****

**TREBALL FINAL DE GRAU**

|  |
| --- |
| **TÍTOL DEL TFG: SISTEMA SENSOR**  **TITULACIÓ: Grau en Enginyeria de Sistemes de Telecomunicació**  **AUTOR: Alejandro García Moreno**  **DIRECTOR: Francesc Josep Robert Sanxis**  **DATA: 23 de setembre del 2016** |

Índice

[Capítulo 0. Abstract 4](#_Toc459417767)

[Capítulo 1. Especificaciones del proyecto 5](#_Toc459417768)

[1.1 Lista de especificaciones 5](#_Toc459417769)

[1.1.1 Sistema sensor capaz de medir magnitudes relacionadas con los cultivos en invernaderos 5](#_Toc459417770)

[1.1.2 Implementación del sistema sensor y de control de actuadores con un MCU 5](#_Toc459417771)

[1.1.3 Comunicación inalámbrica con nuestro sistema sensor 5](#_Toc459417772)

[1.1.4 Sistema sensor autónomo y de bajo consumo 5](#_Toc459417773)

[1.1.5 Monitorización de los parámetros del invernadero con una aplicación de Labview 5](#_Toc459417774)

[1.2 Teoría general 5](#_Toc459417775)

[1.2.1 Magnitudes físicas a medir 5](#_Toc459417776)

[1.2.2 Comunicación inalámbrica 5](#_Toc459417777)

[1.2.3 Energy harvesting 6](#_Toc459417778)

[1.2.4 Actuadores a utilizar 6](#_Toc459417779)

[1.3 Diagrama general 6](#_Toc459417780)

[Capítulo 2. Planificación 7](#_Toc459417781)

[2.1 Fase 1 del proyecto 7](#_Toc459417782)

[2.1.1 Interrupción INT0IF 7](#_Toc459417783)

[2.1.2 Muestreo del ADC con el Timer3 7](#_Toc459417784)

[2.2 Fase 2 del proyecto 7](#_Toc459417785)

[2.2.1 Comunicación con el sensor DHT22 7](#_Toc459417786)

[2.2.2 Comunicación I2C 7](#_Toc459417787)

[2.3 Fase 3 del proyecto 7](#_Toc459417788)

[2.3.1 PWM 7](#_Toc459417789)

[Capítulo 3. Desarrollo 8](#_Toc459417790)

[3.1 Desarrollo de la Fase 1 8](#_Toc459417791)

[3.1.1 Interrupción INT0IF 8](#_Toc459417792)

[3.1.2 Muestreo del ADC con el Timer3 8](#_Toc459417793)

[3.2 Desarrollo de la Fase 2 8](#_Toc459417794)

[3.2.1 Comunicación con el sensor DHT22 9](#_Toc459417795)

[3.2.2 Comunicación I2C 9](#_Toc459417796)

[3.3 Desarrollo de la Fase 3 9](#_Toc459417797)

[3.3.1 PWM 9](#_Toc459417798)

[Capítulo 4. Implementación del sistema 10](#_Toc459417799)

[4.1 Elección de los sensores 10](#_Toc459417800)

[4.2 Implementación de nuestra aplicación Labview 10](#_Toc459417801)

[4.2.1 Funcionalidades 10](#_Toc459417802)

[4.2.2 Diseño de la aplicación 11](#_Toc459417803)

[4.3 Implementación de sistema sensor en placa de pruebas 12](#_Toc459417804)

[4.3.1 Esquema de conexión de los sensores con el MCU 13](#_Toc459417805)

[4.3.2 Esquema de conexión PICKIT y UART-BLUETOOTH 14](#_Toc459417806)

[4.3.3 Esquema de alimentación de la Development Board 15](#_Toc459417807)

[4.4 Implementación del sistema sensor en PCB con Eagle 17](#_Toc459417808)

[4.4.1 Schematic 17](#_Toc459417809)

[4.4.2 Board 18](#_Toc459417810)

[4.4.3 Mecanizado 20](#_Toc459417811)

[Capítulo 5. Conclusiones 22](#_Toc459417812)

# Abstract

El proyecto trata de implementar un sistema de control de riego y lumínico para un invernadero. Para realizar dicho control, el sistema deberá de medir con sensores diferentes paramentos como la temperatura ambiente, la humedad relativa del aire, la humedad de la tierra y la intensidad lumínica. Una vez medidos estos valores, el sistema sensor deberá de enviar las ordenes correspondientes a los diferentes actuadores y además enviar, vía Wireless, esos datos a nuestra aplicación Labview para ser mostrados por pantalla y poder realizar un estudio de estos parámetros en base al tiempo.

El sistema sensor se tendrá que realizar con un microcontrolador PIC18F de bajo consumo para que el sistema pueda ser autónomo con poca energía de alimentación. El PIC18F será programado en C usando el estilo de programación FSM. Los primeros programas se simularán en el simulador virtual Proteus para poder detectar bugs de una manera más eficaz. Una vez realizadas las simulaciones, el programa será cargado en nuestro PIC físico en nuestra placa de pruebas donde comprobaremos que todo lo simulado funciona.

En la elección de sensores, se intentará trabajar con sensores analógicos y digitales indistintamente. Respecto los sensores con interface digital, será importante realizar un estudio de sus protocolos de comunicación.

Finalmente, después de realizar el prototipo final en nuestra placa de pruebas, se diseñará con Eagle la correspondiente placa de circuito impreso(PCB) de nuestro sistema sensor y ,a poder ser, también se hará el mecanizado de la misma.

# Especificaciones del proyecto

## Lista de especificaciones

El proyecto tiene que cumplir los siguientes objetivos.

### Sistema sensor capaz de medir magnitudes relacionadas con los cultivos en invernaderos

El sistema sensor tiene que ser capaz de medir, tanto con sensores analógicos o digitales, las magnitudes físicas necesarias para, posteriormente, controlar el riego y realizar el control de la radiación solar que reciben las plantas.

### Implementación del sistema sensor y de control de actuadores con un MCU

La obtención de los parámetros medidos se realizarán con un MCU y los actuadores serán controlados por el MCU.

### Comunicación inalámbrica con nuestro sistema sensor

Nos podremos comunicar con el sistema sensor de forma inalámbrica para realizar la monitorización de los parámetros medidos.

### Sistema sensor autónomo y de bajo consumo

Se procurará que el sistema funcione de manera autónoma y que consuma la menor energía posible.

### Monitorización de los parámetros del invernadero con una aplicación de Labview

La aplicación de Labview deberá de mostrar los valores medidos. A poder ser, que los parámetros se puedan representar en una gráfica para poder realizar estudio en base al tiempo.

## Teoría general

Actualmente la mayoría de los sistemas de riego actuales realizan riegos programados, 1 o 2 veces al día según la estación del año. Nuestro sistema sensor pretende controlar de forma autónoma el riego y la radiación solar recibida según las condiciones de entorno en tiempo real y no de forma programada.

Detrás de todo lo que implementaremos existe una base científico-técnica que explicaremos a continuación.

### Parámetros de entorno a medir

Nuestro sistema sensor pretende controlar el riego y el control lumínico de un invernadero de forma autónoma. Para conseguirlo es necesario poder medir las magnitudes físicas necesarias para nuestro cometido.

Control de Riego:

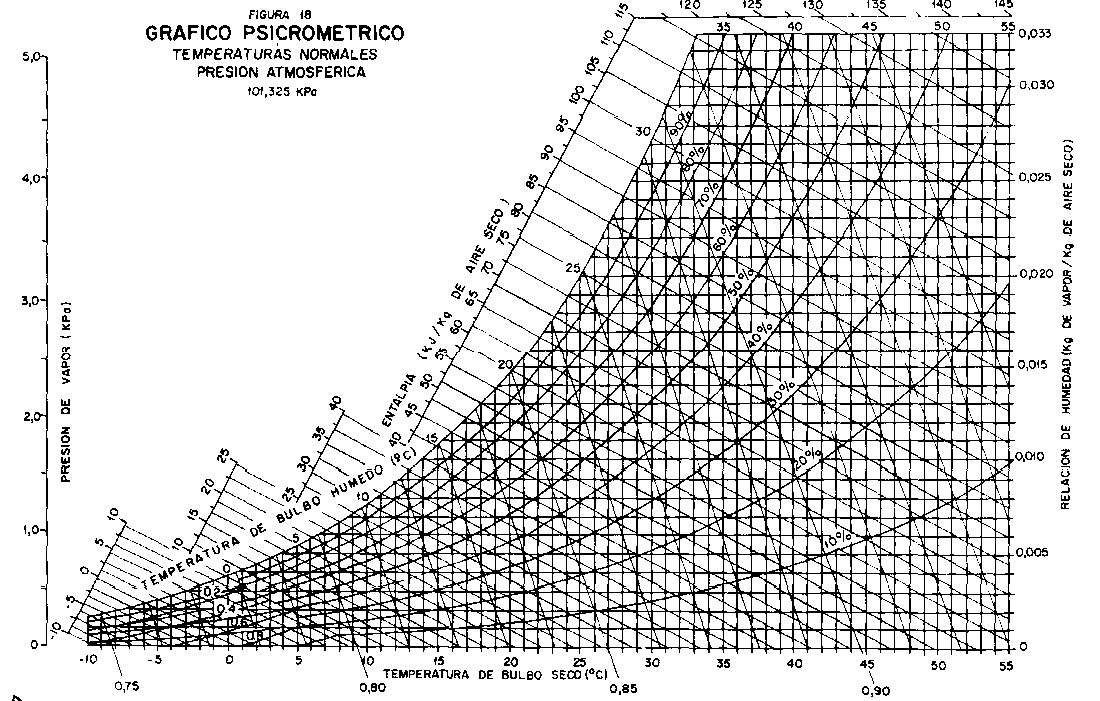
Para realizar el control de riego tendremos en cuenta los parámetros de temperatura ambiente, humedad relativa ambiental y humedad de la tierra. Y según la relación de estos 3 parámetros decidiremos cuándo regar o no.

1. *Temperatura ambiente*

Existen muchos tipos de sensores de temperatura, a continuación mostraremos explicaremos el funcionamiento de los más utilizados para medir la temperatura ambiente.

1. NTC
2. hhsd
3. *Humedad relativa*

La humedad relativa depende de la temperatura ambiente y la cantidad moléculas de H2O que hay en el aire. El sensor de escojamos, además de medir cuantas moléculas de H2O contiene el aire, tendrá que medir la temperatura ambiente para que haga la conversión a humedad relativa.



**Fig. 3.2** Gráfica de la relación humedad relativa-temperatura(www.fao.org)

La **Fig. 3.2** representa que cuando la temperatura ambiente es mayor, mayor es la cantidad de moléculas que puede contener el aire.

1. *Humedad de la tierra*

La humedad de la tierra es la cantidad de moléculas de agua que hay por m2 de tierra pero con un sensor eso no lo podemos medir. Lo que sí que podemos hacer es medir la conductividad de la tierra.

[dibujo]

Control lumínico:

Para realizar el control lumínico tendremos en cuenta la Intensidad lumínica que recibimos del sol.

1. *Intensidad lumínica*

La unidad del sistema internacional de la intensidad lumínica es la candela (Iv, símbolo cd). Esta unidad de medida es proporcional a lumens por estereorradianes (Iv=lm/sr). Pero la unidad que nos será más cómoda para trabajar será la iluminancia (Ev, símbolo lux), en la cual Ev es proporcional a lumens por metro cuadrado(Ev =lm/m2).

Un LDR es una resistencia que varia su resistividad según la luz que capte. Un fabricante de LDRs nos suele proporcionar la relación resistividad-iluminancia(Ω/lux), con lo cual obtener la iluminancia es relativamente sencillo midiendo la resistividad.



**Fig. 3.2** Gráfica Lux-Resistencia de un LDR comercial(Módelo GL5528)

### Comunicación inalámbrica

Una vez configurada la comunicación serial del microcontrolador con un baudrate de 9600, conectaremos al MCU un modulo UART-Bluetooth[Anexo] para poder comunicarnos de manera inalámbrica con un PC. Labview recibirá los datos por un puerto COM virtual a través de la conexión bluetooth.



**Fig. 2.3** Esquema de la comunicación MCU-Labview(Visio)

Una vez se reciban los datos en la aplicación, la idea consiste en hacer aplicación dibuje una gráfica.

### Energy harvesting

Por desarollar

### Actuadores a utilizar

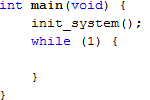
Por desarrollar

## Diagrama general

Esquema general del sistema.[por hacer]

# Planificación

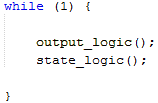
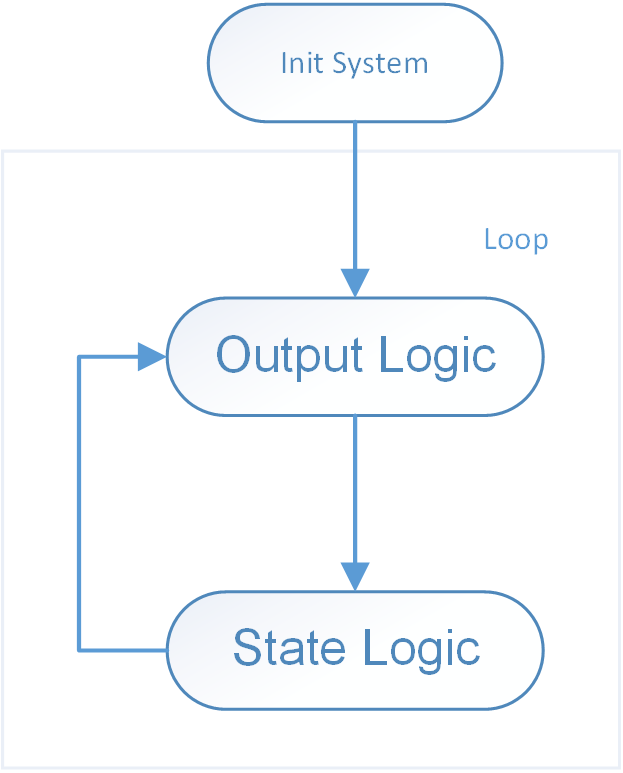
El programa principal tendrá una estructura idéntica a la de un programa Arduino: Setup() y Loop(). Usando el MPLAB-IDE, el programa principal se ejecuta en un *int main(void)* modo que tendremos que emplear una estructura de programa como la siguiente.



**Fig. 4.1** Programa principal

De esta manera, init\_system sería Setup() y while(1){} el Loop().

El programa en C debe seguir el método de programación FSM. Por lo tanto, dentro del loop tendremos 2 funciones ejecutándose para que una se encargue de controlar el cambio de estado(output\_logic()) y otra para ejecutar las tareas del estado(state\_logic()).

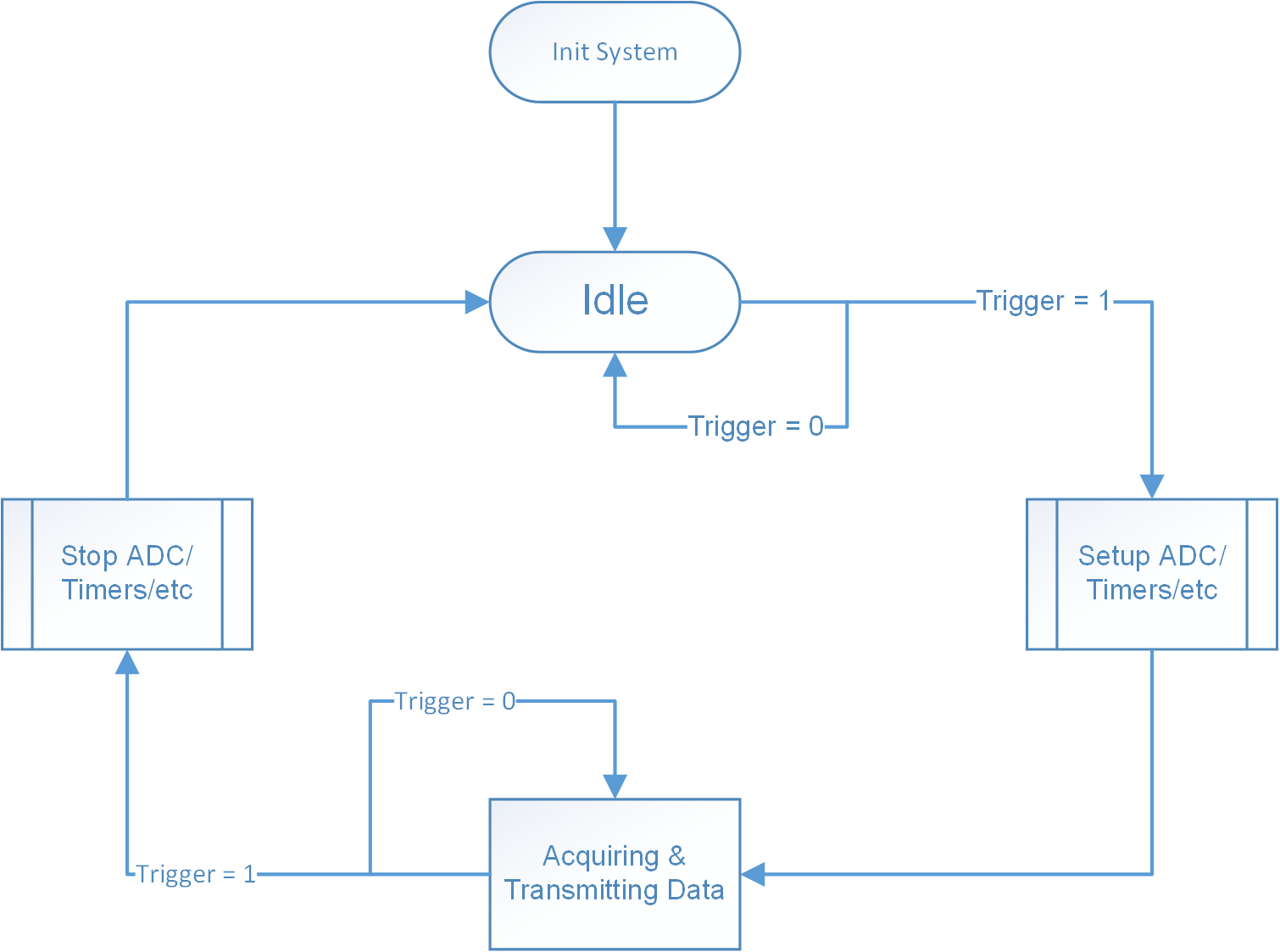


**Fig. 4.2** Loop

## Fase 1 del proyecto

Estados(**Fig. 2.2**):

1. Idle
2. Setup ADC/Timers/etc
3. Acquiring & Transmiting Data
4. Stop ADC/Timers/etc



**Fig. 2.2** Patrón de diseño de nuestra máquina de estados

### Interrupción INT0IF

datasheet

### Muestreo del ADC con el Timer3

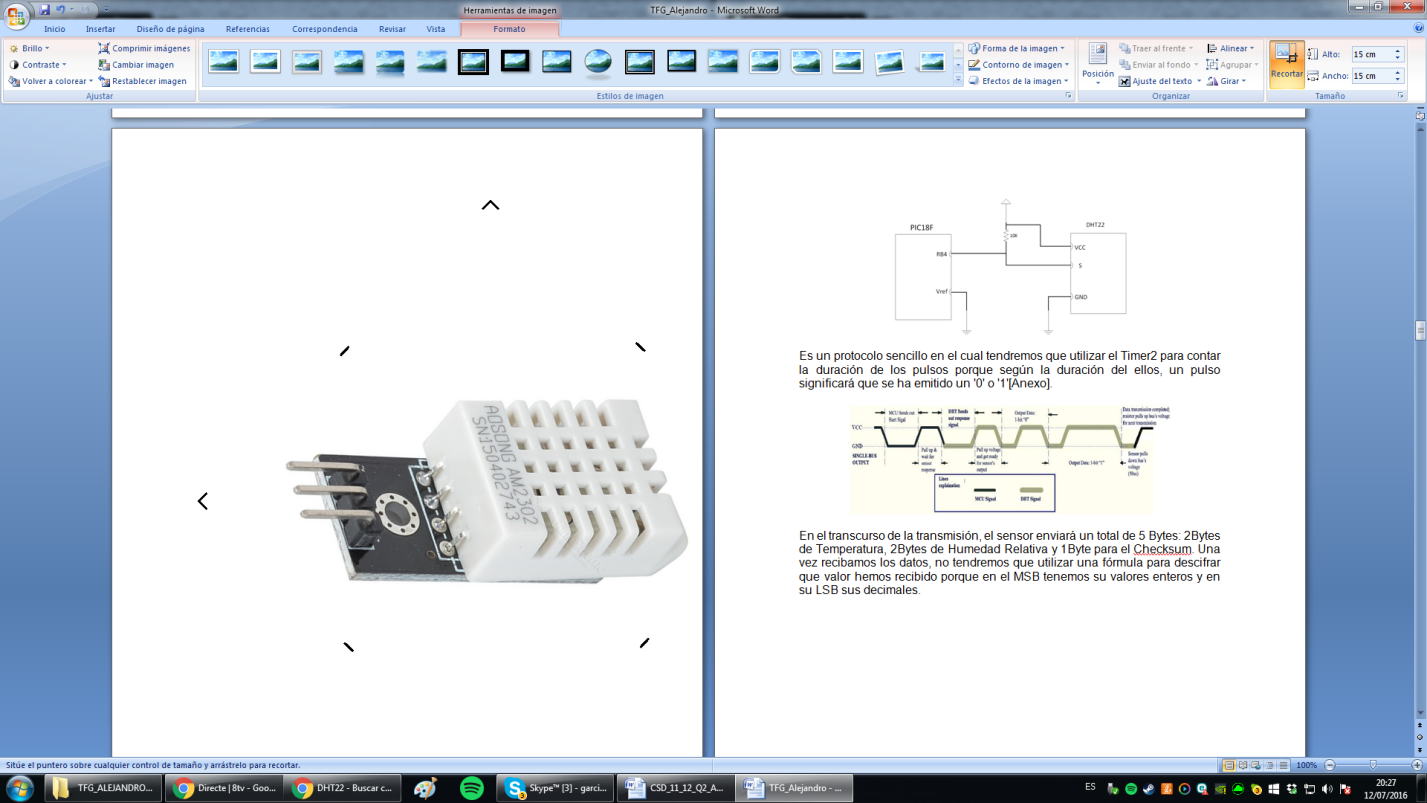
Datasheet adc y timer 3

## Fase 2 del proyecto

Diagrama de estados

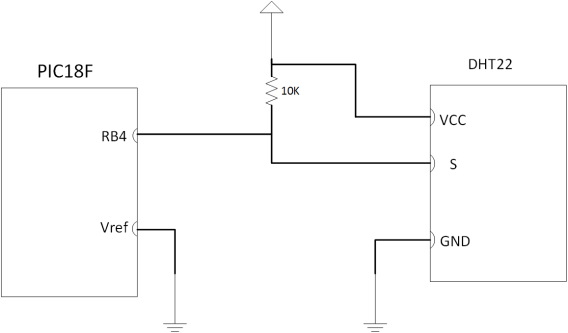
### Comunicación con el sensor DHT22

El DHT22 es un sensor de temperatura y humedad relativa con interface digital. La comunicación de realizará utilizando sólo un pin del MCU y se realizará de forma asíncrona porque no tenemos un clock como en la comunicación I2C.



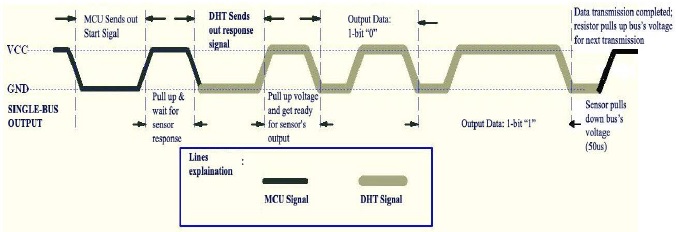
**Fig. 3.13** Sensor DHT22

El sensor utiliza un protocolo propio en el cual se nos explica en el datasheet[Anexo]. Además nos muestra cómo debemos conectar el DHT22 con el MCU.



**Fig. 3.14** Esquema que de conexión del sensor DHT22

Es un protocolo sencillo en el cual tendremos que utilizar el Timer2 para contar la duración de los pulsos porque según la duración del ellos, un pulso significará que se ha emitido un '0' o '1'[Anexo].



**Fig. 3.15** Esquema de la comunicación con el sensor DHT22

En el transcurso de la transmisión, el sensor enviará un total de 5 Bytes: 2Bytes de Temperatura, 2Bytes de Humedad Relativa y 1Byte para el Checksum. Una vez recibamos los datos, no tendremos que utilizar una fórmula para descifrar que valor hemos recibido porque en el MSB tenemos su valores enteros y en su LSB sus decimales.

### Comunicación I2C

## Fase 3 del proyecto

Diagrama de estados

### PWM

comunicación con sensores digitales

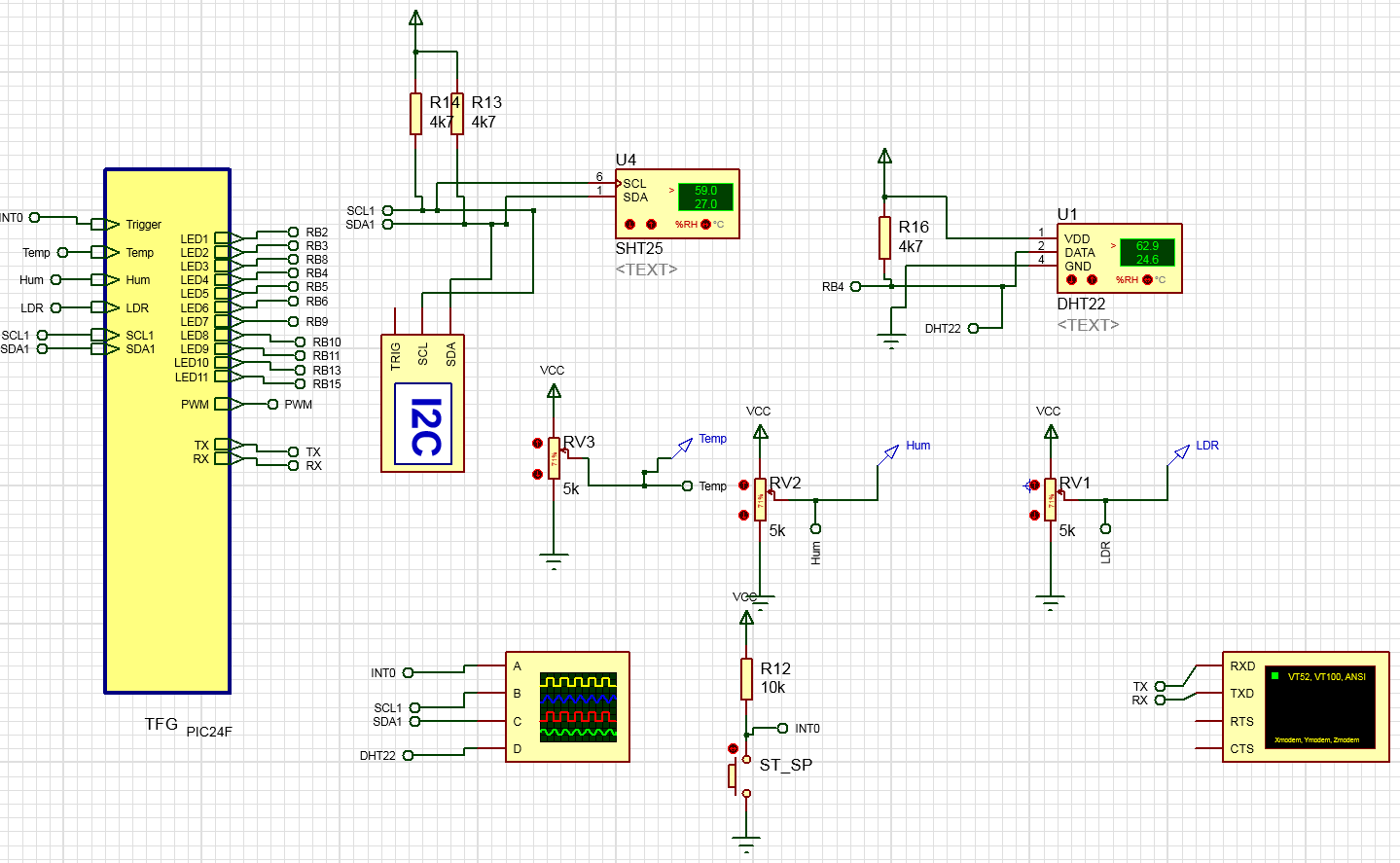
### Comunicación serial

Datasheet configuración del baudrate y puesta en on.

# Desarrollo

Estructura del programa principal 2/2

Para realizar el desarrollo de las diferentes fases de planificación hemos utilizado el simulador Proteus. El simulador Proteus nos permite simular nuestro PIC18F26K20, algunos sensores digitales, la comunicación UART y otros elementos que nos han ayudado a probar el ADC y la comunicación I2C. Además, Proteus nos permite realizar debugging y de esta manera poder encontrar rápidamente soluciones a un bugs en el código.



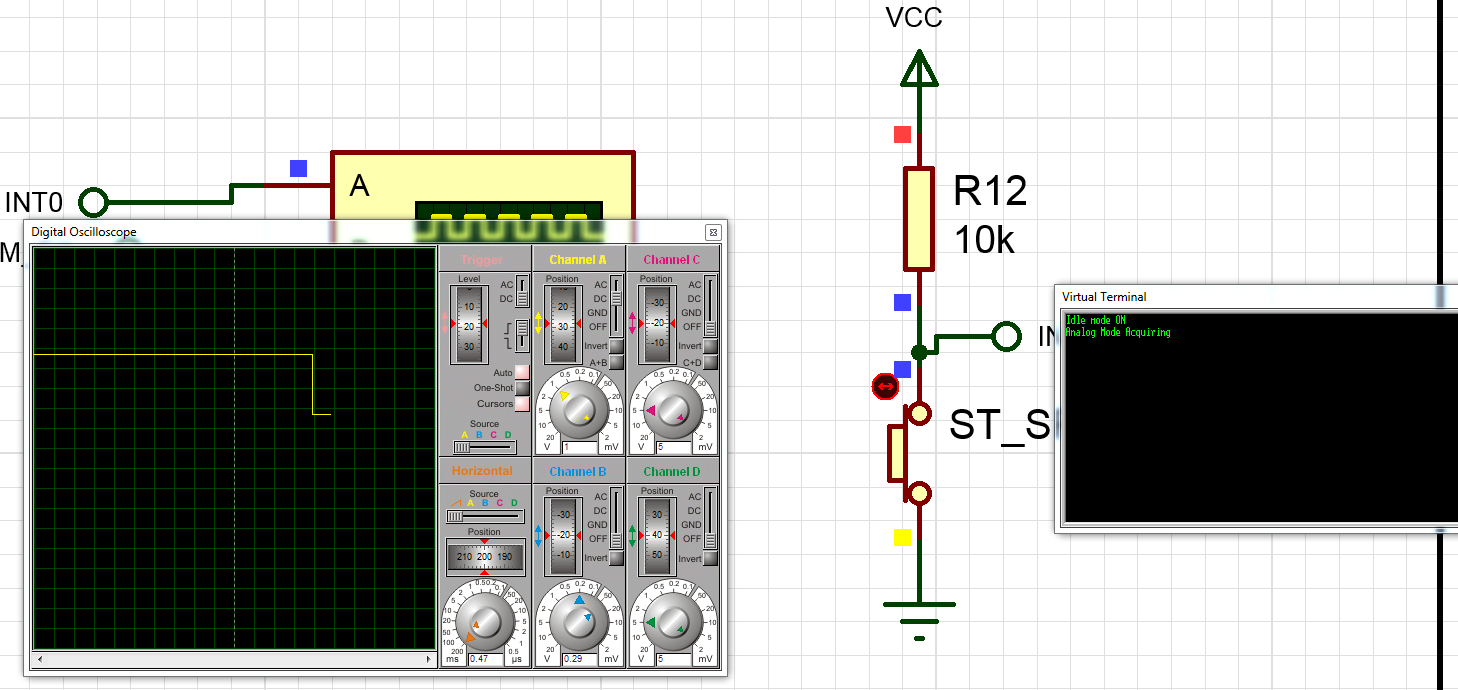
**Fig. 3.1** Esquemático de nuestro proyecto Proteus

## Desarrollo de la Fase 1

[Implementación en C] en los anexos

### Interrupción INT0IF

La entrada INT0 de nuestro PIC18 la tenemos configurada para que la interrupción INT0IF se active cuando detecte un flanco de bajada. Para comprobar que funciona de la manera esperada también podemos utilizar el método el breakpoint pero esta vez hemos utilizado el osciloscopio y el Virtual Terminal.

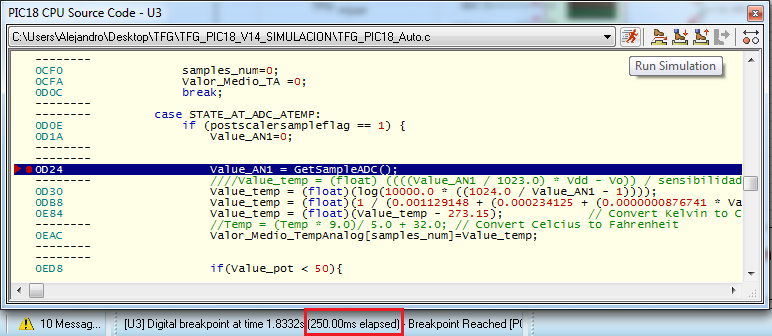


**Fig. 5.3** Simulación de la interrupción INT0

Cómo podemos ver en la **Fig. 5.3**, cuando hay un flanco de bajada en INT0, el PIC envía un mensaje por comunicación UART dejando de estar en el estado Idle.

### Muestreo del ADC con el Timer3

Hemos diseñado el sistema sensor para que tome 4 muestras en 1 segundo, por lo tanto cada vez el entre muestra y muestra deben de pasar 250ms. Utilizando el debugger de Proteus, hemos puesto un breakpoint en la función *GetSampleADC()* sabiendo que en el estado *Acquiring & Transmission ADC Analog Temp* la llamará 4 veces. Proteus nos proporciona el tiempo entre breakpoint, con lo cual tenemos que comprobar si ese intervalo es equivalente a lo previamente definido en nuestro diseño. [Ver]



**Fig. 5.2** Simulación de la interrupción TMR3IF para realizar el periodo de muestreo

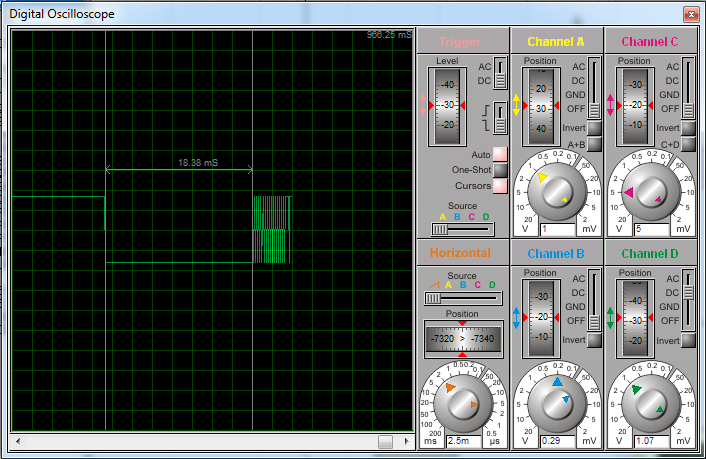
## Desarrollo de la Fase 2

[Implementación en C] los anexos

### Comunicación con el sensor DHT22

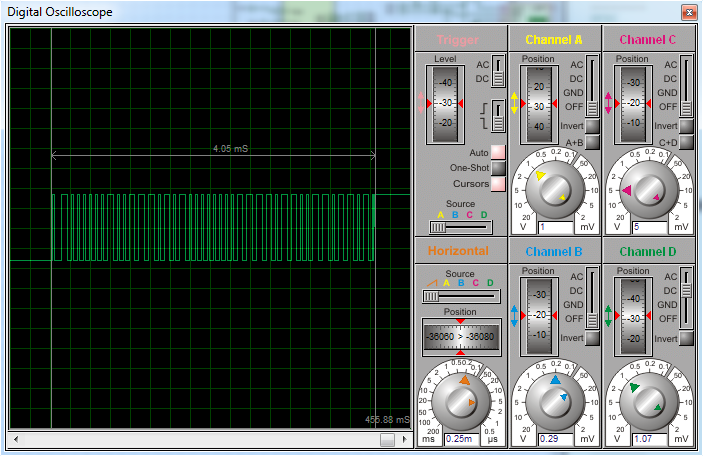
Con el osciloscopio hemos podido comprobar que el formato de trama en la trasmisión de información es igual que en el datasheet del sensor.

En primer lugar tenemos una captura de la señal de inicio de comunicación en el cual el sensor envía una señal de pull down de 18ms y una de pull up hasta que responde el sensor.(Ver **Fig. 5.4**)



**Fig. 5.4** Señal de Start del MCU

En segundo lugar comprobamos que los bits de datos se envían en pulsos cortos '0' y pulsos largos '1'.(Ver **Fig. 5.5)**



**Fig. 5.5** Transmisión de datos del DHT22

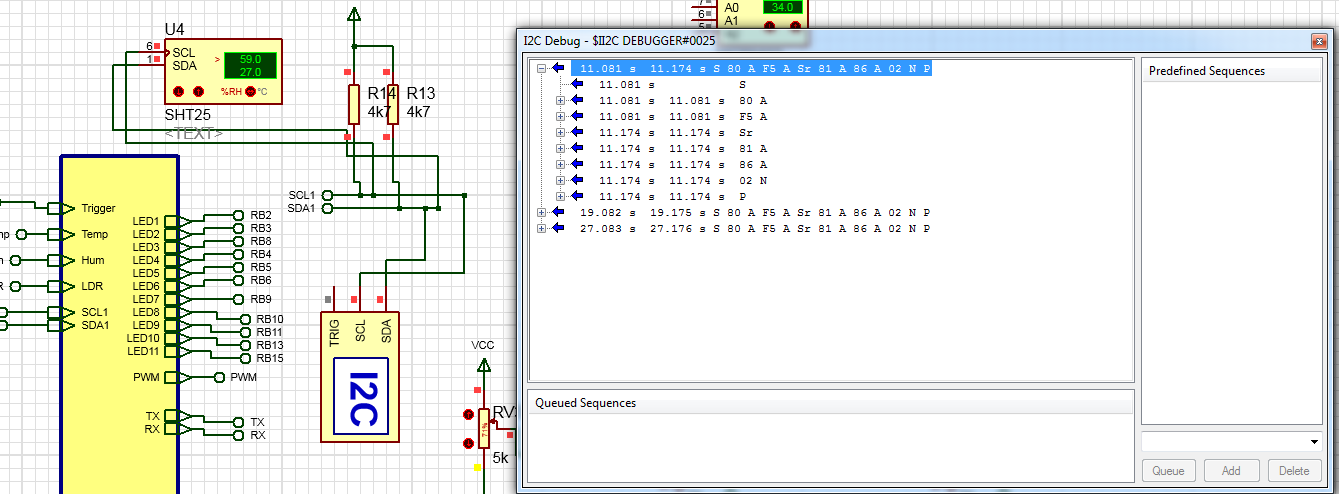
### Comunicación I2C

[Dividir entre planning y desarrollo]

Proteus no contiene el modelo MCP9808 para ser simulado, en su lugar hemos probado el sensor SHT25. El sensor SHT25 tiene la siguiente dirección I2C: '1000000'. Al sensor se le puede solicitar que nos dé los datos de temperatura o humedad relativa. En la simulación que hemos realizado, hemos solicitado los datos de humedad relativa con el comando '11110101' proporcionado en el datasheet del fabricante[Anexo SHT25].

Proteus nos permite hacer debug de la comunicación I2C y esta manera podemos ver que el formato de trama es como explicamos[Apartado I2C, Anexo].

Configuramos el SHT25 con una humedad relativa del 59% y el debugger nos reporta lo siguiente:(ver **Fig. 5.6**)



**Fig. 5.6** Debug de la comunicación I2C

Report en Hexadecimal: [**S 80** A **F5** A **Sr** **81** A 86 **A** 02 **N** **P**]. En Negrita lo enviado por el MCU y en normal lo enviado por el sensor. El formato de trama del protocolo I2C nos indica que el MSB de la información es '86' y el LSB es '02'.

El datasheet del SHT25 por proporciona la fórmula que nos convierte el código recibido a humedad relativa.[Fórmula]

**(5.1)**

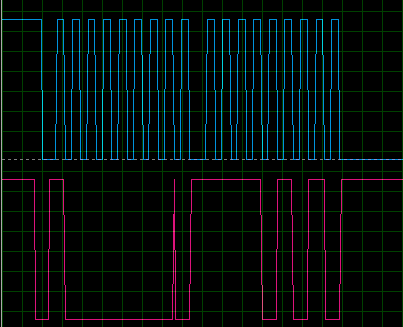
Srh es 8602 en hexadecimal y en decimal es 34306.

**(5.2)**

Por lo tanto comprobamos que funciona bien la comunicación.

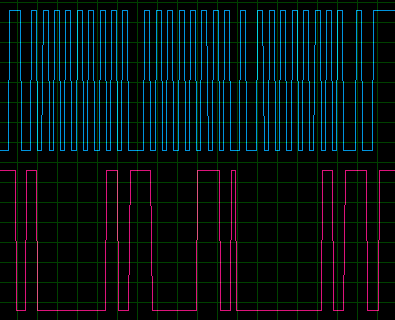
Otro aspecto a analizar de la comunicación I2C es el análisis de la trama en el osciloscopio. Primero mostraremos una captura de la primera parte de la comunicación y luego la segunda.

En la primera parte de la comunicación avisamos al sensor de que datos queremos que nos envíe y esperamos un tiempo para que el sensor tenga tiempo de procesar los datos(en nuestro diseño esperamos 90ms). [Anexo]



**Fig. 5.7** Inicio de la trasmisión I2C: **S 80** A **F5** A; azul clock, rojo SDA

En la segunda parte de la comunicación avisamos al sensor de que estamos preparados para recibir los datos.[Anexos]



**Fig. 5.8** Recepción de los datos: **Sr** **81** A 86 **A** 02 **N** **P**; azul clock , rojo SDA

## Desarrollo de la Fase 3

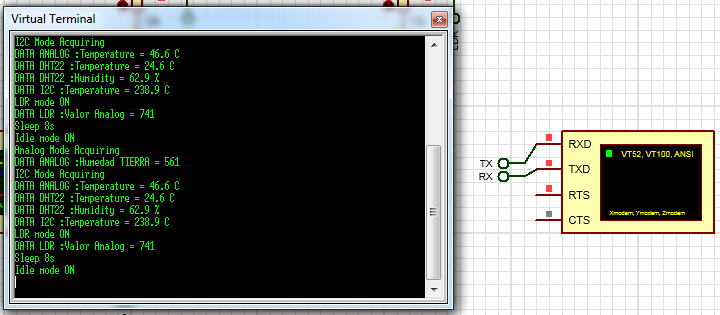
[Implementación en C]

### PWM

pwm acuador

### Comunicación serial

El Virtual Terminal de Proteus nos permite visualizar los mensajes que se transmiten y reciben por el puerto serie. (Ver **Fig. 5.9**)



**Fig. 5.9** Virtual terminal

La aplicación de Labview realizará un parsing de los mensajes que transmite el MCU para obtener los datos de los sensores.

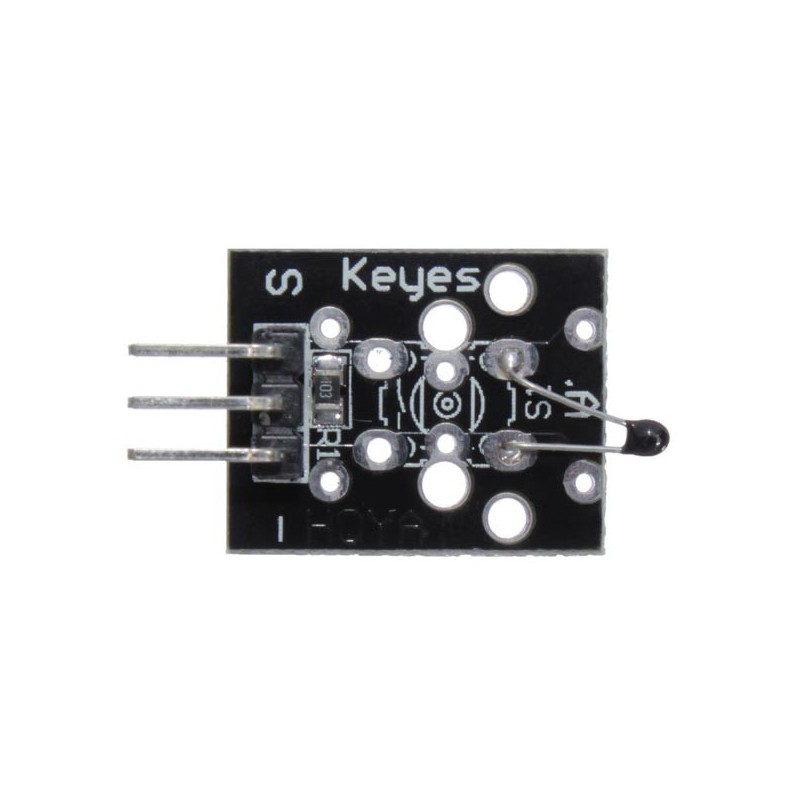
# Implementación del sistema

## Elección de los sensores

Para tomar muestras de las diferentes señales que tenemos que medir utilizaremos tanto sensores con interface de salida analógica como digital, con el fin de estudiar diferentes tipos de sensores y protocolos de comunicación.

### Temperatura Analog KY-013

El KY-013 es un termistor NTC, con lo cual su resistencia varía respecto la temperatura siguiendo un comportamiento exponencial. [foto sensor]

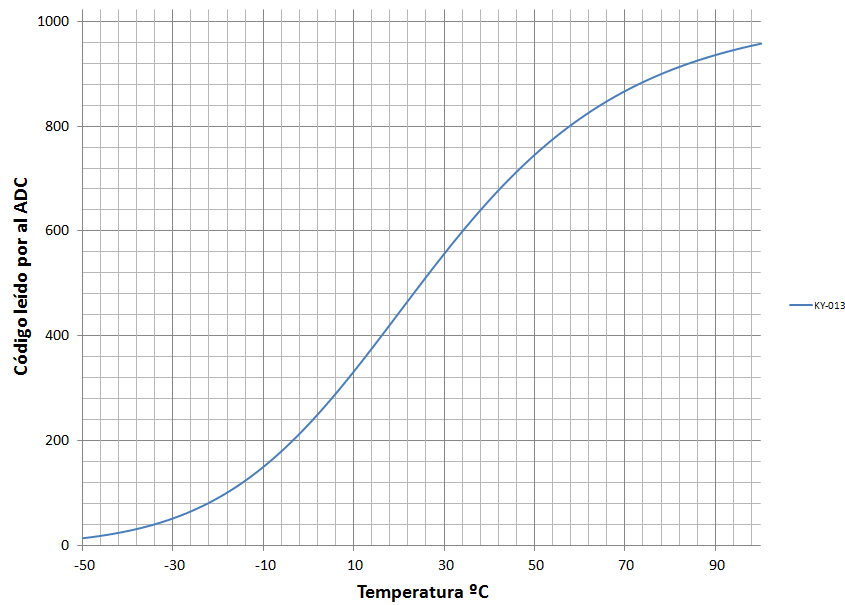


**Fig. 3.2** Módulo de Temperatura ambiental KY-013

El fabricante nos proporciona la fórmula(**3.2**) que nos convierte el código leído por el ADC y Temperatura.[Anexo]

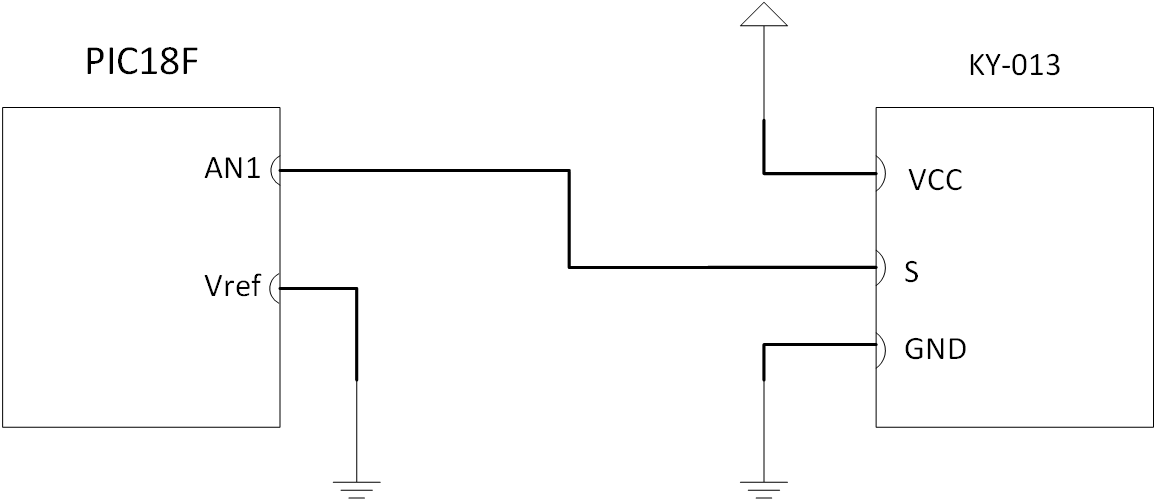
**(3.1)**

**(3.2)**



**Fig. 3.3** Gráfica código ADC-temperatura KY-013

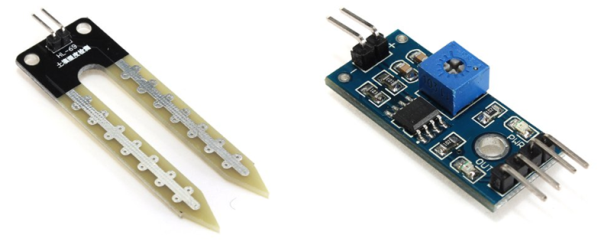
Además el fabricante nos explica cómo conectar el sensor a nuestro MCU. El KY-013 tiene 3 pines: 1 para ser conectado a VCC, 1 para conectar a GND y 1 para conectar la salida analógica(S) a una entrada del ADC(AN1).[figura]



**Fig. 3.3** Esquema de conexión del KY-013 (Visio)

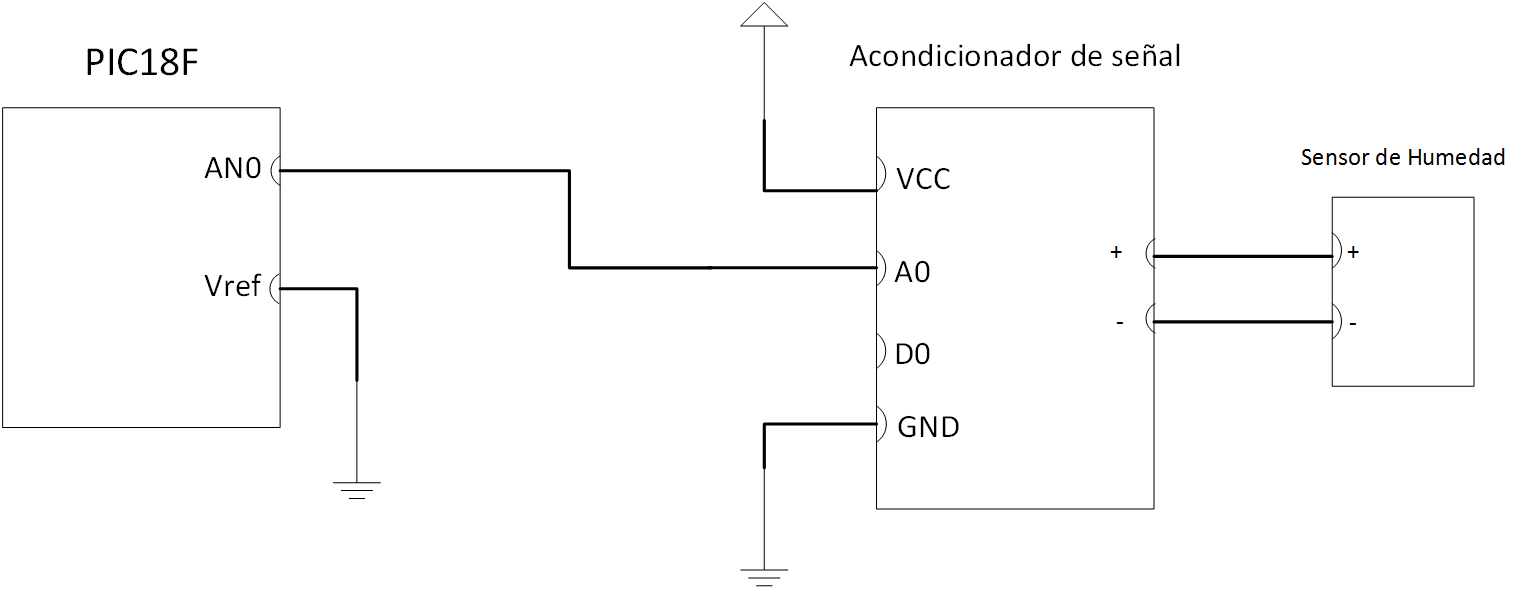
### Humedad Tierra Analog

El HTA es un sensor que detecta la conductancia de la tierra. El HTA tiene integrado un acondicionador de señal para que podamos medir la humedad de la tierra desde Vref hasta VCC con el ADC. Además el sensor tiene una salida digital, que no utilizaremos, que se pone en on/off cuando la tensión de salida del A0 es VCC/2.[Anexo][foto sensor]



**Fig. 3.4** A la izquierda el sensor de humedad y a la derecha el acondicionador de señal

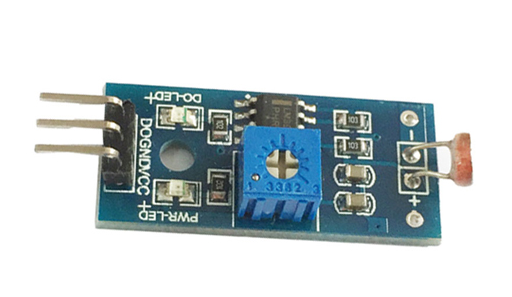
En este caso, conectaremos la salida A0 a la entrada del ADC AN0 y el resto de conexiones tal y cómo se muestra en la figura 1[lea Anexo].



**Fig. 3.5** Esquema que de conexión del módulo sensor de humedad de la tierra (Visio)

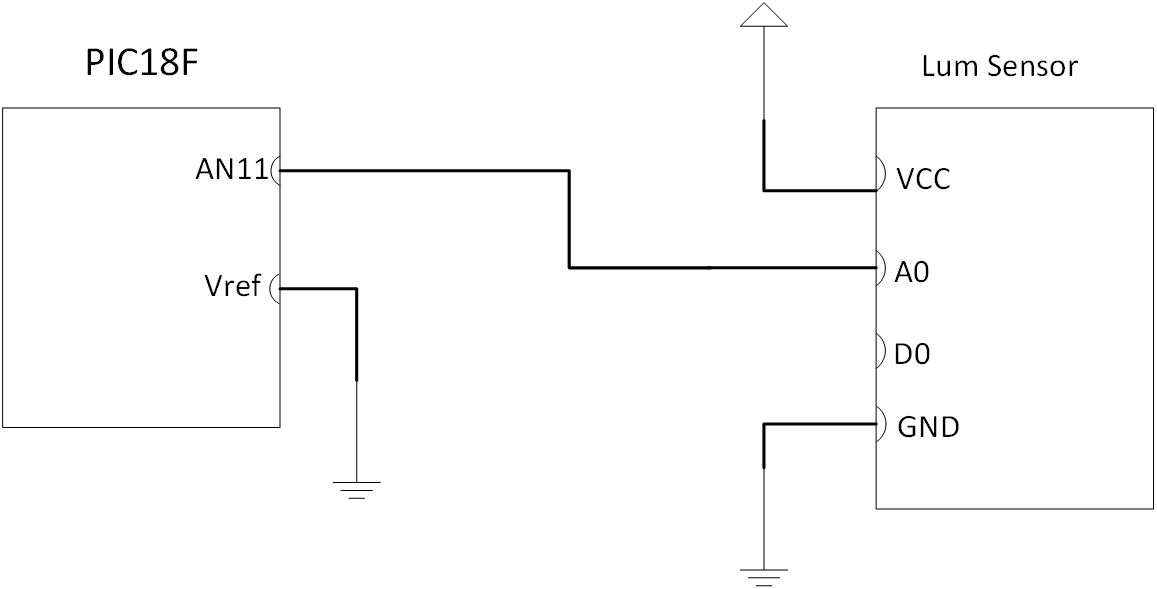
### Iluminancia Analog

El LDR es un sensor que detecta la Iluminancia que capta en su superficie fotosensible y hace variar su resistencia. El sistema sensor del LDR tiene integrado, igual que el HTA, un acondicionador de señal para que podamos medir la humedad de la tierra desde Vref hasta VCC con el ADC. Además el sensor tiene una salida digital, que no utilizaremos, que se pone en on/off cuando la tensión de salida del A0 es VCC/2.[Anexo][foto sensor]



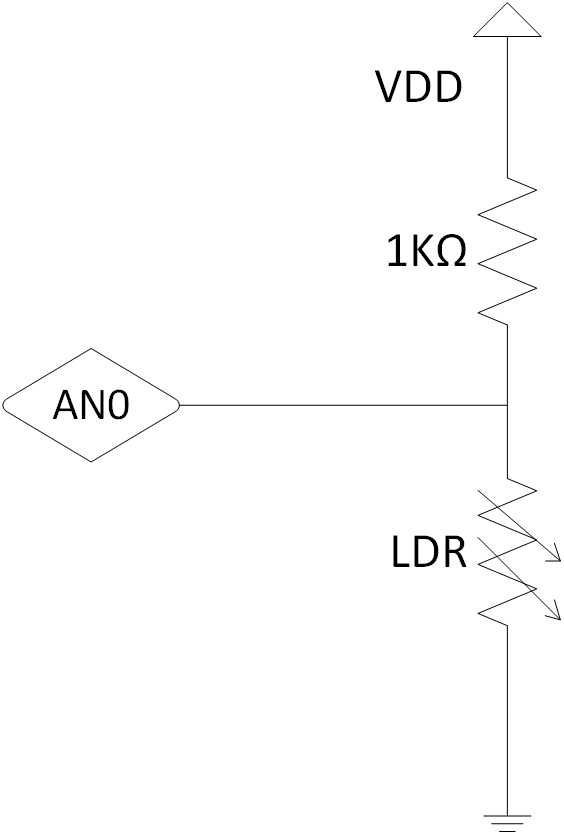
**Fig. 3.6** Módulo sensor de Luz

En este caso, conectaremos la salida A0 a la entrada del ADC AN11 y el resto de conexiones tal y cómo se muestra en la figura 1[lea Anexo].



**Fig. 3.7** Esquema que de conexión del sensor de luz (Visio)

Para conocer la resistencia del LDR debemos tener en cuenta el esquema interno del módulo sensor(fig).



**Fig. 3.7** Estructura del módulo sensor

Siguiendo la estructura del módulo sensor, la resistencia fija de 1KΩ y el LDR forman un divisor de tensión(Fórmula **3.4**). Podemos extraer el valor resistivo del LDR utilizando la siguiente expresión(Fórmula **3.5**):

**(3.4)**

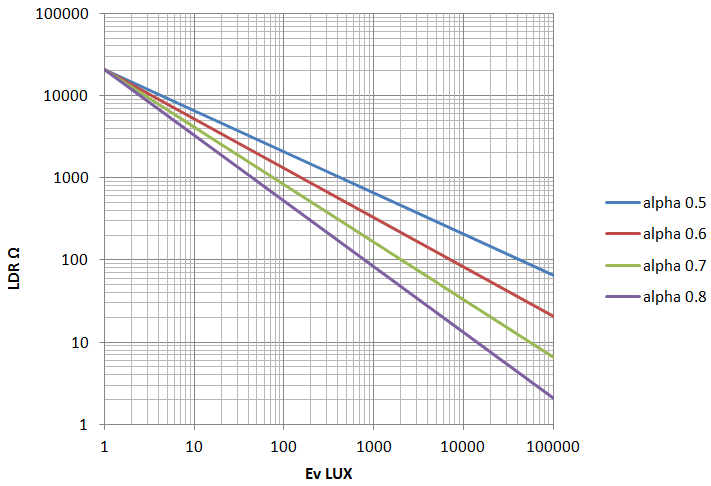
**(3.5)**

El fabricante de nuestro módulo sensor no nos proporciona el modelo de LDR que tenemos. Consecuentemente hemos tenido que caracterizar nuestro LDR basándonos en la forma de caracterización de otros fabricantes.[Anexos]

Dos parámetros son necesarios para tener caracterizado nuestro LDR: La resistencia en la oscuridad y el parámetro α. Nuestro LDR en la oscuridad tiene una resistividad de 21KΩ y el parámetro α debería de proporcionarlo el fabricante pero sabemos los valores típicos de este parámetro que oscilan entre 0.5 y 0.8. Para nuestro diseño hemos escogido 0.6 porque nos daba en la práctica valores de Ev más coherentes(**Fig. 3.8**).

La fórmula para convertir el valor resistivo de nuestro LDR a lux es la siguiente(Fórmula **3.6**):

**(3.6)**



**Fig. 3.8** Valores de Iluminancia para diferentes valores α

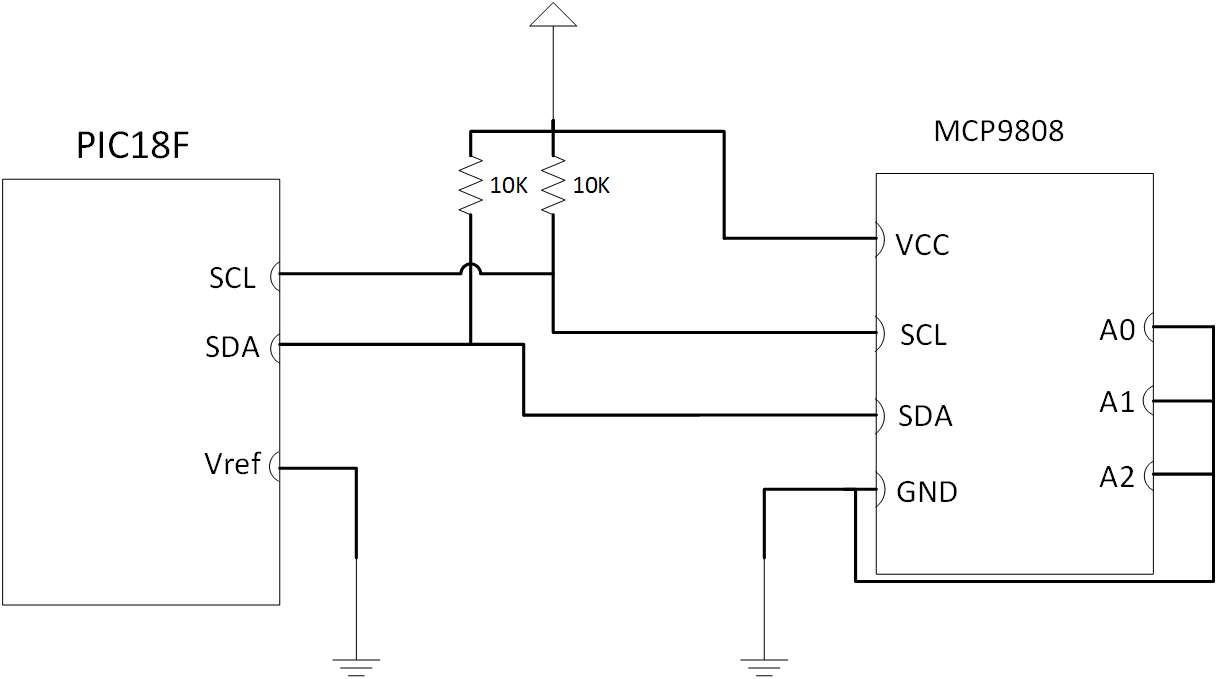
### Temperatura Digital I2C MCP9808

El MCP9808 es un sensor de temperatura con interface digital que utiliza el protocolo de comunicaciones I2C por lo tanto tendremos que configurar la MCU como Master porque el sensor será Slave. [Anexo I2C].



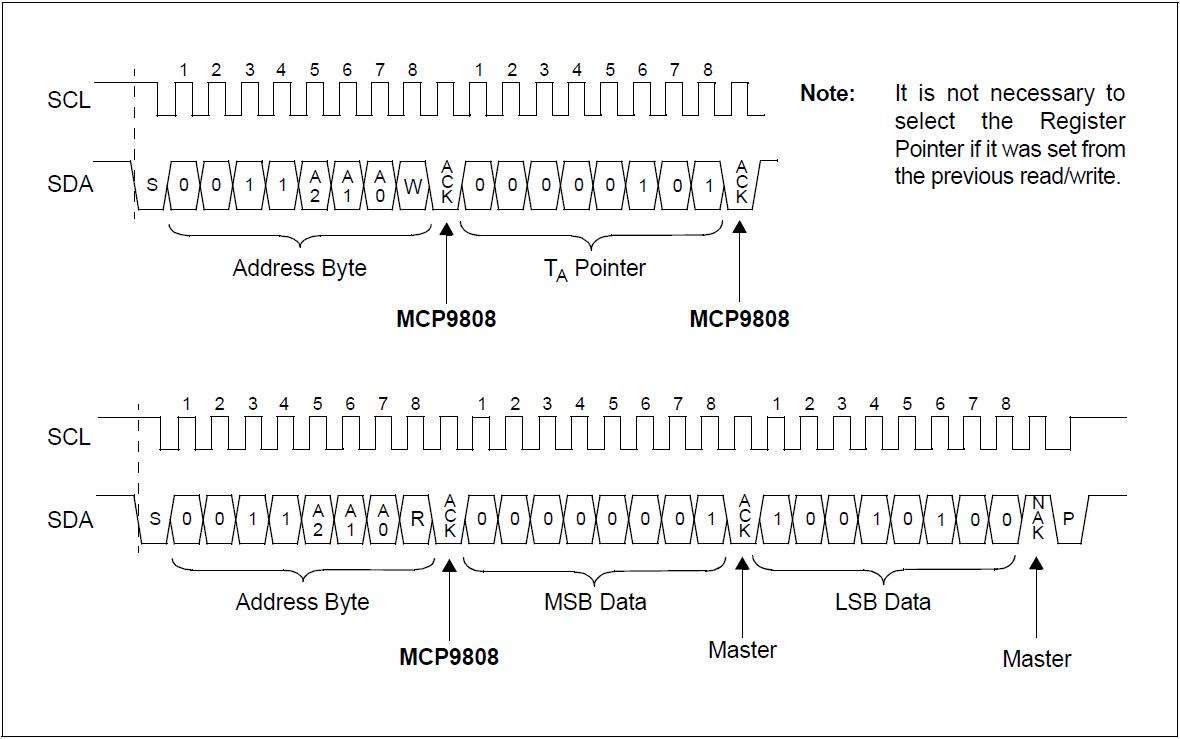
**Fig. 3.9** Sensor MCP9808

El MCP9808 tiene esta dirección I2C 0011XXX donde los últimos 3 bits de la dirección son configurables por vía externa. En el esquema de conexión, los puertos de A0, A1 y A2 están conectados a masa para que sean '0' y por lo tanto la dirección I2C sea 0011000.



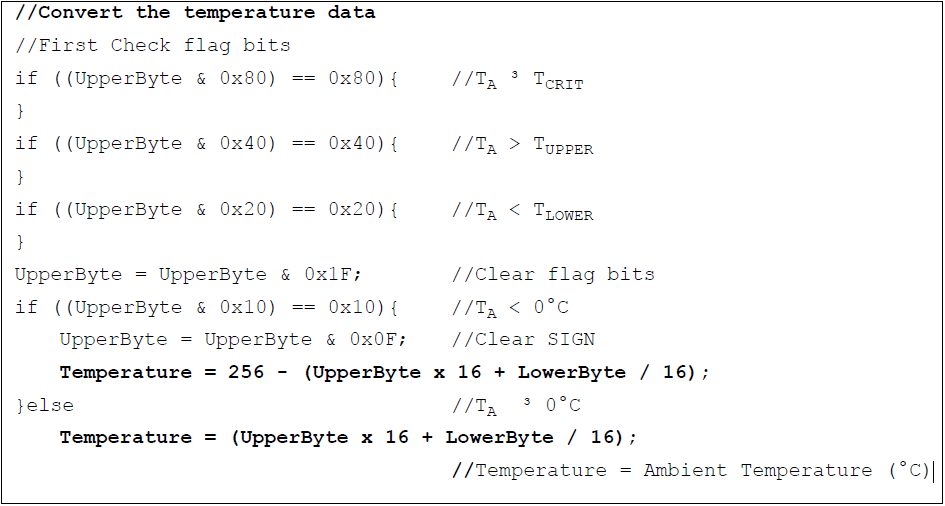
**Fig. 3.10** Esquema que de conexión del sensor MCP9808

Durante la comunicación I2C[Anexo], deberemos de enviar el comando de 8bits '00000101' al sensor para que nos responda proporcionándonos la temperatura.



**Fig. 3.11** Esquema de la comunicación I2C con MCP9808(datasheet)

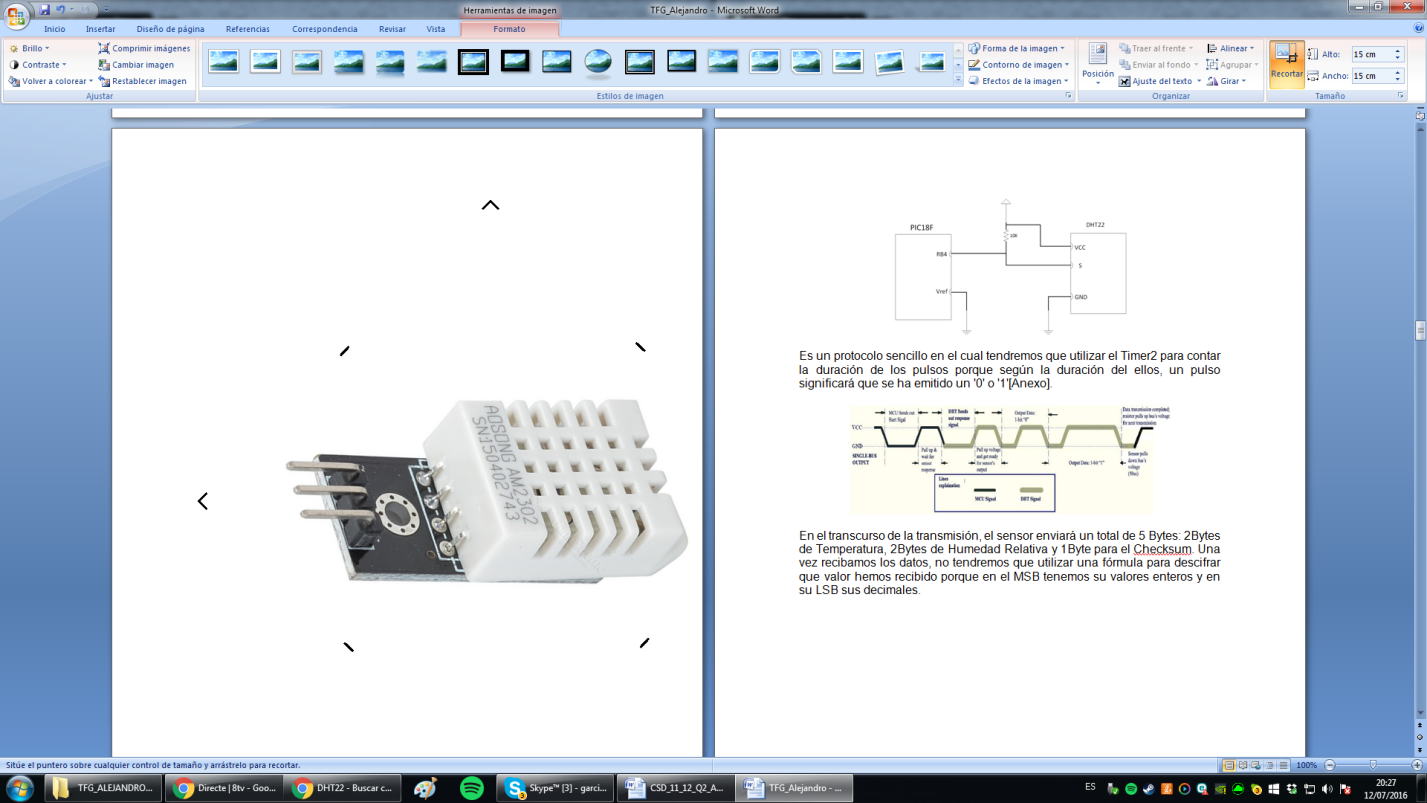
Finalmente el MCP9808 nos responderá con 2 bytes: un MSB y LSB. Entonces, gracias a la fórmula que nos proporciona el fabricante podremos traducir el código a 'ºC'. Sin embargo en el datasheet del fabricante nos proporciona el código C que tenemos que utilizar para hacer la conversión(**Fig. 3.10**).



**Fig. 3.12** Código C para convertirnos los bytes recibidos en Temperatura

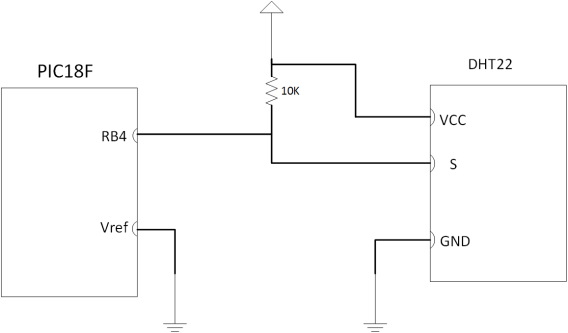
### Temperatura y Humedad Relativa Digital OneWire Protocol DHT22

El DHT22 es un sensor de temperatura y humedad relativa con interface digital. La comunicación de realizará utilizando sólo un pin del MCU y se realizará de forma asíncrona porque no tenemos un clock como en la comunicación I2C.



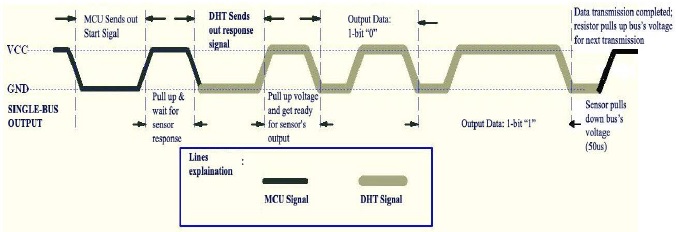
**Fig. 3.13** Sensor DHT22

El sensor utiliza un protocolo propio en el cual se nos explica en el datasheet[Anexo]. Además nos muestra cómo debemos conectar el DHT22 con el MCU.



**Fig. 3.14** Esquema que de conexión del sensor DHT22

Es un protocolo sencillo en el cual tendremos que utilizar el Timer2 para contar la duración de los pulsos porque según la duración del ellos, un pulso significará que se ha emitido un '0' o '1'[Anexo].



**Fig. 3.15** Esquema de la comunicación con el sensor DHT22

En el transcurso de la transmisión, el sensor enviará un total de 5 Bytes: 2Bytes de Temperatura, 2Bytes de Humedad Relativa y 1Byte para el Checksum. Una vez recibamos los datos, no tendremos que utilizar una fórmula para descifrar que valor hemos recibido porque en el MSB tenemos su valores enteros y en su LSB sus decimales.

[Revisar]

## Implementación de nuestra aplicación Labview

Desde nuestra aplicación de Labview podremos recibir los datos y transmitir información por un puerto COM.

### Funcionalidades

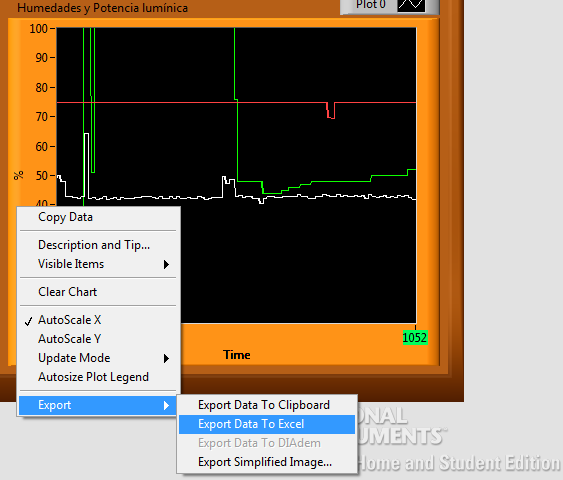
La comunicación aplicación-MCU es bidireccional, desde el Panel de control podemos sacar del modo Sleep a nuestra MCU y que nos envíe los datos de forma instantánea.

Nuestra aplicación realiza un parsing de los mensajes que nos envía el sistema sensor para extraer los datos. Los datos de temperatura los mostramos en los termómetros y los demás datos en medidores en forma de porcentaje.



**Fig. 6.1** Interfaz gráfica de nuestra aplicación Labview

Además, cada dato de los sensores pueden ser dibujados en las gráficas y también podemos exportar las gráficas a Excel.

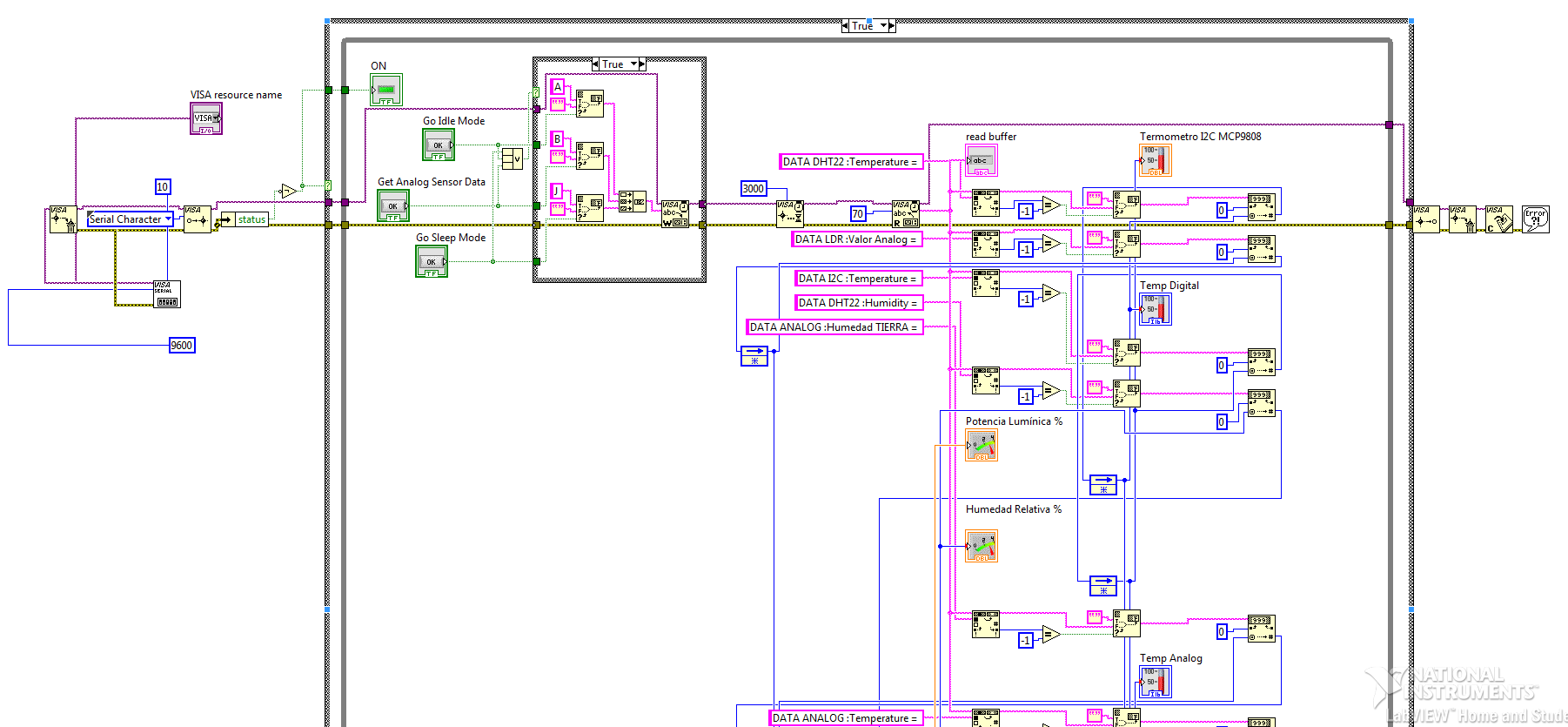


**Fig. 6.2** Exportación de los datos en excel

### Diseño de la aplicación

Labview es una plataforma de diseño de aplicaciones de ingeniería y ciencia pensada para acelerar la productividad de los ingenieros.  Con una sintaxis de programación gráfica que facilita visualizar, crear y codificar sistemas de ingeniería.( http://www.ni.com/labview/esa/)

La creación de la aplicación consiste en ir uniendo módulos y al unirlos formar un diagrama de bloques.(Ver esquemático)

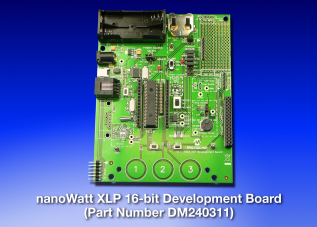


**Fig. 6.3** Esquemático de nuestra aplicación en desarrollo

[Ver explicación detallada de la aplicación ANEXO]

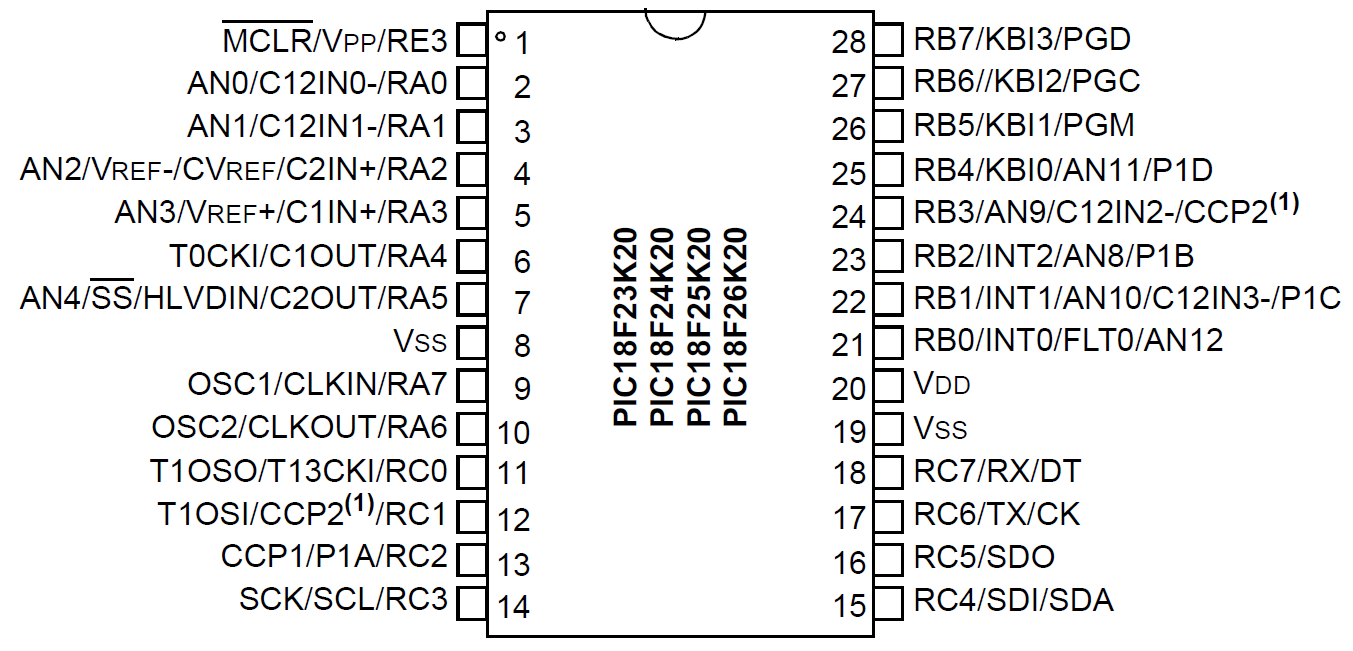
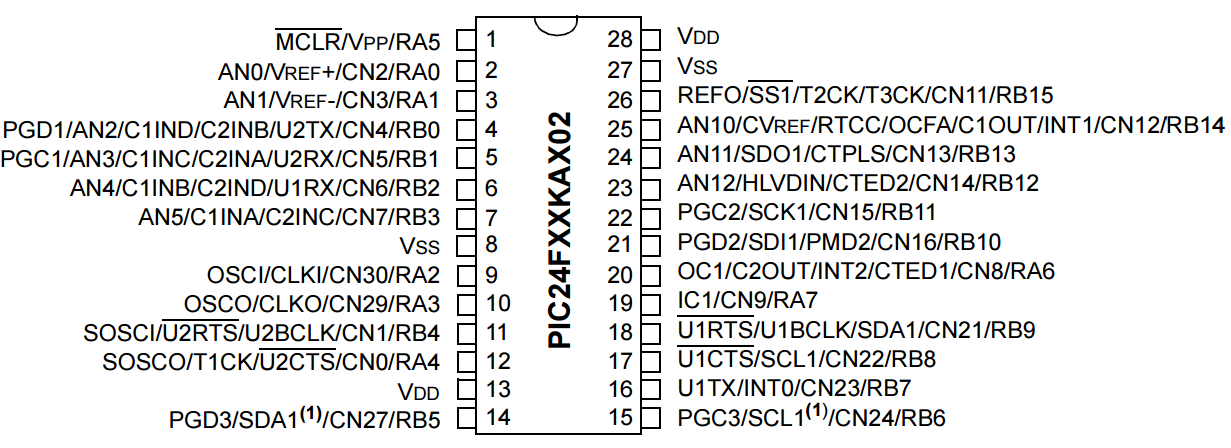
## Implementación de sistema sensor en placa de pruebas

En los inicios del desarrollo del sistema sensor, utilizamos la placa de entrenamiento XLP 16-bits Development Board de Microchip. El mayor inconveniente de trabajar con esta placa era conectar nuestro PIC18 porque la placa está diseñada para un PIC24F16KA102.



**Fig. 7.1** XLP 16-bits Development Board de Microchip

La solución a este problema fue trabajar con una protoboard para poder conectar las I/O del PIC18 con la placa de manera correcta.



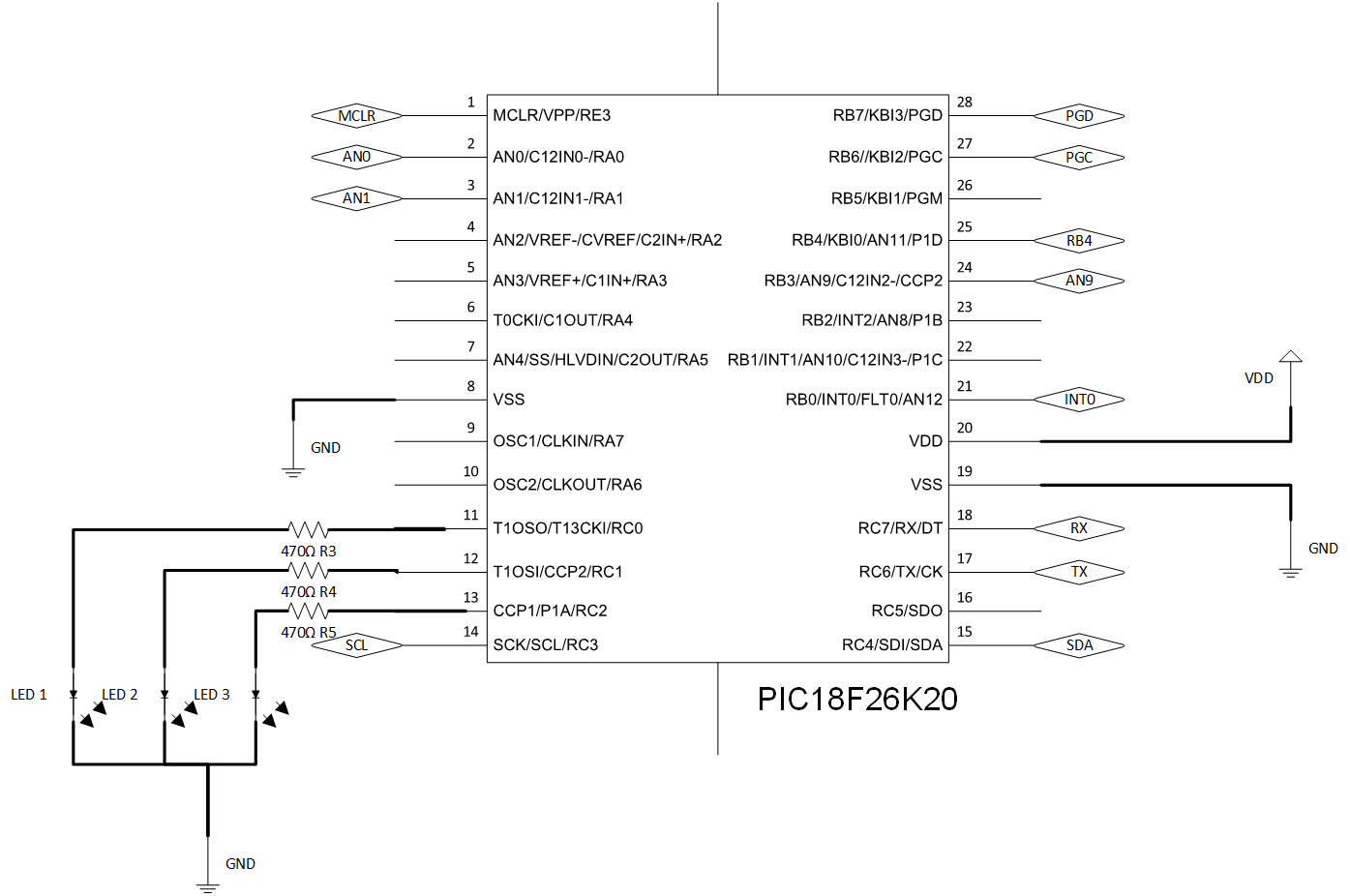
**Fig. 7.2** Diagrama de pines: arriba PIC24F16KA102 y abajo nuestro PIC18F26K20

De este modo fue posible utilizar los dispositivos de la placa de entrenamiento, los más importantes: botones, conector pickit, potenciómetro y circuito de alimentación.



**Fig. 7.3** Implementación del sistema sensor en placa de pruebas

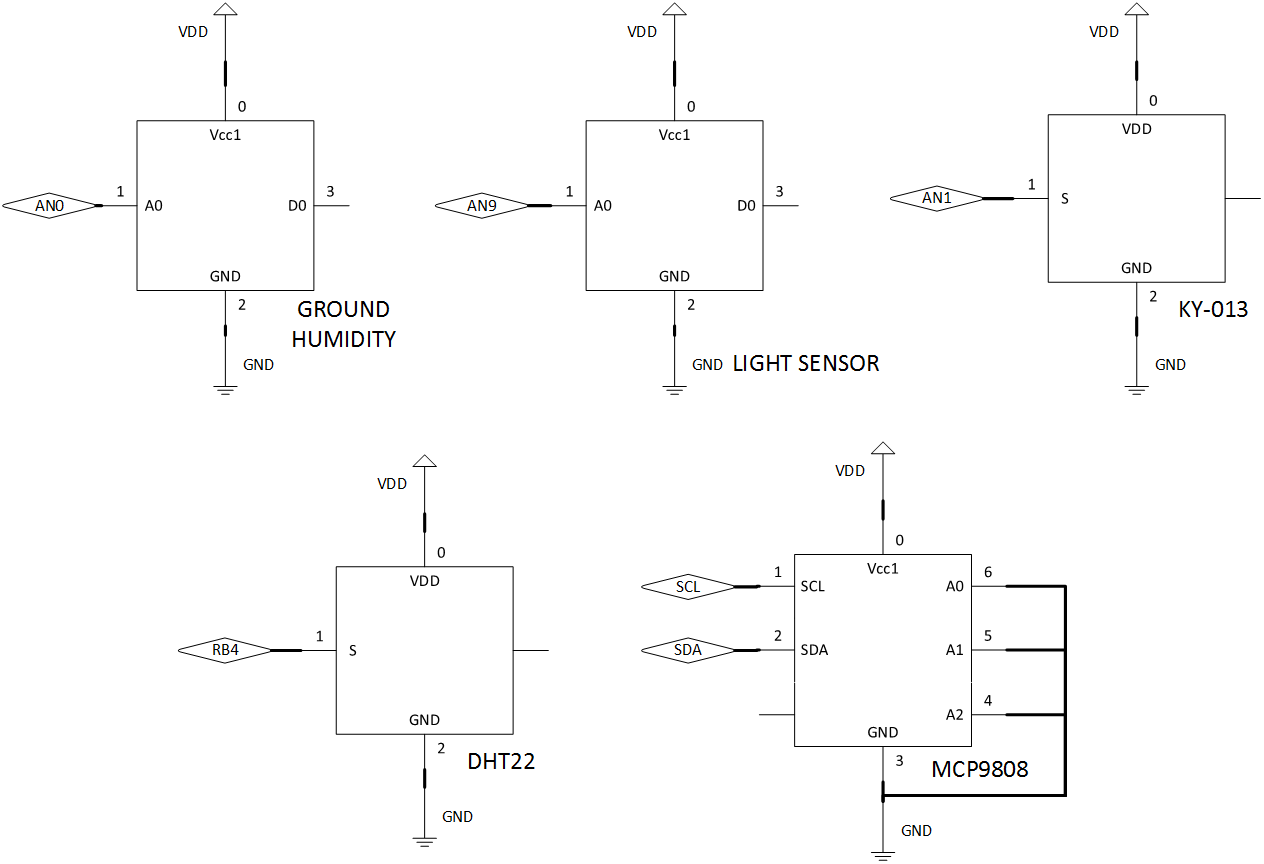
### Esquema de conexión de los sensores con el MCU



**Fig. 7.4** Conexiones con nuestro MCU

Además de conectar los sensores al MCU, hemos conectado unos LEDs en los puertos de salida RC0, RC1, RC2 para tener una referencia del estado en el que se encontraba nuestra máquina de estados.

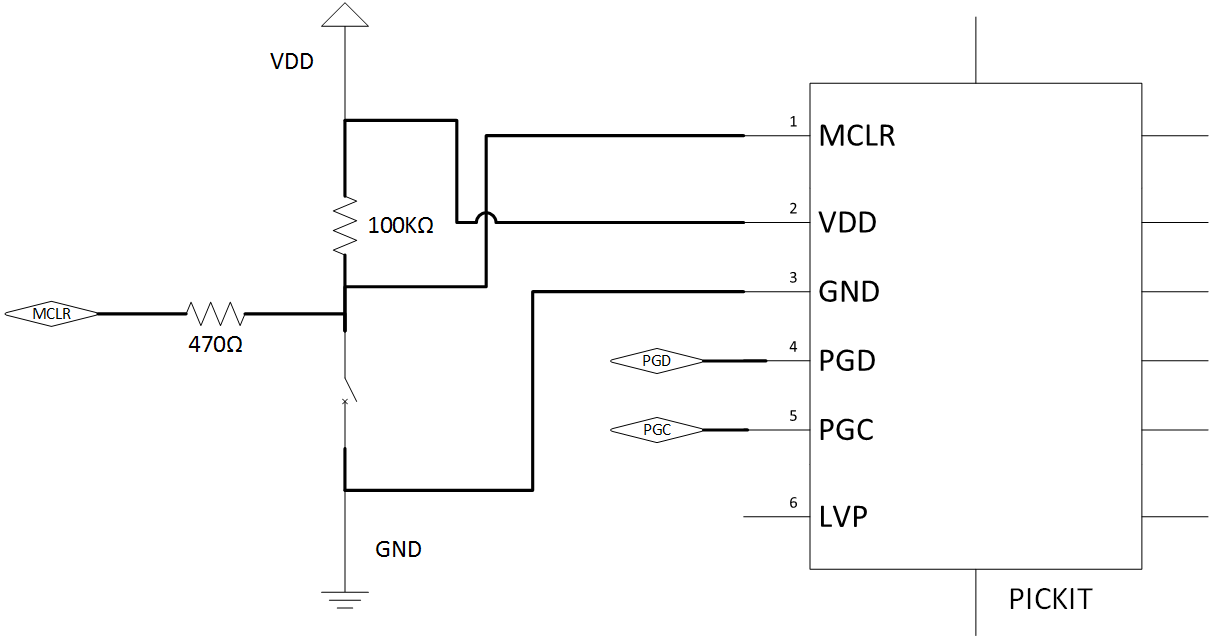
Las conexiones de los sensores se han realizado como explicamos en apartado 4 sin embargo no hemos puesto las resistencias de pull up porque ya vienen integradas en la PCBs que integran los sensores[Anexos].



**Fig. 7.5** Conexiones de nuestros sensores con nuestro MCU

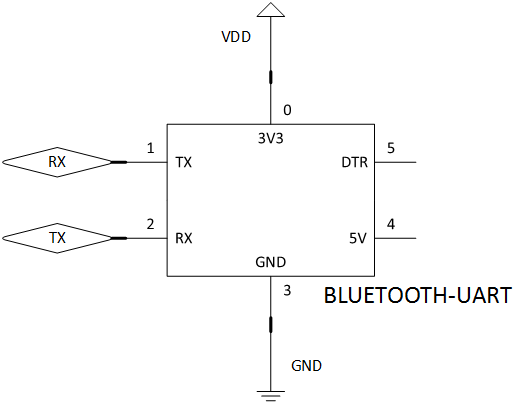
### Esquema de conexión PICKIT y UART-BLUETOOTH

Para programar nuestro PIC es necesario tener acondicionado los pines del PICKIT siguiendo el esquema que nos indica Microchip. Utilizando la placa de entrenamiento no tendremos problema porque ya viene integrada la conexión con PICKIT pero si tuviéramos que conectarlo sin la placa de entrenamiento deberíamos de realizar las conexiones siguiendo el siguiente esquema[Ver Esquema y Anexo PICKIT].



**Fig. 7.6** Conexión con Pickit

Cuando conectemos nuestro transmisor-receptor bluetooth a nuestro MCU, deberemos conectar el puerto RXmcu al TXbluetooth y el TXmcu al RXbluetooth[Ver].

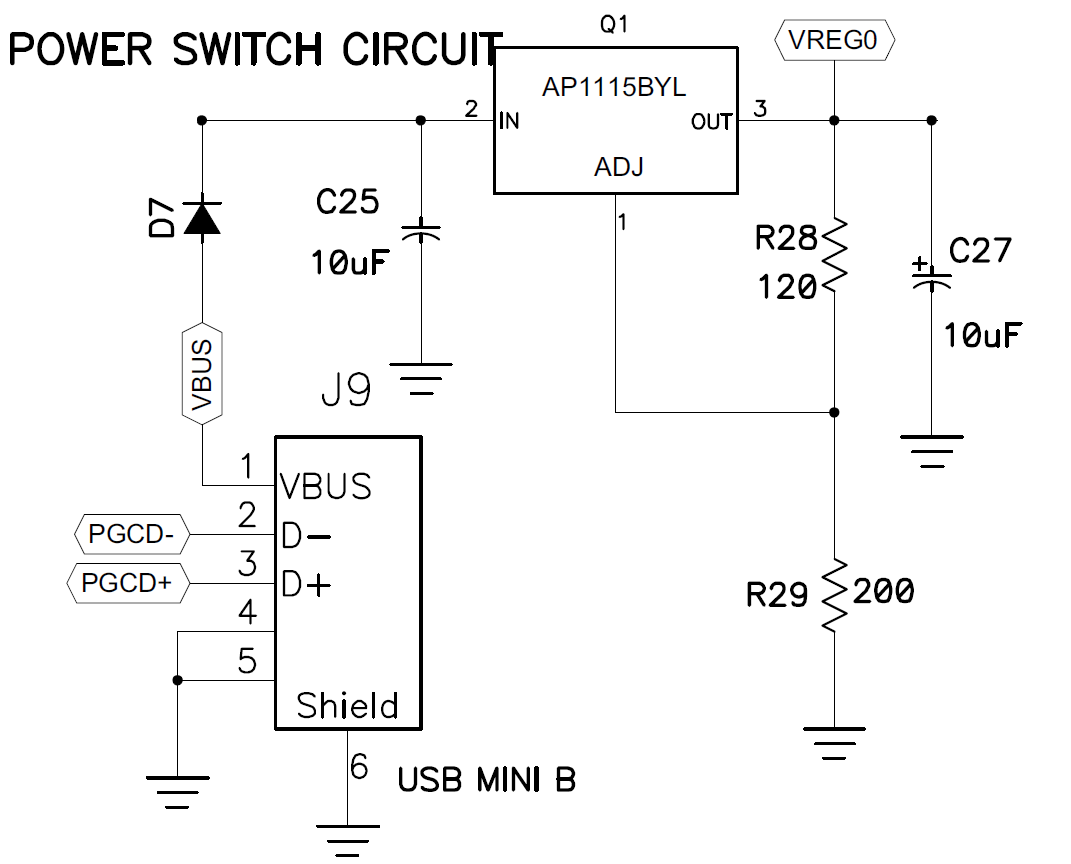


**Fig. 7.7** Conexiones de nuestro módulo bluetooth con nuestro MCU

### Esquema de alimentación de la Development Board

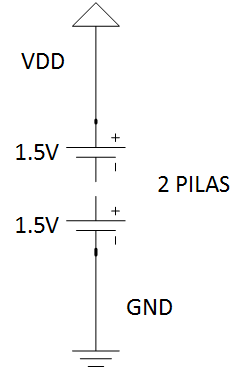
Hay 4 maneras de alimentar nuestro sistemas sensor con nuestra placa de entrenamiento, una utilizar el esquema de conexión del PICKIT y hacer que alimente el PICKIT todo nuestro sistema.

La segunda opción es utilizar la entrada USBmini y con el siguiente sistema regulador convertir los 5V del USB a 3.3V(VDD de nuestro prototipo).[Ver esquema, explicación Anexos]



**Fig. 7.8** Circuito de alimentación por mini USB

La tercera opción es utilizar las pilas de mercurio para realizar una alimentación de 2,9-3V. Este modo de conexión no es el más óptimo porque el voltaje de alimentación es inferior al voltaje de funcionamiento de alguno de nuestros sensores. Aún teniendo este inconveniente el sistema sensor excepto el sensor DHT22 que no funciona.[Ver Esquema]



**Fig. 7.9** Circuito de alimentación con 2 pilas de 1.5V

La cuarta opción consiste en utilizar el Cymbet’s EVAL-08 Solar Energy Harvester que incluye nuestra development board. El mismo está diseñado para ir cargando poco a poco una batería y que el sistema consuma nA, sin embargo nuestro sistema necesita 3.3V de alimentación constante por lo tanto no se puede utilizar para alimentar el sistema sensor.



**Fig. 7.10** Alimentación con Cymbet’s EVAL-08 Solar Energy Harvester

## Implementación del sistema sensor en PCB con Eagle

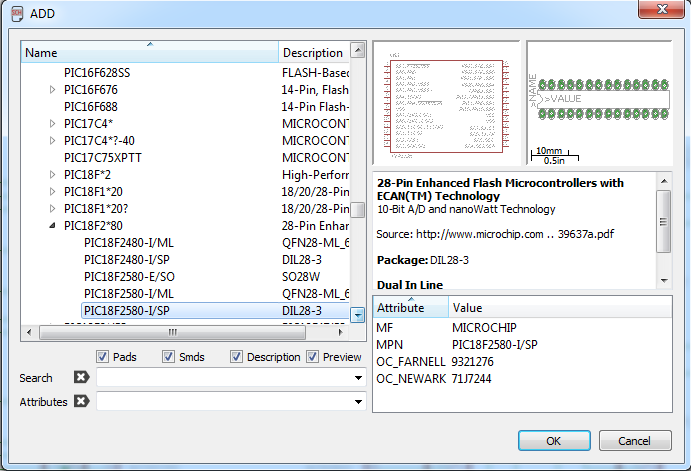
EAGLE, (siglas de Easily Applicable Graphical Layout Editor) es un programa de diseño de diagramas y PCBs con autoenrutador.

1. Lo primero que deberemos realizar para el diseño de nuestra propia PCB, es dibujar el esquema eléctrico de lo que va a ser nuestra placa para luego generar nuestra Broad.
2. Una vez generada la board, deberemos de colocar los elementos del circuito en nuestra placa e ir conectando los elementos.
3. Una vez acabado del diseño de la placa, generaremos los archivos gerber para realizar el mecanizado de nuestra placa.

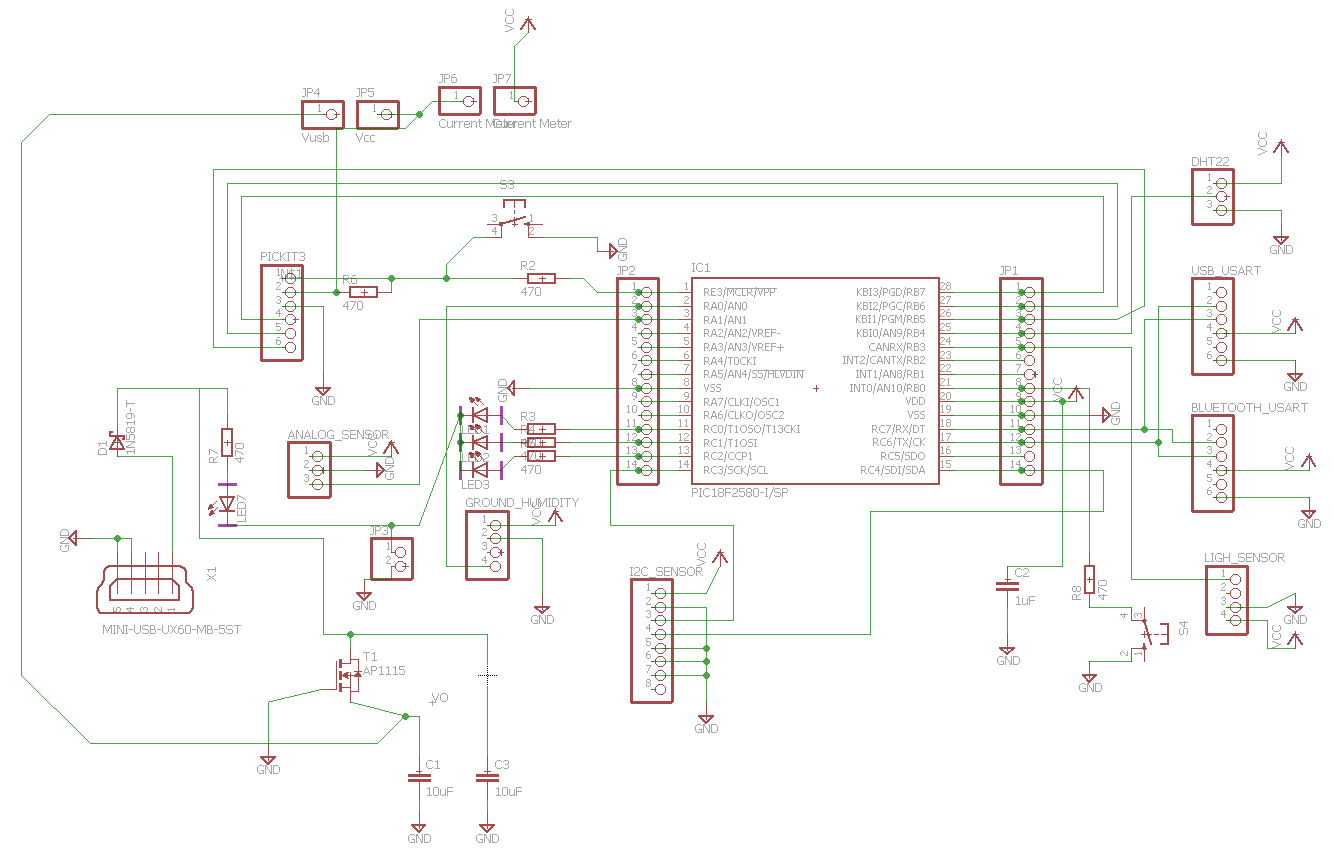
### Schematic

Lo primero que hay que hacer antes de empezar, es saber qué package tienen nuestros elementos del circuito porque el package delimita qué agujeros y/o forma de la pista para luego el dispositivo encaje al soldarlo. Dicho esto, en las librerías de Eagle no tienen todos los dispositivos pero casi todos los tipos de package sí suelen estar. Entonces, cuando no encoramos el dispositivo podemos hacer 2 cosas: crear nuestro propio dispositivo utilizando un package de la librería o coger un dispositivo existente con el mismo package.

En nuestro diseño hemos cogido elementos existentes con el mismo package para hacer el esquemático porque lo importante es el diseño de la board.

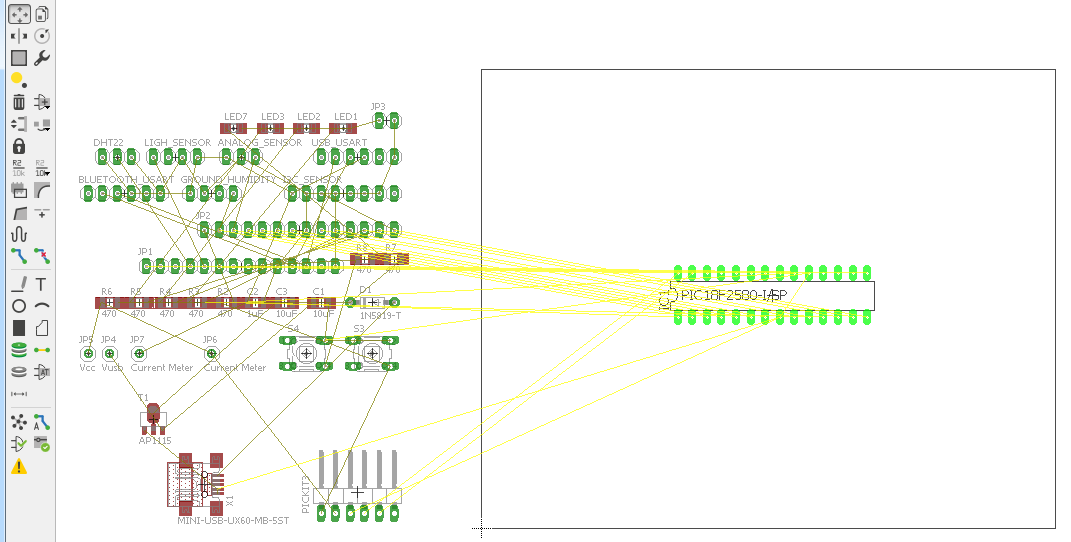


El esquema es casi el mismo que en la placa de pruebas, la única diferencia la tenemos en la alimentación[Anexos], y, pensando en la board, hemos cambiado la mayoría de los package a SMD categoria 1206.[Detalles en anexos] Las conexiones con los sensores y bluetooth se han hecho con puertos jumpers hembra.

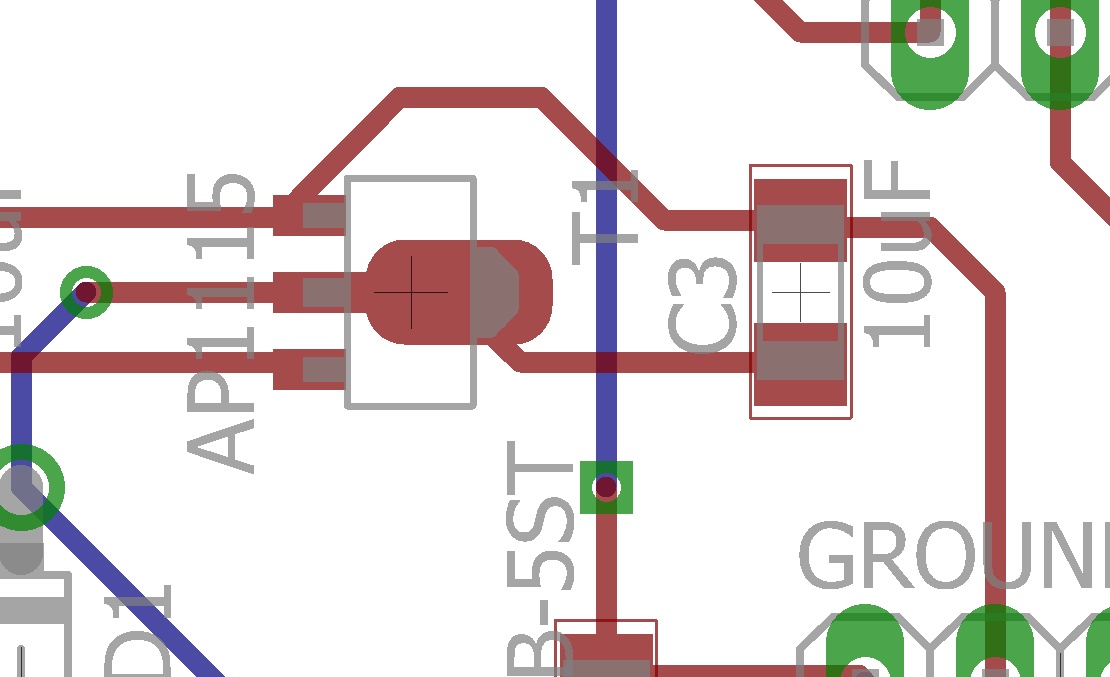


### Board

Una vez realizado el esquematizo, tenemos que generar nuestra board. Con la board generada, tenemos que ir colocando los elementos pensado cómo las pistas tendrán que conectar entre pin y pin.

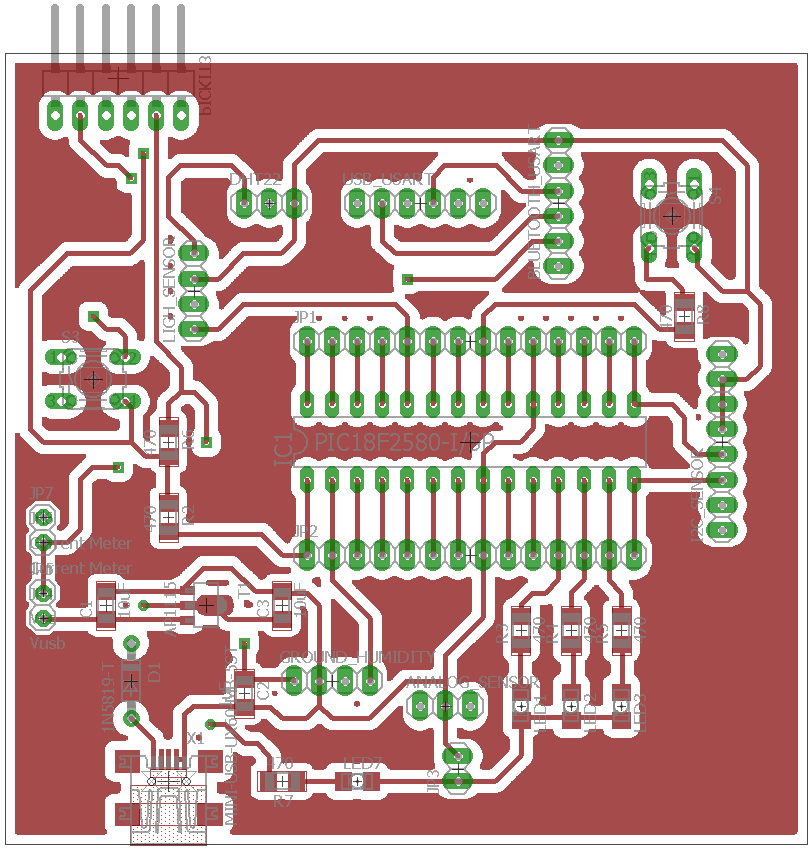


Con todos los elementos colocados, tenemos que dibujar las pistas sin que se solapen entre ellas. Para dibujar las pistas, podemos usar el autorute pero la mejor opción es dibujarlas de forma manual porque normalmente el autorute dibuja rutas incoherentes y dificulta el cumplimiento *design rules*. Nuestra placa es de doble capa(TOP y BOTTOM), entonces cuando dibujamos las pistas podemos usar sin ningún problema vías para pasar una pista por debajo de otra para evitar que se solapen[Ver ].

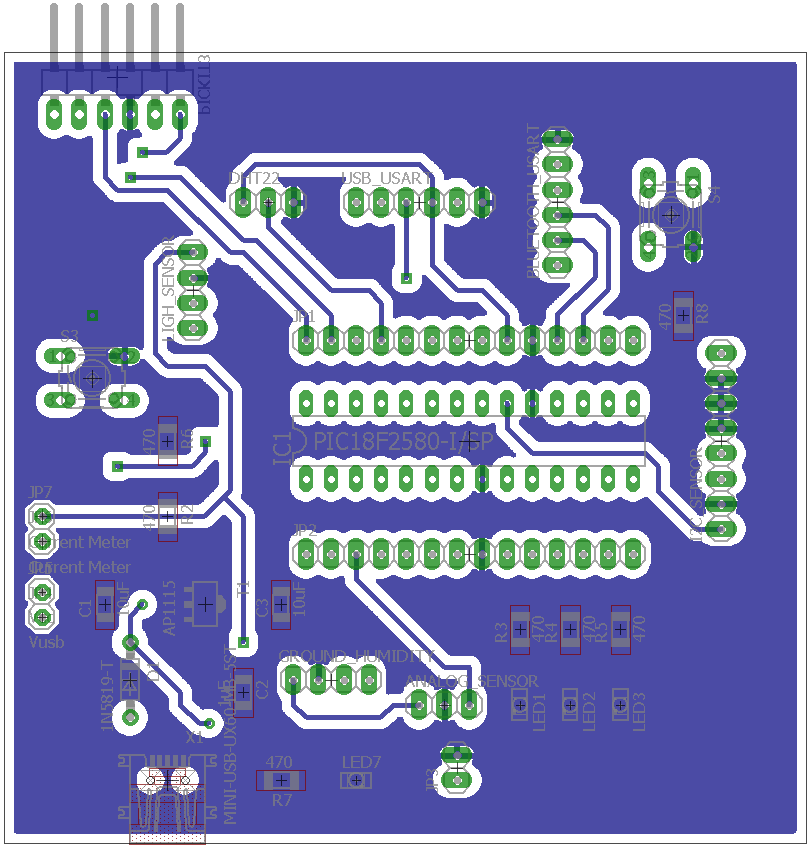


(línea roja capa top y azul bottom)

Finalmente, tenemos que dibujar un polígono que ocupe todo el área de la placa para que Eagle nos dibuje el cobre que no se utiliza que está al lado de las pistas.[Ver Cara Top][Anexos]



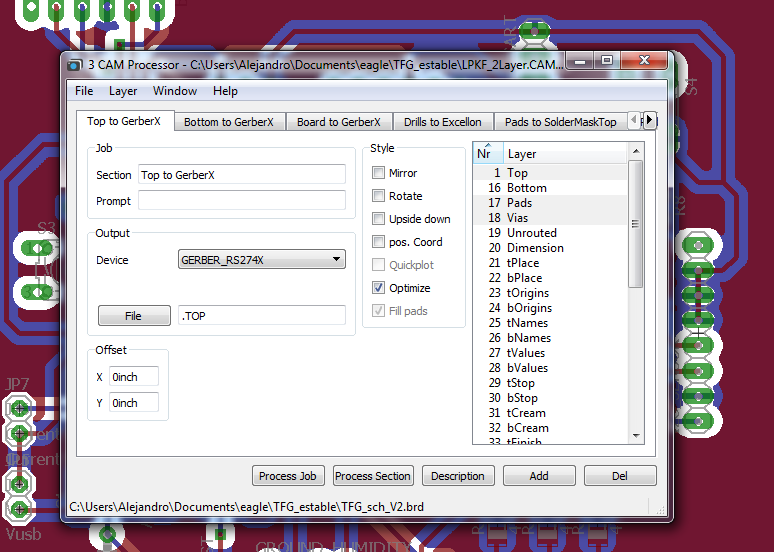
Además en la cara bottom conectaremos el cobre sobrante a masa[Ver cara bottom]



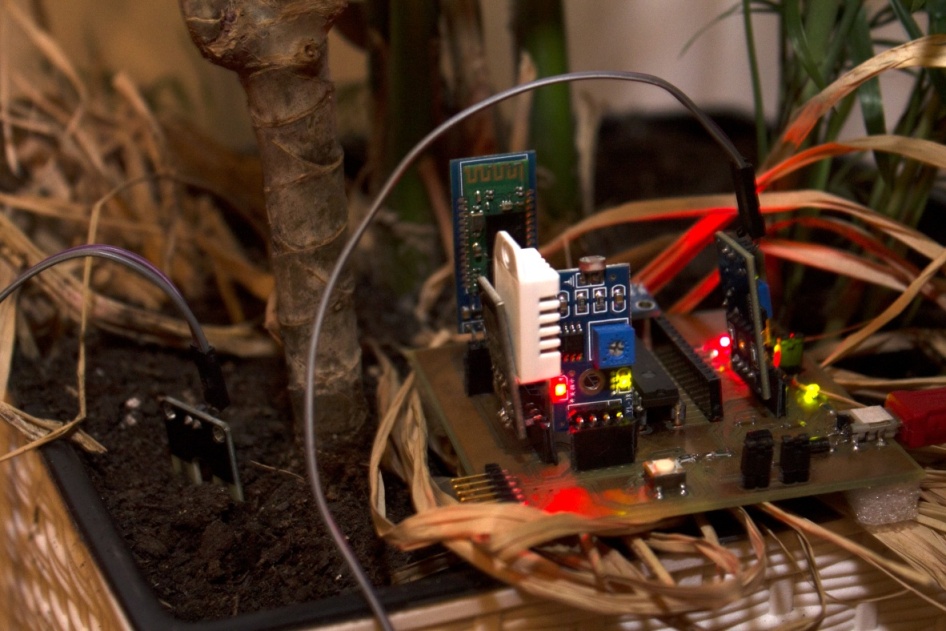
(Cara bottom-plano de masa)

### Mecanizado

Con la placa acabada, sólo falta generar los archivos gerber para realizar el mecanizado de nuestra placa. Deberemos de generar 4 archivos: TOP, BOTTOM, DRILLS y BOARD.



El proceso de mecanizado queda explicado en los anexos.[Anexos]

****

(PCB funcionando)

# Conclusiones

Nuestro sistema sensor funciona correctamente pero, probando el sistema, hay campos donde se puede mejorar.

## Mejoras en la alimentación de los sensores

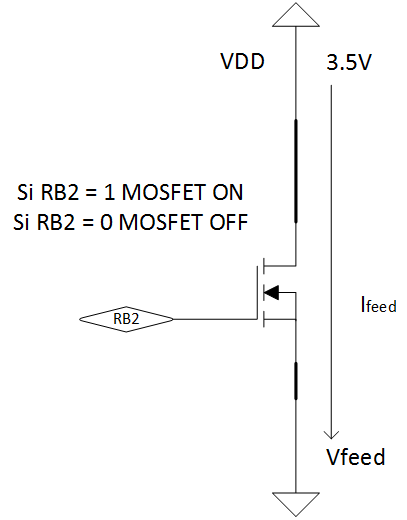
Ahora mismo nuestro sistema sensor siempre mantiene alimentados a los sensores y modulo bluetooth. Tener siempre conectado a la alimentación los sensores implica que cuando entras en modo sleep, si no desconectas los periféricos, el sistema no está optimizado para el bajo consumo.

En nuestro diseño del sistema sensor en PCB pusimos puentes jumpers para poder realizar mediciones de corriente para analizar los consumos. Tabla de consumos teniendo en cuenta que la alimentación de la placa PCB es de 3.5V(la placa de pruebas a 3.3V):

|  |  |  |  |
| --- | --- | --- | --- |
| Dispositivo | A&T Mode  (µA) | Sleep Mode(actual)  (µA) | Sleep Mode Ideal  (µA) |
| PIC18F26K20 | 4530 | 1.5 | 1.5 |
| HC-06(bluetooth) | 21000-32000\* | 3000-7000\* | ~0 |
| KY-013 | 182 | 182 | ~0 |
| MCP9808 | 180 | 180 | ~0 |
| GH Sensor | 2400 | 2400 | ~0 |
| Light Sensor | 2270 | 2270 | ~0 |
| DHT22 | 11 | 11 | ~0 |
| Total | 28mA-39mA\* | 13mA-17mA\* | ~1.5 µA |

(\*)El consumo del módulo blueetooth se dispara hasta los 43mA cuando no está vinculado a ningún dispositivo.

La mejora en el circuito de alimentación consistiría en controlar la alimentación de los sensores y el módulo de comunicaciones con un transistor Mosfet NPN. El transistor estaría en On si uno de los puertos del MCU conectado al Gate está en '1' y si el mismo está en '0' el transistor estaría en Off sin alimentar a los sensores y bluetooth. Con esta solución nos podríamos acercarnos al Sleep Mode Ideal de la tabla.



(podríamos utilizar el puerto RB2 del MCU)

Finalmente el consumo, en comparación con el consumo de corriente que nos indica el fabricante en Sleep mode con el WDT activo, es de 1.5µA alimentado a 3.5V respecto a 800nA alimentado 1.8V. Podemos decir que es un dato muy positivo.

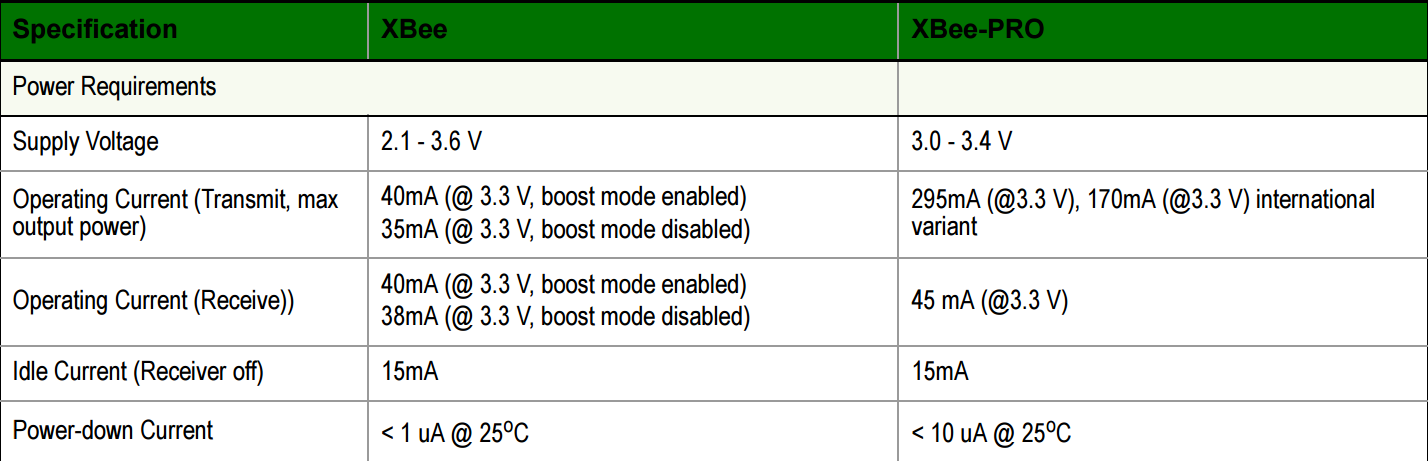
## Problemas de la comunicación bluetooth

Un problema derivado de usar bluetooth se produce cuando se pierde la vinculación entre dispositivos porque no se produce de una manera automática.

Un segundo problema que no afecta al funcionamiento del sistema es el poco alcance, aproximadamente 10m-25m.

Ambos problemas implican realizar el estudio de otras tecnologías de transmisión wireless como: Zigbee, Wi-Fi, GSM, 3G, LTE, etc.

Zigbee es una tecnología pensada en realizar transmisiones de baja velocidad y es bastante más eficiente energéticamente que bluetooth. Cuando el módulo está en espera consume 1µA-10µA respectivamente frente a los 3mA-7mA del bluetooth. Ver en (**Fig. 9.1**).



**Fig. 9.2** Tabla de Power Requirements de ZigBee RF Modules by Digi International

## Implementación de un sistema de energy harvesting

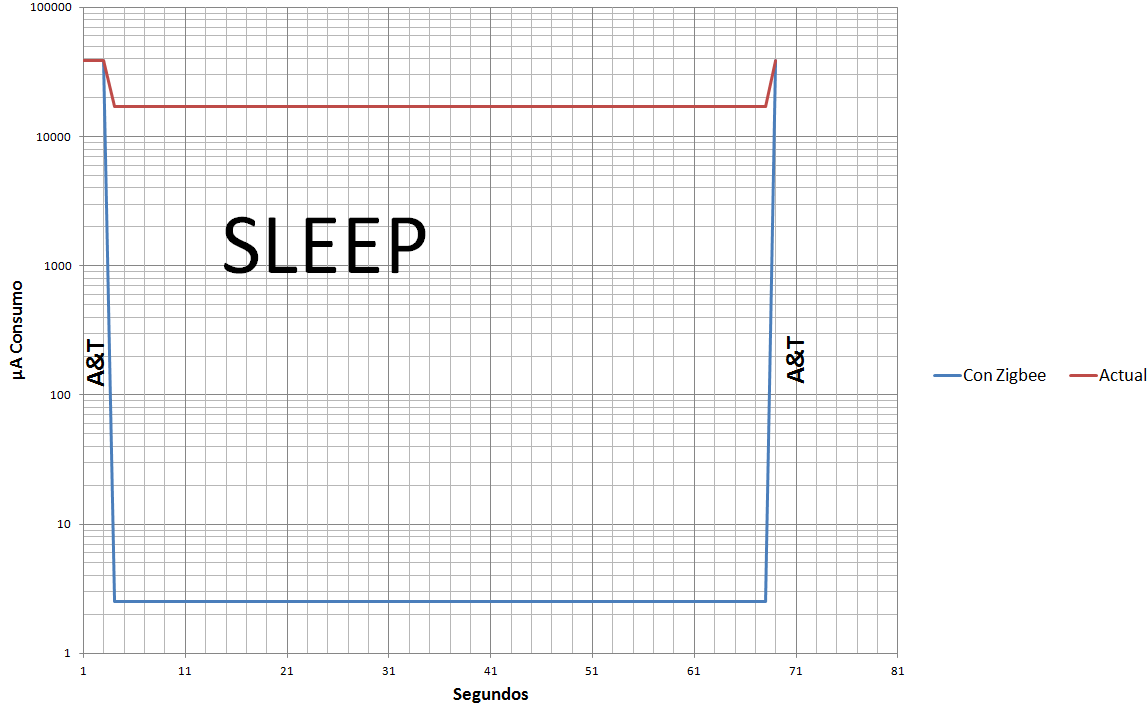
Para implementar un sistema de energy harvesting, antes se tendrían que solucionar los problemas anteriores porque no sería posible realizarlo por exceso de consumo.

Nuestra fuente de energía podría provenir de la [energía solar](https://es.wikipedia.org/wiki/Energ%C3%ADa_solar), [energía geotérmica](https://es.wikipedia.org/wiki/Energ%C3%ADa_geot%C3%A9rmica), [energía eólica](https://es.wikipedia.org/wiki/Energ%C3%ADa_e%C3%B3lica), los [gradientes de salinidad](https://es.wikipedia.org/w/index.php?title=Gradiente_de_salinidad&action=edit&redlink=1), la [energía cinética](https://es.wikipedia.org/wiki/Energ%C3%ADa_cin%C3%A9tica) u otras fuentes de energía.

Una vez escogido el modo en el que realizaremos energy harvesting, deberemos realizar los siguientes cambios o novedades en nuestro sistema.

### Usar Zigbee

Antes de implementar el sistema de energy harvesting cambiaríamos de tecnología de inalámbrica para poder aprovecharnos del Sleep del Zigbee y ser más eficientes cuando no transmitimos datos.



**Fig. 9.3** Predicción de mejora usando Zigbee y desconectando los sensores con un Mosfet

### Implementar un sistema de almacenamiento de energía

La idea consiste en ir almacenando energía mientras nuestro MCU se encuentra en Sleep para que tener la suficiente energía cuando esté adquiriendo y transmitiendo datos.