UNIVERSIDAD CATOLICA DE SANTA MARIA

FACULTAD DE CIENCIAS E INGENIERÍAS FÍSICAS Y FORMALES

PROGRAMA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

“DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA AUTOMATIZADO UTILIZANDO DISPOSITIVOS ANDROID CON INTERFAZ INALÁMBRICA PARA LA SUPERVISIÓN Y ADQUISICIÓN DE DATOS EN UN INVERNADERO”

Plan de Tesis para Optar el Título Profesional de

Ingeniero Electrónico

Presentado por el Bachiller:

Máximo Joel Cruz Chávez

Arequipa – Peru

2014

RESUMEN

El desarrollo de aplicaciones gratuitas para dispositivos Android nos permite tener una gran variedad de posibilidades en el desarrollo de proyectos. El presente trabajo tiene como objetivo utilizar software sin costo para la adquisición, supervisión y control de los procesos de un invernadero. Se implementó una aplicación para dispositivos que utilizan Android. Con esta aplicación el usuario tiene la supervisión de dicho invernadero en su bolsillo, la aplicación se conecta con el proceso de forma inalámbrica. Se implementó además la misma aplicación en Java pero para computadores. Se utilizan sensores de humedad, de temperatura y de luz.

SUMMARY

This research work uses free software to control and supervise the process of a green house. I´ve implemented an application for devices that use Android. With this app the user has absolute control over the green house in his/her pocket. The app is connected to the process using wireless communication. Besides I implemented the same application using Java for computers. Humidity, temperature and light sensors are used, and some other signals are simulated by potentiometers.

DEDICATORIA:

A DIOS por todas las bendiciones dadas, por guiarme en el transcurso de mi vida para alcanzar los logros que tiene para mí y mi familia.

A mi madre por todo su apoyo y confianza que me muestra día a día para alcanzar mis objetivos.

A mi esposa por toda la paciencia y amor que me muestra en todo momento.

Finalmente a mis profesores que en el transcurso de mi vida formaron mi ser. Muchas gracias a todos.

Indice

Índice de Figuras

Índice de Tablas

Introducción

Para obtener el título profesional de ingeniero electrónico presento un proyecto innovador, una solución creativa para un proceso común o que pueda ser aplicada en otros procesos similares, y que además sea realista, viable. Muchas veces somos ajenos a la tecnología por el alto costo de desarrollo y de implementación que implica, y ese detalle puede ser un obstáculo para una sociedad en pleno crecimiento como la nuestra.

El proceso elegido fue un invernadero, por ser un proceso que utiliza diferentes tipos de sensores y actuadores, y porque permite ser controlado desde un celular. Digámoslo de otra forma, un proceso que mueve millones de dólares al día no puede estar controlado absolutamente desde un celular de $150 (ciento cincuenta dólares). Además lo aplicado en un invernadero puede funcionar sin mayor modificación en la agricultura, que constituye una de las principales actividades del Perú.

El proyecto también pretende servir de ejemplo para crear soluciones tecnológicas a nuestro alcance, con herramientas existentes en el mercado y de bajo costo.

Como acabo de mencionar, el proceso se supervisa desde un Smartphone dotado de Android (sistema operativo para dispositivos móviles). A diferencia de los desarrollos para IOS, el sistema de Apple, Las herramientas de desarrollo de Android son completamente libres, cualquier persona puede descargar todo lo necesario para crear aplicaciones Android desde la web. Existen un gran número de formatos de programación, incluida la programación gráfica que utiliza bloques, pero se limitan a aplicaciones que no demandan mucho procesamiento.

La aplicación móvil se comunica con el proceso de forma inalámbrica, utilizando tecnología Bluetooth. Para ello se realizó una búsqueda en el mercado, para determinar que elemento puede cumplir dicho trabajo. Al final se adquirió una tarjeta que convierte señales TTL a señales inalámbricas y viceversa.

También se implementó la misma interface en Java (software libre), sin las características gráficas de la aplicación móvil, pero exactamente con las mismas funciones. De este modo el sistema puede funcionar en ambos casos, de acuerdo a las preferencias del usuario.

.

CAPITULO I
Planteamiento Metodológico

1.1. Problema

Diseño e implementación de un sistema automatizado utilizando dispositivos Android con interfaz inalámbrica para la supervisión, control y adquisición de datos de un invernadero.

1.2. Descripción del problema

Todos los sistemas automatizados necesitan una forma directa de interacción con el operador, esta comunicación se realiza a través de interfaces hombre-máquina o HMI (*ingles:* Human Machine Interface). Puede ser un simple botón para echar a andar un sistema o un panel complejo para sistemas más sofisticados.

El control de supervisión y adquisición de datos o SCADA (*ingles:* Supervisory Control And Data Acquisition) en los últimos años ha tomado gran importancia gracias al fácil acceso a tecnología y a la proliferación de los computadores. Un sistema SCADA permite tener un control minucioso de la información del

proceso, permite ingresar valores y enviar órdenes al proceso, verificar estados de sensores, cambiar valores de referencia, obtener gráficas, etc. Un sistema SCADA cuenta con un HMI que puede ser una pantalla táctil, un PC, etc.

Existen procesos que demandan sistemas SCADA robustos, capaces de soportar las condiciones ambientales industriales, procesos que no pueden registrar fallas porque pueden ocasionar miles o millones de dólares en pérdidas. Para ello se necesitan computadores de última generación, o algunos equipos especiales.

Pero qué hay de aquellos procesos con menor grado de complejidad como por ejemplo el control de un invernadero? La primera opción sería el uso de un PC dedicado para implementar el sistema SCADA. Afortunadamente hoy en día existen otros elementos en el mercado que pueden reemplazar el PC, con un menor costo, otorgando portabilidad y algunos otros beneficios.

1.3. Justificación del Proyecto

Debido al poco interés por mantener y desarrollar cultivos sostenibles en las ciudades por considéralos trabajosos y molestos el presente proyecto nos presenta una alternativa de implementar en casa un invernadero de fácil manipulación

En el presente trabajo de investigación pretendo utilizar un teléfono inteligente para la implementación del sistema SCADA de un invernadero. Se desarrollará una aplicación capaz de ser instalada en cualquier dispositivo que funcione con Android como sistema operativo.

Se podrán controlar y supervisar las diferentes variables involucradas en el proceso de un invernadero. Además funcionará de forma inalámbrica de modo que desde cualquier lugar del invernadero se podrá manejar el sistema, sin tener la necesidad de acercarse a un panel para realizar alguna tarea, se tendrá el panel en las manos.

Este trabajo surge debido a la necesidad de conseguir un sistema de bajo costo y seguro, solo el responsable del invernadero puede controlar el proceso porque lo tendrá en sus bolsillos.

1.4. Objetivos

Objetivo General

Diseñar e implementar un sistema automatizado utilizando software libre para dispositivos Android con conexión inalámbrica y computadores, para la supervisión, control y adquisición de datos de un invernadero.

Objetivos Específicos

- Diseñar una aplicación en Android para supervisar y controlar las variables involucradas en un invernadero.

- Conseguir un sistema SCADA portátil utilizando comunicación inalámbrica.

- Demostrar el uso de teléfonos inteligentes como alternativa segura y económica para la supervisión de datos en procesos automatizados.

- Diseñar una aplicación para computadores personales utilizando el lenguaje de programación Java para supervisar un invernadero.

1.5. Hipótesis

1.5.1. Hipótesis general

Utilizando un Smartphone de uso general será posible controlar una planta pequeña, en este caso se trata de un invernadero. Para ello se utilizarán todas las herramientas de programación disponibles en internet todas ellas de libre descarga. Además se utilizaran los recursos inalámbricos del

Smartphone para que la conexión con el controlador del proceso sea remota.

1.5.2. Hipótesis Específicas

- Android a diferencia de otros sistemas operativos para dispositivos móviles brinda muchas facilidades para sus desarrolladores. Se descargarán gratuitamente todas las herramientas necesarias para crear una aplicación.

- Java es una herramienta completamente libre con grandes prestaciones, no tiene nada que envidiar a los sistemas de pago. Además Java es soportada por todos los sistemas operativos. Por ese motivo utilizaré este lenguaje de programación para implementar una aplicación para computadores personales.

- Se pretende aprovechar la conexión Bluetooth disponible en la gran mayoría de Smartphones como medio de comunicación entre la interfaz del sistema y el proceso.

1.6. Alcance del proyecto

Se diseñara una aplicación en Android para la supervisión y control de las variables involucradas en un invernadero, se podrán ingresar valores como por ejemplo de setpoint, y se visualizarán valores de sensores de nivel, temperatura, ph, etc.

Desde esta interface se podrán iniciar procesos así como manejar actuadores de manera automática y manual. La comunicación entre el smartphone y el controlador del proceso se realizará de manera inalámbrica.

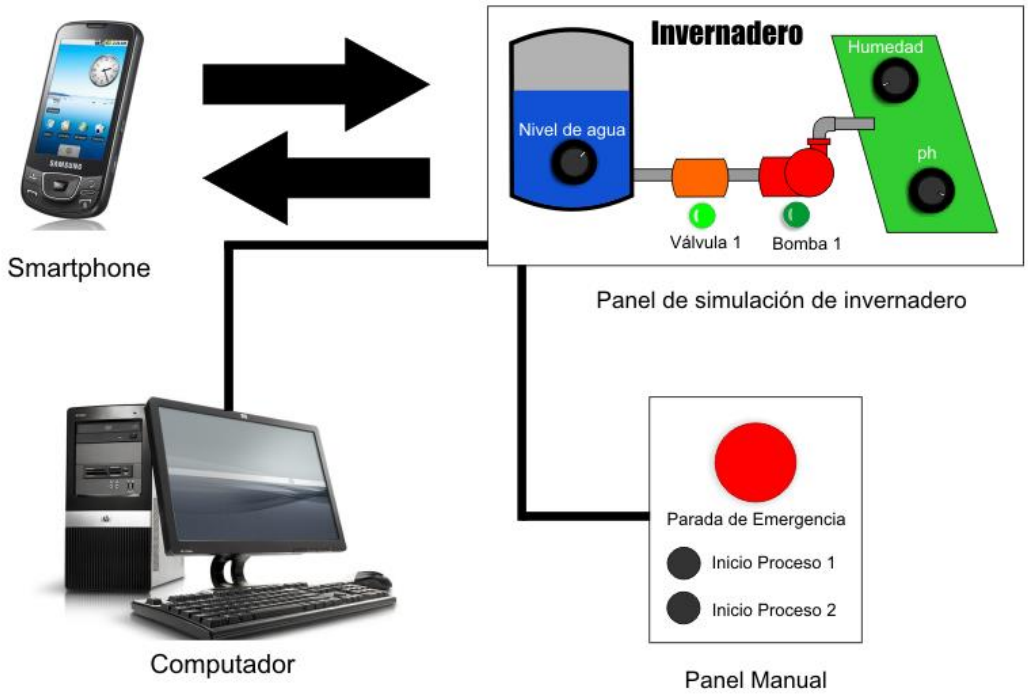


Figura 1. Esquema general del proyecto

Se simularán todas las señales del invernadero. Para los sensores, si son señales analógicas se utilizaran potenciómetros y si son señales discretas, pulsadores o switches. El sensor de temperatura será real, así como el sensor de luz del área 1. Para las señales de salida se utilizarán LED´s, motores DC y bombillas de luz para simular la fuente de temperatura y la fuente de luz del área 1. Todo esto para demostrar el funcionamiento de la aplicación y la comunicación inalámbrica.

Como se indica en las líneas anteriores la demostración del funcionamiento de la aplicación se realizara utilizando un panel de simulación dotado de LED´s, potenciómetros y otros elementos que sean necesarios, pero la aplicación estará lista para funcionar en un invernadero real.

Se utilizará un microcontrolador para digitalizar las señales del invernadero (o de la simulación del invernadero) y las envíe hacia el Smartphone y a su vez para recibir la información enviada desde el Smartphone y ejecutar las acciones correspondientes.

En la figura 1 se muestra un esquema general del proyecto.

En paralelo se implementará una aplicación para computadores personales que reemplace a la aplicación para smartphones, con las mismas funciones, pero con diferentes características gráficas y de conexión. La conexión será por medio del protocolo RS232 utilizando el puerto serial de los computadores.

1.7. Diagrama de bloques

1.7.1. Android y computador

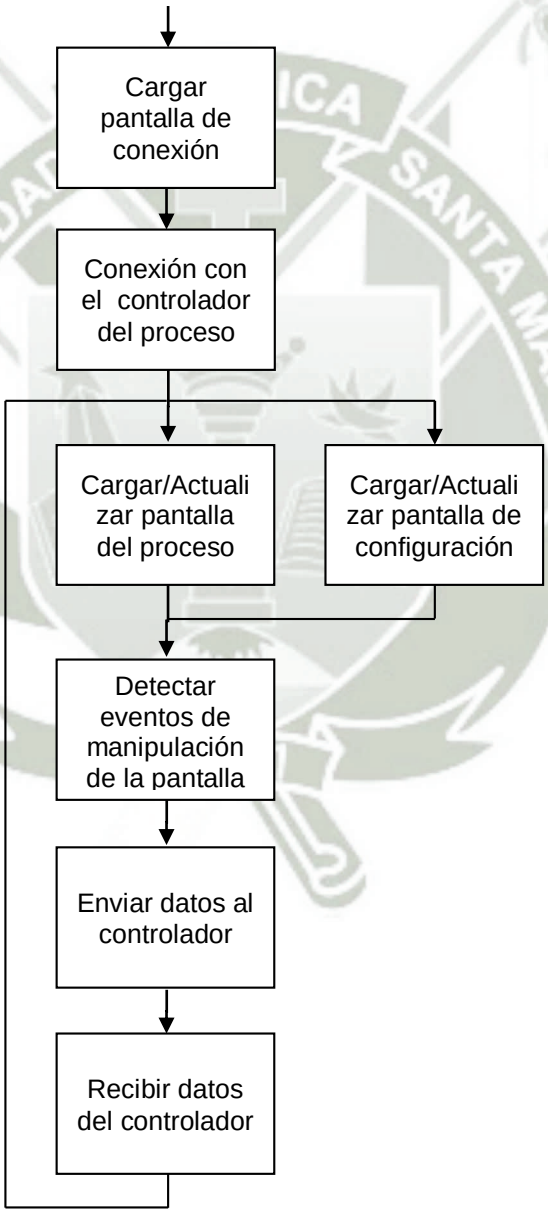


Figura 2 . Diagrama de bloques de la aplicación

1.7.2. Controlador

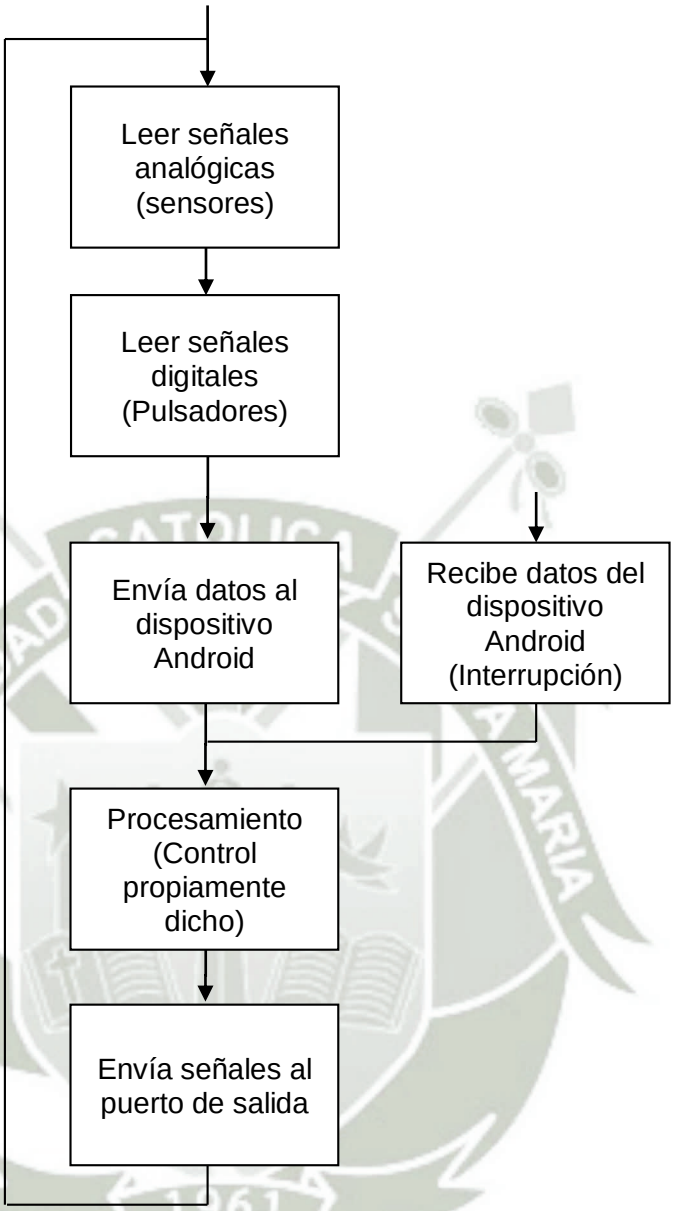


Figura 3 . Diagrama de bloques del controlador

CAPITULO II
Fundamentos Teóricos

2.1. Invernadero

Un invernadero (o invernáculo) es un lugar cerrado, estático y accesible a pie, que se destina a la producción de cultivos, dotado habitualmente de una cubierta exterior translúcida de vidrio o plástico, que permite el control de la temperatura, la humedad y otros factores ambientales para favorecer el desarrollo de las plantas. En la jardinería antigua española, el invernadero se llamaba estufa fría.

Aprovecha el efecto de la radiación solar producida por el sol que, al atravesar un vidrio u otro material traslúcido, calienta los objetos que hay adentro; estos, a su vez, emiten radiación infrarroja, con una longitud de onda mayor que la solar, por lo cual no pueden atravesar los vidrios a su regreso quedando atrapados y produciendo el calentamiento. Las emisiones del sol hacia la tierra son en onda corta mientras que de la tierra al exterior son en onda larga. La radiación visible puede traspasar el vidrio mientras que una parte de la infrarroja no lo puede hacer.

El cristal o plástico usado para un invernadero trabaja como medio selectivo de la transmisión para diversas frecuencias espectrales, y su efecto es atrapar energía dentro del invernadero, que calienta el ambiente interior. También sirve para evitar la pérdida de calor por convección. Esto puede ser demostrada abriendo una ventana pequeña cerca de la azotea de un invernadero: la temperatura cae considerablemente. Este principio es la base del sistema de enfriamiento automático autoventilación.

Figura 4 .Invernadero

En ausencia de un recubrimiento, el calor absorbido se eliminaría por corrientes convectivas y por la emisión de radiación infrarroja (longitud de onda superior a la visible). La presencia de los cristales o plásticos impide el transporte del calor acumulado hacia el exterior por convección y obstruye la salida de una parte de la radiación infrarroja. El efecto neto es la acumulación de calor y el aumento de la temperatura del recinto. Ver invernadero solar (técnica) para una discusión más detallada sobre trabajos técnica de invernadero solar.

Los vidrios tienen muy poca resistencia al paso del calor por transmisión (de hecho, para el acristalamiento sencillo, el coeficiente de transmisión térmica se considera nulo y solo se tiene en cuenta la suma de las resistencias superficiales), de modo que, contra lo que algunos creen, al tener dos temperaturas distintas a cada lado, hay notables pérdidas por transmisión (el vidrio tiene una transmitancia U = 6,4 W/m²·K, aun mayor si está en posición inclinada respecto a la vertical). El resultado es que, a mayor temperatura, menor será el efecto de retención del calor, es decir que al aumentar la temperatura aumentarán las pérdidas disminuyendo el rendimiento del sistema.

Un ejemplo de este efecto es el aumento de temperatura que toma el interior de los coches cuando están al sol. Basta una chapa metálica (los sombrajos habituales de los estacionamientos, sin ningún tipo de aislamiento térmico) que dé sombra, impidiendo el paso del sol por el vidrio, para que no se caliente tanto.

Desde la antigüedad se ha aprovechado este efecto en la construcción, no solo en jardinería. Las ventanas de las casas en países fríos son más grandes que las de los cálidos, y están situadas en los haces exteriores, para que el espesor del muro no produzca sombra. Los miradores acristalados son otro medio de ayudar al calentamiento de los locales.

2.2. SCADA

Figura 5. SCADA

Proviene de las siglas "Supervisory Control And Data Acquisition" (Control de Supervisión y Adquisición de Datos): Es un sistema basado en computadores que permite supervisar y controlar variables de proceso a distancia, proporcionando comunicación con los dispositivos de campo (controladores autónomos) y controlando el proceso de forma automática por medio de un software especializado. También provee de toda la información que se genera en el proceso productivo a diversos usuarios, tanto del mismo nivel como de

otros usuarios supervisores dentro de la empresa (supervisión, control calidad, control de producción, almacenamiento de datos, etc.).

La realimentación, también denominada retroalimentación o feedback es, en una organización, el proceso de compartir observaciones, preocupaciones y sugerencias, con la intención de recabar información, a nivel individual o colectivo, para mejorar o modificar diversos aspectos del funcionamiento de una organización. La realimentación tiene que ser bidireccional de modo que la mejora continua sea posible, en el escalafón jerárquico, de arriba para abajo y de abajo para arriba.

En teoría de la cibernética y de control, la realimentación es un proceso por el que una cierta proporción de la señal de salida de un sistema se redirige de nuevo a la entrada. Esto es de uso frecuente para controlar el comportamiento dinámico del sistema. Los ejemplos de la realimentación se pueden encontrar en la mayoría de los sistemas complejos, tales como ingeniería, arquitectura, economía, y biología. Arturo Rosenblueth, investigador mexicano y médico en cuyo seminaro de 1943 hizo una ponencia llamada “Behavior, Purpose and Teleology“ ("comportamiento, propósito y teleología"), de acuerdo con Norbert Wiener, fijó las bases para la nueva ciencia de la cibernética y propuso que el comportamiento controlado por la realimentación negativa, aplicada a un animal, al ser humano o a las máquinas era un principio determinante y directivo, en la naturaleza o en las creaciones humanas.

2.3. Android

Android es un sistema operativo móvil basado en Linux, que junto con aplicaciones middleware está enfocado para ser utilizado en dispositivos móviles como teléfonos inteligentes, tabletas, Google TV y otros dispositivos. Es desarrollado por la Open Handset Alliance, la cual es liderada por Google. Este sistema por lo general maneja aplicaciones como Google Play.

Fue desarrollado inicialmente por Android Inc., una firma comprada por Google en 2005. Es el principal producto de la Open Handset Alliance, un conglomerado

de fabricantes y desarrolladores de hardware, software y operadores de servicio. Las unidades vendidas de teléfonos inteligentes con Android se ubican en el primer puesto en los Estados Unidos, en el segundo y tercer trimestres de 2010, con una cuota de mercado de 43,6% en el tercer trimestre. A nivel mundial alcanzó una cuota de mercado del 50,9% durante el cuarto trimestre de 2011, más del doble que el segundo sistema operativo (iOS de Apple, Inc.) con más cuota.

Figura 6. Sistema Android

Tiene una gran comunidad de desarrolladores escribiendo aplicaciones para extender la funcionalidad de los dispositivos. A la fecha, se han sobrepasado las 600.000 aplicaciones (de las cuales, dos tercios son gratuitas) disponibles para la tienda de aplicaciones oficial de Android: Google Play, sin tener en cuenta aplicaciones de otras tiendas no oficiales para Android, como pueden ser la App Store de Amazon o la tienda de aplicaciones Samsung Apps de Samsung. Google Play es la tienda de aplicaciones en línea administrada por Google, aunque existe la posibilidad de obtener software externamente. Los programas están escritos en el lenguaje de programación Java. No obstante, no es un sistema operativo libre de malware, aunque la mayoría de ello es descargado de sitios de terceros.

Las aplicaciones se desarrollan habitualmente en el lenguaje Java con Android Software Development Kit (Android SDK), pero están disponibles otras

herramientas de desarrollo, incluyendo un Kit de Desarrollo Nativo para aplicaciones o extensiones en C o C++, Google App Inventor, un entorno visual para programadores novatos y varios cruz aplicaciones de la plataforma web móvil marcos. y también es posible usar las librerías Qt gracias al proyecto Necessitas SDK.

El desarrollo de aplicaciones para Android no requiere aprender lenguajes complejos de programación. Todo lo que se necesita es un conocimiento aceptable de Java y estar en posesión del kit de desarrollo de software o «SDK» provisto por Google el cual se puede descargar gratuitamente.

Todas las aplicaciones están comprimidas en formato APK, que se pueden instalar sin dificultad desde cualquier explorador de archivos en la mayoría de dispositivos.

2.4. Herramientas para Programar en Android

Para programar en Androd se requieren un conjunto de herramientas que trabajan de la mano:

2.4.1. Eclipse

Eclipse es un programa informático compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado *Java Development Toolkit* (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). Sin embargo, también se puede usar para otros tipos de aplicaciones cliente, como BitTorrent o Azureus.

Eclipse es también una comunidad de usuarios, extendiendo constantemente las áreas de aplicación cubiertas. Un ejemplo es el recientemente creado Eclipse Modeling Project, cubriendo casi todas las áreas de Model Driven Engineering.

Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

Eclipse fue liberado originalmente bajo la Common Public License, pero después fue re-licenciado bajo la Eclipse Public License. La Free Software Foundation ha dicho que ambas licencias son licencias de software libre, pero son incompatibles con Licencia pública general de GNU (GNU GPL).

2.4.2. Java

El lenguaje de programación Java fue originalmente desarrollado por James Gosling de Sun Microsystems (la cual fue adquirida por la compañía Oracle) y publicado en el 1995 como un componente fundamental de la plataforma Java de Sun Microsystems. Su sintaxis deriva mucho de C y C++, pero tiene menos facilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son generalmente compiladas a bytecode (clase Java) que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente. Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos y basado en clases que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como *WORA*, o "*write once, run anywhere*"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir del 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos 10 millones de usuarios reportados.

2.4.3. SDK Android

El SDK (Software Development Kit) de Android, incluye un conjunto de herramientas de desarrollo. Comprende un depurador de código, biblioteca, un simulador de teléfono basado en QEMU, documentación, ejemplos de código y tutoriales. Las plataformas de desarrollo soportadas incluyen Linux (cualquier distribución moderna), Max OS X 10.4.9 o posterior, y Windows XP o posterior. La plataforma integral de desarrollo (IDE, Integrated Development Environment) soportada oficialmente es Eclipse junto con el complemento ADT (Android Development Tools plugin), aunque también puede utilizarse un editor de texto para escribir ficheros Java y Xml y utilizar comandos en un terminal (se necesitan los paquetes JDK, Java Development Kit y Apache Ant) para crear y depurar aplicaciones. Además, pueden controlarse dispositivos Android que estén conectados (e.g. reiniciarlos, instalar aplicaciones en remoto).

Las Actualizaciones del SDK están coordinadas con el desarrollo general de Android. El SDK soporta también versiones antiguas de Android, por si los programadores necesitan instalar aplicaciones en dispositivos ya obsoletos o más antiguos. Las herramientas de desarrollo son componentes descargables, de modo que una vez instalada la última versión, pueden instalarse versiones anteriores y hacer pruebas de compatibilidad.

Una aplicación Android está compuesta por un conjunto de ficheros empaquetados en formato .apk y guardada en el directorio /data/app del sistema operativo Android (este directorio necesita permisos de superusuario , root, por razones de seguridad). Un paquete APK incluye ficheros .dex (ejecutables Dalvik, un código intermedio compilado), recursos, etc.

CAPITULO III
Diseño de aplicación en Android

Esta aplicación cumplirá la función de una interfaz hombre máquina (HMI) portátil y de menor costo. Debe mostrar un esquema del proceso a monitorear e intercambiará datos con el procesador de forma inalámbrica utilizando la tecnología Bluetooth.

En ella se pueden ingresar valores de Setpoint, se pueden leer valores de los sensores que intervienen en el proceso, iniciar el sistema en modo automático o manual, y activar o desactivar bombas y válvulas.

La aplicación sólo cumple la función de “interfaz” más no la de “controlador”, cada vez que se ingresan datos o se activa algún elemento en la pantalla del invernadero esta información es enviada al controlador. El controlador periódicamente actualiza la información visible en el dispositivo Android,

A continuación se explicara en detalle los bloques que forman parte del desarrollo de la aplicación. El encabezado y algunos aspectos adicionales de la programación se puede encontrar en la sección de anexos, en las páginas finales del presente trabajo de investigación.

3.1. Cargar pantalla de “Conexión”

Al ejecutar la aplicación lo primero que aparece en pantalla es una imagen que le indica al usuario que el dispositivo Android se está conectando con el controlador por medio de Bluetooth.

Esta imagen es cargada utilizando este código:

```
private Bitmap f1;
f1 = BitmapFactory.decodeResource(getResources(), R.drawable.fondo1);
canvas.scale((float)1, (float)1);
canvas.drawBitmap(f1, 0, 0, null);
```

Primero se debe definir una variable de tipo *Bitmap* en nuestro caso “f1”, luego se carga a la variable creada la imagen “fondo1” en formato *png*, definimos la escala en la que esta imagen se dibujará en la pantalla y por último se ejecuta el comando que dibuja la figura. Las coordenadas de la ubicación de la figuran indican la posición de su esquina superior izquierda.

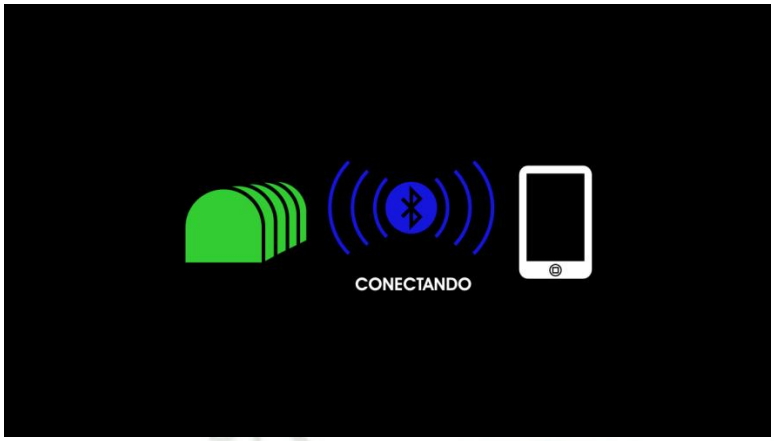
Se trabaja con imágenes en formato *png* debido a que este tipo de archivos permiten utilizar la propiedad de fondo transparente.

3.1.1. Multiresolución

Una sola aplicación puede contener los archivos necesarios para ser ejecutada en cualquier dispositivo Android, las imágenes de las diferentes resoluciones y densidades deben estar contenidas en diferentes carpetas (Fig. 7). El sistema Android dependiendo de la resolución del dispositivo selecciona automáticamente la carpeta desde donde extraer los archivos a utilizar en la aplicación.

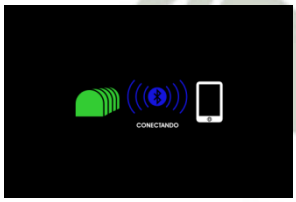
Figura 7. Carpetas del proyecto Android

Para la pantalla inicial los archivos creados son los siguientes:



Densidad: 1280 x 720
Densidad de impresión: 320ppp
Carpeta: drawable-xhdpi

Densidad: 720 x 480
Densidad de impresión: 240ppp
Carpeta: drawable-hdpi



Densidad: 480 x 320
Densidad de impresión: 160ppp
Carpeta: drawable-mdpi

Densidad: 320 x 240
Densidad de impresión: 120ppp
Carpeta: drawable-ldpi

Figura 8. Muestra de diferentes resoluciones

Para la aplicación se diseñó el siguiente icono para ser displayado en el dispositivo Android junto al resto de aplicaciones:

Figura 9. Icono de la aplicación

Este icono también se debe incorporar en todas las carpetas para estar disponible para las diferentes resoluciones.

3.2. Conexión con el controlador del proceso

Mientras la imagen de “conexión” se muestra, se realiza el enlace Bluetooth entre el dispositivo Android y el controlador.

Primero se debe crear un objeto que contenga al dispositivo remoto (controlador), para ello debemos conocer su dirección MAC, luego encendemos el adaptador Bluetooth y por último iniciamos la conexión y asignamos variables para los buffer de entrada y salida

```
myDevice = mBluetoothAdapter.getRemoteDevice("00:12:10:30:06:76");
mBluetoothAdapter.enable();
mySocket.connect();
myOutStream = mySocket.getOutputStream();
myInStream = mySocket.getInputStream();
```

3.3. Cargar/Actualizar pantalla del proceso

El procedimiento para cargar la imagen del invernadero es el mismo que para la pantalla de “conexión”. La imagen creada para el proceso es la siguiente:

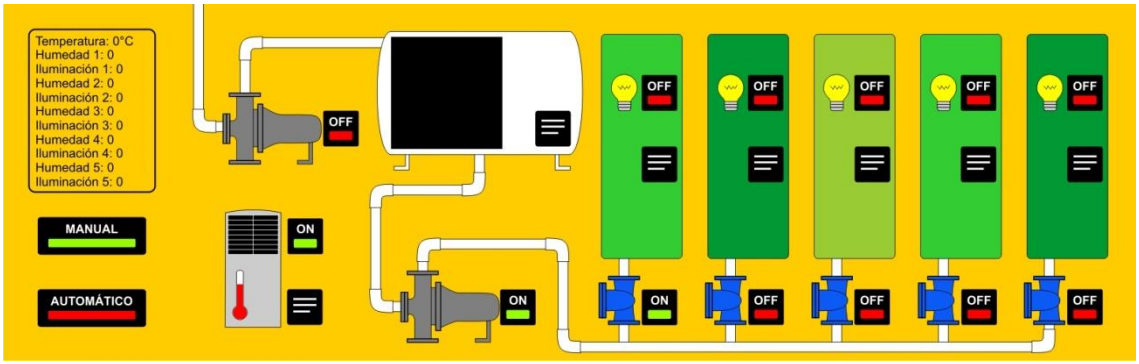


Figura 10. Imagen del proceso

En la parte superior izquierda se cuenta con una vista compacta de la lectura de todos los sensores del invernadero. Como se puede ver la imagen es bastante amplia, y definitivamente no puede ser vista íntegramente en la pantalla de un dispositivo móvil. Sólo se tiene una porción de la imagen visible, y para ver el resto es necesario deslizar un dedo en el sentido que se quiere correr la imagen.

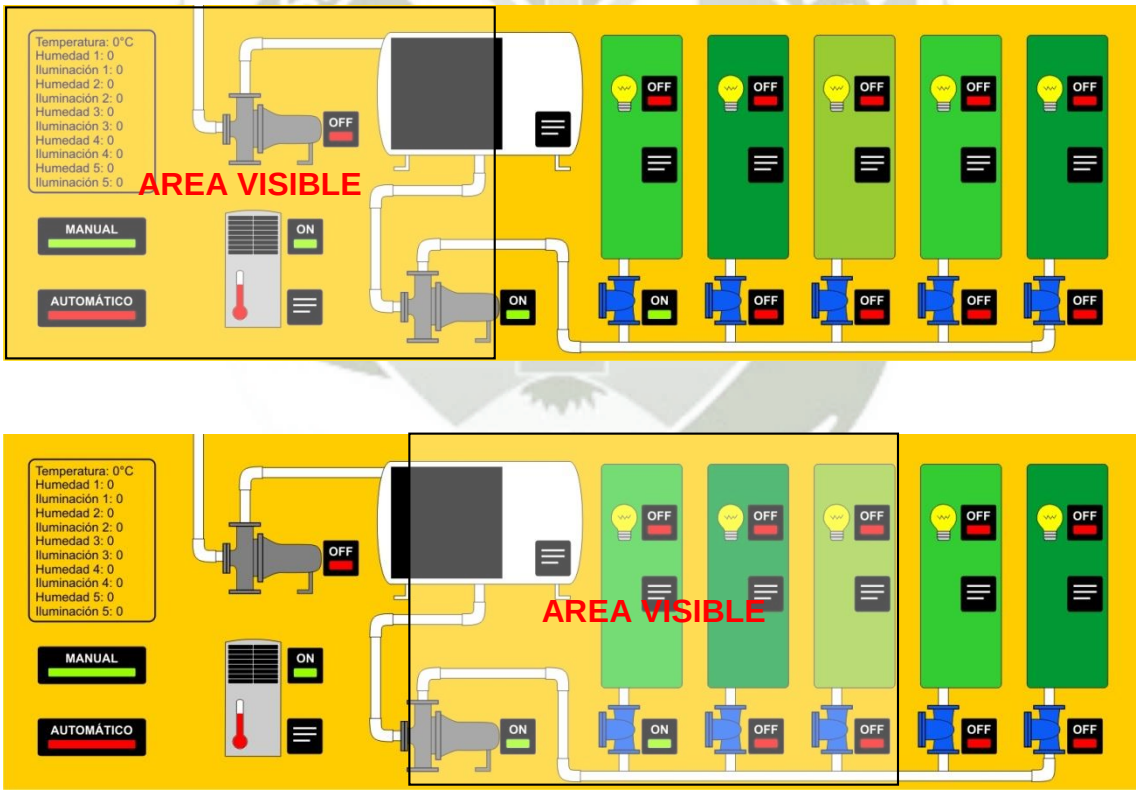


Figura 11. Área visible

En el primer caso la imagen es dibujada en las coordenadas (0,0) del dispositivo Android, mientras que en el segundo es dibujada en las coordenadas

(offset_x,0), donde offset_x es un valor negativo. Es así que para trabajar con una formula general decimos que la imagen es dibujada en las coordenadas (offset_x, offset_y). Más adelante veremos cómo se modifica ligeramente esta fórmula en función de la escala.

3.3.1. Zoom

La función de “zoom” (acercar o alejar la imagen) ha sido implementada en esta aplicación, si bien el proceso no exige esta característica, este proyecto pretende sentar las bases para futuros trabajos en los que se desee supervisar procesos más complejos con un mayor número de elementos.

Para que al hacer “zoom in” (acercar) no se distorsione la imagen o se vea poco clara, las imágenes utilizadas cuentan con el doble de resolución requerida.

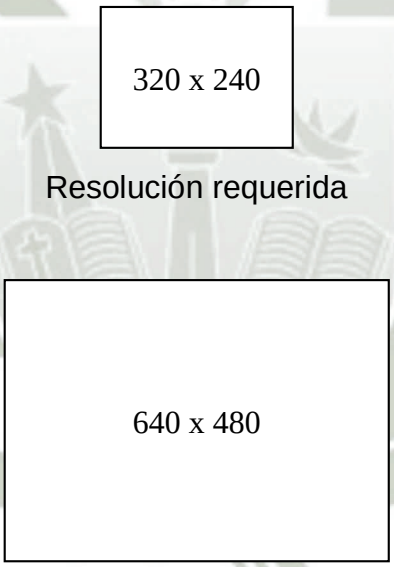


Figura 12. Ejemplo de tamaño de imágenes

Para acercar y alejar la imagen solo se deben deslizar dos dedos, como se muestra a continuación:

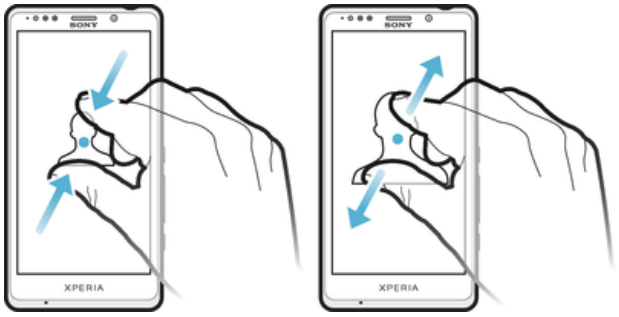


Figura 13. Método para realizar zoom

El nivel máximo de *zoom in* muestra la imagen en tamaño normal, mientras que el máximo nivel de *zoom out* muestra la imagen en la mitad de su tamaño.

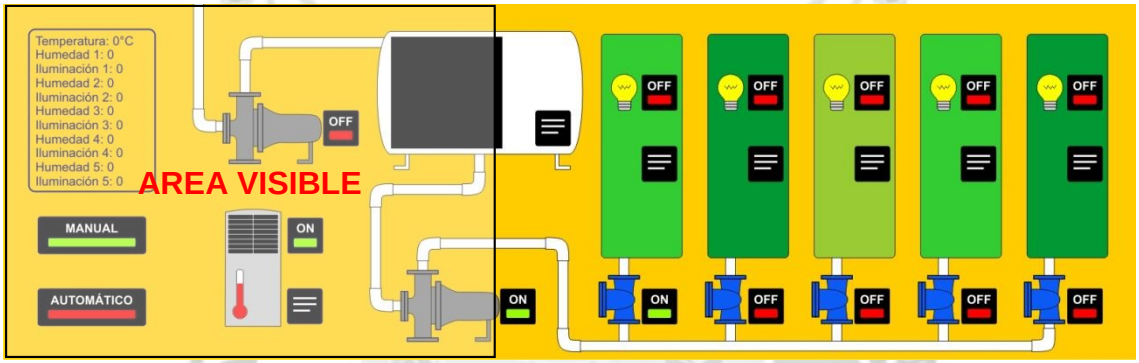


Figura 14. Máximo nivel Zoom Out

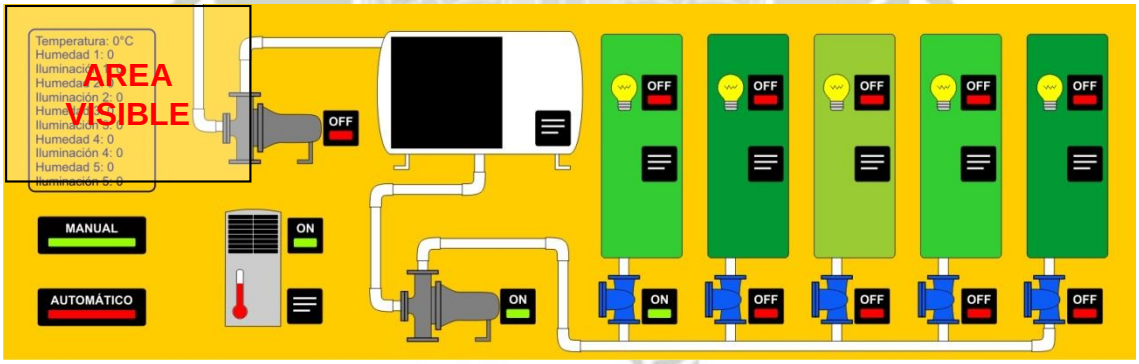


Figura 15. Máximo nivel Zoom In

3.3.2. Las figuras

La siguiente es la imagen de fondo que se utiliza para el proceso, posteriormente dependiendo de la forma de trabajo, si es manual o automático, y dependiendo de cuales elementos están encendidos y cuáles no, se van agregando figuras a la pantalla.

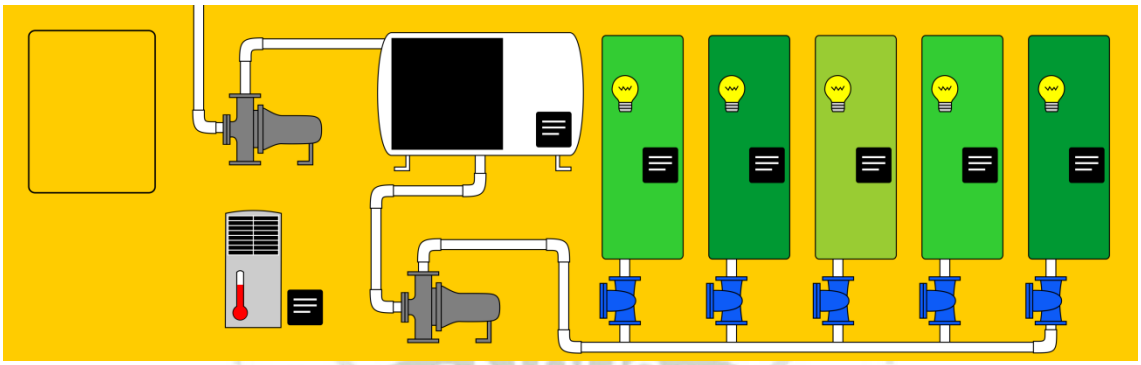


Figura 16. Fondo de la aplicación

Esta imagen tiene el nombre de “fondo3.png”, y para dibujarla con lo explicado anteriormente se utiliza el siguiente código:

```
private Bitmap f3;
f3 = BitmapFactory.decodeResource(getResources(), R.drawable.fondo3);
canvas.scale((float)final_zoom, (float)final_zoom);
canvas.drawBitmap(f3, x_offset , y_offset, paint);
```

Se crearon las siguientes figuras para ser mostradas según el estado del proceso:

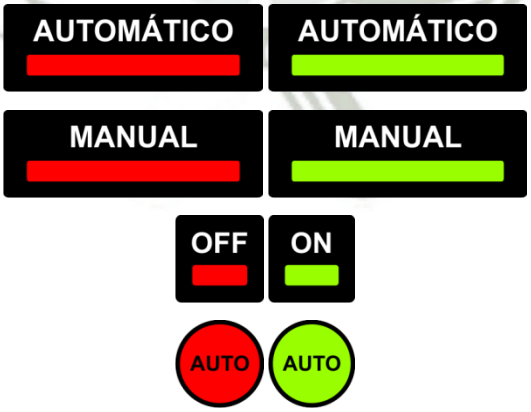


Figura 17. Botones e indicadores

A continuación se muestra un ejemplo de cómo se muestra la pantalla cuando el invernadero funciona en modo manual:

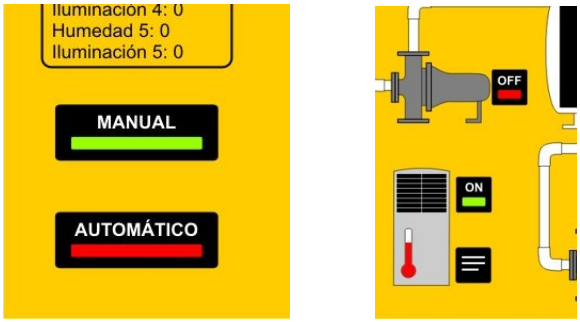


Figura 18. Ejemplo de botones e indicacores

En modo manual se pueden presionar los botones con la leyenda “on” y “off” para encender o apagar los diferentes elementos del invernadero. En modo automático, eso no es posible, sólo se muestran iconos circulares que únicamente indican el estado de determinado componente.

3.3.3. Posición relativa

Como vimos en el caso del fondo, la posición no es absoluta, por el contrario la posición de los objetos en la pantalla del proceso depende de la ubicación del fondo, de la escala y de la resolución del dispositivo.

Debido a eso es que la posición de determinado objeto en la pantalla no puede estar representada por un número de píxeles, sino por un valor proporcional a la resolución del dispositivo. Con esta nueva unidad relativa será posible realizar la programación de modo que pueda funcionar correctamente en todos los dispositivos Android.

Es así que se creó una unidad llamada “up” (unidad de pantalla), que equivale a un décimo del alto de la imagen de fondo del invernadero (Fig. 19). Por ejemplo si el dispositivo soporta una resolución de 480x320 entonces la imagen de fondo del proceso medirá 960x640, por lo tanto la unidad up tiene un valor de 64 píxeles.

480

320

Up=64píxeles

Figura 19. La unidad “UP”

En la siguiente figura se muestra la imagen del invernadero dividida en unidades “up”.

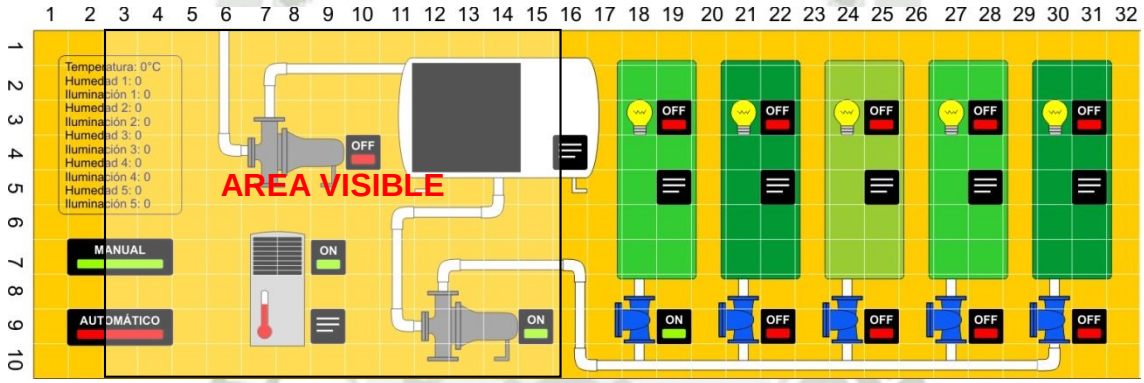


Figura 20. División del gráfico en “up”

Entonces para dibujar por ejemplo el botón de funcionamiento manual se utilizarían las siguientes coordenadas:

(x,y) = (x_offset+1*up, y_offset+6*up)

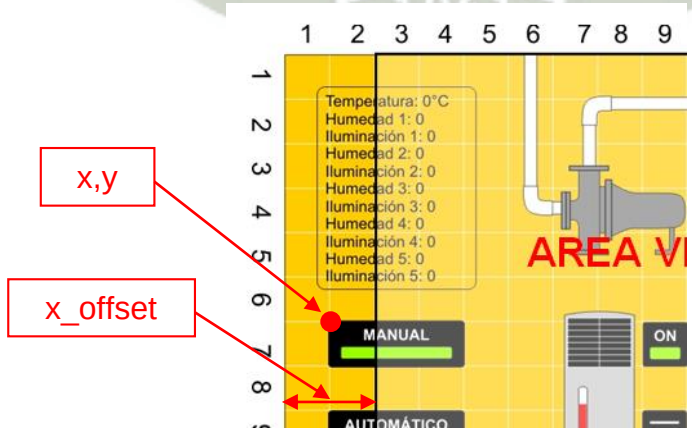


Figura 21. Coordenadas para graficar botones

En el caso del contenido del tanque de agua, se dibuja un rectángulo cuya altura varía según la lectura del sensor de nivel.

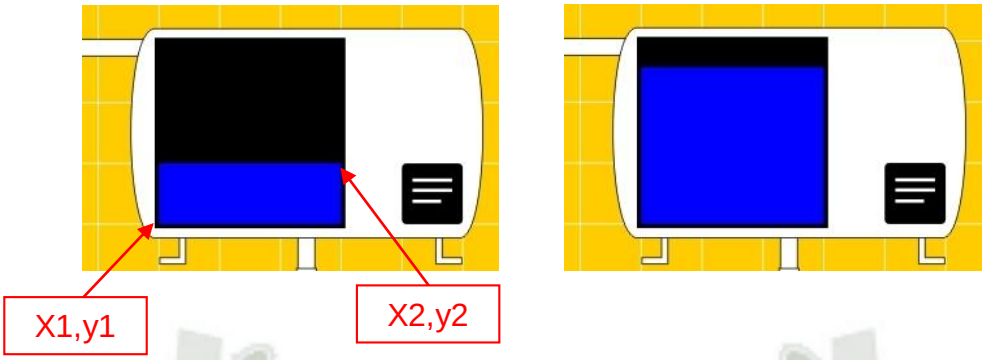


Figura 22. Gráfica del agua en el tanque

Para dibujar un rectángulo se necesitan dos coordenadas, (x1,y1) y (x2,y2). El primer punto será siempre fijo, la parte “x” de la segunda coordenada también, pero la posición vertical de este último punto varía en función del sensor de nivel del tanque de agua. Se debe hallar la ecuación de la recta que traduzca el valor medido por el sensor en “up”.

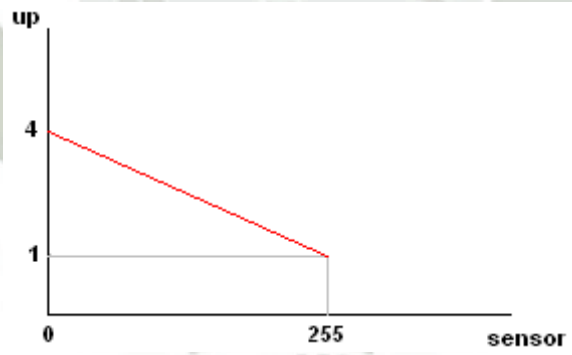


Figura 23. Recta para graficar el agua

Si el sensor lee 255 (valor máximo) el componente vertical del punto 2 debe ser 1up y si el sensor detecta que el nivel de líquido contenido es 0 entonces “y2” debe ser igual a 4up.

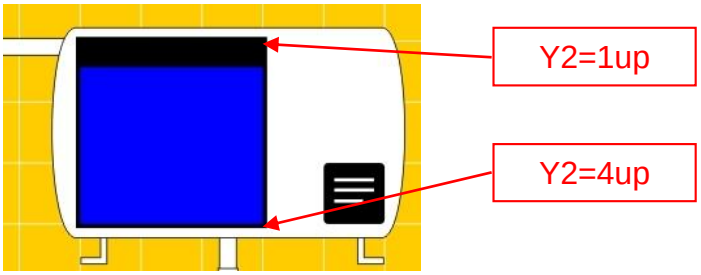


Figura 24. Nivel mínimo y máximo de "y" para graficar el agua

La ecuación de la recta para determinar el valor del sensor de nivel de agua en función de "up" es:

$$f(up)=4-3(sensor)/255$$

El código para dibujar el rectángulo de color azul es el siguiente:

```
paint.setColor(Color.BLUE);
float nivel_agua = 4 - 3*(float)sensor_tanque/255;
canvas.drawRect(x_offset+11*up, y_offset+4*up, x_offset+14*up,
                y_offset+nivel_agua*up, paint );
```

3.4. Cargar/Actualizar pantalla de configuración

En diferentes partes de la pantalla del proceso se pueden encontrar estas figuras:

Figura 25. Botón de configuración

Estos botones de color negro al ser presionados permiten acceder a un panel de configuración desde donde se pueden ingresar valores de setpoint y en donde es posible visualizar el valor actual del sensor del elemento al cual corresponde dicho botón.

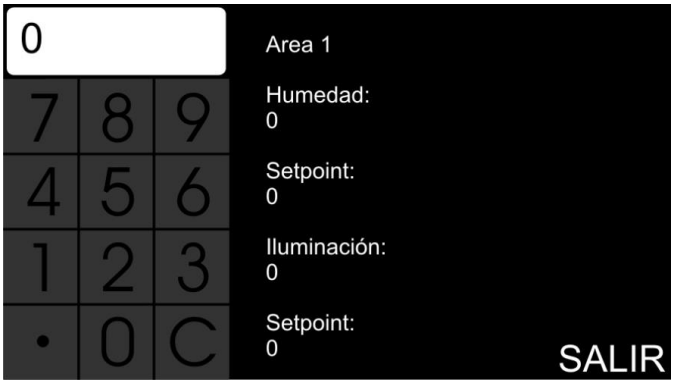


Figura 26. Pantalla de configuración

Para asignar una cantidad a uno de los campos de setpoint se debe ingresar el valor utilizando el teclado numérico y luego presionar el campo a modificar, de esta manera el valor será actualizado. Para salir de esta pantalla se debe presionar “SALIR”.

Aquí no se utilizará la función de zoom por ello la imagen que es usada para el fondo las pantallas de configuración tiene las mismas dimensiones que las del dispositivo, a diferencia de la pantalla del invernadero en donde las imágenes tienen el doble de tamaño.

En la siguiente figura se muestran la pantalla de configuración y la pantalla del proceso, segmentadas en unidades de pantalla (up).

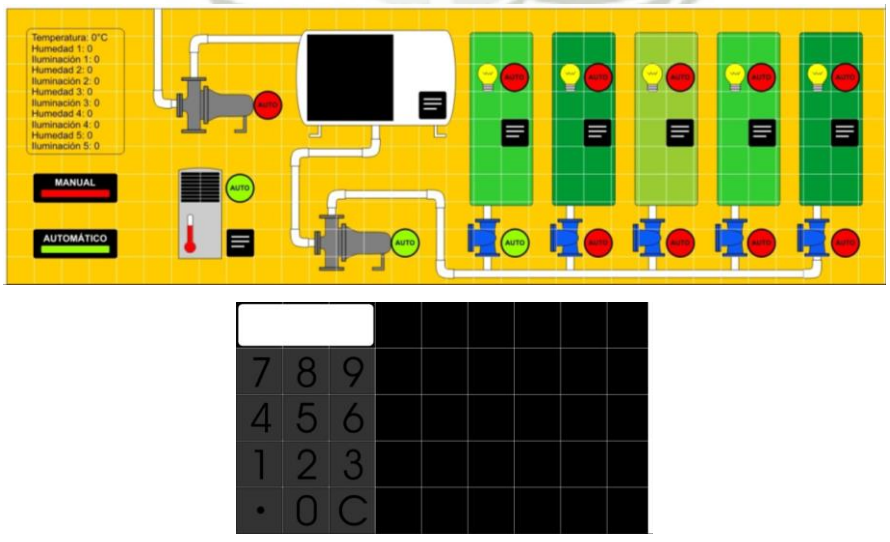


Figura 27. Segmentación virtual

Existen varios componentes del sistema que pueden ser configurables para su funcionamiento en modo automático, estos son: la temperatura, el nivel del tanque de agua, y la humedad e iluminación de cada una de las 5 áreas.

Tomaré como ejemplo el área 1, el código para dibujar el texto en esta pantalla es el siguiente:

```
canvas.drawText("Area 1", up*7/2, 2*up/3, paint);
canvas.drawText("Humedad:", up*7/2, 4*up/3, paint);
canvas.drawText(String.valueOf(sensor_hume1), up*7/2, 5*up/3, paint);
canvas.drawText("Setpoint:", up*7/2, 7*up/3, paint);
canvas.drawText(String.valueOf(setpoint_hum1), up*7/2, 8*up/3, paint);

canvas.drawText("Iluminación:", up*7/2, 10*up/3, paint);
canvas.drawText(String.valueOf(sensor_luz1), up*7/2, 11*up/3, paint);
canvas.drawText("Setpoint:", up*7/2, 13*up/3, paint);
canvas.drawText(String.valueOf(setpoint_luz1), up*7/2, 14*up/3,paint);
```

Figura 28. Graficar texto en pantalla de configuración

3.5. Detectar eventos de manipulación de la pantalla

La aplicación está desarrollada para ser instalada en dispositivos Android del tipo “touchscreen”, entonces la forma de interactuar con el sistema es mediante la pantalla.

Existen diferentes eventos reconocidos por la tecnología “touchscreen”:

- Presión de uno o más dedos en la pantalla (DOWN)
- Retirar uno o más dedos de la pantalla (UP)

- Deslizar los dedos dedo por la pantalla (MOVE)

Cuando hay más de un dedo en la pantalla a cada uno se le asigna un número, del mismo modo a los eventos que se suscitan mientras tanto, para de esa manera distinguir que evento está relacionado con cada dedo.

Para encontrar el número de puntos (dedos) presionados en la pantalla se utiliza el siguiente código:

```
numpointers=event.getPointerCount();
```

Para determinar el porcentaje de zoom que se aplicará a la pantalla del invernadero se toman las coordenadas de ambos dedos y se evalúa su distancia, comparándola con la tomada inmediatamente antes. Si la distancia entre estos puntos incrementó, entonces se amplía la imagen (zoom in), y si esta distancia es menor se aleja la imagen (zoom out). El evento reconocido en aquí es del tipo MOVE.

```
x_pointer_1=(int) event.getX(event.getPointerId(0));
y_pointer_1=(int) event.getY(event.getPointerId(0));
x_pointer_2=(int) event.getX(event.getPointerId(1));
y_pointer_2=(int) event.getY(event.getPointerId(1));
dist_pointers=(int)   Math.sqrt((x_pointer_2-x_pointer_1)*(x_pointer_2-
x_pointer_1)+(y_pointer_2-y_pointer_1)*(y_pointer_2-y_pointer_1));
```

El código anterior muestra como se halla la distancia entre ambos dedos. Por otro lado, si sólo se presiona un botón, la forma de procesar este evento de tipo (cuando se retira el dedo de la pantalla) es más sencilla, simplemente se determina la coordenada del evento y se compara con una tabla de posibilidades donde están registrados todos los botones tanto de la pantalla de invernadero como de configuración. Para determinar si el evento que se registró es ese, se utiliza el siguiente código.

```
if(event.getAction()==MotionEvent.ACTION_UP){

}
```

3.6. Enviar datos al controlador

Cada vez que se modifica un valor en la aplicación Android este es enviado inmediatamente al controlador. Cada valor del proceso tiene un código asignado, de modo que, antes de enviar cada valor se envía su código para que el controlador pueda identificar a que variable pertenece ese dato. Por ejemplo si se está trabajando en modo manual y se enciende la bomba de riego se enviarían los siguientes datos:

```
myOutStream.write(115);
myOutStream.write(flag_bomba_riego);
```

A continuación se muestran una tabla con los códigos asignados a cada valor del proceso:

Código	Variable
100	Setpoint tanque
101	Setpoint temperatura
102	Setpoint luz1
103	Setpoint luz2
104	Setpoint luz3
105	Setpoint luz4
106	Setpoint luz5
107	Setpoint humedad1
108	Setpoint humedad2
109	Setpoint humedad3
110	Setpoint humedad4
111	Setpoint humedad5
112	Manual on/off
113	Automático on/off
114	Bomba tanque on/off
115	Bomba de riego on/off
116	Temperatura on/off

117	Válvula1 on/off
118	Válvula2 on/off
119	Válvula3 on/off
120	Válvula4 on/off
121	Válvula5 on/off
122	Luz1 on/off
123	Luz2 on/off
124	Luz3 on/off
125	Luz4 on/off
126	Luz5 on/off

Tabla 1. Tabla de valores del proceso

Si la variable es de naturaleza digital (on/off) su contenido será “0” o “100”, y si la varialbe es por ejemplo el valor de setpoint de uno de los elementos contiene valores ente 0 y 255.

De manera similar el controlador, a solicitud del dispositivo Android, envía información hacia este, y la forma en que lo hace es mediante una trama. Le interface solicita al controlador el envío de información mandado estos dos números:

```
myOutStream.write(50);
myOutStream.write(50);
```

3.7. Recibir datos del controlador

Los datos enviados por el controlador se leen del buffer con esta línea de código:

```
bbbytes = myInStream.read(bbbuffer);
```

Una vez leídos los datos del buffer procedemos a hacer la búsqueda del encabezado de la trama. Los primeros 5 datos de la trama son letras que forman la abreviatura “INVER”, y los datos a continuación constituyen toda la información relevante del proceso (valores de sensores, pulsadores, estado de válvulas y

bombas, etc). Del mismo modo que para el envío de datos hacia el controlador, la recepción también tiene un orden, a continuación se muestra una tabla con el orden de los datos leídos en el buffer.

Posición en el Buffer	Dato
Bbbytes[0]	I
Bbbytes[1]	N
Bbbytes[2]	V
Bbbytes[3]	E
Bbbytes[4]	R
Bbbytes[5]	Sensor Nivel Tanque
Bbbytes[6]	Sensor Temperatura
Bbbytes[7]	Sensor Luz1
Bbbytes[8]	Sensor Luz2
Bbbytes[9]	Sensor Luz3
Bbbytes[10]	Sensor Luz4
Bbbytes[11]	Sensor Luz5
Bbbytes[12]	Sensor Humedad1
Bbbytes[13]	Sensor Humedad2
Bbbytes[14]	Sensor Humedad3
Bbbytes[15]	Sensor Humedad4
Bbbytes[16]	Sensor Humedad5
Bbbytes[17]	Setpoint Tanque
Bbbytes[18]	Setpoint Temperatura
Bbbytes[19]	Setpoint Luz1
Bbbytes[20]	Setpoint Luz2
Bbbytes[21]	Setpoint Luz3
Bbbytes[22]	Setpoint Luz4
Bbbytes[23]	Setpoint Luz5
Bbbytes[24]	Setpoint Humedad1
Bbbytes[25]	Setpoint Humedad2
Bbbytes[26]	Setpoint Humedad3
Bbbytes[27]	Setpoint Humedad4

Bbbytes[28]	Setpoint Humedad5
Bbbytes[29]	Manual on/off
Bbbytes[30]	Automático on/off
Bbbytes[31]	Temperatura on/off
Bbbytes[32]	Bomba Tanque on/off
Bbbytes[33]	Bomba Riego on/off
Bbbytes[34]	Válvula1 on/off
Bbbytes[35]	Válvula2 on/off
Bbbytes[36]	Válvula3 on/off
Bbbytes[37]	Válvula4 on/off
Bbbytes[38]	Válvula5 on/off
Bbbytes[39]	*Valor de seguridad = 200
Bbbytes[40]	Luz1 on/off
Bbbytes[41]	Luz2 on/off
Bbbytes[42]	Luz3 on/off
Bbbytes[43]	Luz4 on/off
Bbbytes[44]	Luz5 on/off

Tabla 2. Tabla de valores del buffer de entrada

El valor de seguridad se utiliza para corroborar el correcto orden de los datos, si Bbbytes[39] es igual a 200 entonces la información recibida está ordenada, sino se procede a hacer una nueva lectura del buffer.

CAPITULO IV
Diseño e implementación del controlador

Parte importante del proyecto es la implementación de hardware, que constituye la interface física entre el dispositivo Android y el proceso. Como se mencionó con anterioridad el sistema se simula en un panel en el que se pueden proporcionar los valores de los sensores por medio de potenciómetros y se visualiza el estado on/off de algunos elementos mediante indicadores LED.

Como controlador del proceso se seleccionó un microcontrolador PIC de la marca Microchip con soporte para comunicación serial y entradas analógicas. El elemento seleccionado tiene el código 16f877, muy popular en nuestro medio, de fácil acceso y de bajo costo. Además la tecnología necesaria para desarrollar con esta marca también es de fácil acceso. En la sección de anexos se puede encontrar mayor información acerca de este componente.

Figura 29. Microcontrolador

4.1. Bluetooth

La comunicación entre el dispositivo Android y el controlador se realiza por via Bluetooth, puesto que es una solución económica y efectiva y la mayoría de dispositivos Android cuentan con un controlador Bluetooth incorporado.

Del otro lado del sistema, el controlador tiene que interpretar las señales inalámbricas y debe enviar señales del mismo tipo. Afortunadamente en el mercado existen elementos que cumplen esta función, son tarjetas de reducidas dimensiones que convierten las señales Bluetooth en niveles TTL y viceversa.

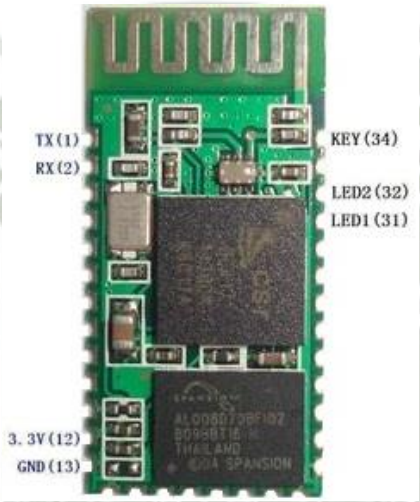


Figura 30. Módulo Bluetooth

La tarjeta utilizada se puede encontrar en la red con el nombre de “módulo Bluetooth HC-05”, como todo circuito de este tipo tiene sus líneas de alimentación y de transmisión y recepción de datos, además cuenta con una salida para un LED indicador que emite una señal intermitente si el módulo aún no está enlazado con otro dispositivo Bluetooth, y una señal constante de nivel lógico “1” si está conectado. También tiene una entrada de configuración con la que se pueden modificar la velocidad de transmisión, el nombre, la contraseña, etc.

Las características más importantes que trae por defecto este módulo son las siguientes:

Característica	Valor
Nombre	Linvor
Contraseña	1234
Velocidad de transmición	9600 bps

Tabla 3. Características del módulo bluetooth

4.2. El circuito

El circuito recibe las señales analógicas de los sensores y potenciómetros que en este proyecto simulan las señales del resto de sensores, las señales discretas de algunos pulsadores que se encuentran en el panel físico de operación manual, y genera las señales para encender y apagar motores, relays e indicadores luminosos que representan válvulas, y otros actuadores.

4.2.1. El microcontrolador

El microcontrolador elegido es de la marca Microchip y tiene el código 16877 debido a que cuenta con gran número de puertos de entrada y salida, posee un módulo interno de comunicación serial y soporta el ingreso de señales analógicas. Para este proyecto el microcontrolador Pic utiliza un cristal de 20Mhz.

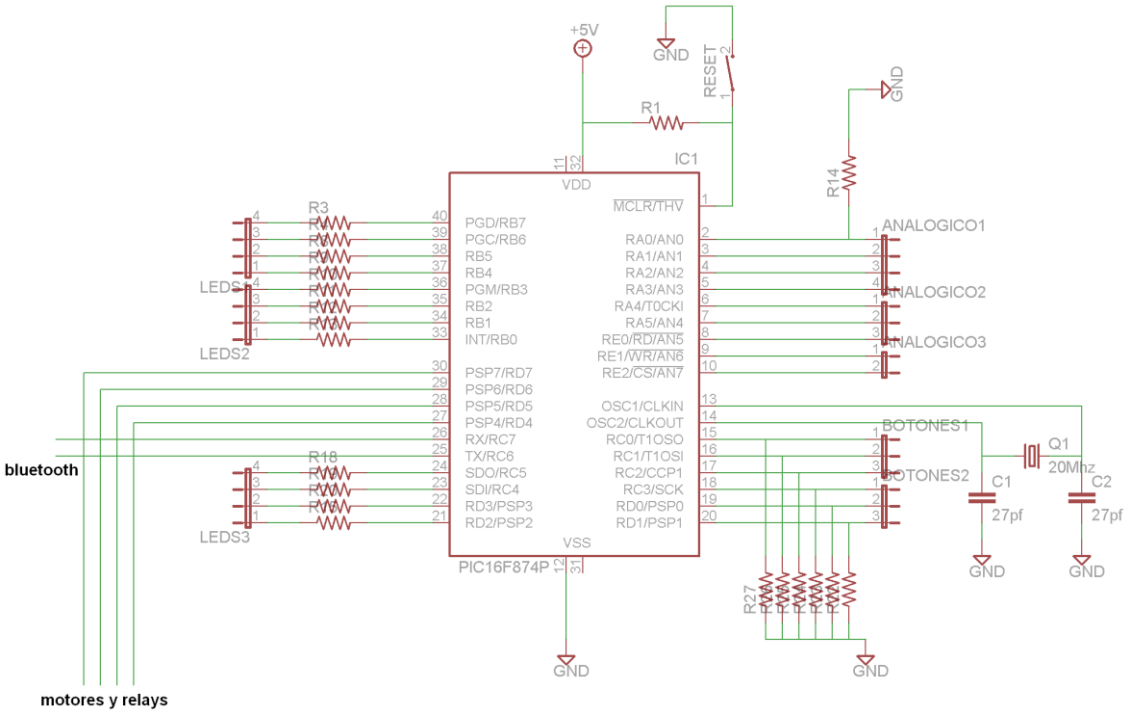


Figura 31. Circuito del sistema

4.2.2. Alimentación

El circuito puede ser alimentado con voltajes mayores de 7v de corriente continua, cuenta con un regulador 5v y otro de 3.3v ya que el módulo Bluetooth opera con ese nivel de tensión.

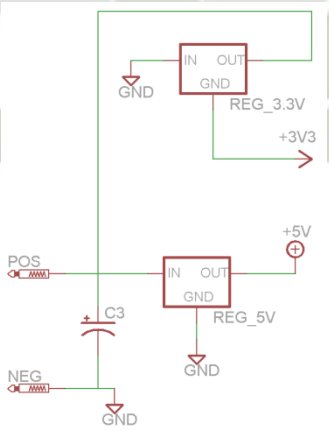


Figura 32. Etapa de alimentación

4.2.3. Acondicionamiento bluetooth

Ya que el microncontrolador se alimenta con 5v y el módulo bluetooth con 3.3v fue necesario implementar un circuito de acondicionamiento para el intercambio de señales entre ambos.

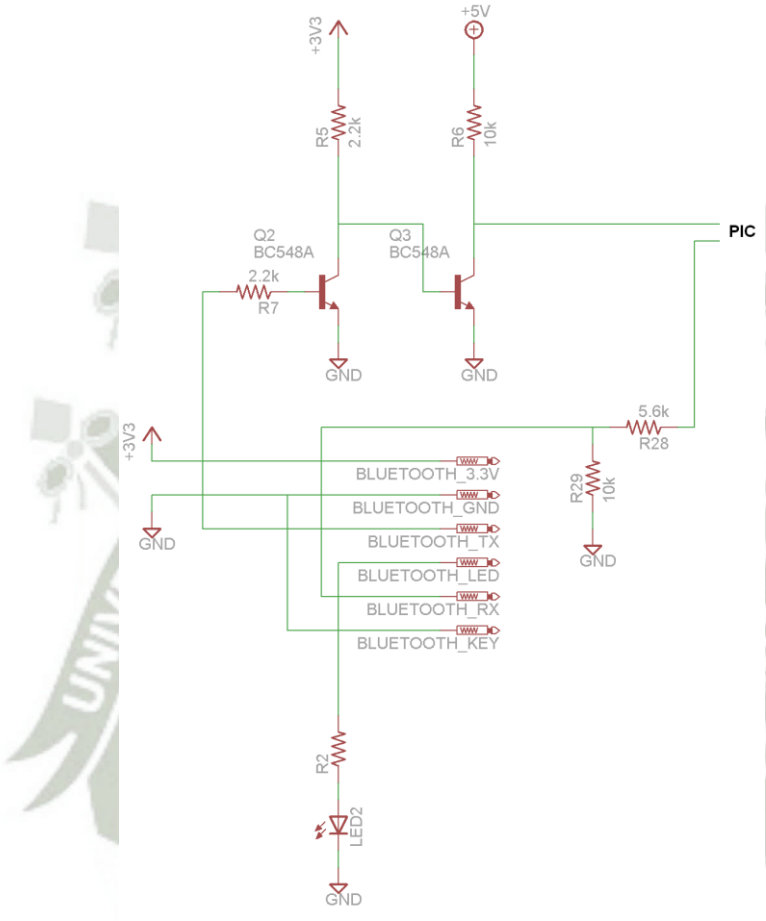


Figura 33. Etapa de acondicionamiento de la etapa bluetooth

En primer lugar convertimos la señal que envía el modulo bluetooth al microcontrolador. Para primero amplificamos la corriente para poder excitar un segundo transistor en configuración “corte-saturacion”.

$$I_B = \frac{V_B - V_{BE}}{R_B}$$

$$I_B = \frac{3.3 - 0.7}{2.2k\Omega}$$

$$I_B = 1.18mA$$

$$I_C = hFe\ x\ I_B$$

Donde:

hFe = 90 según datasheet del transistor BC548

$$I_C = 90\ x\ 1.18$$

$$I_C = 106.36mA$$

Esta corriente ingresa al segundo transistor que es que se encarga de convertir el voltaje de 3.3v a 5v, el cual está configurado como un circuito de "corte-saturación".

Del mismo modo se debe convertir la señal que envía el microcontrolador al PIC, esto se logra implementando un divisor de tensión para reducir el voltaje entregado por el controlador (5v) a 3.3v que es el voltaje al cual funciona el módulo bluetooth.

$$V = \frac{R29}{R29 + R28} VPIC$$

$$V = \frac{10k\Omega}{10k\Omega + 5.6k\Omega} 5v$$

$$V = 3.2v$$

4.2.4. Motores

Se utilizan motores de 5v para simular el funcionamiento de las bombas, puesto que la corriente emitida por el microcontrolador no es lo suficiente para mover un motor, existen circuitos muy utilizados para cubrir esta deficiencia.

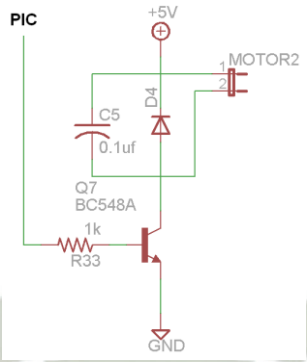


Figura 34. Acondicionamiento de motores

La ecuación que para hallar el valor de la resistencia es la siguiente:

$$R = \frac{Voltaje - 0.7}{\frac{Corriente}{hFe}}$$

Voltaje = 5v

Corriente = 400mA (consumo del motor)

hFe = 90 según datasheet del transistor BC548

$$R = \frac{5 - 0.7}{\frac{400mA}{90}}$$

$$R = 1057.5\Omega$$

$$R = 1k\Omega \ \ (Valor\ comercial)$$

4.2.5. Relays

Para simular la fuente de temperatura y una de las fuentes de luz se utilizan focos de 220vac, y como interruptores se utilizan relays que serán manejados por el microcontrolador.

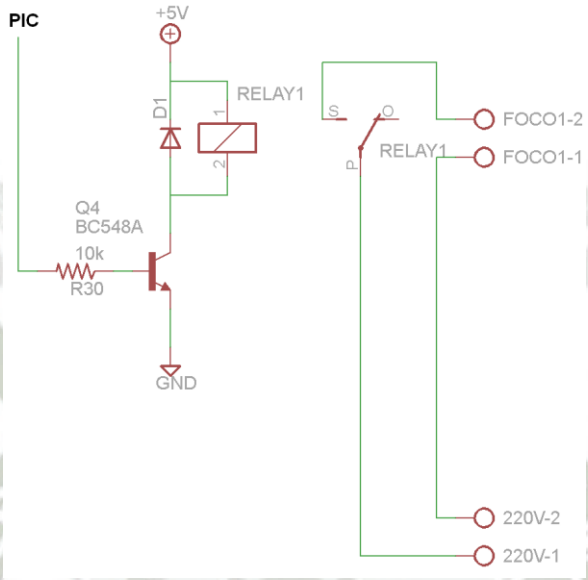


Figura 35. Acondicionamiento de relays

La ecuación que para hallar el valor de la resistencia es la siguiente:

$$R = \frac{Voltaje - 0.7}{\frac{Corriente}{hFe}}$$

Voltaje = 5v

Corriente = 40mA según datasheet del Relay

hFe = 90 según datasheet del transistor BC548

$$R = \frac{5 - 0.7}{\frac{40mA}{90}}$$

$$R = 10{,}575\Omega$$

$$R = 10k\Omega \ \ (Valor\ comercial)$$

4.2.6. Sensor de temperatura

Como sensor de temperatura se eligió el LM35, este sensor no requiere algún circuito de acondicionamiento. La señal de salida de este sensor se entrega directamente a una de las entradas analógicas del microcontrolador.

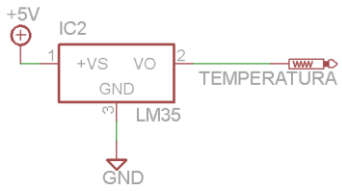


Figura 36. Sensor de temperatura

4.2.7. Sensor de luz

Como sensor de luz se utilizó una fotoresistencia (LDR), cuya resistencia disminuya a mayor cantidad de luz. Necesitan de una resistencia adicional para crear el efecto de divisor de voltaje. Esta resistencia adicional ya está incluida en la placa.

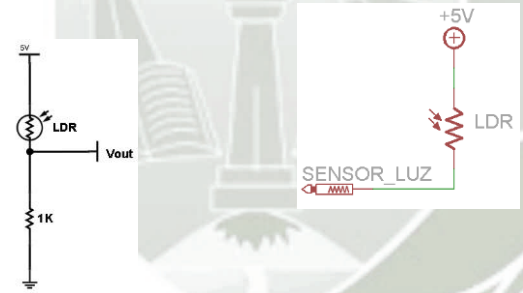


Figura 37. Fotorresistencia

El voltaje que ingresa al controlador está representado por la siguiente ecuación:

$$R = \frac{R}{LDR + R}\ x\ Voltaje$$

Voltaje = 5v

R = 1KΩ

$$R = \frac{5\ 000}{LDR + 1000}$$

4.2.8. Potenciometros

El resto de sensores de luz y los de humedad se simulan utilizando potenciómetros, y están conectados al PIC de la siguiente forma.

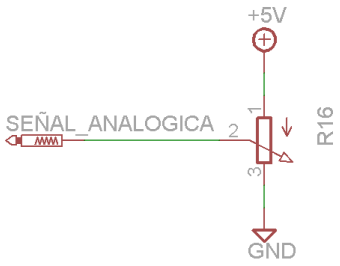


Figura 38. Potenciómetro

4.2.9. Led´s

Para encender un LED se debe conectar este por medio de una resistencia (generalmente de 330ohm) a la salida digital. Para todos los indicadores luminosos las resistencias están incluidas en la placa.

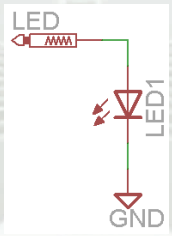


Figura 39. Led´s

La ecuación que para hallar el valor de la resistencia es la siguiente:

$$R = \frac{Voltaje}{Corriente}$$

Voltaje = 5v

Corriente = 20mA según datasheet del LED

$$R = \frac{5v}{20mA}$$

$$R = 250\Omega$$

$$R = 330\Omega \ (Valor\ comercial)$$

4.2.10. Pulsadores

El panel manual contiene indicadores luminosos y pulsadores para poder interactuar con el sistema sin necesidad del dispositivo Android, es evidente que esta interacción se limita a algunas funciones básicas del sistema.

La entrada digital debe estar conectada constantemente a tierra por medio de una resistencia para que reciba un nivel lógico igual a “0”, esta resistencia recibe el nombre de “pull down”, de modo que cuando se presione el pulsador se cree un puente entre 5v y el puerto de entrada. Estas resistencias están incluidas en la placa.

Figura 40. Pulsadores

4.3. Leer señales analógicas (sensores)

Las señales analógicas provenientes de los potenciómetros son:

Puerto analógico del PIC	Señal simulada
RA0	Sensor de nivel de agua en tanque
RA1	Sensor de Temperatura
RA2	Sensor de Luz 1
RA3	Sensor de Luz 2
RA5	Sensor de Luz 3
RE0	Sensor de Humedad 1
RE1	Sensor de Humedad 2
RE2	Sensor de Humedad 3

Tabla 4. Señales analógicas

Para leer una señal analógica primero se debe seleccionar el canal a leer y se inicia el proceso de conversión, se esperan uno milisegundos para que la lectura se realiza de forma satisfactoria, se verifica que haya finalizado la conversión y luego se asigna el valor leído a la variable correspondiente.

4.4. Leer señales digitales (Pulsadores)

Las señales discretas de entrada son generadas por pulsadores que se encuentran en el panel de control manual del proceso. En este panel manual se puede iniciar el sistema en modo manual o automático, y si está corriendo en modo manual se puede manipular la bomba que llena el tanque, la bomba de riego, y la válvula y la iluminación del área 1.

Puerto digital del PIC	Botón en el panel manual

RC0	Manual
RC1	Automático
RC2	ON/OFF Bomba tanque
RC3	ON/OFF Bomba riego
RD0	ON/OFF Válvula área 1
RD1	ON/OFF Luz área 1

Tabla 5. Señales digitales

Las líneas de código para leer las entradas digitales son similares para cada entrada, se lee el puerto para reconocer algún cambio, si la señal está en nivel lógico “1” se engancha el programa del microcontrolador hasta que el nivel vuelva a ser bajo (se suelte el botón). Cada entrada digital tiene asignada una variable del tipo “toogle” (cada vez que se presiona el botón cambia de estado), estas variables posteriormente serán enviadas al dispositivo Android y pueden contener 2 únicos valores: 0 (off) o 100 (on).

Figura 41. Panel manual

4.5. Enviar datos al dispositivo Android

El envío de datos se realiza utilizando la unidad USART (transmisor receptor universal síncrono y asíncrono) del microcontrolador. Se debe configurar la velocidad de transmisión a 9600bps, con 8 bits de datos y un bit de parada.

Cada cierto tiempo el dispositivo Android envía una solicitud de información al microcontrolador a través de dos datos consecutivos “50” y “50”, una vez que el controlador recibe esa orden procede a realizar el traslado de datos vía Bluetooth.

Para enviar esta información se carga el dato deseado en el buffer de salida y se esperan unos milisegundos, hasta que se complete la transmisión y se repite ese procedimiento para todos. El tren de datos que se envía tiene un encabezado el encabezado que servirá para que el Smartphone pueda determinar el inicio de la trama, además, en medio de la trama se envía un dato de verificación, para corroborar que los datos llegan en el orden esperado.

Nro de dato	Dato
1	“I”
2	“N”
3	“V”
4	“E”
5	“R”
6	Sensor de nivel tanque de agua
7	Sensor de temperatura
8	Sensor de luz área 1
9	Sensor de luz área 2
10	Sensor de luz área 3
11	Sensor de luz área 4
12	Sensor de luz área 5
13	Sensor de humedad área 1
14	Sensor de humedad área 2
15	Sensor de humedad área 3

16	Sensor de humedad área 4
17	Sensor de humedad área 5
18	Setpoint nivel tanque de agua
19	Setpoint temperatura
20	Setpoint luz área 1
21	Setpoint luz área 2
22	Setpoint luz área 3
23	Setpoint luz área 4
24	Setpoint luz área 5
25	Setpoint humedad área 1
26	Setpoint humedad área 2
27	Setpoint humedad área 3
28	Setpoint humedad área 4
29	Setpoint humedad área 5
30	Estado botón manual
31	Estado botón automático
32	Estado botón temperatura
33	Estado botón bomba tanque
34	Estado botón bomba riego
35	Estado botón válvula 1
36	Estado botón válvula 2
37	Estado botón válvula 3
38	Estado botón válvula 4
39	Estado botón válvula 5
40	**Byte de verificación = 200**
41	Estado botón luz 1
42	Estado botón luz 2
43	Estado botón luz 3
44	Estado botón luz 4
45	Estado botón luz 5

Tabla 6. Orden de datos enviados hacia el dispositivo Android

4.6. Recibir datos del dispositivo Android

La recepción de datos se rige según la configuración realizada para la transmisión. Cada vez que llega un dato al controlador se genera una interrupción, se almacena ese dato en una variable y se espera la llegada del segundo dato a buffer de entrada. Siempre se reciben dos datos, el primero indica mediante un número a que variable corresponde el dato que viene a continuación. Si ambos datos recibidos tiene el valor de 50, entonces esa es la orden para enviar una nueva trama hacia el dispositivo Android como se explica en el punto anterior.

Dato 1	Dato 2
100	Setpoint tanque de agua
101	Setpoint temperatura
102	Setpoint luz área 1
103	Setpoint luz área 2
104	Setpoint luz área 3
105	Setpoint luz área 4
106	Setpoint luz área 5
107	Setpoint luz humedad 1
108	Setpoint luz humedad 2
109	Setpoint luz humedad 3
110	Setpoint luz humedad 4
111	Setpoint luz humedad 5
112	Estado botón manual
113	Estado botón automático
114	Estado botón bomba de tanque
115	Estado botón bomba de riego
116	Estado botón temperatura
117	Estado botón válvula 1
118	Estado botón válvula 2
119	Estado botón válvula 3
120	Estado botón válvula 4
121	Estado botón válvula 5

122	Estado botón luz área 1
123	Estado botón luz área 2
124	Estado botón luz área 3
125	Estado botón luz área 4
126	Estado botón luz área 5

Tabla 7. Orden de datos enviados desde el dispositivo Android

4.7. Procesamiento (Control propiamente dicho)

El control de este proceso es del tipo ON/OFF, dado que no exige otro de mayor complejidad. En las líneas siguientes se hará una descripción de las condiciones que regulan el funcionamiento del sistema en modo automático.

-Si el valor del sensor de nivel del tanque de agua es menor que el valor de su setpoint, se enciende la bomba de llenado del tanque, y permanecerá prendida hasta que el nivel sea igual o mayor al deseado.

-Si el valor del sensor de temperatura es menor que el valor de su setpoint, se enciende el climatizador hasta que la temperatura sea igual o mayor al deseado.

-Si el valor del sensor de luz del área 1 es menor que el valor de su setpoint, se enciende la fuente de luz, y si el nivel de humedad es menor que el de su setpoint se abre la válvula que permite el flujo de agua hacia el área 1. La misma lógica se aplica para el resto de las 4 áreas.

-Si una de las 5 válvulas está abierta automáticamente se enciende la bomba de riego.

4.8. Enviar señales al puerto de salida

Las salidas del control del invernadero son señales discretas de 0 o 5v, que son el resultado del procesamiento del estado del proceso en modo automático o de órdenes dictadas directamente por el operador en funcionamiento manual. Todas estas salidas son simuladas con indicadores luminosos.

PANEL DE SIMULACIÖN	
Salida digital del PIC	**Elemento**
RD5	ON/OFF Bomba tanque
RD4	ON/OFF Bomba riego
RD6	ON/OFF Temperatura
RC5	ON/OFF Luz área 1
RB6	ON/OFF Luz área 2
RB7	ON/OFF Luz área 3
RB5	ON/OFF válvula 1
RB4	ON/OFF válvula 2
RB3	ON/OFF válvula 3

Tabla 8. Salidas digitales del microcontrolador

PANEL DE CONTROL MANUAL	
Salida digital del PIC	**Elemento**
RB0	Funcionamiento manual
RB1	Funcionamiento atomático
RB2	ON/OFF Bomba tanque
RD3	ON/OFF Bomba riego
RC4	ON/OFF válvula 1
RC5	ON/OFF Luz área 1

Tabla 9. Entradas digitales del microcontrolador

CAPITULO V
Aplicación para Computadores

Para la programación de esta aplicación utilizamos una herramienta de libre descarga conocida como NetBeans, este software permite desarrollar soluciones utilizando el lenguaje de programación Java. NetBeans cuenta con un módulo para implementar interfaces gráficas al estilo de Visual Basic, de este modo se pueden arrastrar componentes como cuadros de texto, botones, etc. hacia la pantalla de la aplicación.

5.1. Descripción

Para darle mayor soporte al proyecto se implementó una aplicación Java para computadores, de modo que el usuario puede elegir con que plataforma trabajar. Este programa cuenta con las mismas funciones que la aplicación móvil, pero sin las características gráficas, es decir que en lugar de visualizar gráficamente el invernadero y sus elementos únicamente se podrán observar botones y texto como instrumentos de interacción.

5.2. Características

Se pueden leer los valores de todos los sensores y se pueden ingresar y leer los valores de “setpoint”. También incluye las funciones de modo automático y modo manual, así como botones para controlar directamente cada elemento en el modo de funcionamiento manual.

La comunicación que se emplea entre el computador y el sistema utiliza el protocolo RS232, quiere decir que la comunicación necesita conexión física.

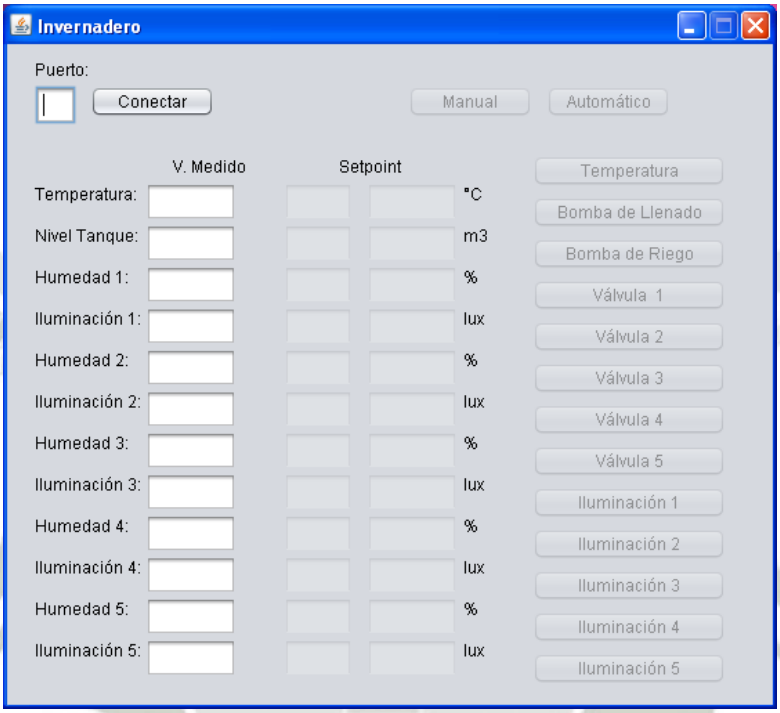


Figura 42. Interface Java para computadores

5.3. Funcionamiento

La figura 42 muestra la interface gráfica para computadores. Lo primero que se debe hacer es seleccionar el puerto serial al que está conectado el invernadero y presionar “conectar”.

Los botones indican el estado de los actuadores, si están en color verde están en funcionamiento, si están en color rojo es porque no están operando. Además en modo manual estos botones pueden encender y apagar cada actuador. Se pueden configurar los valores de setpoint, así como también se pueden leer los

valores presentes en el controlador del proceso. El valor de todos los sensores se encuentran a la izquierda de la interface.

Figura 43. Interface en funcionamiento

5.4. Programación

Lo primero que debemos hacer en la programación es configurar la comunicación serial. En las líneas siguientes se puede observar la manera de ingresar los valores de configuración.

```
SerialPort serialPort = (SerialPort) commPort;
serialPort.setSerialPortParams(9600,SerialPort.DATABITS_8,SerialPort.STOPBITS_1,SerialPort.PARITY_NONE);
```

La comunicación serial tiene una velocidad de 9600bps, 8 bits de datos, 1 bit de parada y ningún bit de paridad.

Para enviar información desde el computador hacia se utilizan los comandos mostrados en las siguientes líneas. La primera parte de este código se ejecuta al modificar el valor de setpoint de “humedad1”.

```
private void setpoint_humedad1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Enviar codigo 107   y   setpoint_humedad1.getText()
    try {Myout.write(107); Myout.write(Integer.parseInt(setpoint_humedad1.getText()));}
    catch (IOException e) { e.printStackTrace();}
}

private void setpoint_luz1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Enviar codigo 102   y   setpoint_luz1.getText()
    try {Myout.write(102); Myout.write(Integer.parseInt(setpoint_luz1.getText()));}
    catch (IOException e) { e.printStackTrace();}
}
```

Se envía el número 107 antes de enviar el valor de setpoint para que el microcontrolador entienda que ese dato corresponde al setpoint de “humedad1”. Cada valor tiene un número asignado como se muestra páginas atrás en la tabla 8.

El valor de setpoint se ingresa en un cuadro de texto, una vez que se presiona la tecla “enter” el valor es enviado hacia el microcontrolador.

Para leer datos provenientes del invernadero se utilizan la línea siguiente:

```
in_bytes = Myin.read(lectura_pic);
```

Luego de leer el buffer completo se procede a distribuir la información proveniente de los sensores en casilleros de texto. Estos casilleros independientes contienen por ejemplo el valor del sensor de nivel, el valor de la temperatura, etc.

```
valor_tanque.setText(String.valueOf(lectura[0]));
valor_temperatura.setText(String.valueOf(lectura[1]));
valor_luz1.setText(String.valueOf(lectura[2]));
valor_luz2.setText(String.valueOf(lectura[3]));
valor_luz3.setText(String.valueOf(lectura[4]));
valor_luz4.setText(String.valueOf(lectura[5]));
valor_luz5.setText(String.valueOf(lectura[6]));
valor_humedad1.setText(String.valueOf(lectura[7]));
valor_humedad2.setText(String.valueOf(lectura[8]));
```

En la tabla 7 se muestra el orden de la información en el buffer de entrada.

CAPITULO VI
Sensores y Actuadores

6.1. Sensor de temperatura

El sensor de temperatura utilizado es un LM35 de fácil instalación.

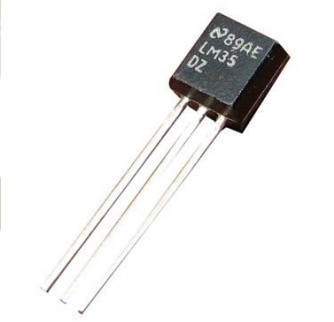


Figura 44. Sensor LM35

El sensor se alimenta y la señal de salida va directamente al microcontrolador, la variación de temperatura es de 10mv por cada grado centígrado.

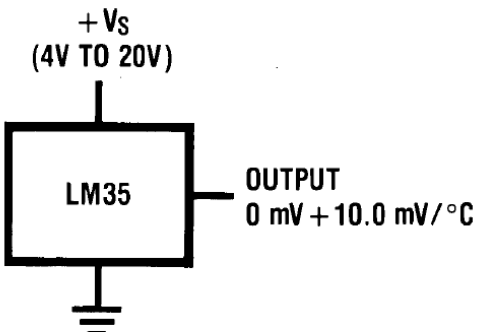


Figura 45. Esquema del sensor LM35

6.2. Calefactor

Como fuente de calor existen muchas opciones, hay las que funcionan con gas propano o con petróleo y las que son eléctricas, en cualquiera de los casos el funcionamiento es el mismo, el calefactor absorbe el aire del invernadero y lo calienta antes de expulsarlo (recirculación de aire).

Figura 46. Calefactor

6.3. Sensor de luz

Como sensores de luz se utilizan fotoresistencias (LDR). Añadiendo una resistencia en serie se crea un divisor de tensión y es este voltaje variable el que se entrega al microcontrolador.

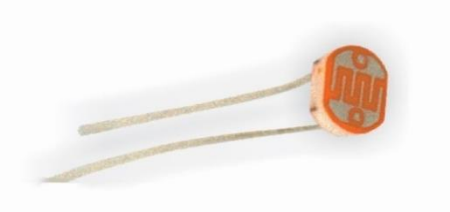


Figura 47. LDR

LDR viene del ingles “light dependent resistor”, los valores típicos de una fotoresistencia son:

Luz del día = 5000 Ω
Oscuridad = 20 MΩ

6.4. Fuente de luz

Para la iluminación se puede utilizar cualquier fuente de luz, pero en este proyecto se recomiendan los fluorescentes por ser de bajo consumo.

Figura 48. Fuente de luz

6.5. Sensor de Humedad

Para la humedad se sugiere el uso de los sensores CHS MSS de la marca TDK, es un sensor de tipo resistivo, es decir, varia su resistencia con el cambio de humedad.

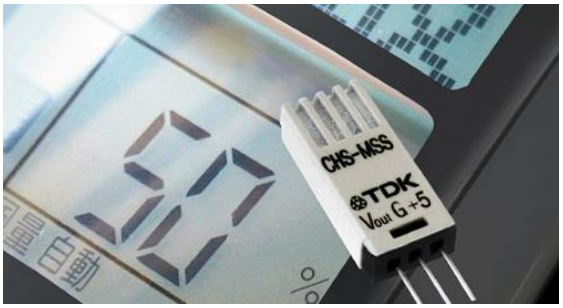


Figura 49. Sensor de humedad

Se debe alimentar el sensor y este empieza a entregar una señal analógica proporcional a la humedad detectada.

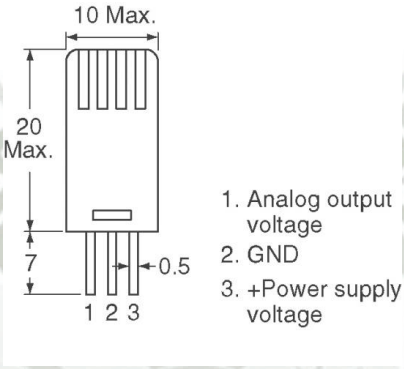


Figura 50. Esquema del sensor de humedad

6.6. Bomba de agua

Se puede utilizar cualquier tipo de bomba tanto para el llenado del tanque como para el riego, puede ser trifásica o no, pero en cualquier caso el microcontrolador gobernará un contactor que cumplirá el papel de interruptor de la bomba.

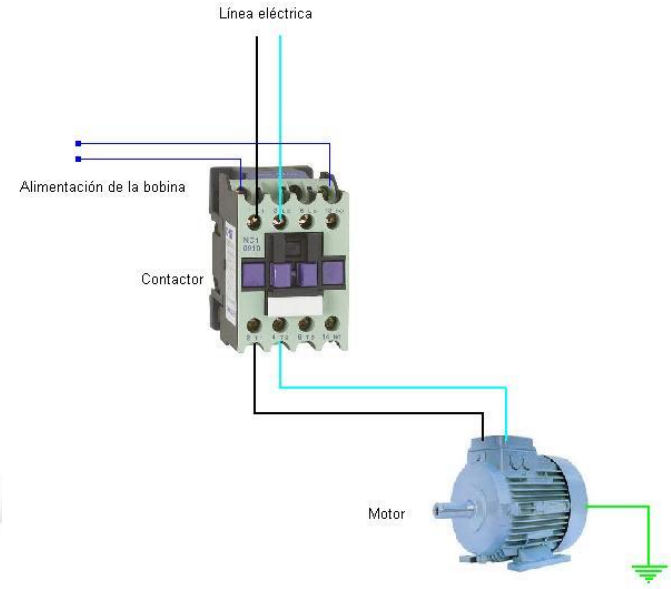


Figura 51. Esquema conexión de bomba

6.7. Electroválvulas

Para el riego se deben utilizar electroválvulas de 12v, el microcontrolador envía la señal de activación a un relay que alimentara la electroválvula.

Figura 52. Electroválvula

6.8. Sensor de nivel de agua

Una forma práctica de medir nivel de líquidos es usar sensores ultrasónicos. Se inyecta un pulso de aprox. 10us y se espera a que el sensor devuelva una señal, este tiempo es proporcional a la distancia recorrida por las ondas.

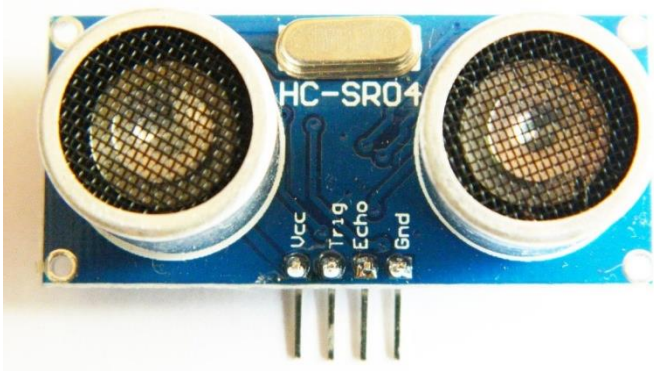


Figura 53. Sensor ultrasónico

CAPITULO VII
Análisis del Proyecto

7.1. Análisis de confiabilidad

La confiabilidad de un sistema es una función exponencial que usa una "tasa de fallas" compuesta (λ_{sis}) que es la suma de las tasas de fallas individuales de los componentes. El Factor de Intensidad de Fallas o Tasa de Fallas se expresa generalmente en FIT, donde:

$$1\ FIT = \frac{1\ falla}{10^9\ horas}$$

Otra medida de la confiabilidad de un componente o de un sistema es el Tiempo Medio Entre Fallas (MTBF), que es el tiempo medio que le toma fallar a un componente o a un sistema. Entonces:

$$MTBF = \frac{1}{\lambda}$$

El Factor de Intensidad de Fallos (λ) de un sistema está dado por la suma algebraica de los factores de intensidad de fallos de cada uno de los componentes:

$$\lambda_{sis} = \sum \lambda_{comp}$$

7.1.1. Tarjeta

Componente	Cantidad	Tasa de Falla (FIT) Total
Circuito impreso	1	1000
Microcontrolador	1	500
MAX 232	1	90
Convertidor Bluetooth	1	500
Transistores	6	600
Resistencias	30	150
Condensadores	11	33
Diodos	4	320
Reguladores de Voltaje	2	160
Relés	2	400
Borneras	6	60
Cristal oscilador	1	30
Pulsador	1	250
Conector DB9 (3 pines)	1	30
TOTAL		**4123**

Tabla 10. Confiabilidad de tarjeta electrónica

7.1.2. Panel de simulación de invernadero

Componente	Cantidad	Tasa de Falla (FIT) Total
LED´s	8	80
Potenciómetro	5	1250
Focos	2	1000
Motores	2	3000
Sensor de Temperatura	1	90
Sensor de Humedad	1	90
Sensor de Luz	1	90
TOTAL		**5600**

Tabla 11. Confiabilidad de panel de simulación

7.1.3. Panel Manual

Componente	Cantidad	Tasa de Falla (FIT) Total
Botones	6	1500
LED´s	6	60
TOTAL		**2100**

Tabla 12. Confiabilidad de panel manual

7.1.4. Computador

Componente	Cantidad	Tasa de Falla (FIT) Total
Computador	1	20
TOTAL		**20**

Tabla 13. Confiabilidad de computador

7.1.5. Total

Como se vio anteriormente:

$$\lambda_{sis} = \sum \lambda_{comp}$$

Entonces:

$$\lambda_{sis} = 4123 \cdot 10^{-9} + 5600 \cdot 10^{-9} + 2100 \cdot 10^{-9} + 20 \cdot 10^{-9}$$

$$\lambda_{sis} = 11843 \cdot 10^{-9}$$

Sabemos también que:

$$MTBF = \frac{1}{\lambda}$$

Entonces:

$$MTBF = \frac{1}{11843 \cdot 10^{-9}}$$

$$MTBF = 84438.06\ horas = 9.6\ años$$

Se concluye entonces que la probabilidad de falla del equipo es de una en 9.6 años.

7.2. Análisis Económico

7.2.1. Análisis de diseño

Componente	Cantidad	Costo S/.
Circuito impreso	1	30.00
Microcontrolador	1	35.00
MAX 232	1	3.00
Convertidor Bluetooth	1	50.00
Transistores	6	1.20
Resistencias	30	2.00
Condensadores	11	4.00
Diodos	4	2.00
Reguladores de Voltaje	2	5.00
Relés	2	7.00
Borneras	6	6.00
Cristal oscilador	1	2.50
Pulsador	1	0.50
Conector DB9 (3 pines)	1	2.00
LED´s	14	5.00
Potenciómetro	5	10.00
Focos	2	4.00
Motores	2	7.00
Sensor de Temperatura	1	14.00
Sensor de Humedad	1	10.00
Sensor de Luz	1	4.00
Botones	6	15.00
Computador	1	1100.00
Smartphone	1	400.00
Mano de obra (horas)	500	3500.00
TOTAL S/.		**5219.20**

Tabla 14. Análisis de costos de diseño

7.2.2. Análisis de comercialización

Para el análisis de comercialización debemos excluir algunos costos de la tabla anterior. Además de la mano de obra de implementación se cobrará un porcentaje de la mano de obra de "diseño" en cada unidad vendida. Y por último se debe considerar una utilidad.

Componente	Cantidad	Costo S/.
Circuito impreso	1	30.00
Microcontrolador	1	35.00
MAX 232	1	3.00
Convertidor Bluetooth	1	50.00
Transistores	6	1.20
Resistencias	30	2.00
Condensadores	11	4.00
Diodos	4	2.00
Reguladores de Voltaje	2	5.00
Relés	2	7.00
Borneras	6	6.00
Cristal oscilador	1	2.50
Pulsador	1	0.50
Conector DB9 (3 pines)	1	2.00
LED´s	14	5.00
Potenciómetro	5	10.00
Focos	2	4.00
Motores	2	7.00
Sensor de Temperatura	1	14.00
Sensor de Humedad	1	10.00
Sensor de Luz	1	4.00
Botones	6	15.00
Computador	1	--
Smartphone	1	--
Mano de obra diseño	20%	700.00
M.O implementación (horas)	20	140.00
Subtotal		1059.20
Utilidad	50%	529.60
TOTAL S/.		**1588.80**

Tabla 15. Análisis de costos de comercialización

Conclusiones

- Se diseñó e implementó un sistema en un dispositivo Android para la supervisión de un invernadero, llegando a la conclusión que es una buena alternativa eficiente y de bajo costo.

- El sistema es portátil, es decir, el operario puede movilizarse por todo el invernadero con el panel en sus manos. Puede leer los valores actuales del invernadero, ingresar valores de setpoint, así como activar elementos de forma manual desde el dispositivo Android.

- Se comprobó mediante el modelo a escala de un invernadero con un Smartphone que la comunicación Bluetooth es lo suficientemente robusta para ser utilizada en un sistema como este, y no solo para conectar teclados, ratones, hands free, etc.

- Se implementó una interface en Java para computadores con las mismas funciones de la aplicación móvil. Java además de ser software libre permite que sus desarrollos puedan ejecutarse en cualquier plataforma (Windows,

Mac, Linux, etc) sin la necesidad de modificar o diseñar nuevamente. Como ellos dicen: “Write once, run everywhere” (Escribe una vez, ejecuta donde sea).

- Se podría pensar que el software de pago está mejor desarrollado y tiene más capacidades que el software libre. Pero en este proyecto se demostró que el uso de software libre cumple completamente con las expectativas de diseño.

Recomendaciones

- Se recomienda integrar el proyecto en un invernadero real, para comparar los resultados con la simulación realizada en la presente tesis, complementando la función de gráficas al sistema, para la temperatura y la humedad para tener un registro del comportamiento de estas variables a lo largo del día.

- Se recomienda agregar a la aplicación en Android la capacidad de guardar datos, para tener un registro histórico de los eventos suscitados en el proceso.

- Posteriormente sería interesante poder aplicar tecnología wi-fi para implementar una aplicación similar o como una actualización de esta.

- Se debe tener en cuenta el uso de software libre en el diseño de futuros desarrollos ya que constituyen herramientas útiles, que cuentan con gran cantidad de información en internet, son de fácil acceso y preparadas para la creación de cualquier tipo de programa.

Bibliografía

Automatización y control de invernadero
Universidad Nacional de Quilmes
Leandro Padovani

Automatización de sistema de riego para el cultivo de flores tipo exportación
Pontificia Universidad Javeriana
Rodolfo Agudelo Sueñas

Sistema de automatización para una planta productora de jabón líquido
Universidad Pontificia Bolivariana
Andrés Mauricio Zapata Gallego

Diseño de una red inalámbrica de sensores para monitorear
Un cultivo de plátanos en el distrito de mala
Pontificia Universidad Católica Del Perú
Ernesto Alonso Antachoque Espinoza

Diseño de un sistema de automatizacion para el Sistema de iluminación de una planta industrial
Pontificia Universidad Católica Del Perú
Anguie María Del Milagro Contreras Iglesias

Diseño de un sistema de riego por goteo controlado y automatizado para uva italia
Pontificia Universidad Católica Del Perú
José Carlos Cruz Concha

Diseño de una ducha automatizada para personas con discapacidad en las extremidades superiores.
Pontificia Universidad Católica Del Perú
María Claudia Dejo Sánchez

Diseño de un sistema de monitoreo a distancia basado en Tecnología web para el proceso de tostado de granos de Kiwicha
Pontificia Universidad Católica Del Perú
Javier Lazarte Paredes

Eclipse
Herramienta para el desarrollo de aplicaciones de código libre
www.eclipse.org

Bluetooth
Conceptos básicos de la tecnología Bluetooth
http://es.wikipedia.org/wiki/Bluetooth

Bluetooth
Información de la página oficial
http://www.bluetooth.com/Pages/How-It-Works.aspx

Android SDK

Herramientas necesarias para programar en Android

https://developer.android.com/sdk

Anexos

PIC16F87X
Data Sheet

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

PIC16F87X

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

Devices Included in this Data Sheet:

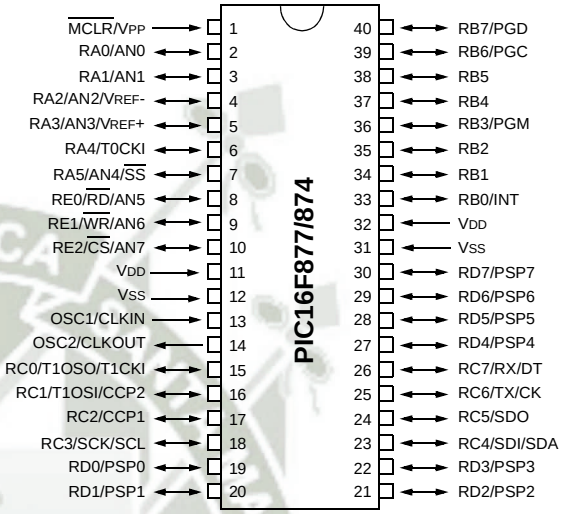
- PIC16F873
- PIC16F874
- PIC16F876
- PIC16F877

Microcontroller Core Features:

- High performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM)
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and
  Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial, Industrial and Extended temperature ranges
- Low-power consumption:
  - < 0.6 mA typical @ 3V, 4 MHz
  - 20 µA typical @ 3V, 32 kHz
  - < 1 µA typical standby current

Pin Diagram

PDIP



Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I^2C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

10.0 ADDRESSABLE UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (USART)

The Universal Synchronous Asynchronous Receiver Transmitter (USART) module is one of the two serial I/O modules. (USART is also known as a Serial Communications Interface or SCI.) The USART can be configured as a full duplex asynchronous system that can communicate with peripheral devices such as CRT terminals and personal computers, or it can be configured as a half duplex synchronous system that can communicate with peripheral devices such as A/D or D/A integrated circuits, serial EEPROMs etc.

The USART can be configured in the following modes:

- Asynchronous (full duplex)
- Synchronous - Master (half duplex)
- Synchronous - Slave (half duplex)

Bit SPEN (RCSTA<7>) and bits TRISC<7:6> have to be set in order to configure pins RC6/TX/CK and RC7/RX/DT as the Universal Synchronous Asynchronous Receiver Transmitter.

The USART module also has a multi-processor communication capability using 9-bit address detection.

REGISTER 10-1: TXSTA: TRANSMIT STATUS AND CONTROL REGISTER (ADDRESS 98h)

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

bit 7 **CSRC:** Clock Source Select bit
Asynchronous mode:
Don't care
Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

bit 6 **TX9**: 9-bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN**: Transmit Enable bit
1 = Transmit enabled
0 = Transmit disabled

Note: SREN/CREN overrides TXEN in SYNC mode.

bit 4 **SYNC**: USART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3 **Unimplemented:** Read as '0'

bit 2 **BRGH**: High Baud Rate Select bit
Asynchronous mode:
1 = High speed
0 = Low speed
Synchronous mode:
Unused in this mode

bit 1 **TRMT**: Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D:** 9th bit of Transmit Data, can be parity bit

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 10-2: RCSTA: RECEIVE STATUS AND CONTROL REGISTER (ADDRESS 18h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

bit 7 **SPEN:** Serial Port Enable bit
1 = Serial port enabled (configures RC7/RX/DT and RC6/TX/CK pins as serial port pins)
0 = Serial port disabled

bit 6 **RX9**: 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN**: Single Receive Enable bit
Asynchronous mode:
Don't care
Synchronous mode - master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode - slave:
Don't care

bit 4 **CREN**: Continuous Receive Enable bit
Asynchronous mode:
1 = Enables continuous receive
0 = Disables continuous receive
Synchronous mode:
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN:** Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enables interrupt and load of the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received, and ninth bit can be used as parity bit

bit 2 **FERR**: Framing Error bit
1 = Framing error (can be updated by reading RCREG register and receive next valid byte)
0 = No framing error

bit 1 **OERR**: Overrun Error bit
1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D:** 9th bit of Received Data (can be parity bit, but must be calculated by user firmware)

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

10.1 USART Baud Rate Generator (BRG)

The BRG supports both the Asynchronous and Synchronous modes of the USART. It is a dedicated 8-bit baud rate generator. The SPBRG register controls the period of a free running 8-bit timer. In Asynchronous mode, bit BRGH (TXSTA<2>) also controls the baud rate. In Synchronous mode, bit BRGH is ignored. Table 10-1 shows the formula for computation of the baud rate for different USART modes which only apply in Master mode (internal clock).

Given the desired baud rate and FOSC, the nearest integer value for the SPBRG register can be calculated using the formula in Table 10-1. From this, the error in baud rate can be determined.

It may be advantageous to use the high baud rate (BRGH = 1), even for slower baud clocks. This is because the FOSC/(16(X + 1)) equation can reduce the baud rate error in some cases.

Writing a new value to the SPBRG register causes the BRG timer to be reset (or cleared). This ensures the BRG does not wait for a timer overflow before outputting the new baud rate.

10.1.1 SAMPLING

The data on the RC7/RX/DT pin is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RX pin.

TABLE 10-1: BAUD RATE FORMULA

SYNC	BRGH = 0 (Low Speed)	BRGH = 1 (High Speed)
0	(Asynchronous) Baud Rate = FOSC/(64(X+1))	Baud Rate = FOSC/(16(X+1))
1	(Synchronous) Baud Rate = FOSC/(4(X+1))	N/A

X = value in SPBRG (0 to 255)

TABLE 10-2: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented, read as '0'. Shaded cells are not used by the BRG.

TABLE 10-3: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 0)

BAUD RATE (K)	Fosc = 20 MHz			Fosc = 16 MHz			Fosc = 10 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-	-	-	-	-	-	-
1.2	1.221	1.75	255	1.202	0.17	207	1.202	0.17	129
2.4	2.404	0.17	129	2.404	0.17	103	2.404	0.17	64
9.6	9.766	1.73	31	9.615	0.16	25	9.766	1.73	15
19.2	19.531	1.72	15	19.231	0.16	12	19.531	1.72	7
28.8	31.250	8.51	9	27.778	3.55	8	31.250	8.51	4
33.6	34.722	3.34	8	35.714	6.29	6	31.250	6.99	4
57.6	62.500	8.51	4	62.500	8.51	3	52.083	9.58	2
HIGH	1.221	-	255	0.977	-	255	0.610	-	255
LOW	312.500	-	0	250.000	-	0	156.250	-	0

BAUD RATE (K)	Fosc = 4 MHz			Fosc = 3.6864 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	0.300	0	207	0.3	0	191
1.2	1.202	0.17	51	1.2	0	47
2.4	2.404	0.17	25	2.4	0	23
9.6	8.929	6.99	6	9.6	0	5
19.2	20.833	8.51	2	19.2	0	2
28.8	31.250	8.51	1	28.8	0	1
33.6	-	-	-	-	-	-
57.6	62.500	8.51	0	57.6	0	0
HIGH	0.244	-	255	0.225	-	255
LOW	62.500	-	0	57.6	-	0

TABLE 10-4: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 1)

BAUD RATE (K)	Fosc = 20 MHz			Fosc = 16 MHz			Fosc = 10 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-	-	-	-	-	-	-
1.2	-	-	-	-	-	-	-	-	-
2.4	-	-	-	-	-	-	2.441	1.71	255
9.6	9.615	0.16	129	9.615	0.16	103	9.615	0.16	64
19.2	19.231	0.16	64	19.231	0.16	51	19.531	1.72	31
28.8	29.070	0.94	42	29.412	2.13	33	28.409	1.36	21
33.6	33.784	0.55	36	33.333	0.79	29	32.895	2.10	18
57.6	59.524	3.34	20	58.824	2.13	16	56.818	1.36	10
HIGH	4.883	-	255	3.906	-	255	2.441	-	255
LOW	1250.000	-	0	1000.000		0	625.000	-	0

BAUD RATE (K)	Fosc = 4 MHz			Fosc = 3.6864 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-	-	-	-
1.2	1.202	0.17	207	1.2	0	191
2.4	2.404	0.17	103	2.4	0	95
9.6	9.615	0.16	25	9.6	0	23
19.2	19.231	0.16	12	19.2	0	11
28.8	27.798	3.55	8	28.8	0	7
33.6	35.714	6.29	6	32.9	2.04	6
57.6	62.500	8.51	3	57.6	0	3
HIGH	0.977	-	255	0.9	-	255
LOW	250.000	-	0	230.4	-	0

10.2 USART Asynchronous Mode

In this mode, the USART uses standard non-return-to-zero (NRZ) format (one START bit, eight or nine data bits, and one STOP bit). The most common data format is 8-bits. An on-chip, dedicated, 8-bit baud rate generator can be used to derive standard baud rate frequencies from the oscillator. The USART transmits and receives the LSb first. The transmitter and receiver are functionally independent, but use the same data format and baud rate. The baud rate generator produces a clock, either x16 or x64 of the bit shift rate, depending on bit BRGH (TXSTA<2>). Parity is not supported by the hardware, but can be implemented in software (and stored as the ninth data bit). Asynchronous mode is stopped during SLEEP.

Asynchronous mode is selected by clearing bit SYNC (TXSTA<4>).

The USART Asynchronous module consists of the following important elements:

- Baud Rate Generator
- Sampling Circuit
- Asynchronous Transmitter
- Asynchronous Receiver

10.2.1 USART ASYNCHRONOUS TRANSMITTER

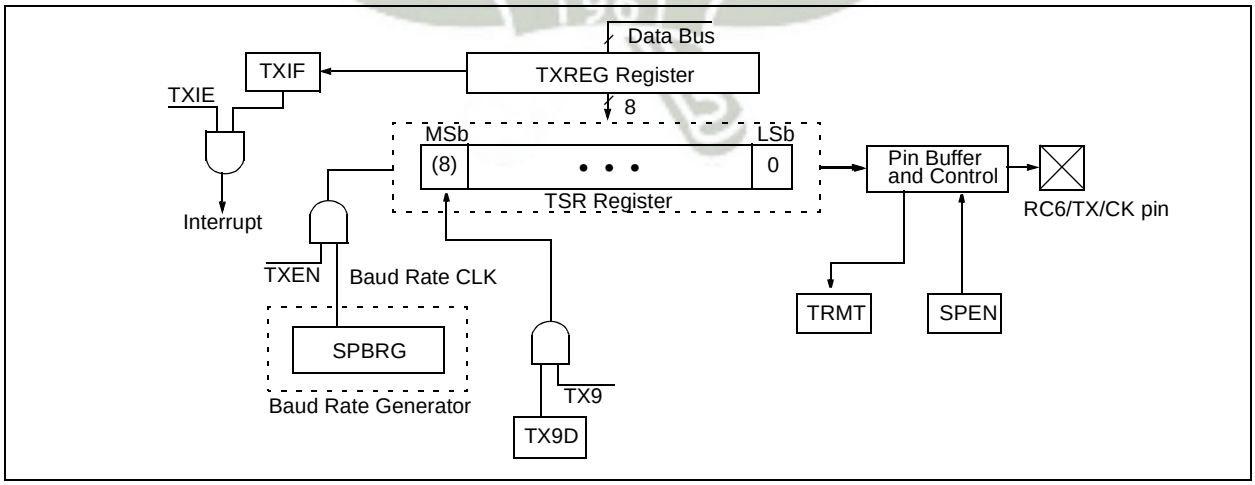
The USART transmitter block diagram is shown in Figure 10-1. The heart of the transmitter is the transmit (serial) shift register (TSR). The shift register obtains its data from the read/write transmit buffer, TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until the STOP bit has been transmitted from the previous load. As soon as the STOP bit is transmitted, the TSR is loaded with new data from the TXREG register (if available). Once the TXREG register transfers the data to the TSR register (occurs in one TCY), the TXREG register is empty and flag bit TXIF (PIR1<4>) is set. This interrupt can be enabled/disabled by setting/clearing enable bit TXIE (PIE1<4>). Flag bit TXIF will be set, regardless of the state of enable bit TXIE and cannot be cleared in software. It will reset only when new data is loaded into the TXREG register. While flag bit TXIF indicates the status of the TXREG register, another bit TRMT (TXSTA<1>) shows the status of the TSR register. Status bit TRMT is a read only bit, which is set when the TSR register is empty. No interrupt logic is tied to this bit, so the user has to poll this bit in order to determine if the TSR register is empty.

Note 1:	The TSR register is not mapped in data memory, so it is not available to the user.
2:	Flag bit TXIF is set when enable bit TXEN is set. TXIF is cleared by loading TXREG.

Transmission is enabled by setting enable bit TXEN (TXSTA<5>). The actual transmission will not occur until the TXREG register has been loaded with data and the baud rate generator (BRG) has produced a shift clock (Figure 10-2). The transmission can also be started by first loading the TXREG register and then setting enable bit TXEN. Normally, when transmission is first started, the TSR register is empty. At that point, transfer to the TXREG register will result in an immediate transfer to TSR, resulting in an empty TXREG. A back-to-back transfer is thus possible (Figure 10-3). Clearing enable bit TXEN during a transmission will cause the transmission to be aborted and will reset the transmitter. As a result, the RC6/TX/CK pin will revert to hi-impedance.

In order to select 9-bit transmission, transmit bit TX9 (TXSTA<6>) should be set and the ninth bit should be written to TX9D (TXSTA<0>). The ninth bit must be written before writing the 8-bit data to the TXREG register. This is because a data write to the TXREG register can result in an immediate transfer of the data to the TSR register (if the TSR is empty). In such a case, an incorrect ninth data bit may be loaded in the TSR register.

FIGURE 10-1: USART TRANSMIT BLOCK DIAGRAM



When setting up an Asynchronous Transmission, follow these steps:

1. Initialize the SPBRG register for the appropriate baud rate. If a high speed baud rate is desired, set bit BRGH (Section 10.1).
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, then set enable bit TXIE.
4. If 9-bit transmission is desired, then set transmit bit TX9.
5. Enable the transmission by setting bit TXEN, which will also set bit TXIF.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Load data to the TXREG register (starts transmission).
8. If using interrupts, ensure that GIE and PEIE (bits 7 and 6) of the INTCON register are set.

FIGURE 10-2: ASYNCHRONOUS MASTER TRANSMISSION

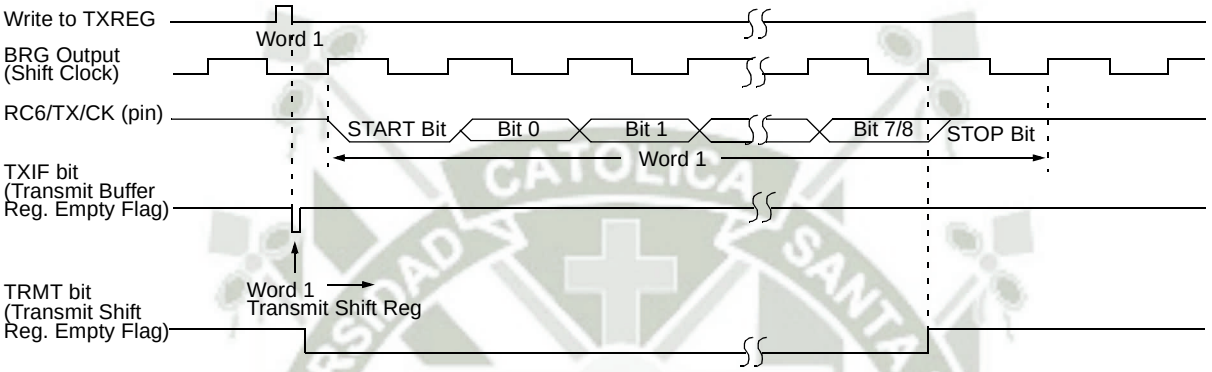
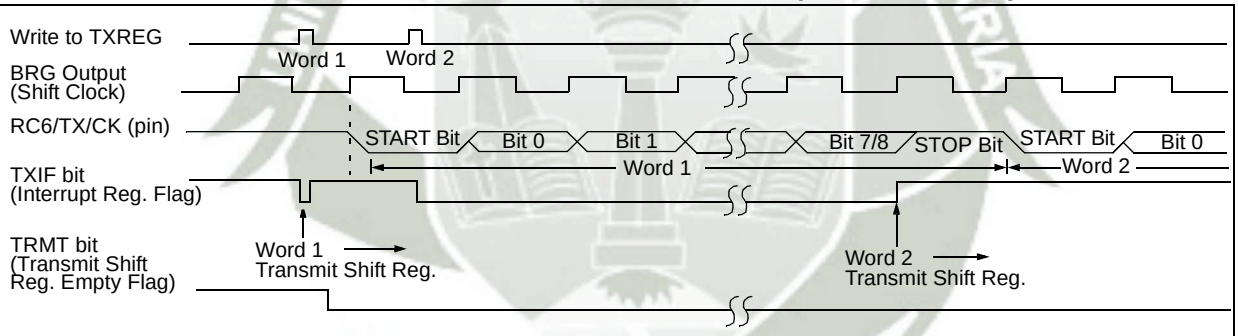


FIGURE 10-3: ASYNCHRONOUS MASTER TRANSMISSION (BACK TO BACK)



Note: This timing diagram shows two consecutive transmissions.

TABLE 10-5: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
0Bh, 8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	R0IF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
19h	TXREG	USART Transmit Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented locations read as '0'. Shaded cells are not used for asynchronous transmission.
Note 1: Bits PSPIE and PSPIF are reserved on the PIC16F873/876; always maintain these bits clear.

10.2.2 USART ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in Figure 10-4. The data is received on the RC7/RX/DT pin and drives the data recovery block. The data recovery block is actually a high speed shifter, operating at x16 times the baud rate; whereas, the main receive serial shifter operates at the bit rate or at FOSC.

Once Asynchronous mode is selected, reception is enabled by setting bit CREN (RCSTA<4>).

The heart of the receiver is the receive (serial) shift register (RSR). After sampling the STOP bit, the received data in the RSR is transferred to the RCREG register (if it is empty). If the transfer is complete, flag bit RCIF (PIR1<5>) is set. The actual interrupt can be enabled/disabled by setting/clearing enable bit RCIE (PIE1<5>). Flag bit RCIF is a read only bit, which is cleared by the hardware. It is cleared when the RCREG register has been read and is empty. The RCREG is a double buffered register (i.e., it is a two deep FIFO). It is possible for two bytes of data to be received and transferred to the RCREG FIFO and a third byte to begin shifting to the RSR register. On the detection of the STOP bit of the third byte, if the RCREG register is still full, the overrun error bit OERR (RCSTA<1>) will be set. The word in the RSR will be lost. The RCREG register can be read twice to retrieve the two bytes in the FIFO. Overrun bit OERR has to be cleared in software. This is done by resetting the receive logic (CREN is cleared and then set). If bit OERR is set, transfers from the RSR register to the RCREG register are inhibited, and no further data will be received. It is therefore, essential to clear error bit OERR if it is set. Framing error bit FERR (RCSTA<2>) is set if a STOP bit is detected as clear. Bit FERR and the 9th receive bit are buffered the same way as the receive data. Reading the RCREG will load bits RX9D and FERR with new values, therefore, it is essential for the user to read the RCSTA register before reading the RCREG register in order not to lose the old FERR and RX9D information.

FIGURE 10-4: USART RECEIVE BLOCK DIAGRAM

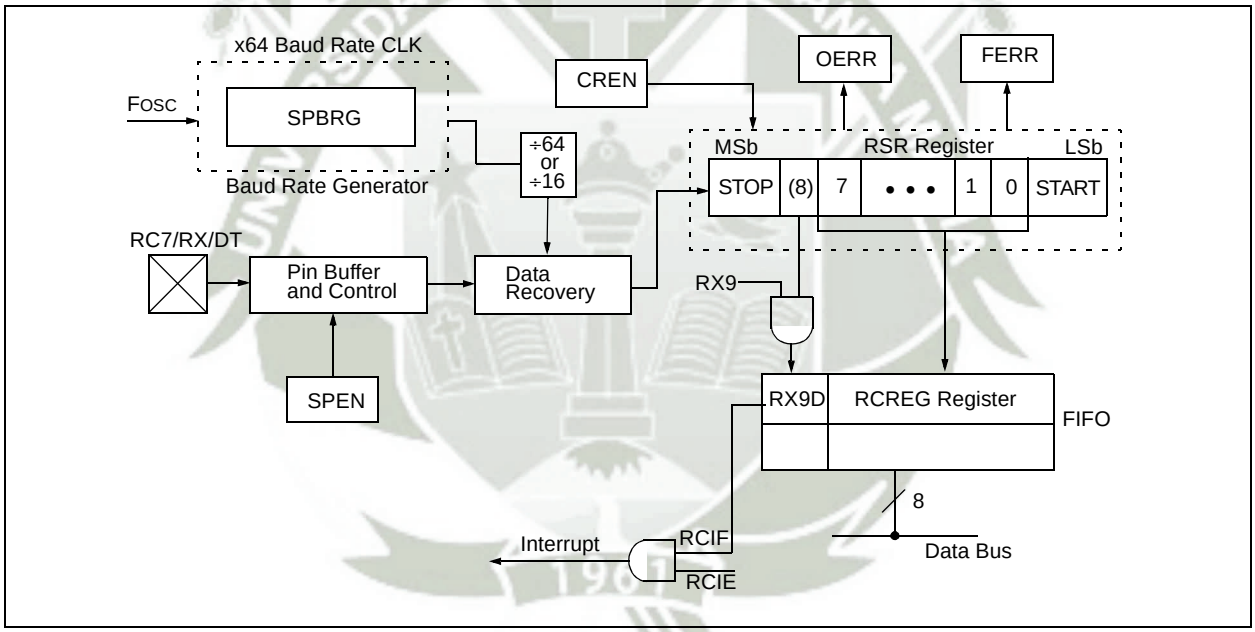
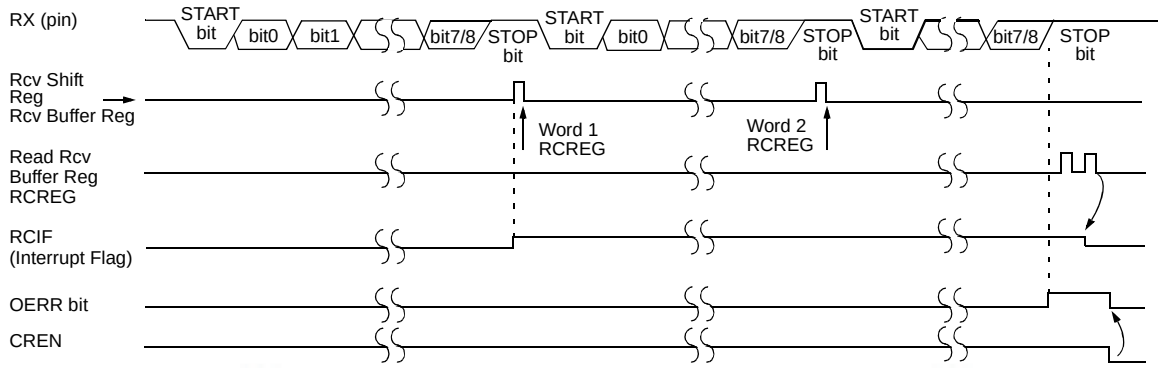


FIGURE 10-5: ASYNCHRONOUS RECEPTION



Note: This timing diagram shows three words appearing on the RX input. The RCREG (receive buffer) is read after the third word, causing the OERR (overrun) bit to be set.

When setting up an Asynchronous Reception, follow these steps:

1. Initialize the SPBRG register for the appropriate baud rate. If a high speed baud rate is desired, set bit BRGH (Section 10.1).
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, then set enable bit RCIE.
4. If 9-bit reception is desired, then set bit RX9.
5. Enable the reception by setting bit CREN.
6. Flag bit RCIF will be set when reception is complete and an interrupt will be generated if enable bit RCIE is set.
7. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
8. Read the 8-bit received data by reading the RCREG register.
9. If any error occurred, clear the error by clearing enable bit CREN.
10. If using interrupts, ensure that GIE and PEIE (bits 7 and 6) of the INTCON register are set.

TABLE 10-6: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
0Bh, 8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	R0IF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
1Ah	RCREG	USART Receive Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented locations read as '0'. Shaded cells are not used for asynchronous reception.
Note 1: Bits PSPIE and PSPIF are reserved on PIC16F873/876 devices; always maintain these bits clear.

11.0 ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The Analog-to-Digital (A/D) Converter module has five inputs for the 28-pin devices and eight for the other devices.

The analog input charges a sample and hold capacitor. The output of the sample and hold capacitor is the input into the converter. The converter then generates a digital result of this analog level via successive approximation. The A/D conversion of the analog input signal results in a corresponding 10-bit digital number. The A/D module has high and low voltage reference input that is software selectable to some combination of VDD, VSS, RA2, or RA3.

The A/D converter has a unique feature of being able to operate while the device is in SLEEP mode. To operate in SLEEP, the A/D clock must be derived from the A/D's internal RC oscillator.

The A/D module has four registers. These registers are:

- A/D Result High Register (ADRESH)
- A/D Result Low Register (ADRESL)
- A/D Control Register0 (ADCON0)
- A/D Control Register1 (ADCON1)

The ADCON0 register, shown in Register 11-1, controls the operation of the A/D module. The ADCON1 register, shown in Register 11-2, configures the functions of the port pins. The port pins can be configured as analog inputs (RA3 can also be the voltage reference), or as digital I/O.

Additional information on using the A/D module can be found in the PICmicro™ Mid-Range MCU Family Reference Manual (DS33023).

REGISTER 11-1: ADCON0 REGISTER (ADDRESS: 1Fh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON
bit 7							bit 0

bit 7-6 **ADCS1:ADCS0:** A/D Conversion Clock Select bits
00 = FOSC/2
01 = FOSC/8
10 = FOSC/32
11 = FRC (clock derived from the internal A/D module RC oscillator)

bit 5-3 **CHS2:CHS0**: Analog Channel Select bits
000 = channel 0, (RA0/AN0)
001 = channel 1, (RA1/AN1)
010 = channel 2, (RA2/AN2)
011 = channel 3, (RA3/AN3)
100 = channel 4, (RA5/AN4)
101 = channel 5, (RE0/AN5)**(1)**
110 = channel 6, (RE1/AN6)**(1)**
111 = channel 7, (RE2/AN7)**(1)**

bit 2 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit
If ADON = 1:
1 = A/D conversion in progress (setting this bit starts the A/D conversion)
0 = A/D conversion not in progress (this bit is automatically cleared by hardware when the A/D conversion is complete)

bit 1 **Unimplemented**: Read as '0'

bit 0 **ADON**: A/D On bit
1 = A/D converter module is operating
0 = A/D converter module is shut-off and consumes no operating current

Note 1: These channels are not available on PIC16F873/876 devices.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 11-2: ADCON1 REGISTER (ADDRESS 9Fh)

U-0	U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 7 **ADFM:** A/D Result Format Select bit
1 = Right justified. 6 Most Significant bits of ADRESH are read as '0'.
0 = Left justified. 6 Least Significant bits of ADRESL are read as '0'.

bit 6-4 **Unimplemented:** Read as '0'

bit 3-0 **PCFG3:PCFG0**: A/D Port Configuration Control bits:

PCFG3: PCFG0	AN7(1) RE2	AN6(1) RE1	AN5(1) RE0	AN4 RA5	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0	VREF+	VREF-	CHAN/ Refs(2)
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	RA3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	RA3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	RA3	VSS	2/1
011x	D	D	D	D	D	D	D	D	VDD	VSS	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	RA3	RA2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	RA3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	RA3	RA2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	RA3	RA2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	RA3	RA2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	RA3	RA2	1/2

A = Analog input D = Digital I/O

Note 1: These channels are not available on PIC16F873/876 devices.
2: This column indicates the number of analog channels available as A/D inputs and the number of analog channels used as voltage reference inputs.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

The ADRESH:ADRESL registers contain the 10-bit result of the A/D conversion. When the A/D conversion is complete, the result is loaded into this A/D result register pair, the GO/DONE bit (ADCON0<2>) is cleared and the A/D interrupt flag bit ADIF is set. The block diagram of the A/D module is shown in Figure 11-1.

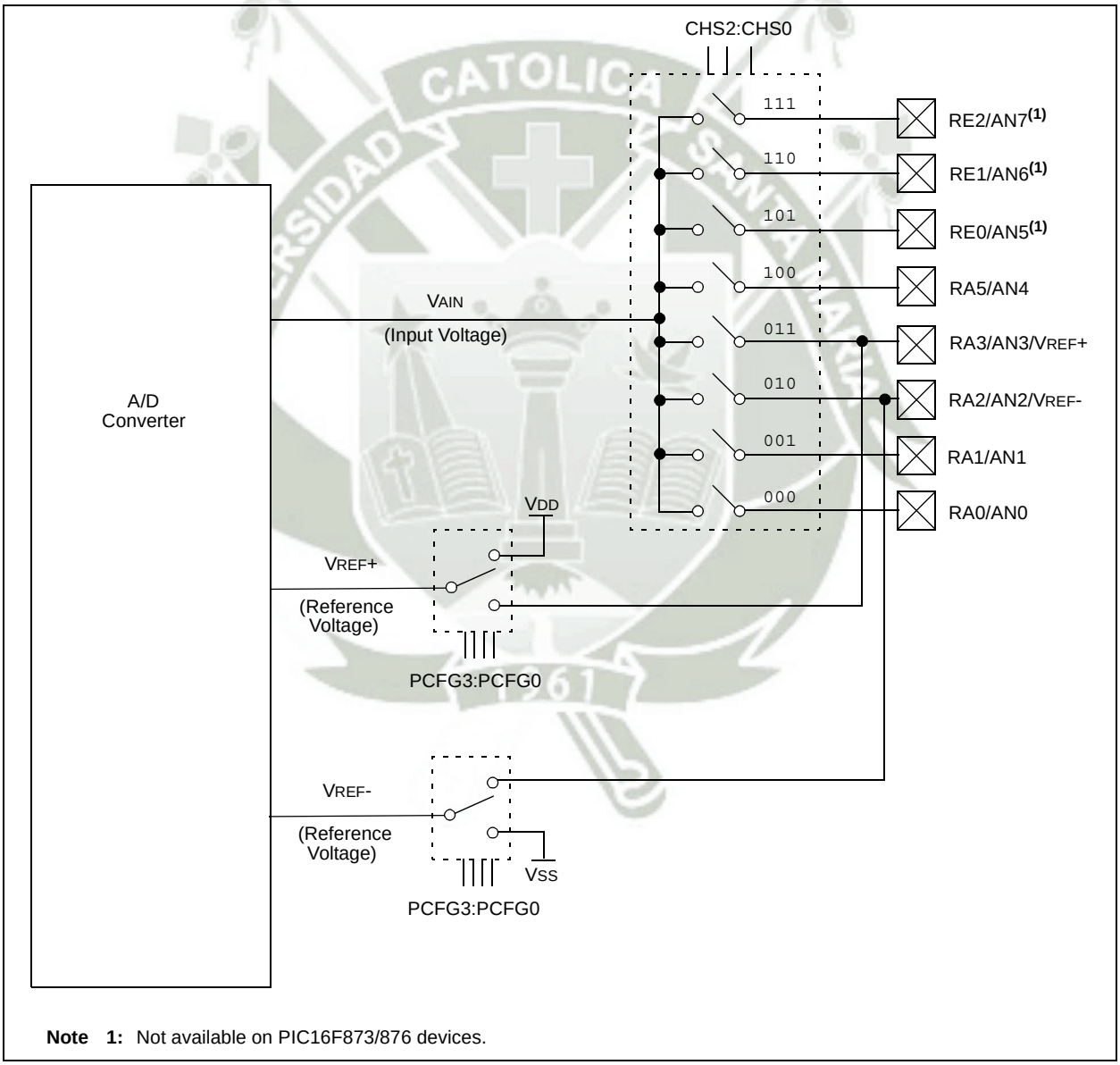
After the A/D module has been configured as desired, the selected channel must be acquired before the conversion is started. The analog input channels must have their corresponding TRIS bits selected as inputs.

To determine sample time, see Section 11.1. After this acquisition time has elapsed, the A/D conversion can be started.

These steps should be followed for doing an A/D Conversion:

1. Configure the A/D module:
   - Configure analog pins/voltage reference and digital I/O (ADCON1)
   - Select A/D input channel (ADCON0)
   - Select A/D conversion clock (ADCON0)
   - Turn on A/D module (ADCON0)
2. Configure A/D interrupt (if desired):
   - Clear ADIF bit
   - Set ADIE bit
   - Set PEIE bit
   - Set GIE bit
3. Wait the required acquisition time.
4. Start conversion:
   - Set GO/$\overline{\text{DONE}}$ bit (ADCON0)
5. Wait for A/D conversion to complete, by either:
   - Polling for the GO/$\overline{\text{DONE}}$ bit to be cleared (with interrupts enabled); OR
   - Waiting for the A/D interrupt
6. Read A/D result register pair (ADRESH:ADRESL), clear bit ADIF if required.
7. For the next conversion, go to step 1 or step 2, as required. The A/D conversion time per bit is defined as TAD. A minimum wait of 2TAD is required before the next acquisition starts.

FIGURE 11-1: A/D BLOCK DIAGRAM



Note 1: Not available on PIC16F873/876 devices.

11.1 A/D Acquisition Requirements

For the A/D converter to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the input channel voltage level. The analog input model is shown in Figure 11-2. The source impedance (RS) and the internal sampling switch (RSS) impedance directly affect the time required to charge the capacitor CHOLD. The sampling switch (RSS) impedance varies over the device voltage (VDD), see Figure 11-2. **The maximum recommended impedance for analog sources is 10 kΩ.** As the impedance is decreased, the acquisition time may be decreased. After the analog input channel is selected (changed), this acquisition must be done before the conversion can be started.

To calculate the minimum acquisition time, Equation 11-1 may be used. This equation assumes that 1/2 LSb error is used (1024 steps for the A/D). The 1/2 LSb error is the maximum error allowed for the A/D to meet its specified resolution.

To calculate the minimum acquisition time, TACQ, see the PICmicro™ Mid-Range Reference Manual (DS33023).

EQUATION 11-1: ACQUISITION TIME

TACQ = Amplifier Settling Time + Hold Capacitor Charging Time + Temperature Coefficient

= TAMP + TC + TCOFF
= 2µs + TC + [(Temperature -25°C)(0.05µs/°C)]
TC = CHOLD (RIC + RSS + RS) In(1/2047)
= - 120pF (1kΩ + 7kΩ + 10kΩ) In(0.0004885)
= 16.47µs
TACQ = 2µs + 16.47µs + [(50°C -25°C)(0.05µs/°C)
= 19.72µs

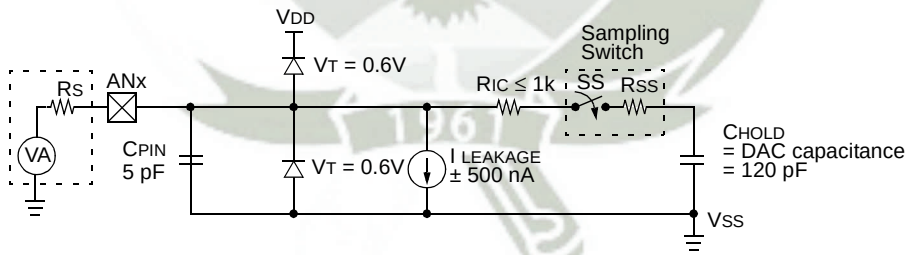
Note 1: The reference voltage (VREF) has no effect on the equation, since it cancels itself out.

2: The charge holding capacitor (CHOLD) is not discharged after each conversion.

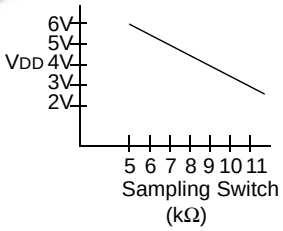
3: The maximum recommended impedance for analog sources is 10 kΩ. This is required to meet the pin leakage specification.

4: After a conversion has completed, a 2.0TAD delay must complete before acquisition can begin again. During this time, the holding capacitor is not connected to the selected A/D input channel.

FIGURE 11-2: ANALOG INPUT MODEL



Legend
CPIN = input capacitance
VT = threshold voltage
I LEAKAGE = leakage current at the pin due to various junctions
RIC = interconnect resistance
SS = sampling switch
CHOLD = sample/hold capacitance (from DAC)



11.2 Selecting the A/D Conversion Clock

The A/D conversion time per bit is defined as TAD. The A/D conversion requires a minimum 12TAD per 10-bit conversion. The source of the A/D conversion clock is software selected. The four possible options for TAD are:

- 2TOSC
- 8TOSC
- 32TOSC
- Internal A/D module RC oscillator (2-6 µs)

For correct A/D conversions, the A/D conversion clock (TAD) must be selected to ensure a minimum TAD time of 1.6 µs.

Table 11-1 shows the resultant TAD times derived from the device operating frequencies and the A/D clock source selected.

TABLE 11-1: TAD vs. MAXIMUM DEVICE OPERATING FREQUENCIES (STANDARD DEVICES (C))

AD Clock Source (TAD)		Maximum Device Frequency
Operation	**ADCS1:ADCS0**	**Max.**
2TOSC	00	1.25 MHz
8TOSC	01	5 MHz
32TOSC	10	20 MHz
RC(1, 2, 3)	11	**(Note 1)**

Note 1: The RC source has a typical TAD time of 4 µs, but can vary between 2-6 µs.
2: When the device frequencies are greater than 1 MHz, the RC A/D conversion clock source is only recommended for SLEEP operation.
3: For extended voltage devices (LC), please refer to the Electrical Characteristics (Sections 15.1 and 15.2).

11.3 Configuring Analog Port Pins

The ADCON1 and TRIS registers control the operation of the A/D port pins. The port pins that are desired as analog inputs must have their corresponding TRIS bits set (input). If the TRIS bit is cleared (output), the digital output level (VOH or VOL) will be converted.

The A/D operation is independent of the state of the CHS2:CHS0 bits and the TRIS bits.

Note 1: When reading the port register, any pin configured as an analog input channel will read as cleared (a low level). Pins configured as digital inputs will convert an analog input. Analog levels on a digitally configured input will not affect the conversion accuracy.

2: Analog levels on any pin that is defined as a digital input (including the AN7:AN0 pins), may cause the input buffer to consume current that is out of the device specifications.

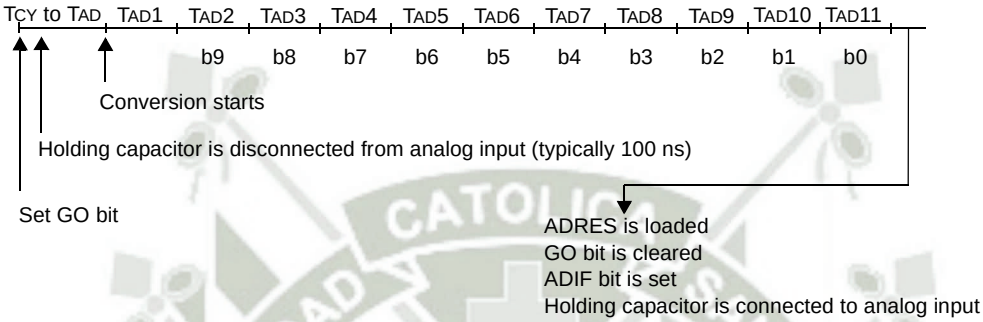
11.4 A/D Conversions

Clearing the GO/$\overline{\text{DONE}}$ bit during a conversion will abort the current conversion. The A/D result register pair will NOT be updated with the partially completed A/D conversion sample. That is, the ADRESH:ADRESL registers will continue to contain the value of the last completed conversion (or the last value written to the ADRESH:ADRESL registers). After the A/D conversion is aborted, a 2TAD wait is required before the next acquisition is started. After this 2TAD wait, acquisition on the selected channel is automatically started. The GO/$\overline{\text{DONE}}$ bit can then be set to start the conversion.

In Figure 11-3, after the GO bit is set, the first time segment has a minimum of TCY and a maximum of TAD.

Note:	The GO/$\overline{\text{DONE}}$ bit should **NOT** be set in the same instruction that turns on the A/D.

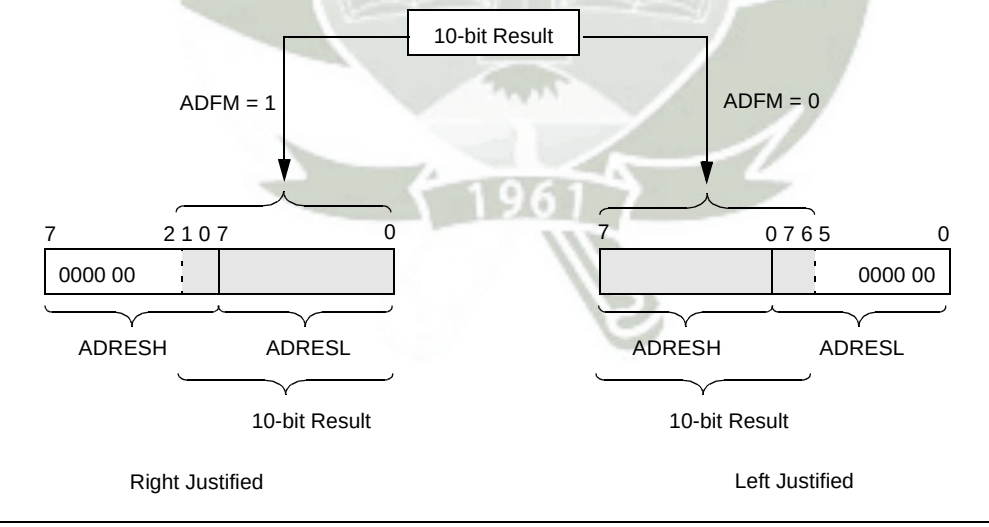
FIGURE 11-3: A/D CONVERSION TAD CYCLES



11.4.1 A/D RESULT REGISTERS

The ADRESH:ADRESL register pair is the location where the 10-bit A/D result is loaded at the completion of the A/D conversion. This register pair is 16-bits wide. The A/D module gives the flexibility to left or right justify the 10-bit result in the 16-bit result register. The A/D Format Select bit (ADFM) controls this justification. Figure 11-4 shows the operation of the A/D result justification. The extra bits are loaded with '0's'. When an A/D result will not overwrite these locations (A/D disable), these registers may be used as two general purpose 8-bit registers.

FIGURE 11-4: A/D RESULT JUSTIFICATION



11.5 A/D Operation During SLEEP

The A/D module can operate during SLEEP mode. This requires that the A/D clock source be set to RC (ADCS1:ADCS0 = 11). When the RC clock source is selected, the A/D module waits one instruction cycle before starting the conversion. This allows the SLEEP instruction to be executed, which eliminates all digital switching noise from the conversion. When the conversion is completed, the GO/$\overline{\text{DONE}}$ bit will be cleared and the result loaded into the ADRES register. If the A/D interrupt is enabled, the device will wake-up from SLEEP. If the A/D interrupt is not enabled, the A/D module will then be turned off, although the ADON bit will remain set.

When the A/D clock source is another clock option (not RC), a SLEEP instruction will cause the present conversion to be aborted and the A/D module to be turned off, though the ADON bit will remain set.

Turning off the A/D places the A/D module in its lowest current consumption state.

Note:	For the A/D module to operate in SLEEP, the A/D clock source must be set to RC (ADCS1:ADCS0 = 11). To allow the conversion to occur during SLEEP, ensure the SLEEP instruction immediately follows the instruction that sets the GO/$\overline{\text{DONE}}$ bit.

11.6 Effects of a RESET

A device RESET forces all registers to their RESET state. This forces the A/D module to be turned off, and any conversion is aborted. All A/D input pins are configured as analog inputs.

The value that is in the ADRESH:ADRESL registers is not modified for a Power-on Reset. The ADRESH:ADRESL registers will contain unknown data after a Power-on Reset.

TABLE 11-2: REGISTERS/BITS ASSOCIATED WITH A/D

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on $\overline{\text{MCLR}}$, WDT
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
1Eh	ADRESH	A/D Result Register High Byte								xxxx xxxx	uuuu uuuu
9Eh	ADRESL	A/D Result Register Low Byte								xxxx xxxx	uuuu uuuu
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON	0000 00-0	0000 00-0
9Fh	ADCON1	ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0	--0- 0000	--0- 0000
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	--11 1111
05h	PORTA	—	—	PORTA Data Latch when written: PORTA pins when read						--0x 0000	--0u 0000
89h(1)	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction bits			0000 -111	0000 -111
09h(1)	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	---- -uuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used for A/D conversion.

Note 1: These registers/bits are not available on the 28-pin devices.

LM35

LM35 Precision Centigrade Temperature Sensors

Literature Number: SNIS159B

National Semiconductor

LM35
Precision Centigrade Temperature Sensors

General Description

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in ˚ Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of ±¼˚C at room temperature and ±¾˚C over a full −55 to +150˚C temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1˚C in still air. The LM35 is rated to operate over a −55˚ to +150˚C temperature range, while the LM35C is rated for a −40˚ to +110˚C range (−10˚ with improved accuracy). The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface mount small outline package and a plastic TO-220 package.

Features

- Calibrated directly in ˚ Celsius (Centigrade)
- Linear + 10.0 mV/˚C scale factor
- 0.5˚C accuracy guaranteeable (at +25˚C)
- Rated for full −55˚ to +150˚C range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts
- Less than 60 µA current drain
- Low self-heating, 0.08˚C in still air
- Nonlinearity only ±¼˚C typical
- Low impedance output, 0.1 Ω for 1 mA load

Typical Applications

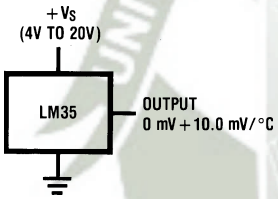
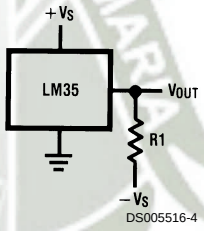


FIGURE 1. Basic Centigrade Temperature Sensor (+2˚C to +150˚C)



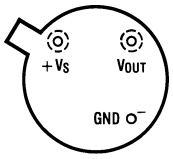
Choose $R_1 = -V_S/50\ \mu A$

V_{OUT} = +1,500 mV at +150˚C
= +250 mV at +25˚C
= −550 mV at −55˚C

FIGURE 2. Full-Range Centigrade Temperature Sensor

Connection Diagrams

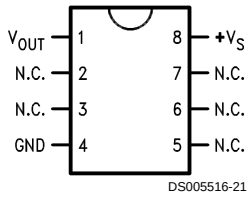
TO-46
Metal Can Package*



*Case is connected to negative pin (GND)

Order Number LM35H, LM35AH, LM35CH, LM35CAH or LM35DH
See NS Package Number H03H

SO-8
Small Outline Molded Package



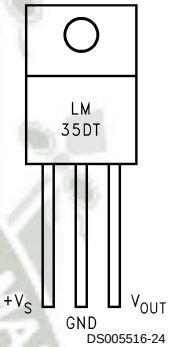
N.C. = No Connection

Top View
Order Number LM35DM
See NS Package Number M08A

TO-92
Plastic Package

Order Number LM35CZ, LM35CAZ or LM35DZ
See NS Package Number Z03A

TO-220
Plastic Package*



*Tab is connected to the negative pin (GND).
Note: The LM35DT pinout is different than the discontinued LM35DP.

Order Number LM35DT
See NS Package Number TA03F

Absolute Maximum Ratings (Note 10)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage	+35V to −0.2V
Output Voltage	+6V to −1.0V
Output Current	10 mA
Storage Temp.;	
TO-46 Package,	−60˚C to +180˚C
TO-92 Package,	−60˚C to +150˚C
SO-8 Package,	−65˚C to +150˚C
TO-220 Package,	−65˚C to +150˚C
Lead Temp.:	
TO-46 Package, (Soldering, 10 seconds)	300˚C
TO-92 and TO-220 Package, (Soldering, 10 seconds)	260˚C
SO Package (Note 12)	
Vapor Phase (60 seconds)	215˚C
Infrared (15 seconds)	220˚C
ESD Susceptibility (Note 11)	2500V
Specified Operating Temperature Range: T_{MIN} to T_{MAX} (Note 2)	
LM35, LM35A	−55˚C to +150˚C
LM35C, LM35CA	−40˚C to +110˚C
LM35D	0˚C to +100˚C

Electrical Characteristics

(Notes 1, 6)

Parameter	Conditions	LM35A			LM35CA			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy (Note 7)	T_A=+25˚C	±0.2	±0.5		±0.2	±0.5		˚C
	T_A=−10˚C	±0.3			±0.3		±1.0	˚C
	T_A=T_{MAX}	±0.4	±1.0		±0.4	±1.0		˚C
	T_A=T_{MIN}	±0.4	±1.0		±0.4		±1.5	˚C
Nonlinearity (Note 8)	T_{MIN}≤T_A≤T_{MAX}	**±0.18**		**±0.35**	**±0.15**		**±0.3**	˚C
Sensor Gain (Average Slope)	T_{MIN}≤T_A≤T_{MAX}	**+10.0**	**+9.9, +10.1**		**+10.0**		**+9.9, +10.1**	mV/˚C
Load Regulation (Note 3) 0≤I_L≤1 mA	T_A=+25˚C	±0.4	±1.0		±0.4	±1.0		mV/mA
	T_{MIN}≤T_A≤T_{MAX}	**±0.5**		**±3.0**	**±0.5**		**±3.0**	mV/mA
Line Regulation (Note 3)	T_A=+25˚C	±0.01	±0.05		±0.01	±0.05		mV/V
	4V≤V_S≤30V	**±0.02**		**±0.1**	**±0.02**		**±0.1**	mV/V
Quiescent Current (Note 9)	V_S=+5V, +25˚C	56	67		56	67		µA
	V_S=+5V	**105**		**131**	**91**		**114**	µA
	V_S=+30V, +25˚C	56.2	68		56.2	68		µA
	V_S=+30V	**105.5**		**133**	**91.5**		**116**	µA
Change of Quiescent Current (Note 3)	4V≤V_S≤30V, +25˚C	0.2	1.0		0.2	1.0		µA
	4V≤V_S≤30V	**0.5**		**2.0**	**0.5**		**2.0**	µA
Temperature Coefficient of Quiescent Current		**+0.39**		**+0.5**	**+0.39**		**+0.5**	µA/˚C
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, I_L=0	+1.5		+2.0	+1.5		+2.0	˚C
Long Term Stability	T_J=T_{MAX}, for 1000 hours	±0.08			±0.08			˚C

Electrical Characteristics

(Notes 1, 6)

Parameter	Conditions	LM35			LM35C, LM35D			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy,	T_A=+25˚C	±0.4	±1.0		±0.4	±1.0		˚C
LM35, LM35C	T_A=−10˚C	±0.5			±0.5		±1.5	˚C
(Note 7)	T_A=T_{MAX}	±0.8	±1.5		±0.8		±1.5	˚C
	T_A=T_{MIN}	±0.8		±1.5	±0.8		±2.0	˚C
Accuracy, LM35D	T_A=+25˚C				±0.6	±1.5		˚C
(Note 7)	T_A=T_{MAX}				±0.9		±2.0	˚C
	T_A=T_{MIN}				±0.9		±2.0	˚C
Nonlinearity	T_{MIN}≤T_A≤T_{MAX}	**±0.3**		**±0.5**	**±0.2**		**±0.5**	˚C
(Note 8)								
Sensor Gain	T_{MIN}≤T_A≤T_{MAX}	**+10.0**	**+9.8, +10.2**		**+10.0**		**+9.8, +10.2**	mV/˚C
(Average Slope)								
Load Regulation	T_A=+25˚C	±0.4	±2.0		±0.4	±2.0		mV/mA
(Note 3) 0≤I_L≤1 mA	T_{MIN}≤T_A≤T_{MAX}	**±0.5**		**±5.0**	**±0.5**		**±5.0**	mV/mA
Line Regulation	T_A=+25˚C	±0.01	±0.1		±0.01	±0.1		mV/V
(Note 3)	4V≤V_S≤30V	**±0.02**		**±0.2**	**±0.02**		**±0.2**	mV/V
Quiescent Current	V_S=+5V, +25˚C	56	80		56	80		µA
(Note 9)	V_S=+5V	**105**		**158**	**91**		**138**	µA
	V_S=+30V, +25˚C	56.2	82		56.2	82		µA
	V_S=+30V	**105.5**		**161**	**91.5**		**141**	µA
Change of	4V≤V_S≤30V, +25˚C	0.2	2.0		0.2	2.0		µA
Quiescent Current	4V≤V_S≤30V	**0.5**		**3.0**	**0.5**		**3.0**	µA
(Note 3)								
Temperature Coefficient of Quiescent Current		**+0.39**		**+0.7**	**+0.39**		**+0.7**	µA/˚C
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, I_L=0	+1.5		+2.0	+1.5		+2.0	˚C
Long Term Stability	T_J=T_{MAX}, for 1000 hours	±0.08			±0.08			˚C

Note 1: Unless otherwise noted, these specifications apply: −55˚C≤T_J≤+150˚C for the LM35 and LM35A; −40˚C≤T_J≤+110˚C for the LM35C and LM35CA; and 0˚C≤T_J≤+100˚C for the LM35D. V_S=+5Vdc and I_{LOAD}=50 µA, in the circuit of *Figure 2*. These specifications also apply from +2˚C to T_{MAX} in the circuit of *Figure 1*. Specifications in **boldface** apply over the full rated temperature range.

Note 2: Thermal resistance of the TO-46 package is 400˚C/W, junction to ambient, and 24˚C/W junction to case. Thermal resistance of the TO-92 package is 180˚C/W junction to ambient. Thermal resistance of the small outline molded package is 220˚C/W junction to ambient. Thermal resistance of the TO-220 package is 90˚C/W junction to ambient. For additional thermal resistance information see table in the Applications section.

Note 3: Regulation is measured at constant junction temperature, using pulse testing with a low duty cycle. Changes in output due to heating effects can be computed by multiplying the internal dissipation by the thermal resistance.

Note 4: Tested Limits are guaranteed and 100% tested in production.

Note 5: Design Limits are guaranteed (but not 100% production tested) over the indicated temperature and supply voltage ranges. These limits are not used to calculate outgoing quality levels.

Note 6: Specifications in **boldface** apply over the full rated temperature range.

Note 7: Accuracy is defined as the error between the output voltage and 10mv/˚C times the device's case temperature, at specified conditions of voltage, current, and temperature (expressed in ˚C).

Note 8: Nonlinearity is defined as the deviation of the output-voltage-versus-temperature curve from the best-fit straight line, over the device's rated temperature range.

Note 9: Quiescent current is defined in the circuit of *Figure 1*.

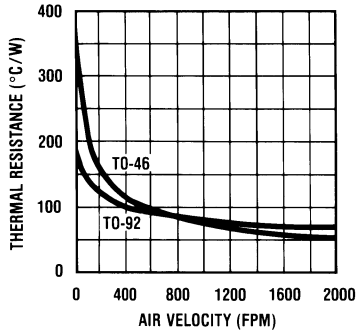
Note 10: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its rated operating conditions. See Note 1.

Note 11: Human body model, 100 pF discharged through a 1.5 kΩ resistor.

Note 12: See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" or the section titled "Surface Mount" found in a current National Semiconductor Linear Data Book for other methods of soldering surface mount devices.

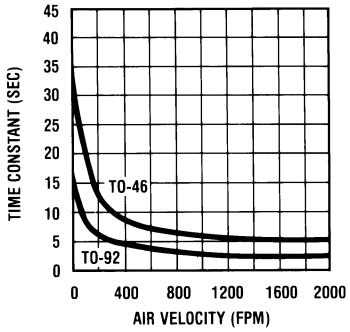
Typical Performance Characteristics

Thermal Resistance Junction to Air



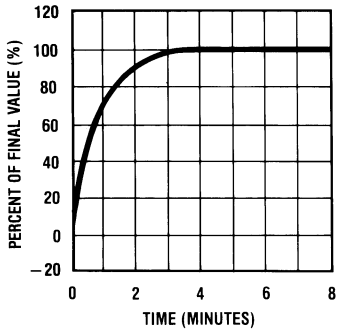
DS005516-25

Thermal Time Constant



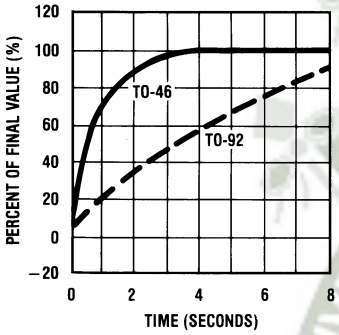
DS005516-26

Thermal Response in Still Air



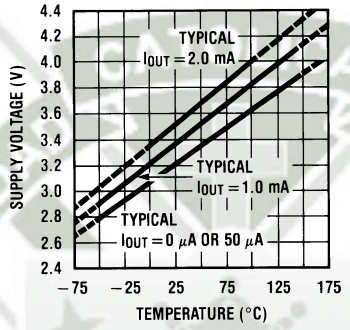
DS005516-27

Thermal Response in Stirred Oil Bath



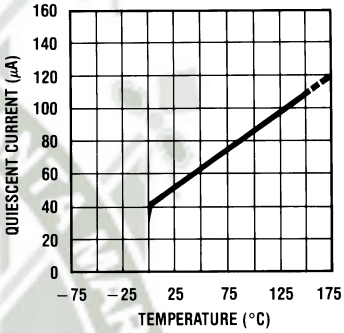
DS005516-28

Minimum Supply Voltage vs. Temperature



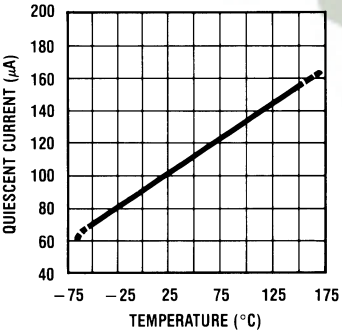
DS005516-29

Quiescent Current vs. Temperature (In Circuit of *Figure 1*.)



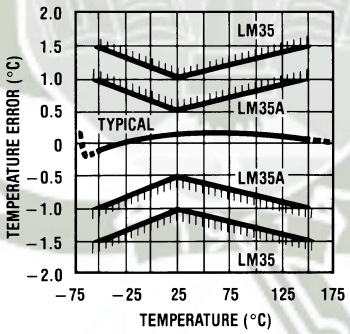
DS005516-30

Quiescent Current vs. Temperature (In Circuit of *Figure 2*.)



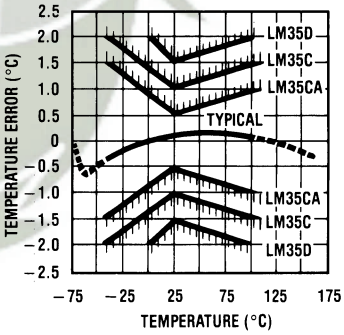
DS005516-31

Accuracy vs. Temperature (Guaranteed)



DS005516-32

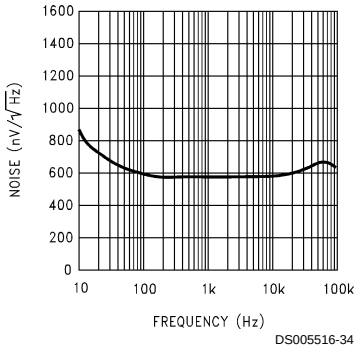
Accuracy vs. Temperature (Guaranteed)



DS005516-33

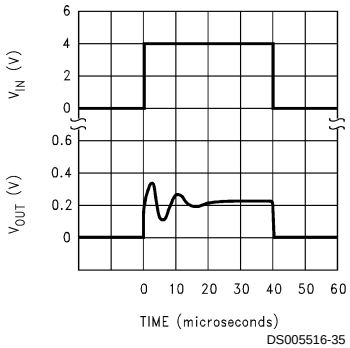
Typical Performance Characteristics (Continued)

Noise Voltage



DS005516-34

Start-Up Response



DS005516-35

Applications

The LM35 can be applied easily in the same way as other integrated-circuit temperature sensors. It can be glued or cemented to a surface and its temperature will be within about 0.01˚C of the surface temperature.

This presumes that the ambient air temperature is almost the same as the surface temperature; if the air temperature were much higher or lower than the surface temperature, the actual temperature of the LM35 die would be at an intermediate temperature between the surface temperature and the air temperature. This is expecially true for the TO-92 plastic package, where the copper leads are the principal thermal path to carry heat into the device, so its temperature might be closer to the air temperature than to the surface temperature.

To minimize this problem, be sure that the wiring to the LM35, as it leaves the device, is held at the same temperature as the surface of interest. The easiest way to do this is to cover up these wires with a bead of epoxy which will insure that the leads and wires are all at the same temperature as the surface, and that the LM35 die's temperature will not be affected by the air temperature.

The TO-46 metal package can also be soldered to a metal surface or pipe without damage. Of course, in that case the V− terminal of the circuit will be grounded to that metal. Alternatively, the LM35 can be mounted inside a sealed-end metal tube, and can then be dipped into a bath or screwed into a threaded hole in a tank. As with any IC, the LM35 and accompanying wiring and circuits must be kept insulated and dry, to avoid leakage and corrosion. This is especially true if the circuit may operate at cold temperatures where condensation can occur. Printed-circuit coatings and varnishes such as Humiseal and epoxy paints or dips are often used to insure that moisture cannot corrode the LM35 or its connections.

These devices are sometimes soldered to a small light-weight heat fin, to decrease the thermal time constant and speed up the response in slowly-moving air. On the other hand, a small thermal mass may be added to the sensor, to give the steadiest reading despite small deviations in the air temperature.

Temperature Rise of LM35 Due To Self-heating (Thermal Resistance, θ_{JA})

	TO-46, no heat sink	TO-46*, small heat fin	TO-92, no heat sink	TO-92**, small heat fin	SO-8 no heat sink	SO-8** small heat fin	TO-220 no heat sink
Still air	400˚C/W	100˚C/W	180˚C/W	140˚C/W	220˚C/W	110˚C/W	90˚C/W
Moving air	100˚C/W	40˚C/W	90˚C/W	70˚C/W	105˚C/W	90˚C/W	26˚C/W
Still oil	100˚C/W	40˚C/W	90˚C/W	70˚C/W			
Stirred oil	50˚C/W	30˚C/W	45˚C/W	40˚C/W			
(Clamped to metal, Infinite heat sink)	(24˚C/W)				(55˚C/W)		

*Wakefield type 201, or 1" disc of 0.020" sheet brass, soldered to case, or similar.
**TO-92 and SO-8 packages glued and leads soldered to 1" square of 1/16" printed circuit board with 2 oz. foil or similar.

Typical Applications

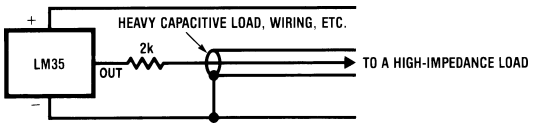


FIGURE 3. LM35 with Decoupling from Capacitive Load

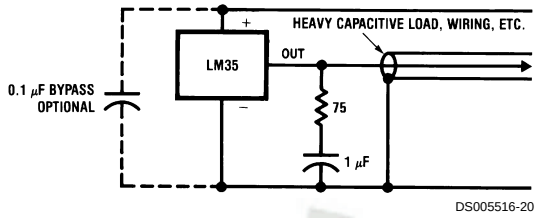


FIGURE 4. LM35 with R-C Damper

CAPACITIVE LOADS

Like most micropower circuits, the LM35 has a limited ability to drive heavy capacitive loads. The LM35 by itself is able to drive 50 pf without special precautions. If heavier loads are anticipated, it is easy to isolate or decouple the load with a resistor; see *Figure 3*. Or you can improve the tolerance of capacitance with a series R-C damper from output to ground; see *Figure 4*.

When the LM35 is applied with a 200Ω load resistor as shown in *Figure 5*, *Figure 6* or *Figure 8* it is relatively immune to wiring capacitance because the capacitance forms a by-pass from ground to input, not on the output. However, as with any linear circuit connected to wires in a hostile environment, its performance can be affected adversely by intense electromagnetic sources such as relays, radio transmitters, motors with arcing brushes, SCR transients, etc, as its wiring can act as a receiving antenna and its internal junctions can act as rectifiers. For best results in such cases, a bypass capacitor from V_{IN} to ground and a series R-C damper such as 75Ω in series with 0.2 or 1 μF from output to ground are often useful. These are shown in *Figure 13*, *Figure 14*, and *Figure 16*.

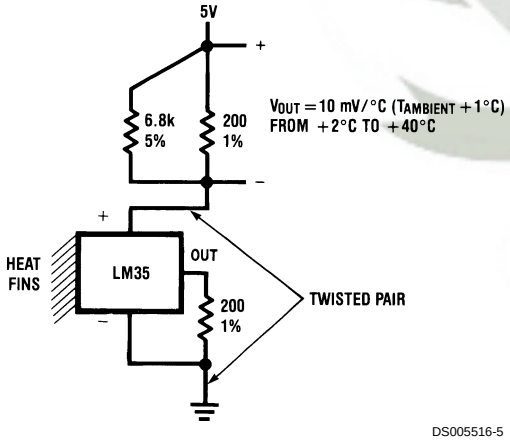


FIGURE 5. Two-Wire Remote Temperature Sensor (Grounded Sensor)

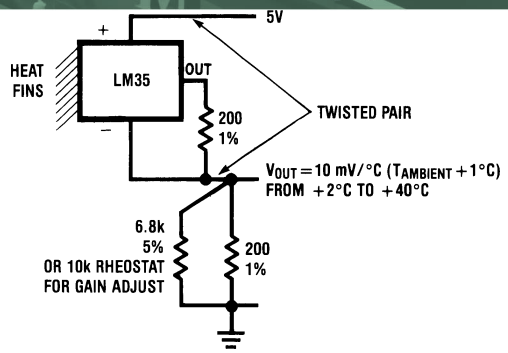


FIGURE 6. Two-Wire Remote Temperature Sensor (Output Referred to Ground)

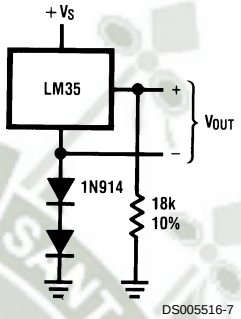


FIGURE 7. Temperature Sensor, Single Supply, −55˚ to +150˚C

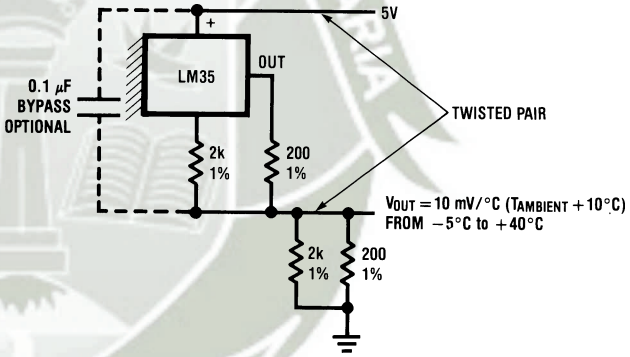


FIGURE 8. Two-Wire Remote Temperature Sensor (Output Referred to Ground)

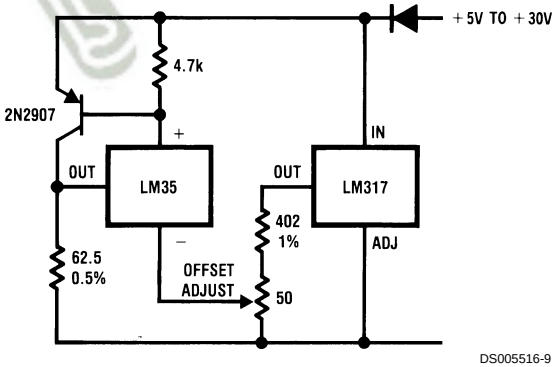


FIGURE 9. 4-To-20 mA Current Source (0˚C to +100˚C)

Typical Applications (Continued)

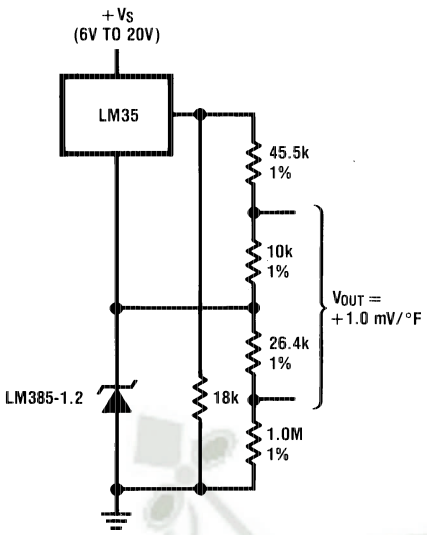


FIGURE 10. Fahrenheit Thermometer

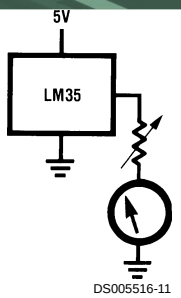


FIGURE 11. Centigrade Thermometer (Analog Meter)

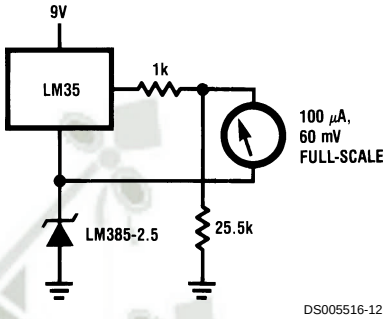


FIGURE 12. Fahrenheit ThermometerExpanded Scale Thermometer
(50˚ to 80˚ Fahrenheit, for Example Shown)

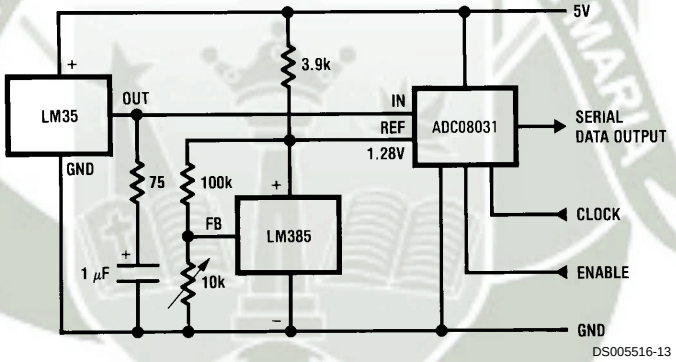


FIGURE 13. Temperature To Digital Converter (Serial Output) (+128˚C Full Scale)

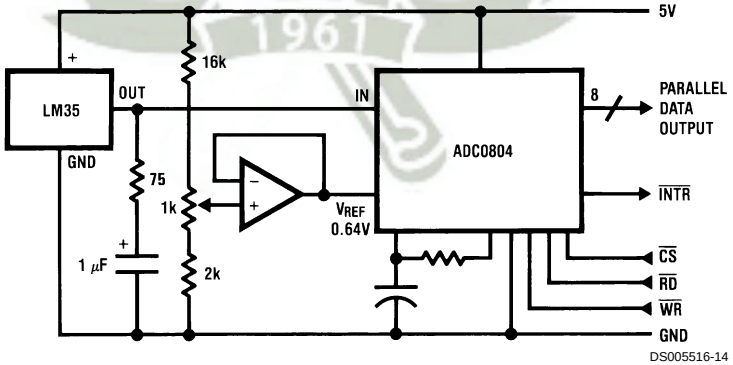
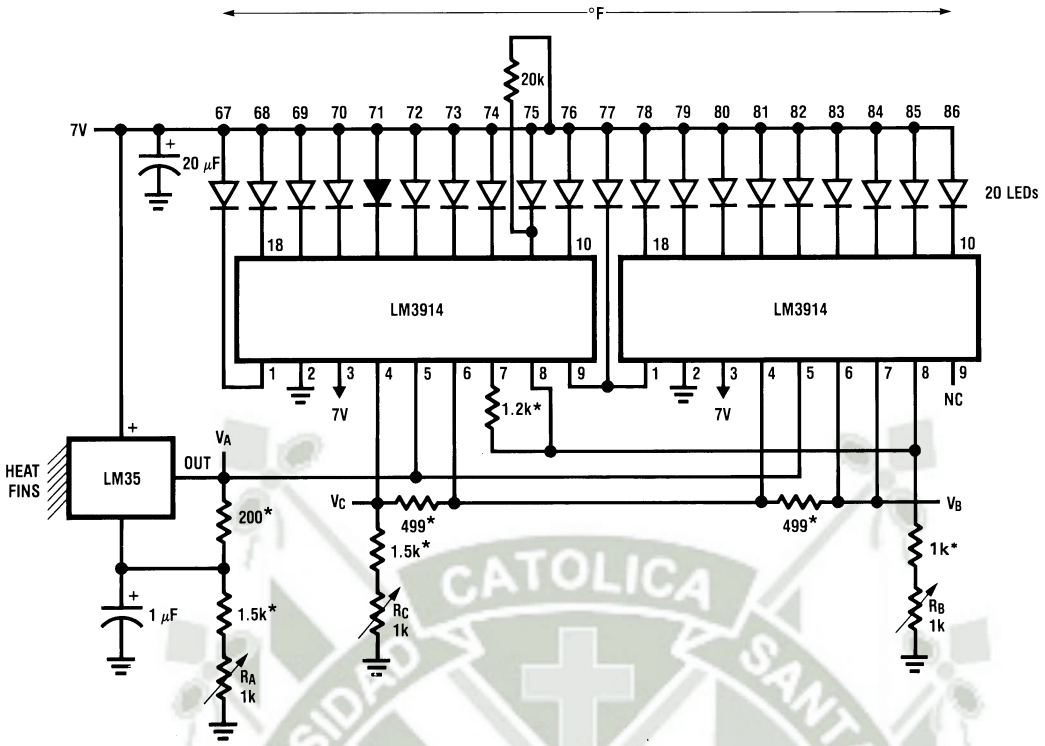


FIGURE 14. Temperature To Digital Converter (Parallel TRI-STATE™ Outputs for Standard Data Bus to µP Interface) (128˚C Full Scale)

Typical Applications (Continued)



*=1% or 2% film resistor
Trim R_B for V_B=3.075V
Trim R_C for V_C=1.955V
Trim R_A for V_A=0.075V + 100mV/°C x $T_{ambient}$
Example, V_A=2.275V at 22°C

FIGURE 15. Bar-Graph Temperature Display (Dot Mode)

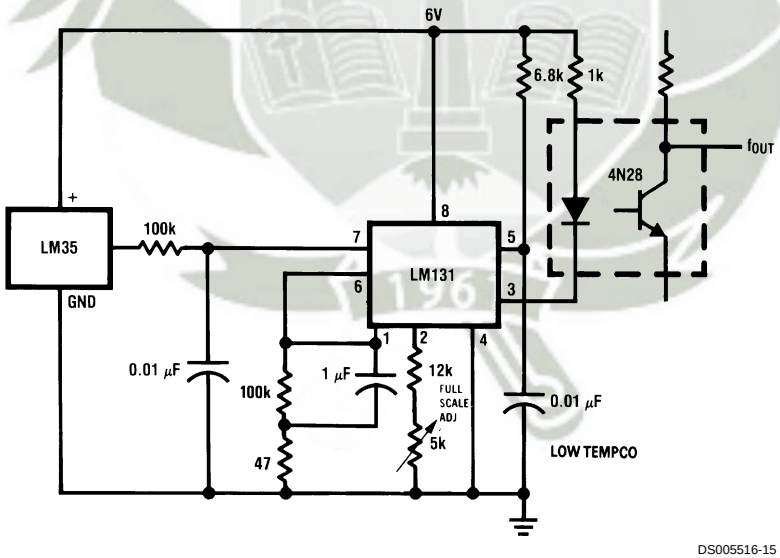
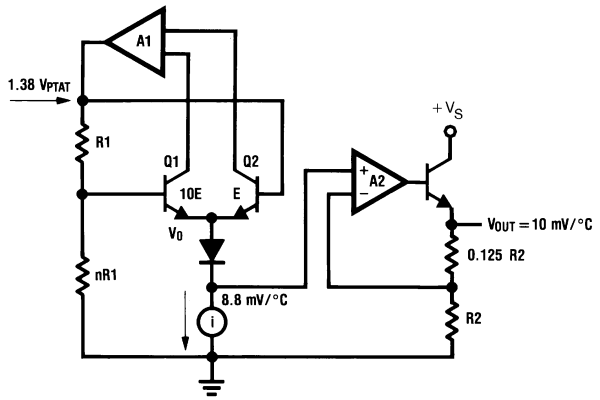


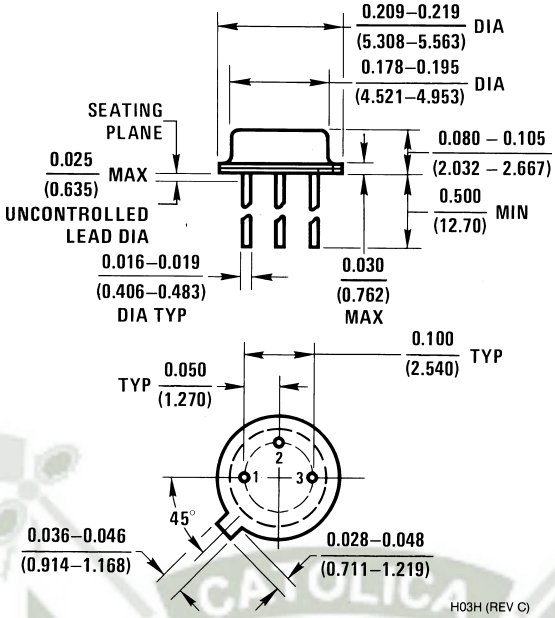
FIGURE 16. LM35 With Voltage-To-Frequency Converter And Isolated Output (2°C to +150°C; 20 Hz to 1500 Hz)

Block Diagram

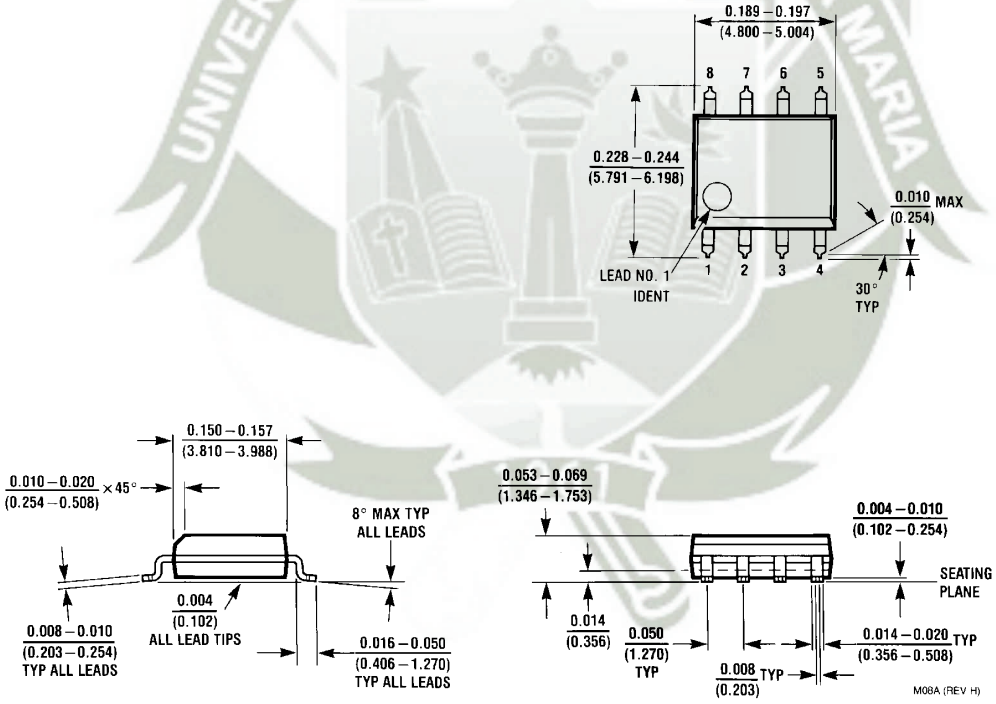


DS005516-23

Physical Dimensions inches (millimeters) unless otherwise noted

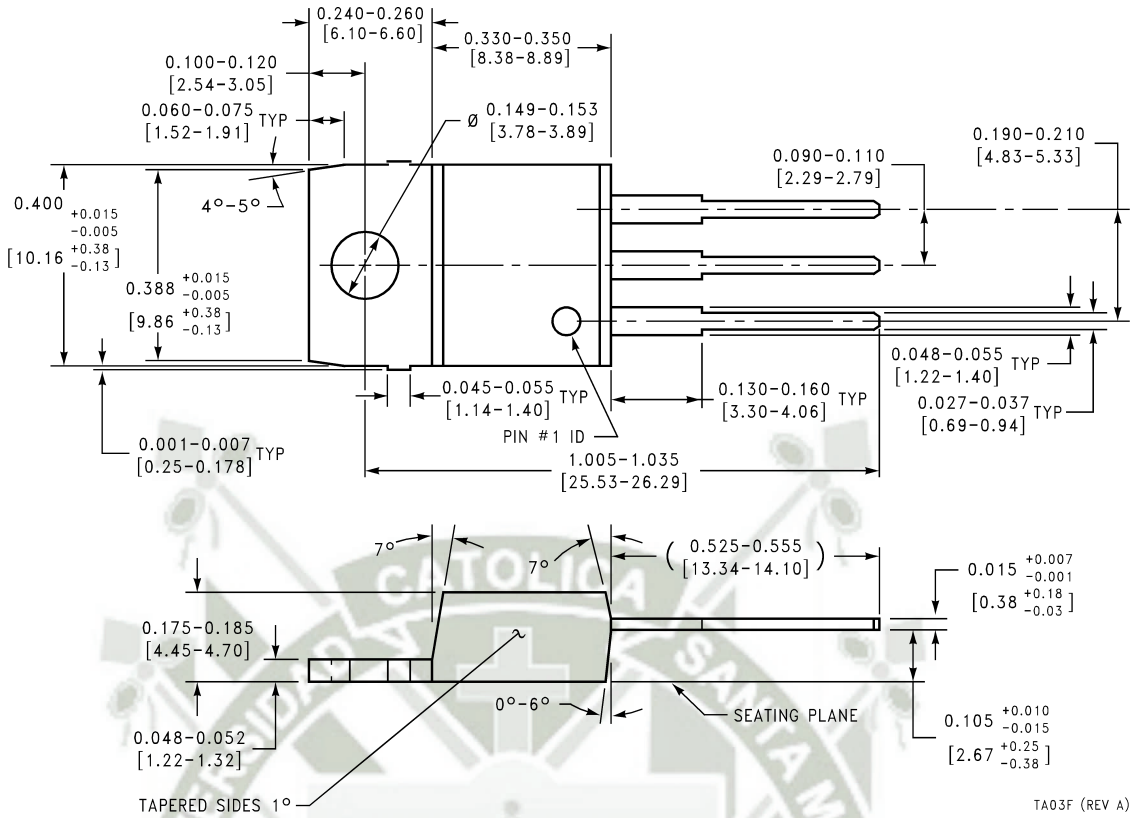


TO-46 Metal Can Package (H)
Order Number LM35H, LM35AH, LM35CH, LM35CAH, or LM35DH
NS Package Number H03H



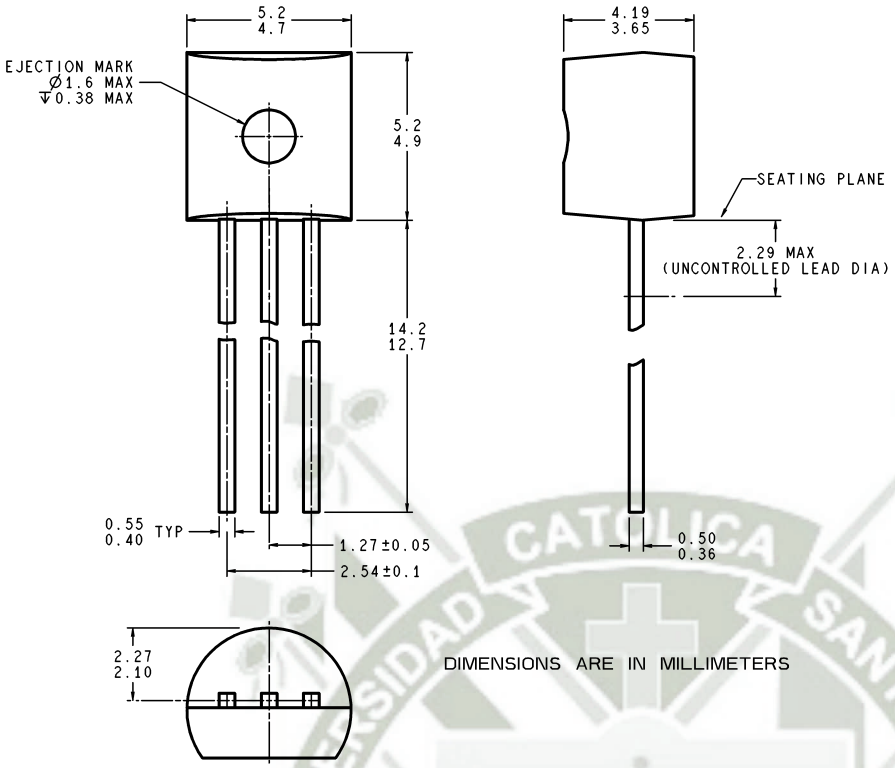
SO-8 Molded Small Outline Package (M)
Order Number LM35DM
NS Package Number M08A

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



Power Package TO-220 (T)
Order Number LM35DT
NS Package Number TA03F

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



TO-92 Plastic Package (Z)
Order Number LM35CZ, LM35CAZ or LM35DZ
NS Package Number Z03A

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
Americas
Tel: 1-800-272-9959
Fax: 1-800-737-7018
Email: support@nsc.com
www.national.com

National Semiconductor Europe
Fax: +49 (0) 180-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 69 9508 6208
English Tel: +44 (0) 870 24 0 2171
Français Tel: +33 (0) 1 41 91 8790

National Semiconductor Asia Pacific Customer Response Group
Tel: 65-2544466
Fax: 65-2504466
Email: ap.support@nsc.com

National Semiconductor Japan Ltd.
Tel: 81-3-5639-7560
Fax: 81-3-5639-7507

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Audio	www.ti.com/audio	Communications and Telecom	www.ti.com/communications
Amplifiers	amplifier.ti.com	Computers and Peripherals	www.ti.com/computers
Data Converters	dataconverter.ti.com	Consumer Electronics	www.ti.com/consumer-apps
DLP® Products	www.dlp.com	Energy and Lighting	www.ti.com/energy
DSP	dsp.ti.com	Industrial	www.ti.com/industrial
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Security	www.ti.com/security
Logic	logic.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Power Mgmt	power.ti.com	Transportation and Automotive	www.ti.com/automotive
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Mobile Processors	www.ti.com/omap		
Wireless Connectivity	www.ti.com/wirelessconnectivity		

TI E2E Community Home Page e2e.ti.com

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

SUNROM TECHNOLOGIES
Your Source for Embedded Systems

Email: info@sunrom.com or sunrom@gmail.com

Visit us at http://www.sunrom.com

Document: Datasheet Date: 28-Jul-08 Model #: 3190 Product's Page: www.sunrom.com/p-510.html

Light Dependent Resistor - LDR

Two cadmium sulphide(cds) photoconductive cells with spectral responses similar to that of the human eye. The cell resistance falls with increasing light intensity. Applications include smoke detection, automatic lighting control, batch counting and burglar alarm systems.

Applications

Photoconductive cells are used in many different types of circuits and applications.

Analog Applications

- Camera Exposure Control
- Auto Slide Focus - dual cell
- Photocopy Machines - density of toner
- Colorimetric Test Equipment
- Densitometer
- Electronic Scales - dual cell
- Automatic Gain Control – modulated light source
- Automated Rear View Mirror

Digital Applications

- Automatic Headlight Dimmer
- Night Light Control
- Oil Burner Flame Out
- Street Light Control
- Absence / Presence (beam breaker)
- Position Sensor

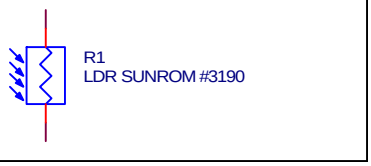
Electrical Characteristics

Parameter	Conditions	Min	Typ	Max	Unit
Cell resistance	1000 LUX 10 LUX	- -	400 9	- -	Ohm K Ohm
Dark Resistance	-	-	1	-	M Ohm
Dark Capacitance	-	-	3.5	-	pF
Rise Time	1000 LUX 10 LUX	- -	2.8 18	- -	ms ms
Fall Time	1000 LUX 10 LUX	- -	48 120	- -	ms ms
Voltage AC/DC Peak		-	-	320	V max
Current		-	-	75	mA max
Power Dissipation				100	mW max
Operating Temperature		-60	-	+75	Deg. C

Guide to source illuminations

Light source Illumination	LUX
Moonlight	0.1
60W Bulb at 1m	50
1W MES Bulb at 0.1m	100
Fluorescent Lighting	500
Bright Sunlight	30,000

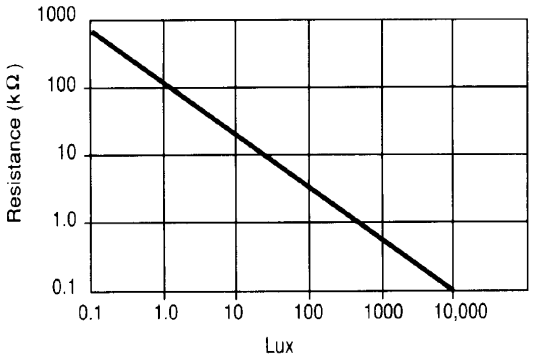
FIGURE 1 CIRCUIT SYMBOL



Sensitivity

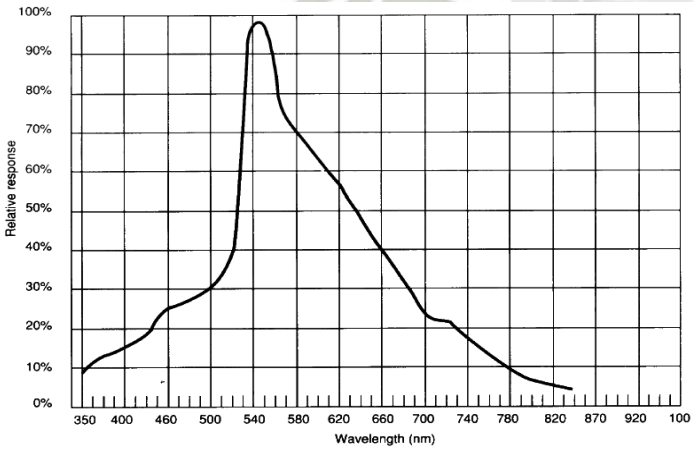
The sensitivity of a photodetector is the relationship between the light falling on the device and the resulting output signal. In the case of a photocell, one is dealing with the relationship between the incident light and the corresponding resistance of the cell.

FIGURE 2 RESISTANCE AS FUNCTION OF ILLUMINATION



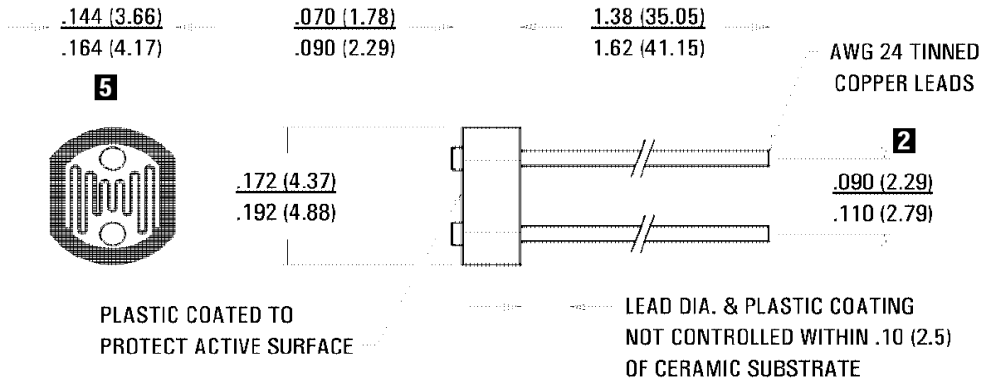
Spectral Response

Figure 3 Spectral response



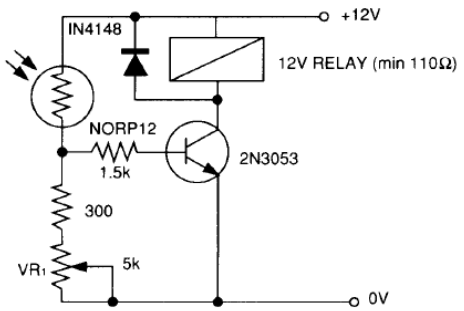
Like the human eye, the relative sensitivity of a photoconductive cell is dependent on the wavelength (color) of the incident light. Each photoconductor material type has its own unique spectral response curve or plot of the relative response of the photocell versus wavelength of light.

Dimensions



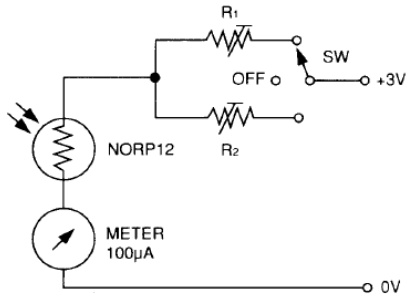
Typical Application Circuits

Figure 6 Sensitive light operated relay



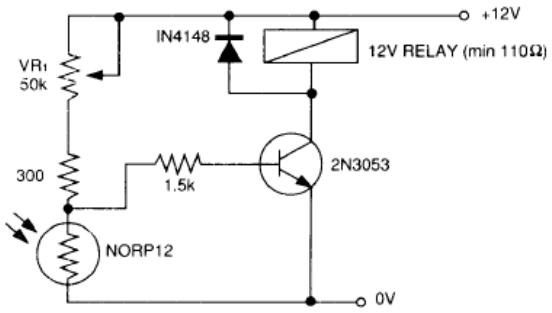
Relay energised when light level increases above the level set by VR_1

Figure 9 Logarithmic law photographic light meter



Typical value $R^1 = 100k\Omega$
$R^2 = 200k\Omega$ preset to give two overlapping ranges.
(Calibration should be made against an accurate meter.)

Figure 7 Light interruption detector



As Figure 6 relay energised when light level drops below the level set by VR_1

Figure 8 Automatic light circuit

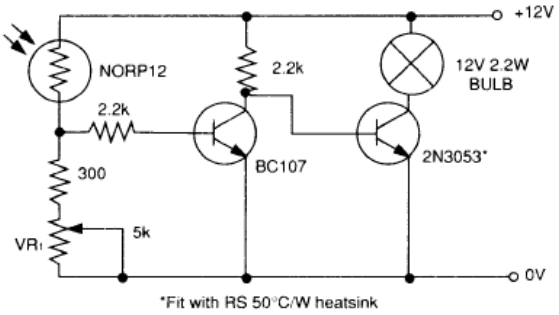
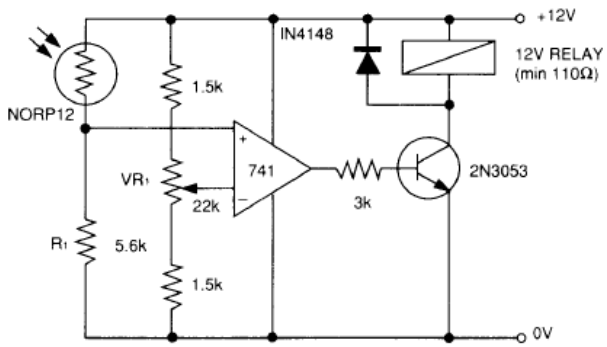
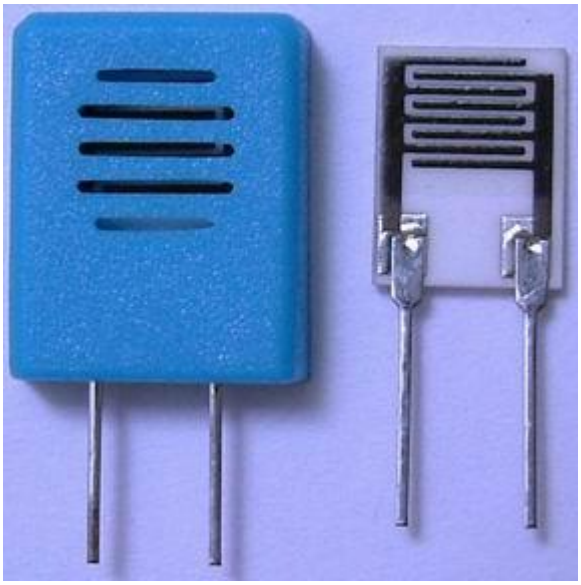


Figure 10 Extremely sensitive light operated relay



(Relay energised when light exceeds preset level.) Incorporates a balancing bridge and op-amp. R_1 and NORP12 may be interchanged for the reverse function.

Resistive humidity sensor, Model: HR202



1. HR202 is a new kind of humidity-sensitive resistor made from organic macromolecule materials, it can be used in occasions like: hospitals, storage, workshop, textile industry, tobaccos, pharmaceutical field, meteorology, etc.

2. Features:

Excellent linearity, low power consumption, wide measurement range, quick response, anti-pollution, high stability, high performance-price ratio.

3. Technical Specification:

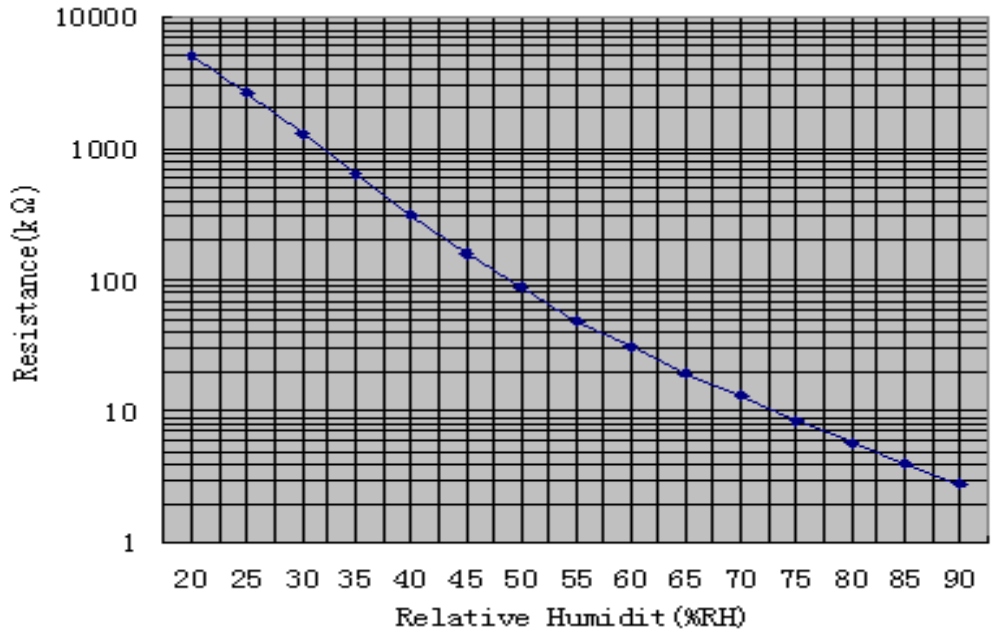
Operating range:	humidity(20-95%RH) temperature(0-60Celsius)
Power supply:	1.5V AC(Max sine)
Operating frequency:	500Hz-2kHz
Rated power:	0.2mW(Max sine)
Central value:	31kΩ(at 25Celsius, 1kHz ,1V AC, 60%RH)
Impedance range:	19.8-50.2kΩ(at 25Celsius, 1kHz ,1V AC, 60%RH)
Accuracy:	+-5%RH
Hysteresis:	+-1%RH
Long-term stability:	+-1%RH/year
Response time:	<10s
Dimensions:	with case 12*15*5mm, without case 8*10*0.7mm

4.Performance parameter(at 1KHz) Unit: ohm

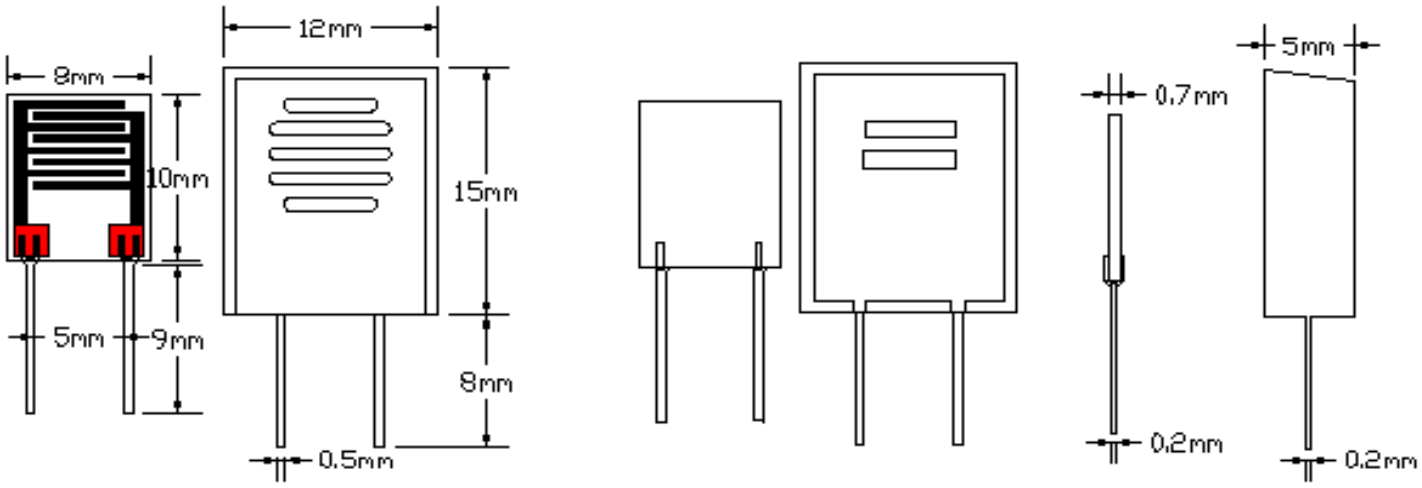
_

	0℃	5℃	10℃	15℃	20℃	25℃	30℃	35℃	40℃	45℃	50℃	55℃	60℃
20%RH				10M	6.7M	5.0M	3.9M	3.0M	2.4M	1.75M	1.45M	1.15M	970K
25%RH		10M	7.0M	5.0M	3.4M	2.6M	1.9M	1.5M	1.1M	880K	700K	560K	450K
30%RH	6.4M	4.6M	3.2M	2.3M	1.75M	1.3M	970K	740K	570K	420K	340K	270K	215K
35%RH	2.9M	2.1M	1.5M	1.1M	850K	630K	460K	380K	280K	210K	170K	130K	150K
40%RH	1.4M	1.0M	750K	540K	420K	310K	235K	190K	140K	110K	88K	70K	57K
45%RH	700K	500K	380K	280K	210K	160K	125K	100K	78K	64K	50K	41K	34K
50%RH	370K	26K	200K	150K	115K	87K	69K	56K	45K	38K	31K	25K	21K
55%RH	190K	140K	110K	84K	64K	49K	39K	33K	27K	24K	19.5K	17K	14K
60%RH	105K	80K	62K	50K	39K	31K	25K	20K	17.5K	15K	13K	11K	9.4K
65%RH	62K	48K	37K	30K	24K	19.5K	16K	13K	11.5K	10K	8.6K	7.6K	6.8K
70%RH	38K	30K	24K	19K	15.5K	13K	10.5K	9.0K	8.0K	7.0K	6.0K	5.4K	4.8K
75%RH	23K	18K	15K	12K	10K	8.4K	7.2K	6.2K	5.6K	4.9K	4.2K	3.8K	3.4K
80%RH	15.5K	12.0K	10.0K	8.0K	7.0K	5.7K	5.0K	4.3K	3.9K	3.4K	3.0K	2.7K	2.5K
85%RH	10.5K	8.2K	6.8K	5.5K	4.8K	4.0K	3.5K	3.1K	2.8K	2.4K	2.1K	1.9K	1.8K
90%RH	7.1K	5.3K	4.7K	4.0K	3.3K	2.8K	2.5K	2.2K	2.0K	1.8K	1.55K	1.4K	1.3K

5.Impedance performance (at25℃ 1V AC 1kHz)



6.Dimensions



Cautions:

(1) Avoid polarization, driving voltage or current should be 100% alternative.
(2) Please measure the sensor with LCR alternative-current bridge, don't use multimeter.
(3) Avoid dew condensation.
(4) Recommended storage conditions: temperature 0-60Celsius; humidity <80%RH

HC-05

-Bluetooth to Serial Port Module

Overview

HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup.

Serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation with complete 2.4GHz radio transceiver and baseband. It uses CSR Bluecore 04-External single chip Bluetooth system with CMOS technology and with AFH(Adaptive Frequency Hopping Feature). It has the footprint as small as 12.7mmx27mm. Hope it will simplify your overall design/development cycle.

Specifications

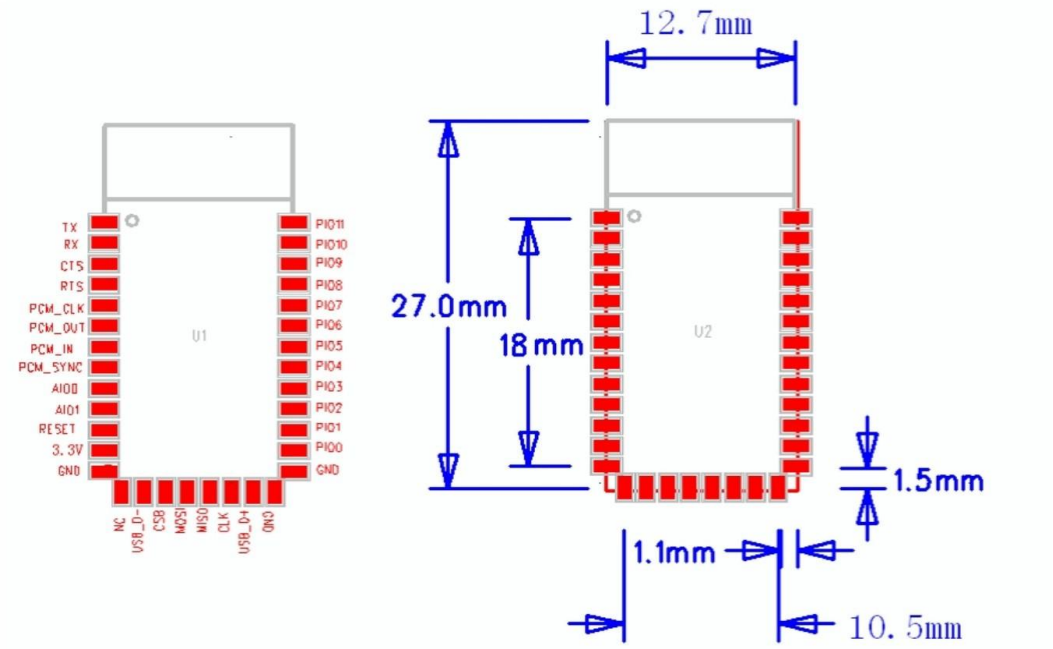
Hardware features

- Typical -80dBm sensitivity
- Up to +4dBm RF transmit power
- Low Power 1.8V Operation ,1.8 to 3.6V I/O
- PIO control
- UART interface with programmable baud rate
- With integrated antenna
- With edge connector

Software features

- Default Baud rate: 38400, Data bits:8, Stop bit:1,Parity:No parity, Data control: has. Supported baud rate: 9600,19200,38400,57600,115200,230400,460800.
- Given a rising pulse in PIO0, device will be disconnected.
- Status instruction port PIO1: low-disconnected, high-connected;
- PIO10 and PIO11 can be connected to red and blue led separately. When master and slave are paired, red and blue led blinks 1time/2s in interval, while disconnected only blue led blinks 2times/s.
- Auto-connect to the last device on power as default.
- Permit pairing device to connect as default.
- Auto-pairing PINCODE:"0000" as default
- Auto-reconnect in 30 min when disconnected as a result of beyond the range of connection.

Hardware



PIN Name	PIN #	Pad type	Description	Note
GND	13 21 22	VSS	Ground pot	
3.3 VCC	12	3.3V	Integrated 3.3V (+) supply with On-chip linear regulator output within 3.15-3.3V	
AIO0	9	Bi-Directional	Programmable input/output line	
AIO1	10	Bi-Directional	Programmable input/output line	
PIO0	23	Bi-Directional RX EN	Programmable input/output line, control output for LNA(if fitted)	
PIO1	24	Bi-Directional TX EN	Programmable input/output line, control output for PA(if fitted)	
PIO2	25	Bi-Directional	Programmable input/output line	
PIO3	26	Bi-Directional	Programmable input/output line	
PIO4	27	Bi-Directional	Programmable input/output line	
PIO5	28	Bi-Directional	Programmable input/output line	
PIO6	29	Bi-Directional	Programmable input/output line	
PIO7	30	Bi-Directional	Programmable input/output line	
PIO8	31	Bi-Directional	Programmable input/output line	
PIO9	32	Bi-Directional	Programmable input/output line	
PIO10	33	Bi-Directional	Programmable input/output line	
PIO11	34	Bi-Directional	Programmable input/output line	

RESETB	11	CMOS input with weak internal pull-up	Reset if low.input debouncde so must be low for >5MS to cause a reset	
UART_RTS	4	CMOS output, tri-stable with weak internal pull-up	UART request to send, active low	
UART_CTS	3	CMOS input with weak internal pull-down	UART clear to send, active low	
UART_RX	2	CMOS input with weak internal pull-down	UART Data input	
UART_TX	1	CMOS output, Tri-stable with weak internal pull-up	UART Data output	
SPI_MOSI	17	CMOS input with weak internal pull-down	Serial peripheral interface data input	
SPI_CSB	16	CMOS input with weak internal pull-up	Chip select for serial peripheral interface, active low	
SPI_CLK	19	CMOS input with weak internal pull-down	Serial peripheral interface clock	
SPI_MISO	18	CMOS input with weak internal pull-down	Serial peripheral interface data Output	
USB_-	15	Bi-Directional		

USB_+	20	Bi-Directional		
NC	14			
PCM_CLK	5	Bi-Directional	Synchronous PCM data clock	
PCM_OUT	6	CMOS output	Synchronous PCM data output	
PCM_IN	7	CMOS Input	Synchronous PCM data input	
PCM_SYNC	8	Bi-Directional	Synchronous PCM data strobe	

AT command Default:

How to set the mode to server (master):
1. Connect PIO11 to high level.
2. Power on, module into command state.
3. Using baud rate 38400, sent the "AT+ROLE=1\r\n" to module, with "OK\r\n" means setting successes.
4. Connect the PIO11 to low level, repower the module, the module work as server (master).

AT commands: (all end with \r\n)

1. Test command:

Command	Respond	Parameter
AT	OK	-

2. Reset

Command	Respond	Parameter
AT+RESET	OK	-

3. Get firmware version

Command	Respond	Parameter
AT+VERSION?	+VERSION:<Param> OK	Param : firmware version

Example:
AT+VERSION?\r\n
+VERSION:2.0-20100601
OK

4. Restore default

Command	Respond	Parameter
AT+ORGL	OK	-

Default state:

Slave mode, pin code :1234, device name: H-C-2010-06-01 ,Baud 38400bits/s.

5. Get module address

Command	Respond	Parameter
AT+ADDR?	+ADDR:<Param> OK	Param: address of Bluetooth module

Bluetooth address: NAP: UAP : LAP

Example:

AT+ADDR?\r\n

+ADDR:1234:56:abcdef

OK

6. Set/Check module name:

Command	Respond	Parameter
AT+NAME=<Param>	OK	Param: Bluetooth module name (Default :HC-05)
AT+NAME?	+NAME:<Param> OK (/FAIL)	

Example:

AT+NAME=HC-05\r\n set the module name to “HC-05”

OK

AT+NAME=ITeadStudio\r\n

OK

AT+NAME?\r\n

+NAME: ITeadStudio

OK

7. Get the Bluetooth device name:

Command	Respond	Parameter
AT+RNAME?<Param1>	1. +NAME:<Param2> OK 2. FAIL	Param1,Param 2 : the address of Bluetooth device

Example: (Device address 00:02:72:od:22:24，name：ITead)

AT+RNAME? 0002，72，od2224\r\n

+RNAME:ITead

OK

8. Set/Check module mode:

Command	Respond	Parameter
AT+ROLE=<Param>	OK	Param: 0- Slave
AT+ ROLE?	+ROLE:<Param>	

	OK	1-Master 2-Slave-Loop

9. Set/Check device class

Command	Respond	Parameter
AT+CLASS=<Param>	OK	Param: Device Class
AT+ CLASS?	1. +CLASS:<Param> OK 2. FAIL	

10. Set/Check GIAC (General Inquire Access Code)

Command	Respond	Parameter
AT+IAC=<Param>	1.OK 2. FAIL	Param: GIAC (Default : 9e8b33)
AT+IAC	+IAC:<Param> OK	

Example:

AT+IAC=9e8b3f\r\n

OK

AT+IAC?\r\n

+IAC: 9e8b3f

OK

11. Set/Check -- Query access patterns

Command	Respond	Parameter
AT+INQM=<Param>,<Param2>,<Param3>	1.OK 2. FAIL	Param: 0——inquiry_mode_standard 1——inquiry_mode_rssi Param2: Maximum number of Bluetooth devices to respond to Param3: Timeout (1-48 : 1.28s to 61.44s)
AT+ INQM?	+INQM： <Param>,<Param2>,<Param3> OK	

Example:

AT+INQM=1,9,48\r\n

OK

AT+INQM\r\n

+INQM:1, 9, 48

OK

12. Set/Check PIN code:

Command	Respond	Parameter
AT+PSWD=<Param>	OK	Param: PIN code (Default 1234)
AT+ PSWD?	+ PSWD ： <Param> OK	

13. Set/Check serial parameter:

Command	Respond	Parameter
AT+UART=<Param>,<Param2>,<Param3>	OK	Param1: Baud Param2: Stop bit Param3: Parity
AT+ UART?	+UART=<Param>,<Param2>,<Param3> OK	

Example:

AT+UART=115200，1,2,\r\n

OK

AT+UART?

+UART:115200,1,2

OK

14. Set/Check connect mode:

Command	Respond	Parameter
AT+CMODE=<Param>	OK	Param: 0 - connect fixed address 1 - connect any address 2 - slave-Loop
AT+ CMODE?	+ CMODE:<Param> OK	

15. Set/Check fixed address:

Command	Respond	Parameter
AT+BIND=<Param>	OK	Param: Fixed address (Default 00:00:00:00:00:00)
AT+ BIND?	+ BIND:<Param> OK	

Example:

AT+BIND=1234，56，abcdef\r\n

OK

AT+BIND?\r\n

+BIND:1234:56:abcdef

OK

16. Set/Check LED I/O

Command	Respond	Parameter
AT+POLAR=<Param1,<Param2>	OK	Param1: 0- PIO8 low drive LED 1- PIO8 high drive LED
AT+ POLAR?	+ POLAR=<Param1>,<Param2> OK	

		Param2: 0- PIO9 low drive LED 1- PIO9 high drive LED

17. Set PIO output

Command	Respond	Parameter
AT+PIO=<Param1>,<Param2>	OK	Param1: PIO number Param2: PIO level 0- low 1- high

Example:

1. PIO10 output high level

AT+PI0=10， 1\r\n

OK

18. Set/Check – scan parameter

Command	Respond	Parameter
AT+IPSCAN=<Param1>,<Param2>,<Param3>,<Param4>	OK	Param1: Query time interval Param2：Query duration Param3：Paging interval Param4：Call duration
AT+IPSCAN?	+IPSCAN:<Param1>,<Param2>,<Param3>,<Param4> OK	

Example:

AT+IPSCAN =1234,500,1200,250\r\n

OK

AT+IPSCAN?

+IPSCAN:1234,500,1200,250

19. Set/Check – SHIFF parameter

Command	Respond	Parameter
AT+SNIFF=<Param1>,<Param2>,<Param3>,<Param4>	OK	Param1: Max time Param2: Min time Param3: Retry time Param4: Time out
AT+ SNIFF?	+SNIFF:<Param1>,<Param2>,<Param3>,<Param4> OK	

20. Set/Check security mode

Command	Respond	Parameter
AT+SENM=<Param1>,<Param2>	1. OK 2. FAIL	Param1: 0——sec_mode0+off 1——sec_mode1+non_se
AT+ SENM?	+ SENM:<Param1>,<Param2>	

	OK	cure 2——sec_mode2_service 3——sec_mode3_link 4——sec_mode_unknow n Param2: 0——hci_enc_mode_off 1——hci_enc_mode_pt_t o_pt 2——hci_enc_mode_pt_t o_pt_and_bcast

21. Delete Authenticated Device

Command	Respond	Parameter
AT+PMSAD=<Param>	OK	Param: Authenticated Device Address

Example:
AT+PMSAD =1234,56,abcdef\r\n
OK

22. Delete All Authenticated Device

Command	Respond	Parameter
AT+ RMAAD	OK	-

23. Search Authenticated Device

Command	Respond	Parameter
AT+FSAD=<Param>	1. OK 2. FAIL	Param: Device address

24. Get Authenticated Device Count

Command	Respond	Parameter
AT+ADCN?	+ADCN：<Param> OK	Param: Device Count

25. Most Recently Used Authenticated Device

Command	Respond	Parameter
AT+MRAD?	+ MRAD：<Param> OK	Param: Recently Authenticated Device Address

26. Get the module working state

Command	Respond	Parameter

AT+ STATE?	+ STATE：<Param> OK	Param: “INITIALIZED” “READY” “PAIRABLE” “PAIRED” “INQUIRING” “CONNECTING” “CONNECTED” “DISCONNECTED” “NUKNOW”

27. Initialize the SPP profile lib

Command	Respond	Parameter
AT+INIT	1. OK 2. FAIL	-

28. Inquiry Bluetooth Device

Command	Respond	Parameter
AT+INQ	+INQ: <Param1>，<Param2>，<Param3> …. OK	Param1：Address Param2：Device Class Param3：RSSI Signal strength

Example:

AT+INIT\r\n

OK

AT+IAC=9e8b33\r\n

OK

AT+CLASS=0\r\n

AT+INQM=1,9,48\r\n

At+INQ\r\n

+INQ:2:72:D2224,3E0104,FFBC

+INQ:1234:56:0,1F1F,FFC1

+INQ:1234:56:0,1F1F,FFC0

+INQ:1234:56:0,1F1F,FFC1

+INQ:2:72:D2224,3F0104,FFAD

+INQ:1234:56:0,1F1F,FFBE

+INQ:1234:56:0,1F1F,FFC2

+INQ:1234:56:0,1F1F,FFBE

+INQ:2:72:D2224,3F0104,FFBC

OK

28. Cancel Inquiring Bluetooth Device

Command	Respond	Parameter
AT+ INQC	OK	-

29. Equipment Matching

Command	Respond	Parameter
AT+PAIR=<Param1>,<Param2>	1. OK 2. FAIL	Param1： Device Address Param2： Time out

30. Connect Device

Command	Respond	Parameter
AT+LINK=<Param>	1. OK 2. FAIL	Param： Device Address

Example:

AT+FSAD=1234,56,abcdef\r\n

OK

AT+LINK=1234,56,abcdef\r\n

OK

31. Disconnect

Command	Respond	Parameter
AT+DISC	1. +DISC:SUCCESS OK 2. +DISC:LINK_LOSS OK 3. +DISC:NO_SLC OK 4. +DISC:TIMEOUT OK 5. +DISC:ERROR OK	Param： Device Address

32. Energy-saving mode

Command	Respond	Parameter
AT+ENSNIFF=<Param>	OK	Param： Device Address

33. Exerts Energy-saving mode

Command	Respond	Parameter
AT+ EXSNIFF =<Param>	OK	Param： Device Address

Revision History

Rev.	Description	Release date
v1.0	Initial version	7/18/2010