Código programable para el CC2530

Gabriel Andrés Hernández Rugeles

Escuela Colombiana de ingeniería Julio Garavito
Bogotá, Colombia
Gabriel.hernandez-r@mail.escuelaing.edu.co

***Resumen*—En este documento se explicará paso a paso como programar un CC2530, los requerimientos, sus funciones y el uso del conversor análogo a digital y la transmisión por antena a velocidad de 2.4Ghz.**

***Abstract*— In this document the step by step is explained as the programming in CC2530, the requirements, the functions and the use of the converter, the digital analysis and the transmission by antenna at a speed of 2.4GHz.**

I. INTRODUCCIÓN

El presente informe detallara el uso y el proceso de programación llevado a cabo para el microprocesador CC2530 usando el programa IAR WORKBENCH, sus parámetros respectivos y la funcionalidad la cual es una micro estación meteorológica con capacidad de transmitir sus valores que son: Temperatura, %LUZ, %HR, % Humedad del suelo; vía radio frecuencia.

II. OBJETIVOS

- Configurar e implementar el control de un CC2530 por medio de RF y ADC.
- Desarrollar adecuadamente el código por medio de IAR Workbench.
- Programar el microcontrolador por medio de un CCdebugger.

III. REQUERIMIENTOS

1. CC debugger para programar el microcontrolador.
2. Tarjeta de programación y aplicación usando un CC2530, este va conectado al CC debugger como lo indica la *Figura 1*, teniendo en cuenta que el programador suministra energía por su PIN 9

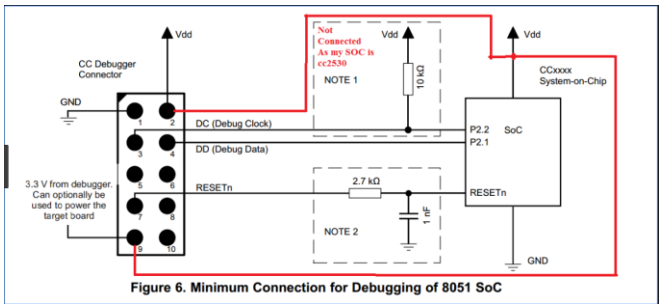

Figura 1

3. Se necesita un compilador para realizar el código, en este caso se usará el IAR Workbench para para arquitecturas 8051 la cual es la de nuestro microcontrolador.
4. Aplicación para descargar un archivo .hex desde el computador al microcontrolador por medio del CCdebugger el cual será el resultado del compilador, los cuales pueden ser : rf flash programmer 1 o 2.

IV.CODIGO DESARROLLADO

Como inicio se agregarán 2 librerías importantes *Figura 2* que son: “basic_rf_security.c” y “basic_rf.c” dichas librerías se encargarán del empaquetado de la información, esto quiere decir todas las capas que se transmitirán por medio de RF teniendo en cuenta un código de seguridad el cual servirá para poder decodificar el mensaje que se transmite.

```
/*****************************
* INCLUDES
*/

#include "basic_rf.h"
#include "basic_rf.c"
#include "basic_rf_security.h"
#include "basic_rf_security.c"
```

Figura 2

Una vez agregadas estas librerías inicializamos las variables mínimas para configurar el sistema de transmisión las cuales son

una dirección de origen y destino, el tamaño de los datos y una clave de seguridad si así se desea. Paso seguido empezaremos las rutinas de inicialización para el ADC.

Este periférico soporta conversiones de 7 a 14 Bits de resolución y una velocidad de conversión desde 4Khz hasta 30Khz, por lo tanto, al necesitar leer valores precisos y en un tiempo rápido usaremos una configuración de 30Khz y 12Bits; estas configuraciones se realizan a través de la modificación de registros de control en el microcontrolador como se muestra en la *Figura 3*.

```
static void init_ADC(){
  ADCCON1 = 00110011;
  ADCCON2 = 00110000;
  ADCCON3 = 00110000;
}
```

Figura 3

Ya configurado el periférico, y entendiendo que la resolución será de 14 bits, estos quedaran guardados en 2 registros independientes, ya que el microcontrolador tiene registros de 8 Bits máximo, al crear la función para leer el ADC solo se debe dar inicio a la conversión, y esperar a que finalice, esto se realiza con el bit ADCCON1.ST y el bit ADCCON1.EOC, concatenar estos dos registros usando desplazamientos de bits, ya que esta conversión queda guardada 8 bits en la parte alta 6 bits en la parte baja, y mostrar el resultado como se muestra en la *figura 4*.

```
static int leer_adc (){
  int tempL,tempH;
  ADCCON1.ST = 1;
  while(ADCCON1.EOC=0);

  tempL= ADCL; //parte alta
  tempH= ADCH; // parte baja
  tempH= (tempH << 6) | (tempL >> 2) ;
  return tempH;

}
```

Figura 4

Al finalizar éstas rutinas de inicialización lectura, podremos iniciar el código principal “Main” en el cual lo primero que se hará es inicializar cada proceso, periférico etc. Para este caso, se lleva a cabo la inicialización de la transmisión y el ADC como se muestra en la *Figura 5*.

```
// Config basicRF
    basicRfConfig.panId = PAN_ID;
    basicRfConfig.channel = RF_CHANNEL;
    basicRfConfig.ackRequest = TRUE;
#ifdef SECURITY_CCM
    basicRfConfig.securityKey = key;
#endif

   init_ADC();
```

Figura 5.

En esta inicialización del RF se ingresa la ID origen el canal (25) el cual es la velocidad de 2.4Ghz, un request de acknowledge y la clave de seguridad si es requerida, en ese orden. Asi mismo se usa la función “init_ADC();” que previamente fue creada.

Una vez se complete este paso (Inicializaciones), el microcontrolador esta listo para su uso, así que se empezara la transmisión a la espera de la recepción de un bit de START (Configurado por diseño en la posición 7 del dato de recepción), al recibir este bit se apaga la recepción para ahorro de energía y se empieza la toma de datos por medio de los diferentes canales(Puertos) en este caso son 4 canales diferentes los cuales se varían por medio del registro “ADCCON2.SCH” y el valor obtenido se multiplica por una constante para que la visualización de datos sea mas cómoda para el usuario, al poder ver porcentajes de humedad, luz o bien la temperatura en grados y no solo valores de voltaje (El cual es lo que nos entrega el microcontrolador), seguido a esto se puede proceder al empaquetado y envio de datos, añadiendo una dirección destino final como se puede ver en la *Figura 6*.

```
while(1){
 basicRfReceiveOn();

  while(1){
   while(!basicRfPacketIsReady());
    if(basicRfReceive(pRxData, APP_PAYLOAD_LENGTH, NULL)>0) {
      if (pRxData[7] = 1)     //posicion a definir
        break;
    }
  }
basicRfReceiveOff();

ADCCON2.SCH=0000;
pTxData[0]=leer_adc()*const_temp;

ADCCON2.SCH=0001;
pTxData[1]=leer_adc()*const_HuSue;

ADCCON2.SCH=0010;
pTxData[2]=leer_adc()*const_HuAir;

ADCCON2.SCH=0011;
pTxData[3]=leer_adc()*const_Luz;

basicRfSendPacket(Host_add, pTxData, APP_PAYLOAD_LENGTH);


}
```

Figura 6

Al completar esto, se tiene lo más básico en configuración e implementación para los requerimientos solicitados. Procedemos a compilar el código el cual nos dará un archivo .hex que servirá para programar el microcontrolador.
Para esto lo primero es conectar los puertos de programación de la tarjeta al CC debugger y este conectarlo al computador, seguido a esto usamos el programa FLASH PROGRAMER 2 *Figura 7* y procedemos a cargar el código en el microcontrolador.

Figura 7

V. SENSORES

Tablas de comportamiento para los diferentes tipos de sensores obtenidos, para la conversión de voltaje a sus diferentes medidas.

1. TEMPERATURA.

Figura 8.

Se usa un "NTC Thermistor" el cual tiene la siguiente tabla de resistencia vs temperatura *Figura 9:*

R/T No.	**8016**
T (°C)	$B_{25/100}$ = 3988 K
	R_T/R_{25}
−55.0	96.3
−50.0	67.01
−45.0	47.17
−40.0	33.65
−35.0	24.26
−30.0	17.7
−25.0	13.04
−20.0	9.707
−15.0	7.293
−10.0	5.533
−5.0	4.232
0.0	3.265
5.0	2.539
10.0	1.99
15.0	1.571
20.0	1.249
25.0	1.0000
30.0	0.8057
35.0	0.6531
40.0	0.5327
45.0	0.4369
50.0	0.3603
55.0	0.2986

Figura 10

Con esta tabla sabemos la resistencia que tendrá el sensor multiplicando por 10kΩ el valor de la gráfica, ya que este es el R_{25} con esta resistencia obtenemos los valores reales de voltaje en el circuito a temperaturas ambientales, esto se traduce en el microcontrolador por valores entre 0 y 16384 (2^{14}) como se muestra en la *Figura 11*:

Temperatura	Resistencia(KΩ)	Voltaje	Valor final en el micro
-15	72,93	2,02	10028,99394
-10	55,33	2,02	10028,99394
-5	42,32	2,02	10028,99394
0	32,65	2,02	10028,99394
5	25,39	2,02	10028,99394
10	19,9	1,91	9482,860606
15	15,71	1,59	7894,109091
20	12,49	1,26	6255,709091
25	10	0,932	4627,238788
30	8,057	0,605	3003,733333
35	6,531	0,292	1449,735758
40	5,327	0,00617	30,63311515
45	4,36	0,00617	30,63311515
50	3,603	0,00617	30,63311515

Figura 11

Al analizar la *Figura 11* podemos observar que solo es útil los valores de temperatura entre 10 a 35 grados, con estos valores se obtendrá una recta de tendencia para obtener una ecuación para poder calibrar este sensor en el código del microcontrolador.

Resultando la siguiente ecuación para calibrar el sensor de temperatura:

$$Temperatura = \frac{Valor\ ADC}{-321.32} + 39.51$$

Ecuación 1

2. LUZ

Para la luz se usará una Fotocelda VT43N1

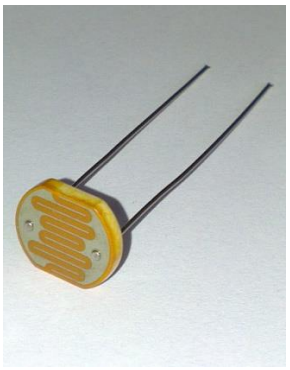

Figura 12

La cual es de coeficiente resistivo negativo, lo que quiere decir que a mayor luz menor será su resistencia, la cual tiene una curva de tendencia que se puede aproximar a una de tipo lineal, variando su resistencia de 0 a 4kΩ dando un voltaje máximo de referencia = 2.25 voltios por lo tanto en el microcontrolador encontramos el valor 11171 teniendo la siguiente ecuación dado un porcentaje completo de luz 100% y un mínimo de 0 %

$$\%Luz = \frac{Valor\ ADC * 100}{11171}$$

Ecuacion 2

3. HUMEDAD DEL SUELO

Se usara el sensor SEN0193 de humedad mostrado en la *figura 13*

Figura 13

Este nos entregara una señal de voltaje que nos dira que tan húmedo esta el suelo entre 2.1 V y 1.65 V

Los cuales al pasar por el circuito el microcontrolador los tomara como valores de 9930 y 4310 teniendo asi la calibración siguiente

$$\%HumedadSuelo = \frac{Valor\ ADC - 4310}{5620} * 100$$

Ecuacion 3

4. HUMEDAD RELATIVA(AIRE)

Para este sensor usaremos el HIH-5030-001

Figura 14

Este sensor nos entregara un voltaje de 0.5 a 2.5 V en su salida y al ser procesado por nuestra tarjeta tendremos valor de 0.0045 V y 2.45 V por lo tanto sus valores de ADC serán: 23 y 12164 y su ecuación:

$$\%HR = \frac{Valor\ ADC - 23}{12141} * 100$$

Ecuacion 4

VI. Referencias

[1] https://co.mouser.com/datasheet/2/400/NTC_Probe_ass_M703-1382262.pdf

[2] http://www.ti.com/lit/ds/symlink/cc2530.pdf

[3] http://www.ti.com/lit/ug/swru191f/swru191f.pdf

[4] http://www.ti.com/product/CC2530/technicaldocuments

[5] https://co.mouser.com/