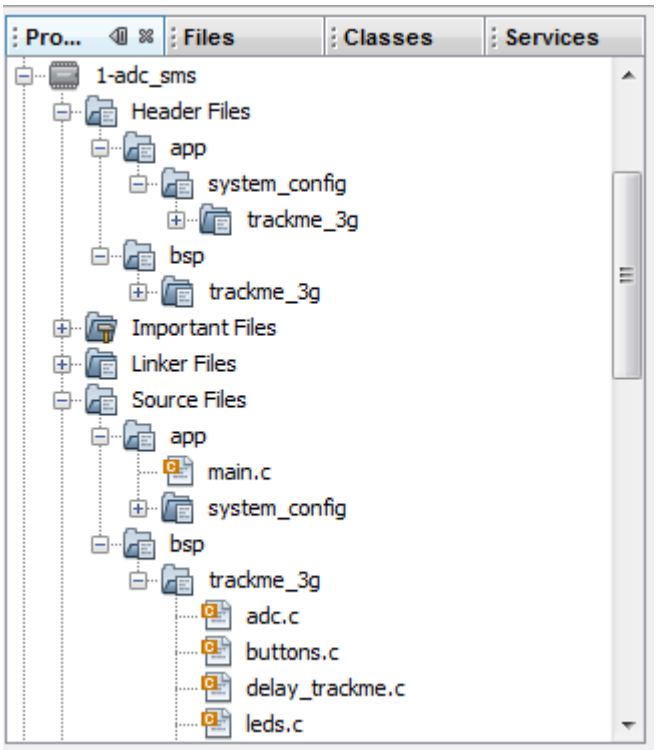
Documentación de los Proyectos

El siguiente documento define el formato que presentan los Proyectos desarrollados en MPLABX para la placa TrackMe 3G. Estos están escritos enteramente en XC8 y compilados para el PIC18F26J50.

Estructura de Archivos

La estructura de archivos de los proyectos están hechos de manera similar a los proyectos de la Librería de Aplicaciones de Microchip. Los proyectos más simples se encuentran separados en dos carpetas, una llamada bsp y la otra app. La carpeta bsp contiene los códigos fuentes que son comunes a todos los proyectos y tiene en su interior las librerías genéricas del dispositivo a programar. Es así que en esta carpeta encontramos los fuentes de la librería del modulo SIM5320 o las funciones para el manejo de la USART y el ADC, entre otras cosas, de la placa TrackMe. Es decir, que si en nuestro proyecto queremos utilizar la USART para comunicarnos con el modulo, simplemente agregamos las funciones de esta librería y obtendremos la comunicación con el modulo. Lo mismo para el modulo SIM, si queremos utilizar las funciones generales del modulo SIM solamente debemos agregar la librería sim5320.h el cual contiene un set de funciones para el manejo de comandos para enviar un SMS o, por ejemplo obtener las coordenadas GPS. Separar estas librerías en distintos archivos permiten si en un futuro se utiliza otro modulo SIM para la comunicación, simplemente se debe agregar la nueva librería simXXXX.h para adaptar el código a la nueva librería.

La otra carpeta se llama app y contiene los códigos fuentes propios del proyecto, aquí básicamente se encuentra el archivo main.c y system.c que se utilizara para la inicialización y las tareas que deben ejecutar el sistema. Estos son los fuentes que deben modificar el programador para crear los nuevos proyectos.



Estructura de la carpeta app

La estructura de archivo de la carpeta app debe ser el siguiente: En su interior se debe encontrar el main.c, que es donde se inicia la aplicacion, y una carpeta llamada "system". Esta carpeta a su vez debe contener otra carpeta cuyo nombre debe hacer una referencia al hardware que administrara, en este caso lo llamamos trackme_3g. Un archivo main.c de este proyecto debe tener el siguiente formato:

```
MAIN_RETURN main(void)
{

    SYSTEM_Initialize(SYSTEM_STATE_START);

    SIM_On();
    SIM_Init();

    while(1)
    {
        SYSTEM_Tasks();


    }//end while
}//end main
```

En este codigo se pueden visualizar dos funciones del sistema: SystemInit() y SystemTask(). Estas funciones se definen en el archivo system.c y son las encargadas de realizar la inicializacion del sistema(Aqui se configuran lo pines GPIO, el ADC o la USART). La funcion SystemTask debe contener las tareas del sistema que deben ejecutarse

periódicamente. Se debe mantener este formato en el archivo main para todos los proyectos con la placa TrackMe, ya que de esta manera el código se mantiene ordenado.

Estructura de la Carpeta bsp

Esta carpeta es la que contiene los archivos fuentes que son comunes a todos los proyectos. Aquí se encontraran librerías para el manejo de la USART para el modulo SIM y para el FTDI de la placa, funciones para el conversor ADC y la librería del modulo sim5320. Aquí también debe existir otra carpeta cuyo nombre nos haga referencia al hardware a la que está destinada las librerías de los archivos fuentes, que en este caso llamamos trackme_3g.

Crear un nuevo proyecto a partir de los Ejemplos:

Para crear un nuevo proyecto desde los ejemplos se deben realizar los siguientes pasos:

1- Copiar y pegar en el "mismo directorio" la carpeta que contiene los fuentes editables del proyecto, cambiar el nombre de la carpeta que contiene el proyecto.

2- Ejecutar el MPLABX y abrir el proyecto

3- Hacer click derecho sobre el proyecto y seleccionar entre las opciones “Rename”

4- Modificar y compilar el proyecto.

Librería de la placa TrackMe

Librería SIM5320

En el siguiente apartado se detallan las funcionalidades implementadas en la librería del modulo sim5320

Estructura simResponse.

La librería del sim5320.h define una estructura declarada de manera global para tener acceso a los valores que se capturan del modulo SIM como la respuesta a los comandos, coordenadas del GPS y si se recibe un mensaje de texto, esta estructura almacena el

numero telefónico y el mensaje recibido. Para utilizarlo, simplemente se debe declararla como variable externa, con la siguiente sintaxis.

```
extern sSimStruct simResponse;      //Declaracion global de la estructura
```

Estas son las variables que componen la estructura

sSimStruct

.-*bufferRes*: Buffer de recepción de las respuestas del modulo SIM

.-*enterResponse*: flag que indica que se completo una respuesta.

.-*newResponse*: booleano para indicar que se requiere una nueva respuesta

.-*gpsInfo*: estructura que contiene datos del GPS

.-*latitud*: almacena latitud de la posición actual

.-*longitud*: almacena la longitud de la posición actual.

.-*NSIndicator*: indica polaridad N o S

.-*EWIndicato*r: indica polaridad W o E

.-*horaUTC*: almacena el string con la hora UTC

.-*date*: almacena el string con la fecha tomada del GPS

.-*speed*: contiene la velocidad actual del dispositivo indicado por el GPS en Km/h

.-*smsRead*:Estructura con datos del último SMS recibido

.-*newSMS*: Indicador de que se completo la recepción de un SMS

.-*receiveSMS*: Indicador de que se está recibiendo un nuevo SMS

.-*phoneSMS*: Almacena el número telefónico del dispositivo que envió el mensaje

.-*messenger*: almacena el mensaje recibido

.-*maxCRLF*: Contador máximo de caracteres <CR><LF> que puede variar según la respuesta de un determinado comando

.-*countCRLF*: Variable que contiene la cantidad de caracteres <CR><LF> recibidos hasta el momento, cuando este valor alcanza el maxCRLF, la respuesta completa se almacena en bufferRes.

Función: SIM_On

-**Firma de la funcion:** void SIM_On(void)

-Descripción: Enciende el modulo SIM. Realiza la secuencia de encendido en el PIN POWER_KEY del modulo y espera un tiempo suficiente hasta que sea posible comunicarse con el SIM.

-**Parámetros de entrada:** Ninguno

-Retorno: Ninguno

-Ejemplo de aplicación.

```
//Encendido del modulo SIM
SIM_On();
printf("Modulo Encendido\r\n");
```

Función: SIM_Off

-Firma de la función: void SIM_Off(void)

-Descripción: Apaga el modulo SIM. Realiza la secuencia de apagado en el PIN POWER_KEY del modulo y espera un tiempo suficiente hasta que se deshabiliten las comunicaciones.

-Parámetros de entrada: Ninguno

-Retorno: Ninguno

-Ejemplo de aplicación.

```
//Apagado del modulo SIM
SIM_Off();
printf("Modulo Apagado\r\n");
```

Función: SIM_Init

-Firma de la función: void SIM_Init(void)

-Descripción: Realiza la inicialización del modulo SIM. Configura el modulo SIM para que no haya ecos, los SMS se manejen en formato Texto y ante la llegada de un mensaje lo envié por la USART y no lo almacene en memoria.

-Parámetros de entrada: Ninguno

-Retorno: Ninguno

-Ejemplo de aplicación.

```
SIM_Init();
```

Función: SIM_Sync

-Firma de la función: void SIM_Sync(void)

-Descripción: Sincroniza la comunicación con el modulo SIM, envía varios comandos AT para sincronizar velocidades.

-Parámetros de entrada: Ninguno

-Retorno: Ninguno

-Ejemplo de aplicación.

```
SIM_Sync();
```

Función: SIM_WriteCommand

-Firma de la funcion: int SIM_WriteCommand(const char* command,const char* value)

-Descripción: Envía un comando AT con el formato AT+<command>=<value>\r

-Parámetros de entrada:

command: String con el comando AT a enviar
value: valor en String del comando.

-Retorno: SIM_STATUS_OK si se pudo enviar el comando

-Ejemplo de aplicación.

```
int stat;

stat = SIM_WriteCommand("CMGF", "1");
```

Función: SIM_ReadResponse

-Firma de la función: int SIM_ReadResponse(uint8_t maxCountCRLF)

-Descripción: Lee la respuesta del modulo SIM ante un comando

-Parámetros de entrada: maxCountCRLF: Cantidad de caracteres <CR> y <LF> que contiene la respuesta

-Retorno: SIM_STATUS_OK si se recibió OK

-Ejemplo de aplicación.

```
int stat;

stat = SIM_ReadResponse(CRLF_COUNT_GPS_GPRMC);
```

Función: GSM_SendSMS

-Firma de la función: int GSM_SendSMS(char* cel_phone, char* message)

-Descripción: Envia un mensaje SMS desde el modulo SIM

-Parámetros de entrada:

cel_phone: Número de teléfono
message: Mensaje de texto

-Retorno: SIM_STATUS_OK si se pudieron enviar los comandos

-Ejemplo de aplicación.

```
    char mensaje[256];
    char telefono[16]="1234567890";
    int stat;

    stat = GSM_SendSMS(telefono,mensaje);
```

Función: GPS_UpdateData

-Firma de la función: int GPS_UpdateData(void)

-Descripción: Actualiza los valores de la estructura gpsInfo.

-Parámetros de entrada: Ninguno.

-Retorno: SIM_STATUS_OK si se obtuvo respuesta del modulo SIM

-Ejemplo de aplicación.

```
int stat;

stat = GPS_UpdateData();

sprintf(mensaje,"%shttp://maps.google.com/maps?q=%s,%s",mensaje,
simResponse.gpsInfo.latitude, simResponse.gpsInfo.longitude);
```

Función: GPS_UpdateNMEAData

-Firma de la función: int GPS_UpdateNMEAData(void)

-Descripción: Actualiza los valores de la estructura gpsInfo desde la trama NMEA

-Parámetros de entrada: Ninguno.

-Retorno: SIM_STATUS_OK si se obtuvo respuesta del modulo SIM

-Ejemplo de aplicación.

```
int stat;

stat = GPS_UpdateNMEAData();

sprintf(mensaje,"%shttp://maps.google.com/maps?q=%s,%s",mensaje,
simResponse.gpsInfo.latitude, simResponse.gpsInfo.longitude);
```

Función: GPS_DMMtoDDD

-Firma de la función: void GPS_DMMtoDDD(char* valor)

-Descripción: Realiza la conversión de coordenadas GPS que se encuentran en notación grados y minutos decimales(DMM), obtenidos desde la trama NMEA, a grados Decimales (DDD).

-Parámetros de entrada: Coordenadas en formato DMM(grados y minutos decimales)

-Retorno:

-Ejemplo de aplicación.

```
if(GPS_UpdateNMEAData() == SIM_STATUS_OK)
{
    //Cambio de formato las coordenadas
    GPS_DMMtoDDD(simResponse.gpsInfo.latitude);
    GPS_DMMtoDDD(simResponse.gpsInfo.longitude);
}
```

-Función: SIM_Tasks

-Firma de la función: void SIM_Tasks(void)

-Descripción: Tarea del modulo SIM, realiza un constante monitoreo para saber si se recibió un mensaje SMS. Una vez que se recibe el mensaje, este lo procesa y llena la estructura simResponse con los datos del mensaje recibido. Esta tarea trabaja en conjunto con la rutina que recibe los caracteres por la USART, ya que esta le indica cuando se recibe un SMS.

Nota Importante: Los mensajes recibidos no deben contener coma (,) o de lo contrario los mensajes se verán truncados.

-Parámetros de entrada:

-Retorno: Actualiza la estructura simResponse con los datos de un nuevo mensaje

-Ejemplo de aplicación.

Librería de Ubidots

En el siguiente apartado se detallan las funcionalidades implementadas en la librería ubidots de la placa de desarrollo TrackMe 3G

Mediante esta librería podemos sincronizar datos con los servidores de ubidots utilizando simplemente un token de autenticación y suministrando el ID de la variable.

Función: UBI_SendData

-Firma de la función: int UBI_SendData(const char* token,const char* auth_id_var, float value, BOOL coord_gps)

-Descripción: Esta funcion se encarga de abrir un socket al servidor HTTP de ubidots y de enviar los datos sumnistrados por el usuario.

-Parámetros: Se debe ingresar el token del dispositivo, el ID de la variable, el valor a enviar y si se desea enviar las coordenadas GPS junto con el valor de la variable

-Retorno: Retorna SIM_STATUS_OK si se putdo realizar el envio de los datos

-Ejemplo de aplicación.

```
MAIN_RETURN main(void)
{
    char mensaje[SIZE_SMS];

    SYSTEM_Initialize(SYSTEM_STATE_START);

    SIM_On();
    SIM_Init();

    //Configurar datos del APN
    delay_s(1);
    SIM_WriteCommand(CGSOCKCOUNT, APN_PERSONAL);
    SIM_WriteCommand(CSOCKAUTH, APN_USER_PASS_PERSONAL);

    //Se inicia una sesion GPS
    delay_s(1);
    SIM_WriteCommand(CGPS, OPEN_GPS_SESSION);

    while(1)
    {
        SYSTEM_Tasks();


        if(BUTTON_ActivatePanic())
```

```
        {
            LED_On(LED_USB_DEVICE_STATE);
            UBI_SendData(UBI_TOKEN_AUTH_TRACKME, UBI_ID_TEMP,
      ADCtrackme.temp, TRUE);
        }

    }//end while
}//end main
```

Función: UBI_SendPOST

-Firma de la función: int UBI_SendPOST(const char* token, const char* auth_id_var, float value, BOOL coord_gps)

-Descripción: Esta función arma el mensaje POST a enviar al servidor HTTP de ubidots. Según sea el valor booleano de coord_gps envia o no las coordenadas GPS junto con el valor de la variable.

-Parámetros: Se debe ingresar el token del dispositivo, el ID de la variable, el valor a enviar y si se desea enviar las coordenadas GPS junto con el valor de la variable

-Retorno: Retorna SIM_STATUS_OK si se pudo realizar el envió de los datos

-Ejemplo de aplicación.

```
UBI_SendPOST(token, auth_id_var, value, coord_gps);
```

Librería del ADC de la placa trackme.

En el siguiente apartado se detallan las funcionalidades implementadas en la librería ADC de la placa de desarrollo TrackMe 3G

Estructura ADCtrackme.

Esta estructura contiene los resultados de la conversión AD de los periféricos analógico de la placa TrackMe 3G. Esta estructura se actualiza mediante la ejecución periódica de la tarea ADC_task() que provee la librería. La estructura ADCtrackme contiene los siguientes campos:

ADCtrackme

.-*float temp:* Almacena el resultado de la conversión del sensor de temperatura. El valor se guarda en ºC

.-*float Xout:* Contiene la conversión del eje X del acelerómetro.

.-*float Yout:* Contiene la conversión del eje Y del acelerómetro.

.-*float Zout:* Contiene la conversión del eje Z del acelerómetro.

.-*uint16_t ldr:* resultado de la conversión del sensor LDR de la placa.

Función: ADC_Init

-Firma de la función: void ADC_Init(void)

-Descripción: Inicializa el modulo ADC para la placa TrackMe, configura los pines que seran utilizados como canales analógicos y los deshabilita para los pines que son utilizados como digitales.

-Parámetros de entrada: Ninguno.

-Retorno: Ninguno.

-Ejemplo de aplicación.

```
ADC_Init();
```

Función: ADC_task

-Firma de la función: void ADC_task(void)

-Descripción: Tarea del conversor ADC, debe ejecutarse periódicamente para que se realicen las sucesivas conversiones AD de los canales analógicos.

-Parámetros de entrada: Ninguno.

-Retorno: Ninguno.

-Ejemplo de aplicación.

```
void SYSTEM_Tasks(void)
```

```
{
    //Tareas del sistema

//-----Tarea del ADC
    ADC_task();
//-----Fin Tarea ADC
}
```

Librería USART de la placa trackme.

En el siguiente apartado se detallan las funcionalidades implementadas en la librería USART de la placa de desarrollo TrackMe 3G.

Función: USART_Initialize

-Firma de la función: void USART_Initialize (void)

-Descripción: Inicializa el modulo USART2 para la placa TrackMe, configura los pines que serán utilizados para la comunicación con el modulo SIM. Además configura la USART1 como puerto de comunicación para mensajes de depuración de código.

-Parámetros de entrada: Ninguno.

-Retorno: Ninguno.

-Ejemplo de aplicación.

```
USART_Initialize();
```

Función: putcSIM_USART

-Firma de la función: void putcSIM_USART (char data)

-Descripción: Envía un carácter desde la USART2 al modulo SIM

-Parámetros de entrada:

data: carácter a enviar.

-Retorno: Ninguno.

-Ejemplo de aplicación.

```
char data;
putcSIM_USART(data);
```

Función: putsSIM_USART

-Firma de la función: void putsSIM_USART (char* data)

-Descripción: Envía una cadena de string desde la USART2 al modulo SIM

-Parámetros de entrada:

data: puntero de la cadena de string a enviar.

-Retorno: Ninguno.

-Ejemplo de aplicación.

```
char data[256];
putsSIM_USART(data);
```

Función: putrsSIM_USART

-Firma de la función: void putrsSIM_USART (const char* data)

-Descripción: Envía una cadena de string desde la USART2 al modulo SIM

-Parámetros de entrada:

data: puntero de la cadena constante de string a enviar.

-Retorno: Ninguno.

-Ejemplo de aplicación.

```
putrssSIM_USART("ATE0\r");
```

Función: ReadSIM_USART

-Firma de la función: char ReadSIM_USART (void)

-Descripción: Lee un carácter que llega desde la interfaz de comunicación con el modulo SIM.

-Parámetros de entrada:

data: puntero de la cadena constante de string a enviar.

-Retorno: Ninguno.

-Ejemplo de aplicación.

```
char c:
c = ReadSIM_USART();
```