PROYECTO SISTEMAS ELECTRÓNICOS DIGITALES

MEMORIA TRABAJO MICROS

CONTROL DE COCINA

NATALIA BORLAF NIETO, 54516

JESÚS GARCÍA SÁNCHEZ, 54622

ISMAEL FERNÁNDEZ DE LA COTERA LORENZO, 54594

CURSO 2022-2023

Repositorio github:

https://github.com/nborlaf/TrabajoSED-Micros.git

ÍNDICE

Funcionamiento del proyecto

Con nuestro proyecto hemos querido realizar algunas de las funciones principales que tiene una cocina utilizando microcontroladores:

Interruptor "User" con LED externo: Representa el extractor de la cocina.

Con presionamos el botón User de la placa, se enciende el led rojo externo en la protoboard. Para realizar esto se han utilizado las interrupciones.

Detector de presencia con LED verde de la placa: Representa luz de la cocina.

Para esta parte hemos utilizado el sensor de ultrasonidos hc-sr04. Cuando detecta algo a menos de 10 cm se enciende la luz verde de la placa, que se mantendrá encendida mientras siga detectando algo. Cuando deja de detectar algo, la luz verde estará 10 segundos encendida y procederá a apagarse.

Detector de temperatura con LED rojo de la placa: Representa luz de emergencia.

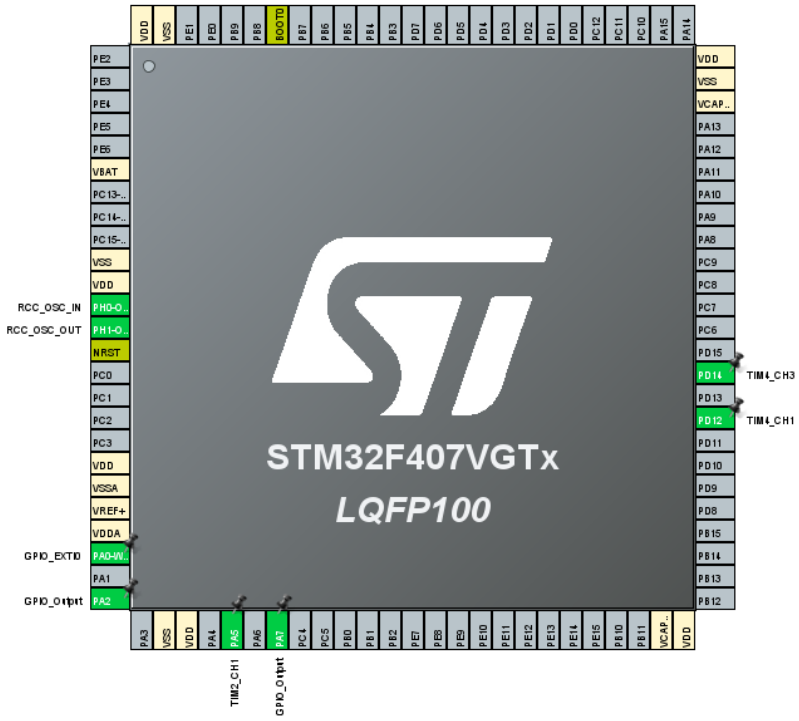
En esta última parte hemos utilizado el sensor de temperatura que viene incluido en la placa STM32. Para poder mostrar el funcionamiento se ha establecido, que un led rojo empezará a parpadear cuando se superen los 20ºC, ya que la temperatura media de una habitación suele estar en torno a los 19ºC. Aunque en una situación real habría que poner algún valor superior a los 30ºC.

Material

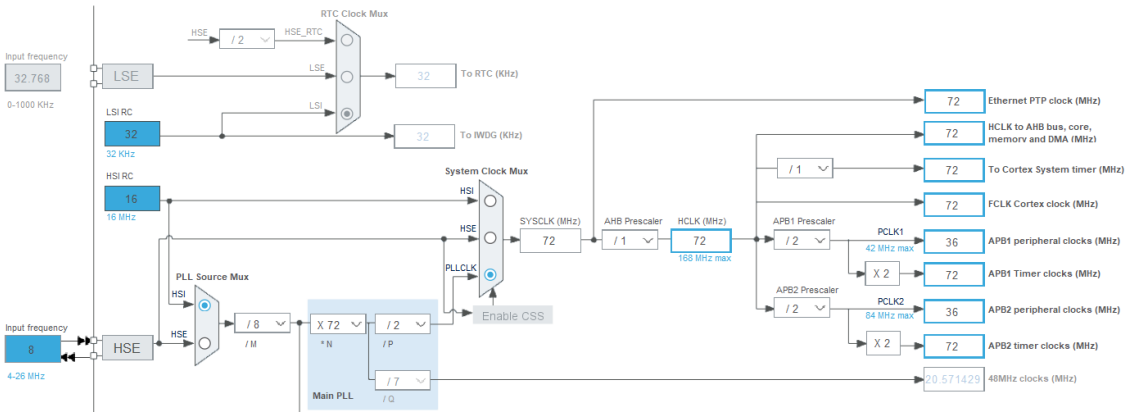
El material utilizado ha sido el siguiente:

- Placa STM32F407
- Protoboard
- Led rojo
- Cables
- Sensor de ultrasonidos hc-sr04

Configuración de pines



Configuración del reloj

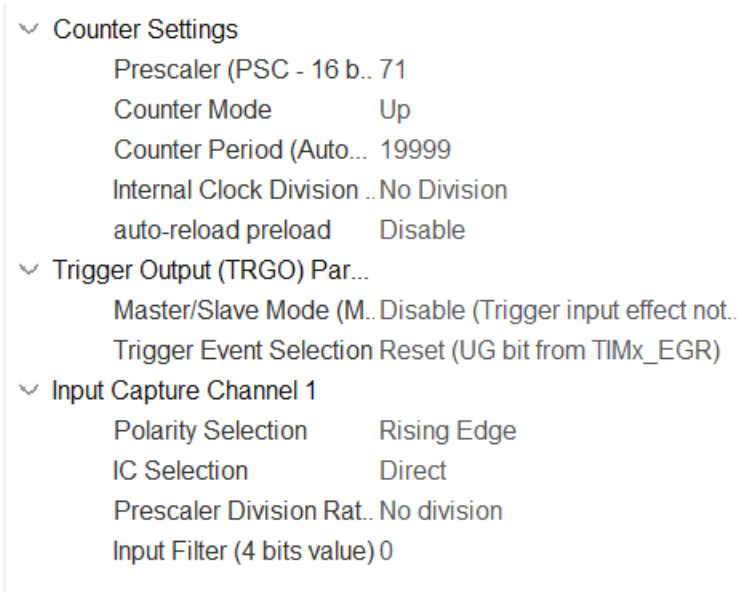


El HCLK le hemos configurado a 72 MHz, el cual será importante a la hora de configurar lo “TIMERS” utilizados a lo largo del proyecto.

TIM 2

Este TIMER se ha utilizado para la lectura y el correcto funcionamiento del HC-SR04. Por lo que se ha utilizado un *Input Capture Direct Mode*, ya que es capaz de detectar flancos en los pines de entrada y almacenar los valores en el contador, que es lo más ideal para el funcionamiento que buscamos.

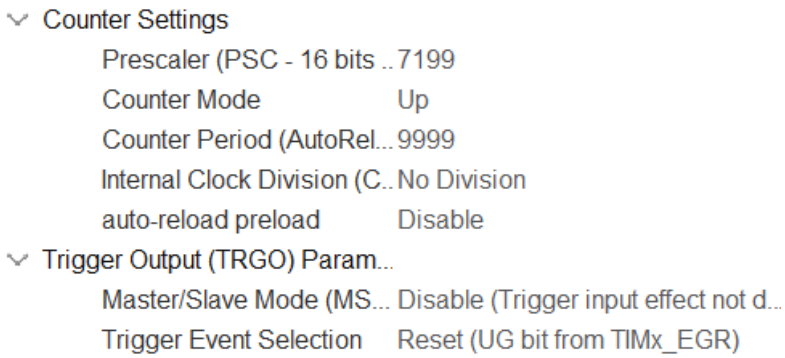
El TIMER lo hemos configurado de la siguiente manera, ya que buscamos que el sensor funcione a 50 Hz con un periodo de unos 20 ms.



TIM 4

Este TIMER se ha utilizado para 2 funciones distintas, en el canal 1 para establecer los 10 segundos que se tiene que mantener encendido el led verde una vez deja de detectar la presencia y en el canal 3 para hacer parpadear el led rojo una vez se superan los 20ºC.

En ambos casos se ha utilizado un *PWM*. Y el TIMER lo hemos configurado de la siguiente manera, ya que buscamos funcione a 1 Hz con un periodo de 1 ms.



- CH1

PWM Generation Channel 1	
Mode	PWM mode 1
Pulse (16 bits value)	0
Output compare preload	Enable
Fast Mode	Disable
CH Polarity	High

- CH3

PWM Generation Channel 3	
Mode	PWM mode 1
Pulse (16 bits value)	0
Output compare preload	Enable
Fast Mode	Disable
CH Polarity	High

Código en STM32CubeIDE

- Variables globales

```
/* Private user code ---------------------------
/* USER CODE BEGIN 0 */

//Variables globales del sensor ultrasonidos.
uint32_t IC_Val1 = 0;
uint32_t IC_Val2 = 0;
uint32_t Difference = 0;
uint8_t Is_First_Captured = 0;
uint8_t Distance  = 0;
int encendido=0;
uint32_t ten=0;

//Variables globales interrupción.
volatile int boton=0;

//Variables para la temperatura.
volatile uint32_t adcvalue;
float temp=0;
```

- Función antirrebotes

```
int debouncer(volatile int* button, GPIO_TypeDef* GPIO_port, uint16_t GPIO_number){
    static uint8_t button_count=0;
    static int cuenta=0;

    if (*button==1){
        if (button_count==0) {
            cuenta=HAL_GetTick();
            button_count++;
        }
        if (HAL_GetTick()-cuenta>=20){
            cuenta=HAL_GetTick();
            if (HAL_GPIO_ReadPin(GPIO_port, GPIO_number)!=1){
                button_count=1;
            }
            else{
                button_count++;
            }
            if (button_count==4){
                button_count=0;
                *button=0;
                return 1;
            }
        }

    }
    return 0;
}
```

- Funciones del ultrasonido

Lo primero que hacemos es crear una función para hacer el delay en la lectura del ultrasonidos:

```
//Necesario para la lectura del sensor ultrasonidos. Con HalDelay se detiene
//Función para hacer el delay en el ultrasonidos.
void delay (uint16_t time)
{
    __HAL_TIM_SET_COUNTER(&htim2, 0);
    while (__HAL_TIM_GET_COUNTER (&htim2) < time);
}
```

Y ya a continuación se crea la función para la lectura del ultrasonidos:

```
//Función de lectura del ultrasonidos
void HCSR04_Read (void)
{
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, GPIO_PIN_SET); //Se envia pulso a la patilla trig del ultrasonidos
    delay(5);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, GPIO_PIN_RESET);

    __HAL_TIM_ENABLE_IT(&htim2, TIM_IT_CC1);
}
```

Por último, se crea la función que nos da la distancia del objeto al sensor:

```
88 //Código del funcionamiento del HC-SR04.
89 void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim) {
90     if (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1)
91     {
92         if (Is_First_Captured==0)
93         {
94             IC_Val1 = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_1); //lee el primer valor
95             Is_First_Captured = 1;  // selecciona la primera captura
96             __HAL_TIM_SET_CAPTUREPOLARITY(htim, TIM_CHANNEL_1, TIM_INPUTCHANNELPOLARITY_FALLING);
97         }
98
99         else if (Is_First_Captured==1)   // si ha leido el primer valor
100        {
101            IC_Val2 = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_1);  // lee el segundo valor
102            __HAL_TIM_SET_COUNTER(htim, 0);  // reset the counter
103
104            if (IC_Val2 > IC_Val1)
105            {
106                Difference = IC_Val2-IC_Val1;
107            }
108
109            else if (IC_Val1 > IC_Val2)
110            {
111                Difference = (0xffff - IC_Val1) + IC_Val2;
112            }
113
114            Distance = Difference * .034/2;//conversion a centimetros
115            Is_First_Captured = 0;
116            __HAL_TIM_SET_CAPTUREPOLARITY(htim, TIM_CHANNEL_1, TIM_INPUTCHANNELPOLARITY_RISING);
117            __HAL_TIM_DISABLE_IT(&htim2, TIM_IT_CC1);
118        }
119    }
120 }
```

- Función para la interrupción

```
132 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){
133
134     if (GPIO_Pin==GPIO_PIN_0)
135     {
136         boton=1;
137     }
138 }
```

- Funcionamiento (while)

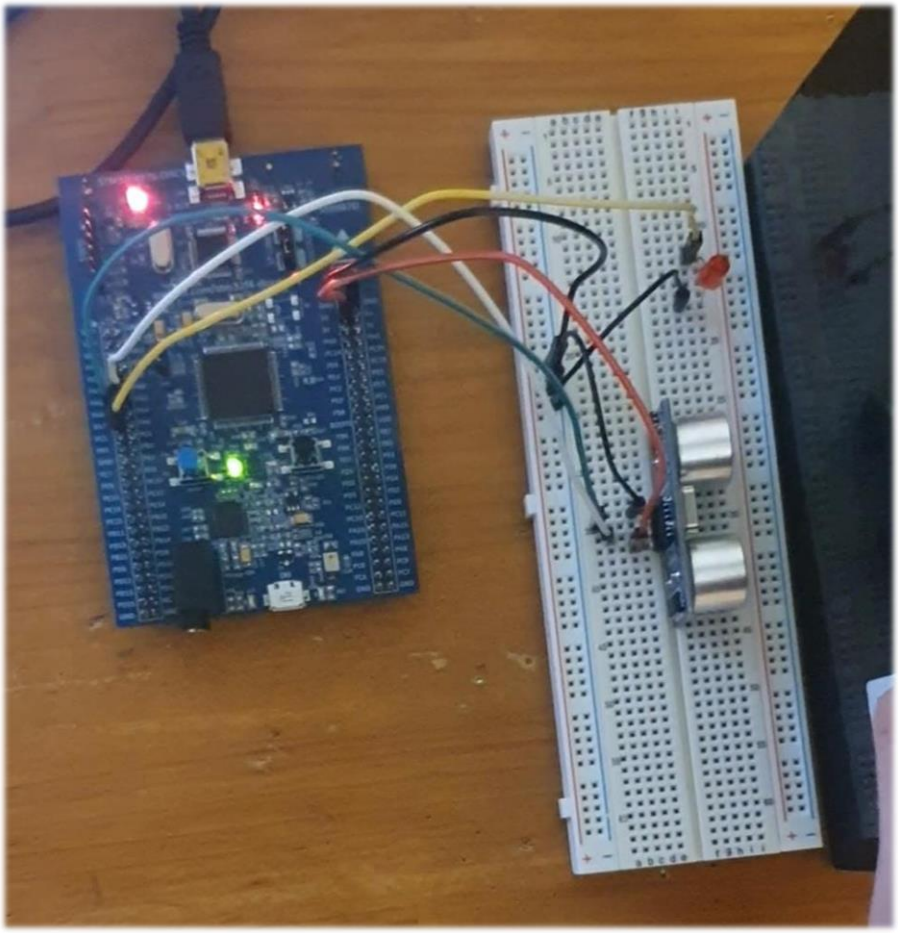
```
209   while (1)
210   {
211     /* USER CODE END WHILE */
212
213     /* USER CODE BEGIN 3 */
214       HCSR04_Read(); //Lee el valor del ultrasonidos
215       HAL_Delay(100);
216
217       if(Distance<10)
218       {
219           ten=HAL_GetTick();
220             __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_1, 9999);
221           //encendido=1;
222       }
223       else if((HAL_GetTick()-ten)>10000){
224            __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_1, 0);
225       }
226       if (debouncer(&boton, GPIOA, GPIO_PIN_0)){
227           encendido=1;
228           HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_7);
229       }
230       HAL_ADC_Start(&hadc1);
231       if(HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY) == HAL_OK)
232       {
233           adcvalue=HAL_ADC_GetValue(&hadc1);
234           //La temperatura la hemos convertido de Fahrenheit a Celsius.
235           temp=(((float)adcvalue)-32)/1.8;
236       }
237       //Lo hemos puesto a 20 para no subir demasiado la temperatura y no dañar la placa.
238           if(temp>20)
239           __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_3, 6666);
240           else
241           __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_3, 0);
242
243   }
244   /* USER CODE END 3 */
```

Dentro del while lo único que se realiza es las llamadas a las funciones de lectura tanto del sensor de distancia como del de temperatura. En el caso del sensor distancia, si se detecta presencia a menos de 10cm se encenderá el led correspondiente durante 10 segundos. Si nos fijamos en el sensor de temperatura, vemos que el led correspondiente comenzara a parpadear cuando la temperatura supere los 20 grasos.

Además, también se llama a la función debouncer desde el while. Esto encenderá el led correspondiente cuando se pulse el botón y el programa se asegure de que es una pulsación real.

Montaje

Para el correcto funcionamiento se ha tenido que realizar un montaje ayudándonos de una protoboard.



Se han utilizado, además de los recursos que nos ofrece la placa, un sensor de ultrasonido hc-sr04 y un Led de color rojo.