**Otras formas de programación**

El módulo Black pill o Blue Pill puede programarse no solamente con Embitz; hay programas con los que también se puede programar, como Arduino y Embed, o el programa del fabricante STM32IDE, que cuenta con una librería HAL.

Programación con Arduino

Para la programación con Arduino se debe hacer una configuración previa. Una vez se haga, se puede acceder a los pines y diferentes periféricos haciendo algunos ajustes a los programas que están como ejemplo.

*Puertos GPIO.*

Acceder a los pines GPIO del Stm32f103x es definiendo los pines como, por ejemplo, para el puerto A, PA0, PA1, ... , PA15. Para el puerto B: PB0, PB1, ..., PB15. Por ejemplo, para el led que se ubica en el pin 12 del puerto B, del BlackPill, y usando el ejemplo de Blink, hay que definir el led como:

#define LED_BUILTIN PB12

Para la tarjeta azul, que tiene el led en el pin 13 de l puerto GPIOC,

#define LED_BUILTIN PC13

*Puerto Serial.*

El Stm32f103x tiene tres puertos seriales. En Arduino los puertos seriales están definidos así:

- Serial.begin(9600), inicializa el puerto serial Uart1 con los pines TX1= PA9 y RX1= PA10.
- Serial1.begin(9600), inicializa el puerto serial Uart2 con los pines TX2= PA2 y Rx2= PA3.
- Serial2.begin(9600), inicializa el puerto serial Uart3 con los pines TX3= PB10 y RX3= PB11.

Como ejemplo, el programa en Arduino DigitalReadSerial envía el estado de un pin al puerto serial. La modificación necesaria es determinar el pin:

int pushButton = PA1;

*Convertidor Analógico a Digital.*

El Stm32f103x tiene dos convertidores ADC y diez canales. Los canales, que se pueden acceder por ADC1 o ADC2, son PA0 a PA7 (ADC0 a ADC7), PB0 y PB1 (ADC8 y ADC9). Como ejemplo del uso del convertidor analógico a digital, el programa de Arduino AnalogReadSerial. Dentro del Loop, se debe modificar la línea de llamada de lectura ADC:

int sensorValue = analogRead(PA1);

*Timers.*

Para la medición de tiempo no hay acceso a los contadores como Tim1, 2, 3 ni 4, ni tampoco al RTC. Para esto es más conveniente buscar una librería que los soporte. Arduino tiene unas funciones para contar el tiempo como:

- micros(). Devuelve el número de microsegundos desde que el microcontrolador empieza a correr el programa. Este valor vuelve a cero en 70 minutos en los Arduinos. El valor de retorno puede ser múltiplo de 4 o de 8.
- millis(). Devuelve el número de milisegundos desde que el microcontrolador empieza a correr el programa. Este valor vuelve a cero en aproximadamente 50 días.

Hay funciones de retardo como delayMicroseconds(us) o delay(ms), que detienen la ejecución del programa por un tiempo determinado en microsegundos o en milisegundos, respectivamente.

*SPI.*

La comunicación serial síncrona con SPI se hace con la configuración de cada uno de los dos SPIs que tiene el microcontrolador STM32F103x. Se debe incluir el archivo:

```
#include <SPI.h>
```

Hay que crear una instancia para SPI2:

```
SPIClass SPI_2(2);
```

La inicialización de cada uno de los SPIs, con SPI1 como maestro y SPI2 como esclavo, como ejemplo:

```
// configuración SPI1
  SPI.begin(); //Initialize the SPI_1 port.
  SPI.setBitOrder(MSBFIRST); // Set the SPI_1 bit order
  SPI.setDataMode(SPI_MODE0); //Set the  SPI_2 data mode 0
  SPI.setClockDivider(SPI_CLOCK_DIV8);      // Slow speed (72 / 8 = 9 MHz SPI_1)

  // configuración SPI2
  SPI_2.beginSlave(); //Initialize the SPI_2 port.
  SPI_2.setBitOrder(MSBFIRST); // Set the SPI_2 bit order
  SPI_2.setDataMode(SPI_MODE0); //Set the  SPI_2 data mode 0
  SPI_2.setClockDivider(SPI_CLOCK_DIV4);  // (36/4= 9 MhzSPI_2)
```

Luego el llamado para escritura en el maestro y de lectura en el esclavo:

```
SPI.transfer(data);
data = SPI_2.read();
```

Embed.

Para programar el Stm32f103x, tarjeta azul o negra, se debe escoger la tarjeta que sea compatible con estas, o sea, que tenga el mismo microcontrolador. Embed tiene la tarjeta NUCLEO-F103RB.

*Puertos GPIO.*

Los nombres para los pines de la tarjeta están definidos como PA_0, PA_1,... , PA_15, para el puerto A, PB_0, PB_1,... , PB_15, para el puerto B y PC_0, PC_1, ..., PC_15, para el puerto C. Como ejemplo, para encender el led de la tarjeta negra:

```
#include "mbed.h"

DigitalOut  myled(PB_12);

int main()
```

```
{
    while (1) {
        myled = !myled;
        wait_ms(1000);
    }
}
```

La función `DigitalOut myled(PB_12)` configura la línea 12 del puerto B como salida y la llama `myled`.

*Puerto Serial.*

Para el puerto serial se usa la función `Serial pc(Tx, Rx)`. Aquí se escoge la línea que va a estar como transmisión, Tx, y luego la línea que va a estar funcionando como recepción, Rx, y el puerto se le nombra `pc`.

*SPI.*

Hay dos SPIs en el microcontrolador y el software los reconoce. Para inicializar el SPI1 como maestro, se usa, al comienzo, por fuera del main():

```
SPI Spi1(PA_7, PA_6, PA_5); // SPI_MOSI, SPI_MISO, SPI_SCK
```

Para SPI2 como esclavo:

```
SPISlave Spi2(PB_15, PB_14, PB_13, PB_12); // mosi, miso, sclk, ssel
```

La configuración de la frecuencia, el número de bits y el modo, que por lo general se usa el modo cero, tanto para el maestro como para el esclavo:

```
Spi1.frequency(9000000);
Spi1.format(8,0);
```

Si el maestro va a escribir un dato:

```
Spi1.write(dato);
```

Si se recibe un dato y se va a leer en el esclavo:

```
if(Spi2.receive())
{
   int dato = Spi2.read();
}
```