Trabajo Micros

SAMUEL SÁNCHEZ CRUZ - 55450
MARÍA SANZ PIÑA – 55463
CÉSAR ZABALA MARTÍN - 55540

Índice

Introducción

Se ha implementado un sistema de domótica de control de luces y apertura de puertas.

En el control de apagado y encendido de las luces se ha establecido dos modos de funcionamiento: manual, gobernado mediante un botón y automático mediante una fotorresistencia. En cuanto al control de apertura y cierre de la puerta, se empleará un sensor para detectar la presencia o no presencia de una persona y un servomotor que moverá la puerta a la posición indicada.

Es decir, nuestro sistema tiene los siguientes objetivos:

1. Detección de un objeto a una distancia determinada
2. Accionamiento de un servomotor
3. Indicador de luz incidente en la casa
4. Detección del botón pulsado

Componentes

Placa STM32F411-E

El microcontrolador que se empelará será la placa STM32f411 de ARM Cortex-M4 .

Sensor HC-SR04

El sensor ultrasónico HC-SR04 mide el tiempo que tarda un pulso ultrasónico en viajar desde el sensor hasta el objeto y regresar. En nuestro caso, lo emplearemos para detectar si existe una persona delante de la puerta y calcular la distancia a la que se haya. En función de dicha distancia, se activará o no el servomotor para abrir dicha puerta. Además, si la presencia del objeto sigue activa (distancia detectada por el sensor), el HC-SR04 le enviará un mensaje al servomotor para que mantenga su posición y no cierre la puerta hasta que el sensor deje de detectar objeto alguno.

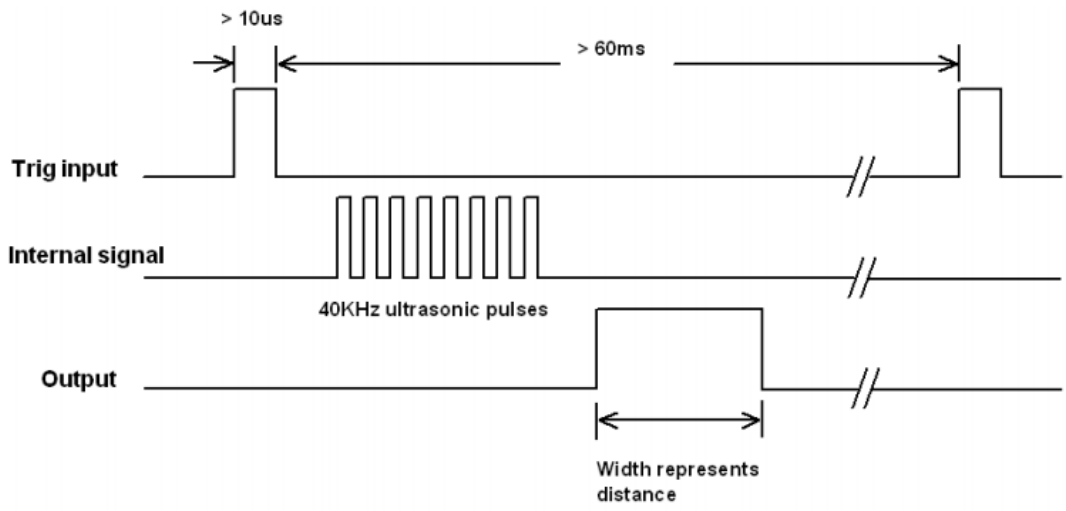
Las cuatro patillas del sensor son las siguientes:

TRIG: este pin se utiliza para iniciar la medición enviando un pulso ultrasónico, la conectaremos al pin de salida PA5 (GPIO_Output). Es el pin del canal1 del TIM3.

GND: pin conectado a tierra

ECHO: pin donde el sensor devuelve el eco del pulso ultrasónico, la conectaremos al pin de entrada PA7 (GPIO_Input). Es el pin del canal1 del TIM3.

VCC: conectado al pin de alimentación de 5V (voltaje de trabajo de este sensor).



Para hacer uso de este sensor hemos dispuesto 2 temporizadores, el primero, "TIM2", va conectado a la patilla ECHO en modo *Input Capture direct mode* que se encarga de generar una interrupción cada vez que recoge el valor de tiempo que ha tardado la señal en llegar. Lo hemos configurado con un periodo de 2ms usando la siguiente formula:

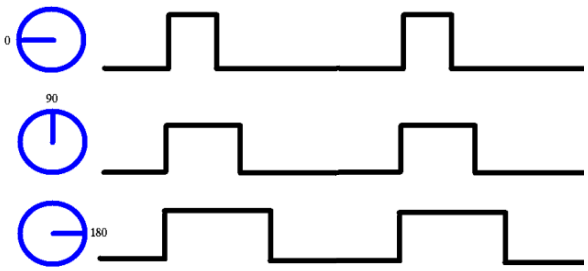
$$Période = \frac{1}{\frac{Clk_{TIM}}{Prescaler * Contador}}$$

Elegimos un Prescaler de 50 y un contador de 2000.

El segundo temporizador, "TIM3", será el encargado de mandarle pulsos de PWM a la patilla Trigger del HC-SR04. Como muestra la anterior gráfica aportada por la hoja de características, el mínimo tiempo entre pulsos de Trigger del HC-SR04 ha de ser de 60 ms. Por lo tanto, haciendo uso de la fórmula anterior y sustituyendo el periodo por 60 ms elegimos un Prescaler de 50 y un contador de 60000.

Servomotor SG90

El servomotor será el encargado de abrir y cerrar la puerta en función de la información detectada por el sensor ultrasónico.



Para hacer uso de este motor hemos dispuesto de un tercer temporizador, "TIM4" que se encargará de mandar pulsos PWM al motor. Como muestra la imagen adjunta dependiendo del ancho de pulso que se le mande irá a una posición u a otra. Para 0° serán 0,5ms de pulso y para 180° 2ms.

Sabiendo que en la hoja de características la frecuencia a la que funciona este motor es de 50Hz, es decir, un periodo de 20ms elegimos un prescaler de 1000 y un contador de 1000 para facilitar los cálculos.

Dependiendo del valor del contador se mandará un pulso de duración mayor o menor. En nuestro caso, al haber puesto un contador de 1000 y tener una frecuencia de reloj del temporizador de 50MHz, si se mandara un pulso del 100% se estaría mandando un pulso de ancho 20ms.

Esto implica que si queremos un valor de 120° (1,8ms) lo único que tenemos que hacer es:

$$1{,}8 * \frac{1000}{20} = 90$$

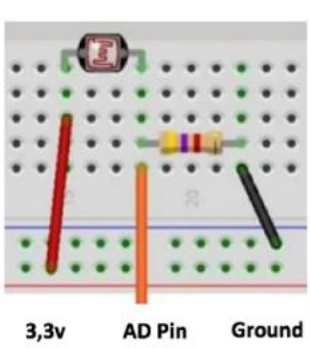
Ese valor de noventa es el que le pasamos al motor. En nuestro montaje la puerta cerrada es el motor a 120° y para abrirla se manda el motor a 0° es decir un pulso de 0,5ms que es mandarle un valor de 25. Por eso aparecen esos valores en el código.

Led

Emplearemos dos diodos led para simular el comportamiento de las luces de la casa. Ambos estarán conectados en serie con una resistencia de 220Ω.

Fotorresistencia LDR

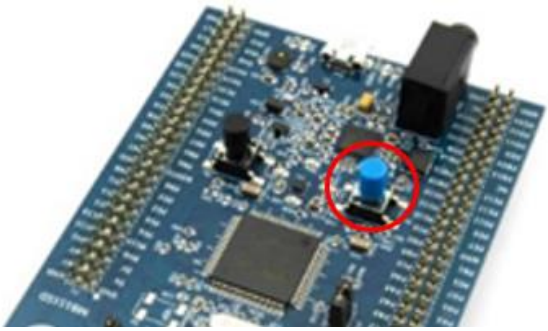
El LDR es una resistencia eléctrica la cual varía su valor en función de la cantidad de luz que incide sobre su superficie. Si la luz detectada está por debajo de un umbral, mandará una señal al led para que se encienda. En caso contrario, detectará que existe suficiente luz y no será necesario el encendido del led



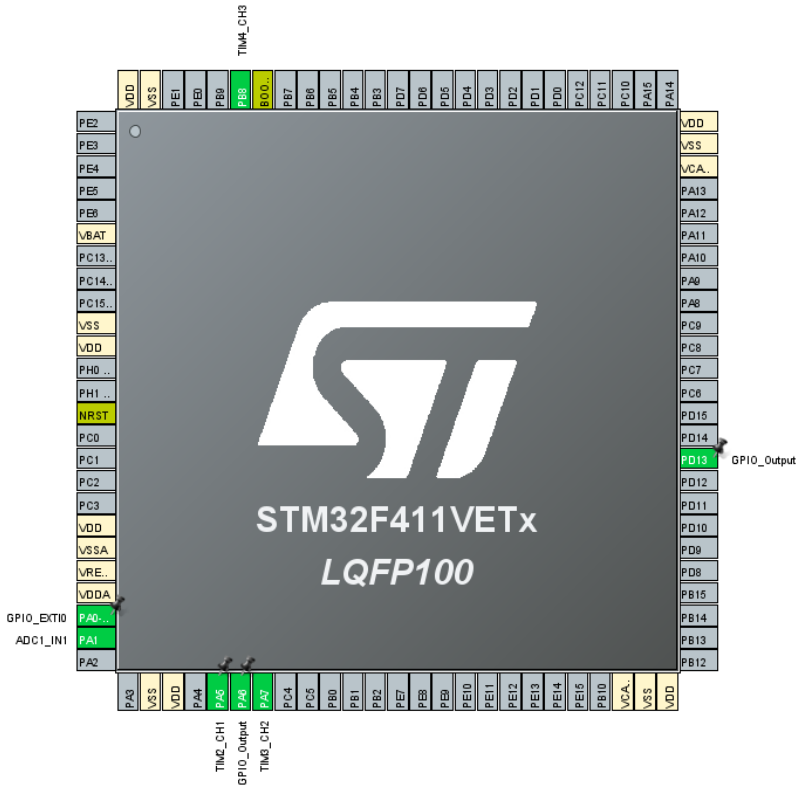
Conectaremos una resistencia 330Ω en serie con la LDR como se muestra en la imagen superior.

Botón

Para controlar las luces de manera manual emplearemos el botón de la placa. (PA0). Se ha configurado para que, una vez pulsado el botón para encender las luces, aunque el LDR detecte que hay suficiente luz exterior para que las luces de nuestra vivienda estén apagadas, estas no se apaguen ya que el botón ha sido pulsado.



Configuración



Placa

PA0: botón de la placa (GPIO_EXTI0).

PA1: patilla común del LDR y la resistencia (ADC1).

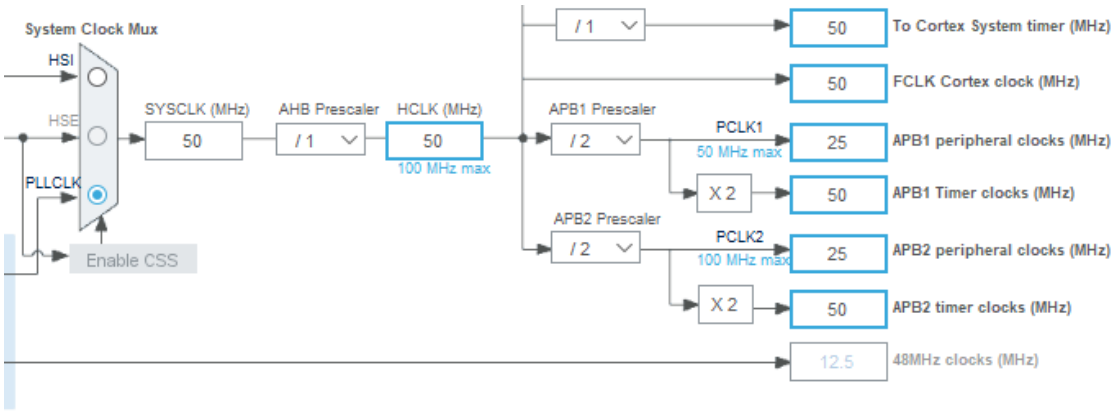
PA5: Trigger del sensor ultrasónico. TIM2- Chanel 1.

PA6: ánodo del led (patilla larga).

PA7: Echo del sensor ultrasónico. TIM3- Chanel 2.

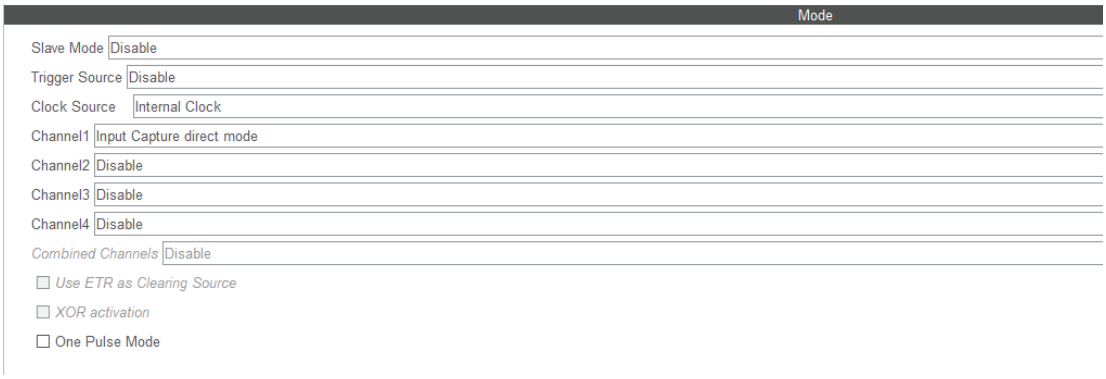
PB8: patilla del PWM del servomotor. TIM4- Chanel 3.

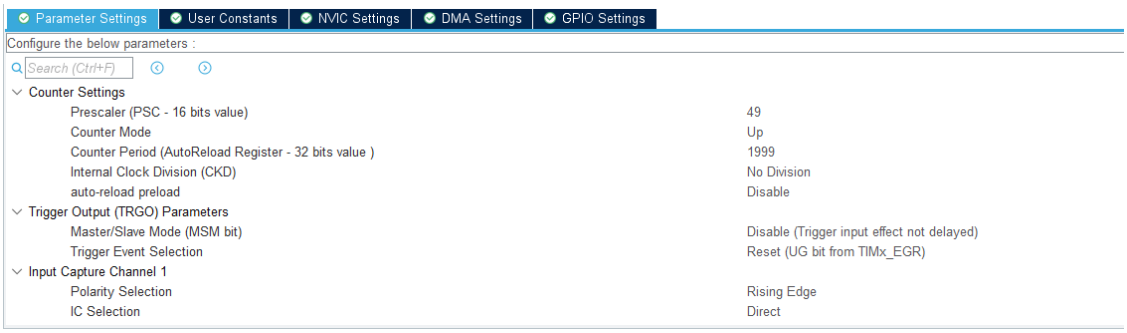
Reloj para los temporizadores de 50Mhz.



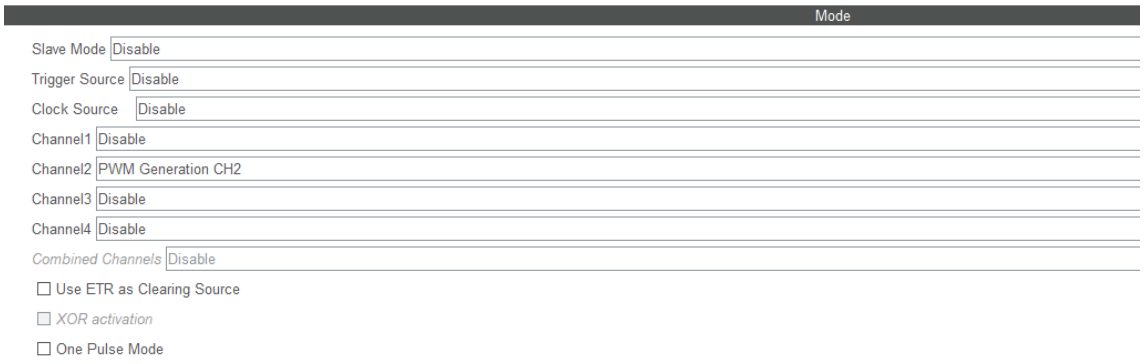
Temporizadores

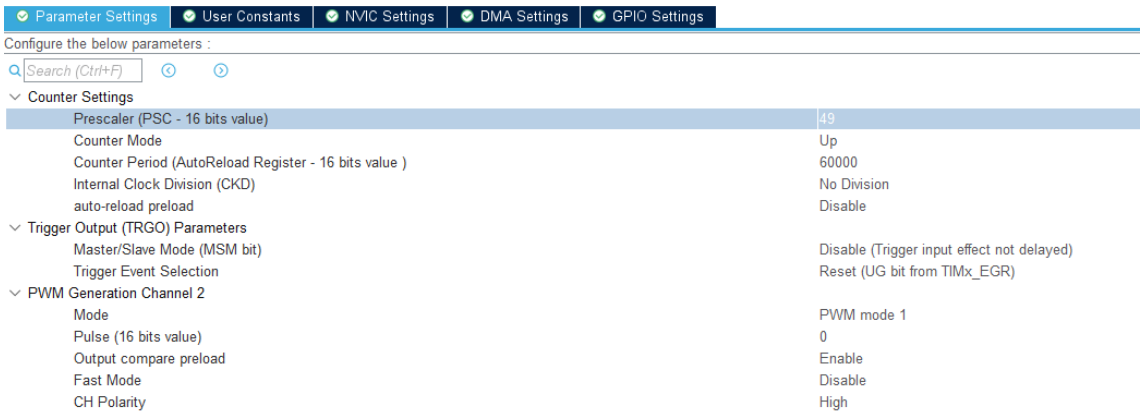
TIM2:



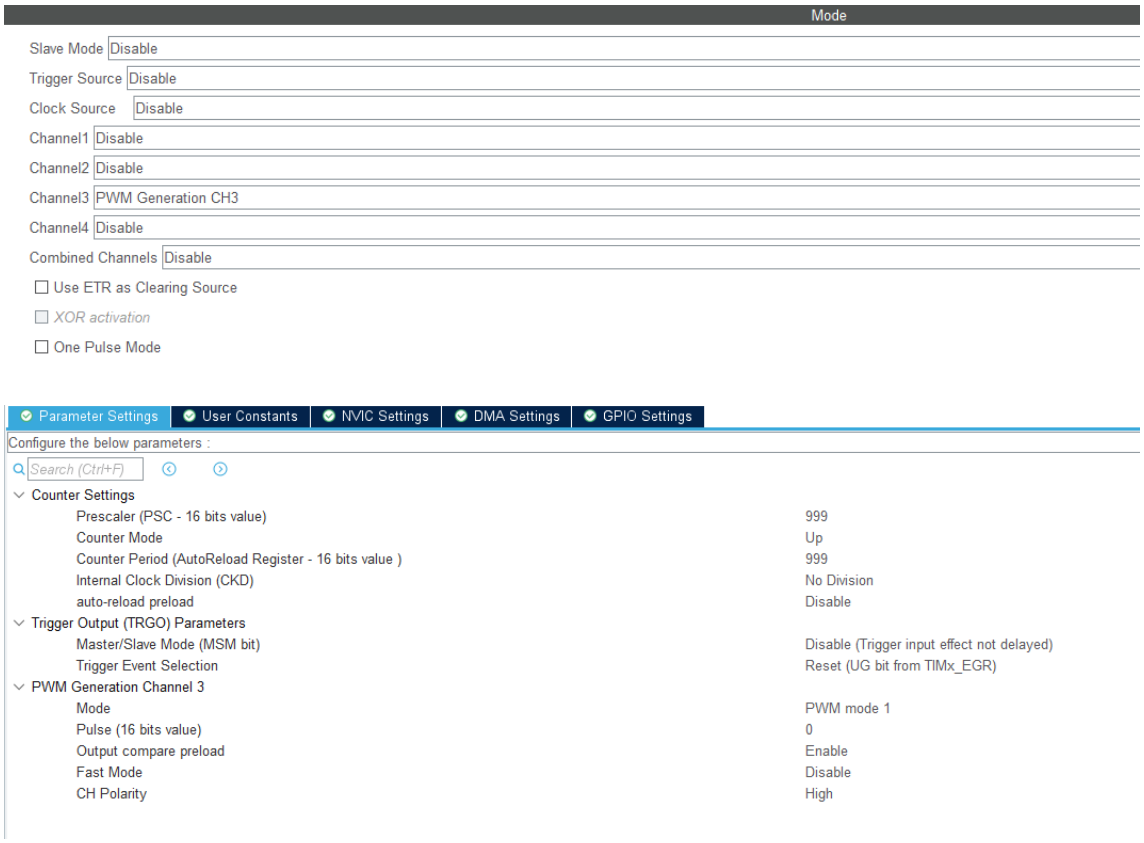


TIM3:

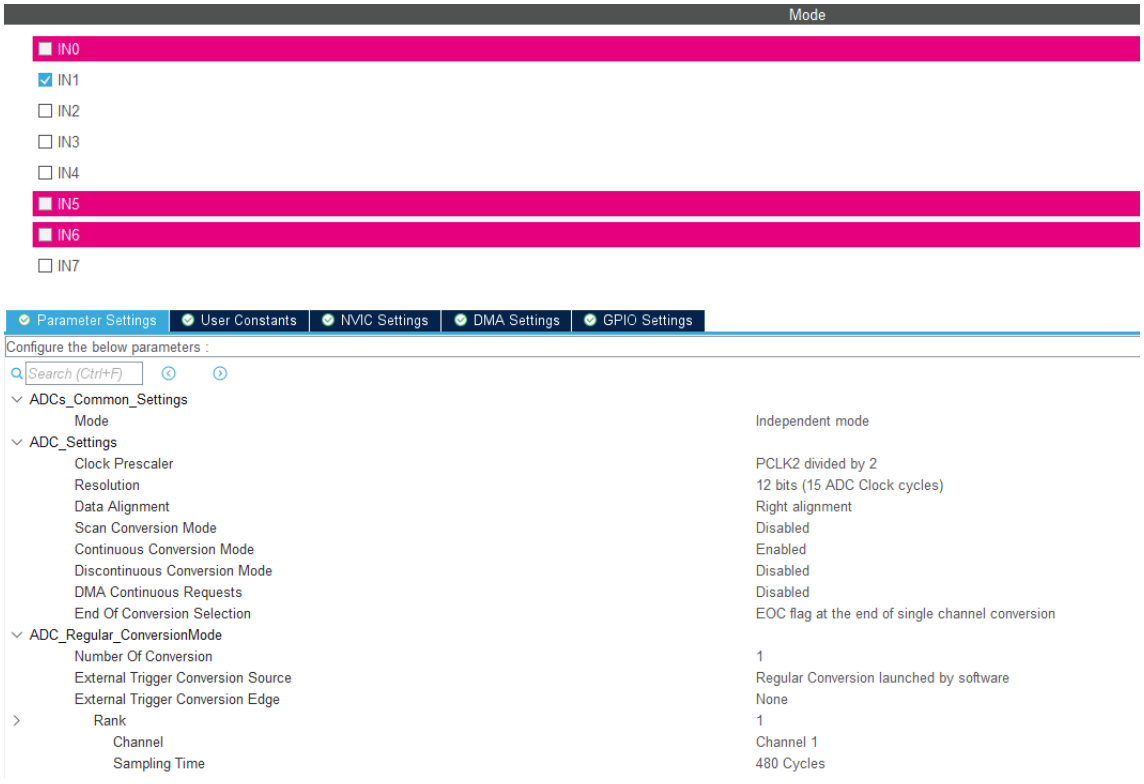




TIM4:



Conversor ADC



Para el conversor analógico-digital elegimos el canal 1 que se corresponde con el pin PA1. Para su modo de funcionamiento hemos elegido que sea en modo de conversión continuo para que esté haciendo conversiones continuamente y un sampling time de 480 ciclos.

Interrupciones

NVIC Interrupt Table	Enabled
Non maskable interrupt	☑
Hard fault interrupt	☑
Memory management fault	☑
Pre-fetch fault, memory access fault	☑
Undefined instruction or illegal state	☑
System service call via SWI instruction	☑
Debug monitor	☑
Pendable request for system service	☑
Time base: System tick timer	☑
PVD interrupt through EXTI line 16	☐
Flash global interrupt	☐
RCC global interrupt	☐
EXTI line0 interrupt	☑
ADC1 global interrupt	☑
TIM2 global interrupt	☑
TIM3 global interrupt	☑
TIM4 global interrupt	☑
FPU global interrupt	☐

Están habilitadas todas las interrupciones de temporizadores, conversor y el botón de la placa.

Código

Nuestro código va a funcionar principalmente por interrupciones, por lo tanto, tenemos 3 funciones principales con la mayoría del código y el main es usado principalmente para inicializar los temporizadores y gestionar la apertura de la puerta. Nuestro código persigue conseguir lo siguiente:

Si el sensor de distancia detecta que hay algo a menos de 5cm la puerta se abrirá a través del servomotor y permanecerá abierta hasta que deje de detectar el objeto tras lo cual se cerrará. Además, si detecta un valor bajo de luminosidad el LDR se encenderán los leds ubicados en el techo. Si el botón de la placa es pulsado los leds se encenderán igualmente, aunque haya mucha luz.

A continuación, vamos a explicar las funciones:

```
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
```

Esta función se hace cargo de gestionar las interrupciones del temporizador 2, el cual está asociado al ECHO del sensor de posición. Cuenta con 2 flags, Objeto y Is_First_Captured. Esta última sirve para saber si ya se ha mandado un pulso o no.

Como el sensor manda un pulso que continua hasta que recibe la señal de vuelta vamos a codificar que cuando empiece este pulso se cambie la flag y comience a medir el temporizador en modo flanco de bajada.

Así, cuando ocurra este flanco de bajada se calculará la diferencia de tiempos recogidos y volverá a cambiarse la flag y el modo a flanco de subida.

Si cuando mida, la distancia es inferior a 5cm también se encargará de hacer cambiar la flag de Objeto y encender el led naranja de la placa.

Su código es el siguiente:

```
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
    if (htim->Instance==TIM2)  //  interrupción es el canal 1 (canal configurado TIM2)
    {
        if (Is_First_Captured == 0) // Si el primer valor no ha sido capturado
        {
            IC_Val1 = HAL_TIM_ReadCapturedValue(&htim2, TIM_CHANNEL_1); // Leer el primer valor capturado
            Is_First_Captured = 1;  // Marcar como el primer valor capturado
            __HAL_TIM_SET_CAPTUREPOLARITY(&htim2, TIM_CHANNEL_1, TIM_INPUTCHANNELPOLARITY_FALLING); // Cambiar la polaridad al flanco descendente
        }
        else if (Is_First_Captured == 1)   // Si el primer valor ya ha sido capturado
        {
            IC_Val2 = HAL_TIM_ReadCapturedValue(&htim2, TIM_CHANNEL_1);  // Leer el segundo valor capturado
            __HAL_TIM_SET_COUNTER(htim, 0);  // Reiniciar el contador

            if (IC_Val2 > IC_Val1)
            {
                Difference = IC_Val2 - IC_Val1; // Calcular la diferencia entre los valores capturados
            }
            Distance = Difference * .034 / 2; // Calcular la distancia basándose en la diferencia de tiempo

            if (Distance <= 5)
            {
                HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_SET); // Encender LED en naranja

                Objeto=1;
            }
            else
            {
                HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_RESET);// Apagar el LED
                Objeto=0;
            }

            Is_First_Captured = 0; // Restablecer el estado para la próxima medición

            // Establecer la polaridad para el flanco ascendente
            __HAL_TIM_SET_CAPTUREPOLARITY(&htim2, TIM_CHANNEL_1, TIM_INPUTCHANNELPOLARITY_RISING);

        }
    }
}
```

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
```

Encargada de las interrupciones generadas por el botón de la placa. Esta función tiene una construcción muy simple vista en las prácticas de la asignatura. Además, se le ha añadido un circuito antirrebotes para evitar falsas lecturas del botón. Solo tomará lecturas que tengan 100ms de diferencia. Si cumplen esta condición los leds se encenderán, aunque haya un valor de luminosidad elevado sin importar el umbral de luminosidad. El flag luz sirve para saber si el botón esta pulsado o no. Su código es el siguiente:

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) //Interrupciones del botón
{
 tiempoactual=HAL_GetTick();
 if((tiempoactual-ultimapulsacion)>debouncer) // si el boton a sido pulsado o ha detectado mas de una pulsación solo valida si han pasado 100ms
 {
     ultimapulsacion=tiempoactual;
     if(GPIO_Pin== GPIO_PIN_0)
     {
        luz=!luz; //Cambiar el estado de la luz cuando pulsamos el botón
        if (luz)
        {
         HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13,GPIO_PIN_SET);
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_SET);
        }
        else
        {
         HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13,GPIO_PIN_RESET);
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_RESET);
        }
     }
 }
}
```

```
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
```

Por último, esta función se encarga del conversor analógico-digital el cual si detecta un valor de luminosidad inferior a 500 y no está el botón pulsado procederá a encender los LEDs. Tiene el siguiente código:

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) //Interrupciones del botón
{
 tiempoactual=HAL_GetTick();
 if((tiempoactual-ultimapulsacion)>debouncer) // si el boton a sido pulsado o ha detectado mas de una pulsación solo valida si han pasado 100ms
 {
     ultimapulsacion=tiempoactual;
     if(GPIO_Pin== GPIO_PIN_0)
     {
        luz=!luz; //Cambiar el estado de la luz cuando pulsamos el botón
        if (luz)
        {
         HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13,GPIO_PIN_SET);
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_SET);
        }
        else
        {
         HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13,GPIO_PIN_RESET);
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_RESET);
        }
     }
 }
}
```

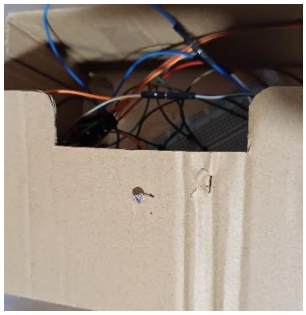
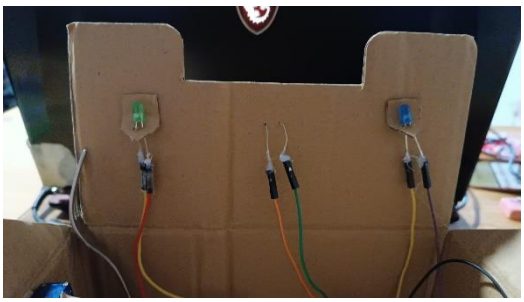
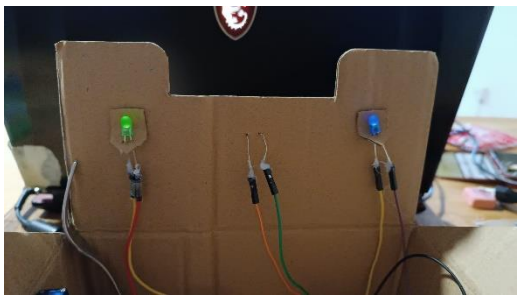
Finalmente, el código al completo quedaría así:

```
#include "main.h"

/* Private variables ---------------------------------------------------------*/
ADC_HandleTypeDef hadc1;

TIM_HandleTypeDef htim2;
TIM_HandleTypeDef htim3;
TIM_HandleTypeDef htim4;

/* USER CODE BEGIN PV */
uint32_t IC_Val1 = 0;
uint32_t IC_Val2 = 0;
uint32_t Difference = 0;
uint8_t Is_First_Captured = 0;  // primer valor
uint8_t Distance  = 0;
uint8_t Objeto=0;//flag usada para saber si ha detectado algo
uint8_t i=90;
uint32_t valor_ldr;
uint8_t luz=0; //Estado de la luz: 0 apagado, 1 encendido
uint32_t debouncer=100; //tiempo mínimo entre pulsaciones de botón
uint32_t tiempoactual;
uint32_t ultimapulsacion;

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM2_Init(void);
static void MX_TIM3_Init(void);
static void MX_TIM4_Init(void);
static void MX_ADC1_Init(void);

void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim);
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) //Interrupciones del botón
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc) //Interrupciones del conversor ADC
int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_TIM2_Init();
  MX_TIM3_Init();
  MX_TIM4_Init();
  MX_ADC1_Init();

  HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_1); //Iniciar y habilitar interrupciones
  HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_2); //iniciar pulso de trigger
  __HAL_TIM_SET_COMPARE(&htim3,TIM_CHANNEL_2,10); //Hace que el pulso de trigger dure 10us
  HAL_ADC_Start_IT(&hadc1);

  while (1)
  {

      if (Objeto==1){ //si detecta presencia abre la puerta

          HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_3); //enciende el motor
          htim4.Instance->CCR3=i;
          while(i>25){
                htim4.Instance->CCR3=i;
                i--;
                HAL_Delay(100);
            }
      }
      if (Objeto==0){ // Cuando no hay presencia cierra la puerta si esta abierta y si no apaga el motor

          if (i<90){
              while(i<90){
                  htim4.Instance->CCR3=i;
                  i++;
                  HAL_Delay(100);
                  }
              }
          else {
              HAL_TIM_PWM_Stop(&htim4, TIM_CHANNEL_3); //apaga motor

          }
      }

  }
```

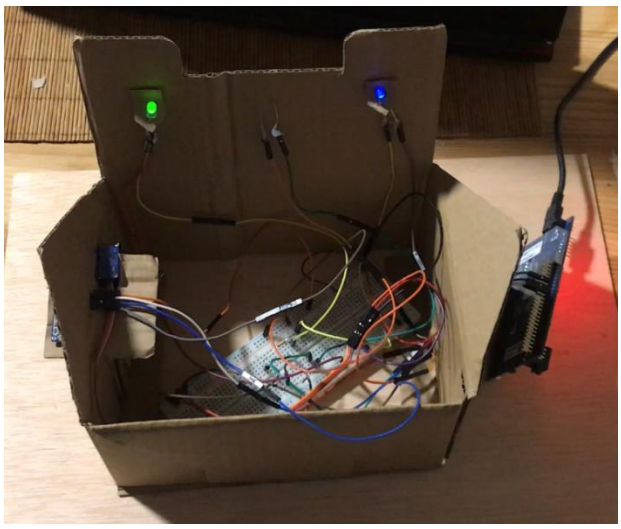
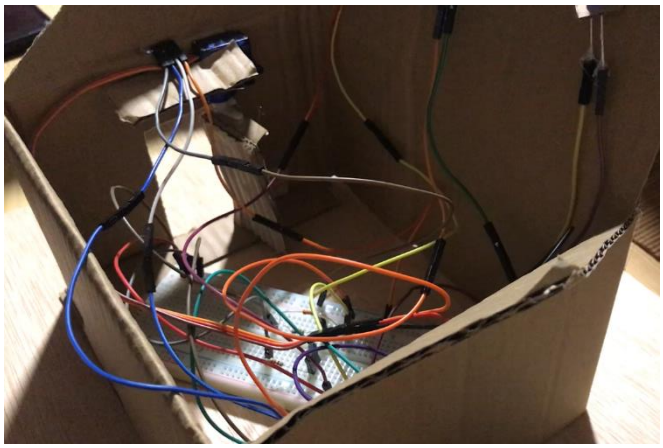
Montaje

A continuación, se muestran las fotos del montaje:



El montaje completo:

Video de funcionamiento

https://mega.nz/file/GcgFnaCI#TEZe-jjvHKsxZVudeoBnjcngpsLXhD1OhQsFpy46sXA