution
 *                        of 1/256°C
 ******************************************************************************/
sint16 DS1820_GetTempRaw(void)
{
   uint8 i;
   uint16 temp_u16;
   uint16 highres_u16;
   uint8 scrpad[DS1820_SCRPADMEM_LEN];

   /* --- start temperature conversion ---------------------------------- */
   DS1820_Reset();
   DS1820_AddrDevice(DS1820_CMD_MATCHROM);     /* address the device */
   output_high(DS1820_DATAPIN);
   DS1820_WriteByte(DS1820_CMD_CONVERTTEMP);   /* start conversion */
   //DS1820_DelayMs(DS1820_TEMPCONVERT_DLY);   /* wait for temperature conversion */
   DS1820_DelayMs(750);


   /* --- read sratchpad ------------------------------------------------ */
   DS1820_Reset();
   DS1820_AddrDevice(DS1820_CMD_MATCHROM);   /* address the device */
   DS1820_WriteByte(DS1820_CMD_READSCRPAD);  /* read scratch pad */

   /* read scratch pad data */
   for (i=0; i < DS1820_SCRPADMEM_LEN; i++)
   {
      scrpad[i] = DS1820_ReadByte();
   }


   /* --- calculate temperature ----------------------------------------- */
   /* Formular for temperature calculation: */
   /* Temp = Temp_read - 0.25 + ((Count_per_C - Count_Remain)/Count_per_C) */

   /* get raw value of temperature (0.5°C resolution) */
   temp_u16 = 0;
   temp_u16 = (uint16)((uint16)scrpad[DS1820_REG_TEMPMSB] << 8);
   temp_u16 |= (uint16)(scrpad[DS1820_REG_TEMPLSB]);

   if (nRomAddr_au8[0] == DS1820_FAMILY_CODE_DS18S20)
   {
      /* get temperature value in 1°C resolution */
      temp_u16 >>= 1;
```

```
      /* temperature resolution is TEMP_RES (0x100), so 1°C equals 0x100 */
      /* => convert to temperature to 1/256°C resolution */
      temp_u16 = ((uint16)temp_u16 << 8);

      /* now substract 0.25°C */
      temp_u16 -= ((uint16)TEMP_RES >> 2);

      /* now calculate high resolution */
      highres_u16 = scrpad[DS1820_REG_CNTPERSEC] - scrpad[DS1820_REG_CNTREMAIN];
      highres_u16 = ((uint16)highres_u16 << 8);
      if (scrpad[DS1820_REG_CNTPERSEC])
      {
         highres_u16 = highres_u16 / (uint16)scrpad[DS1820_REG_CNTPERSEC];
      }

      /* now calculate result */
      highres_u16 = highres_u16 + temp_u16;
   }
   else
   {
      /* 12 bit temperature value has 0.0625°C resolution */
      /* shift left by 4 to get 1/256°C resolution */
      highres_u16 = temp_u16;
      highres_u16 <<= 4;
   }

   return (highres_u16);
}


/*****************************************************************************
 * FUNCTION:   DS1820_GetTempFloat
 * PURPOSE:    Converts internal temperature value to string (physical value).
 *
 * INPUT:      none
 * OUTPUT:     none
 * RETURN:     float          temperature value with as float value
 *****************************************************************************/
float DS1820_GetTempFloat(void)
{
   return ((float)DS1820_GetTempRaw() / (float)TEMP_RES);
}


/*****************************************************************************
 * FUNCTION:   DS1820_GetTempString
 * PURPOSE:    Converts internal temperature value to string (physical value).
 *
 * INPUT:      tRaw_s16       internal temperature value
 * OUTPUT:     strTemp_pc     user string buffer to write temperature value
 * RETURN:     sint16         temperature value with an internal resolution
 *                            of TEMP_RES
 *****************************************************************************/
void DS1820_GetTempString(sint16 tRaw_s16, char *strTemp_pc)
{
   sint16 tPhyLow_s16;
   sint8 tPhy_s8;

   /* convert from raw value (1/256°C resolution) to physical value */
   tPhy_s8 = (sint8)(tRaw_s16/TEMP_RES);

   /* convert digits from raw value (1/256°C resolution) to physical value */
   /*tPhyLow_u16 = tInt_s16 % TEMP_RES;*/
   tPhyLow_s16 = tRaw_s16 & 0xFF;      /* this operation is the same as */
                                       /* but saves flash memory tInt_s16 % TEMP_RES */
   tPhyLow_s16 = tPhyLow_s16 * 100;
   tPhyLow_s16 = (uint16)tPhyLow_s16 / TEMP_RES;
```

```
   /* write physical temperature value to string */
   sprintf(strTemp_pc, "%d.%02d", tPhy_s8, (sint8)tPhyLow_s16);
}

#endif /* _DS1820_H */
```

B4 Sub Librería DS1820

```
/*** FILEHEADER ********************************************************************
 *
 *           FILENAME:    types.h
 *           DATE:        21.11.2004
 *           AUTHOR:                          Christian Stadler
 *
 *           DESCRIPTION: Definition of common types.
 *
 ***********************************************************************************/

/*** HISTORY OF CHANGE *************************************************************
 *
 *           $Log: /pic/_inc/types.h $
 *
 * 3     1.11.10 11:42 Stadler
 * - type definitons dependend on compiler
 *
 * 2     26.02.05 18:44 Stadler
 * added bool type
 *
 ***********************************************************************************/

#ifndef _TYPES_H
#define _TYPES_H

/*---------------------------------------------------------------------------*/
/* Type definitions for Microchip C18 Compiler                               */
/*---------------------------------------------------------------------------*/
#if defined(__18CXX)

typedef unsigned char bool;
typedef signed char sint8;
typedef signed int sint16;
typedef signed long sint32;
typedef unsigned char  uint8;
typedef unsigned int uint16;
typedef unsigned long uint32;

#endif /* #if defined(__18CXX) */



/*---------------------------------------------------------------------------*/
/* Type definitions for CCS C Compiler                                       */
/*---------------------------------------------------------------------------*/
#if (defined(__PCB__) || defined(__PCH__) || defined(__PCM__))

typedef int1 bool;
typedef signed int8 sint8;
typedef signed int16 sint16;
typedef signed int32 sint32;
typedef unsigned int8  uint8;
typedef unsigned int16 uint16;
typedef unsigned int32 uint32;

#endif /* (defined(__PCB__) || defined(__PCH__) || defined(__PCM__)) */

#endif /*_TYPES_H */
```

B5 Programa Envío de Temperatura 1 sensor

```
#include <18f2550.h>
#fuses HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL1,CPUDIV1,VREGEN          //PLL1 usando un
cristal de 4MHz.
#use delay(clock=48M)
#include <usb_cdc.h>                                    //libreria de emulacion de puerto serie.
#include <usb_desc_cdc.h>                                  //libreria para configurar el VID (0x04D8) y el PID (0x000A)
#define LEDR PIN_c1                                     //LED rojo para la espera de la conexion USB.
#define LEDV PIN_c0                                     //LED  verde, se enciende cuando el USB esta conectado.
#define LED_OFF   output_low
#define LED_ON    output_high

/* ---  Configuracion de Sensor de Temperatura DS1820 --- */
#define DS1820_DATAPIN PIN_B0                               //Definicion de pin de entrada one-wire.
#include <typesds1820.h>                                  //Sub-libreria de simplificacion de declaracion para ds1820
#include <ds1820.h>                                     //Libreria ds1820 para sensores DS18B20.


void main() {
  usb_cdc_init();                                     //Inicializacion del modo CDC.
  usb_init();                                       //Inicializacion del puerto USB.
  LED_ON(LEDR);
  LED_OFF(LEDV);
  delay_ms(1000);
  usb_enumerated();                                 //Sentencia para esperar hasta que la PC configure el PIC.
  LED_ON(LEDV);
  LED_OFF(LEDR);
  LED_OFF(PIN_b7);
  char temp_char[10];                                   //String de salida float hacia PC.
  float temperature_float;

  while (TRUE) {
    usb_task();
    if (usb_enumerated()){
      if(DS1820_FindFirstDevice ()){                    //Funcion para encontrar el primer sensor DS18B20.
        LED_ON(PIN_B7);
        do{
          temperature_float = DS1820_GetTempFloat ();          //Obtener el dato de Temp. en float.
          sprintf(temp_char,"%3.4f",temperature_float);        //Guardar el dato float de temp en un string, 3 enteros y
4 decimales.
          usb_cdc_putc(temp_char[0]);                   //Se envian cada uno de los char hacia la PC.
          usb_cdc_putc(temp_char[1]);
          usb_cdc_putc(temp_char[2]);
          usb_cdc_putc(temp_char[3]);
          usb_cdc_putc(temp_char[4]);
          usb_cdc_putc(temp_char[5]);
          usb_cdc_putc(temp_char[6]);
          usb_cdc_putc(temp_char[7]);
        }WHILE (DS1820_FindNextDevice ());                  //While para regresar si hay otro sensor en el one-wire.
      }
      LED_OFF(PIN_B7);
      delay_ms(2000);
    }
  }
}
```

B6 Programa PIC Main para envió de varias Temperaturas

```
void main() {
  usb_cdc_init();                                     //Inicializacion del modo CDC.
  usb_init();                                       //Inicializacion del puerto USB.
  LED_ON(LEDR);
```

```
  LED_OFF(LEDV);
  delay_ms(1000);
  usb_enumerated();                                   //Sentencia para esperar hasta que la PC configure el PIC.
  LED_ON(LEDV);
  LED_OFF(LEDR);
  LED_OFF(PIN_b7);

  char temp_char[6];                                  //String de salida float hacia PC.
  char dato,stop;
  int count;
  float temperature_float;
  stop=1;
  while (TRUE) {
    usb_task();
    if (usb_enumerated()){
      if (usb_cdc_kbhit()==1){                        //detecta algo en el bus de entrada.
        dato=usb_cdc_getc();
        if(dato=='T'){                                //Enviar Temperaturas
          stop=0;
        }
        if (dato=='C'){                               //Para envio de datos
          stop=1;
        }
      }
      if (stop==0){
          count=0;
          if(DS1820_FindFirstDevice ()){              //Funcion para encontrar el primer sensor DS18B20.
            LED_ON(PIN_B7);
            do{
              count++;
              temperature_float = DS1820_GetTempFloat ();     //Obtener el dato de Temp. en float.
              sprintf(temp_char,"%2.2f",temperature_float);   //Guardar el dato float de temp en un string, 3
enteros y 4 decimales.
              usb_cdc_putc(count+48);                 //Se envia numero de Sensor.
              usb_cdc_putc(temp_char[0]);             //Se envian cada uno de los char hacia la PC.
              usb_cdc_putc(temp_char[1]);
              usb_cdc_putc(temp_char[2]);
              usb_cdc_putc(temp_char[3]);
              usb_cdc_putc(temp_char[4]);
              delay_ms(1000);
            }WHILE (DS1820_FindNextDevice ());        //While para regresar si hay otro sensor en el one-
wire.
          }
          LED_OFF(PIN_B7);
      }
    }
  }
}
```

B7 Programa PIC lectura de Irradiación

```
#include <18f2550.h>
#device ADC=10
#fuses HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL1,CPUDIV1,VREGEN          //PLL1 usando un
cristal de 4MHz.
#use delay(clock=48M)
#include <usb_cdc.h>                                  //libreria de emulacion de puerto serie.
#include <usb_desc_cdc.h>                             //libreria para configurar el VID (0x04D8) y el PID (0x000A)
#define LEDR PIN_c1                                   //LED rojo para la espera de la conexion USB.
#define LEDV PIN_c0                                   //LED  verde, se enciende cuando el USB esta conectado.
#define LED_OFF   output_low
#define LED_ON    output_high


void main() {
  usb_cdc_init();                                     //Inicializacion del modo CDC.
  usb_init();                                         //Inicializacion del puerto USB.
```

```
  setup_adc (ADC_CLOCK_INTERNAL) ;                              //configuracionde ADC
  setup_adc_ports (AN0) ;
  set_tris_a (0b00000001) ;
  LED_ON(LEDR);
  LED_OFF(LEDV);
  delay_ms(1000);
  usb_enumerated();                                          //Sentencia para esperar hasta que la PC configure el PIC.
  LED_ON(LEDV);
  LED_OFF(LEDR);
  LED_OFF(PIN_b7);

  char irra_char[6];                                        //String de salida float hacia PC.
  char dato,stop;
  float irradiacion_float;
  stop=1;

  while (TRUE) {
    usb_task();
    if (usb_enumerated()){
      if (usb_cdc_kbhit()==1){                                 //detecta algo en el bus de entrada.
        dato=usb_cdc_getc();
        if(dato=='T'){                                      //Enviar Temperaturas
          stop=0;
        }
        if (dato=='C'){                                     //Para envio de datos
          stop=1;
        }
      }
      if (stop==0){
          LED_ON(PIN_B7);
          set_adc_channel (0) ;
          delay_us (10) ;
          irradiacion_float = read_adc () ;                        //recojo la lectura de 0 a 1023 (10bits)
          irradiacion_float=(float)(irradiacion_float*(1.551710655));
          sprintf(irra_char,"%4.0f",irradiacion_float);              //Guardar el dato float de temp en un string, 3 enteros y 4
decimales.
          usb_cdc_putc(irra_char[0]);                       //Se envian cada uno de los char hacia la PC.
          usb_cdc_putc(irra_char[1]);
          usb_cdc_putc(irra_char[2]);
          usb_cdc_putc(irra_char[3]);
          usb_cdc_putc(irra_char[4]);
          delay_ms(1000);
          LED_OFF(PIN_B7);
      }
    }
  }
}
```