UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA EMPOTRADO TOLERANTE A FALLOS PARA EL CONTROL DE CONDICIONES METEOROLÓGICAS

AUTOR: PABLO MARTÍN TABARES

TUTOR: JOAQUÍN GRACIA MORÁN

COTUTOR: LUIS JOSÉ SAIZ ADALID

Curso Académico: 2020-21

Resumen

En el presente Trabajo de Fin de Grado se ha realizado el estudio de un sistema empotrado formado por la tarjeta `STM32F429i-DISCOVERY`, un sensor de temperatura y humedad y un actuador LED.

Se presentan dos objetivos principales:

Por un lado, implementar un sistema de control con el que tener unos valores concretos de temperatura y humedad. Este sistema sirve de modelo para el control de un cultivo interior ecológico. La aplicación que se ha llevado a cabo es sencilla pero tiene mucha trascendencia para la futura automatización de los invernaderos.

Por otro lado, el otro objetivo principal es la implementación de un `Código de corrección de errores(ECC)` el cual garantiza la reducción de errores de los datos almacenados en memoria. Dentro de este apartado, la principal novedad es el entrelazado de distintas matrices para formar un código con mayor alcance de corrección y la utilización de distintas coberturas de error para comparar la sobrecarga introducida en cada caso. El ECC proporciona una mejora en la tolerancia a fallos y la confiabilidad del sistema.

Finalmente, se han analizado los resultados de los distintos códigos implementados, se han comparado los tiempos de ejecución y el tamaño de los mismos y se ha constatado que el sistema de control funciona correctamente.

Palabras clave: Sistema Empotrado, STM32F429i-DISCOVERY, Sensor y Actuador, Códigos de Corrección de Errores, Microcontroladores.

Abstract

In this Final Degree Project, the study of a built-in system has been carried out, formed by the `STM32F429i-DISCOVERY` card, a temperature and humidity sensor and a LED actuator.

Two main objectives are presented:

Firstly, a control system to measure specific values of temperature and humidity has been implemented. The system can model the control of an ecological indoor cultivation. The application that has been carried out is simple but it has a great value for the future automation of greenhouses.

Secondly, the other main objective is the implementation of an `Error Correction Code (ECC)` which guarantees the reduction of errors in the data stored in the memory. Within this section, the biggest novelty is the interlacing of different arrays to form a code with a greater scope of correction. The ECC provides improved fault tolerance and system reliability.

Finally, the results of the different implemented codes have been analyzed, their execution times and sizes have been compared and it has been verified that the control system works correctly.

Key words: Embedded Systems, STM32F429i-DISCOVERY, Sensor and Actuator, Error Correction Code, Fault Tolerance, Reliability, Microcontrollers.

Resum

En el present Treball de Fi de Grau s'ha realitzat l'estudi d'un sistema encastat format per la targeta STM32F429i-DISCOVERY, un sensor de temperatura i humitat i un actuador LED.

Es presenten dos objectius principals:

Per un costat, realitzar un sistema de control per a tindre uns valors concrets de temperatura i humitat. El sistema ha sigut desenvolupat en vistes a controlar un cultiu interior ecològic. L'aplicació que s'ha portat a cap es senzilla però té molta transcendència per a la futura automatització dels hivernacles.

Per un altre costat, l'altre objectiu principal és la implementació d'un Codi de correcció d'errors (ECC) el qual garanteix la reducció d'errors de les dades emmagatzemades en memòria. Dintre d'aquest apartat, la principal novetat és l'entramat de distintes matrius per a formar un codi amb un major abast de correcció. L'ECC proporciona una millora en la tolerància a errades i la fiabilitat de el sistema.

Finalment, s'han analitzat els resultats dels distints codis implementats, s'han comparat els temps d'execució i pes dels mateixos, i s'ha constatat que el sistema de control funciona correctament.

Paraules clau: Sistema Encastat, STM32F429i-DISCOVERY, Sensor i Actuador, Codi de Correcció d'Errors , Microcontroladors.

Agradecimientos

En primer lugar, quiero agradecer a mi madre y padre que me permitieron comenzar mi carrera universitaria en Valencia. Agradezco tanto su esfuerzo económico como emocional ya que no es fácil dejar marchar a un hijo aún sabiendo que es lo mejor para el.

A mi hermano Daniel, que siempre me ha motivado para seguir aprendiendo, formándome y no darme por vencido.

Al resto de mi familia que siempre han supuesto un apoyo a lo largo de los años. Mención especial a mi abuela Mita que sin sus velas no habría aprobado ni la mitad de los exámenes.

Todos los amigos que he hecho en Valencia que desde el principio me hicieron sentir como en casa. De todos ellos me llevo un gran recuerdo a donde quiera que me lleve el destino.

A Juan, mi compañero de batallas, mi Sancho Panza, mi amigo... gracias por todo, sin ti esto no hubiera sido posible.

A mis tutores Joaquín y Luis José sin los cuales este proyecto no se podría haber llevado a cabo.

Quiero mencionar a mi tío Jose y mi perro Yanko que estuvieron en el comienzo pero no pudieron presenciar el final. A mi abuela Susana y abuelo Adolfo.

Índice general

Índice de figuras

Índice de tablas

Siglas

CMOS Complementary Metal-Oxide Semiconductor. 5

DMA Direct Memory Access. 35

dps Degrees per Second. 6

FMC Flexible Memory Controller. 37

HSE High Speed External Clock Signal. 32

IC Integrated Circuit. 6, 11

JTAG Joint Test Action Group. 5

LSI Low Speed Internal Clock Signal. 32

NRE Non-recoverable Error. 26

NRST Natural Resources Science and Technology. 6

NTC Negative Temperature Coefficient. 10

OTG On The Go. 6

PCB Printed Circuit Board. 11

PHY Physical Layer. 6

PWM Pulse Width Modulation. 34

SDRAM Synchronous dynamic random access-memory. 5

SEC-DAEC Single Error Correction and Double Adjacent Error Correction. 3

SEC-DED Single Error Correction and Double Error Detection. 3

SPI Serial Peripheral Interface. 6, 39

SWD Serial Wire Debug. 5

VLSI Very Large-Scale Integration. 18

Capítulo 1

Introducción

El presente documento relata el Trabajo de Fin de Grado `Desarrollo e implementación de un sistema empotrado tolerante a fallos para el control de condiciones meteorológicas`.

Este proyecto se centra en la ventaja del uso de Códigos de Corrección de Errores (ECC, del inglés Error Correction Codes) en memorias. En concreto se hace uso de un `Código ultrarrápido para corregir múltiples errores adyacentes y detectar dobles errores`. Además de la implementación del ECC y su comprobación con inyección de fallos, también se desarrolla el sistema empotrado formado por un sensor de humedad y temperatura, un actuador LED y un microcontrolador STM32. A continuación, se resume la estructura de la presente memoria:

- **Capítulo 1:** se lleva a cabo una descripción detallada de las diferentes partes del sistema empotrado así como de los programas que se utilizan. La comprensión de algunos de los componentes, como el DHT22 o el ECC, es más importante que otros para la correcta apreciación del trabajo y por tanto, se dedica una explicación más profunda y detallada.
- **Capítulo 2:** se procede con la metodología la cuál incluye la inicialización de los periféricos con la herramienta STM32Cube-Mx, la configuración del Cube-IDE y la escritura del código, la muestra por pantalla de los resultados y la implementación del ECC. Por último, se realiza el debug del proyecto.
- **Capítulo 3:** se analizan los resultados obtenidos tras aplicar a cada uno de los códigos varias inyecciones de fallos. A su vez se comprueba el tamaño de archivo y el tiempo de ejecución de cada uno de los códigos propuestos y se verifica que el sistema de control cumple con los objetivos deseados.

- **Capítulo 4:** se desarrolla la conclusión del proyecto con los aspectos favorables y desfavorables del mismo. Se finaliza con las posibles aplicaciones futuras de este proyecto.

1.1. Objetivos

El objetivo de este trabajo es conseguir un sistema confiable para pequeños huertos dentro de hogares donde se requieran unas condiciones concretas de temperatura y humedad. El sistema es bastante simple pero otorga las herramientas al usuario para añadir más actuadores y sensores con el consiguiente aumento del rendimiento y la automatización de la plantación. La idea es implementar un sistema de control sencillo al cual se le pueden aplicar diferentes códigos de corrección de errores para estudiar sus prestaciones.

Este trabajo se ha divido en dos objetivos principales:

- Por una parte, el control del sistema empotrado formado por la tarjeta `STM32F429i-DISCOVERY`, el sensor `DHT-22` y un actuador `LED-RGB`.
- Por otro lado, la protección de la memoria del sistema frente a errores haciendo uso de varios códigos de corrección de errores ultrarrápidos.

1.2. Motivación

Durante la cuarentena provocada por el COVID seguí muy de cerca todo tipo de iniciativas fomentadoras del consumo propio. Desde siempre me ha parecido una idea brillante, la cual nunca he tenido la oportunidad de conocer ni llevar a cabo. Si bien mis conocimientos sobre agricultura son bastante escasos, me gusta aportar soluciones desde el lado de la ingeniería a todos los problemas relacionados con el cambio climático y considero que esta es una buena manera de empezar. A su vez, durante los estudios de grado he visto contadas asignaturas enfocadas a programas informáticos y programación y me parecen fundamentales para cualquier ingeniero que se precie, por tanto, he querido indagar un poco más en este aspecto y me he encontrado con la oportunidad perfecta.

1.3. Antecedentes

Desde que se crearon los primeros invernaderos en 1850, los agricultores siempre han buscado una manera de aumentar la eficiencia de los mismos para obtener los productos en el mejor estado y menor tiempo posible. Con la aparición de los primeros microcontroladores, estos se pudieron utilizar para incrementar la eficiencia. Sin embargo, estos eran muy caros y muy complicados de usar. Hoy en día con los nuevos y sofisticados microcontroladores que existen, podemos controlar innumerable cantidad de aplicaciones de una manera bastante más sencilla (aunque siempre requiere algo de conocimiento de la materia) y a un coste muchísimo más bajo.

Por otro lado, los ECC han cambiado mucho desde que Hamming creó su famoso código de corrección. Los códigos de Hamming pueden detectar hasta 2 bits erróneos y corregir 1 bit erróneo. Con el paso de los años la tecnología ha avanzado y con ella se han desarrollado aplicaciones que requieren una mayor confiabilidad. Los códigos de Hamming se han quedado obsoletos para muchos de estos procesos y por tanto, se han tenido que modificar/mejorar para llegar a cumplir la confiabilidad esperada. Entre ellos se pueden encontrar: Hamming extendido, SEC-DED, SEC-DAEC, Ultrafast SEC-xAEC-DED ... todos ellos con mejorías notables con respecto a los códigos de Hamming originales. Aún así, todos presentan inconvenientes, por tanto, es de vital importancia saber bien cual es nuestro propósito para luego escoger el que mejor se adapte a nuestro proyecto.

Capítulo 2

Fundamentación teórica

En el capítulo 2, se ha llevado a cabo un análisis teórico de cada uno de los componentes del sistema empotrado, así como de los programas a utilizar. El funcionamiento del sensor no es muy complicado pero es necesario entender su funcionamiento para situarlo en el lugar más apropiado y así evitar mediciones innecesarias o incluso inútiles.

Por otra parte, los programas que se han utilizado presentan una innumerable cantidad de aplicaciones de las cuáles se han visto las más importantes enfocadas al proyecto. Los aspectos prácticos de estos programas se verán en la metodología del proyecto.

Por último, es fundamental entender o por lo menos, tener una idea de qué es y cómo funciona un `ECC` para poder luego implementarlo en el código de la manera menos tediosa y más efectiva posible. Esta parte es la más difícil de comprender para las personas sin conocimientos en informática y por tanto, se ha intentado resumir y concretar las ideas de una forma lo más clara posible.

2.1. Tarjeta STM32F429i-DISCOVERY

Para que un proyecto de sistemas empotrados triunfe necesita apoyarse en una base sólida y fiable que le permita cumplir los objetivos que se quieren llevar a cabo.

La tarjeta `STM32F429i-DISCOVERY` es un kit de desarrollo creado por la empresa ST. La empresa ST es una de las empresas punteras en desarrollo de circuitos integrados, microcontroladores, chips de tarjetas inteligentes, microprocesadores... todos ellos con componentes de gran calidad. Fue la primera empresa en introducir un kit de desarrollo (`STM32F407VGT6`) con la arquitectura ARM Cortex-M4 [**1**] lo cual supuso un gran avance en las aplicaciones de los microcontroladores.

En nuestro caso, hemos optado por un kit más completo y potente. La tarjeta `STM32F429i-DISCOVERY` cuenta, entre otras muchas prestaciones, con [**1**]:

- **Arquitectura `ARM-Cortex M4`** Esta es una de las características más importantes de esta tarjeta y por tanto, será explicada con mayor detalle en la sección siguiente.
- **Herramienta de depuración integrada `ST-Link/V2-B`** La herramienta `ST-Link/V2-B` está empotrada en la tarjeta donde actúa como una herramienta de programación y depuración.

  Esta herramienta se puede encontrar tanto en tarjetas STM8 como en STM32, sin embargo, esta última cuenta con SWD, una alternativa al famoso JTAG. Una de las mejoras con respecto a la `ST-Link/V2` es que este modelo más avanzado cuenta con una interfaz virtual de puerto COM (USB) y una interfaz de almacenamiento de masa, lo que permite la transferencia de archivos entre el host y el dispositivo USB [**2**].

- **Pantalla de 2,4"`QVGA TFT LCD`** La pantalla LCD TFT es una pantalla de 2,4"de 262K colores. Su definición es `QVGA (240 x 320)` y es impulsada directamente por el `STM32F429ZIT6` usando el protocolo RGB. Además incluye el controlador `LCD ILI9341`.

- **`SDRAM` de 64M-bit externa** La SDRAM de 64 Mbit es una CMOS de alta velocidad, diseñada para operar en sistemas de memoria de 3,3V que contienen 67.108.864 bits. Está configurado internamente como una DRAM de cuatro bancos con una interfaz síncrona. Cada banco de 16.777.216 bits está organizado como 4.096 filas por 256 columnas por 16 bits. La SDRAM de 64 Mbit incluye modos de actualización automática, ahorro de energía y apagado. Todas las señales se registran en el borde positivo de la señal de reloj, CLK.

- **Giroscopio**

  El `I3G4250D` es un sensor de velocidad angular de tres ejes de baja potencia. Incluye un elemento sensor y una interfaz IC capaz de proporcionar la tasa angular medida al mundo externo a través de la interfaz serie I2C / SPI. El `I3G4250D` tiene una escala completa de ± 245 / ± 500 / ± 2000 dps y es capaz de medir tasas con un ancho de banda seleccionable por el usuario. El `STM32F429ZIT6` controla este sensor de movimiento a través de la interfaz SPI.

- **USB OTG**

  El `STM32F429ZIT6` presenta USB OTG de alta velocidad a través de su `PHY` interno. El conector USB Micro-AB (CN6) permite al usuario conectar un host o componente del dispositivo, como una llave USB, un ratón u otro.

  Hay dos LED dedicados a este módulo:

  - LD5 (LED verde) indica cuando VBUS está activo.
  - LD6 (LED rojo) indica una sobrecorriente de un dispositivo conectado.

- **LEDS**

  - LD1 COM: El estado predeterminado de LD1 es rojo. LD1 cambia a verde para indicar que hay comunicaciones en curso entre el PC y el ST-LINK/V2-B.
  - LD2 PWR: El LED rojo indica que la placa está encendida.
  - Usuario LD3: es un LED de usuario conectado a la E/S PG13 del `STM32F429ZIT6`.
  - Usuario LD4: El LED rojo es un LED de usuario conectado a la E/S PG14 del `STM32F429ZIT6`.
  - Usuario LD5: El LED verde indica cuando VBUS está presente en CN6 y está conectado a PB13 del `STM32F429ZIT6`.
  - Usuario LD6: El LED rojo indica una sobrecorriente del VBUS de CN6 y mientras está conectado a la E/S PC5 del `STM32F429ZIT6` [**1**].

- **Pulsadores**

  - Usuario B1: botón de usuario y despertador conectado a la E/S PA0.
  - Reinicio B2: el pulsador conectado a NRST es usado para reiniciar la tarjeta [**1**].

2.1.1. ARM Cortex-M4

La familia ARM Cortex-M son un conjunto de núcleos de microprocesadores ARM diseñados para usarse en microcontroladores **ASICs, ASSPs, FPGAs, SoCs**. Los núcleos Cortex-M son usados normalmente como chips microcontroladores pero también como E/S de controladores, controladores de sistema, de pantalla táctil,de baterías inteligentes, de sensores...[**3**].

En concreto, el Cortex-M4 está desarrollado para llevar a cabo el control de señales digitales con alta eficiencia, baja potencia, bajo coste y un uso sencillo. Es por ello, que la familia M4 está tan bien posicionada en el mercado, siendo uno de los mejores microprocesadores en cuanto a relación calidad/precio [**4**].

Feature	M0	M1	M3	M4	M7
Instruction set	Armv6-M	Armv6-M	Armv7-M	Armv7-M	Armv7-M
Digital signal processing	No	No	No	Yes	Yes
DMIPS/MHz*	0.87	0.8	1.25	1.25	2.14
CoreMark/MHz*	2.33	1.85	3.34	3.42	5.01
External interrupts	32	32	240	240	240
Bus protocol	AHB lite	AHB lite	AHB lite	AHB lite	AXI
Data cache	No	No	No	No	0-64kB
Dual Core Lock-Step	No	No	No	No	0-16MB

Tabla 2.1: Tabla comparativa de los `Cortex-M` [**4**]

El `Cortex-M4` incluye [**5**]:

- **Núcleo del procesador**.
- **Controlador de interrupción vectorial anidado (NVIC)** que está estrechamente integrado con el núcleo del procesador para lograr un procesamiento de interrupciones de baja latencia.
- **Múltiples interfaces de bus de alto rendimiento**.
- **Solución de depuración de bajo costo** con la capacidad opcional de:
  - Implementar puntos de interrupción y parches de código.
  - Implementar puntos de vigilancia, rastreo y creación de perfiles del sistema.
  - Admite la depuración de estilo printf.

- **Unidad de protección de memoria (MPU) opcional.**
- **Procesamiento de operaciones en coma flotante (FPU).**

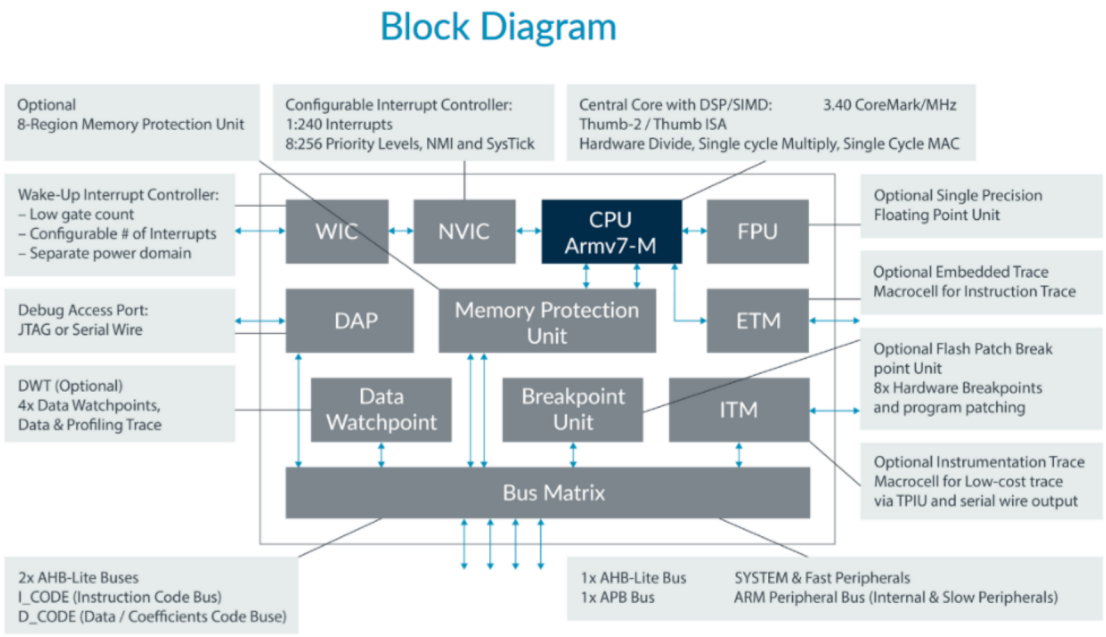


Figura 2.1: Componentes del Cortex-M4 [**5**]

Además, el Cortex-M4 incluye el temporizador Systick. Esta herramienta sirve para crear interrupciones precisas en diferentes tareas, crear retrasos o medir tiempos de ejecución. Este periférico usa un contador descendente (resta 1 por evento) con un registro de 24 bits. Cuando la cuenta está en 0 y llega un nuevo evento, el registro de cuenta se recarga con un valor de "precarga" establecido por programa y sigue descontando a partir de ese valor. El ritmo de descuento se deriva del reloj del sistema [**4**].

2.2. Sensor DHT22

El sensor DHT22 es un transductor capaz de medir la humedad y la temperatura del ambiente gracias a que presenta un sensor capacitivo (humedad) y un termoresistor (temperatura). Las prestaciones que ofrece este sensor en cuanto a calidad/precio son de las mejores del mercado y es por ello que se ha apostado por él en este trabajo.

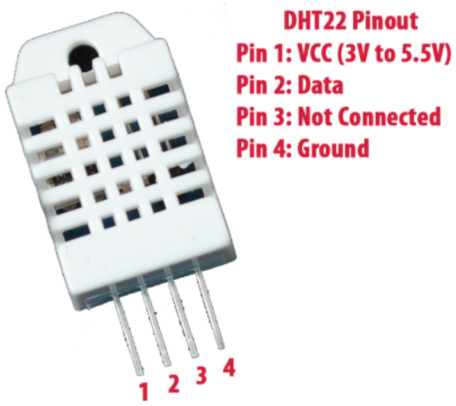


Figura 2.2: Sensor DHT22

Es un sensor muy versátil, que se puede usar tanto en estaciones meteorológicas como en invernaderos o en museos, donde se requiere una humedad y temperatura concretas para preservar las obras de arte, así como en salas de música donde la humedad y la temperatura influyen notablemente en los instrumentos. La idea de nuestro proyecto es crear un invernadero monitorizado y este sensor es fundamental para lograrlo.

Este sensor tiene un rango de mediciones del 0 % al 100 % de humedad relativa y de -40ºC hasta 125ºC. Si bien, estos rangos de temperatura no son soportados por casi ningún circuito electrónico, estas mediciones son unas muy buenas garantías por parte del sensor. La precisión de la temperatura es de $\pm$ 0.2ºC y la de la humedad de $\pm$ 5 % .Unos rangos más que aceptables [**6**].

El componente sensor de humedad tiene dos electrodos con un sustrato de retención de humedad (generalmente una sal o un polímero plástico conductor) intercalados entre ellos. Los iones son liberados por el sustrato a medida que absorbe el vapor de agua, lo que a su vez, aumenta la conductividad entre los electrodos. El cambio de resistencia entre los dos electrodos es proporcional a la humedad relativa. Una humedad relativa más alta disminuye la resistencia entre los electrodos, mientras que la humedad relativa más baja aumenta la resistencia entre los electrodos [7].

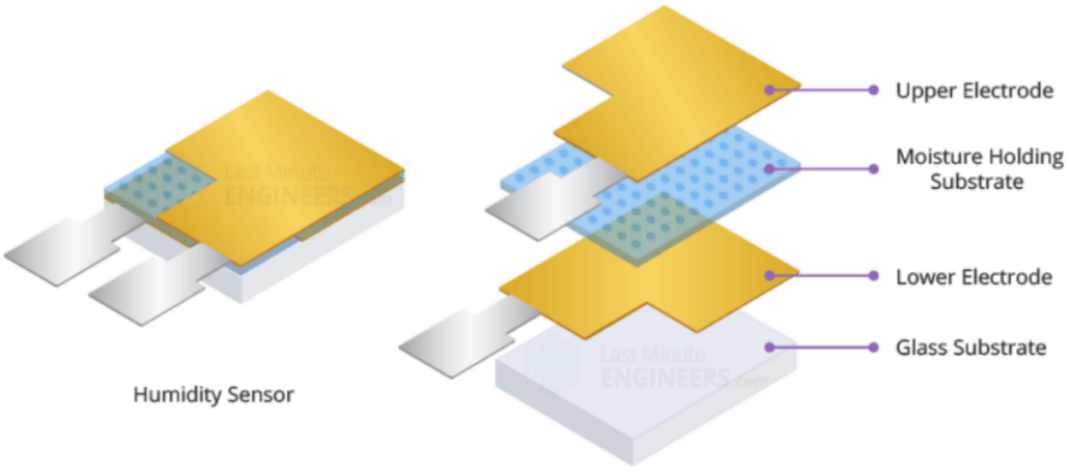


Figura 2.3: Estructura interna del DHT22 [7]

Además, consta de un sensor Termistor NTC para medir la temperatura. Un termistor es una resistencia térmica, es decir, cambia su resistencia con la temperatura.

Los termistores están hechos para que la resistencia cambie drásticamente con la temperatura, de modo que el cambio pueda ser de 100 ohmios o más por grado. El término "NTC" significa "Coeficiente de temperatura negativo", lo que implica que la resistencia disminuye con el aumento de la temperatura.

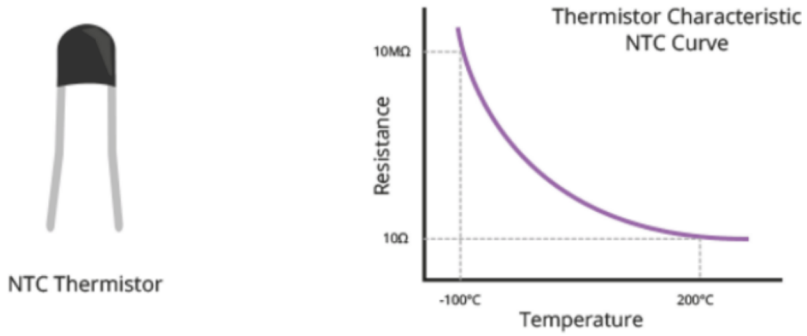


Figura 2.4: Curva característica del termistor NTC [7]

Además, hay una pequeña PCB con un IC empaquetado SOIC-14 de 8 bits. Este IC mide y procesa la señal analógica con los coeficientes de calibración almacenados, realiza la conversión de analógico a digital y manda una señal digital con la temperatura y la humedad [7].

Inicialización

La línea negra representa la señal del microcontrolador y la línea blanca la del DHT22. Para inicializar el sensor, se tiene que poner la línea de datos baja (GND) durante al menos $1ms$ y luego ponerla en alta (VCC) durante unos 20-40μs. Al recibir la señal de inicio, el DHT22 indicará su presencia tirando de la línea hacia abajo durante 80μs y luego hacia arriba otros 80μs.

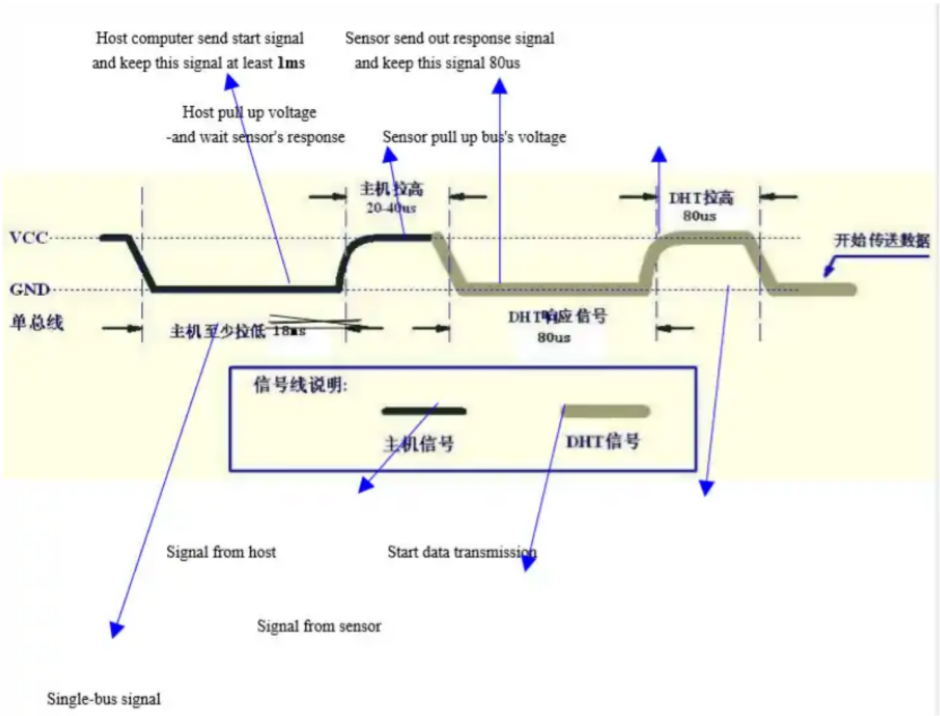


Figura 2.5: Inicialización del sensor DHT22 [8]

Los pasos para inicializar el sensor son los siguientes:

- Fijar el pin de datos como salida (output).
- Poner el pin bajo y esperar durante $\geq 1ms$.
- Poner el pin alto y esperar durante $30\mu s$.
- Por último, fijar el pin como entrada (input).

Transmisión de datos

Ahora el DHT22 envía 40 bits de datos. La transmisión de cada bit comienza con un nivel de voltaje bajo que dura $50\mu s$. La siguiente longitud de la señal a nivel alto decide si el bit es "1" o "0".

- Si la longitud del nivel de alto voltaje es alrededor de 26-28μs, el bit es "0".
- Si la longitud es de alrededor de 70μs, entonces el bit es "1".

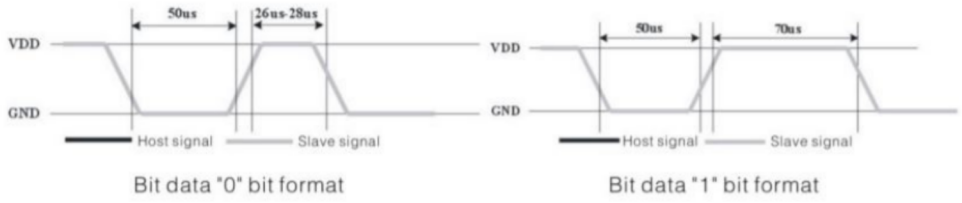


Figura 2.6: Transmisión de datos del DHT22 [8]

Los 40 bits enviados por DHT22 son los siguientes:

- **Datos** = RH enteros de 8 bits + RH decimales de 8 bits + T enteros de 8 bits + T decimales de 8 bits + suma de comprobación de 8 bits (check-sum).

Si la transmisión de datos es correcta, **check-sum** debe ser igual a los últimos 8 bits de "RH entero +RH decimal + T entero + T decimal" [8].

Lectura de datos

Para obtener los resultados se deben seguir los siguientes pasos:

- Esperar al pin a que esté a nivel alto.
- Esperar 40μs. Esto es debido a que, como se ha visto anteriormente, la duración cuando el bit es "0" es de 26-28μs y si el pin está a nivel alto después de estos 40μs entonces se sabe que el bit es "1".
- Por último, se escriben los respectivos valores en las variables.

2.3. LED RGB

Para completar el sistema de control se utiliza un LED RGB. Este indica si el sistema se encuentra en un estado óptimo, en cuanto a temperatura y humedad (LED azul activado), o si por el contrario, no se están cumpliendo las condiciones necesarias (LED rojo encendido).

Figura 2.7: LED RGB

Básicamente, cada color tiene un rango de intensidad de 0 a 255 que depende del voltaje proporcionado al pin respectivo. Se pueden combinar estas intensidades (0 a 255) de estos tres colores para producir 16 millones (256x256x256) de colores diferentes.

2.4. Entorno de programación

El ecosistema STM32Cube es una solución de software completa para microcontroladores y microprocesadores STM32. Está dirigido a usuarios que buscan un entorno de desarrollo completo y gratuito para STM32 en el que puedan integrar fácilmente los distintos componentes como STM32CubeMx, STM32CubeProgrammer o STM32CubeMonitor [**9**].

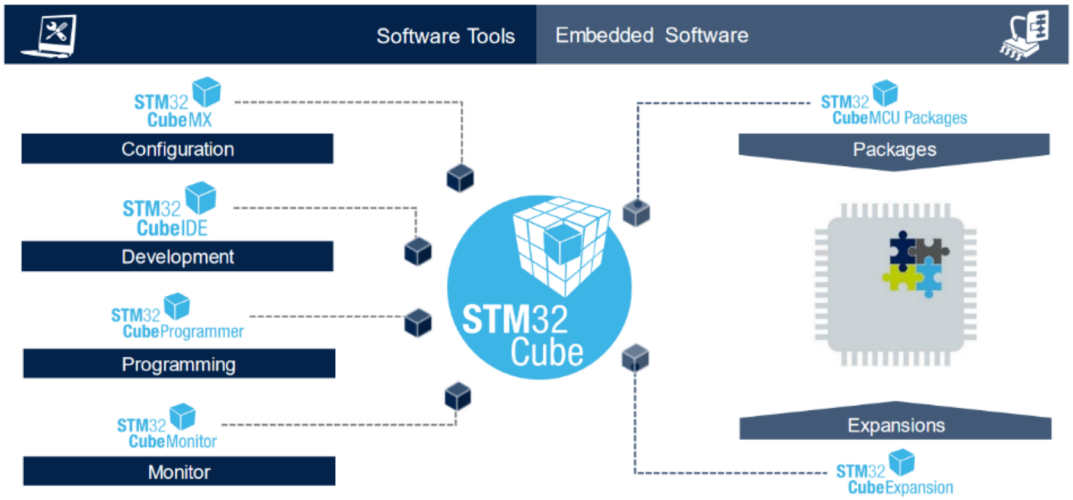


Figura 2.8: Ecosistema STM32Cube [**9**]

2.4.1. STM32Cube-Mx

STM32CubeMx es una herramienta gráfica que permite una configuración muy sencilla de microcontroladores y microprocesadores STM32, así como la generación del código C de inicialización correspondiente para el núcleo ARM Cortex-M.

Una de las grandes ventajas del CubeMx es la facilidad para configurar aspectos a priori bastante complicados como los relojes o el consumo del dispositivo. También incluye una gran cantidad de bibliotecas que ayudan al usuario a trabajar de manera más cómoda [**10**].

2.4.2. STM32Cube-IDE

En este proyecto se cuenta con otro de los pilares del entorno de programación de STM32Cube, el Cube-IDE.

Este software es una plataforma de desarrollo C/C++ avanzada con configuración de periféricos, generación de código, compilación de código y funciones de depuración para microcontroladores y microprocesadores STM32. Se basa en el marco Eclipse CDT y la cadena de herramientas GCC para el desarrollo y GDB para la depuración.

STM32CubeIDE trabaja conjuntamente con el STM32CubeMx ofreciendo una experiencia de herramienta todo en uno facilitando la ejecución de las tareas y ahorrando tiempo de instalación y desarrollo.

STM32CubeIDE incluye analizadores de compilación y pila que brindan al usuario información útil sobre el estado del proyecto y los requisitos de memoria. También incluye funciones de depuración estándar y avanzadas como vistas de los registros del núcleo de la CPU, las memorias y los registros de periféricos, así como la vigilancia de variables en vivo, la interfaz del visor de cables en serie o el analizador de fallos [**11**].

2.4.3. ST-LINK Utility

Esta herramienta es capaz de proporcionar un entorno eficiente y fácil de usar para leer, escribir y verificar un dispositivo de memoria. Ofrece una amplia gama de funciones para programar memorias internas STM32 (Flash, RAM, OTP y otras), memorias externas, verificar el contenido de la programación (suma de comprobación, verificar durante y después de la programación, comparar con archivo) y automatizar la programación STM32 [**12**].

2.5. Sistema de protección

2.5.1. Códigos de Corrección de Errores (ECCs)

Actualmente, se ha conseguido una gran capacidad de almacenamiento en los sistemas de memoria gracias a la continua reducción de tamaño de la tecnología CMOS. Sin embargo, la tasa de errores de la memoria también ha aumentado con esta disminución de tamaño [**13**][**14**]. De esta forma, se pueden generar fallos simples o múltiples con el impacto de una partícula de radiación cósmica [**15**].

Usualmente, los sistemas de memoria se han protegido mediante Códigos de Corrección de Errores (ECCs), siendo los más comunes los códigos SEC o SEC-DED [**16**][**17**][**18**]. Los primeros pueden corregir un error en una única celda de memoria, mientras que los segundos pueden corregir un error en una celda de memoria, o detectar dos errores en dos celdas independientes.

Los ECC están diseñados tradicionalmente para minimizar el número de bits redundantes, ya que se agregan a cada palabra en toda la memoria. Sin embargo, el uso de un ECC introduce latencias de codificación y decodificación, aumenta el área de silicio y el consumo de energía. En otras unidades informáticas, estos parámetros se han optimizado y la redundancia ha perdido importancia. Por ejemplo, la protección de los registros contra errores sigue siendo una preocupación importante para los sistemas submicrónicos profundos debido al escalado de la tecnología. En este caso, un requisito importante para la protección del registro es mantener las latencias de codificación y decodificación lo más cortas posible [**19**].

Introducción a los ECCs

Un sistema informático está formado por el hardware (parte física), el software (parte lógica) y la información que almacena y procesa. Ser capaz de mantener esta información de manera íntegra es de vital importancia para el correcto funcionamiento del sistema o por lo menos, para la fiabilidad de las operaciones llevadas a cabo.

Los ECCs se usan en aplicaciones de [**20**]:

- **Comunicación** : buses, I/O, transmisión de información...
- **Almacenamiento de información**: Registros, RAM, HDDs, CDs/DVDs...

Los códigos de corrección de errores (ECCs) permiten proteger la información transmitida por un canal poco fiable. Pueden ser de dos tipos:

- **Espacio**: sistema de buses, señales inalámbricas...
- **Tiempo**: elementos de almacenamiento.

Existen una gran cantidad de tipos de ECCs, por tanto, para llevar a cabo una buena elección hay que tener unos parámetros en consideración. El más importante es la cantidad de información que se va a transmitir. Se distinguen entre [**20**]:

- **Transmisiones masivas**: comunicación de satélites, CD/DVDs...
- **Microprocesadores y otros sistemas VLSI**, normalmente con pequeños tamaños de palabras (8,16,32,64...).

Antes de continuar se debe conocer la distinción entre fallos, errores y averías [**20**].

- **Fallo**: se define como la causa u origen probable de un error. Se trata de un defecto o imperfección en el hardware o en el software del sistema que al activarse provoca un error. Hay tres tipos de fallos:
  - ***Transitorios***: Este tipo de fallos están presentes en el sistema durante un tiempo limitado. Son generados por condiciones medioambientales temporales (por ejemplo, interferencias electromagnéticas, radiación cósmica, etc.). Son difíciles de detectar por su ocurrencia aleatoria. En el caso concreto de los elementos de almacenamiento (memoria y registros), el fallo que típicamente se produce es el bit-flip, en el que un bit cambia de 1→0 o de 0→1.
  - ***Permanentes***: Son fallos que una vez activados, permanecen en el sistema hasta que se reemplace el componente afectado. Están provocados por cambios irreversibles producidos por procesos de fabricación, o de funcionamiento del circuito (proceso de desgaste). En el caso concreto de los elementos de almacenamiento, el fallo típico es el Stuck-at, o fallo de pegado a, en el que el valor de un bit se fija a 0 o a 1.
  - ***Intermitentes***: Son ráfagas de fallos que ocurren repetidamente en un mismo lugar y con un efecto temporal. Están provocados por defectos introducidos en los procesos de fabricación, por procesos de desgaste, o por cambios en la temperatura, en el voltaje o en la frecuencia. También son fallos difíciles de detectar por su ocurrencia aleatoria. En el caso de los elementos de almacenamiento, el fallo intermitente típico es el intermittent stuck-at, en el que se fija durante un período de tiempo a 0 o a 1.
- **Error**: Es el estado incorrecto del sistema (o de alguno/a de sus componentes/partes). Está provocado por un fallo y puede causar una avería. Se puede distinguir entre:

- **Errores simples:** El error afecta a un bit.
- **Errores múltiples:** El error afecta a más de un bit.
  - ***Aleatorios***: distribución aleatoria de los errores a lo largo de la palabra.
  - ***Unidireccionales***: Todos los bits se vuelven 1 o 0 (seguidamente se verá, a través de un ejemplo,que esto no tiene porque originar un error).
  - ***Adyacentes***: errores en bits contiguos.
  - ***De ráfaga***: conjunto de bits adyacentes donde, al menos, el primer y el último bit son erróneos.

- **Avería**: Se produce una avería cuando el servicio entregado por el sistema no es el esperado. En este caso, el usuario es capaz de percibir que el sistema no funciona bien.

En este proyecto, los 4 ECCs implementados son capaces de corregir (8, 12, 16 y 20) errores adyacentes, todos corrigen errores en ráfaga de tamaño 8 y las detectan de tamaño 12. Además, corrigen errores simples y detectan errores dobles aleatorios.

Se ha considerdado un ejemplo sencillo para aclarar la diferencia entre fallo y error. Si tenemos un fallo permanente en el que las posiciones 0,1 y 2 (menor peso) de la palabra se mantienen en 0, si la palabra original es 101000 entonces aunque haya un fallo, este no induce a ningún error ya que el código se mantiene intacto. Si la original fuera 101001 entonces si tendríamos un error ya que el fallo convierte esta palabra en 101000. Sin embargo en este caso, el bit modificado es el de menor peso y aunque haya un error puede que este no sea percibido por el usuario. Por poner un caso sencillo, si estamos midiendo la temperatura y esta palabra modifica la parte centesimal o milesimal, el usuario no lo percibirá ya que la temperatura mantendrá unos valores aceptables.

Otros conceptos básicos a tener en cuenta para saber cómo funcionan los ECCs, por qué ocurren los errores y cómo se solucionan son [**20**]:

Un código es un medio para representar datos usando un conjunto de reglas definidas, y una palabra codificada es un elemento del código que satisface las reglas de codificación establecidas.

La palabra codificada se obtiene codificando los datos. Si se decodifica la palabra codificada se volverán a obtener los datos originales.

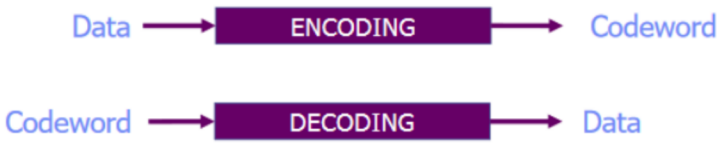


Figura 2.9: Codificación y decodificación [**20**]

Se debe aclarar que aunque se les llame códigos de corrección de errores estos pueden cumplir dos funciones diferentes.La distinción más clara que divide a la familia de ECCs es la capacidad de detección y la capacidad de corrección.

Un código de detección de errores `S` es un código que, habiendo leído o recibido un mensaje corrompido por errores `S`, es capaz de reconocer el mensaje como erróneo. Por esta razón, se produce un fallo de detección cuando el código toma un mensaje dañado por uno correcto. Un ejemplo muy sencillo de un código de detección de errores capaz de reconocer un error es el código de paridad. Al tener un código binario, se agrega un bit adicional al mensaje como un OR exclusivo (XOR) entre todos los bits del mensaje. Si ocurre un error, este bit adicional ya no será el XOR entre todos los bits recibidos. De esta forma se detecta un solo error. Si ocurren dos errores, el bit adicional no puede reconocer los errores y la detección falla. Por esta razón, el código de paridad es un código de detección de sólo un error [**20**].

Un ejemplo sencillo de detección de errores es el de paridad par:

- El 0 se codifica como 00.
- El 1 se codifica como 11.

Decodificación	Análisis
00	Dato correcto: es el 0.
01	No se sabe si el dato es 0 o 1, pero sí que el dato es incorrecto.
10	Igual que en el caso anterior.
11	Dato correcto: es el 1.

Tabla 2.2: Ejemplo: detección de errores

Por otro lado, un código de corrección de errores es un código que no solo es capaz de reconocer si un mensaje está dañado, sino que también identifica y corrige las posiciones incorrectas. Cuando se dice que hay un error de corrección se hace referencia a la incapacidad del código para identificar las posiciones incorrectas y por tanto, no poder corregir el bit erróneo [**20**].

Antes de ver como funcionan los ECCs, se debe saber cuando es preferible emplear un sistema de corrección y cuando uno de detección.

El uso de corrección de errores en lugar de códigos de detección de errores depende del tipo de aplicación. De manera general, se puede decir que [**21**]:

- Usamos un código de detección si suponemos que los datos originales son correctos y por lo tanto pueden ser releídos. Por ejemplo, si el error en un paquete de datos ocurre en la línea de transmisión, con un código de detección simplemente podemos pedir su retransmisión.
- Si la frecuencia de tal evento aumenta por encima de un cierto límite, puede ser más eficiente en términos de ancho de banda utilizar un código de corrección.
- Si el error está en la fuente de la información transmitida, por ejemplo, en la memoria, solo podemos usar un código de corrección.

Funcionamiento de los ECCs

Un ECC binario $(\mathbf{n}, \mathbf{k})$ codifica una palabra de entrada (el dato original) de k bits en una palabra de salida de n bits [19]. La figura 2.10 sintetiza los procesos de codificación y decodificación. La palabra de entrada $\mathbf{u} = (\mathbf{u_0}, \mathbf{u_1}, ..., \mathbf{u_{k-1}})$ es un vector de $\mathbf{k}$-bit que representa los datos originales. La palabra código $\mathbf{b} = (\mathbf{b_0}, \mathbf{b_1}, ..., \mathbf{b_{n-1}})$ es un vector de $\mathbf{n}$ bits, donde el código agrega la redundancia requerida. Se transmite a través del canal que entrega la palabra recibida $\mathbf{r} = (\mathbf{r_0}, \mathbf{r_1}, ..., \mathbf{r_{n-1}})$ [**22**].

El vector de error $\mathbf{e} = (\mathbf{e_0}, \mathbf{e_1}, ..., \mathbf{e_{n-1}})$ modela el error inducido por el canal. Si no ocurre ningún error en el bit $\mathbf{i}$, entonces $\mathbf{e_i} = \mathbf{0}$. Si ocurre un error, $\mathbf{e_i} = \mathbf{1}$. De esta forma, $\mathbf{r}$ puede ser interpretada como $\mathbf{r} = \mathbf{b} \oplus \mathbf{e}$ [**22**].

La matriz de paridad $\mathbf{H}$ de un código lineal de bloque define el código. Para el proceso de codificación se debe satisfacer $\mathbf{H} \cdot \mathbf{b^T} = \mathbf{0}$. En la decodificación por síndrome se define al mismo como $\mathbf{s^T} = \mathbf{H} \cdot \mathbf{r^T}$, y depende exclusivamente de $\mathbf{e}$:

$$\mathbf{s^T} = \mathbf{H} \cdot \mathbf{r^T} = \mathbf{H} \cdot (\mathbf{b} \oplus \mathbf{e})^\mathbf{T} = \mathbf{H} \cdot \mathbf{b^T} \oplus \mathbf{H} \cdot \mathbf{e^T} = \mathbf{H} \cdot \mathbf{e^T}$$

Debe haber un $\mathbf{s}$ diferente para cada vector de error $\mathbf{e}$. Si $\mathbf{s} = \mathbf{0}$, entonces $\mathbf{e} = \mathbf{0}$. Por tanto, $\mathbf{r}$ es correcta. Por otro lado, la decodificación por síndrome se realiza direccionando una tabla de búsqueda (lookup table) que relaciona cada $\mathbf{s}$ con el vector de error estimado $\hat{\mathbf{e}}$. Haciendo la or-exclusiva entre la palabra recibida $\mathbf{r}$ y el vector de error estimado $\hat{\mathbf{e}}$ se obtiene la palabra de código estimada $\hat{\mathbf{b}}$,es decir: $\hat{\mathbf{b}} = \mathbf{r} \oplus \hat{\mathbf{e}}$. Si $\mathbf{s}$ es distinto de cero, pero

la tabla de búsqueda no tiene una entrada para ese síndrome, el error es detectado pero no puede ser corregido [**23**].

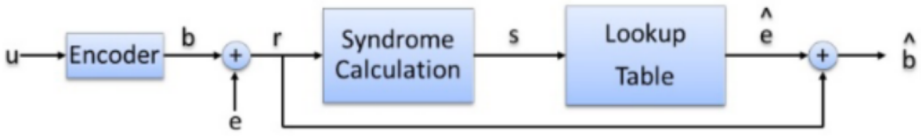


Figura 2.10: Codificación, cruce de canales y proceso de decodificación [**23**]

A continuación se describe la metodología de búsqueda de ECCs a partir de los vectores de error que se desea corregir o detectar, desarrollada por el grupo de Sistemas Tolerantes a Fallos (STF) de la Universitat Politècnica de València. Este trabajo es parte de mi colaboración con este grupo, y los códigos utilizados en el mismo se han obtenido con esta metodología.

Metodología de búsqueda de ECCs a partir de vectores de error

Aunque la metodología que se describe en este apartado se diseñó inicialmente para generar códigos FUEC (códigos con varias zonas de cobertura), esta metodología se puede generalizar para el diseño de cualquier ECC [**24**].

Esta metodología plantea la búsqueda de una matriz de paridad para un determinado ECC como un problema de satisfacibilidad booleana (Boolean Satisfiability problem) [**19**]. Un ECC se define con los siguientes tres pasos:

- Se determina el conjunto de errores a corregir ($\mathbf{E_+}$) y/o detectar ($\mathbf{E_\Delta}$).
- Se determina la longitud de la palabra codificada (n). Este número se calcula usando los conjuntos definidos en el paso 1, junto con la longitud de la palabra sin codificar (k), de tal manera que el número de bits redundantes que se van a utilizar vienen definidos por (n – k).
- El último paso es encontrar, si existe, la matriz de paridad $\mathbf{H(n-k)\times n}$.

La metodología de búsqueda consiste en considerar todas las matrices posibles, comprobando si la matriz obtenida cumple las condiciones (1) y (2), en las que se modelan los errores con sus vectores de error e. . Como se indicó anteriormente, **s** depende exclusivamente de **e** ($\mathbf{s^T = H \cdot e^T}$).

$$\mathbf{H} \cdot \mathbf{e_i^T} \neq \mathbf{H} \cdot \mathbf{e_j^T}; \forall \mathbf{e_i}, \mathbf{e_j} \in \mathbf{E_+} | \mathbf{e_i} \neq \mathbf{e_j}; \qquad (1)$$

$$\mathbf{H} \cdot \mathbf{e_i^T} \neq \mathbf{H} \cdot \mathbf{e_j^T}; \forall \mathbf{e_i} \in \mathbf{E_\Delta}, \mathbf{e_j} \in \mathbf{E_+}; \qquad (2)$$

Es decir, los síndromes utilizados para identificar los errores que se corregirán deben ser diferentes entre ellos, y los síndromes utilizados para identificar los errores que se detectarán deben ser diferentes de los síndromes utilizados para la corrección de errores. Sin embargo, varios errores detectables pueden tener el mismo síndrome. La búsqueda de **H** utiliza un algoritmo recursivo que se inicia con una matriz vacía y va añadiendo todos los posibles valores para cada columna. Cuando se añade una columna, se comprueba si la matriz parcial obtenida cumple las condiciones (1) y (2). Si no es así, se descarta y se prueba un nuevo valor. Si cumple, se añade una nueva columna, repitiendo el proceso hasta que se completa la matriz y se cumplen las condiciones.

Una vez calculada la matriz **H**, las fórmulas de codificación y decodificación se pueden obtener fácilmente de la misma. Más detalles acerca del algoritmo, y de la metodología en general, pueden obtenerse en [24].

Códigos ultrarrápidos

Con la metodología descrita en el apartado anterior, el grupo STF ha diseñado los códigos Ultrafast [**19**]. Estos códigos, diseñados inicialmente para su implementación en hardware, permiten una muy rápida codificación y decodificación. En este trabajo se utilizarán versiones software de estos códigos para estudiar la influencia en el tamaño del programa y en el tiempo de ejecución de varias funciones de decodificación con distintas coberturas, incluyendo la corrección de errores múltiples adyacentes, lo que supone un gran avance en estos tipos de ECC.

Los requerimientos para este tipo de código son [**19**]:

(1) Cada columna de la matriz de paridad H debe ser diferente y distinta de 0.

(2) Cada columna asignada a los bits del código debe tener un solo uno.

(3) Cada columna asignada a los bits de datos debe tener tres unos.

(4) Cada fila debe tener cuatro unos.

(5) Todos los errores corregibles deben tener síndromes diferentes.

(6) Todos los errores detectables deben tener un síndrome diferente a todos los síndromes reservados para la corrección.

La condición 1 permite la corrección de los errores simples. Las condiciones 2 y 3 permiten añadir la detección de errores dobles a la corrección de errores simples: todas las columnas de la matriz de paridad tienen un peso impar. Por tanto, todos los síndromes de errores simples tienen un peso impar, mientras que todos los síndromes de errores dobles tienen un peso par. Esto permite la detección de errores de 2 bits. La condición 4 permite que los circuitos de codificación y de cálculo del síndrome en la decodificación no dependan de la longitud de la palabra. Las condiciones 5 y 6 permiten que se puedan corregir los errores simples y dobles adyacentes.

Siguiendo la metodología descrita anteriormente se obtiene, como ejemplo, una matriz **H** de (16,8). Este código ultrarrápido equivale a un SEC-5AEC-DED esto es que puede corregir errores simples, hasta 5 errores adyacentes y detectar errores dobles no adyacentes. Esta matriz se representa como:

$$H8 = [I_{8x8} \quad A_{8x8}] = \begin{bmatrix} 10000000 & 10100010 \\ 01000000 & 01000101 \\ 00100000 & 10101000 \\ 00010000 & 01010100 \\ 00001000 & 10001010 \\ 00000100 & 01010001 \\ 00000010 & 00101010 \\ 00000001 & 00010101 \end{bmatrix}$$

donde I es la matriz identidad con las columnas para los bits de paridad y A es la segunda mitad de la matriz con las columnas para los bits de datos.

Utilizando esta matriz se pueden diseñar decodificadores con distintas coberturas de error, como se describirá más adelante. Además, los códigos ultrarrápidos para palabras largas de

datos se pueden obtener combinando matrices para palabras de datos más cortas. Como las longitudes de palabras comunes en las computadoras son potencias de dos (8,16,32,64...), podemos usar matrices Ultrarrápidas (16,8) para generar códigos (32,16), (64,32), etc. Por ejemplo, considerando la matriz H8 que se ha mostrado anteriormente, la matriz de verificación de paridad de 16 × 32 para un código ultrarrápido (32, 16), con la misma redundancia y cobertura de errores, se puede generar como:

$$H16 = \begin{bmatrix} & A_{8x8} & 0_{8x8} \\ I_{16x16} & & \\ & 0_{8x8} & A_{8x8} \end{bmatrix}$$

Este nuevo código tiene la misma complejidad, ya que la construcción equivale a tener dos subpalabras de 16 bits (0..7,16..23) y bits (8..15, 24..31), cada una cubierta por códigos independientes (16, 8). También mantiene la cobertura de errores:

- Los errores adyacentes simples y múltiples en cada mitad de palabra se corrigen como en el código (16, 8).
- Los errores adyacentes que afectan a ambas medias palabras se convierten en dos errores adyacentes más cortos para los códigos (16, 8).
- Errores dobles no adyacentes, donde cada bit pertenece a una mitad diferente, se convierten en errores únicos para cada código.
- Los errores dobles no adyacentes dentro de una mitad son detectados por el código correspondiente (16, 8).

Más aún, podemos modificar H16 mediante permutaciones de columna para lograr una mejor cobertura. Columnas alternas de ambas matrices H8, la nueva matriz H16' representa un ultrarrápido (32, 16) SEC-10AEC-DED, donde CiH8 representa la i-ésima columna de la matriz H8. Como se puede observar, un error adyacente de 10 bits se trata como dos errores adyacentes de 5 bits, que pueden corregirse con cada código original.

$$H16' = \begin{bmatrix} C_{0H8} & 0_{8x1} & C_{1H8} & 0_{8x1} & \\ 0_{8x1} & C_{0H8} & 0_{8x1} & C_{1H8} & ... \end{bmatrix}$$

Incluso se pueden generalizar las matrices H16 y H16' para obtener otros tamaños de palabras (32,64...bits). Por ejemplo:

$$H32 = \begin{bmatrix} & A_{8x8} & 0_{8x8} & 0_{8x8} & 0_{8x8} \\ & 0_{8x8} & A_{8x8} & 0_{8x8} & 0_{8x8} \\ I_{32x32} & 0_{8x8} & 0_{8x8} & A_{8x8} & 0_{8x8} \\ & 0_{8x8} & 0_{8x8} & 0_{8x8} & A_{8x8} \end{bmatrix}$$

$$H32' = \begin{bmatrix} C_{0H8} & 0_{8x1} & 0_{8x1} & 0_{8x1} & C_{1H8} & \\ 0_{8x1} & C_{0H8} & 0_{8x1} & 0_{8x1} & 0_{8x1} & ... \\ 0_{8x1} & 0_{8x1} & C_{0H8} & 0_{8x1} & 0_{8x1} & \\ 0_{8x1} & 0_{8x1} & 0_{8x1} & C_{0H8} & 0_{8x1} & \end{bmatrix}$$

Como se indicó anteriormente, el objetivo de los códigos ultrarrápidos es lograr la codificación y decodificación de circuitos de manera lo más rápidamente posible. El rendimiento real de un circuito dependerá de varios factores, como la tecnología de implementación, el nivel lógico de las señales o la complejidad de las ecuaciones lógicas.

Como se describe en el punto 2.5.1, las operaciones de codificación y el cálculo del síndrome (la primera parte de la decodificación), se pueden obtener fácilmente de la matriz de verificación de paridad. En la mayoría de los ECC, la complejidad de estas operaciones depende de la longitud de la palabra. Por el contrario, las expresiones para códigos ultrarrápidos tienen una complejidad baja y constante, independientemente de la longitud de la palabra.

De todos modos, la parte más compleja del decodificador es la implementación de la tabla de búsqueda, especialmente cuando varios síndromes pueden indicar un error en el mismo bit (es decir, cuando el decodificador permite la corrección de errores múltiples). Es necesario simplificar las ecuaciones lógicas necesarias para obtener $\hat{e}_i$ y las señales NRE. La simplificación de las expresiones puede reducir el retardo, el área y el consumo de energía del circuito corrector.

Se pueden obtener ecuaciones no simplificadas de la tabla de búsqueda como suma de minitérminos o producto de maxitérminos. Los códigos ultrarrápidos aprovechan en gran medida su redundancia para obtener una gran cantidad de síndromes libres, que se convierten en indiferentes, lo que permite simplificar las funciones lógicas.

Si se requiere, se puede obtener una simplificación adicional reduciendo la cobertura de error de un código. Por ejemplo, el código SEC-5AEC-DED presentado en 2.5.1 (la matriz H8) permite una gran cobertura de errores, quizás excesiva. Sabiendo que con una hipótesis de fallo dada, los errores adyacentes pueden afectar a dos bits adyacentes como máximo, se puede diseñar un decodificador SEC-DAEC-DED (DAEC es el acrónimo de Double Adjacent Error Correction) utilizando la misma matriz de paridad. En este caso, la tabla de búsqueda considerará como indiferentes los síndromes asignados a errores adyacentes de 3 a 5 bits, mejorando la simplificación. De igual forma se pueden diseñar decodificadores 3AEC/4AEC [**25**].

Siguiendo con el ejemplo anterior, se puede obtener $\hat{e}_{10}$ como la suma de 15 minitérminos (de 8 variables) considerando la cobertura máxima de error (SEC-5AEC-DED) y sin simplificar:

$$\begin{aligned}\hat{e}_{10} = \bar{s}_7\bar{s}_6\bar{s}_5s_4s_3\bar{s}_2\bar{s}_1s_0 + \bar{s}_7\bar{s}_6s_5s_4\bar{s}_3\bar{s}_2s_1s_0 + \\ \bar{s}_7s_6\bar{s}_5\bar{s}_4\bar{s}_3s_2\bar{s}_1s_0 + \bar{s}_7s_6\bar{s}_5s_4\bar{s}_3\bar{s}_2s_1\bar{s}_0 + \\ \bar{s}_7s_6s_5\bar{s}_4\bar{s}_3\bar{s}_2s_1s_0 + \bar{s}_7s_6s_5\bar{s}_4s_3s_2s_1s_0 + \\ \bar{s}_7s_6s_5s_4s_3\bar{s}_2s_1\bar{s}_0 + s_7\bar{s}_6\bar{s}_5\bar{s}_4\bar{s}_3s_2s_1\bar{s}_0 + \\ s_7\bar{s}_6\bar{s}_5s_4\bar{s}_3\bar{s}_2s_1s_0 + s_7\bar{s}_6s_5s_4s_3\bar{s}_2\bar{s}_1s_0 + \\ s_7\bar{s}_6s_5s_4s_3\bar{s}_2s_1\bar{s}_0 + s_7s_6\bar{s}_5\bar{s}_4\bar{s}_3s_2s_1s_0 + \\ s_7s_6\bar{s}_5s_4\bar{s}_3\bar{s}_2s_1\bar{s}_0 + s_7s_6s_5\bar{s}_4s_3s_2\bar{s}_1s_0 + \\ s_7s_6s_5s_4s_3\bar{s}_2s_1\bar{s}_0\end{aligned}$$

Simplificando para obtener la máxima cobertura de errores, se obtiene:

$$\begin{aligned}\hat{e}_{10} = s_6\bar{s}_4s_3 + s_7\bar{s}_5\bar{s}_3s_1 + \bar{s}_6s_4\bar{s}_2s_0 + \\ s_6\bar{s}_2s_1\bar{s}_0 + s_4\bar{s}_2s_1\bar{s}_0\end{aligned}$$

Y simplificando una vez más para obtener la cobertura de errores para SEC-DAEC-DED:

$$\hat{e}_{10} = \bar{s}_4s_2s_0$$

En este trabajo se han implementado cuatro códigos distintos del tipo (64,32) SEC-DED-XAEC, construidos a partir de la matriz H32':

- SEC-DED-8AEC (combinando el decodificador del código (16, 8) SEC-DED-DAEC)
- SEC-DED-12AEC (combinando el decodificador del código (16, 8) SEC-DED-3AEC)
- SEC-DED-16AEC (combinando el decodificador del código (16, 8) SEC-DED-4AEC)
- SEC-DED-20AEC (combinando el decodificador del código (16, 8) SEC-DED-5AEC)

Todos ellos son capaces de corregir errores en ráfaga de 8 bits de tamaño y detectar errores de ráfaga de hasta 12 bits. A su vez, cada uno es capaz de corregir los errores adyacentes indicados (8,12,16 y 20).

2.5.2. Inyección de fallos

La inyección de fallos es una técnica de validación experimental de sistemas tolerantes a fallos. Consiste en la introducción, de forma deliberada, de errores en el sistema para comprobar que los mecanismos de tolerancia a fallos, en este caso Códigos de Corrección de Errores, funcionan según las especificaciones con las que han sido diseñados.

Las pruebas de inyección de fallos en el software se pueden realizar utilizando inyecciones en tiempo de compilación o en tiempo de ejecución.

- La inyección en tiempo de compilación consiste en cambiar el código fuente para simular fallos en el sistema de software.
- La inyección en tiempo de ejecución simula la ocurrencia de un fallo modificando uno (o varios bits) en la memoria (de datos o de código) o en los registros de la CPU.

En este proyecto se lleva a cabo una inyección de fallos en tiempo de ejecución para comprobar si los códigos actúan de forma correcta [**26**].

La inyección se lleva a cabo utilizando la operación XOR entre los datos codificados y una máscara. Se pone en la máscara un 1 en la misma posición que se desea alterar del dato codificado. Con esta operación se simulan los bit-flips, que son los errores transitorios típicos en memoria. Se debe tener en cuenta que el dato codificado es de 16 bits y las posiciones de '1' en la máscara se determinan mediante el sistema hexadecimal.

Capítulo 3

Metodología

Se procede a detallar los pasos a seguir en la inicialización de los periféricos, implementación del código, método de corrección de errores, etc.

En primer lugar, se deben instalar los programas explicados anteriormente (Cube-Mx, Cube-IDE y ST-Link Utility).

Para llevar a cabo este proyecto es necesario descargar unas librerías encargadas del funcionamiento de la pantalla LCD de la tarjeta (controlador BSP). Además son necesarias las librerías predeterminadas de STM32, las cuáles se pueden descargar desde su página web o directamente desde cualquiera de las dos aplicaciones (Cube-IDE o Cube-Mx). Se recomienda esta segunda opción porque así, en caso de una actualización, estas lo harían automáticamente. Las carpetas del controlador BSP también se pueden obtener desde la página web de STM.

Las librerías que se van a utilizar forman parte del controlador BSP y son:

- BSP-Components-Common
- BSP-Components-ili9341
- BSP-Components-stmpe811
- BSP-STM32F429I-Discovery

En el apartado de `Programación con STM32Cube-IDE` se explica como agregar estos archivos al proyecto.

3.1. Montaje

Este sistema empotrado consta de un sensor DHT22 el cual se conecta, como se ve en la figura 2.2:

- El primer pin a VCC (5V).
- El segundo pin a los datos (PE3).
- El tercer pin no se conecta.
- El cuarto pin se conecta a tierra (GND).

En cuanto al `LED RGB`, las conexiones son las siguientes:

- El primer pin se conecta a tierra (GND).
- El segundo pin (azul) se conecta a PC8.
- El tercer pin (verde) se conecta a PA7.
- El cuarto pin (rojo) se conecta a PB4.

3.2. STM32Cube-Mx

El proyecto se comienza con la selección del microcontrolador utilizado, en nuestro caso, el modelo STM32F429i-Discovery.

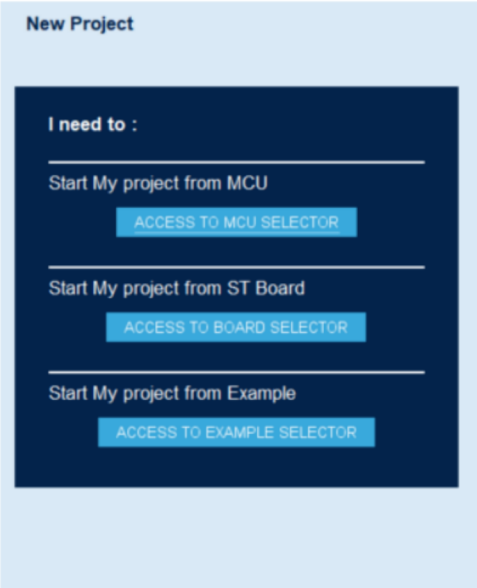


Figura 3.1: Selección de modelo

Una vez se ha escogido el modelo del microcontrolador, en la pantalla principal se pueden ver los pines y parámetros correspondientes al modelo elegido.

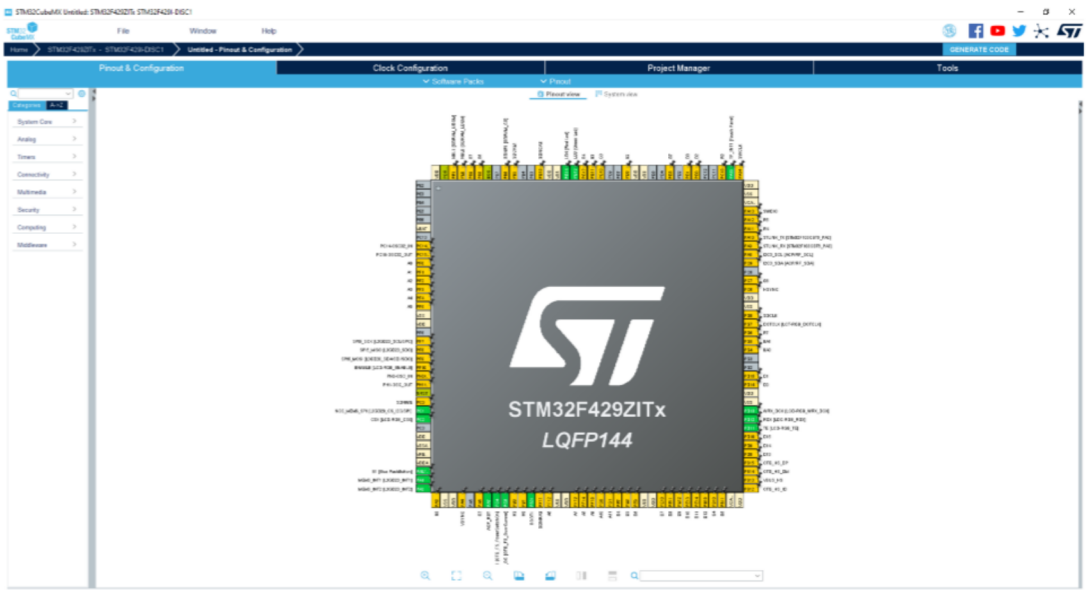


Figura 3.2: CubeMx

3.2.1. DHT22

System Core: RCC

El RCC (del inglés, Reset and Clock Control) se utiliza para controlar los tiempos de reseteo y las frecuencias del reloj. En primer lugar se va a escoger el reloj externo HSE.

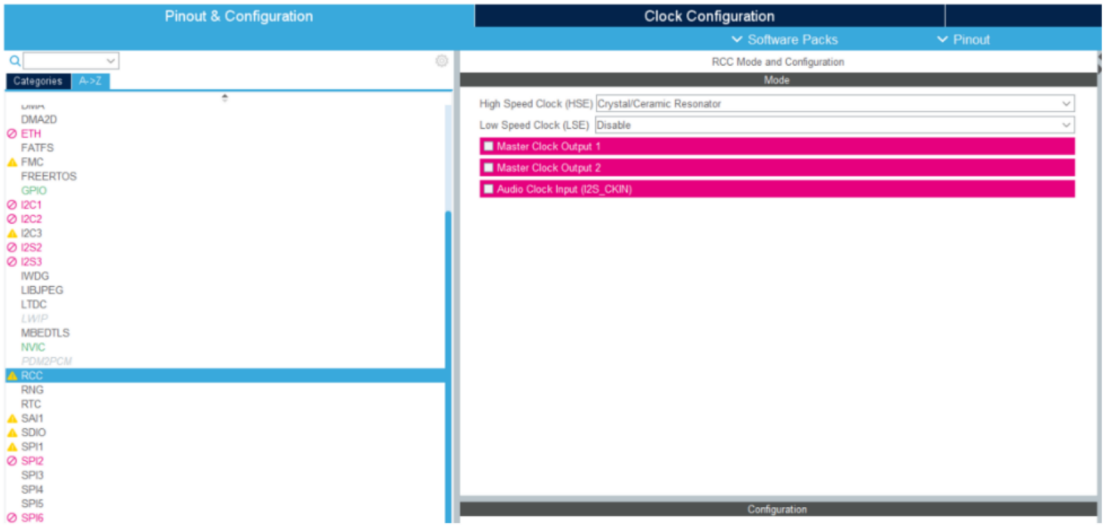
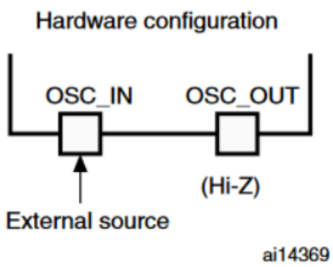


Figura 3.3: Configuración del RCC

En concreto se escogerá el resonador externo de cristal/cerámica el cuál tiene una configuración de Hardware distinta al HSE (Fig 3.4) de usuario y nos garantiza una mayor precisión en el muestreo.



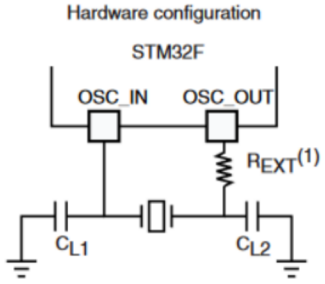


Figura 3.4: a)User external clock [1] b) External crystal/ceramic resonator [1]

Este modo presenta una velocidad superior de reloj en comparación con el LSI.

Timers: TIM6

Para crear un delay (retraso) en microsegundos se activa el modo TIM6

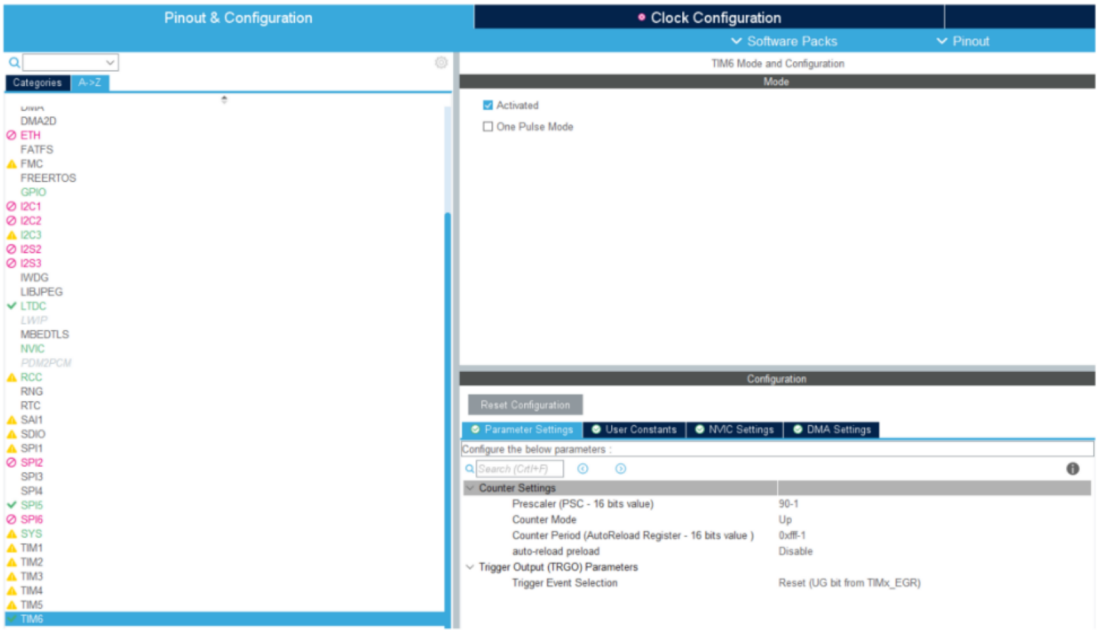


Figura 3.5: Configuración del TIM

Se debe cambiar el preescaler a 90-1 y el counter period a 0xffff-1

Pinout

El pin de salida escogido es el PE2.

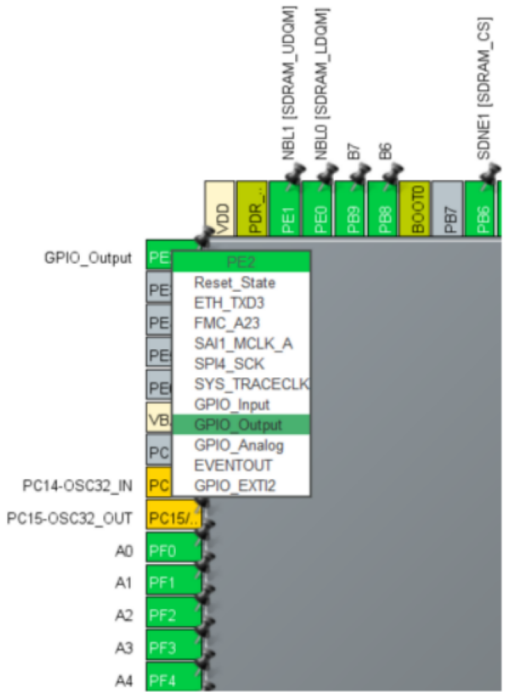


Figura 3.6: Configuración de periféricos

3.2.2. LED RGB

Como se ha visto en el punto 2.3 la intensidad de cada color es de (0,255). Para variar el voltaje de los pines se necesita usar el PWM y para ello se debe activar el TIM3.

El ciclo de trabajo describe la cantidad de tiempo que la señal está en estado ALTO como porcentaje del tiempo total que se necesita para completar un ciclo. Como se mencionó, el valor de intensidad varía de 0 a 255, así que se configura el temporizador de tal manera que el ciclo de trabajo del 100 % sea equivalente a 255 .

Más adelante se configurará un temporizador para que sea 90 MHz. Con este dato se puede calcular el período de conteo necesario:

$$90MHz/255 = 352941/255 = 1384$$

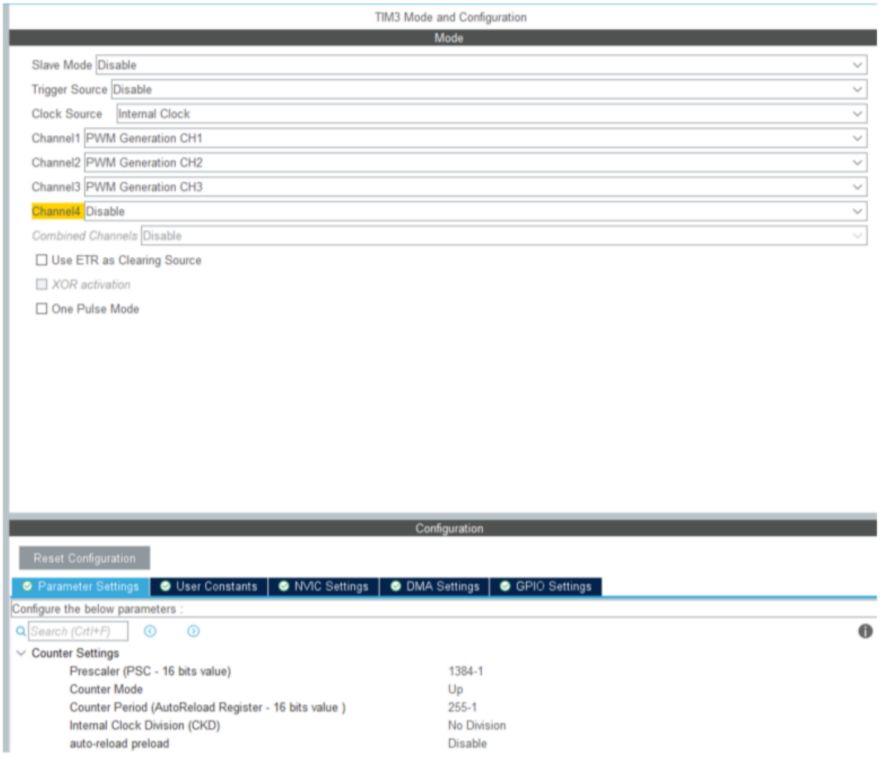


Figura 3.7: Configuración LED RGB

La activación del CH1,CH2 y CH3 provoca la inicialización de los periféricos PC8, PA7 y PB4 donde irán conectados los colores azul, verde y rojo.

3.2.3. Pantalla LCD

Para poder representar los valores por pantalla se deben configurar los siguientes parámetros.

Multimedia: DMA2D y LTDC

- DMA es un mecanismo (Hardware) que permite transferir a los periféricos la información de entrada y salida desde la memoria principal sin necesidad de involucrar al procesador, lo cual, ahorra tiempo de computación [**28**].

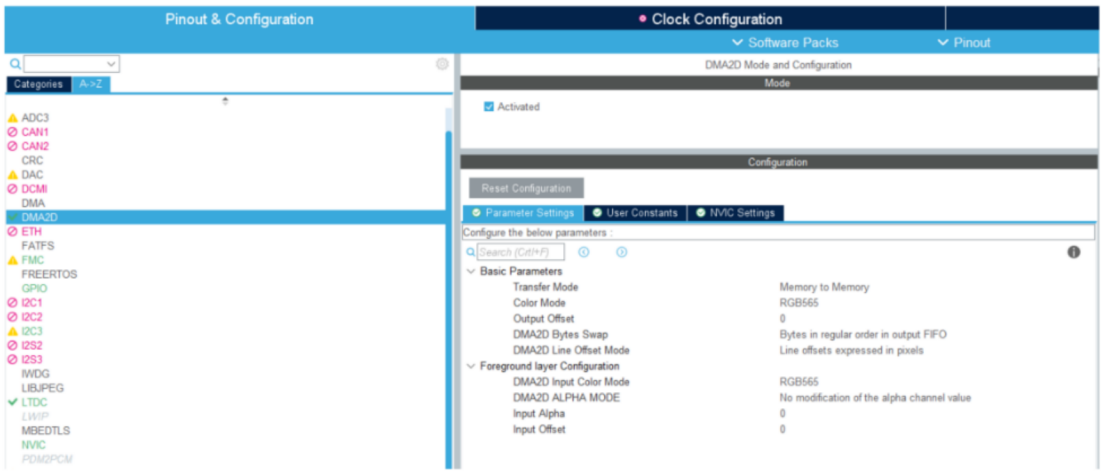


Figura 3.8: Configuración DMA2D

El acelerador Chrom-ART (DMA2D) ofrece una aceleración de hardware para operaciones gráficas. Se basa en un motor DMA 2D para una copia de datos rápida con funciones específicas para admitir la conversión de formato de píxeles (imágenes) [**29**].

- A continuación se activa el LTDC. Este controlador se utiliza para configurar interfaces RGB [**30**]. Se deben modificar los datos de Active Width:240 y Active Height:320 para que estén acordes con el tamaño de nuestra pantalla y así poder visualizar de forma correcta todo el texto.

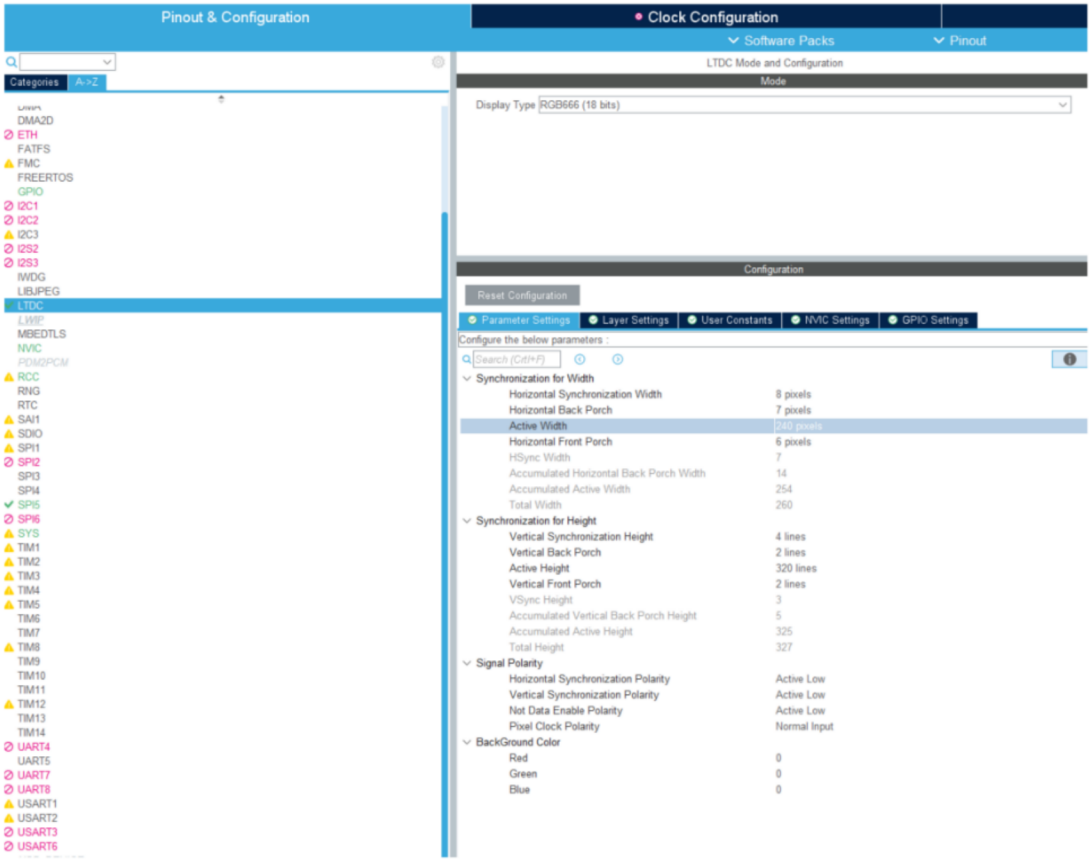


Figura 3.9: Configuración LTDC

Conectividad: FMC, I2C3 y SPI5

- En cuánto a la conectividad, en primer lugar se debe activar el FMC. Este controlador permite utilizar la memoria externa SDRAM del dispositivo. Esta interfaz es totalmente configurable, lo que permite una fácil conexión con memorias externas u otras interfaces paralelas.

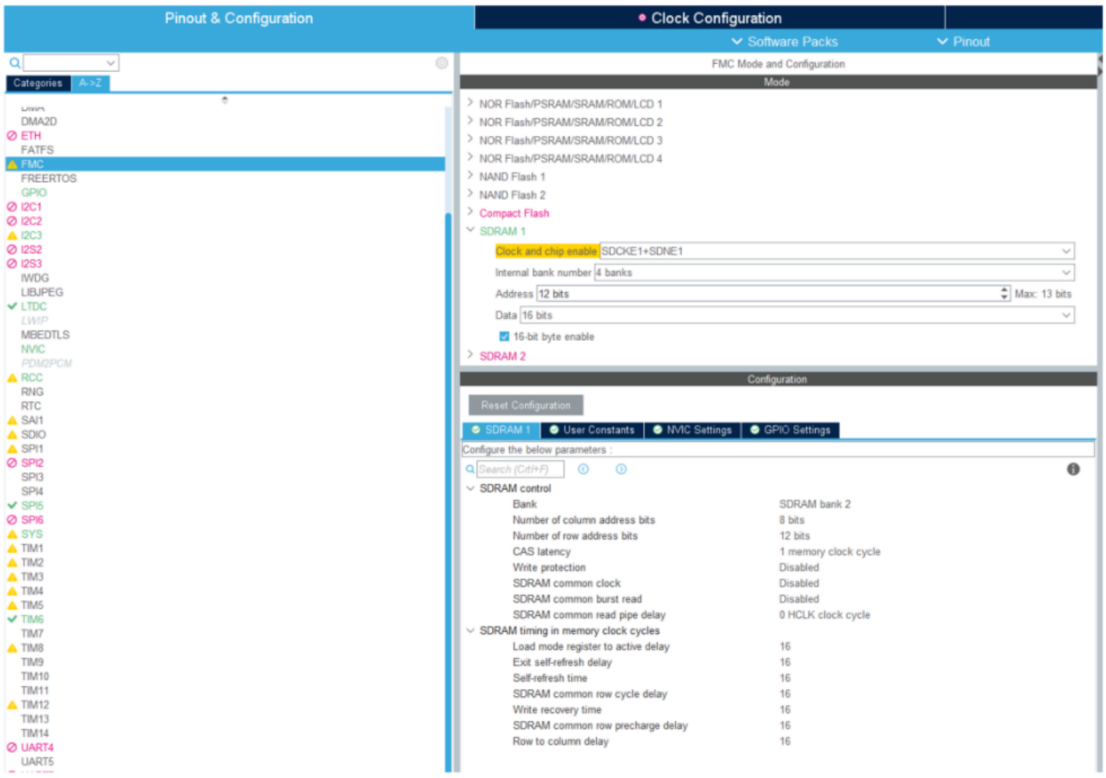


Figura 3.10: Configuración FMC

Los beneficios del controlador FMC incluyen no solo la extensión del espacio de memoria RAM y Flash, sino también la capacidad de interactuar sin problemas con la mayoría de los controladores LCD. Esta capacidad de interfaz paralela de LCD facilita la creación de aplicaciones gráficas rentables utilizando módulos LCD que contienen controladores integrados o soluciones de alto rendimiento que utilizan controladores externos con aceleración dedicada [**31**].

- Por otro lado, se debe activar el I2C, que es un protocolo de comunicación entre circuitos integrados (ICs) que se encuentran próximos. Este protocolo está cobrando mucha importancia, principalmente por su uso en empresas [**32**].

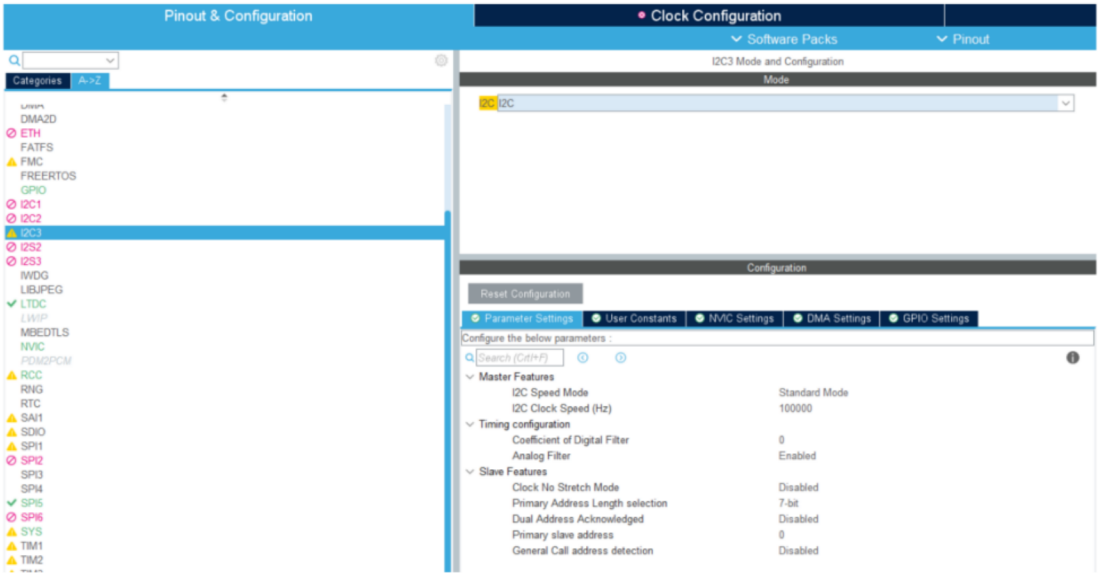


Figura 3.11: Configuración I2C

Los detalles del I2C son [**32**]:

- Cómo se debe enviar la información.
- Cómo se debe recibir la información.
- Cómo se debe manejar los errores.
- Cómo deben manejar la información el remitente y el receptor.

- Por último, se conecta el protocolo SPI. Este permite la comunicación entre un dispositivo principal y otro o varios dispositivos secundarios como pueden ser los sensores,una pantalla, una tarjeta SD, un chip EEPROM,etc. Este protocolo es fundamental para el intercambio de información entre estos dispositivos [**33**].

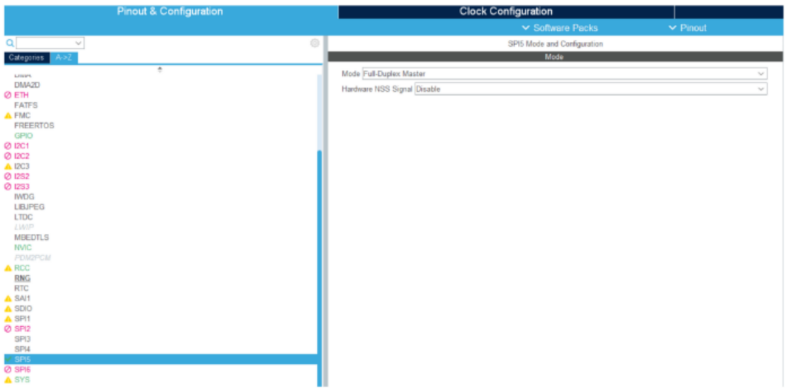


Figura 3.12: Configuración SPI5

3.2.4. Generación del código

El último parámetro que se va a modificar es el del reloj. Se pone la máxima frecuencia permitida para el dispositivo: 180 MHz. Además, se debe marcar la casilla de HSE para terminar de configurarlo. Se confirma que los temporizadores (APB1) están a 90MHz.

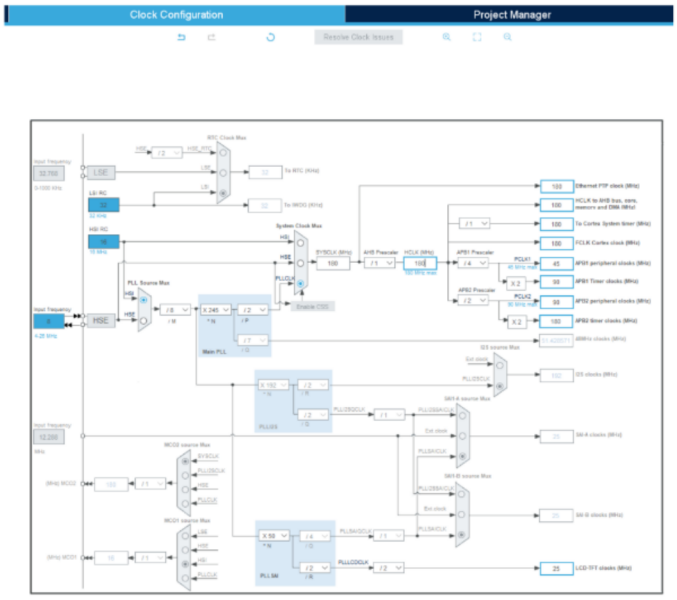


Figura 3.13: Configuración del reloj

Finalmente, se debe indicar el nombre del proyecto así como la IDE con la que se va a trabajar. En nuestro caso, como se ha comentado anteriormente, se ha utilizado el entorno de STM por tanto, se selecciona STM32Cube-IDE.

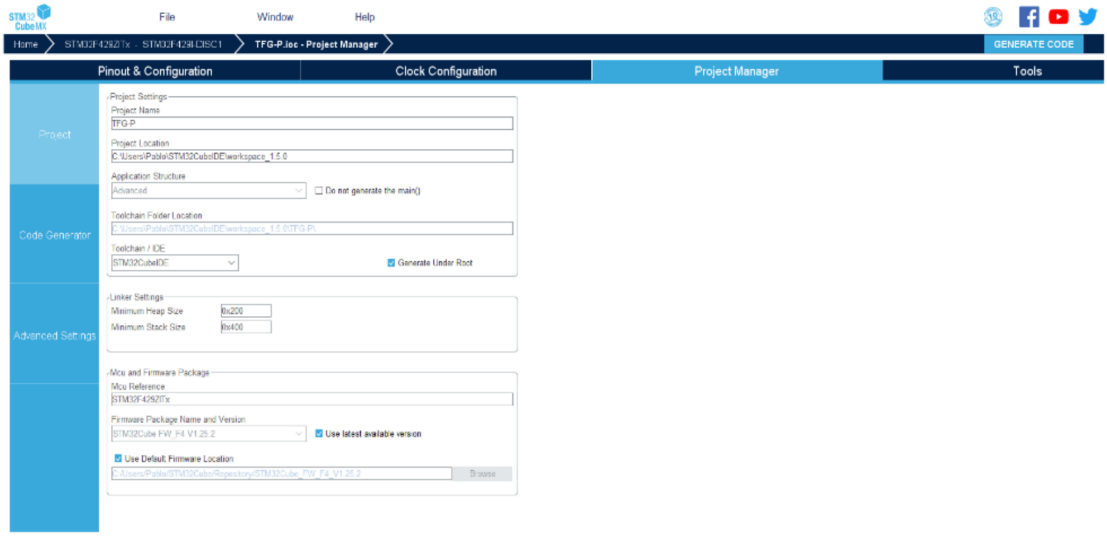


Figura 3.14: Generación del código

3.3. Programación con STM32Cube-IDE

Al generar el código, se acepta la opción de abrir proyecto y automáticamente se nos dirige a la aplicación STM32Cube-IDE.

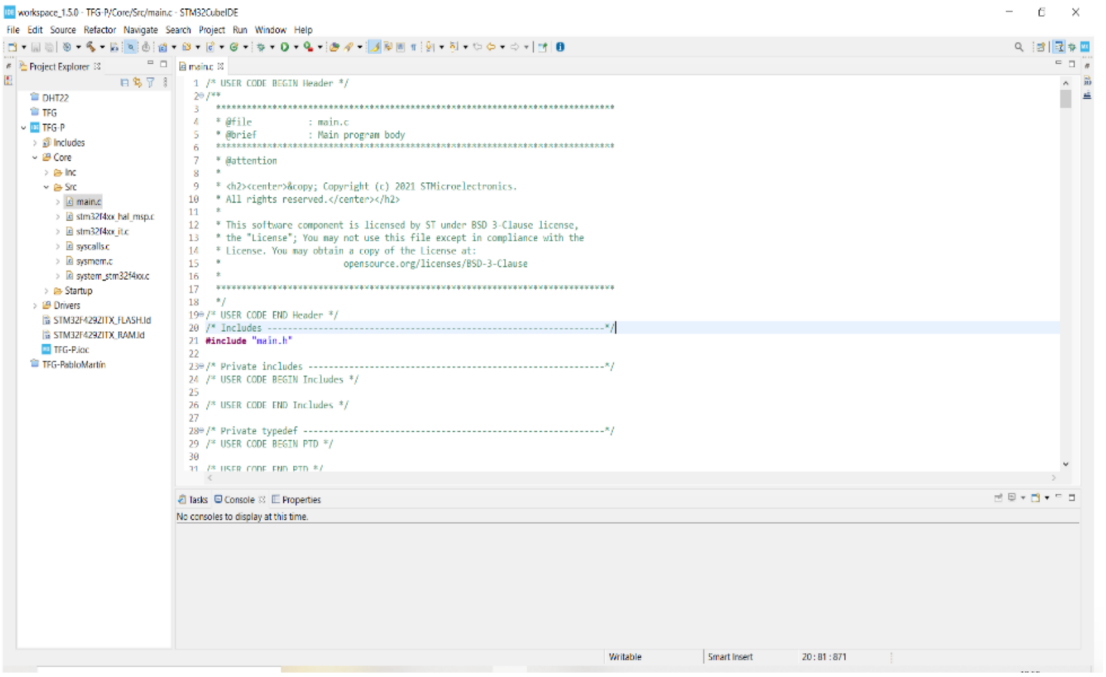


Figura 3.15: STM32Cube-IDE

En primer lugar, se deben configurar una serie de parámetros antes de comenzar con el código. Se procede a la configuración de las bibliotecas, las cuales hay que descargar para que el programa funcione. Una vez descargadas se deben agregar al proyecto (simplemente copiar y pegar estas carpetas en la carpeta especificada) y también se debe indicar dónde están siguiendo las instrucciones de la imagen siguiente.

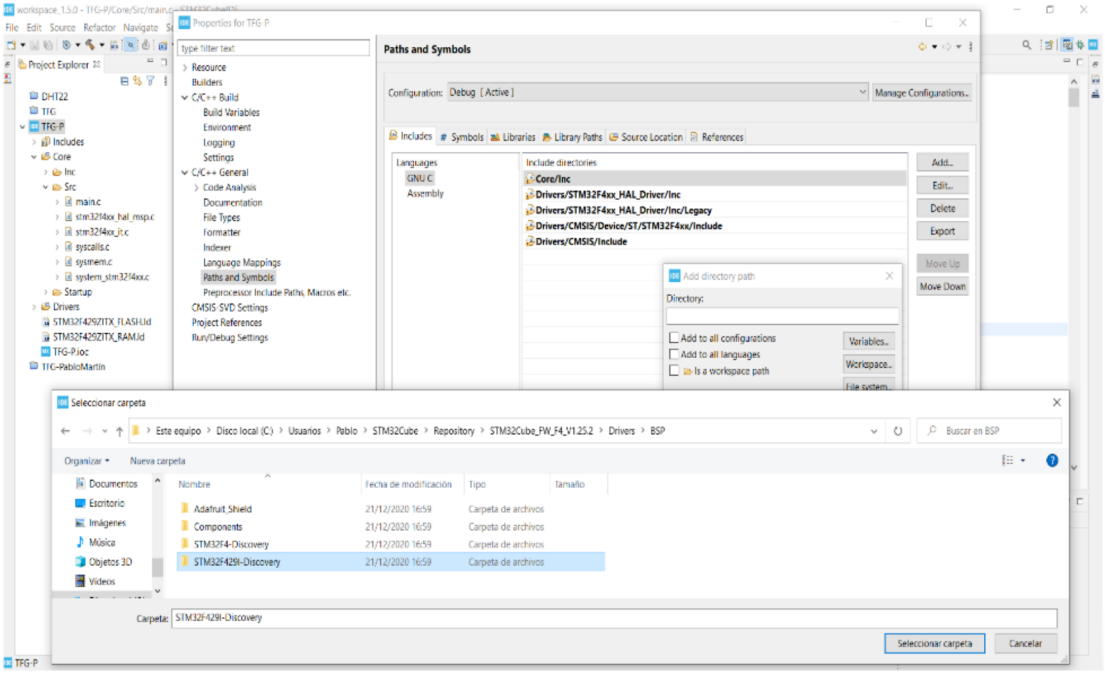


Figura 3.16: Configuración de las bibliotecas

La carpeta BSP (con los archivos de `Common`, `ili9341`, `stmpe811` y `STM32F429-Discovery`) se debe copiar a los Drivers del proyecto. Por otro lado, debemos agregar la carpeta Utilities en el mismo nivel que las carpetas principales. Esta carpeta se encuentra entre `los archivos` propios de STM32Cube.

Como se verá más adelante, se va a utilizar la función sprintf con valores de tipo float. En estos casos, para evitar un error debemos configurar la MCU (Micro Controller Unit) permitiendo que se pueda utilizar la librería newlib-nano gracias a la cual se pueden escanear y mostrar por pantalla variables de tipo float.

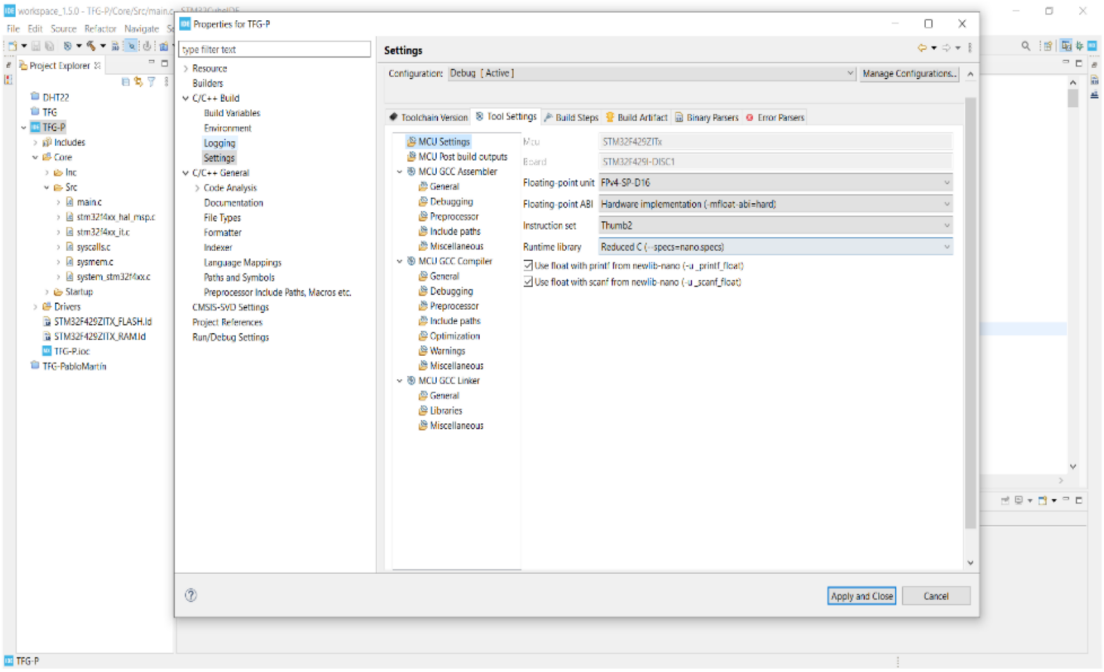


Figura 3.17: Configuración de las propiedades

Se comieza el código incluyendo las librerías necesarias:

```
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "stm32f429i_discovery_lcd.h"
#include "stdio.h"
#include "DHT.h"
#include "stm32f429i_discovery.h"
#include <string.h>

/* USER CODE END Includes */
```

Figura 3.18: Includes

3.3.1. User Code Begin 0

Una vez se han incluido las librerías, se prosigue con el User Code Begin 0. En esta parte del código se declaran las variables necesarias en el proyecto así como las funciones que se van a utilizar.

DHT22

En primer lugar se tiene una función **delay** que crea un retraso en microsegundos. Básicamente lo que hace esta función es poner el contador en 0 y luego esperar a que el contador llegue al tiempo insertado.

Luego se añaden variables que se van a necesitar en el código.

A continuación, se insertan las funciones para modificar los puertos GPIO. En estas funciones se declara el pin como salida o entrada en función de los requerimientos.

Se define el puerto GPIO y el pin utilizados.

```
/* USER CODE BEGIN 0 */
void delay (uint16_t time)
{

    __HAL_TIM_SET_COUNTER(&htim6,0);
    while ((__HAL_TIM_GET_COUNTER(&htim6))<time);
}

uint8_t Rh_byte1, Rh_byte2, Temp_byte1, Temp_byte2;
uint16_t SUM, RH, TEMP;

float Temperature = 0;
float Humidity = 0;
uint8_t Presence = 0;

void Set_Pin_Output (GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin)
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};
    GPIO_InitStruct.Pin = GPIO_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOx, &GPIO_InitStruct);
}

void Set_Pin_Input (GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin)
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};
    GPIO_InitStruct.Pin = GPIO_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
    GPIO_InitStruct.Pull = GPIO_PULLUP;
    HAL_GPIO_Init(GPIOx, &GPIO_InitStruct);
}


#define DHT22_PORT GPIOE
#define DHT22_PIN GPIO_PIN_3

```

Figura 3.19: User Code Begin 0 - DHT22

Se procede a añadir las funciones relacionadas con la `inicialización`, `transmisión de datos` y `lectura de datos`. Todas estas funciones se han explicado en la fundamentación teórica del DHT22.

```
119 void DHT22_Start (void)
120 {
121     Set_Pin_Output(DHT22_PORT, DHT22_PIN); // set the pin as output
122     HAL_GPIO_WritePin (DHT22_PORT, DHT22_PIN, 0);   // pull the pin low
123     delay(1200);   // wait for > 1ms
124
125     HAL_GPIO_WritePin (DHT22_PORT, DHT22_PIN, 1);   // pull the pin high
126     delay (20);   // wait for 30us
127
128     Set_Pin_Input(DHT22_PORT, DHT22_PIN);   // set as input
129 }
130
131 uint8_t DHT22_Check_Response (void)
132 {
133     Set_Pin_Input(DHT22_PORT, DHT22_PIN);   // set as input
134     uint8_t Response = 0;
135     delay (40);  // wait for 40us
136     if (!(HAL_GPIO_ReadPin (DHT22_PORT, DHT22_PIN))) // if the pin is low
137     {
138         delay (80);   // wait for 80us
139
140         if ((HAL_GPIO_ReadPin (DHT22_PORT, DHT22_PIN))) Response = 1;  // if the pin is high, response is ok
141         else Response = -1;
142     }
143
144     while ((HAL_GPIO_ReadPin (DHT22_PORT, DHT22_PIN)));   // wait for the pin to go low
145     return Response;
146 }
147
```

Figura 3.20: User Code Begin 0 - DHT22

```
148 uint8_t DHT22_Read (void)
149 {
150     uint8_t i,j;
151     for (j=0;j<8;j++)
152     {
153         while (!(HAL_GPIO_ReadPin (DHT22_PORT, DHT22_PIN)));   // wait for the pin to go high
154         delay (40);   // wait for 40 us
155
156         if (!(HAL_GPIO_ReadPin (DHT22_PORT, DHT22_PIN)))   // if the pin is low
157         {
158             i&= ~(1<<(7-j));   // write 0
159         }
160         else i|= (1<<(7-j));  // if the pin is high, write 1
161         while ((HAL_GPIO_ReadPin (DHT22_PORT, DHT22_PIN)));  // wait for the pin to go low
162     }
163
164     return i;
165 }
166
```

Figura 3.21: User Code Begin 0 - DHT22

Variables

Se añaden las variables necesarias tanto en las funciones propias del ECC como del sistema de control o el temporizador SysTick.

```
168
169 #define bitvector unsigned int
170 #define N 64
171 #define M 32
172
173
174
175
176 uint64_t Temp, Hum;
177 float T,H;
178 int resultado;
179
180 _Bool ErrD;
181
182 unsigned int start_time, stop_time, cycle_count;
183
```

Figura 3.22: Variables

LED RGB

Se escribe la función encargada de alimentar los valores desde los diferentes canales que están conectados a cada pin.

```
458 void rgb_set (uint8_t red, uint8_t blue, uint8_t green) {
459
460     htim3.Instance->CCR1=red;
461     htim3.Instance->CCR2=blue;
462     htim3.Instance->CCR3=green;
463 }
```

Figura 3.23: User Code Begin 0 - LED RGB

ECC

Debido al tamaño de las funciones `codificar y decodificar`, estas se incluyen en el Anexo D.

Además, también se declaran las funciones **EscrMem y LeerMem** encargadas de escribir en memoria la palabra codificada y de devolver el resultado de la palabra decodificada.

```
419 void EscrMem(float u, uint64_t*b)
420 {
421     int i;
422     uint32_t U;
423     bitvector enc[N],dat[M];
424 
425     memcpy(&U,&u,M/8);
426 
427     for (i=0;i<M;i++)
428     dat[i]=(U&(1<<i))>>i;
429     codificar(enc,dat);
430     *b=0;
431 
432     for(i=0;i<N;i++)
433     *b=*b|((uint64_t)enc[i]<<i);
434 
435 }
436 
437 float LeerMem(uint64_t r)
438 {
439     int i;
440     float u;
441     uint32_t U;
442     bitvector rec[N],dec[M];
443 
444     for(i=0;i<N;i++)
445         rec[i]=(r&(1ULL<<i))>>i;
446     decodificar(rec,dec);
447     U=0;
448     for (i=0;i<M;i++)
449         U=U|((uint32_t)dec[i]<<i);
450     memcpy(&u,&U,M/8);
451     return u;
452 
453 }
454 
```

Figura 3.24: User Code Begin 0 - ECC

Inyección de fallos

Se declara en último lugar la función **InytMem** necesaria para la inyección de fallos.

```
466 void InytMem(uint64_t *mem, uint64_t err)
467 {
468 *mem = *mem ^ err;
469 }
470 
```

Figura 3.25: User Code Begin 0 - Inyección de fallos

3.3.2. User Code Begin 2

En esta parte del código se inicializan las funciones de los temporizadores.

DHT22

A continuación, se añade la función que inicializa el TIM6.

```
515  /* USER CODE BEGIN 2 */
516
517
518  HAL_TIM_Base_Start(&htim6);
519
```

Figura 3.26: User Code Begin 2 - DHT22

LED RGB

Se inicializa el TIM3 y los respectivos canales.

```
520
521  HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_1);
522  HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_2);
523  HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_3);
524
525
```

Figura 3.27: User Code Begin 2 - LED RGB

Temporizador SysTick

Se añaden los parámetros necesarios para el uso del temporizador SysTick.

```
475
476  SysTick->CTRL=0;
477  SysTick->LOAD=0xFFFFFFFF;
478  SysTick->VAL=0;
479  SysTick->CTRL=0x5;
480
```

Figura 3.28: SysTick

Pantalla LCD

Finalmente, se inicializa la pantalla LCD del microcontrolador.

```
483  BSP_LCD_Init();
484  BSP_LCD_LayerDefaultInit(1, LCD_FRAME_BUFFER);
485  BSP_LCD_SelectLayer(1);
486  BSP_LCD_Clear(LCD_COLOR_WHITE);
487  BSP_LCD_SetColorKeying(1, LCD_COLOR_WHITE);
488  BSP_LCD_SetLayerVisible(1, DISABLE);
489  BSP_LCD_LayerDefaultInit(0, LCD_FRAME_BUFFER + 0x130000);
490  BSP_LCD_SelectLayer(0);
491  BSP_LCD_DisplayOn();
492  BSP_LCD_Clear(LCD_COLOR_RED);
493  BSP_LCD_DisplayStringAt(10, 10,(uint8_t*) "DATOS SENSOR:", LEFT_MODE);
494
```

Figura 3.29: Pantalla LCD

3.3.3. User Code Begin 3

En esta sección del programa se llevan a cabo las llamadas a las funciones, se muestran los resultados y se programa el sistema de control deseado.

DHT22

Se llaman a las funciones necesarias del DHT22 para obtener los resultados, y una vez obtenidos se declaran como variables tipo float. El proceso de obtención de estos resultados se ha comentado con anterioridad.

```
534  while (1)
535  {
536    /* USER CODE END WHILE */
537    /* USER CODE BEGIN 3 */
538
539      DHT22_Start();
540      Presence = DHT22_Check_Response();
541      Rh_byte1 = DHT22_Read ();
542        Rh_byte2 = DHT22_Read ();
543      Temp_byte1 = DHT22_Read ();
544      Temp_byte2 = DHT22_Read ();
545      SUM = DHT22_Read();
546
547      TEMP = ((Temp_byte1<<8)|Temp_byte2);
548      RH = ((Rh_byte1<<8)|Rh_byte2);
549
550      Temperature = (float) (TEMP/10.0);
551      Humidity = (float) (RH/10.0);
552
```

Figura 3.30: DHT22

Pantalla LCD, SysTick, ECC e Inyección de fallos

En primer lugar, se muestra por pantalla los datos sin proteger. Luego, se escriben en memoria los valores obtenidos para proceder con la protección de los datos. Luego se inyectan los fallos para luego leer el valor en memoria y comprobar que el código funciona correctamente. Gracias a la funciones del SysTick, se obtiene el tiempo de ejecución de estas operaciones. Finalmente, los datos protegidos se enseñan por pantalla.

```
523      sprintf(str,"Tem: %f",Temperature);
524       BSP_LCD_DisplayStringAt(10,50, (uint8_t*)str,LEFT_MODE);
525      sprintf(str,"Hum: %f",Humidity);
526       BSP_LCD_DisplayStringAt(10,80, (uint8_t*)str,LEFT_MODE);
527
528       start_time=SysTick->VAL;
529
530     EscrMem((float) (TEMP/10.0), &Temp);
531     EscrMem((float) (RH/10.0), &Hum);
532
533       InytMem(&Temp, 0xFFFFFF);
534       InytMem(&Hum,  0xFFFFFF);
535
536     T=LeerMem(Temp);
537     H=LeerMem(Hum);
538
539     stop_time=SysTick->VAL;
540     cycle_count=start_time-stop_time;
541
542     sprintf(str,"Tem: %f",T);
543     BSP_LCD_DisplayStringAt(10,120, (uint8_t*)str,LEFT_MODE);
544
545     sprintf(str,"Hum: %f",H);
546     BSP_LCD_DisplayStringAt(10,150, (uint8_t*)str,LEFT_MODE);
```

Figura 3.31: SysTick + ECC + Inyección de fallos

Sistema de control

Se utiliza un sistema sencillo de control en el que en primer lugar se comprueba que los resultados de temperatura y humedad protegidos coinciden con los resultados sin proteger (teniendo en cuenta que la inyección de fallos puede provocar errores). Si esto se cumple la variable resultado vale 1 y si no se cumple entonces vale 0. Esta variable se muestra por la pantalla LCD.

Finalmente se determinan unas condiciones óptimas para el huerto y en función de si se cumplen o no estas condiciones, el LED se pone de color verde o rojo respectivamente.

Este sistema de control, a pesar de su sencillez, es suficiente para comprobar el funcionamiento de los ECCs, y cómo afecta la mayor o menor tolerancia a fallos de cada ECC probado al sistema.

```
549
550     if (Temperature==LeerMem(Temp) && Humidity==LeerMem(Hum)) {
551        resultado=1;
552      }
553      else {
554        resultado=0;
555      }
556
557     sprintf(str,"Resultado: %d",resultado);
558     BSP_LCD_DisplayStringAt(10,200, (uint8_t*)str,LEFT_MODE);
559
560
561       HAL_Delay(3000);
562
563       if (Humidity<70 || Temperature>25 || Temperature<15 || resultado==0)
564       {
565           rgb_set (255,0,0); //Red
566           HAL_Delay (5000);
567       }
568
569       else {
570       rgb_set (0,255,255); //Blue + Green
571       HAL_Delay (5000);
572       }
573
574   }
575   /* USER CODE END 3 */
576 }
```

Figura 3.32: Sistema de control

3.4. Depuración del programa

Por último, se depura el proyecto. Primero se debe contruir para ver que no se tiene ningún error. Para ello se va a la ventana de Project→Build Project.

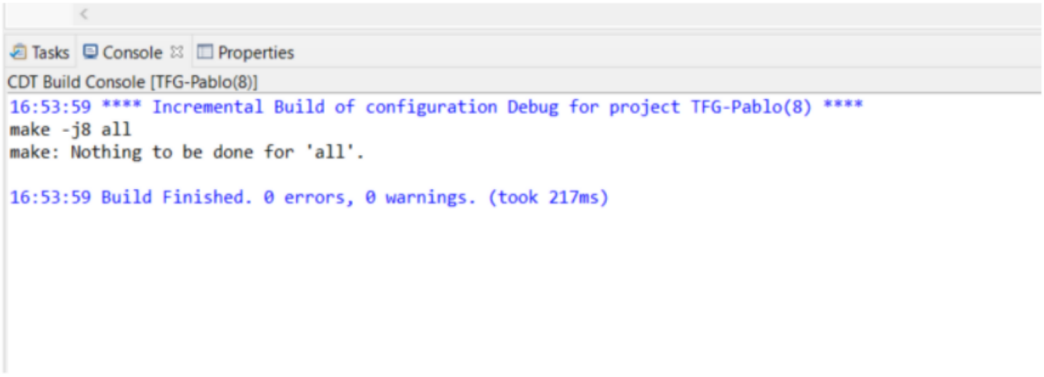


Figura 3.33: Build Project

Una vez que se ha construido sin errores se prosigue con la depuración. Los pasos son los siguientes :

- Se configura el depurador:

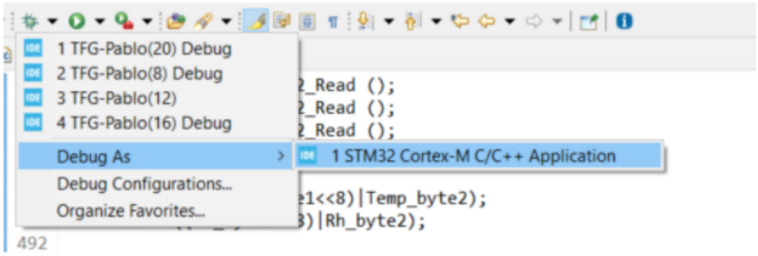


Figura 3.34: Configuración Debugger

En este apartado únicamente se pone el nombre del depurador igual que el del proyecto (para evitar confusiones con otros proyectos), y se le da a Debug.

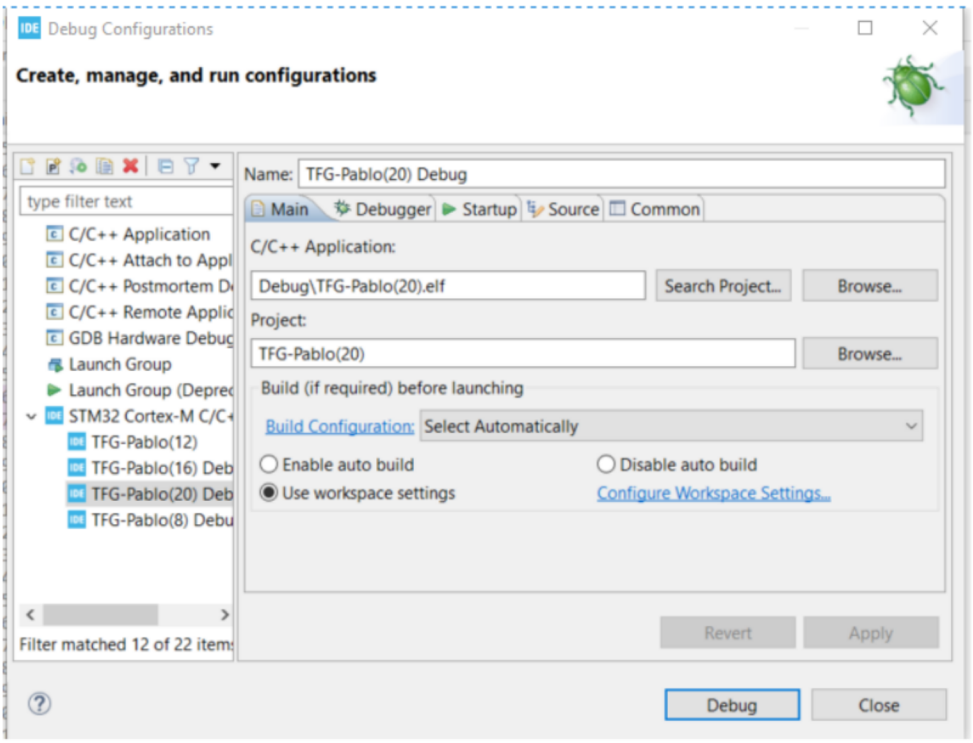


Figura 3.35: Configuración Debugger

- Una vez se ha creado la configuración del Debug si queremos volver a debuggear entonces:

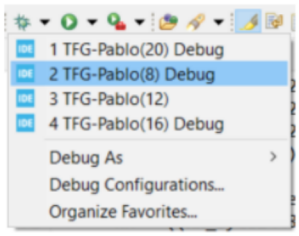


Figura 3.36: Elección Debugger

- Una vez se ha debuggeado para que arranque el programa debemos darle al botón verde (Play) y se obtiene esta pantalla:

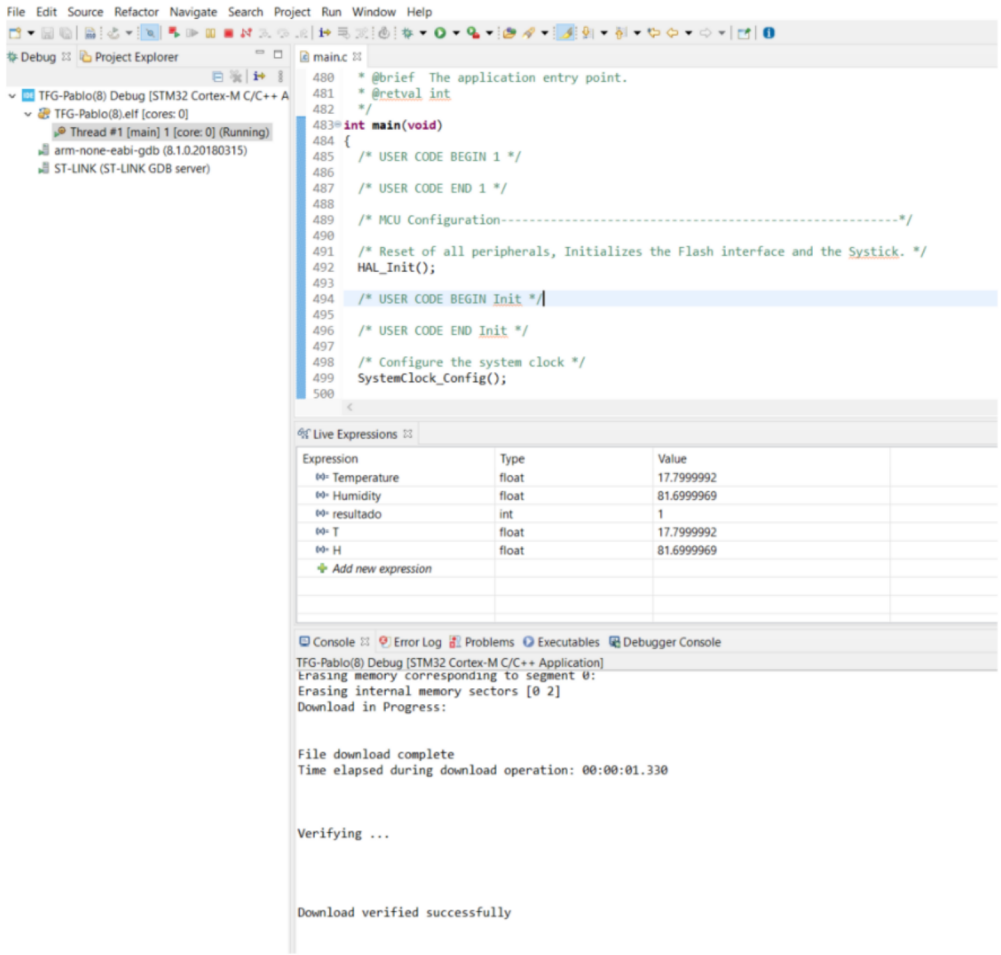


Figura 3.37: Resultado final

Capítulo 4

Análisis de los resultados

Se presenta una tabla con los resultados obtenidos tras la inyección de fallos. Hay que aclarar una serie de aspectos:

- La variable resultado determina si los datos de temperatura y humedad protegidos coinciden con los datos sin proteger (humedad=1) o si por el contrario, no coinciden (humedad=0).
- El valor de referencia de temperatura y humedad de cada tipo de ECC es el obtenido con la inyección del primer fallo (0x0001) ya que en ese caso se sabe que no va a inducir a ningún error. Por tanto, un resultado tiene marcada la función no cumple cuando el valor de temperatura y humedad no coincide con el valor del primer ensayo.
- Los fallos se introducen en Hexadecimal. En la tabla se muestra ese valor y debajo el equivalente en binario. Se especifican los dos porque así es más fácil para el lector identificar dónde están localizados los fallos y si el código debe o no debe corregirlos/detectarlos.
- El motivo porque el no coinciden las temperaturas y humedades de los diferentes ECC es porque los resultados se han obtenido en diferentes momentos del día.
- Los cuatro códigos presentan el mismo consumo de memoria RAM 2.32kB (1,21 %). Además, la función codificar es la misma en todos ellos. Por tanto, también ocupa lo mismo (1.26kB). Se especifica el tamaño de estas funciones ya que son las que más memoria FLASH ocupan, con bastante diferencia, además de ser una de las bases del trabajo.
- Para obtener los tiempos de ejecución en ms se ha divido el resultado proporcionado por cycle_count entre el valor del reloj del sistema: 180MHz.

Tipo	Inyección(Hex & Binario)	Resultado	Cumple	Temperatura	Humedad
8	0x0000 0x0000000000000000	1	sí	16.89	71.8
	0xFF00 0x1111111100000000	1	sí	16.91	72.1
	0x3FF0 0x0011111111110000	0	sí	22.89	87.1
	0x11F2 0x001000111110010	0	sí	19	71.3
12	0x0000 0x0000000000000000	1	sí	16.2	77.6
	0xDE00 0x1101111000000000	1	sí	16.1	77.8
	0x9F0 0x0000100111110000	1	sí	16.2	77.8
	0xFFFF 0x1111111111111111	0	sí	1.08e+9	322961408
16	0x0000 0x0000000000000000	1	sí	17.5	78.2
	0xDE20 0x1101111000100000	1	sí	17.6	78.2
	0xFFFF 0x1111111111100010	1	sí	17.6	78
	0xFFFFF 0x11111111111111111111	0	sí	18647.9	20047.4
20	0x0000 0x0000000000000000	1	sí	18	75.2
	0xFFE0 0x1111111111100000	1	sí	18.1	75.1
	0xFFFFF 0x11111111111111111111	1	sí	18.1	75.2
	0xFFFEE 0x11111111111111101110	0	sí	35260	7849.4

Tabla 4.1: Resultados

Tipo	Tamaño de archivo	Tiempo de ejecución ECC
8	FLASH:29kB (1.45 %) Decodificar: 7.39kB	175.3 ms
12	FLASH:33.4kB (1.67 %) Decodificar: 10.76kB	189.8 ms
16	FLASH:35kB (1.75 %) Decodificar: 13.57kB	202 ms
20	FLASH:40.58kB (1.98 %) Decodificar: 17.21kB	233.7 ms

Tabla 4.2: Tamaño de archivo y tiempo de ejecución del ECC

Se verifica en la tabla de resultados que todos los códigos han sido implementados correctamente ya que cumplen con su función de corregir los errores. Se observa que cuando un error no puede ser corregido el valor en muchos casos varía enormemente mientras que en otros casos esta diferencia no es tan notable. Comparando los resultados obtenidos se puede verificar que esta diferencia de valores aumenta proporcionalmente a la cantidad de fallos inyectados.

El tiempo de ejecución de la codificación (EscrMem) y de la inyección de fallos (InytMem) es el mismo en todos los ECCs.

Función	Tiempo de ejecución
EscrMem	71.1 ms
InytMem	0.67 ms

Tabla 4.3: Tiempo de ejecución: codificación e inyección de errores

Por tanto, los valores de la decodificación (LeerMem) son los siguientes:

Tipo	Tiempo de ejecución
8	103.53 ms
12	118.03 ms
16	130.23 ms
20	161.93 ms

Tabla 4.4: Tiempo de ejecución: decodificación

Se comprueba que los tiempos de decodificación dependen directamente del tamaño del ECC, es decir, cuánto más errores corrigen más tiempo tarda en ejecutarse.

Figura 4.1: Muestra por pantalla

Por otro lado, se ve que el sistema empotrado funciona correctamente. El LED cumple con el sistema de control deseado y los resultados se muestran correctamente por pantalla.

Capítulo 5

Conclusión

Una vez finalizado el proyecto, se prosigue con las conclusiones y las líneas futuras del mismo.

En primer lugar, se ha dejado constancia de la facilidad de realizar un sistema empotrado simple. Con unos pocos conocimientos del hardware y software utilizados y con un seguimiento paso a paso de las instrucciones, se puede implementar de manera fácil y sencilla. Si bien, cabe destacar que durante el proyecto se han tenido que solucionar muchos problemas, algunos usuales con fácil solución y otros no tan típicos que requieren de una gran capacidad para buscar alternativas y también de mucha paciencia. Este tipo de proyectos suelen ser desesperantes cuando no se obtienen los resultados deseados o sobre todo cuando la compilación da problemas que no sabes como solucionar. En estos casos, lo importante es mantener la calma y ampliar la búsqueda de información.

Por otro lado, se ha comprobado la funcionalidad de los ECC y su clara ventaja frente al no uso de los mismos. Se demuestra que los códigos ultrarápidos son óptimos en procesos donde se requieren bajas latencias y donde es asequible un aumento de la redundancia, como en los registros (memorias de alta velocidad) de los microcontroladores.

También se verifica que los códigos han sido entrelazados correctamente ya que todos ellos están formados a partir de la misma matriz H32'. Este hecho es un claro avance en las líneas actuales de los ECC ya que permite la corrección de un mayor número de errores y por consiguiente, un aumento de la confiabilidad del sistema.

5.1. Líneas futuras

El aumento del número de sensores y actuadores posibilita la automatización de diferentes proyectos. En este caso se sugiere el empleo del sistema para un huerto particular, por tanto, se pueden añadir sensores de humedad de la tierra, luminosidad y otros para parámetros necesarios en los cultivos. Como actuadores se pueden añadir medios de refrigeración y ventilación como puede ser un aspersor automático. Se ha buscado información sobre como añadir estos sensores y actuadores al proyecto y se ha constatado que no presentan mayores complicaciones. Una vez se tienen las bases de como funciona el ecosistema STM y con el conocimiento requerido de cada componente que se añada es sencillo incrementar el sistema empotrado. Cuantos más componentes se añadan más datos se obtendrán y más eficaz será el sistema de control.

Por otro lado, los ECC tienen un abanico enorme de aplicación y suponen un gran avance en la confiabilidad de los sistemas informáticos. En este siglo, la automatización está teniendo un gran desarrollo el cual necesita garantizar la mayor confiabilidad posible. Por ello, los ECC son de vital importancia ya que cuantos menos errores se produzcan en estos sistemas, más fiables y eficientes serán los mismo. Uno de los casos más importantes es el de los coches autónomos donde un error puede ocasionar la muerte de una persona. Es necesario seguir avanzando en la creación de códigos de errores que abarquen un mayor número de corrección de errores intentando reducir el área utilizada, el consumo, la redundancia, así como el tiempo de ejecución y el tamaño del código.

Bibliografía

[1] ST-Electronics. “User Manual. Discovery kit with STM32F429ZI MCU”. Recuperado (28/11/2020) de: /www.st.com/resource/en/usermanual.pdf

[2] Tecnología fácil. ”¿Qué es un controlador ”Mass Storage¿Recuperado (01/12/2021) de: https://tecnologia-facil.com/que-es/que-es-un-controlador-mass-storage/.

[3] Yiu, Joseph (2013). ”The definitive guide to ARM Cortex-M3 and Cortex-M4 processors”. ISBN: 0124080820.

[4] Perles,A (2019). “ARM Cortex-M práctico 1 - Introducción a los microcontroladores STM32 de ST”.https://aperles.blogs.upv.es/.

[5] ARM Developer.“Cortex-M4 Processor Datasheet”. Recuperado (04/12/2020) de: https://developer.arm.com/ip-products/processors/cortex-m/cortex-m4.

[6] Unknown[ABCelectronica].”Sensor de humedad y temperatura DHT11 y DHT22”. Recuperado (05/12/2020) de: http://www.datasheet.es/PDF/792209/DHT22-pdf.html.

[7] Last minute engineers. ”How DHT11 DHT22 Sensors Work and Interface With Arduino”. Recuperado (12/12/2020) de: HowDHT11DHT22SensorsWork&InterfaceWithArduino.

[8] Controllers Tech.”How to use DHT22 with STM32”. Recuperado (05/12/2020) de: https://controllerstech.com/temperature-measurement-using-dht22-in-stm32/.

[9] ST-Electronics. ”STM32Cube Ecosystem”. Recuperado (14/12/2020) de: https://www.st.com/content/st_com/en/stm32cube-ecosystem.html.

[10] ST-Electronics. ”STM32Cube initialization code generator”. Recuperado (14/12/2020) de: www.st.com/content/st.com/en/products/development/tools.html

[11] ST-Electronics. ”Integrated Development Environment for STM32”. Recuperado (14/12/2020) de: www.st.com/content/st-com/en/products/development-tools.html.

[12] ST-Electronics. ”STM32 ST-Link Utility”. Recuperado (14/12/2020) de : https:

//www.st.com/en/development-tools/stsw-link004.html.

[13] The International Technology Roadmap for Semiconductors 2013. [Online]. Available at: http://www.itrs2.net/2013-itrs.html

[14] S.K. Kurinec and K. Iniewsky. Nanoscale Semiconductor Memories: Technology and Application, CRC Press, Taylor and Francis Group, 2014.

[15] E. Ibe, H. Taniguchi, Y. Yahagi, K. Shimbo, and T. Toba, "Impact of scaling on neutron-induced soft error in SRAMs from a 250 nm to a 22 nm design rule", IEEE Trans. Electron Devices, vol. 57, no. 7, pp. 1527–1538, July 2010.

[16] E. Fujiwara, Code Design for Dependable Systems: Theory and Practical Application, Ed. Wiley-Interscience, 2006.

[17]R. W. Hamming, "Error detecting and error correcting codes," Bell System Technical Journal, vol. 29, pp. 147–160, 1950.

[18] C.L. Chen and M.Y. Hsiao, "Error-correcting codes for semiconductor memory applications: a state-of-the-art review", IBM Journal of Research and Development, vol. 58, no. 2, pp. 124–134, March 1984.

[19] Saiz-Adalid, Luis-J; Gracia-Moran, Joaquin; Gil-Tomas, Daniel; Baraza-Calvo, J.-Carlos; Gil-Vicente, Pedro-J " Ultrafast Codes for Multiple Adjacent Error Correction and Double Error Detection". DOI: 10.1109/ACCESS.2019.2947315.

[20] Micheloni,R. Marelli,A y Ravasio,R. (2008). " Error Correction Codes for Non-Volatile Memories". ISBN 1-281-49224-8.

[21] Neubauer, A., Freudenberger, J., Kühn, V.: "Coding Theory: Algorithms, Architectures and Applications". John Wiley and Sons (2007).

[22] J. Gracia-Moran, L.J. Saiz-Adalid, D. Gil-Tomás, and P.J. Gil-Vicente, "Improving Error Correction Codes for Multiple Cell Upsets in Space Applications", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26(10), pp. 2132-2142, October 2018.

[23] Fault Tolerant Systems Group, DISCA (2020). "Evolving Error Control Codes for increasing the robustness of systems and communications. Part one: Introduction to error detection and correction codes".

[24] Saiz-Adalid, Luis-J ; Gracia-Moran, Joaquin ; Gil-Tomas, Daniel ; Baraza-Calvo, J.-Carlos ; Gil-Vicente, Pedro-J; Ruíz-García, J.-Carlos. "Flexible Unequal Error Control Codes with Selectable Error Detection and Correction Levels". en Proc. SAFECOMP,Toulouse,France, 2013, pp. 178–189.

[25] Benso,A. Prinetto, P. "Fault Injection Techniques and Tools for Embedded Systems Reliability Evaluation". ISBN: 0-306-48711-X (2003).

[26] Franco, J. "What is Direct Memory Access (DMA)and Why Should We Know About it?". Recuperado (21/12/2020) de: http://gauss.ececs.uc.edu/Courses/c4029/lectures/dma.pdf.

[27] ST-Electronics. "STM32F7-Chrom-ART". Recuperado (21/12/2020) de : https://www.st.com/content/ccc/resource/training/technical/product_training/group0/06/38/cb/60/21/70/4e/d8/STM32F7_System_DMA2D/files/STM32F7_System_DMA2D.pdf/jcr:content/translations/en.STM32F7_System_DMA2D.pdf.

[28] ST-Electronics. "STM32F7-LTDC". Recuperado (21/12/2020) de: https://www.st.com/content/ccc/resource/training/technical/product_training/group0/3f/7b/af/97/88/ba/48/33/STM32F7_Peripheral_LTDC/files/STM32F7_Peripheral_LTDC.pdf/jcr:content/translations/en.STM32F7_Peripheral_LTDC.pdf.

[29] ST-Electronics. "STM32F7-FMC". Recuperado (22/12/2020) de : https://www.st.com/content/ccc/resource/training/technical/product_training/group0/51/a3/68/fd/47/6d/43/b8/STM32F7_Memory_FSMC/files/STM32F7_Memory_FSMC.pdf/_jcr_content/translations/en.STM32F7_Memory_FSMC.pdf.

[30] Main, John. "I2C tutorial". Recuperado (08/01/2021) de: https://www.best-microcontroller-projects.com/i2c-tutorial.html

[31] Nayak, K. "Master Microcontroller and Embedded Driver Development (MCU1)". Udemy. https://www.udemy.com/course/mastering-microcontroller.

ANEXOS

Dentro del anexo se diferencian cuatro capítulos. En primer lugar se tiene el Pliego de condiciones donde se especifica el material y equipo que se ha utilizado así como el entorno de trabajo. Luego se encuentra el Presupuesto del proyecto seguido de los planos de los equipos usados. Por último, se localizan las funciones codificar y decodificar del ECC.

Anexo A

Pliego de condiciones

A.1. Material y equipo

En este apartado se deben diferenciar dos partes:

- **Hardware:**
  - Por un lado, se necesita el sistema empotrado formado por un microcontrolador y un sensor. En este caso, el microcontrolador elegido es el STM32F429i-Discovery y el sensor utilizado es el DHT22. Sin embargo, se podrían utilizar una gran variedad de microcontroladores de la familia STM (teniendo en cuenta el cambio de librerías) o de otras IDEs como Arduino (con cambio de librerías y código). A su vez, se podrían elegir otros sensores en función de las necesidades del proyecto.
  - Por otro lado, se necesita un equipo informático para trabajar con este sistema empotrado. En cuanto al Hardware, el único requisito es que el equipo (portátil o PC fijo) tenga entrada USB para poder conectar la tarjeta.
- **Software:**
  - Los programas informáticos que se han utilizado son:
    - STM32Cube-IDE
    - STM32Cube-Mx
    - STLink-Utility

    El equipo utilizado para hacer funcionar estos programas cumple con las especificaciones siguientes:

- Tipo: Portátil.
- Marca: Acer Aspire 5.
- Procesador: Intel Core i7-10510U.
- RAM: 8,00 Gb.
- Sistema operativo: Windows 10 Home.
- SDD: 500 Gb.

Estos programas no necesitan gran cantidad de prestaciones para poder ejecutarlos, por tanto, basta tener un ordenador con más de 4Gb de RAM y espacio suficiente en el disco duro para guardar los programas y proyectos.

A.2. Entorno de trabajo

Este proyecto se ha realizado tanto en el laboratorio del Grupo de Sistemas Tolerantes a Fallos del Instituto ITACA en la Universidad Politécnica de Valencia como en la casa del alumno gracias a la portabilidad del material y equipos necesarios.

El entorno de trabajo para este proyecto no necesita requisitos específicos y depende de la comodidad del usuario y las preferencias que tenga a la hora de trabajar. Cabe resaltar la obligatoriedad de llevar a cabo las medidas preventivas anti-Covid cuando se ha trabajado desde el aula de la universidad.

Anexo B

Presupuesto

B.1. Equipo y material

Concepto	Cantidad	Coste unitario (€)	Subtotal(€)
Tarjeta STM32F429i-Discovery	1	27,66	27,66
Sensor DHT22	1	9,99	9,99
Cableado	1	6,59	6,59
Portátil Acer Aspire 5	1	680	680
Coste total: Equipo y material			**724,24 €**

Tabla B.1: Coste del Equipo y material

B.2. Mano de obra

Concepto	Cantidad (horas)	Coste unitario (€/h)	Subtotal (€)
Ingeniero en prácticas	300	18	5400
Ingeniero tutor UPV	25	60	1500
Ingeniero tutor	25	60	1500
Coste total: Mano de obra			**8400 €**

Tabla B.2: Coste de la Mano de obra

B.3. Consumo

Para calcular el coste unitario se ha observado el consumo del ordenador en los puntos de mayor carga. De ahí se ha observado que el PC trabaja al 85 % lo que equivale a un consumo de 0.148 kW/h. Se ha calculado un precio medio aproximado de 0.11365 €/ kWh. Con esto se obtiene el **coste unitario de consumo: 0.01682 €/h.**

Concepto	Cantidad (horas)	Coste unitario (€/h)	Subtotal(€)
STM32Cube-Mx	175	0,0168	2,94
STM32Cube-IDE	250	0,0168	4,2
STLINK-Utility	25	0,0168	0,42
Búsqueda de información	150	0,0168	2,52
Memoria	120	0,0168	2,01
Coste total: Consumo			**12,09 €**

Tabla B.3: Coste del Consumo

B.4. Presupuesto global

Concepto	Coste (€)
Equipo y material	724,24
Mano de obra	8400
Consumo	12,09
TOTAL	**9136.33 €**
Costes indirectos	5 %
TOTAL	**9593.15 €**
Costes IVA	21 %
TOTAL	**11607.71 €**

Tabla B.4: Coste total del proyecto

Anexo C

Planos

C.1. STM32F429i-Discovery

Figura C.1: Tarjeta STM32F429i-Discovery

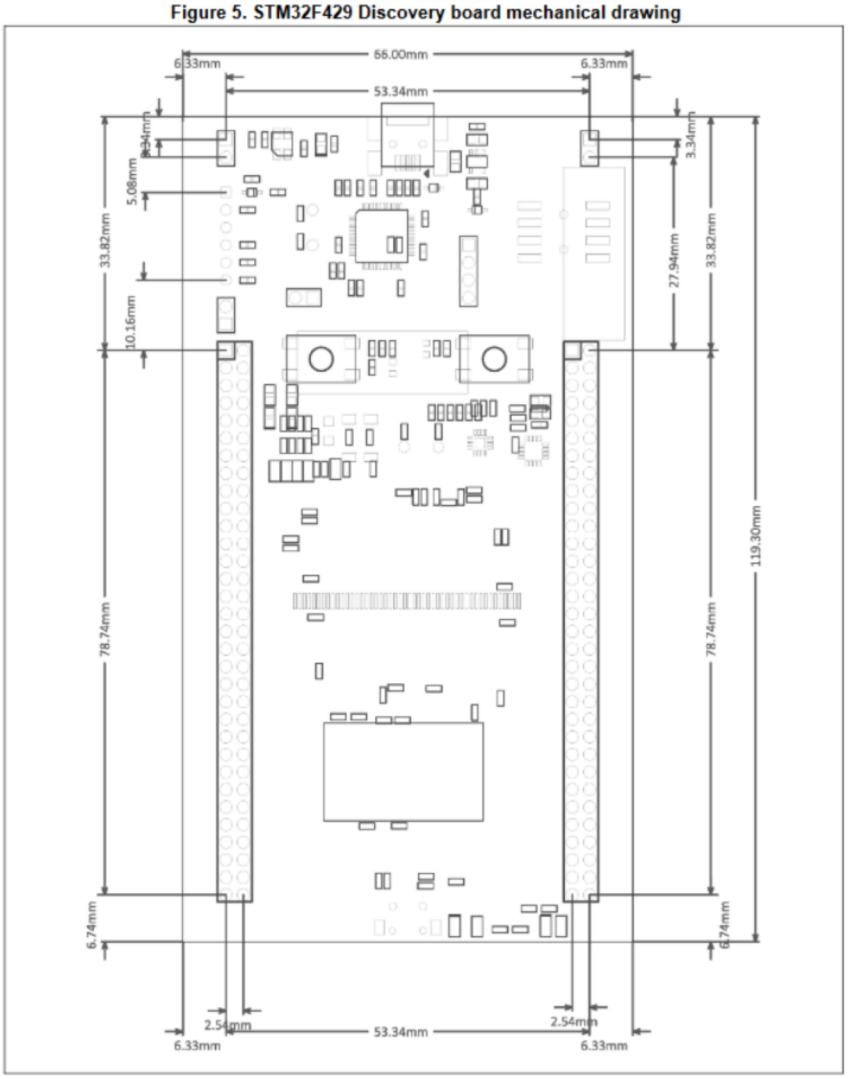


Figura C.2: Plano STM32F429i-Discovery

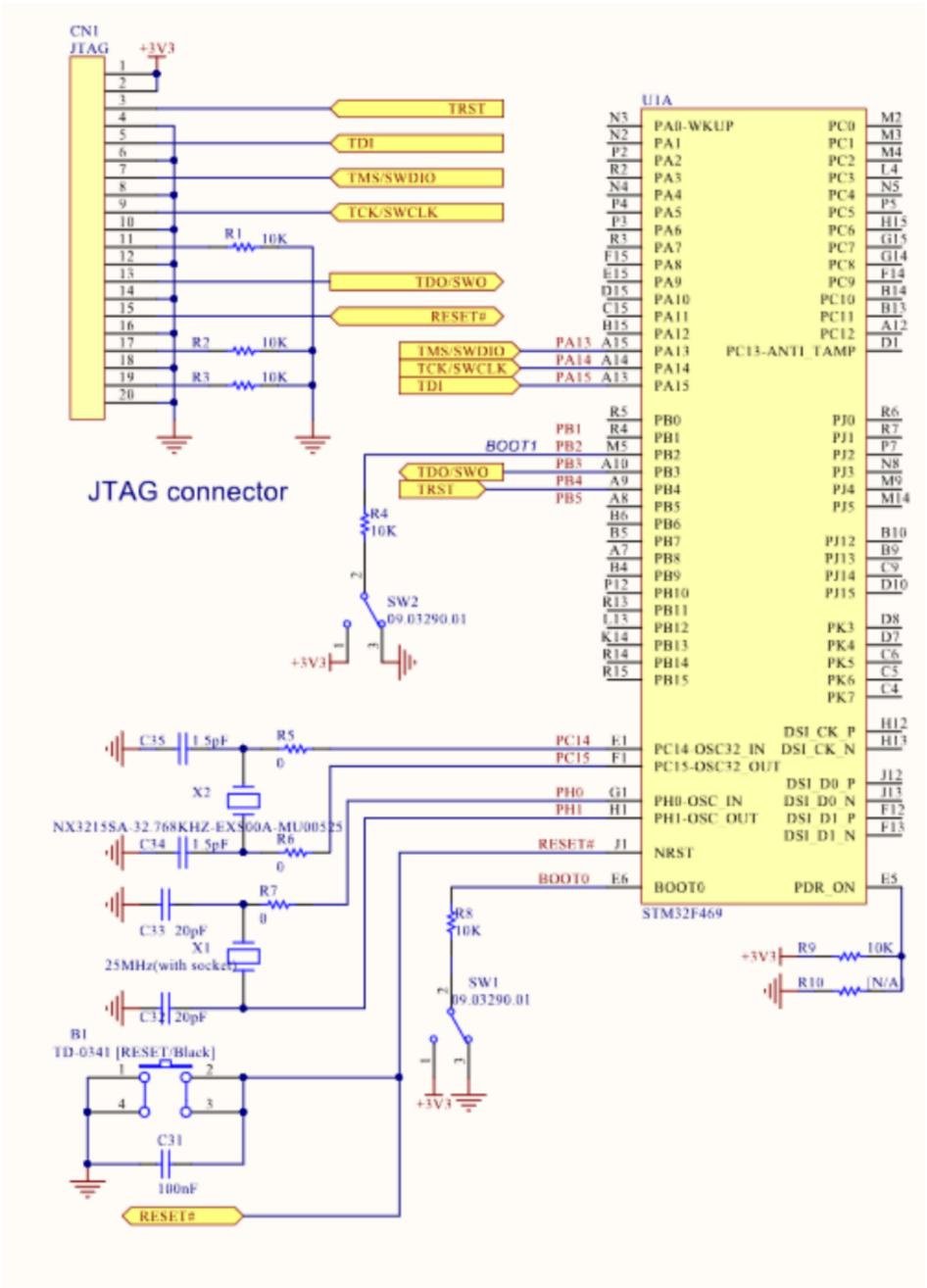


Figura C.3: Esquema de referencia 1

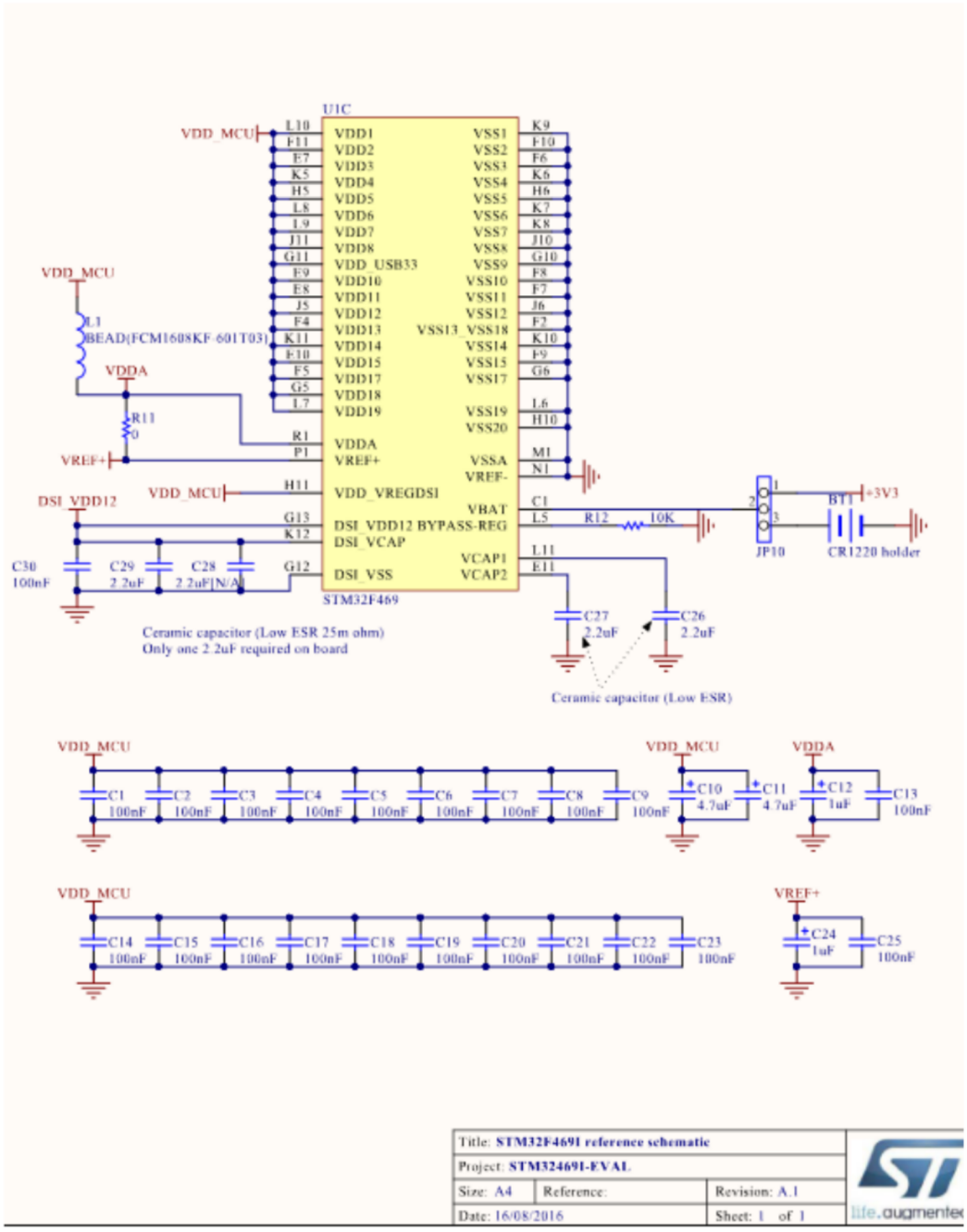


Figura C.4: Esquema de referencia 2

C.2. Sensor DHT22

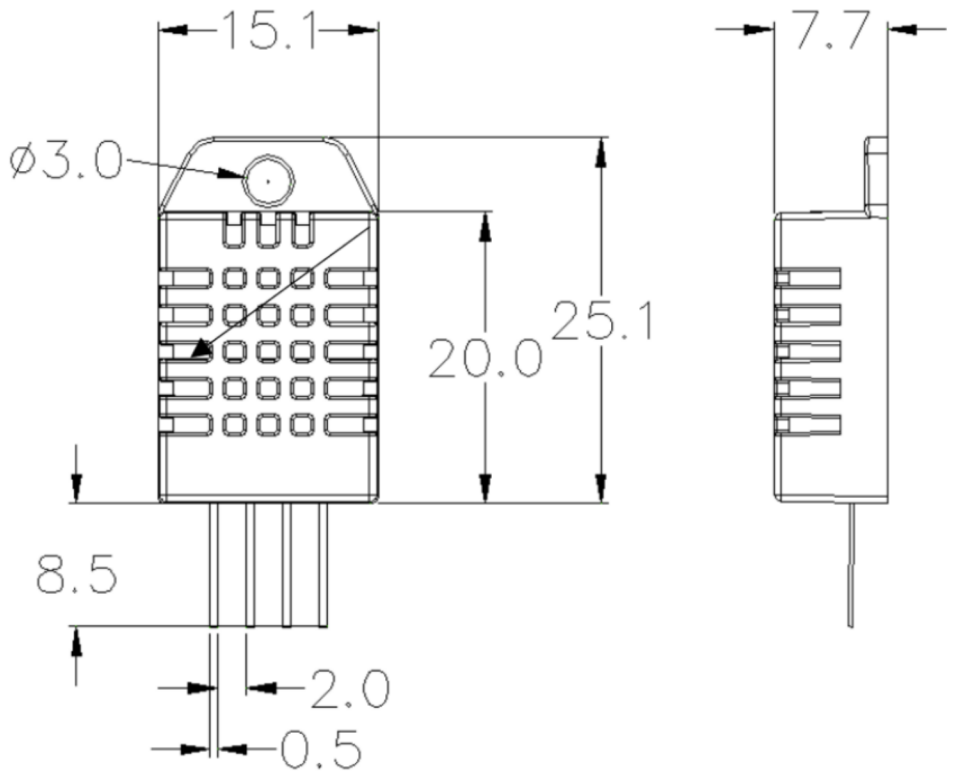


Figura C.5: Plano DHT22

C.3. LED RGB

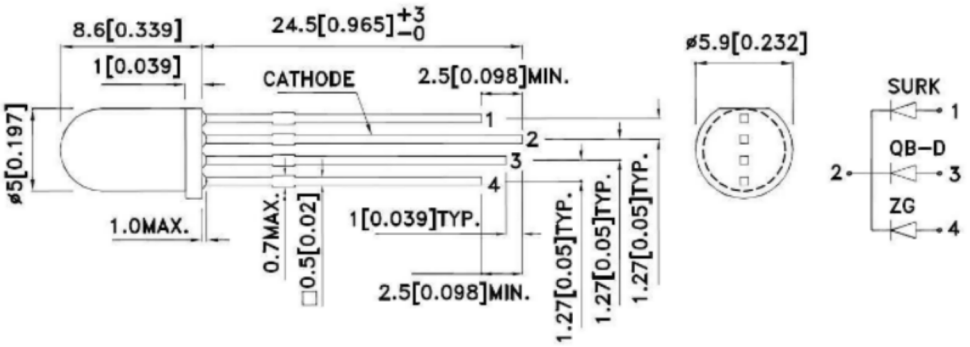


Figura C.6: Plano LED RGB

Anexo D

Código

Función codificar:

```
void codificar (bitvector *enc, bitvector *dec)
{
enc[ 0] = dec[0]^dec[8]^dec[24];
enc[ 4] = dec[4]^dec[20]^dec[28];
enc[ 8] = dec[0]^dec[8]^dec[16];
enc[12] = dec[4]^dec[12]^dec[20];
enc[16] = dec[0]^dec[16]^dec[24];
enc[20] = dec[4]^dec[12]^dec[28];
enc[24] = dec[8]^dec[16]^dec[24];
enc[28] = dec[12]^dec[20]^dec[28];
enc[ 1] = dec[1]^dec[9]^dec[25];
enc[ 5] = dec[5]^dec[21]^dec[29];
enc[ 9] = dec[1]^dec[9]^dec[17];
enc[13] = dec[5]^dec[13]^dec[21];
enc[17] = dec[1]^dec[17]^dec[25];
enc[21] = dec[5]^dec[13]^dec[29];
enc[25] = dec[9]^dec[17]^dec[25];
enc[29] = dec[13]^dec[21]^dec[29];
enc[ 2] = dec[2]^dec[10]^dec[26];
enc[ 6] = dec[6]^dec[22]^dec[30];
enc[10] = dec[2]^dec[10]^dec[18];
enc[14] = dec[6]^dec[14]^dec[22];
enc[18] = dec[2]^dec[18]^dec[26];
enc[22] = dec[6]^dec[14]^dec[30];
enc[26] = dec[10]^dec[18]^dec[26];
enc[30] = dec[14]^dec[22]^dec[30];
enc[ 3] = dec[3]^dec[11]^dec[27];
enc[ 7] = dec[7]^dec[23]^dec[31];
enc[11] = dec[3]^dec[11]^dec[19];
enc[15] = dec[7]^dec[15]^dec[23];
enc[19] = dec[3]^dec[19]^dec[27];
enc[23] = dec[7]^dec[15]^dec[31];
enc[27] = dec[11]^dec[19]^dec[27];
enc[31] = dec[15]^dec[23]^dec[31];
enc[32] = dec[0];
enc[33] = dec[1];
enc[34] = dec[2];
enc[35] = dec[3];
enc[36] = dec[4];
enc[37] = dec[5];
enc[38] = dec[6];
enc[39] = dec[7];
enc[40] = dec[8];
enc[41] = dec[9];
enc[42] = dec[10];
enc[43] = dec[11];
enc[44] = dec[12];
enc[45] = dec[13];
enc[46] = dec[14];
enc[47] = dec[15];
enc[48] = dec[16];
enc[49] = dec[17];
enc[50] = dec[18];
enc[51] = dec[19];
enc[52] = dec[20];
enc[53] = dec[21];
enc[54] = dec[22];
enc[55] = dec[23];
enc[56] = dec[24];
enc[57] = dec[25];
enc[58] = dec[26];
enc[59] = dec[27];
enc[60] = dec[28];
enc[61] = dec[29];
enc[62] = dec[30];
enc[63] = dec[31];
}
```

Figura D.1: Codificar

Funciones decodificar:

- Tipo 8:

```
void decodificar (bitvector *enc, bitvector *dec)
{
    bitvector sindrome[N-M];
    bitvector o[N];

sindrome[ 0] = enc[0]^enc[32]^enc[40]^enc[56];
sindrome[ 1] = enc[4]^enc[36]^enc[52]^enc[60];
sindrome[ 2] = enc[8]^enc[32]^enc[40]^enc[48];
sindrome[ 3] = enc[12]^enc[36]^enc[44]^enc[52];
sindrome[ 4] = enc[16]^enc[32]^enc[48]^enc[56];
sindrome[ 5] = enc[20]^enc[36]^enc[44]^enc[60];
sindrome[ 6] = enc[24]^enc[40]^enc[48]^enc[56];
sindrome[ 7] = enc[28]^enc[44]^enc[52]^enc[60];
sindrome[ 8] = enc[1]^enc[33]^enc[41]^enc[57];
sindrome[ 9] = enc[5]^enc[37]^enc[53]^enc[61];
sindrome[10] = enc[9]^enc[33]^enc[41]^enc[49];
sindrome[11] = enc[13]^enc[37]^enc[45]^enc[53];
sindrome[12] = enc[17]^enc[33]^enc[49]^enc[57];
sindrome[13] = enc[21]^enc[37]^enc[45]^enc[61];
sindrome[14] = enc[25]^enc[41]^enc[49]^enc[57];
sindrome[15] = enc[29]^enc[45]^enc[53]^enc[61];
sindrome[16] = enc[2]^enc[34]^enc[42]^enc[58];
sindrome[17] = enc[6]^enc[38]^enc[54]^enc[62];
sindrome[18] = enc[10]^enc[34]^enc[42]^enc[50];
sindrome[19] = enc[14]^enc[38]^enc[46]^enc[54];
sindrome[20] = enc[18]^enc[34]^enc[50]^enc[58];
sindrome[21] = enc[22]^enc[38]^enc[46]^enc[62];
sindrome[22] = enc[26]^enc[42]^enc[50]^enc[58];
sindrome[23] = enc[30]^enc[46]^enc[54]^enc[62];
sindrome[24] = enc[3]^enc[35]^enc[43]^enc[59];
sindrome[25] = enc[7]^enc[39]^enc[55]^enc[63];
sindrome[26] = enc[11]^enc[35]^enc[43]^enc[51];
sindrome[27] = enc[15]^enc[39]^enc[47]^enc[55];
sindrome[28] = enc[19]^enc[35]^enc[51]^enc[59];
sindrome[29] = enc[23]^enc[39]^enc[47]^enc[63];
sindrome[30] = enc[27]^enc[43]^enc[51]^enc[59];
sindrome[31] = enc[31]^enc[47]^enc[55]^enc[63];

o[60] = (sindrome[5]&!sindrome[3]&sindrome[1]);
o[56] = (sindrome[4]&!sindrome[2]&sindrome[0]);
o[52] = (!sindrome[5]&sindrome[3]&sindrome[1]);
o[48] = (sindrome[4]&sindrome[2]&!sindrome[0]);
o[44] = (sindrome[5]&sindrome[3]&!sindrome[1]);
o[40] = (!sindrome[4]&sindrome[2]&sindrome[0]);
o[36] = (sindrome[5]&sindrome[3]&sindrome[1]);
o[32] = (sindrome[4]&sindrome[2]&sindrome[0]);

o[61] = (sindrome[13]&!sindrome[11]&sindrome[9]);
o[57] = (sindrome[12]&!sindrome[10]&sindrome[8]);
o[53] = (!sindrome[13]&sindrome[11]&sindrome[9]);
o[49] = (sindrome[12]&sindrome[10]&!sindrome[8]);
o[45] = (sindrome[13]&sindrome[11]&!sindrome[9]);
o[41] = (!sindrome[12]&sindrome[10]&sindrome[8]);
o[37] = (sindrome[13]&sindrome[11]&sindrome[9]);
o[33] = (sindrome[12]&sindrome[10]&sindrome[8]);

o[62] = (sindrome[21]&!sindrome[19]&sindrome[17]);
o[58] = (sindrome[20]&!sindrome[18]&sindrome[16]);
o[54] = (!sindrome[21]&sindrome[19]&sindrome[17]);
o[50] = (sindrome[20]&sindrome[18]&!sindrome[16]);
o[46] = (sindrome[21]&sindrome[19]&!sindrome[17]);
o[42] = (!sindrome[20]&sindrome[18]&sindrome[16]);
o[38] = (sindrome[21]&sindrome[19]&sindrome[17]);
o[34] = (sindrome[20]&sindrome[18]&sindrome[16]);

o[63] = (sindrome[29]&!sindrome[27]&sindrome[25]);
o[59] = (sindrome[28]&!sindrome[26]&sindrome[24]);
o[55] = (!sindrome[29]&sindrome[27]&sindrome[25]);
o[51] = (sindrome[28]&sindrome[26]&!sindrome[24]);
o[47] = (sindrome[29]&sindrome[27]&!sindrome[25]);
o[43] = (!sindrome[28]&sindrome[26]&sindrome[24]);
o[39] = (sindrome[29]&sindrome[27]&sindrome[25]);
o[35] = (sindrome[28]&sindrome[26]&sindrome[24]);
```

Figura D.2: Decodificar (8)

```
dec[ 0] = enc[32]^o[32];
dec[ 1] = enc[33]^o[33];
dec[ 2] = enc[34]^o[34];
dec[ 3] = enc[35]^o[35];
dec[ 4] = enc[36]^o[36];
dec[ 5] = enc[37]^o[37];
dec[ 6] = enc[38]^o[38];
dec[ 7] = enc[39]^o[39];
dec[ 8] = enc[40]^o[40];
dec[ 9] = enc[41]^o[41];
dec[10] = enc[42]^o[42];
dec[11] = enc[43]^o[43];
dec[12] = enc[44]^o[44];
dec[13] = enc[45]^o[45];
dec[14] = enc[46]^o[46];
dec[15] = enc[47]^o[47];
dec[16] = enc[48]^o[48];
dec[17] = enc[49]^o[49];
dec[18] = enc[50]^o[50];
dec[19] = enc[51]^o[51];
dec[20] = enc[52]^o[52];
dec[21] = enc[53]^o[53];
dec[22] = enc[54]^o[54];
dec[23] = enc[55]^o[55];
dec[24] = enc[56]^o[56];
dec[25] = enc[57]^o[57];
dec[26] = enc[58]^o[58];
dec[27] = enc[59]^o[59];
dec[28] = enc[60]^o[60];
dec[29] = enc[61]^o[61];
dec[30] = enc[62]^o[62];
dec[31] = enc[63]^o[63];
```

Figura D.3: Decodificar (8)

```
ErrD = (sindrome[7]&sindrome[5]&sindrome[3]&sindrome[1]) | (!sindrome[7]&!sindrome[5]&sindrome[3]&sindrome[1]) |
       (!sindrome[7]&sindrome[5]&!sindrome[3]&sindrome[1]) | (sindrome[7]&!sindrome[5]&!sindrome[3]&sindrome[1]) |
       (!sindrome[7]&sindrome[5]&sindrome[3]&!sindrome[1]) | (sindrome[7]&!sindrome[5]&sindrome[3]&!sindrome[1]) |
       (sindrome[7]&sindrome[5]&!sindrome[3]&!sindrome[1]) | (sindrome[6]&sindrome[4]&sindrome[2]&sindrome[0]) |
       (!sindrome[6]&!sindrome[4]&sindrome[2]&sindrome[0]) | (!sindrome[6]&sindrome[4]&!sindrome[2]&sindrome[0]) |
       (sindrome[6]&!sindrome[4]&!sindrome[2]&sindrome[0]) | (!sindrome[6]&sindrome[4]&sindrome[2]&!sindrome[0]) |
       (sindrome[6]&!sindrome[4]&sindrome[2]&!sindrome[0]) | (sindrome[6]&sindrome[4]&!sindrome[2]&!sindrome[0]) |
       (sindrome[5]&!sindrome[3]&sindrome[2]) | (sindrome[7]&!sindrome[6]&sindrome[3]&sindrome[2]) |
       (sindrome[7]&sindrome[6]&!sindrome[3]&sindrome[2]) | (!sindrome[5]&sindrome[4]&!sindrome[3]&sindrome[1]) |
       (sindrome[6]&!sindrome[4]&!sindrome[3]&sindrome[1]) | (sindrome[6]&!sindrome[5]&sindrome[3]&!sindrome[1]) |
       (sindrome[7]&sindrome[4]&!sindrome[2]&!sindrome[1]) | (!sindrome[5]&sindrome[3]&sindrome[2]&sindrome[0]) |
       (!sindrome[7]&sindrome[5]&!sindrome[2]&sindrome[0]) | (sindrome[7]&!sindrome[4]&!sindrome[2]&sindrome[0]) |
       (sindrome[6]&sindrome[3]&!sindrome[2]&!sindrome[0]) | (sindrome[5]&sindrome[2]&sindrome[1]&!sindrome[0]) |
       (sindrome[4]&!sindrome[2]&sindrome[1]&!sindrome[0]) |
       (sindrome[15]&sindrome[13]&sindrome[11]&sindrome[9]) | (!sindrome[15]&!sindrome[13]&sindrome[11]&sindrome[9]) |
       (!sindrome[15]&sindrome[13]&!sindrome[11]&sindrome[9]) | (sindrome[15]&!sindrome[13]&!sindrome[11]&sindrome[9]) |
       (!sindrome[15]&sindrome[13]&sindrome[11]&!sindrome[9]) | (sindrome[15]&!sindrome[13]&sindrome[11]&!sindrome[9]) |
       (sindrome[15]&sindrome[13]&!sindrome[11]&!sindrome[9]) | (sindrome[14]&sindrome[12]&sindrome[10]&sindrome[8]) |
       (!sindrome[14]&!sindrome[12]&sindrome[10]&sindrome[8]) | (!sindrome[14]&sindrome[12]&!sindrome[10]&sindrome[8]) |
       (sindrome[14]&!sindrome[12]&!sindrome[10]&sindrome[8]) | (!sindrome[14]&sindrome[12]&sindrome[10]&!sindrome[8]) |
       (sindrome[14]&!sindrome[12]&sindrome[10]&!sindrome[8]) | (sindrome[14]&sindrome[12]&!sindrome[10]&!sindrome[8]) |
       (sindrome[13]&!sindrome[11]&sindrome[10]) | (sindrome[15]&!sindrome[14]&sindrome[11]&sindrome[10]) |
       (sindrome[15]&sindrome[14]&!sindrome[11]&sindrome[10]) | (!sindrome[13]&sindrome[12]&!sindrome[11]&sindrome[9]) |
       (sindrome[14]&!sindrome[12]&!sindrome[11]&sindrome[9]) | (sindrome[14]&!sindrome[13]&sindrome[11]&!sindrome[9]) |
       (sindrome[15]&sindrome[12]&!sindrome[10]&!sindrome[9]) | (!sindrome[13]&sindrome[11]&sindrome[10]&sindrome[8]) |
       (!sindrome[15]&sindrome[13]&!sindrome[10]&sindrome[8]) | (sindrome[15]&!sindrome[12]&!sindrome[10]&sindrome[8]) |
       (sindrome[14]&sindrome[11]&!sindrome[10]&!sindrome[8]) | (sindrome[13]&sindrome[10]&sindrome[9]&!sindrome[8]) |
       (sindrome[12]&!sindrome[10]&sindrome[9]&!sindrome[8]) |
       (sindrome[23]&sindrome[21]&sindrome[19]&sindrome[17]) | (!sindrome[23]&!sindrome[21]&sindrome[19]&sindrome[17]) |
       (!sindrome[23]&sindrome[21]&!sindrome[19]&sindrome[17]) | (sindrome[23]&!sindrome[21]&!sindrome[19]&sindrome[17]) |
       (!sindrome[23]&sindrome[21]&sindrome[19]&!sindrome[17]) | (sindrome[23]&!sindrome[21]&sindrome[19]&!sindrome[17]) |
       (sindrome[23]&sindrome[21]&!sindrome[19]&!sindrome[17]) | (sindrome[22]&sindrome[20]&sindrome[18]&sindrome[16]) |
       (!sindrome[22]&!sindrome[20]&sindrome[18]&sindrome[16]) | (!sindrome[22]&sindrome[20]&!sindrome[18]&sindrome[16]) |
       (sindrome[22]&!sindrome[20]&!sindrome[18]&sindrome[16]) | (!sindrome[22]&sindrome[20]&sindrome[18]&!sindrome[16]) |
       (sindrome[22]&!sindrome[20]&sindrome[18]&!sindrome[16]) | (sindrome[22]&sindrome[20]&!sindrome[18]&!sindrome[16]) |
       (sindrome[21]&!sindrome[19]&sindrome[18]) | (sindrome[23]&!sindrome[22]&sindrome[19]&sindrome[18]) |
       (sindrome[23]&sindrome[22]&!sindrome[19]&sindrome[18]) | (!sindrome[21]&sindrome[20]&!sindrome[19]&sindrome[17]) |
       (sindrome[22]&!sindrome[20]&!sindrome[19]&sindrome[17]) | (sindrome[22]&!sindrome[21]&sindrome[19]&!sindrome[17]) |
       (sindrome[23]&sindrome[20]&!sindrome[18]&!sindrome[17]) | (!sindrome[21]&sindrome[19]&sindrome[18]&sindrome[16]) |
       (!sindrome[23]&sindrome[21]&!sindrome[18]&sindrome[16]) | (sindrome[23]&!sindrome[20]&!sindrome[18]&sindrome[16]) |
       (sindrome[22]&sindrome[19]&!sindrome[18]&!sindrome[16]) | (sindrome[21]&sindrome[18]&sindrome[17]&!sindrome[16]) |
       (sindrome[20]&!sindrome[18]&sindrome[17]&!sindrome[16]) |
       (sindrome[31]&sindrome[29]&sindrome[27]&sindrome[25]) | (!sindrome[31]&!sindrome[29]&sindrome[27]&sindrome[25]) |
       (!sindrome[31]&sindrome[29]&!sindrome[27]&sindrome[25]) | (sindrome[31]&!sindrome[29]&!sindrome[27]&sindrome[25]) |
       (!sindrome[31]&sindrome[29]&sindrome[27]&!sindrome[25]) | (sindrome[31]&!sindrome[29]&sindrome[27]&!sindrome[25]) |
       (sindrome[31]&sindrome[29]&!sindrome[27]&!sindrome[25]) | (sindrome[30]&sindrome[28]&sindrome[26]&sindrome[24]) |
       (!sindrome[30]&!sindrome[28]&sindrome[26]&sindrome[24]) | (!sindrome[30]&sindrome[28]&!sindrome[26]&sindrome[24]) |
       (sindrome[30]&!sindrome[28]&!sindrome[26]&sindrome[24]) | (!sindrome[30]&sindrome[28]&sindrome[26]&!sindrome[24]) |
       (sindrome[30]&!sindrome[28]&sindrome[26]&!sindrome[24]) | (sindrome[30]&sindrome[28]&!sindrome[26]&!sindrome[24]) |
       (sindrome[29]&!sindrome[27]&sindrome[26]) | (sindrome[31]&!sindrome[30]&sindrome[27]&sindrome[26]) |
       (sindrome[31]&sindrome[30]&!sindrome[27]&sindrome[26]) | (!sindrome[29]&sindrome[28]&!sindrome[27]&sindrome[25]) |
       (sindrome[30]&!sindrome[28]&!sindrome[27]&sindrome[25]) | (sindrome[30]&!sindrome[29]&sindrome[27]&!sindrome[25]) |
       (sindrome[31]&sindrome[28]&!sindrome[26]&!sindrome[25]) | (!sindrome[29]&sindrome[27]&sindrome[26]&sindrome[24]) |
       (!sindrome[31]&sindrome[29]&!sindrome[26]&sindrome[24]) | (sindrome[31]&!sindrome[28]&!sindrome[26]&sindrome[24]) |
       (sindrome[30]&sindrome[27]&!sindrome[26]&!sindrome[24]) | (sindrome[29]&sindrome[26]&sindrome[25]&!sindrome[24]) |
       (sindrome[28]&!sindrome[26]&sindrome[25]&!sindrome[24]);
}
```

Figura D.4: Decodificar (8)

- Tipo 12:

```
void decodificar (bitvector *enc, bitvector *dec)
{
    bitvector sindrome[N-M];
    bitvector o[N];

sindrome[ 0] = enc[0]^enc[32]^enc[40]^enc[56];
sindrome[ 1] = enc[4]^enc[36]^enc[52]^enc[60];
sindrome[ 2] = enc[8]^enc[32]^enc[40]^enc[48];
sindrome[ 3] = enc[12]^enc[36]^enc[44]^enc[52];
sindrome[ 4] = enc[16]^enc[32]^enc[48]^enc[56];
sindrome[ 5] = enc[20]^enc[36]^enc[44]^enc[60];
sindrome[ 6] = enc[24]^enc[40]^enc[48]^enc[56];
sindrome[ 7] = enc[28]^enc[44]^enc[52]^enc[60];
sindrome[ 8] = enc[1]^enc[33]^enc[41]^enc[57];
sindrome[ 9] = enc[5]^enc[37]^enc[53]^enc[61];
sindrome[10] = enc[9]^enc[33]^enc[41]^enc[49];
sindrome[11] = enc[13]^enc[37]^enc[45]^enc[53];
sindrome[12] = enc[17]^enc[33]^enc[49]^enc[57];
sindrome[13] = enc[21]^enc[37]^enc[45]^enc[61];
sindrome[14] = enc[25]^enc[41]^enc[49]^enc[57];
sindrome[15] = enc[29]^enc[45]^enc[53]^enc[61];
sindrome[16] = enc[2]^enc[34]^enc[42]^enc[58];
sindrome[17] = enc[6]^enc[38]^enc[54]^enc[62];
sindrome[18] = enc[10]^enc[34]^enc[42]^enc[50];
sindrome[19] = enc[14]^enc[38]^enc[46]^enc[54];
sindrome[20] = enc[18]^enc[34]^enc[50]^enc[58];
sindrome[21] = enc[22]^enc[38]^enc[46]^enc[62];
sindrome[22] = enc[26]^enc[42]^enc[50]^enc[58];
sindrome[23] = enc[30]^enc[46]^enc[54]^enc[62];
sindrome[24] = enc[3]^enc[35]^enc[43]^enc[59];
sindrome[25] = enc[7]^enc[39]^enc[55]^enc[63];
sindrome[26] = enc[11]^enc[35]^enc[43]^enc[51];
sindrome[27] = enc[15]^enc[39]^enc[47]^enc[55];
sindrome[28] = enc[19]^enc[35]^enc[51]^enc[59];
sindrome[29] = enc[23]^enc[39]^enc[47]^enc[63];
sindrome[30] = enc[27]^enc[43]^enc[51]^enc[59];
sindrome[31] = enc[31]^enc[47]^enc[55]^enc[63];

o[60] = (sindrome[5]&!sindrome[3]&!sindrome[2]&sindrome[1]) | (sindrome[6]&sindrome[5]&!sindrome[2]&sindrome[0]);
o[56] = (!sindrome[5]&sindrome[3]&sindrome[0]) | (sindrome[6]&!sindrome[2]&sindrome[0]);
o[52] = (sindrome[7]&!sindrome[5]&sindrome[3]) | (sindrome[6]&sindrome[3]&!sindrome[2]&sindrome[0]) | (sindrome[4]&sindrome[2]&sindrome[1]&!sindrome[0]);
o[48] = (sindrome[6]&sindrome[2]&!sindrome[0]) | (sindrome[7]&!sindrome[5]&sindrome[3]&sindrome[2]) | (sindrome[7]&sindrome[4]&sindrome[3]&!sindrome[1]);
o[44] = (sindrome[7]&sindrome[3]&!sindrome[1]) | (sindrome[6]&!sindrome[3]&sindrome[2]&sindrome[1]);
o[40] = (sindrome[6]&!sindrome[4]&sindrome[0]) | (sindrome[6]&sindrome[5]&sindrome[3]&sindrome[1]) | (sindrome[7]&sindrome[3]&!sindrome[1]&sindrome[0]);
o[36] = (sindrome[5]&sindrome[3]&sindrome[1]) | (sindrome[6]&sindrome[2]&sindrome[1]&sindrome[0]);
o[32] = (sindrome[4]&sindrome[2]&sindrome[0]) | (sindrome[5]&sindrome[4]&sindrome[3]&sindrome[1]);

o[61] = (sindrome[13]&!sindrome[11]&!sindrome[10]&sindrome[9]) | (sindrome[14]&sindrome[13]&!sindrome[10]&sindrome[8]);
o[57] = (!sindrome[13]&sindrome[11]&sindrome[8]) | (sindrome[14]&!sindrome[10]&sindrome[8]);
o[53] = (sindrome[15]&!sindrome[13]&sindrome[11]) | (sindrome[14]&sindrome[11]&!sindrome[10]&sindrome[8]) | (sindrome[12]&sindrome[10]&sindrome[9]&!sindrome[8]);
o[49] = (sindrome[14]&sindrome[10]&!sindrome[8]) | (sindrome[15]&!sindrome[13]&sindrome[11]&sindrome[10]) | (sindrome[15]&sindrome[12]&sindrome[11]&!sindrome[9]);
o[45] = (sindrome[15]&sindrome[11]&!sindrome[9]) | (sindrome[14]&!sindrome[11]&sindrome[10]&sindrome[9]);
o[41] = (sindrome[14]&!sindrome[12]&sindrome[8]) | (sindrome[14]&sindrome[13]&sindrome[11]&sindrome[9]) | (sindrome[15]&sindrome[11]&!sindrome[9]&sindrome[8]);
o[37] = (sindrome[13]&sindrome[11]&sindrome[9]) | (sindrome[14]&sindrome[10]&sindrome[9]&sindrome[8]);
o[33] = (sindrome[12]&sindrome[10]&sindrome[8]) | (sindrome[13]&sindrome[12]&sindrome[11]&sindrome[9]);

o[62] = (sindrome[21]&!sindrome[19]&!sindrome[18]&sindrome[17]) | (sindrome[22]&sindrome[21]&!sindrome[18]&sindrome[16]);
o[58] = (!sindrome[21]&sindrome[19]&sindrome[16]) | (sindrome[22]&!sindrome[18]&sindrome[16]);
o[54] = (sindrome[23]&!sindrome[21]&sindrome[19]) | (sindrome[22]&sindrome[19]&!sindrome[18]&sindrome[16]) | (sindrome[20]&sindrome[18]&sindrome[17]&!sindrome[16]);
o[50] = (sindrome[22]&sindrome[18]&!sindrome[16]) | (sindrome[23]&!sindrome[21]&sindrome[19]&sindrome[18]) | (sindrome[23]&sindrome[20]&sindrome[19]&!sindrome[17]);
o[46] = (sindrome[23]&sindrome[19]&!sindrome[17]) | (sindrome[22]&!sindrome[19]&sindrome[18]&sindrome[17]);
o[42] = (sindrome[22]&!sindrome[20]&sindrome[16]) | (sindrome[22]&sindrome[21]&sindrome[19]&sindrome[17]) | (sindrome[23]&sindrome[19]&!sindrome[17]&sindrome[16]);
o[38] = (sindrome[21]&sindrome[19]&sindrome[17]) | (sindrome[22]&sindrome[18]&sindrome[17]&sindrome[16]);
o[34] = (sindrome[20]&sindrome[18]&sindrome[16]) | (sindrome[21]&sindrome[20]&sindrome[19]&sindrome[17]);
```

Figura D.5: Decodificar (12)

```
o[63] = (sindrome[29]&!sindrome[27]&!sindrome[26]&sindrome[25]) |
(sindrome[30]&sindrome[29]&!sindrome[26]&sindrome[24]);

o[59] = (!sindrome[29]&sindrome[27]&sindrome[24]) | (sindrome[30]&!sindrome[26]&sindrome[24]);

o[55] = (sindrome[31]&!sindrome[29]&sindrome[27]) | (sindrome[30]&sindrome[27]&!sindrome[26]&sindrome[24]) |
(sindrome[28]&sindrome[26]&sindrome[25]&!sindrome[24]);

o[51] = (sindrome[30]&sindrome[26]&!sindrome[24]) | (sindrome[31]&!sindrome[29]&sindrome[27]&sindrome[26]) |
(sindrome[31]&sindrome[28]&sindrome[27]&!sindrome[25]);

o[47] = (sindrome[31]&sindrome[27]&!sindrome[25]) | (sindrome[30]&!sindrome[27]&sindrome[26]&sindrome[25]);

o[43] = (sindrome[30]&!sindrome[28]&sindrome[24]) | (sindrome[30]&sindrome[29]&sindrome[27]&sindrome[25]) |
(sindrome[31]&sindrome[27]&!sindrome[25]&sindrome[24]);

o[39] = (sindrome[29]&sindrome[27]&sindrome[25]) | (sindrome[30]&sindrome[26]&sindrome[25]&sindrome[24]);

o[35] = (sindrome[28]&sindrome[26]&sindrome[24]) | (sindrome[29]&sindrome[28]&sindrome[27]&sindrome[25]);


dec[ 0] = enc[32]^o[32];

dec[ 1] = enc[33]^o[33];

dec[ 2] = enc[34]^o[34];

dec[ 3] = enc[35]^o[35];

dec[ 4] = enc[36]^o[36];

dec[ 5] = enc[37]^o[37];

dec[ 6] = enc[38]^o[38];

dec[ 7] = enc[39]^o[39];

dec[ 8] = enc[40]^o[40];

dec[ 9] = enc[41]^o[41];

dec[10] = enc[42]^o[42];

dec[11] = enc[43]^o[43];

dec[12] = enc[44]^o[44];

dec[13] = enc[45]^o[45];

dec[14] = enc[46]^o[46];

dec[15] = enc[47]^o[47];

dec[16] = enc[48]^o[48];

dec[17] = enc[49]^o[49];

dec[18] = enc[50]^o[50];

dec[19] = enc[51]^o[51];

dec[20] = enc[52]^o[52];

dec[21] = enc[53]^o[53];

dec[22] = enc[54]^o[54];

dec[23] = enc[55]^o[55];

dec[24] = enc[56]^o[56];

dec[25] = enc[57]^o[57];

dec[26] = enc[58]^o[58];
```

Figura D.6: Decodificar (12)

```
dec[27] = enc[59]^o[59];

dec[28] = enc[60]^o[60];

dec[29] = enc[61]^o[61];

dec[30] = enc[62]^o[62];

dec[31] = enc[63]^o[63];


ErrD = (sindrome[7]&!sindrome[5]&sindrome[3]&!sindrome[1]) | (sindrome[6]&!sindrome[4]&!sindrome[2]&sindrome[0]) |
(sindrome[6]&!sindrome[4]&sindrome[2]&!sindrome[0]) | (!sindrome[7]&!sindrome[5]&sindrome[3]&!sindrome[2]&sindrome[1])
| (!sindrome[7]&sindrome[5]&!sindrome[4]&sindrome[3]&!sindrome[1]) |
(sindrome[7]&!sindrome[6]&sindrome[5]&!sindrome[3]&!sindrome[1]) |
(!sindrome[7]&sindrome[6]&sindrome[4]&sindrome[2]&sindrome[0]) |
(!sindrome[6]&!sindrome[4]&sindrome[2]&!sindrome[1]&sindrome[0]) |
(!sindrome[6]&sindrome[4]&!sindrome[3]&sindrome[2]&!sindrome[0]) |
(sindrome[6]&!sindrome[5]&sindrome[4]&!sindrome[2]&!sindrome[0]) | (sindrome[6]&!sindrome[5]&!sindrome[4]&sindrome[3])
| (!sindrome[7]&sindrome[6]&!sindrome[5]&sindrome[1]) | (!sindrome[6]&!sindrome[5]&sindrome[4]&sindrome[1]) |
(sindrome[6]&!sindrome[5]&sindrome[3]&!sindrome[1]) | (sindrome[5]&!sindrome[3]&sindrome[2]&!sindrome[1]) |
(!sindrome[6]&sindrome[5]&!sindrome[3]&sindrome[0]) | (!sindrome[4]&sindrome[3]&!sindrome[2]&sindrome[0]) |
(sindrome[5]&!sindrome[3]&!sindrome[1]&sindrome[0]) | (sindrome[6]&!sindrome[4]&sindrome[3]&!sindrome[0]) |
(sindrome[7]&!sindrome[4]&sindrome[2]&!sindrome[0]) | (sindrome[7]&!sindrome[3]&sindrome[2]&!sindrome[0]) |
(sindrome[7]&sindrome[4]&!sindrome[2]&!sindrome[0]) | (!sindrome[6]&sindrome[4]&sindrome[1]&!sindrome[0]) |
(sindrome[6]&sindrome[5]&!sindrome[4]&!sindrome[3]&sindrome[1]) |
(!sindrome[7]&sindrome[5]&!sindrome[3]&!sindrome[2]&sindrome[1]) |
(!sindrome[6]&sindrome[4]&!sindrome[3]&!sindrome[2]&sindrome[0]) |
(sindrome[5]&sindrome[3]&!sindrome[2]&sindrome[1]&sindrome[0]) |
(sindrome[7]&!sindrome[4]&!sindrome[3]&!sindrome[1]&sindrome[0]) |
(sindrome[4]&sindrome[3]&sindrome[2]&!sindrome[1]&sindrome[0]) |
(sindrome[7]&sindrome[6]&!sindrome[2]&!sindrome[1]&sindrome[0]) |
(sindrome[7]&sindrome[5]&sindrome[3]&sindrome[1]&!sindrome[0]) |
(sindrome[7]&!sindrome[5]&!sindrome[3]&sindrome[1]&!sindrome[0]) |
(sindrome[5]&sindrome[3]&sindrome[2]&sindrome[1]&!sindrome[0])

 | (sindrome[15]&!sindrome[13]&sindrome[11]&!sindrome[9]) | (sindrome[14]&!sindrome[12]&!sindrome[10]&sindrome[8]) |
(sindrome[14]&!sindrome[12]&sindrome[10]&!sindrome[8]) |
(!sindrome[15]&!sindrome[13]&sindrome[11]&!sindrome[10]&sindrome[9]) |
(!sindrome[15]&sindrome[13]&!sindrome[12]&sindrome[11]&!sindrome[9]) |
(sindrome[15]&!sindrome[14]&sindrome[13]&!sindrome[11]&!sindrome[9]) |
(!sindrome[15]&sindrome[14]&sindrome[12]&sindrome[10]&sindrome[8]) |
(!sindrome[14]&!sindrome[12]&sindrome[10]&!sindrome[9]&sindrome[8]) |
(!sindrome[14]&sindrome[12]&!sindrome[11]&sindrome[10]&!sindrome[8]) |
(sindrome[14]&!sindrome[13]&sindrome[12]&!sindrome[10]&!sindrome[8]) |
(sindrome[14]&!sindrome[13]&!sindrome[12]&sindrome[11]) | (!sindrome[15]&sindrome[14]&!sindrome[13]&sindrome[9]) |
(!sindrome[14]&!sindrome[13]&sindrome[12]&sindrome[9]) | (sindrome[14]&!sindrome[13]&sindrome[11]&!sindrome[9]) |
(sindrome[13]&!sindrome[11]&sindrome[10]&!sindrome[9]) | (!sindrome[14]&sindrome[13]&!sindrome[11]&sindrome[8]) |
(!sindrome[12]&sindrome[11]&!sindrome[10]&sindrome[8]) | (sindrome[13]&!sindrome[11]&!sindrome[9]&sindrome[8]) |
(sindrome[14]&!sindrome[12]&sindrome[11]&!sindrome[8]) | (sindrome[15]&!sindrome[12]&sindrome[10]&!sindrome[8]) |
(sindrome[15]&!sindrome[11]&sindrome[10]&!sindrome[8]) | (sindrome[15]&sindrome[12]&!sindrome[10]&!sindrome[8]) |
(!sindrome[14]&sindrome[12]&sindrome[9]&!sindrome[8]) |
(sindrome[14]&sindrome[13]&!sindrome[12]&!sindrome[11]&sindrome[9]) |
(!sindrome[15]&sindrome[13]&!sindrome[11]&!sindrome[10]&sindrome[9]) |
(!sindrome[14]&sindrome[12]&!sindrome[11]&!sindrome[10]&sindrome[8]) |
(sindrome[13]&sindrome[11]&!sindrome[10]&sindrome[9]&sindrome[8]) |
(sindrome[15]&!sindrome[12]&!sindrome[11]&!sindrome[9]&sindrome[8]) |
(sindrome[12]&sindrome[11]&sindrome[10]&!sindrome[9]&sindrome[8]) |
(sindrome[15]&sindrome[14]&!sindrome[10]&!sindrome[9]&sindrome[8]) |
(sindrome[15]&sindrome[13]&sindrome[11]&sindrome[9]&!sindrome[8]) |
(sindrome[15]&!sindrome[13]&!sindrome[11]&sindrome[9]&!sindrome[8]) |
(sindrome[13]&sindrome[11]&sindrome[10]&sindrome[9]&!sindrome[8])

 | (sindrome[23]&!sindrome[21]&sindrome[19]&!sindrome[17]) | (sindrome[22]&!sindrome[20]&!sindrome[18]&sindrome[16]) |
(sindrome[22]&!sindrome[20]&sindrome[18]&!sindrome[16]) |
```

Figura D.7: Decodificar (12)

```
(!sindrome[23]&!sindrome[21]&sindrome[19]&!sindrome[18]&sindrome[17]) |
(!sindrome[23]&sindrome[21]&!sindrome[20]&sindrome[19]&!sindrome[17]) |
(sindrome[23]&!sindrome[22]&sindrome[21]&!sindrome[19]&!sindrome[17]) |
(!sindrome[23]&sindrome[22]&sindrome[20]&sindrome[18]&sindrome[16]) |
(!sindrome[22]&!sindrome[20]&sindrome[18]&!sindrome[17]&sindrome[16]) |
(!sindrome[22]&sindrome[20]&!sindrome[19]&sindrome[18]&!sindrome[16]) |
(sindrome[22]&!sindrome[21]&sindrome[20]&!sindrome[18]&!sindrome[16]) |
(sindrome[22]&!sindrome[21]&!sindrome[20]&sindrome[19]) | (!sindrome[23]&sindrome[22]&!sindrome[21]&sindrome[17]) |
(!sindrome[22]&!sindrome[21]&sindrome[20]&sindrome[17]) | (sindrome[22]&!sindrome[21]&sindrome[19]&!sindrome[17]) |
(sindrome[21]&!sindrome[19]&sindrome[18]&!sindrome[17]) | (!sindrome[22]&sindrome[21]&!sindrome[19]&sindrome[16]) |
(!sindrome[20]&sindrome[19]&!sindrome[18]&sindrome[16]) | (sindrome[21]&!sindrome[19]&!sindrome[17]&sindrome[16]) |
(sindrome[22]&!sindrome[20]&sindrome[19]&!sindrome[16]) | (sindrome[23]&!sindrome[20]&sindrome[18]&!sindrome[16]) |
(sindrome[23]&!sindrome[19]&sindrome[18]&!sindrome[16]) | (sindrome[23]&sindrome[20]&!sindrome[18]&!sindrome[16]) |
(!sindrome[22]&sindrome[20]&sindrome[17]&!sindrome[16]) |
(sindrome[22]&sindrome[21]&!sindrome[20]&!sindrome[19]&sindrome[17]) |
(!sindrome[23]&sindrome[21]&!sindrome[19]&!sindrome[18]&sindrome[17]) |
(!sindrome[22]&sindrome[20]&!sindrome[19]&!sindrome[18]&sindrome[16]) |
(sindrome[21]&sindrome[19]&!sindrome[18]&sindrome[17]&sindrome[16]) |
(sindrome[23]&!sindrome[20]&!sindrome[19]&!sindrome[17]&sindrome[16]) |
(sindrome[20]&sindrome[19]&sindrome[18]&!sindrome[17]&sindrome[16]) |
(sindrome[23]&sindrome[22]&!sindrome[18]&!sindrome[17]&sindrome[16]) |
(sindrome[23]&sindrome[21]&sindrome[19]&sindrome[17]&!sindrome[16]) |
(sindrome[23]&!sindrome[21]&!sindrome[19]&sindrome[17]&!sindrome[16]) |
(sindrome[21]&sindrome[19]&sindrome[18]&sindrome[17]&!sindrome[16])

 | (sindrome[31]&!sindrome[29]&sindrome[27]&!sindrome[25]) | (sindrome[30]&!sindrome[28]&!sindrome[26]&sindrome[24]) |
(sindrome[30]&!sindrome[28]&sindrome[26]&!sindrome[24]) |
(!sindrome[31]&!sindrome[29]&sindrome[27]&!sindrome[26]&sindrome[25]) |
(!sindrome[31]&sindrome[29]&!sindrome[28]&sindrome[27]&!sindrome[25]) |
(sindrome[31]&!sindrome[30]&sindrome[29]&!sindrome[27]&!sindrome[25]) |
(!sindrome[31]&sindrome[30]&sindrome[28]&sindrome[26]&sindrome[24]) |
(!sindrome[30]&!sindrome[28]&sindrome[26]&!sindrome[25]&sindrome[24]) |
(!sindrome[30]&sindrome[28]&!sindrome[27]&sindrome[26]&!sindrome[24]) |
(sindrome[30]&!sindrome[29]&sindrome[28]&!sindrome[26]&!sindrome[24]) |
(sindrome[30]&!sindrome[29]&!sindrome[28]&sindrome[27]) | (!sindrome[31]&sindrome[30]&!sindrome[29]&sindrome[25]) |
(!sindrome[30]&!sindrome[29]&sindrome[28]&sindrome[25]) | (sindrome[30]&!sindrome[29]&sindrome[27]&!sindrome[25]) |
(sindrome[29]&!sindrome[27]&sindrome[26]&!sindrome[25]) | (!sindrome[30]&sindrome[29]&!sindrome[27]&sindrome[24]) |
(!sindrome[28]&sindrome[27]&!sindrome[26]&sindrome[24]) | (sindrome[29]&!sindrome[27]&!sindrome[25]&sindrome[24]) |
(sindrome[30]&!sindrome[28]&sindrome[27]&!sindrome[24]) | (sindrome[31]&!sindrome[28]&sindrome[26]&!sindrome[24]) |
(sindrome[31]&!sindrome[27]&sindrome[26]&!sindrome[24]) | (sindrome[31]&sindrome[28]&!sindrome[26]&!sindrome[24]) |
(!sindrome[30]&sindrome[28]&sindrome[25]&!sindrome[24]) |
(sindrome[30]&sindrome[29]&!sindrome[28]&!sindrome[27]&sindrome[25]) |
(!sindrome[31]&sindrome[29]&!sindrome[27]&!sindrome[26]&sindrome[25]) |
(!sindrome[30]&sindrome[28]&!sindrome[27]&!sindrome[26]&sindrome[24]) |
(sindrome[29]&sindrome[27]&!sindrome[26]&sindrome[25]&sindrome[24]) |
(sindrome[31]&!sindrome[28]&!sindrome[27]&!sindrome[25]&sindrome[24]) |
(sindrome[28]&sindrome[27]&sindrome[26]&!sindrome[25]&sindrome[24]) |
(sindrome[31]&sindrome[30]&!sindrome[26]&!sindrome[25]&sindrome[24]) |
(sindrome[31]&sindrome[29]&sindrome[27]&sindrome[25]&!sindrome[24]) |
(sindrome[31]&!sindrome[29]&!sindrome[27]&sindrome[25]&!sindrome[24]) |
(sindrome[29]&sindrome[27]&sindrome[26]&sindrome[25]&!sindrome[24]);

}
```

Figura D.8: Decodificar (12)

- Tipo 16:

```
void decodificar (bitvector *enc, bitvector *dec)
{
  bitvector sindrome[N-M];
  bitvector o[N];

sindrome[ 0] = enc[0]^enc[32]^enc[40]^enc[56];
sindrome[ 1] = enc[4]^enc[36]^enc[52]^enc[60];
sindrome[ 2] = enc[8]^enc[32]^enc[40]^enc[48];
sindrome[ 3] = enc[12]^enc[36]^enc[44]^enc[52];
sindrome[ 4] = enc[16]^enc[32]^enc[48]^enc[56];
sindrome[ 5] = enc[20]^enc[36]^enc[44]^enc[60];
sindrome[ 6] = enc[24]^enc[40]^enc[48]^enc[56];
sindrome[ 7] = enc[28]^enc[44]^enc[52]^enc[60];
sindrome[ 8] = enc[1]^enc[33]^enc[41]^enc[57];
sindrome[ 9] = enc[5]^enc[37]^enc[53]^enc[61];
sindrome[10] = enc[9]^enc[33]^enc[41]^enc[49];
sindrome[11] = enc[13]^enc[37]^enc[45]^enc[53];
sindrome[12] = enc[17]^enc[33]^enc[49]^enc[57];
sindrome[13] = enc[21]^enc[37]^enc[45]^enc[61];
sindrome[14] = enc[25]^enc[41]^enc[49]^enc[57];
sindrome[15] = enc[29]^enc[45]^enc[53]^enc[61];
sindrome[16] = enc[2]^enc[34]^enc[42]^enc[58];
sindrome[17] = enc[6]^enc[38]^enc[54]^enc[62];
sindrome[18] = enc[10]^enc[34]^enc[42]^enc[50];
sindrome[19] = enc[14]^enc[38]^enc[46]^enc[54];
sindrome[20] = enc[18]^enc[34]^enc[50]^enc[58];
sindrome[21] = enc[22]^enc[38]^enc[46]^enc[62];
sindrome[22] = enc[26]^enc[42]^enc[50]^enc[58];
sindrome[23] = enc[30]^enc[46]^enc[54]^enc[62];
sindrome[24] = enc[3]^enc[35]^enc[43]^enc[59];
sindrome[25] = enc[7]^enc[39]^enc[55]^enc[63];
sindrome[26] = enc[11]^enc[35]^enc[43]^enc[51];
sindrome[27] = enc[15]^enc[39]^enc[47]^enc[55];
sindrome[28] = enc[19]^enc[35]^enc[51]^enc[59];
sindrome[29] = enc[23]^enc[39]^enc[47]^enc[63];
sindrome[30] = enc[27]^enc[43]^enc[51]^enc[59];
sindrome[31] = enc[31]^enc[47]^enc[55]^enc[63];
```

Figura D.9: Decodificar (16)

```
o[60] = (sindrome[7]&sindrome[5]&!sindrome[3]&sindrome[1]) | (!sindrome[7]&sindrome[3]&!sindrome[1]&sindrome[0]);

o[56] = (sindrome[6]&!sindrome[2]&sindrome[0]) | (!sindrome[6]&sindrome[5]&!sindrome[4]&sindrome[0]) |
(sindrome[7]&!sindrome[5]&sindrome[3]&sindrome[0]);

o[52] = (sindrome[7]&!sindrome[5]&sindrome[3]) | (!sindrome[7]&sindrome[5]&!sindrome[3]&sindrome[1]) |
(!sindrome[7]&sindrome[3]&!sindrome[1]&sindrome[0]);

o[48] = (sindrome[6]&sindrome[2]&!sindrome[0]) | (!sindrome[7]&sindrome[5]&!sindrome[3]&sindrome[1]) |
(!sindrome[6]&sindrome[3]&!sindrome[1]&sindrome[0]) | (sindrome[7]&!sindrome[6]&!sindrome[5]&sindrome[1]&sindrome[0]);

o[44] = (sindrome[7]&sindrome[3]&!sindrome[1]) | (!sindrome[7]&sindrome[5]&!sindrome[3]&sindrome[1]) |
(sindrome[7]&!sindrome[5]&!sindrome[3]&sindrome[1]);

o[40] = (sindrome[6]&!sindrome[4]&sindrome[0]) | (!sindrome[6]&sindrome[4]&!sindrome[2]&sindrome[0]) |
(sindrome[4]&!sindrome[2]&sindrome[1]&!sindrome[0]);

o[36] = (sindrome[5]&sindrome[3]&sindrome[1]) | (sindrome[7]&!sindrome[5]&!sindrome[3]&sindrome[1]);

o[32] = (sindrome[4]&sindrome[2]&sindrome[0]) | (sindrome[4]&!sindrome[2]&sindrome[1]&!sindrome[0]);


o[61] = (sindrome[15]&sindrome[13]&!sindrome[11]&sindrome[9]) | (!sindrome[15]&sindrome[11]&!sindrome[9]&sindrome[8]);

o[57] = (sindrome[14]&!sindrome[10]&sindrome[8]) | (!sindrome[14]&sindrome[13]&!sindrome[12]&sindrome[8]) |
(sindrome[15]&!sindrome[13]&sindrome[11]&sindrome[8]);

o[53] = (sindrome[15]&!sindrome[13]&sindrome[11]) | (!sindrome[15]&sindrome[13]&!sindrome[11]&sindrome[9]) |
(!sindrome[15]&sindrome[11]&!sindrome[9]&sindrome[8]);

o[49] = (sindrome[14]&sindrome[10]&!sindrome[8]) | (!sindrome[15]&sindrome[13]&!sindrome[11]&sindrome[9]) |
(!sindrome[14]&sindrome[11]&!sindrome[9]&sindrome[8]) |
(sindrome[15]&!sindrome[14]&!sindrome[13]&sindrome[9]&sindrome[8]);

o[45] = (sindrome[15]&sindrome[11]&!sindrome[9]) | (!sindrome[15]&sindrome[13]&!sindrome[11]&sindrome[9]) |
(sindrome[15]&!sindrome[13]&!sindrome[11]&sindrome[9]);

o[41] = (sindrome[14]&!sindrome[12]&sindrome[8]) | (!sindrome[14]&sindrome[12]&!sindrome[10]&sindrome[8]) |
(sindrome[12]&!sindrome[10]&sindrome[9]&!sindrome[8]);

o[37] = (sindrome[13]&sindrome[11]&sindrome[9]) | (sindrome[15]&!sindrome[13]&!sindrome[11]&sindrome[9]);

o[33] = (sindrome[12]&sindrome[10]&sindrome[8]) | (sindrome[12]&!sindrome[10]&sindrome[9]&!sindrome[8]);


o[62] = (sindrome[23]&sindrome[21]&!sindrome[19]&sindrome[17]) |
(!sindrome[23]&sindrome[19]&!sindrome[17]&sindrome[16]);

o[58] = (sindrome[22]&!sindrome[18]&sindrome[16]) | (!sindrome[22]&sindrome[21]&!sindrome[20]&sindrome[16]) |
(sindrome[23]&!sindrome[21]&sindrome[19]&sindrome[16]);

o[54] = (sindrome[23]&!sindrome[21]&sindrome[19]) | (!sindrome[23]&sindrome[21]&!sindrome[19]&sindrome[17]) |
(!sindrome[23]&sindrome[19]&!sindrome[17]&sindrome[16]);

o[50] = (sindrome[22]&sindrome[18]&!sindrome[16]) | (!sindrome[23]&sindrome[21]&!sindrome[19]&sindrome[17]) |
(!sindrome[22]&sindrome[19]&!sindrome[17]&sindrome[16]) |
(sindrome[23]&!sindrome[22]&!sindrome[21]&sindrome[17]&sindrome[16]);

o[46] = (sindrome[23]&sindrome[19]&!sindrome[17]) | (!sindrome[23]&sindrome[21]&!sindrome[19]&sindrome[17]) |
(sindrome[23]&!sindrome[21]&!sindrome[19]&sindrome[17]);

o[42] = (sindrome[22]&!sindrome[20]&sindrome[16]) | (!sindrome[22]&sindrome[20]&!sindrome[18]&sindrome[16]) |
(sindrome[20]&!sindrome[18]&sindrome[17]&!sindrome[16]);

o[38] = (sindrome[21]&sindrome[19]&sindrome[17]) | (sindrome[23]&!sindrome[21]&!sindrome[19]&sindrome[17]);

o[34] = (sindrome[20]&sindrome[18]&sindrome[16]) | (sindrome[20]&!sindrome[18]&sindrome[17]&!sindrome[16]);
```

Figura D.10: Decodificar (16)

```
o[63] = (sindrome[31]&sindrome[29]&!sindrome[27]&sindrome[25]) |
(!sindrome[31]&sindrome[27]&!sindrome[25]&sindrome[24]);

o[59] = (sindrome[30]&!sindrome[26]&sindrome[24]) | (!sindrome[30]&sindrome[29]&!sindrome[28]&sindrome[24]) |
(sindrome[31]&!sindrome[29]&sindrome[27]&sindrome[24]);

o[55] = (sindrome[31]&!sindrome[29]&sindrome[27]) | (!sindrome[31]&sindrome[29]&!sindrome[27]&sindrome[25]) |
(!sindrome[31]&sindrome[27]&!sindrome[25]&sindrome[24]);

o[51] = (sindrome[30]&sindrome[26]&!sindrome[24]) | (!sindrome[31]&sindrome[29]&!sindrome[27]&sindrome[25]) |
(!sindrome[30]&sindrome[27]&!sindrome[25]&sindrome[24]) |
(sindrome[31]&!sindrome[30]&!sindrome[29]&sindrome[25]&sindrome[24]);

o[47] = (sindrome[31]&sindrome[27]&!sindrome[25]) | (!sindrome[31]&sindrome[29]&!sindrome[27]&sindrome[25]) |
(sindrome[31]&!sindrome[29]&!sindrome[27]&sindrome[25]);

o[43] = (sindrome[30]&!sindrome[28]&sindrome[24]) | (!sindrome[30]&sindrome[28]&!sindrome[26]&sindrome[24]) |
(sindrome[28]&!sindrome[26]&sindrome[25]&!sindrome[24]);

o[39] = (sindrome[29]&sindrome[27]&sindrome[25]) | (sindrome[31]&!sindrome[29]&!sindrome[27]&sindrome[25]);

o[35] = (sindrome[28]&sindrome[26]&sindrome[24]) | (sindrome[28]&!sindrome[26]&sindrome[25]&!sindrome[24]);


dec[ 0] = enc[32]^o[32];

dec[ 1] = enc[33]^o[33];

dec[ 2] = enc[34]^o[34];

dec[ 3] = enc[35]^o[35];

dec[ 4] = enc[36]^o[36];

dec[ 5] = enc[37]^o[37];

dec[ 6] = enc[38]^o[38];

dec[ 7] = enc[39]^o[39];

dec[ 8] = enc[40]^o[40];

dec[ 9] = enc[41]^o[41];

dec[10] = enc[42]^o[42];

dec[11] = enc[43]^o[43];

dec[12] = enc[44]^o[44];

dec[13] = enc[45]^o[45];

dec[14] = enc[46]^o[46];

dec[15] = enc[47]^o[47];

dec[16] = enc[48]^o[48];

dec[17] = enc[49]^o[49];

dec[18] = enc[50]^o[50];

dec[19] = enc[51]^o[51];

dec[20] = enc[52]^o[52];

dec[21] = enc[53]^o[53];

dec[22] = enc[54]^o[54];

dec[23] = enc[55]^o[55];

dec[24] = enc[56]^o[56];
```

Figura D.11: Decodificar (16)

```
dec[25] = enc[57]^o[57];

dec[26] = enc[58]^o[58];

dec[27] = enc[59]^o[59];

dec[28] = enc[60]^o[60];

dec[29] = enc[61]^o[61];

dec[30] = enc[62]^o[62];

dec[31] = enc[63]^o[63];


ErrD = (sindrome[7]&!sindrome[5]&sindrome[3]&!sindrome[1]) | (sindrome[6]&!sindrome[4]&!sindrome[2]&sindrome[0]) |
(sindrome[6]&!sindrome[4]&sindrome[2]&!sindrome[0]) | (!sindrome[7]&!sindrome[5]&sindrome[3]&!sindrome[2]&sindrome[1])
| (sindrome[7]&!sindrome[6]&sindrome[5]&!sindrome[3]&!sindrome[1]) |
(!sindrome[7]&sindrome[6]&sindrome[4]&sindrome[2]&sindrome[0]) |
(!sindrome[6]&sindrome[4]&!sindrome[3]&sindrome[2]&!sindrome[0]) | (sindrome[6]&!sindrome[5]&!sindrome[4]&sindrome[3])
| (!sindrome[7]&sindrome[6]&!sindrome[5]&sindrome[1]) | (sindrome[6]&!sindrome[5]&sindrome[3]&!sindrome[1]) |
(!sindrome[4]&sindrome[3]&!sindrome[2]&sindrome[0]) | (sindrome[7]&!sindrome[6]&sindrome[4]&!sindrome[0]) |
(sindrome[6]&!sindrome[4]&sindrome[3]&!sindrome[0]) | (sindrome[7]&!sindrome[4]&sindrome[2]&!sindrome[0]) |
(sindrome[7]&!sindrome[3]&sindrome[2]&!sindrome[0]) | (sindrome[6]&sindrome[5]&!sindrome[4]&!sindrome[3]&sindrome[1]) |
(!sindrome[7]&sindrome[5]&!sindrome[3]&sindrome[2]&!sindrome[1]) |
(sindrome[7]&!sindrome[6]&sindrome[5]&!sindrome[3]&sindrome[0]) |
(!sindrome[5]&sindrome[4]&sindrome[2]&sindrome[1]&sindrome[0]) |
(sindrome[5]&sindrome[3]&!sindrome[2]&sindrome[1]&sindrome[0]) |
(!sindrome[7]&sindrome[5]&!sindrome[3]&!sindrome[1]&sindrome[0]) |
(sindrome[7]&!sindrome[4]&!sindrome[3]&!sindrome[1]&sindrome[0]) |
(sindrome[4]&sindrome[3]&sindrome[2]&!sindrome[1]&sindrome[0]) |
(sindrome[7]&sindrome[6]&!sindrome[2]&!sindrome[1]&sindrome[0]) |
(sindrome[5]&sindrome[3]&sindrome[2]&sindrome[1]&!sindrome[0]) |
(!sindrome[6]&sindrome[4]&!sindrome[2]&sindrome[1]&!sindrome[0]) |
(!sindrome[6]&!sindrome[4]&!sindrome[3]&sindrome[2]&!sindrome[1]&sindrome[0]) |
(!sindrome[6]&sindrome[4]&!sindrome[3]&!sindrome[2]&!sindrome[1]&sindrome[0]) |
(sindrome[7]&!sindrome[6]&sindrome[5]&sindrome[3]&sindrome[1]&!sindrome[0]) |
(sindrome[7]&!sindrome[5]&!sindrome[4]&!sindrome[3]&sindrome[1]&!sindrome[0]) |
(!sindrome[7]&sindrome[5]&!sindrome[3]&!sindrome[2]&sindrome[1]&!sindrome[0]) |
(!sindrome[7]&sindrome[5]&!sindrome[4]&sindrome[3]&!sindrome[1]&!sindrome[0]) |
(sindrome[6]&!sindrome[5]&sindrome[4]&!sindrome[2]&!sindrome[1]&!sindrome[0])

 | (sindrome[15]&!sindrome[13]&sindrome[11]&!sindrome[9]) | (sindrome[14]&!sindrome[12]&!sindrome[10]&sindrome[8]) |
(sindrome[14]&!sindrome[12]&sindrome[10]&!sindrome[8]) |
(!sindrome[15]&!sindrome[13]&sindrome[11]&!sindrome[10]&sindrome[9]) |
(sindrome[15]&!sindrome[14]&sindrome[13]&!sindrome[11]&!sindrome[9]) |
(!sindrome[15]&sindrome[14]&sindrome[12]&sindrome[10]&sindrome[8]) |
(!sindrome[14]&sindrome[12]&!sindrome[11]&sindrome[10]&!sindrome[8]) |
(sindrome[14]&!sindrome[13]&!sindrome[12]&sindrome[11]) | (!sindrome[15]&sindrome[14]&!sindrome[13]&sindrome[9]) |
(sindrome[14]&!sindrome[13]&sindrome[11]&!sindrome[9]) | (!sindrome[12]&sindrome[11]&!sindrome[10]&sindrome[8]) |
(sindrome[15]&!sindrome[14]&sindrome[12]&!sindrome[8]) | (sindrome[14]&!sindrome[12]&sindrome[11]&!sindrome[8]) |
(sindrome[15]&!sindrome[12]&sindrome[10]&!sindrome[8]) | (sindrome[15]&!sindrome[11]&sindrome[10]&!sindrome[8]) |
(sindrome[14]&sindrome[13]&!sindrome[12]&!sindrome[11]&sindrome[9]) |
(!sindrome[15]&sindrome[13]&!sindrome[11]&sindrome[10]&!sindrome[9]) |
(sindrome[15]&!sindrome[14]&sindrome[13]&!sindrome[11]&sindrome[8]) |
(!sindrome[13]&sindrome[12]&sindrome[10]&sindrome[9]&sindrome[8]) |
(sindrome[13]&sindrome[11]&!sindrome[10]&sindrome[9]&sindrome[8]) |
(!sindrome[15]&sindrome[13]&!sindrome[11]&!sindrome[9]&sindrome[8]) |
(sindrome[15]&!sindrome[12]&!sindrome[11]&!sindrome[9]&sindrome[8]) |
(sindrome[12]&sindrome[11]&sindrome[10]&!sindrome[9]&sindrome[8]) |
(sindrome[15]&sindrome[14]&!sindrome[10]&!sindrome[9]&sindrome[8]) |
(sindrome[13]&sindrome[11]&sindrome[10]&sindrome[9]&!sindrome[8]) |
(!sindrome[14]&sindrome[12]&!sindrome[10]&sindrome[9]&!sindrome[8]) |
(!sindrome[14]&!sindrome[12]&!sindrome[11]&sindrome[10]&!sindrome[9]&sindrome[8]) |
(!sindrome[14]&sindrome[12]&!sindrome[11]&!sindrome[10]&!sindrome[9]&sindrome[8]) |
```

Figura D.12: Decodificar (16)

```
(sindrome[15]&!sindrome[14]&sindrome[13]&sindrome[11]&sindrome[9]&!sindrome[8]) |
(sindrome[15]&!sindrome[13]&!sindrome[12]&!sindrome[11]&sindrome[9]&!sindrome[8]) |
(!sindrome[15]&sindrome[13]&!sindrome[11]&!sindrome[10]&sindrome[9]&!sindrome[8]) |
(!sindrome[15]&sindrome[13]&!sindrome[12]&sindrome[11]&!sindrome[9]&!sindrome[8]) |
(sindrome[14]&!sindrome[13]&sindrome[12]&!sindrome[10]&!sindrome[9]&!sindrome[8]) |
(sindrome[23]&!sindrome[21]&sindrome[19]&!sindrome[17]) | (sindrome[22]&!sindrome[20]&!sindrome[18]&sindrome[16]) |
(sindrome[22]&!sindrome[20]&sindrome[18]&!sindrome[16]) |
(!sindrome[23]&!sindrome[21]&sindrome[19]&!sindrome[18]&sindrome[17]) |
(sindrome[23]&!sindrome[22]&sindrome[21]&!sindrome[19]&!sindrome[17]) |
(!sindrome[23]&sindrome[22]&sindrome[20]&sindrome[18]&sindrome[16]) |
(!sindrome[22]&sindrome[20]&!sindrome[19]&sindrome[18]&!sindrome[16]) |
(sindrome[22]&!sindrome[21]&!sindrome[20]&sindrome[19]) | (!sindrome[23]&sindrome[22]&!sindrome[21]&sindrome[17]) |
(sindrome[22]&!sindrome[21]&sindrome[19]&!sindrome[17]) | (!sindrome[20]&sindrome[19]&!sindrome[18]&sindrome[16]) |
(sindrome[23]&!sindrome[22]&sindrome[20]&!sindrome[16]) | (sindrome[22]&!sindrome[20]&sindrome[19]&!sindrome[16]) |
(sindrome[23]&!sindrome[20]&sindrome[18]&!sindrome[16]) | (sindrome[23]&!sindrome[19]&sindrome[18]&!sindrome[16]) |
(sindrome[22]&sindrome[21]&!sindrome[20]&!sindrome[19]&sindrome[17]) |
(!sindrome[23]&sindrome[21]&!sindrome[19]&sindrome[18]&!sindrome[17]) |
(sindrome[23]&!sindrome[22]&sindrome[21]&!sindrome[19]&sindrome[16]) |
(!sindrome[21]&sindrome[20]&sindrome[18]&sindrome[17]&sindrome[16]) |
(sindrome[21]&sindrome[19]&!sindrome[18]&sindrome[17]&sindrome[16]) |
(!sindrome[23]&sindrome[21]&!sindrome[19]&!sindrome[17]&sindrome[16]) |
(sindrome[23]&!sindrome[20]&!sindrome[19]&!sindrome[17]&sindrome[16]) |
(sindrome[20]&sindrome[19]&sindrome[18]&!sindrome[17]&sindrome[16]) |
(sindrome[23]&sindrome[22]&!sindrome[18]&!sindrome[17]&sindrome[16]) |
(sindrome[21]&sindrome[19]&sindrome[18]&sindrome[17]&!sindrome[16]) |
(!sindrome[22]&sindrome[20]&!sindrome[18]&sindrome[17]&!sindrome[16]) |
(!sindrome[22]&!sindrome[20]&!sindrome[19]&sindrome[18]&!sindrome[17]&sindrome[16]) |
(!sindrome[22]&sindrome[20]&!sindrome[19]&!sindrome[18]&!sindrome[17]&sindrome[16]) |
(sindrome[23]&!sindrome[22]&sindrome[21]&sindrome[19]&sindrome[17]&!sindrome[16]) |
(sindrome[23]&!sindrome[21]&!sindrome[20]&!sindrome[19]&sindrome[17]&!sindrome[16]) |
(!sindrome[23]&sindrome[21]&!sindrome[19]&!sindrome[18]&sindrome[17]&!sindrome[16]) |
(!sindrome[23]&sindrome[21]&!sindrome[20]&sindrome[19]&!sindrome[17]&!sindrome[16]) |
(sindrome[22]&!sindrome[21]&sindrome[20]&!sindrome[18]&!sindrome[17]&!sindrome[16])

 | (sindrome[31]&!sindrome[29]&sindrome[27]&!sindrome[25]) | (sindrome[30]&!sindrome[28]&!sindrome[26]&sindrome[24]) |
(sindrome[30]&!sindrome[28]&sindrome[26]&!sindrome[24]) |
(!sindrome[31]&!sindrome[29]&sindrome[27]&!sindrome[26]&sindrome[25]) |
(sindrome[31]&!sindrome[30]&sindrome[29]&!sindrome[27]&!sindrome[25]) |
(!sindrome[31]&sindrome[30]&sindrome[28]&sindrome[26]&sindrome[24]) |
(!sindrome[30]&sindrome[28]&!sindrome[27]&sindrome[26]&!sindrome[24]) |
(sindrome[30]&!sindrome[29]&!sindrome[28]&sindrome[27]) | (!sindrome[31]&sindrome[30]&!sindrome[29]&sindrome[25]) |
(sindrome[30]&!sindrome[29]&sindrome[27]&!sindrome[25]) | (!sindrome[28]&sindrome[27]&!sindrome[26]&sindrome[24]) |
(sindrome[31]&!sindrome[30]&sindrome[28]&!sindrome[24]) | (sindrome[30]&!sindrome[28]&sindrome[27]&!sindrome[24]) |
(sindrome[31]&!sindrome[28]&sindrome[26]&!sindrome[24]) | (sindrome[31]&!sindrome[27]&sindrome[26]&!sindrome[24]) |
(sindrome[30]&sindrome[29]&!sindrome[28]&!sindrome[27]&sindrome[25]) |
(!sindrome[31]&sindrome[29]&!sindrome[27]&sindrome[26]&!sindrome[25]) |
(sindrome[31]&!sindrome[30]&sindrome[29]&!sindrome[27]&sindrome[24]) |
(!sindrome[29]&sindrome[28]&sindrome[26]&sindrome[25]&sindrome[24]) |
(sindrome[29]&sindrome[27]&!sindrome[26]&sindrome[25]&sindrome[24]) |
(!sindrome[31]&sindrome[29]&!sindrome[27]&!sindrome[25]&sindrome[24]) |
(sindrome[31]&!sindrome[28]&!sindrome[27]&!sindrome[25]&sindrome[24]) |
(sindrome[28]&sindrome[27]&sindrome[26]&!sindrome[25]&sindrome[24]) |
(sindrome[31]&sindrome[30]&!sindrome[26]&!sindrome[25]&sindrome[24]) |
(sindrome[29]&sindrome[27]&sindrome[26]&sindrome[25]&!sindrome[24]) |
(!sindrome[30]&sindrome[28]&!sindrome[26]&sindrome[25]&!sindrome[24]) |
(!sindrome[30]&!sindrome[28]&!sindrome[27]&sindrome[26]&!sindrome[25]&sindrome[24]) |
(!sindrome[30]&sindrome[28]&!sindrome[27]&!sindrome[26]&!sindrome[25]&sindrome[24]) |
(sindrome[31]&!sindrome[30]&sindrome[29]&sindrome[27]&sindrome[25]&!sindrome[24]) |
(sindrome[31]&!sindrome[29]&!sindrome[28]&!sindrome[27]&sindrome[25]&!sindrome[24]) |
(!sindrome[31]&sindrome[29]&!sindrome[27]&!sindrome[26]&sindrome[25]&!sindrome[24]) |
(!sindrome[31]&sindrome[29]&!sindrome[28]&sindrome[27]&!sindrome[25]&!sindrome[24]) |
(sindrome[30]&!sindrome[29]&sindrome[28]&!sindrome[26]&!sindrome[25]&!sindrome[24]); }
```

Figura D.13: Decodificar (16)

- Tipo 20:

```
void decodificar (bitvector *enc, bitvector *dec)
{
  bitvector sindrome[N-M];
  bitvector o[N];

sindrome[ 0] = enc[0]^enc[32]^enc[40]^enc[56];
sindrome[ 1] = enc[4]^enc[36]^enc[52]^enc[60];
sindrome[ 2] = enc[8]^enc[32]^enc[40]^enc[48];
sindrome[ 3] = enc[12]^enc[36]^enc[44]^enc[52];
sindrome[ 4] = enc[16]^enc[32]^enc[48]^enc[56];
sindrome[ 5] = enc[20]^enc[36]^enc[44]^enc[60];
sindrome[ 6] = enc[24]^enc[40]^enc[48]^enc[56];
sindrome[ 7] = enc[28]^enc[44]^enc[52]^enc[60];
sindrome[ 8] = enc[1]^enc[33]^enc[41]^enc[57];
sindrome[ 9] = enc[5]^enc[37]^enc[53]^enc[61];
sindrome[10] = enc[9]^enc[33]^enc[41]^enc[49];
sindrome[11] = enc[13]^enc[37]^enc[45]^enc[53];
sindrome[12] = enc[17]^enc[33]^enc[49]^enc[57];
sindrome[13] = enc[21]^enc[37]^enc[45]^enc[61];
sindrome[14] = enc[25]^enc[41]^enc[49]^enc[57];
sindrome[15] = enc[29]^enc[45]^enc[53]^enc[61];
sindrome[16] = enc[2]^enc[34]^enc[42]^enc[58];
sindrome[17] = enc[6]^enc[38]^enc[54]^enc[62];
sindrome[18] = enc[10]^enc[34]^enc[42]^enc[50];
sindrome[19] = enc[14]^enc[38]^enc[46]^enc[54];
sindrome[20] = enc[18]^enc[34]^enc[50]^enc[58];
sindrome[21] = enc[22]^enc[38]^enc[46]^enc[62];
sindrome[22] = enc[26]^enc[42]^enc[50]^enc[58];
sindrome[23] = enc[30]^enc[46]^enc[54]^enc[62];
sindrome[24] = enc[3]^enc[35]^enc[43]^enc[59];
sindrome[25] = enc[7]^enc[39]^enc[55]^enc[63];
sindrome[26] = enc[11]^enc[35]^enc[43]^enc[51];
sindrome[27] = enc[15]^enc[39]^enc[47]^enc[55];
sindrome[28] = enc[19]^enc[35]^enc[51]^enc[59];
sindrome[29] = enc[23]^enc[39]^enc[47]^enc[63];
sindrome[30] = enc[27]^enc[43]^enc[51]^enc[59];
sindrome[31] = enc[31]^enc[47]^enc[55]^enc[63];
```

Figura D.14: Decodificar (20)

```
o[60] = (sindrome[7]&sindrome[5]&!sindrome[3]&sindrome[1]) | (!sindrome[7]&sindrome[5]&!sindrome[1]&sindrome[0]) |
(!sindrome[6]&!sindrome[4]&sindrome[2]&!sindrome[1]&sindrome[0]);

o[56] = (sindrome[6]&!sindrome[2]&sindrome[0]) | (sindrome[7]&!sindrome[6]&!sindrome[4]&sindrome[0]) |
(!sindrome[6]&sindrome[5]&!sindrome[4]&sindrome[0]) | (!sindrome[7]&sindrome[5]&!sindrome[4]&!sindrome[3]&sindrome[1]);

o[52] = (!sindrome[7]&sindrome[5]&!sindrome[3]&sindrome[1]) | (sindrome[7]&!sindrome[6]&!sindrome[4]&sindrome[0]) |
(!sindrome[5]&sindrome[3]&!sindrome[2]&sindrome[0]) | (!sindrome[7]&sindrome[3]&!sindrome[1]&sindrome[0]) |
(sindrome[7]&!sindrome[5]&sindrome[3]&!sindrome[0]);

o[48] = (sindrome[7]&sindrome[2]&!sindrome[0]) | (!sindrome[7]&sindrome[5]&!sindrome[3]&sindrome[1]) |
(!sindrome[6]&sindrome[4]&!sindrome[2]&sindrome[0]) | (!sindrome[7]&sindrome[6]&sindrome[4]&!sindrome[3]&sindrome[2]) |
(sindrome[7]&!sindrome[6]&!sindrome[5]&sindrome[1]&sindrome[0]) |
(!sindrome[6]&!sindrome[4]&sindrome[2]&!sindrome[1]&sindrome[0]);

o[44] = (!sindrome[7]&sindrome[5]&!sindrome[3]&sindrome[1]) | (sindrome[7]&!sindrome[4]&sindrome[3]&!sindrome[1]) |
(sindrome[7]&sindrome[3]&sindrome[2]&!sindrome[1]) | (!sindrome[6]&sindrome[4]&!sindrome[2]&sindrome[0]) |
(sindrome[6]&!sindrome[3]&sindrome[1]&!sindrome[0]) | (sindrome[7]&!sindrome[5]&!sindrome[4]&!sindrome[3]&sindrome[2]);

o[40] = (sindrome[6]&!sindrome[4]&sindrome[3]) | (sindrome[7]&!sindrome[5]&!sindrome[3]&sindrome[1]) |
(!sindrome[7]&sindrome[6]&sindrome[2]&sindrome[0]) | (!sindrome[6]&sindrome[4]&!sindrome[2]&sindrome[0]) |
(sindrome[6]&!sindrome[2]&sindrome[1]&!sindrome[0]) | (sindrome[4]&!sindrome[2]&sindrome[1]&!sindrome[0]);

o[36] = (sindrome[7]&!sindrome[5]&!sindrome[3]&sindrome[1]) | (sindrome[5]&sindrome[3]&!sindrome[2]&sindrome[1]) |
(sindrome[5]&sindrome[3]&sindrome[1]&sindrome[0]) | (sindrome[4]&!sindrome[2]&sindrome[1]&!sindrome[0]) |
(!sindrome[6]&!sindrome[5]&sindrome[4]&!sindrome[2]&sindrome[0]) |
(sindrome[7]&sindrome[4]&sindrome[2]&sindrome[1]&sindrome[0]);

o[32] = (sindrome[7]&sindrome[4]&sindrome[2]&sindrome[0]) | (sindrome[5]&sindrome[4]&sindrome[2]&sindrome[0]) |
(sindrome[4]&sindrome[2]&!sindrome[1]&sindrome[0]) | (sindrome[4]&!sindrome[2]&sindrome[1]&!sindrome[0]) |
(sindrome[7]&sindrome[6]&!sindrome[3]&!sindrome[1]&sindrome[0]) |
(sindrome[7]&!sindrome[5]&!sindrome[3]&sindrome[1]&!sindrome[0]);


o[61] = (sindrome[15]&sindrome[13]&!sindrome[11]&sindrome[9]) | (!sindrome[15]&sindrome[13]&!sindrome[9]&sindrome[8]) |
(!sindrome[14]&!sindrome[12]&sindrome[10]&!sindrome[9]&sindrome[8]);

o[57] = (sindrome[14]&!sindrome[10]&sindrome[8]) | (sindrome[15]&!sindrome[14]&!sindrome[12]&sindrome[8]) |
(!sindrome[14]&sindrome[13]&!sindrome[12]&sindrome[8]) |
(!sindrome[15]&sindrome[13]&!sindrome[12]&!sindrome[11]&sindrome[9]);

o[53] = (!sindrome[15]&sindrome[13]&!sindrome[11]&sindrome[9]) | (sindrome[15]&!sindrome[14]&!sindrome[12]&sindrome[8])
| (!sindrome[13]&sindrome[11]&!sindrome[10]&sindrome[8]) | (!sindrome[15]&sindrome[11]&!sindrome[9]&sindrome[8]) |
(sindrome[15]&!sindrome[13]&sindrome[11]&!sindrome[8]);

o[49] = (sindrome[15]&sindrome[10]&!sindrome[8]) | (!sindrome[15]&sindrome[13]&!sindrome[11]&sindrome[9]) |
(!sindrome[14]&sindrome[12]&!sindrome[10]&sindrome[8]) |
(!sindrome[15]&sindrome[14]&sindrome[12]&!sindrome[11]&sindrome[10]) |
(sindrome[15]&!sindrome[14]&!sindrome[13]&sindrome[9]&sindrome[8]) |
(!sindrome[14]&!sindrome[12]&sindrome[10]&!sindrome[9]&sindrome[8]);

o[45] = (!sindrome[15]&sindrome[13]&!sindrome[11]&sindrome[9]) | (sindrome[15]&!sindrome[12]&sindrome[11]&!sindrome[9])
| (sindrome[15]&sindrome[11]&sindrome[10]&!sindrome[9]) | (!sindrome[14]&sindrome[12]&!sindrome[10]&sindrome[8]) |
(sindrome[14]&!sindrome[11]&sindrome[9]&!sindrome[8]) |
(sindrome[15]&!sindrome[13]&!sindrome[12]&!sindrome[11]&sindrome[10]);

o[41] = (sindrome[14]&!sindrome[12]&sindrome[11]) | (sindrome[15]&!sindrome[13]&!sindrome[11]&sindrome[9]) |
(!sindrome[15]&sindrome[14]&sindrome[10]&sindrome[8]) | (!sindrome[14]&sindrome[12]&!sindrome[10]&sindrome[8]) |
(sindrome[14]&!sindrome[10]&sindrome[9]&!sindrome[8]) | (sindrome[12]&!sindrome[10]&sindrome[9]&!sindrome[8]);

o[37] = (sindrome[15]&!sindrome[13]&!sindrome[11]&sindrome[9]) | (sindrome[13]&sindrome[11]&!sindrome[10]&sindrome[9]) |
(sindrome[13]&sindrome[11]&sindrome[9]&sindrome[8]) | (sindrome[12]&!sindrome[10]&sindrome[9]&!sindrome[8]) |
(!sindrome[14]&!sindrome[13]&sindrome[12]&!sindrome[10]&sindrome[8]) |
(sindrome[15]&sindrome[12]&sindrome[10]&sindrome[9]&sindrome[8]);

o[33] = (sindrome[15]&sindrome[12]&sindrome[10]&sindrome[8]) | (sindrome[13]&sindrome[12]&sindrome[10]&sindrome[8]) |
(sindrome[12]&sindrome[10]&!sindrome[9]&sindrome[8]) | (sindrome[12]&!sindrome[10]&sindrome[9]&!sindrome[8]) |
```

Figura D.15: Decodificar (20)

```
(sindrome[15]&sindrome[14]&!sindrome[11]&!sindrome[9]&sindrome[8]) |
(sindrome[15]&!sindrome[13]&!sindrome[11]&sindrome[9]&!sindrome[8]);


o[62] = (sindrome[23]&sindrome[21]&!sindrome[19]&sindrome[17]) |
(!sindrome[23]&sindrome[21]&!sindrome[17]&sindrome[16]) |
(!sindrome[22]&!sindrome[20]&sindrome[18]&!sindrome[17]&sindrome[16]);

o[58] = (sindrome[22]&!sindrome[18]&sindrome[16]) | (sindrome[23]&!sindrome[22]&!sindrome[20]&sindrome[16]) |
(!sindrome[22]&sindrome[21]&!sindrome[20]&sindrome[16]) |
(!sindrome[23]&sindrome[21]&!sindrome[20]&!sindrome[19]&sindrome[17]);

o[54] = (!sindrome[23]&sindrome[21]&!sindrome[19]&sindrome[17]) |
(sindrome[23]&!sindrome[22]&!sindrome[20]&sindrome[16]) | (!sindrome[21]&sindrome[19]&!sindrome[18]&sindrome[16]) |
(!sindrome[23]&sindrome[19]&!sindrome[17]&sindrome[16]) | (sindrome[23]&!sindrome[21]&sindrome[19]&!sindrome[16]);

o[50] = (sindrome[23]&sindrome[18]&!sindrome[16]) | (!sindrome[23]&sindrome[21]&!sindrome[19]&sindrome[17]) |
(!sindrome[22]&sindrome[20]&!sindrome[18]&sindrome[16]) |
(!sindrome[23]&sindrome[22]&sindrome[20]&!sindrome[19]&sindrome[18]) |
(sindrome[23]&!sindrome[22]&!sindrome[21]&sindrome[17]&sindrome[16]) |
(!sindrome[22]&!sindrome[20]&sindrome[18]&!sindrome[17]&sindrome[16]);

o[46] = (!sindrome[23]&sindrome[21]&!sindrome[19]&sindrome[17]) |
(sindrome[23]&!sindrome[20]&sindrome[19]&!sindrome[17]) | (sindrome[23]&sindrome[19]&sindrome[18]&!sindrome[17]) |
(!sindrome[22]&sindrome[20]&!sindrome[18]&sindrome[16]) | (sindrome[22]&!sindrome[19]&sindrome[17]&!sindrome[16]) |
(sindrome[23]&!sindrome[21]&!sindrome[20]&!sindrome[19]&sindrome[18]);

o[42] = (sindrome[22]&!sindrome[20]&sindrome[19]) | (sindrome[23]&!sindrome[21]&!sindrome[19]&sindrome[17]) |
(!sindrome[23]&sindrome[22]&sindrome[18]&sindrome[16]) | (!sindrome[22]&sindrome[20]&!sindrome[18]&sindrome[16]) |
(sindrome[22]&!sindrome[18]&sindrome[17]&!sindrome[16]) | (sindrome[20]&!sindrome[18]&sindrome[17]&!sindrome[16]);

o[38] = (sindrome[23]&!sindrome[21]&!sindrome[19]&sindrome[17]) |
(sindrome[21]&sindrome[19]&!sindrome[18]&sindrome[17]) | (sindrome[21]&sindrome[19]&sindrome[17]&sindrome[16]) |
(sindrome[20]&!sindrome[18]&sindrome[17]&!sindrome[16]) |
(!sindrome[22]&!sindrome[21]&sindrome[20]&!sindrome[18]&sindrome[16]) |
(sindrome[23]&sindrome[20]&sindrome[18]&sindrome[17]&sindrome[16]);

o[34] = (sindrome[23]&sindrome[20]&sindrome[18]&sindrome[16]) | (sindrome[21]&sindrome[20]&sindrome[18]&sindrome[16]) |
(sindrome[20]&sindrome[18]&!sindrome[17]&sindrome[16]) | (sindrome[20]&!sindrome[18]&sindrome[17]&!sindrome[16]) |
(sindrome[23]&sindrome[22]&!sindrome[19]&!sindrome[17]&sindrome[16]) |
(sindrome[23]&!sindrome[21]&!sindrome[19]&sindrome[17]&!sindrome[16]);


o[63] = (sindrome[31]&sindrome[29]&!sindrome[27]&sindrome[25]) |
(!sindrome[31]&sindrome[29]&!sindrome[25]&sindrome[24]) |
(!sindrome[30]&!sindrome[28]&sindrome[26]&!sindrome[25]&sindrome[24]);

o[59] = (sindrome[30]&!sindrome[26]&sindrome[24]) | (sindrome[31]&!sindrome[30]&!sindrome[28]&sindrome[24]) |
(!sindrome[30]&sindrome[29]&!sindrome[28]&sindrome[24]) |
(!sindrome[31]&sindrome[29]&!sindrome[28]&!sindrome[27]&sindrome[25]);

o[55] = (!sindrome[31]&sindrome[29]&!sindrome[27]&sindrome[25]) |
(sindrome[31]&!sindrome[30]&!sindrome[28]&sindrome[24]) | (!sindrome[29]&sindrome[27]&!sindrome[26]&sindrome[24]) |
(!sindrome[31]&sindrome[27]&!sindrome[25]&sindrome[24]) | (sindrome[31]&!sindrome[29]&sindrome[27]&!sindrome[24]);

o[51] = (sindrome[31]&sindrome[26]&!sindrome[24]) | (!sindrome[31]&sindrome[29]&!sindrome[27]&sindrome[25]) |
(!sindrome[30]&sindrome[28]&!sindrome[26]&sindrome[24]) |
(!sindrome[31]&sindrome[30]&sindrome[28]&!sindrome[27]&sindrome[26]) |
(sindrome[31]&!sindrome[30]&!sindrome[29]&sindrome[25]&sindrome[24]) |
(!sindrome[30]&!sindrome[28]&sindrome[26]&!sindrome[25]&sindrome[24]);

o[47] = (!sindrome[31]&sindrome[29]&!sindrome[27]&sindrome[25]) |
(sindrome[31]&!sindrome[28]&sindrome[27]&!sindrome[25]) | (sindrome[31]&sindrome[27]&sindrome[26]&!sindrome[25]) |
(!sindrome[30]&sindrome[28]&!sindrome[26]&sindrome[24]) | (sindrome[30]&!sindrome[27]&sindrome[25]&!sindrome[24]) |
(sindrome[31]&!sindrome[29]&!sindrome[28]&!sindrome[27]&sindrome[26]);
```

Figura D.16: Decodificar (20)

```
o[43] = (sindrome[30]&!sindrome[28]&sindrome[27]) | (sindrome[31]&!sindrome[29]&!sindrome[27]&sindrome[25]) |
(!sindrome[31]&sindrome[30]&sindrome[26]&sindrome[24]) | (!sindrome[30]&sindrome[28]&!sindrome[26]&sindrome[24]) |
(sindrome[30]&!sindrome[26]&sindrome[25]&!sindrome[24]) | (sindrome[28]&!sindrome[26]&sindrome[25]&!sindrome[24]);

o[39] = (sindrome[31]&!sindrome[29]&!sindrome[27]&sindrome[25]) |
(sindrome[29]&sindrome[27]&!sindrome[26]&sindrome[25]) | (sindrome[29]&sindrome[27]&sindrome[25]&sindrome[24]) |
(sindrome[28]&!sindrome[26]&sindrome[25]&!sindrome[24]) |
(!sindrome[30]&!sindrome[29]&sindrome[28]&!sindrome[26]&sindrome[24]) |
(sindrome[31]&sindrome[28]&sindrome[26]&sindrome[25]&sindrome[24]);

o[35] = (sindrome[31]&sindrome[28]&sindrome[26]&sindrome[24]) | (sindrome[29]&sindrome[28]&sindrome[26]&sindrome[24]) |
(sindrome[28]&sindrome[26]&!sindrome[25]&sindrome[24]) | (sindrome[28]&!sindrome[26]&sindrome[25]&!sindrome[24]) |
(sindrome[31]&sindrome[30]&!sindrome[27]&!sindrome[25]&sindrome[24]) |
(sindrome[31]&!sindrome[29]&!sindrome[27]&sindrome[25]&!sindrome[24]);


dec[ 0] = enc[32]^o[32];

dec[ 1] = enc[33]^o[33];

dec[ 2] = enc[34]^o[34];

dec[ 3] = enc[35]^o[35];

dec[ 4] = enc[36]^o[36];

dec[ 5] = enc[37]^o[37];

dec[ 6] = enc[38]^o[38];

dec[ 7] = enc[39]^o[39];

dec[ 8] = enc[40]^o[40];

dec[ 9] = enc[41]^o[41];

dec[10] = enc[42]^o[42];

dec[11] = enc[43]^o[43];

dec[12] = enc[44]^o[44];

dec[13] = enc[45]^o[45];

dec[14] = enc[46]^o[46];

dec[15] = enc[47]^o[47];

dec[16] = enc[48]^o[48];

dec[17] = enc[49]^o[49];

dec[18] = enc[50]^o[50];

dec[19] = enc[51]^o[51];

dec[20] = enc[52]^o[52];

dec[21] = enc[53]^o[53];

dec[22] = enc[54]^o[54];

dec[23] = enc[55]^o[55];

dec[24] = enc[56]^o[56];

dec[25] = enc[57]^o[57];

dec[26] = enc[58]^o[58];

dec[27] = enc[59]^o[59];

dec[28] = enc[60]^o[60];
```

Figura D.17: Decodificar (20)

```
dec[29] = enc[61]^o[61];

dec[30] = enc[62]^o[62];

dec[31] = enc[63]^o[63];


ErrD = (sindrome[7]&!sindrome[5]&sindrome[3]&!sindrome[1]) | (sindrome[6]&!sindrome[4]&!sindrome[2]&sindrome[0]) |
(sindrome[6]&!sindrome[4]&sindrome[2]&!sindrome[0]) | (sindrome[7]&sindrome[5]&!sindrome[4]&sindrome[3]&sindrome[1]) |
(!sindrome[7]&!sindrome[5]&sindrome[3]&!sindrome[2]&sindrome[1]) |
(sindrome[7]&!sindrome[6]&sindrome[5]&!sindrome[3]&!sindrome[1]) |
(!sindrome[7]&sindrome[6]&sindrome[4]&sindrome[2]&sindrome[0]) |
(!sindrome[6]&sindrome[4]&!sindrome[3]&sindrome[2]&!sindrome[0]) |
(sindrome[7]&!sindrome[6]&sindrome[4]&!sindrome[1]&!sindrome[0]) |
(sindrome[6]&!sindrome[5]&sindrome[4]&!sindrome[2]&!sindrome[1]&!sindrome[0]) |
(sindrome[6]&!sindrome[5]&!sindrome[4]&sindrome[3]) | (sindrome[6]&!sindrome[5]&sindrome[3]&!sindrome[1]) |
(sindrome[7]&!sindrome[4]&!sindrome[2]&sindrome[0]) | (!sindrome[4]&sindrome[3]&!sindrome[2]&sindrome[0]) |
(sindrome[6]&!sindrome[4]&sindrome[3]&!sindrome[0]) | (sindrome[5]&!sindrome[4]&sindrome[2]&!sindrome[0]) |
(sindrome[7]&!sindrome[6]&!sindrome[5]&sindrome[4]&sindrome[3]) |
(!sindrome[7]&sindrome[6]&!sindrome[5]&!sindrome[4]&sindrome[1]) |
(sindrome[7]&sindrome[5]&!sindrome[3]&sindrome[2]&sindrome[1]) |
(!sindrome[5]&sindrome[4]&!sindrome[3]&sindrome[2]&sindrome[1]) |
(!sindrome[7]&sindrome[5]&!sindrome[3]&sindrome[2]&!sindrome[1]) |
(!sindrome[7]&sindrome[6]&!sindrome[2]&sindrome[1]&sindrome[0]) |
(!sindrome[7]&sindrome[5]&!sindrome[3]&!sindrome[1]&sindrome[0]) |
(sindrome[4]&sindrome[3]&sindrome[2]&!sindrome[1]&sindrome[0]) |
(sindrome[7]&sindrome[6]&!sindrome[2]&!sindrome[1]&sindrome[0]) |
(sindrome[7]&!sindrome[4]&sindrome[3]&sindrome[2]&!sindrome[0]) |
(sindrome[7]&!sindrome[3]&sindrome[2]&!sindrome[1]&!sindrome[0]) |
(sindrome[7]&sindrome[6]&!sindrome[5]&!sindrome[4]&!sindrome[1]&sindrome[0]) |
(sindrome[7]&sindrome[6]&sindrome[5]&!sindrome[3]&sindrome[1]&!sindrome[0]) |
(sindrome[7]&sindrome[5]&sindrome[4]&!sindrome[3]&sindrome[1]&!sindrome[0]) |
(sindrome[6]&sindrome[5]&sindrome[3]&sindrome[2]&sindrome[1]&!sindrome[0]) |
(!sindrome[7]&!sindrome[6]&sindrome[4]&!sindrome[2]&sindrome[1]&!sindrome[0]) |
(!sindrome[7]&sindrome[5]&!sindrome[4]&sindrome[3]&!sindrome[1]&!sindrome[0]) |
(!sindrome[7]&!sindrome[6]&sindrome[5]&!sindrome[4]&!sindrome[3]&!sindrome[2]&sindrome[1]) |
(!sindrome[7]&!sindrome[6]&!sindrome[5]&sindrome[4]&!sindrome[3]&!sindrome[2]&sindrome[0]) |
(!sindrome[7]&!sindrome[6]&!sindrome[4]&!sindrome[3]&sindrome[2]&!sindrome[1]&sindrome[0]) |
(sindrome[7]&!sindrome[6]&!sindrome[5]&!sindrome[3]&!sindrome[2]&sindrome[1]&!sindrome[0])

 | (sindrome[15]&!sindrome[13]&sindrome[11]&!sindrome[9]) | (sindrome[14]&!sindrome[12]&!sindrome[10]&sindrome[8]) |
(sindrome[14]&!sindrome[12]&sindrome[10]&!sindrome[8]) |
(sindrome[15]&sindrome[13]&!sindrome[12]&sindrome[11]&sindrome[9]) |
(!sindrome[15]&!sindrome[13]&sindrome[11]&!sindrome[10]&sindrome[9]) |
(sindrome[15]&!sindrome[14]&sindrome[13]&!sindrome[11]&!sindrome[9]) |
(!sindrome[15]&sindrome[14]&sindrome[12]&sindrome[10]&sindrome[8]) |
(!sindrome[14]&sindrome[12]&!sindrome[11]&sindrome[10]&!sindrome[8]) |
(sindrome[15]&!sindrome[14]&sindrome[12]&!sindrome[9]&!sindrome[8]) |
(sindrome[14]&!sindrome[13]&sindrome[12]&!sindrome[10]&!sindrome[9]&!sindrome[8]) |
(sindrome[14]&!sindrome[13]&!sindrome[12]&sindrome[11]) | (sindrome[14]&!sindrome[13]&sindrome[11]&!sindrome[9]) |
(sindrome[15]&!sindrome[12]&!sindrome[10]&sindrome[8]) | (!sindrome[12]&sindrome[11]&!sindrome[10]&sindrome[8]) |
(sindrome[14]&!sindrome[12]&sindrome[11]&!sindrome[8]) | (sindrome[13]&!sindrome[12]&sindrome[10]&!sindrome[8]) |
(sindrome[15]&!sindrome[14]&!sindrome[13]&sindrome[12]&sindrome[11]) |
(!sindrome[15]&sindrome[14]&!sindrome[13]&!sindrome[12]&sindrome[9]) |
(sindrome[15]&sindrome[13]&!sindrome[11]&sindrome[10]&sindrome[9]) |
(!sindrome[13]&sindrome[12]&!sindrome[11]&sindrome[10]&sindrome[9]) |
(!sindrome[15]&sindrome[13]&!sindrome[11]&sindrome[10]&!sindrome[9]) |
(!sindrome[15]&sindrome[14]&!sindrome[10]&sindrome[9]&sindrome[8]) |
(!sindrome[15]&sindrome[13]&!sindrome[11]&!sindrome[9]&sindrome[8]) |
(sindrome[12]&sindrome[11]&sindrome[10]&!sindrome[9]&sindrome[8]) |
(sindrome[15]&sindrome[14]&!sindrome[10]&!sindrome[9]&sindrome[8]) |
(sindrome[15]&!sindrome[12]&sindrome[11]&sindrome[10]&!sindrome[8]) |
(sindrome[15]&!sindrome[11]&sindrome[10]&!sindrome[9]&!sindrome[8]) |
```

Figura D.18: Decodificar (20)

```
(sindrome[15]&sindrome[14]&!sindrome[13]&!sindrome[12]&!sindrome[9]&sindrome[8]) |
(sindrome[15]&sindrome[14]&sindrome[13]&!sindrome[11]&sindrome[9]&!sindrome[8]) |
(sindrome[15]&sindrome[13]&sindrome[12]&!sindrome[11]&sindrome[9]&!sindrome[8]) |
(sindrome[14]&sindrome[13]&sindrome[11]&sindrome[10]&sindrome[9]&!sindrome[8]) |
(!sindrome[15]&!sindrome[14]&sindrome[12]&!sindrome[10]&sindrome[9]&!sindrome[8]) |
(!sindrome[15]&sindrome[13]&!sindrome[12]&sindrome[11]&!sindrome[9]&!sindrome[8]) |
(!sindrome[15]&!sindrome[14]&sindrome[13]&!sindrome[12]&!sindrome[11]&!sindrome[10]&sindrome[9]) |
(!sindrome[15]&!sindrome[14]&!sindrome[13]&sindrome[12]&!sindrome[11]&!sindrome[10]&sindrome[8]) |
(!sindrome[15]&!sindrome[14]&!sindrome[12]&!sindrome[11]&sindrome[10]&!sindrome[9]&sindrome[8]) |
(sindrome[15]&!sindrome[14]&!sindrome[13]&!sindrome[11]&!sindrome[10]&sindrome[9]&!sindrome[8])

 | (sindrome[23]&!sindrome[21]&sindrome[19]&!sindrome[17]) | (sindrome[22]&!sindrome[20]&!sindrome[18]&sindrome[16]) |
(sindrome[22]&!sindrome[20]&sindrome[18]&!sindrome[16]) |
(sindrome[23]&sindrome[21]&!sindrome[20]&sindrome[19]&sindrome[17]) |
(!sindrome[23]&!sindrome[21]&sindrome[19]&!sindrome[18]&sindrome[17]) |
(sindrome[23]&!sindrome[22]&sindrome[21]&!sindrome[19]&!sindrome[17]) |
(!sindrome[23]&sindrome[22]&sindrome[20]&sindrome[18]&sindrome[16]) |
(!sindrome[22]&sindrome[20]&!sindrome[19]&sindrome[18]&!sindrome[16]) |
(sindrome[23]&!sindrome[22]&sindrome[20]&!sindrome[17]&!sindrome[16]) |
(sindrome[22]&!sindrome[21]&sindrome[20]&!sindrome[18]&!sindrome[17]&!sindrome[16]) |
(sindrome[22]&!sindrome[21]&!sindrome[20]&sindrome[19]) | (sindrome[22]&!sindrome[21]&sindrome[19]&!sindrome[17]) |
(sindrome[23]&!sindrome[20]&!sindrome[18]&sindrome[16]) | (!sindrome[20]&sindrome[19]&!sindrome[18]&sindrome[16]) |
(sindrome[22]&!sindrome[20]&sindrome[19]&!sindrome[16]) | (sindrome[21]&!sindrome[20]&sindrome[18]&!sindrome[16]) |
(sindrome[23]&!sindrome[22]&!sindrome[21]&sindrome[20]&sindrome[19]) |
(!sindrome[23]&sindrome[22]&!sindrome[21]&!sindrome[20]&sindrome[17]) |
(sindrome[23]&sindrome[21]&!sindrome[19]&sindrome[18]&sindrome[17]) |
(!sindrome[21]&sindrome[20]&!sindrome[19]&sindrome[18]&sindrome[17]) |
(!sindrome[23]&sindrome[21]&!sindrome[19]&sindrome[18]&!sindrome[17]) |
(!sindrome[23]&sindrome[22]&!sindrome[18]&sindrome[17]&sindrome[16]) |
(!sindrome[23]&sindrome[21]&!sindrome[19]&!sindrome[17]&sindrome[16]) |
(sindrome[20]&sindrome[19]&sindrome[18]&!sindrome[17]&sindrome[16]) |
(sindrome[23]&sindrome[22]&!sindrome[18]&!sindrome[17]&sindrome[16]) |
(sindrome[23]&!sindrome[20]&sindrome[19]&sindrome[18]&!sindrome[16]) |
(sindrome[23]&!sindrome[19]&sindrome[18]&!sindrome[17]&!sindrome[16]) |
(sindrome[23]&sindrome[22]&!sindrome[21]&!sindrome[20]&!sindrome[17]&sindrome[16]) |
(sindrome[23]&sindrome[22]&sindrome[21]&!sindrome[19]&sindrome[17]&!sindrome[16]) |
(sindrome[23]&sindrome[21]&sindrome[20]&!sindrome[19]&sindrome[17]&!sindrome[16]) |
(sindrome[22]&sindrome[21]&sindrome[19]&sindrome[18]&sindrome[17]&!sindrome[16]) |
(!sindrome[23]&!sindrome[22]&sindrome[20]&!sindrome[18]&sindrome[17]&!sindrome[16]) |
(!sindrome[23]&sindrome[21]&!sindrome[20]&sindrome[19]&!sindrome[17]&!sindrome[16]) |
(!sindrome[23]&!sindrome[22]&sindrome[21]&!sindrome[20]&!sindrome[19]&!sindrome[18]&sindrome[17]) |
(!sindrome[23]&!sindrome[22]&!sindrome[21]&sindrome[20]&!sindrome[19]&!sindrome[18]&sindrome[16]) |
(!sindrome[23]&!sindrome[22]&!sindrome[20]&!sindrome[19]&sindrome[18]&!sindrome[17]&sindrome[16]) |
(sindrome[23]&!sindrome[22]&!sindrome[21]&!sindrome[19]&!sindrome[18]&sindrome[17]&!sindrome[16])

 | (sindrome[31]&!sindrome[29]&sindrome[27]&!sindrome[25]) | (sindrome[30]&!sindrome[28]&!sindrome[26]&sindrome[24]) |
(sindrome[30]&!sindrome[28]&sindrome[26]&!sindrome[24]) |
(sindrome[31]&sindrome[29]&!sindrome[28]&sindrome[27]&sindrome[25]) |
(!sindrome[31]&!sindrome[29]&sindrome[27]&!sindrome[26]&sindrome[25]) |
(sindrome[31]&!sindrome[30]&sindrome[29]&!sindrome[27]&!sindrome[25]) |
(!sindrome[31]&sindrome[30]&sindrome[28]&sindrome[26]&sindrome[24]) |
(!sindrome[30]&sindrome[28]&!sindrome[27]&sindrome[26]&!sindrome[24]) |
(sindrome[31]&!sindrome[30]&sindrome[28]&!sindrome[25]&!sindrome[24]) |
(sindrome[30]&!sindrome[29]&sindrome[28]&!sindrome[26]&!sindrome[25]&!sindrome[24]) |
(sindrome[30]&!sindrome[29]&!sindrome[28]&sindrome[27]) | (sindrome[30]&!sindrome[29]&sindrome[27]&!sindrome[25]) |
(sindrome[31]&!sindrome[28]&!sindrome[26]&sindrome[24]) | (!sindrome[28]&sindrome[27]&!sindrome[26]&sindrome[24]) |
(sindrome[30]&!sindrome[28]&sindrome[27]&!sindrome[24]) | (sindrome[29]&!sindrome[28]&sindrome[26]&!sindrome[24]) |
(sindrome[31]&!sindrome[30]&!sindrome[29]&sindrome[28]&sindrome[27]) |
(!sindrome[31]&sindrome[30]&!sindrome[29]&!sindrome[28]&sindrome[25]) |
(sindrome[31]&sindrome[29]&!sindrome[27]&sindrome[26]&sindrome[25]) |
(!sindrome[29]&sindrome[28]&!sindrome[27]&sindrome[26]&sindrome[25]) |
(!sindrome[31]&sindrome[29]&!sindrome[27]&sindrome[26]&!sindrome[25]) |
(!sindrome[31]&sindrome[30]&!sindrome[26]&sindrome[25]&sindrome[24]) |
```

Figura D.19: Decodificar (20)

```
(!sindrome[31]&sindrome[29]&!sindrome[27]&!sindrome[25]&sindrome[24]) |
(sindrome[28]&sindrome[27]&sindrome[26]&!sindrome[25]&sindrome[24]) |
(sindrome[31]&sindrome[30]&!sindrome[26]&!sindrome[25]&sindrome[24]) |
(sindrome[31]&!sindrome[28]&sindrome[27]&sindrome[26]&!sindrome[24]) |
(sindrome[31]&!sindrome[27]&sindrome[26]&!sindrome[25]&!sindrome[24]) |
(sindrome[31]&sindrome[30]&!sindrome[29]&!sindrome[28]&!sindrome[25]&sindrome[24]) |
(sindrome[31]&sindrome[30]&sindrome[29]&!sindrome[27]&sindrome[25]&!sindrome[24]) |
(sindrome[31]&sindrome[29]&sindrome[28]&!sindrome[27]&sindrome[25]&!sindrome[24]) |
(sindrome[30]&sindrome[29]&sindrome[27]&sindrome[26]&sindrome[25]&!sindrome[24]) |
(!sindrome[31]&!sindrome[30]&sindrome[28]&!sindrome[26]&sindrome[25]&!sindrome[24]) |
(!sindrome[31]&sindrome[29]&!sindrome[28]&sindrome[27]&!sindrome[25]&!sindrome[24]) |
(!sindrome[31]&!sindrome[30]&sindrome[29]&!sindrome[28]&!sindrome[27]&!sindrome[26]&sindrome[25]) |
(!sindrome[31]&!sindrome[30]&!sindrome[29]&sindrome[28]&!sindrome[27]&!sindrome[26]&sindrome[24]) |
(!sindrome[31]&!sindrome[30]&!sindrome[28]&!sindrome[27]&sindrome[26]&!sindrome[25]&sindrome[24]) |
(sindrome[31]&!sindrome[30]&!sindrome[29]&!sindrome[27]&!sindrome[26]&sindrome[25]&!sindrome[24]);

}
```

Figura D.20: Decodificar (20)