3. GPIO (General Purpose Input Output).

Como Gpio se conoce el grupo de líneas de entrada y salida digitales del microcontrolador. En el STM32F103C8T6 hay 37 pines Gpio.

- El puerto A tiene 16 pines, desde PA0 hasta PA15.
- El puerto B tiene 16 pines, desde PB0 hasta PB15.
- Puerto C tiene 3 pines, desde PC13 hasta PC15.
- El puerto D tiene apenas 2 pines, PD0 y PD1, que sirven como oscilador externo.

Sin embargo, en las tarjetas azul y negra los pines PA13 y PA14 del puerto A se usan como DIO y CLK, respectivamente, del conector JP2 para depuración y programación en la parte superior de la tarjeta. El pin PB2 se usa como BOOT1 en el conector de selección de modo, JP1. Los dos pines del puerto D están usados en ambas tarjetas para el oscilador de 8Mhz. En la tarjeta negra los pines PC14 y PC15 se usan para el oscilador de tiempo de 32.768 khz y solo está disponible el PC13 del puerto C.

Para trabajar con los pines digitales de entrada o de salida es necesario hacer la misma configuración que se mostró en el capítulo anterior sobre descripción de programación en lenguaje ensamblador:

- Habilitar el reloj del puerto.
- Configurar el modo del puerto.
- Hacer lectura o escritura.

Configuración del reloj del puerto:

Todos los periféricos requieren de un reloj para poderlos habilitar. Los puertos Gpio tienen el registro RCC_APB2ENR para habilitar o deshabilitar el reloj. Un ejemplo para habilitar el reloj del puerto A es con la función:

```
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
```

Esta función se encuentra en el archivo “stm32f10x_rcc.h” de la librería SPL. Para el puerto se pueden usar los siguientes parámetros:

`RCC_APB2Periph_GPIOA`, para el puerto A
`RCC_APB2Periph_GPIOB`, para el puerto B
`RCC_APB2Periph_GPIOC`, para el puerto C
`RCC_APB2Periph_GPIOD`, para el puerto D
Y el estado de habilitación `ENABLE` o deshabilitación, `DISABLE`.

Configuración del modo del puerto.

El modo del puerto, como ejemplo para el puerto A, se configura con la función que se encuentra en el archivo “stm32f10x_gpio.h”:

```
GPIO_Init(GPIOA, &GPIOA_Struct);
```

Con esta función se configura el pin del puerto, la velocidad de muestreo y el modo del pin. El puerto puede ser `GPIOx`, donde x puede ser A, B, C o D. Los detalles del pin para su

configuración deben estar en una estructura que se define de la siguiente manera, como ejemplo, para el puerto A, pin 12, velocidad de 2 Mhz y modo de salida:

```
GPIO_InitTypeDef GPIOA_Struct;
GPIOA_Struct.GPIO_Pin  = GPIO_Pin_12;
GPIOA_Struct.GPIO_Speed = GPIO_Speed_2MHz;
GPIOA_Struct.GPIO_Mode = GPIO_Mode_Out_PP;
```

El pin del puerto puede ser `GPIO_Pin_0` al `GPIO_Pin_15` para los puertos A y B, `GPIO_Pin_13` al `GPIO_Pin_15` para el puerto C y `GPIO_Pin_0` y `GPIO_Pin_1` para el puerto D. También se pueden escoger todos los pines del puerto con `GPIO_Pin_All`, o se pueden escoger más de un pin usando una lógica Or, como por ejemplo, `GPIOB_Struct.GPIO_Pin  = GPIO_Pin_12 | GPIO_Pin_13`.

La velocidad de muestreo del puerto tiene las opciones `GPIO_Speed_2MHz`, `GPIO_Speed_10MHz`, y `GPIO_Speed_50MHz`. En cuanto a los modos de los pines del puerto, se pueden escoger:
`GPIO_Mode_AF_OD`, función alterna en Open- Drain.
`GPIO_Mode_AF_PP`, función alterna en Push- Pull.
`GPIO_Mode_AIN`, señal analógica de entrada.
`GPIO_Mode_IN_FLOATING`, entrada flotante.
`GPIO_Mode_IPD`, entrada en Pull- Down.
`GPIO_Mode_IPU`, entrada Pull- Up.
`GPIO_Mode_Out_OD`, salida en Open- Drain.
`GPIO_Mode_Out_PP`, salida en Push- Pull.

La ejecución de la función de configuración de los pines no borra la anterior, así que se pueden hacer configuraciones diferentes, por ejemplo, para pines de entrada y luego para salida. Las diferentes configuraciones de conexión al pin de los puertos Gpio se describen a continuación.

Configuración en salida Open- Drain: Esta configuración es de salida. Tiene la posibilidad de alimentar una carga con un voltaje diferente al voltaje de alimentación del microcontrolador. Ver la figura 32.

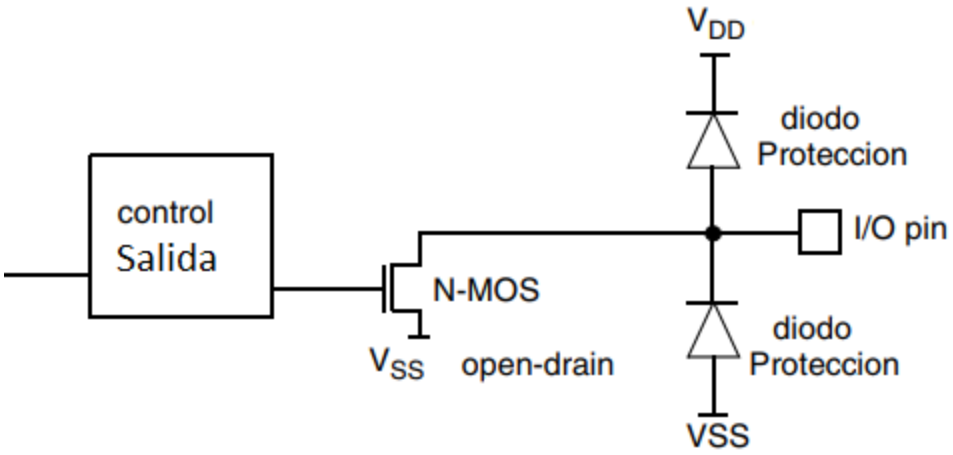

Figura 32. Configuración de salida Open Drain, o drenaje abierto, llamado así por el pin de drenaje del mosfet.

Configuración en salida Push- Pull. En esta configuración se consigue una salida de corriente como para encender un led conectado con una resistencia en serie hacia VDD o hacia VSS. Ver figura 33.

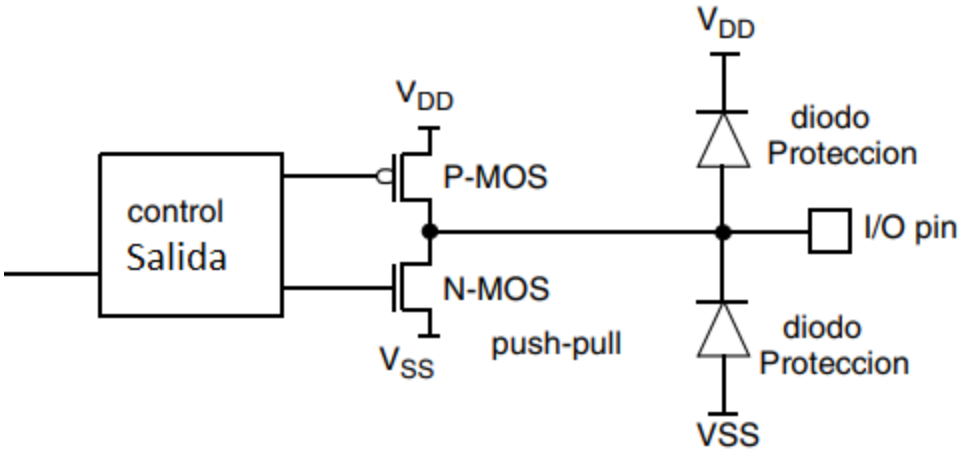

Figura 33. Configuración de salida en Push- Pull, llamado así por la disposición de los transistores.

Configuración en entrada flotante. Como se aprecia en la figura 34, ninguna de las resistencias están conectadas, por lo que la entrada queda "flotando". Esta configuración se usa para entradas de funciones alternas, como una entrada al módulo ADC.

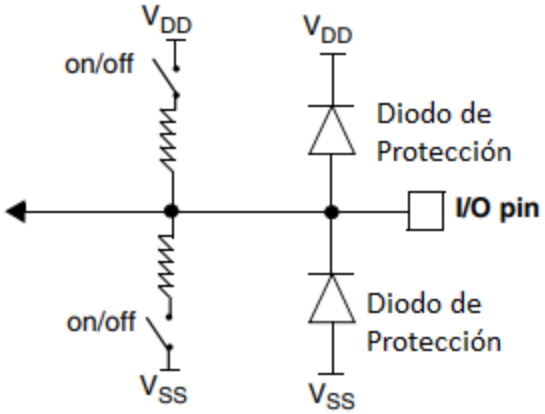

Figura 34. Configuración de pin en entrada flotante.

Configuración entrada en Pull- Down. Esta configuración de entrada tiene conectada la resistencia que va a tierra. Se usa para leer siempre un "cero" lógico, por ejemplo, si se conecta un conmutador a VDD. En caso que el conmutador se cierre, la entrada lógica ve VDD. Ver figura 35.

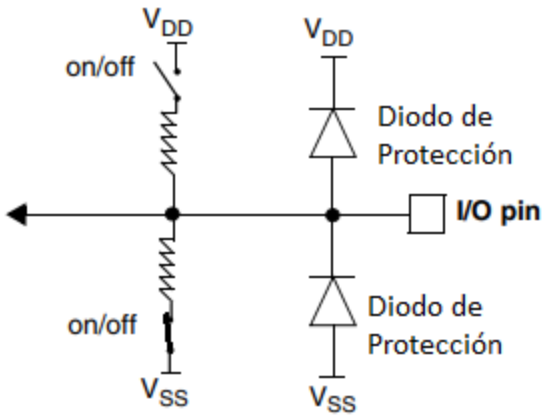

Figura 35. Configuración de entrada en Pull- Down o con resistencia a tierra.

Configuración entrada en Pull- Up. En esta configuración de entrada la resistencia superior va conectada a VDD y la entrada va a leer siempre VDD. Si se conecta un conmutador entre el pin y VSS, la lectura va a ver VDD si el conmutador está abierto, o VSS si el conmutador está cerrado.

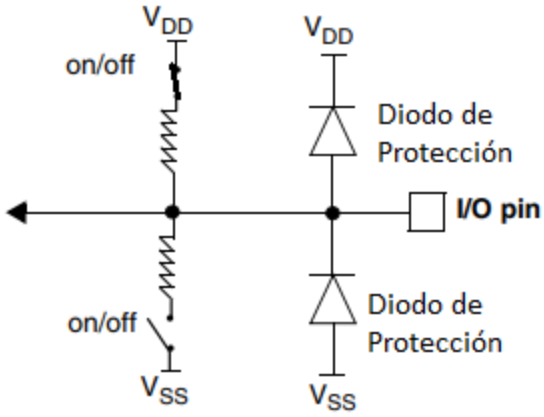

Figura 36. Configuración de entrada en Pull- Up o resistencia en Vcc.

Lectura o escritura de un pin de un puerto Gpio:
Dentro del mismo archivo “stm32f10x_gpio.h” se encuentran las diferentes funciones para leer o escribir en un pin de un puerto.

Funciones para leer un puerto Gpio. Estas funciones devuelven un valor entero sin signo de 8 bits para lectura de un solo pin, o de 16 bits para la lectura del puerto. `GPIOx` puede ser `GPIOA`, `GPIOB`, `GPIOC`, o `GPIOD`, que define el puerto, mientras que `GPIO_Pin` puede ser `GPIO_Pin_0` al `GPIO_Pin_15`. Se pueden hacer lecturas al puerto definido como entrada, para las dos primeras funciones, o lecturas a un puerto definido como salida, para las dos últimas funciones a continuación.

```
uint8_t GPIO_ReadInputDataBit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
uint16_t GPIO_ReadInputData(GPIO_TypeDef* GPIOx);
uint8_t GPIO_ReadOutputDataBit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
```

```
uint16_t GPIO_ReadOutputData(GPIO_TypeDef* GPIOx);
```

Funciones para escribir a un puerto Gpio. Estas funciones de escritura sobre un puerto pueden poner bits en uno lógico (`GPIO_SetBits()`), en cero lógico (`GPIO_ResetBits()`), o escribir cualquier valor lógico en un bit específico del puerto (`GPIO_WriteBit()`), o cualquier valor en todo el puerto (`GPIO_Write()`).

```
void GPIO_SetBits(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
void GPIO_ResetBits(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
void GPIO_WriteBit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, BitAction BitVal);
void GPIO_Write(GPIO_TypeDef* GPIOx, uint16_t PortVal);
```

A forma de integración lo que se ha visto como uso de un bit de un puerto, el siguiente ejemplo cambia el estado del bit 12 del puerto B por un tiempo, donde está conectado un led en la tarjeta negra, como se muestra en la figura 37.

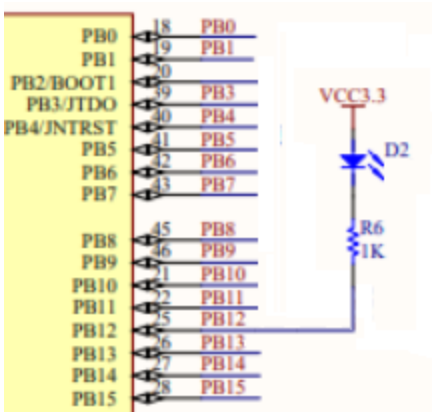

Figura 37. Led de la tarjeta negra conectado al microcontrolador.

Para encender el led D2 se debe poner un cero lógico, como se ve en la figura 37 donde el ánodo del led está conectado a Vcc.

```
#include "stm32f10x_conf.h"

void LED_Init(void);

int main(void)
{
  LED_Init();
  while(1)
  {
    if(GPIO_ReadOutputDataBit(GPIOB, GPIO_Pin_12) == Bit_SET)
        GPIO_ResetBits(GPIOB, GPIO_Pin_12);
    else
        GPIO_SetBits(GPIOB, GPIO_Pin_12);
    for (int i = 0; i < 2000000; ++i) asm("nop");
  }
}
```

```
/*******************************************
 * Inicialización del puerto B, pin 12
 *******************************************/
void LED_Init(void)
{
    GPIO_InitTypeDef GPIOB_Struct;
    GPIOB_Struct.GPIO_Pin  = GPIO_Pin_12;
    GPIOB_Struct.GPIO_Speed = GPIO_Speed_2MHz;
    GPIOB_Struct.GPIO_Mode = GPIO_Mode_Out_PP;

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
    GPIO_Init(GPIOB, &GPIOB_Struct);
}
```

La subrutina `LED_Init()` configura el pin 12 del puerto B como salida para que pueda usarse para encender y apagar el led de la tarjeta. Dentro del while se lee el estado del bit 12 y si está en uno lógico, se pone en cero, de otra forma se pone en uno lógico.
El archivo stm32f10x_gpio de la librería SPL contiene las siguientes funciones.

```
1.  void GPIO_DeInit(GPIO_TypeDef* GPIOx);
2.  void GPIO_AFIODeInit(void);
3.  void GPIO_Init(GPIO_TypeDef* GPIOx, GPIO_InitTypeDef* GPIO_InitStruct);
4.  void GPIO_StructInit(GPIO_InitTypeDef* GPIO_InitStruct);
5.  uint8_t GPIO_ReadInputDataBit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
6.  uint16_t GPIO_ReadInputData(GPIO_TypeDef* GPIOx);
7.  uint8_t GPIO_ReadOutputDataBit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
8.  uint16_t GPIO_ReadOutputData(GPIO_TypeDef* GPIOx);
9.  void GPIO_SetBits(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
10. void GPIO_ResetBits(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
11. void GPIO_WriteBit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, BitAction BitVal);
12. void GPIO_Write(GPIO_TypeDef* GPIOx, uint16_t PortVal);
13. void GPIO_PinLockConfig(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
14. void GPIO_EventOutputConfig(uint8_t GPIO_PortSource, uint8_t GPIO_PinSource);
15. void GPIO_EventOutputCmd(FunctionalState NewState);
16. void GPIO_PinRemapConfig(uint32_t GPIO_Remap, FunctionalState NewState);
17. void GPIO_EXTILineConfig(uint8_t GPIO_PortSource, uint8_t GPIO_PinSource);
18. void GPIO_ETH_MediaInterfaceConfig(uint32_t GPIO_ETH_MediaInterface);
```

Las primeras dos funciones son para desconfigurar una configuración previa al puerto, la número 4 es para inicializar la estructura con información del puerto, la número 13 es para desbloquear algunos pines especiales que cumplen doble propósito, la 14 y 15 son para programar eventos en los pines, los cuales son funciones alternas, la 16 es para configurar el pin con funciones alternas, la 17 es para configurar un pin del puerto como fuente de interrupción externa y la función en 18 es para configuración de internet, pero el microcontrolador STM32F103C8T6 no lo tiene disponible. Las demás funciones se mostraron como funcionan en este capítulo.