

SISTEMAS ELECTRÓNICOS DIGITALES



TRABAJO MICROS CONTROL DE PERSIANAS

GRUPO 1

Profesor: Pedro Luis Castedo Cepeda

Participantes:

Mario Sánchez García (55452)

Felipe de Gracia Ruiz (55199)

Adrián Juan Drag (55219)



ÍNDICE

INTRODUCCIÓN	3
COMPONENTES Y MÉTODOS	3
FUNCIONAMIENTO PREVISTO	3
PLANTEAMIENTO INICIAL	4
DISTRIBUCIÓN .IOC	4
DESARROLLO DEL PROYECTO	5
CLOCK CONFIGURATION	5
PRIMER MODO: SERVOMOTOR TEMP + BOTÓN CON IT	6
MODOS CON SENSOR DE PRESENCIA Y LUMINOSIDAD	8
COMUNICACIÓN SERIE UART	9
COMUNICACIÓN I2C DISPLAY	12
PROBLEMAS	12
REPOSITORIO DE GITHUB	13
CONCLUSION	13

INTRODUCCIÓN

El proyecto seleccionado para el trabajo de microcontroladores de la asignatura ha sido una aplicación de domótica basada en el control de una persiana y sistema de alumbrado a través de determinados modos mediante sensores. Se ha implementado en el microcontrolador STM32F411 visto en la asignatura. Además, se ha recreado en una maqueta real con distintos sensores, leds y un servomotor para una mejor comprensión de su funcionamiento.

La inspiración del trabajo realizado se basa en el control de sistemas utilizados por el ser humano día a día como son las propias luces o las persianas del hogar. La domótica es conocida como la agrupación de una serie de técnicas orientadas a la automatización de una vivienda gracias a la integración de la tecnología en ella.

COMPONENTES Y MÉTODOS

- Microcontrolador STM32F411
- Servomotor
- Sensor de presencia
- Sensor de iluminación LDR
- Comunicación serie UART con adaptador USB a TTL
- Display mediante comunicación serie I2C
- Luces LED
- Botón del propio microcontrolador

FUNCIONAMIENTO PREVISTO

Existe una continua interacción con la consola del programa mediante comunicación serie UART. La consola pedirá introducir un modo cada vez que sea conveniente, de forma que introduciendo uno de los tres modos configurados se active ese funcionamiento específico. Existen tres modos:

- MODO1: Control de un servomotor mediante un botón.
- MODO2: Control de luz LED mediante un sensor de presencia.
- MODO2: Control de luz LED mediante sensor de luminosidad.

Además, existe un display en el que se muestran determinados mensajes en función del desarrollo del programa.

- RCC
- PA13,PA14: Systick
- PB6,PB7: SCL y SDA de la comunicación serie I2C.

DESARROLLO DEL PROYECTO

CLOCK CONFIGURATION

Lo primero de todo fue configurar el reloj ya que se tienen elementos como el servomotor o la comunicación serie I2C que dependiendo de la frecuencia que le llegue funciona de una forma u otra. Para ello se tuvo que activar el reloj externo ya que la pantalla que da salida el I2C necesitaba dicho reloj.

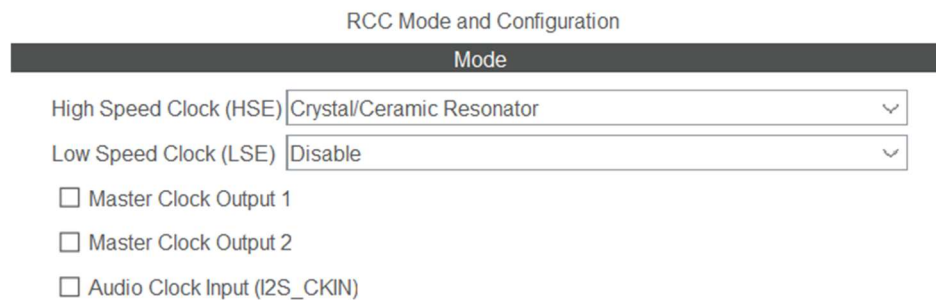


Ilustración 2: Reloj externo

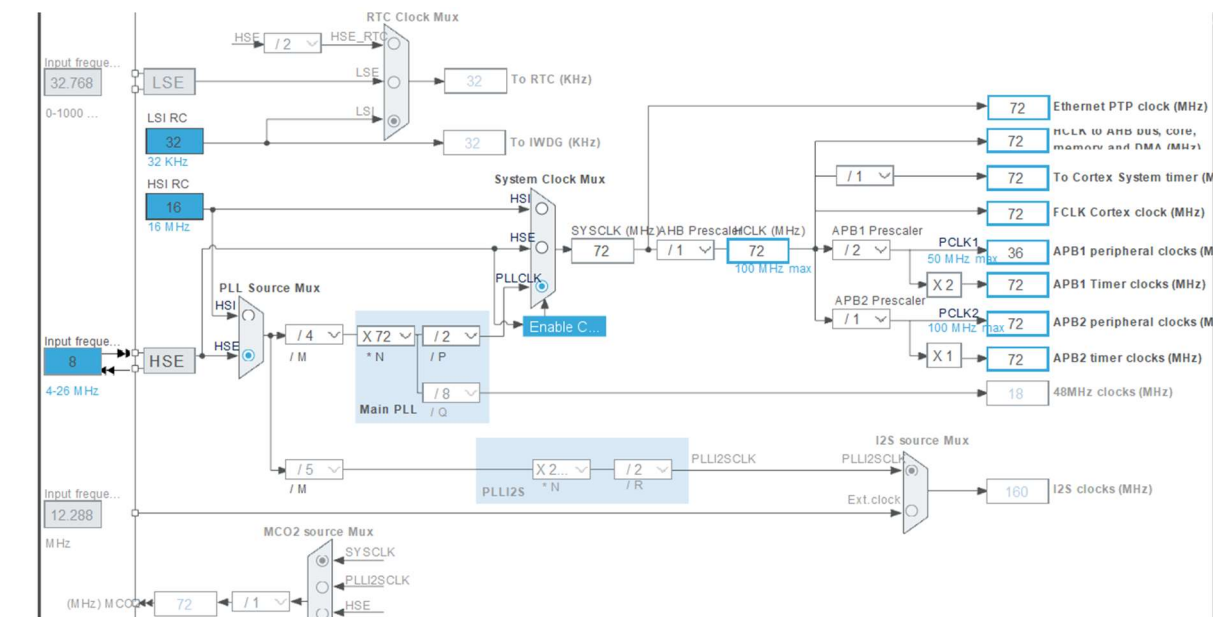


Ilustración 3: Configuración Reloj

Gracias a esta configuración se ha conseguido que el display funcionara y que el servomotor no tuviera problemas para moverse.

PRIMER MODO: SERVOMOTOR TEMP + BOTÓN CON IT

Este modo consiste en hacer girar el servomotor configurado con un temporizador PWM de un extremo a otro mediante un botón con interrupción que provoque que el flag se ponga a 1. Esto simulará el control de la persiana introducido en este proyecto.

Para ello fue necesario modificar el Prescaler y su contador ya que dicho servomotor trabaja a una frecuencia de 50 Hz. Se realizaron los cálculos pertinentes y sabiendo que el reloj interno que le llega es el APB1 (72 MHz) se obtiene dicha frecuencia de funcionamiento.

$$Update\ evente = \frac{APB1(frec)}{Prescaler * counter}$$

$$50Hz = \frac{72MHz}{720 * 2000}$$

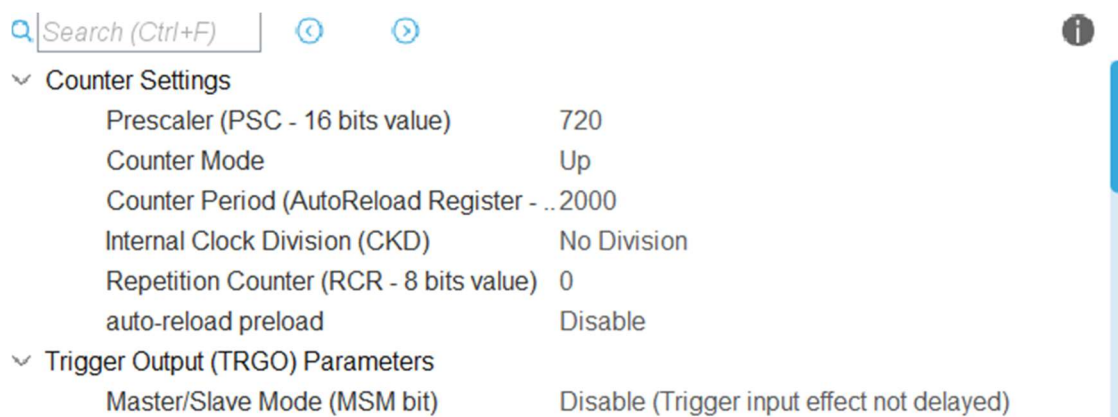


Ilustración 4: configuración parámetros TIM1

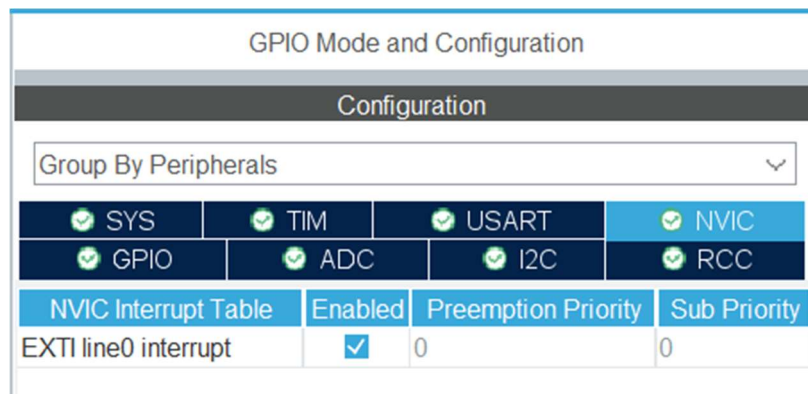


Ilustración 5: Configuración GPIO input

```

75 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){
76     if (GPIO_Pin==GPIO_PIN_0)
77     {
78         char texto3[30]="Has elegido el modo boton\r\n";
79         HAL_UART_Transmit(&huart2, (uint8_t *)texto3, 30,HAL_MAX_DELAY);
80
81         flag=1;
82     }
83 }
84

```

Ilustración 6: Código Callback GPIO

Una vez ha saltado la interrupción la variable declarada como flag pasa a valer 1 por lo que el código dentro del while se ejecuta, es decir, se mueve el servomotor según ha sido configurado.

El código de este primer modo se desarrolla en el while y se puede ver a continuación:

```

219 while (1)
220 {
221     /* USER CODE END WHILE */
222
223     /* USER CODE BEGIN 3 */
224     HAL_ADC_Start_DMA(&hadc1, adcbuffer, 2);
225     if(flag==1)
226     {
227         HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15,0);
228         HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14,0);
229         j=0;
230         if(contador%2==0)
231         {
232             for(int i=0;i<90;i++)
233             {
234                 HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_1,i);
235                 HAL_Delay(10);
236                 j++;
237             }
238
239             HAL_UART_Transmit_IT(&huart2,(uint8_t *)textol, 30);
240             flag=0;
241             contador++;
242
243         }
244     }
245     else
246     {
247         for(int i=90;i>0;i--)
248         {
249             HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_1,i);
250             HAL_Delay(10);
251             j++;
252         }
253         HAL_UART_Transmit_IT(&huart2,(uint8_t *)textol, 30);
254         flag=0;
255         contador++;
256     }

```

Ilustración 7: Código movimiento servomotor

MODOS CON SENSOR DE PRESENCIA Y LUMINOSIDAD

Al tener dos sensores, el de luminosidad y el de presencia, se ha utilizado DMA ya que la placa STM32F411 solo tiene un convertidor (ADC1). Al utilizar DMA se ha ahorrado código ya que con este método no hay preocupación alguna por la transferencia de datos ya que lo hace el propio DMA. Es una técnica muy eficiente.

```
HAL_ADC_Start_DMA(&hadc1, adcbuffer, 2);
```


DMA Mode and Configuration

Configuration			
<div style="display: flex; justify-content: space-around;"> ✓ DMA1 ✓ DMA2 ✓ MemToMem </div>			
DMA Request	Stream	Direction	
ADC1	DMA2 Stream 0	Peripheral To Memory	Low

Add
Delete

DMA Request Settings

Mode	<div>Circular</div>	Increment Address	<input type="checkbox"/>
Use Fifo	<input type="checkbox"/>	Threshold	<div></div>
Data Width	<div>Word</div>	Burst Size	<div></div>

Ilustración 8: configuración DMA

Para la configuración del ADC, al tener configurado ya el DMA, en el número de conversiones se ponen las utilizadas en el proyecto, en este caso dos. Al tener dos sensores el DMA va a depositar los datos convertidos en diferentes canales. Se crea un buffer en el que almacene ambas medidas de los sensores para poder trabajar con ellos en cada modo.

ADC_Regular_ConversionMode
2

Number Of Conversion
2

External Trigger Conversion Source
Regular Conversion launched by software

External Trigger Conversion Edge
None

Rank
1

Channel
Channel 4

Sampling Time
3 Cycles

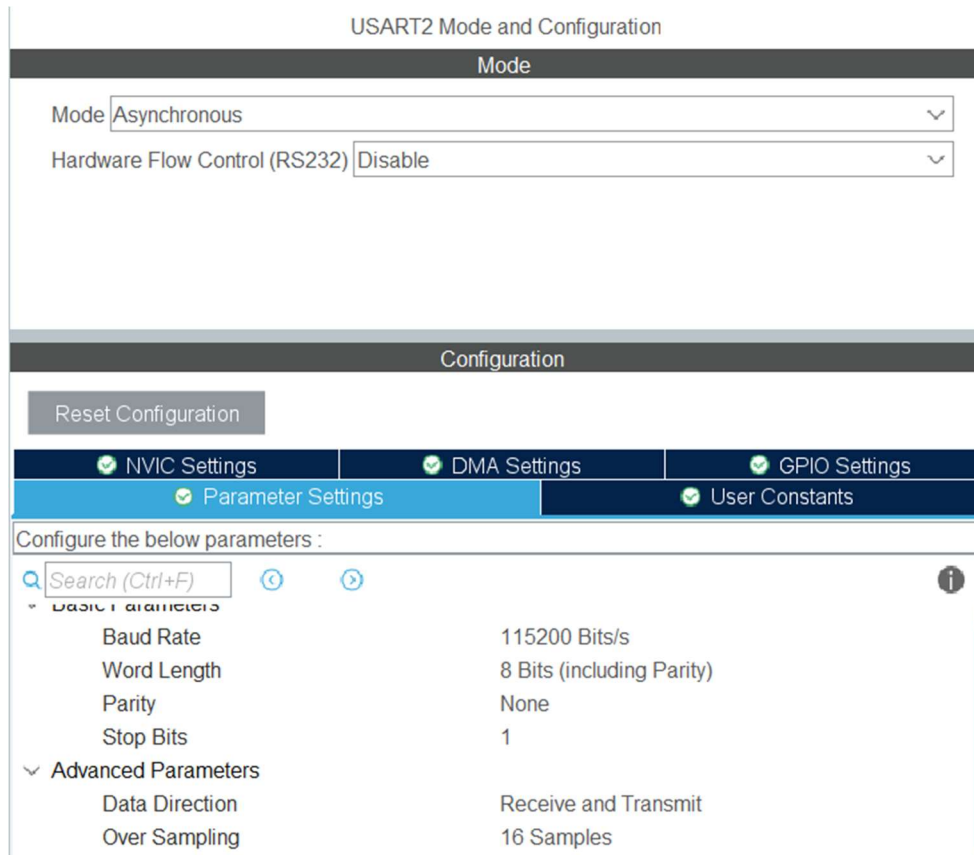
Rank
2

Channel
Channel 1

Ilustración 9: Configuración ADC

COMUNICACIÓN SERIE UART

Gracias a la comunicación serie asíncrona UART, se consigue que en el programa desarrollado haya una interacción constante entre el usuario (persona) y el microcontrolador. Para ello, se tiene que configurar los mismos parámetros tanto en el puerto serie como en el microcontrolador.



USART2 Mode and Configuration	
Mode	
Mode	Asynchronous
Hardware Flow Control (RS232)	Disable
Configuration	
Reset Configuration	
NVIC Settings DMA Settings GPIO Settings	
Parameter Settings User Constants	
Configure the below parameters :	
Search (Ctrl+F)	
Basic Parameters	
Baud Rate	115200 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1
Advanced Parameters	
Data Direction	Receive and Transmit
Over Sampling	16 Samples

Ilustración 10: configuración UART

La consola configurada es la siguiente:

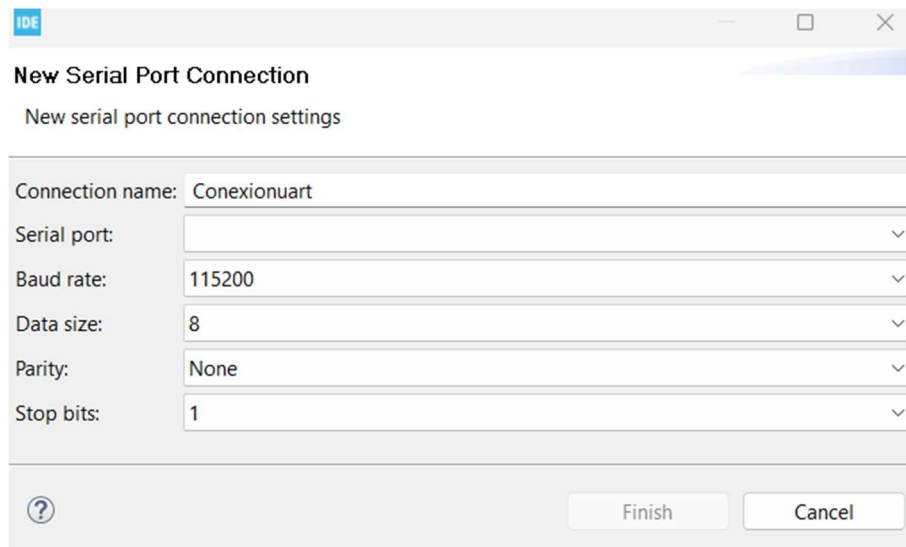


Ilustración 11: Configuración puerto serie

La comunicación UART se ha realizado mediante interrupciones y así se ha conseguido poder controlar cuando se quiera recibir o transmitir mensajes con más seguridad en comparación con el caso de si estar ubicado dentro del while. Además, gracias a la callback de recibir se ha conseguido realizar la funcionalidad de la logística a la perfección.

Se ha conseguido controlar todo el programa creado mediante esta comunicación serie.

```

86 void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart) {
87     char texto2[30]="Buenas, seleccione modo\r\n";
88
89     HAL_UART_Transmit(huart, (uint8_t *)texto2, 30,HAL_MAX_DELAY);
90
91 }
92 }

```

Ilustración 12: Función Callback transmitir

```

93 void HAL_UART_RxCpltCallback(UART_HandleTypeDef *UartHandle) {
94     /* Se recibe el caracter y se pide el siguiente*/
95
96     HAL_UART_Receive_IT(&huart2, (uint8_t *)recibido, 1);
97     if(UartHandle->Instance == USART2){
98         modo = recibido[0];
99         switch(modo){
100             //if( modo=='1')
101
102
103             case '1':
104                 {
105                     HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14,0);
106                     if((adcbuffer[0]<300) ){
107                         //va de 527 a 1023 mas o menos el sensor
108                         char texto4[50]="Has elegido el sensor de LUMINOSIDAD\r\n";
109                         HAL_UART_Transmit(&huart2, (uint8_t *)texto4, 50,HAL_MAX_DELAY);
110                         HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15,0);
111
112                         modo ='0';
113                         HAL_UART_Transmit_IT(&huart2,(uint8_t *)texto1, 30);
114

```

Ilustración 13: Función Callback Recibir

COMUNICACIÓN I2C DISPLAY

La comunicación serie I2C fue utilizada para poder hacer uso de la pantalla lcd, mediante el uso de un módulo adaptador que se trata de un expensor de entradas y salidas. A través de este módulo se pudo variar el contraste de la pantalla mediante un potenciómetro azul que tiene incorporado.

Para poder hacer uso de este conjunto de LCD e I2C se tuvo que añadir la librería correspondiente con sus funciones ya programadas (i2c-lcd.h e i2c-lcd.c).

Estas funciones son:

- void lcd_init(void): Función de inicialización
- void lcd_clear(void) : Limpia todos los caracteres que haya en ese momento en la pantalla y posiciona el cursor en la esquina superior izquierda.
- void lcd_put_cur(int row, int col): Posiciona el cursor en la línea y columna deseados.
- void lcd_send_string(char *str): Esta función imprime por la pantalla el texto que se le pase.

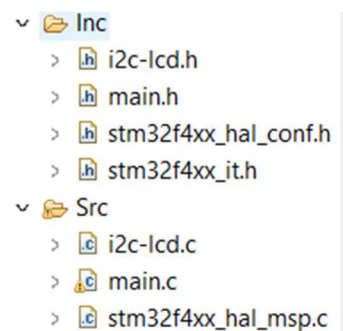
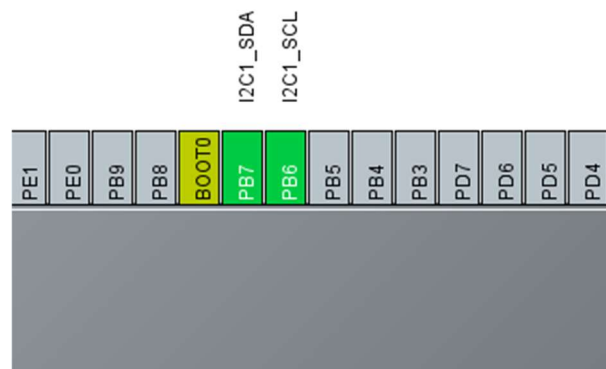


Ilustración 14: Librería y pines I2C

PROBLEMAS

Se han tenido varios tipos de problemas y contratiempos a la hora de llevar a cabo el proyecto descrito.

En primer lugar, se tuvieron problemas con la lectura sin interrupción de los valores de los sensores mediante DMA. El mayor detonante fue que realizamos distintas interrupciones en las que se involucraba y solo leía valores de los sensores en el instante de la interrupción. Este ha sido el único problema que ha sido solventando en menor medida.

En segundo lugar, la comunicación serie UART dio bastante trabajo a realizar ya que hasta no tener un conocimiento suficiente de su funcionamiento y de la consola de comunicación del programa, no fue posible su funcionamiento correcto. Finalmente, se ha realizado la comunicación serie UART mediante interrupciones, solucionando así, todos los inconvenientes que no estaba aportando y dando al proyecto un toque sofisticado.

Por otro lado, la interrupción del botón tuvo que ser probada de distintos modos pero finalmente se ha conseguido su correcto funcionamiento.

La comunicación serie I2C para el display fue bastante tediosa debido a la necesidad que tenía de limpiar y posicionar de nuevo los mensajes mostrados. Finalmente, se solucionó a pesar de darle un uso escaso en el desarrollo de la ejecución del programa.

Por último, la configuración del reloj fue modificada varias veces por la frecuencia necesaria del servomotor y por el RCC necesario para el display. Se entendió su funcionamiento correctamente y esta es la prueba de como funciona el proyecto.

REPOSITORIO DE GITHUB

<https://github.com/mariooot13/Trabajo-MICROS-G1>

CONCLUSIÓN

Tras llevar a cabo el proyecto entero con la integración de todos los modos y comunicaciones que contiene se puede observar que su ejecución ha sido exitosa y eficiente.

El trabajo realizado simula a la perfección la aplicación de domótica como objetivo principal de ello.

Además, se han adquirido una serie de conocimientos sobre todos los conceptos vistos en la asignatura con una gran profundidad en cada uno de ellos ya que se han utilizado técnicas como DMA para los sensores, interrupciones para las comunicaciones serie y un código estructurado de todo el programa.

Por último, el resultado ha sido el esperado y ha sido un proyecto con un gran valor para cada uno de los integrantes del proyecto.