****

**TREBALL FINAL DE GRAU**

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

Índice

[1. Objetivos/Apstract 4](#_Toc455330062)

[2. Especificaciones del sistema sensor 5](#_Toc455330063)

[2.1 Diagrama de estados (FSM) 5](#_Toc455330064)

[2.2 Registros de configuración del PIC18F 6](#_Toc455330065)

[2.3 Timers y ADC 6](#_Toc455330066)

[2.4 Comunicación con sensores digitales(I2C y otros protocolos) 7](#_Toc455330067)

[2.5 Comunicación serial(UART) con Labview 7](#_Toc455330068)

[2.6 Aplicación Labview 7](#_Toc455330069)

[3. Análisis de las señales a captar con los sensores 8](#_Toc455330070)

[3.1 Temperatura ambiente 8](#_Toc455330071)

[3.2 Humedad relativa 8](#_Toc455330072)

[3.3 Humedad de la tierra 8](#_Toc455330073)

[3.4 Intensidad lumínica 9](#_Toc455330074)

[4. SENSORES 10](#_Toc455330075)

[4.1 Temperatura Analog KY-013 10](#_Toc455330076)

[4.2 Humedad Tierra Analog 10](#_Toc455330077)

[4.3 Iluminancia Analog 11](#_Toc455330078)

[4.4 Temperatura Digital I2C MCP9808 12](#_Toc455330079)

[4.5 Temperatura y Humedad Relativa Digital OneWire Protocol DHT22 14](#_Toc455330080)

[5. Estructura del programa principal 16](#_Toc455330081)

[5.1 Estados 16](#_Toc455330082)

[5.1.1 Idle 18](#_Toc455330083)

[5.1.2 Setup ADC (Ground Humidity) 18](#_Toc455330084)

[5.1.3 Acquiring & Transmission ADC (Ground Humidity) 18](#_Toc455330085)

[5.1.4 Stop ADC (Ground Humidity) 18](#_Toc455330086)

[5.1.5 Setup ADC (Analog Temp) 18](#_Toc455330087)

[5.1.6 Acquiring & Transmission ADC (Analog Temp) 18](#_Toc455330088)

[5.1.7 Stop ADC (Analog Temp) 19](#_Toc455330089)

[5.1.8 Setup OneWire (DHT22) 19](#_Toc455330090)

[5.1.9 Acquiring & Transmission OneWire (DHT22) 19](#_Toc455330091)

[5.1.10 Stop OneWire (DHT22) 19](#_Toc455330092)

[5.1.11 Setup I2C (MCP9808) 19](#_Toc455330093)

[5.1.12 Acquiring & Transmission I2C (MCP9808) 19](#_Toc455330094)

[5.1.13 Stop I2C (MCP9808) 20](#_Toc455330095)

[5.1.14 Setup ADC (Analog LDR) 20](#_Toc455330096)

[5.1.15 Acquiring & Transmission ADC (Analog LDR) 20](#_Toc455330097)

[5.1.16 Stop ADC (Analog LDR) 20](#_Toc455330098)

[5.2 Interrupciones 20](#_Toc455330099)

[5.2.1 RCIF 20](#_Toc455330100)

[5.2.2 INT0IF 21](#_Toc455330101)

[5.2.3 TMR3IF 21](#_Toc455330102)

[5.2.4 TMR2IF 21](#_Toc455330103)

[6. Simulación con Proteus. paso a paso 22](#_Toc455330104)

[6.1 FRECUENCIA DE MUESTREO CON TIMER3 22](#_Toc455330105)

[6.2 INTERRUPCIÓN INT0IF 23](#_Toc455330106)

[6.3 COMUNICACIÓN CON EL SENSOR DHT22 23](#_Toc455330107)

[6.4 COMUNCIACIÓN I2C 25](#_Toc455330108)

[6.5 COMUNICACIÓN SERIAL 25](#_Toc455330109)

[7. implementación real en una placa de pruebas 26](#_Toc455330110)

[8. Trasmison serial(USB o BLuetooth) 27](#_Toc455330111)

[9. Aplicacion LABview paso a paso 28](#_Toc455330112)

[10. PCB eagle. 29](#_Toc455330113)

[11. mejoras: 30](#_Toc455330114)

[11.1 mejoras en la alimentacion de los sensores: 30](#_Toc455330115)

[11.2 mejoras a la hora de escoger el modo de transmision de los datos wireless 30](#_Toc455330116)

[11.3 mejora a la hora de realizar el energy harvesting. 30](#_Toc455330117)

[12. 32](#_Toc455330118)

# Objetivos/Apstract

El objetivo del proyecto es realizar 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 enviar, vía Wireless, esos datos a nuestra aplicación de control de 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 sistema sensor

El sistema sensor se implementará con el microcontrolador PIC18F26K20.



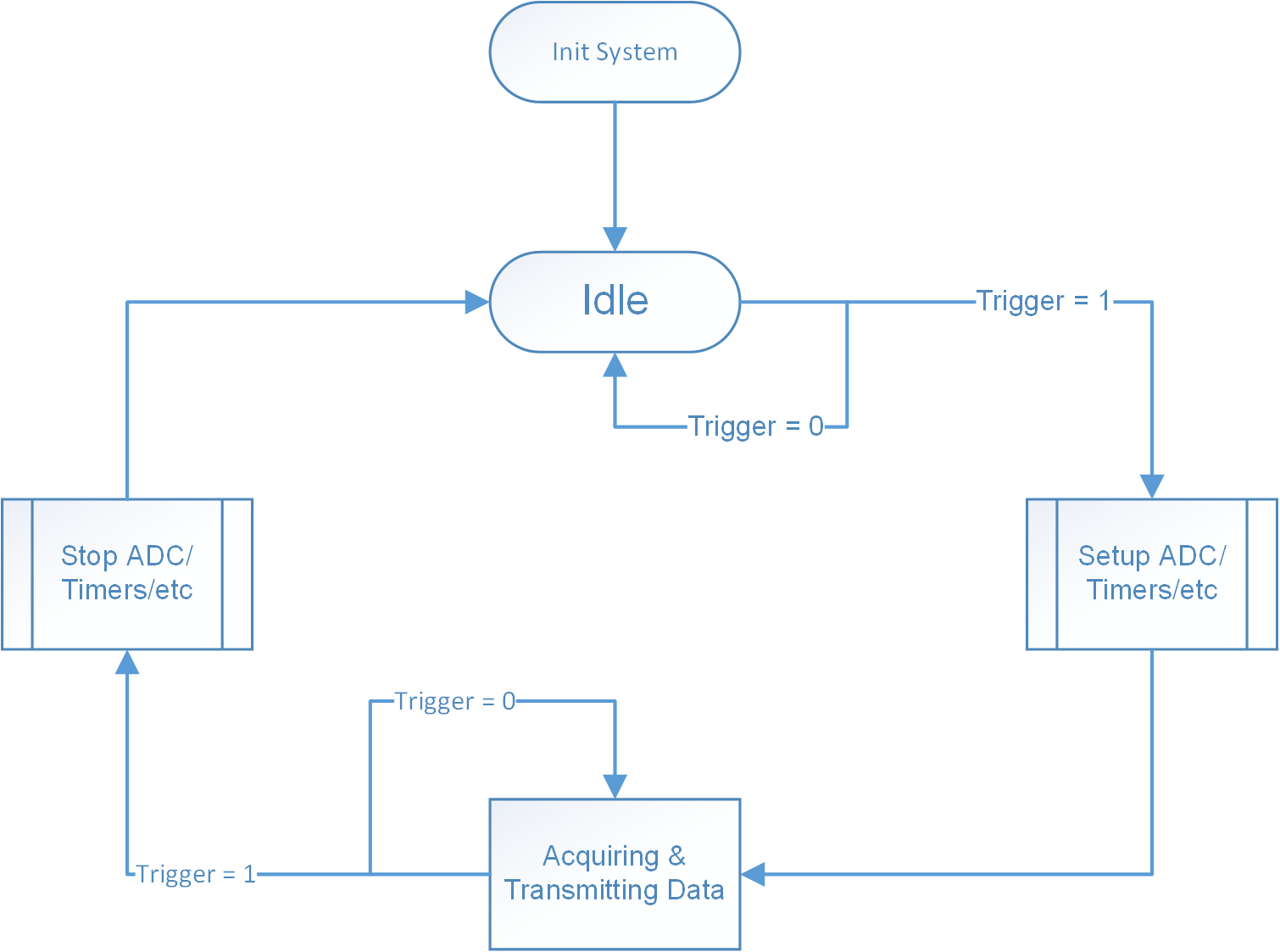
Características principales del microcontrolador son:

* Es un microcontrolador de 28 pines.
* Reloj interno de 16MHz/32.756KHz.
* Comunicación UART, SPI, I2C y PWM.
* 12 entradas al ADC-10bits.
* Alimentación 1.8V-3.6V.
* Sleep mode.

El sistema deberá de ser autónomo y de bajo consumo. De este modo, el sistema tendrá que ponerse a trabajar durante un periodo corto de tiempo en el cual estará tomando medidas y transmitiendo los datos a nuestra aplicación de *Labview.* Una vez se hayan envidado todos los datos, el microcontrolador se pondrá en modo Sleep durante un periodo largo de tiempo.

## Diagrama de estados (FSM)

El programa en C se programará con el estilo FSM(finite state machine) en el cual seguiremos el patrón del siguiente diagrama de estados.



Estados(ejemplo):

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

En Init System, configuraremos todo lo relacionado con los registros de configuración. Después, empezaremos en modo Idle y cambiaremos de estado cuando la variable trigger cambie a 1. Cuando queramos hacer una tarea en un estado, siempre tendremos el estado de Setup y Stop para hacer setup y reset de los registros necesarios para realizar dicha tarea. La tarea se dará por finalizada cuando la variable trigger sea = 1 y se pasará al siguiente estado tal y cómo mostramos en el diagrama.

## Registros de configuración del PIC18F26K20

Configuración del oscilador interno.

Configuración del baudrate de la comunicación UART

## Timers y ADC

Queremos hacer que el microcontrolador entre en modo Sleep cuando acabe de adquirir y transmitir los datos. Con lo cual, tendremos que configurar un Timer para que cuente el tiempo que pasa mientras estamos en modo Sleep y que despierte al microcontrolador.

Cuando tomemos muestras con el ADC tomaremos 4 muestras en 1 segundo y haremos la media de las 4 muestras para obtener una muestra filtrada. Dicho esto, configuraremos otro Timer para que nos fije la frecuencia de muestreo en 4Hz.

[Cuando se activa el modo Sleep, se activa el *Watchdog Timer* que funciona con el oscilador interno de 32.756KHz y consume menos que el de16MHz.]

## Comunicación con sensores digitales(I2C y otros protocolos)

Realizaremos el estudio del protocolo I2C que nos permite comunicarnos con muchos sensores utilizando solamente 2 pines de PIC. Un sensor con interfaz digital I2C tiene por lo general una dirección de 7bits asignada o con una dirección I2C configurable. Para la adquisición de los datos, el microcontrolador llama al sensor por su dirección y éste contesta. [Anexo]

Trabajaremos con otro tipo de sensores digitales que tienen otros protocolos de comunicación no estandarizados y que el fabricante explica cómo implementarlos en tu microcontrolador. [1 Wire Comunication Anexos]

## Comunicación serial(UART) con Labview

Una vez configurada la comunicación serial del microcontrolador con un baudrate a 9600, conectaremos al microcontrolador 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.



## Aplicación Labview

Una vez recibamos los datos en el aplicación, los iremos mostrando por pantalla en tiempo real. Además podremos mandarle al microcontrolador comandos para que se ponga en modo Sleep o empiece a tomar medidas.

# Análisis de las señales a captar con los sensores

El teorema de Nyquist nos dice que para muestrear una señal analógica, sin perder información de la misma, tenemos que usar una frecuencia de muestreo el doble de su ancho de banda o superior.

Las señales que tenemos que medir tienen poca variancia en el tiempo por lo que podemos escoger frecuencias de muestreo muy bajas.

## Temperatura ambiente

Si nos ponemos en un caso extremo, nos podemos situar en el desierto del Sahara donde se una gran oscilación de temperatura entre el día y la noche. En los días de más oscilación se puede llegar a haber diferencias entre máximas y mínimas de 20-30ºC.[Referencia]

Queremos medir la temperatura con una precisión de 0,1ºC y queremos saber el máximo periodo de muestreo que podemos escoger. Definiremos el ancho de banda de la señal como el diferencial de la temperatura(∆T) entre la presión de lectura(p) y todo entre el diferencial del tiempo(∆t). Y por último, entre que se alcanza la máxima temperatura y la mínima transcurre en media unas 12h.

∆T=30ºC; p=0.1ºC; ∆t=12h;

Por lo tanto, si no queremos perder información de la señal de la temperatura no podemos muestrear con un periodo superior a 72s.

## Humedad relativa

La humedad relativa depende de la temperatura ambiente y la cantidad moléculas de H2O que hay en el aire.

La señal de la humedad relativa se obtendrá con un porcentaje y una precisión de 1%. Para esta señal no hará falta un estudio para obtener la frecuencia de muestro mínima porque tiene mucha menos variancia y se optará por una frecuencia de muestro única.

## 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. Por lo tanto, mediremos la conductividad de la tierra con un sensor que nos dé una salida en forma de voltaje para que nuestro ADC capte la señal.

Para esta señal no hará falta un estudio para obtener la frecuencia de muestro mínima porque tiene mucha menos variancia que la temperatura ambiente y se optará por una frecuencia de muestro única.

## 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.

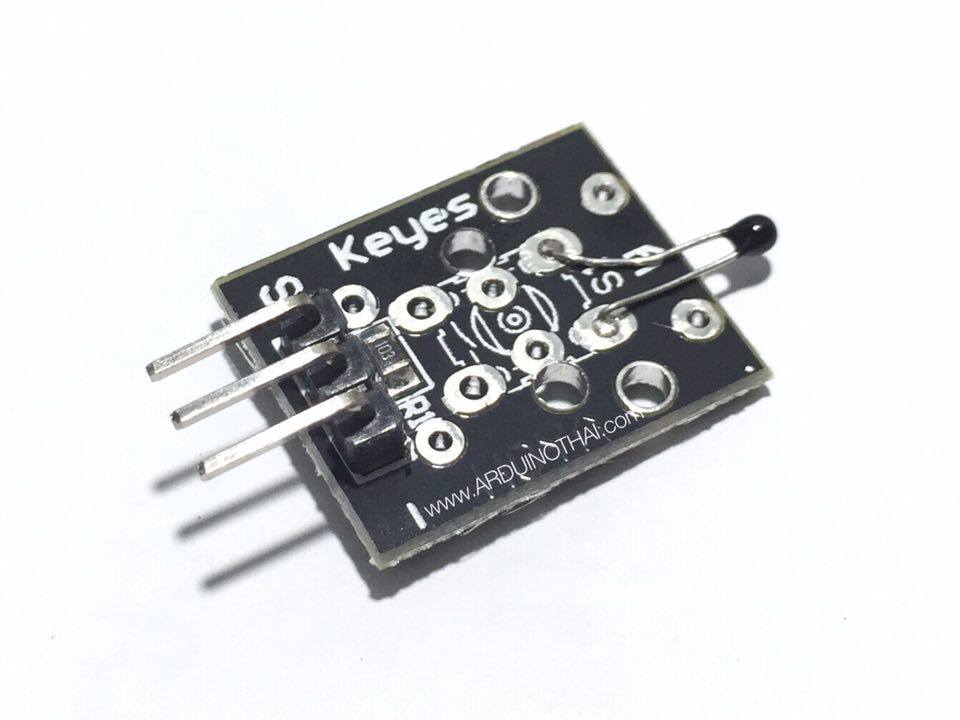
Más adelante explicaremos cómo detectaremos esta señal y qué estrategia seguiremos para su muestreo.

# SENSORES

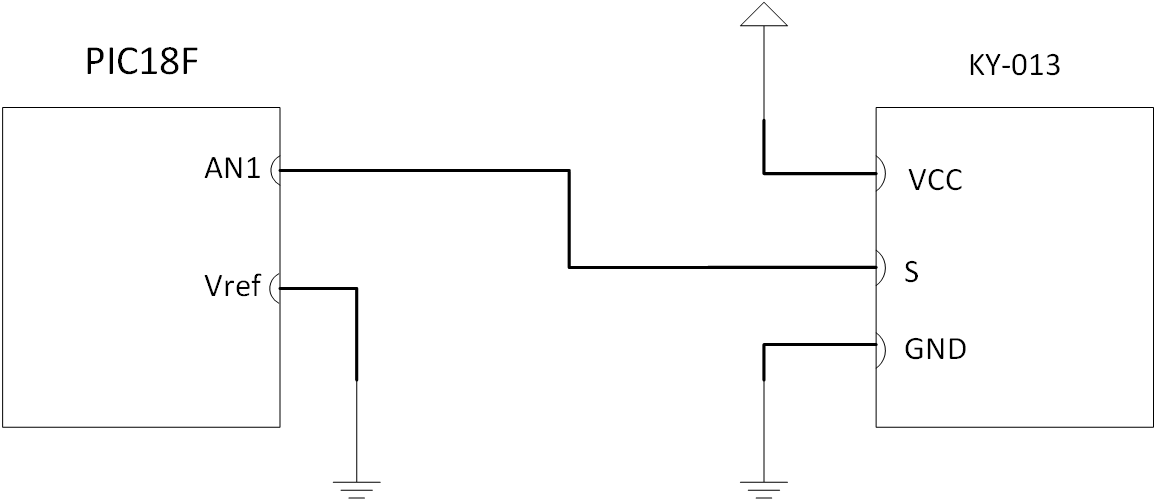
Para tomar muestras de las diferentes señales que tenemos que medir utilizaremos tanto sensores con interface de salida analógica como digital, con el fin de estudiar diferentes tipos de sensores y protocolos de comunicación.

## Temperatura Analog KY-013

El KY-013 es un termistor NTC, con lo cual su resistencia varía respecto la temperatura siguiendo un comportamiento exponencial. El fabricante nos proporciona la fórmula que nos convierte el código leído por el ADC y Temperatura.[Anexo][foto sensor]



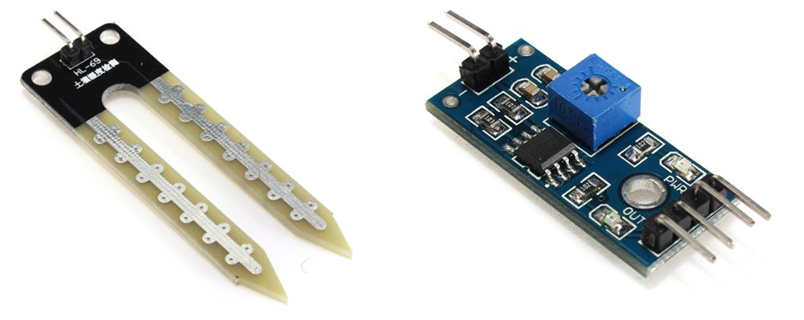
Además el fabricante nos explica cómo conectar el sensor a nuestro MCU. El KY-013 tiene 3 pines: 1 para ser conectado a VCC, 1 para conectar a GND y 1 para conectar la salida analógica(S) a una entrada del ADC(AN1).[figura]



(figura)

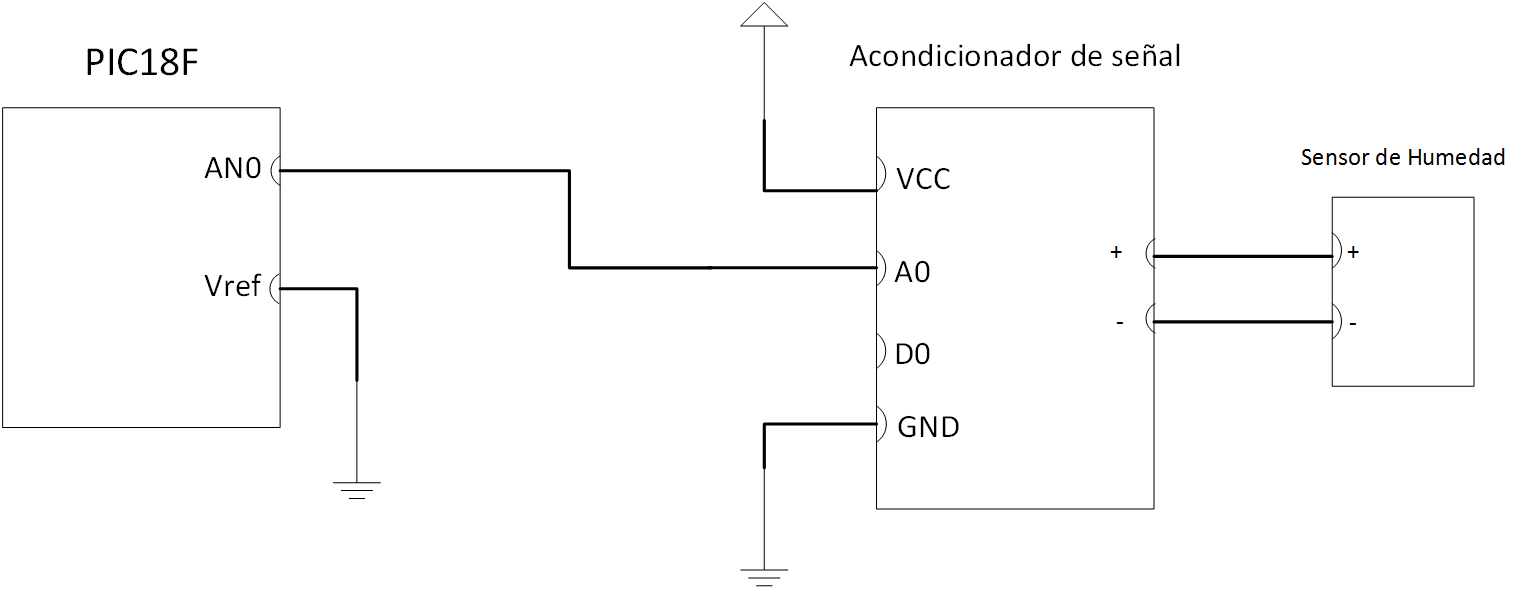
## Humedad Tierra Analog

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]



(izquierda sensor, derecha acondicionador)

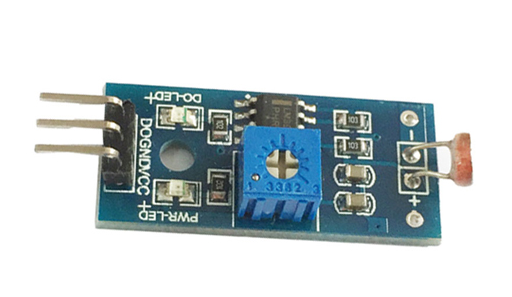
En este caso, conectaremos la salida A0 a la entrada del ADC AN0 y el resto de conexiones tal y cómo se muestra en la figura 1[lea Anexo].



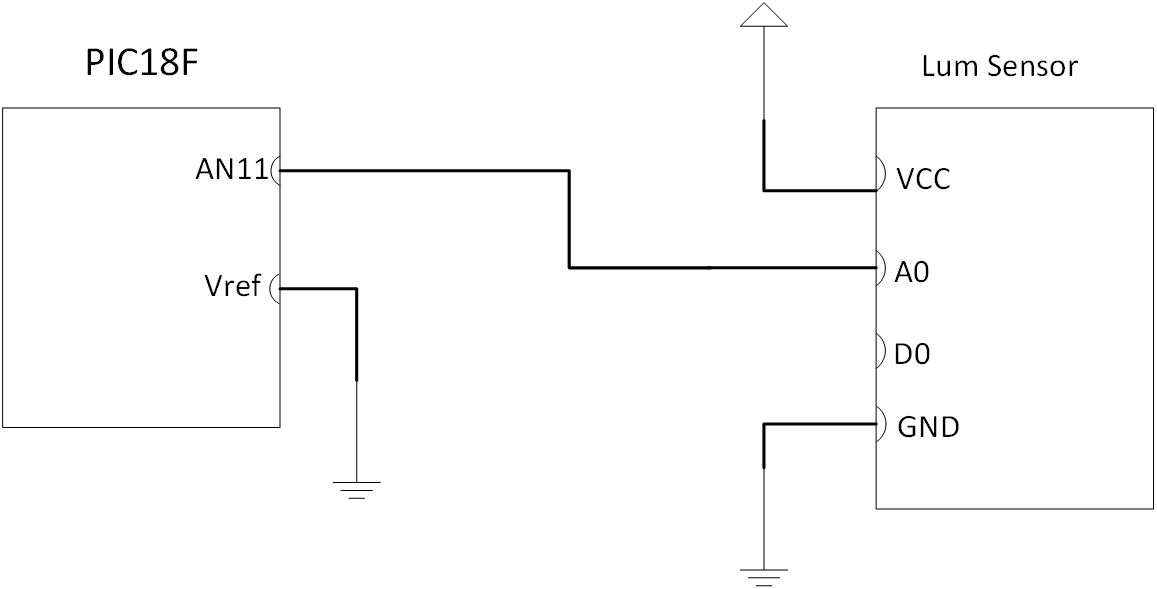
(figura1)

## 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]



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].

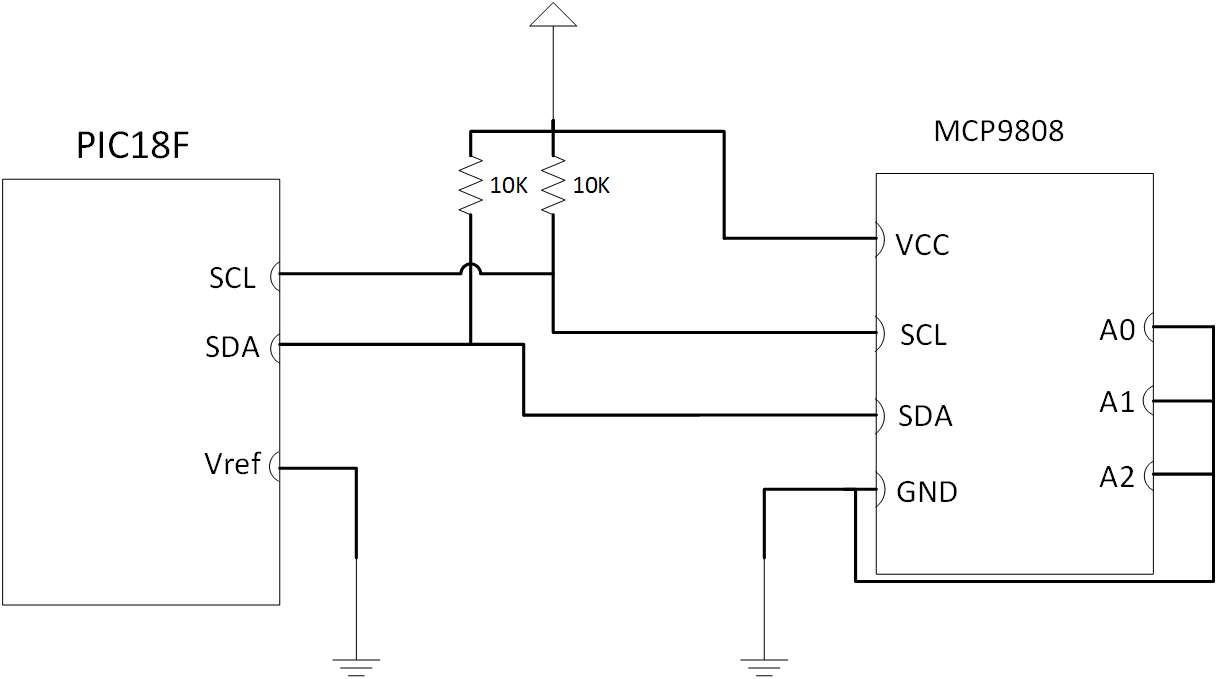


## Temperatura Digital I2C MCP9808

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].

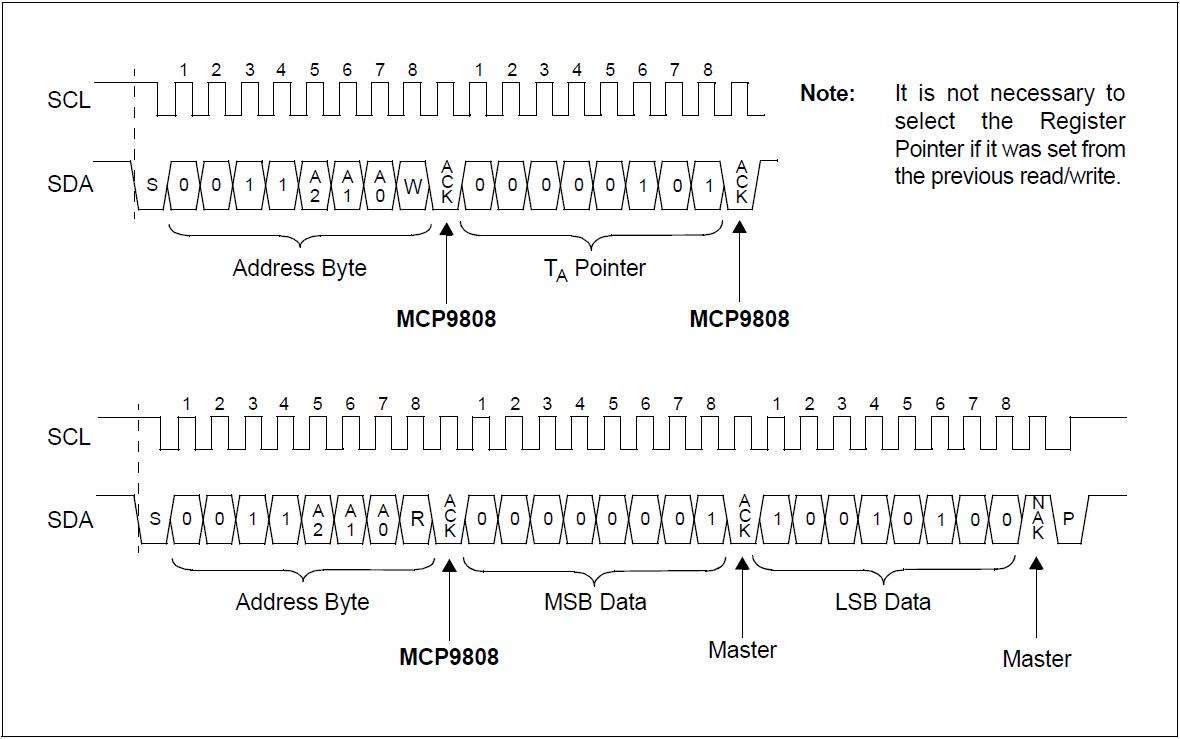


El MCP9808 tiene esta dirección I2C 0011XXX donde los últimos 3 bits de la dirección son configurables por vía externa. En el esquema de conexión, los puertos de A0, A1 y A2 están conectados a masa para que sean '0' y por lo tanto la dirección I2C sea 0011000.

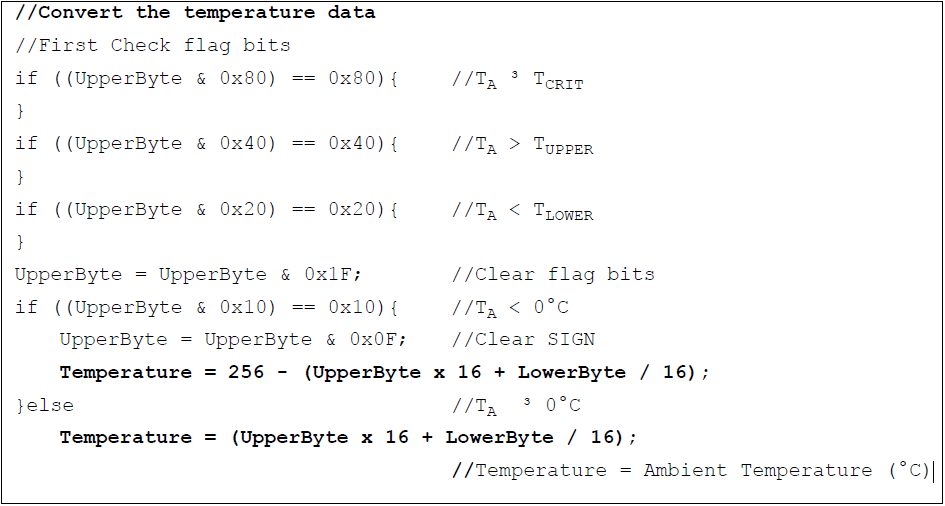


(Esquema de conexión con la MCU)

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



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.

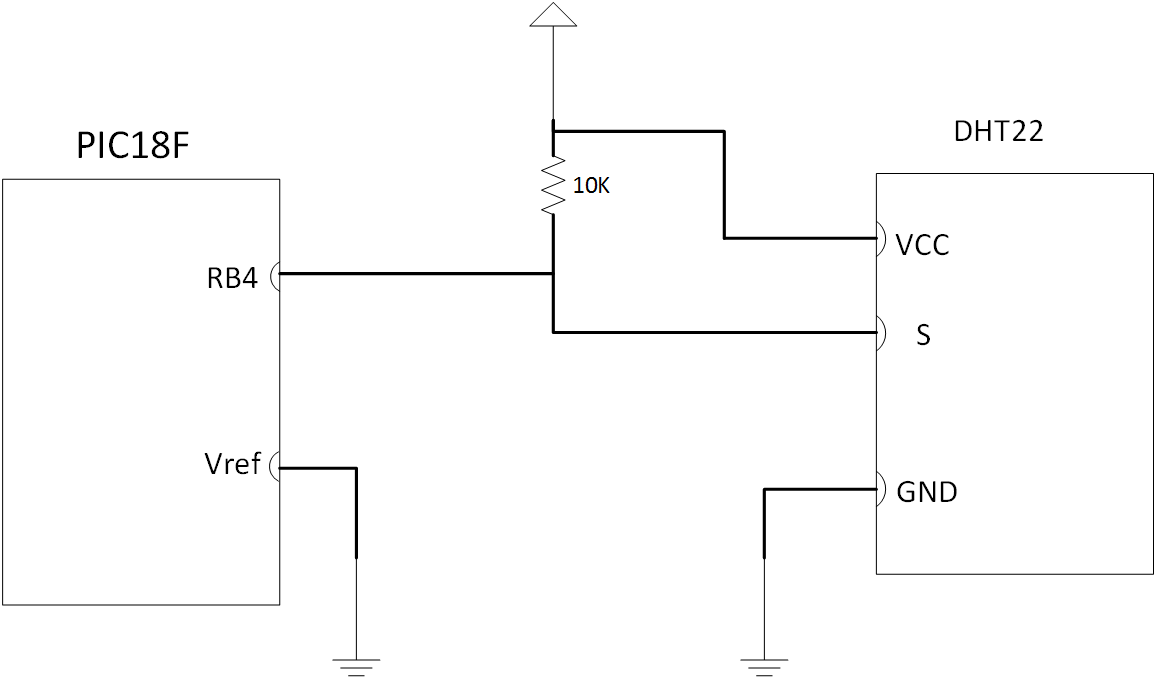


## Temperatura y Humedad Relativa Digital OneWire Protocol DHT22

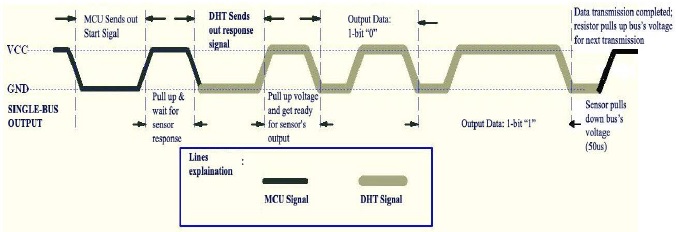
El DHT22 es un sensor de temperatura y humedad relativa con interface digital. La comunicación de realizará utilizando sólo un pin del MCU y se realizará de forma asíncrona porque no tenemos un clock como en la comunicación I2C.



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.



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].



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.

# Trasmisión UART

Un transmisor UART(Universal Asynchronous Receiver-Transmitter) es un dispositivo que controla los puertos y dispositivos serie.

En nuestro proyecto utilizaremos el transmisor UART del MCU para comunicarnos de forma serial con nuestra aplicación de Labview. La aplicación de Labview estará presente en un ordenador y normalmente un ordenador no tienen puertos UART. La solución es encontrar la manera de convertir la comunicación UART en USB o bluetooth(preferiblemente).

## USB-UART

Un convertidor USB-UART se puede adquirir en el mercado y sólo tenemos que instalar los drivers del dispositivo para que nos cree un puerto virtual COM para que la aplicación Labview pueda comunicarse por ese puerto.



(CP2102)

## BLUETOOTH-UART

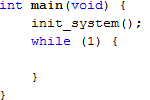
De la misma manera, un convertidor Bluetooth-UART se puede adquirir en el mercado y sólo tenemos que instalar los drivers del dispositivo para que nos cree un puerto virtual COM para que la aplicación Labview pueda comunicarse por ese puerto.



(HC-06)

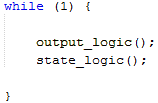
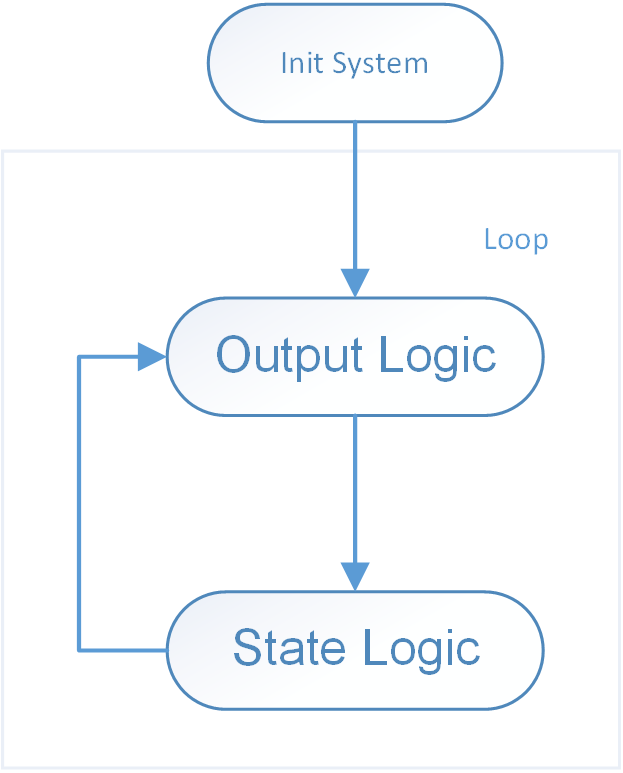
# Estructura del programa principal

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.



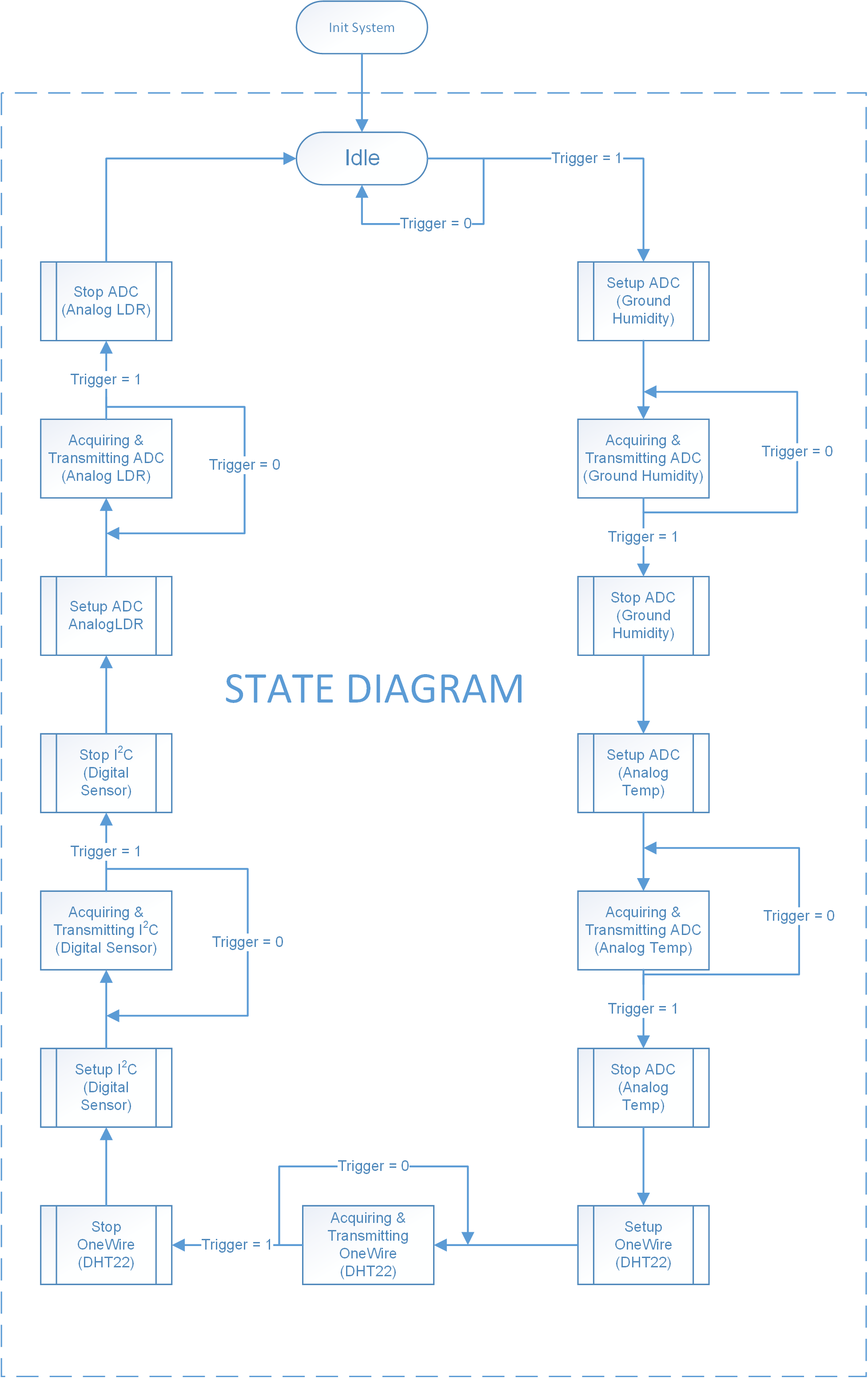
De esta manera, init\_system sería Setup() y while(1){} el Loop().

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 encargará de controlar el cambio de estado(output\_logic()) y otra para ejecutar las tareas del estado(state\_logic()).



## Estados

Tenemos un total de 5 sensores y seguiremos el patrón de diagrama de estados del apartado 2.1 por lo que tendremos 15 estados más el de Idle. En el siguiente diagrama de estado podemos ver que cuando acabamos de finalizar la adquisición y transmisión de datos de un sensor, pasamos a realizar las estas mismas tareas del siguiente sensor.(Ver diagrama)



### Idle

En el estado Idle, el MCU no está realizando ninguna tarea. Además cuando activemos el modo Sleep[Anexos] éste es el estado donde se estará. Y sólo una interrupción que haga que trigger sea '1' hará empiece a capturar datos.

### Setup ADC (Ground Humidity)

En el estado Setup ADC Ground Humidity, el MCU realiza un reset de todos los registros y variables que se utilizarán para medir en el siguiente estado la humedad de la tierra.

Para tomar las medidas de la temperatura seleccionaremos el canal de entrada del ADC AN0(apartado 4.2). Después configuraremos los registros del TIMER3, utilizando además un postscaler, para que podamos conseguir una frecuencia de muestreo de 4Hz[Anexo].

Una vez puesto en marcha el ADC y el TIMER3 pasaremos al estado A&T ADC(GH).

### Acquiring & Transmission ADC (Ground Humidity)

En el estado Acquiring & Transmission ADC Ground Humidity, el MCU tomará un total de 4 muestras y realizará la media de ellas. Por último, enviará por puerto serie los datos a la aplicación Labview y pondrá el flag trigger a '1' para pasar al estado siguiente.

### Stop ADC (Ground Humidity)

En el estado Stop ADC Ground Humidity, el MCU pondrá a '0' el flag que activa la toma de una muestra del ADC. Además, aprovecharemos que tenemos encendidos el TIMER3 y el ADC para no apagarlos porque los utilizaremos para tomar medidas en los siguientes estados.

### Setup ADC (Analog Temp)

En el estado Setup ADC Analog Temp, el MCU hará un reset a las variables que utilizarán para medir en el siguiente estado la temperatura ambiental. Finalmente, seleccionaremos el canal de entrada del ADC AN1(apartado 4.1).

### Acquiring & Transmission ADC (Analog Temp)

En el estado Acquiring & Transmission ADC Analog Temp, el MCU tomará un total de 4 muestras, aplicará la fórmula de conversión código-celcius y realizará la media de ellas. Por último, enviará por puerto serie los datos a la aplicación Labview y pondrá el flag trigger a '1' para pasar al estado siguiente.

### Stop ADC (Analog Temp)

En el estado Stop ADC Analog Temp, el MCU pondrá a '0' el flag que activa la toma de una muestra del ADC. Además, aprovecharemos que tenemos encendidos el TIMER3 y el ADC para no apagarlos porque los utilizaremos para tomar medidas en los siguientes estados.

### Setup OneWire (DHT22)

En el estado Setup OneWire DHT22, el MCU pondrá el puerto RB4 como puerto de salida y en '1' para preparar la secuencia de START. Además, encenderemos el TIMER2 para que lea los bits que nos envíe el DHT22.[Anexo]

### Acquiring & Transmission OneWire (DHT22)

En el estado Acquiring & Transmission OneWire DHT22, el MCU se comunicará con el sensor DHT22 utilizando el protocolo proporcionado por el fabricante. Una vez reciba los 5 bytes de datos, el MCU enviará los datos de temperatura ambiente y humedad relativa a la aplicación Labview. Finalmente, el MCU pondrá el flag trigger a '1' para pasar al estado siguiente.

### Stop OneWire (DHT22)

En el estado Stop OneWire DHT22, el MCU apagará el TIMER2 porque no se utilizará en ningún estado de los siguientes.

### Setup I2C (MCP9808)

En el estado Setup I2C MCP9808, el MCU encenderá la comunicación I2C configurando el clock a 100KHz, poniendo la MCU como Master y en Idle mode.[Anexo]

### Acquiring & Transmission I2C (MCP9808)

En el estado Acquiring & Transmission I2C MCP9808, el MCU se comunicará con el MCP9808 utilizando el protocolo I2C[Anexo]. Una vez recibidos los datos, el MCU utilizará la fórmula(apartado4.4) para convertir los 2Bytes de datos en celcius. Por último, enviará por puerto serie los datos a la aplicación Labview y pondrá el flag trigger a '1' para pasar al estado siguiente.

### Stop I2C (MCP9808)

En el estado Setup I2C MCP9808, el MCU apagará la comunicación I2C porque no se utilizará en ningún estado de los siguientes.

### Setup ADC (Analog LDR)

En el estado Setup ADC Analog LDR, el MCU hará un reset a las variables que utilizarán para medir en el siguiente estado la iluminancia. Finalmente, seleccionaremos el canal de entrada del ADC AN11(apartado 4.3).

### Acquiring & Transmission ADC (Analog LDR)

En el estado Acquiring & Transmission ADC Analog LDR, el MCU tomará un total de 4 muestras y realizará la media de ellas. Por último, enviará por puerto serie los datos a la aplicación Labview y pondrá el flag trigger a '1' para pasar al estado siguiente.

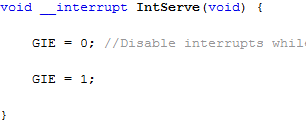
### Stop ADC (Analog LDR)

En el estado Stop ADC Analog LDR, el MCU apagará el ADC y TIMER3 utilizados para capturar los datos de temperatura ambiente, humedad relativa y iluminancia.

Por último el MCU volverá al estado Idle y estará en modo Sleep(durante un 1min) hasta que una interrupción del WDT despierte el MCU.

## Interrupciones

Las interrupciones son esenciales para hacer este tipo de diseños en los que utilizamos TIMERs, comunicación serie y algún botón. Los registros de interrupción se procesarán en la siguiente función[Anexos]:

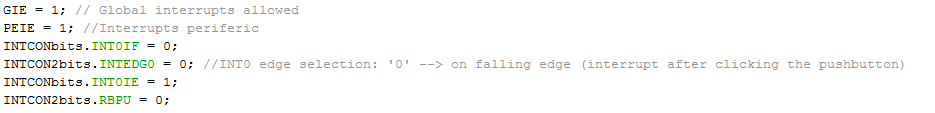


### RCIF

Cuando llegue un mensaje por puerto serie, el registro RCIF se pondrá en '1'.

### INT0IF

Cuando haya un cambio un flanco de bajada en el puerto INT0, el registro INT0IF se pondrá en '1'. Que INT0 haya se active con un flanco de bajada es algo que ha sido previamente configurado en la función InitSystem[Ver abajo, explicación en Anexos].



Configuración INT0.

### TMR3IF

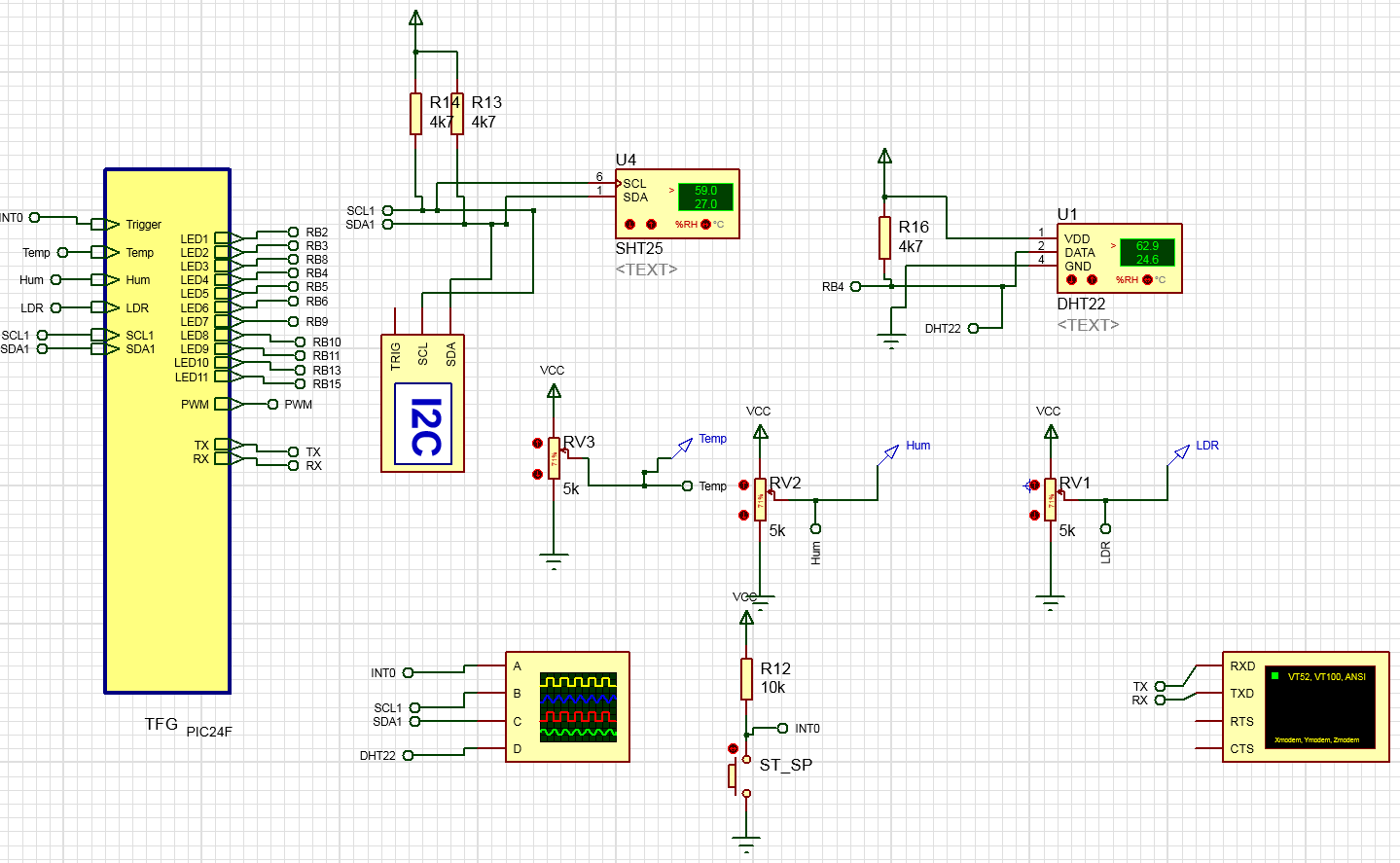
Cuando el TIMER3 haya acabado de contar los pulsos que le hemos configurado en los registros de configuración, el registro TMR3IF se pondrá en '1' y tendremos que volver a cargar los registros de configuración otra vez.[Anexo]

### TMR2IF

Cuando el TIMER2 haya acabado de contar los pulsos que le hemos configurado en los registros de configuración, el registro TMR3IF se pondrá en '1' y en este caso no hará falta volver a cargar ningún registro del TIMER[Anexos].

# Simulación con Proteus. paso a paso

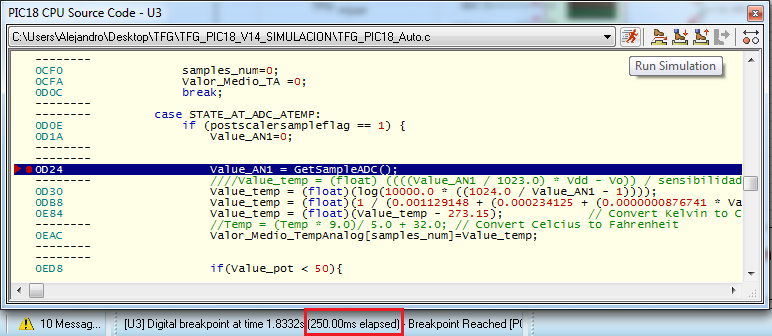
El simulador Proteus hemos nos permite simular nuestro PIC18F26K20, el sensor DHT22, la comunicación UART y otros sensores que no hemos utilizado en el diseño de nuestro sistema sensor pero que nos ha ayudado a probar el ADC y la comunicación I2C. Además, Proteus nos permite realizar debugging y de esta manera poder encontrar rápidamente soluciones a un bugs en el código.



El sensor analógico de temperatura KY-013, el sensor de humedad de la tierra y el sensor de iluminancia, los hemos simulado con un potenciómetro. Para simular la comunicación I2C, hemos utilizado el sensor de temperatura y humedad relativa SHT25.

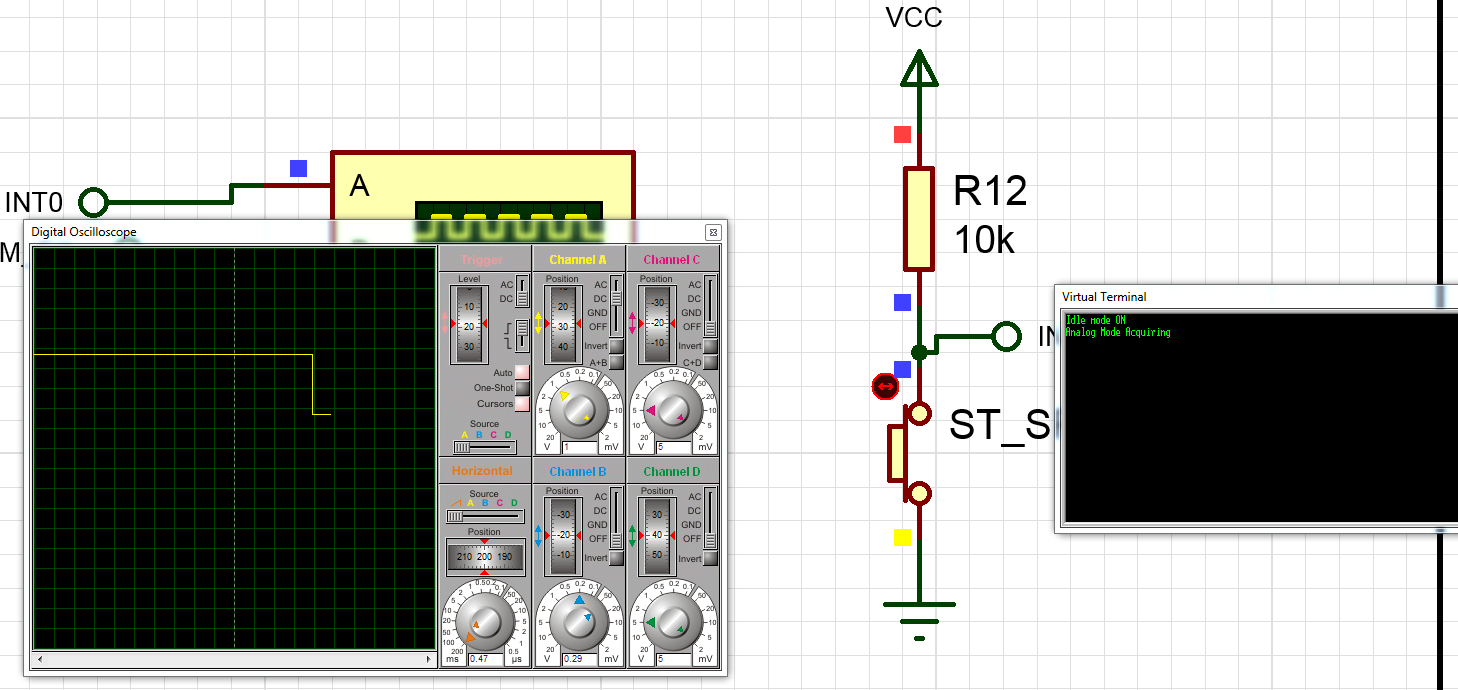
## FRECUENCIA DE MUESTREO CON TIMER3

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]



## 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.

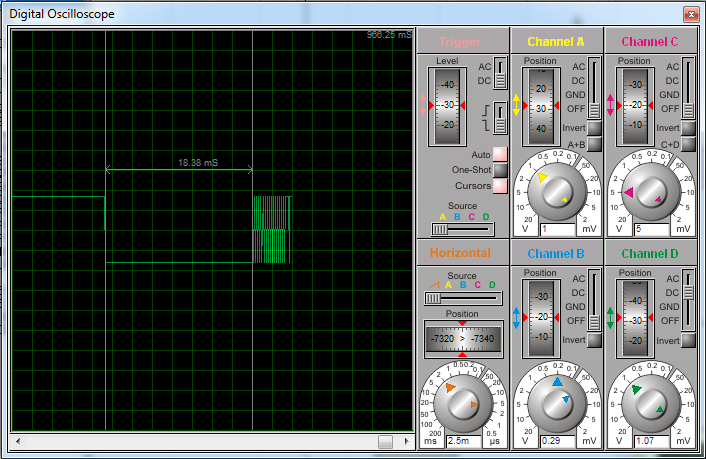


Cómo podemos ver, cuando hay un flanco de bajada en INT0, el PIC envía un mensaje por comunicación UART dejando de estar en el estado Idle.

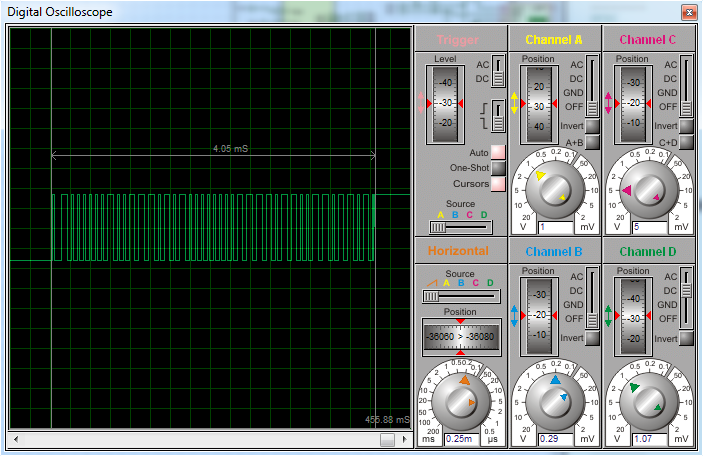
## COMUNICACIÓN CON EL SENSOR DHT22

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]



En segundo lugar comprobamos que los bits de datos se envían en pulsos cortos '0' y pulsos largos '1'.

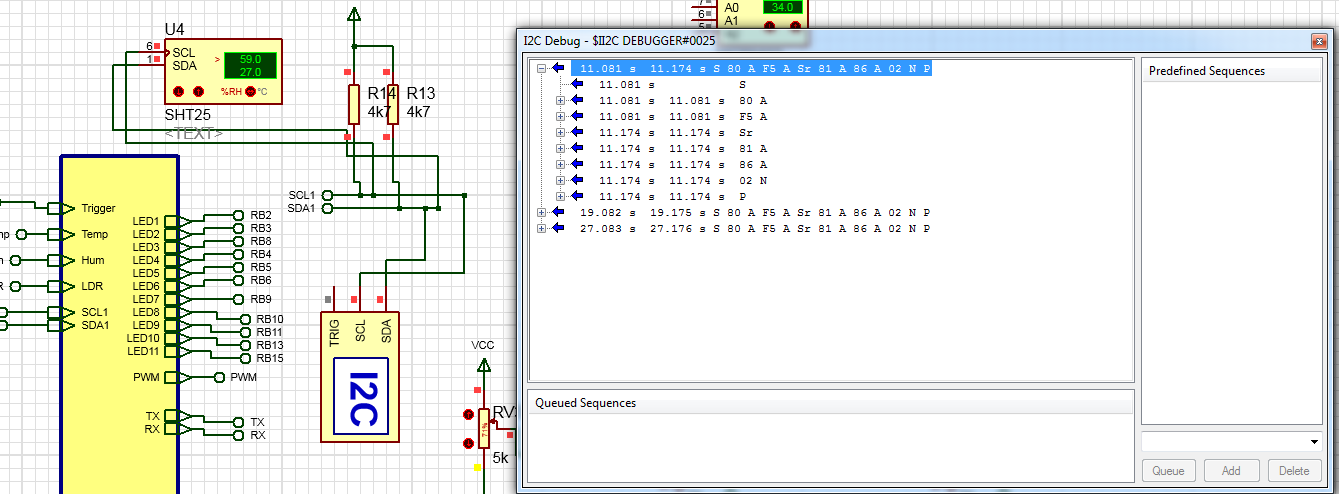


## COMUNCIACIÓN I2C

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]



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'.

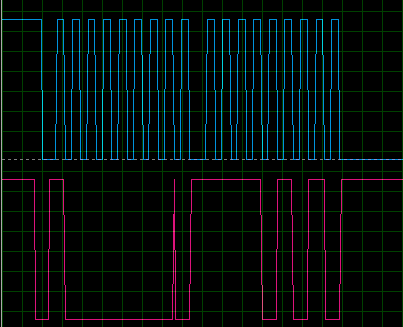
El datasheet del SHT25 por proporciona la fórmula que nos convierte el código recibido a humedad relativa.[Fórmula]

Srh es 8602 en hexadecimal y en decimal es 34306.

Por lo tanto comprobamos que funciona bien la comunicación.

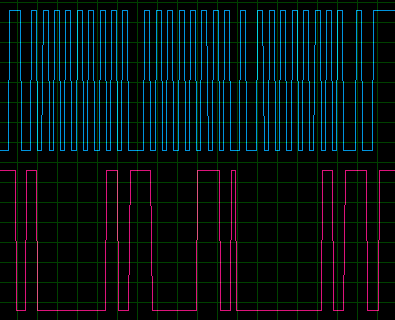
Otro aspecto a analizar de la comunicación I2C es el análisis de la trama en el osciloscopio. Primero mostraremos una captura de la primera parte de la comunicación y luego la segunda.

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]



(**S 80** A **F5** A, azul clock, rojo SDA)

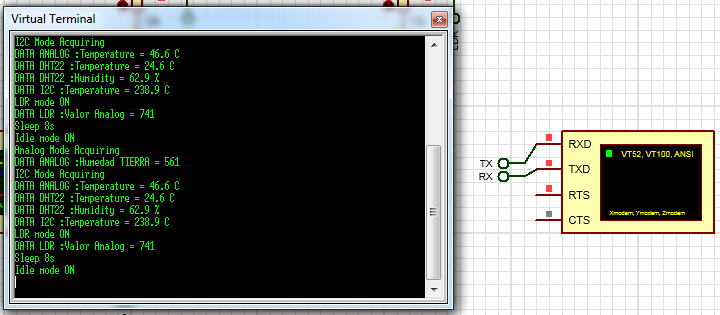
En la segunda parte de la comunicación avisamos al sensor de que estamos preparados para recibir los datos.[Anexos]



(**Sr** **81** A 86 **A** 02 **N** **P, azul clock , rojo SDA**)

## COMUNICACIÓN SERIAL

El Virtual Terminal de Proteus nos permite visualizar los mensajes que se transmiten y reciben por el puerto serie. [Ver virtual terminal]



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

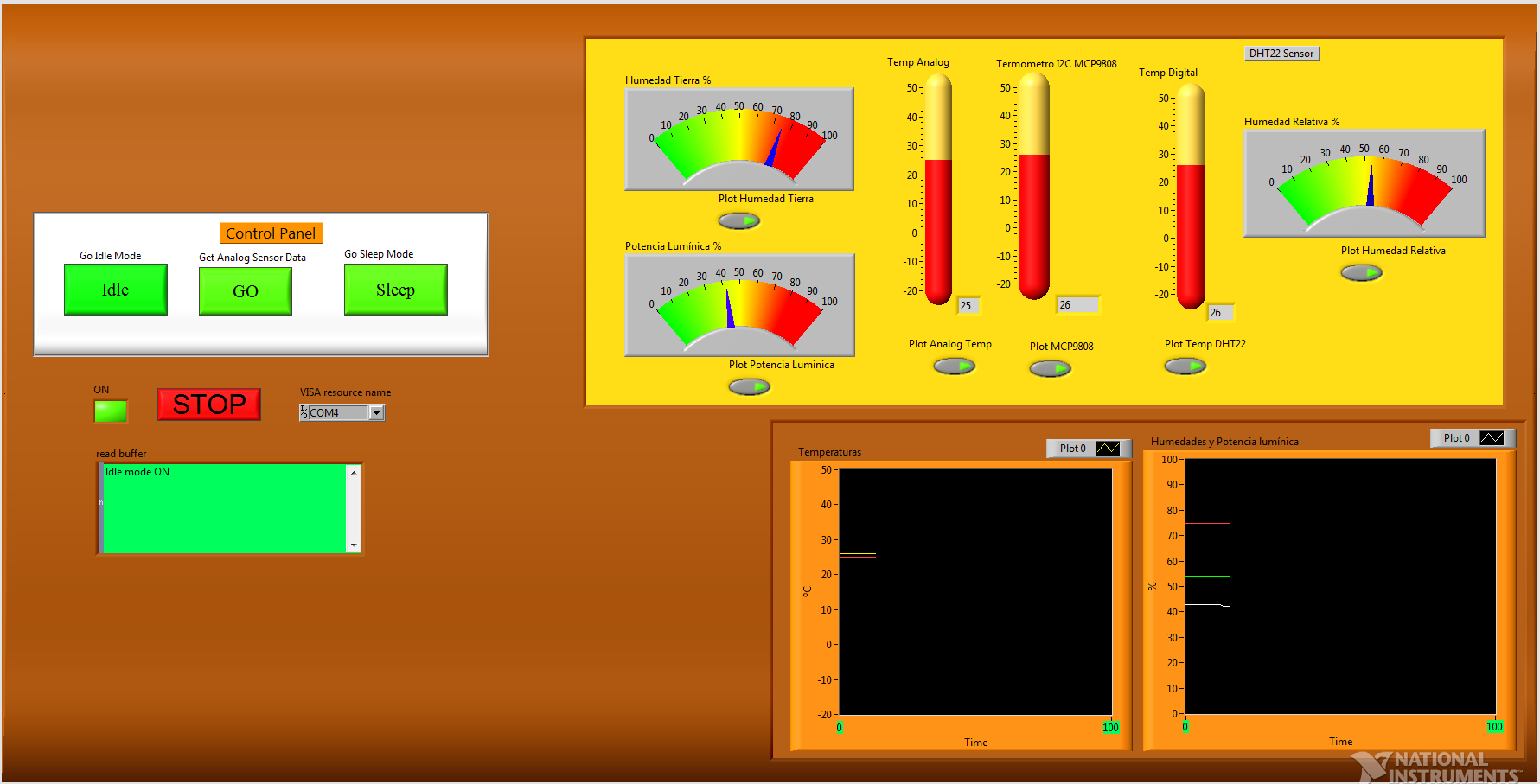
# Aplicación Labview

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

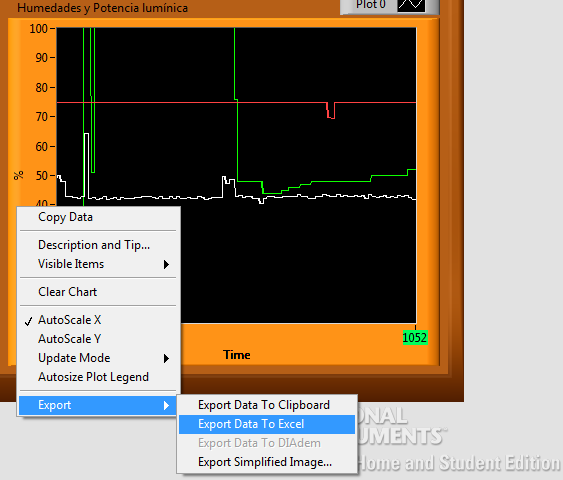
## Funcionalidades

La comunicación aplicación-MCU es bidireccional, desde el Panel de control podemos enviar 3 ordenes: poner el MCU en modo Idle, poner el MCU a tomar medidas y poner en modo Sleep el MCU.

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.



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

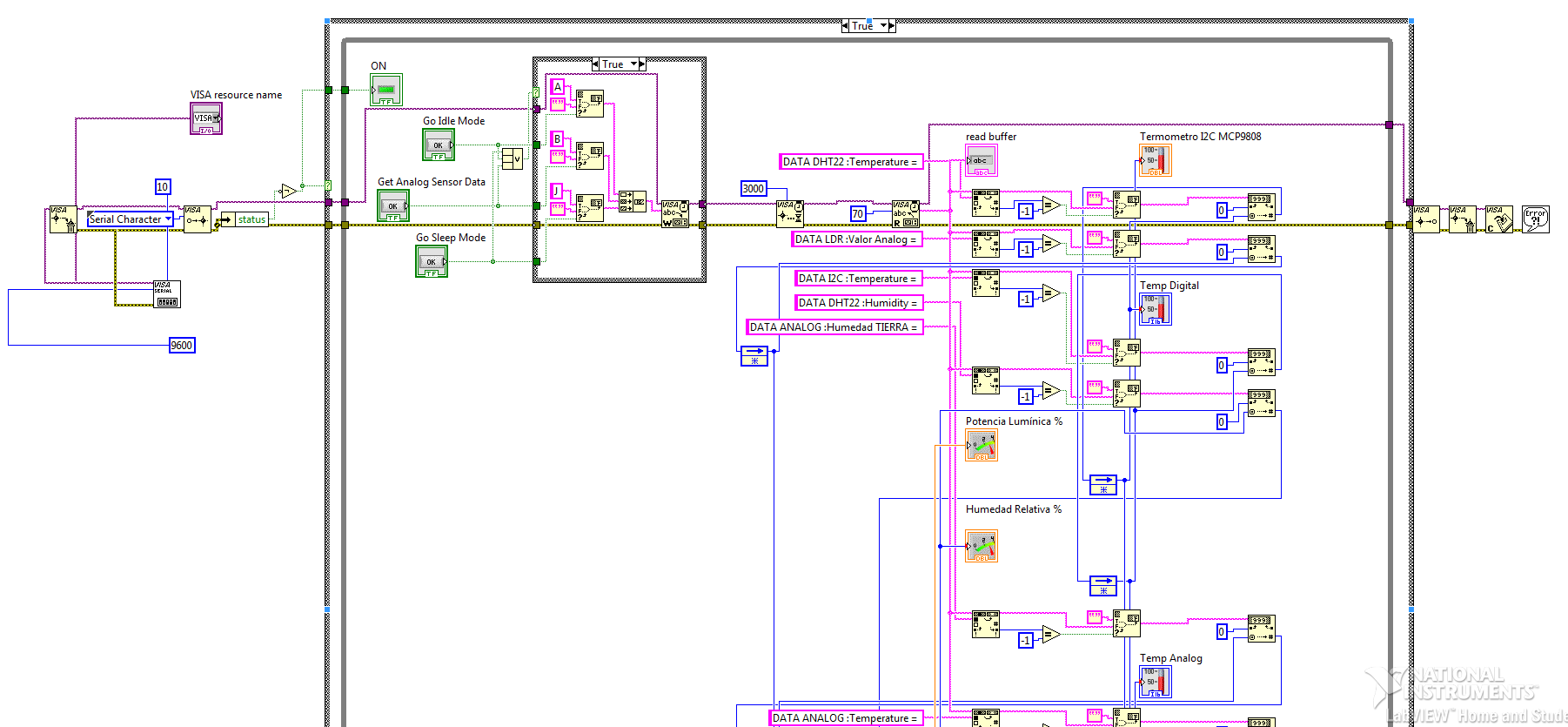


(Exportar excel)

## 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/)

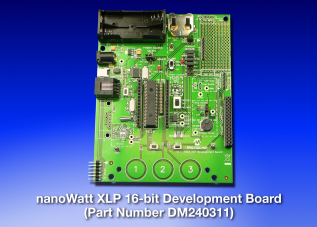
La creación de la aplicación consiste en ir uniendo módulos y al unir forman un diagrama de bloques.(Ver esquemático)



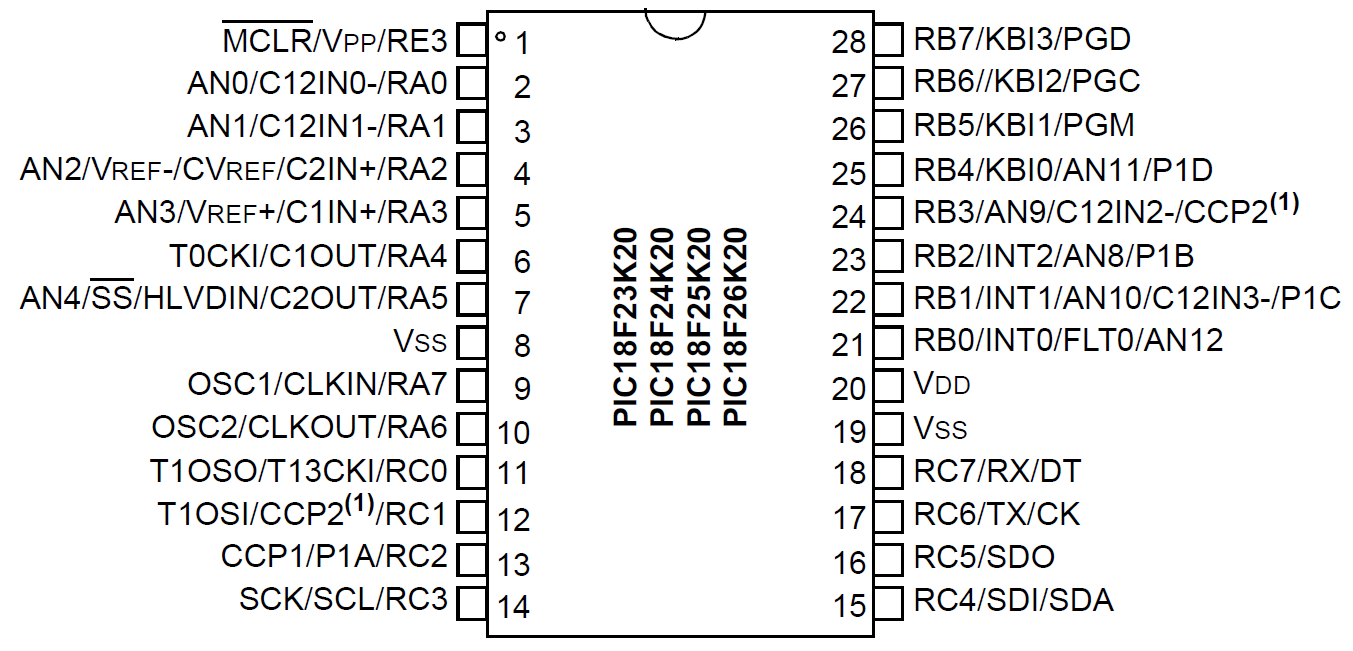
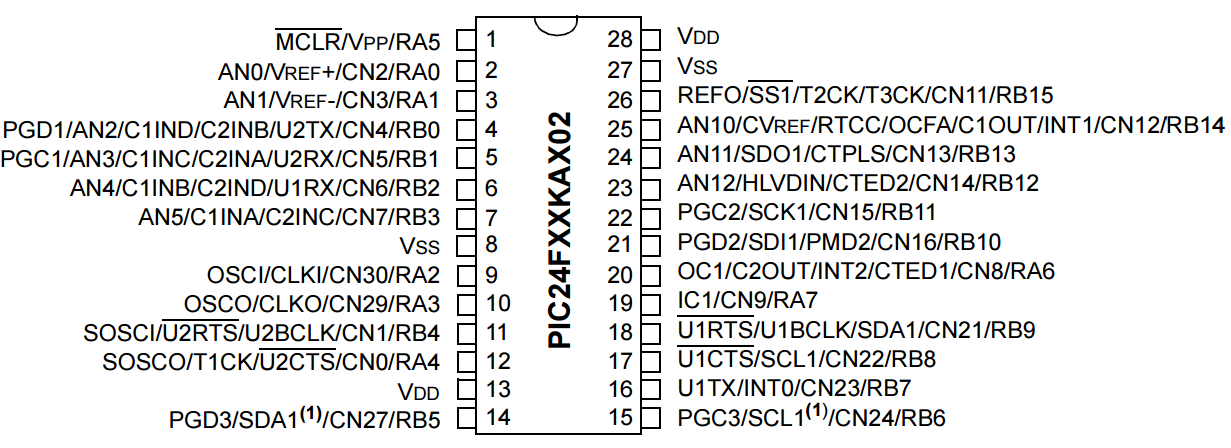
[Ver explicación de tallada de la aplicación ANEXO]

# Implementación 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.



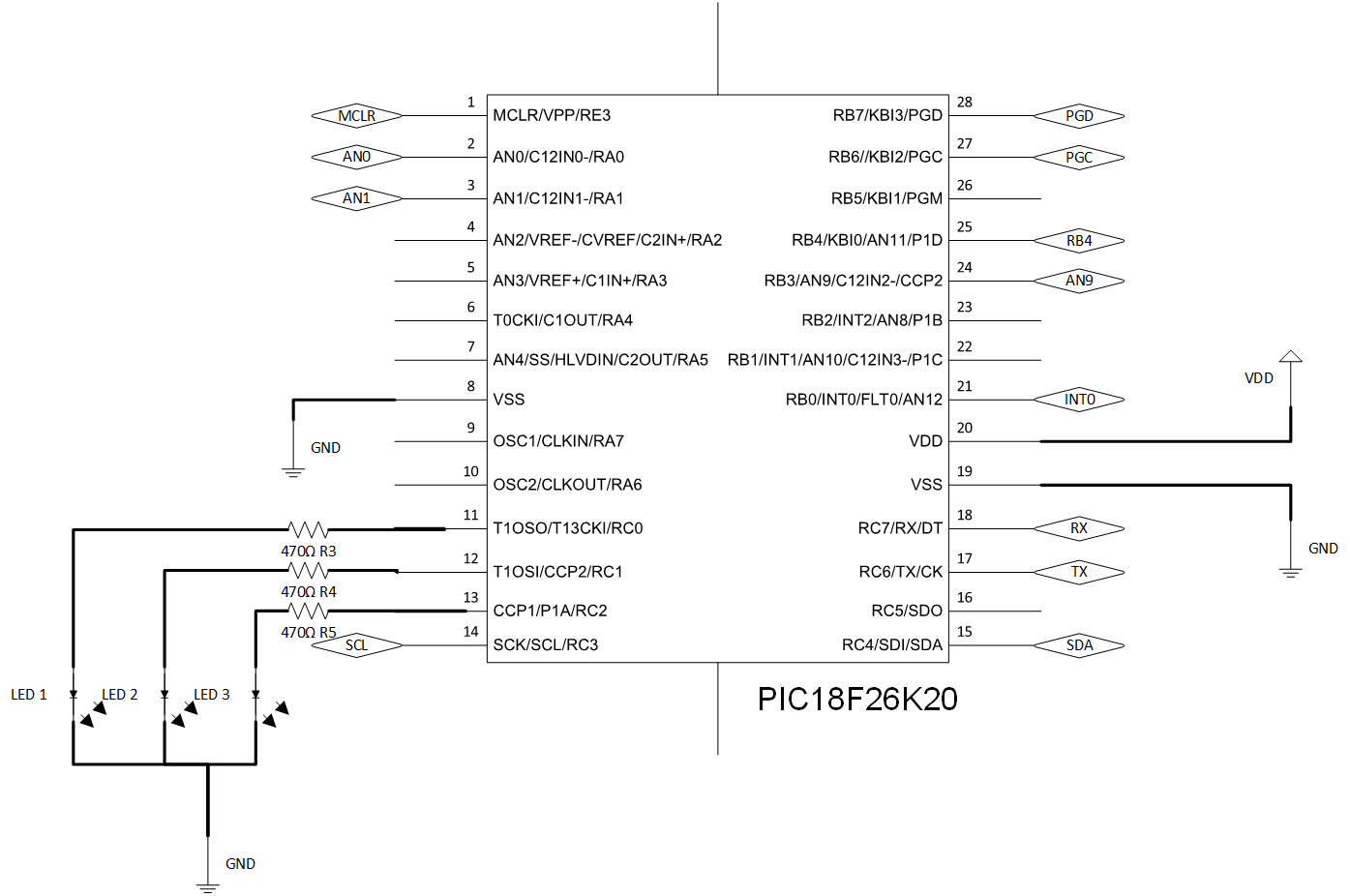
La solución a este problema fue trabajar con una protoboard para poder conectar las I/O del PIC18 con la placa de manera correcta.



(Diferencia de pines)

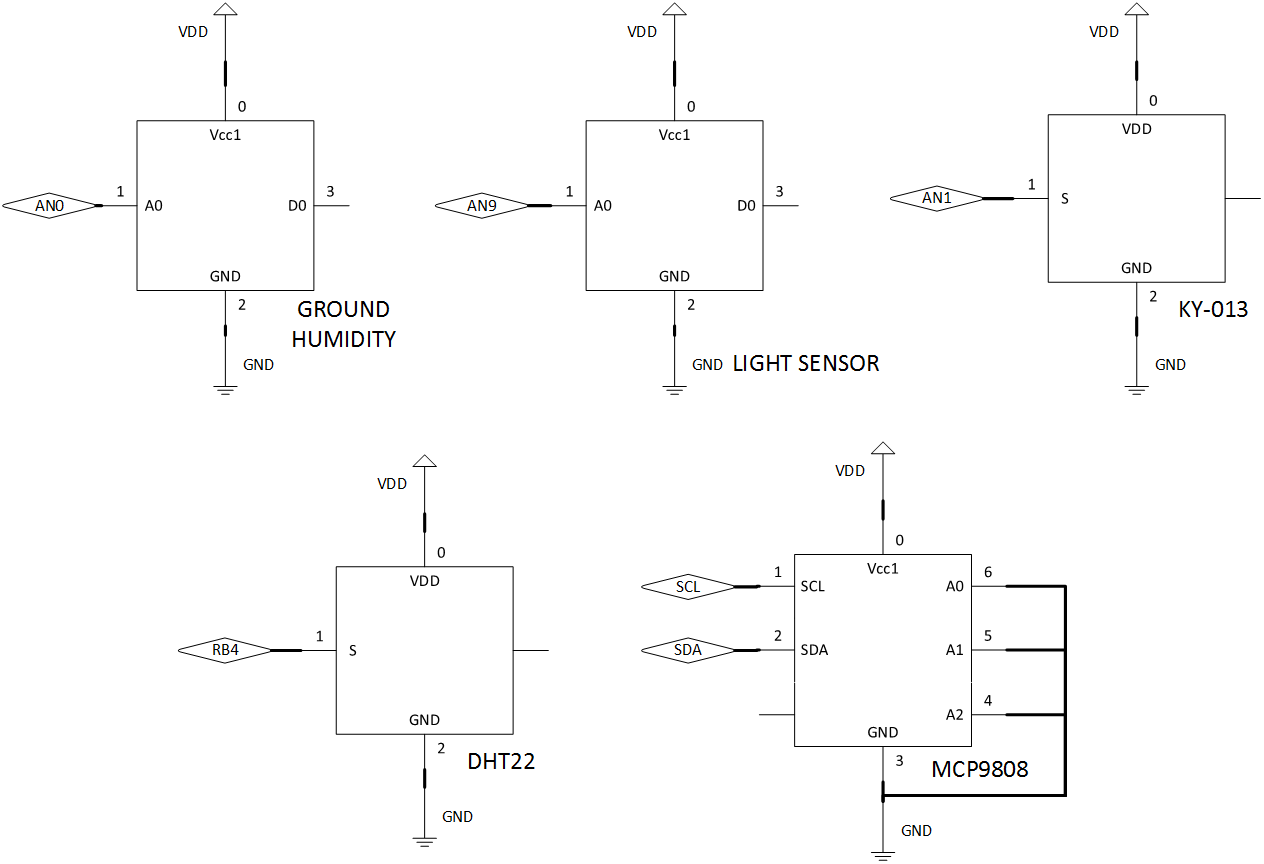
De este modo fue posible utilizar los dispositivos de la placa de entrenamiento, los más importantes: botones, conector pickit, potenciómetro y circuito de alimentación.

## Esquema de conexión de los sensores con el MCU



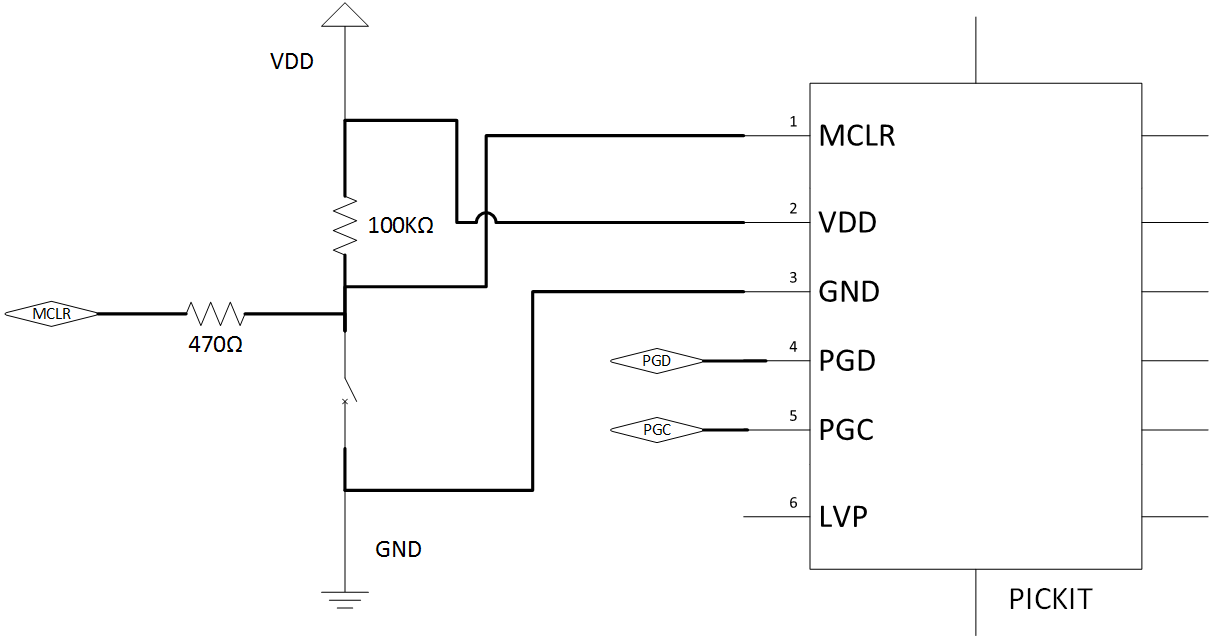
Además de conectar los sensores al MCU, hemos conectado unos LEDs en los puertos de salida RC0, RC1, RC2 para tener una referencia del estado en el que se encontraba nuestra máquina de estados.

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].

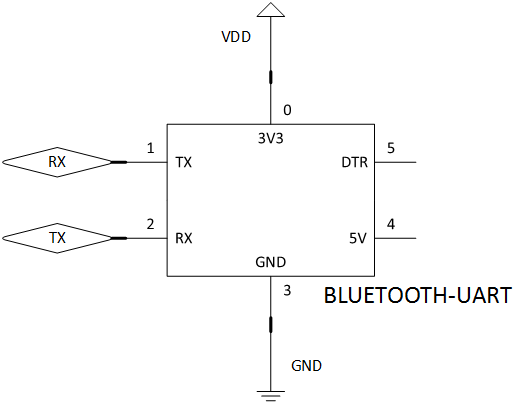


## Esquema de conexión PICKIT y UART-BLUETOOTH

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].



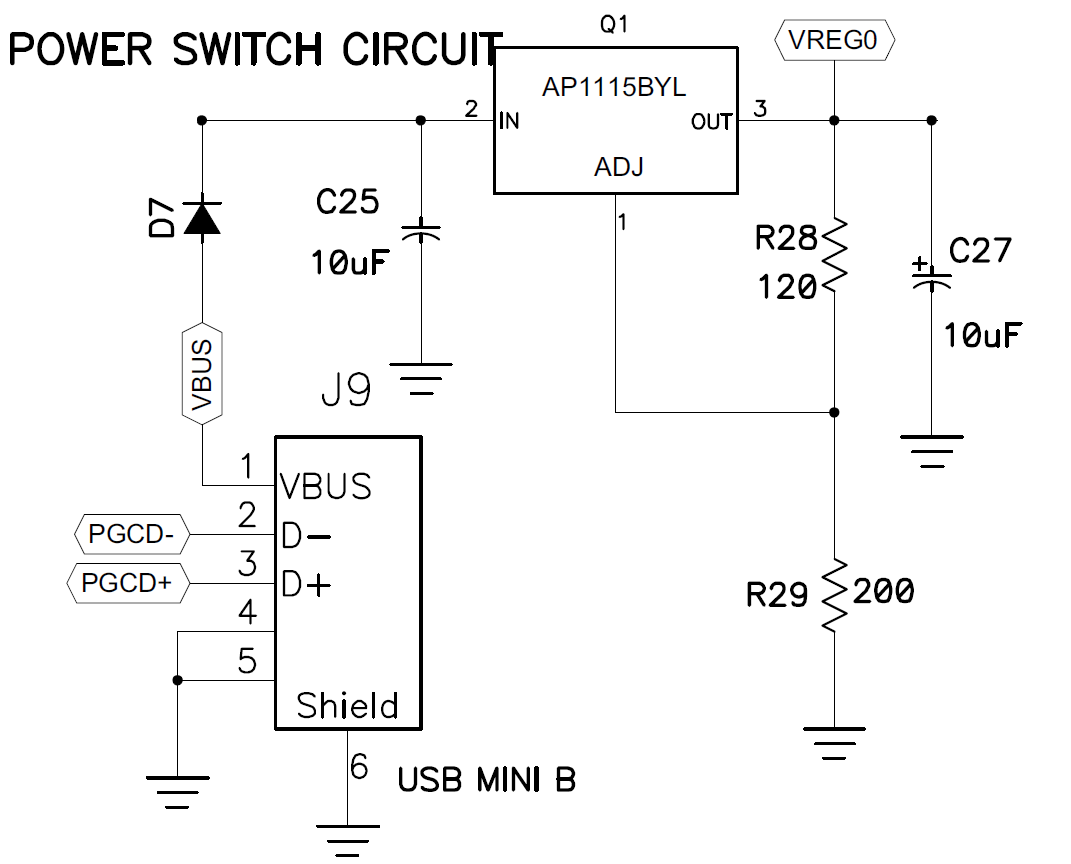
Cuando conectemos nuestro transmisor-receptor bluetooth a nuestro MCU, deberemos conectar el puerto RXmcu al TXbluetooth y el TXmcu al RXbluetooth[Ver].



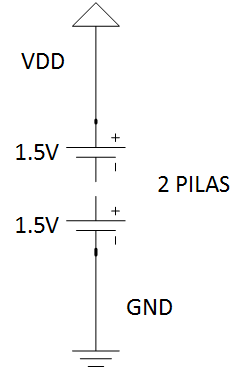
## Esquema de alimentación de la Development Board

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]



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]



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.



## Capturas de nuestra placa de pruebas definitiva

# Diseño del sistema sensor en placa de circuito impreso(PCB) con Eagle.

# mejoras:

## mejoras en la alimentacion de los sensores:

## mejoras a la hora de escoger el modo de transmision de los datos wireless

## mejora a la hora de realizar el energy harvesting.

# 

Objetivo:

descripcion

TEMAS

ESTructura del programa principal(ide microchip)

SENSORES:

temp

lum

hum

analog temp

ter hum.

I2c, 1 wire analog read, timers, sleep mode.

Simulacion con proteus. paso a paso

implementacion real en una placa de pruebas

Trasmison serial(USB o BLuetooth)

Aplicacion LABview paso a paso

PCB eagle.

mejoras:

mejoras en la alimentacion de los sensores:

mejoras a la hora de escoger el modo de transmision de los datos wireless

mejora a la hora de realizar el energy harvesting.