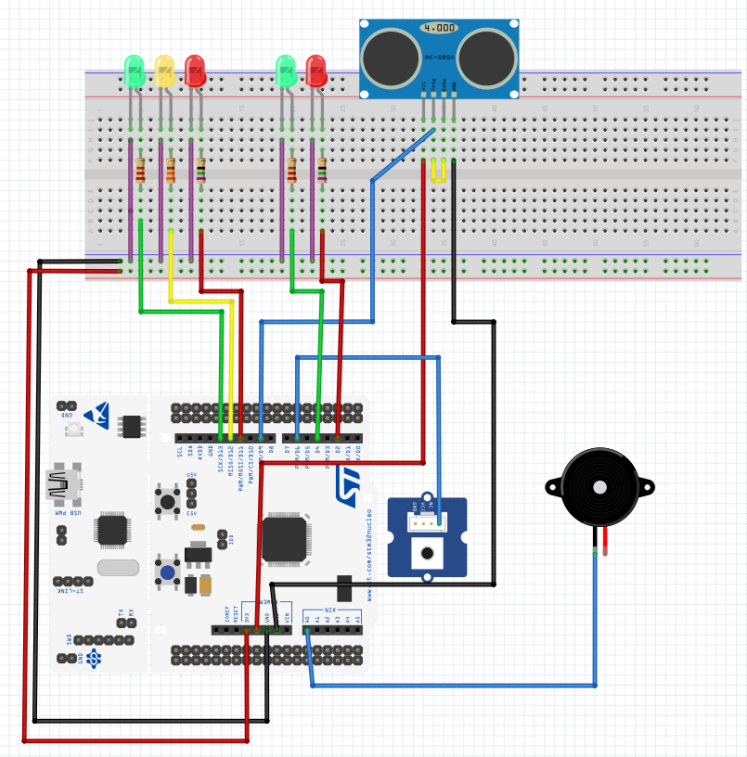
– Lab 5: Adding sound alarm for blind people (DAC/PWM) –

Objetivo:

Nuestro cliente quiere ampliar la funcionalidad del semáforo desarrollado hasta la fecha y ha decido incluir un sistema que sea capaz de avisar a los/as invidentes cuándo es posible cruzar la calle de forma segura. Además, cuando el indicador de peatones empiece a parpadear el sonido emitido se deberá acelerar con el objetivo de indicar que el tiempo para cruzar está finalizando.

En nuestro caso, hemos decidió realizar la implementación mediante la placa F411RE.

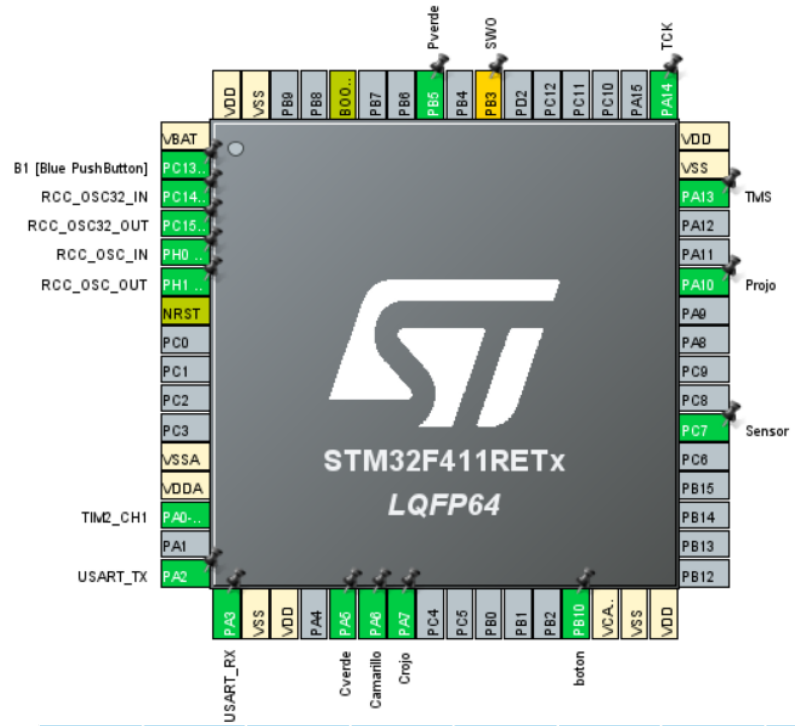
Esquema:



Componentes:

- Placa STM32 F411RE
- Placa Grove
- Botón para cable de cuatro canales
- Ultrasonidos HC-SR04
- 2 LED rojos, 2 LED verdes, LED Amarillo
- Respectivas resistencias (resistencia explicada en prácticas anteriores)
- Buzzer

Esquema STM32CubeMX:

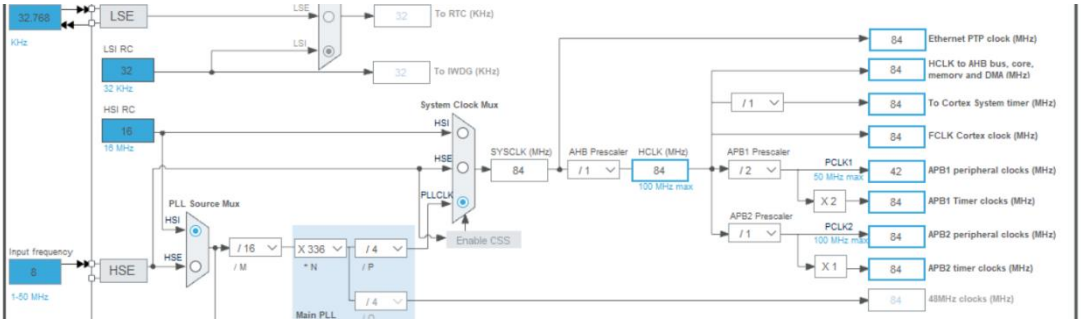


Pin Name	Signal on Pin	GPIO outpu...	GPIO mode	GPIO Pull-u...	Maximum o...	User Label	Modified
PA5	n/a	Low	Output Pus...	No pull-up ...	Low	Cverde	☑
PA6	n/a	Low	Output Pus...	No pull-up ...	Low	Camarillo	☑
PA7	n/a	Low	Output Pus...	No pull-up ...	Low	Crojo	☑
PA10	n/a	Low	Output Pus...	No pull-up ...	Low	Projo	☑
PB5	n/a	Low	Output Pus...	No pull-up ...	Low	Pverde	☑
PB10	n/a	n/a	External Int...	No pull-up ...	n/a	boton	☑
PC7	n/a	Low	Output Pus...	No pull-up ...	Low	Sensor	☑
PC13-ANTI...	n/a	n/a	External Int...	No pull-up ...	n/a	B1 [Blue P...	☑

Para esta práctica además hemos habilitado el Timer 2, activando la generación de señal PWM por el canal 1.

Hemos configurado el prescaler y el registro ARR de la siguiente forma:

Esto es debido a que nuestro timer actúa con una frecuencia de 84 MHz.



Calculamos el valor del prescaler de tal forma:

(84 MHz / (255*255)) = 1291

El -1 es debido a que el mínimo valor (0) se indica como un 1 en los registros, tal y como explicamos en clase (se desplaza un bit a la izquierda).

El resto del proyecto STM se mantiene igual respecto a la última práctica.

Explicación de la solución:

Declaramos el timer2.

```
/* USER CODE BEGIN 2 */
HAL_TIM_Base_Start_IT(&htim11);
HAL_TIM_PWM_Start(&htim2,TIM_CHANNEL_1);
/* USER CODE END 2 */
```

En el código main no hemos modificado nada, únicamente hemos añadido una instrucción para evitar que comience a pitar el buzzer.

```
while (1)
{
  /* USER CODE END WHILE */
  htim2.Instance->CCR1=0;
  secuencia(next_state);


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

Dentro de la función secuencia, en ST_PVerde (caso de que el LED verde de peatones está activo) hemos añadido un bucle for para añadir la secuencia de pitido.

```
case ST_PVerde:
  GPIOA->ODR = GPIO_ODR_OD7_Msk; //Coche rojo activo
  GPIOB -> ODR = GPIO_ODR_OD5_Msk; //Peaton verde activo
  //HAL_Delay(5000);
  for (size_t i = 0; i < 5; i++)
  {
    htim2.Instance->CCR1=0;
    HAL_Delay(500);
    htim2.Instance->CCR1=50;
    HAL_Delay(500);
  }
```

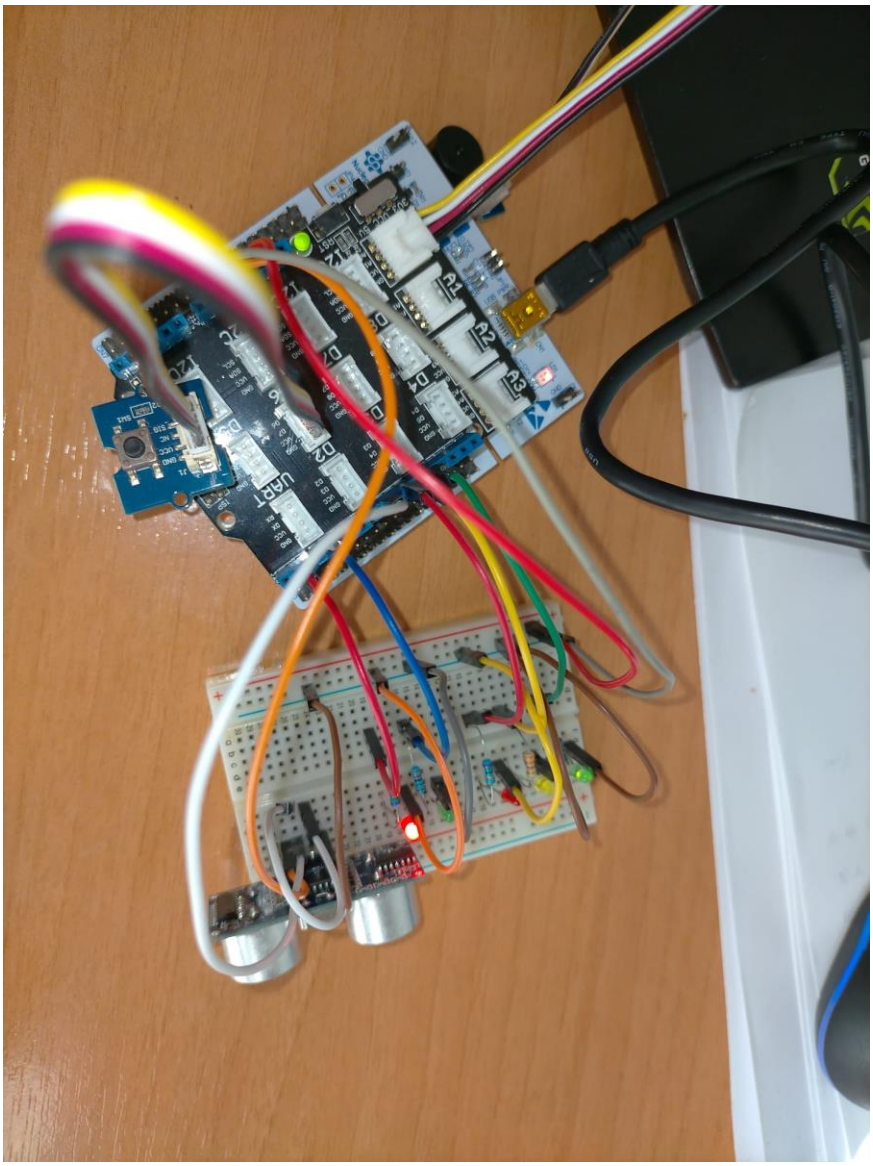
De igual forma lo hemos añadido en la función task_parpadeo, cambiando la frecuencia para que ofrezca un tono distinto.

```
void task_parpadeo(void){ //coche rojo activo y peaton verde parpadea
    for (size_t i = 0; i < 6; i++)
    {
      HAL_GPIO_WritePin(Pverde_GPIO_Port,Pverde_Pin,GPIO_PIN_RESET);
      htim2.Instance->CCR1= 0;
      HAL_Delay(250);
      HAL_GPIO_WritePin(Pverde_GPIO_Port,Pverde_Pin,GPIO_PIN_SET);
      htim2.Instance->CCR1= 200;
      HAL_Delay(250);
      HAL_GPIO_WritePin(Pverde_GPIO_Port,Pverde_Pin,GPIO_PIN_RESET);
    }
}
```

En el siguiente fragmento de código declaramos todos los valores del timer (generados con STM32 Cube MX).

```
/* USER CODE END TIM2_Init 1 */
htim2.Instance = TIM2;
htim2.Init.Prescaler = 1291-1;
htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
htim2.Init.Period = 255;
htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
{
  Error_Handler();
}
sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
```

Montaje:



Ejemplo de salida por terminal: