UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA

DESARROLLO DEL SOFTWARE DE UN SISTEMA DE SOPORTE PARA SONDAS MÓVILES ANDROID CON YOCTO PROJECT Y ECLIPSE

Autor: Bernardo López Panadero

Tutor: Vicente Torres Carot

Tutor de Empresa: Eduardo Giménez Cebrián

Trabajo Fin de Máster presentado en el Departamento de Ingeniería Electrónica de la Universitat Politècnica de València para la obtención del Título de Máster Universitario en Ingeniería de Sistemas Electrónicos

Curso 2021-22

Valencia, Marzo de 2022

A mis padres.

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. - Linus Torvalds, 1991.

Resumen

En este trabajo se ha desarrollado el software de un sistema electrónico de soporte para móviles Android. La finalidad del sistema de soporte es convertir un móvil Android en una sonda móvil autónoma, llamada Picosonda, accesible y configurable desde el exterior, capaz de realizar pruebas que determinan la calidad de la red de un operador. Esto permitirá conocer la experiencia real del usuario de las redes de telecomunicaciones, lo cual tiene gran importancia para los operadores de red.

El software del sistema de soporte está basado en una distribución de Linux embebido, generada con Yocto Project debido a las características que ofrece para realizar configuraciones y posibles migraciones de manera rápida y ordenada. Se describirán los pasos a seguir para realizar configuraciones sobre el bootloader, el kernel y el espacio de usuario de la distribución de Linux creada para este trabajo.

Para controlar el hardware y llevar a cabo todas las tareas del sistema de soporte se han desarrollado aplicaciones y librerías compartidas en C/C++ con Eclipse IDE. Mediante el uso de librerías compartidas se pretende conseguir un software modular que permita reutilizar las funciones creadas para controlar los periféricos, sensores y actuadores del sistema. La aplicación principal del sistema de soporte carga las librerías y lanza varios hilos que controlan cada una de las tareas que debe realizar.

Por último se expondrán las conclusiones finales tras la realización de este trabajo, así cómo una serie de propuestas de actividades a desarrollar en el futuro.

Resum

En aquest treball s'ha desenvolupat el programari d'un sistema electrònic de suport per a mòbils Android. La finalitat del sistema de suport és convertir un mòbil Android en una sonda mòbil autònoma, anomenada Picosonda, accessible i configurable des de l'exterior, capaç de realitzar proves que determinen la qualitat de la xarxa d'un operador. Això permetrà conéixer l'experiència real de l'usuari de les xarxes de telecomunicacions, la qual cosa té gran importància per als operadors de xarxa.

El programari del sistema de suport està basat en una distribució de Linux embegut, generada amb Yocto Project a causa de les característiques que ofereix per a realitzar configuracions i possibles migracions de manera ràpida i ordenada. Es descriuran els passos a seguir per a realitzar configuracions sobre el bootloader, el kernel i l'espai d'usuari de la distribució de Linux creada per a aquest treball.

Per a controlar el maquinari i dur a terme totes les tasques del sistema de suport s'han desenvolupat aplicacions i llibreries compartides en C/C++ amb Eclipsi IDE. Mitjançant l'ús de llibreries compartides es pretén aconseguir un programari modular que permeta reutilitzar les funcions creades per a controlar els perifèrics, sensors i actuadors del sistema. L'aplicació principal del sistema de suport carrega les llibreries i llança diversos fils que controlen cadascuna de les tasques que ha de realitzar.

Finalment s'exposaran les conclusions finals després de la realització d'aquest treball, així com una sèrie de propostes d'activitats a desenvolupar en el futur.

Resume

In this work has been developed the software of an electronic support system for Android mobile phones. The purpose of the support system is to turn an Android mobile phone into an autonomous mobile probe, called Picosonda, accessible and configurable, capable of performing tests that determine the quality of an operator's network. This will allow to know the real experience of user's telecommunications networks, which is important for operator's network.

The support system software is based on an embedded Linux distribution, built with Yocto Project due to the features it offers to make configurations and migrations quickly and orderly. The steps to configure and modify the bootloader, the kernel and the user space have been described throughout the development of this work.

To control the hardware and carry out all the tasks of the support system, applications and shared libraries coded in C/C++ have been developed in Eclipse IDE. Using shared libraries help to perform a modular software that allows easy testing and reuse functions created to control the peripherals, sensors and actuators of the system. The main application of support system loads the libraries and launches several threads that control each of the tasks that must perform.

Final conclusions will be presented after the completion of this work, as well as a series of proposals for activities to be carried out in the future.

Índice general

Índice de figuras

Listado de siglas empleadas

ADB Android Debug Bridge.

ARM Advanced RISC Machine.

CAN Controller Area Network.

CI Circuito Integrado.

eMMC embedded MultiMediaCard.

GPIO General Purpose Input Output.

I2C Inter-Integrated Circuit.

LED Light Emitting Diode.

PCB Circuito Impreso (Printed Circuit Board).

PWM Pulse Width Modulation.

RAM Random Access Memory.

ROM Read Only Memory.

SAI Sistema de alimentación ininterrumpida.

SD Secure Digital.

SDIO Secure Digital Input Output.

SIM Subscriber Identity Module.

SOM System On Module.

SPI Serial Peripheral Interface.

SPL Secondary Program Loader.

SRAM Static Random Access Memory.

TCP Transmission Control Protocol.

UART Universal Asynchronous Receiver and Transmitter.

USB Universal Serial Bus.

Parte I

Memoria

Capítulo 1

Introducción

1.1. Acerca de la compañía MedUX

Actualmente, la calidad del servicio y la experiencia del usuario en redes de telecomunicación fijas y móviles son variables esenciales para el negocio de los operadores de red. Para determinar la calidad real de la red es necesario analizar los datos desde la experiencia real del usuario. Por ello, la compañía MedUX ha desarrollado una aplicación para dispositivos fijos y móviles Android capaz de realizar pruebas que determinan la calidad de la red de los operadores de telecomunicaciones.

La ejecución de pruebas en redes fijas se puede realizar sobre dispositivos Android diseñados para funcionar de manera estática en un lugar determinado, como puede ser un Android TV X96. Este tipo de dispositivos están provistos de un sistema de alimentación capaz de obtener energía de la red eléctrica, lo cual les permite operar de manera ininterrumpida. En cambio, la ejecución de pruebas en redes móviles requiere que las mismas se realicen en un dispositivo móvil Android que no está preparado para funcionar de forma totalmente autónoma, por lo que se necesita un sistema de soporte para lograr este objetivo.

1.2. Motivación

El sistema de soporte necesita utilidades y herramientas de software complejas para llevar a cabo todas las tareas que le han sido asignadas, por lo que se ha optado por utilizar el sistema operativo Linux en una distribución personalizada. En una primera instancia se comenzó a desarrollar la distribución de Linux con Docker, donde se creaba un entorno de Ubuntu 18.04 en el que se descargaban e instalaban todas las herramientas necesarias para la generación de la distribución, siguiendo una serie de instrucciones establecidas por el programador.

El problema de desarrollar la imagen de esta forma es que la capacidad de migrar toda la capa de software a otra plataforma hardware queda muy reducida, pues se tendría que volver a revisar todo el proceso de generación de la imagen. Para solventar este problema de portabilidad, la imagen Linux se generará con Yocto Project, lo cual asegura que una posible migración de hardware se realice de una manera más rápida y eficiente.

1.3. Objetivos

El objetivo principal de este trabajo es el desarrollo del software de un sistema electrónico de soporte para móviles Android, el cual permitirá convertir un movil Android en una sonda que funciona de manera autónoma, capaz de medir la calidad de las redes de telecomunicación. El sistema de soporte permitirá monitorizar las condiciones ambientales y eléctricas del terminal y realizar configuraciones o actualizaciones de forma remota.

La implementación del software se realizará de manera ordenada, escalable y modular, lo cual permitirá reutilizar el código fuente en otras aplicaciones o proyectos. Se realizará también la verificación de cada módulo con tests unitarios que aseguran que las distintas piezas de software funcionan correctamente antes de combinarlas en una aplicación más compleja.

El desarrollo de este trabajo parte de un diseño hardware ya establecido, por tanto, únicamente será objeto de estudio el software que correrá sobre el mismo.

Capítulo 2

Sistema de soporte para móviles Android

2.1. Introducción

El sistema de soporte para móviles Android, mostrado en la siguiente figura, monitoriza que las condiciones ambientales y eléctricas de funcionamiento sean correctas. Es configurable y capaz de realizar actualizaciones del móvil Android cuando es necesario.

Para llevar a cabo todas estas funciones, el sistema de soporte cuenta con un SOM Variscite DART-6UL-5G, sensor de temperatura, ventiladores, conexión USB entre el móvil Android y el sistema de soporte, un sensor de potencia, comunicación CAN, Wifi, LEDs, un portaSIM y dos adaptadores de tarjeta micro SD. La energía eléctrica que consume el sistema de soporte en el escenario real de aplicación está suministrada por un SAI externo.

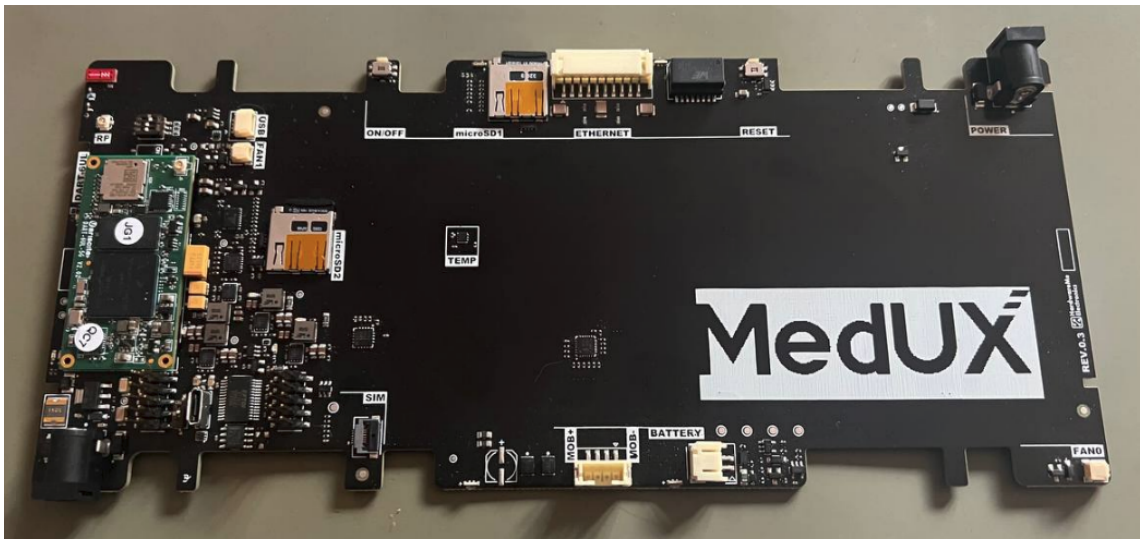


Figura 2.1: Sistema de soporte

El software que correrá en el sistema de soporte será una distribución personalizada de Linux, debido a la gran cantidad de drivers y utilidades que permiten agilizar el proceso de desarrollo y mejorar la interacción con el mismo.

La comunicación entre el sistema de soporte y el móvil Android se realizará mediante USB. Sobre éste se establecerá una comunicación entre ambos mediante la herramienta de software ADB[1], la cual permite al usuario comunicarse con un dispositivo Android y realizar una gran variedad de acciones sobre el mismo, como instalar y depurar aplicaciones, subir y descargar ficheros o ejecutar comandos en el dispositivo, entre otras.

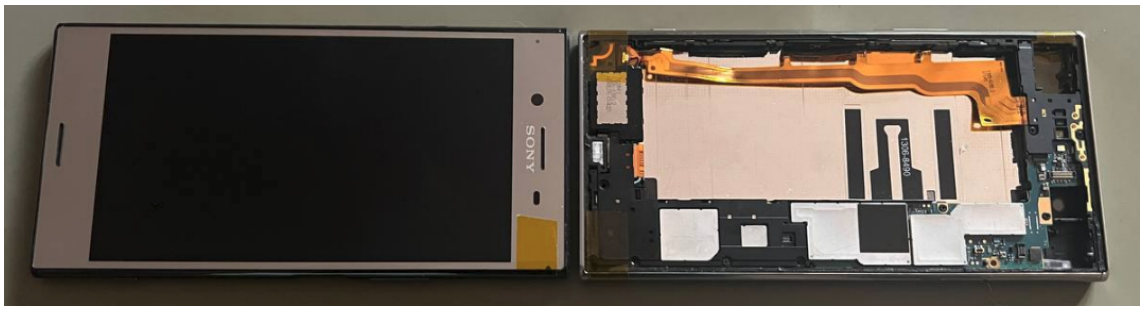
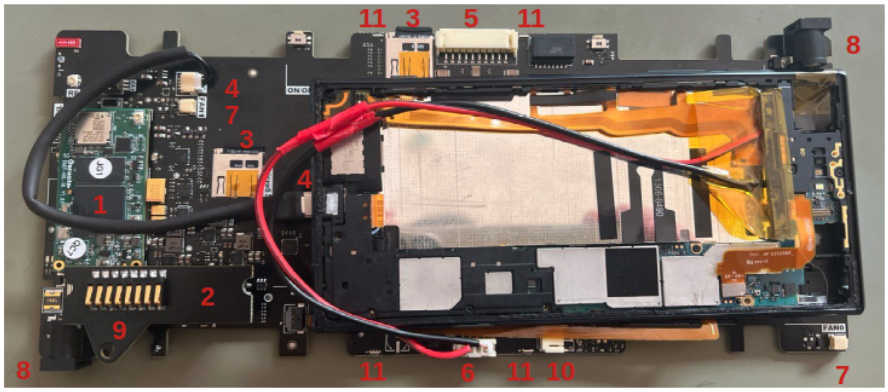


Figura 2.2: Móvil Android

El móvil Android se conecta al sistema de soporte sin batería, ya que es el propio sistema quien suministra la tensión de 4V2 que necesita el terminal para funcionar. Las pruebas para determinar la calidad de la red del teleoperador se realizan haciendo uso de las aplicaciones y acciones más comunes entre usuarios, como puede ser visualizar un video de Youtube, realizar una llamada de voz o cargar una página web con Google Chrome, entre otras. Antes de instalar el móvil Android en el sistema de soporte, es necesario activar en el mismo las opciones de desarrollador, habilitar la depuración por USB y rootearlo para tener control total del sistema operativo.

Al conjunto formado entre el móvil Android y el sistema de soporte se le llamará Picosonda.



1. SOM Variscite
2. PortaSIM
3. Tarjeta SD
4. Conector USB móvil
5. Conector Ethernet
6. Conector batería móvil
7. Conector ventilador
8. Conector alimentación
9. Conector bus CAN
10. Conector para la detección de alimentación por SAI
11. Led RGB

Figura 2.3: Picosonda

2.2. SOM Variscite

El sistema de soporte cuenta con un SOM Variscite DART-6UL-5G, del cual se pueden destacar las siguientes características principales[2]:

- Procesador IMX6ULL de NXP con un ARM Cortex A7 hasta 900MHz
- Memoria RAM DDR3L de 256MB
- Memoria FLASH (eMMC) de 8GB
- Módulo de comunicaciones Wifi-Bluetooth
- Interfaz para tarjeta SD
- Ethernet
- Comunicación CAN, I2C, SPI y UART
- GPIOs

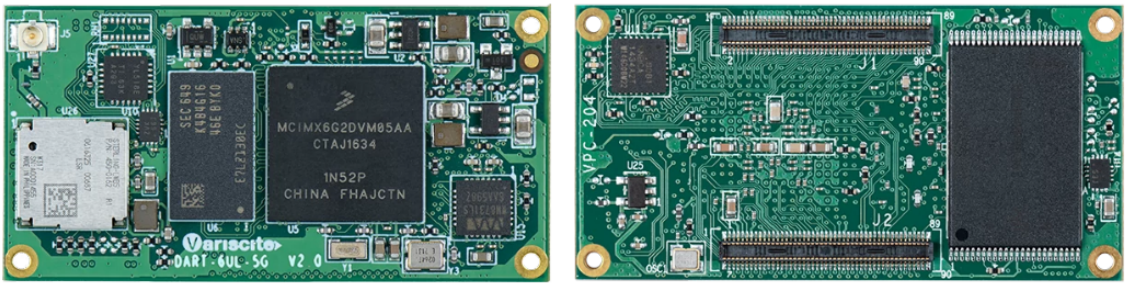


Figura 2.4: SOM Variscite Dart-6UL-5G [2]

Este módulo es el encargado de correr la distribución Linux que controlará todos los elementos hardware del sistema de soporte. El hecho de utilizar un SOM facilita el diseño de la PCB, ya que el diseñador hardware no tiene que preocuparse de rutar líneas de transmisión críticas, como pueden ser las que conectan la memoria RAM DDR3L con el procesador.

Variscite ofrece soporte para generar la imagen Linux con los siguientes proyectos:

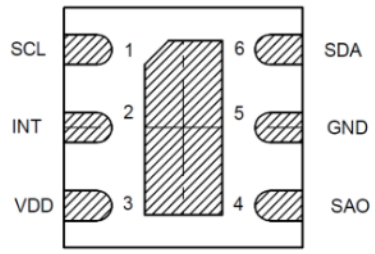
- Yocto Project
- Debian
- Boot2Qt

En este caso se ha optado por elegir Yocto Project debido a las interesantes características y ventajas que ofrece frente a los otros, las cuales se describirán más detalladamente en el Capítulo 3.

2.3. Sensor de temperatura

La temperatura de la Picosonda se monitoriza de manera constante para que no sufra daños debido a un exceso de calor.

El sensor utilizado en este caso es un WSEN-TIDS de Würth Elektronik, el cual incluye una interfaz de comunicación I2C para su configuración y posterior lectura del valor de temperatura. En la siguiente figura se muestra la descripción de los pines del sensor.



Pin No.	Name	Function	I/O	Comments
1	*SCL*	I²C serial clock	Input	
2	*INT*	Interrupt	Output	Do not connect if not used
3	*VDD*	Positive supply voltage	Supply	
4	*SAO*	I²C device address selection	Input	High: device address is 0111000b Low: device address is 0111111b
5	*GND*	Negative supply voltage	Supply	
6	*SDA*	I²C serial data	Input/Output	

Figura 2.5: Descripción de pines del sensor WSEN-TIDS [3]

La inicialización del sensor se realiza escribiendo en algunos de los registros de configuración que se muestran a continuación.

Addr.	Register Name	Type	Bits								Comments
			7	6	5	4	3	2	1	0	
0x01	*DEVICE_ID*	R	1	0	1	0	0	0	0	0	Device ID register
0x02	*T_H_LIMIT*	R/W	THL[7:0]								Temperature limit register
0x03	*T_L_LIMIT*	R/W	TLL[7:0]								
0x04	*CTRL*	R/W	0	BDU	AVG[1:0]		IF_ADD_INC	FREE-RUN	0	ONE_SHOT	Control register
0x05	*STATUS*	R	0	0	0	0	0	UNDER_TLL	OVER_THL	BUSY	Status register
0x06	*DATA_T_L*	R	DATA_T[7:0]								Temperature output registers
0x07	*DATA_T_H*	R	DATA_T[15:8]								
0x0C	*SOFT_RESET*	R/W	0	0	0	0	0	0	SW RE-SET	0	Software reset register

Figura 2.6: Registros del sensor WSEN-TIDS [3]

En este caso se ha optado por una configuración en modo continuo de operación, en el cual el sensor realiza medidas de temperatura de manera continua, refrescando el valor obtenido en los registros DATA_T_L (0x06) y DATA_T_H (0x07) de forma periódica. Para llevar a cabo esta configuración se ha seguido el siguiente diagrama proporcionado por el fabricante del sensor.

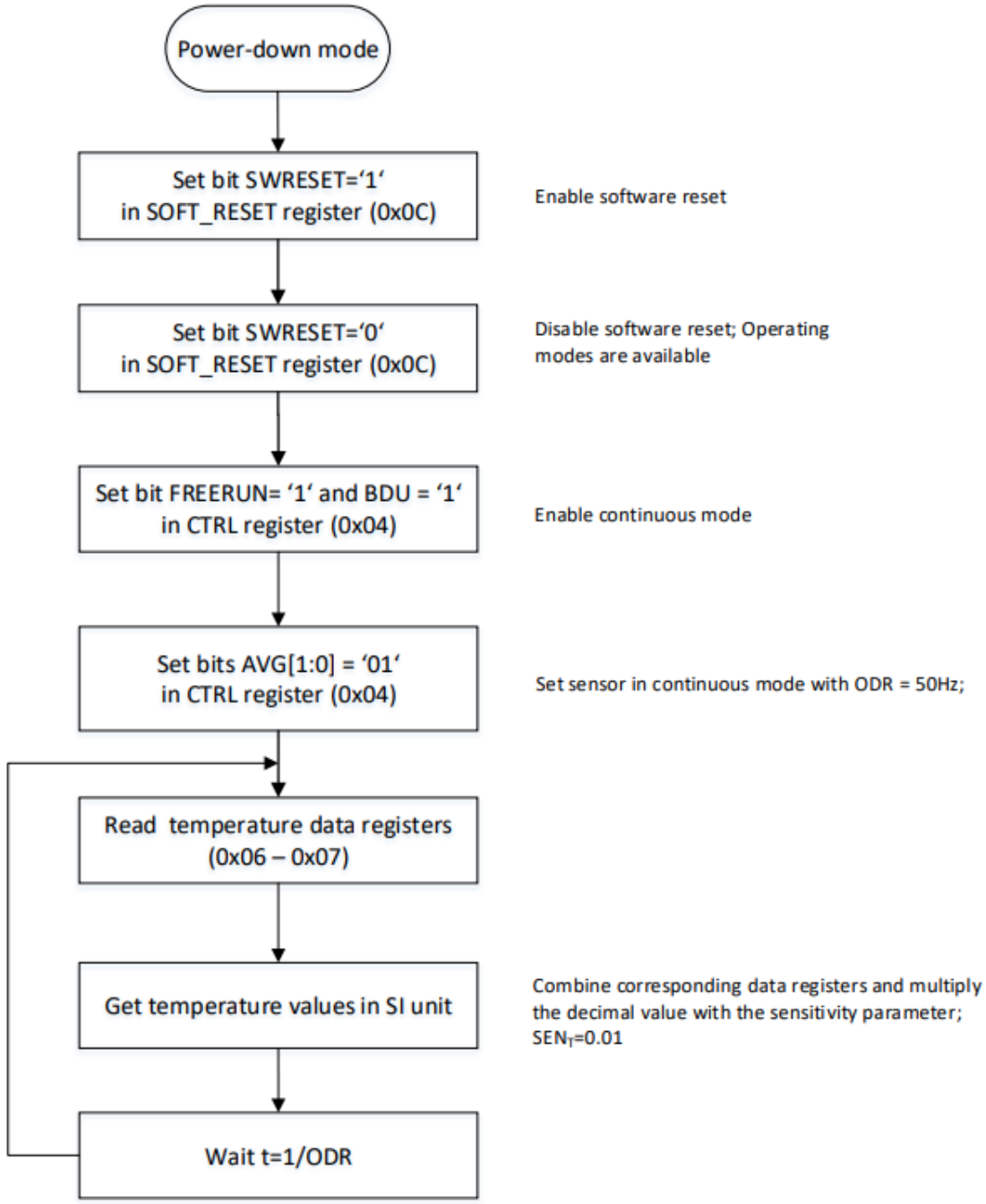


Figura 2.7: Diagrama de configuración y puesta en marcha del sensor WSEN-TIDS [3]

Una vez configurado el sensor en el modo de operación descrito anteriormente, es posible comenzar a leer el valor de temperatura de los registros DATA_T_L (0x06) y DATA_T_H (0x07). El valor de temperatura se obtiene mediante la siguiente expresión:

$$\text{Temperature} = [(\text{DATA\_T\_H} * 256) + (\text{DATA\_T\_L})] * 0{,}01 \tag{2.1}$$

2.4. Ventiladores

El sistema de soporte consta de dos ventiladores. Tienen la función de disipar el calor del interior de la Picosonda en caso de que la temperatura sea demasiado elevada. Su velocidad de giro está controlada por una señal PWM.

Figura 2.8: Ventiladores de la Picosonda

El hecho de instalar ventiladores se debe a que la Picosonda tiene muchas posibilidades de funcionar en entornos de calor extremos, como podría ser el maletero de un taxi en Ciudad de México en pleno verano.

2.5. Sensor de Potencia

El sensor de potencia PAC1932 de Microchip es el encargado de chequear los cuatro canales de alimentación del sistema de soporte.

- Canal 1: Alimentación 3V3 del SOM Variscite.
- Canal 2: Alimentación 3V3 de los sensores del sistema de soporte.
- Canal 3: Alimentación 4V2 de la batería del móvil Android.
- Canal 4: Alimentación 5V del USB del móvil Android.

Entre sus principales características, cabe destacar que tiene la capacidad de medir rangos de voltaje entre 0V y 32V, es configurable para un modo bidireccional de medida de la corriente y la resolución de las mediciones que realiza es de 16 bits. Este sensor solo se encarga de monitorizar los canales de alimentación, es decir, no tiene capacidad de iniciar o detener el flujo de corriente por los mismos.

Las alimentaciones de la batería del móvil Android de 4V2 y del conector USB de 5V se pueden controlar, respectivamente, mediante los niveles lógicos de dos GPIOs, de modo que la alimentación del canal estará habilitada si el GPIO correspondiente se encuentra a nivel alto (1) o estará deshabilitada si el GPIO se encuentra a nivel bajo (0).

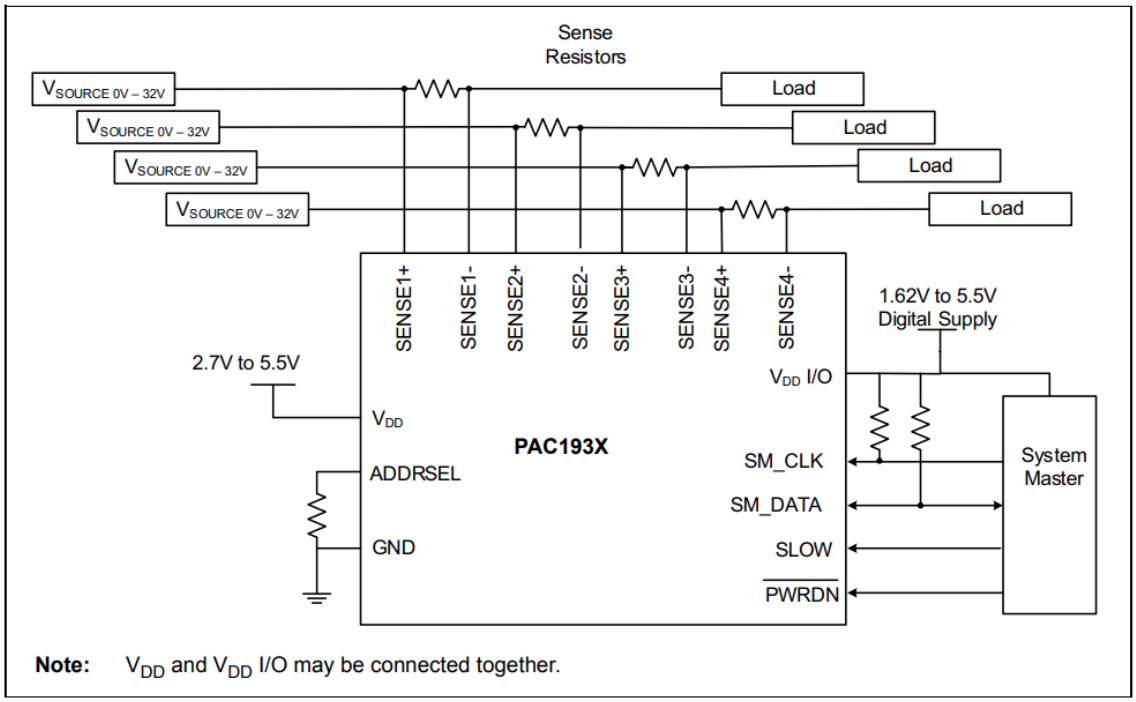


Figura 2.9: Diagrama del sensor PAC1932 [4]

La comunicación entre el sensor y el SOM se realiza mediante el bus I2C. En la siguiente figura se muestra la tabla de registros:

0x00	W	1 byte	REFRESH	0x0F	R	2 bytes	VBUS1_AVG				
0x01	R/W	1 byte	CTRL	0x10	R	2 bytes	VBUS2_AVG				
0x02	R	3 bytes	ACC_COUNT	0x11	R	2 bytes	VBUS3_AVG				
0x03	R	6 bytes	VPOWER1_ACC	0x12	R	2 bytes	VBUS4_AVG	0x1F	W	1 byte	REFRESH_V
0x04	R	6 bytes	VPOWER2_ACC	0x13	R	2 bytes	VSENSE1_AVG	0x20	R/W	1 byte	SLOW
0x05	R	6 bytes	VPOWER3_ACC	0x14	R	2 bytes	VSENSE2_AVG	0x21	R	1 byte	CTRL_ACT
0x06	R	6 bytes	VPOWER4_ACC	0x15	R	2 bytes	VSENSE3_AVG	0x22	R	1 byte	CHANNEL_DIS_ACT
0x07	R	2 bytes	VBUS1	0x16	R	2 bytes	VSENSE4_AVG	0x23	R	1 byte	NEG_PWR_ACT
0x08	R	2 bytes	VBUS2	0x17	R	4 bytes	VPOWER1	0x24	R	1 byte	CTRL_LAT
0x09	R	2 bytes	VBUS3	0x18	R	4 bytes	VPOWER2	0x25	R	1 byte	CHANNEL_DIS_LAT
0x0A	R	2 bytes	VBUS4	0x19	R	4 bytes	VPOWER3	0x26	R	1 byte	NEG_PWR_LAT
0x0B	R	2 bytes	VSENSE1	0x1A	R	4 bytes	VPOWER4				
0x0C	R	2 bytes	VSENSE2	0x1C	R/W	1 byte	CHANNEL_DIS	0xFD	R	1 byte	PID
0x0D	R	2 bytes	VSENSE3	0x1D	R/W	1 byte	NEG_PWR	0xFE	R	1 byte	MID
0x0E	R	2 bytes	VSENSE4	0x1E	W	1 byte	REFRESH_G	0xFF	R	1 byte	REV

Figura 2.10: Registros del sensor PAC1932 [4]

El encendido y apagado del sensor también se puede realizar mediante un GPIO del SOM, alimentando el mismo si el GPIO correspondiente se encuentra a nivel alto (1) o apagándolo si el GPIO se encuentra a nivel bajo (0).

En el sensor PAC1932 una dirección de registro puede contener varios bytes. Por ejemplo, la dirección VPOWER1_ACC (0x03) es de 6 bytes y la primera lectura que se obtiene de ese registro es la del byte más significativo. Se necesitan cinco lecturas más de ese registro para completar el valor de VPOWER1_ACC.

Para realizar el cálculo de los valores eléctricos de cada canal es necesario calcular, en primer lugar, el valor constante "Full-Scale Current". Para ello se ha utilizado la siguiente expresión, la cual viene dada en la ecuación 4.3 del datasheet del sensor [4], donde RSENSE es el valor de la resistencia externa, que en este caso es de 0.22Ω.

$$\mathrm{PAC1932\_FSC} = \frac{100mV}{\mathrm{RSENSE}} \tag{2.2}$$

Después hay que calcular otro valor constante, el "Power Full-Scale Range", que se calcula aplicando la ecuación 4.5 del datasheet del sensor [4].

$$\mathrm{PAC1932\_PowerFSR} = \frac{3{,}2}{\mathrm{RSENSE}} \tag{2.3}$$

Puesto que las lecturas de tensión y corriente se realizarán en modo unipolar, se han definido los siguientes valores en hexadecimal indicados en el datasheet del sensor para trabajar en este modo:

$$\mathrm{UNIPOLAR\_MODE} = 0x10000000 \tag{2.4}$$

$$\mathrm{UNIPOLAR\_MEAS} = 0x10000 \tag{2.5}$$

Una vez realizado el cálculo de estos valores constantes, se procede a hacer las lecturas de los canales de alimentación. En primer lugar se envia el comando REFRESH para que se actualicen los valores de los registros del sensor. Después hay que esperar al menos 1ms para que los valores de los registros se estabilicen. Transcurrido un tiempo prudencial de 1,5ms se procede a realizar la lectura de los registros ACC_COUNT, VPOWER_ACC, VBUS y VSENSE de los cuatro canales. El valor ACC_COUNT se corresponde con el total de muestras acumuladas. El valor VPOWER_ACC es el valor de potencia acumulado, mientras que los valores VBUS y VSENSE son valores de tensión y corriente instantáneos.

Para obtener el valor de potencia, voltaje o corriente de un canal hay que aplicar la siguientes expresiones:

$$\mathrm{Power\_Channel\_x} = \frac{\mathrm{VPOWERx\_ACC} * \mathrm{PAC1932\_PowerFSR}}{\mathrm{UNIPOLAR\_MODE} * \mathrm{ACC\_COUNT}} \tag{2.6}$$

$$\mathrm{Voltage\_Channel\_x} = \frac{\mathrm{VBUSx} * 32}{\mathrm{UNIPOLAR\_MEAS}} \tag{2.7}$$

$$\mathrm{Current\_Channel\_x} = \frac{\mathrm{VSENSEx} * \mathrm{PAC1932\_FSC}}{\mathrm{UNIPOLAR\_MEAS}} \tag{2.8}$$

2.6. PortaSIM

El sistema de soporte incluye una pequeña PCB extraíble denominada portaSIM, la cual está conectada al mismo mediante unos pines. Su cometido es portar hasta dos tarjetas SIM que puedan ser introducidas o extraídas desde el exterior por un técnico sin necesidad de abrir la Picosonda completamente para acceder al móvil Android.

El SOM se encarga de multiplexar con un GPIO para que el móvil Android tenga acceso a una u otra. Además, en el SOM también se ha configurado un pin de entrada que detecta si el portaSIM está conectado al sistema de soporte o no.

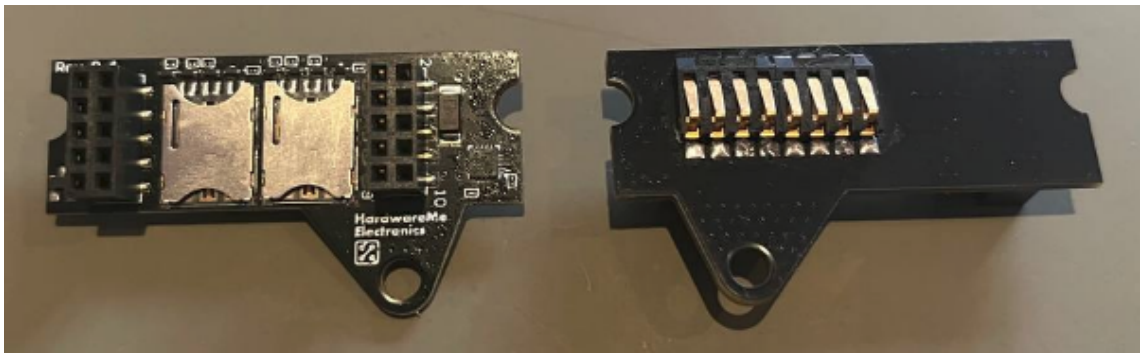


Figura 2.11: PortaSIM del sistema de soporte

La conexión entre el portaSIM y el móvil Android se realiza mediante un cable flex, el cual se conecta en un extremo a un conector del sistema de soporte y el otro extremo se introduce en el móvil Android simulando la presencia de tarjeta SIM. De esta forma se puede acceder a la tarjeta SIM fácilmente sin necesidad de abrir la caja de la Picosonda.

Figura 2.12: Cable flex de tarjeta SIM

En el portaSIM también se encuentra el bus de comunicación CAN, accesible desde el exterior por el conector que aparece en el portaSIM de la derecha de la Figura 2.11.

2.7. Bus CAN

La comunicación CAN que incorpora el sistema de soporte permite conectar por cable más sistemas de soporte apilados uno encima de otro en forma de tándem. Este bus permitirá compartir información entre distintas Picosondas: por ejemplo, una Picosonda funcionando en modo máster podría enviar información a otras para sincronizar la ejecución de pruebas.

Durante el desarrollo de la comunicación CAN se ha utilizado un cable PEAK [5] con conexión USB en un extremo y líneas CAN TX y CAN RX en el otro extremo.

Figura 2.13: Cable PEAK [5]

La visualización de las tramas CAN enviadas y recibidas se puede realizar a través de PCAN-View, un software gratuito para los usuarios del cable PEAK.

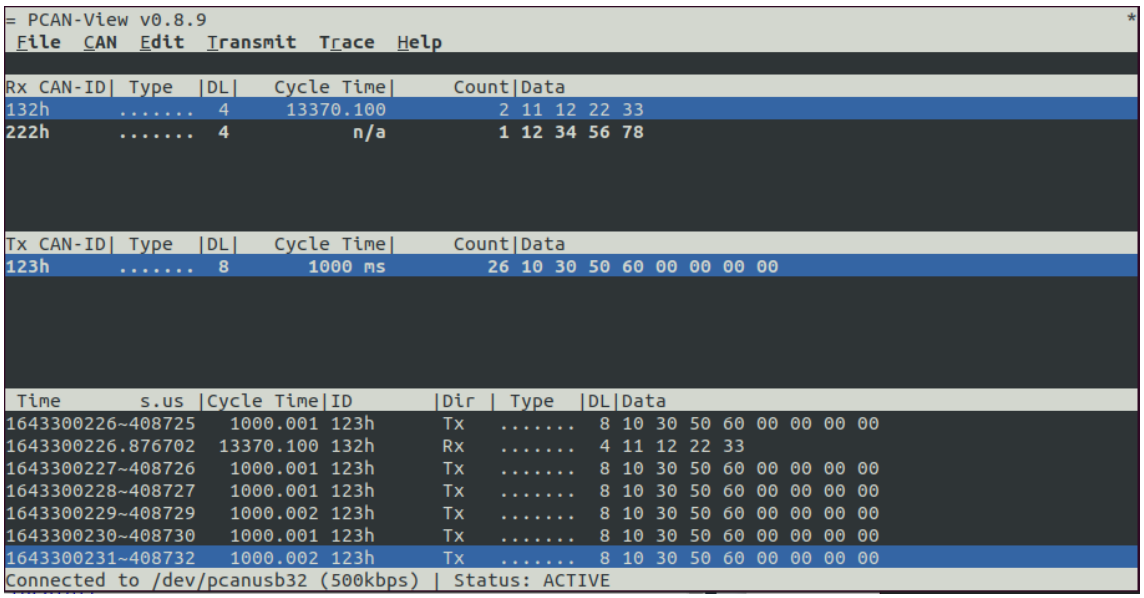


Figura 2.14: PCAN-View

2.8. Controlador de LEDs

El sistema de soporte consta del circuito integrado PCA9532 capaz de controlar hasta 16 entradas/salidas. El CI ofrece la posibilidad de configurar las salidas con PWM, lo cual lo hace ideal para controlar LEDs. La comunicación entre el CI y el SOM se realiza mediante el bus I2C. En la siguiente figura se muestra un circuito de aplicación típico para el PCA9532.

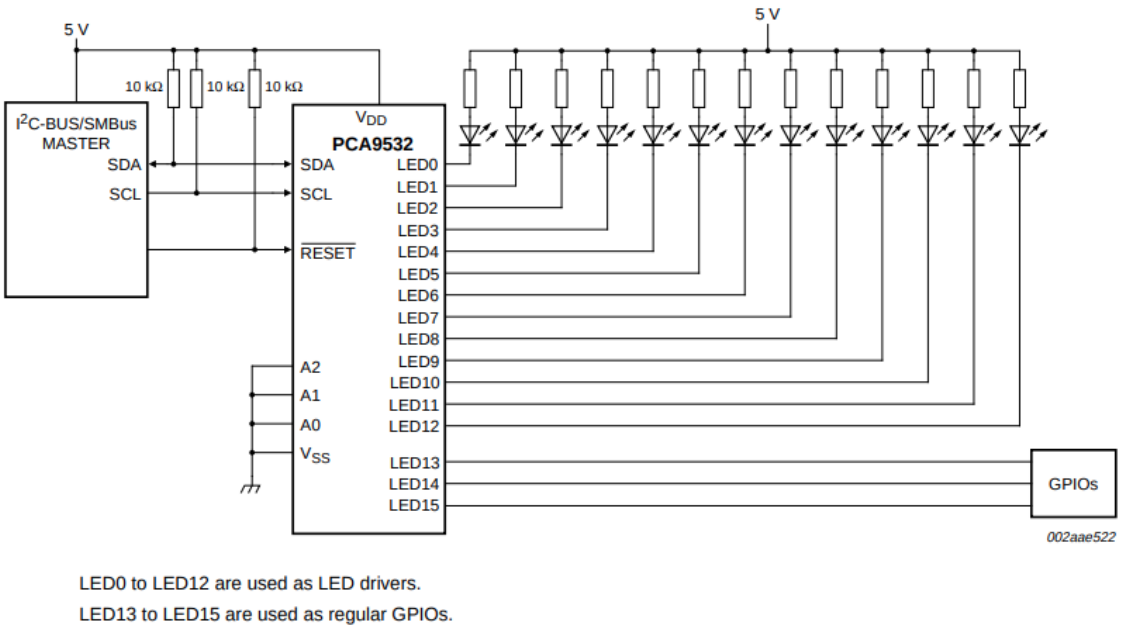


Figura 2.15: Circuito de aplicación típico para PCA9532 [6]

El encendido y apagado del controlador de LEDs también se puede realizar mediante un GPIO del SOM, encendiéndolo si el GPIO correspondiente se encuentra a nivel alto (1) o apagándolo si el GPIO se encuentra a nivel bajo (0).

El sistema de soporte consta de 4 LEDs RGB que representan mediante colores el estado en que se encuentra la Picosonda. De esta forma se podría detectar cualquier tipo de incidencia de forma visual.

- LED 1: Indica el estado de la temperatura del interior de la Picosonda.
- LED 2: Indica el estado de los canales de alimentación de la Picosonda.
- LED 3: Indica el estado de la comunicación con el móvil Android.
- LED 4: Indica si el portaSIM está conectado al sistema de soporte.

A continuación se muestra la tabla de registros del PCA9532:

B3	B2	B1	B0	Symbol	Access	Description
0	0	0	0	INPUT0	read only	input register 0
0	0	0	1	INPUT1	read only	input register 1
0	0	1	0	PSC0	read/write	frequency prescaler 0
0	0	1	1	PWM0	read/write	PWM register 0
0	1	0	0	PSC1	read/write	frequency prescaler 1
0	1	0	1	PWM1	read/write	PWM register 1
0	1	1	0	LS0	read/write	LED0 to LED3 selector
0	1	1	1	LS1	read/write	LED4 to LED7 selector
1	0	0	0	LS2	read/write	LED8 to LED11 selector
1	0	0	1	LS3	read/write	LED12 to LED15 selector

Figura 2.16: Registros del PCA9532 [6]

Los registros INPUT0 e INPUT1 sirven para leer el estado de las 16 entradas/salidas. Los registros PSC0, PWM0, PSC1 y PWM1 sirven para configurar dos valores de preescaler y dos valores de PWM. Por último, los registros LS0, LS1, LS2 y LS3 sirven para configurar las salidas del PCA9532. La siguiente figura muestra los bits de selección para cada salida:

Register	Bit	Value	Description
LS0 - LED0 to LED3 selector			
LS0	7:6	00*	LED3 selected
	5:4	00*	LED2 selected
	3:2	00*	LED1 selected
	1:0	00*	LED0 selected
LS1 - LED4 to LED7 selector			
LS1	7:6	00*	LED7 selected
	5:4	00*	LED6 selected
	3:2	00*	LED5 selected
	1:0	00*	LED4 selected
LS2 - LED8 to LED11 selector			
LS2	7:6	00*	LED11 selected
	5:4	00*	LED10 selected
	3:2	00*	LED9 selected
	1:0	00*	LED8 selected
LS3 - LED12 to LED15 selector			
LS3	7:6	00*	LED15 selected
	5:4	00*	LED14 selected
	3:2	00*	LED13 selected
	1:0	00*	LED12 selected

Figura 2.17: Bits de selección para las salidas del PCA9532 [6]

Los posibles valores para cada salida son los siguientes:

- 00 = LED off
- 01 = LED on
- 10 = Output blinks at PWM0 rate
- 11 = Output blinks at PWM1 rate

2.9. Tarjetas micro SD

El SOM Variscite tiene una sola interfaz SDIO. Sin embargo, en los requisitos de funcionalidad del sistema de soporte se necesitan dos tarjetas SD:

- SD 1: Tarjeta SD interna, la cual no es accesible desde el exterior y cuya funcionalidad es almacenar una imagen del móvil Android que permita actualizarlo en caso de que la Picosonda se encuentre en una localización con poca cobertura y se requiera esta acción.
- SD 2: Tarjeta SD externa, accesible desde el exterior y cuya funcionalidad es permitir a un técnico leer o escribir datos de la Picosonda en ella sin necesidad de abrirla por completo. Al ser un componente accesible desde el exterior, en principio este adaptador de tarjetas SD irá vacío y solo se utilizará en presencia de un técnico para evitar posibles hurtos.

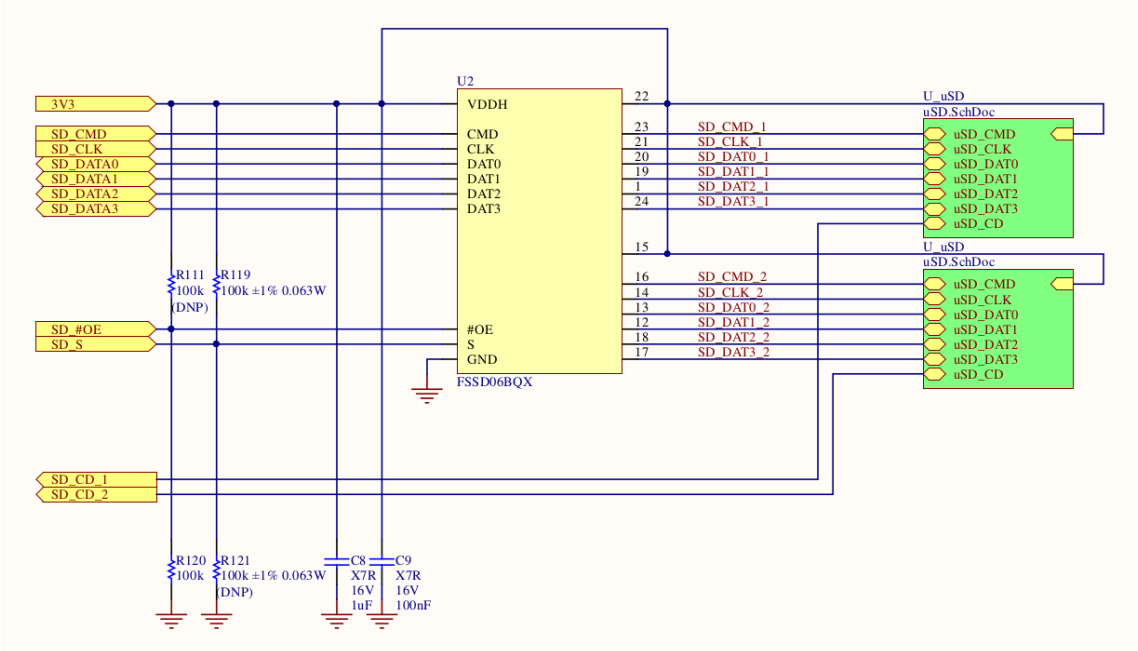


Figura 2.18: Esquemático de tarjetas SD

En la figura 2.18 se muestra cómo se ha resuelto el problema de tener solo una interfaz SDIO disponible, instalando el multiplexor de interfaz SDIO FSSD06BQX [7] en la PCB. La multiplexión entre una tarjeta SD y otra se realiza mediante los pines SD_OE y SD_S controlados por GPIOs del SOM.

Por otro lado, la tarjeta SD puede servir, además de dispositivo de almacenamiento, como fuente de arranque del SOM. El SOM tiene un switch hardware mediante el cual se selecciona si el sistema operativo arrancará de la tarjeta SD, de la eMMC o de la memoria NAND. En condiciones normales el switch estará configurado para que el SOM arranque desde la eMMC. Sin embargo, si por cualquier motivo se necesitase arrancar desde la tarjeta SD sería un técnico quien tendría que cambiar de posición ese switch e introducir una SD booteable en el adaptador externo (SD 2).

2.10. Punto de acceso Wifi

El sistema de soporte es capaz de crear un punto de acceso Wifi que permitirá a todos los dispositivos conectados a su red acceder a una pagina web de configuración. Mediante dicha web, un técnico podrá realizar cambios o identificar problemas en la Picosonda.

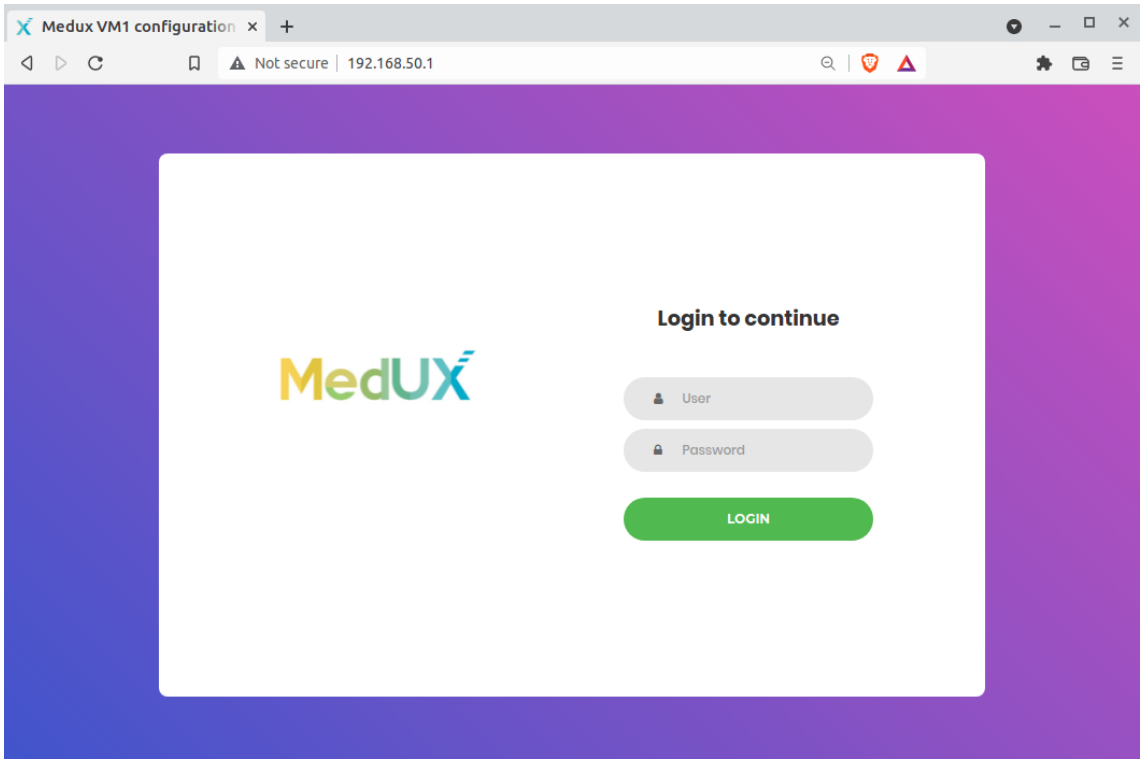


Figura 2.19: Web de configuración

En el SOM Variscite DART-6UL-5G la interfaz del módulo Wifi-Bluetooth se comparte con la interfaz de tarjetas SD. No es posible tener activo el módulo de Wifi-Bluetooth y la tarjeta SD al mismo tiempo.

Capítulo 3

Yocto Project

3.1. Introducción

Yocto Project es un proyecto Open Source que proporciona una estructura de directorios estándar para realizar configuraciones, compilaciones y migraciones de distribuciones de Linux embebido de forma sencilla y escalable. Yocto Project es la combinación de varios proyectos y elementos[8]:

- Utiliza el build system OpenEmbedded, mantenido también por OpenEmbedded Project, el cual soporta gran cantidad de arquitecturas hardware. Bitbake es la herramienta principal que ejecuta las acciones que puede realizar el build system.
- Su distribución Linux de referencia se llama Poky.
- Consta de un conjunto de herramientas integradas que permiten realizar compilaciones, integraciones, pruebas, etc... de forma automatizada.

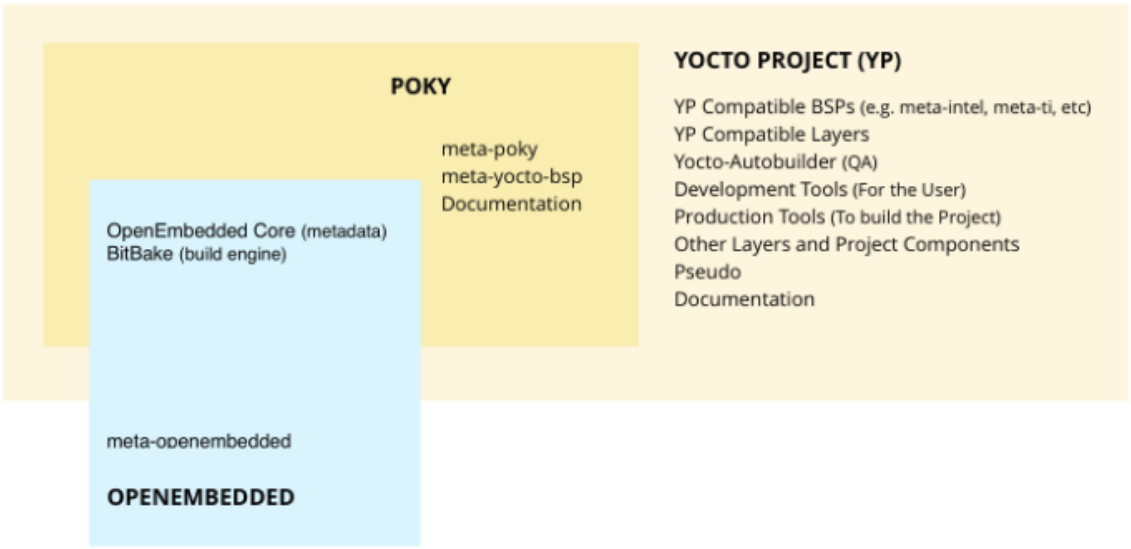


Figura 3.1: Elementos de Yocto Project [9]

3.2. Cómo funciona

El build system OpenEmbedded, integrado en Yocto Proyect, sigue un flujo de trabajo ordenado hasta generar la imagen de la distribución Linux, el cual se muestra en la siguiente figura:

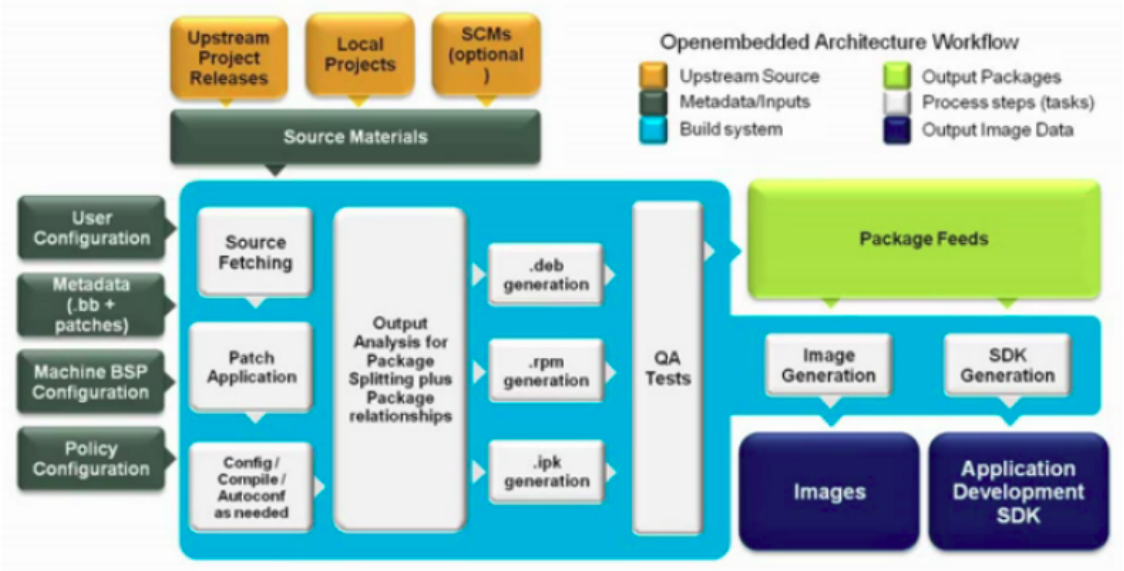


Figura 3.2: OpenEmbedded build system workflow

En primer lugar el desarrollador debe especificar la arquitectura, parches, licencias y detalles de configuración. Después el build system descarga y descomprime el código fuente, principalmente de sistemas de repositorios Git.

Una vez el código está descargado, el build system extrae las fuentes en un área local de trabajo dentro del equipo donde las parchea, configura y compila. Los binarios generados se instalan en un fichero de imagen que contiene el root filesystem, el kernel de Linux, el device tree blob, el U-Boot y el SPL.

Bitbake parsea todas las instrucciones que debe ejecutar el build system, crea un árbol de dependencias para realizar una compilación ordenada y finalmente genera la imagen de la distribución de Linux. Es similar a la herramienta “make”. Durante la generación de la imagen, el build system realiza un proceso de compilación cruzada de todos los paquetes de software.

Bitbake permite hacer generaciones parciales de la imagen, es decir, se podría ejecutar para que, por ejemplo, solo descargue, parchee, configure y compile el kernel de Linux. O bien, para que solo lo configure. O que solo lo compile. Esto permite agilizar el proceso de desarrollo de la distribución Linux.

3.3. Sistema de Capas

Yocto Project tiene un modelo de desarrollo de imágenes de Linux embebido que se distingue de otros modelos al estar estructurado en capas. Este modelo está diseñado para que pueda ser soportado tanto por nuevas personalizaciones, como por mantenedores del software ya existente.

Las capas en Yocto Project son directorios que comienzan por el nombre de "meta-". En el contenido del repositorio que ofrece Variscite para comenzar a trabajar con Yocto Project se pueden encontrar los siguientes ficheros y directorios:

- build_x11: En este directorio se almacenan todos los paquetes de código fuente que se descargan e instalan durante la generación de la imagen. Además, también hay algunos ficheros de configuración de algunos parámetros de la compilación de la imagen.
- README: Fichero con información relativa a las capas que contiene el repositorio.
- setup-environment: Script que configura algunas variables de entorno y que se debe ejecutar antes de comenzar a trabajar con el build system.
- sources: Bajo este directorio se encuentran todas las capas del proyecto. Entre ellas se pueden destacar las mas relevantes, las cuales son la de Openembedded, Poky, Freescale y las de Variscite.

Figura 3.3: Sistema de capas del Variscite DART-6UL-5G

Una vez descargado el repositorio de Variscite, se procederá a inicializar el entorno de desarrollo mediante el siguiente comando:

```
$ source setup-environment build_x11
```

A partir de este momento ya se puede comenzar a ejecutar acciones del build system con Bitbake. En primer lugar se creará y añadirá la capa de software de este proyecto al conjunto de capas proporcionado por Variscite, la cual se llamará “meta-app”.

```
$ bitbake-layers create-layer ../sources/meta-app/
$ bitbake-layers add-layer ../sources/meta-app/
```

Tras ejecutar los comandos anteriores, se puede observar que la capa “meta-app” ha sido creada bajo el directorio sources/

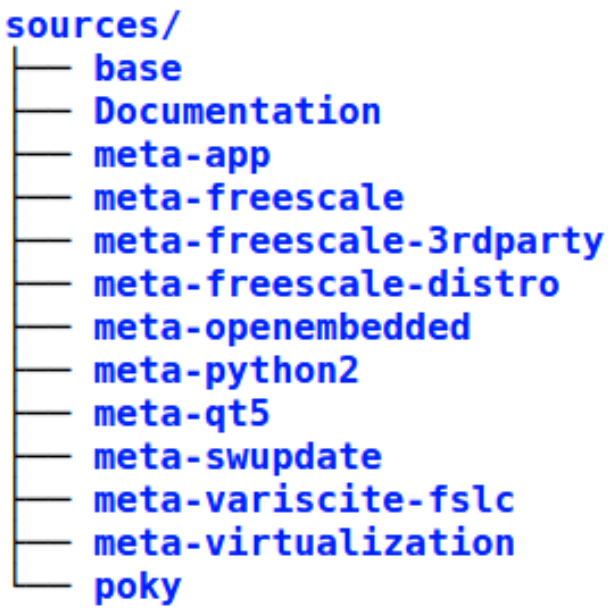


Figura 3.4: Capa meta-app añadida al directorio sources/

Las capas separan de forma lógica la información de la imagen de la distribución Linux. Por ejemplo, una capa podría tener la información del BSP, de una GUI o de una aplicación en concreto, entre otros. Incluir en una sola capa toda la información acerca de las instrucciones o configuraciones que tiene que llevar a cabo el build system complica futuras personalizaciones de la imagen.

Las capas también pueden contener cambios en instrucciones o configuraciones que habían sido definidas en otras capas. Esto es posible debido a que Yocto Project organiza las capas dándoles distintos valores de prioridad, donde la mayor prioridad se corresponde con el valor más alto. Para ver la prioridad asignada a cada capa basta con ejecutar el siguiente comando:

```
$ bitbake-layers show-layers
NOTE: Starting bitbake server...
layer                 path                                      priority
==========================================================================
meta                  ~/var-fslc-yocto/sources/poky/meta  5
meta-poky             ~/var-fslc-yocto/sources/poky/meta-poky  5
meta-oe               ~/var-fslc-yocto/sources/meta-openembedded/meta-oe  6
meta-multimedia       ~/var-fslc-yocto/sources/meta-openembedded/meta-multimedia  6
meta-python           ~/var-fslc-yocto/sources/meta-openembedded/meta-python  7
meta-filesystems      ~/var-fslc-yocto/sources/meta-openembedded/meta-filesystems  6
meta-gnome            ~/var-fslc-yocto/sources/meta-openembedded/meta-gnome  7
meta-networking       ~/var-fslc-yocto/sources/meta-openembedded/meta-networking  5
meta-freescale        ~/var-fslc-yocto/sources/meta-freescale  5
meta-freescale-3rdparty ~/var-fslc-yocto/sources/meta-freescale-3rdparty  4
meta-freescale-distro   ~/var-fslc-yocto/sources/meta-freescale-distro  4
meta-qt5              ~/var-fslc-yocto/sources/meta-qt5  7
meta-swupdate         ~/var-fslc-yocto/sources/meta-swupdate  6
meta-virtualization   ~/var-fslc-yocto/sources/meta-virtualization  8
meta-variscite-fslc   ~/var-fslc-yocto/sources/meta-variscite-fslc  9
meta-app              ~/var-fslc-yocto/sources/meta-app  30
```

Si en una capa de, por ejemplo, prioridad 5, se realiza una configuración y en otra capa con una prioridad superior se modifica dicha configuración, la que prevalecerá será la de la capa de mayor prioridad. A la capa “meta-app” se le ha dado un valor de prioridad 30 para que todas las configuraciones que se realicen en esta capa se apliquen en el caso de que alguna de ellas ya esté definida en capas inferiores. En Yocto Project el valor máximo de prioridad de una capa es 99.

3.4. Recipes

Por lo general, las capas son repositorios que contienen un conjunto de ficheros denominados recipes. Un conjunto de recipes relacionados entre sí forman una capa. Son ficheros con el sufijo .bb que contienen una lista de configuraciones e instrucciones que le dicen al build system qué hacer. También contienen la información de dónde se encuentra el código fuente y los parches que deben aplicarse. Pueden contener información acerca de las dependencias del paquete o de dependencias de otros recipes. En la capa meta-app los recipes se agrupan en cuatro tipos:

- Recipes del filesystem (recipes-filesystem): Contiene los recipes relacionados con la configuración del filesystem. En este caso se han creado dos recipes: uno para añadir ficheros de configuración y otro para añadir servicios de arranque. Consulta en Anexo 1 y Anexo 2.
- Recipes de la imagen Linux (recipes-images): Contiene un recipe con el nombre de la imagen que se va a generar, que en este caso se llamará “app-image.bb”. En este recipe están definidos los paquetes y utilidades que se instalarán en la imagen. Consulta en Anexo 3.
- Recipes del kernel (recipes-kernel): Contiene los recipes que aplican las configuraciones y los parches que se hayan realizado en el kernel o en el device tree. Consulta en Anexo 4.
- Recipes de software de usuario (recipes-software): Contiene los recipes que descargan, compilan e instalan las aplicaciones, librerias, scripts y programas de test creados por el usuario. Consulta en Anexo 5, Anexo 6, Anexo 7 y Anexo 8.

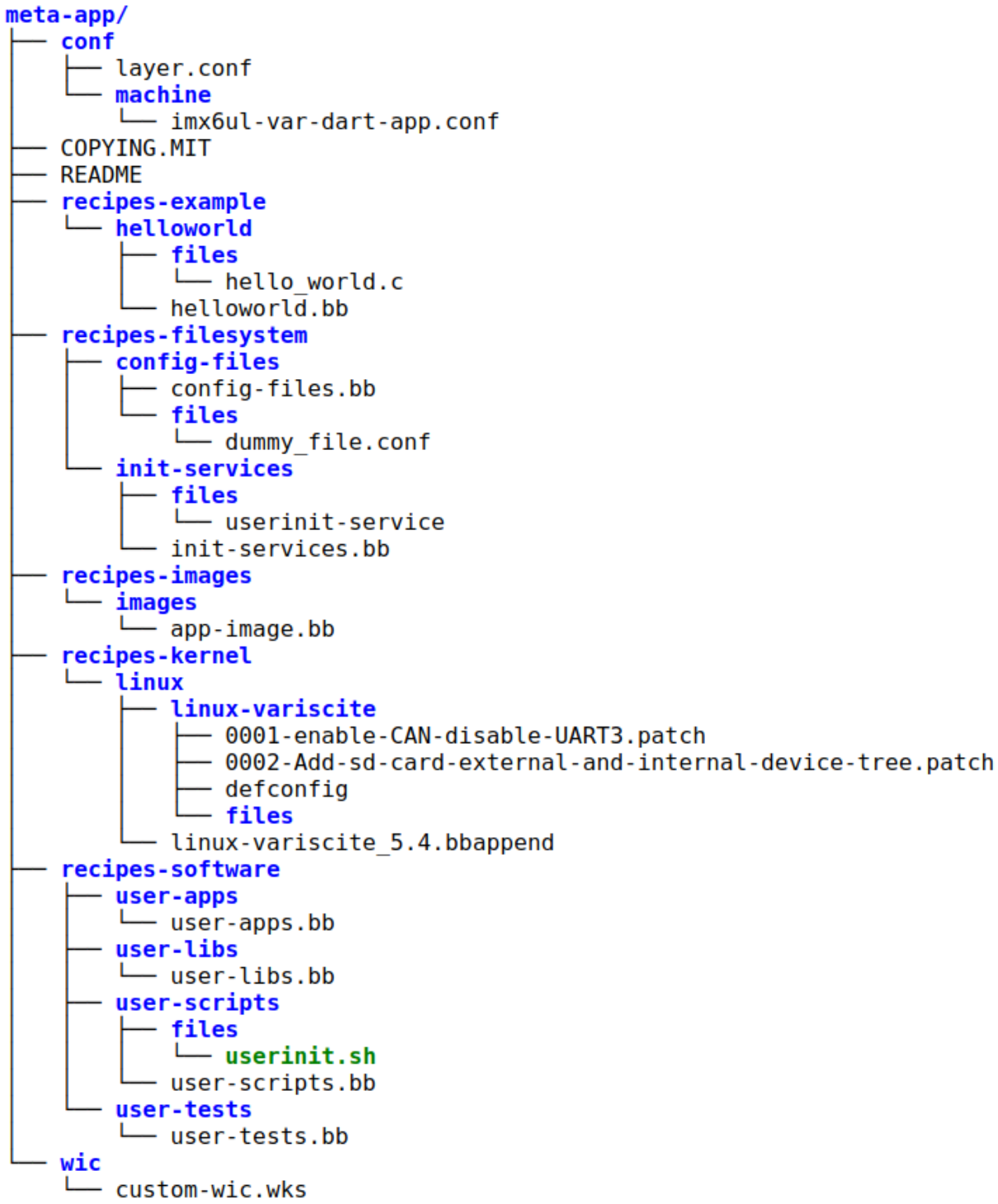


Figura 3.5: Directorios capa meta-app

En caso de querer añadir más instrucciones o configuraciones a un recipe que ya existe en alguna capa, lo recomendable no es modificar dicho recipe, sino crear uno nuevo con la extension .bbappend en la capa sobre la que se esté trabajando y añadir en este las nuevas modificaciones.

A continuación se muestra el recipe “helloworld.bb” a modo de ejemplo, en el que se compila un fichero en C y se instala en la imagen.

```
SUMMARY = "A simple Hello World application"
LICENSE = "MIT"
LIC_FILES_CHKSUM = "file://${COMMON_LICENSE_DIR}/MIT;md5=0835
    ade698e0bcf8506ecda2f7b4f302"
SRC_URI = "file://hello_world.c"

TARGET_CC_ARCH += "${LDFLAGS}"

S = "${WORKDIR}"

do_compile() {
  ${CC} -Wall hello_world.c -o hello_world
}

do_install() {
  install -d ${D}${bindir}
  install -m 0755 ${S}/hello_world ${D}${bindir}
}
```

El recipe consta en primer lugar de una pequeña descripción y del tipo de licencia que tiene. Después hay que indicar dónde se encuentran las fuentes mediante la variable SRC_URI. En este caso la fuente se encuentra en el mismo directorio que el recipe, pero también se podría indicar que está en un repositorio de, por ejemplo, Github, indicando la URL del repositorio en dicha variable.

En la variable TARGET_CC_ARCH se especifican los flags de compilación necesarios para que la aplicación pueda funcionar sobre la arquitectura especificada.

La variable ${S} (source) almacena el directorio de trabajo.

La función do_compile() se le indica al build system qué comando debe ejecutar para compilar las fuentes. La variable ${CC} hace referencia al compilador que se esté usando. En este caso el compilador será arm-linux-gnueabihf.

Por último, la función do_install() se encarga de instalar las fuentes en la imagen. En este caso, al tratarse de un fichero ejecutable, se instalarán en el directorio ${bindir}, el cual se corresponde con /usr/bin en la imagen del sistema operativo.

Bitbake permite generar recipes de forma individual. Para ello, únicamente hay que indicarle el nombre del recipe sin la extensión .bb. Una vez finalizado el recipe, se ejecutará el siguiente comando para generarlo y comprobar que no hay errores:

```
$ bitbake helloworld
```

De esta forma se puede comprobar que la sintaxis del recipe que se ha creado es correcta si el build system no reporta ningún error. Además de esto, se puede comprobar que las configuraciones e instrucciones del recipe se aplicarán correctamente. Para ello se utilizará la herramienta "devshell":

```
$ bitbake -c devshell helloworld
```

El comando anterior abrirá una nueva shell en el directorio donde se hayan depositado todos los objetos relacionados con el recipe. En este caso, nos lleva al siguiente directorio:

```
~/var-fslc-yocto/build_x11/tmp/work/cortexa7t2hf-neon-fslc-linux-gnueabi/helloworld#
```

En él podemos buscar si el ejecutable "hello_world" se instalará correctamente en la imagen. Si mostramos el contenido del directorio se puede visualizar lo siguiente:

```
~/var-fslc-yocto/build_x11/tmp/work/cortexa7t2hf-neon-fslc-linux-gnueabi/helloworld# ls

-rw-r--r-- 1 1000 1000    65 Jan 29 12:37 configure.sstate
-rw-r--r-- 1 root root   492 Jan 29 12:37 debugsources.list
drwxr-xr-x 3 1000 1000  4096 Jan 29 12:37 deploy-rpms
drwxrwxr-x 2 1000 1000  4096 Jan 29 12:37 deploy-source-date-epoch
-rwxr-xr-x 1 1000 1000 11548 Jan 29 12:37 hello_world
-rw-rw-r-- 1 1000 1000   110 Jan 29 12:36 hello_world.c
-rw-r--r-- 1 root root  2274 Jan 29 12:37 helloworld.spec
drwxr-xr-x 3 root root  4096 Jan 29 12:37 image
drwxrwxr-x 3 1000 1000  4096 Jan 29 12:37 license-destdir
drwxr-xr-x 3 root root  4096 Jan 29 12:37 package
drwxr-xr-x 9 root root  4096 Jan 29 12:37 packages-split
drwxr-xr-x 2 1000 1000  4096 Jan 29 12:37 patches
drwxr-xr-x 6 root root  4096 Jan 29 12:37 pkgdata
drwxr-xr-x 6 1000 1000  4096 Jan 29 12:37 pkgdata-pdata-input
drwxr-xr-x 6 1000 1000  4096 Jan 29 12:37 pkgdata-sysroot
drwxrwxr-x 2 1000 1000  4096 Jan 29 16:01 pseudo
drwxrwxr-x 5 1000 1000  4096 Jan 29 16:01 recipe-sysroot
drwxrwxr-x 9 1000 1000  4096 Jan 29 12:37 recipe-sysroot-native
drwxr-xr-x 2 1000 1000  4096 Jan 29 12:37 source-date-epoch
drwxrwxr-x 2 1000 1000  4096 Jan 29 12:37 sstate-install-deploy_source_date_epoch
drwxr-xr-x 2 root root  4096 Jan 29 12:37 sstate-install-package
drwxrwxr-x 2 1000 1000  4096 Jan 29 12:37 sstate-install-packagedata
drwxrwxr-x 2 1000 1000  4096 Jan 29 12:37 sstate-install-package_qa
drwxr-xr-x 2 1000 1000  4096 Jan 29 12:37 sstate-install-package_write_rpm
drwxrwxr-x 2 1000 1000  4096 Jan 29 12:37 sstate-install-populate_lic
drwxr-xr-x 2 1000 1000  4096 Jan 29 12:37 sstate-install-populate_sysroot
drwxr-xr-x 3 1000 1000  4096 Jan 29 12:37 sysroot-destdir
drwxrwxr-x 2 1000 1000 12288 Jan 29 16:01 temp
```

El ejecutable hello_world se puede encontrar en este mismo directorio en quinto lugar empezando por arriba. Si además se desea comprobar que se ha instalado en el directorio correcto en la imagen se mostrará el contenido del directorio image/:

```
$ ls -R image/

image/usr/bin:
hello_world
```

El ejecutable se instalará en el directorio /usr/bin de la imagen Linux, que es el directorio donde se pueden encontrar los ejecutables de usuario.

Bajo el directorio de la capa meta-app también se pueden encontrar, a parte de los recipes, los directorios conf/ y wic/.

- conf: Contiene los ficheros de configuración de la capa. Consulta en Anexo 9 y Anexo 10.
- wic: Contiene un fichero que indica al build system las particiones de memoria que tendrá el fichero de imagen. Consulta en Anexo 11. La distribución del fichero de imagen se verá más detalladamente en el Capítulo 4.

3.5. Objetos generados

En el directorio build/ (build_x11 en el caso de trabajar con esta distribución) se almacena el código fuente de los paquetes descargados, configuraciones, información sobre el estado de la compilación de la imagen y los objetos que se generan tras su generación, entre otros.

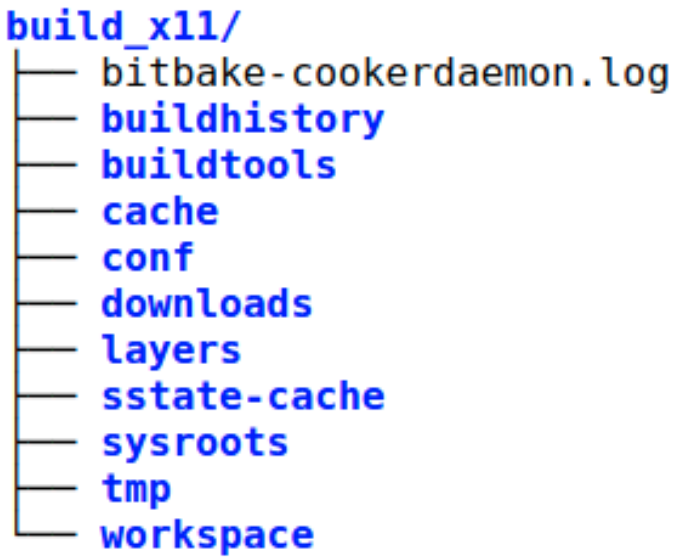


Figura 3.6: Directorio build_x11

Bajo el directorio build/conf/ se puede configurar la compilación del sistema operativo. Contiene los siguientes ficheros:

```
$ ls -l ~/var-fslc-yocto/build_x11/conf

-rw-rw-r-- 1 user user   876 nov 23 18:01 bblayers.conf
-rw-rw-r-- 1 user user   629 ene 11 15:15 local.conf
-rw-rw-r-- 1 user user 11726 nov  9 12:51 local.conf.sample
-rw-rw-r-- 1 user user    15 nov  9 12:51 templateconf.cfg
```

A continuación se muestra el fichero bblayers.conf, el cual contiene la información sobre las capas que se van a tener en cuenta durante el proceso de generación de la imagen. Este fichero se actualiza automáticamente cuando se crean las capas con Bitbake.

```
LCONF_VERSION = "7"

BBPATH = "${TOPDIR}"
BSPDIR := "${@os.path.abspath(os.path.dirname(d.getVar('FILE', True)) +
    '/../..')}"

BBFILES ?= ""
BBLAYERS = " \
  ${BSPDIR}/sources/poky/meta \
  ${BSPDIR}/sources/poky/meta-poky \
  ${BSPDIR}/sources/meta-openembedded/meta-oe \
  ${BSPDIR}/sources/meta-openembedded/meta-multimedia \
  ${BSPDIR}/sources/meta-openembedded/meta-python \
  ${BSPDIR}/sources/meta-openembedded/meta-filesystems \
  ${BSPDIR}/sources/meta-openembedded/meta-gnome \
  ${BSPDIR}/sources/meta-openembedded/meta-networking \
  ${BSPDIR}/sources/meta-freescale \
  ${BSPDIR}/sources/meta-freescale-3rdparty \
  ${BSPDIR}/sources/meta-freescale-distro \
  ${BSPDIR}/sources/meta-qt5 \
  ${BSPDIR}/sources/meta-swupdate \
  ${BSPDIR}/sources/meta-virtualization \
  ${BSPDIR}/sources/meta-variscite-fslc \
  ${BSPDIR}/sources/meta-app \
  "
```

El fichero local.conf, que se muestra a continuación, contiene la información acerca de la arquitectura hardware y de los paquetes que se generan. En este fichero cabe destacar la variable MACHINE, la cual indica al build system para qué arquitectura debe compilar la distribución.

```
#MACHINE ??= 'qemuarm'
MACHINE ??= 'imx6ul-var-dart-app'
DISTRO ?= 'fslc-x11'
PACKAGE_CLASSES ?= 'package_rpm'
EXTRA_IMAGE_FEATURES ?= "debug-tweaks"
USER_CLASSES ?= "buildstats image-mklibs image-prelink"
```

```
PATCHRESOLVE = "noop"
BB_DISKMON_DIRS ??= "\
    STOPTASKS,${TMPDIR},1G,100K \
    STOPTASKS,${DL_DIR},1G,100K \
    STOPTASKS,${SSTATE_DIR},1G,100K \
    STOPTASKS,/tmp,100M,100K \
    ABORT,${TMPDIR},100M,1K \
    ABORT,${DL_DIR},100M,1K \
    ABORT,${SSTATE_DIR},100M,1K \
    ABORT,/tmp,10M,1K"
PACKAGECONFIG_append_pn-qemu-system-native = " sdl"
CONF_VERSION = "1"

DL_DIR ?= "${BSPDIR}/downloads/"
ACCEPT_FSL_EULA = "1"
```

En este caso se podría configurar para que la imagen que se genere funcione en el hardware real de Variscite dándole el valor “imx6ul-var-dart-app”, o bien para que genere una imagen que funcione en un entorno virtual creado por Qemu, dándole el valor de “qemuarm” (actualmente está comentado poniendo un # delante de la variable).

Qemu es un emulador Open Source capaz de virtualizar la ejecución de una distribución Linux diseñada para funcionar sobre una arquitectura hardware distinta. Una vez finalizada la generación de la imagen con la variable MACHINE=’qemuarm’ se puede arrancar en Qemu con el siguiente comando:

```
$ runqemu qemuarm
```

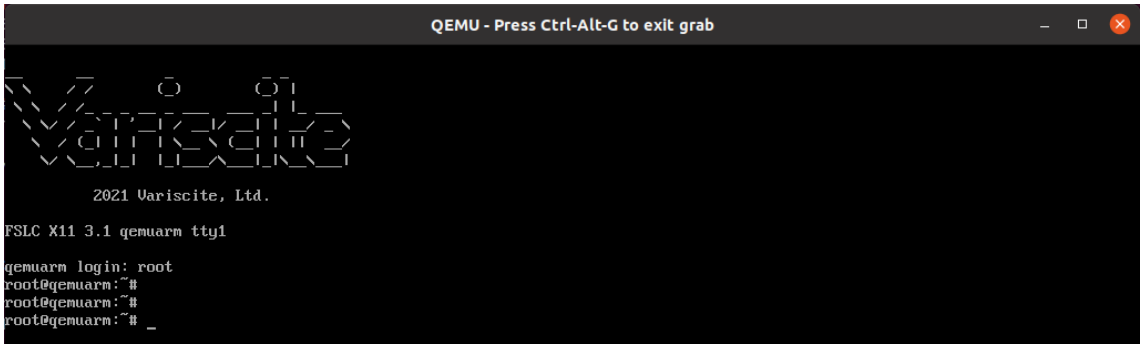


Figura 3.7: Emulador Qemu

En el siguiente directorio se pueden encontrar los objetos que se generan tras la generación de la imagen, tanto para qemuarm como para imx6ul-var-dart y imx6ul-var-dart-app.

```
~/var-fslc-yocto/build_x11/tmp/deploy/images$ ls -l

drwxr-xr-x 2 user user 24576 ene 29 11:44 imx6ul-var-dart
drwxr-xr-x 3 user user 20480 mar  5 15:22 imx6ul-var-dart-app
drwxr-xr-x 2 user user  4096 ene  9 18:45 qemuarm
```

Bajo estos directorios se encuentran todas las piezas de software que forman la distribución Linux: SPL, U-Boot, kernel de Linux, device tree y filesystem.

```
~/var-fslc-yocto/build_x11/tmp/deploy/images/imx6ul-var-dart-app$ ls -l

-rw-r--r-- 2 user user 104588642 ene 28 09:25  app-image-imx6ul-var-dart.rootfs.tar.gz
-rw-r--r-- 2 user user 111906303 ene 28 09:25  app-image-imx6ul-var-dart.rootfs.wic.gz
lrwxrwxrwx 2 user user        36 nov  4 22:40  SPL -> SPL-imx6ul-var-dart-1.0-r0-sd
-rw-r--r-- 2 user user     60416 nov  4 22:40  SPL-imx6ul-var-dart-1.0-r0-sd
lrwxrwxrwx 2 user user        20 nov  4 22:40  u-boot.img -> u-boot-sd-1.0-r0.img
-rw-r--r-- 2 user user    367072 nov  4 22:40  u-boot-sd-1.0-r0.img
lrwxrwxrwx 2 user user        75 feb  2 08:25  zImage -> zImage--5.4.142.bin
-rw-r--r-- 2 user user   8017040 feb  2 08:25  zImage--5.4.142.bin
-rw-r--r-- 2 user user     34459 feb  2 08:25  imx6ull-var-dart-emmc-wifi.dtb
-rw-r--r-- 2 user user     34380 feb  2 08:25  imx6ull-var-dart-emmc-sd-card.dtb
-rw-r--r-- 2 user user     34380 feb  2 08:25 imx6ull-var-dart-emmc-sd-card-external.dtb
-rw-r--r-- 2 user user     34380 feb  2 08:25 imx6ull-var-dart-emmc-sd-card-internal.dtb
```

Al descomprimir el fichero app-image-imx6ul-var-dart.rootfs.wic.gz se obtiene el archivo de imagen que contiene de manera ordenada todas las partes de la distribución.

Capítulo 4

Configuración de la distribución Linux

4.1. Introducción

Las partes fundamentales de una distribución Linux son el SPL, el U-Boot, el kernel de Linux, el device tree y el filesystem. En la siguiente figura se muestra un esquema básico del funcionamiento del sistema.

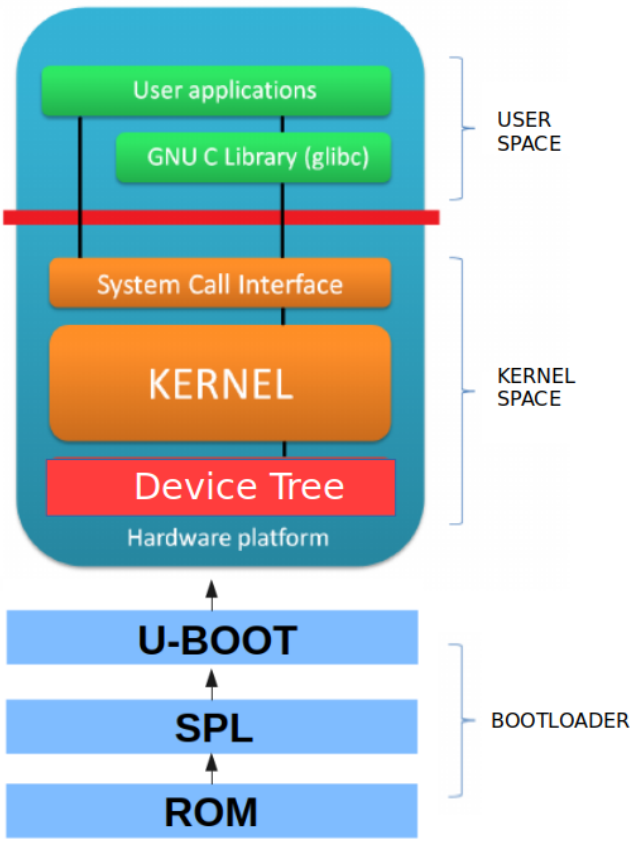


Figura 4.1: Esquema de distribución Linux [10]

En la imagen anterior se pueden distinguir tres partes fundamentales: el bootloader, el espacio de kernel y el espacio de usuario.

- Bootloader: Se divide en tres etapas: ROM, SPL y U-Boot. Se encarga de inicializar las partes fundamentales del procesador y de arrancar el sistema operativo.
- Espacio de kernel: Está formado por el sistema operativo Linux y el device tree.
- Espacio de usuario: En él se monta el filesystem con todos los directorios y ficheros que contiene la distribución.

4.2. ROM

La ROM es un programa de solo lectura que viene implementado de fábrica en el procesador. Inicializa las partes más básicas del mismo, entre ellas la SRAM. Es capaz de ejecutar un número muy reducido de instrucciones, pero resultan suficientes para cargar un binario (el SPL) en la SRAM. Al ser un programa que ya viene grabado en la memoria del procesador, no se considera parte de la distribución Linux.

4.3. SPL

El SPL (Secondary Program Loader) es un pequeño binario generado a partir de U-Boot que, estando cargado en la SRAM, es capaz de inicializar la RAM principal (DDR3L) y cargar el binario del U-Boot en ella. El SPL, a diferencia de la ROM, es un programa configurable por el usuario.

4.4. U-Boot

El U-Boot es un programa más grande y complejo que el SPL que se encarga de cargar en la RAM DDR3L el kernel de Linux y el device tree (y el fichero de RAMDisk en caso de querer montar el filesystem en la memoria RAM). En el U-Boot se pueden configurar parámetros como la dirección de memoria en la que se va a cargar el kernel o el device tree, la partición de memoria donde se encuentra el filesystem o la velocidad de la UART de depuración, entre otros. En Yocto Project se puede ver el código fuente del U-Boot ejecutando el siguiente comando:

```
$ bitbake -c devshell virtual/bootloader
```

Tras lanzar el comando anterior se abrirá una nueva shell en el directorio del código fuente del SPL y del U-Boot.

```
~/var-fslc-yocto/build_x11/tmp/work/imx6ul_var_dart-fslc-linux-gnueabi/u-boot-variscite
    /1.0-r0/git# ls -l
total 464
drwxr-xr-x   2 1000 1000   4096 Nov  4 18:45 api
drwxr-xr-x  15 1000 1000   4096 Nov  4 18:45 arch
drwxr-xr-x 187 1000 1000   4096 Nov  4 18:46 board
-rw-r--r--   1 1000 1000   2297 Nov  4 18:45 CleanSpec.mk
drwxr-xr-x   6 1000 1000   4096 Nov  4 18:46 cmd
drwxr-xr-x   5 1000 1000   4096 Nov  4 18:46 common
-rw-r--r--   1 1000 1000   2260 Nov  4 18:46 config.mk
drwxr-xr-x   2 1000 1000  69632 Nov  4 18:46 configs
drwxr-xr-x   2 1000 1000   4096 Nov  4 18:46 disk
drwxr-xr-x  11 1000 1000  12288 Nov  4 18:46 doc
drwxr-xr-x  53 1000 1000   4096 Nov  4 18:46 drivers
drwxr-xr-x   2 1000 1000   4096 Nov  4 18:46 dts
drwxr-xr-x   2 1000 1000   4096 Nov  4 18:46 env
drwxr-xr-x   4 1000 1000   4096 Nov  4 18:46 examples
drwxr-xr-x  13 1000 1000   4096 Nov  4 18:46 fs
drwxr-xr-x  31 1000 1000  16384 Nov  4 18:46 include
-rw-r--r--   1 1000 1000   1863 Nov  4 18:45 Kbuild
-rw-r--r--   1 1000 1000  15799 Nov  4 18:45 Kconfig
drwxr-xr-x  17 1000 1000   4096 Nov  4 18:46 lib
drwxr-xr-x   2 1000 1000   4096 Nov  4 18:45 Licenses
-rw-r--r--   1 1000 1000  13557 Nov  4 18:45 MAINTAINERS
-rw-r--r--   1 1000 1000  58716 Nov  4 18:45 Makefile
drwxr-xr-x  18 1000 1000   4096 Nov  4 20:37 mx6ul_var_dart_mmc_defconfig
drwxr-xr-x  18 1000 1000   4096 Nov  4 20:36 mx6ul_var_dart_nand_defconfig
drwxr-xr-x   2 1000 1000   4096 Nov  4 18:46 net
drwxr-xr-x   5 1000 1000   4096 Nov  4 18:46 post
-rw-r--r--   1 1000 1000 183354 Nov  4 18:45 README
drwxr-xr-x   6 1000 1000   4096 Nov  4 18:46 scripts
-rw-r--r--   1 1000 1000     17 Nov  4 18:46 snapshot.commit
drwxr-xr-x  11 1000 1000   4096 Nov  4 18:46 test
drwxr-xr-x  14 1000 1000   4096 Nov  4 18:46 tools
```

Para modificar la configuración actual del SPL o del U-Boot hay que entrar en el siguiente directorio:

```
# cd include/configs/
```

En este directorio se encuentran todos los ficheros de cabecera de los procesadores que están incluidos en el proyecto de U-Boot. Los ficheros de cabecera del imx6ull se pueden buscar ejecutando el siguiente comando:

```
# ls -l imx6*

-rw-r--r-- 1 1000 1000 5674 Nov  4 18:46 imx6-engicam.h
-rw-r--r-- 1 1000 1000 5150 Nov  4 18:46 imx6_logic.h
-rw-r--r-- 1 1000 1000 2700 Nov  4 18:46 imx6_spl.h
```

Modificando los ficheros de cabecera anteriores se pueden añadir o quitar funcionalidades del SPL y del U-Boot. También es posible modificar configuraciones del U-Boot y del SPL a través de un menú de configuración ejecutando el siguiente comando:

```
$ bitbake -c menuconfig virtual/bootloader
```

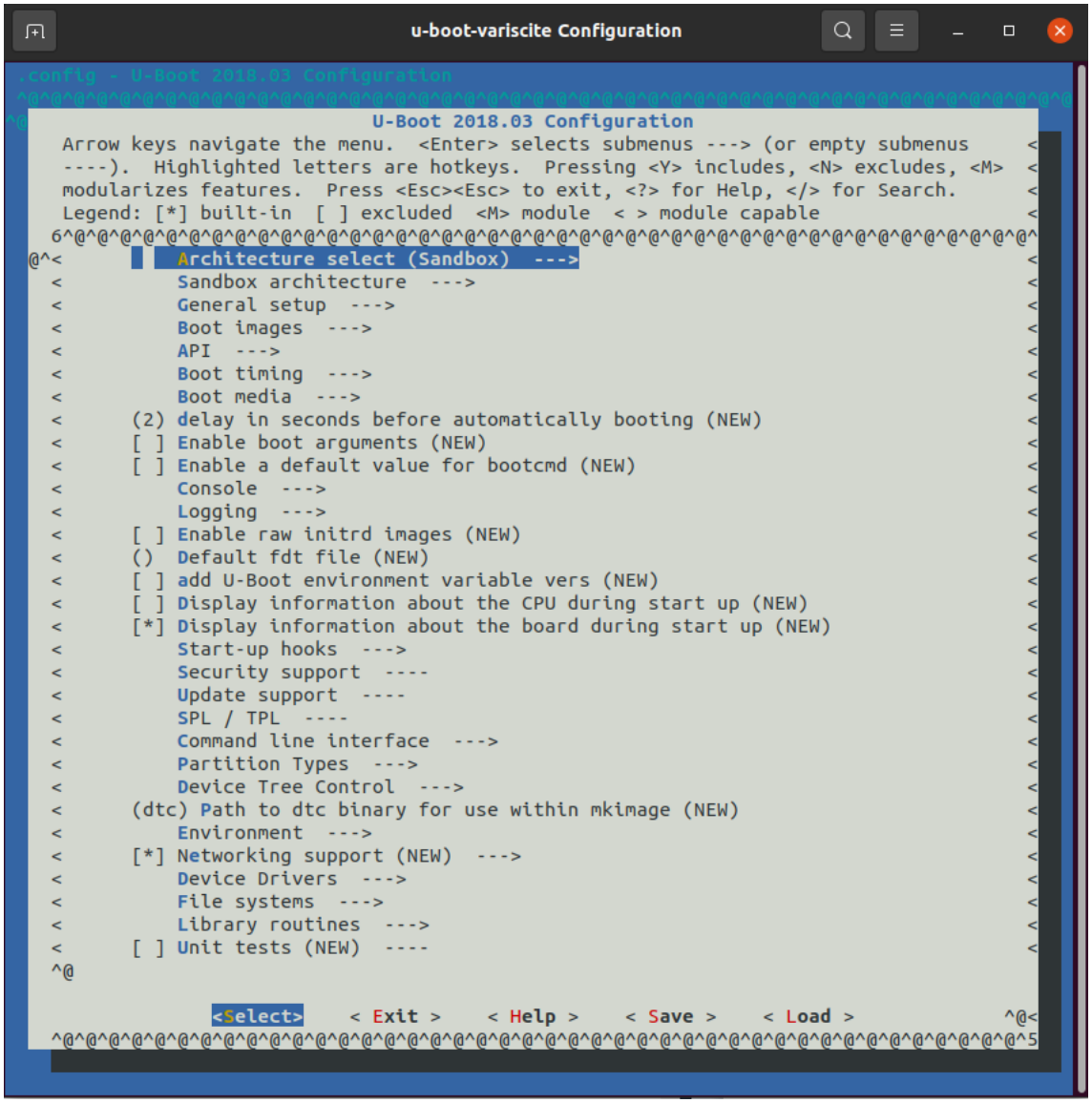


Figura 4.2: Menú de configuración del U-Boot y del SPL

Por defecto, el U-Boot lee un fichero de configuración llamado “uEnv.txt”, en el que se pueden hacer algunas configuraciones como, por ejemplo, el fichero de device tree que debe cargar antes de arrancar. Este fichero se puede modificar con el sistema operativo inicializado y los cambios se aplicarán en el siguiente reinicio.

A pesar de todas las configuraciones posibles, hasta ahora no ha sido necesario realizar ninguna modificación.

4.5. Kernel de Linux

El kernel de Linux es el sistema operativo que se encarga de controlar y gestionar los recursos hardware, además de controlar otras tareas de software como protocolos de bajo nivel, planificación de la ejecución de las tareas, comunicación entre procesos, etc. Fue publicado en internet por primera vez en 1991 por Linus Torvalds cuando todavía era estudiante. Hoy en día es el proyecto Open Source más popular del mundo. En Yocto Project es posible configurar el kernel para añadir o quitar funcionalidades mediante menuconfig:

```
$ bitbake -c menuconfig virtual/kernel
```

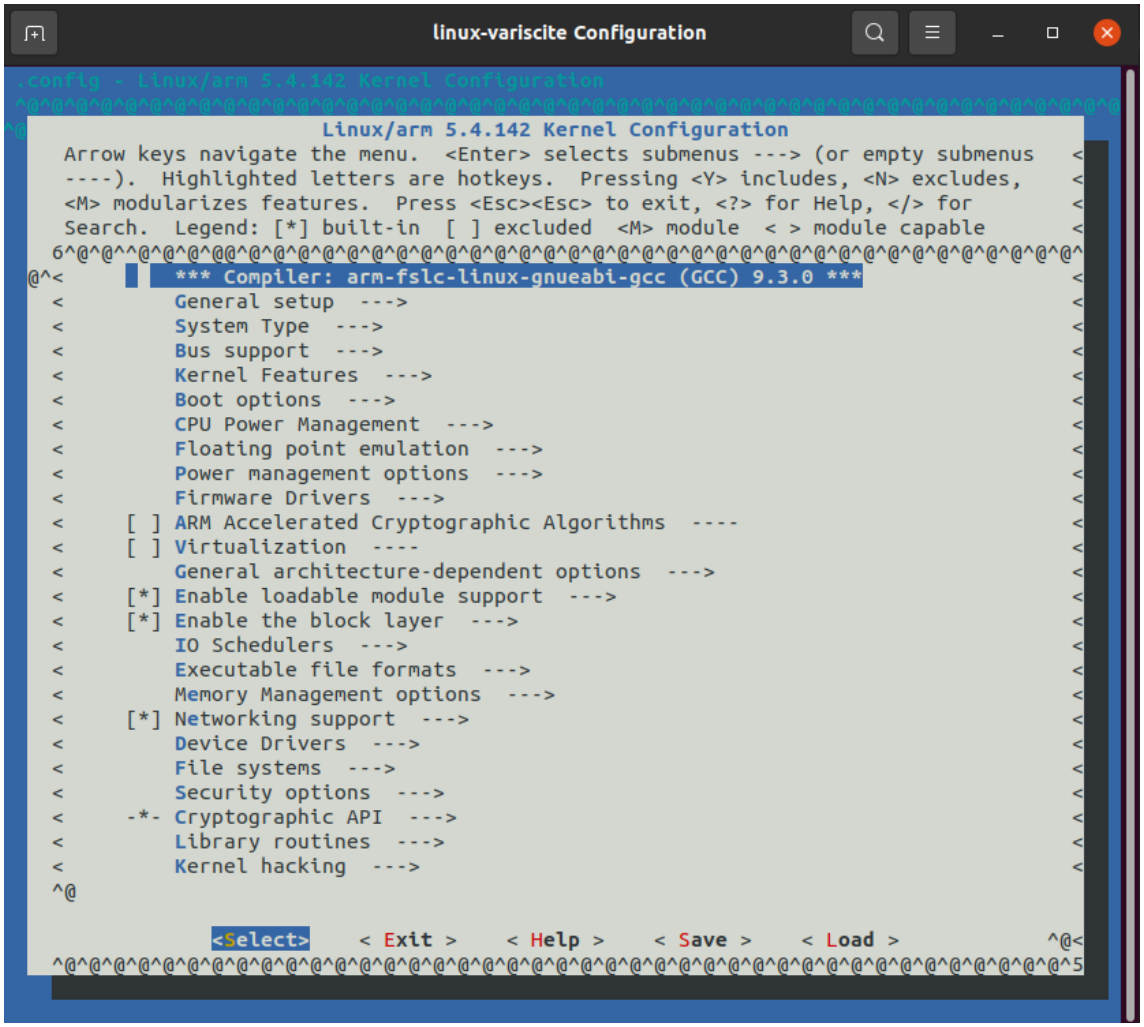


Figura 4.3: Menú de configuración del kernel

La configuración del kernel está dividida en secciones. Cada sección contiene las opciones de una funcionalidad del kernel. Cada una de esas funcionalidades contiene a su vez subsecciones con opciones de configuración más específicas.

En la configuración del kernel se pueden seleccionar los drivers para que se compilen en el mismo binario del kernel seleccionándolos con un asterisco (*) o también se pueden seleccionar para que se compilen como módulos (M). Si se selecciona un driver para que se compile como un módulo, no se añadirá al binario del kernel, sino que se guardará en el filesystem en un fichero con la extensión .ko y se podrá cargar y descargar al kernel en tiempo de ejecución.

En este proyecto se ha partido de la configuración del kernel que trae Variscite por defecto y se ha añadido un driver más: SAE J1939.

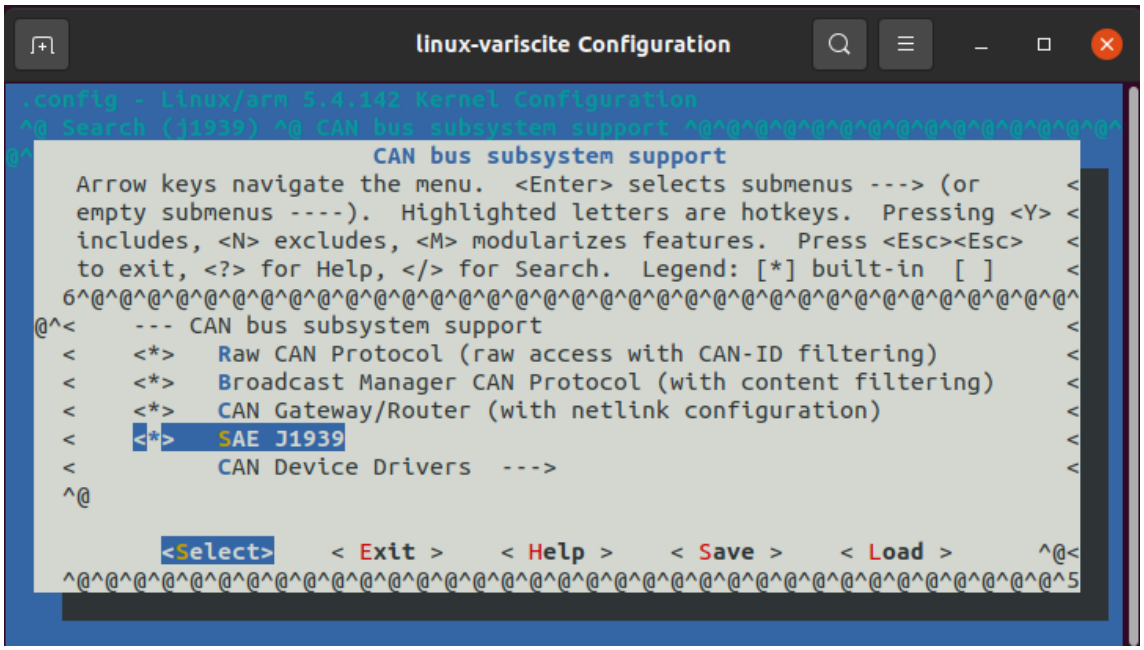


Figura 4.4: Selección del driver SAE J1939

Este driver implementa un protocolo en la capa de transporte de la comunicación CAN que permite enviar y recibir mensajes de más de 8 bytes, lo cual facilitará el intercambio de información entre Picosondas que estén apiladas en forma de tándem, siempre que no se supere el tamaño máximo por mensaje, que son 1785 bytes.

Para guardar esta nueva configuración del kernel hay que ejecutar el siguiente comando:

```
$ bitbake -c savedefconfig virtual/kernel
```

Se generará un nuevo fichero defconfig con la configuración realizada.

```
1 ...
2 CONFIG_CAN=y
3 CONFIG_CAN_J1939=y
4 ...
```

4.6. Device Tree

El device tree se corresponde con uno o más ficheros con la extensión .dts o .dtsi que describen la configuración de los elementos hardware que forman el dispositivo. En él se pueden configurar parámetros tales como la velocidad del bus I2C, GPIOs de entrada y salida, frecuencia de muestreo del ADC o los pines que forman la interfaz de una UART, entre muchos otros.

El device tree puede estar contenido en el mismo fichero binario del kernel o compilarse de forma independiente. En caso de compilarlo de forma independiente, el binario que se genera se llama device tree blob y tiene la extensión .dtb.

El siguiente comando abre una nueva consola en el directorio donde se sitúa el código fuente del kernel de Linux.

```
$ bitbake -c devshell virtual/kernel
```

En este caso, al estar trabajando con una arquitectura ARM, los ficheros de device tree se encuentran en la siguiente ruta del kernel de Linux: arch/arm/boot/dts/. En este proyecto ha sido necesario hacer algunos cambios en la configuración por defecto del SOM:

- Bus CAN: Se ha cambiado la configuración de los pines que venían por defecto configurados para funcionar como señales RTS y CTS de la UART3 y se han configurado para que funcionen como señales CAN_TX y CAN_RX. Para aplicar dicha modificación se ha creado un parche en el fichero correspondiente del device tree. El parche se puede consultar en el Anexo 12.
- Interfaz SDIO: El sistema de soporte consta de dos adaptadores para tarjetas SD. Se han creado dos ficheros de device tree que parten del original para realizar las modificaciones en cada uno de ellos, los cuales hacen referencia a las tarjetas SD interna y externa: En el fichero de device tree original, la señal de interrupción que indica la presencia de tarjeta SD se activa a nivel bajo (0). En el hardware del sistema de soporte esta señal se activa a nivel alto (1). Además, la señal de interrupción es distinta entre ambas tarjetas SD. Consulta en Anexo 13.

Las modificaciones en el código fuente del kernel pueden realizarse con la utilidad “devtool”:

```
$ devtool modify virtual/kernel
```

Una vez realizadas las modificaciones se ejecuta el siguiente comando que genera tanto el parche como el recipe linux-variscite.bbappend, en el cual se indica que se aplique el parche cuando se genere el recipe original linux-variscite.bb.

```
$ devtool update-recipe -a ~/var-fslc-yocto/sources/meta-app linux-variscite
```

4.7. Filesystem

En el filesystem se almacenan todas las aplicaciones de usuario, además de ficheros de configuración, servicios de arranque, librerías, módulos del kernel, scripts... etc. El filesystem puede montarse sobre una memoria flash (ya sea una tarjeta SD, memoria eMMC, memoria NAND...) o sobre la memoria RAM en caso de trabajar con RAMDisk.

Las aplicaciones que hay en el filesystem, salvo los módulos del kernel de Linux, se ejecutan en el espacio de usuario. Son las denominadas aplicaciones "de escritorio", las cuales en momentos determinados de su ejecución hacen peticiones al kernel a través de las llamadas al sistema (system calls). Por ejemplo, una aplicación del espacio de usuario realizará una llamada al sistema cuando necesite guardar un dato en memoria y será el kernel quien diga sí guardará el dato o no, en caso de que la memoria esté llena.

Conocer el árbol de directorios de un sistema Linux es importante para encontrar ficheros de configuración, aplicaciones de usuario o logs del sistema, entre otros [11].

- /: Directorio raiz (root) del que parten el resto de directorios.
- /bin: Binarios que utiliza el sistema operativo (cp, echo, grep, mv, rm, ls...) para tareas administrativas.
- /sbin: Binarios que utiliza el sistema operativo (mkfs, reboot, mount...) para tareas de arranque, restauración del filesystem, etc.
- /boot: Directorio donde se almacena el binario del kernel de Linux, el o los ficheros de device tree y otros ficheros de configuración del gestor de arranque.
- /dev: En este directorio se encuentran los ficheros que representan los bloques de memoria, buses de comunicación, etc.
- /mnt: Es un directorio que se utiliza normalmente como punto de montaje de memorias extraibles, recursos de red compartidos, etc.
- /etc: En él se almacenan los ficheros de configuración del sistema.
- /home: Directorio que almacena ficheros personales de usuarios estándar.
- /lib: Almacena librerías necesarias para los binarios del sistema.
- /proc: En este directorio se almacena toda la información referente a los procesos que hay en ejecución, además de ficheros donde se pueden obtener datos acerca de la CPU o el estado de la memoria entre otros.
- /sys: Contiene ficheros con información acerca del kernel.
- /tmp: Directorio para ficheros que se quieran guardar temporalmente. Al realizar un reinicio del sistema los ficheros que se guarden en este directorio desaparecen.
- /var: En este directorio se almacenan logs, bases de datos, registros del sistema... etc.

- /usr: Almacena ficheros y utilidades de usuario. En /usr/bin se almacenan los ejecutables de usuario, mientras que en /usr/lib se almacenan librerías compartidas de usuario.

Busybox es un binario que contiene gran cantidad de utilidades del espacio de usuario, las cuales se pueden añadir o quitar a través de un menú de configuración.

```
$ bitbake -c menuconfig busybox
```

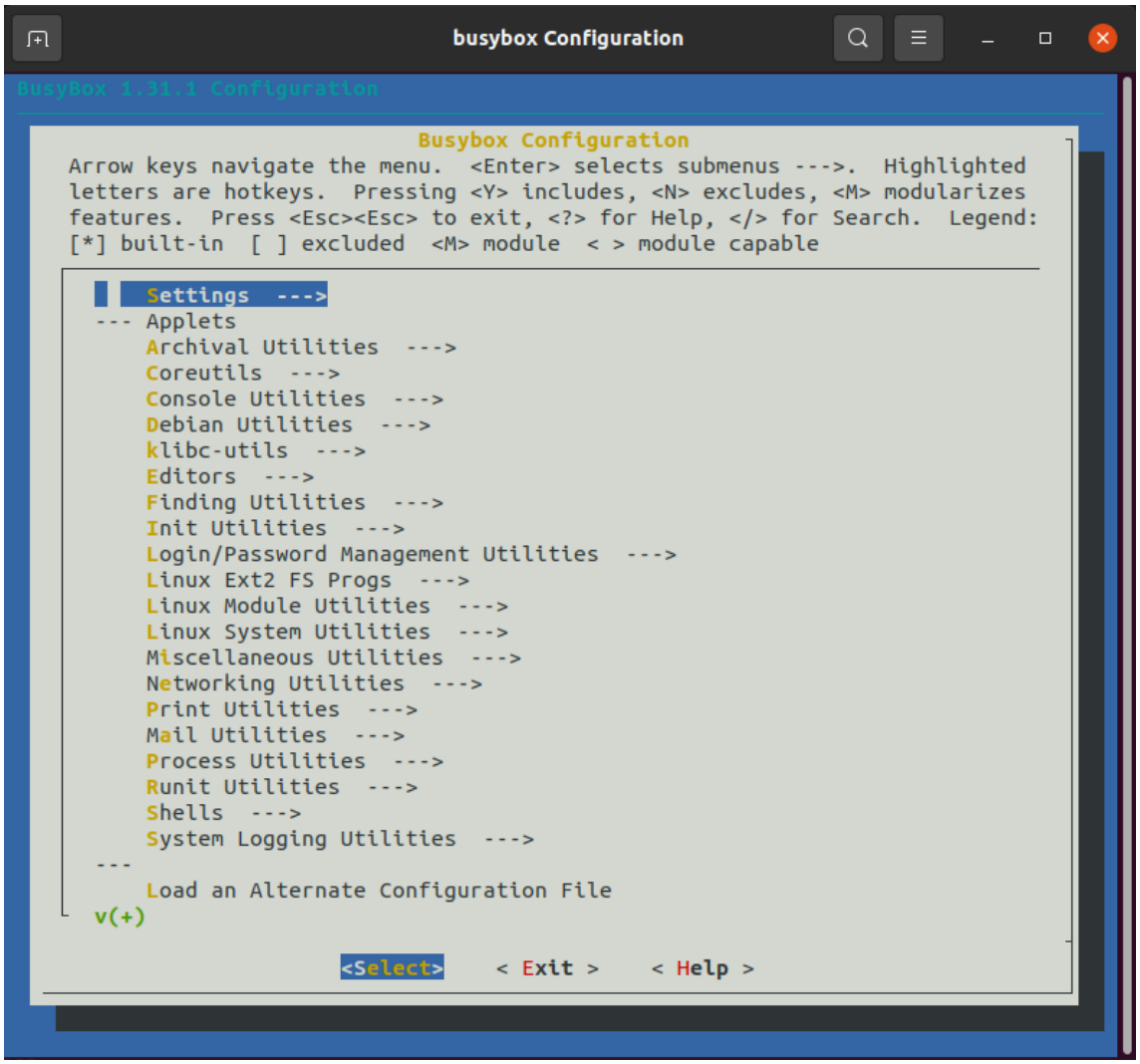


Figura 4.5: Menú de configuración de Busybox

Las demás utilidades y herramientas del espacio de usuario añadidas a la distribución están definidas en el recipe de imagen “app-image.bb” del Anexo 3.

Al filesystem también se le ha añadido el servicio de arranque “userinit-service” que lanza el script “userinit.sh”, el cual inicializa el funcionamiento del sistema de soporte. El script se puede consultar en el Anexo 14.

4.8. Fichero de la imagen Linux

El fichero de la imagen Linux contiene el SPL, el U-Boot, el kernel de Linux, el device tree y el filesystem. Todas estas partes de la distribución de Linux deben situarse en el fichero de imagen de forma lógica y ordenada. En la siguiente figura se muestra cómo se ha distribuido el fichero de imagen:

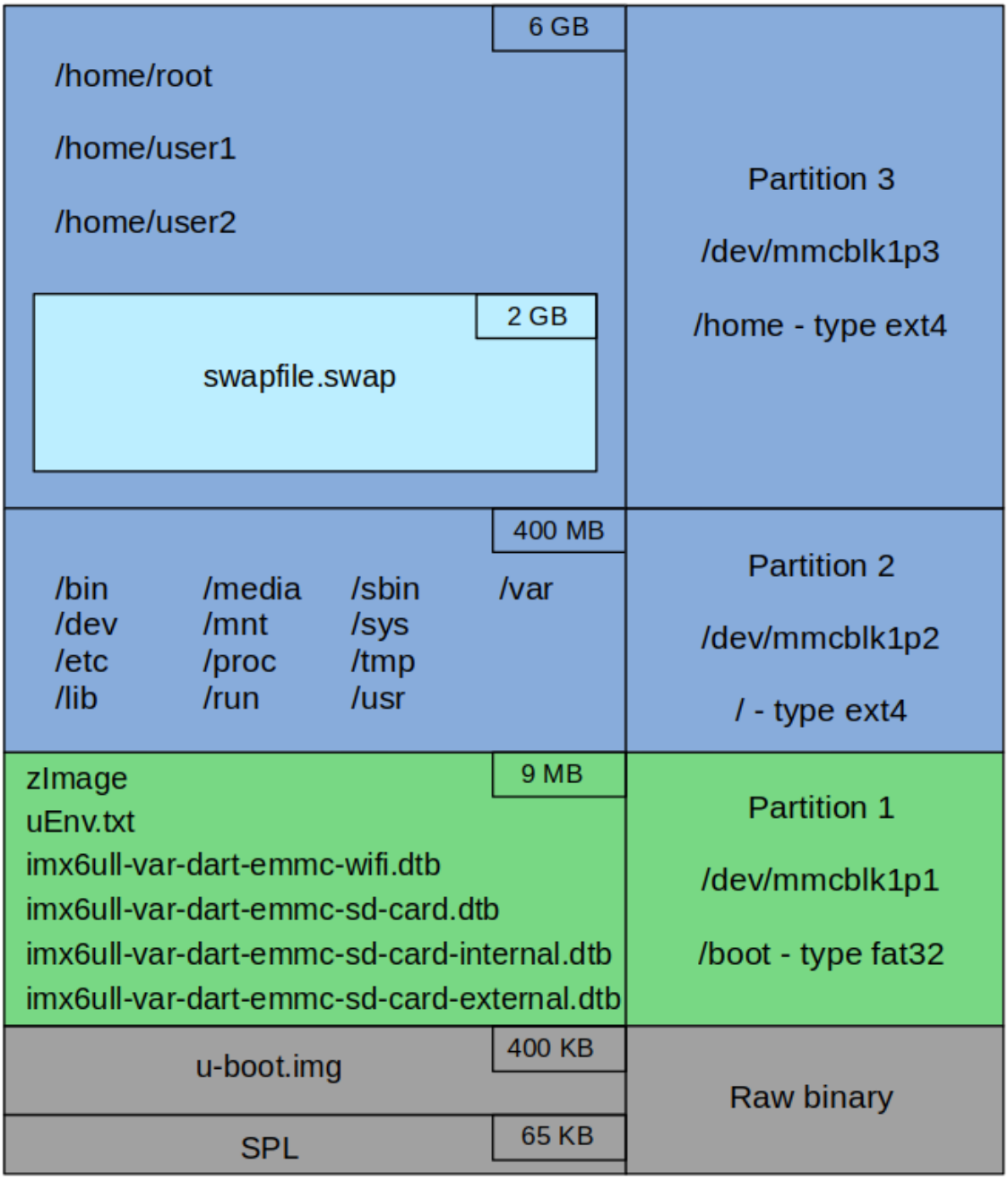


Figura 4.6: Distribución del fichero de imagen

Teniendo en cuenta que el comienzo del fichero de imagen empieza por la parte inferior de la figura anterior, en primer lugar está mapeado el binario del SPL, seguido del binario del U-Boot. Esta primera parte del fichero de imagen no es una partición que sea legible por un sistema operativo, sino que es una zona de memoria donde hay dos ficheros binarios mapeados en bruto.

La partición 1 comienza tras los binarios del bootloader. Está montada en el directorio /boot y formateada en fat32. Contiene los ficheros de device tree, el kernel de Linux y el fichero de configuración del bootloader uEnv.txt.

La partición 2 es de tipo ext4, tiene un tamaño lo más reducido posible y contiene todos los directorios del filesystem, salvo los directorios /boot y /home.

La partición 3, con formato ext4, contiene únicamente el directorio /home y su uso es exclusivamente para almacenar ficheros y utilidades de la aplicación del sistema de soporte. Al generar el fichero de imagen, este fichero tiene el tamaño más reducido posible.

La memoria RAM del SOM es de 256MB, lo cual resulta insuficiente para llevar a cabo algunas actualizaciones del móvil Android. Para solventar este problema se ha creado un fichero swap de 2GB que permite ampliar la capacidad de la memoria RAM a cambio de ceder espacio de la memoria eMMC.

En el arranque del sistema operativo, el script “userinit.sh” comprueba el tamaño de esta partición y la expande si es necesario hasta alcanzar la capacidad máxima que pueda soportar la memoria. Después de esto se crea el fichero swap en ella. Consulta en Anexo 15. De esta forma se consigue generar ficheros de imagen de un tamaño reducido que se expanden cuando ya están grabadas en el dispositivo.

Para arrancar el sistema operativo, es necesario quemar la imagen generada sobre una memoria. En Linux está disponible la utilidad “dd” para quemar imágenes sobre una memoria. Si, por ejemplo, se pretende crear una tarjeta SD booteable y ésta es accesible a través del fichero /dev/sda, habría que ejecutar el siguiente comando para quemar la imagen sobre la tarjeta SD:

```
$ dd if=app-image.img of=/dev/sda
```

Si en lugar de crear una tarjeta SD booteable queremos grabar la imagen en la memoria eMMC del Variscite, accesible desde /dev/mmcblk1, habría que ejecutar el siguiente comando:

```
$ dd if=app-image.img of=/dev/mmcblk1
```

Capítulo 5

Workspace de usuario

5.1. Introducción

El workspace de usuario contiene las aplicaciones, librerías, tests, definiciones de tipos y definiciones de errores creados para este trabajo. El desarrollo se ha realizado con el IDE Eclipse, configurando la compilación cruzada para el microprocesador IMX6ULL en el apartado de Settings de Eclipse.

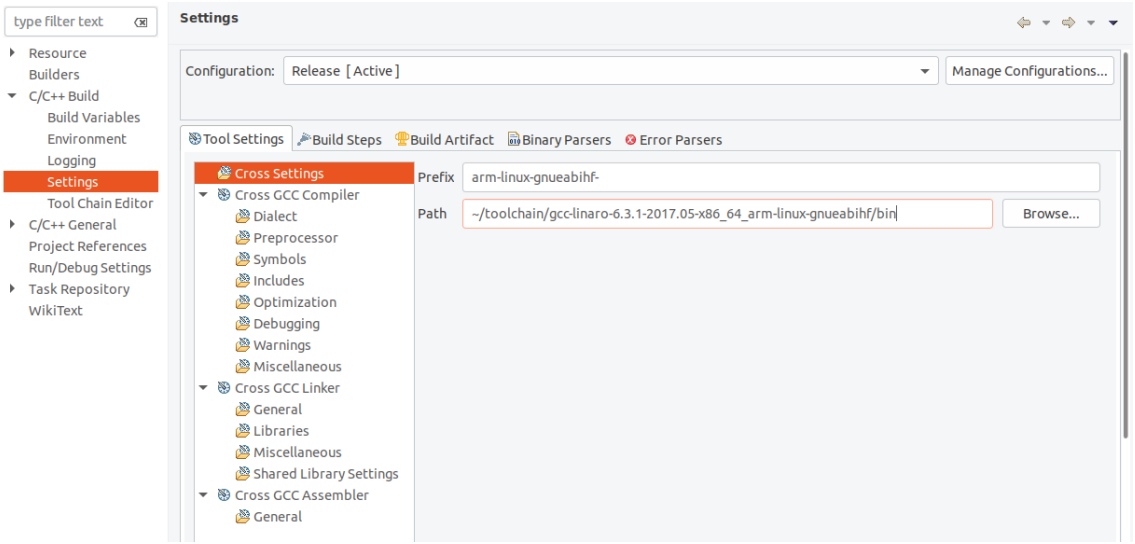


Figura 5.1: Eclipse IDE Settings

Durante el desarrollo de todo el contenido del workspace se ha utilizado el software Filezilla para enviar los binarios generados por Eclipse al sistema de soporte, el cual debe estar conectado a la misma red local que el equipo que realiza el envío desde Filezilla. De esta forma se agiliza el proceso de desarrollo, ya que no es necesario generar una nueva imagen Linux cada vez que se quiera probar una nueva utilidad.

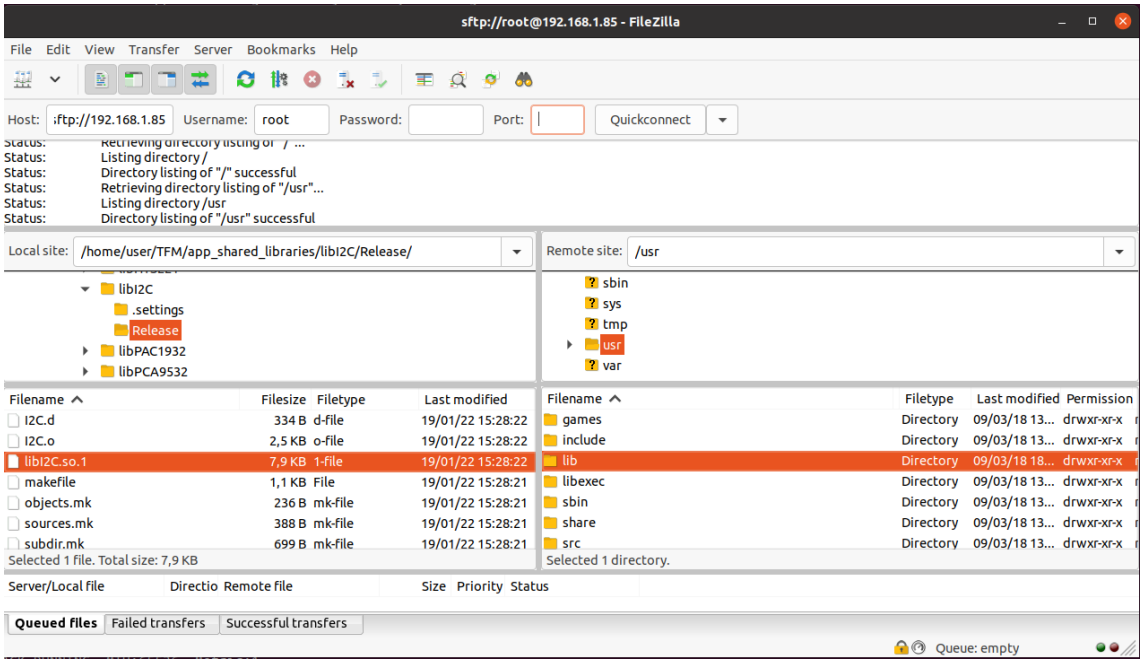


Figura 5.2: Filezilla

El workspace de usuario está organizado en cuatro repositorios que se describirán en las siguientes secciones.

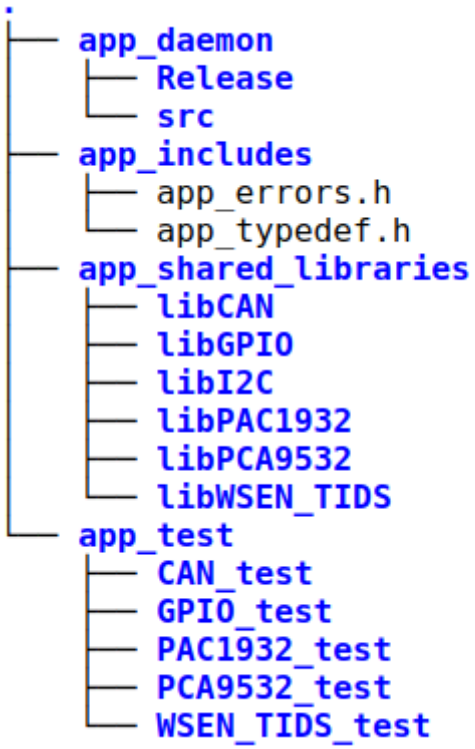


Figura 5.3: Workspace de usuario

5.2. app_includes

El repositorio app_includes contiene las definiciones de tipos de datos y las definiciones de los errores del sistema de soporte en dos ficheros de cabecera:

- app_typedef.h: Contiene las definiciones de los GPIOs que debe controlar el sistema de soporte y otras estructuras de datos. Se puede consultar en el Anexo 16.
- app_errors.h: En este fichero se definen los errores que pueden producirse en el sistema de soporte. Puede consultarse en el Anexo 17. Para reportar los errores en función de su tipo y del número de error que se trata, se ha definido la siguiente macro:

```
1 #define APP_REPORT(type, id_error) ((type << 8) | id_error)
```

De esta forma se reportará un número entero de 16 bits del que se podrá obtener el tipo de error y su identificador. Además, en este fichero también se ha definido el siguiente tipo de dato:

```
1 typedef uint16_t error_type;
```

En caso de que en un futuro existan mas de 256 tipos de errores o identificadores de error, bastaría con cambiar este tipo de dato de uint16_t a uint32_t en esta única línea de código.

5.3. app_shared_libraries

Este repositorio contiene las librerías compartidas del sistema de soporte. Utiliza las definiciones de tipos de datos y errores del repositorio app_includes. Las librerías compartidas son ficheros binarios con la extensión .so que contienen funciones que permiten controlar los sensores y actuadores del sistema de soporte. Implementando esta lógica de control en forma de librerías se consigue un software portable, pues las funciones que se han desarrollado se podrían utilizar en otra aplicación o proyecto que utilice estos mismos componentes. En este repositorio se pueden encontrar las siguientes librerías con todas estas funciones ya implementadas y funcionales:

libI2C.so: Librería de comunicación I2C. Anexo 18.

```
1 error_type (*read_i2c_data)(i2c_device_struct *i2c_dev);
2 error_type (*write_i2c_data)(i2c_device_struct *i2c_dev);
```

libGPIO.so: Librería de control y monitorización del estado de los GPIOs. Anexo 19.

```
1 error_type (*configGPIO)(uint8_t gpio, char* direction);
2 error_type (*freeGPIO)(uint8_t gpio);
3 error_type (*getGPIO_Value)(uint8_t gpio, uint8_t* value);
4 error_type (*getGPIO_Direction)(uint8_t gpio, char* direction);
5 error_type (*setGPIO_Value)(uint8_t gpio, uint8_t value);
```

libWSEN_TIDS.so: Librería de control del sensor de temperatura. En la función WSEN_TIDS_Initialize() se cargan las funciones de la librería libI2C.so. Anexo 20.

```
error_type (*WSEN_TIDS_Initialize)(void);
error_type (*WSEN_TIDS_getTemperature)(float*);
```

libPCA9532.so: Librería de control de Leds. En la función PCA9532_Initialize() se cargan las funciones de las librerías libI2C.so y libGPIO.so. Anexo 21.

```
error_type (*setLED_Value)(uint8_t LEDn, uint8_t color);
error_type (*PCA9532_Initialize)(void);
```

libPAC1932.so: Librería de control del sensor de potencia. En la función PAC1932_Initialize() se cargan las fuciones de las librerías libI2C.so y libGPIO.so. Anexo 22.

```
error_type (*PAC1932_Initialize)(void);
error_type (*PAC1932_GetAllValues)(PAC1932_struct* PAC1932_Value);
```

libCAN.so: Librería de control de la comunicación CAN. Anexo 23.

```
error_type (*CAN_Initialize)(struct can_message * can_data);
error_type (*CAN_Configure)(struct can_message * can_data);
error_type (*CAN_Send)(struct can_message * can_data);
error_type (*CAN_SendFile)(struct can_message * can_data, char * path);
error_type (*CAN_Receive)(struct can_message * can_data);
```

Las funciones de las librerías compartidas interactúan con los drivers del kernel a través de las llamadas al sistema. Para ello, en primer lugar hay que abrir el fichero correspondiente mediante open(). Una vez abierto el fichero es posible intercambiar información con el driver mediante las funciones write(), read(), ioctl(), etc. Tras finalizar la comunicación entre el espacio de usuario y el kernel, hay que cerrar el fichero con la función close().

En lugar de haber implementado toda esta lógica en forma de librerías compartidas, se podrían haber desarrollado drivers que funcionen en el espacio de kernel y acceder a ellos desde las aplicaciones de usuario a través de las llamadas al sistema. También sería una opción válida. Sin embargo, en este caso se ha optado por implementar toda esta lógica en el espacio de usuario, ya que en caso de un fallo crítico el sistema tendría posibilidades de reiniciarse si hay un watchdog monitorizando que la aplicación principal se está ejecutando. Por el contrario, si el fallo se produce en el kernel, el sistema puede quedarse colgado por completo y requeriría de intervención manual para restaurarse.

Los sensores de temperatura, potencia y el controlador de LEDs se comunican con el SOM por el bus I2C. En las funciones de inicialización de estos componentes (WSEN_TIDS_Initialize(), PCA9532_Initialize() y PAC1932_Initialize()) se cargan las funciones de lectura y escritura del bus I2C de la librería libI2C.so y las funciones de configuración de los GPIO de la librería libGPIO.so.

Puesto que estos sensores comparten el mismo bus I2C y la lectura de los mismos se realizará de forma paralela desde distintos hilos de ejecución, se ha declarado un semáforo mutex en la librería libI2C.so para evitar que sucedan condiciones de carrera cuando se trate de acceder a este recurso. Lo mismo ocurre para la librería libGPIO.so, en la cual se ha creado otro semáforo mutex para evitar que los hilos accedan al mismo tiempo a un GPIO.

5.4. app_test

Los tests unitarios comprueban que pequeños módulos de software funcionan correctamente antes de combinarlos en programas más complejos. Con cada aplicación de test se genera un binario ejecutable que carga la librería a testear mediante la función dlopen() y obtiene las direcciones de memoria donde se encuentran las funciones con la función dlsym(). A partir de ahí es capaz de ejecutarlas. Este repositorio contiene los siguientes test unitarios:

- GPIO_test: Aplicación de test para GPIOs. Anexo 24.
- PAC1932_test: Aplicación de test para el sensor de potencia. Anexo 25.
- PCA9532_test: Aplicación de test para el controlador de LEDs. Anexo 26.
- WSEN_TIDS_test: Aplicación de test para el sensor de temperatura. Anexo 27.
- CAN_test: Aplicación de test de envío y recepción CAN. Anexo 28.

5.5. app_daemon

En este directorio solo hay un proyecto, que será la aplicación principal del sistema de soporte, implementada en C++. Este ejecutable carga todas las funciones de las librerías compartidas para tener control total de los elementos hardware del sistema de soporte. Consulta en Anexo 29.

La aplicación app_daemon se encarga de llevar a cabo todas las tareas de monitorización y control del sistema de soporte en paralelo. Para ello, se lanzan cuatro hilos con una máquina de estados en cada uno de ellos.

Hilo de monitorización de temperatura: Realiza lecturas periódicas del valor de temperatura. A continuación se muestran los posibles estados en los que puede encontrarse esta tarea:

- TEMP_IDDLE: Sensor sin inicializar.
- TEMP_INITIALIZE: Inicialización del sensor. LED 1 de color azul.
- TEMP_NORMAL: Temperatura normal. LED 1 de color verde. Ventiladores apagados. Móvil Android encendido.

- TEMP_ALARM_1: Temperatura algo elevada. LED 1 de color amarillo. Se encienden los ventiladores. El móvil Android permanece encendido.
- TEMP_ALARM_2: Temperatura muy elevada. LED 1 de color rojo. Ventiladores encendidos. El móvil Android se apaga.
- TEMP_ERROR: Error de lectura del sensor. LED 1 de color blanco.

Hilo de monitorización de la alimentación. Realiza lecturas periódicas de los canales de alimentación. Posibles estados:

- POWER_IDDLE: Sensor sin inicializar.
- POWER_INITIALIZE: Inicialización del sensor. LED 2 de color azul.
- POWER_READY: Todos los valores de alimentación se encuentran dentro del rango normal. LED 2 de color verde.
- POWER_FAULT_ALL_SOURCES: Todos los valores de alimentación se encuentran fuera de rango. LED 2 de color rojo.
- POWER_FAULT_PERIPHERALS: La alimentación 3V3 de los sensores y/o del SOM se encuentra fuera de rango. LED 2 de color púrpura.
- POWER_FAULT_TERMINAL: La alimentación de 5V del conector USB y/o la alimentación 4V2 de la batería del móvil Android están fuera de rango. LED Amarillo encendido.
- POWER_ERROR_READ: Error de lectura del sensor. LED 2 de color blanco.

Hilo de comunicación CAN: Controla el envío y recepción de mensajes CAN.

- CAN_IDDLE: Bus CAN y sin inicializar.
- CAN_INITIALIZE: Inicialización del bus CAN.
- CAN_CONFIGURE: Configuración del socket de comunicación CAN.
- CAN_LISTEN: Recepción de mensajes CAN.
- CAN_ERROR: Error en el bus CAN.

Hilo de la comunicación con el móvil Android: Se emplea el protocolo de comunicación TCP, donde el sistema de soporte asume el rol de cliente y el móvil Android actúa como servidor y atiende a las peticiones que le hace el sistema de soporte. Para redirigir la comunicación TCP a través de ADB y de un puerto determinado, se ejecutará el siguiente comando:

```
$ adb forward tcp:5555 tcp:5555
```

Los mensajes entrantes y salientes del servidor local 127.0.0.1 del sistema de soporte pasarán por el puerto 5555. Están codificados en formato JSON y parseo de los mismos se ha realizado con ayuda de la clase “nlohmann” [12]. Podrán distinguirse los siguientes tipos de mensaje:

- Tipo 1: Información del móvil Android y de la aplicación de pruebas.

```
{
    "header":{
        "type":1,
        "message_id":131
    },
    "body":{
        "android_dev":{
            "id_sonda":"0001",
            "serial_no":"ABCD1234",
            "model":"ONEPLUS"
        },
        "metrics_app":{
            "interfaces":2,
            "versions":{
                "instrumentation":"10.0.2",
                "metrics":"15.2.3",
                "script":"20.1.1"
            }
        }
    }
}
```

- Tipo 2: Actualización del móvil Android disponible.

```
{
    "header":{
        "type":2,
        "message_id":145,
    },
    "body":{
        "path":"/sdcard/new_android_image.img",
        "md5":"dd79e1f387a3ae1848eebda16815cd5f",
        "version":"9.3.3"
    }
}
```

- Tipo 3: Solicitud de reinicio del móvil Android.

```
{
    "header":{
        "type":3,
        "message_id":153
    }
}
```

- Tipo 4: Solicitud de reinicio del sistema de soporte.

```
{
    "header":{
        "type":4,
        "message_id":177
    }
}
```

La comunicación con el móvil Android puede tener los siguientes estados:

- ADB_IDDLE: Comunicación no inicializada.
- ADB_INITIALIZE: Inicializando comunicación.
- ADB_CONNECTING: Conectando con el móvil Android.
- ADB_ESTABLISHED: Comunicación establecida. LED 3 de color verde.
- ADB_CLOSE_SOCKET: Cierre del socket de comunicación. LED 3 de color blanco.

En este hilo también se monitorizará si el portaSIM está conectado al sistema de soporte. Su estado se reportará con el color del LED 4:

- PortaSIM conectado: LED 4 de color verde.
- PortaSIM desconectado: LED 4 de color blanco.

Capítulo 6

Conclusiones

Un Linux embebido permite al desarrollador interactuar con un sistema operativo complejo que cuenta con muchísimas utilidades y documentación Open Source sin coste alguno. Gracias a esta gran cantidad de recursos, se facilita el trabajo de desarrollo de nuevas aplicaciones y se amplía el rango de posibilidades para nuevas implementaciones.

La versatilidad y modularidad de Linux permite crear sistemas embebidos de todos los grados de complejidad en prácticamente cualquier arquitectura hardware. Además, el proceso de desarrollo de drivers para el kernel de Linux se ha convertido en un estándar de los fabricantes de hardware más importantes del mundo, lo cual permite personalizar el sistema con una gran variedad de componentes electrónicos.

Los conocimientos adquiridos administrando un sistema Linux son aplicables a dispositivos muy comunes en la vida cotidiana de hoy en día, como puede ser un móvil Android, una tablet o una Smart TV, entre otros. Hacer ingeniería inversa de estos dispositivos puede ser interesante para crear otros nuevos.

Yocto Project mantendrá actualizados el build system Openembedded, la distribución de referencia Poky y la herramienta de trabajo Bitbake. Se podrán añadir, modificar o quitar funcionalidades de manera sencilla, con la posibilidad de verificar cada módulo de software de manera independiente mediante tests unitarios.

La información reportada por la Picosonda permitirá a los operadores de redes de telecomunicación a mejorar la experiencia de sus usuarios. Esto supone aumentar la calidad del servicio tanto para dispositivos de telefonía móvil como para sistemas IOT, cada vez mas presentes en entornos industriales y en núcleos urbanos. También ayudará a hacer más eficiente el despliegue de nuevas redes de telecomunicación, como puede ser la de 5G.

Capítulo 7

Propuestas futuras

7.1. Aplicación para test de producción

Aparte de la aplicación principal “app_daemon”, lo mas conveniente sería desarrollar una aplicación para test de producción “app_production” que se encargue de arrancar el sistema de soporte por primera vez, grabar la imagen de la distribución Linux en el SOM y testear que todos los elementos de hardware funcionan correctamente.

La implementación de este software de test utilizaría las mismas librerías compartidas que la aplicación principal “app_daemon”, solo sería necesario crear la parte lógica de la aplicación de producción.

7.2. Migrar a otro SOM o SOC

Personalmente, considero que el SOM Variscite DART-6UL-5G está muy limitado en cuanto a recursos para las funciones que tiene que ejecutar de forma simultánea en el sistema de soporte.

- En primer lugar, este SOM no puede estar funcionando con el Wifi y la tarjeta SD al mismo tiempo. Es necesario realizar un reinicio del sistema operativo para multiplexar entre una configuración u otra.
- El bus de arranque del SOM se configura estrictamente con un switch hardware, en el que se indica arranque desde la tarjeta SD, eMMC (flash) o NAND. No hay posibilidad de cambiar la fuente de arranque vía software, lo cual obliga a que tenga que intervenir personal técnico cada vez que se requiera arrancar el sistema operativo desde otra fuente.
- Solo hay disponible una interfaz para tarjeta SD, de modo que y hay que reiniciar el sistema cada vez que se quiere multiplexar entre una y otra.

7.3. Crear una imagen Recovery

Una imagen de Recovery permitiría al sistema de soporte actualizarse así mismo. Si se genera una imagen basada en RAMDisk en la que, en lugar de montar el filesystem en la memoria eMMC se monta en la memoria RAM, se podría grabar un nuevo fichero de imagen en la memoria eMMC. Esto permitiría realizar actualizaciones de la Picosonda en remoto.

7.4. Integrar el SAI en el sistema de soporte

Actualmente el SAI que utiliza el sistema de soporte es un dispositivo comercial. Sería interesante estimar los costes de integrar esa electrónica en nuestro sistema y hacer una comparativa.

Bibliografía

[1] *Android Debug Bridge (ADB)*. 2021. URL: https://developer.android.com/studio/command-line/adb.

[2] *Variscite Products*. 2021. URL: https://www.variscite.com/product/system-on-module-som/cortex-a7/dart-6ul-freescale-imx-6ul/#general-info.

[3] *Manual temperature sensor WSEN-TIDS*. 2021. URL: https://www.we-online.com/catalog/manual/2521020222501_WSEN-TIDS%202521020222501%20Manual_rev1.1.pdf.

[4] *Datasheet multichannel power monitor PAC1932*. 2021. URL: https://ww1.microchip.com/downloads/en/DeviceDoc/PAC1931-Family-Data-Sheet-DS20005850E.pdf.

[5] *Peak System*. 2021. URL: https://www.peak-system.com/PCAN-USB.199.0.html?&L=1.

[6] *Controlador de LEDs*. 2021. URL: https://www.nxp.com/docs/en/data-sheet/PCA9532.pdf.

[7] *Multiplexor SDIO*. 2021. URL: https://www.mouser.es/datasheet/2/308/1/FSSD06_D-2314271.pdf.

[8] *Yocto Project Overview*. 2021. URL: https://www.yoctoproject.org/software-overview/.

[9] *Mega manual Yocto Project*. 2021. URL: https://www.yoctoproject.org/docs/current/mega-manual/mega-manual.html.

[10] *Research Gate - Linux user and kernel space*. 2021. URL: https://www.researchgate.net/figure/Linux-User-and-Kernel-space_fig1_245022829.

[11] *Explicación sencilla del arbol de directorios de GNU/Linux*. 2021. URL: https://www.linuxadictos.com/explicacion-sencilla-del-arbol-de-directorios-de-gnu-linux.html.

[12] *JSON for Modern C++*. 2021. URL: https://github.com/nlohmann/json.

Parte II

Anexos

Apéndice A

Anexo 1

```
DESCRIPTION = "User Init Service"
SRC_URI = "file://userinit-service"
LICENSE = "MIT"
LIC_FILES_CHKSUM = "file://${COMMON_LICENSE_DIR}/MIT;md5=0835
    ade698e0bcf8506ecda2f7b4f302"

inherit update-rc.d
INITSCRIPT_PACKAGES = "${PN}"
INITSCRIPT_NAME = "userinit-service"

do_install() {
        install -d ${D}${INIT_D_DIR}
        install -m 0755 ${WORKDIR}/userinit-service ${D}${INIT_D_DIR}/
            userinit-service
}

# package it as it is not installed in a standard location*
FILES_${PN} += "${INIT_D_DIR}"
```

Volver

Apéndice B

Anexo 2

```
DESCRIPTION = "User config files"
SECTION = "Configuracion de usuario"
LICENSE = "MIT"
LIC_FILES_CHKSUM = "file://${COMMON_LICENSE_DIR}/MIT;md5=0835
    ade698e0bcf8506ecda2f7b4f302"
SRC_URI = "file://dummy_file.conf "

S = "${WORKDIR}"

do_install() {
        install -d ${D}${sysconfdir}
        install -m 755 dummy_file.conf ${D}${sysconfdir}
}

FILES_${PN} += "${sysconfdir}"
```

Volver

Apéndice C

Anexo 3

```
SUMMARY = "Simple app image"

inherit image

IMAGE_FEATURES += "package-management"
IMAGE_LINGUAS = "en-us"


CORE_OS = "\
    openssh openssh-keygen openssh-sftp-server \
    packagegroup-core-boot \
    tzdata \
    kernel-modules \
"

DEV_SDK = " \
    binutils \
    binutils-symlinks \
    coreutils \
    cpp \
    cpp-symlinks \
    diffutils \
    elfutils elfutils-binutils \
    file \
    gcc \
    gcc-symlinks \
    g++ \
    g++-symlinks \
    gettext \
    git \
    ldd \
    libstdc++ \
    libstdc++-dev \
    libtool \
    ltrace \
    make \
```

```
    perl-modules \
    pkgconfig \
    python3-modules \
    strace \
"

EXTRA_TOOLS = " \
    android-tools \
    can-utils \
    coreutils \
    curl \
    diffutils \
    dosfstools \
    e2fsprogs-mke2fs \
    ethtool \
    findutils \
    i2c-tools \
    ifupdown \
    iperf3 \
    iproute2 \
    iptables \
    lsof \
    ntp ntp-tickadj \
    procps \
    sysfsutils \
    tcpdump \
    util-linux \
    util-linux-blkid \
    unzip \
    wget \
    zip \
"

WIFI = " \
    crda \
    iw \
    bcm43xx-utils \
    bluez5 \
    bluez-alsa \
    brcm-patchram-plus \
    wpa-supplicant \
    wlconf \
    hostapd \
    linux-firmware-bcm4339 \
    linux-firmware-bcm43430 \
"

USER_SOFTWARE = " \
    user-apps \
    user-libs \
    user-tests \
    user-scripts \
"
```

```
90
91  USER_CONFIG = " \
92      config-files \
93      init-services \
94  "
95
96  IMAGE_INSTALL += " \
97      ${CORE_OS} \
98      ${DEV_SDK} \
99      ${EXTRA_TOOLS} \
100     ${WIFI} \
101     ${USER_CONFIG} \
102     ${USER_SOFTWARE} \
103 "
104
105 inherit extrausers
106 EXTRA_USERS_PARAMS = "\
107                       useradd -P passwd1 user1; \
108                       useradd -P passwd2 user2; \
109                       "
110
111 mount_smackfs () {
112
113     cat >> ${IMAGE_ROOTFS}/etc/fstab <<EOF
114
115 /dev/mmcblk1p1       /boot           vfat    defaults        1       2
116 /dev/mmcblk1p2       /               ext4    defaults        1       1
117 /dev/mmcblk1p3       /home           ext4    defaults        0       0
118
119 EOF
120 }
121 ROOTFS_POSTPROCESS_COMMAND += "mount_smackfs; "
122
123 export IMAGE_BASENAME = "app-image"
```

Volver

Apéndice D

Anexo 4

```
FILESEXTRAPATHS_prepend := "${THISDIR}/${PN}:"

SRC_URI += "file://0001-enable-CAN-disable-UART3.patch \
            file://0002-Add-sd-card-external-and-internal-device-tree.
               patch \
            file://defconfig \
            "
```

Volver

Apéndice E

Anexo 5

```
DESCRIPTION = "Apps user"
SECTION = "User Applications"
LICENSE = "MIT"
LIC_FILES_CHKSUM = "file://${COMMON_LICENSE_DIR}/MIT;md5=0835
    ade698e0bcf8506ecda2f7b4f302"

FILESEXTRAPATHS_prepend := "${THISDIR}/${PN}-${PV}:"

SRCREV = "${AUTOREV}"
SRCREV_FORMAT = "none"

SRC_URI = " \
 git://github.com/blpanadero/app_daemon.git;destsuffix=git/app_daemon;
     protocol=https \
 git://github.com/blpanadero/app_includes.git;destsuffix=git/
     app_includes;protocol=https"

S = "${WORKDIR}/git"

FILES_${PN} = "${bindir} "

do_compile() {
 ${CC} ${CFLAGS} ${LDFLAGS} -lpthread -shared ${S}/app_daemon/src/main.
     cpp -o app_daemon/Release/app_daemon
}

do_install() {
 install -d ${D}${bindir}
 install -m 0755 ${S}/app_daemon/Release/app_daemon ${D}${bindir}
}
```

Volver

Apéndice F

Anexo 6

```
DESCRIPTION = "Shared libraries for user app"
SECTION = "Libraries"
LICENSE = "MIT"
LIC_FILES_CHKSUM = "file://${COMMON_LICENSE_DIR}/MIT;md5=0835
    ade698e0bcf8506ecda2f7b4f302"

FILESEXTRAPATHS_prepend := "${THISDIR}/${PN}-${PV}:"

SRCREV = "${AUTOREV}"
SRCREV_FORMAT = "none"

SRC_URI = " \
 git://github.com/blpanadero/app_shared_libraries.git;destsuffix=git/
     app_shared_libraries;protocol=https \
 git://github.com/blpanadero/app_includes.git;destsuffix=git/
     app_includes;protocol=https"

S = "${WORKDIR}/git"
PV = "1"
FILES_${PN} = "${libdir} "

do_compile() {

 ${CC} ${CFLAGS} ${LDFLAGS} -shared -fPIC -Wl,-soname,libGPIO.so.${PV}
     ${S}/app_shared_libraries/libGPIO/GPIO.c -o libGPIO.so.${PV}
 ${CC} ${CFLAGS} ${LDFLAGS} -shared -fPIC -Wl,-soname,libI2C.so.${PV} $
     {S}/app_shared_libraries/libI2C/I2C.c -o libI2C.so.${PV}
 ${CC} ${CFLAGS} ${LDFLAGS} -shared -fPIC -Wl,-soname,libWSEN_TIDS.so.$
     {PV} ${S}/app_shared_libraries/libWSEN_TIDS/WSEN_TIDS.c -o
     libWSEN_TIDS.so.${PV}
 ${CC} ${CFLAGS} ${LDFLAGS} -shared -fPIC -Wl,-soname,libPAC1932.so.${
     PV} ${S}/app_shared_libraries/libPAC1932/PAC1932.c -o libPAC1932.
     so.${PV}
 ${CC} ${CFLAGS} ${LDFLAGS} -shared -fPIC -Wl,-soname,libPCA9532.so.${
     PV} ${S}/app_shared_libraries/libPCA9532/PCA9532.c -o libPCA9532.
     so.${PV}
```

```
26    ${CC} ${CFLAGS} ${LDFLAGS} -shared -fPIC -Wl,-soname,libCAN.so.${PV}
          ${S}/app_shared_libraries/libCAN/CAN.c -o libCAN.so.${PV}
27  }
28
29  do_install() {
30
31   install -d ${D}${libdir}
32   install -m 0755 ${S}/lib*.so* ${D}${libdir}
33  }
```

Volver

Apéndice G

Anexo 7

```
DESCRIPTION = "Scripts"
SECTION = "Scripts de usuario"
LICENSE = "MIT"
LIC_FILES_CHKSUM = "file://${COMMON_LICENSE_DIR}/MIT;md5=0835
    ade698e0bcf8506ecda2f7b4f302"
SRC_URI = "file://userinit.sh "

S = "${WORKDIR}"

do_install() {
        install -d ${D}${sbindir}
        install -m 755 userinit.sh ${D}${sbindir}
}

FILES_${PN} += "${sbindir}"
```

Volver

Apéndice H

Anexo 8

```
DESCRIPTION = "Test user"
SECTION = "Test user applications"
LICENSE = "MIT"
LIC_FILES_CHKSUM = "file://${COMMON_LICENSE_DIR}/MIT;md5=0835
    ade698e0bcf8506ecda2f7b4f302"

FILESEXTRAPATHS_prepend := "${THISDIR}/${PN}-${PV}:"

SRCREV = "${AUTOREV}"
SRCREV_FORMAT = "none"

SRC_URI = " \
 git://github.com/blpanadero/app_shared_libraries.git;destsuffix=git/
    app_shared_libraries;protocol=https \
 git://github.com/blpanadero/app_tests.git;destsuffix=git/app_tests;
    protocol=https \
 git://github.com/blpanadero/app_includes.git;destsuffix=git/
    app_includes;protocol=https "


S = "${WORKDIR}/git"
FILES_${PN} = "${bindir} "


do_compile() {
 ${CC} ${CFLAGS} ${LDFLAGS} app_tests/GPIO_test/main.c -o GPIO_test -Wl
    ,--no-as-needed -ldl
 ${CC} ${CFLAGS} ${LDFLAGS} app_tests/WSEN_TIDS_test/main.c -o
    WSEN_TIDS_test -Wl,--no-as-needed -ldl
 ${CC} ${CFLAGS} ${LDFLAGS} app_tests/PAC1932_test/main.c -o
    PAC1932_test -Wl,--no-as-needed -ldl
 ${CC} ${CFLAGS} ${LDFLAGS} app_tests/PCA9532_test/main.c -o
    PCA9532_test -Wl,--no-as-needed -ldl
 ${CC} ${CFLAGS} ${LDFLAGS} app_tests/CAN_test/main.c -o CAN_test -Wl
    ,--no-as-needed -ldl
}
```

```
28
29 do_install() {
30
31  install -d ${D}${bindir}
32  install -m 0755 ${S}/GPIO_test ${D}${bindir}
33  install -m 0755 ${S}/WSEN_TIDS_test ${D}${bindir}
34  install -m 0755 ${S}/PAC1932_test ${D}${bindir}
35  install -m 0755 ${S}/PCA9532_test ${D}${bindir}
36  install -m 0755 ${S}/CAN_test ${D}${bindir}
37 }
```

Volver

Apéndice I

Anexo 9

```
# local.conf
# We have a conf and classes directory, add to BBPATH
BBPATH .= ":${LAYERDIR}"

# We have recipes-* directories, add to BBFILES
BBFILES += "${LAYERDIR}/recipes-*/*/*.bb \
            ${LAYERDIR}/recipes-*/*/*.bbappend"

BBFILE_COLLECTIONS += "meta-app"
BBFILE_PATTERN_meta-app = "^${LAYERDIR}/"
BBFILE_PRIORITY_meta-app = "30"

LAYERDEPENDS_meta-app = "core"
LAYERSERIES_COMPAT_meta-app = "dunfell"

IMAGE_FSTYPES += "wic"
WKS_FILE = "custom-wic.wks"
```

Volver

Apéndice J

Anexo 10

```
# imx6ul-var-dart-app.conf
require conf/machine/imx6ul-var-dart.conf
MACHINEOVERRIDES_EXTENDER_imx6ul-var-dart-app = "imx6ul-var-dart"

KERNEL_DEVICETREE = "\
        imx6ull-var-dart-6ulcustomboard-emmc-wifi.dtb \
        imx6ull-var-dart-6ulcustomboard-emmc-sd-card.dtb \
        imx6ull-var-dart-6ulcustomboard-emmc-sd-card-external.dtb \
        imx6ull-var-dart-6ulcustomboard-emmc-sd-card-internal.dtb \
        "
```

Volver

Apéndice K

Anexo 11

```
# Image Creator .wks
part SPL --source rawcopy --sourceparams="file=SPL" --ondisk mmcblk --
    no-table --align 1
part u-boot --source rawcopy --sourceparams="file=u-boot.img" --ondisk
    mmcblk --no-table --align 69
# Boot partition
part /boot --source bootimg-partition --ondisk mmcblk --fstype=vfat --
    label boot --active --align 4096 --size 8M --extra-space 0
# Rootfs partition
part / --source rootfs --ondisk mmcblk --fstype=ext4 --label system --
    exclude-path=home/
part /home --source rootfs --rootfs-dir=${IMAGE_ROOTFS}/home --ondisk
    mmcblk --fstype=ext4 --label home
```

Volver

Apéndice L

Anexo 12

```
From 2494b62df638eb2f0ed27fcb1b026bbf1c458d44 Mon Sep 17 00:00:00 2001
From: blpanadero <bloppan@etsid.upv.es>
Date: Wed, 16 Feb 2022 10:18:36 +0100
Subject: [PATCH] enable CAN disable UART3

---
 .../dts/imx6ul-imx6ull-var-dart-6ulcustomboard.dtsi | 13 ++++++++-----
 1 file changed, 8 insertions(+), 5 deletions(-)

diff --git a/arch/arm/boot/dts/imx6ul-imx6ull-var-dart-6ulcustomboard.
   dtsi b/arch/arm/boot/dts/imx6ul-imx6ull-var-dart-6ulcustomboard.
   dtsi
index 965f64c87c56..7de87d414726 100644
--- a/arch/arm/boot/dts/imx6ul-imx6ull-var-dart-6ulcustomboard.dtsi
+++ b/arch/arm/boot/dts/imx6ul-imx6ull-var-dart-6ulcustomboard.dtsi
@@ -148,8 +148,8 @@

        pinctrl_flexcan1: flexcan1grp {
                fsl,pins = <
-                       MX6UL_PAD_LCD_DATA09__FLEXCAN1_RX       0x1b020
-                       MX6UL_PAD_LCD_DATA08__FLEXCAN1_TX       0x1b020
+                       MX6UL_PAD_UART3_RTS_B__FLEXCAN1_RX      0x1b020
+                       MX6UL_PAD_UART3_CTS_B__FLEXCAN1_TX      0x1b020
                >;
        };

@@ -220,7 +220,8 @@
                        MX6UL_PAD_UART1_RX_DATA__UART1_DCE_RX   0x1b0b1
                >;
        };
-
+
+/*     CAN Enabled on UART3 pins
        pinctrl_uart3: uart3grp {
                fsl,pins = <
                        MX6UL_PAD_UART3_TX_DATA__UART3_DCE_TX   0x1b0b1
```

```
@@ -229,7 +230,7 @@
                          MX6UL_PAD_UART3_RTS_B__UART3_DCE_RTS    0x1b0b1
                  >;
         };
-
+*/
         pinctrl_wdog: wdoggrp {
                 fsl,pins = <
                          MX6UL_PAD_GPIO1_IO08__WDOG1_WDOG_B      0x78b0
@@ -287,13 +288,15 @@
 };

 /* ttymxc2 UART */
+
+/*      CAN Enabled on UART3 pins
 &uart3 {
         pinctrl-names = "default";
         pinctrl-0 = <&pinctrl_uart3>;
         fsl,uart-has-rtscts;
         status = "okay";
 };
-
+*/
 &usbotg1 {
         dr_mode = "host";
         status = "okay";
```

Volver

Apéndice M

Anexo 13

```
From e398c6029edf762575207f7da943d737621ce482 Mon Sep 17 00:00:00 2001
From: blpanadero <bloppan@etsid.upv.es>
Date: Wed, 16 Feb 2022 11:29:29 +0100
Subject: [PATCH] Add sd card external and internal device tree

---
 arch/arm/boot/dts/Makefile                      |  2 +
 ...-dart-6ulcustomboard-sd-card-external.dtsi | 71 ++++++++++++++++++++
 ...-dart-6ulcustomboard-sd-card-internal.dtsi | 71 ++++++++++++++++++++
 ...t-6ulcustomboard-emmc-sd-card-external.dts | 22 ++++++
 ...t-6ulcustomboard-emmc-sd-card-internal.dts | 22 ++++++
 5 files changed, 188 insertions(+)
 create mode 100644 arch/arm/boot/dts/imx6ul-imx6ull-var-dart-6
    ulcustomboard-sd-card-external.dtsi
 create mode 100644 arch/arm/boot/dts/imx6ul-imx6ull-var-dart-6
    ulcustomboard-sd-card-internal.dtsi
 create mode 100644 arch/arm/boot/dts/imx6ull-var-dart-6ulcustomboard-
    emmc-sd-card-external.dts
 create mode 100644 arch/arm/boot/dts/imx6ull-var-dart-6ulcustomboard-
    emmc-sd-card-internal.dts

diff --git a/arch/arm/boot/dts/Makefile b/arch/arm/boot/dts/Makefile
index b513d67a9a4c..25ad5e051fd7 100644
--- a/arch/arm/boot/dts/Makefile
+++ b/arch/arm/boot/dts/Makefile
@@ -674,6 +674,8 @@ dtb-$(CONFIG_SOC_IMX6UL) += \
        imx6ull-phytec-segin-ff-rdk-emmc.dtb \
        imx6ull-phytec-segin-lc-rdk-nand.dtb \
        imx6ull-var-dart-6ulcustomboard-emmc-sd-card.dtb \
+       imx6ull-var-dart-6ulcustomboard-emmc-sd-card-internal.dtb \
+       imx6ull-var-dart-6ulcustomboard-emmc-sd-card-external.dtb \
        imx6ull-var-dart-6ulcustomboard-emmc-wifi.dtb \
        imx6ull-var-dart-6ulcustomboard-nand-sd-card.dtb \
        imx6ull-var-dart-6ulcustomboard-nand-wifi.dtb \
diff --git a/arch/arm/boot/dts/imx6ul-imx6ull-var-dart-6ulcustomboard-
   sd-card-external.dtsi b/arch/arm/boot/dts/imx6ul-imx6ull-var-dart-6
```

```
    ulcustomboard-sd-card-external.dtsi
new file mode 100644
index 000000000000..5c1371933784
--- /dev/null
+++ b/arch/arm/boot/dts/imx6ul-imx6ull-var-dart-6ulcustomboard-sd-card-
    external.dtsi
@@ -0,0 +1,71 @@
+/*
+ * Copyright (C) 2015-2019 Variscite Ltd. - https://www.variscite.com
+ *
+ * This program is free software; you can redistribute it and/or
    modify
+ * it under the terms of the GNU General Public License version 2 as
+ * published by the Free Software Foundation.
+ */
+
+/ {
+       regulators {
+               reg_sd1_vmmc: regulator_sd1_vmmc {
+                       compatible = "regulator-fixed";
+                       regulator-name = "VSD_3V3";
+                       regulator-min-microvolt = <3300000>;
+                       regulator-max-microvolt = <3300000>;
+               };
+       };
+};
+
+&iomuxc {
+       pinctrl_usdhc1: usdhc1grp {
+               fsl,pins = <
+                       MX6UL_PAD_SD1_CMD__USDHC1_CMD           0x17059
+                       MX6UL_PAD_SD1_CLK__USDHC1_CLK           0x17059
+                       MX6UL_PAD_SD1_DATA0__USDHC1_DATA0       0x17059
+                       MX6UL_PAD_SD1_DATA1__USDHC1_DATA1       0x17059
+                       MX6UL_PAD_SD1_DATA2__USDHC1_DATA2       0x17059
+                       MX6UL_PAD_SD1_DATA3__USDHC1_DATA3       0x17059
+               >;
+       };
+
+       pinctrl_usdhc1_100mhz: usdhc1grp100mhz {
+               fsl,pins = <
+                       MX6UL_PAD_SD1_CMD__USDHC1_CMD           0x170b9
+                       MX6UL_PAD_SD1_CLK__USDHC1_CLK           0x100b9
+                       MX6UL_PAD_SD1_DATA0__USDHC1_DATA0       0x170b9
+                       MX6UL_PAD_SD1_DATA1__USDHC1_DATA1       0x170b9
+                       MX6UL_PAD_SD1_DATA2__USDHC1_DATA2       0x170b9
+                       MX6UL_PAD_SD1_DATA3__USDHC1_DATA3       0x170b9
+               >;
+       };
+
+       pinctrl_usdhc1_200mhz: usdhc1grp200mhz {
+               fsl,pins = <
+                       MX6UL_PAD_SD1_CMD__USDHC1_CMD           0x170f9
```

```
+                       MX6UL_PAD_SD1_CLK__USDHC1_CLK           0x100f9
+                       MX6UL_PAD_SD1_DATA0__USDHC1_DATA0       0x170f9
+                       MX6UL_PAD_SD1_DATA1__USDHC1_DATA1       0x170f9
+                       MX6UL_PAD_SD1_DATA2__USDHC1_DATA2       0x170f9
+                       MX6UL_PAD_SD1_DATA3__USDHC1_DATA3       0x170f9
+               >;
+       };
+
+       pinctrl_usdhc1_gpio: usdhc1_gpiogrp {
+               fsl,pins = <
+                       MX6UL_PAD_CSI_DATA03__GPIO4_IO24
          0x1b0b1 /* CD */
+               >;
+       };
+};
+
+&usdhc1 {
+       pinctrl-names = "default", "state_100mhz", "state_200mhz";
+       pinctrl-0 = <&pinctrl_usdhc1>, <&pinctrl_usdhc1_gpio>;
+       pinctrl-1 = <&pinctrl_usdhc1_100mhz>, <&pinctrl_usdhc1_gpio>;
+       pinctrl-2 = <&pinctrl_usdhc1_200mhz>, <&pinctrl_usdhc1_gpio>;
+       cd-gpios = <&gpio4 24 GPIO_ACTIVE_HIGH>;
+       no-1-8-v;
+       keep-power-in-suspend;
+       enable-sdio-wakeup;
+       status = "okay";
+};
diff --git a/arch/arm/boot/dts/imx6ul-imx6ull-var-dart-6ulcustomboard-
   sd-card-internal.dtsi b/arch/arm/boot/dts/imx6ul-imx6ull-var-dart-6
   ulcustomboard-sd-card-internal.dtsi
new file mode 100644
index 000000000000..2a2c5ddd8557
--- /dev/null
+++ b/arch/arm/boot/dts/imx6ul-imx6ull-var-dart-6ulcustomboard-sd-card-
   internal.dtsi
@@ -0,0 +1,71 @@
+/*
+ * Copyright (C) 2015-2019 Variscite Ltd. - https://www.variscite.com
+ *
+ * This program is free software; you can redistribute it and/or
   modify
+ * it under the terms of the GNU General Public License version 2 as
+ * published by the Free Software Foundation.
+ */
+
+/ {
+       regulators {
+               reg_sd1_vmmc: regulator_sd1_vmmc {
+                       compatible = "regulator-fixed";
+                       regulator-name = "VSD_3V3";
+                       regulator-min-microvolt = <3300000>;
+                       regulator-max-microvolt = <3300000>;
+               };
```

```
130 +        };
131 +};
132 +
133 +&iomuxc {
134 +        pinctrl_usdhc1: usdhc1grp {
135 +                fsl,pins = <
136 +                        MX6UL_PAD_SD1_CMD__USDHC1_CMD           0x17059
137 +                        MX6UL_PAD_SD1_CLK__USDHC1_CLK           0x17059
138 +                        MX6UL_PAD_SD1_DATA0__USDHC1_DATA0       0x17059
139 +                        MX6UL_PAD_SD1_DATA1__USDHC1_DATA1       0x17059
140 +                        MX6UL_PAD_SD1_DATA2__USDHC1_DATA2       0x17059
141 +                        MX6UL_PAD_SD1_DATA3__USDHC1_DATA3       0x17059
142 +                >;
143 +        };
144 +
145 +        pinctrl_usdhc1_100mhz: usdhc1grp100mhz {
146 +                fsl,pins = <
147 +                        MX6UL_PAD_SD1_CMD__USDHC1_CMD           0x170b9
148 +                        MX6UL_PAD_SD1_CLK__USDHC1_CLK           0x100b9
149 +                        MX6UL_PAD_SD1_DATA0__USDHC1_DATA0       0x170b9
150 +                        MX6UL_PAD_SD1_DATA1__USDHC1_DATA1       0x170b9
151 +                        MX6UL_PAD_SD1_DATA2__USDHC1_DATA2       0x170b9
152 +                        MX6UL_PAD_SD1_DATA3__USDHC1_DATA3       0x170b9
153 +                >;
154 +        };
155 +
156 +        pinctrl_usdhc1_200mhz: usdhc1grp200mhz {
157 +                fsl,pins = <
158 +                        MX6UL_PAD_SD1_CMD__USDHC1_CMD           0x170f9
159 +                        MX6UL_PAD_SD1_CLK__USDHC1_CLK           0x100f9
160 +                        MX6UL_PAD_SD1_DATA0__USDHC1_DATA0       0x170f9
161 +                        MX6UL_PAD_SD1_DATA1__USDHC1_DATA1       0x170f9
162 +                        MX6UL_PAD_SD1_DATA2__USDHC1_DATA2       0x170f9
163 +                        MX6UL_PAD_SD1_DATA3__USDHC1_DATA3       0x170f9
164 +                >;
165 +        };
166 +
167 +        pinctrl_usdhc1_gpio: usdhc1_gpiogrp {
168 +                fsl,pins = <
169 +                        MX6UL_PAD_CSI_VSYNC__GPIO4_IO19         0x1b0b1
    /* CD */
170 +                >;
171 +        };
172 +};
173 +
174 +&usdhc1 {
175 +        pinctrl-names = "default", "state_100mhz", "state_200mhz";
176 +        pinctrl-0 = <&pinctrl_usdhc1>, <&pinctrl_usdhc1_gpio>;
177 +        pinctrl-1 = <&pinctrl_usdhc1_100mhz>, <&pinctrl_usdhc1_gpio>;
178 +        pinctrl-2 = <&pinctrl_usdhc1_200mhz>, <&pinctrl_usdhc1_gpio>;
179 +        cd-gpios = <&gpio4 19 GPIO_ACTIVE_HIGH>;
180 +        no-1-8-v;
181 +        keep-power-in-suspend;
```

```
+       enable-sdio-wakeup;
+       status = "okay";
+};
diff --git a/arch/arm/boot/dts/imx6ull-var-dart-6ulcustomboard-emmc-sd-card-external.dts b/arch/arm/boot/dts/imx6ull-var-dart-6ulcustomboard-emmc-sd-card-external.dts
new file mode 100644
index 000000000000..07923684b4fd
--- /dev/null
+++ b/arch/arm/boot/dts/imx6ull-var-dart-6ulcustomboard-emmc-sd-card-external.dts
@@ -0,0 +1,22 @@
+/*
+ * Copyright (C) 2016-2019 Variscite Ltd. - https://www.variscite.com
+ *
+ * This program is free software; you can redistribute it and/or modify
+ * it under the terms of the GNU General Public License version 2 as
+ * published by the Free Software Foundation.
+ */
+
+/dts-v1/;
+
+#include "imx6ull.dtsi"
+
+#include "imx6ull-var-dart.dtsi"
+#include "imx6ul-imx6ull-var-dart-emmc.dtsi"
+
+#include "imx6ull-var-dart-6ulcustomboard.dtsi"
+#include "imx6ul-imx6ull-var-dart-6ulcustomboard-sd-card-external.dtsi"
+
+/ {
+       model = "Variscite DART-6UL with i.MX6ULL, eMMC & SD card support on VAR-6ULCustomBoard";
+       compatible = "variscite,imx6ul-var-dart", "fsl,imx6ull";
+};
diff --git a/arch/arm/boot/dts/imx6ull-var-dart-6ulcustomboard-emmc-sd-card-internal.dts b/arch/arm/boot/dts/imx6ull-var-dart-6ulcustomboard-emmc-sd-card-internal.dts
new file mode 100644
index 000000000000..165945c7c1d4
--- /dev/null
+++ b/arch/arm/boot/dts/imx6ull-var-dart-6ulcustomboard-emmc-sd-card-internal.dts
@@ -0,0 +1,22 @@
+/*
+ * Copyright (C) 2016-2019 Variscite Ltd. - https://www.variscite.com
+ *
+ * This program is free software; you can redistribute it and/or modify
+ * it under the terms of the GNU General Public License version 2 as
+ * published by the Free Software Foundation.
```

```
225 + */
226 +
227 +/dts-v1/;
228 +
229 +#include "imx6ull.dtsi"
230 +
231 +#include "imx6ull-var-dart.dtsi"
232 +#include "imx6ul-imx6ull-var-dart-emmc.dtsi"
233 +
234 +#include "imx6ull-var-dart-6ulcustomboard.dtsi"
235 +#include "imx6ul-imx6ull-var-dart-6ulcustomboard-sd-card-internal.dtsi
    "
236 +
237 +/ {
238 +       model = "Variscite DART-6UL with i.MX6ULL, eMMC & SD card
    support on VAR-6ULCustomBoard";
239 +       compatible = "variscite,imx6ul-var-dart", "fsl,imx6ull";
240 +};
```

Volver

Apéndice N

Anexo 14

```
#!/bin/bash

#Configura la interfaz ethernet
ifconfig eth1 192.168.1.85 up
route add -net 192.168.1.0 netmask 255.255.255.0 dev eth1


#Comprueba si existe el fichero de configuracion del U-Boot (uEnv.txt)
if ! [ -f /boot/uEnv.txt ]; then

        echo "uEnv.txt not found"
        #Si no existe, posiblemente sea el primer arranque desde que se
            ha grabado la imagen en la memoria

        CURRENTSIZEB=`fdisk -l /dev/mmcblk1p3 | grep "Disk /dev/
           mmcblk1p3" | cut -d' ' -f5`
        CURRENTSIZE=`expr $CURRENTSIZEB / 1024 / 1024`

        #Comprueba si el tamaño de la particion 3 es menor que 20MB
        if [ "$CURRENTSIZE" < 20 ]; then

            #Obtiene tamaño maximo que puede tener
            MAXSIZEMB=`printf %s\\n 'unit MB print list' | parted |
               grep "Disk /dev/mmcblk1p3" | cut -d' ' -f3 | tr -d MB`

            #Redimensiona la particion /dev/mmcblk1p3
            parted /dev/mmcblk1 resizepart p3 $MAXSIZEMB

            #Crea el fichero de SWAP
            create_swap_file.sh
        fi


        #Crea el fichero uEnv.txt
        touch /boot/uEnv.txt

```

```
34                  if grep "SD" /sys/devices/soc0/machine; then
35                          echo "fdt_file = external"
36                          echo fdt_file=imx6ull-var-dart-6ulcustomboard-
                               emmc-sd-card-external.dtb > /boot/uEnv.txt
37                  else
38                          echo "fdt_file = wifi"
39                          echo fdt_file=imx6ull-var-dart-6ulcustomboard-
                               emmc-wifi.dtb > /boot/uEnv.txt
40                  fi
41  fi
42
43
44  if grep "external" /boot/uEnv.txt; then
45
46          #Multiplexa a la tarjeta SD externa
47          echo 119 > /sys/class/gpio/export
48          echo out > /sys/class/gpio/gpio119/direction
49          echo 0 > /sys/class/gpio/gpio119/value
50
51          echo 118 > /sys/class/gpio/export
52          echo out > /sys/class/gpio/gpio118/direction
53          echo 1 > /sys/class/gpio/gpio118/value
54
55  elif grep "internal" /boot/uEnv.txt; then
56
57          #Multiplexa a la tarjeta SD interna
58          echo 119 > /sys/class/gpio/export
59          echo out > /sys/class/gpio/gpio119/direction
60          echo 0 > /sys/class/gpio/gpio119/value
61
62          echo 118 > /sys/class/gpio/export
63          echo out > /sys/class/gpio/gpio118/direction
64          echo 0 > /sys/class/gpio/gpio118/value
65  else
66          #Crea el punto de acceso Wifi
67          echo 1 > /proc/sys/net/ipv4/ip_forward
68          ifconfig wlan0 192.168.5.1
69          hostapd -B /etc/hostapd.conf -P /var/run/hostapd.pid
70          udhcpd /etc/udhcpd.conf
71          iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
72  fi
73
74  #Lanza la aplicacion principal del sistema de soporte
75  app_daemon &
```

Volver

Apéndice Ñ

Anexo 15

```
#!/bin/bash
# Script fichero SWAP

#Crea un fichero de 2GB
dd if=/dev/zero of=/home/swapfile.swap bs=1M count=2K

#Configura los permisos del fichero
chmod 600 /home/swapfile.swap

#Formatea el fichero como swap
mkswap /home/swapfile.swap

#Activa el fichero para que funcione como swap
swapon /home/swapfile.swap
```

Volver

Apéndice O

Anexo 16

```
/*
 * app_typedef.h
 *
 *  Created on: Jul 26, 2021
 *      Author: bernardo
 */
#ifndef APP_TYPEDEF_H_
#define APP_TYPEDEF_H_

#include "app_errors.h"
#include <stdint.h>

#define GPIO_NUMBER(bank, index) ((((bank)-1)*32)+((index)&31))

// GPIO Definition
#define SIM_CTRL GPIO_NUMBER(5, 7)      // Multiplexor de tarjeta SIM
#define PORTASIM_PRES GPIO_NUMBER(2, 4) // Detector del PortaSIM
#define EN_5V_USB_MOB GPIO_NUMBER(5, 9) // Conector USB 5V del movil
#define EN_4V2 GPIO_NUMBER(1, 4) // Conector 4V2 bateria del movil
#define PWDN_CURR_SENSOR GPIO_NUMBER(5, 1)  // Enable PAC1932
#define ALERT_CURR_SENSOR GPIO_NUMBER(5, 8)     // Alert PAC1932
#define TEMP_DRDY GPIO_NUMBER(1, 0)     // Temp data ready
#define LED_RST GPIO_NUMBER(1, 2) // Enable PCA9532
#define FANOUT_1 GPIO_NUMBER(1, 5)      // Ventilador 1
#define FANOUT_2 GPIO_NUMBER(1, 9)      // Ventilador 2
/*
 * @brief       Enum de canales de medida de valores electricos
 */
typedef enum {

        SOM,
        Peripherals,
        Terminal_Battery,
        USB_Connector

}Channels_Enum;
```

```
37 /*
38  * @brief       Estructura que almacena valores electricos de un canal
39  */
40 typedef struct {
41
42         float   Intensity; // Valor de intensidad en miliamperios (mA)
43         float   Voltage; // Valor de voltaje en voltios (V)
44         float   Power; // Valor de potencia consumida en watios (W)
45
46 }Electrial_Values_struct;
47 /*
48  * @brief       Estructura que almacena los datos del sensor PAC1932
49  */
50 typedef struct {
51
52         Electrial_Values_struct SOM;
53         Electrial_Values_struct Peripherals;
54         Electrial_Values_struct Terminal_Battery;
55         Electrial_Values_struct USB_Connector;
56
57         uint32_t        Acc_Count;      // Numero de acumulaciones
               realizadas en los registros VPOWERn_ACC
58
59 }PAC1932_struct;
60 /*
61  * @brief Estructura que almacena la informacion de un mensaje CAN
62  */
63 struct can_message {
64
65         struct can_frame frame;
66         struct sockaddr_can addr;
67         struct ifreq ifr;
68         int socket;
69
70 };
71 #endif /* APP_TYPEDEF_H_ */
```

Volver

Apéndice P

Anexo 17

```
/*
 * app_errors.h
 *
 *  Created on: 22 nov. 2021
 *      Author: bernardo
 */

#ifndef APP_ERRORS_H_
#define APP_ERRORS_H_

#include "app_typedef.h"
#include <stdint.h>

typedef uint16_t error_type     ;

#define APP_REPORT(type, id_error)      ((type << 8) | id_error)

/*
 * Type definition (type error)
 */
#define         NO_ERROR                0
#define         I2C                     1
#define         HTS221                  2
#define         GPIO                    3
#define         PAC1932                 4
#define         PCA9532                 5
#define         CAN                     6
#define         WSEN_TIDS               7
/*
 * Identifier definition        (id_error)
 */
#define         OPENING_EXPORT_FILE                     1
#define         WRITING_EXPORT_FILE                     2
#define         OPENING_DIRECTION_FILE                  3
#define         WRITING_DIRECTION_FILE                  4
#define         INVALID_OUTPUT_VALUE                    5
```

```
37 #define          OPENING_UNEXPORT_FILE                    6
38 #define          WRITING_UNEXPORT_FILE                    7
39 #define          OPENING_VALUE_FILE                       8
40 #define          INVALID_DIRECTION_VALUE                  9
41 #define          WRITING_VALUE_FILE                       10
42 #define          READING_VALUE_FILE                       11
43 #define          READING_DIRECTION_FILE                   12
44 #define          OPENING_SOCKET                           13
45 #define          WRITING_IOCTL_FILE                       14
46 #define          BIND_CALL                                15
47 #define          SEND_FRAME                               16
48 #define          RECEIVE_FRAME                            17
49 #define          LOADING_I2C_LIBRARY                      18
50 #define          READING_CALIBRATION_VALUES               19
51 #define          READING_HUMIDITY                         20
52 #define          READING_TEMPERATURE                      21
53 #define          OPENING_I2C_PORT                         22
54 #define          LOADING_GPIO_LIBRARY                     23
55 #define          READING_PAC1932_VALUES                   24
56 #define          INCORRECT_COLOR                          25
57 #define          SETTING_LED_COLOR                        26
58 #define          SEND_RESET_COMMAND                       27
59 #define          SEND_CTRL_COMMAND                        28
60
61 #endif /* APP_ERRORS_H_ */
```

Volver

Apéndice Q

Anexo 18

```
/*
 * i2c.c
 *
 *  Created on: Jun 30, 2021
 *      Author: bernardo
 */

#include "I2C.h"

static pthread_mutex_t i2c_mutex = PTHREAD_MUTEX_INITIALIZER;
/*
 * @brief       Funcion que lee datos a traves del puerto I2C
 * @param       i2c_dev         Puntero a la estructura de datos I2C
 * @return      errorCode       Codigo de error al ejecutar la funcion
 */
error_type read_i2c_data(i2c_device_struct *i2c_dev)
{
        struct i2c_rdwr_ioctl_data i2c_packet;
        struct i2c_msg message[2];
        uint32_t i2c_file;
        uint8_t outbuf[1];
        error_type errorCode = NO_ERROR;

        // Bloqueo mutex
        pthread_mutex_lock (&i2c_mutex);

        i2c_file = open(i2c_dev->i2c_port, O_RDWR);

        if(i2c_file > 0){

                /*
                 * Antes de leer un registro, se debe indicar mediante
                    una operacion de escritura el registro que se
                    quiere leer:
                 */

```

```
35                  outbuf[0] = i2c_dev->reg_addr;
36
37                  message[0].addr         = i2c_dev->dev_addr;
38                  message[0].flags        = 0;    // 0 -> Write
39                  message[0].len          = 1;    // 1 byte
40                  message[0].buf          = outbuf;
41
42                  /* Ahora se configura la estructura de lectura: */
43
44                  message[1].addr         = i2c_dev->dev_addr;
45                  message[1].flags        = 1;    // 1 -> Read
46                  message[1].len          = i2c_dev->length;
47                  message[1].buf          = i2c_dev->data;
48
49                  i2c_packet.msgs         = message;
50                  i2c_packet.nmsgs        = 2;
51
52                  if(ioctl(i2c_file, I2C_RDWR, &i2c_packet) < 0){
53
54  #ifdef DEBUG
55                  printf("[ERROR]\t[I2C]\t\tioctl: %s\n", strerror(errno)
                        );
56  #endif
57                  errorCode = APP_REPORT(I2C, WRITING_IOCTL_FILE);
58                  }
59
60          }else{
61
62  #ifdef DEBUG
63                  printf("[ERROR]\t[I2C]\t\tOpen I2C port file: %s \n",
                        strerror(errno));
64  #endif
65                  errorCode = APP_REPORT(I2C, OPENING_I2C_PORT);
66          }
67
68          close(i2c_file);
69
70          // Desbloqueo mutex
71          pthread_mutex_unlock (&i2c_mutex);
72
73          return errorCode;
74  }
75
76
77
78
79
80  /*
81   * @brief       Funcion que escribe datos a traves del puerto I2C
82   * @param       i2c_dev         Puntero a la estructura de datos I2C
83   * @return      errorCode       Codigo de error al ejecutar la funcion
84   */
```

```
85  error_type write_i2c_data(i2c_device_struct *i2c_dev)
86  {
87          struct i2c_rdwr_ioctl_data i2c_packet;
88          struct i2c_msg message[1];
89          uint32_t i2c_file;
90          uint8_t outbuf[128] = {0};
91          error_type errorCode = NO_ERROR;
92
93          // Bloqueo mutex
94          pthread_mutex_lock (&i2c_mutex);
95
96          i2c_file = open(i2c_dev->i2c_port, O_RDWR);
97
98          if(i2c_file > 0){
99
100                 memset(outbuf, 0, sizeof(outbuf));
101                 outbuf[0] = i2c_dev->reg_addr;
102
103                 for(int i = 0; i < i2c_dev->length; i++){
104
105                         outbuf[i + 1] = i2c_dev->data[i];
106                 }
107                 message[0].addr = i2c_dev->dev_addr;
108                 message[0].flags = 0; // 0 -> Write
109                 message[0].len   = i2c_dev->length + 1;  // N bytes
110                 message[0].buf   = outbuf;
111
112                 i2c_packet.msgs = message;
113                 i2c_packet.nmsgs = 1;
114
115                 if(ioctl(i2c_file, I2C_RDWR, &i2c_packet) < 0){
116 #ifdef DEBUG
117                 printf("[ERROR]\t[I2C]\t\tioctl: %s\n", strerror(errno)
                        );
118 #endif
119                 errorCode = APP_REPORT(I2C, WRITING_IOCTL_FILE);
120                 }
121         }else{
122
123 #ifdef DEBUG
124                 printf("[ERROR]\t[I2C]\t\tOpen I2C port file: %s \n",
                        strerror(errno));
125 #endif
126                 errorCode = APP_REPORT(I2C, OPENING_I2C_PORT);
127         }
128
129         close(i2c_file);
130         // Desbloqueo mutex
131         pthread_mutex_unlock (&i2c_mutex);
132
133         return errorCode;
134 }
```

```c
/*
 * i2c.h
 *
 *  Created on: Jun 30, 2021
 *      Author: bernardo
 */

#ifndef I2C_H_
#define I2C_H_

#include <stdio.h>
#include <linux/i2c.h>
#include <linux/i2c-dev.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <stdint.h>
#include <string.h>
#include <pthread.h>

#include "../../app_includes/app_typedef.h"
#include "../../app_includes/app_errors.h"

//#define       DEBUG

#define         I2C_0   "/dev/i2c-0"
#define         I2C_1   "/dev/i2c-1"

/*
 * I2C library location
 */
#define         I2C_LIBRARY_DIRECTORY   "/usr/lib/"
#define         I2C_LIBRARY_NAME        "libI2C.so.1"

/*
 * @brief       Estructura de parametros I2C de un dispositivo
 */
typedef struct {

        char*           i2c_port; // Puerto I2C al que esta conectado
        uint16_t        dev_addr; // Direccion I2C del dispositivo
        uint8_t         reg_addr; // Direccion del registro
        uint8_t*        data; // Puntero al buffer de datos
        uint16_t        length; // Numero de bytes

}i2c_device_struct;


#endif /* I2C_H_ */
```

```c
/*
 * i2c_export.h
 *
 *  Created on: Jun 30, 2021
 *      Author: bernardo
 */

#ifndef I2C_EXPORT_H_
#define I2C_EXPORT_H_

#include "I2C.h"

error_type (*read_i2c_data)(i2c_device_struct *i2c_dev);
error_type (*write_i2c_data)(i2c_device_struct *i2c_dev);

#endif /* I2C_EXPORT_H_ */
```

Volver

Apéndice R

Anexo 19

```c
/*
 * gpio.c
 *
 *  Created on: Jul 23, 2021
 *      Author: bernardo
 */

#include "GPIO.h"

static pthread_mutex_t gpio_mutex = PTHREAD_MUTEX_INITIALIZER;
/**
 * @brief       Funcion que exporta un gpio y lo configura
 * @param       gpio            Numero de GPIO que se va a configurar
 * @param       direction       Direccion (out o in)
 * @retval      Codigo de error
 */
error_type configGPIO(uint8_t gpio, char* direction)
{
 error_type codeError = NO_ERROR;
 int8_t     fd_export; // File descriptor export file
 int8_t     fd_direction; // File descriptor direction file
 char       gpio_str[4] = {0}; // String gpio
 char       gpio_path_str[50] = {0}; // String path gpio

 // Bloqueo mutex
 pthread_mutex_lock (&gpio_mutex);

 fd_export = open("/sys/class/gpio/export", O_WRONLY);

 if(fd_export < 0){

    codeError = APP_REPORT(GPIO, OPENING_EXPORT_FILE);
#ifdef DEBUG
    printf("[ERROR]\t[GPIO]\t\tOpening GPIO %d export file: %s \n",
       gpio, strerror(errno));
#endif
```

```
 }else{
#ifdef DEBUG
    printf("[OK]\t[GPIO]\t\tExport file opened \n");
#endif
    // Numero de gpio como cadena de caracteres
    sprintf(gpio_str, "%d", gpio);

    // Exporta el pin escribiendo el numero de gpio en el fichero /sys/
       class/gpio/export
    if(write(fd_export, gpio_str, strlen(gpio_str)) < 1){

    codeError = APP_REPORT(GPIO, WRITING_EXPORT_FILE);
#ifdef DEBUG
    printf("[ERROR]\t[GPIO]\t\tWriting GPIO %d export file: %s. Maybe
       the GPIO is already exported \n", gpio, strerror(errno));
#endif
    }else{
        sprintf(gpio_path_str, "/sys/class/gpio/gpio%d/direction", gpio
           );

        fd_direction = open(gpio_path_str, O_WRONLY);

        if(fd_direction < 0){

            codeError = APP_REPORT(GPIO, OPENING_DIRECTION_FILE);
#ifdef DEBUG
            printf("[ERROR]\t[GPIO]\t\tOpening gpio direction file: %s
               \n", strerror(errno));
#endif
        }else{
            if(!strncmp("out", direction, 3)){

                if(write(fd_direction, "out", strlen("out")) < 0){
                    codeError = APP_REPORT(GPIO, WRITING_DIRECTION_FILE
                       );
#ifdef DEBUG
                    printf("[ERROR]\t[GPIO]\t\tSetting GPIO %d as
                       output \n", gpio);
#endif
                }else{
#ifdef DEBUG
                printf("[OK]\t[GPIO]\t\tGPIO %d set as out\n", gpio);
#endif
                }
            }else if(!strncmp("in", direction, 2)){

                if(write(fd_direction, "in", strlen("in")) < 0){

                codeError = APP_REPORT(GPIO, WRITING_DIRECTION_FILE);
#ifdef DEBUG
                printf("[ERROR]\t[GPIO]\t\tSetting GPIO %d as input \n"
                   , gpio);
#endif
```

```
                }else{
#ifdef DEBUG
                printf("[OK]\t[GPIO]\t\tGPIO %d set as in\n", gpio);
#endif
                }
            }else{
                codeError = APP_REPORT(GPIO, INVALID_DIRECTION_VALUE);
#ifdef DEBUG
                printf("[ERROR]\t[GPIO]\t\tDirection \"%s\" is not
                   valid. Must be \"out\" or \"in\" \n", direction);
#endif
            }
    close(fd_direction);
        }
    }
    close(fd_export);
}
 // Desbloqueo mutex
 pthread_mutex_unlock (&gpio_mutex);

 return codeError;
}

/**
 * @brief       Funcion que libera un GPIO
 * @param       gpio    Numero de GPIO que se quiere liberar
 * @retval      Codigo de error
 */
error_type freeGPIO(uint8_t gpio)
{
error_type  codeError = NO_ERROR;
int8_t      fd_unexport; // File descriptor export file
char        gpio_str[4] = {0}; // String gpio

 // Bloqueo mutex
 pthread_mutex_lock (&gpio_mutex);

 fd_unexport = open("/sys/class/gpio/unexport", O_WRONLY);

 if(fd_unexport < 0){
    codeError = APP_REPORT(GPIO, OPENING_UNEXPORT_FILE);
#ifdef DEBUG
    printf("[ERROR]\t[GPIO]\t\tOpening gpio unexport file: %s \n",
       strerror(errno));
#endif
 }else{
#ifdef DEBUG
    printf("[OK]\t[GPIO]\t\tUnexport file opened \n");
#endif

    // Numero de gpio como cadena de caracteres
    sprintf(gpio_str, "%d", gpio);

```

```
133     // Exporta el pin escribiendo el numero de gpio en el fichero /sys/
            class/gpio/export
134     if(write(fd_unexport, gpio_str, strlen(gpio_str)) < 1){
135
136         codeError = APP_REPORT(GPIO, WRITING_UNEXPORT_FILE);
137 #ifdef DEBUG
138         printf("[ERROR]\t[GPIO]\t\tWriting GPIO %d unexport file: %s.
                Maybe the GPIO is already unexported \n", gpio, strerror(
                errno));
139 #endif
140     }else{
141 #ifdef DEBUG
142         printf("[OK]\t[GPIO]\t\tUnexported GPIO %d \n", gpio);
143 #endif
144     }
145
146  close(fd_unexport);
147 }
148
149  // Desbloqueo mutex
150  pthread_mutex_unlock (&gpio_mutex);
151
152  return codeError;
153 }
154
155 /**
156  * @brief Funcion que obtiene el valor actual de un GPIO
157  * @param gpio Numero de GPIO del que se quiere obtener el valor
158  * @param value Puntero a la variable donde se almacenara el valor
159  * @retval Codigo de error
160  */
161 error_type getGPIO_Value(uint8_t gpio, uint8_t* value)
162 {
163  error_type codeError = NO_ERROR;
164
165  char gpio_value_str[2] = {0};  // String gpio
166  char gpio_path_str[50] = {0};  // String path gpio
167
168  int8_t fd_value;               // File descriptor
169
170  sprintf(gpio_path_str, "/sys/class/gpio/gpio%d/value", gpio);
171
172  // Bloqueo mutex
173  pthread_mutex_lock (&gpio_mutex);
174
175  fd_value = open(gpio_path_str, O_RDONLY);
176
177  if(fd_value < 0){
178     codeError = APP_REPORT(GPIO, OPENING_VALUE_FILE);
179 #ifdef DEBUG
180     printf("[ERROR]\t[GPIO]\t\tOpening gpio %d value file: %s \n", gpio
            , strerror(errno));
181 #endif
```

```
182  }else{
183     if(read(fd_value, gpio_value_str, 1) < 1){
184
185         codeError = APP_REPORT(GPIO, READING_VALUE_FILE);
186 #ifdef DEBUG
187         printf("[ERROR]\t[GPIO]\t\tOpening gpio %d value file: %s \n",
                gpio, strerror(errno));
188 #endif
189     }else{
190
191         *value = atoi(gpio_value_str);
192 #ifdef DEBUG
193         printf("[OK]\t[GPIO]\t\tGPIO %d value is: %d \n", gpio, *value)
                ;
194 #endif
195     }
196
197  close(fd_value);
198
199 }
200  // Desbloqueo mutex
201  pthread_mutex_unlock (&gpio_mutex);
202
203  return codeError;
204 }
205
206 /**
207  * @brief Obtiene la direccion actual de un GPIO (output o input)
208  * @param gpio Numero de GPIO del cual se quiere obtener la direccion
209  * @param direction     Puntero a la cadena de caracteres
210  * @retval Codigo de error
211  */
212 error_type getGPIO_Direction(uint8_t gpio, char* direction)
213 {
214  error_type codeError = NO_ERROR;
215
216  char gpio_path_str[50] = {0}; // String path gpio
217  int8_t fd_direction; // File descriptor
218
219  sprintf(gpio_path_str, "/sys/class/gpio/gpio%d/direction", gpio);
220
221  // Bloqueo mutex
222  pthread_mutex_lock (&gpio_mutex);
223
224  fd_direction = open(gpio_path_str, O_RDONLY);
225
226  if(fd_direction < 0){
227
228     codeError = APP_REPORT(GPIO, OPENING_DIRECTION_FILE);
229 #ifdef DEBUG
230     printf("[ERROR]\t[GPIO]\t\tOpening gpio %d value file: %s \n", gpio
        , strerror(errno));
231 #endif
```

```
232  }else{
233      if(read(fd_direction, direction, 4) < 2){
234          codeError = APP_REPORT(GPIO, READING_DIRECTION_FILE);
235 #ifdef DEBUG
236          printf("[ERROR]\t[GPIO]\t\tOpening gpio %d value file: %s \n",
                 gpio, strerror(errno));
237 #endif
238      }else{
239 #ifdef DEBUG
240      printf("[OK]\t[GPIO]\t\tGPIO %d direction is: %s \n", gpio,
             direction);
241 #endif
242      }
243      close(fd_direction);
244 }
245  // Desbloqueo mutex
246  pthread_mutex_unlock (&gpio_mutex);
247
248  return codeError;
249 }
250
251 /**
252  * @brief Funcion que cambia el valor de un GPIO de salida
253  * @param gpio Numero de GPIO del cual se quiere cambiar el valor
254  * @param value Valor que se quiere dar al GPIO (debe ser 0 o 1)
255  * @reval Codigo de error
256  */
257 error_type setGPIO_Value(uint8_t gpio, uint8_t value)
258 {
259  error_type     codeError = NO_ERROR;
260
261  char gpio_value_str[2] = {0}; // String gpio
262  char gpio_path_str[50] = {0}; // String path gpio
263
264  int8_t fd_value;
265
266  // Bloqueo mutex
267  pthread_mutex_lock (&gpio_mutex);
268
269  if(value >= 0 && value <= 1) {
270
271      sprintf(gpio_path_str, "/sys/class/gpio/gpio%d/value", gpio);
272      fd_value = open(gpio_path_str, O_WRONLY);
273
274      if(fd_value < 0){
275          codeError = APP_REPORT(GPIO, OPENING_VALUE_FILE);
276 #ifdef DEBUG
277      printf("[ERROR]\t\t[GPIO]\tOpening GPIO %d value file: %s \n", gpio
             , strerror(errno));
278 #endif
279      }else {
280          sprintf(gpio_value_str, "%d", value);
```

```
281         if(write(fd_value, gpio_value_str, strlen(gpio_value_str)) < 0)
                {
282
283             codeError = APP_REPORT(GPIO, WRITING_VALUE_FILE);
284 #ifdef DEBUG
285             printf("[ERROR]\t\t[GPIO]\tWriting GPIO %d value file: %s \
                    n", gpio, strerror(errno));
286 #endif
287
288         }else{
289 #ifdef DEBUG
290             printf("[OK]\t[GPIO]\t\tGPIO %d value set: %s \n", gpio,
                    gpio_value_str);
291 #endif
292         }
293     close(fd_value);
294     }
295  }else{
296     codeError = APP_REPORT(GPIO, INVALID_OUTPUT_VALUE);
297 #ifdef DEBUG
298     printf("[ERROR]\t[GPIO]\t\tValue must be \"0\" or \"1\" \n");
299 #endif
300     }
301  // Desbloqueo mutex
302  pthread_mutex_unlock (&gpio_mutex);
303
304  return codeError;
305 }
```

```c
/*
 * GPIO.h
 *
 *  Created on: Jul 23, 2021
 *      Author: bernardo
 */

#ifndef GPIO_H_
#define GPIO_H_

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <stdint.h>
#include <string.h>
#include <dirent.h>
#include <dlfcn.h>
#include <pthread.h>

#include "../../app_includes/app_typedef.h"
#include "../../app_includes/app_errors.h"

#define         GPIO_LIBRARY_DIRECTORY          "/usr/lib/"
#define         GPIO_LIBRARY_NAME               "libGPIO.so.1"

#endif /* GPIO_H_ */
```

```c
/*
 * GPIO_export.h
 *
 *  Created on: Jul 23, 2021
 *      Author: bernardo
 */

#ifndef GPIO_EXPORT_H_
#define GPIO_EXPORT_H_

#include "GPIO.h"

error_type (*configGPIO)(uint8_t gpio, char* direction);
error_type (*freeGPIO)(uint8_t gpio);
error_type (*getGPIO_Value)(uint8_t gpio, uint8_t* value);
error_type (*getGPIO_Direction)(uint8_t gpio, char* direction);
error_type (*setGPIO_Value)(uint8_t gpio, uint8_t value);


#endif /* GPIO_EXPORT_H_ */
```

Volver

Apéndice S

Anexo 20

```
/*
 * libWSEN_TIDS.c
 *
 *  Created on: Jan 26, 2022
 *      Author: bernardo
 */
#include "WSEN_TIDS.h"

static i2c_device_struct WSEN_i2c_info = {0}; // Estructura I2C del sensor WSEN-TIDS
/*
 * @brief       Funcion que inicializa el sensor de WSEN_TIDS
 * @retval      Codigo de error
 */
error_type WSEN_TIDS_Initialize(void)
{
 void *libHandlerI2C = NULL;
 char route[256] = {0};
 // Carga la libreria I2C
 strcat(route, I2C_LIBRARY_DIRECTORY);
 strcat(route, I2C_LIBRARY_NAME);

 libHandlerI2C = dlopen(route, RTLD_LAZY);

 if(libHandlerI2C == NULL) {
#ifdef DEBUG
        printf("[ERROR]\t[WSEN-TIDS]\tLoading I2C library: %s \n", strerror(errno));
#endif
        return APP_REPORT(WSEN_TIDS, LOADING_I2C_LIBRARY);
 }else {
#ifdef DEBUG
        printf("[OK]\t[WSEN-TIDS]\tI2C library loaded successfully \n");
#endif
        // Carga de funciones de la libreria I2C:
```

```
34          read_i2c_data = (error_type ( *)(i2c_device_struct *)) dlsym(
               libHandlerI2C, "read_i2c_data");
35          write_i2c_data = (error_type ( *)(i2c_device_struct *)) dlsym(
               libHandlerI2C, "write_i2c_data");
36
37          /*      Inicializacion de datos I2C del sensor WSEN-TIDS */
38          uint8_t buffer[1];
39
40          WSEN_i2c_info.i2c_port          = I2C_0;
41          WSEN_i2c_info.dev_addr          = WSEN_ADDR;
42          WSEN_i2c_info.data              = buffer;
43          WSEN_i2c_info.length            = 1;
44          WSEN_i2c_info.reg_addr          = SOFT_RESET;
45
46          buffer[0] = 0x2;
47
48          if(write_i2c_data(&WSEN_i2c_info) != NO_ERROR){
49
50  #ifdef DEBUG
51          printf("[ERROR]\t[WSEN-TIDS]\tEnable software reset: %s \n",
               strerror(errno));
52  #endif
53
54          return APP_REPORT(WSEN_TIDS, SEND_RESET_COMMAND);
55          }
56          sleep(0.3);
57
58          buffer[0] = 0x0;
59
60          if(write_i2c_data(&WSEN_i2c_info) != NO_ERROR){
61
62  #ifdef DEBUG
63          printf("[ERROR]\t[WSEN-TIDS]\tDisable software reset: %s \n",
               strerror(errno));
64  #endif
65          return APP_REPORT(WSEN_TIDS, SEND_RESET_COMMAND);
66          }
67          sleep(0.3);
68
69          WSEN_i2c_info.reg_addr          = WSEN_CTRL;
70          buffer[0] = 0x5C;
71
72          if(write_i2c_data(&WSEN_i2c_info) != NO_ERROR){
73
74  #ifdef DEBUG
75          printf("[ERROR]\t[WSEN-TIDS]\tSend control command: %s \n",
               strerror(errno));
76  #endif
77
78          return APP_REPORT(WSEN_TIDS, SEND_CTRL_COMMAND);
79          }
80  }
81
```

```
82  #ifdef DEBUG
83          printf("[OK]\t[WSEN-TIDS]\tSensor Initialized \n");
84  #endif
85
86   return NO_ERROR;
87  }
88  /*
89   * @brief       Funcion que configura un LED de un color
90   * @param       Temp    Led que se quiere configurar
91   * @retval      Codigo de error
92   */
93  error_type WSEN_TIDS_getTemperature(float* Temp)
94  {
95   uint8_t array_temp[2] = {0, 0};
96
97   WSEN_i2c_info.i2c_port = I2C_0;
98   WSEN_i2c_info.dev_addr = WSEN_ADDR;
99   WSEN_i2c_info.data     = array_temp;
100  WSEN_i2c_info.reg_addr = DATA_T_L;
101  WSEN_i2c_info.length   = 2;
102
103  if(read_i2c_data(&WSEN_i2c_info) != NO_ERROR){
104
105 #ifdef DEBUG
106         printf("[ERROR]\t[WSEN-TIDS]\tReading temperature value: %s \n"
               , strerror(errno));
107 #endif
108
109         return APP_REPORT(WSEN_TIDS, READING_TEMPERATURE);
110 }
111
112  uint32_t raw_value = 0;
113
114  raw_value = ((array_temp[1] << 8) | (array_temp[0]));
115
116  *Temp = (float)(raw_value * 0.01);
117
118 #ifdef DEBUG
119  printf("[OK]\t[WSEN-TIDS]\tTemperature value: %.1f C\n", *Temp);
120 #endif
121
122  return NO_ERROR;
123 }
```

```c
/*
 * libWSEN_TIDS.h
 *
 *  Created on: Jan 26, 2022
 *      Author: bernardo
 */
#ifndef LIBWSEN_TIDS_H_
#define LIBWSEN_TIDS_H_

#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <stdint.h>
#include <string.h>
#include <dirent.h>
#include <dlfcn.h>

#include "../libI2C/I2C_export.h"
#include "../../app_includes/app_errors.h"
#include "../../app_includes/app_typedef.h"

#define WSEN_ADDR       0x3F // Sensor I2C address
#define DEVICE_ID       0x01 // Device ID Register
#define T_H_LIMIT       0x02 // Temperature limit register H
#define T_L_LIMIT       0x03 // Temperature limit register L
#define WSEN_CTRL       0x04 // Control register
#define STATUS          0x05 // Status register
#define DATA_T_L        0x06 // Temperature output register L
#define DATA_T_H        0x07 // Temperature output register H
#define SOFT_RESET      0x0C // Software reset register

#endif /* LIBWSEN_TIDS_H_ */
```

```c
/*
 * libWSEN_TIDS_export.h
 *
 *  Created on: Jan 26, 2022
 *      Author: bernardo
 */
#ifndef LIBWSEN_TIDS_EXPORT_H_
#define LIBWSEN_TIDS_EXPORT_H_

#include "WSEN_TIDS.h"

error_type (*WSEN_TIDS_Initialize)(void);
error_type (*WSEN_TIDS_getTemperature)(float*);

#endif /* LIBWSEN_TIDS_EXPORT_H_ */
```

Volver

Apéndice T

Anexo 21

```
/*
 * PCA9532.c
 *
 *  Created on: Aug 2, 2021
 *      Author: bernardo
 */
#include "PCA9532.h"

LED_struct      LED[12] = {0};

static i2c_device_struct        PCA9532_I2C_info = {0};


/*
 * @brief       Funcion que configura un LED de un color
 * @param       LEDn    Led que se quiere configurar
 * @param       color   Color del led
 * @retval      Codigo de error
 */
error_type setLED_Value(uint8_t LEDn, uint8_t color)
{
 error_type errorCode = NO_ERROR;
 uint8_t buffer[3] = {0};

 switch(color) {

        case BLUE:
                LED[LEDn*3 + BLUE].value        = 1;
                LED[LEDn*3 + GREEN].value       = 0;
                LED[LEDn*3 + RED].value         = 0;
                break;
        case GREEN:
                LED[LEDn*3 + BLUE].value        = 0;
                LED[LEDn*3 + GREEN].value       = 1;
                LED[LEDn*3 + RED].value         = 0;
                break;
```

```
37          case RED:
38                  LED[LEDn*3 + BLUE].value        = 0;
39                  LED[LEDn*3 + GREEN].value       = 0;
40                  LED[LEDn*3 + RED].value         = 1;
41                  break;
42          case PURPLE:
43                  LED[LEDn*3 + BLUE].value        = 1;
44                  LED[LEDn*3 + GREEN].value       = 0;
45                  LED[LEDn*3 + RED].value         = 1;
46                  break;
47          case YELLOW:
48                  LED[LEDn*3 + BLUE].value        = 0;
49                  LED[LEDn*3 + GREEN].value       = 1;
50                  LED[LEDn*3 + RED].value         = 1;
51                  break;
52          case CYAN:
53                  LED[LEDn*3 + BLUE].value        = 1;
54                  LED[LEDn*3 + GREEN].value       = 1;
55                  LED[LEDn*3 + RED].value         = 0;
56                  break;
57          case WHITE:
58                  LED[LEDn*3 + BLUE].value        = 1;
59                  LED[LEDn*3 + GREEN].value       = 1;
60                  LED[LEDn*3 + RED].value         = 1;
61                  break;
62          case LED_OFF:
63                  LED[LEDn*3 + BLUE].value        = 0;
64                  LED[LEDn*3 + GREEN].value       = 0;
65                  LED[LEDn*3 + RED].value         = 0;
66                  break;
67          default:
68                  errorCode = APP_REPORT(PCA9532, INCORRECT_COLOR);
69
70  #ifdef DEBUG
71                  printf("[ERROR]\t[PCA9532]\tIncorrect color \n");
72  #endif
73                  break;
74  }
75
76   buffer[0] = LED[LED0].value | (LED[LED1].value << 2) | (LED[LED2].
        value << 4) | (LED[LED3].value << 6);
77   buffer[1] = LED[LED4].value | (LED[LED5].value << 2) | (LED[LED6].
        value << 4) | (LED[LED7].value << 6);
78   buffer[2] = LED[LED8].value | (LED[LED9].value << 2) | (LED[LED10].
        value << 4) | (LED[LED11].value << 6);
79
80
81   // Datos del envio I2C
82   PCA9532_I2C_info.reg_addr = 0x16; // LED0 to LED4 with Autoincrement
        (0x10 | 0x06)
83   PCA9532_I2C_info.length    = 3;
84   PCA9532_I2C_info.data      = buffer;
85
```

```
 if(write_i2c_data(&PCA9532_I2C_info) != NO_ERROR) {

#ifdef DEBUG
        errorCode = APP_REPORT(PCA9532, SETTING_LED_COLOR);
        printf("[ERROR]\t[PCA9532]\tWriting LED color \n");
#endif
 }else{
#ifdef DEBUG
        printf("[OK]\t[PCA9532]\tLED color set \n");
#endif
 }

 return errorCode;
}
/*
 * @brief       Funcion de inicializacion del PCA9532
 * @retval      Codigo de error
 */
error_type PCA9532_Initialize(void)
{
 error_type errorCode = NO_ERROR;
 void *libHandlerI2C = NULL;
 void *libHandlerGPIO   = NULL;
 char route[256] = {0};

 // Carga la libreria I2C
 strcat(route, I2C_LIBRARY_DIRECTORY);
 strcat(route, I2C_LIBRARY_NAME);
 libHandlerI2C = dlopen(route, RTLD_LAZY);

 if(libHandlerI2C == NULL) {

    errorCode = APP_REPORT(PCA9532, LOADING_I2C_LIBRARY);
#ifdef DEBUG
        printf("[ERROR]\t[PCA9532]\tLoading I2C library\n");
#endif
 }else {
#ifdef DEBUG
        printf("[OK]\t[PCA9532]\tI2C library loaded successfully \n");
#endif
        // Carga la libreria GPIO
        memset(route, 0, sizeof(route));                // Inicializa
           de nuevo el array a 0
        strcat(route, GPIO_LIBRARY_DIRECTORY);
        strcat(route, GPIO_LIBRARY_NAME);

        libHandlerGPIO = dlopen(route, RTLD_LAZY);

        if(libHandlerGPIO == NULL){

            errorCode = APP_REPORT(PCA9532, LOADING_GPIO_LIBRARY);
#ifdef DEBUG
                printf("[ERROR]\t[PCA9532]\tLoading GPIO library \n");
```

```
138 #endif
139         }else {
140 #ifdef DEBUG
141                 printf("[OK]\t[PCA9532]\tGPIO Library loaded
                        successfully \n");
142 #endif
143                 // Carga de funciones de la libreria GPIO
144                 configGPIO = (error_type ( *)(uint8_t, char*)) dlsym(
                        libHandlerGPIO, "configGPIO");
145                 freeGPIO = (error_type ( *)(uint8_t)) dlsym(
                        libHandlerGPIO, "freeGPIO");
146                 getGPIO_Value = (error_type ( *)(uint8_t, uint8_t*))
                        dlsym(libHandlerGPIO, "getGPIO_Value");
147                 getGPIO_Direction = (error_type ( *)(uint8_t, char*))
                        dlsym(libHandlerGPIO, "getGPIO_Direction");
148                 setGPIO_Value = (error_type ( *)(uint8_t, uint8_t))
                        dlsym(libHandlerGPIO, "setGPIO_Value");
149
150                 // Habilita el PCA9532 a nivel alto (1)
151                 configGPIO(LED_RST, "out");
152                 setGPIO_Value(LED_RST, 1);
153
154                 // Carga de funciones de la libreria I2C:
155                 read_i2c_data = (error_type ( *)(i2c_device_struct *))
                        dlsym(libHandlerI2C, "read_i2c_data");
156                 write_i2c_data = (error_type ( *)(i2c_device_struct *))
                         dlsym(libHandlerI2C, "write_i2c_data");
157
158                 /*      Inicializacion de la estructura de datos I2C
                        del sensor PCA9532 */
159                 PCA9532_I2C_info.i2c_port       = I2C_0;
160                 PCA9532_I2C_info.dev_addr       = PCA9532_ADDR;
161             }
162     }
163  return errorCode;
164 }
```

```
1  /*
2   * PCA9532.h
3   *
4   *  Created on: Aug 2, 2021
5   *      Author: bernardo
6   */
7
8  #ifndef PCA9532_H_
9  #define PCA9532_H_
10
11 #include <dlfcn.h>
12 #include "../libI2C/I2C_export.h"
13 #include "../libGPIO/GPIO_export.h"
14 #include "../../app_includes/app_errors.h"
15 #include "../../app_includes/app_typedef.h"
```

```
16 /*
17  * PCA9532 library location
18  */
19 #define         PCA9532_LIBRARY_DIRECTORY       "/usr/lib/"
20 #define         PCA9532_LIBRARY_NAME            "libPCA9532.so.1"
21
22 /*
23  * LED colors
24  */
25 #define BLUE            0
26 #define GREEN           1
27 #define RED             2
28 #define PURPLE          3
29 #define YELLOW          4
30 #define CYAN            5
31 #define WHITE           6
32 #define LED_OFF         7
33
34 // PCA9532 I2C Address
35 #define PCA9532_ADDR    0x67
36
37 /*
38  * LED struct
39  */
40 typedef struct {
41
42         uint8_t         value;
43
44 }LED_struct;
45
46 /*
47  * PCA9532 Outputs
48  */
49
50 #define LED0    0
51 #define LED1    1
52 #define LED2    2
53 #define LED3    3
54 #define LED4    4
55 #define LED5    5
56 #define LED6    6
57 #define LED7    7
58 #define LED8    8
59 #define LED9    9
60 #define LED10   10
61 #define LED11   11
62
63 #endif /* PCA9532_H_ */
```

```c
/*
 * PCA9532_export.h
 *
 *  Created on: Aug 2, 2021
 *      Author: bernardo
 */

#ifndef PCA9532_EXPORT_H_
#define PCA9532_EXPORT_H_

#include "PCA9532.h"

error_type (*setLED_Value)(uint8_t LEDn, uint8_t color);
error_type (*PCA9532_Initialize)(void);

#endif /* PCA9532_EXPORT_H_ */
```

Volver

Apéndice U

Anexo 22

```
/*
 * PAC1932.c
 *
 *  Created on: Jul 6, 2021
 *      Author: bernardo
 */

#include "PAC1932.h"

static i2c_device_struct PAC1932_i2c_info = {0}; // Estructura I2C
static float PAC1932_FSC        = 0; // Full Scale Current
static float PAC1932_PowerFSC   = 0; // Power Full Scale Current
/*
 * @brief       Funcion de inicializacion del sensor PAC1932
 * @return      errorCode       Codigo de error
 */
error_type PAC1932_Initialize(void)
 {
 error_type errorCode   = NO_ERROR;
 void *libHandlerI2C    = NULL;
 void *libHandlerGPIO   = NULL;
 char route[256]                = {0};

 // Carga la libreria I2C
 strcat(route, I2C_LIBRARY_DIRECTORY);
 strcat(route, I2C_LIBRARY_NAME);

 libHandlerI2C = dlopen(route, RTLD_LAZY);

 if(libHandlerI2C == NULL) {

#ifdef DEBUG
        printf("[ERROR]\t[PAC1932]\tLoading I2C library \n");
#endif
        errorCode = APP_REPORT(PAC1932, LOADING_I2C_LIBRARY);
 }else {
```

```
37 #ifdef DEBUG
38         printf("[OK]\t[PAC1932]\tI2C Library loaded successfully \n");
39 #endif
40     // Carga la libreria GPIO
41         memset(route, 0, sizeof(route));                // Inicializa
               de nuevo el array a 0
42         strcat(route, GPIO_LIBRARY_DIRECTORY);
43         strcat(route, GPIO_LIBRARY_NAME);
44 
45         libHandlerGPIO = dlopen(route, RTLD_LAZY);
46 
47         if(libHandlerGPIO == NULL){
48                 errorCode = APP_REPORT(PAC1932, LOADING_GPIO_LIBRARY);
49 #ifdef DEBUG
50         printf("[ERROR]\t[PAC1932]\tLoading GPIO library \n");
51 #endif
52         }else {
53 #ifdef DEBUG
54         printf("[OK]\t[PAC1932]\tGPIO Library loaded successfully \n");
55 #endif
56         // Carga de funciones de la libreria GPIO
57         configGPIO = (error_type ( *)(uint8_t, char*)) dlsym(
               libHandlerGPIO, "configGPIO");
58         freeGPIO = (error_type ( *)(uint8_t)) dlsym(libHandlerGPIO, "
               freeGPIO");
59         getGPIO_Value = (error_type ( *)(uint8_t, uint8_t*)) dlsym(
               libHandlerGPIO, "getGPIO_Value");
60         getGPIO_Direction = (error_type ( *)(uint8_t, char*)) dlsym(
               libHandlerGPIO, "getGPIO_Direction");
61         setGPIO_Value = (error_type ( *)(uint8_t, uint8_t)) dlsym(
               libHandlerGPIO, "setGPIO_Value");
62 
63         // Habilita el sensor de potencia a nivel bajo (0)
64         configGPIO(PWDN_CURR_SENSOR, "out");
65         setGPIO_Value(PWDN_CURR_SENSOR, 1);
66 
67         // Carga de funciones de la libreria I2C:
68         read_i2c_data = (error_type( *)(i2c_device_struct *)) dlsym(
               libHandlerI2C, "read_i2c_data");
69         write_i2c_data = (error_type( *)(i2c_device_struct *)) dlsym(
               libHandlerI2C, "write_i2c_data");
70 
71         // Inicializacion de la estructura I2C del sensor PAC1932. A
               continuacion se indica el puerto I2C y la direccion del
               sensor
72         PAC1932_i2c_info.i2c_port       = I2C_0;
73         PAC1932_i2c_info.dev_addr       = PAC1932_ADDR;
74 
75         // Calculo del FULL-SCALE CURRENT (Equation 4.3 del Datasheet)
76         PAC1932_FSC = (100 / R_SENSE);
77         // Calculo del Power FULL-SCALE CURRENT (Equation 4.5 del
               Datasheet)
78         PAC1932_PowerFSC = (3.2 / R_SENSE);
```

```
79
80      }
81   }
82
83   return errorCode;
84  }
85  /*
86   * @brief Obtiene los valores de voltaje, intensidad y potencia
87   * @param PAC1932_Value Puntero a la estructura de datos
88   * @return errorCode Codigo de error
89   */
90  error_type PAC1932_GetAllValues(PAC1932_struct* PAC1932_Value)
91  {
92  uint8_t buffer[50] = {0}; // Buffer de recepcion de datos del bus I2C
93  uint64_t valor_aux  = 0; // Variable auxiliar para obtener los valores
        de potencia, voltaje e intensidad
94  uint8_t index_reg = 0; // Indice de registros leidos
95  uint8_t index_byte = 0; // Indice de bytes leidos
96  uint8_t i, j;
97
98  // En primer lugar se envia el comando REFRESH para que se actualicen
        los valores de los registros del sensor
99  PAC1932_i2c_info.data           = buffer;
100 PAC1932_i2c_info.reg_addr       = REFRESH;
101 PAC1932_i2c_info.length         = 0;
102
103 if(write_i2c_data(&PAC1932_i2c_info) != NO_ERROR) goto error;
104
105 // Segun el datasheet hay que esperar al menos 1ms para que los valores
        de los registros se estabilicen
106 usleep(1500); // Espera 1,5ms
107
108 // Se leen los valores ACC_COUNT, VPOWER_ACC, VBUS y VSENSE de todos
        los canales
109 PAC1932_i2c_info.reg_addr = ACC_COUNT;
110 PAC1932_i2c_info.length = (ACC_COUNT_LENGTH + VPOWER_ACC_LENGTH +
        VBUS_LENGTH + VSENSE_LENGTH) * 4; // Longitud total bytes
111 if(read_i2c_data(&PAC1932_i2c_info) != NO_ERROR) goto error;
112
113 // Valor del ACC_COUNT
114 PAC1932_Value->Acc_Count = 0;
115
116 for(i = 0; i < ACC_COUNT_LENGTH; i++){
117         PAC1932_Value->Acc_Count = (PAC1932_Value->Acc_Count << 8) |
              buffer[i];
118         index_byte++;
119 }
120 index_reg = index_byte;
121
122 // Valores de POTENCIA (W)
123 for(i = 0; i < TOTAL_CHANNELS; i ++){
124         valor_aux = 0;
125         for(j = index_reg; j < (VPOWER_ACC_LENGTH + index_reg); j++){
```

```
126                 valor_aux = (valor_aux << 8) | buffer[j];
127                 index_byte++;
128         }
129         index_reg = index_byte;
130 
131 /*
132   El valor de potencia obtenido es la acumulacion (suma) de varias
      muestras. Por eso hay que dividirlo por el numero de veces que se
       han realizado dichas acumulaciones (ACC_COUNT)
133  */
134 
135         switch(i){
136 
137         case(SOM):
138         PAC1932_Value->SOM.Power = ((valor_aux * PAC1932_PowerFSC) / (
               DEN_UNIPOLAR_MODE)) / PAC1932_Value->Acc_Count;
139         break;
140         case(Peripherals):
141         PAC1932_Value->Peripherals.Power = ((valor_aux *
               PAC1932_PowerFSC) / (DEN_UNIPOLAR_MODE)) / PAC1932_Value->
               Acc_Count;
142         break;
143         case(Terminal_Battery):
144         PAC1932_Value->Terminal_Battery.Power = ((valor_aux *
               PAC1932_PowerFSC) / (DEN_UNIPOLAR_MODE)) / PAC1932_Value->
               Acc_Count;
145         break;
146         case(USB_Connector):
147         PAC1932_Value->USB_Connector.Power = ((valor_aux *
               PAC1932_PowerFSC) / (DEN_UNIPOLAR_MODE)) / PAC1932_Value->
               Acc_Count;
148         break;
149         default:
150                 goto error;
151         break;
152             }
153 
154         }
155 
156 // Valores de VOLTAJE (V)
157 for(i = 0; i < TOTAL_CHANNELS; i ++){
158 
159         valor_aux = 0;
160 
161         for(j = index_reg; j < (VBUS_LENGTH + index_reg); j++){
162 
163                 valor_aux = (valor_aux << 8) | buffer[j];
164                 index_byte++;
165         }
166 
167         index_reg = index_byte;
168 
169 
```

```
170         switch(i){
171
172         case(SOM):
173         PAC1932_Value->SOM.Voltage = (valor_aux * 32.0) /
                DEN_UNIPOLAR_MEAS;
174         break;
175         case(Peripherals):
176         PAC1932_Value->Peripherals.Voltage = (valor_aux * 32.0) /
                DEN_UNIPOLAR_MEAS;
177         break;
178         case(Terminal_Battery):
179         PAC1932_Value->Terminal_Battery.Voltage = (valor_aux * 32.0) /
                DEN_UNIPOLAR_MEAS;
180         break;
181         case(USB_Connector):
182         PAC1932_Value->USB_Connector.Voltage = (valor_aux * 32.0) /
                DEN_UNIPOLAR_MEAS;
183         break;
184         default:
185                 goto error;
186         break;
187         }
188 }
189
190 // Valores de INTENSIDAD (mA)
191 for(i = 0; i < TOTAL_CHANNELS; i ++){
192
193         valor_aux = 0;
194
195         for(j = index_reg; j < (VSENSE_LENGTH + index_reg); j++){
196                 valor_aux = (valor_aux << 8) | buffer[j];
197                 index_byte++;
198         }
199         index_reg = index_byte;
200
201         switch(i){
202
203         case(SOM):
204         PAC1932_Value->SOM.Intensity = (valor_aux * PAC1932_FSC) /
                DEN_UNIPOLAR_MEAS;
205         break;
206         case(Peripherals):
207         PAC1932_Value->Peripherals.Intensity = (valor_aux * PAC1932_FSC
                ) / DEN_UNIPOLAR_MEAS;
208         break;
209         case(Terminal_Battery):
210         PAC1932_Value->Terminal_Battery.Intensity = (valor_aux *
                PAC1932_FSC) / DEN_UNIPOLAR_MEAS;
211         break;
212         case(USB_Connector):
213         PAC1932_Value->USB_Connector.Intensity = (valor_aux *
                PAC1932_FSC) / DEN_UNIPOLAR_MEAS;
214         break;
```

```
215         default:
216                 goto error;
217         break;
218         }
219 }
220
221 #ifdef DEBUG
222  printf("[OK]\t[PAC1932]\tChannel 0 -> Voltaje: %f V; Intensidad: %f mA
        ; Potencia: %f W\n",PAC1932_Value->SOM.Voltage, PAC1932_Value->SOM
        .Intensity, PAC1932_Value->SOM.Power);
223  printf("[OK]\t[PAC1932]\tChannel 1 -> Voltaje: %f V; Intensidad: %f mA
        ; Potencia: %f W\n",PAC1932_Value->Peripherals.Voltage,
        PAC1932_Value->Peripherals.Intensity, PAC1932_Value->Peripherals.
        Power);
224  printf("[OK]\t[PAC1932]\tChannel 2 -> Voltaje: %f V; Intensidad: %f mA
        ; Potencia: %f W\n",PAC1932_Value->Terminal_Battery.Voltage,
        PAC1932_Value->Terminal_Battery.Intensity, PAC1932_Value->
        Terminal_Battery.Power);
225  printf("[OK]\t[PAC1932]\tChannel 3 -> Voltaje: %f V; Intensidad: %f mA
        ; Potencia: %f W\n",PAC1932_Value->USB_Connector.Voltage,
        PAC1932_Value->USB_Connector.Intensity, PAC1932_Value->
        USB_Connector.Power);
226 #endif
227
228  return (error_type)NO_ERROR;
229
230  error:
231 #ifdef DEBUG
232  printf("[ERROR]\t[PAC1932]\tReading PAC1932 values \n");
233 #endif
234
235  return APP_REPORT(PAC1932, READING_PAC1932_VALUES);
236 }
```

```
1  /*
2   * PAC1932.h
3   *
4   *  Created on: Jul 6, 2021
5   *      Author: bernardo
6   */
7
8  #ifndef PAC1932_H_
9  #define PAC1932_H_
10
11 #include <stdio.h>
12 #include <stdlib.h>
13 #include <fcntl.h>
14 #include <unistd.h>
15 #include <errno.h>
16 #include <stdint.h>
17 #include <string.h>
18 #include <dirent.h>
```

```
19 #include <dlfcn.h>
20
21 #include "../libI2C/I2C_export.h"
22 #include "../libGPIO/GPIO_export.h"
23 #include "../../app_includes/app_typedef.h"
24 #include "../../app_includes/app_errors.h"
25
26 //#define       DEBUG
27 /*
28  * PAC1932 library location
29  */
30 #define         PAC1932_LIBRARY_DIRECTORY       "/usr/lib/"
31 #define         PAC1932_LIBRARY_NAME            "libPAC1932.so.1"
32 /*
33  * PAC1932 Static Params
34  */
35 #define TOTAL_CHANNELS  4 // Numero total de canales que mide el sensor
36 #define R_SENSE 0.22f // Valor en ohmnios de la resistencia externa
37 #define DEN_UNIPOLAR_MEAS 0x10000 // Denominador para medidas
       unipolares -> 2^16
38 #define DEN_BIPOLAR_MEAS 0x1000 //Denominador para medidas bipolares
       -> 2^15
39 #define DEN_UNIPOLAR_MODE 0x10000000 //Denominador para modo unipolar
       -> 2^28
40 #define DEN_BIPOLAR_MODE 0x1000000 //Denominador para modo bipolar  ->
       2^27
41
42 /*
43  * PAC1932 I2C DATA
44  */
45 #define         PAC1932_ADDR    0x10 /* PAC1932 I2C Address     */
46 /*
47  * PAC1932 I2C Registers
48  */
49 #define         REFRESH                         0x00
50 #define         CTRL_PAC1932                    0x01
51 #define         ACC_COUNT                       0x02
52 #define         VPOWER1_ACC                     0x03
53 #define         VPOWER2_ACC                     0x04
54 #define         VPOWER3_ACC                     0x05
55 #define         VPOWER4_ACC                     0x06
56 #define         VBUS1                           0x07
57 #define         VBUS2                           0x08
58 #define         VBUS3                           0x09
59 #define         VBUS4                           0x0A
60 #define         VSENSE1                         0x0B
61 #define         VSENSE2                         0x0C
62 #define         VSENSE3                         0x0D
63 #define         VSENSE4                         0x0E
64 #define         VBUS1_AVG                       0x0F
65 #define         VBUS2_AVG                       0x10
66 #define         VBUS3_AVG                       0x11
67 #define         VBUS4_AVG                       0x12
```

```
68  #define         VSENSE1_AVG                     0x13
69  #define         VSENSE2_AVG                     0x14
70  #define         VSENSE3_AVG                     0x15
71  #define         VSENSE4_AVG                     0x16
72  #define         VPOWER1                         0x17
73  #define         VPOWER2                         0x18
74  #define         VPOWER3                         0x19
75  #define         VPOWER4                         0x1A
76  #define         CHANNEL_DIS                     0x1C
77  #define         NEG_PWR                         0x1D
78  #define         REFRESH_G                       0x1E
79  #define         REFRESH_V                       0x1F
80  #define         SLOW                            0x20
81  #define         CTRL_ACT                        0x21
82  #define         CHANNEL_DIS_ACT                 0x22
83  #define         NEG_PWR_ACT                     0x23
84  #define         CTRL_LAT                        0x24
85  #define         CHANNEL_DIS_LAT                 0x25
86  #define         NEG_PWR_LAT                     0x26
87  #define         PID                             0xFD
88  #define         MID                             0xFE
89  #define         REV                             0xFF
90  /*
91   * PAC1932 I2C Registers Length
92   */
93  #define         REFRESH_LENGTH                  1
94  #define         CTRL_LENGTH                     1
95  #define         ACC_COUNT_LENGTH                3
96  #define         VPOWER_ACC_LENGTH               6
97  #define         VBUS_LENGTH                     2
98  #define         VSENSE_LENGTH                   2
99  #define         VBUS_AVG_LENGTH                 2
100 #define         VSENSE_AVG_LENGTH               2
101 #define         VPOWER_LENGTH                   4
102 #define         CHANNEL_DIS_LENGTH              1
103 #define         NEG_PWR_LENGTH                  1
104 #define         REFRESH_G_LENGTH                1
105 #define         REFRESH_V_LENGTH                1
106 #define         SLOW_LENGTH                     1
107 #define         CTRL_ACT_LENGTH                 1
108 #define         CHANNEL_DIS_ACT_LENGTH          1
109 #define         NEG_PWR_ACT_LENGTH              1
110 #define         CTRL_LAT_LENGTH                 1
111 #define         CHANNEL_DIS_LAT_LENGTH          1
112 #define         NEG_PWR_LAT_LENGTH              1
113 #define         PID_LENGTH                      1
114 #define         MID_LENGTH                      1
115 #define         REV_LENGTH                      1
116
117
118 #endif /* PAC1932_H_ */
```

```c
/*
 * PAC1932_export.h
 *
 *  Created on: Jul 6, 2021
 *      Author: bernardo
 */

#ifndef PAC1932_EXPORT_H_
#define PAC1932_EXPORT_H_

#include "PAC1932.h"

error_type (*PAC1932_Initialize)(void);
error_type (*PAC1932_GetAllValues)(PAC1932_struct* PAC1932_Value);

#endif /* PAC1932_EXPORT_H_ */
```

Volver

Apéndice V

Anexo 23

```
/*
 * CAN.c
 *
 *  Created on: Dec 16, 2021
 *      Author: bernardo
 */

#include "CAN.h"

/**
 * @brief       Inicializa la interfaz CAN
 * @param       can_data Estructura del mensaje CAN
 * @retval      Codigo de error
 */
error_type CAN_Initialize(struct can_message * can_data)
{
 error_type codeError = NO_ERROR;
 int todo_broadcast = 1;
 int ret = 0;
 char command[100];

 memset(command, 0, sizeof(command));
 sprintf(command, "/sbin/ifconfig can0 down");
 printf("command = %s \n", command);
 popen(command,"r");

 sleep(1);

 memset(command, 0, sizeof(command));
 sprintf(command, "ip link set can0 type can bitrate 500000");
 printf("command = %s \n", command);
 popen(command,"r");

 sleep(1);

 memset(command, 0, sizeof(command));
```

```
37  sprintf(command, "/sbin/ifconfig can0 up");
38  printf("command = %s \n", command);
39  popen(command,"r");
40
41  sleep(1);
42
43  can_data->socket = socket(PF_CAN, SOCK_DGRAM, CAN_J1939);
44
45  if(setsockopt(can_data->socket, SOL_SOCKET, SO_BROADCAST, &
        todo_broadcast, sizeof(todo_broadcast))){
46
47          return APP_REPORT(CAN, OPENING_SOCKET);
48  }
49
50  return NO_ERROR;
51  }
52  /**
53  * @brief Configura la interfaz CAN
54  * @param can_data Puntero a estructura de datos CAN
55  * @retval Codigo de error
56  */
57  error_type CAN_Configure(struct can_message * can_data)
58  {
59  can_data->addr.can_family = AF_CAN;
60  can_data->addr.can_addr.j1939.name = J1939_NO_NAME;
61  can_data->addr.can_addr.j1939.addr = 0x20;
62  can_data->addr.can_addr.j1939.pgn = J1939_NO_PGN;
63  can_data->addr.can_ifindex = if_nametoindex("can0");
64
65  if(bind(can_data->socket, (struct sockaddr *)&can_data->addr, sizeof(
        can_data->addr))){
66
67          return APP_REPORT(CAN, BIND_CALL);
68  }
69
70  return NO_ERROR;
71  }
72  /**
73  * @brief Envia una trama CAN
74  * @param can_data Puntero a estructura de datos CAN
75  * @retval Codigo de error
76  */
77  error_type CAN_Send(struct can_message * can_data)
78  {
79  if (write(can_data->socket, &can_data->frame, sizeof(struct can_frame)
        ) != sizeof(struct can_frame)) {
80
81  perror("CAN Send: ");
82      return APP_REPORT(CAN, SEND_FRAME);
83  }
84
85  return NO_ERROR;
86  }
```

```
87  /**
88  * @brief Envia un fichero por el bus CAN
89  * @param can_data Puntero a estructura de datos CAN
90  * @param path Ruta donde se encuentra el fichero
91  * @retval Codigo de error
92  */
93  error_type CAN_SendFile(struct can_message * can_data, char * path)
94  {
95   char json_array[2000];
96   uint32_t n_bytes = 0, j = 0;
97
98   FILE *file = fopen(path, "r");
99
100  memset(json_array, 0, sizeof(json_array));
101
102  while(!feof(file)){
103
104         fscanf(file, "%c", &json_array[n_bytes++]);
105  }
106
107  fclose(file);
108
109  for(j = 0; j < n_bytes; j++){
110
111         printf("%c ", json_array[j]);
112  }
113
114  if (write(can_data->socket, json_array, n_bytes) != n_bytes) {
115
116         perror("CAN Send file: ");
117         return APP_REPORT(CAN, SEND_FRAME);
118  }
119
120  return NO_ERROR;
121 }
122 /**
123 * @brief Recepcion de tramas CAN
124 * @param can_data Estructura del mensaje CAN a recibir
125 * @retval Codigo de error
126 */
127 error_type CAN_Receive(struct can_message * can_data)
128 {
129  int nbytes, i;
130
131  nbytes = read(can_data->socket, &can_data->frame, sizeof(struct
        can_frame));
132
133  if (nbytes < 0) {
134
135         perror("CAN Read: ");
136         return APP_REPORT(CAN, RECEIVE_FRAME);
137  }
138
```

```
 // Imprime la trama recibida
 printf("0x%03X [%d] ",can_data->frame.can_id, can_data->frame.can_dlc)
     ;

 for (i = 0; i < can_data->frame.can_dlc; i++){

        printf("%02X ", can_data->frame.data[i]);
}

 printf("\r\n");

 return NO_ERROR;
}
```

```
/*
 * CAN.h
 *
 *  Created on: Dec 16, 2021
 *      Author: bernardo
 */

#ifndef CAN_H_
#define CAN_H_
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <errno.h>
#include <net/if.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <linux/can.h>
#include <linux/can/raw.h>
#include <linux/can/j1939.h>
#include "../../app_includes/app_typedef.h"
#include "../../app_includes/app_errors.h"
#endif /* CAN_H_ */
```

```
/*
 * CAN_export.h
 *
 *  Created on: Dec 16, 2021
 *      Author: bernardo
 */

#ifndef CAN_EXPORT_H_
#define CAN_EXPORT_H_

#include "CAN.h"

error_type (*CAN_Initialize)(struct can_message *);
```

```
14 error_type (*CAN_Configure)(struct can_message *);
15 error_type (*CAN_Send)(struct can_message *);
16 error_type (*CAN_SendFile)(struct can_message *, char *);
17 error_type (*CAN_Receive)(struct can_message *);
18
19
20 #endif /* CAN_EXPORT_H_ */
```

Volver

Apéndice W

Anexo 24

```
/*
 * main.c
 *
 *  Created on: Jul 26, 2021
 *      Author: bernardo
 */

#include "../../app_shared_libraries/libGPIO/GPIO_export.h"


int main(void){

 uint8_t value;
 char direction[10] = {0};


 void *libHandler = NULL;
 char route[256] = {0};

 // Carga la libreria libPAC1932
 strcat(route, GPIO_LIBRARY_DIRECTORY);
 strcat(route, GPIO_LIBRARY_NAME);

 libHandler = dlopen(route, RTLD_LAZY);

 if(libHandler == NULL) {

   printf("[ERROR]\t[GPIO]\tLoading %s library \n", route);

 }else {

   printf("[OK]\t[GPIO]\tLibrary %s loaded successfully \n", route);

   configGPIO = (error_type ( *)(uint8_t, char*)) dlsym(libHandler, "
       configGPIO");
   freeGPIO = (error_type ( *)(uint8_t)) dlsym(libHandler, "freeGPIO");
```

```
36     getGPIO_Value = (error_type ( *)(uint8_t, uint8_t*)) dlsym(
           libHandler, "getGPIO_Value");
37     getGPIO_Direction = (error_type ( *)(uint8_t, char*)) dlsym(
           libHandler, "getGPIO_Direction");
38     setGPIO_Value = (error_type ( *)(uint8_t, uint8_t)) dlsym(libHandler
           , "setGPIO_Value");
39
40
41
42     configGPIO(EN_4V2, "out");
43     configGPIO(EN_5V_USB_MOB, "out");
44
45     setGPIO_Value(EN_4V2, 0);
46     setGPIO_Value(EN_5V_USB_MOB, 0);
47
48     printf("Pulsa enter \n");
49
50     getchar();
51
52     setGPIO_Value(EN_4V2, 1);
53     setGPIO_Value(EN_5V_USB_MOB, 1);
54
55     printf("Fin del programa \n");
56
57     freeGPIO(EN_4V2);
58     freeGPIO(EN_5V_USB_MOB);
59   }
60
61   return 0;
62 }
```

Volver

Apéndice X

Anexo 25

```
/*
 * main.c
 *
 *  Created on: Jul 9, 2021
 *      Author: bernardo
 */

#include "../../app_shared_libraries/libPAC1932/PAC1932_export.h"

static PAC1932_struct PAC1932_data = {0};

int main (void)
{

 void *libHandler = NULL;
 char route[256] = {0};

 // Carga la libreria libPAC1932
 strcat(route, PAC1932_LIBRARY_DIRECTORY);
 strcat(route, PAC1932_LIBRARY_NAME);

 libHandler = dlopen(route, RTLD_LAZY);

 if(libHandler == NULL) {

   printf("[ERROR]\t[PAC1932]\tLoading %s library \n", route);
 }else {

   printf("[OK]\t[PAC1932]\tLibrary %s loaded successfully \n", route);

   // Carga de funciones de la libreria del sensor PAC1932
   PAC1932_Initialize = (error_type ( *)(void)) dlsym(libHandler, "
       PAC1932_Initialize");
     PAC1932_GetAllValues = (error_type ( *)(PAC1932_struct*)) dlsym
         (libHandler, "PAC1932_GetAllValues");
 }
```

```
35
36  PAC1932_Initialize();
37
38  while(1){
39
40    PAC1932_GetAllValues(&PAC1932_data);
41    printf("\n");
42    sleep(1);
43  }
44
45  return 0;
46 }
```

Volver

Apéndice Y

Anexo 26

```
/*
 * main.c
 *
 *  Created on: Jul 30, 2021
 *      Author: bernardo
 */

#include "../../app_shared_libraries/libPCA9532/PCA9532_export.h"

#include <dlfcn.h>

int main (void)
{
 void *libHandlerPCA9532 = NULL;
 char route[256] = {0};

 // Carga la libreria PCA9532
 strcat(route, PCA9532_LIBRARY_DIRECTORY);
 strcat(route, PCA9532_LIBRARY_NAME);

 libHandlerPCA9532 = dlopen(route, RTLD_LAZY);

 if(libHandlerPCA9532 == NULL) {

   printf("[ERROR]\t[PCA9532]\tLoading %s library \n", route);
 }else {

   printf("[OK]\t[PCA9532]\tLibrary %s loaded successfully \n", route);

   // Carga de funciones de la libreria I2C:
   setLED_Value = (error_type ( *)(uint8_t, uint8_t)) dlsym(
      libHandlerPCA9532, "setLED_Value");

   PCA9532_Initialize = (error_type ( *)(void)) dlsym(libHandlerPCA9532
      , "PCA9532_Initialize");

```

```
35
36    PCA9532_Initialize();
37
38    for(int i = 0; i < 4; i ++){
39
40       for(int j = 0; j < 3; j++){
41
42                setLED_Value(i, j);
43                sleep(1);
44           }
45         }
46         setLED_Value(LED0, PURPLE);
47         sleep(1);
48         setLED_Value(LED1, CYAN);
49         sleep(1);
50         setLED_Value(LED2, YELLOW);
51         sleep(1);
52         setLED_Value(LED3, WHITE);
53         sleep(1);
54
55         for(int i = 0; i < 4; i ++){
56
57            setLED_Value(i, LED_OFF);
58            sleep(1);
59         }
60  }
61
62  return 0;
63 }
```

Volver

Apéndice Z

Anexo 27

```c
/*
 * main.c
 *
 *  Created on: Jan 26, 2022
 *      Author: bernardo
 */
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <stdint.h>
#include <string.h>
#include <dirent.h>
#include <dlfcn.h>

#include "../../app_shared_libraries/libWSEN_TIDS/WSEN_TIDS_export.h"
#include "../../app_includes/app_errors.h"
#include "../../app_includes/app_typedef.h"

int main(void)
{

 float Temperature = 0;
 void *libHandlerWSEN = NULL;
 char route[256] = {0};

 strcat(route, "/usr/lib/");
 strcat(route, "libWSEN_TIDS.so.1");

 libHandlerWSEN = dlopen(route, RTLD_LAZY);

 if(libHandlerWSEN == NULL) {
    printf("[ERROR]\t[WSEN-TIDS]\tLoading %s library \n", route);
 }else {
    printf("[OK]\t[WSEN-TIDS]\tLibrary %s loaded successfully \n",
        route);
```

```
36
37          WSEN_TIDS_Initialize = (error_type ( *)(void)) dlsym(
                libHandlerWSEN, "WSEN_TIDS_Initialize");
38          WSEN_TIDS_getTemperature = (error_type ( *)(float *)) dlsym(
                libHandlerWSEN, "WSEN_TIDS_getTemperature");
39
40          WSEN_TIDS_Initialize();
41
42          while(1){
43
44                  WSEN_TIDS_getTemperature(&Temperature);
45                  sleep(1);
46          }
47  }
48
49  return 0;
50 }
```

Volver

Apéndice AA

Anexo 28

```c
/*
 * main.c
 *
 *  Created on: 15/12/2021
 *      Author: bernardo
 */

#include <dlfcn.h>
#include <stdio.h>

#include "../../app_shared_libraries/libCAN/CAN_export.h"

int main(void)
{
 void *libHandler = NULL;
 struct can_message can_info;
 char route[256] = {0};

 // Carga la libreria libCAN
 strcat(route, "/usr/lib/libCAN.so.1");

 libHandler = dlopen(route, RTLD_LAZY);

 if(libHandler == NULL) {

        printf("[ERROR] Loading CAN library \n");

 }else {

        printf("[OK] CAN library loaded \n");

        CAN_Initialize = (error_type ( *)(struct can_message *)) dlsym(
            libHandler, "CAN_Initialize");
        CAN_Configure = (error_type ( *)(struct can_message *)) dlsym(
            libHandler, "CAN_Configure");
```

```
34          CAN_Send = (error_type ( *)(struct can_message *)) dlsym(
                libHandler, "CAN_Send");
35          CAN_SendFile = (error_type ( *)(struct can_message *, char *))
                dlsym(libHandler, "CAN_SendFile");
36          CAN_Receive     = (error_type ( *)(struct can_message *)) dlsym
                (libHandler, "CAN_Receive");
37
38          CAN_Initialize(&can_info);
39          CAN_Configure(&can_info);
40          CAN_SendFile(&can_info, "/home/info.json");
41  }
42
43  return 0;
44 }
```

Volver

Apéndice AB

Anexo 29

```
//
    ============================================================================

// Name        : main.cpp
// Author      : Bernardo
// Version     :
// Copyright   : Your copyright
// Description : Hello World in C++, Ansi-style
//
    ============================================================================

#include <iostream>
#include <mutex>
#include <thread>
#include <stdio.h>
#include <unistd.h>
#include "StartDaemon.h"

using namespace std;
using namespace nlohmann;

uint32_t TempThread_isActive, PowerThread_isActive;
uint32_t AndroidThread_isActive, CanThread_isActive;

uint32_t TempState, PowerState;
uint32_t AndroidState, CanState;

int main() {

        StartDaemon daemon;

        TempState               = 0;
        PowerState              = 0;
        AndroidState            = 0;
        CanState                = 0;

```

```
        TempThread_isActive     = 1;
        PowerThread_isActive    = 1;
        AndroidThread_isActive  = 1;
        CanThread_isActive      = 1;

        daemon.LaunchThreads();

        return 0;
}
```

```
/*
 * StartDaemon.h
 *
 *  Created on: Nov 29, 2021
 *      Author: bernardo
 */

#ifndef STARTDAEMON_H_
#define STARTDAEMON_H_

#include <iostream>

#include "AndroidControl.h"

class StartDaemon: public AndroidControl {
public:
        StartDaemon();
        virtual ~StartDaemon();

        void get_ThreadsState();
        void stopThreads();
        void LaunchThreads();
};

#endif /* STARTDAEMON_H_ */
```

```
/*
 * StartDaemon.cpp
 *
 *  Created on: Nov 29, 2021
 *      Author: bernardo
 */
#include <thread>
/**
 * Constructor de la clase StartDaemon
 */
StartDaemon::StartDaemon() {
        // TODO Auto-generated constructor stub

        std::cout << "StartDaemon Constructor" << std::endl;
}
```

```
/**
 * Lanza los hilos del sistema de soporte
 */
void StartDaemon::LaunchThreads()
{

 std::cout << "\nDAEMON - Launching Threads... \n" << std::endl;

 std::thread TempThread(&TemperatureControl::TempStateMachine, *this);
 std::thread PowerThread(&PowerControl::PowerStateMachine, *this);
 std::thread AndroidThread(&AndroidControl::communication_state_machine
    , *this);
 std::thread CanThread(&CanControl::CanStateMachine, *this);

 // Finalizacion de los hilos
 TempThread.join();
 PowerThread.join();
 AndroidThread.join();
 CanThread.join();
}
/**
 * Destructor de la clase StartDaemon
 */
StartDaemon::~StartDaemon() {
        // TODO Auto-generated destructor stub
}
```

```
/*
 * SharedLibraries.h
 *
 *  Created on: Nov 29, 2021
 *      Author: bernardo
 */

#ifndef SHAREDLIBRARIES_H_
#define SHAREDLIBRARIES_H_

#include <iostream>
#include <cstring>
#include <cstdio>
#include <mutex>
#include <thread>

#include <stdio.h>
#include <dlfcn.h>


#include "../../app_includes/app_typedef.h"
#include "../../app_includes/app_errors.h"

/*
 * LED colors
```

```
 */
#define BLUE            0
#define GREEN           1
#define RED             2
#define PURPLE          3
#define YELLOW          4
#define CYAN            5
#define WHITE           6
#define LED_OFF         7


/*
 * LEDs Tasks
 */
#define LED_TEMP                0
#define LED_PWR                 1
#define LED_ANDROID             2
#define LED_SIM                 3


class SharedLibraries {
public:
        SharedLibraries();
        virtual ~SharedLibraries();

        void* SearchLibrary(const char*, const char*);

        error_type LoadLibrary(uint32_t);

        error_type (*WSEN_TIDS_Initialize)(void);
        error_type (*WSEN_TIDS_getTemperature)(float *);

        error_type (*configGPIO)(uint8_t, char*);
        error_type (*freeGPIO)(uint8_t);
        error_type (*getGPIO_Value)(uint8_t, uint8_t*);
        error_type (*getGPIO_Direction)(uint8_t, char*);
        error_type (*setGPIO_Value)(uint8_t, uint8_t);

        error_type (*PAC1932_Initialize)(void);
        error_type (*PAC1932_GetAllValues)(PAC1932_struct*);

        error_type (*setLED_Value)(uint8_t, uint8_t);
        error_type (*PCA9532_Initialize)(void);

        error_type (*CAN_Initialize)(struct can_message *);
        error_type (*CAN_Configure)(struct can_message *);
        error_type (*CAN_Send)(struct can_message *);
        error_type (*CAN_SendFile)(struct can_message *, char *);
        error_type (*CAN_Receive)(struct can_message *);
};

#endif /* SHAREDLIBRARIES_H_ */
```

```
/*
 * SharedLibraries.cpp
 *
 *  Created on: Nov 29, 2021
 *      Author: bernardo
 */

#include "SharedLibraries.h"

/**
 * Constructor de la clase SharedLibraries
 */
SharedLibraries::SharedLibraries() {
// TODO Auto-generated constructor stub

 std::cout << "SharedLibraries Constructor" << std::endl;

 this->WSEN_TIDS_Initialize             = NULL;
 this->WSEN_TIDS_getTemperature         = NULL;

 this->configGPIO               = NULL;
 this->freeGPIO                 = NULL;
 this->getGPIO_Value            = NULL;
 this->getGPIO_Direction        = NULL;
 this->setGPIO_Value            = NULL;

 this->PAC1932_Initialize       = NULL;
 this->PAC1932_GetAllValues     = NULL;

 this->setLED_Value             = NULL;
 this->PCA9532_Initialize       = NULL;

 this->CAN_Initialize           = NULL;
 this->CAN_Send                 = NULL;
 this->CAN_SendFile             = NULL;
 this->CAN_Receive              = NULL;
}

SharedLibraries::~SharedLibraries() {
// TODO Auto-generated destructor stub
}
/**
* Carga las librerias compartidas
*/
error_type SharedLibraries::LoadLibrary(uint32_t Library)
{
 error_type codeError = NO_ERROR;
 void* libHandlerWSEN_TIDS = NULL;
 void* libHandlerGPIO = NULL;
 void *libHandlerPAC1932 = NULL;
 void *libHandlerPCA9532 = NULL;
 void *libHandlerCAN = NULL;
```

```
53
54  switch(Library){
55
56  case WSEN_TIDS:
57
58          // Load WSEN-TIDS library
59          libHandlerWSEN_TIDS = this->SearchLibrary("libWSEN_TIDS.so.1",
                "/usr/lib/");
60
61          this->WSEN_TIDS_Initialize = (error_type ( *)(void)) dlsym(
                libHandlerWSEN_TIDS, "WSEN_TIDS_Initialize");
62          this->WSEN_TIDS_getTemperature = (error_type ( *)(float *))
                dlsym(libHandlerWSEN_TIDS, "WSEN_TIDS_getTemperature");
63          break;
64
65  case GPIO:
66
67          // Load GPIO shared library
68          libHandlerGPIO = this->SearchLibrary("libGPIO.so.1", "/usr/lib/
                ");
69
70          this->configGPIO = (error_type ( *)(uint8_t, char*)) dlsym(
                libHandlerGPIO, "configGPIO");
71          this->freeGPIO = (error_type ( *)(uint8_t)) dlsym(
                libHandlerGPIO, "freeGPIO");
72          this->getGPIO_Value     = (error_type ( *)(uint8_t, uint8_t*))
                dlsym(libHandlerGPIO, "getGPIO_Value");
73          this->getGPIO_Direction = (error_type ( *)(uint8_t, char*))
                dlsym(libHandlerGPIO, "getGPIO_Direction");
74          this->setGPIO_Value = (error_type ( *)(uint8_t, uint8_t)) dlsym
                (libHandlerGPIO, "setGPIO_Value");
75          break;
76
77  case PAC1932:
78
79          // Load PAC1932 shared library
80          libHandlerPAC1932 = this->SearchLibrary("libPAC1932.so.1", "/
                usr/lib/");
81
82          this->PAC1932_Initialize = (error_type ( *)(void)) dlsym(
                libHandlerPAC1932, "PAC1932_Initialize");
83          this->PAC1932_GetAllValues = (error_type ( *)(PAC1932_struct*))
                 dlsym(libHandlerPAC1932, "PAC1932_GetAllValues");
84          break;
85
86  case PCA9532:
87
88          // Load PCA9532 shared library
89          libHandlerPCA9532 = this->SearchLibrary("libPCA9532.so.1", "/
                usr/lib/");
90
91          this->setLED_Value = (error_type ( *)(uint8_t, uint8_t)) dlsym(
                libHandlerPCA9532, "setLED_Value");
```

```
92          this->PCA9532_Initialize = (error_type ( *)(void)) dlsym(
                libHandlerPCA9532, "PCA9532_Initialize");
93
94          break;
95
96  case CAN:
97
98          // Load PCA9532 shared library
99          libHandlerCAN = this->SearchLibrary("libCAN.so.1", "/usr/lib/")
                ;
100
101         this->CAN_Initialize = (error_type ( *)(struct can_message *))
                dlsym(libHandlerCAN, "CAN_Initialize");
102         this->CAN_Configure     = (error_type ( *)(struct can_message
                *)) dlsym(libHandlerCAN, "CAN_Configure");
103         this->CAN_Send = (error_type ( *)(struct can_message *)) dlsym(
                libHandlerCAN, "CAN_Send");
104         this->CAN_SendFile      = (error_type ( *)(struct can_message
                *, char *)) dlsym(libHandlerCAN, "CAN_SendFile");
105         this->CAN_Receive = (error_type ( *)(struct can_message *))
                dlsym(libHandlerCAN, "CAN_Receive");
106
107         break;
108
109 default:
110         break;
111
112 }
113
114 return codeError;
115 }
116 /**
117 * Busca la libreria compartida en el filesystem
118 */
119 void* SharedLibraries::SearchLibrary(const char* name, const char*
        directory)
120 {
121  void *libHandler = NULL;
122  char route[256] = {0};
123
124  std::strcat(route, directory);
125  std::strcat(route, name);
126
127  libHandler = dlopen(route, RTLD_LAZY);
128
129  if(libHandler == NULL) printf("[ERROR]\tLoading %s \n", route);
130
131  else printf("[OK]\tLibrary %s loaded \n", name);
132
133  return libHandler;
134 }
```

```
/*
 * TemperatureControl.h
 *
 *  Created on: Nov 29, 2021
 *      Author: bernardo
 */

#ifndef TEMPERATURECONTROL_H_
#define TEMPERATURECONTROL_H_

#include <iostream>
#include <stdint.h>
#include "PowerControl.h"

class TemperatureControl: public PowerControl {
public:
        TemperatureControl();
        virtual ~TemperatureControl();
        void    TempChangeState(uint32_t State);
        void    TempStateMachine();
};

#endif /* TEMPERATURECONTROL_H_ */
```

```
/*
 * TemperatureControl.cpp
 *
 *  Created on: Nov 29, 2021
 *      Author: bernardo
 */
#include "TemperatureControl.h"

extern uint32_t TempThread_isActive;
extern uint32_t TempState;

#define TEMP_IDDLE      0 // Funciones del sensor sin cargar
#define TEMP_INITIALIZE 1 // Inicializacion del sensor
#define TEMP_NORMAL     2 // Sensor inicializado y temperatura normal
#define TEMP_ALARM_1    3 // Temperatura elevada, se activan los
   ventiladores
#define TEMP_ALARM_2    4 // Temperatura muy alta, ventiladores
   activados y movil apagado
#define TEMP_ERROR      5 // Error de lectura del sensor de temperatura
/**
 * Constructor de la clase TemperatureControl
 */
TemperatureControl::TemperatureControl() {
        // TODO Auto-generated constructor stub

        std::cout << "TemperatureControl Constructor" << std::endl;

}
```

```
/**
 * Cambio de estado del control de temperatura
 */
void TemperatureControl::TempChangeState(uint32_t State)
{

switch(State) {

        case TEMP_IDDLE:
                // Temperatura normal, LED azul
                this->setLED_Value(LED_TEMP, BLUE);
                break;

        case TEMP_NORMAL:
                // Temperatura normal, LED verde
                this->setLED_Value(LED_TEMP, GREEN);

                // Apagar ventiladores
                this->setGPIO_Value(FANOUT_1, 0);
                this->setGPIO_Value(FANOUT_2, 0);

                // Encender el movil
                this->setGPIO_Value(EN_5V_USB_MOB, 1);
                this->setGPIO_Value(EN_4V2, 1);
                break;
        case TEMP_ALARM_1:
                // Encender LED amarillo
                this->setLED_Value(LED_TEMP, YELLOW);

                // Encender ventiladores
                this->setGPIO_Value(FANOUT_1, 1);
                this->setGPIO_Value(FANOUT_2, 1);

                // Encender el movil
                this->setGPIO_Value(EN_5V_USB_MOB, 0);
                this->setGPIO_Value(EN_4V2, 0);
                break;
        case TEMP_ALARM_2:
                // Encender LED rojo
                this->setLED_Value(LED_TEMP, RED);

                // Encender ventiladores
                this->setGPIO_Value(FANOUT_1, 1);
                this->setGPIO_Value(FANOUT_2, 1);

                // Apagar el movil
                this->setGPIO_Value(EN_5V_USB_MOB, 1);
                this->setGPIO_Value(EN_4V2, 1);

                break;


```

```
80          case TEMP_ERROR:
81                  // Encender LED blanco
82                  this->setLED_Value(LED_TEMP, WHITE);
83
84                  break;
85
86          default:
87                  break;
88   }
89  }
90  /**
91   * Maquina de estados del control de temperatura
92   */
93  void TemperatureControl::TempStateMachine()
94  {
95
96  float TempData = 0;
97
98  char direction[4] = "out";
99
100 uint32_t old_state;
101
102 old_state = TempState;
103
104 while(TempThread_isActive){
105
106  switch(TempState){
107
108    case TEMP_IDDLE:
109
110     if(!this->LoadLibrary(WSEN_TIDS) && !this->LoadLibrary(PCA9532) &&
        !this->LoadLibrary(GPIO) && !this->setLED_Value(LED_TEMP, BLUE)
        ){
111
112              TempState = TEMP_INITIALIZE;
113         }
114
115                 break;
116    case TEMP_INITIALIZE:
117
118          std::cout << "\n-DAEMON TempControl state READY \n" << std::
              endl;
119
120          //GPIOs de los ventiladores
121          this->configGPIO(FANOUT_1, direction);
122          this->configGPIO(FANOUT_2, direction);
123
124          //GPIOs de la alimentacion del terminal
125          this->configGPIO(EN_5V_USB_MOB, direction);
126          this->configGPIO(EN_4V2, direction);
127
128          this->WSEN_TIDS_Initialize();
129          this->PCA9532_Initialize();
```

```
130
131         TempState = TEMP_NORMAL;
132         break;
133
134     default:
135
136         std::cout << "\n-DAEMON TempControl thread \n" << std::endl;
137
138         // Lee el valor de temperatura del sensor
139         if(this->WSEN_TIDS_getTemperature(&TempData) == NO_ERROR){
140                 //TempData = temperatura; // DEBUG
141
142                 if(TempData < 40){
143                         // Temperatura normal
144                         TempState = TEMP_NORMAL;
145                 }else if(TempData >= 40 && TempData < 50){
146                         // Temperatura elevada. Alarma nivel 1
147                         TempState = TEMP_ALARM_1;
148                 }else{
149                         // Temperatura muy elevada. Alarma nivel 2
150                         TempState = TEMP_ALARM_2;
151                 }
152         }else{
153                 // Error de lectura del sensor. No se pueden leer datos
                        del sensor
154                 std::cout << "\nERROR reading Temperature sensor \n" <<
                        std::endl;
155                 TempState = TEMP_ERROR;
156         }
157
158         break;
159  }
160
161  // Comprueba si se tiene que cambiar de estado
162  if(TempState != old_state){
163
164         this->TempChangeState(TempState);
165         old_state = TempState;
166 }
167  // El hilo se duerme durante 1 segundo
168  std::this_thread::sleep_for(std::chrono::seconds(1));
169  }
170 }
171
172
173 /**
174  * Destructor de la clase TemperatureControl
175  */
176 TemperatureControl::~TemperatureControl() {
177         // TODO Auto-generated destructor stub
178 }
```

```
/*
 * PowerControl.h
 *
 *  Created on: Nov 29, 2021
 *      Author: bernardo
 */

#ifndef POWERCONTROL_H_
#define POWERCONTROL_H_

#include <iostream>

#include "SharedLibraries.h"

#include "../../app_includes/app_typedef.h"

class PowerControl: public SharedLibraries {
public:
        PowerControl();
        virtual ~PowerControl();

        void  PowerStateMachine();
        void PowerChangeState(uint32_t);
};

#endif /* POWERCONTROL_H_ */
```

```
/*
 * PowerControl.cpp
 *
 *  Created on: Nov 29, 2021
 *      Author: bernardo
 */

#include "PowerControl.h"

extern std::mutex mutex_hardware;
extern uint32_t PowerThread_ON;
extern uint32_t PowerState;

/*

USB, BATERY, PERIPHERALS, SOM

OK OK OK OK - VERDE // Todos los voltajes funcionan correctamente
OK OK OK NOK - YELLOW // Falla la alimentacion de los perifericos
OK OK NOK OK - YELLOW // Falla la alimentacion de los perifericos
OK OK NOK NOK - YELLOW // Falla la alimentacion de los perifericos
OK NOK OK OK - PURPLE // Falla la alimentacion del movil
OK NOK OK NOK - CYAN // Falla la alimentacion del MOVIL y los
   PERIFERICOS
OK NOK NOK OK - CYAN // Falla la alimentacion del MOVIL y los
```

```
   PERIFERICOS
OK NOK NOK NOK - CYAN // Falla la alimentacion del MOVIL y los
   PERIFERICOS
NOK OK OK OK - PURPLE // Falla la alimentacion del movil
NOK OK OK NOK - CYAN // Falla la alimentacion del MOVIL y los
   PERIFERICOS
NOK OK NOK OK - CYAN // Falla la alimentacion del MOVIL y los
   PERIFERICOS
NOK OK NOK NOK - CYAN // Falla la alimentacion del MOVIL y los
   PERIFERICOS
NOK NOK OK OK - PURPLE // Falla la alimentacion del movil
NOK NOK OK NOK - CYAN   // Falla la alimentacion del MOVIL y los
   PERIFERICOS
NOK NOK NOK OK - CYAN // Falla la alimentacion del MOVIL y los
   PERIFERICOS
NOK NOK NOK NOK - RED // Falla todas las alimentaciones

Reset   -> LED_OFF              POWER_IDDLE
Loaded  -> BLUE                 POWER_INITIALIZE

(00 00) -> VERDE                POWER_READY (Todo ceros)
(11 11) -> RED                  POWER_FAULT_ALL_SOURCES (Todo unos)

(00 X1) -> PURPLE               POWER_FAULT_TERMINAL
(X1 00) -> YELLOW               POWER_FAULT_PERIPHERALS
(X1 X1) -> CYAN                 POWER_FAULT_TERMINAL_AND_PERIPHERALS

Bad Read -> WHITE               POWER_ERROR_READ

*/

#define POWER_IDDLE         0 // Librerias sin cargar
#define POWER_INITIALIZE    1 // Sensor de voltaje inicializado
#define POWER_READY         2 // Lectura correcta de todos los voltajes
#define POWER_FAULT_ALL_SOURCES 3 // Fallo en todas las alimentaciones
#define POWER_FAULT_PERIPHERALS 4 // Fallo en la alimentacion de los
   perifericos
#define POWER_FAULT_TERMINAL 5 // Fallo en la alimentacion de terminal
#define POWER_FAULT_TERMINAL_AND_PERIPHERALS 6 // Fallo en la
   alimentacion de terminal y perifericos
#define POWER_ERROR_READ     7 // Fallo lectura del sensor



PowerControl::PowerControl() {
        // TODO Auto-generated constructor stub

        std::cout << "PowerControl Constructor" << std::endl;

}


```

```
69  void PowerControl::PowerChangeState(uint32_t State)
70  {
71
72          switch(State){
73
74                  case POWER_READY: // Lectura correcta de todos los
                        voltajes (0x0000)
75                          this->setLED_Value(LED_PWR, GREEN);
76                  break;
77                  case POWER_FAULT_ALL_SOURCES: // 0x1111
78                          this->setLED_Value(LED_PWR, RED);
79                  break;
80                  case POWER_FAULT_PERIPHERALS: // 0x01
81                          this->setLED_Value(LED_PWR, PURPLE);
82                          break;
83                  case POWER_FAULT_TERMINAL: // 0x10
84                          this->setLED_Value(LED_PWR, YELLOW);
85                          break;
86                  case POWER_FAULT_TERMINAL_AND_PERIPHERALS: // 0x11
87                          this->setLED_Value(LED_PWR, CYAN);
88                          break;
89                  case POWER_ERROR_READ:
90                          this->setLED_Value(LED_PWR, WHITE);
91                          break;
92                  default:
93                          break;
94          }
95  }
96
97  void PowerControl::PowerStateMachine()
98  {
99   PAC1932_struct PowerData = {0};
100
101  uint32_t power_error_code = 0;
102  uint32_t old_state;
103
104  old_state = PowerState;
105
106  while(PowerThread_isActive){
107
108    switch(PowerState){
109
110         case POWER_IDDLE:
111
112         if(!this->LoadLibrary(PAC1932) && !this->LoadLibrary(PCA9532)
               && !this->LoadLibrary(GPIO) && !this->setLED_Value(LED_PWR,
                BLUE)){
113
114             PowerState = POWER_INITIALIZE;
115             std::cout << "\n-DAEMON PowerControl state READY \n" <<
                        std::endl;
116         }
117
```

```
118         break;
119
120         case POWER_INITIALIZE:
121
122                 if(!this->PAC1932_Initialize() && !this->
                        PCA9532_Initialize()){
123
124                         PowerState = POWER_READY;
125                 }
126
127         break;
128
129         default:
130
131         std::cout << "\n-DAEMON PowerControl Thread: \n" << std::endl;
132
133         if(!this->PAC1932_GetAllValues(&PowerData)){
134
135                 power_error_code = 0x0000;
136
137         // Se comprueban si los voltajes de los perifericos y del
                terminal estan dentro del rango de lecturas correctas. En
                caso de que
138         // alguno de ellos no este dentro del rango se marca 1 bandera
                de error
139            if(PowerData.USB_Connector.Voltage < 4.8 || PowerData.
                   USB_Connector.Voltage > 5.0){
140
141                         power_error_code |= 0x0001;
142                 }
143                 if(PowerData.Terminal_Battery.Voltage < 4.1 ||
                        PowerData.Terminal_Battery.Voltage > 4.3){
144
145                         power_error_code |= 0x0010;
146                 }
147                 if(PowerData.Peripherals.Voltage < 3.1 || PowerData.
                        Peripherals.Voltage > 3.3){
148
149                         power_error_code |= 0x0100;
150                 }
151                 if(PowerData.SOM.Voltage < 3.1 || PowerData.SOM.Voltage
                         > 3.3){
152
153                         power_error_code |= 0x1000;
154                 }
155
156                 // Si todo esta bien (0x0000) se ignora esta
                        comprobacion
157                 // Si todo esta mal (0x1111) se ignora esta
                        comprobacion
158                 // Si hay algo mal, se obtiene el codigo de error (0x01
                        , 0x10 o 0x11)
159
```

```
160                 if(power_error_code != 0x0000 && power_error_code != 0
                        x1111){
161
162                     power_error_code = (power_error_code & 0x0001) | ((
                            power_error_code & 0x0010) >> 8) | ((
                            power_error_code & 0x0100) >> 8) | ((
                            power_error_code & 0x1000) >> 16);
163                 }
164
165                 // Se actualiza el estado del control de potencia en
                        funcion del codigo de error obtenido
166                 if(power_error_code == 0x0000){
167
168                         PowerState = POWER_READY;
169
170                 }else if(power_error_code == 0x1111){
171
172                         PowerState = POWER_FAULT_ALL_SOURCES;
173
174                 }else if(power_error_code == 0x01){
175
176                     PowerState = POWER_FAULT_PERIPHERALS;
177
178                 }else if(power_error_code == 0x10){
179
180                         PowerState = POWER_FAULT_TERMINAL;
181
182                 }else if(power_error_code == 0x11){
183
184                         PowerState =
                                POWER_FAULT_TERMINAL_AND_PERIPHERALS;
185
186                 }else{
187
188                         PowerState = POWER_ERROR_READ;
189                 }
190         }else{
191                 std::cout << "\nERROR reading Power sensor \n" << std::
                        endl;
192                 // Encender LED blanco
193                 this->setLED_Value(LED_PWR, WHITE);
194         }
195
196                         std::cout << "\n" << std::endl;
197                         break;
198         }
199
200         // Comprueba si se tiene que cambiar de estado
201         if(PowerState != old_state){
202
203                 this->PowerChangeState(PowerState);
204                 old_state = PowerState;
205         }
```

```
206
207         std::this_thread::sleep_for(std::chrono::seconds(1));
208  }
209 }
210 /**
211  * Destructor de la clase PowerControl
212  */
213 PowerControl::~PowerControl() {
214         // TODO Auto-generated destructor stub
215 }
```

```
1  /*
2   * AndroidControl.h
3   *
4   *  Created on: Mar 4, 2022
5   *      Author: bernardo
6   */
7
8  #ifndef ANDROIDCONTROL_H_
9  #define ANDROIDCONTROL_H_
10
11
12 #include <iostream>
13 #include <mutex>
14 #include <thread>
15 #include <string>
16
17 #include <sys/socket.h>
18 #include <netinet/in.h>
19 #include <arpa/inet.h>
20 #include <stdio.h>
21 #include <unistd.h>
22 #include <netdb.h>
23 #include <stdlib.h>
24 #include <stdint.h>
25
26 #include "json.hpp"
27
28 using namespace std;
29 using namespace nlohmann;
30
31 #define MAX 600
32 #define PORT 5555
33 #define SA struct sockaddr
34
35
36 #define ADB_IDDLE               0
37 #define ADB_INITIALIZE          1
38 #define ADB_CONNECTING          2
39 #define ADB_ESTABLISHED         3
40 #define ADB_CLOSE_SOCKET        4
41
```

```
42  struct support_system_info {
43
44          string serial_no;
45          string local_version;
46  };
47
48  struct android_app {
49
50          int interfaces;
51          string metrics_version;
52          string instrumentation_version;
53          string script_version;
54  };
55
56  struct android_update {
57
58          string  path;
59          string  md5;
60          string  version;
61  };
62
63  struct android_device {
64
65          string id_sonda;
66          string serial_no;
67          string model;
68  };
69
70
71  struct android_message {
72
73          int type;
74          int message_id;
75
76          json json_message;
77
78          struct android_app metrics_app;
79          struct android_update android_dev_update;
80          struct android_device android_dev;
81  };
82
83  class AndroidControl {
84  public:
85
86          struct android_message android_data;
87
88          AndroidControl();
89          virtual ~AndroidControl();
90          void android_data_initialize();
91          void change_state_communication(uint32_t state);
92          void communication_state_machine();
93          int create_socket(int *);
94          int connect_socket(struct sockaddr_in *, int *);
```

```
95          int listen_socket(int *);
96          int send_socket(int * sockfd, const void * data, int lenght);
97          int close_socket(int *);
98          int save_message_info();
99  };
100
101
102 #endif /* ANDROIDCONTROL_H_ */
```

```
1   /*
2    * AndroidControl.cpp
3    *
4    *  Created on: Mar 4, 2022
5    *      Author: bernardo
6    */
7   #include "AndroidControl.h"
8   #include <thread>
9
10  using namespace std;
11
12  AndroidControl::AndroidControl() {
13          // TODO Auto-generated constructor stub
14          std::cout << "AndroidControl Constructor" << std::endl;
15  }
16  AndroidControl::~AndroidControl() {
17          // TODO Auto-generated destructor stub
18  }
19  /**
20   * Maquina de estados de la comunicacion con el movil Android
21   */
22  void AndroidControl::communication_state_machine(void)
23  {
24   int sockfd = 0;
25   struct sockaddr_in servaddr;
26   int communication_state = ADB_IDDLE;
27   char command[100];
28
29   while(AndroidThread_isActive){
30
31   switch(communication_state){
32
33   case ADB_IDDLE:
34
35          this->android_data_initialize();
36
37          if(!this->LoadLibrary(PCA9532) && !this->PCA9532_Initialize()
               && !this->setLED_Value(LED_ANDROID, BLUE)){
38
39                  communication_state = ADB_INITIALIZE;
40          }
41   break;
42
```

```
43  case ADB_INITIALIZE:
44
45          /* command contiene la string del comando a ejecutar (a
                character array) */
46          bzero(command, sizeof(command));
47          sprintf(command, "adb devices");
48          // Ejecuta "adb devices"
49          popen(command,"r");
50
51          sleep(3);
52
53          bzero(command, sizeof(command));
54          sprintf(command, "adb forward tcp:%d tcp:%d", PORT, PORT);
55          // Ejecuta el comando
56          popen(command,"r");
57
58          sleep(1);
59          // Apertura del socket
60          if(this->create_socket(&sockfd) == NO_ERROR){
61
62          this->setLED_Value(LED_ANDROID, WHITE);
63                  communication_state = ADB_CONNECTING;
64          }
65  break;
66  case ADB_CONNECTING:
67          // Conexion al socket
68          if(this->connect_socket(&servaddr, &sockfd) == NO_ERROR){
69
70                  this->setLED_Value(LED_ANDROID, GREEN);
71                  communication_state = ADB_ESTABLISHED;
72          }else {
73                  communication_state = ADB_INITIALIZE;
74                  sleep(3);
75          }
76  break;
77  case ADB_ESTABLISHED:
78          // Escucha el socket. Lo cierra en caso de perder la
                comunicacion
79          if(this->listen_socket(&sockfd) != NO_ERROR){
80
81                  this->setLED_Value(LED_ANDROID, WHITE);
82                  communication_state = ADB_CLOSE_SOCKET;
83          }
84  break;
85  case ADB_CLOSE_SOCKET:
86          // Cierra el socket
87          if(this->close_socket(&sockfd) == NO_ERROR){
88
89                  communication_state = ADB_INITIALIZE;
90
91                  printf("[DAEMON]\t\tTrying reconnect to server...\n");
92          }
93  break;
```

```
 default:
 break;
  }
 // Comprueba si el portaSIM esta conectado
 if(this->getGPIO_Value(PORTASIM_PRES)){
        // Si se lee un 1, el portaSIM esta desconectado. Led blanco.
        this->setLED_Value(LED_SIM, WHITE);
 }else{
        // Si se lee un 0, el portaSIM esta conectado. Led verde.
        this->setLED_Value(LED_SIM, GREEN);
 }
 usleep(200000);
 }
}

/**
 * Inicializacion de la estructura que contiene la informacion del
    movil Android
 */
void AndroidControl::android_data_initialize()
{

this->android_data.type         = 0;
this->android_data.message_id   = 0;

this->android_data.metrics_app.interfaces               = 0;
this->android_data.metrics_app.instrumentation_version  = "0.0";
this->android_data.metrics_app.metrics_version          = "0.0";
this->android_data.metrics_app.script_version           = "0.0";

this->android_data.android_dev.id_sonda         = "S/N";
this->android_data.android_dev.model            = "S/N";
this->android_data.android_dev.serial_no        = "S/N";

this->android_data.android_dev_update.md5       = "S/N";
this->android_data.android_dev_update.path      = "S/N";
this->android_data.android_dev_update.version   = "S/N";
}

/**
 * Cierre del socket
 */
int AndroidControl::close_socket(int * sockfd){

        if(close(*sockfd)){

                return APP_REPORT(APP_DAEMON, CLOSING_SOCKET);
        }else {

                return NO_ERROR;
        }
}

```

```
146 /**
147  * Creacion del socket
148  */
149 int AndroidControl::create_socket(int * sockfd){
150 
151         // socket create and verification
152         *sockfd = socket(AF_INET, SOCK_STREAM, 0);
153         if (*sockfd == -1) {
154                 printf("[DAEMON]\t\tSocket creation failed...\n");
155                 return APP_REPORT(APP_DAEMON, OPENING_SOCKET);
156         }
157         else{
158                 printf("[DAEMON]\t\tSocket successfully created...\n");
159                 return NO_ERROR;
160         }
161 }
162 
163 /**
164  * Conexion del socket
165  */
166 int AndroidControl::connect_socket(struct sockaddr_in * servaddr, int *
        sockfd){
167 
168         bzero(servaddr, sizeof(*servaddr));
169         // assign IP, PORT
170         servaddr->sin_family = AF_INET;
171         servaddr->sin_addr.s_addr = inet_addr("127.0.0.1");
172         servaddr->sin_port = htons(PORT);
173 
174         // connect the client socket to server socket
175         if (connect(*sockfd, (SA*)servaddr, sizeof(*servaddr)) != 0) {
176                 printf("[DAEMON]\t\tConnection with server failed...\n"
                        );
177                 return APP_REPORT(APP_DAEMON, CONNECTING_SOCKET);
178         }else{
179                 printf("[DAEMON]\t\tConnected to server... Waiting
                        response...\n");
180                 return NO_ERROR;
181         }
182 }
183 
184 /**
185  * Enviar a traves del socket
186  */
187 int AndroidControl::send_socket(int * sockfd, const void * data , int
       lenght)
188 {
189         // Escribe datos a traves del socket
190         if (write(*sockfd, data, lenght) == lenght){
191                 printf("[DAEMON]\t\tWrite socket = %s \n", (char *)
                        data);
192                 return NO_ERROR;
193         } else {
```

```
194                 perror("[DAEMON]\t\tWrite error: ");
195                 return APP_REPORT(APP_DAEMON, WRITING_SOCKET);
196         }
197 }
198 /**
199  * Escucha los mensajes que llegan al socket
200  */
201 int AndroidControl::listen_socket(int * sockfd)
202 {
203 char buff[MAX];
204 static int reintentos = 0;
205 int nbytes = 0;
206
207 fd_set rfds;
208 struct timeval tv;
209
210 tv.tv_sec = 3;
211 tv.tv_usec = 0;
212
213 bzero(buff, sizeof(buff));
214
215 FD_ZERO(&rfds);
216 FD_SET(*sockfd, &rfds);
217
218 int retval = select(*sockfd + 1, &rfds, NULL, NULL, &tv);
219
220 if(retval > 0){
221
222  usleep(100000);
223  nbytes = read(*sockfd, buff, sizeof(buff));
224
225  if(nbytes == 0){
226
227    if(reintentos++ > 3){
228
229          return CONNECTION_LOST;
230    }
231  sleep(1);
232
233  }else{
234
235  reintentos = 0;
236
237  try{
238
239  this->android_data.json_message = nlohmann::json::parse(buff);
240  this->save_message_info();
241
242  }catch(const std::exception& e){
243
244   printf("[DAEMON]\t\tJSON Parse error, received from server : %s",
        buff);
245  }
```

```
246  }
247  }else if(retval == 0){
248   //printf("Timeout! \n"); // do nothing
249  }else{
250
251   perror("[DAEMON]\t\tError select:  \n");
252   return APP_REPORT(APP_DAEMON, SELECT_SOCKET);
253  }
254
255 return NO_ERROR;
256 }
257 /**
258  * Guarda la informacion del mensaje JSON recibida
259  */
260 int AndroidControl::save_message_info()
261 {
262
263 char command[100];
264
265 printf("[DAEMON]\t\tJSON Message received from server \n");
266
267 this->android_data.type = this->android_data.json_message["header"]["
    type"];
268 this->android_data.message_id = this->android_data.json_message["header
    "]["message_id"];
269
270 switch(this->android_data.type){
271
272 case 1:
273         printf("Type 1: Android Info \n");
274
275         // Guarda la informacion del movil Android
276         this->android_data.android_dev.id_sonda = this->android_data.
             json_message["body"]["android_dev"]["id_sonda"];
277         this->android_data.android_dev.serial_no = this->android_data.
             json_message["body"]["android_dev"]["serial_no"];
278         this->android_data.android_dev.model = this->android_data.
             json_message["body"]["android_dev"]["model"];
279
280         this->android_data.metrics_app.interfaces = this->android_data.
             json_message["body"]["metrics_app"]["interfaces"];
281         this->android_data.metrics_app.instrumentation_version = this->
             android_data.json_message["body"]["metrics_app"]["versions"
             ]["instrumentation"];
282         this->android_data.metrics_app.metrics_version
                     = this->android_data.json_message["body"]["
             metrics_app"]["versions"]["metrics"];
283         this->android_data.metrics_app.script_version
                     = this->android_data.json_message["body"]["
             metrics_app"]["versions"]["script"];
284
285         cout << "Android Device: " << endl << "\t\tID Sonda: " << this
             ->android_data.android_dev.id_sonda << endl << "\t\tSerial
```

```
            number: " << this->android_data.android_dev.serial_no <<
            endl << "\t\tDevice model: " << this->android_data.
            android_dev.model << endl << endl; cout << "Metrics App: "
            << endl << "\t\tInterfaces: " << this->android_data.
            metrics_app.interfaces << endl << "\t\tInstrumentation
            version: " << this->android_data.metrics_app.
            instrumentation_version << endl <<"\t\tMetrics version: "
            << this->android_data.metrics_app.metrics_version << endl
            <<
286             "\t\tScript version: " << this->android_data.
                   metrics_app.script_version << endl << endl;
287         break;
288
289 case 2:
290         printf("Type 2: Android update available \n");
291
292         // Guarda la informacion de la actualizacion
293         this->android_data.android_dev_update.path = this->android_data
               .json_message["body"]["path"];
294         this->android_data.android_dev_update.md5 = this->android_data.
               json_message["body"]["md5"];
295         this->android_data.android_dev_update.version = this->
               android_data.json_message["body"]["version"];
296
297         cout << "Android Update available: " << endl << "\t\tpath: " <<
                this->android_data.android_dev_update.path << endl << "\t\
               tmd5: " << this->android_data.android_dev_update.md5 <<
               endl << "\t\tversion: " << this->android_data.
               android_dev_update.version << endl << endl;
298
299         // Descarga el fichero de actualizacion del movil Android en el
                directorio /home
300         bzero(command, sizeof(command));
301         sprintf(command, "adb pull %s /home/", this->android_data.
               android_dev_update.path.c_str());
302         // Ejecuta el comando
303         popen(command,"r");
304         // Lanza la actualizacion
305         system("/home/update_android/install.sh");
306
307         break;
308
309 case 3:
310         printf("Type 3: Reboot Android device \n");
311         // Reinicia el movil Android
312         system("adb reboot");
313         break;
314
315 case 4:
316         printf("Type 4: Reboot system support \n");
317         // Reinicia el sistema de soporte
318         system("reboot");
319         break;
```

```

default:
        printf("Another type \n");
        break;
 }

 return 0;
}
```

```
/*
 * CanControl.h
 *
 *  Created on: Mar 7, 2022
 *      Author: bernardo
 */

#ifndef CANCONTROL_H_
#define CANCONTROL_H_

#include "TemperatureControl.h"

class CanControl: public TemperatureControl {
public:
        CanControl();
        virtual ~CanControl();

        void CanStateMachine();
};

#endif /* CANCONTROL_H_ */
```

```
/*
 * CanControl.cpp
 *
 *  Created on: Mar 7, 2022
 *      Author: bernardo
 */
#include "CanControl.h"

extern uint32_t CanThread_isActive;
extern uint32_t CanState;

#define CAN_IDDLE               0
#define CAN_INITIALIZE          1
#define CAN_CONFIGURE           2
#define CAN_LISTEN              3
#define CAN_ERROR               4
// Constructor de la clase CanControl
CanControl::CanControl() {
        // TODO Auto-generated constructor stub
}
```

```
/*
 * Maquina de estados de la comunicacion CAN
 */
void CanControl::CanStateMachine()
{
 struct can_message can_data;

 while(CanThread_isActive){

  switch(CanState){

        case CAN_IDDLE:
                if(!this->LoadLibrary(CAN)){

                        CanState = CAN_INITIALIZE;
                }
                break;
        case CAN_INITIALIZE:
                // Inicializa el bus CAN
                if(!this->CAN_Initialize(&can_data)){

                        CanState = CAN_CONFIGURE;
                }
                break;
        case CAN_CONFIGURE:
                // Configura el bus CAN
                if(!this->CAN_Configure(&can_data)){

                        CanState = CAN_LISTEN;
                }else{

                        CanState = CAN_INITIALIZE;
                }
                break;
        case CAN_LISTEN:
                // Escucha mensajes del bus CAN
                if(this->CAN_Receive){ // Si se detecta algun error, se
                    reinicia el bus

                        CanState = CAN_INITIALIZE;
                }
                break;
        default:
                        CanState = CAN_IDDLE;
                break;
 }
 std::this_thread::sleep_for(std::chrono::seconds(1));
 }
}
```

Volver