



Instituto Tecnológico Superior De Atlixco.

Ingeniería Mecatrónica.

IM140744.

7-A.

Microcontroladores.

PRÁCTICA 1 Control de puertos de entrada y salida
del microcontrolador STM32F407VGT6.

CATEDRÁTICO: Dra. Mariana Natalia Ibarra
Bonilla.

INTEGRANTE: José Ángel Balbuena

Palma. IM140744

Fecha de realización:

17/09/2017

Fecha de entrega:

18/09/2017

Objetivo general.

El objetivo de estas prácticas es una Introducción al manejo y entorno de softwares para la programación y configuración de puertos, así como la carga de los programas a la tarjeta de desarrollo Ophyra que contiene u microcontrolador STM32F407VGT6 ,el cual es de 32 bits así mismo es de la familia Cortex-M4 .

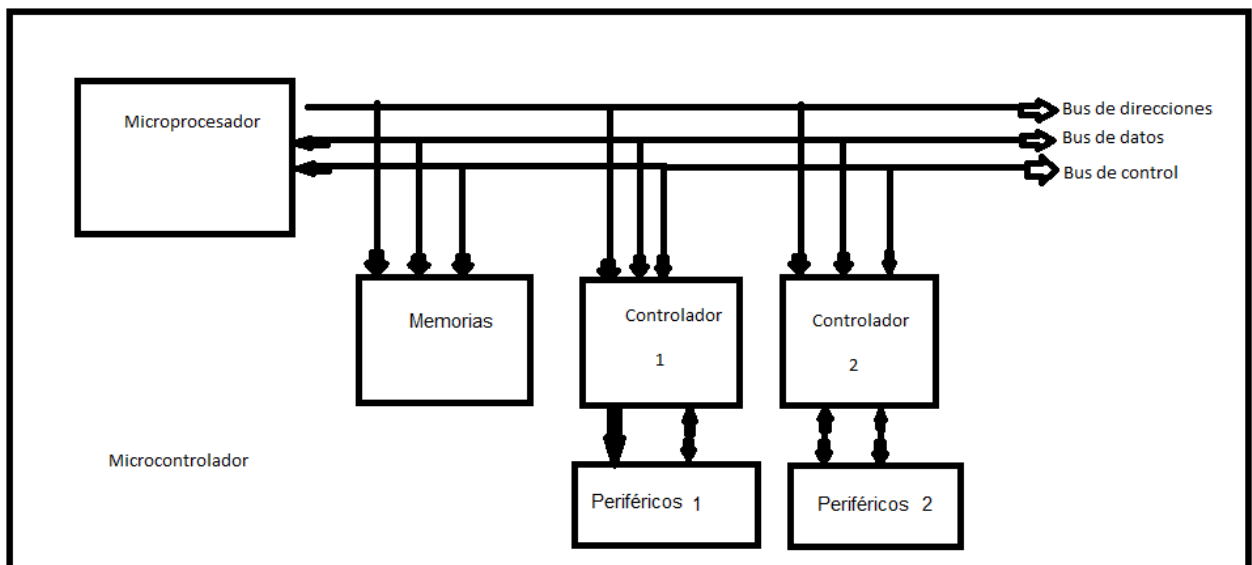
Materiales.

- Protoboard.
- Módulos de LEDS.
- Tarjeta de desarrollo Ophyra.
- Jumpers.
- Computadora con el software Atollic TrueStudio, STM32CubeMX y STMFlashLoader .

Marco Teórico.

Microcontrolador.

Es un computador completo (unidad central de procesamiento, memoria y periféricos de entrada y salida) con delimitadas prestaciones, es un circuito integrado que se destina a gobernar una determinada tarea.



Tamaño de palabra.

Se hace referencia el número de bits con el que el CPU trabaja, pueden ser de 8, 16 o 32 bits en este caso el microcontrolador STM32F407VGT6 trabaja a 32 bits lo que nos da una idea de que este micro controlador tiene un capacidad de procesamiento alta y rápida.

Ejecución de instrucciones.

Este microcontrolador realiza su ejecución de instrucciones de arquitectura RISC (Computador con conjunto de instrucciones reducidas). El conjunto de instrucciones que el procesador es capaz de ejecutar es pequeño, simple y rápido.

Microcontroladores de la gama CORTEX-M.

Son microcontroladores de alta aplicación los cuales son empleados de manera amplia en el sector industrial, pues proporciona un alto nivel de eficiencia energética, estos también incluyen características especiales para la industria como protocolos Can, USB, entre otros más. Este también cuenta con MCU de 32 bits más reducida.

Microcontrolador CORTEX-M4 STM32F407VGT6.

Este es un micro controlador el cual trabaja a una frecuencia de 168Mhz, con un empaquetado de 100 pines, una unidad de punto flotante ,un convertidor AD (3 a 12 bits) y DA (2 a 12 bits) , dos temporizador de 32 bit, interfaz Ethernet, PWM de 16 bits , protocolos de comunicación CAN, entre otras más características.

Entradas y salidas digital.

Es un mecanismo básico de cualquier controlador que permite escribir y leer en dos estados posibles alto o bajo.

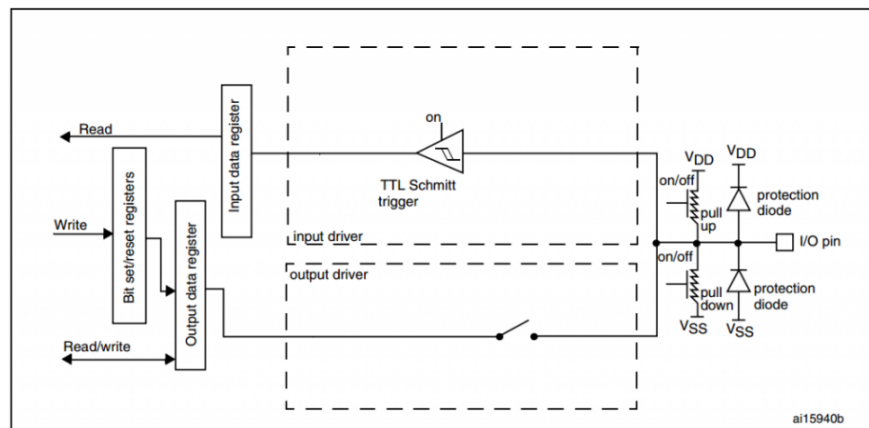
Puertos.

Al subsistema encargado de la escritura y lectura de señales se le denomina GPIO (General Purpose Input/Output) los puertos de micro controlador se agrupan en 16 líneas o pines, estos se enumeran con letras A,B,C,D... y la línea con el numero entre 0 y el número de línea bits de puerto. Por ejemplo PD0 puerto D, línea 0.

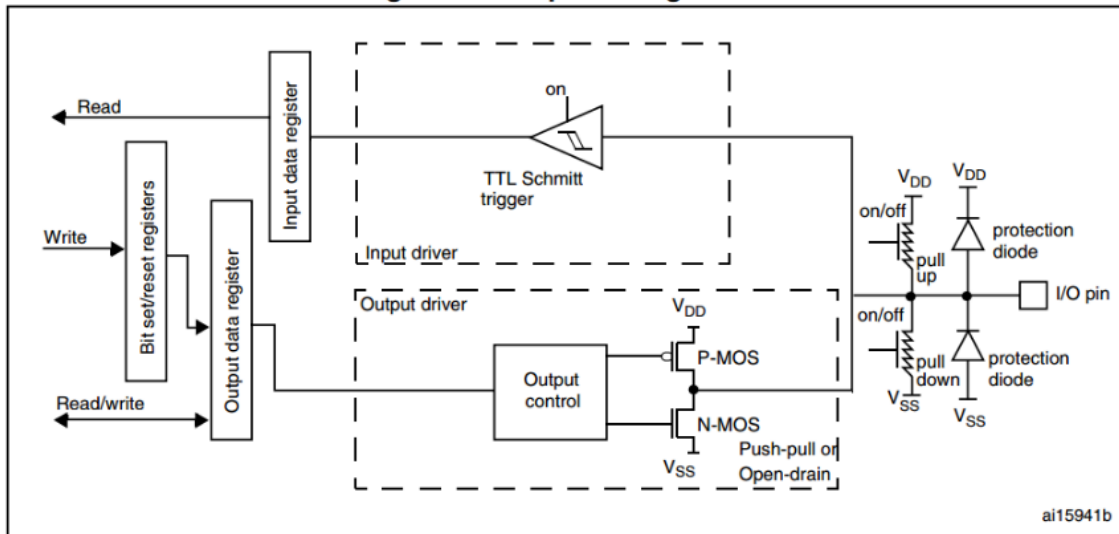
Configuración GPIO.

Configuración del GPIO como entrada.

Se activa la entrada Schmitt Trigger, las resistencias pull down y pull up pueden ser habilitadas, los estados de entrada son flotante,pull up , pull down y analógica.



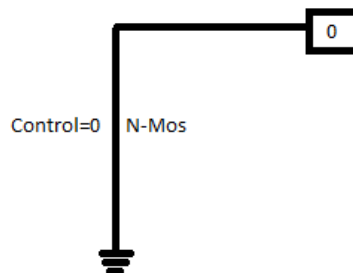
Configuración de GPIO como salida digital.



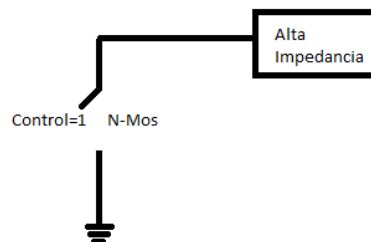
El buffer de salida es habilitado en push pull u open Drain, se activa la entrada Schmitt Trigger, las resistencias pull up y pull down están listas para hacer activadas.

Configuración Open Drain.

Un cero en el registro de salida activara el N-Mos , por lo que en el pin tendremos un cero lógico al inicio.



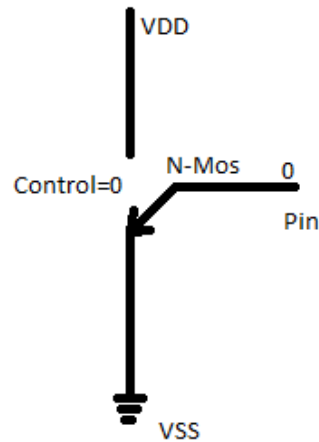
Un uno en el registro de salida deja el pin en alta impedancia (P-Mos nunca es empleado).



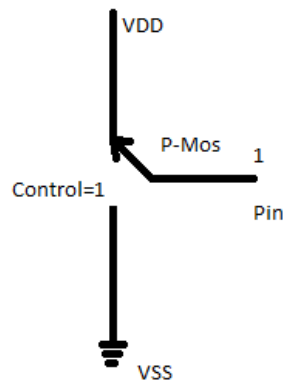
Configuración Push-Pull.

Es el modo recomendado para trabajar inicialmente el estado del pin.

Un 0 en el registro de salida activa el N-Mos, por lo que tendremos un cero lógico inicialmente en el pin.

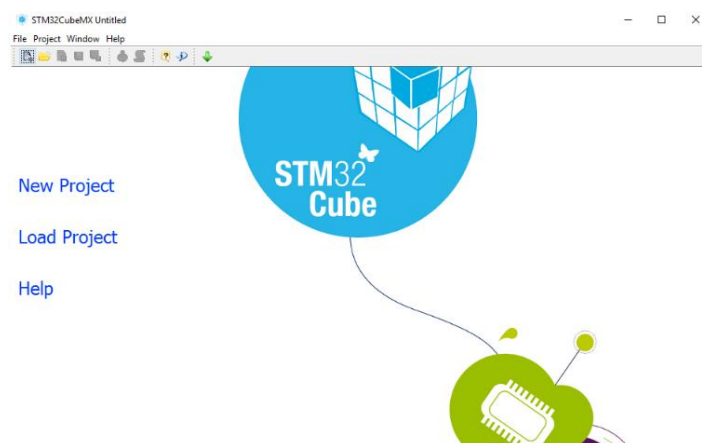


Un 1 en el registro se activa el P-Mos, por lo que tendremos uno lógico inicialmente en el pin.



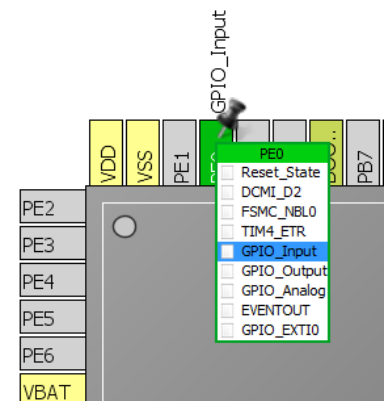
Software STM32CubeMX.

Este es el software que se emplea para las configuraciones de los pines o puertos de nuestro microcontrolador. Aquí se puede seleccionar los pines a utilizar, configurar si serán entradas o salidas y el tipo de estas.



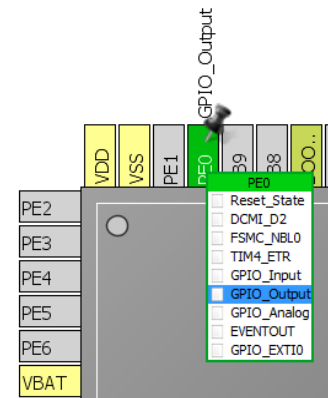
Para buscar nuestro microcontrolador nos situamos en la barra de búsqueda con la cual podremos el nombre y serie del microcontrolador para después seleccionarlo.

Selección de pines así como configuración de estos como entrada y salida.



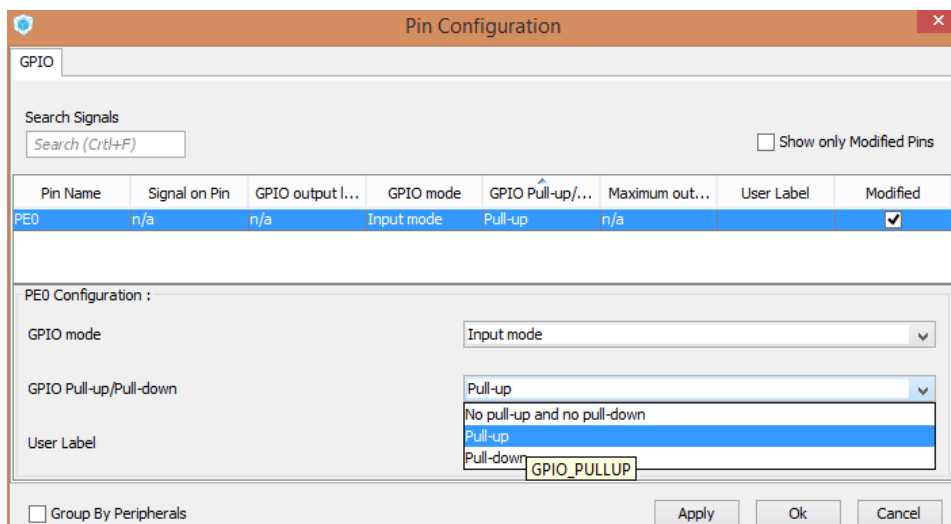
Configuración del pin como salida.

Seleccionamos el pin deseado como salida, se debe configurar como GPIO_Output.

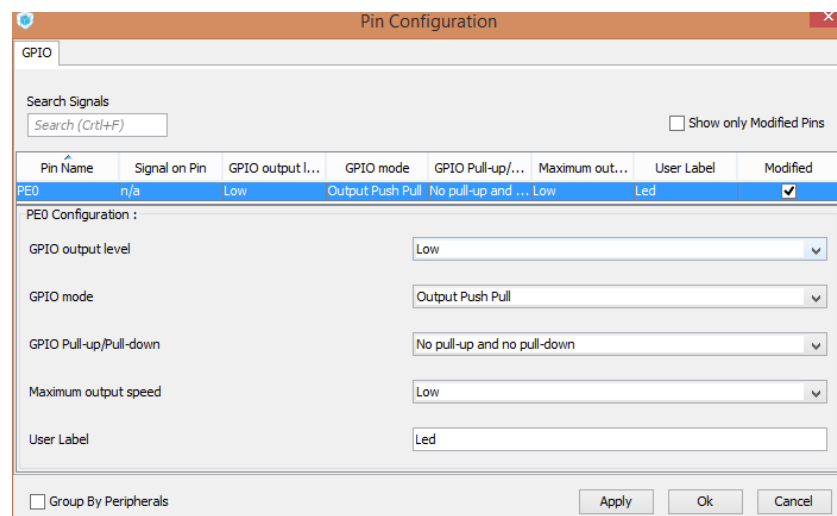


Configuración GPIO.

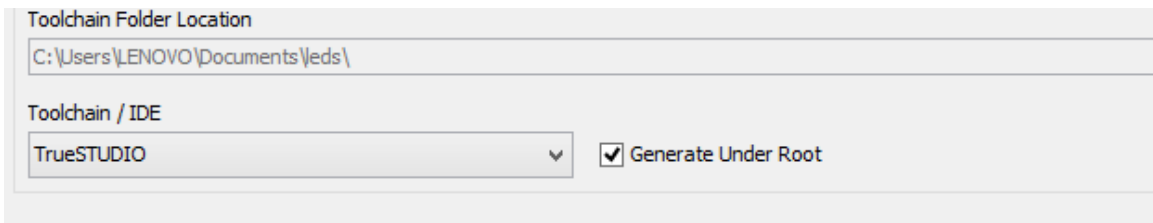
El pin como entrada se puede configurar como Pull-up o Pull-down dependiendo del arreglo del botón Pull-up o Pull-down.



Para configurar pin como salida se debe seleccionar la opción de Push-Pull pues es la recomendada, podemos establecer un nivel lógico inicial al pin dependiendo de la aplicación deseada.



Se debe seccionar el entorno de desarrollo para la programación del microcontrolador puesto que STM32CubeMX nos genera un código con las configuraciones previamente realizadas de entradas y salidas.



Toolchain Folder Location
C:\Users\LENOVO\Documents\eds\
Toolchain / IDE
TrueSTUDIO
☒ Generate Under Root

Atollic TrueStudio.

Es el IDE designado para la programación del microcontrolador en el cual podemos generar el código requerido para la aplicación deseada.

Partes principales de un programa en Atollic TrueStudio.

```
36 *****  
37 */  
38 /* Includes -----  
39 #include "main.h"  
40 #include "stm32f4xx_hal.h"  
41  
42 /* USER CODE BEGIN Includes */
```



Inclusión de librerías del microcontrolador para operar.

```
53 /* Private function prototypes -----*/  
54 void SystemClock_Config(void);  
55 static void MX_GPIO_Init(void);
```



Prototipos de funciones para la configuración del reloj a utilizar y de la función con las configuraciones de entradas y salidas realizadas con anterioridad.

```
66 int main(void)
```



Función principal

```
67 {
```

```
75 /* Reset of all peripherals, Initializes the
```

```
76 HAL_Init();
```



Función resetea e inicializa la interfaz Flash

```
77 /* USER CODE BEGIN Init */
```

```
78 /* USER CODE END Init */
```

```
79 /* Configure the system clock */
```

```
80 SystemClock_Config();
```



Inicia las configuraciones del reloj

```
81 /* USER CODE BEGIN SysInit */
```

```
82 /* USER CODE END SysInit */
```

```
83 /* Initialize all configured peripherals */
```

```
84 MX_GPIO_Init();
```



Configura los pines de entrada-salida

```
85 /* USER CODE BEGIN 2 */
```

```
86 /* USER CODE END 2 */
```

```
87 /* Infinite loop */
```

```
88 /* USER CODE BEGIN WHILE */
```

```
89 while (1)
```



Ciclo infinito donde crearemos el código a emplear

```
90 {
```

```
91 /* USER CODE END WHILE */
```

```
92
```

```
93 /* USER CODE BEGIN 3 */
```

```
94
```

```
95 /* USER CODE END 3 */
```

```
96
```

```
97 }
```

```
98 }
```


Funciones propias de Atollic TrueStudio.

HAL_GPIO_ReadPin(); Esta función permite leer el estado lógico con el que cuenta un pin. Esta función requiere como parámetros: el puerto a emplear (GPIO (letra del puerto)) y el pin a leer (etiqueta del pin_Pin) .

Ejemplo: HAL_GPIO_ReadPin(GPIOC, Boton_Pin);

HAL_GPIO_WritePin(); Esta función permite escribir un estado lógico en un pin. Esta función requiere como parámetros: el puerto a emplear (GPIO (letra del puerto)), el pin a escribir (etiqueta del pin_Pin) y estado a escribir en el pin (GPIO_PIN_Estado lógico).

GPIO_PIN_SET pone un uno lógico en el pin.

GPIO_PIN_RESET pone un cero lógico en el pin.

Ejemplo: HAL_GPIO_WritePin(GPIOC, Led1_Pin,GPIO_PIN_SET);

HAL_GPIO_TooglePin(); Invierte el estado lógico presente en el pin. Esta función requiere como parámetros: el puerto a emplear (GPIO (letra del puerto)) y el pin a escribir (etiqueta del pin_Pin).

HAL_Delay(); Permite crear retardos en milisegundos. Esta función requiere como parámetro el número de milisegundos del retardo.

Estructuras de control.

If-else.

Esta estructura permite evaluar una condición y se está resulta ser verdadera ejecutara un bloque de instrucciones, else es el caso de no cumplir esta condición ejecutara otro bloque de instrucciones.

If (condición a evaluar)

{Bloque de Instrucciones si se cumple la condición}

else

{Bloque de instrucciones si no se cumple la condición }

Switch-case.

Eta estructura nos permite ejecutar diferentes bloques de instrucciones para diferentes valores que puede presentar una variable.

switch(expresión) { se coloca la variable a comparar

case valor1: Bloque de instrucciones 1;

break;

case valor2: Bloque de instrucciones 2;

break;

case valor3: Bloque de instrucciones 3;

```
break;
```

default: Bloque de instrucciones por defecto; este bloque se ejecuta si ningún otro caso fue empleado

```
}
```

For.

Es una estructura de control iterativa. Este indica el mínimo de iteraciones que realizarán las instrucciones.

```
for(iniciación; condición de finalización; incremento)
```

```
{
```

```
    Bloque de instrucciones;
```

```
}
```

Ciclo while.

El bloque de instrucciones se ejecutan mientras una condición permanezca verdadera sino se rompe el ciclo.

```
while(Condición a evaluar)
```

```
{
```

```
    Bloque de instrucciones;
```

```
}
```

Ciclo do- while.

El bloque de instrucciones se ejecutan mientras una condición permanezca verdadera sino se rompe el ciclo. Este bloque de instrucciones se ejecutan al menos una vez.

```
do
```

```
{
```

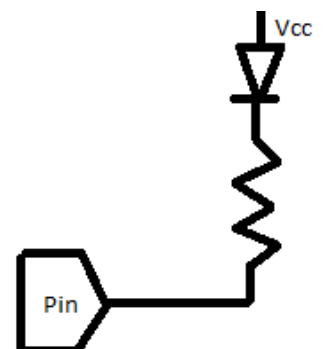
```
    Bloque de instrucciones;
```

```
} while(Condición a evaluar)
```

Características propias de la tarjeta de desarrollo Ophyra.

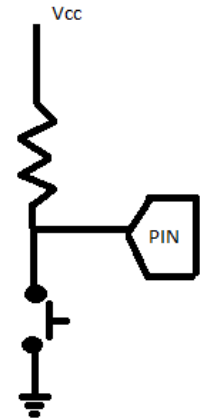
Led RGB.

El led de color rojo se encuentra en el pin PE0, el led de color verde se encuentra conectado en PE1 y el led de color azul en el pin PE2. En la configuración que se muestra en la imagen.



Push Botón.

La placa de desarrollo cuenta con 4 Push botón los cuales están conectados a PC2, PD5, PD4 y PD3. Conectados En configuración Pull Up.



Desarrollo.

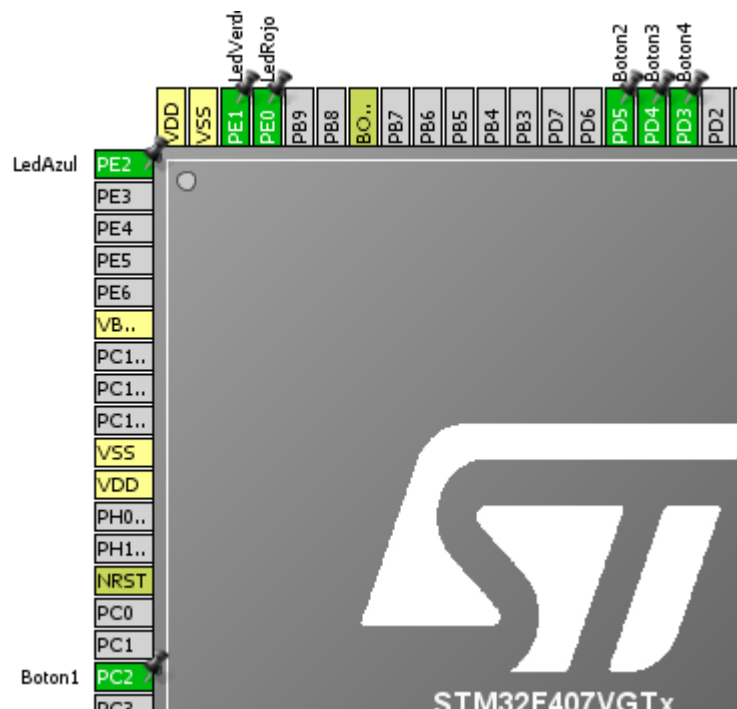
Practica 1.- Encendido y apagado de led RGB usando 4 interruptores.

Descripción: Se deben usar 4 push botón de la placa de desarrollo Ophyra cada uno de ellos al ser pulsado debe encender un led el rojo, el verde, el azul o una combinación de estos, al dejar de ser pulsados los interruptores se deberán apagar los leds, al inicio deben estar apagados los leds.

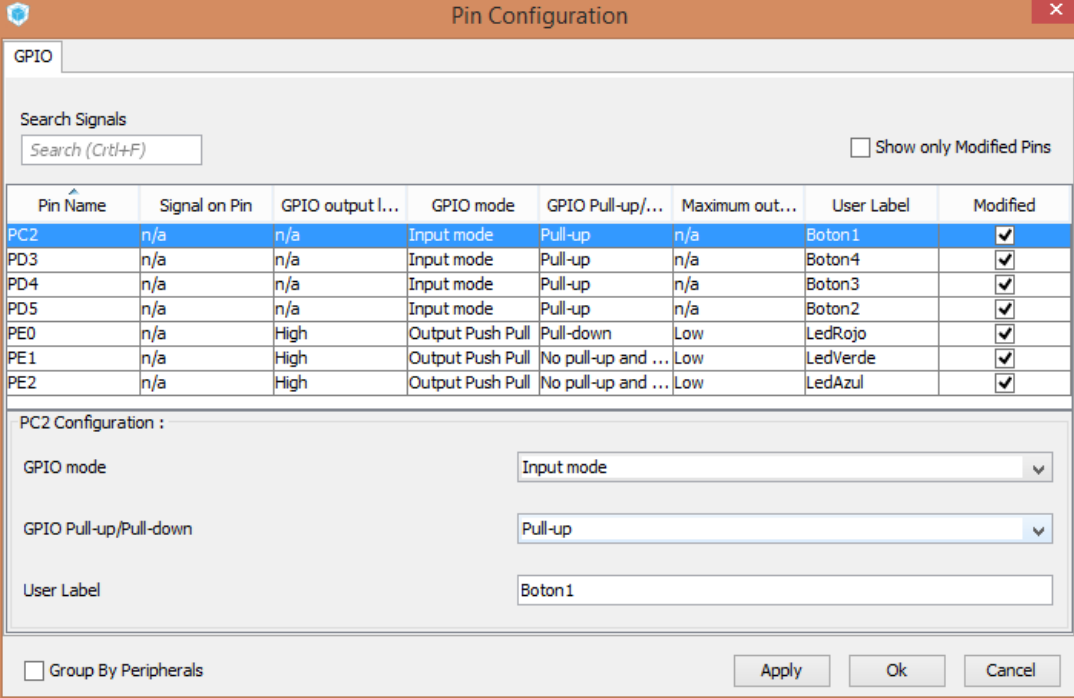
Configuración de pines en STM32CubeMX.

Los botones se configuran como entradas (GPIO_Input), el Boton1 se configura al pin PC2, el Boton2 se configura al pin PD5, el Boton3 se configura al pin PD4 y el Boton4 se configura al pin PD3.

Los Leds se configuran como salidas (GPIO_Output), el LedRojo se configura al pin PE0, el LedVerde se configura al pin PE1 y el LedAzul se configura al pin PE2.



Para los botones debemos seleccionar GPIO Pull-up por la configuración en que están conectados estos en la placa que es Pull-Up todos los botones se deben seleccionar esta forma.



Pin Configuration

GPIO

Search Signals
Search (Ctrl+F)

☐ Show only Modified Pins

Pin Name	Signal on Pin	GPIO output l...	GPIO mode	GPIO Pull-up/...	Maximum out...	User Label	Modified
PC2	n/a	n/a	Input mode	Pull-up	n/a	Boton 1	<input checked="" type="checkbox"/>
PD3	n/a	n/a	Input mode	Pull-up	n/a	Boton4	<input checked="" type="checkbox"/>
PD4	n/a	n/a	Input mode	Pull-up	n/a	Boton3	<input checked="" type="checkbox"/>
PD5	n/a	n/a	Input mode	Pull-up	n/a	Boton2	<input checked="" type="checkbox"/>
PE0	n/a	High	Output Push Pull	Pull-down	Low	LedRojo	<input checked="" type="checkbox"/>
PE1	n/a	High	Output Push Pull	No pull-up and ...	Low	LedVerde	<input checked="" type="checkbox"/>
PE2	n/a	High	Output Push Pull	No pull-up and ...	Low	LedAzul	<input checked="" type="checkbox"/>

PC2 Configuration :

GPIO mode: Input mode

GPIO Pull-up/Pull-down: Pull-up

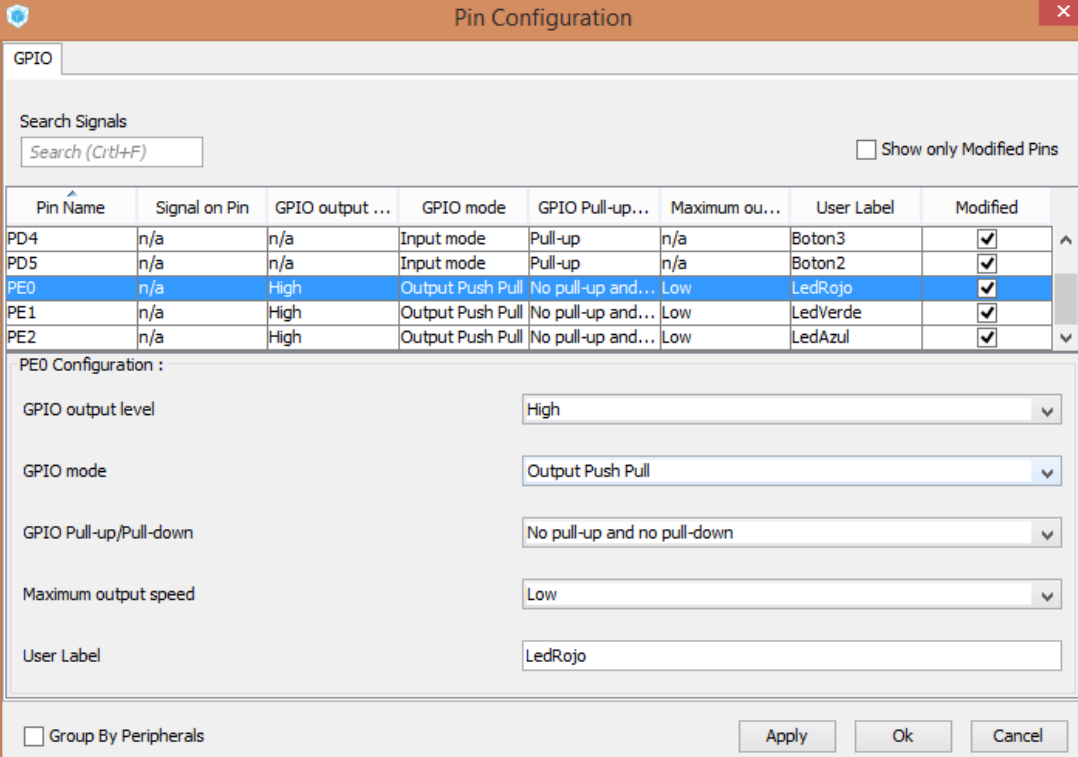
User Label: Boton 1

☐ Group By Peripherals

Apply Ok Cancel

En
En

cuanto a los Leds se deben configurar con un nivel alto al inicio para que los leds comiencen apagados y se debe seleccionar la característica Output Push Pull debido a que es la recomendada. Las resistencias Pull Up y Pull Down deben permanece desactivadas ya que no las emplearemos.



Pin Configuration

GPIO

Search Signals
Search (Ctrl+F)

☐ Show only Modified Pins

Pin Name	Signal on Pin	GPIO output ...	GPIO mode	GPIO Pull-up...	Maximum ou...	User Label	Modified
PD4	n/a	n/a	Input mode	Pull-up	n/a	Boton3	<input checked="" type="checkbox"/>
PD5	n/a	n/a	Input mode	Pull-up	n/a	Boton2	<input checked="" type="checkbox"/>
PE0	n/a	High	Output Push Pull	No pull-up and...	Low	LedRojo	<input checked="" type="checkbox"/>
PE1	n/a	High	Output Push Pull	No pull-up and...	Low	LedVerde	<input checked="" type="checkbox"/>
PE2	n/a	High	Output Push Pull	No pull-up and...	Low	LedAzul	<input checked="" type="checkbox"/>

PE0 Configuration :

GPIO output level: High

GPIO mode: Output Push Pull

GPIO Pull-up/Pull-down: No pull-up and no pull-down

Maximum output speed: Low

User Label: LedRojo

☐ Group By Peripherals

Apply Ok Cancel

Programación de la aplicación Atollic TrueStudio.

Una vez realizadas las configuraciones de los pines en STM32CubeMX nos generara un código con un código podemos empezar a programar Atollic TrueStudio.

Código empleado.

```
98 while (1)
99 {
100 /* USER CODE END WHILE */
101 /* USER CODE BEGIN 3 */
102 HAL_GPIO_WritePin(GPIOE, LedRojo_Pin, GPIO_PIN_SET);
103 HAL_GPIO_WritePin(GPIOE, LedAzul_Pin, GPIO_PIN_SET);
104 HAL_GPIO_WritePin(GPIOE, LedVerde_Pin, GPIO_PIN_SET);
105 if((HAL_GPIO_ReadPin(GPIOC, Boton1_Pin))==0){
106     HAL_GPIO_WritePin(GPIOE, LedRojo_Pin, GPIO_PIN_RESET);
107 }
108 if((HAL_GPIO_ReadPin(GPIOD, Boton2_Pin))==0){
109     HAL_GPIO_WritePin(GPIOE, LedAzul_Pin, GPIO_PIN_RESET);
110 }
111 if((HAL_GPIO_ReadPin(GPIOD, Boton3_Pin))==0){
112     HAL_GPIO_WritePin(GPIOE, LedVerde_Pin, GPIO_PIN_RESET);
113 }
114 if((HAL_GPIO_ReadPin(GPIOD, Boton4_Pin))==0){
115     HAL_GPIO_WritePin(GPIOE, LedRojo_Pin, GPIO_PIN_RESET);
116     HAL_GPIO_WritePin(GPIOE, LedVerde_Pin, GPIO_PIN_RESET);
117 }
118 }
119 }
```

CICLO INFINITO → `while (1)`

Inicialmente apagamos los tres leds → `HAL_GPIO_WritePin(GPIOE, LedRojo_Pin, GPIO_PIN_SET);`
`HAL_GPIO_WritePin(GPIOE, LedAzul_Pin, GPIO_PIN_SET);`
`HAL_GPIO_WritePin(GPIOE, LedVerde_Pin, GPIO_PIN_SET);`

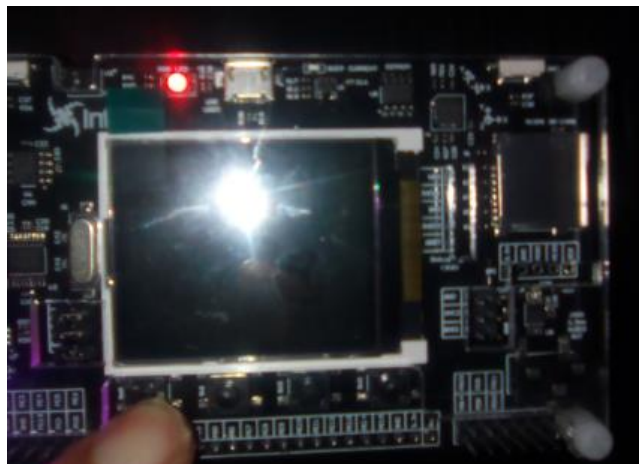
Preguntamos si el estado del Boton1 es cero si es así el botón fue pulsado Prendemos el LedRojo mandandole un cero. → `if((HAL_GPIO_ReadPin(GPIOC, Boton1_Pin))==0){`
`HAL_GPIO_WritePin(GPIOE, LedRojo_Pin, GPIO_PIN_RESET);`

Preguntamos si Boton2 fue pulsado si es así encendemos el LedAzul. → `if((HAL_GPIO_ReadPin(GPIOD, Boton2_Pin))==0){`
`HAL_GPIO_WritePin(GPIOE, LedAzul_Pin, GPIO_PIN_RESET);`

Preguntamos si Boton3 fue pulsado si es así encendemos el LedVerde. → `if((HAL_GPIO_ReadPin(GPIOD, Boton3_Pin))==0){`
`HAL_GPIO_WritePin(GPIOE, LedVerde_Pin, GPIO_PIN_RESET);`

Por ultimo preguntamos si Boton4 fue pulsado si es así encendemos el LedVerde y el LedRojo. → `if((HAL_GPIO_ReadPin(GPIOD, Boton4_Pin))==0){`
`HAL_GPIO_WritePin(GPIOE, LedRojo_Pin, GPIO_PIN_RESET);`
`HAL_GPIO_WritePin(GPIOE, LedVerde_Pin, GPIO_PIN_RESET);`

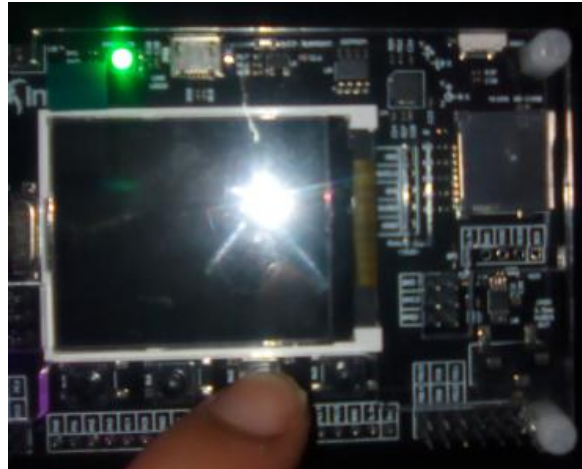
Boton1 pulsado.



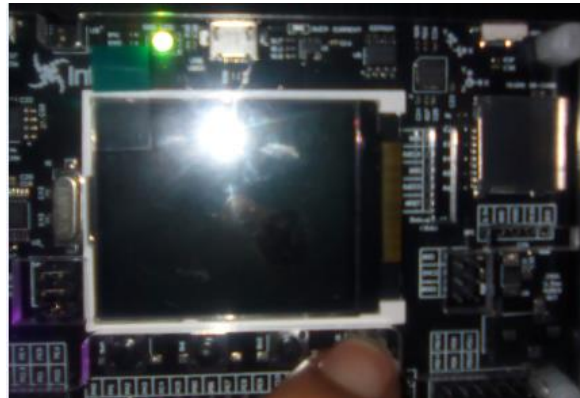
Boton2 pulsado.



Boton3 pulsado.



Boton4 pulsado.



Practica 2.-Secuencia de 8 leds con botón.

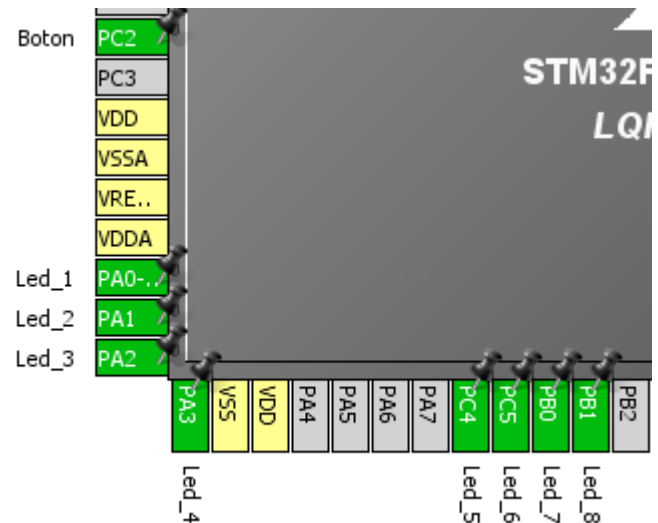
Descripción: Se deben usar un push botón de la placa de desarrollo Ophyra al pulsar este botón debe encender el Led_1 y los demás leds apagados de un módulo de 8 leds externo a la placa, al pulsar por segunda vez se debe encender el Led_2 y así hasta llegar a la pulsación nueve donde deberá prender el Led_7, la pulsación 10 deberá prender el Led_6 así hasta la pulsación 14 donde vuelve a comenzar.

Configuración de pines en STM32CubeMX.

Configuración de los 8 leds en el módulo externo empleado por lo que para encender el led se requiere un 1 lógico.

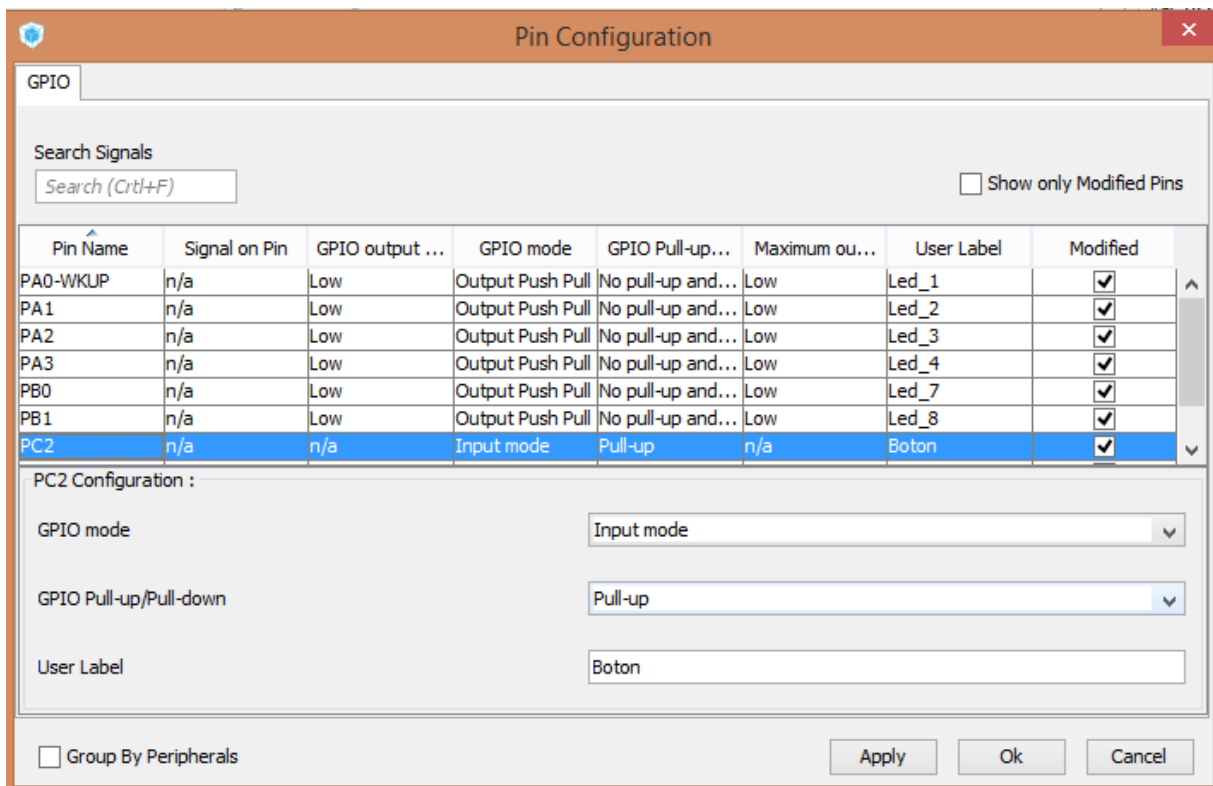


Para esta aplicación requerimos una entrada llamada Boton conectada al botón 1 (PC2) de la placa Ophyra. Utilizamos 8 salidas para encender los 8 leds del módulo externo, empleamos 4 pines de puerto A (PA0, PA1, PA2, PA3) llamados Led_1, Led_2, Led_3, Led_4, dos pines del puerto C (PC4, PC5) Led_5, Led_6 y otros dos pines de puerto B (PB0, PB1) Led_7 y Led_8.



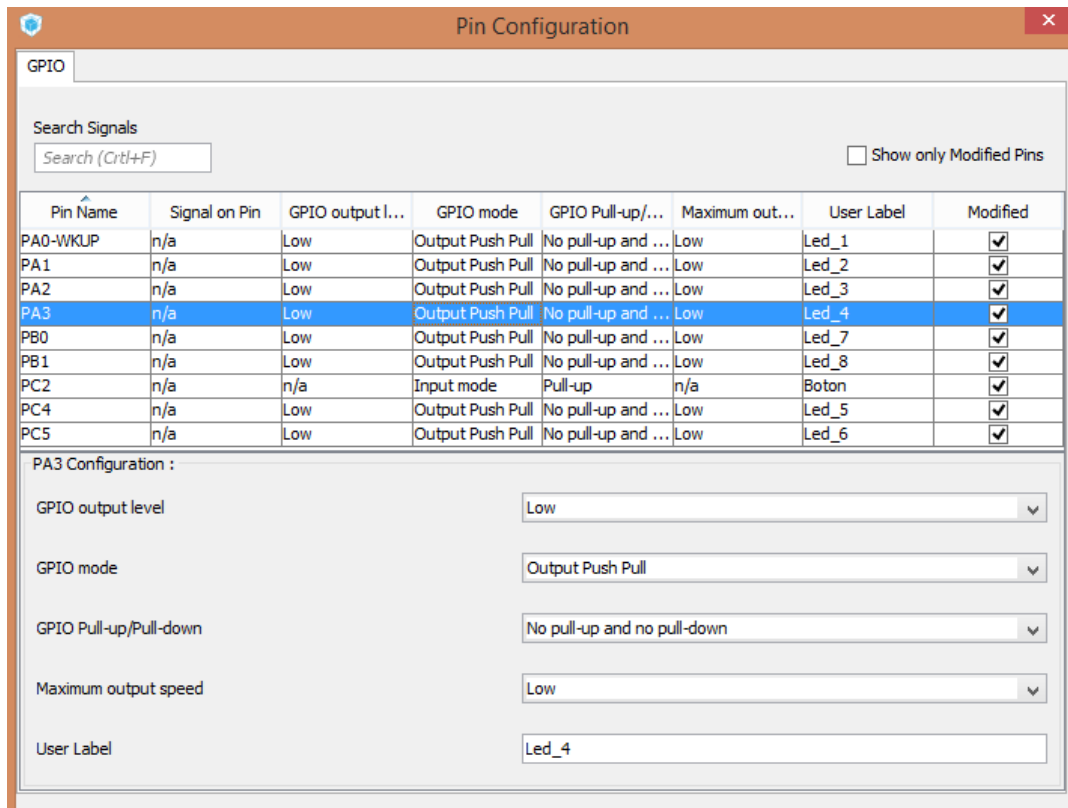
Configuración del botón.

El Boton se debe configurar en GPIO Pull-up, debido a la configuración del botón en la placa es Pull-Up.



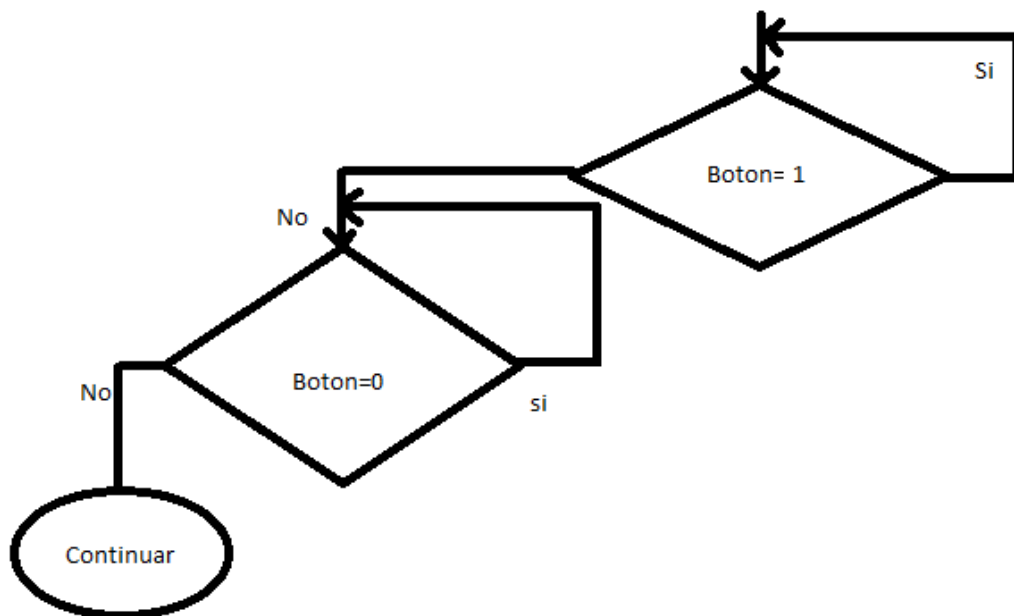
Configuración de los leds.

La configuración a seleccionar debe ser nivel lógico Low puesto que los leds deben iniciar apagados. Se debe colocar Output Push Pull pues es la configuración recomendada y no emplearemos las resistencias Pull-Up y Pull-Down no deben ser activadas.



Código en Atollic TrueStudio.

Algoritmo anti rebotes este algoritmo nos permite pulsar el botón y al pulsar eliminar efecto de rebote en la lectura del estado presente en el botón.




```
while (1)
```

```
{
```

```
while(HAL_GPIO_ReadPin(GPIOC, Boton_Pin)){
```

```
while(!HAL_GPIO_ReadPin(GPIOC, Boton_Pin)){//
```

Implementación del algoritmo anti rebote usando ciclos while.

```
HAL_GPIO_WritePin(GPIOA, Led_1_Pin,GPIO_PIN_RESET);
```

```
HAL_GPIO_WritePin(GPIOA, Led_2_Pin,GPIO_PIN_RESET);
```

```
HAL_GPIO_WritePin(GPIOA, Led_3_Pin,GPIO_PIN_RESET);
```

```
HAL_GPIO_WritePin(GPIOA, Led_4_Pin,GPIO_PIN_RESET);
```

```
HAL_GPIO_WritePin(GPIOC, Led_5_Pin,GPIO_PIN_RESET);
```

```
HAL_GPIO_WritePin(GPIOC, Led_6_Pin,GPIO_PIN_RESET);
```

```
HAL_GPIO_WritePin(GPIOB, Led_7_Pin,GPIO_PIN_RESET);
```

```
HAL_GPIO_WritePin(GPIOB, Led_8_Pin,GPIO_PIN_RESET);
```

Inicialmente todos los Leds son apagados.

```
contador++;
```

Incremento en el contador cada vez que es pulsado el botón

```
switch(contador){
```

Variable a comparar para saber en qué pulsación nos encontramos

```
case 1:
```

```
HAL_GPIO_WritePin(GPIOA, Led_1_Pin,GPIO_PIN_SET);
```

En la pulsación 1 encendemos el Led1

```
break;
```

```
case 2:
```

```
HAL_GPIO_WritePin(GPIOA, Led_2_Pin,GPIO_PIN_SET);
```

En la pulsación 2 encendemos el Led_2

```
break;
```

```
case 3:
```

```
HAL_GPIO_WritePin(GPIOA, Led_3_Pin,GPIO_PIN_SET);
```

En la pulsación 3 encendemos el Led_3

```
break;
```

```
case 4:
```

```
HAL_GPIO_WritePin(GPIOA, Led_4_Pin,GPIO_PIN_SET);
```

En la pulsación 4 encendemos el Led_4

```
break;
```

```
case 5:
```

```
HAL_GPIO_WritePin(GPIOC, Led_5_Pin,GPIO_PIN_SET);
```

En la pulsación 5 encendemos el Led_5

```
break;
```

```
case 6:
```

```
HAL_GPIO_WritePin(GPIOC, Led_6_Pin,GPIO_PIN_SET);
```

En la pulsación 6 encendemos el Led_6

```
break;
```

```
case 7:
```

```
HAL_GPIO_WritePin(GPIOB, Led_7_Pin,GPIO_PIN_SET);
```

En la pulsación 7 encendemos el Led_7

```
break;
```

```
case 8:
```

```
    HAL_GPIO_WritePin(GPIOB, Led_8_Pin, GPIO_PIN_SET);  
    break;
```



En pulsación 8 encendemos el Led_8

```
case 9:
```

```
    HAL_GPIO_WritePin(GPIOB, Led_7_Pin, GPIO_PIN_SET);  
    break;
```



En pulsación 9 encendemos el Led_7, a partir de aquí la secuencia ira en sentido contrario

```
case 10:
```

```
    HAL_GPIO_WritePin(GPIOC, Led_6_Pin, GPIO_PIN_SET);  
    break;
```



En pulsación 10 encendemos el Led_6

```
case 11:
```

```
    HAL_GPIO_WritePin(GPIOC, Led_5_Pin, GPIO_PIN_SET);  
    break;
```



En pulsación 11 encendemos el Led_5

```
case 12:
```

```
    HAL_GPIO_WritePin(GPIOA, Led_4_Pin, GPIO_PIN_SET);  
    break;
```



En pulsación 12 encendemos el Led_4

```
case 13:
```

```
    HAL_GPIO_WritePin(GPIOA, Led_3_Pin, GPIO_PIN_SET);  
    break;
```



En pulsación 13 encendemos el Led_3

```
case 14:
```

```
    HAL_GPIO_WritePin(GPIOA, Led_2_Pin, GPIO_PIN_SET);  
    contador=0;  
    break;
```



En pulsación 14 encendemos el Led_2, reiniciamos contador lo cual nos reiniciara el número de pulsaciones , nos posicionara en el caso1 si se hace una nueva pulsación

```
}
```

```
    HAL_Delay(300);
```

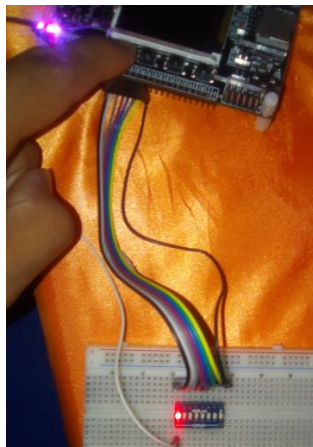


Este pequeño retardo ayuda al algoritmo anti rebote porque nos da un retardo entre pulsar y dejar de presionar el botón ya que la velocidad del micro controlador es muy alta y no permite visualizarlo de manera correcta.

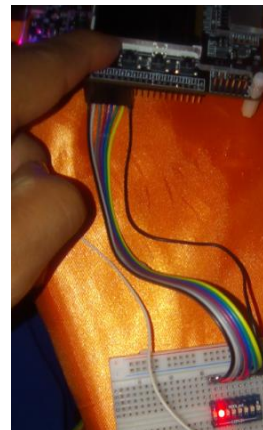
```
}  
/* USER CODE END 3 */
```

```
}
```

Pulsación 1.



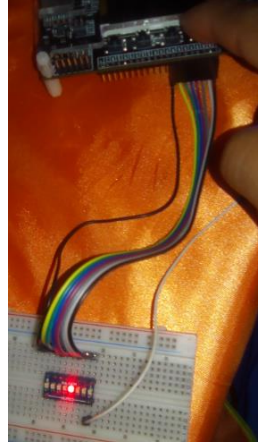
Pulsación 2.



Pulsación 3



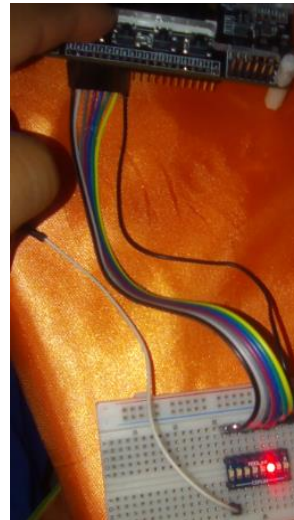
Pulsación 4



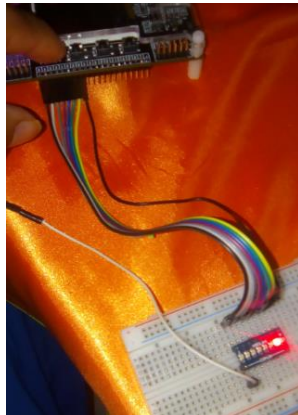
Pulsación 5



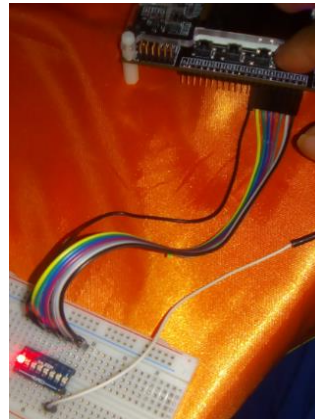
Pulsación 6



Pulsación 7



Pulsación 8

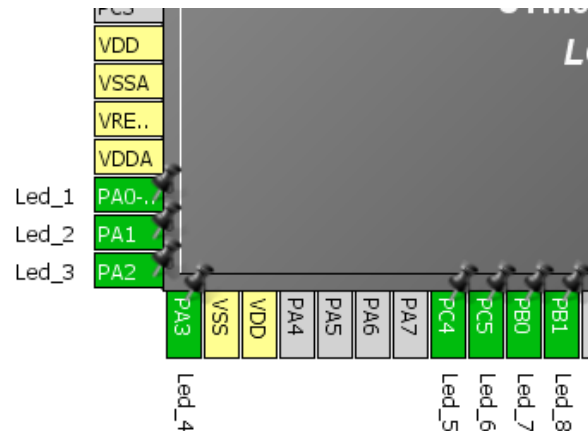


Practica 3.-Secuencia de 8 leds automática.

Descripción: Se utilizara la placa de desarrollo Ophyra y un módulo de 8 leds externos a la placa con la configuración de la practica 2, la secuencia debe empezar con encender el Led_1 y todos los demás apagados, después encender el Led_2 y todos los demás apagados, así hasta llegar al Led_8 a partir de aquí se debe hacer el proceso al revés en la posición 9 se de encender el Led_7 y todos los demás apagados hasta llegar a la posición 1 donde se reinicia el proceso.

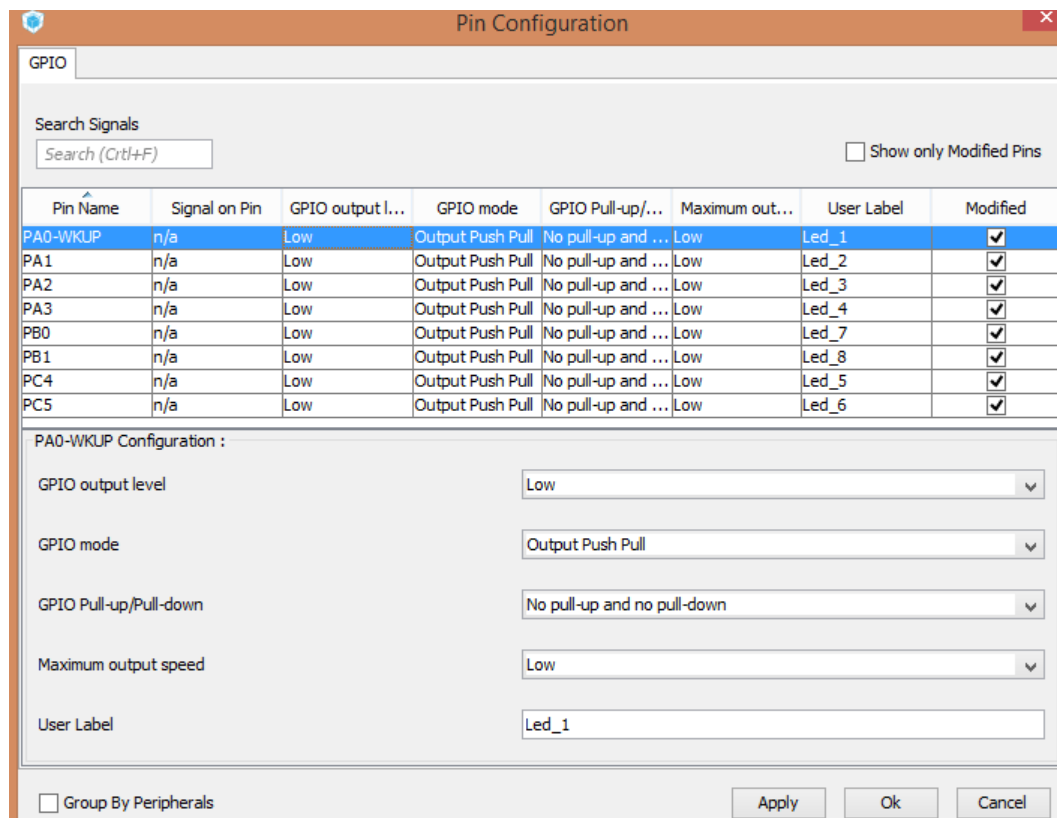
Configuración de pines para los Leds en STM32CubeMX.

Emplearemos 8 salidas (GPIO_Output) para encender los 8 leds del módulo externo, empleamos 4 pines de puerto A (PA0, PA1, PA2, PA3) llamados Led_1, Led_2, Led_3, Led_4, dos pines del puerto C (PC4, PC5) Led_5, Led_6 y otros dos pines de puerto B (PB0, PB1) Led_7 y Led_8.



Configuración GPIO de los Leds.

Se selecciona el nivel de salida Low para inicialmente tener los Leds se apagados, se debe seleccionar un Output Push Pull pues es la configuración recomendada y no emplearemos las resistencias Pull-up y Pull-down.



Programación de la aplicación en Atollic TrueStudio.

```
/* USER CODE BEGIN WHILE */
```

```
while (1) {
```

⇒ Ciclo Infinito

```
/* USER CODE END WHILE */
```

```
/* USER CODE BEGIN 3 */
```

```
HAL_GPIO_WritePin(GPIOA, Led_1_Pin,GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOA, Led_2_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOA, Led_3_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOA, Led_4_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOC, Led_5_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOC, Led_6_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOB, Led_7_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOB, Led_8_Pin,GPIO_PIN_RESET);
```

Se prende el Led_1 y todos los demás se apagan.

```
HAL_Delay(1000);
```

⇒ Damos un retardo de 1 segundo

```
HAL_GPIO_WritePin(GPIOA, Led_1_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOA, Led_2_Pin,GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOA, Led_3_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOA, Led_4_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOC, Led_5_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOC, Led_6_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOB, Led_7_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOB, Led_8_Pin,GPIO_PIN_RESET);
```

Se prende el Led_2 y todos los demás se apagan.

```
HAL_Delay(1000);
```

⇒ Damos un retardo de 1 segundo

```
HAL_GPIO_WritePin(GPIOA, Led_1_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOA, Led_2_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOA, Led_3_Pin,GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOA, Led_4_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOC, Led_5_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOC, Led_6_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOB, Led_7_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOB, Led_8_Pin,GPIO_PIN_RESET);
HAL_Delay(1000);
```

⇒ Prendemos el Led_3 y apagamos los demas. Damos un retardo de 1 segundo.

```
HAL_GPIO_WritePin(GPIOA, Led_1_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOA, Led_2_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOA, Led_3_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOA, Led_4_Pin,GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOC, Led_5_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOC, Led_6_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOB, Led_7_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOB, Led_8_Pin,GPIO_PIN_RESET);
HAL_Delay(1000);
```

⇒ Prendemos el Led_4 y apagamos los demas. Damos un retardo de 1 segundo.

```
HAL_GPIO_WritePin(GPIOA, Led_1_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOA, Led_2_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOA, Led_3_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOA, Led_4_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOC, Led_5_Pin,GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOC, Led_6_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOB, Led_7_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOB, Led_8_Pin,GPIO_PIN_RESET);
HAL_Delay(1000);
```

⇒ Prendemos el Led_5 y apagamos los demas. Damos un retardo de 1 segundo.

Encendemos el Led_6
y apagamos los demás
Damos un retardo de
1 segundo

Encendemos el Led_7 y apagamos los demas.Damos un retardo de 1 segundo.

Encendemos el Led_8 y apagamos los demas. Damos un retardo de 1 segundo.

Encendemos el Led_7
y apagamos los
demás.

Damos un retardo de 1
segundo.

Encendemos el Led_6
y apagamos los
demás.
Damos un retardo de 1
segundo.

Encendemos el Led_5
y apagamos los
demás.
Damos un retardo de 1
segundo.

```

HAL_GPIO_WritePin(GPIOA, Led_3_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOA, Led_4_Pin,GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOC, Led_5_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOC, Led_6_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOB, Led_7_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOB, Led_8_Pin,GPIO_PIN_RESET);
HAL_Delay(1000);

```

Encendemos el Led_4
y apagamos los
demás.

Damos un retardo de 1
segundo.

```

HAL_GPIO_WritePin(GPIOA, Led_1_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOA, Led_2_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOA, Led_3_Pin,GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOA, Led_4_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOC, Led_5_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOC, Led_6_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOB, Led_7_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOB, Led_8_Pin,GPIO_PIN_RESET);
HAL_Delay(1000);

```

Encendemos el Led_3
y apagamos los
demás.

Damos un retardo de 1
segundo.

```

HAL_GPIO_WritePin(GPIOA, Led_1_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOA, Led_2_Pin,GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOA, Led_3_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOA, Led_4_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOC, Led_5_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOC, Led_6_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOB, Led_7_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOB, Led_8_Pin,GPIO_PIN_RESET);
HAL_Delay(1000);

```

Encendemos el Led_2
y apagamos los
demás.

Damos un retardo de 1
segundo.

```

}

```

```

/* USER CODE END 3 */

```

```

}

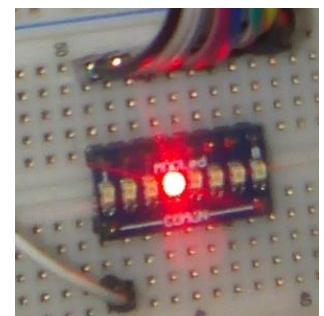
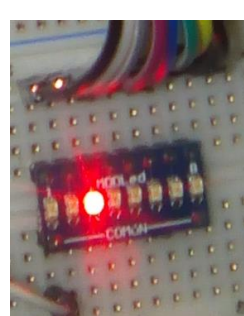
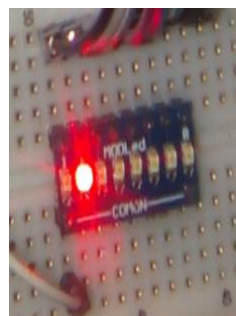
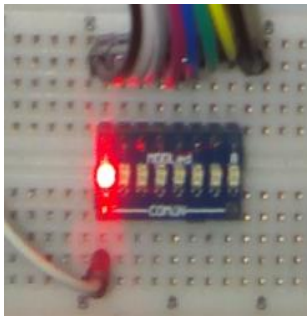
```

Posición 1

Posición 2

Posición 3

Posición 4

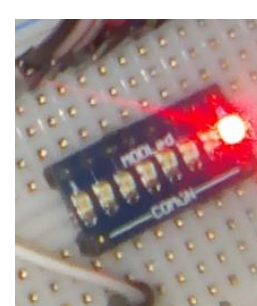
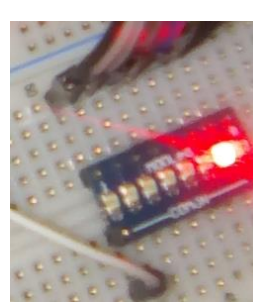
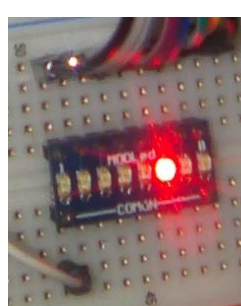
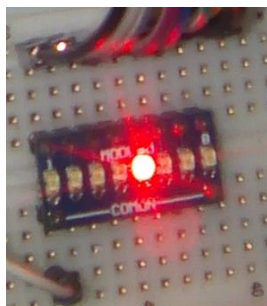


Posición 5

Posición 6

Posición 7

Posición 8



Conclusiones.

El micro controlador a emplear es uno muy potente pues posee una arquitectura de 32 bits, sus instrucciones se ejecutan bajo una arquitectura RISC, puede trabajar a una velocidad de 168 MHz, puede ejecutar una instrucción en un solo ciclo de reloj. Para poder programar este microcontrolador es necesario el empleo principalmente de dos softwares STM32CubeMX donde se realizarán las configuraciones de los Pines GPIO dependiendo de la aplicación previamente establecida y planteada. El otro software es Atollic TrueStudio en este programa podremos desarrollar el código necesario para que nuestra aplicación en el microcontrolador funcione; dentro de este entorno de programación basado en C++ aprendimos las sintaxis para la lectura y escritura en los pines del microcontrolador, también se aprendió a cómo generar los retardos con su instrucción en Atollic el cual es muy estricto en la escritura de estas instrucciones por lo que debe tener cuidado al empleo de mayúsculas y minúsculas pues puede provocar un error de compilación.

Bibliografía.

Perles Àngel; ARM Cortex-M práctico. 1 - Introducción a los microcontroladores STM32 de St. Universidad Politécnica de Valencia, Valencia 2017, 206 págs.