

INGENIERÍA DE SISTEMAS ELECTRÓNICOS

Curso 2023-2024. BLOQUE 2

Título proyecto: The Snitch

Estudiantes:

Luis González-Gallego Sosa

Pablo Martínez Lafarga

Sara Jiménez Muñoz

Luis López Izquierdo



Contenido

1	ESPECIFICACIONES.	3
1.1	Especificaciones iniciales del sistema a diseñar y construir.	3
1.2	Especificaciones finales del sistema diseñado y construido.	4
2	DESARROLLO DE SUBSISTEMAS.	6
2.1	Analógicos.	6
2.2	ALIMENTACIÓN.	10
	ESQUEMA ELECTRÓNICO COMPLETO.	16
3	SOFTWARE.	20
3.1	Interfaz de usuario.	20
3.2	Descripción de cada uno de los módulos del sistema.	20
3.2.1	Buzzer.	20
3.2.2	FT 21	
3.2.3	Keypad	22
3.2.4	LCD24	
	• Descripción:	25
	• Configuración:	25
	• Entradas y Salidas:	25
	• Funciones Principales:	25
	• Paso por Paso:	25
	• Descripción:	26
	• Funciones Principales:	26
3.2.5	NFC	26
	• Descripción de los Registros del MFRC522 utilizados en nuestro programa:	26
	• Funciones de Bajo Nivel de SPI	27
	• Funciones de Comunicación y Control del MFRC522	27
	• Funciones de Inicialización y Configuración	27
	• Funciones de Gestión de Hilos	28
	• Funciones de Inicialización del Sistema	28
	• Gestión de Interrupciones y Temporizadores	28
	• Configuración de Hardware y Entorno de Ejecución	28
	• Funciones Principales del Sistema	28
	• Funcionamiento Paso a Paso:	28
3.2.6	RGB.	30
	• Entradas y Salidas:	30
	Funciones Principales:	31
	Paso por Paso:	31
3.2.7	RTC31	
3.3	Pruebas realizadas software.	36
3.4	Pruebas realizadas hardware.	37
4	CONCLUSIONES.	39
5	PRESUPUESTO FINAL.	40

6	EQUIPO DE TRABAJO.....	41
7	ACRÓNIMOS UTILIZADOS.	42
8	BIBLIOGRAFÍA UTILIZADA.	43

Tabla de ilustraciones

Figura 1. Diagrama de bloques	5
Figura 2. Detector de conmutación	6
Figura 3. Catalogo del PIR	8
Figura 4. Medidor de consumo	8
Figura 5. Alimentación externa	9
Figura 6. Circuito de alimentación	10
Figura 7. Circuito battery back-up	10
Figura 8. Datasheet del BMS	11
Figura 9. Datasheet regulador LM78XX	12
Figura 10. Circuio alimentación simétrica	13
Figura 11. Integrado ICL7660	14
Figura 12. Datasheet ICL7660	15
Figura 13. Sistema completo	16
Figura 14. Circuito de conexión	16
Figura 15. Diseño PCB	17
Figura 16. Esquema de pines	18
Figura 17. Diseño caja final	19
Figura 18. Representación final hardware	19
Figura 17. Buzzer	20
Figura 18. MSGQUEUE_OBJ_BUZ	20
Figura 19. Configuración Buzzer	21
Figura 20. MSG TTF	22
Figura 21. Configuración TTF	22
Figura 22. Pines Keypad	23
Figura 23. Configuración pines GPIO	24
Figura 24 25. LCD	24
Figura 24. NFC	26
Figura 26 RGB	30
Figura 27. configuracion RTE_Device.h	32
Figura 28. PHY LAN8742A-CZ-TR	32
Figura 29 . RTC Network	33
Figura 30. Visualización Servidor	34
Figura 31. Configuración Net_Config_ETH_0.h	35

1 ESPECIFICACIONES.

1.1 Especificaciones iniciales del sistema a diseñar y construir.

Este proyecto se centra en garantizar la seguridad y el control eficiente del personal en entornos de alto riesgo mediante la implementación de un sistema de autenticación en dos pasos. Cada usuario contará con una tarjeta de identificación, que será leída mediante un lector de tarjetas RFID, además se le solicitará al usuario que introduzca, mediante un teclado numérico, una contraseña personal única, consiguiendo así, una autenticación sólida.

Otro aspecto de seguridad extra, que se planteó en un inicio, es el registro mediante cámara. Cada vez que un usuario intente acceder a la instalación, sea un acceso autorizado o no, se le realizará una fotografía que será almacenada junto con la hora de acceso. Adicionalmente, cualquier persona que se acerque al acceso, intente o no autenticarse, será fotografiado, evitando de esta manera que se manipule el dispositivo sin autorización. Toda la información referente al registro, que sea recogida por la aplicación, será mostrada en el servidor web.

Como medida extra de seguridad, el dispositivo de acceso podrá seguir funcionando cuando se vaya la red eléctrica, alimentado a través de baterías. De esta manera se pueden diferenciar dos modos diferentes de funcionamiento: modo normal y modo de emergencia.

- **Modo normal:** la aplicación estará conectado a la red eléctrica. La aplicación funcionará como se ha descrito previamente, es decir el servidor web estará funcionando de manera interactiva y los diferentes sensores estarán activos. Se pasará del modo normal al modo de emergencia cuando se detecte una caída de alimentación en la red eléctrica.
- **Modo emergencia:** la aplicación estará alimentada mediante baterías. La aplicación podrá entrar en modo de bajo consumo para alargar el tiempo de duración de la batería. En este modo se reducirá el consumo del dispositivo al máximo posible. Se saldrá del modo de emergencia cuando se detecte alimentación a través de la red eléctrica o cuando se detecte movimiento a través de un sensor de presencia.

1.2 Especificaciones finales del sistema diseñado y construido.

La funcionalidad principal del proyecto sigue siendo la misma que se definió anteriormente, con algunas excepciones importantes:

- La cámara no se ha implementado en el sistema debido a la complejidad en la programación y a la falta de tiempo necesaria para implementarla.
- Los componentes finales (microprocesador y sensores) se han cableado directamente a la STM debido a un error de diseño en los conectores de la PCB.
- Se ha modificado el circuito de alimentación simétrica para el amplificador de instrumentación del medidor de consumo respecto al diseño original. En el diseño original, se utilizaba un circuito con un amplificador operacional para generar una masa virtual y proporcionar la alimentación simétrica. Sin embargo, nos dimos cuenta de que esta masa virtual no era adecuada para conectar todos los sensores del sistema junto con la STM, ya que no representa una masa real.

Para resolver este problema, decidimos incorporar un integrado adicional que genera la alimentación negativa para el medidor de consumo, utilizando la misma referencia de masa de la alimentación principal.

Nuestro enfoque inicial de utilizar una masa virtual para la alimentación simétrica resultó en el diseño de un circuito con un voltaje mayor al necesario para nuestro sistema. Originalmente, requeríamos al menos 10V en la entrada del circuito de alimentación simétrica para generar aproximadamente $\pm 5V$ y así alimentar los amplificadores operacionales del amplificador de instrumentación. Además, enfrentamos el desafío de emplear amplificadores operacionales tipo rail-to-rail, ya que estos necesitaban ser alimentados con $\pm 5V$ mientras que la entrada era de 5V.

Diagrama de bloques final del sistema:

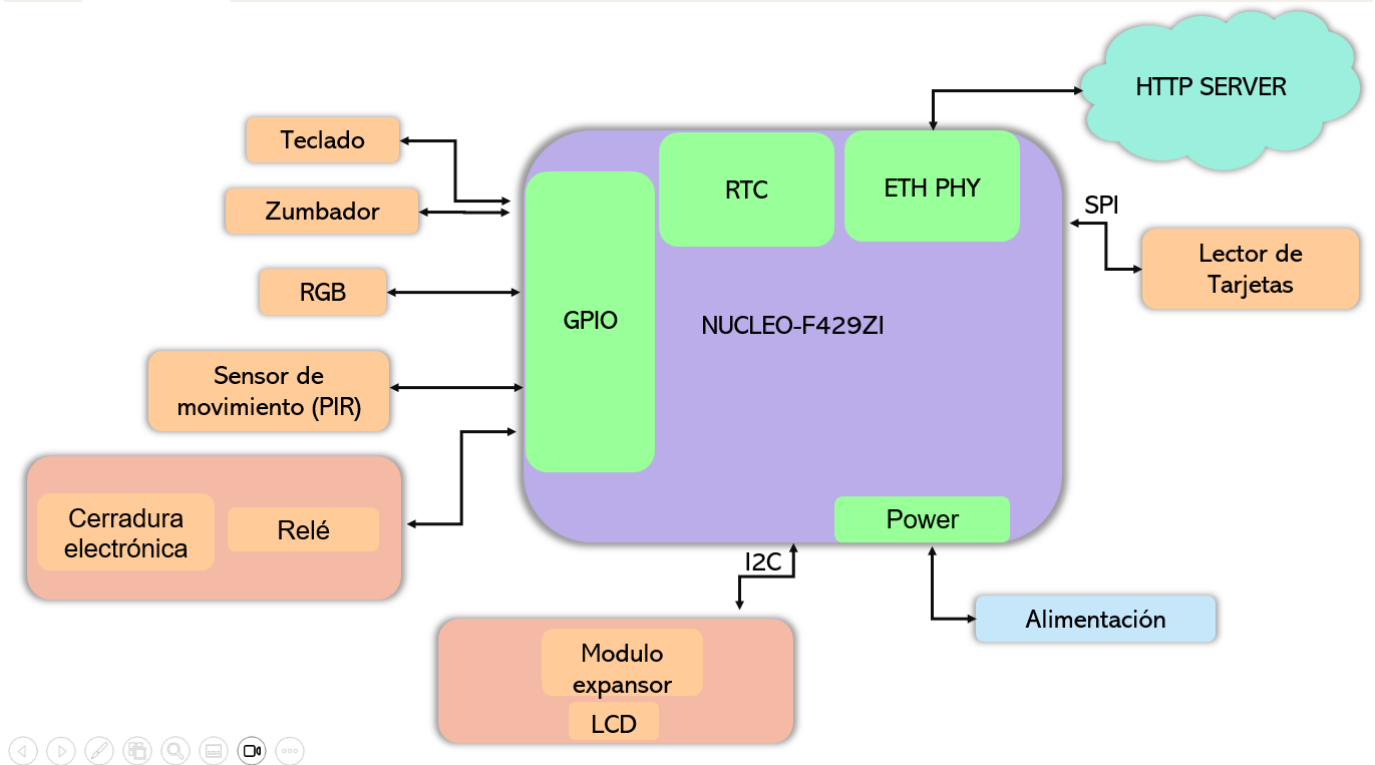


Figura 1. Diagrama de bloques

2 DESARROLLO DE SUBSISTEMAS.

2.1 Analógicos.

Circuito detección de conmutación:

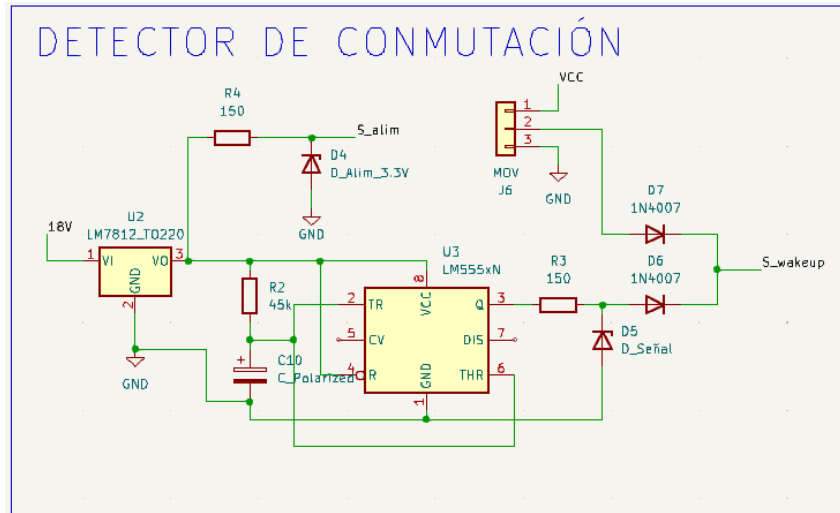


Figura 2. Detector de conmutación

Objetivo:

Este circuito es el encargado de enviar la señal de wakeup a la STM para que despierte el micro del modo de bajo consumo y de activar la funcionalidad de registro de acceso de personal cuando se detecte movimiento. Por otro lado, también tiene la funcionalidad de despertar al micro si se detecta que ha vuelto la red eléctrica.

Componentes:

- Integrado 555NE.
- Regulador LM7812
- Diodos zener BZX85C3V3
- Resistencia de 45k (para controlar el ancho del pulso de la señal de wakeup)
- Condensador de 10μF (para controlar el ancho del pulso de la señal de wakeup)

Funcionalidad:

Para habilitar funcionalidades específicas en nuestro sistema (salir del modo de bajo consumo al alimentarse el sistema desde la batería y detectar la presencia de una persona que se quiere autenticar o realizar un registro de acceso cuando se detecta la presencia de una persona y el sistema se alimenta a través de la red eléctrica), necesitábamos integrar dos métodos para activar el microcontrolador, conectados al mismo pin de la STM (ya que tiene solo un pin de activación). Uno de ellos se activa mediante la señal generada por un sensor de movimiento, mientras que el otro utiliza un circuito integrado 555 configurado en modo monoestable para generar pulsos de duración específica.

Además, requeríamos una señal para determinar si el circuito está siendo alimentado por la red eléctrica o por una batería, y detectar la presencia de una persona a través del sensor de movimiento. Con nuestro sistema, es esencial diferenciar si detectamos a alguien en modo de bajo consumo o en modo normal, dependiendo de cómo esté siendo alimentado el circuito y si se detecta movimiento.

El regulador colocado a entrada LM7812 nos reduce la tensión a la entrada del integrado a 12V ya que, según el datasheet, su tensión máxima a la entrada es de:

Absolute Maximum Ratings (TA = 25°C)

Parameter	Symbol	Value	Unit
Supply Voltage	VCC	16	V
Lead Temperature (Soldering 10sec)	TLEAD	300	°C
Power Dissipation	PD	600	mW
Operating Temperature Range LM555/NE555 SA555	TOPR	0 ~ +70 -40 ~ +85	°C
Storage Temperature Range	TSTG	-65 ~ +150	°C

Por otro lado, hemos colocado un diodo zener de 3.3V/1W antes de introducir estas señales al micro para regular la tensión a este valor, que es el cual este interpreta que tenemos un nivel alto.

Cálculos:

- **Periodo del pulso de salida del integrado NE555:**

Fórmula para el cálculo del periodo:

$$T = 1.1 \cdot R2 \cdot C10$$

Como queremos generar pulsos de una duración de 500ms, podemos fijar el valor del condensador a 10uF para obtener el valor de la resistencia de la siguiente manera:

$$R2 = \frac{T}{1.1 \cdot C10} = \frac{500 \cdot 10^{-3}}{1.1 \cdot 10 \cdot 10^{-6}} = 45.45kOhms$$

Escogeremos un valor de resistencia de 44.2Kohms que es el valor comercial más cercano.

- **Cálculos del diodo zener D5:**

Resistencia en serie mínima:

$$Rz \min \square = \frac{Vzener \cdot (Vin - Vzener)}{Pzener} = \frac{3.3V(12V - 3.3V)}{1W} = 28.710hms$$

Hemos escogido un valor de 150 ohms que es más que suficiente.

Tensión de entrada mínima de funcionamiento del zener:

$$Vzin \min \square = \frac{(RL + R) \cdot Vzener}{RL} = \frac{(132 + 150)3.3V}{132} = 7.05V$$

Siendo RL la resistencia de carga que, sabiendo que el consumo máximo de este pin de la stm es de 25mA, esta será de un valor de:

$$RL = \frac{3.3V}{25 \cdot 10^{-3}}$$

Y Siendo R el valor de la resistencia calculada anteriormente.

Como la tensión de entrada al zener es de 12V es suficiente para que funcione correctamente.

La tensión a la salida en el pin de wakeup será de $3.3V - V_{D6} = 3.3 - 0.7 = 2.6V$ que es suficiente para que el micro lo entienda como un nivel alto. Será el mismo valor de tensión de la señal proveniente del sensor de movimiento puesto que su señal de salida, según el datasheet es de:

HC-SR501 PIR MOTION DETECTOR

Product Discription

HC-SR501 is based on infrared technology, automatic control module, using Germany imported LHI778 probe design, high sensitivity, high reliability, ultra-low-voltage operating mode, widely used in various auto-sensing electrical equipment, especially for battery-powered automatic controlled products.

Specification:

- Voltage: 5V – 20V
- Power Consumption: 65mA
- **TTL output: 3.3V, 0V**
- Delay time: Adjustable (.3->5min)
- Lock time: 0.2 sec
- Trigger methods: L – disable repeat trigger, H enable repeat trigger
- Sensing range: less than 120 degree, within 7 meters
- Temperature: – 15 ~ +70
- Dimension: 32*24 mm, distance between screw 28mm, M2, Lens dimension in diameter: 23mm

Figura 3. Catalogo del PIR

Circuito medidor de consumo (Sensor de corriente):

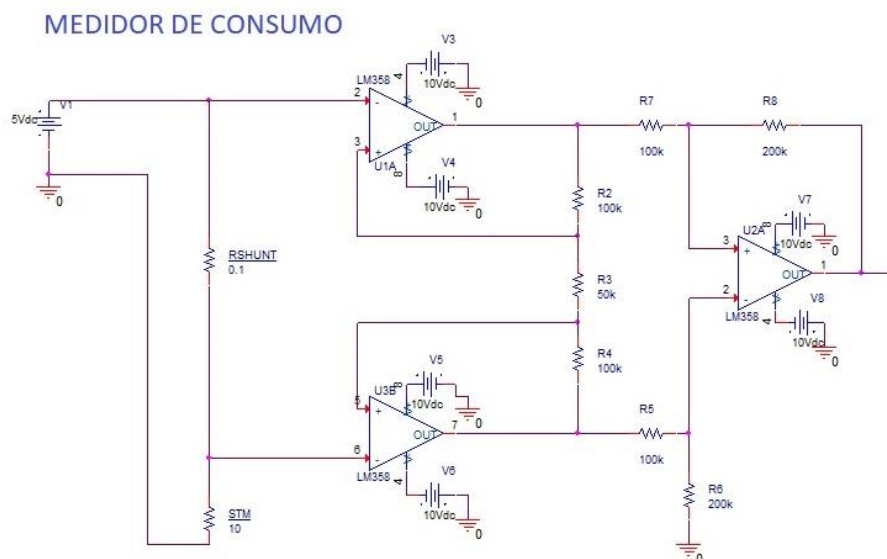


Figura 4. Medidor de consumo

Objetivo:

Este circuito mide el consumo del sistema desde la medida indirecta de la corriente con una resistencia de valor resistivo muy pequeño y con un amplificador de instrumentación para amplificar el valor de tensión entre los terminales de esta resistencia que corresponderá al valor de corriente que consume todo el sistema.

Componentes:

- Integrado LMC6484 (cuatro amplificadores operacionales en uno).
- Resistencias de 100k y 200k
- Potenciómetro de 50k para regular la ganancia.

Funcionalidad:

El diseño del circuito implica la medición indirecta de la corriente que pasa a través de una resistencia de valor muy bajo (para minimizar el consumo), seguido de la amplificación de la diferencia de tensión a través de esta resistencia. Esto nos permite conocer la tensión caída a través de los sistemas conectados y, a partir de ahí, determinar la corriente total consumida por la aplicación.

Elegimos utilizar un amplificador operacional rail-to-rail porque inicialmente estábamos considerando un circuito de alimentación simétrica que generaba una tensión de $\pm 5V$, con la entrada operando a 5V. Por lo tanto, fue necesario utilizar amplificadores operacionales rail-to-rail para poder amplificar una señal de entrada muy cercana al rango de la alimentación simétrica. Hemos seguido utilizando este integrado ya que igualmente nos sirve para nuestra aplicación.

Cálculos:

- **Cálculo de los componentes:**

La tensión máxima de consumo de la stm cuando se alimenta externamente en el pin de 5V según el datasheet es de:

Table 7. External power sources

Input power name	Connector pins	Voltage range	Max current	Limitation
V_{IN}	CN8 pin 15 CN11 pin 24	7 V to 12 V	800 mA	From 7 V to 12 V only and input current capability is linked to input voltage: 800 mA input current when $V_{IN}=7$ V 450 mA input current when $7\text{ V} < V_{IN} < 9\text{ V}$ 250 mA input current when $9\text{ V} < V_{IN} < 12\text{ V}$
E5V	CN11 pin 6	4.75 V to 5.25 V	500 mA	-
+3.3 V	CN8 pin 7 CN11 pin 16	3 V to 3.6 V	-	Two possibilities: ST-LINK PCB is cut SB3 and SB111 OFF (ST-LINK not powered)

Figura 5. Alimentación externa

Luego, para poder medir esa corriente, podemos diseñar el circuito a partir de la siguiente fórmula:

$$V_{out} = \left(\frac{R_6}{R_7}\right) \cdot \left(1 + \frac{2 \cdot R_4}{R_G}\right) \cdot (V_1 - V_{stm})$$

Siendo $R_6 = R_8$, $R_7 = R_5$ y $R_2 = R_4$.

La tensión de salida V_{out} refleja el consumo de corriente de nuestro sistema. Por ejemplo, si nuestra fuente de alimentación es de 5V y experimentamos una caída de tensión de aproximadamente 30mV al conectar la STM (microcontrolador), utilizando una resistencia de shunt (R_{shunt}) de 0.1 ohmios, la corriente que fluye a través de esta resistencia será de 300 mA. Esta corriente representa el consumo combinado del microprocesador y otros sensores conectados al sistema.

2.2 ALIMENTACIÓN.

Circuito de alimentación:

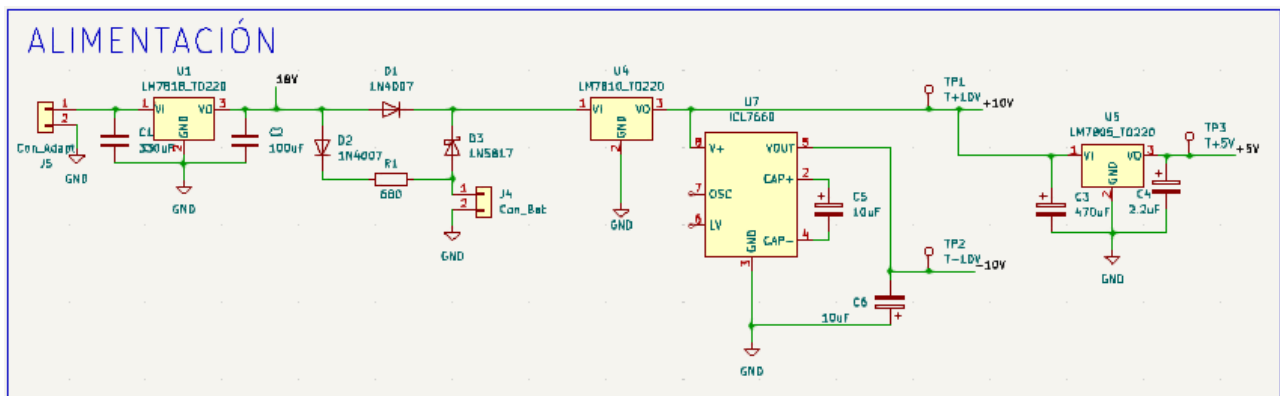


Figura 6. Circuito de alimentación

Circuito battery back-up:

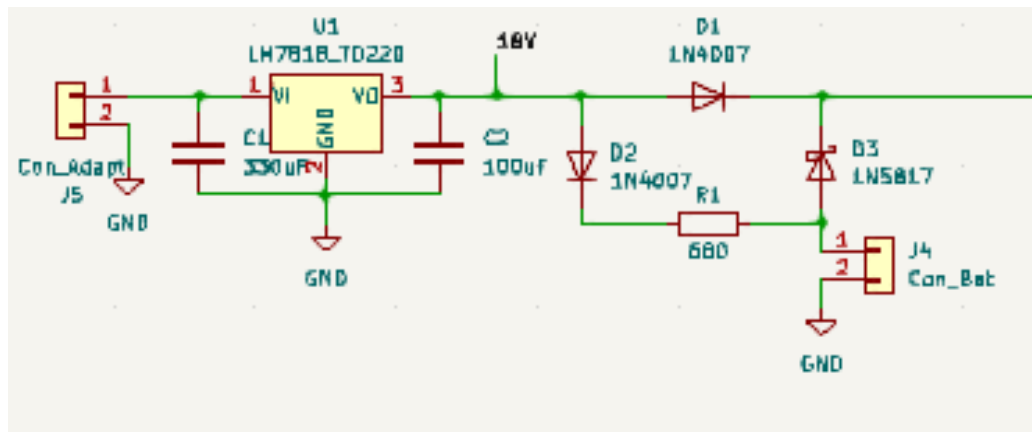


Figura 7. Circuito battery back-up

Objetivo:

Con este circuito, conseguimos la conmutación entre la alimentación por red y la alimentación por baterías. El circuito alimentará el sistema a través de la red eléctrica utilizando un adaptador de 24V/1A, pudiendo cargar a su vez la batería de 16,8V (de voltaje máximo) y con el juego de diodos, se guía a la corriente para poder conmutar de la red eléctrica a la batería haciendo que nunca haya un fallo en el sistema por corte de luz.

Componentes:

- Adaptador de red de 24V/1A.
- 4 baterías de ion-litio 18650 de 3.7V nominales y 4.2V de máxima.
- Regulador LM7818.
- Disipador térmico To220 para el regulador con resistencia térmica de 9°C/W
- Diodos para guiar la corriente D1, D2 y D3.
- Resistencia de 680ohms para regulación de carga de la batería.

Funcionalidad:

Cuando alimentamos el circuito desde la red eléctrica, la corriente primero pasa a través de un regulador de 18V. Esto se debe a que la batería conectada al conector J4 está equipada con un módulo BMS (Sistema de Gestión de Baterías) que supervisa y regula la carga de cada célula de la batería de forma individual. Según las especificaciones del datasheet del BMS, la tensión máxima de alimentación permitida es de 18V:

4 cell 16.8V 40A 18650 lithium battery protection board (with recovery function - AUTO Recovery)

Model: BMS-40A-4S-E / B / S

Integrated with 10 low internal resistance MOSFETS, continuous discharge 40A, 100mA equalizing current, suitable for electric drills, sprayers, LED lights, low power inverters (400W).

BMS-40A-4S-S Standard BMS 4 cell 16.8V 40A lithium battery protection board (with recovery function - AUTO Recovery)
Starting motor less than 60A / power less than 100W.

BMS-40A-4S-E Same as above with the difference of Starting motor less than 80A / power less than 135W.

BMS-40A-4S-B Battery balance version, Starting motor less than 80A / power less than 135W.

Application range: Suitable for lithium batteries with a normal voltage of 3.7V and fully charged 4.2V.
including 1860 to 26650, Polymer lithium batteries.

PCB Size: -Std & Enh. 55 x 45 x 3.4mm
"PCB Size: -Balance 60 x 46 x 3.4mm
Weight: 9.3g and 10.3g

Charging Voltage: 16.8 ~ 18.1V

Continues discharge max: **40A**
If heat dissipation environment is not adequate please reduce load

Continues discharge normal: **40A**

Standard version (-S): Applicable to start electric motor with current below 60A and power up to 100W.

Enhanced version (-E): Applicable to start electric motor with current below 80A and power up to 135W, with interference function.

Figura 8. Datasheet del BMS

El diodo D2 se encarga de dirigir la corriente hacia la batería cuando el sistema se alimenta a través del adaptador para permitir la carga. Por otro lado, el diodo D1 canaliza la corriente procedente del adaptador hacia el resto del sistema. Además, el diodo D1 protege al adaptador cuando se cambia la alimentación a la batería, evitando que la corriente fluya hacia él.

En caso de un corte en la red eléctrica, el sistema dejará de recibir alimentación a través de los diodos D1 y D2 desde el adaptador, y en su lugar, se alimentará de la batería a través del diodo D3.

Cálculos:

- Tensión de salida a partir de la red eléctrica:

$$V_{o_{reg}} = 18V$$

$$V_o = V_{o_{reg}} - VD1 = 18V - 0.7V = 17.3V$$

- Tensión de salida a partir de la batería:

$$V_{o_{max}} = V_{o_{bat-max}} - VD3 = 16.8V - 0.7V = 16.1V$$

$$V_{o_{nom}} = V_{o_{bat-nom}} - VD3 = 14.8V - 0.7V = 14.1V$$

- Cálculo de disipador para el regulador:

Datos del circuito:

$V_{in} = 24V$ (tensión a la salida del adaptador)

$V_{out} = 18V$ (tensión a la salida del regulador)

$I_{out} = 1A$ (corriente máxima de salida del transformador).

Datos del disipador:

$R_{jc} = 5^{\circ}C/W$ (Resistencia unión cápsula del disipador)

$R_{cs} = 1^{\circ}C/W$ (Resistencia capsula fuente)

$R_{sa} = 9^{\circ}C/W$ (Resistencia térmica)

Datos del regulador:

LM78XX (KA78XX, MC78XX) FIXED VOLTAGE REGULATOR (POSITIVE)

ABSOLUTE MAXIMUM RATINGS ($T_A = +25^{\circ}C$, unless otherwise specified)

Characteristic	Symbol	Value	Unit
Input Voltage (for $V_o = 5V$ to $18V$)	V_i	35	V
(for $V_o = 24V$)	V_i	40	V
Thermal Resistance Junction-Cases	$R_{\theta JC}$	5	$^{\circ}C/W$
Thermal Resistance Junction-Air	$R_{\theta JA}$	65	$^{\circ}C/W$
Operating Temperature Range KA78XX/A/R/RA	T_{OPR}	0 ~ +125	$^{\circ}C$
KA78XXI/RI		-40 ~ +125	$^{\circ}C$
Storage Temperature Range	T_{STG}	-65 ~ +150	$^{\circ}C$

Figura 9. Datasheet regurador LM78XX

Temperatura máxima de funcionamiento: -65/150 $^{\circ}C$

- Comprobar si es necesario colocar disipador:

$$\Delta V \cdot I_{in} = (24 - 18)V \cdot 1A = 6W$$

$$T_{Real} = 25 + 6 = 31$$

$$T = 6 \cdot 31 = 186^{\circ}C$$

$$T_{Total} = T + T_{ambiente} = 186^{\circ}C + 25^{\circ}C = 211^{\circ}C$$

Como la temperatura es de 211°C que es mayor que la temperatura máxima de funcionamiento del regulador según el datasheet, habría que ponerle un disipador.

2º) Comprobar si con regulador de resistencia térmica 9°C/W es suficiente:

- Resistencia térmica del disipador:

$$R_{ja} = R_{jc} + R_{cs} + R_{sa}$$

- Escoger disipador con resistencia térmica de 9°C/W:

$$R_{ja} = 5^{\circ}\text{C/W} + 1^{\circ}\text{C/W} + 9^{\circ}\text{C/W} = 14^{\circ}\text{C/W}$$

$$T^{\text{real}} = 25^{\circ}\text{C} + 14^{\circ}\text{C/W} * 6\text{W} = 109^{\circ}\text{C}$$

Se comprueba como utilizando un regulador de resistencia térmica 9°C/W es suficiente para disminuir la temperatura a un valor dentro del rango máximo permitido del regulador.

Circuito alimentación simétrica:

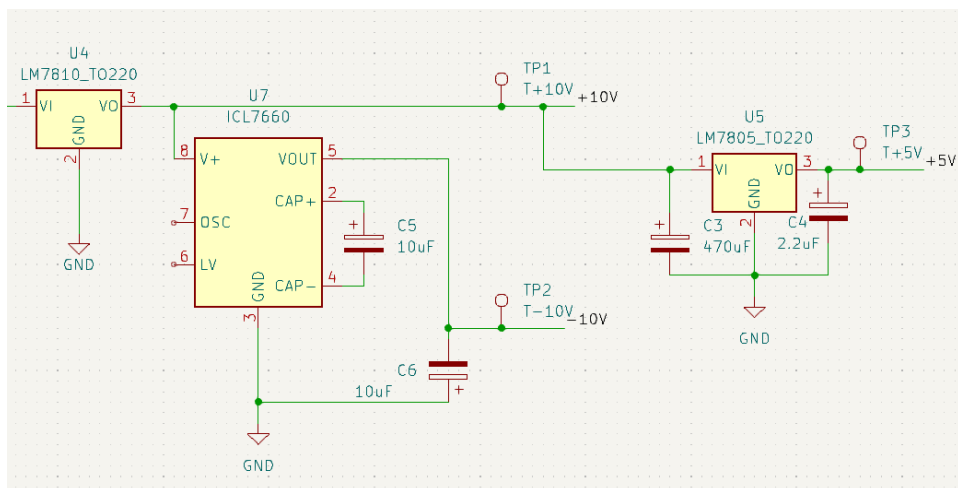


Figura 10. Circuito alimentación simétrica

Objetivo:

Con este circuito nos encargamos de aportar la alimentación simétrica necesaria para los amplificadores operacionales de nuestro amplificador de instrumentación. La alimentación será de +-10V suficiente para nuestro amplificador de instrumentación.

Componentes:

- Regulador LM7810.
- Regulador LM7805.
- Integrado ICL7660 para obtener la tensión negativa.
- Condensador C5 de 10uF

Funcionalidad:

Este circuito se basa en el integrado ICL7660, un convertidor de voltaje DC-DC inversor (DC-DC charge pump voltage inverter). El funcionamiento es el siguiente:

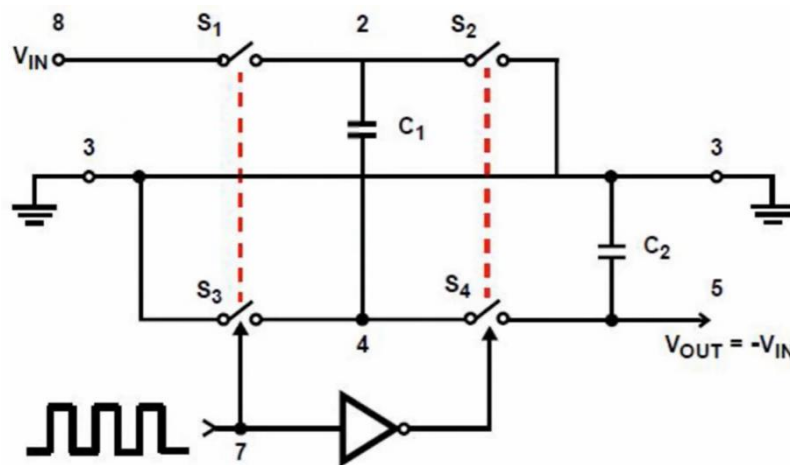


Figura 11. Integrado ICL7660

Cuando los interruptores S_1 y S_3 están cerrados y los interruptores S_2 y S_4 están abiertos, durante la primera mitad del ciclo de funcionamiento se carga el condensador C_1 con un voltaje V_{IN} . En la segunda mitad del ciclo, los interruptores S_2 y S_4 se cierran mientras que S_1 y S_3 se abren. En este momento, la carga almacenada en el condensador C_1 se transfiere al condensador C_2 , lo que proporciona la tensión negativa requerida en la salida del circuito integrado. A partir del condensador C_2 se puede regular la velocidad de carga y descarga de estos condensadores, pero nosotros hemos elegido ese valor de $10\mu F$ puesto que es el valor recomendado por el fabricante.

Principio de la bomba de carga: este utiliza un principio de bomba de carga para generar un voltaje de salida mayor o menor que el voltaje de entrada, con la polaridad invertida. La bomba de carga es un tipo de convertidor de voltaje que utiliza condensadores y un circuito de conmutación para generar un voltaje de salida diferente del voltaje de entrada.

Ciclo de carga y descarga de capacitores: El ICL7660 utiliza dos condensadores internos y un circuito de conmutación para alternar entre ciclos de carga y descarga. Durante el ciclo de carga, un condensador se carga desde la entrada. Durante el ciclo de descarga, el condensador cargado se conecta en serie para sumar sus voltajes con el voltaje de entrada, generando así un voltaje de salida más alto.

Generación de voltaje de salida invertido: Mediante la conmutación y combinación de voltajes a través de estos condensadores durante múltiples ciclos, el ICL7660 genera un voltaje de salida que es la inversa del voltaje de entrada. Por ejemplo, si la entrada es positiva (+V), el ICL7660 puede generar una salida negativa (-V).

Este integrado admite distintas formas de configuración, y en nuestro caso, solo lo utilizaremos como inversor de alimentación de la entrada. El voltaje de entrada en este caso es de 10V que es el máximo permitido según el datasheet por este integrado:

ICL7660, ICL7660A

Absolute Maximum Ratings

Supply Voltage	
ICL7660	+10.5V
ICL7660A	+13.0V
LV and OSC Input Voltage	-0.3V to (V+ +0.3V) for V+ < 5.5V (Note 2)
Current into LV (Note 2)	20µA for V+ > 5.5V
Output Short Duration (V _{SUPPLY} ≤ 5.5V)	Continuous

Operating Conditions

Temperature Range	
ICL7660C, ICL7660AC	0°C to 70°C
ICL7660AI	-40°C to 85°C

Thermal Information

Thermal Resistance (Typical, Note 1)	θ _{JA} (°C/W)	θ _{JC} (°C/W)
PDIP Package*	110	N/A
SOIC Package	160	N/A
Maximum Storage Temperature Range	-65°C to 150°C	
Maximum Lead Temperature (Soldering, 10s)	300°C (SOIC - Lead Tips Only)	

*Pb-free PDIPs can be used for through hole wave solder processing only. They are not intended for use in Reflow solder processing applications.

CAUTION: Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

Figura 12. Datasheet ICL7660

obteniendo así una salida de -10V. Por último, está el regulador de 5V que es el que irá directamente a alimentar todos los componentes del sistema (STM, sensores, etc...).

Cálculos:

- Rendimiento del ICL7660:

$$n = \frac{V_{out}}{V_{in}} \cdot 100\%$$

Como V_{out} = V_{in} tendremos un rendimiento del 100%.

ESQUEMA ELECTRÓNICO COMPLETO.

Sistema completo:

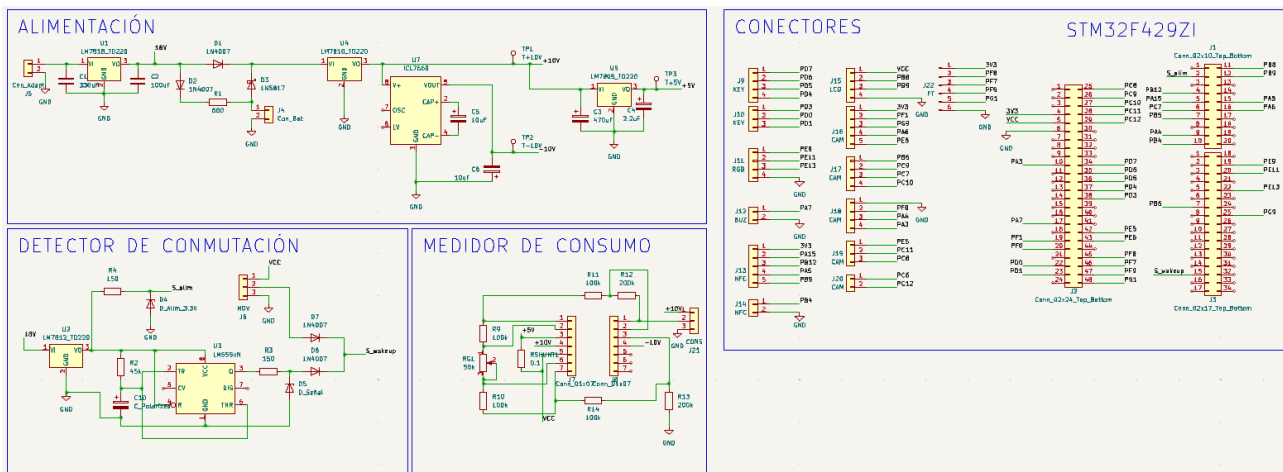


Figura 13. Sistema completo

Conexión de circuito con la stm:

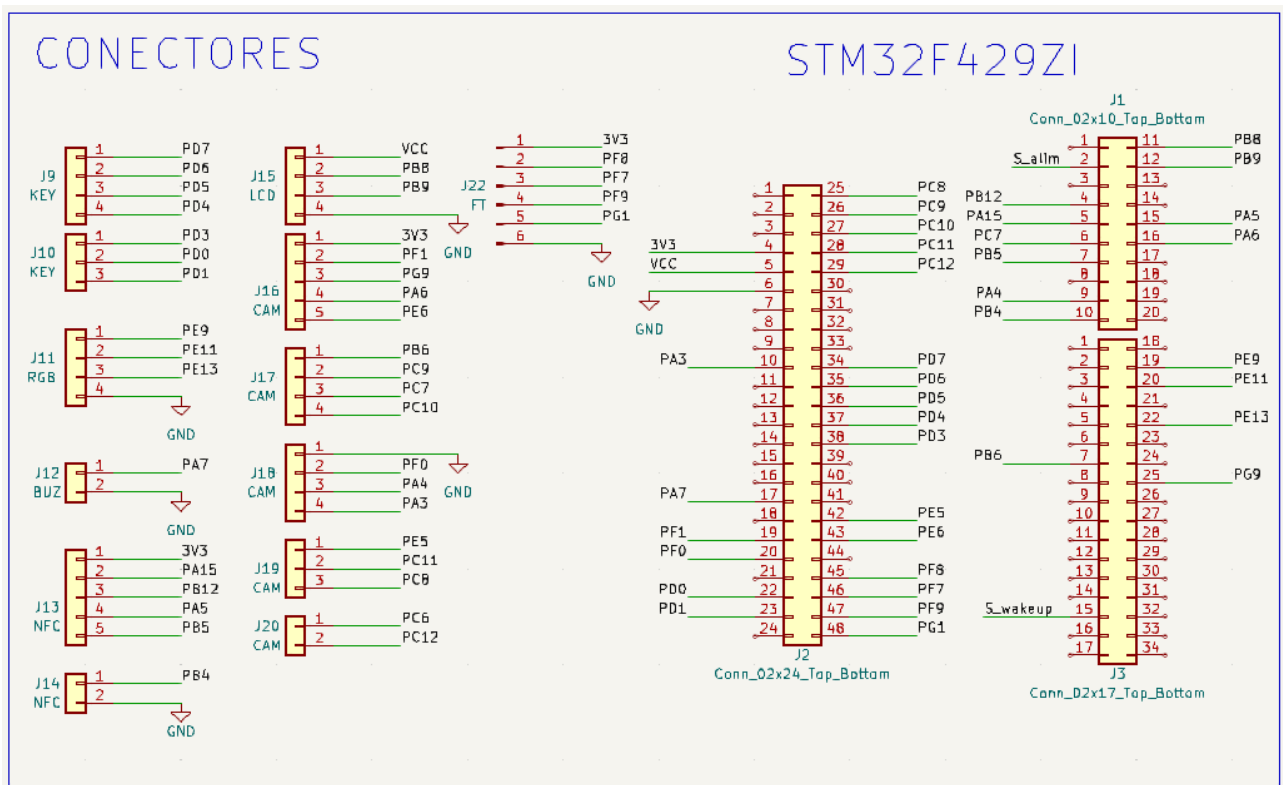


Figura 14. Circuito de conexión

Diseño de PCB:

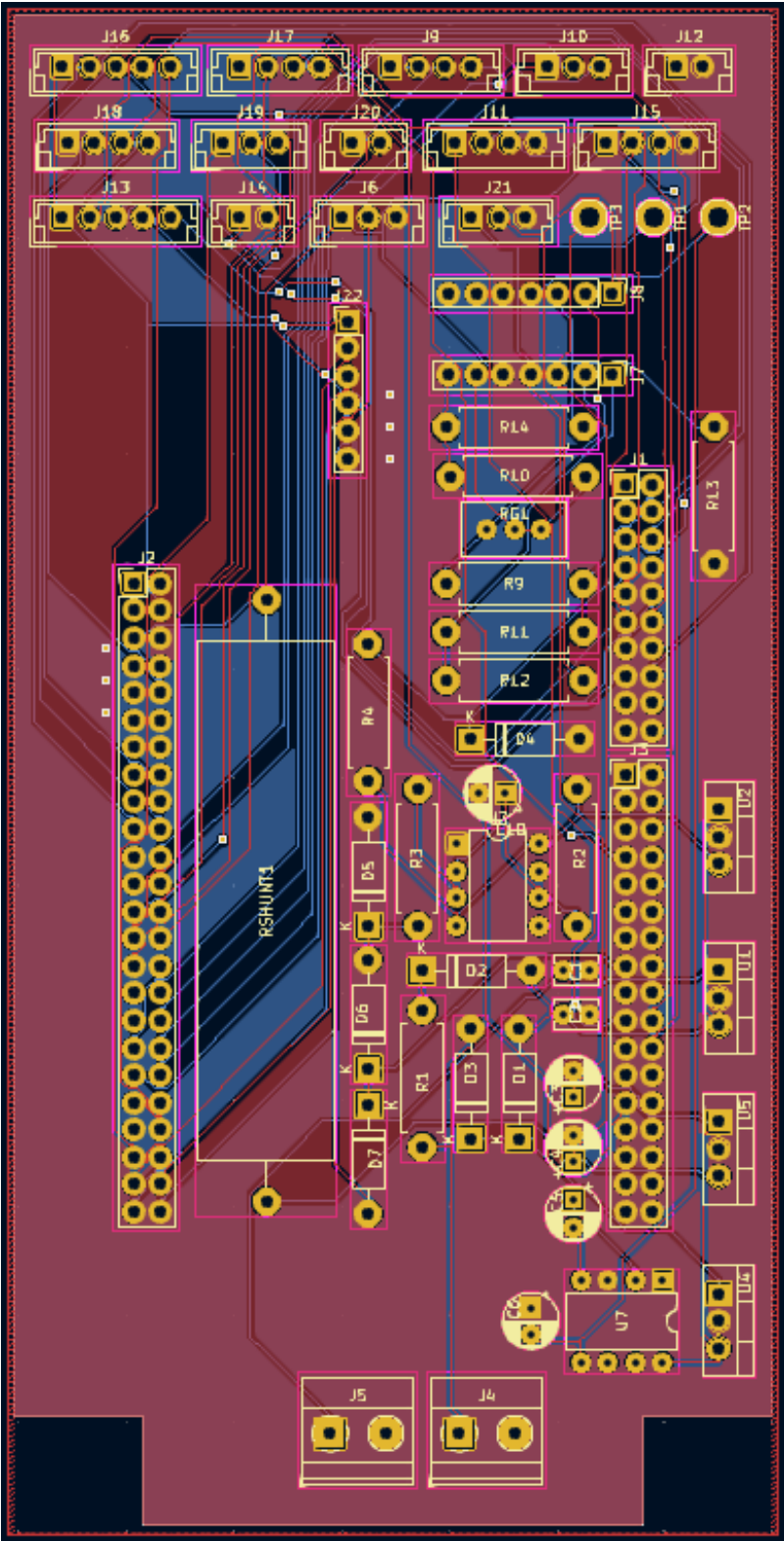


Figura 15. Diseño PCB

Pines utilizados:

		TOP		
			CH7	
			PC6	PB8
			PB15	PB9
			PB13	AVDD
			PB12	GND
			PA15	PA5
			PC7	PA6
			PB5	PA7
			PB3	PD14
			PA4	PD15
			PB4	PF12
			AVDD	PF13
			AGND	PE9
			GND	PE11
			PB1	PF14
			PC2	PE13
			PF4	PF15
			PB6	PG14
			PB2	PG9
			GND	PE8
			PD13	PE7
			PD12	GND
			PD11	PE10
			PE2	PE12
			GND	PE14
			PA0	PE15
			PB0	PB10
			PE0	PB11
			CH10	

Figura 16. Esquema de pines

Visualización hardware:

El diseño inicial de la caja que contendría todos los componentes es la siguiente:

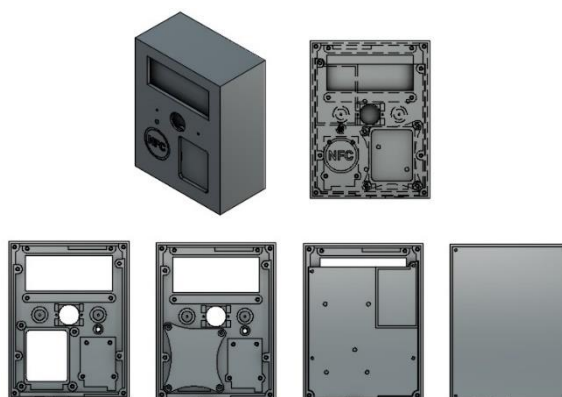


Figura 17. Diseño caja final

Y la presentación final:

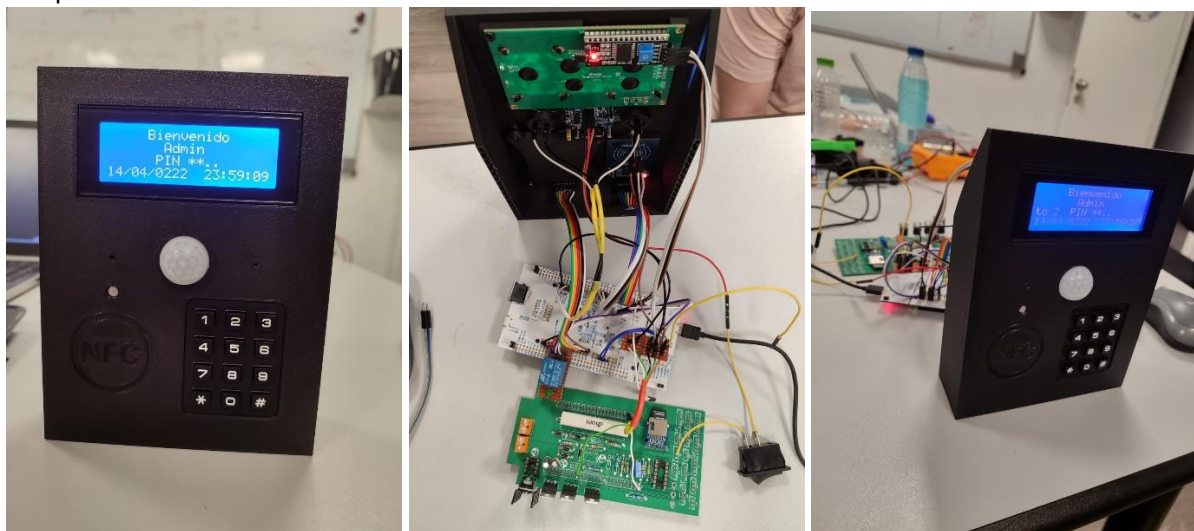


Figura 18. Representación final hardware

3 SOFTWARE.

3.1 Interfaz de usuario.

Se presentará el front-end para interactuar con el sistema, y el funcionamiento asociado a cada elemento de dicha interface.

3.2 Descripción de cada uno de los módulos del sistema.

3.2.1 Buzzer



Figura 1719. Buzzer

Ficheros: *buz.c* y *buz.h*

Descripción

Este código implementa un controlador para un zumbador (buzzer) utilizando un temporizador PWM. El zumbador puede generar tonos de diferentes frecuencias y volúmenes, controlados a través de mensajes en una cola.

El módulo lleva a cabo las siguientes operaciones:

- 1) **Inicialización del Zumbador (Th_buz):**
 - Configura el zumbador y el temporizador PWM al inicio.
 - Entra en un bucle infinito para esperar mensajes de tono desde la cola de mensajes.
- 2) **Recepción de Mensajes (osMessageQueueGet):**
 - Espera a recibir un mensaje desde la cola de mensajes que especifica la frecuencia, volumen y duración del tono a reproducir.

```
typedef struct{
    uint32_t frequency_hz;
    uint32_t duration_ms;
    uint8_t volume;
} MSGQUEUE_OBJ_BUZ;
```

Figura 1820. MSGQUEUE_OBJ_BUZ

3) Configuración del Tono (set_tone):

- Calcula y configura la frecuencia y el volumen del tono utilizando la función **set_tone**.
- Reinicializa el temporizador PWM con los nuevos parámetros de configuración.

4) Reproducción del Tono (HAL_TIM_PWM_Start/Stop):

- Inicia la reproducción del tono generando una señal PWM con la frecuencia y el volumen configurados.
- Espera la duración del tono especificada en el mensaje.
- Detiene la reproducción del tono al finalizar la duración especificada

Configuración

Se utiliza el pin PB0 de la STM32, configurado como un timer en modo PWM.

```
static void init_buz(TIM_HandleTypeDef *htim, TIM_OC_InitTypeDef *hsoc){
    static GPIO_InitTypeDef sgpio = {0};

    __HAL_RCC_GPIOB_CLK_ENABLE();
    sgpio.Pin= GPIO_PIN_0;
    sgpio.Mode= GPIO_MODE_AF_PP;
    sgpio.Alternate= GPIO_AF2_TIM3;
    sgpio.Pull= GPIO_NOPULL;
    sgpio.Speed= GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOB, &sgpio);

    __HAL_RCC_TIM3_CLK_ENABLE();
    htim->Instance= TIM3;
    htim->Init.Prescaler = 840; // 84MHz/840 = 100KHz -> 10us
    htim->Init.Period = 100000 - 1; // 100KHz/100000 = 1Hz technically no sound
    htim->Init.CounterMode = TIM_COUNTERMODE_UP;
    htim->Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
    HAL_TIM_PWM_Init(htim);

    hsoc->OCMode = TIM_OCMode_PWM1;
    hsoc->OCpolarity = TIM_OCPolarity_HIGH;
    hsoc->OCFastMode = TIM_OCFAST_DISABLE;
    hsoc->Pulse = 0; // 0% (volume)
    HAL_TIM_PWM_ConfigChannel(htim, hsoc, TIM_CHANNEL_3);
}
```

Figura 1921. Configuración Buzzer

Entradas

La entrada de este módulo es un mensaje a la cola *id_MsgQueue_buz*, enviado por el hilo principal.

Salidas

Este módulo no tiene ninguna salida hacia otro módulo.

3.2.2 FT

Ficheros: *tff.c* y *tff.h*

Descripción

En este módulo se escribe y se lee en una tarjeta SD por el protocolo de comunicaciones SPI. Se compone de dos colas *get_id_MsgQueue_ttf_miso* y *get_id_MsgQueue_ttf_mosi*. La cola miso espera recibir mensajes de tipo *MSGQUEUE_OBJ_TTF_MISO* y la cola mosi *MSGQUEUE_OBJ_TTF_MOSI*.

Se utiliza un sistema de fichero para almacenar los datos. Los datos se almacenan en un fichero .csv, es decir se almacenan con una coma entre datos.

```
typedef struct{
    cmd_t cmd;
    char data [5][24];
} MSGQUEUE_OBJ_TTF_MOSI;

typedef struct{
    char adtos[50][24];
} MSGQUEUE_OBJ_TTF_MISO;
```

Figura 2022. MSG TTF

Cuando se recibe un mensaje procedente del hilo principal que indique escritura se escribe el dato indicado en el fichero data.csv.

Cuando se recibe el mensaje de lectura se lee todo el fichero y se envía a la cola.

Configuración

La tarjeta utiliza SPI5 y se configura de la siguiente manera.


	SPI5 (Serial Peripheral Interface 5) [Driver_SPI5]	<input checked="" type="checkbox"/>
	SPI5_MISO Pin	PF8
	SPI5_MOSI Pin	PF9
	SPI5_SCK Pin	PF7
	SPI5_NSS Pin	PF6

Figura 2123. Configuración TTF

Entradas

Entrada msg de escritura procedente del hilo principal.

Salidas

Salida de msg, en el cual se almacenan todos los datos leídos en el fichero.

3.2.3 Keypad

Ficheros: *key.c* y *key.h*

Descripción

Este módulo es el encargado del keypad 4x3, 4 filas, 3 columnas. Las filas del teclado están conectadas a salidas del circuito y las columnas a entradas. La tecla pulsada se detecta cuando su fila y salida correspondiente están activas, por lo que el teclado debe ir muestreando las filas con el objetivo de detectar la columna pulsada.

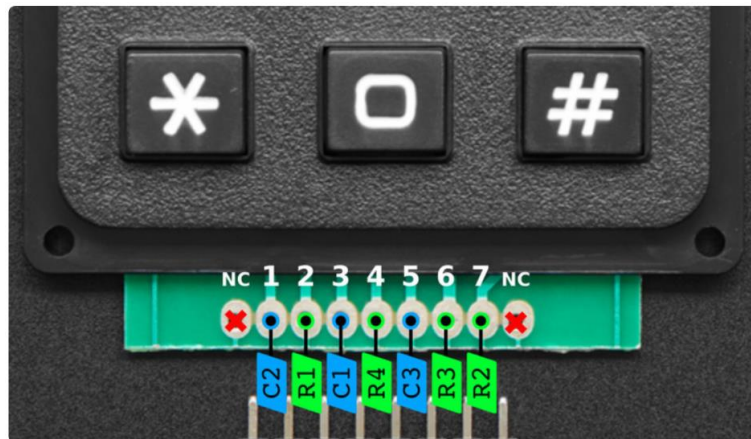


Figura 2224. Pines Keypad

El código implementado utiliza una cola *id_MsgQueue_key* y los timers *tim_id_Muestreo*, *tim_id_Rebotes* y *tim_id_KeyOn*.

La función principal de este módulo es esperar a que se reciba alguna señal. A continuación, se analizan las señales a recibir y la función que se debe realizar en cada caso.

Key_FLAG_ON: Se activa en el hilo principal cuando este quiere activar la funcionalidad del teclado. Su función es iniciar el timer periódico *tim_id_Muestreo*, encargado de muestrear las filas del teclado.

FLAG_IRQ: Se activa cuando se genera una pulsación en una tecla y su función es iniciar el *tim_id_Rebotes*, encargado de esperar un tiempo para evitar rebotes.

FLAG_REBOTES: Se activa cuando ha transcurrido el tiempo de rebotes y su función es detectar la tecla pulsada y mandarla a la cola *id_MsgQueue_key*.

FLAG_MUESTREO: Se activa cada vez transcurrido el tiempo del timer *tim_id_Muestreo*, cada vez que se activa cambia la fila activa.

FLAG_CHECK_KEY: Se activa cada vez trascurrido el tiempo del timer periodico *tim_id_KeyOn*, encargado de comprobar si una tecla sigue pulsada, de esta manera solo se envía un mensaje de tecla pulsada cuando se envía una tecla.

Configuración

Los pines GPIO se configuran como se muestra a continuación:


```

static void GPIO_Init(void) {
    static GPIO_InitTypeDef  GPIO_InitStruct;

    __HAL_RCC_GPIOC_CLK_ENABLE();

    GPIO_InitStruct.Pin = GPIO_PIN_7 | GPIO_PIN_6 | GPIO_PIN_5;
    GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
    GPIO_InitStruct.Pull = GPIO_PULLUP;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

    GPIO_InitStruct.Pin = GPIO_PIN_1 | GPIO_PIN_3 | GPIO_PIN_4 | GPIO_PIN_0;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_PULLUP;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

    HAL_NVIC_EnableIRQ(EXTI9_5_IRQn);
}

```

Figura 2325. Configuración pines GPIO

Entradas

La entrada de este módulo es la activación del flag KEY_FLAG_ON.

Salidas

Las salidas de este módulo son las teclas pulsadas que se introducen en la cola de mensajes *id_MsgQueue_key*, los mensajes serán extraídos por el hilo principal.

3.2.4 LCD

Ficheros: *LCD.c*, *LCD.h*



Figura 24. LCD

Descripción

Este módulo es el encargado de controlar un LCD basado en I2C

LCD.c:

- **Descripción:**
- **Configuración:**
 - **Librerías y definiciones:** El código incluye varias librerías estándar, así como definiciones y encabezados necesarios para trabajar con I2C y el LCD.
 - **Definiciones de constantes:** Se definen constantes como la dirección del esclavo del LCD (**SLAVE_ADDRESS_LCD**) y el número máximo de objetos de mensaje en la cola (**MSGQUEUE_OBJECTS_LCD**).
- **Entradas y Salidas:**
 - **Entradas:**
Mensajes a la cola de mensajes (**id_MsgQueue_lcd**) para actualizar y controlar el contenido de la pantalla LCD.
 - **Salidas:**
Control de una pantalla LCD mediante I2C, enviando comandos y datos específicos para configurar y mostrar texto en la pantalla.
- **Funciones Principales:**
 - 1) init_Th_lcd:** Función para inicializar el hilo (**Thread**) responsable de manejar la lógica del LCD.
 - 2) Init_MsgQueue_lcd:** Función para inicializar la cola de mensajes que será utilizada para enviar comandos y datos al LCD.
 - 3) init_lcd:** Función de inicialización del LCD, configurando la comunicación I2C y la pantalla para su uso.
 - 4) send_cmd:** Función para enviar comandos al LCD, como configuraciones de modo, limpieza de pantalla, etc.
 - 5) send_data:** Función para enviar datos al LCD, como caracteres individuales para mostrar en la pantalla.
 - 6) send_string:** Función para enviar una cadena de caracteres al LCD.
 - 7) set_cursor:** Función para mover el cursor a una posición específica en la pantalla LCD.
 - 8) clear:** Función para borrar el contenido de la pantalla LCD.
- **Paso por Paso:**
 - 1º) Inicialización:**
 - Se inicializa el hilo (**Thread**) y la cola de mensajes del LCD (**init_Th_lcd**).
 - Dentro de **init_Th_lcd**, se llama a **Init_MsgQueue_lcd** para crear la cola de mensajes.
 - 2º) Configuración del LCD (init_lcd):**
 - Se inicializa la comunicación I2C con el LCD (**I2Cdrv>Initialize(callback_i2c)**).
 - Se configuran parámetros como velocidad de bus I2C y modo de funcionamiento del LCD.
 - Se envían una serie de comandos específicos (**send_cmd**) para inicializar el LCD en modo de 4 bits.
 - 3º) Funcionamiento Principal (Th_lcd):**
 - En este hilo, se inicializa y configura el LCD (**init_lcd** y **clear**).
 - El hilo espera constantemente por mensajes en la cola (**osMessageQueueGet**).
 - Dependiendo del mensaje recibido, se actualiza el contenido de la pantalla:
 - Se ajusta el estado del fondo (**back_light**).
 - Se actualizan las líneas individuales de texto en la pantalla, moviendo el cursor y enviando cadenas de texto (**send_string**).

LCD.h

- **Descripción:**

El archivo **lcd.h** define las estructuras de datos y las funciones necesarias para interactuar con la pantalla LCD a través de I2C en el sistema embebido con CMSIS-OS. Proporciona abstracciones para el estado del LCD y las operaciones básicas de inicialización y obtención de identificadores de cola de mensajes para controlar la pantalla desde otras partes del programa.

- **Funciones Principales:**

- **Enumeración (lcd_state_t):**

Se define un tipo de enumeración llamado **lcd_state_t** que representa el estado de encendido/apagado de la pantalla LCD (**ON** o **OFF**). Esto se utiliza para controlar el fondo de la pantalla.

- **Estructura (MSGQUEUE_OBJ_LCD):**

Se define una estructura llamada **MSGQUEUE_OBJ_LCD** que encapsula el estado de la pantalla LCD y las líneas de texto que se mostrarán en ella.

Contiene un campo **state** de tipo **lcd_state_t** para el estado de la pantalla (encendido o apagado). Contiene cuatro arrays de caracteres (**L0**, **L1**, **L2**, **L3**), cada uno con espacio para 21 caracteres (incluyendo el carácter nulo **\0**) que es el máximo tamaño de las 4 líneas del LCD. Estos campos representan las líneas de texto que se mostrarán en la pantalla.

3.2.5 NFC

- **Ficheros:**

NFC.c, NFC.h



Figura 2526. NFC

NFC.c:

- **Descripción:**

Este código implementa la funcionalidad de un lector NFC basado en el chip MFRC522 a través de una interfaz SPI

- **Descripción de los Registros del MFRC522 utilizados en nuestro programa:**

- 1) **CommandReg (0x01):**

Este registro se utiliza para enviar comandos al chip MFRC522. En nuestra aplicación, se utiliza para enviar comandos como **PCD_RESETPHASE** para reiniciar el chip.

- 2) **CommIrqReg (0x04):**

Este registro indica las fuentes de interrupción activas. En nuestra aplicación, se utiliza para configurar y verificar interrupciones durante la comunicación con las tarjetas RFID.

3) FIFODataReg (0x09):

Este registro se utiliza para escribir y leer datos en el FIFO (First In, First Out) del chip MFRC522. En nuestra aplicación, se usa para almacenar datos durante la comunicación con las tarjetas RFID.

4) FIFOLevelReg (0x0A):

Este registro indica el nivel actual de ocupación del FIFO del chip MFRC522. Se utiliza para gestionar la transferencia de datos entre el chip y el sistema host.

5) ControlReg (0x0C):

Este registro se utiliza para configurar varios parámetros de control del chip MFRC522. En nuestra aplicación, se emplea para configurar modos de operación y activar la antena.

6) BitFramingReg (0x0D):

El registro BitFramingReg controla ciertos aspectos del marco de bits durante la transmisión y recepción de datos. En nuestra aplicación, se utiliza para configurar la transmisión de datos.

- **Funciones de Bajo Nivel de SPI**

- **RC522_SPI_Transfer(uint8_t data):** Realiza una transferencia SPI con el MFRC522.
- **Write_Reg(uint8_t addr, uint8_t val):** Escribe un valor en un registro del MFRC522.
- **Read_Reg(uint8_t addr):** Lee el valor de un registro del MFRC522.
- **SetBitMask(uint8_t reg, uint8_t mask):** Establece bits específicos en un registro del MFRC522.
- **ClearBitMask(uint8_t reg, uint8_t mask):** Borra bits específicos en un registro del MFRC522.
- **AntennaOn(void):** Activa la antena del MFRC522.
- **Reset(void):** Envía un comando de reinicio al MFRC522.

- **Funciones de Comunicación y Control del MFRC522**

- **Request(uint8_t reqMode, uint8_t* TagType):** Realiza una solicitud al PICC (Proximity Integrated Circuit Card, como una tarjeta RFID).
- **ToCard(uint8_t command, uint8_t *sendData, uint8_t sendLen, uint8_t *backData, uint16_t *backLen):** Envía datos al PICC y recibe una respuesta.
- **Anticoll(uint8_t *serNum):** Realiza el procedimiento de anticollisión para identificar las tarjetas.
- **CalulateCRC(uint8_t *pIndata, uint8_t len, uint8_t *pOutData):** Calcula un código de redundancia cíclica (CRC) para los datos.
- **Halt(void):** Envía un comando de hibernación al PICC.

- **Funciones de Inicialización y Configuración**

- **Init(void):** Inicializa el MFRC522 configurando registros y parámetros.

- **Check(uint8_t* id):** Realiza un chequeo para detectar tarjetas cercanas y recupera su identificador.
- **Funciones de Gestión de Hilos**
 - **Th_nfc(void *arg):** Función del hilo principal del lector NFC. Se ejecuta en un bucle infinito esperando activación y realizando comprobaciones de presencia de tarjetas.
- **Funciones de Inicialización del Sistema**
 - **init_Th_nfc(void):** Inicializa el hilo principal del lector NFC y la cola de mensajes asociada.
 - **Init_MsgQueue_nfc(void):** Inicializa la cola de mensajes utilizada para comunicarse con otros componentes del sistema.
- **Gestión de Interrupciones y Temporizadores**
 - **callback_spi(uint32_t event):** Función de callback para manejar eventos de la interfaz SPI.
 - **Timer_Callback(void):** Función de callback para manejar eventos de temporización.
- **Configuración de Hardware y Entorno de Ejecución**
 - Se realizan inicializaciones de pines GPIO y de la interfaz SPI.
 - Se configuran temporizadores y manejo de interrupciones.
- **Funciones Principales del Sistema**
 - El hilo principal (**Th_nfc**) ejecuta un bucle donde espera activación para detectar tarjetas y realizar las operaciones de comunicación con el MFRC522.
- **Funcionamiento Paso a Paso:**
 - 1) Inicialización del Sistema:**
 - El programa comienza inicializando el sistema, configurando los pines GPIO necesarios y estableciendo la comunicación SPI con el chip MFRC522.
 - Se inicializa también un temporizador para gestionar ciertas operaciones temporales.
 - 2) Configuración del MFRC522:**
 - Se configura el MFRC522 enviando una secuencia de comandos de configuración al chip mediante el protocolo SPI. Esto incluye la configuración de parámetros como el modo de operación, la potencia de la señal, la frecuencia de transmisión y la activación de la antena.
 - 3) Funciones de Comunicación con las Tarjetas NFC:**
 - El programa implementa funciones específicas para realizar diferentes operaciones con las tarjetas NFC, como detectar tarjetas, autenticar, leer y escribir datos, entre otras.
 - 4) Verificación y Autenticación:**

- Cuando se detecta una tarjeta NFC, el programa ejecuta una secuencia para verificar su presencia y autenticarla. Esto implica enviar comandos al chip MFRC522 para iniciar la comunicación con la tarjeta, realizar la autenticación con una clave específica y validar la respuesta de la tarjeta.

5) Lectura o Escritura de Datos:

- Una vez autenticada, la aplicación puede leer o escribir datos en la tarjeta NFC. Se utilizan comandos específicos del MFRC522 para transmitir datos de ida y vuelta entre la tarjeta NFC y el sistema host a través del protocolo SPI.

6) Finalización de la Operación:

- Después de completar las operaciones de lectura o escritura, el programa finaliza la comunicación con la tarjeta NFC y vuelve al estado de espera para detectar nuevas tarjetas.

NFC.h:

- **Descripción:**

Define una interfaz y estructuras para manejar las operaciones del NFC.

- **Constantes:**

- **NFC_FLAG_ON:** Define flag que indica el estado de encendido (**ON**) de la operación NFC.
- **NFC_TIMEOUT_MS:** Define el tiempo de espera en milisegundos para las operaciones NFC.

- **Estructura de Datos (MSGQUEUE_OBJ_NFC):**

- Define una estructura llamada **MSGQUEUE_OBJ_NFC** que contiene un campo **sNum** de tipo array de **uint8_t** con longitud 5. Esta estructura se utiliza para almacenar datos relacionados con la operación NFC.

- **Prototipos de Funciones:**

- **int init_Th_nfc(void):**
 - Prototipo de una función que inicializa el hilo (thread) responsable de las operaciones NFC.
 - Retorna un entero (**int**) que indica el estado de la inicialización.
- **osThreadId_t get_id_Th_nfc(void):**
 - Prototipo de una función que retorna el identificador (**osThreadId_t**) del hilo NFC creado.
- **osMessageQueueId_t get_id_MsgQueue_nfc(void):**
 - Prototipo de una función que retorna el identificador (**osMessageQueueId_t**) de la cola de mensajes utilizada para la comunicación relacionada con NFC.

3.2.6 RGB

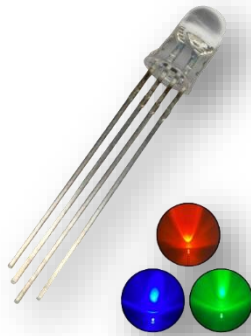


Figura 2627. RGB

Ficheros: *RGB.c* y *RGB.h*

Descripción

Este módulo es el encargado de controlar un dispositivo RGB utilizando un temporizador PWM. El hilo *TH_rgb* espera a la recepción del mensaje *msg_rgb* que contiene todo el espectro posible del rgb.

```
typedef struct{
    uint8_t r;
    uint8_t g;
    uint8_t b;
} MSGQUEUE_OBJ_RGB;
```

Configuración:

- **Inicialización de la Cola de Mensajes (Init_MsgQueue_rgb):**
 - Esta función inicializa la cola de mensajes utilizada para las operaciones RGB.
 - Crea una nueva cola de mensajes con un tamaño específico (**MSGQUEUE_OBJECTS_RGB**).
- **Inicialización del Hilo RGB (init_Th_rgb):**
 - Crea un nuevo hilo (**Th_rgb**) para controlar las operaciones RGB.
 - Llama a **Init_MsgQueue_rgb** para inicializar la cola de mensajes.
- **Configuración del PWM (init_rgb):**
 - Esta función configura los pines GPIO y el temporizador (**TIM1**) para el funcionamiento PWM.
 - Inicia el temporizador PWM y sus canales para controlar el dispositivo RGB.
- **Entradas y Salidas:**
- **Entradas (msg_rgb):**
 - Mensajes de la cola de mensajes que contienen información sobre los niveles de intensidad de los colores rojo (**r**), verde (**g**) y azul (**b**) para controlar el dispositivo RGB.
- **Salidas (TIM1):**
 - Se utiliza el temporizador (**TIM1**) con canales PWM para generar señales de control que ajustan la intensidad de los colores (**r, g, b**) del dispositivo RGB.

Funciones Principales:

- **init_rgb:**
 - Inicializa los pines GPIO y el temporizador (**TIM1**) para generar señales PWM.
 - Configura el temporizador para operar en modo PWM y establece la frecuencia de oscilación y la resolución.
- **Th_rgb:**
 - Hilo principal encargado de controlar el dispositivo RGB.
 - Espera continuamente la recepción de mensajes de la cola de mensajes RGB.
 - Cuando recibe un mensaje, ajusta los niveles de intensidad de los colores (**r, g, b**) utilizando el temporizador PWM.

Paso por Paso:

Inicialización del Sistema (init_Th_rgb):

- Crea un nuevo hilo (**Th_rgb**) para controlar las operaciones RGB.
- Llama a **Init_MsgQueue_rgb** para inicializar la cola de mensajes.

Configuración del PWM (init_rgb):

- Configura los pines GPIO (**GPIOE**) para su uso como salidas alternas para el control PWM.
- Inicializa el temporizador (**TIM1**) para generar señales PWM con una frecuencia y resolución específicas.

Bucle Principal del Hilo (Th_rgb):

- Entra en un bucle infinito.
- Espera la recepción de mensajes de la cola de mensajes RGB.
- Cuando se recibe un mensaje, ajusta los niveles de intensidad de los colores (**r, g, b**) utilizando el temporizador PWM (**TIM1**).

3.2.7 RTC

Ficheros: *rtc.c* y *rtc.h*

Descripción

Este módulo es el encargado de hacer uso del periférico RTC incluido en el microcontrolador para mantener la hora localmente, además se dará soporte para sincronizar la hora mediante un servidor de tiempo externo. Para la realización de este código se parte del código de ejemplo proporcionado por STM.

El módulo lleva a cabo las siguientes operaciones:

1. En primer lugar, se configura con una hora por defecto, mediante la llamada a la función *RTC_CalendarConfig()*. La llamada a esta función hace que se escriba en el registro RTC_BKP_DR1 el valor 0x32F2. Es un registro de backup, que será leído cada vez que se produce un reset en el sistema, si el dato leído se corresponde con 0x32F2 no se vuelve a inicializar el sistema.
2. Se establece conexión con el servidor SNTP *tic.redcayle.es*, cuya IP es 185.179.104.7, mediante la llamada a la función *init_Sntp()*. Cada llamada a esta función también escribe 0x32F2 en el registro RTC_BKP_DR1. En caso de que no sea posible establecer comunicación con el servidor SNTP la hora del RTC continuará siendo la configurada previamente mediante *RTC_CalendarConfig()*.
3. Se arranca un timer de 3 minutos con el objetivo de establecer comunicación con el servidor SNTP cada 3 minutos para evitar pérdidas de tiempo del RTC.
4. Mediante un *osDelay()* cada segundo se llama a función *RTC_Show()*, función encargada de actualizar la hora en la variable global *g_time*.

Toda esta funcionalidad se lleva a cabo mediante el *thread* *Th_rtc* que será inicializado en el thread principal del sistema.

Configuración

La STM32 utiliza el PHY LAN8742A-CZ-TR por lo que es necesario configurar el fichero `RTE_Device.h` como se muestra a continuación:

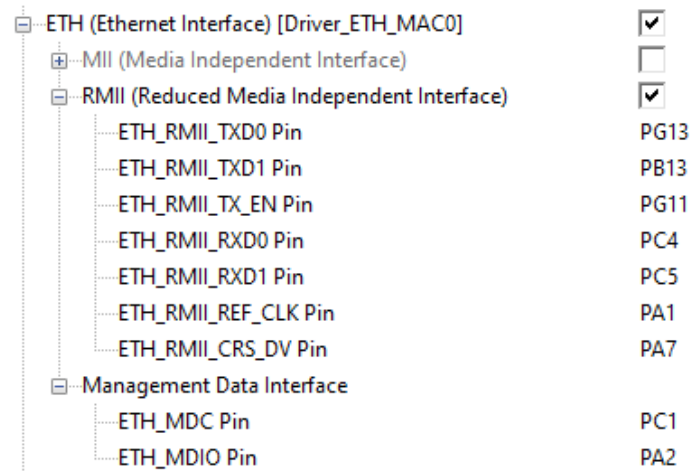


Figura 2728. configuracion RTE_Device.h

Figura . Configuración RTC RTE_Device.h

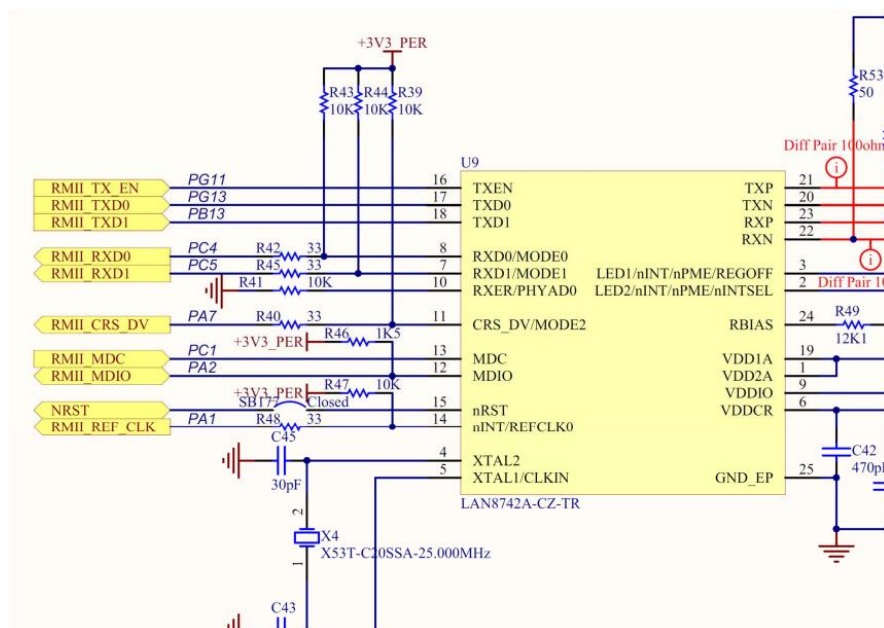


Figura 2829. PHY LAN8742A-CZ-TR

Figura . PHY LAN8742A-CZ

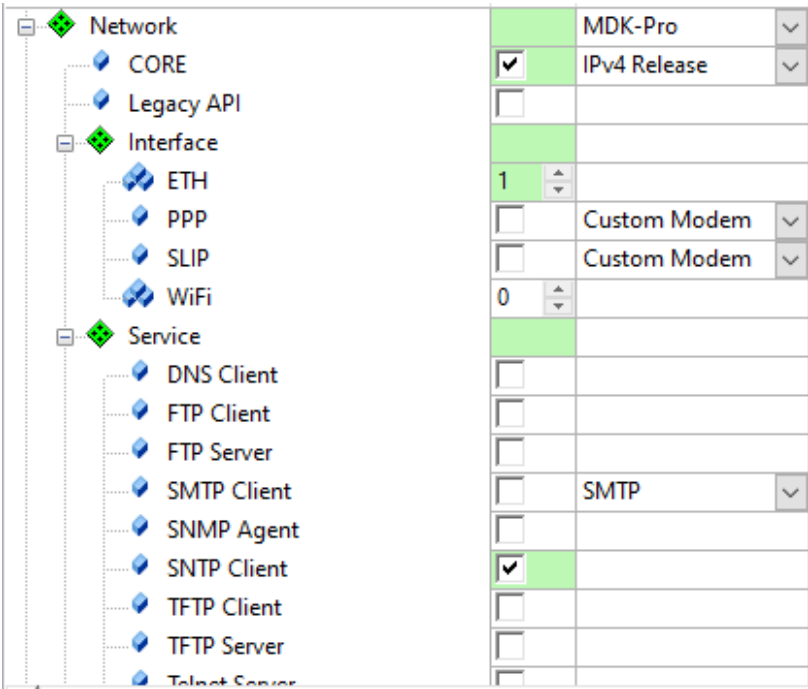


Figura 29 30. RTC Network

Entradas

Este módulo no cuenta con entradas externas.

Salidas

La salida del módulo es la variable global *g_time* del tipo *mytime_t*, que tiene la siguiente estructura:

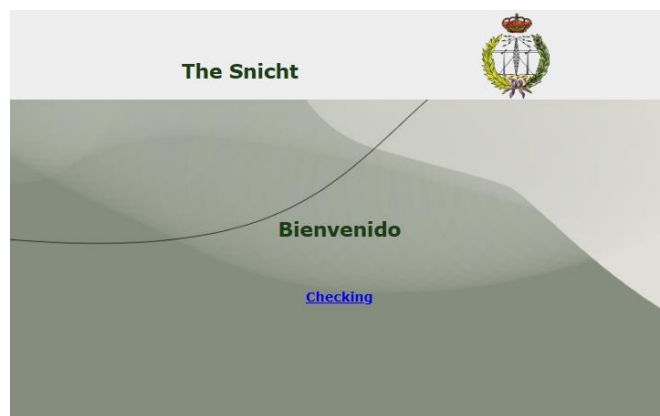
```
typedef struct{
    uint8_t sec;
    uint8_t min;
    uint8_t hour;
    uint8_t day;
    uint8_t month;
    uint8_t year;
} mytime_t;
```

3.2.8 Servidor:

Ficheros: *srv.c* y *srv.h*

Descripción

Este módulo se encarga de configurar un servidor que se lanzara mediante la tarjeta stm32, en el servidor se mostraran las entradas registradas en la tarjeta ft



Hora y Fecha	Nombre	Identificacion	Acceso
12:11:29 12/02/2024	Claudia	968168729	denegado
12:45:21 12/02/2024	Roberto	665479236	permitido
14:22:43 12/02/2024	--	--	desconocido
13:41:45 12/02/2024	Rosana	968168729	permitido

Figura 31. Visualización Servidor

Como se puede comprobar en la imagen, cada entrada constara de los siguientes campos:

- **Hora y fecha:** contendrá la hora y fecha del momento en el que se complete el acceso, ya de permitido, denegado o desconocido.
- **Nombre:** nombre del usuario que ha solicitado el acceso.
- **Identificación:** número de serie de la tarjeta de acceso, este número es único para cada tarjeta.
- **Acceso:** respuesta final a la solicitud de la entrada, esta puede tomar los siguientes valores:
 - *Permitido:* la solicitud ha resultado exitosa y se le permite el acceso registrando todos los datos.
 - *Denegado:* la solicitud ha sido denegada ya sea porque el número de serie de la tarjeta no esté registrada o porque se haya introducido la contraseña mal los tres intentos.
 - *Desconocido:* una persona que hay sido detectada por el sensor de movimiento no ha iniciado la solicitud de acceso pasando la tarjeta.

Ahora bien, dado que en el modo ahorro no se encuentra el servidor activo tan solo mostrara un mensaje por pantalla indicando que el servidor no se encuentra disponible.

Para poder llevar a cabo esta tarea el módulo realiza las siguientes operaciones:

En primer lugar, se extrae de la cola (nombres de la cola) una variable de tipo (nombre del tipo) con la siguiente estructura:

```
typedef struct{
    char adtos[50][20];
    uint8_t standBy;
} MSGQUEUE_OBJ_SRV;
```

El campo *adtos* contiene todas las palabras a representar mientras que el campo *standBy* indica si el dispositivo se encuentra en modo ahorro o no.

Una vez extraído el mensaje de la cola el primer campo que se examina es el *standBy*, que dependiendo de su valor se ejecutará una de las siguientes funciones.

Si la variable *standBy* tiene un valor de 0 se ejecutara la función *asignacion()*, la cual ira extrayendo y asignando a las listas correspondientes de *nombre*, *fechaHora*, *identificacion* y *tipoAcceso*, además se vaciará el mensaje de información que contiene el aviso de que el servidor no se encuentra disponible.

En cambio, si el campo *standBy* se encuentra a 1, se vaciarán todas las listas y se configurará el mensaje de información.

Por último, se encuentra la función *netCGI_Script ()* que le proporcionara la información necesaria de cada entrada al *checking.cgi*, cada case del segundo switch se corresponde con una entrada independiente.

Configuración

La configuración del diseño de la página web se encuentra en los ficheros *intex.htm* *checking.cgi* y *checking.cgx*, además, para hacer uso del servidor hay que configurar el fichero *Net_Config_ETH_0.h* que tiene la siguiente configuración:

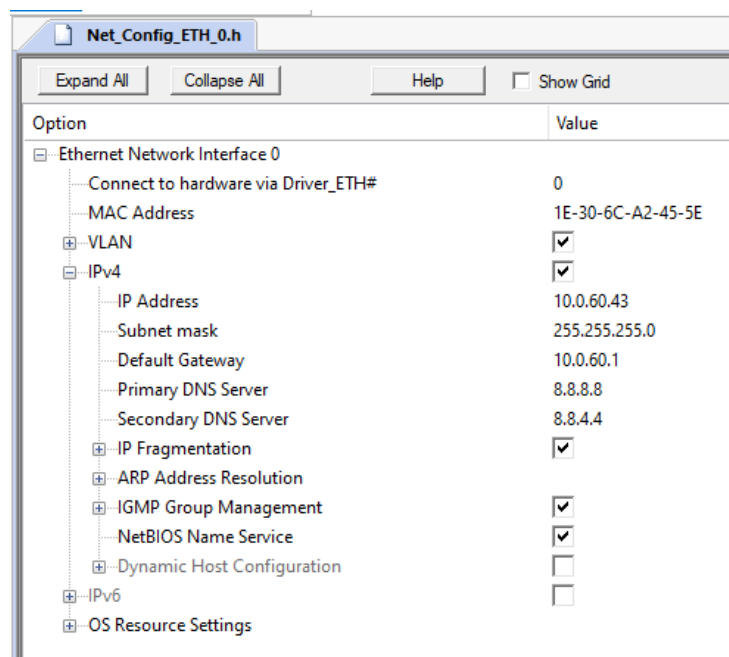


Figura 32. Configuración Net_Config_ETH_0.h

Entradas

Mensaje de la cola (nombre de la cola)

Salidas

Este módulo no cuenta con salidas externas

3.3 Pruebas realizadas software.

1º) Comprobar NFC:

Enviar señales al hilo NFC para activar la comunicación, esperar un segundo, y luego intentar recibir un mensaje del módulo NFC.

- **Resultados esperados:**
Si se recibe un mensaje, imprime su ID; de lo contrario, imprime "TO" indicando que se produjo un timeout al esperar el mensaje.
- **Resultados obtenidos:**
Éxito.

2º) Comprobar teclado:

Comprobar la correcta activación de cada una de las teclas.

- **Resultados esperados:**
El hilo se debe de quedar esperando a que reciba un flag por haberse pulsado una tecla y una vez pulsada se almacena en una cola de mensajes
- **Resultados obtenidos:**
Éxito.

3º) Comprobar LCD:

Encender la pantalla durante un registro de autenticación mostrando los mensajes necesarios.

- **Resultados esperados:**
Mensaje en espera de autenticación.
Mensaje acceso permitido.
Mensaje acceso incorrecto.
Mensaje acceso denegado.
- **Resultados obtenidos:**
Éxito.

4º) Comprobar RGB

Encender el diodo led con distintas tonalidades dependiendo del uso de la aplicación.

- **Resultados esperados:**
Azul enviado al servidor.
Verde acceso permitido.
Rojo acceso denegado.
Amarillo acceso incorrecto
Morado se va a dormir (modo de bajo consumo).
Cian cuando se despierta el microprocesador.
- **Resultados obtenidos:**
Éxito.

5º) Comprobar zumbador:

- **Resultados esperados:**

Comprobar que se activa el sonido siempre que se activa una pulsación en el teclado matricial.

- **Resultados obtenidos:**

Éxito:

6º) Comprobar servidor:

Para poder verificar el comportamiento del módulo del servidor se creó un hilo de prueba el cual mete en la cola (nombre de la cola) los siguientes valores de prueba

(Imagen de los valores de prueba)

- **Resultados esperados:**

Mostrar en la página cada entrada

7º) Comprobar RTC:

Comprobar que el sistema actualiza la hora cada segundo

- **Resultados esperados:**

La hora debe de seguir contando en todo momento y se debe de reiniciar si pasamos al modo de bajo consumo.

- **Resultados obtenidos:**

Éxito.

8º) Comprobar Tarjeta SD:

Comprobar que se almacenan y se leen mensajes en la SD

- **Resultados esperados:**

Se almacenan y se leen mensajes en la SD

- **Resultados obtenidos:**

Individualmente el módulo funciona, pero no funciona a la hora de integrarlo.

3.4 Pruebas realizadas hardware.

1º) Comprobar circuito de battery back-up:

Comprobar que al pasar de alimentación por la red a alimentación por batería sigue funcionando el sistema.

- **Resultados esperados:**

Se mantiene la alimentación de la STM y los módulos.
Se mantiene la alimentación del medidor de consumo.
No se producen cortes en el sistema.

- **Resultado:**

Éxito.

2º) Comprobar circuito de detección de conmutación:

Comprobar que se despierta el micro si vuelve la alimentación por la red o si se detecta movimiento.

- **Resultados esperados:**

El microcontrolador sale del modo sleep y también se activa la funcionalidad del registro de las personas cuando son detectadas.

- **Resultado:**

Éxito.

3º) Comprobar medidor de consumo:

Comprobar el consumo de la aplicación en diferentes casos de funcionamiento.

- **Resultados esperados:**

Consumo del circuito total de alimentación montado en PCB sin conectar nada: 90mA.

Consumo de la STM: 100mA.

Consumo de los sensores: 120mA.

- **Resultado:**

Salida del medidor de consumo:

4º) Comprobar circuito de alimentación simétrica:

Comprobar que genera la alimentación simétrica suficiente para alimentar los amplificadores operacionales del amplificador de instrumentación del medidor de consumo.

- **Resultados esperados:**

Obtener +-10V que entran dentro del rango de alimentación de los operacionales.

- **Resultado:**

Éxito.

4 CONCLUSIONES.

Este proyecto ofrece un sistema efectivo para controlar el acceso en áreas de alto riesgo al garantizar la autenticación doble de las personas. Esto se logra mediante el uso combinado de tarjetas NFC y un teclado numérico para ingresar un código. Además, el sistema asegura la continuidad operativa incluso en caso de cortes de energía eléctrica, manteniendo la seguridad del acceso a pesar de la falta de alimentación de la red eléctrica.

Además de esto, el sistema registra la información de acceso de los usuarios en una memoria externa y permite su visualización a través de un servidor en un ordenador. Esta función facilita la gestión y el seguimiento de los registros de acceso de manera conveniente y remota.

Por último, también nos aseguramos de que el sistema no utilice todos los recursos en todo momento, optimizando así su eficiencia energética y su rendimiento general.

5 PRESUPUESTO FINAL.

Componente	Precio
Keypad Lilypad	3'20 €
Buzzer (10)	2'03 €
Lector de tarjetas RFID MRFC522	2'57 €
Cámara OV7690	1'77 €
LCD 2004	1'5 €
Módulo expansor PCF8574	3'50 €
LED RGB	0'20 €
Sensor PIR HC SR501	1'15 €
Modulo BMS 40A-4S	1'14€
Componentes electrónicos (Estimación)	8 €
PCB	35 €
Adaptador 230V/24V-1A	9,32 €
4 células 18650	17,05 €
Núcleo-F429ZI	30'25 €
TOTAL	116,98 €

6 EQUIPO DE TRABAJO.

Se debe completar una tabla en la que se incluyan los miembros del equipo de trabajo, las tareas asignadas inicialmente, las tareas realmente desarrolladas y el tiempo utilizado en las mismas.

También deben incluirse las tareas relacionadas con la integración y los tests o pruebas realizadas.

Miembros del equipo	Tareas iniciales	Tareas finales	Horas
Luis González-Gallego Sosa	Analógica + PCB	Diseño Analógica + PCB +Test Analógica + Programación de bajo consumo	148
Luis López Izquierdo	Cámara + Teclado + TarjetaSD + RTC	Teclado + TarjetaSD + Programación de bajo consumo + RTC	146
Pablo Martínez Lafarga	Buzzer + NFC + RGB + LCD	Buzzer + NFC + RGB + LCD + Integración + Principal	186
Sara Jiménez Muñoz	Servidor	Servidor + Integración	139

7 ACRÓNIMOS UTILIZADOS.

Identifique los acrónimos usados en su documento.

UART	Universal Asynchronous Receiver Transmitter
SPI	Serial Peripheral Interface
I2C	Inter-Integrated Circuit
RGB	Red-Green-Blue
RTC	Real Time Clock
PIR	Passive Infrared
NFC	Near Field Communication
RFID	Radio Frecuency ID
BMS	Battery Manage Sistem
PCB	Printed Circuit Board
SNTP	Simple Network Protocol

8 BIBLIOGRAFÍA UTILIZADA.

- Catálogo integrado amplificadores operacionales: <https://pdf1.alldatasheet.com/datasheet-pdf/view/545518/TI/LMC6484IN.html>
- Catálogo de los reguladores: <https://pdf1.alldatasheet.com/datasheet-pdf/view/105716/FAIRCHILD/LM7818.html>
- Núcleo STM32F429ZI: https://www.st.com/resource/en/user_manual/um1974-stm32-nucleo144-boards-mb1137-stmicroelectronics.pdf
- Catálogo integrado conversor DC-DC inversor: <https://pdf1.alldatasheet.es/datasheet-pdf/view/532554/INTERSIL/ICL7660.html>
- Catálogo baterías ion-litio: <https://www.ineltro.ch/media/downloads/SAAItem/45/45958/36e3e7f3-2049-4adb-a2a7-79c654d92915.pdf>
- Catálogo módulo BMS: https://www.mantech.co.za/datasheets/products/BMS-40A-4S_SGT.pdf
- Catálogo sensor de movimiento: <https://www.mpja.com/download/31227sc.pdf>
- Catálogo sensor NFC: https://github.com/PabloMalf/ISE/blob/main/B2/Docu/nfc_RC522.pdf
- Catálogo módulo expansor del LCD: https://github.com/PabloMalf/ISE/blob/main/B2/Docu/lcd_PCF8574.pdf
- Catálogo pantalla LCD https://github.com/PabloMalf/ISE/blob/main/B2/Docu/lcd_HD44780U.pdf
- Catálogo keypad: <https://github.com/PabloMalf/ISE/blob/main/B2/Docu/keypad.pdf>
- Catálogo cámara: https://github.com/PabloMalf/ISE/blob/main/B2/Docu/cam_OV7670.pdf
- Página oficial ARM Keil: <https://www.keil.com/pack/doc/mw/Network/html/index.html>