

UNIVERSITAT OBERTA DE CATALUNYA

GRAU DE TECNOLOGIES DE TELECOMUNICACIÓ



**DISSENY E IMPLEMENTACIÓ D'UNA
INTERFICIE D'ADMINISTRACIÓ WEB PER A
SMARTCITIZEN KIT**

TREBALL DE FINAL DE GRAU

Autor: **Alejandro Castro Vílchez**

Consultor: **Pere Tuset**

Junio-2015

Agraïments

A tots els meus amics i família per tots els ànims donats durant tota la carrera i especialment aquest últim trajecte. Gràcies Manolo i Carlos per totes les xerrades d'ànim.

I especialment, a Cristina, per estar sempre al meu costat als moments més difícils, aguantant el meu estrès i per haver-se esforçat, fins i tot més que jo, aquest últim mes.

- Alejandro Castro -

RESUM

El present projecte proposa el desenvolupament d'un sistema de gestió web que permeti la configuració i visualització de paràmetres de la placa SmartCitizen (SCK), juntament amb altres paràmetres de connectivitat WIFI mitjançant la creació d'un servidor web i la implementació d'un mode Soft AP que permeti la connexió directa de dispositius amb connexió sense fils. Aquestes funcions son implementades com a colapps dins del mòdul Wi-fi RTX4100.

El SmartCitizen Kit DVK esta format per una placa Arduino Due i el Shield Arduino, SmartCitizen DVK, dissenyat per ajudar als desenvolupadors a la creació de noves aplicacions per la nova placa SmartCitizen Kit 2.0. Aquest kit inclou el mòdul wifi RTX4100, que permet el desenvolupament d'aplicacions pròpies (colapps), a part d'altres característiques que s'analitzaran en aquest document, on es descriuran els avantatges de l'actual kit amb l'anterior model, SCK 1.1.

Per d'altre banda, s'analitzarà el firmware Arduino, el qual ens permet la visualització de temperatures, la carga de colapps i firmware al mòdul RTX. Una de les funcions a desenvolupar és la implementació de comandes per activar el Mode Soft AP i Servidor web, a part de l'enviament de temperatures al mòdul wifi mitjançant la comunicació SPI. Per tant, s'haurà d'implementar una nova llibreria que implementi aquestes funcions, tant per Arduino, com pel mòdul RTX.

Finalment, el disseny de la colapp del Servidor Web, recollirà les temperatures dels sensors i proporcionarà els paràmetres de configuració del sistema mitjançant el disseny d'una interfície web, on l'usuari podrà visualitzar o configurar els paràmetres del dispositiu. Aquesta es realitzarà amb Html i, es analitzarà les limitacions del mòdul RTX per poder inserir altres tipus de codi de programació, com Css o Javascript.

Paraules claus: WI-FI, SmartCitizen Kit, RTX4100, Soft AP, Servidor Web, colapp, Protothreads, SPI

ABSTRACT

This project proposes the development of a web management system that allows the configuration and display settings of SmartCitizen Kit (SCK), also other parameters like WIFI parameters through the implementation of a web server Soft AP mode that allows direct connection of devices with wireless connection. These functions are implemented like colapps within RTX4100 Wi-fi module.

The SmartCitizen Kit DVK is formed by ArduinoDue Databoard and SmartCitizen DVK Arduino Shield. This was designed to help developers for create new applications for the new board SmartCitizen Kit 2.0. This kit includes the RTX4100 wifi module, which allows the development of own applications (colapps), and other features that will be analyzed in this document, which described the advantages of this kit with the previous model, SCK 1.1.

On the other hand, will analyze the Arduino firmware, which allows the visualization of temperatures, loading firmware and colapps to RTX. One of the functions to develop is the implementation of a comand to trigger Soft AP mode and Web server and send temperatures through communication SPI to the RTX chip. Therefore, it's necessary develop a new library that implements these functions, both for Arduino as the RTX module.

Finally, design a Web Server colapp that read temperature sensors and provide the configuration parameters of the system, and design a web interface where the user can view or configure device settings. This will be programming with HTML, and it will analyze the limitations of RTX module to implement other types of programming web code, such as JavaScript or CSS.

Keywords: WI-FI, SmartCitizen Kit, RTX4100, Soft AP, Servidor Web, colaapp, Protothreads, SPI

Índex de continguts

1	Introducció	10
1.1	Justificació del projecte	10
1.2	Objectius	12
1.3	Beneficis	13
1.4	Organització de la memòria	14
1.5	Metodologia	15
2	Estat de l'art	16
2.1	SCK 1.1 – RN131	16
2.2	SmartCitizen kit DVK	18
2.2.1	Arduino DUE	18
2.2.2	SmartCitizen DVK shield	20
2.2.3	Modes de funcionament	23
2.3	Comunicació SPI	24
2.3.1	Polaritat del rellotge i fase	25
2.4	Programació c++ -Protothreads-	27
2.5	Protocol HTTP	29
2.6	Llenguatges de programació web	31
2.7	Anàlisi Software RTX4100	31
2.7.1	Bloc Co-Located Application (Colapps)	32
2.7.2	Bloc Mòdul Firmware	32
2.7.3	Framework de les Aplicacions	33
2.7.4	El sistema operatiu de RTX (ROS)	34
2.7.5	API AmelieSDK	35
2.8	Anàlisi COLApp SmartCitizen-RTX4100	36
3	Preparació de l'entorn de desenvolupament	38
3.1	Software de Desenvolupament RTX:	38
3.1.1	Creació y edició	40
3.1.2	Compilació	41
3.1.3	Carrega (Loading)	42
3.1.4	Depuració (Debugging)	44
3.1.5	Carrega de Firmware RTX	46
3.2	Arduino IDE	48

3.2.1	Llibreries Arduino.....	50
4	Proces de Desenvolupament	52
4.1	Mode Soft Ap.....	53
4.1.1	Llibreries utilitzades	53
4.1.2	Desenvolupament.....	54
4.2	Servidor Web.....	55
4.2.1	Llibreries utilitzades	56
4.2.2	Proves: Desenvolupament del servidor web amb EASY WEB.....	58
4.3	Desenvolupament del servidor web final.....	61
4.3.1	Creació de pàgines Html	61
4.3.2	Interpretació de les peticions HTTP client-servidor	63
4.3.3	Implementació de formularis.....	64
4.3.4	Iniciació del Servidor web	65
4.4	Comunicació SPI Arduino-RTX.....	66
4.4.1	Llibreries Arduino-SPI.....	66
4.4.2	Modificació Firmware Arduino (Master).....	67
4.4.3	Llibreries SPI RTX	69
4.4.4	RTX (Slave) -Recepció de temperatures i comandes SPI-.....	70
5	Fase de proves	72
5.1	Proves	72
5.2	Problemes i solucions.....	73
6	Conclusio	75
6.1	Futurs treballs.....	76
7	Glossari	77
8	Bibliografia	79
9	Annex	81
9.1	Diagrama de flux colapp Firmware SmartCitizenRTX4100	81
9.2	Diagrama de Flux Colapp final.....	82
9.3	Codi Scdk.cpp -SmartCitizen_SDK.ino-	83
9.4	Codi Colapp – SmartCitizenRTX4100-.....	90
9.5	Codi prova Server Web amb EASYWEB	110

Índex de figures

Figura 1. Aplicació Web SmartCitizen. www.SmartCitizen.me	11
Figura 2. Vista Frontal i posterior del SKC 1.1	16
Figura 3. Modul RN131 "Wifly"	17
Figura 4. Vista Frontal Arduino DUE	19
Figura 5. Vista frontal Modul RTX4100	20
Figura 6. Circuit intern mòdul Wi-fi RTX4100	21
Figura 7. Components SmartCitizen DVK Shield	22
Figura 8. Comunicació SPI Master-Slave.....	25
Figura 9. Diagrama de temps segons configuració CPOL I CPHA.....	26
Figura 10. Arquitectura Software RTX4100.....	32
Figura 11. Procés d'enviament de missatges i execució de protothreads	35
Figura 12. Comunicació Colapps amb l'API AmelieSDK	36
Figura 13. Creació del directori del projecte mitjançant la comanda CreateApp.bat.....	40
Figura 14. Compilació de la colapp	41
Figura 15. Exemple d'error de compilació	41
Figura 16. Configuració RTX EAI Port Server	42
Figura 17. ColaController: Error detecció del mòdul RTX.....	43
Figura 18. ColaController: Importació correcta de la colapp al mòdul	44
Figura 19. Aplicació Amelie SDK trace (RSX).....	45
Figura 20. Configuració de les ColaTask en RSX	45
Figura 21. Missatges Colatask en RSX.....	46
Figura 22 . Procés de Carga firmware RTX4100.....	47
Figura 23. Finalització carga firmware RTX4100.....	47
Figura 24. Drivers instal·lats Arduino DUE en port COM3.....	48
Figura 25. Instal·lació Core Arduino Due en l'aplicació Arduino IDE	49
Figura 26. Board Manager Arduino IDE.	49
Figura 27. Monitor Sèrie Arduino IDE	50
Figura 28. Llibrerias instal·lades Arduino IDE	51
Figura 29. Diagrama Flux SOFT AP	53
Figura 30. Diagrama flux Servidor Web	56
Figura 31. Demostració Servidor Web amb EASYWEB	60
Figura 32. Pagina Web Final. Visualització de Temperatures	62
Figura 33 Pagina Web Final. Informació del sistema	62
Figura 34. Pagina Web Final. Configuració de paràmetres SOFT AP	63
Figura 35. Enviament correcte de temperatures mitjançant comunicació SPI	68
Figura 36. Diagrama Flux Recepció de dades i comandes SPI.....	70

Índex de taules

Tabla 1. Mètodes petició HTTP.....	30
Tabla 2. Codis Resposta HTTP	30
Tabla 3. Comandes SPI firmware SmartCitizenRTX4100	37
Tabla 4. Lliberies AppWifi.h per implementar SOFT AP	53
Tabla 5. Estructura Header	56
Tabla 6. Estructura Body (Cos).....	57
Tabla 7. Estructura Recurs de Resposta HTTP	57
Tabla 8. Lliberies ApiHttp.h per implementar Servidor Web	58
Tabla 9. Lliberies AppWebConfig.H per implementar Servidor Web amb EASYWEB	59
Tabla 10. Modes d'enviament de dades SPI	67
Tabla 11. Lliberies DrvSpiSlave.h per implementar comunicació SPI en la colapp	70

1 INTRODUCCIÓ

En aquest projecte de final de grau, es descriu el disseny i el desenvolupament de noves funcionalitats al firmware del mòdul RTX4100 on es pretén implementar un sistema de gestió web que permeti la visualització i modificació de paràmetres del SmartCitizen Kit (SCK), juntament amb altres paràmetres de connectivitat Wifi i la visualització de temperatures dels sensors del SCK, mitjançant la implementació d'un mode Soft AP que permeti la connexió directa dels usuaris al dispositiu.

En aquest capítol, es descriuran els motius que han justificat l'elecció d'aquest projecte i els antecedents relacionats amb l'empresaproveïdora, SmartCitizen; els objectius a assolir, els beneficis del projecte i una breu descripció de l'organització de la memòria. Finalment s'analitza la metodologia i les principals etapes per realitzar aquest projecte.

1.1 JUSTIFICACIÓ DEL PROJECTE

L'objectiu d'aquest projecte de final de grau, es implementar i millorar noves funcionalitats a l'actual firmware del mòdul RTX4100 per els nous kits de sensors de SmartCitizen.

L'empresa SmartCitizen té com a objectiu fomentar la recollida de dades ambientals, com són els nivells de soroll, concentració de gasos, temperatures i humitat, a través de sensors de baix cost, permetent l'anàlisi d'aquestes dades i compartir-les amb la resta del món. Tots els usuaris interessats poden formar part de la xarxa de sensors i compartir les dades que es podran veure a través de la web del projecte des de qualsevol dispositiu amb accés a Internet, permetent la comparació de les dades entre diferents ciutats del món. Les dades recollides per aquests sensors són centralitzats en el sistema de SmartCitizen. Aquests dispositius poden ser utilitzats per qualsevol usuari a part d'empreses relacionades amb la millora mediambiental. Actualment s'està implementant una xarxa de sensors a la ciutat de Barcelona, però com es pot veure a la Web, hi ha milers de sensors repartits per tot el món



Figura 1. Aplicació Web SmartCitizen. www.SmartCitizen.me

La necessitat de SmartCitizen per millorar els actuals sensors SCK 1.1, ha permès el desenvolupament dels futurs kit de sensors, SCK 2.0. Actualment està en fase de desenvolupament, distribuint equips que permetin als desenvolupadors el disseny d'aplicacions que permetin millorar el futur firmware de la placa. Aquesta nova placa inclou el mòdul WIFI RTX4100, el qual permet dissenyar les seves pròpies aplicacions i gestionar-les des del propi mòdul, millorant els temps de resposta del conjunt i el consum energètic.

Gràcies al treball de final de grau desenvolupat l'anterior semestre per Miguel Colom, anomenat "“Analysis, Improvement, and Development of New Firmware for the Smart Citizen Kit Ambient Board “, va realitzar un ampli estudi sobre les característiques hardware i software de la placa SCK1.1, incloent millores al codi actual del firmware. També va realitzar un primer disseny del firmware del mòdul RTX4100 per a les noves plaques el qual es pot trobar en el repositori "“<https://github.com/fablabbcn/SmartCitizenRTX4100> “ . Aquest firmware desenvolupat amb l'API del propi mòdul permet l'execució de diverses comandes a través d'una consola de comandes, però també dóna l'opció d'executar-les mitjançant comandes SPI.

En aquest projecte es vol dissenyar un sistema de gestió web que permeti la visualització i modificació de paràmetres de la placa SmartCitizen SCK i la implementació d'un mode Soft AP que permetí la connexió directa al mòdul RTX des de qualsevol dispositiu amb connexió WIFI, millorant la connectivitat i les funcionalitats del

dispositiu. D'altra banda es vol millorar la comunicació SPI, en part ja implementada, millorant el sistema de comandes i l'enviament de dades entre els diferents mòduls.

1.2 OBJECTIUS

L'**objectiu principal** del projecte es expandir les funcionalitats de l'actual firmware per suportar un mode Soft AccesPoint que permeti a altres dispositius associar-se per facilitar la configuració i visualització dels paràmetres de la placa sense la necessitat d'utilitzar recursos externs, es a dir, millorar les prestacions del dispositiu i facilitar l'ús.

El desenvolupament del projecte serà principalment crear les funcions per implementar un mode SOFT AP i servidor web com una extensió de la colapp ja desenvolupada .

Per realitzar aquesta implementació s'haurà d'estudiar l'actual colapp per mantenir una arquitectura i una API semblant. Per un altre banda, s'haurà d'estudiar la documentació del fabricant, per entendre l'API de les aplicacions (Colapps). El software de desenvolupament de Colapps inclou exemples útils que poden ser utilitzats en aquest projecte.

A continuació es descriuran els **objectius específics**:

- Desenvolupament **mode SOFT AP** que permeti la connexió dels usuaris al mòdul. Aquest ha de suportar les següents funcionalitats:
 - Activació del mode SOFT AP mitjançant una comanda per SPI des de el SCK
 - Creació d'una xarxa oberta amb un SSID únic per kit que consisteix d'un denominador comú i un identificador únic (MAC)
 - Servidor DHCP per assignar IP's als dispositius que s'associïn
- Desenvolupament **deservidor web** que permeti la visualització i configuració dels paràmetres del SCK. Aquest servidor ha de suportar les següents funcionalitats
 - Activació del servidor web mitjançant una comanda per SPI des de el SCK
 - Suport de HTTP/1.1 que realitzi les funcionalitats bàsiques de GET i POST.

- Creació d'una única pagina html que pugui contenir distints formularis amb CSS i Javascript incrustats. S'ha de tenir en compte les limitacions d'espai en la flash de RTX4100 per poder desenvolupar-la
- Creació de formularis HTML que permeti enviar informació al mòdul, on la funcionalitat sigui configurar els paràmetres d'aquest.
- Complementar aquests funcionalitats, analitzant si hi ha possibilitat de incloure continguts estàtics mes pesats com imatges, contemplant les limitacions d'espai de la colapp.
- L 'interfície web ha de permetre visualitzar i configurar els següents paràmetres
 - Paràmetres de xarxa (SSID, Seguretat, clau)
 - Mostres des de de l'ultima lectura dels sensors de temperatura i humitat
- Desenvolupament d' una llibreria per el firmware SmartCitizen_DVK.ino que realitzi la **comunicació SPI** necessària per gestionar les funcionalitats dels modes SOFT API i Web Server i l'enviament de temperatures al mòdul RTX.
- Elaborar la **documentació de l'aplicació** desenvolupada amb els resultats de la investigació i les proves realitzades. Aquesta fase es intrínseca en el desenvolupament del projecte, i es necessari que la documentació realitzada sigui completa i clara, ja que el projecte pot ser utilitzat en un futur per tercers per la implantació de noves funcionalitats o millores

1.3 BENEFICIS

Els beneficis d'aquestes noves funcionalitats permetran als usuaris accedir al mòdul directament des de qualsevol dispositiu que disposi connexió WIFI, permeten la configuració i la visualització dels paràmetres del kit SmartCitizen, millorant la connectivitat i la interoperabilitat entre dispositius.

D'altre banda, l'avantatgedel chip RTX4100 es que permet la creació i modificació del firmware, i per tant la creació de nou contingut permet la utilització del chip sense modificar les característiques de la placa SCK, millorant així la funcionalitat del kit. A part, gràcies a la pròpia CPU del xip, els temps de resposta del SCK no es veuran afectats ja que les colapps son gestionades per el propi mòdul, millorant així les funcionalitats del

SCK. Per altre banda, el consum d'aquest xip es molt mes baix que el seu germà petit RN131 millorant el consum elèctric del kit.

1.4 ORGANITZACIÓ DE LA MEMÒRIA

Aquest document s'ha organitzat en sis capítols, a part dels annexos, glossari i bibliografia.

En aquest **primer capítol**, es fa una introducció de la memòria on es justifica els motius per realitzar el projecte, els objectius que es pretenen assolir i els futurs beneficis d'aquest. També es farà un resum de la metodologia i les etapes del projecte.

El **segon capítol** es farà un repàs sobre l'estat de l'art, realitzant un anàlisi i comparativa de la placa SCK 1.1 i el mòdul Wifi RN131 amb el SCDK que s'utilitzarà per desenvolupar el futur firmware per SCK 2.0. També es realitzarà un breu estudi de les tecnologies utilitzades per poder assolir el projecte: Comunicació SPI, programació de protothreads, llenguatges de programació WEB. Per últim, s'analitzarà el software del RTX4100, el qual permet crear les seves pròpies aplicacions, anomenades *colapps*.

El **tercer capítol** descriu l'entorn de desenvolupament, on serà necessari la instal·lació del software de desenvolupament del RTX (AmelieSDK) per desenvolupar/modificar el firmware del mòdul, i l'entorn Arduino IDE per modificar l'actual firmware de la placa. A part es descriurà el mètode per compilar, debuggar, i fer la carrega d'aplicacions i firmware al mòdul i a la placa Arduino.

El **quart capítol** es descriu el procés de disseny, desenvolupament i programació del projecte. Primer es descriurà el procés per implementar la comunicació SPI Master-Slave (Arduino-RTX) que permet l'enviament de comandes i la lectura de temperatures, a continuació es descriurà la programació del mode Soft AP i Servidor Web, en aquesta última part s'analitzarà un mode alternatiu per realitzar el Servidor WEB (Mètode Easy WEB), qual s'ha descartat per les seves limitacions. Finalment es realitzarà la integració amb el firmware actual del RTX (SmartCitizenRTX4100). Cadascuna de les parts s'inclourà un anàlisi de les llibreries utilitzades i les limitacions trobades.

El **cinquè capítol** descriu les proves de funcionament del sistema i els problemes que han sorgit durant el procés i les solucions trobades.

Amb les dades obtingudes al punt anterior, es detallarà al **capítol sis** les conclusions obtingudes i els futurs treballs per millorar el sistema.

Finalment s' inclouen els annexos i la bibliografia.

1.5 METODOLOGIA

Aquest projecte s'ha executat seguint el pla de treball que es va proposar al principi del semestre.

Un cop definit el pla de treball, es va programar una reunió amb l'equip de SmartCitizen i després del primer contacte, es va analitzar les necessitats i els objectius del projecte i la disponibilitat del material de desenvolupament, ja que al inici del semestre, el kit encara estava en procés de producció i per tant l'entrega del material sofriria d'un retard significatiu afectant al pla del treball, al procés de desenvolupament del firmware i sobretot, en el procés d'aprenentatge.

En aquest període d'espera, es va realitzar una investigació del mètode de programació per realitzar el projecte. Aquest període d'estudi i aprenentatge es va dirigir en l'anàlisi de l'actual firmware desenvolupat per Miguel Colom, i l'estudi de la documentació existent del mòdul RTX4100, on l'objectiu era conèixer la API de les colaps, on es va veure que era necessari aprendre a programar amb protothreads.

D'altre banda, es necessari aprendre la estructura de programació d 'Arduino, que inclou fer un estudi de la comunicació SPI i les llibreries necessàries per implantar-ho.

Al no tenir el material, es van realitzar codis de proves de la pagina web per el futur servidor. Això inclou a l'aprenentatge del llenguatge de programació Html5, Css i Javascript.

Un cop amb el material a les mans, el procés de desenvolupament va començar immediatament ,ja que únicament es disposava d'un mes per dur a terme tot el projecte.

Finalment, es procedeix a la redacció de la memòria i la presentació.

2 ESTAT DE L'ART

En aquest capítol s'analitza el hardware del SmartCitizen kit 1.1 (SCK 1.1) i el kit que s'utilitzarà en el desenvolupament del projecte, SmartCitizen Development Kit (SCDK), per tal de realitzar una comparativa. D'altra banda, es realitzarà un estudi de les diferents tecnologies utilitzades per poder realitzar el desenvolupament del sistema. Finalment s'analitzarà el software del RTX4100, necessari per entendre el funcionament i el mètode de programació de les aplicacions pròpies d'aquest mòdul, anomenades colapps.

2.1 SCK 1.1 – RN131

La versió actual de el SCK consisteix en dues PCB muntades una sobre l'altre. La primera està dedicada al processament, transmissió y emmagatzematge de dades (DataBoard). També conté el mòdul WIFI i la bateria. La segona conté els sensors i els seus complementos. Tot dues plaques podem treballar de forma independent i per tant es possible intercanvia-les o instal·lar nous components. Per un altre banda, l'alimentació per bateria no és estrictament necessària, ja que es pot alimentar a través del connector micro USB.

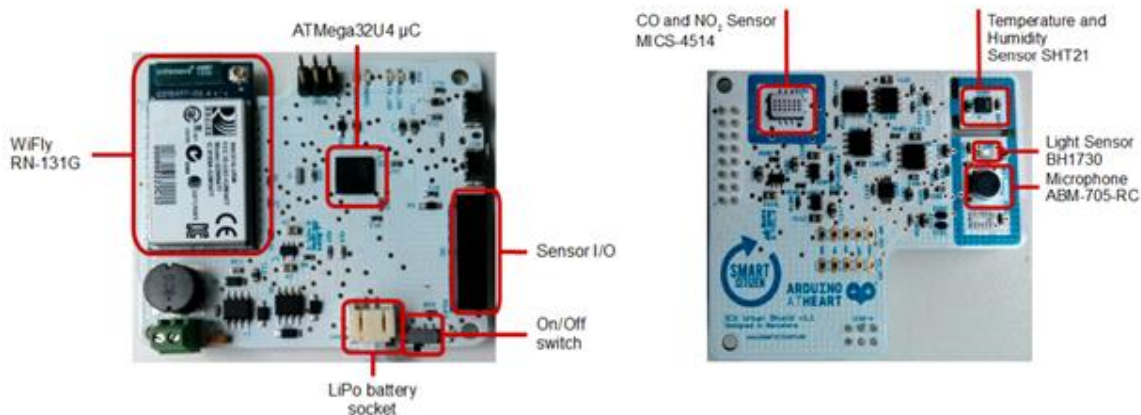


Figura 2. Vista Frontal i posterior del SKC 1.1

El MCU (Microcontrolador) de la databoard de la versió actual (SCK 1.1), és un Atmel ATMEGA32U4 (AVR 8-bits) amb arquitectura RISC (Reduced Instruction Set

Computing). Aquesta MCU es utilitzada en Arduino Leonard, Arduino Pro Micro, LilyPad USB o Arduino YUN. Això permet que el codi existent d'Arduino pugui ser adaptat o reutilitzat en el firmware del SCK. Aquest tipus de CPU té l'avantatge de que disposa de suport USB natiu i per tant no es necessari un convertidor usb-serie. Per un altre banda, té un baix consum elèctric (200mA) i disposa d'un mode "sleep" per estalviar energia. A continuació s'especifica les característiques principals d'aquest microcontrolador:

- Baix consum 200 mA
- 16/32Kbytes de EEPROM disponible pel firmware
- 1.25/2.5kbytes Internal SRAM
- 512Bytes/1K bytes Internal EEPROM
- Interfície JTAG
- Suport USB 2.0 natiu.
- Sis modes de Sleep: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- Voltatges 2.7 a 5-5V
- Temperatures de entre -40° C a 85°C , suficients per treballar en exteriors
- Ràpid. Freqüències entre 8Mhz a 2.7V i 16 Mhz a 4.5V

Una de les limitacions d'aquest MCU és la poca memòria EEPROM que disposa pel firmware (fins 32 kbytes). Actualment els llibres d'Arduino són de 28kb i la versió actual del SCK es de 27kB. Això pot ser un problema en un futur per la implantació de nous algorismes necessaris per la instal·lació de nous sensors que ho necessitin.

El mòdul Wifi, RN131 "Wifly", que integra les funcionalitats bàsiques de radio 802.11b/g i el stack TCP/IP. Segons el fabricant, aquesta és una solució "*Ultra-low power embedded TCP/IP solution*". La combinació del poc consum i l'habilitat de connectar-se a xarxes sense fils, enviar dades i tornar al mode standby en menys de 100 ms, permet que aquest mòdul funcioni durant anys amb dues piles estàndard AAA. El



Figura 3. Modul RN131 "Wifly"

consum d'aquest dispositiu es de 210mA en actiu, i de 4µA en mode Standby. Aquest consum "Ultra-low" és relatiu, ja que es poden trobar solucions millors. D'altre banda, aquest mòdul té un rang de temperatures de 0 a 70 Celsius. Aquesta limitació pot ser un problema, ja que normalment el SCK s'utilitza a l'exterior i alguns països la temperatura pot baixar dels 0 graus.

Un altra limitació es la interacció entre el MCU i el RN131 on queda limitat el sistema de comandes d'aquest últim, ja que no existeix ninguna API per treballar directament amb el firmware d'aquest mòdul. Les comandes que executa RN131 son:

- Command Syntax
- Command Organization
- Set Commands
- Get Commands
- Status Commands
- Action Commands
- File I/O Commands

2.2 SMARTCITIZEN KIT DVK

Aquest kit de desenvolupament està format per dues plaques: la primera una placa Arduino DUE, on en un futur serà substituïda per Arduino Zero, i la segona el SmartCitizen DVK shield. Aquesta segona placa conté el mòdul wifi RTX4100, una ranura per a targetes microSD i un sensor digital de temperatures i humitat. Aquest conjunt està destinat per ajudar als desenvolupadors a realitzar els prototips d'aplicacions per la nova "SmartCitizen Databoard 2.0".

2.2.1 Arduino DUE

El MCU d'Arduino Zero , ATMEL ARM Cortex M0+ (SAMD21) , encara està en desenvolupament, per tant, per aquest kit s'utilitzarà el MCU d'Arduino DUE, Atmel SAM3X8E 10 ARM CortexM3, per les seves característiques semblants amb el futur microcontrolador amb una arquitectura de 32-bit ARM similar.

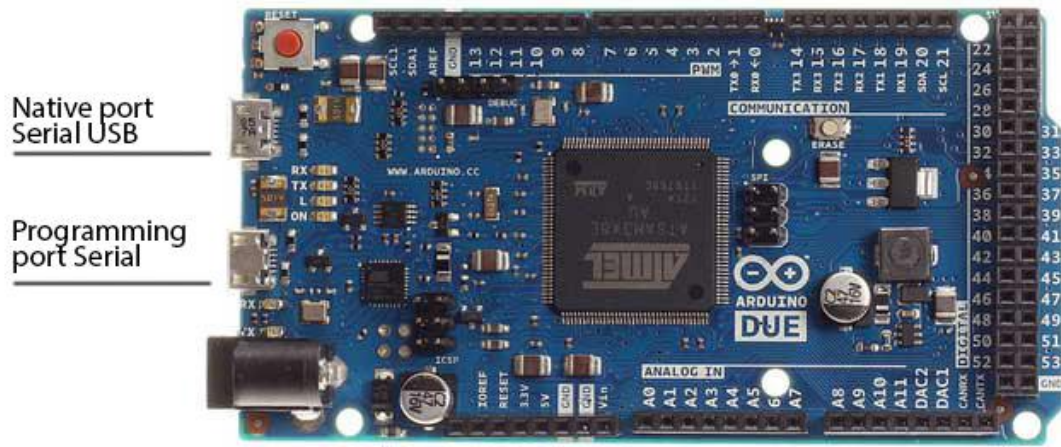


Figura 4. Vista Frontal Arduino DUE

Les característiques principals de la placa son:

- 96 Kbytes de SRAM
- 512 Kbytes de memòria flash per a Codi i aplicacions d'usuari
- Un controlador DMA que pot alleujar la CPU de realitzar tasques intenses de memòria.
- 54 pins digitals de entrada/sortida (de els quals 12 es poden utilitzar per a sortides PWN)
- 12 entrades analògiques,
- 4 UARTS (ports series),
- Un rellotge CPU de 84 Mhz,
- Dos connectors USB (Un natiu i un altre de programació)
- 2 DAC (pins de sortida Digital-Analogic),
- 2 connectors TWI,
- Un connector d'alimentació jack,
- Connectors SPI
- JTAG
- Un botó de reinici i d'esborrat de memòria flash

L'alimentació operativa d'aquesta placa, a diferència d'altres, es de 3.3V. El màxim voltatge que pot suportar els pins I/O es de 3.3V. En cas de alimentar-la amb voltatges més alts, com per exemple 5V, la placa pot danyar-se. Aquesta placa pot alimentar-se externament amb unconnector USB, alimentació per adaptador AC/DC (connector jack) o per una bateria. Els límits d'alimentació es de 6-16V i el recomanable es de 7-12V.

D'altre banda, la interfície SPI d'Arduino DUE funciona de forma totalment diferent que qualsevol altre placa Arduino. Les llibreries es poden utilitzar de la mateixa manera que altres plaques o utilitzant mètodes estesos. Els mètodes estesos exploten el processador SAM3X i permet algunes característiques interessants com:

- Selecció automàtica del mode esclau
- Selecció automàtica dels paràmetres de configuració de cada dispositiu (velocitat de rellotge, mode de enviament, etc) pel que cadascun pot tenir seleccionada la seva pròpia configuració automàticament.
- Arduino Due té tres pins SS (Slave Selected) per als dispositius Slave (pins 4, 10 i 52).

2.2.2 SmartCitizen DVK shield

El SmartCitizen DVK shield disposa d'un mòdul **Wifi RTX4100**. Aquest mòdul és una solució de baix consum gràcies a la combinació del microcontrolador Energy Micro Gecko EFM32G230F128 i el xip de baix consum Atheros AR4100 Wi-Fi (b/g/n) amb

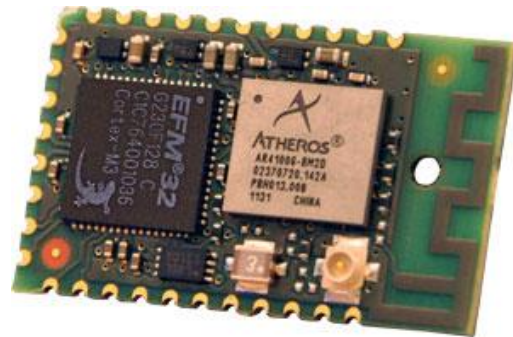


Figura 5. Vista frontal Modul RTX4100

unes dimensions de 18x30mm. Aquesta combinació fa que aquest mòdul sigui una solució ideal per aplicacions de automatització i per a sensors, gràcies a la seva eficiència i el seu consum reduït. El mòdul inclou una antena integrada però es pot instal·lar una externa mitjançant el connector respectiu.

El microcontrolador EFM32G230F128 disposa d'un processador de 32 Mhz ARM Cortex-M3, memòria flash interna de 128 kb per emmagatzemar el firmware WIFI i altres aplicacions, i 16 kb memòria RAM útils per executar aquestes aplicacions i la interacció

amb altres mòduls. L'alimentació del processador es realitza a través de LDO i controla l'alimentació a través de dos LDOS addicionals que alimenten al mòdul WIFI i la memòria Flash. Això permet que el xip WIFI pugui ser desconnectat quan no sigui necessari i reduir el consum energètic. Per un altre banda, el microcontrolador permet 5 diferents modes energètics que es molt important per el desenvolupament d'aplicacions de baix consum.

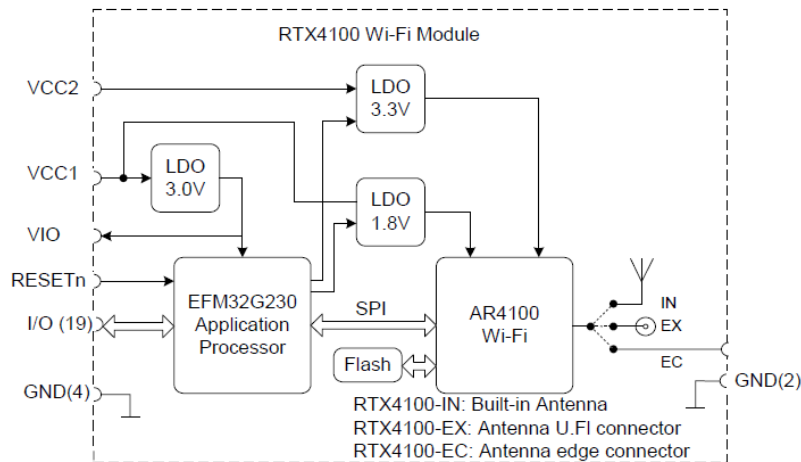


Figura 6. Circuit intern mòdul Wi-fi RTX4100

Les característiques principals d'aquest mòdul es poden resumir:

- Chip Wifi de Baix Consum (85mA)
- Permet posar el mòdul WIFI en mode stand-by per reduir el consum energètic quan no esta actiu
- Suport als stacks IPv4 i Ipv6
- Suport als estàndards de encriptació WEP,WPA i WPA2
- 32 Mhz ARM Cortex-M3 CPU
- 128 kb memòria Flash I 16 kb memòria RAM
- El MCU disposa de 5 diferents modes de consum energètic
- Es pot comunicar amb altres mòduls gràcies a les interfícies UART,SPI I I2C
- Ports GPIO
- ADC I DAC ports
- Timers
- Dimensions reduïdes 18x30mm
- Suporta temperatures de entre -40^o + 85^o. Útil per utilitzar en exteriors

D'altre banda, aquest mòdul permet el desenvolupament d'aplicacions a partir de la seva plataforma (AmelieSDK), on el propi proveïdor posa a disposició documentació de referència sobre aquesta API i varis exemples útils per conèixer el funcionament d'aquest. Aquestes aplicacions son anomenades colapps. Per a més informació sobre el funcionament software consulteu el punt específic de RTX4100 on s'analitzarà en profunditat les característiques del software d'aquest mòdul, ja que es fonamental per poder realitzar el projecte

Els **avantatges** principals d'utilitzar aquest mòdul comparant-lo amb el RN131 són:

- Consum reduït: 85 mA RTX4100 i 210 mA RN131
- Opera en un rang mes gran de temperatures: -40° + 85° RTX4100 i 0-70° RN131
- Permet desenvolupament d'aplicacions

Els altres components de la placa SmartCitizen DVK Shield són:

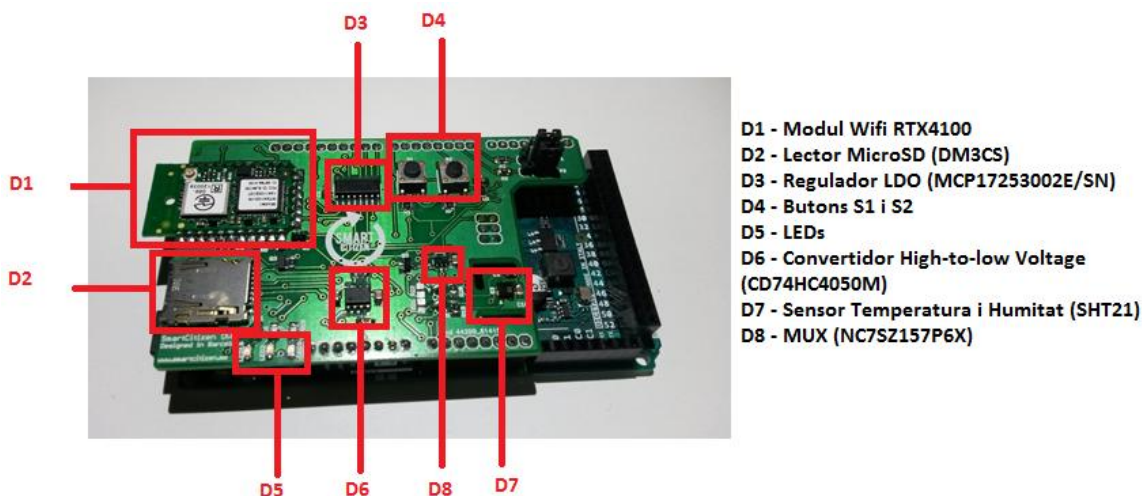


Figura 7. Components SmartCitizen DVK Shield

- **NC7SZ157P6X – MUX:** El MUX (Multiplexador) es utilitzat per seleccionar des d'Arduino quin serial port del RTX4100 es vol utilitzar. Es pot seleccionar dos port series:
 - LEU1 (LEUART) és utilitzat per l' aplicació terminal (com es Putty)
 - US1: (USART) és utilitzat per la carga de colapps, SW debugging i la plataforma de updates (utilitzat pel ColaController)

- **MCP1725-3002E/SN LDO REGULATOR:** És un regulador de voltatge per disposar de 3V fixos
- **CD74HC4050M High-to-Low Voltage Level Converter** IC utilitzat com a un convertidor a nivell lògic entre la Placa Arduino i el mòdul WIFI RTX4100
- **DM3CS SDCARD:** S'utilitza la interfície DM3CS com a dispositiu d'entrada per a targetes MicroSD. Aquesta interfície es alimentada amb 3V i es comunica amb el RTX4100 a través del protocol SPI
- **SHT21 Sensor de Temperatura i Humitat SCK** utilitza el sensor SHT21 per obtenir informació sobre la temperatura ambiental i la humitat relativa.
- **LEDs:**
 - LED1 Groc: Connectat al RTX4100 amb el pin 27, per testejar el Colapp Blinky
 - LED2 Blau : Connectat a Arduino amb el pin Analog0
 - LED3 Verd: Connectat a Arduino amb el pin Analog1
- **Botons S1 i S2:** Per activar els modes de la placa SmartCitizen SDK, on la comunicació es realitza a través del protocol SPI

2.2.3 Modes de funcionament

La placa SmartCitizen DVK disposa de dos botons (S1 i S2) que permeten l'activament de quatre modes:

- **MODE 1 :** És el mode default de la placa on el LED Verd és actiu. Aquest mode envia a través del connector USB els valors capturats pel sensor de temperatura i humitat, i a més, informa si es detecta si hi ha alguna targeta MicroSD connectada.
- **MODE 2:** Al prémer el botó **S1**, el LED verd començarà a parpellejar. Això significa que s'ha activat el mode terminal. En aquest mode es pot realitzar una comunicació amb el mòdul RTX4100 a través del USB. Al pujar una colapp, el LED groc s'activarà. Si es torna a prémer el boto S1, tornarà al MODE 1
- **MODE3:** Al prémer el boto S2, el LED blue s'activarà i es podran pujar aplicacions colapps amb l'aplicació ColaController, l'aplicació de PC utilitzada per descarregar COLAPS al mòdul via UART.

- **MODE4:** Al prémer de nou el botó S2, els LEDS verd i blau s'activaren i es podrà accedir al firmware de RTX, permeten l'actualització d'aquest. El LED blau parpellejarà si detecta que el firmware es modificat/actualitzat.

D'altre banda, els modes de funcionament poden ser modificats per implementar l'enviament de comandes SPI per l'activament dels modes Servidor WEB/Soft AP, on un dels objectius és modificar l'actual firmware de l'Arduino.

Els modes 1, 3 y 4 es recomana no modificar, ja que implementen les funcionalitats bàsiques per poder pujar colapps o modificar el firmware. Per tant, el mode 2 es el candidat per ser modificat i adaptar les funcionalitats necessàries pel projecte.

2.3 COMUNICACIÓ SPI

Abans d'analitzar el funcionament de les colapps és necessari entendre la comunicació entre els diferents elements de la placa. El sistema de comunicació entre els elements es realitza a través de protocols como és el I2C o el SPI. En aquest punt, analitzarem únicament la comunicació SPI, ja que el I2C presenta alguns desavantatges per el desenvolupament de l'aplicació que es vol implementar (comunicació no full dúplex, consum elèctric). Per un altre banda, les comandes actuals implementades en la colapp realitzada per M.Colom estan desenvolupades per ser utilitzades com comandes SPI.

El protocol SPI és un protocol síncron que treballa en full dúplex per rebre i transmetre informació, permetent que dos dispositius puguin comunicar-se entre si al mateix temps utilitzant canals diferents o línies diferents en el mateix cable. En ser un protocol síncron el sistema compta amb una línia addicional per a dades, que s' encarrega de portar el procés de sincronisme.

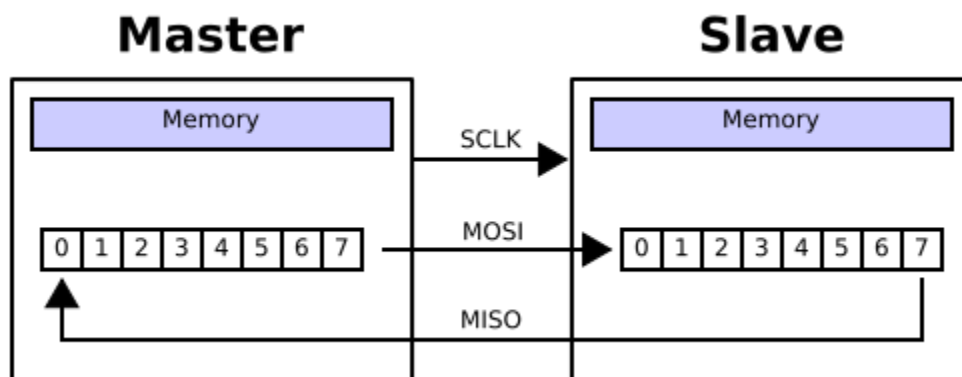


Figura 8. Comunicació SPI Master-Slave

Dins d'aquest protocol es defineix un mestre que serà aquell dispositiu encarregat de transmetre informació als esclaus. Els esclaus seran aquells dispositius que s'encarreguin de rebre i enviar informació al mestre.. Perquè aquest procés es faci realitat és necessari l'existència de dos registres de desplaçament, un per al mestre i un per a l'esclau respectivament. Els registres de desplaçament s'encarreguen d'emmagatzemar els bits, per realitzar una conversió paral·lela a serial per a la transmissió d'informació. Existeixen quatre línies lògiques encarregades de realitzar tot el procés:

- **MOSI (Master Out Slave In):**. Línia utilitzada per portar els bits que provenen del mestre cap a l'esclau
- **MISO (Master In Slave Out):**. Línia utilitzada per portar els bits que provenen de l'esclau cap al mestre
- **CLK (Clock):**. Línia que prové del mestre que s'encarrega d'enviar el senyal de rellotge per sincronitzar els dispositius
- **SS (Slave Select):**. Línia encarregada de seleccionar i a la vegada, habilitar un esclau.

2.3.1 Polaritat del rellotge i fase

A més d'establir un sincronisme per la comunicació, el master també ha de configurar la polaritat del rellotge i fase respecte a l'enviament de dades. Els noms per aquestes opcions son CPOL (polarització) i CPHA (fase)

El següent diagrama de temps mostra els diferents tipus de configuració d'aquests paràmetres

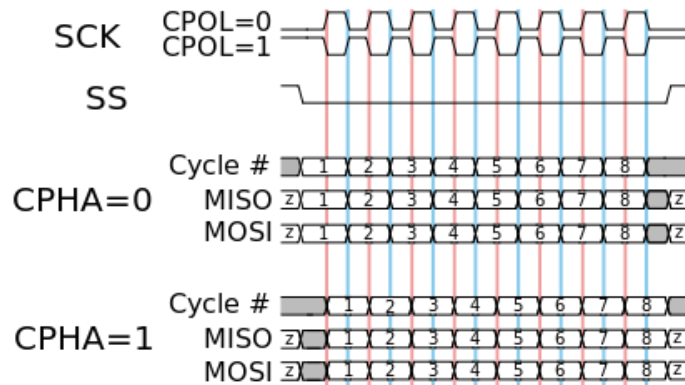


Figura 9. Diagrama de temps segons configuració CPOL i CPHA

En **CPOL = 0** el valor base del rellotge és zero.

- Per CPHA = 0, les dades són capturades en el flanc ascendent del rellotge i son enviades al flanc de baixada.
- Per CPHA = 1, les dades són capturades en flanc de baixada del rellotge i les enviades al flanc ascendent.

En **CPOL = 1** el valor base del rellotge és un (inversió de CPOL = 0)

- Per CPHA = 0, les dades són capturades en el flanc ascendent del rellotge i son enviades al flanc de baixada.
- Per CPHA = 1, les dades són capturades en flanc de baixada del rellotge i les enviades al flanc ascendent.

A continuació s'esmentarà els avantatges e inconvenients d'utilitzar aquest sistema de comunicació

Avantatges:

- Comunicació Ful Duplex
- Major velocitat de transmissió que amb I²C o SMBus
- Protocol flexible en què es pot tenir un control absolut sobre els bits transmesos
- No està limitat a la transferència de blocs de 8 bits
- Elecció de la grandària de la trama de bits,
- La seva implementació en dispositius és extremadament simple

- Consumeix menys energia que I²C o que SMBus degut a que posseeix menys circuits i aquests són més simples
- No és necessari arbitratge o mecanisme de resposta davant fallades
- Els dispositius clients utilitzen el rellotge que envia el servidor. No necessiten per tant el seu propi rellotge
- No és obligatori implementar un transceptor (emissor i receptor), el dispositiu connectat pot configurar-se perquè solament envii, només rebí o ambdues coses alhora
- Utilitza molts menys terminals en cada xip/connector que una interfície equivalent
- Com a molt, un únic senyal específic per a cada client (senyal SS), les altres senyals poden ser compartides

Desavantatges

- Utilitzen més pins que I²C, fins i tot en la variant de 3 fils
- L'adreçament es fa mitjançant línies específiques (senyalització fora de banda) a diferència del que ocorre en I²C que se selecciona cada xip mitjançant una adreça de 7 bits que s'envia per les mateixes línies del bus
- No hi ha control de flux
- No hi ha senyal d'assentiment. El servidor podria estar enviant informació sense que estigués connectat cap client i no s'adonaria de res
- No permet tenir diversos servidors connectats al bus
- Només funciona en les distàncies curtes a diferència de, per exemple, RS-232, RS-485, o Bus CAN

2.4 PROGRAMACIO C++ -PROTOTHREADS-

Els protothreads són un conjunt de “*macros*”, que permeten realitzar un codi més llegible i més econòmic a mantenir, permeten el disseny de firmwares on les funcions del codi s'executen de forma similar als threads. A diferència dels threads normals, no necessita reservar una quantitat de memòria per a cadascun, sino que únicament utilitza uns 2 bytes de RAM per a cada protothreads, el qual el fa ideal per a MCUs de 8 o 16 bits, como ara és el cas del RTX4100, i per tant, al no utilitzar un sistema amb gaire recursos , permet una reducció de consum d'elèctric ideal per la MCU.

Un protothread pot romandre actiu fins que s'activa altres threads. Això ocorre quan una tasca ha finalitzat o quan arriba a un punt on ha d'esperar a que es produeixi un esdeveniment, per tant, varis threads poden estar en espera fins que arriba aquest esdeveniment. Aquest model de execució s'anomena "execució dirigida per esdeveniments". El programador ha de tenir en compte que les variables locals no es conservaran quan siguin dirigits al threads, o sigui emmagatzemades, encara que s'hagi trobat una situació de bloqueig i que s'espera una represa. Les variables que es volen preservar hauran de ser declarades com globals.

A continuació s'analitzen els diferents tipus de macros utilitzades pel desenvolupament de les colapps :

Inicialització

- PT_INIT(pt): Inicia un protothread. Abans de ser cridats, els protothreads necessiten ser inicialitzats.

Declaració y definició

- PT_THREAD(name_args) : Declara un protothread. Tots els protothreads necessiten aquesta macro per ser definits.
- PT_BEGIN (pt): Declara l'inici d'un protothread dins de la funció C implementada del protothread.
- PT_END(pt): Declara el final d'un protothread. Ha de ser utilitzada sempre junt a la macro PT_BEGIN

Bloqueig

- PT_WAIT_UNTIL(pt, condition): Bloqueja i espera fins que la condició sigui certa
- PT_WAIT_WHILE(pt, condition): Bloqueja i espera mentre la condició sigui certa

Protothreads jeràrquics

- PT_WAIT_THREAD(pt, thread): Bloqueja i espera fins que un protothread fill finalitzi
- PT_SPAWN(pt, child, thread): Crida a un protothread fill y espera fins que finalitzi.

Sortir y reiniciar

- PT_RESTART(pt): Reinicia el protothread. Bloqueja i reinicia l'execució.
- PT_EXIT(pt): Surt d'un protothread. Si el protothread es cridat per altre protothread, el protothread pare es desbloquejarà i podrà continuar funcionant.

Cedir protothreads

- PT_YIELD(pt): Bloqueja l'execució d'un protothread y cedeix l'execució a la resta. El protothread continuarà la seva execució al punt on es va quedar bloquejat
- PT_YIELD_UNTIL(pt, condition): Bloqueja l'execució d'un protothread i cedeix l'execució a la resta fins que la condició sigui certa El protothread continuarà la seva execució al punt on es va quedar bloquejat

2.5 PROTOCOL HTTP

El Hypertext Transfer Protocol o HTTP és un protocol de tipus client- servidor per la realització de transaccions que permeten l'intercanvi d'informació entre un client (navegador web) i un servidor mitjançant l'enviament d'una cadena de caràcters anomenada direcció URL

HTTP es basa en senzilles operacions de sol·licitud/resposta. A continuació s'analitza el procés d'intercanvi d'informació:

- L'usuari introdueix una URL al navegador web (client) realitzant una petició al servidor . El client web descodifica la URL, separant les diferents parts, on s'identifica el protocol d'accés, la direcció DNS o IP del servidor, el port (per defecte el 80) i l'objecte o recurs sol·licitat.
- S'obre una connexió TCP/IP amb el servidor per el port corresponent i es realitza la petició HTTP. Per realitzar aquestapetició s'envia:
 - La comanda necessària o mètode (GET,POST, HEAD)
 - La direcció del recurs (URL),
 - La versió del protocol HTTP utilitzada (normalment HTTP 1.0)
 - Els camps de capçalera de la sol·licitud , el qual és un conjunt de dades opcionals que varien segons la petició i que aporten informació addicional sobre la sol·licitud i/o el client (Navegador, sistema operatiu, etc)
 - Cos de la sol·licitud: Conjunt de línies addicionals separades de la resta de petició per una línia en blanc que permet l'enviament de dades addicionals
- El servidor retorna la resposta al client que consisteix :
 - Línia d'estat : Especifica la versió del protocol utilitzada i l'estat de la sol·licitud mitjançant un text explicatiu i el codicorresponent.

- Camps de capçalera de resposta: Conjunt de dades addicionals que aporten informació sobre la resposta i/o servidor
- Cos de la resposta amb la informació sol·licitada
- Finalment es tanca la connexió TCP

A continuació es descriuen els mètodes per realitzar peticions, on s'indica l'acció que s'ha d'executar sobre el recurs sol·licitat

Metode	Descripció
GET	Sol·licita el recurs situat en la URL especificada
HEAD	Sol·licita la capçalera del recurs ubicat en la URL especificada
POST	Envia dades per a que siguin processades pel programa ubicat en la URL especificada
PUT	Envia dades a la URL especificada
DELETE	Esborra el recurs ubicat en la URL especificada

Tabla 1. Mètodes petició HTTP

Els codis de resposta de part del servidor més habituals:

Codi	Descripció
10x	Missatge de informació, aquest codi no s'utilitza en HTTP1.0
20x	Indica la correcta execució de la transacció
30x	Indica el recurs ja no es troba en la ubicació especificada
40x	Error per part del client. La sol·licitud és incorrecta
50x	Error per part del servidor. Possiblement hagi un error intern

Tabla 2. Codis Resposta HTTP

2.6 LLENGUATGES DE PROGRAMACIO WEB

Els llenguatges que s'utilitzaran són interpretats per part del client. El nostre navegador és una aplicació que pot interpretar les ordres rebudes, en forma de codi HTML principalment, i convertir-les en pàgines. Quan es carrega una pàgina s'estableix una petició d'un arxiu al servidor, el qual s'envia al nostre navegador per que l'interpreti. Els llenguatges utilitzats per implementar l'interfecte web son:

- HTML5: per la estructura de la pàgina
- CSS3: Per la edició d'estils

Segons les limitacions del mòdul RTX4100, s'intentarà implementar javascript per donar dinamisme a la pàgina i estudiar si és viable la implementació d'AJAX per l'intercanvi dinàmic d'informació entre la web allotjada al servidor Web i el navegador, per evitar ser actualitzada constantment i únicaments'actualitzi quan arribi un nou valor (como els valors de les temperatures).

2.7 ANALISIS SOFTWARE RTX4100

El Firmware del RTX4100 està dividit en dues parts:

- L'aplicació d'usuari, o també coneguda com Co-Located Application (Colapp)
- Mòdul firmware de RTX que dona suport a les colapp.

Els desenvolupadors d'aplicacions poden programar colapps utilitzant les API's definides per la plataforma AmelieSDK i el framework de lescolapp. L'aplicació pot ser descarregada al mòdul sense modificar la resta del firmware, mitjançant la comunicació USB (Connector USB Programming) i utilitzant l'aplicació ColaController, prèviament activant el mode 3, corresponent a la pujada de colapps.

A continuació es pot veure l'arquitectura Software del RTX4100 i una breu definició de cada part:

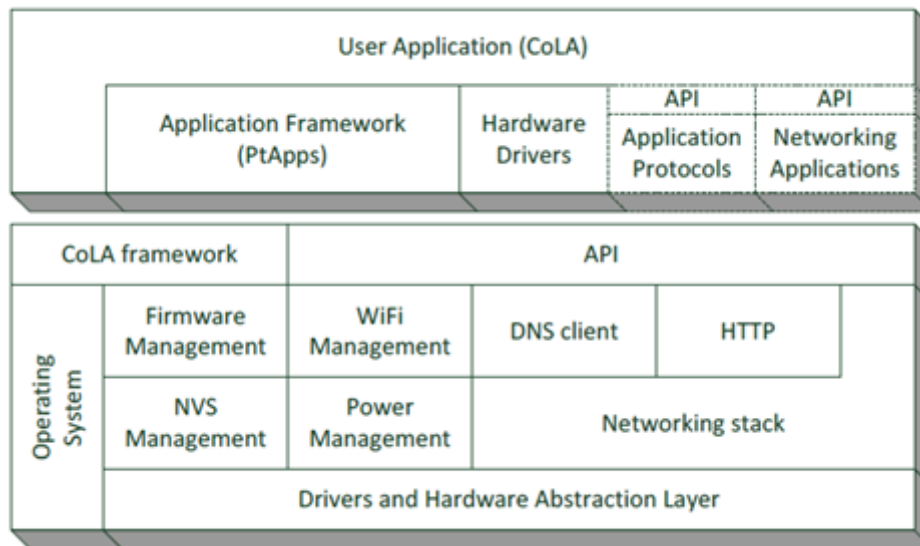


Figura 10. Arquitectura Software RTX4100

2.7.1 Bloc Co-Located Application (Colapps)

- **User Application:** component que implementa la funcionalitat de l'aplicació del dispositiu WIFI.
- **Application Framework (PtApps):** Es un conjunt d'aplicacions que funcionen a partir de protothreads i implementa les funcionalitats principals. Com per exemple: control dels leds, consum wifi, etc
- **Drivers Hardware:** Components software per accedir als drivers hardware
- **API's per Protocols d'Aplicacions :** Components opcionals per la implementació d'aplicacions especificques.
- **API's per Aplicacions de Xarxa:** Component opcional per implementar aplicacions de xarxa que com FTP, TFTP,HTTP, Web server.

2.7.2 Bloc Mòdul Firmware

- **Co-Located Application (Colapp) Framework:** Aquest component implementa un model de programació on l'aplicació s'enllaça dinàmicament amb els serveis prestats per les capes inferiors . L'aplicació es compila i s'enllaça com un programa separat que s'executa com una tasca al Sistema Operatiu (ROS)

- **API:** Aquesta és la interfície exposada per la Plataforma de firmware. S'exposa tota les funcionalitats que necessita l'aplicació per implementar un dispositiu Wi-Fi
- **OS:** sistema operatiu de RTX de baix consum amb les funcionalitats necessàries per organitzar tasques internes, així com les colapps
- **Networking Stack:** Aquesta és una capa funcional que implementa els protocols de xarxa UDP, TCP/IP i xarxes IPv4 i IPv6
- **Client DNS:** El client DNS s'utilitza per traduir noms de domini a adreces IP mitjançant la consulta d'un servidor DNS.
- **HTTP:** servidor HTTP comú i l'aplicació client. El servidor HTTP implementada inclou el maneig de connexions TCP, l'anàlisi dels missatges de petició HTTP, generació/ enviament de missatges de resposta HTTP, l'emmagatzematge de recursos HTTP (pàgines web), implementació del client que inclou el maneig de connexions TCP, generació / enviament de missatges de sol·licituds HTTP , i l'anàlisi dels missatges de resposta HTTP.
- **Gestió de WiFi:** Aquest component s'encarrega de tots els aspectes de la connexió Wi-Fi amb un AP, incloent la seguretat i el maneig de claus per assegurar la connexió sense fils, gestió de l'energia Wi-Fi, etc.
- **Administració d'energia:** Aquest component gestiona la freqüència del MCU, així com l'energia del mòdul. S'assegura que qualsevol part del MCU només s'està executant la quantitat de temps necessari per estalviar consum elèctric.
- **Gestió de Firmware:** Implementa la funcionalitat per realitzar actualitzacions del firmware de colapps.
- **Gestió NVS:** Implementa un sistema d'emmagatzematge no volàtil (NVS) en una part de la FLASH interna a la MCU.
- **Drivers:** Aquesta és una capa funcional que implementa una sèrie de Drivers per a la MCU, així com la interfície física per al subcomponent Wi-Fi

Un cop definida l'arquitectura software del RTX4100 es pot definir el framework de les aplicacions.

2.7.3 Framework de les Aplicacions

Abans de fer un anàlisi dels codis de les colapps és interessant saber quin tipus de framework estan basades aquestes aplicacions. Hem de tenir en compte que el codi

d'aquest es realitza en C++ i que el sistema te uns recursos molt reduïts, i per tant és interessant trobar un sistema de programació que tingui un consum de recursos el més reduït possible. El framework de les aplicacions està basat en protothreads, facilitant la implementació de qualsevol aplicació.

2.7.4 El sistema operatiu de RTX (ROS)

El sistema operatiu de RTX (ROS), és un sistema operatiu cooperatiu basat en un sistema de missatges, permeten que tasques individuals es comuniquin entre si a través del sistema. El framework de les aplicacions COLA (Cola Task) funciona como una única tasca i s'executarà en el seu propi thread(PtMain) on rebrà tots els "mails" necessaris per a que els processi. Aquest protothread principal (PTmain) crearà tots els protothreads necessaris en el firmware de l'aplicació. Cada mail enviat està etiquetat per el seu identificador descriptiu i un conjunt de paràmetres i aquests són enviats al "gestionador de missatges" (Mail Scheduler) on posarà a la cua de e-mails indicant a quin thread s'ha de dirigir.

Totes les aplicacions de tipus colapp necessitaran la funció Colatask el qual enviarà els missatges del sistema. La majoria d'aplicacions estan basades en protothreads i són enviades pel Colatask on preseleccionarà els mails per organitzar la inicialització i finalització de la tasca.

El sistema operatiu de RTX disposa també d'un simple sistema temporitzador (Timer mails) que permet que la aplicació pugui subscriure's a unmail des de el sistema operatiu després de transcorre un temps determinat. Aquest sistema de timer es pot donar de baixa abans de expirar-se el temps.

Finalment, hi han aplicacions que necessiten que les dades generades o la seva pròpia configuració sigui emmagatzemada de forma persistent per recuperar-les en cas d'una aturada del sistema. Com per exemple, el firmware del dispositiu, claus wifi, SSIDs, etc. ROS disposa d'un sistema de Emmagatzematge no volàtil (NVS). RTX4100 disposa d'una memòria flash de 128 kbytes, on 512 bytes estan reservats pel firmware i la resta per la utilització de Colapps. Segons. D'aquests 512 bytes del firmware, es pot assumir que 256 bytes estan disponibles per a NVS. El offset i la mida està disponible en les

variables ColaNvsOffset i ColaNvsSize. Es pot accedir a la NVS a través de les funcions NvsRead() i NvsWrite() .

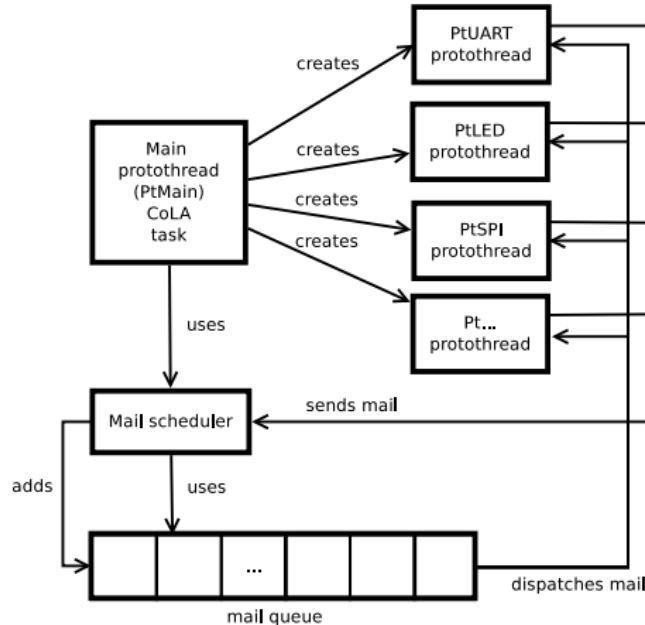


Figura 11. Procés d'enviament de missatges i execució de protothreads

2.7.5 API AmelieSDK

La API utilitzada per les Co-Located Applications (Colapps) es la AmelieSDK. Aquesta està formada per les funcions de la Cola Interface i el conjunt de mails utilitzats per la pròpia Api per accedir a les funcions implementades per la pròpia plataforma. Els mails en que es basa la API son:

- Amelie Management API
- Atheros Wi-Fi API
- Socket API
- IpConfig API
- DNS client API

La descripció de cada sistema de mail de cada API es pot veure en la documentació de la pròpia API AmelieSDK. Cada secció te un amplia descripció de les funcions utilitzades.

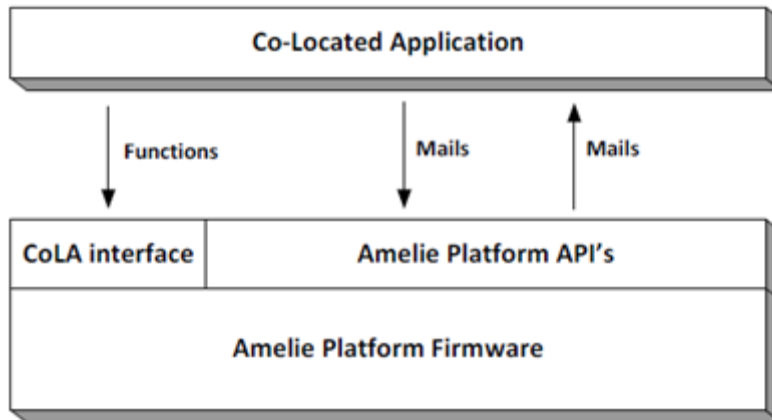


Figura 12. Comunicació Colapps amb l'API AmelieSDK

D'altre banda, la Cola Interface exporta totes les funcions del sistema operatiu de RTX (ROS) mitjançant la utilització d'un bloc de dades especial emmagatzemades en la memòria flash. Aquesta Interface ofereix funcions d'accés mitjançant la utilització de la llibreria RtxCore.h. Les funcions són les següents:

- OS: enviament/recepció de mails i Timers
- HEAP : assignació dinàmica de memòria
- NVS: Emmagatzematge de dades en la NVS
- System- controlador de rendiment i gestió d'energia
- RtxEai- Registre de comentaris en els logs de mails i screen output

2.8 ANALISIS COLAPP SMARTCITIZEN-RTX4100

Actualment hi ha una Colapp que va ser desenvolupada en el treball de final de grau de M.Colom "Analysis, Improvement, and Development of New Firmware for the Smart Citizen Kit Ambient Board" i que implementa les funcionalitats principals necessàries per el funcionament del SCK. La arquitectura i la API SPI desenvolupades en aquest codi es faran servir com a referència per totes les funcionalitats necessàries per realitzar el projecte.

Les funcionalitats principals d'aquesta colapp són la de associar-se a una xarxa WIFI i la connexió remota mitjançant un socket TCP. Aquestes funcionalitats es realitzen mitjançant l'enviament de comandes SPI del mòdul WIFI al MCU del SCK. M.Colom al tenir

únicament disponible el mòdul WIFI i no la nova placa SCK, va realitzar la comunicació enviant comandes de forma manual a traves d'una Raspberry per els ports GPIO.

Actualment hi ha 15 comandes SPI funcionals:

Command #1 (get status)
Command #2 (DNS33 resolve)
Command #3 (IP config)
Command #4 (TCP start)
Command #5 (associate and connect to the AP)
Command #6 (disassociate and disconnect from the AP)
Command #7 (setup AP)
Command #8 (close TCP connection)
Command #9 (TCP receive)
Command #10 (TCP send)
Command #11 (Wifi chip power on/off)
Command #12 (Wifi set powersave profile)
Command #13 (Wifi set transmit power)
Command #14 (Wifi chip suspend)
Command #15 (Wifi chip resume)

Tabla 3. Comandes SPI firmware SmartCitizenRTX4100

3 PREPARACIÓ DE L'ENTORN DE DESENVOLUPAMENT

En aquest capítol es descriu l'entorn de desenvolupament necessari per poder implementar el sistema. Per poder desenvolupar i carregar les colapps al mòdul és necessari la instal·lació del software del proveïdor del mòdul, AmelieSDK, juntament amb les eines de compilació. D'altra banda, és necessari la instal·lació de l'entorn de programació Arduino IDE per poder modificar el firmware actual de la placa Arduino.

3.1 SOFTWARE DE DESENVOLUPAMENT RTX:

Els desenvolupadors d'aplicacions hauran d'instal·lar el software de desenvolupament del RTX (SDK) i un editor de text per poder desenvolupar les Colapps. A continuació es detalla el procediment i el que es trobarà dins de el software instal·lat

El primer que s'ha de realitzar es instal·lar el "toolchain" necessari per compilar les aplicacions, que inclou el preprocessador, un compilador y un linker. Es pot descarregar l'arxiu des de <https://launchpad.net/gcc-arm-embedded/+milestone/4.7-2013-q3-update>

Un cop instal·lat el Toolchain, s'haurà d'instal·lar l'ultima versió del SDK des de la pàgina oficial de RTX . És necessari estar registrar en la pàgina web per realitzar la descàrrega. Un cop registrats, a part de poder descarregar el SDK, es podrà tenir accés a tota la documentació necessària per entendre el funcionament del mòdul.

Un cop instal·lat el SDK es generaran les següents carpetes (si no s'ha modificat la ruta estàndard d'instal·lació) en c:\AmelieSdk\vX.XX :

- **3Party** – Inclou el source files de tercers. El arxiu mes important que es pot trobar es el "efm32lib" de Energy Micro que es utilitzat per el accés dels perifèrics de la MCU.
- **ColaController** – L'aplicació utilitzada per carregar les colapps al mòdul via UART. Aquesta aplicació és essencial per instal·lar les aplicacions desenvolupades al mòdul RTX.
- **Components** – Componentes necessaris pel desenvolupament de les colapps
- **Documents** – Arxius de documentació

- **Include** – Arxius de capçaleres (headers) comuns
- **Projects** – Aquí es trobaran totes les colapps d'exemple generades en la instal·lació i on es guardaran de forma estàndard les colapps desenvolupades
- **Tools** – Conjunt d'eines utilitzades per el desenvolupament de colapps. Inclou Doxygen, i altres eines per documentació de codi i el sistema de RtxBuild utilitzada per el procés de compilació.

La part més important per entendre el funcionament de les Colapps es pot trobar dins dels projectes d'exemple. Dins de la carpeta Project (c:\AmelieSdk\vx.XXProjects\Amelie) es troben les carpetes Components y colapps.

Dins de la carpeta Components trobarem:

- **Api** – Capçaleres de fitxers de les API del sistema
- **ApiMps** – funcions d'ajuda per l'enviament de missatges al sistema.
- **Drivers** – Implementació dels drivers necessaris
- **PtApps** – Protothreads basats en el framework de 'aplicació que es poden utilitzar en les aplicacions per desenvolupar

Finalment en la carpeta de colapp trobarem:

- **Apps** – Conjunt d'aplicacions d'exemple
- **Config** – Arxius de configuració necessaris per desenvolupar les aplicacions d'exemple.
- **Rsx** – Eina per la API d'enviament de mails via UART
- **Scripts** – Scripts utilitzats en el constructor/linker

A continuació s'analitzaran de forma resumida les Colapps d'exemple que ens servirà de gran ajuda entendre i poder desenvolupar el projecte

- Blinky: Aplicaciósenzilla que utilitza el timer per encendre el LED de la placa
- Servidor/client TCP
- Servidor/client UDP
- Aplicació Terminal
- 3 diferents implantacions de sensors de temperatura on es demostra les capacitats de poc consum del mòdul.

- TempSensorSoftApCfg, Demostració de com es pot configurar el mòdul per a que iniciï en SOFTAP amb suport Telnet.
- SoftApTcpServer :Implantació d'un server TCP en mode SOFTAP
- WebServer:Exemple de implementació d'un simple servidor WEB

3.1.1 Creació y edició

Per crear o modificar les colapps es necessari un editor de text o un entorn de desenvolupament per programar en C. En aquest projecte s'utilitzarà el editor de text gratuït **Notepad++**.

D'altra banda, per crear un projecte és necessari crear els arxius principals per poder compilar-los amb el toolchain. Per poder fer-ho és necessari obrir la consola de comandes de Windows (cmd) i navegar fins la carpeta de projectes del AmelieSDK, on és troba l'arxiu **CreateApp.bat**, el qual s'executarà amb el nom del projecte.

```

Administrador: C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\Losco>cd C:\AmelieSdk\1.6.0.58\Projects\Amelie\COLApps\Apps

C:\AmelieSdk\1.6.0.58\Projects\Amelie\COLApps\Apps>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 2AE8-AE56

Directorio de C:\AmelieSdk\1.6.0.58\Projects\Amelie\COLApps\Apps

28/05/2015  14:18  <DIR>          .
28/05/2015  14:18  <DIR>          ..
17/05/2015  17:43  <DIR>          2lemetry
17/05/2015  17:43  <DIR>          Blinky
07/11/2013  12:45  <FILE>          3.167 CreateApp.bat
17/05/2015  17:43  <DIR>          Exosite
17/05/2015  17:43  <DIR>          Nabto
28/05/2015  14:18  <DIR>          proyecto
17/05/2015  17:43  <DIR>          Sensinode
21/05/2015  20:33  <DIR>          SmartCitizenRTX4100
17/05/2015  17:43  <DIR>          SoftApTcpServer
17/05/2015  17:43  <DIR>          Tcp2Uart
17/05/2015  17:43  <DIR>          TcpClient
17/05/2015  17:43  <DIR>          TcpServer
17/05/2015  17:43  <DIR>          TempSensor
17/05/2015  17:43  <DIR>          TempSensor2
17/05/2015  17:43  <DIR>          TempSensor3
17/05/2015  17:43  <DIR>          TempSensorSoftApCfg
17/05/2015  17:43  <DIR>          Terminal
17/05/2015  17:43  <DIR>          UdpClient
17/05/2015  17:43  <DIR>          UdpServer
20/05/2015  19:16  <DIR>          WebServer
                1 archivos          3.167 bytes
                21 dirs  21.907.451.904 bytes libres

C:\AmelieSdk\1.6.0.58\Projects\Amelie\COLApps\Apps>CreateApp NuevoProyecto
Project NuevoProyecto created successfully!

C:\AmelieSdk\1.6.0.58\Projects\Amelie\COLApps\Apps>

```

Figura 13. Creació del directori del projecte mitjançant la comanda CreateApp.bat

Dins de la carpeta creada, es troba els següents arxius i directoris:

- Build
 - RTX4100_WSAB: Inclou tots els arxius necessaris per compilar el projecte
 - Ba.bat
 - Bp.bat

- Makefile
- Node.cnf
- Nx4
- RTX4110_WSAB: mateix arxius que RTX4100_WSAB
- Main.c : Codi principal del projecte
- Node.ncf: Document on s'especifica els arxius que formen part del projecte.

3.1.2 Compilació

Per compilar i linkar les colapps és necessari fer-ho també des de la consola de comandes (cmd) i navegar fins a la carpeta Build\RTX4100_WSAB del projecte. A continuació s'executa la comanda batch ba.bat (BA:Build All). Una vegada el projecte es compilat correctament, el resultat serà com a continuació.

```

Administrador: C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\Losco>cd C:\AmelieSdk\1.6.0.58\Projects\Amelie\COLApps\Apps\Blinky\Build\RTX4100_WSAB
C:\AmelieSdk\1.6.0.58\Projects\Amelie\COLApps\Apps\Blinky\Build\RTX4100_WSAB>ba.bat
Cleaning up
Tool check: "C:\Program Files (x86)\GNU Tools ARM Embedded\4.7.2013q3\bin\arm-none-eabi-gcc.exe"
4.7.4
Building filelist.mak
Processing: Projects\Amelie\COLApps\Apps\Blinky\Build\RTX4100_WSAB
Finished: Projects\Amelie\COLApps\Apps\Blinky\Build\RTX4100_WSAB
Time: 0:1
Updating C:\AmelieSdk\1.6.0.58\Projects\Amelie\COLApps\Config\BuildInfo.inc

Main.c
BuildInfo.c
ImageHeader.c
PsiInfo.c

Linking Blinky.hex

   500 bytes code
    14 bytes data
    96 bytes const
    596 bytes flash

Creating intel-hex file
Creating binary file
Creating FW update file
Time: 00:02:246

C:\AmelieSdk\1.6.0.58\Projects\Amelie\COLApps\Apps\Blinky\Build\RTX4100_WSAB>

```

Figura 14. Compilació de la colapp

En cas de fallar, el compilador indicarà les línies on es troben els errors, com per exemple:

```

C:\AmelieSdk\1.6.0.58\Projects\Amelie\COLApps\Apps\Blinky\Build\RTX4100_WSAB>ba.bat
Cleaning up
Tool check: "C:\Program Files (x86)\GNU Tools ARM Embedded\4.7.2013q3\bin\arm-none-eabi-gcc.exe"
4.7.4
Building filelist.mak
Processing: Projects\Amelie\COLApps\Apps\Blinky\Build\RTX4100_WSAB
Finished: Projects\Amelie\COLApps\Apps\Blinky\Build\RTX4100_WSAB
Time: 0:1
Updating C:\AmelieSdk\1.6.0.58\Projects\Amelie\COLApps\Config\BuildInfo.inc

Main.c
BuildInfo.c
ImageHeader.c
PsiInfo.c
C:\AmelieSdk\1.6.0.58\Projects\Amelie\COLApps\Apps\Blinky\Main.c: In function 'ColaTask':
C:\AmelieSdk\1.6.0.58\Projects\Amelie\COLApps\Apps\Blinky\Main.c:67:21: error: 'APP_LED_TIME' undeclared (first use in this function)
C:\AmelieSdk\1.6.0.58\Projects\Amelie\COLApps\Apps\Blinky\Main.c:67:21: note: each undeclared identifier is reported only once for each function it appears in
gnunake: *** OBJ/obj.o: Error 1
gnunake: *** Waiting for unfinished jobs....
Time: 00:01:669

C:\AmelieSdk\1.6.0.58\Projects\Amelie\COLApps\Apps\Blinky\Build\RTX4100_WSAB>

```

Figura 15. Exemple d'error de compilació

On la línia 67:21 indica que s'ha trobar un error d'identificador no declarat. En cas de necessitar el log, es pot trobar dins de l'arxiu BuildLog.txt dins de la carpeta Build\RTX4100_WSAB del propi projecte

3.1.3 Carrega (Loading)

Un cop compilat el projecte, es generar un arxiu .fws. Aquest arxiu es la colaps que haurà de ser importada al mòdul RTX4100 amb l'aplicació ColaController.

Abans d'executar-la es necessari configurar el RTX EAI Port Server. Per a això es dóna clic dret a la icona que es troba dins la barra de tasques i es selecciona Setup al menú. També es pot executar directament des de el directori de l'aplicació.

A continuació s'obrirà l'aplicació de configuració del RTX EAI Port Server. El menú a configurar és el UART, on s'indicarà:

- a) El port COM assignat a la placa Arduino. En aquest cas el COM3
- b) La velocitat en bps (Speed). En aquest cas 115200 bps
- c) Activar "Use Windows Based Flow Control"
- d) Desactivar Packet Based i UART CTS del Output flow control

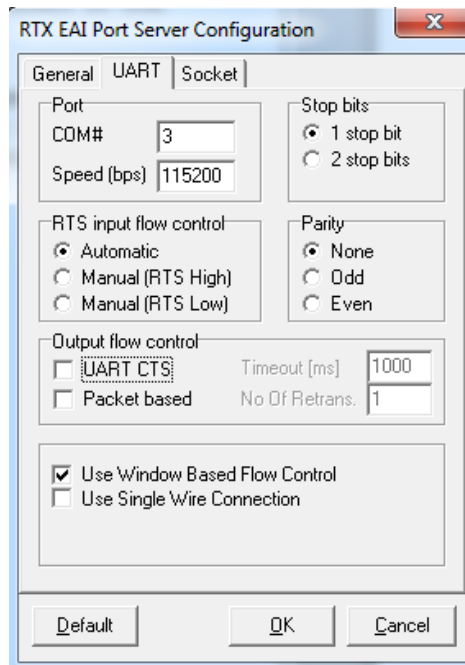


Figura 16. Configuració RTX EAI Port Server

Els passos per importar les colapps un cop configurar el RTX EAI Port Server són els següents:

- 1) Connectar el cable micro-USBal port USB-Programming
- 2) Activar el mode 2 de la placa SVDK prenen el botó 2 fins que s'il·luminin els leds blau i groc
- 3) Localitzar la carpeta AmelieSDK en el menú d'inici i obrir l'aplicació ColaController
- 4) Un cop oberta l'aplicació, donar al boto "Refresh". En cas d'error sortirà el següent missatge:

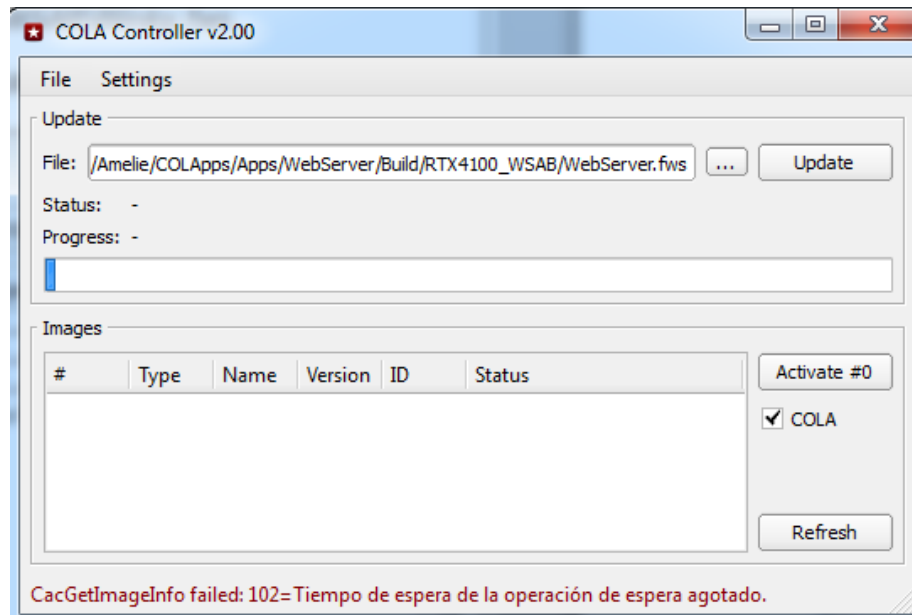


Figura 17. ColaController: Error detecció del mòdul RTX

Això potser provocat per què no estigui en Mode 2, o estigui obert la monitorització d'Arduino IDE. També pot ser que no se hagi iniciat correctament la placa. En tot cas és necessari que únicament estigui oberta l'aplicació ColaController i reiniciar el dispositiu. En la finestra "Status Windows" del RTX AI Server es pot veure si s'ha iniciat correctament el dispositiu..

- 5) Per importar colapps, s'ha de seleccionar el botó "..." i trobar l'arxiu .fws dins del directori \Build\RTX4100_WSAB del projecte.
- 6) Seleccionar Update per carregar l'arxiu al mòdul.
- 7) Finalment seleccionar Refresh per a que es vegi la correcta carrega de la colapp

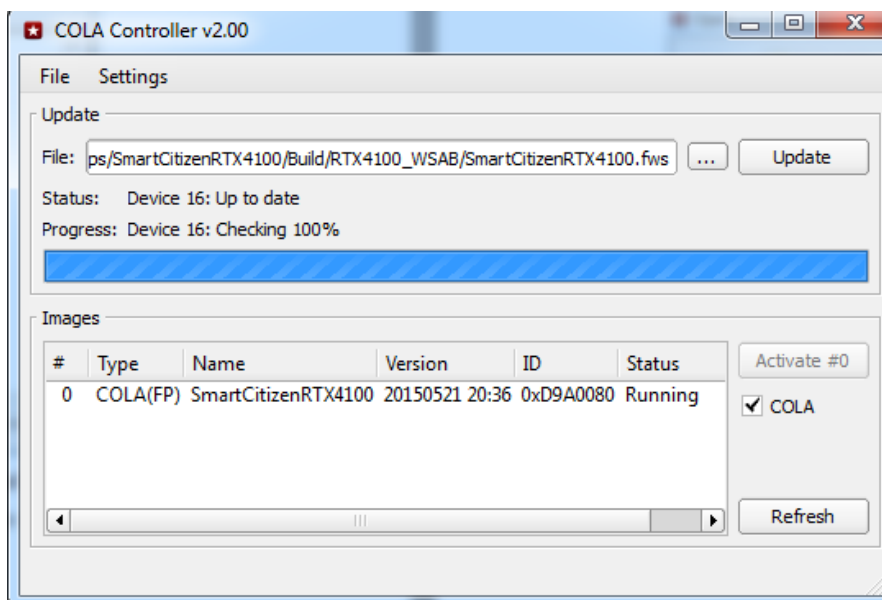


Figura 18. ColaController: Importació correcta de la colapp al mòdul

3.1.4 Depuració (Debugging)

Una vegada la colapp es importada al mòdul és necessari veure si l'aplicació funciona correctament. En cas que s'hagi compilat correctament però el funcionament d'aquesta no és l'esperat és necessari veure que ocorre internament amb les eines de depuració que disposa el programari de RTX4100. En aquest cas, l'aplicació es l'Amelie SDK trace (RSX) que permet monitoritzar els mails enviats pel sistema operatiu de RTX. Aquest aplicació es pot executar des de la carpeta AmelieSDK des de el menú d'inici o al directori C:\AmelieSdk\v1.6.0.58\Projects\Amelie\COLApps\Rsx

Es necessari iniciar la placa en mode 2 amb la mateixa configuració del RTX EAI Port server per poder utilitzar aquesta aplicació.

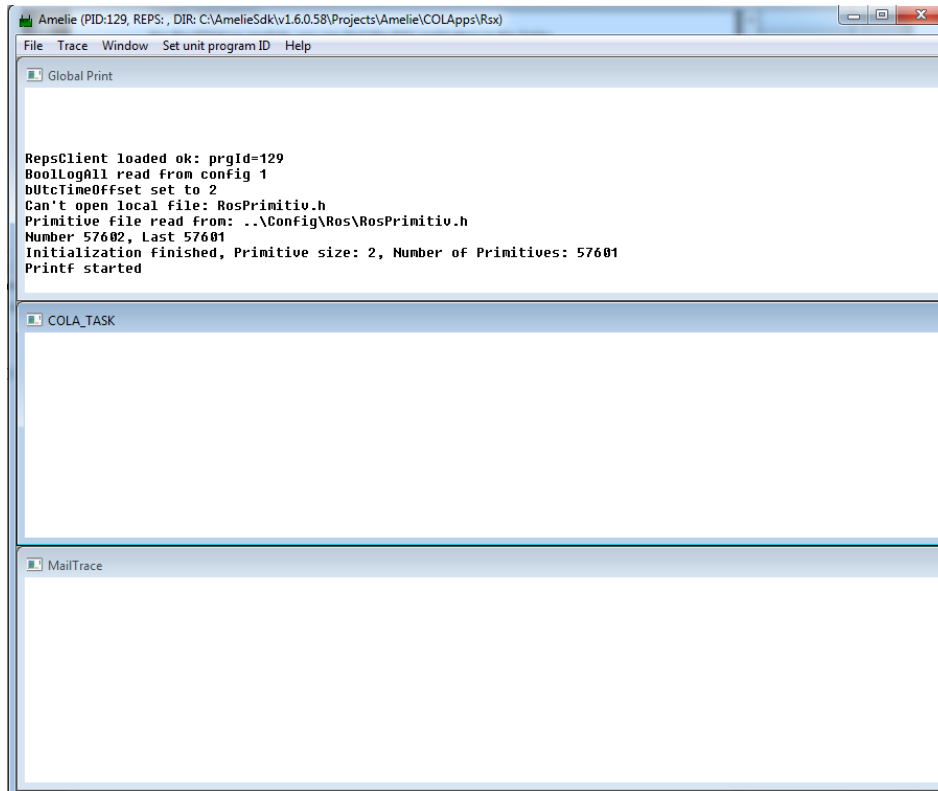


Figura 19. Aplicació Amelie SDK trace (RSX)

Una vegada oberta l'aplicació, es pot seleccionar les "colatask" que es volen monitoritzar. Per això, s'ha de seleccionar la finestra "MailTrace" i obrir el menú "Task Options". Un cop obert el menú, es seleccionaren els mails que es volen monitoritzar. Un cop finalitzat, es selecciona de nou la finestra Mail trace i s'activarà l'opció "Download and Start" per iniciar la monitorització de les colatask seleccionades.

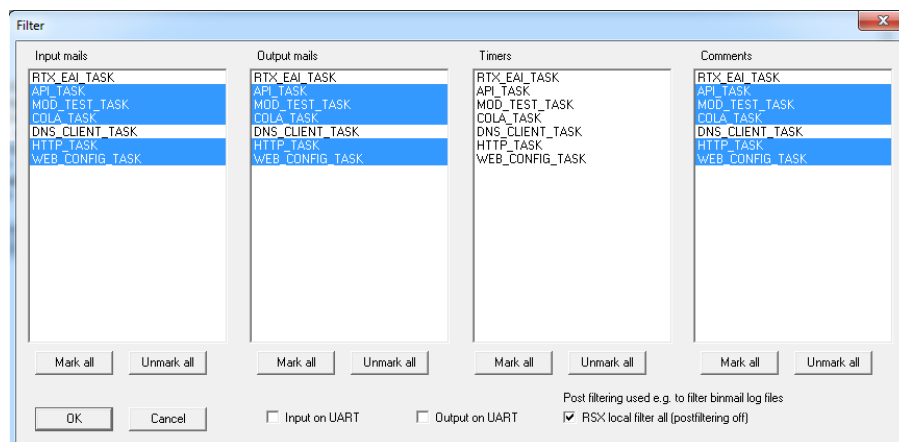


Figura 20. Configuració de les ColaTask en RSX

Un cop seleccionats els mails, es donarà de nou click dreta la finestra Mailtrace i es seleccionarà del menú l'opció "Download and Start", iniciant la monitorització de les colatask seleccionades.

```

MailTrace
15.32.44.786 ----- Mailtrace started -----
02.01.50.080_s ----- Options downloaded -----
15.32.55.978 ----- Mailtrace started -----
15.33.11.859 I: Task 9      -> API_TASK      , 5800, 00
15.33.11.867 I: API_TASK  -> COLA_TASK     , 5801, 00 00 06 00 00 00 80 00 9A 00 15 05 21 20 36 13 00
54 58 34 31 30 30
15.33.11.871 I: Task 9      -> API_TASK      , 5800, 01
15.33.11.871 I: API_TASK  -> COLA_TASK     , 5801, 15 FF
15.33.17.052 I: Task 9      -> API_TASK      , 5800, 00
15.33.17.060 I: API_TASK  -> COLA_TASK     , 5801, 00 00 06 00 00 00 80 00 9A 00 15 05 21 20 36 13 00
54 58 34 31 30 30
15.33.17.064 I: Task 9      -> API_TASK      , 5800, 01
15.33.17.064 I: API_TASK  -> COLA_TASK     , 5801, 15 FF

```

Figura 21. Missatges Colatask en RSX

Per altra banda, altre forma útil per veure si el codi s'executa de forma correcta es declarant funcions "print". L'aplicació RSX permet monitoritzar aquests missatges definint la funció RtxEaiPrintf() dins del codi del projecte

```
#define printf(...) RtxEaiPrintf(Colalf->ColaTaskId,__VA_ARGS__)
```

A continuació es defineix els printf en les seccions del codi que interessa controlar i s'executa la monitorització RSX, on es veurà els missatges dins de la finestra COLA_TASK

3.1.5 Carga de Firmware RTX

En cas que es vulgui actualitzar el firmware del modulo RTX4100 a l'última versió o solucionar un problema del mòdul , és necessari carregar la ultima versió del firmware.En el nostre cas, va ser necessari carregar de nou el firmware ja que es va executar de forma incorrecta un bucle en la colapp . Això provocava que fos impossible carregar noves colapps i el reinici continu del mòdul. El mètode per carregar el firmware és el següent:

1. Descarregar el firmwaredes de la pagina principal de RTX i descomprimir l'arxiu al directori C:
2. Activar Mode 4, premen dos cops el boto S2.Una vegada comenci a parpellejar el LED blau, serà possible carregar el nou firmware

3. Entrar per cmd al director que s'ha descomprimit y executar l'arxiu FwuRTX41xx_V0106_N0060.exe. A continuació demanarà que s'introdueixi el port on estigui connectada la placa. En aquest cas el port COM 3
4. A continuació la placa es reiniciarà i l'aplicació demanarà que torni a connectar-se la placa.

```
C:\firmware>FwuRTX41xx_V0106_N0060.exe

FwuRTX41xx.exe                               Platform version 1.6.0.60
RTX Telecom A/S                               Copyright (c) 2013 by RTX Telecom A/S

Please enter the COM port number: 3
Power on the module with the "Boot" button pressed.....
```

Figura 22 . Procés de Carga firmware RTX4100

5. Activar de nou el mode 4. Una vegada activat, començarà la càrrega del nou firmware. Si el procés ha estat correcte, acabarà amb la frase "firmware update successfully".

```
C:\firmware>FwuRTX41xx_V0106_N0060.exe

FwuRTX41xx.exe                               Platform version 1.6.0.60
RTX Telecom A/S                               Copyright (c) 2013 by RTX Telecom A/S

Please enter the COM port number: 3
Power on the module with the "Boot" button pressed.....
RTX4100 module detected
Updating platform firmware to version 1.6.0.60
Preparing...
Writing 100%
Platform firmware updated successfully!
Updating the AR4100 firmware to version 2.1.9.11.
Writing 100%
Writing 100%
AR4100 firmware updated successfully!
Erasing the CoLA image
Done
```

Figura 23. Finalització carga firmware RTX4100

6. Finalment, es comprova el correcte funcionament carregant qualsevol colap al mòdul..

3.2 ARDUINO IDE

Per realitzar les modificacions en el firmware d'Arduino, SmartCitizenSDKV, es necessari instal·lar l'entorn de desenvolupament d'Arduino (Arduino IDE), però prèviament hem d'instal·lar els drivers de la placa Arduino Due.

Arduino DUE disposa de dos ports USB, un serial i un altre de programming. La primera vegada que es connecta la placa, Windows haurà d'instal·lar els drivers automàticament. És important que una vegada s'hagi instal·lat vegem en el gestor de dispositius (Panell de control\Sistema i seguretat\Sistema\Administrador de dispositius) en què port COM s'ha instal·lat, en aquest cas, COM3.

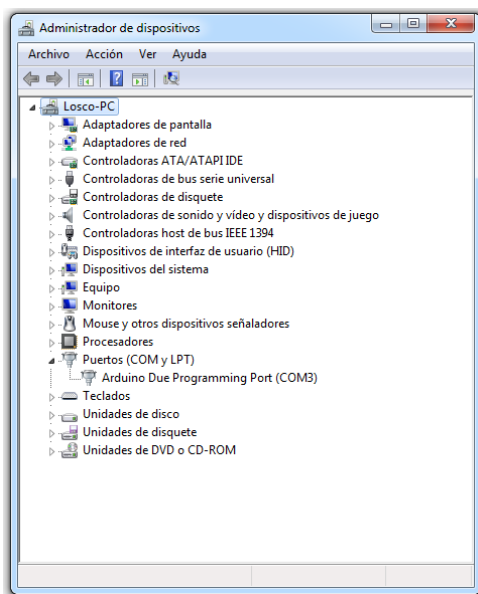


Figura 24. Drivers instal·lats Arduino DUE en port COM3

Una cop instal·lat els drivers, es procedeix a la instal·lació de l'aplicació Arduino IDE i a continuació instal·lar el core que da suport a l'Arduino Due, ja que únicament te instal·lada per defecte els cores d'AVR-Arduino. Per realitzar la instal·lació, s'ha de seleccionar el menú d'eines i a continuació "Placa\Boards Manager"

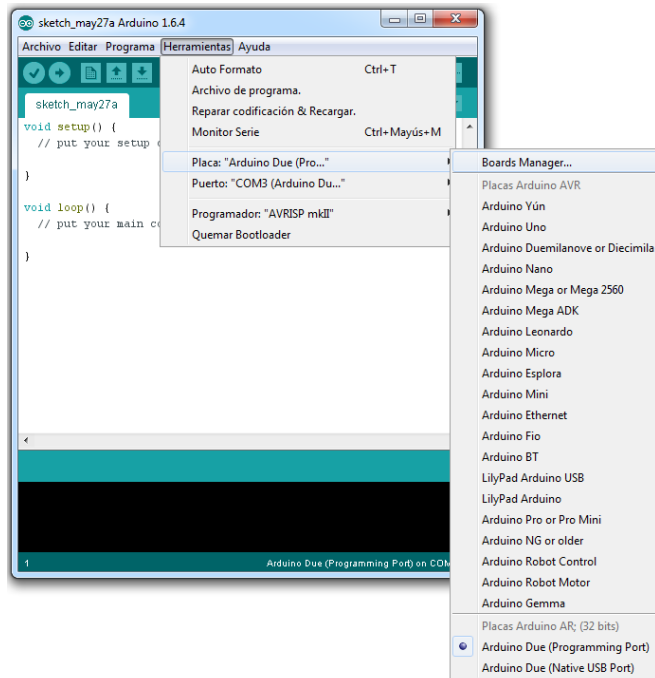


Figura 25. Instal·lació Core Arduino Due en l'aplicació Arduino IDE

En la finestra que s'obra, sortirà una llista dels cores disponibles i per instal·lar. En aquest cas es seleccionarà el SAM core i s'escollirà la versió més actual.

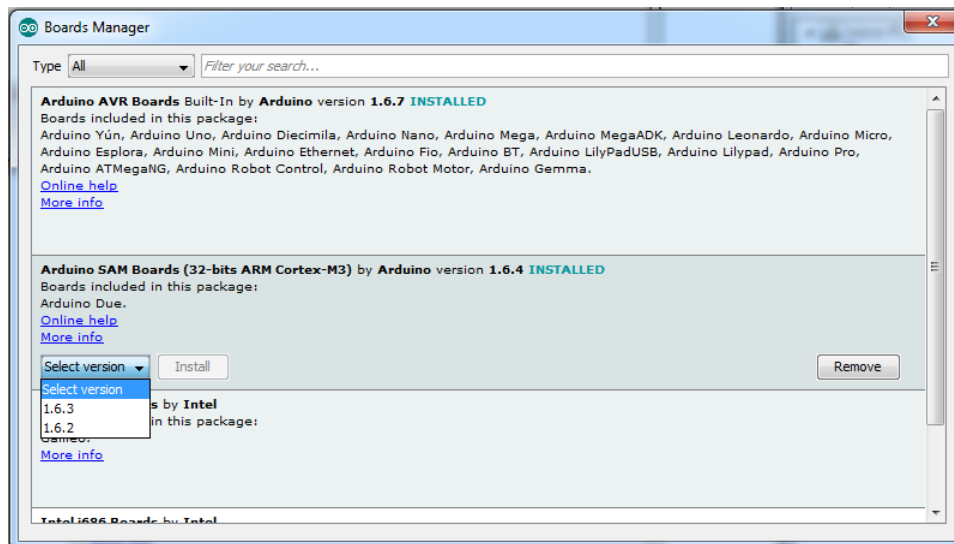


Figura 26. Board Manager Arduino IDE.

Un cop instal·lat el core, es reinicia l'aplicació i es selecciona la placa i el port des del menú d'eines. Per comprovar si es funcional, es pot activar el Monitor sèrie des de el

La instal·lació de les llibreries es pot realitzar de dues formes, des del propi Arduino IDE si es disposa d'un arxiu .zip o arrossegant els arxius directament a la carpeta de llibreries. En el nostre cas s'ha triat la segona opció. Aquesta carpeta es pot trobar des de la ruta per defecte C:\Users\Documents\Arduino\libraries.

Dins d'aquesta carpeta, es pot crear diferents carpetes que incloguin les llibreries desitjades. En aquest cas, s'ha creat la carpeta SmartCitizen que inclou els arxius Constants.h, SCDVK.cpp i SCDVK.h. Una vegada carregada les llibreries, es

recomana reiniciar l'aplicació Arduino IDE. Des del menú Programa>Include Library es pot veure i afegir totes les llibreries disponibles en els futurs projectes/sketchs

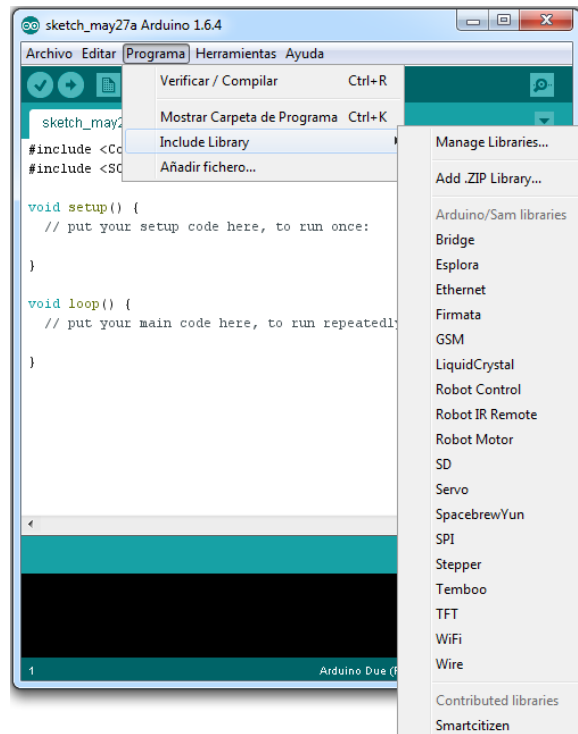




Figura 28. Llibreries instal·lades Arduino IDE

Un cop instal·lades les llibreries, es procedeix a obrir l'arxiu SmartCitizen_DVK.ino en l'aplicació per poder ser importat a la placa. Normalment es poden programar directament des de el propi sketch/projecte, però les llibreries faciliten el procés, permetent un disseny modular i reutilitzable. Dins d'aquesta arxiu es defineixen les llibreries que s'utilitzessin i el bucle perquè s'executi infinitament fins que s'interrompi o apagni el dispositiu.

Per importar el firmware primer s'ha de verificar el codi amb el botó  Verificar y finalment compilar y carregar amb el botó  Subir.

4 PROCES DE DESENVOLUPAMENT

Com ja s'ha dit, per falta de temps s'ha tingut que modificar el pla de treball, i per tant el desenvolupament de l'aplicació .

Les modificacions del firmware SmartCitizen consisteixen en la integració de les següents parts dels codi:

- **Recepció comandes SPI:** Es defineix un protothread fill que serà cridat dins del protothread principal del firmware, el qual s'executarà infinitament fins que arribi el valor de la comanda SPI destinada a l'activació del mode Servidor WEB i el mode Soft AP. Aquesta funcionalitat no ha sigut possible per problemes que no s'han aconseguit resoldre . En la secció referent a la implementació d'aquesta part del codi hi ha mes informació sobre el problema. L 'implementació està comentada en el codi.
- **Mode Soft AP:**En un principi, l'objectiu era creauna funció que seria cridada quan arribi la comanda corresponent a aquest mode. Per els motius explicats en l'anterior punt , el codi d'aquest comanda actualment no es pot activar, per tant anirà comentada. Actualment el mode Soft ap va unit a la funció que inicialitzà el servidor web.
- **Mode Servidor WEB:** Per facilitar la modificació del codi HTML es defineix el codi del servidor web com una llibreria externa, on es crearà un arxiu header que serà definit al principi del codi del firmware principal. La inicialització del protothread esta definit dins de la comanda i a part d'activar el Servidor Web, també activarà el mode Soft AP i la recepció de temperatures

Per un altre banda, les modificacions realitzades al firmware de l 'Arduino, SmartcitizenSDK.ino, tenen la funció d' enviar els valors temperatures i humitat del sensor i les comandes al mòdul RTX mitjançant la comunicació SPI. Cal recordar que en una comunicació SPI ha de haver-hi un Master i un SLAVE, en aquest cas, el master serà la placa Arduino, i el slave serà el RTX.

En el annex es pot veure el diagrama de flux del sistema dissenyat.

4.1 MODE SOFT AP

En aquesta secció es descriu el procediment per implementar el mode Soft AP

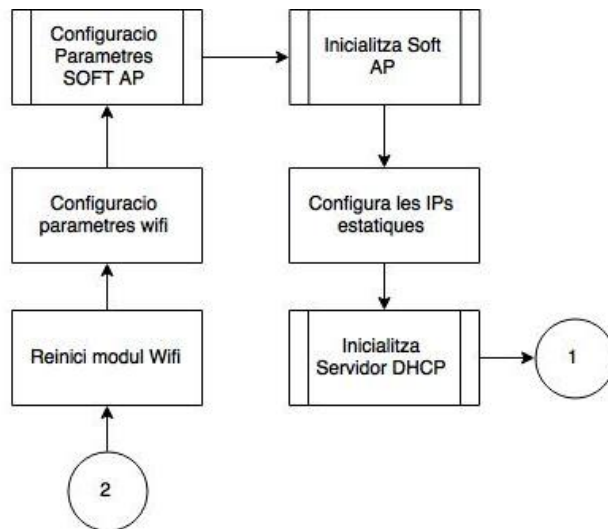


Figura 29. Diagrama Flux SOFT AP

4.1.1 Llibreries utilitzades

Les llibreries necessàries per desenvolupar el mode SOFT AP es troben dins de Projects\Amelie\Components\PtApps \AppWifi.h.

Funció	Descripció
rsuint8* AppWifiApGetSoftApSsid(void);	Valor del ssid del softap
rsbool AppWifiIsSoftAp(void);	Estat que esta el softap (activat/desactivat)
RsStatusType AppWifiApSetSoftApInfo(rsuint8 *Ssid, ApiWifiSecTypeType SecurityType, rsbool HiddenSsid, rsuint8 KeyLength, rsuint8 *Key, rsuint16 Channel, rsuint32 ConnInactPeriod, rsuint8 *CountryCode, rsuint16 BeaconInterval);	Especifica els paràmetres del Soft AP
PT_THREAD(PtAppWifiStartSoftAp(struct pt *Pt, const RosMailType* Mail));	Protothread per iniciar el SOFT AP

Tabla 4. Llibreries AppWifi.h per implementar SOFT AP

4.1.2 Desenvolupament

La definició del thread que inicialitza el mode de SOFT AP es troba dins de la llibreria del Servidor WEB. Això degut a que les probes per la realització de formularis estaven dirigides a la modificació dels paràmetres SOFT AP. No ha sigut possible realitzar un codi que realitzi correctament les modificacions i per tant, la part del codi destinat a aquestes funcions estan comentades (línies X X) per poder ser millorades en futurs treballs. La part que realitza les modificacions estan analitzades dins de la secció del servidor web.

Aquestes circumstàncies afecten al mode de definició dels paràmetres del SOFT AP. Segons el mètode de programació, els paràmetres es defineixen com valors estàtics (define SOFT_AP_SSID) o es defineixen com valors “dinàmics”, els quals son definits com variables globals que aniran emmagatzemades a la NVS i que permetran ser modificats (AppData.Ssid). En aquest cas, i pels motius esmentats, els paràmetres són definits de forma estàtica.

A continuació es descriu el codi implementat:

- Abans d'iniciar el Soft AP, és necessari iniciar/reiniciar el mòdul WIFI.
- A continuació es defineixen els paràmetres del SOFT AP amb els valors definits anteriorment. En aquest cas, el valor de ssid esta format pel SSID definit y els valors de la mac del mòdul, ja que aquestes son úniques per cada mòdul. Per un altre banda, no s'ha definit cap tipus de seguretat ni clau (Accés lliure).
- S'inicialitza el SOFT AP, el servidor DHCP i es defineix la IPestàtica del servidor web
- Finalment s'espera a que un client connecti

Referent a la part desenvolupada relacionada amb la modificació dels paràmetres, el disseny del mode SOFT AP es una mica diferent:

- Al inicialitzar el mòdul o al no ficar cap paràmetre en l'input de l'aplicació web, es considerarà com un valor NULL i per tant es defineix el valor per defecte del SSID (SSID +MAC).El tipus de seguretat i la clau no estan definides per defecte i per tant es d'accés lliure.
- El valors modificables es guarden a la NVS, i cada cop que es reinicia el mòdul, el valor es llegit des d'aquesta.

Aquestes funcions es poden veure comentades al codi, ja que es pretén modificar en futurs treballs.

4.2 SERVIDOR WEB

Per realitzar el servidor web, es pren com a referència el projecte d'exemple Web server. Un dels punts importants es decidir quina forma es vol implementar el servidor web. L'objectiu principal, tant del servidor web com la pròpia interfície web, es que siguin independents i per tant no s'utilitzi recursos externs per ser executat. Això vol dir, que s'ha de desenvolupar un servidor/interfície web que pugui ser executat amb els propis recursos del mòdul RTX, i per tant s'haurà d'avaluar les limitacions..

Segons l'analitzat a partir dels codis d'exemples i la documentació, es pot desenvolupar el servidor web de dues formes:

- Modificació de la pàgina web de configuració per defecte "EASY WEB" a partir de les llibreries AppWebConfig.
- Creació del servidor/interfície web a partir de les llibreries ApiHttp.

Aquestes opcions no són les úniques, ja que es possible que hagin d'altres les quals no s'han analitzat amb profunditat.

Per desenvolupar el projecte s'ha escollit la segona opció. Igualment, s'han realitzat proves amb la primera, on en el següents s'analitzaren, enumerant les limitacions i avantatges.

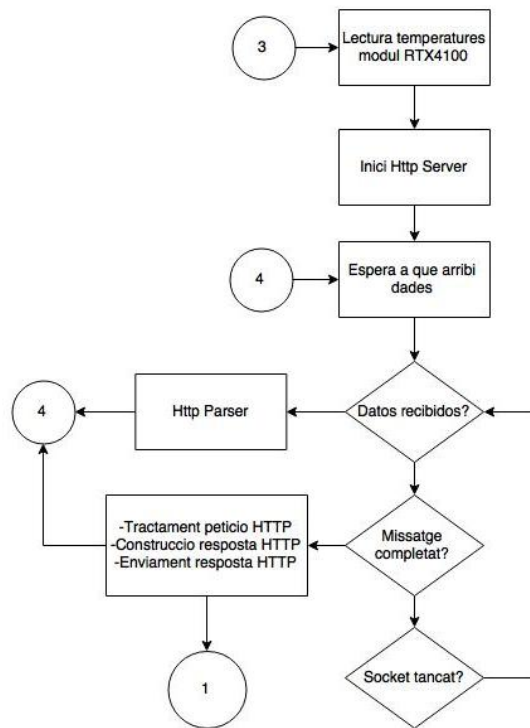


Figura 30. Diagrama flux Servidor Web

4.2.1 Llibreries utilitzades

Tant en la fase de proves como en la fase final del servidor, la pagina web es realitzada amb les llibreries /Include/IOT/App/AppiHttp.h . Per generar la pagina web s'utilitzarà 3 estructures bàsicament

Header
Descripció: Funció que s'utilitza per generar per afegir camps de capçalera a HTTP alMissatge HTTP que s'envia
Retorna: Número de bytes afegits
Estructura <pre>typedef rsuint32 (*AhAddHeaderCbType)(rsuint8 *BufferPtr, rsuint32 BufferLength,rsuint8* DataPtr,rsuint32 Instance);</pre>

Tabla 5. Estructura Header

Cos (Body)
Descripció: Funció que s'utilitza per annexar dades del cos al missatge HTTPenviat.
Retorna: Número de bytes afegits
Estructura: typedef rsuint32 (*AhAddBodyCbType)(rsuint8 *BufferPtr,rsuint32 BufferLength,rsuint32 Offset,rsuint8* DataPtr, rsuint32 Instance)

Tabla 6. Estructura Body (Cos)

Recurs de Resposta HTTP
Descripció: Funció que s' utilitza per manejar les peticions d' un recurs determinat. Les dades de recursos són enviades directament per l'ús del missatge API_HTTP_SERVER_SEND_RESPONSE_REQ El servidor HTTP genera un missatge de resposta HTTP amb l'estat: <ul style="list-style-type: none"> • 404 si el client sol·licita un recurs desconegut • 501si aquesta devolució de trucada retorna RSS_NOT_SUPPORTED • 500 si aquesta devolució de trucada retorna RSS_FAILED.
Retorna: Aquesta funció ha de retornar RSS_SUCCESS si genera les dades dels recursos sol·licitats amb èxit, RSS_NOT_SUPPORTED si el mètode HTTP de la petició no està justificada, o RSS_FAILED si la funció falla per generar les dades sol·licitada.
Estructura: typedef RsStatusType (*AhResourceCbType)(AhHttpMethodIdType HttpMethod, rschar *PathPtr,rschar *QueryPtr, rsuint32 Instance);

Tabla 7. Estructura Recurs de Resposta HTTP

Les funcions principals utilitzades es troben dins de la mateixa llibreria (ApiHttp.h)

Funció	Descripció
void SendApiHttpServerInitReq (RosTaskIdType Src, rsuint32 Port, AhCallbackType Callback	Inicia el servidor HTTP
void SendApiHttpServerSendResponseReq (RosTaskIdType Src, rsuint32 Instance, rsuint16 StatusCode, rschar* ReasonPtr,	Funció que envia/genera la pàgina web sol·licitada

rsuint32 BodyDataLength, rsuint8* DataPtr, AhAddHeaderCbType AddHeaderCb, AhAddBodyCbType AddBodyCb);	
void SendApiHttpServerAddResourceReq (RosTaskIdType Src, rsuint32 Instance, rschar* PathPtr,AhResourceCbType CallBack);	Afegeix la pàgina Web sol·licitada
rsuint8 HttpSplitQueryString(rsuint8 Count, AhQueryDataType Query[1], rschar *QueryStringPtr);	Funció que divideix la cadena de caràcters d'una sol·licitud HTTP.

Tabla 8. Llibreries ApiHttp.h per implementar Servidor Web

4.2.2 Proves: Desenvolupament del servidor web amb EASY WEB

Aquest mètode permet modificar la pàgina de configuració “EASY WEB”, que permet la modificació dels paràmetres IPs (DHCP o Ip estàtica) i la configuració dels paràmetres per la connexió a un AP, on en la mateixa pàgina es pot veure les APs disponibles que detecta el mòdul.

L'execució de la web únicament es realitza en mode SOFT AP i per sortir d'aquesta és necessari reiniciar el mòdul. Totes les aplicacions d'exemple criden aquesta funció abans d'executar la resta de l'aplicació per configurar els paràmetres del AP.

Segons la documentació del fabricant, aquesta web permet ser modificada i a la vegada permet afegir noves pàgines, mitjançant l'execució de funcions a través del WebServer. Aquestes funcions permet la modificació dels paràmetres del SOF AP, com el SSID, el tipus de seguretat, etc. Per realitzar aquestes modificacions es necessari definir les funcions que es troben dins de AppWebConfig.H. Aquesta llibreria s'encarrega de construir la web i modificar totes les dades que són enviades des del WebServer. Totes les funcions han de ser definides abans d'executar el thread que s'encarrega d'iniciar el AppWebConfig (PtAppWebconfig).

Per un altre banda, per carregar les pàgines web implementades al Webserver es necessari executar la següent funció que es troba dins de la llibreria ApiWebConfig.h

```
SendApiWebConfigAddPageReq(COLA_TASK, "/prueba", "Temperaturas", OnMainPage);
```

A continuació s'enumeren les funcions principals de la llibreria:

Funció	Descripció
void AppWebConfigSetSsid(rsuint8 *SsidPtr);	Canvia el valor SSID
void AppWebConfigSetKey(rsuint8 *KeyPtr	Modificació del Password
void AppWebConfigSetChannel(rsuint16 Channel);	Modificació Canal
void AppWebConfigSetSecType(ApiWifiSecTypeType SecType)	Tipus de seguretat
void AppWebConfigSetTitle(rschar *TitlePtr);	Canvia el títol principal de la web
void AppWebConfigSetInfoText(rschar *InfoTextPtr);	Introdueix informació sota el títol principal
void AppWebConfigAddAppSetupPage(rschar* TitleTextPtr, rsuint8 SettingsCount, AppWebConfigSettingsType *SettingsPtr);	Aquesta funció es utilitzada per afegir noves pàgines de configuració
PT_THREAD(PtAppWebConfig(struct pt *Pt, const RosMailType* Mail, RsListEntryType *PtList));	Inicia SOFT AP, DHCP Server i el constructor web

Tabla 9. Llibreries AppWebConfig.H per implementar Servidor Web amb EASYWEB

Aquest sistema té les següents limitacions:

- Appwebconfig inicia el servidor web, servidor DHCP y mode SOFT AP sense la possibilitat de separar el codi per executar independentment el mode SOFT AP i el servidor web a partir de comandes SPI, a menys que es modifiqui completament la llibreria.
- No permet modificar la presentació de la interfície web, per tant, visualment és molt pobre

Demostració Servidor WEB-EasyWeb

La pàgina principal es pot veure el títol principal "RTX Web Server" que va ser enviat/modificat des del Web server. Sota del títol principal es pot llegir SSID del SOFT AP i la temperatura del sensor.

A continuació es presenta un menú amb les diferents webs que es poden accedir:

- Reboot the module: Reinicia el mòdul RTX
- Set Wifi Acces Point: Menú per configurar el mòdul com client i permet introduir les dades necessàries per connectar-se a qualsevol AP que detecti el mòdul. També mostra la llista de AP disponibles.
- Temperatures: Llista de temperatures que es poden llegir des del mòdul RTX
- Set Ip Config: Configura l'equip per què utilitzi una IP estàtica (Configuració de la IP, DNS, Gateway, Subred)

Finalment mostra al final de la pàgina la informació del sistema. Com es la COLA que s'està executant, la versió d'aquesta, la mac y la versió del firmware del mòdul.

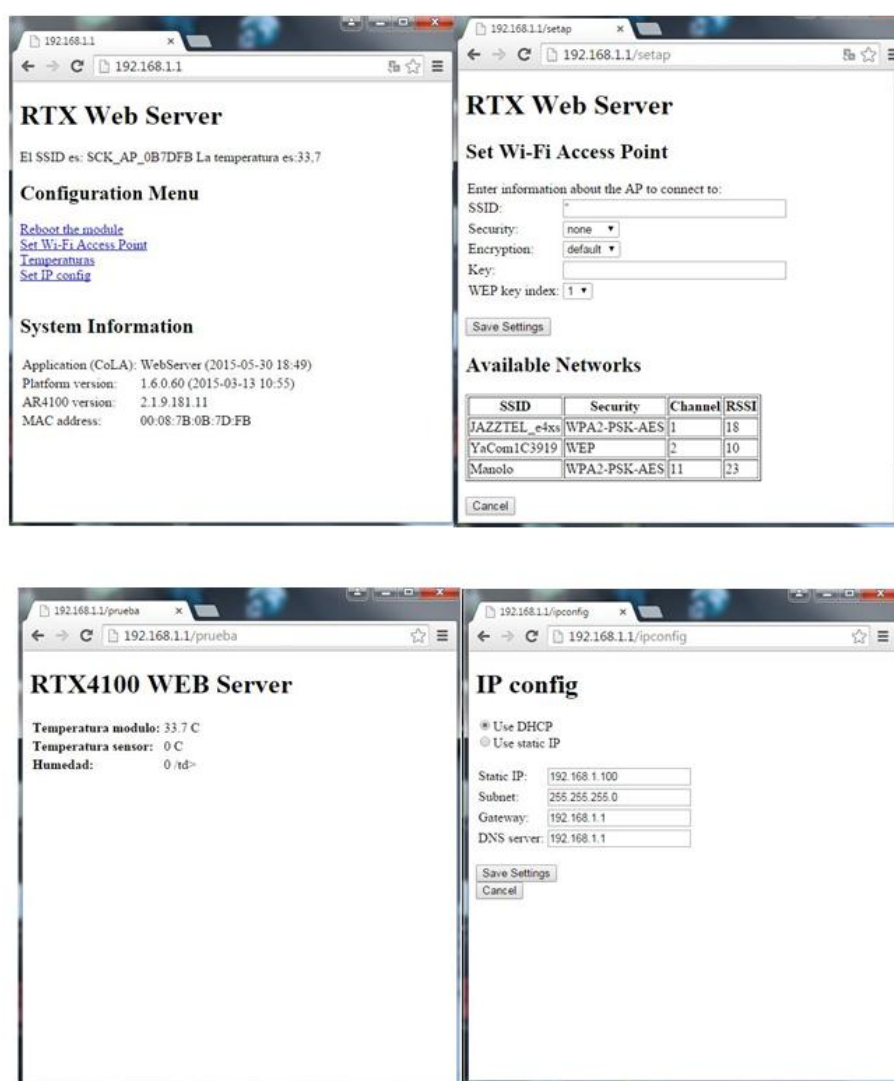


Figura 31. Demostració Servidor Web amb EASYWEB

En l'annex es pot trobar el codi de prova

4.3 DESENVOLUPAMENT DEL SERVIDOR WEB FINAL

4.3.1 Creació de pàginesHtml

Dins de la estructura del Body es defineix el codi HTML&CSS. Al disposar d'un buffer de 32 bytes (rsuint32), el contingut d'aquestes pàgines és limitat,. Les proves que s'han realitzat per incloure pàgines amb presentació demostren que únicament es possible generar codi CSS de forma limitada, per tant, s'ha realitzat de forma mínima i optimitzada. Per altre banda, no ha sigut possible incloure petits scripts amb javascript, ja que les limitacions del buffer no permeten fer una correcte lectura dels paràmetres.

La definició del html es realitza com una cadena de caràcters. Cada línia del codi s'introdueix dins de cometes dobles (" "). Però hi ha part del codi que l'interpretador no podrà entendre i s'ha de modificar:

- **Paràmetres del html dins de cometes.** Per exemple, la definició dels valors. En aquest cas s'ha de definir el paràmetre dins de " \"
- **Símbols.** Els símbols com "%" o "o" no poder ser interpretats directament des de el html i són necessaris definir-los externament des d'una variable.

Un cop generat el codi HTML, aquest s'introdueix dins del buffer, juntament amb els paràmetres que s'han de llegir (o introduir en cas dels formularis). Aquest buffer és el nombre de bytes que enviaràal client.

Per solucionar els problemes de limitacions de buffer, s'han generat pàgines independents en la que es divideix el contingut en cadascuna d'elles:

- **Pàgina de Temperatures (Inici):** Conté la informació de les últimes lectures del sensors.



Figura 32. Pagina Web Final. Visualització de Temperatures

- **Pàgina d' Informació del sistema:** Conté informació del sistema.



Figura 33 Pagina Web Final. Informació del sistema

- **Pàgina de Configuració Soft AP:** Visualització i modificació dels paràmetresde configuració Soft AP.

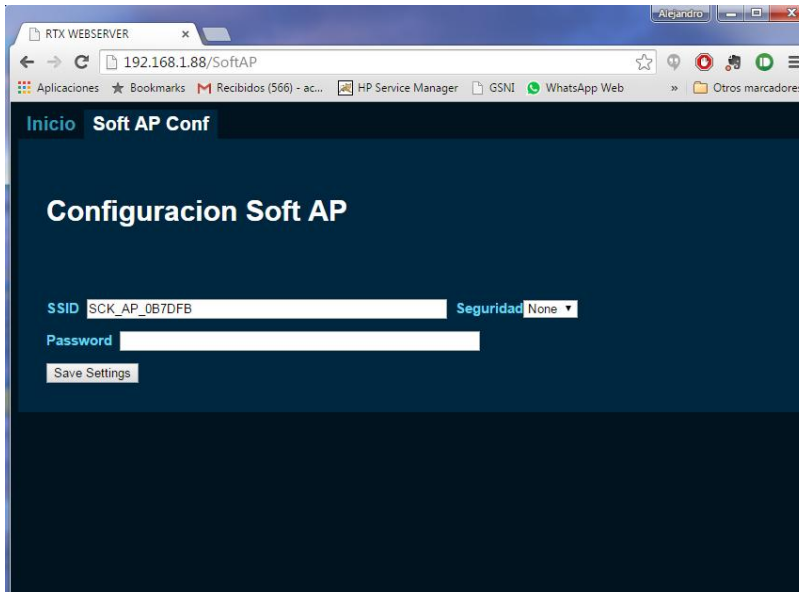


Figura 34. Pagina Web Final. Configuració de paràmetres SOFT AP

4.3.2 Interpretació de les peticions HTTP client-servidor

Cadascuna de les pàgines és necessari la funció de resposta per part del client-servidor. Quan el servidor rep una petició del client (GET), aquest ha de respondre de forma afirmativa si ha pogut generar els recurs de la pagina web (Resposta "200"). A continuació s'enumera com es realitza aquest procés dins del codi:

- Condició de que si al rebre una petició diferent a GET, aquest enviï un missatge de que no s'ha pogut generar la petició del recurs
- En cas afirmatiu. Es defineix la funció per el enviament del recurs per la generació de la pagina web amb el codi HTML generat. Aquesta funció esta formada per el codi de confirmació de enviament "200", el header, i la pròpia pagina html)
- Finalment retorna el valor RSS_SUCCESS on es confirma la correcta generació del recurs.

Aquesta part del codi s'utilitza únicament amb les pàgines on es realitza consulta dels paràmetres del mòdul. En la implementació de formularis, la interpretació de les peticions client-servidor es diferent.

4.3.3 Implementació de formularis

Aquesta part del codi està comentat ja que no s'ha pogut finalitzar amb les funcionalitats esperades. Igualment s'analitzarà el codi que es volia implementar

L'objectiu inicial dels formularis és modificar els valors de ssid, tipus de seguretat i el claua través de l'aplicació web. Un cop modificades, és necessari el reinici del mòdul RTX per fer efectiu el canvi. La resta dels paràmetres no es consideren necessàries ser modificades (country code, beacon, channel).

Dins de la pàgina de configuració de Soft AP s'implementa:

- Es defineix el mètode GET a l'inici del codi html per enviar les dades en forma d'URL un cop confirmada les modificacions amb el boto submit.
- Input text : Camp per inserir el nom del SSID. En cas de no ficar cap valor, aquest s'interpretarà com un valor NULL i es modificar el SSID per el valor per defecte. Per defecte es mostra l'actual SSID del mòdul.
- Menú desplegable per escollir el tipus de seguretat. En cas d'escollir l'opció afirmativa, serà necessària la introducció de la clau i confirmar els canvis. Si l'opció és negativa, únicament s'haurà de confirmar els canvis
- Boto Submit: Confirma les modificacions realitzades al prémer el botó.

Un cop definida la pàgina de modificació, és necessari interpretar el bytes del buffer del html creat i l'URL enviada mitjançant el mètode GET. En aquest cas és una mica diferent a la interpretació de les peticions client-servidor. A continuació s'analitzarà el codi:

- Primer és defineix el buffer de tipus query el qual s'utilitzarà per emmagatzemar les dades rebudes.
- Es defineix la funció necessària per separar els valors dins la URL enviada. Aquests valors son guardats dins del buffer query. Aquesta funció conta el numero de paràmetres introduïts i passa el valor a la variable "comptador".
- Es defineix una condició de que si el comptador és el mateix al nombre màxim del número de paràmetres a modificar, es realitza l'anàlisi dels paràmetres al buffer:
 - En cas de que SSID=NULL (camp del input buit): Es carrega l'últim valor guardat al NVS
 - En cas de que SSID sigui diferent a NULL: Es guardarà el nou valor a la NVS

- Condició de que si el tipus de seguretat és igual a NONE: El tipus de seguretat serà NONE i no serà necessari llegir el valor de la clau.
- En cas contrari es modificarà el tipus de seguretat per AWST_WPA2 i passarà a la condició llegirà el valor de la clau.
- Condició de la clau, en cas de que el valor sigui nul s'enviarà el codi de resposta al client i s'obrirà una pàgina d'error indicant que falta la clau i no es farà cap canvi. En cas de que hagi algun valor, es guardarà aquest en la NVS.
- Finalment s'envia el codi de resposta al client i els recursos html per generar la pàgina de confirmació dels canvis, avisant de que és necessari reiniciar el mòdul.
- En cas de que no es compleixi les anteriors condicions, es generarà la pàgina de Configuració SOFT AP, ja que significa que no s'ha generat cap enviament de modificació de dades.

Per algun motiu, el codi no funciona correctament, i això que la compilació d'aquest és correcte sense sortir cap advertència ni error. En futurs treballs s'espera millorar el codi i descobrir finalment quin es el motiu del mal funcionament.

4.3.4 Iniciació del Servidor web

La fase de inicialització del servidor web es realitza en el protothread principal de l'aplicació. A continuació es descriu el procés d'inici i generació de les pàgines web:

- S'envia el recurs de inici del servidor HTTP mitjançant una COLATASK i s'especifica el port per realitzar la connexió amb el client
- Espera a que es rebí un paquet indicant que la connexió s'ha realitzat amb èxit per el port definit
- Un cop es rep el paquet, es defineix la condició que si s'ha pogut generar el recurs sol·licitat, s'inicià el servidor definint la instància HTTP on s'enviarà les pàgines webs.
- Finalment es generen les pàgines web, enviant la instància HTTP, la ruta d'on estarà ubicada les pàgines i el recursos generats.

4.4 COMUNICACIÓ SPI ARDUINO-RTX

4.4.1 Llibreries Arduino-SPI

És necessari conèixer les funcions que utilitza Arduino per implementar la comunicació SPI. Aquestes són:

- SPI.Begin () : Inicia el bus SPI. Amb Arduino Due es pot especificar el pin SS
- SPI.End() : Finalitza la comunicació SPI.
- SPI.beginTransaction(SPISettings (speed,dataOrder,datamode)). Inicia la comunicació SPI amb les propietats definides
 - Speed= Velocitat de la comunicació.
 - dataOrder=Ordre de les dades. MSBFIRST (bit mes significatiu) o LSBFIRST (bit menys significatiu)
 - Datamode=Mode d'enviament de les dades SPI_MODE0, SPI_MODE1, SPI_MODE2, or SPI_MODE3
- SPI.endTransaction() = Finalitza l'enviament de dades
- SPI.setBitOrder(order) = Especifica l'ordre de les dades (MSBFIRST o LSBFIRST)
- SPI.setClockDivider(divider) = Estableix el rellotge divisor en relació amb el rellotge del sistema . En les plaques AVR , els divisors disponibles són 2 , 4 , 8 , 16 , 32 , 64 o 128 mitjançant la definició SPI_CLOCK_DIV<valor del divisor>. El valor per defecte es el DIV4, que estableix el rellotge SPI com un quart de la freqüència del rellotge del sistema). En Arduino DUE , el rellotge del sistema es pot dividir per valors d'1 a 255. El valor per defecte és 21, que estableix el rellotge a 4 Mhz.
- SPI.setDataMode(modo) = Especifica el mode que envia les dades(SPI_MODE0, SPI_MODE1, SPI_MODE2, or SPI_MODE3)
- SPI.transfer():Transfereix (envia i rep) un byte sobre el bus SPI . Funció necessària per transferir dades

En la funció beginTransaction es pot especificar l'ordre de les dades (dataOrder) i la manera d'enviar-les (DataMode). Aquest últim es pot especificar quatre modes, els quals es poden veure en la taula següent, prenent com a referència el descrit al capítol “*estat de l'art*” referent a la polaritat del rellotge i fase en la secció de comunicació SPI.

Mode	Clock Polarity (CPOL)	Clock Phase (CPHA)
SPI_MODE0	0	0
SPI_MODE1	0	1
SPI_MODE2	1	0
SPI_MODE3	1	1

Tabla 10. Modes d'enviament de dades SPI

El mode que es seleccionarà serà el SPI_MODE0 per la comunicació entre l'Arduino i el mòdul RTX4100.

4.4.2 Modificació Firmware Arduino (Master)

Enviament de Temperatures

La funció SCDVK::Main() serà modificada per realitzar l'enviament de temperatures. Actualment aquesta part del codi s'executa en mode normal on únicament mostra les temperatures actuals i el missatge de comprovació de disponibilitat de targeta SD.. A continuació es descriu les modificacions realitzades al codi:

- Primer es defineix l'inici del bus SPI i l'activació del pin RTX. Aquestes definicions es realitzen a la funció SCDVK::begin(), on s'encarrega d'iniciar totes les interfícies.
- A continuació es defineix les propietats del canal SPI a la funció SCDVK::Main() , on s'assigna la velocitat, l'ordre dels bytes i el mode. En aquest cas, es definirà la configuració per defecte i una velocitat d'un quart del processador.
- Es defineix els bytes que rebran/enviaren les dades del canal SPI. Les dades a enviar serà els valors de la temperatura i la humitat.
- Finalment es tanca la comunicació, tancant la comunicació SPI i desactivant el pin de RTX.

Enviament comandes SPI

Respecte a les comandes SPI, les modificacions en el firmware únicament afectarà a la modificació del funcionament del mode 2. Aquest mode és activat quan es pren el boto 1. Un cop activat, el LED verd començarà a parpellejar i (per defecte) s'activarà el mode

terminal. Aquest mode es considera prescindible a comparació de la resta (carga de colapps i modificació de firmware) i per tant serà modificat per poder configurar l'enviament de comandes SPI.

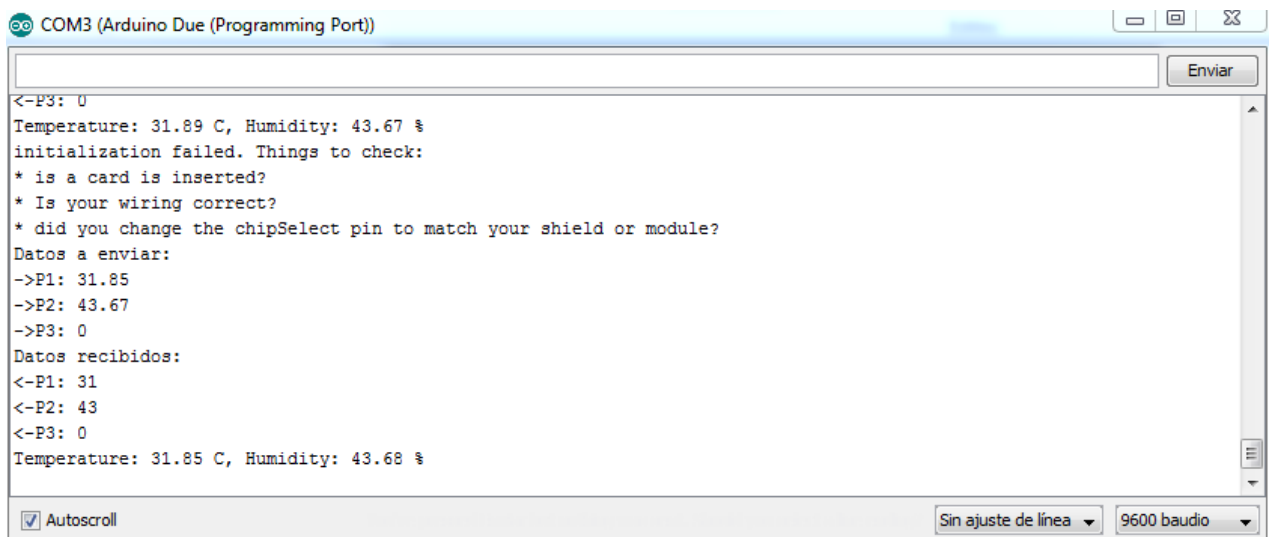
Únicament es podrà enviar dos comandes SPI, ja que la placa únicament disposa de dos botons i únicament podrà ser utilitzat el boto 1. El funcionament de les comandes és el següent:

- El mode normal (cap boto activat) la placa enviarà una byte (p3) amb valor 0 pel canal SPI. Els leds actius seran el verd i el groc
- Al prémer el boto 1, el mode 2 s'activarà i començarà a parpellejar el LED verd. El valor del byte amb valor 0 canviarà i s'enviarà el mateix byte però amb valor 16, el qual es valor de la comanda d'activament en la colapp.
- Per tornar al mode normal, es tornarà a prémer el boto, on el valor del byte tornarà a ser 0.

Les modificacions del codi son les següents

- Definició de variable global byte command = 0
- Es defineix command=0 en el bucle d'activació del mode normal
- Es defineix command =16 en el bucle d'activació del mode terminal i l'activació de la funció main().
- Modificació de la funció main::SCVDK () amb la definició d'una nova línia per la transferència SPI de la comanda, on es guardarà el valor en el byte p3.

A continuació es pot veure com realitza correctament l'enviament de temperatures:



```
COM3 (Arduino Due (Programming Port))
<-P3: 0
Temperature: 31.89 C, Humidity: 43.67 %
initialization failed. Things to check:
* is a card is inserted?
* Is your wiring correct?
* did you change the chipSelect pin to match your shield or module?
Datos a enviar:
->P1: 31.85
->P2: 43.67
->P3: 0
Datos recibidos:
<-P1: 31
<-P2: 43
<-P3: 0
Temperature: 31.85 C, Humidity: 43.68 %
```

Figura 35. Enviament correcte de temperatures mitjançant comunicació SPI

El codi complet del firmware de l'Arduino es pot veure en el annex.

4.4.3 Llibreries SPI RTX

Un cop definides les funcions necessàries per modificar el firmware d'Arduino, s'ha de fer el mateix pel mòdul RTX. Gràcies a l'ajuda de SmartCitizen, es va descobrir un error el qual no permetia la comunicació del mòdul amb la databoard. El problema era que el mòdul RTX estava funcionant com a Master i no es possible tenir dos dispositius en aquest mode en una comunicació SPI. En aquest cas, el master es la placa Arduino i RTX actuarà com esclau.

Al principi, el proveïdor ens va indicar que aquest problema era provocat per un error de fabricació i era necessari modificar la placa dessoldant el component erroni. Finalment, l'error es podia solucionar modificant les llibreries dels drivers SPI del mòdul RTX. (.\\v1.6.0.58\\Projects\\Amelie\\Components\\Drivers\\DrvSpiSlave.c.) La modificació es realitza a les línies 274-276, afegint :

```
// Enable Chip Select Invert  
USART->CTRL |= USART_CTRL_CSINV;
```

Això permet declarar el mòdul RTX com a esclau sense necessitat de modificar el hardware.

A continuació es nombraran les funcions necessàries per realitzar la comunicació SPI. Aquestes es troben a l'arxiu DrvSpiSlave.h

Funció	Descripció
PT_THREAD(PtDrvSpiSlaveInit(struct pt *Pt, const RosMailType* Mail, rsuint32 BaudRate));	Inicialitza comunicació SPI
PT_THREAD(PtDrvSpiSlaveClose(struct pt *Pt, const RosMailType* Mail));	Finalitza comunicació SPI
void DrvSpiSlaveInit(rsuint32 BaudRate);	Funció d'inici
rsuint16 DrvSpiSlaveRxGetSize(void);	Obté el número de bytes emmagatzemats al buffer (RX)
rsuint16 DrvSpiSlaveRx(rsuint8 *RxBufferPtr, rsuint16 RxBufferLength);	Llegeix les dades del buffer RX

<code>void DrvSpiSlaveRxFlush(void);</code>	Buida el buffer RX
<code>RsStatusType DrvSpiSlaveTxStart(rsuint8 *TxDataPtr, rsuint16 TxDataLength);</code>	Inicialitza el buffer de transmissió (TX)

Tabla 11. Llibreries DrvSpiSlave.h per implementar comunicació SPI en la colapp

4.4.4 RTX (Slave) -Recepció de temperatures i comandes SPI-

El desenvolupament de la colapp per la captura de temperatures i comandes al mòdul RTX es realitza amb la definició d'un protothread el qual executarà en bucle infinit fins que rebí la comanda per desactivar la lectura. El codi és defineix al firmware del SmartcitizenRTX4100 per fer la lectura de comandes a l'inici de l'aplicació i un altre molt semblant dins del Servidor web, que te la funció de realitzar la lectura de temperatures.

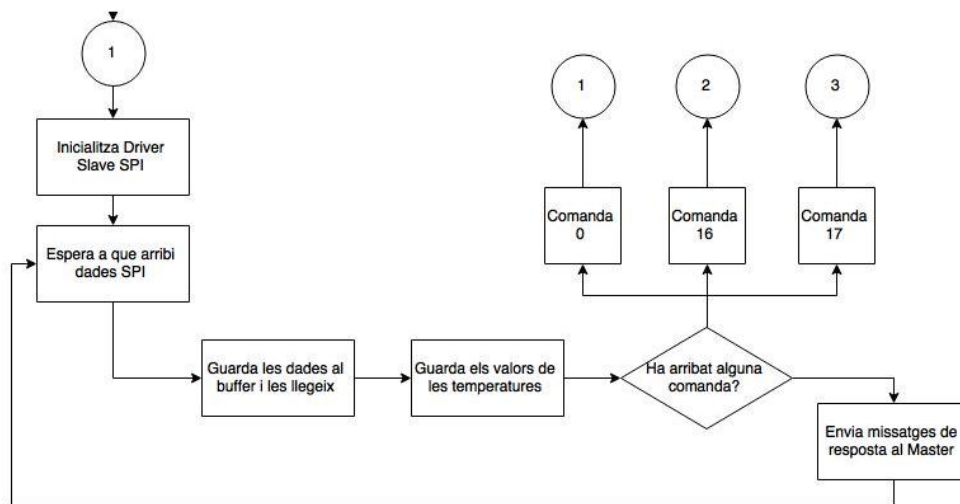


Figura 36. Diagrama Flux Recepció de dades i comandes SPI

Les estructures del codi són molt similars, únicament es diferencia amb la definició de quan s'ha de sortir del bucle.

- Abans d'iniciar el bucle s'inicia el bus de comunicació SPI, definint la velocitat en bauds.
- A continuació es defineix el bucle. En el cas de la lectura de comandes SPI, sortirà d'aquest quan es rebí el valor de la comanda 16, activant el mode Servidor WEB.
- En el cas del lector de temperatures, el bucle finalitzarà quan es rebí el valor 0. Això es degut a que l'execució del thread de captura de temperatures s'executa al final del codi del Servidor WEB, on al finalitzar el bucle, també finalitzarà l'execució de l'aplicació tornant al selector de comandes del firmware SmartCitizenRTX4100.

- Dins del bucle es defineix una condició d'espera fins que es rebí el primer byte al bus SPI
- Un cop el canal comença a rebre els bytes d'informació, aquests s'introdueixen dins del buffer RXspi[] i es llegeixen
- A continuació es passa als valors a les variables que seran utilitzades per mostrar a la pagina web i a la variable que s'utilitza per control del bucle.
- Finalment s'envia els mateixos valors a l'Arduino per indicar que s'ha realitzat correctament la recepció i es buida el buffer.

5 FASE DE PROVES

A continuació es descriu els diversos passos i proves realitzades per comprovar si el sistema funciona correctament, juntament amb els problemes trobats durant tot el procés.

5.1 PROVES

Per la realització del projecte ha sigut necessari la realització de multitud de proves per integrar les noves parts del codi que s'implementaven. Cadascuna de aquestes parts era estudiada prèviament., ja que, com s'ha dit, el procés d'aprenentatge intervenia també en el procés d'integració.

En les proves finals i cada cop que s' integrava una nova funcionalitat es procedia de la següent forma:

- Muntatge i alimentació del Kit de desenvolupament
- Iniciació de l'aplicació ColaController
- Compilació de la colapp mitjançant el procediment descrit en l'apartat de "preparació de l'entorn de desenvolupament".
- Carrega de la colapp al mòdul amb l'aplicacióColaController
- En cas d'actualitzar del firmware Arduino, es tanca l'aplicacióColaController i s'inicia l'aplicació Arduino IDE.
- Compilació i carrega del firmware Arduino a la placa
- S'activa el mode Servidor WEB prenen el boto S1. El LED verd començarà a parpellejar indicant l'activació del mode.
- Al monitor sèrie de l'Arduino IDE es veurà el correcte 'enviament de temperatures i la comanda d'activació. És obligatori que l'aplicacióColaController no estigui activa.
- Connexió amb el mòdul RTX4100 mitjançant un dispositiu amb connexióWifi (Ordinador portàtil o mòbil). Es busca el SSID corresponent amb la placa i es realitza la connexió, en aquest cas no serà necessari la introducció de cap clau. En cas de no activar el mode Servidor WEB, no es veuria el SSID
- S'obre el navegador web s'introdueix la IP de la pagina web En aquest cas la direcció es **192.168.1.88**.

- Es comprova que la pagina web mostra correctament les temperatures i els paràmetres del mòdul.

5.2 PROBLEMES I SOLUCIONS

Durant tot el procés de desenvolupament van sorgir problemes, sigui per motius de falta de coneixement o per les limitacions del propi sistema, on va ser necessari trobar solucions.

- No es pot tenir obertes les aplicacions d'Arduino IDE i el software de desenvolupament de RTX a la vegada. En cas de fer-ho, l'aplicació que predomina és la primera en ser executada, per tant, en el cas de carregar una colapp amb el ColaController, no es podrà carregar firmware o veure el Monitor Sèrie amb Arduino IDE.
- En cas de que el firmware RTX es quedi bloquejat per l'execució errònia d'una colapp, serà impossible carregar noves aplicacions. Per tant, la solució es carregar de nou el firmware del mòdul amb el procediment descrit en la "preparació de l'entorn de desenvolupament".
- La limitació del mòdul RTX (32 bytes) no permet la creació d'elaborades pàgines web. Aquesta situació es va poder veure a executar la web i veure que part del codi programat no es veia o mostrava codi estrany. La solució va ser optimitzar el codi el màxim possible, reduint el codi de presentació (CSS) al mínim i dividir el contingut en varies pàgines webs. Es va descartar directament la utilització de javascript i AJAX per les limitacions observades.
- Al llarg de tot el procés de programació han sorgit infinitat de comportaments inesperats del sistema. Després d'analitzar el codi, es va observar errors de declaracions de funcions, mala utilització de variables i declaracions de protothreads. Gran part dels problemes s'han solucionat gràcies als punts de debug que s'han introduït, però hi ha parts d'aquest que per falta de temps no ha sigut possible resoldre
- Per algun motiu que es desconeix, el canal SPI deixa de llegir dades en activar el mode SOFT AP i al tancar el servidor web. Això provoca que no es pogués enviar correctament els valors dels sensors i les comandes per tancar els modes.

Després de moltes proves, no s'ha solucionat el problema, però sembla que es un problema amb la missatgeria de ColaTask. Per tant, no s'ha pogut implementar l'enviament de comandes SPI. Tot el codi desenvolupat està comentat, on en futurs treballs es pretén resoldre el problema.

- S'ha tingut que modificar des de quin protothread inicia l'aplicació, desactivant mode de comandes SPI, i a continuació indicant que iniciï directament des de el thread del Web server, on també inicia el Soft AP i la lectura de temperatures un cop s'inicia el dispositiu

Segons els problemes descrits, s'ha tingut que modificar la estructura de l'aplicació final. En l'annex es pot veure el diagrama de flux del sistema final. En futurs treballs s'espera seguir la estructura dissenyada al principi del projecte.

6 CONCLUSIO

Durant el procés de desenvolupament del projecte, s'ha realitzat un anàlisi complet de totes les tecnologies necessàries per dissenyar les funcionalitats del sistema i el compliment dels objectius, destacant la anàlisi i estudi de les aplicacions exclusives del mòdul RTX, anomenades Colapps, on va ser necessari també aprendre a programar amb protothreads i conèixer el funcionament de la comunicació SPI

Un dels objectius finalitzats amb èxit, ha sigut la implementació del mode Soft AP ,el qual permet la connexió directa al mòdul des de qualsevol dispositiu amb connexió sense fils, millorant les funcions de connectivitat proposades al principi del projecte.

La connexió amb el mòdul permet obrir, des de qualsevol navegador, la pagina web dissenyada per veure els paràmetres del dispositiu y els valors de les temperatures capturades per la placa de desenvolupament.

Referent al servidor web, s'ha parlat sobre els dos mètodes per desenvolupar-lo, on finalment s'ha decidit dissenyar el sistema des de zero, sense les limitacions que te el mètode per EASYWEB. La web ha sigut implementada respectant les limitacions del mòdul RTX i s'ha programat el codi de forma optimitzada, on finalment, no ha sigut possible desenvolupar la pagina per modificar els paràmetres del mòdul.

Finalment, la comunicació SPI entre el mòdul RTX i la placa Arduino s'ha pogut implementar correctament, amb la modificació del firmware d'Arduino, Implementant les funcions necessàries per l'enviament de temperatures i comandes, i el RTX per a rebre-les. Per els motius descrits en el punt anterior, no ha sigut possible implementar el sistema de comandes d'activació de modes, i per tant, s'ha deixat per futurs treballs millorar aquesta funció. Actualment el codi envia correctament les temperatures on es poden veure a la pagina web.

Personalment, aquest projecte ha suposat tot un repte acadèmic per les dificultats trobades, les quals han estat la majoria solucionades per mitja de la recerca i l'estudi, sent això, un èxit personal en l'avanç de la formació acadèmia i professional. El desenvolupament, des de les fases inicials fins la fase final, ha implicat el coneixement de noves tecnologies i sistemes de programació i ha implicat posar en practica les diferents competències que componen el conjunt d'assignatures que s'han estudiat al Grau de

tecnologies de telecomunicació, on l'objectiu final es demostrar els coneixements adquirits.

6.1 FUTURS TREBALLS

Per falta de temps i altres problemes sorgits descrits al capítol anterior i durant el transcurs del projecte, no ha sigut possible desenvolupar totes les característiques sol·licitades en el pla del projecte i per tant, part dels objectius no han sigut possibles de complir. Els futurs treballs que es proposen realitzar son majoritàriament solucionar els problemes esdevinguts:

- Optimitzar i solucionar els problemes de programació descrits en punts anteriors:
 - Implementar el sistema de formularis per modificar paràmetres del mòdul RTX
 - Solucionar els problemes amb l'enviament de comandes SPI.
 - Optimitzar el codi.

Per un altre banda, es proposa els següents futurs treballs per implementar millores sobre el sistema actual

- Implementar un sistema més complet per l'activació dels modes. En aquest cas es podria implementar una aplicació per dispositiu mòbil que permeti seleccionar els modes disponibles.
- Estudiar la forma de poder implementar codi web mes elaborat. Per solucionar aquesta limitació es pot utilitzar el sistema que s'ha vist en els projectes d'exemple que hi dins dels software AmelieSDK (Exosite, Nabto, 2lementary), que implementen la lectura dels paràmetres mitjançant serveis en la núvol (Cloud Service", on la presentació de la web esta ubicada en un servidor extern.
- Un altra solució que es podria estudiar podria ser utilitzar el lector de targetes SD per implementar el servidor web dins d'aquesta.

7 GLOSSARI

SCK: SmartCitizen Kit

SCDK: SmartCitizen Development Kit

WIFI: Wireless Fidelity

MAC: media access control;

SSID: Service Set Identifie

AP: Access Point.

SOFTAP: Soft Acces Point

TCP: Transmission Control Protocol.

IP: Internet Protocol.

DNS: Domain Name System

PortCOM: Port serie

MCU: microcontroller unit

GPIO: General Purpose Input/Output.

EEPROM: Electrically Erasable Programmable Read-Only Memory

SRAM: Static Random Access Memory

I/O: Input/Ouput

DAC: Digital-to-Analog Conversión

ADC: Analog-to-Digital Conversión

USB: Universal Serial Bus

JTAG: Joint Test Action Group

DMA: Digital media adapater

UART: Universal Asynchronous Receiver-Transmitter

API: Application Programming Interface.

SDK: Software development kit

IDE: Integrated Development Environment.

Colapp: Co-Located Application.

URL: uniform resource locator

HTTP: HyperText Transfer Protocol

HTML: HyperText Markup Language

CSS: cascading style sheets

I2C: Inter-Integrated Circuit.

SPI: Serial Peripheral Interface.

MISO: Master Input Slave Output (SPI signal).

MOSI: Master Output Slave Input (SPI signal).CLK

SS: Selected Slave

CPOL: Clock Polarity

CPHA: Clock Phase

RISC: Reduced Instruction Set Computing.

ROS: RTX Operating System.

NVS: Non-Volatile Storage

RSX: PC tool for API mail trace via UART.

8 BIBLIOGRAFIA

[2]M.Colom. “Analysis, Improvement, and Development of New Firmware for the Smart Citizen Kit Ambient Board”
<http://openaccess.uoc.edu/webapps/o2/bitstream/10609/40042/6/mcolombTFG0115memoria.pdf>,

[1]Smart Citizen , “The Smart Citizen platform”, <http://www.smartcitizen.me/>,

[3] Arduino Team, “Arduino IDE”, <http://arduino.cc/en/Main/Software/>,

[4] Atmel Corporation, “The ATmega32U4 microcontroller unit”, <http://www.atmel.com/devices/atmega32u4.aspx/>,

[5] F. Leens, “An introduction to I2C and SPI protocols”, Instrumentation Measurement Magazine, IEEE, vol. 12, no. 1, pp. 8–13,

[6] Freescale Semiconductor, “Official SPI block guide v03.06”, <http://www.ee.nmt.edu/~teare/ee308/datasheets/S12SPIV3.pdf>

[7] Qualcomm, “AR4100 single-stream 802.11n SIP for the Internet of Everything”, <http://www.qca.qualcomm.com/wp-content/uploads/2013/11/AR4100.pdf>, Abril 2015.

[8] Silicon Labs, “EFM32G200 datasheet”, <http://www.silabs.com/Support%20Documents/TechnicalDocs/EFM32G200.pdf>,

[9] Adam Dunkels, Oliver Schmidt, Thiemo Voigt, and Muneeb Ali, “Protothreads: simplifying event-driven programming of memoryconstrained embedded systems”, in Proceedings of the 4th international conference on Embedded networked sensor systems.

[10]Code, “HTTP: The Protocol Every Web Developer Must Know - Part 1”
<http://code.tutsplus.com/tutorials/http-the-protocol-every-web-developer-must-know-part-1--net-31177>

[11] RTX, “RTX41xx Wi-Fi modules, user guide UG3 application development”,

[12] RTX, “RTX41xx Wi-Fi modules, user guide UG4 Application Debugging”,

[13] RTX, “RTX41xx Wi-Fi modules, user guide UG2 Tools Installation”,

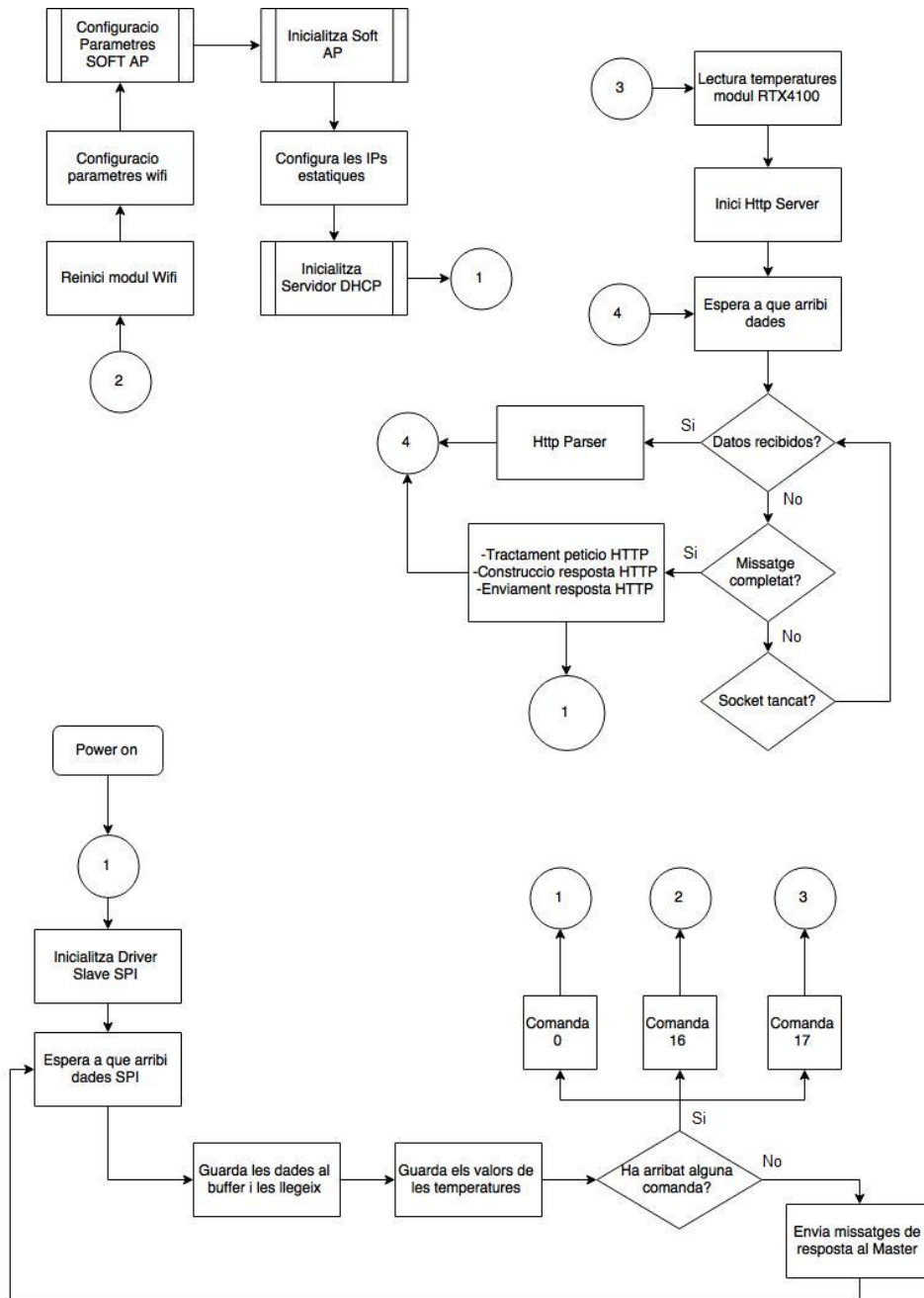
[14]RTX, RTX41xx Wi-Fi modules, Application Note AN8 Soft Acces point Mode, Maig 2015

[15]RTX , “Amelie Api Documentation”, Maig 2015

[16]RTX, RTX41 Wi-Fi module , “RTX4100 Datasheet DS1”, Maig 2015

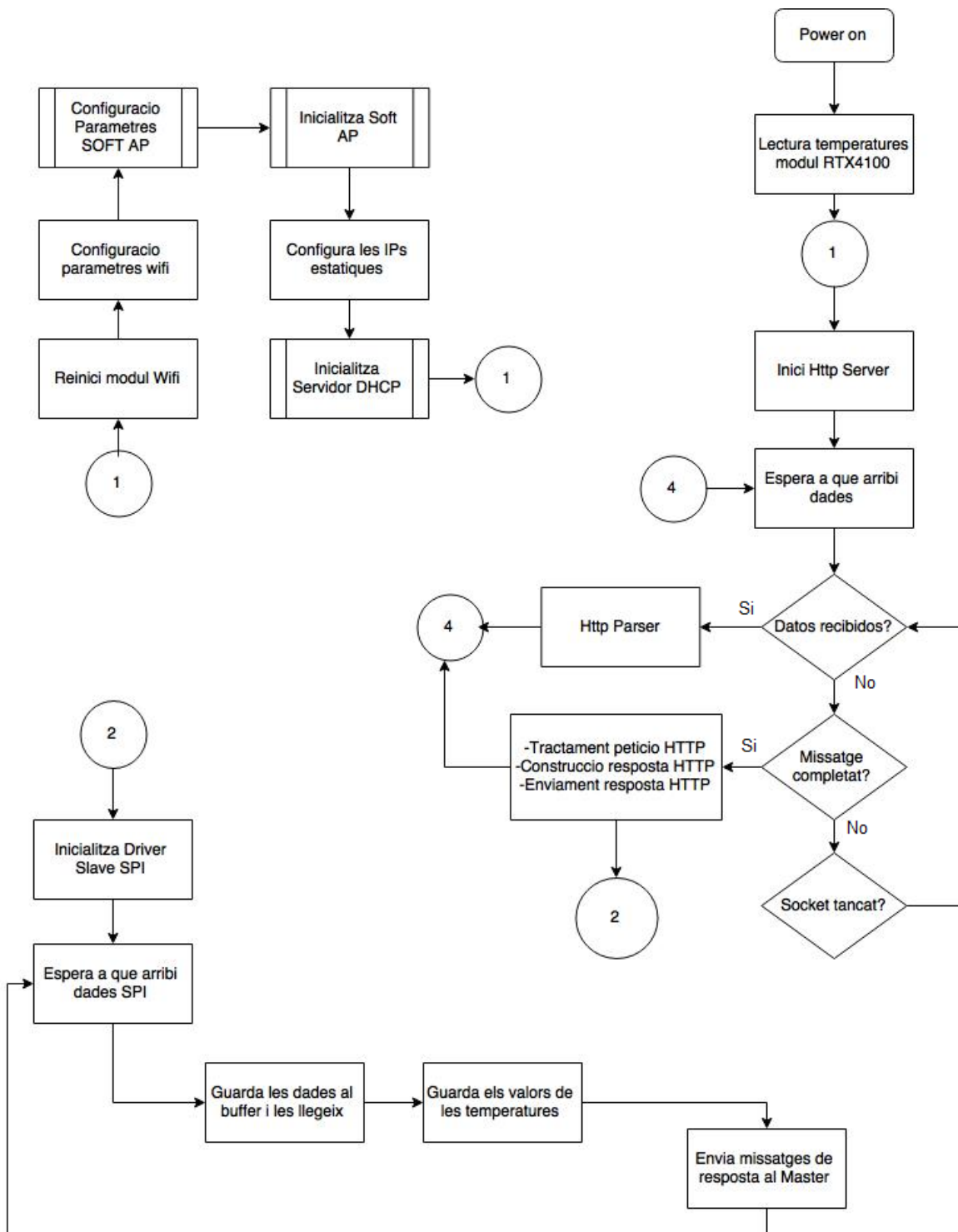
9.1 DIAGRAMA DE FLUX COLAPP FIRMWARE SMARTCITZENRTX4100

Proposta de diagrama del flux del sistema.



9.2 DIAGRAMA DE FLUX COLAPP FINAL

Diagrama del flux de l'aplicació final dissenyada, modificada pels problemes sorgits.



9.3 CODI SCDK.CPP -SMARTCITIZEN_SDK.INO-

```
#include "Constants.h"
#include "SCDVK.h"
#include <Wire.h>
#include <SPI.h>
#include <SD.h>

#define debug true

int firmware_mode = 0;
int operation_mode = 0;
unsigned long t = 0;
byte seq_byte = 0;
byte command = 0;

unsigned long timeisr1 = 0;

void ISR1()
{
  if ((millis()-timeisr1)>1000)
  {
    timeisr1 = millis();
    if (operation_mode<1)operation_mode++;
    else operation_mode=0;
  }
}

unsigned long timeisr2 = 0;

void ISR2()
{
  if ((millis()-timeisr2)>1000)
  {
    timeisr2 = millis();
    if ((operation_mode>1)&&(operation_mode<3)) operation_mode++;
    else operation_mode=2;
  }
}

void SCDVK::begin() {
  pinMode(RESET, OUTPUT);
  digitalWrite(RESET, LOW); //RESET ON
  delay(100);
  digitalWrite(RESET, HIGH); //RESET OFF
  pinMode(MUX, OUTPUT);
  digitalWrite(MUX, LOW); //NORMAL MODE
```

```

pinMode(PROG_MODE, OUTPUT);
digitalWrite(PROG_MODE, LOW); //PROG_MODE OFF
pinMode(GREEN, OUTPUT);
digitalWrite(GREEN, HIGH); //GREEN ON
pinMode(BLUE, OUTPUT);
digitalWrite(BLUE, LOW); //BLUE OFF
pinMode(MMC_CS, OUTPUT);
pinMode(52, OUTPUT);
digitalWrite(MMC_CS, HIGH); //SPI MMC NO SELECT
pinMode(RTX_CS, OUTPUT);
digitalWrite(RTX_CS, HIGH); //SPI RTX NO SELECT
pinMode(POWER, OUTPUT);
digitalWrite(POWER, HIGH); //ENABLE POWER RTX
pinMode(SELECT_MODE_1, INPUT);
pinMode(SELECT_MODE_2, INPUT);
SPI.begin();
Serial.begin(9600);
Serial1.begin(9600);
Wire.begin();
attachInterrupt(SELECT_MODE_1, ISR1, FALLING);
attachInterrupt(SELECT_MODE_2, ISR2, FALLING);
}

```

```

void SCDVK::disable_all()
{
  Serial1.end();
  digitalWrite(MUX, HIGH); //PROGRAM MODE
  digitalWrite(SCK, LOW); //BLUE OFF
  digitalWrite(MISO, LOW); //BLUE OFF
  digitalWrite(MOSI, LOW); //BLUE OFF
}

```

```

void SCDVK::enable_all() //Provisional function
{
  Serial.begin(115200);
  Serial1.begin(115200);
}

```

```

void SCDVK::bootloader_flash()
{
  disable_all();
  digitalWrite(RESET, LOW); //RESET OFF
  digitalWrite(GREEN, HIGH); //GREEN LED ON
  digitalWrite(BLUE, HIGH); //BLUE LED ON
  digitalWrite(POWER, LOW); //DISABLE POWER RTX
  delay(100);
  digitalWrite(PROG_MODE, HIGH);
}

```

```

digitalWrite(RESET, HIGH); //RESET OFF
delay(100);
digitalWrite(POWER, HIGH); //ENABLE POWER RTX
enable_all();
delay(4000);
digitalWrite(PROG_MODE, LOW);
delay(100);
digitalWrite(GREEN, LOW); //GREEN LED ON
while (operation_mode==3)
{
    blink_led_blue(500);
    echo();
}
}

void SCDVK::firmware_flash()
{
    digitalWrite(RESET, LOW); //RESET
    delay(100);
    digitalWrite(RESET, HIGH); //RESET OFF
    digitalWrite(MUX, HIGH); //PROGRAM MODE
    Serial.begin(115200);
    Serial1.begin(115200);
    digitalWrite(GREEN, LOW); //GREEN LED OFF
    digitalWrite(BLUE, HIGH); //BLUE LED ON
    delay(500);
    while (operation_mode==2)
    {
        echo();
    }
}

void SCDVK::terminal_mode()
{
    digitalWrite(MUX, LOW); //NORMAL MODE
    Serial.begin(9600);
    Serial1.begin(9600);
    digitalWrite(GREEN, HIGH); //GREEN LED ON
    digitalWrite(BLUE, LOW); //BLUE LED OFF
    delay(500);
    while (operation_mode==1)
    {
        command = 16;
        main();
        blink_led_green(500);
        echo();
    }
}

```

```

}

void SCDVK::normal_mode()
{
    SPI.endTransaction();
    digitalWrite(RTX_CS, LOW);
    Serial.begin(9600);
    Serial1.begin(9600);
    digitalWrite(GREEN, HIGH); //GREEN LED ON
    digitalWrite(BLUE, LOW); //GREEN LED ON
    while (operation_mode==0)
    {
        command = 0;
        main();
    }
}

boolean ledStateblue = LOW;
unsigned long previousMillisblue = 0;

void SCDVK::blink_led_blue(unsigned long interval)
{
    unsigned long currentMillisblue = millis();

    if(currentMillisblue - previousMillisblue >= interval) {
        // save the last time you blinked the LED
        previousMillisblue = currentMillisblue;

        // if the LED is off turn it on and vice-versa:
        if (ledStateblue == LOW)
            ledStateblue = HIGH;
        else
            ledStateblue = LOW;

        // set the LED with the ledState of the variable:
        digitalWrite(BLUE, ledStateblue);
    }
}

boolean ledStategreen = LOW;
unsigned long previousMillisgreen = 0;

void SCDVK::blink_led_green(unsigned long interval)
{
    unsigned long currentMillisgreen = millis();

    if(currentMillisgreen - previousMillisgreen >= interval) {
        // save the last time you blinked the LED

```

```

previousMillisgreen = currentMillisgreen;

// if the LED is off turn it on and vice-versa:
if (ledStategreen == LOW)
  ledStategreen = HIGH;
else
  ledStategreen = LOW;

// set the LED with the ledState of the variable:
digitalWrite(GREEN, ledStategreen);
}
}

uint16_t SCDVK::readSHT21(uint8_t type){
  uint16_t DATA = 0;
  Wire.beginTransmission(SHT21_ADDRESS);
  Wire.write(type);
  Wire.endTransmission();
  Wire.requestFrom(SHT21_ADDRESS,2);
  unsigned long time = millis();
  while (!Wire.available()) if ((millis() - time)>500) return 0x00;
  DATA = Wire.read()<<8;
  while (!Wire.available());
  DATA = (DATA|Wire.read());
  DATA &= ~0x0003;
  return DATA;
}

uint32_t lastHumidity;
uint32_t lastTemperature;

float SCDVK::getTemperature()
{
  return (-46.85 + ((175.72 * (float)readSHT21(0xE3)) / 65536.0) );
}

float SCDVK::getHumidity()
{
  return (-6.0 + ((125.0 * (float)readSHT21(0xE5)) / 65536.0));
}

// set up variables using the SD utility library functions:
Sd2Card card;

void SCDVK::check_MMC()
{
  // we'll use the initialization code from the utility libraries
  // since we're just testing if the card is working!

```

```

    if (!card.init(SPI_QUARTER_SPEED, MMC_CS)) {
        Serial.println("initialization failed. Things to check:");
        Serial.println("* is a card is inserted?");
        Serial.println("* Is your wiring correct?");
        Serial.println("* did you change the chipSelect pin to match your shield or module?");
        return;
    } else {
        Serial.println("Wiring is correct and a card is present.");
    }
}

void SCDVK::echo()
{
    if (Serial.available())
        Serial1.write(Serial.read());
    if (Serial1.available())
        Serial.write(Serial1.read());
}

void SCDVK::execute()
{
    switch (operation_mode) {
        case 0:
            normal_mode();
            break;
        case 1:
            terminal_mode();
            break;
        case 2:
            firmware_flash();
            break;
        case 3:
            bootloader_flash();
            break;
    }
}

void SCDVK::main()
{
    Serial.print("Temperature: ");
    Serial.print(getTemperature());
    Serial.print(" C, Humidity: ");
    Serial.print(getHumidity());
    Serial.println(" %");
    delay(1000);
    check_MMC();

    if (millis() > (t + 3000)) {

```



```
//Test SPI (Arduino=Master)

digitalWrite(RTX_CS, HIGH); //Hay que ponerlo ya que SPI.begin no lo baja
SPI.beginTransaction(SPISettings(SPI_CLOCK_DIV4, LSBFIRST, SPI_MODE0));

Serial.println("Datos a enviar:");
Serial.print("->P1: "); Serial.println(getTemperature());
Serial.print("->P2: "); Serial.println(getHumidity());
Serial.print("->P3: "); Serial.println(command);

byte p1 = SPI.transfer(getTemperature());
byte p2 = SPI.transfer(getHumidity());
byte p3 = SPI.transfer(command);

Serial.println("Datos recibidos:");
Serial.print("<-P1: "); Serial.println(p1);
Serial.print("<-P2: "); Serial.println(p2);
Serial.print("<-P3: "); Serial.println(p3);
SPI.endTransaction();

digitalWrite(RTX_CS, LOW);
t = millis();
}
}
```

9.4 CODI COLAPP – SMARTCITIZIENRTX4100-

Únicament es mostra les línies que s'han modificat del firmware original, ja que en total són més de 2000 línies de codi. La resta del codi es pot trobar dins del directori `\SmartCitizenRTX4100`: a l'arxiu comprimit del projecte o al github: <https://github.com/acasvi/SmartCitizenRTX4100>:

Linies 68-138 (Definició paràmetres)

```
//*****TEST ALEJANDRO CASTRO*****//

#include <RtxEai/RtxEai.h>
#include <Drivers/DrvSpiSlave.h>
#include <BuildInfo/BuildInfo.h>
// For debug (EAI)
#define printf(...) RtxEaiPrintf(Colalf->ColaTaskId,__VA_ARGS__)
// Data for configure softAP
#define SOFT_AP_STATIC_IP    "192.168.1.88"
#define SOFT_AP_SUBNET_MASK  "255.255.255.0"
#define SOFT_AP_GATEWAY     "192.168.1.1"
#define SOFT_AP_DHCP        TRUE
#define SOFT_AP_ESSID       "SCK_AP_" //partial SSID
#define SOFT_AP_SECURITY_TYPE AWST_NONE
#define SOFT_AP_COUNTRY_CODE "ES"
#define SOFT_AP_CHANNEL     2437 //Channel 6 center frequency
#define SOFT_AP_INACT       10 // minutes
#define SOFT_AP_BEACON      100 // ms
#define GATEWAY             "192.168.1.1"
#define DATE_STR_LEN 30
#define HTTP_PORT 80
#define RX_IDLE_TIMEOUT (30*RS_T1SEC)
#define SOFT_AP_DHCP_POOL_START 0xC0A801C8 //"192.168.1.200"
#define SOFT_AP_DHCP_POOL_END  0xC0A801FF //"192.168.1.255"
#define SOFT_AP_LEASE_TIME     86400

/*****
*                               Macro definitions
*****/

// Max WiFi transmission power
#define MAX_TX_POWER 18

// Macros to print with the LUART
```

```

#define PRINT(x) UartPrint((rsuint8*)x)
#define PRINTLN(x) UartPrintLn((rsuint8*)x)

// Buffer sizes
#define TMP_STR_LENGTH 100
#define CMD_STR_LENGTH TMP_STR_LENGTH
#define TX_BUFFER_LENGTH 500

// Maximum number of arguments for terminal commands
#define MAX_ARGV 3

// Timeout in seconds for DNS resolutions
#define APP_DNS_RESOLVE_RSP_TIMEOUT (10*RS_T1SEC)

/*****
*       Enumerations/Type definitions/Structs
*****/
// Application data stored at the NVS
typedef struct {
    ApInfoType ap_info;
    rsuint8 use_dhcp;
    struct pt ChildPt;
    PtEntryType* PtEntryPtr;
    rschar DateStr[DATE_STR_LEN];
    rsint16 Temperature;
    rsint16 Humidity;
    rsint32 Pressure;
    rsint16 Lux;
    rsbool MultiSensor;
    rsbool HttpServerStarted;
    rsuint8 SPI1 [5];
    rschar HWVERSION[25];
    rschar SWVERSION[25];
    rschar platform [30];
    rsuint32 HttpInstance;
    ApiSocketAddrType static_address, static_subnet, static_gateway;
} app_dataType;

```

Linies 979-1514(Implementacio Codi Soft AP, WebServer i comunicació SPI)

```

//////////*****ALEJANDRO CASTRO IMPLEMENTATION*****//////////

```

```

/*
 * @brief Sets the softAP mode
 * @param Pt : current protothread pointer
 * @param Mail : protothread mail
 */
static PT_THREAD(SoftAPMode(struct pt *Pt, const RosMailType *Mail)) {
    static struct pt childPt;
    PT_BEGIN(Pt);
    // Reset Wifi
    PT_SPAWN(Pt, &childPt, PtAppWifiReset(&childPt, Mail));

    // Set power save parameters
    AppWifiSetPowerSaveProfile(POWER_SAVE_HIGH_IDLE); // 200ms idle interval
    AppWifiSetListenInterval(100); // 100ms

    const ApiWifiMacAddrType *pMacAddr = AppWifiGetMacAddr();

    //Set parameteres SoftAP
    //SSID = MASK + MAC
    rsuint8 ssid[32];
    snprintf((char*)ssid, sizeof(ssid), "%s%02X%02X%02X",
        SOFT_AP_ESSID, (*pMacAddr)[3], (*pMacAddr)[4], (*pMacAddr)[5]);
    AppWifiApSetSoftApInfo((rsuint8*)ssid, SOFT_AP_SECURITY_TYPE,
        FALSE, 0, NULL, SOFT_AP_CHANNEL, SOFT_AP_INACT,
        (rsuint8*)SOFT_AP_COUNTRY_CODE, SOFT_AP_BEACON);

    //Start SoftAP
    PT_SPAWN(Pt, &childPt, PtAppWifiStartSoftAp(&childPt, Mail));
    //static IP
    ApiIpV4AddressType address, subnetMask;
    inet_aton(SOFT_AP_STATIC_IP, &address);
    inet_aton(SOFT_AP_SUBNET_MASK, &subnetMask);
    AppWifiIpV4Config(true, address, subnetMask, address, address);
    //start SOFT Dhcp
    SendApiWifiApSetDhcpPoolReq(COLA_TASK, SOFT_AP_DHCP_POOL_START,
    SOFT_AP_DHCP_POOL_END, SOFT_AP_LEASE_TIME);
    //wait a client
    PT_WAIT_UNTIL(Pt, IS_RECEIVED(API_WIFI_CONNECT_IND));

    PT_END(Pt);
}

```

```

// Definition Header HTTP
static rsuint32 AddHeader(rsuint8 *BufferPtr, rsuint32 BufferLength, rsuint8* DataPtr, rsuint32
Instance)
{
    rsuint8 *p = BufferPtr;
    rsuint32 l = 0;

    l += HttpAddHeader(p+l, BufferLength-l, "Content-Type", "text/html; charset=utf-8");
    l += HttpAddHeader(p+l, BufferLength-l, "Server", "RTX41xx demo WEB server");
    l += HttpAddHeader(p+l, BufferLength-l, "Date", app_data.DateStr);

    return l;
}
// Definition Main Page. Display temperatures of SHT21 and RTX4100
static rsuint32 GenerateMainPage(rsuint8 *BufferPtr, rsuint32 BufferLength, rsuint32 Offset,
rsuint8 *DataPtr, rsuint32 Instance)
{
    rsuint32 n = 0;
    rsuint16 l = BufferLength;
    rschar *p = (rschar*)BufferPtr;
    const char* const start =
"<!DOCTYPE html>"
"<html>"
"<head>"
    "    <meta charset='UTF-8'>"
    "    <title>RTX WEBSERVER</title>"
    "    <style>"
"body{"
    "font-family: nunito, sans-serif;"
    "background: #00131F;"
"}"
".boton{"
    "font-size: 20px;"
    "background: #00131F;"
    "color:#329AB4;"
    "border: solid 3px #00131F;"
    "font-weight: bolder;"
    "transition: 2s;"
"}"
".boton:hover{"
    "color:#fff;"

```

```

    "border-bottom: solid 3px #fff;"
  }"
  ".selected{"
    "font-size: 20px;"
    "background: #002740;"
    "color:white;"
    "border: solid 3px #002740;"
    "font-weight: bolder;"
    "transition: 2s;"
  }"
  ".content{"
    "background: #002740;"
    "padding: 2em;"
    "text-align: center;"
    "width: 100%;"
    "margin: 0 auto;"
  }"
  ".content p{"
    "margin: 0 auto;"
    "margin-bottom: 1em;"
    "margin-top: 1em;"
    "background-color: #163B52;"
    "width: 50%;"
    "padding: 1em;"
    "font-weight: bolder;"
    "color:#fff;"
    "border: solid 0px white;"
    "border-radius: 10px;"
    "box-shadow: 0 0 8px #414141;"
  }"
  ".content p span{"
    "color:#66d5f0;"
    "font-size: 1.5em;"
  }"
</style>"
</head>"
<body>"
  <div>"
    <input class="selected" type="button" value="Inicio">"
    <input class="boton" type="button" value="Informacion"
onclick="location.href='Info'">"

```

```
"<input class=\"boton\" type=\"button\" value=\"Soft AP Conf\"
onclick=\"location.href='SoftAP'\">"
```

```
"<div class=\"content\">"
  "<p>Temperatura modulo:<br><br><span>%d.%d C</span></p>"
  "<p>Temperatura sensor:<br><br><span>%d C</span></p>"
  "<p>Humedad:<br><br><span>%d %s</span></p>"
"</div> "
"</div>"
"</body>"
"</html>";
```

```
n = sprintf(p, l, start, "%", app_data.Temperature/10,
app_data.Temperature%10, app_data.SPI1[0], app_data.SPI1[1], "%");
return n;
}
```

```
// Definition Info Page. Displays Mac, last version of colapp, Wifi Hardware and Software Version
and platform version.
```

```
static rsuint32 Info(rsuint8 *BufferPtr, rsuint32 BufferLength, rsuint32 Offset, rsuint8 *DataPtr,
rsuint32 Instance)
```

```
{
```

```
rsuint32 n = 0;
rsuint16 l = BufferLength;
rschar *p = (rschar*)BufferPtr;
const ApiWifiMacAddrType *pMacAddr = AppWifiGetMacAddr();
const char* const start =
```

```
"<!DOCTYPE html>"
```

```
"<html>"
```

```
"<head>"
```

```
  "<meta charset='UTF-8'>"
```

```
  "<title>RTX WEBSERVER</title>"
```

```
  "<style>"
```

```
  "body{"
```

```
    "font-family: nunito, sans-serif;"
```

```
    "background: #00131F;"
```

```
  "}"
```

```
  ".boton{"
```

```
    "font-size: 20px;"
```

```
    "background: #00131F;"
```

```
    "color:#329AB4;"
```

```
    "border: solid 3px #00131F;"
```

```

        "font-weight: bolder;"
        "transition: 2s;"
    }"
    ".boton:hover{"
        "color:#fff;"
        "border-bottom: solid 3px #fff;"
    }"
    ".selected{"
        "font-size: 20px;"
        "background: #002740;"
        "color:white;"
        "border: solid 3px #002740;"
        "font-weight: bolder;"
        "transition: 2s;"
    }"
    ".content{"
        "background: #002740;"
        "padding: 1em;"
        "text-align: center;"
        "width: 100%;"
        "margin: 0 auto;"
        "font-size: 30px;"
        "font-weight: bolder;"
        "color:#fff;"
    }"
        ".content p sub{"
            "color:#fff;"
            "font-size: 15px;"
        }"
    "</style>"
"</head>"
"<body>"
"<div>"
    "<input class=\"boton\" type=\"button\" value=\"Inicio\" onclick=\"location.href='/\">"
    "<input class=\"selected\" type=\"button\" value=\"Informacion\" >"
    "<div class=\"content\"> "
        "<p>Informacion del sistema<br>"
        "<sub>MAC: <font color=\"#66d5f0\">%02X-%02X-%02X-%02X-%02X-
%02X</font></sub>"
        "<br><sub>WebServer Version: <font color=\"#66d5f0\">%s 20%02X-
%02X-%02X %02X:%02X</font></sub><br> "
        "<sub>Wifi Hw Version: <font color=\"#66d5f0\">%s</font></sub>"

```



```

        "<br><sub>Wifi SW Version: <font color=\"#66d5f0\">%s
</font></sub><br> "
        "<sub>Platform Version: <font color=\"#66d5f0\">%s</font></sub>"
    "</div> "
    "</div>"
    "</body>"
    "</html>";

```

```

    n = snprintf(p, l, start,(*pMacAddr)[0], (*pMacAddr)[1], (*pMacAddr)[2], (*pMacAddr)[3],
    (*pMacAddr)[4], (*pMacAddr)[5],(const char*)ReleaseLabel,LinkDate[0], LinkDate[1], LinkDate[2],
    LinkDate[3], LinkDate[4],app_data.HWVERSION,app_data.SWVERSION,app_data.platform);

```

```

    return n;
}

```

// Definition of Configure Page. Configure SSID, Security Type and password. In SSID Input display the actual SSID.

```

static rsuint32 SOFTAPPage(rsuint8 *BufferPtr, rsuint32 BufferLength, rsuint32 Offset, rsuint8
*DataPtr, rsuint32 Instance)
{

```

```

    rsuint32 n = 0;
    rsuint16 l = BufferLength;
    rschar *p = (rschar*)BufferPtr;

```

```

    const char* const start =
    "<!DOCTYPE html>"
    "<html>"
    "<head>"
        "<meta charset='UTF-8'>"
        "<title>RTX WEBSERVER</title>"
        "<style>"
    "body{"
        "font-family: nunito, sans-serif;"
        "background: #00131F;"
    "}"
    ".boton{"
        "font-size: 20px;"
        "background: #00131F;"
        "color:#329AB4;"
        "border: solid 3px #00131F;"
        "font-weight: bolder;"

```

```

    "transition: 2s;"
  }"
  ".boton:hover{"
    "color:#fff;"
    "border-bottom: solid 3px #fff;"
  }"
  ".selected{"
    "font-size: 20px;"
    "background: #002740;"
    "color:white;"
    "border: solid 3px #002740;"
    "font-weight: bolder;"
    "transition: 2s;"
  }"
  ".content{"
    "background: #002740;"
    "padding: 1em;"
    "text-align: left;"
    "width: 100%;"
    "margin: 0 auto;"
    "font-size: 30px;"
    "font-weight: bolder;"
    "color:#fff;"
  }"
  ".content span{"
    "color:#66d5f0;"
    "font-size: 15px;"
  }"

"</style>"
"</head>"
"<body>"
  "<div>"
    "<input class=\"boton\" type=\"button\" value=\"Inicio\" onclick=\"location.href='/\">"
    "<input class=\"selected\" type=\"button\" value=\"Soft AP Conf\" >"
    "<div class=\"content\"> "
      "<p>Configuracion Soft AP<br>"
        "<form action=\"softap\" method=\"get\">"
          "<br><span>SSID</span><input type=\"text\" size= 50 name=\"ssid\"
value=\"%s\"> "
          "<span>Seguridad</span>"
          "<select name=\"Seguridad \">"

```

```

        "<option value=\"None\">None </option>"
        "<option value=\"WPA\">WPA</option>"
    "</select>"
        "<br><span>Password</span><input type=\"text\" size= 50
name=\"ssid\" value=\"\"> "
        "<br><input type=\"submit\" value=\"Save Settings\">"
        "<form method=\"link\" action=\"/\">"
        "</form>"
    "</div> "
"</div>"
"</body>"
"</html>";

n = snprintf(p, l, start,AppWifiApGetSoftApSsid());
return n;
}
//Resource Callback of GenerationMainPage. Generates the Main Page resource
static RsStatusType OnMainPage(AhHttpMethodIdType HttpMethod, rschar *PathPtr, rschar
*QueryPtr, rsuint32 Instance)
{
    if (HttpMethod != AHM_GET)
    {
        return RSS_NOT_SUPPORTED;
    }
}

SendApiHttpServerSendResponseReq(COLA_TASK, Instance, 200, "OK",
    GenerateMainPage(NULL, 0, 0, NULL, 0), // Calc the size of the main page
    NULL, // Static body not used
    AddHeader, // Call back used to add HTTP headers
    GenerateMainPage); // Callback used to generate the main page
return RSS_SUCCESS;
}
//Resource Callback of InfoPage. Generates the Info Page resource
static RsStatusType InfoPage(AhHttpMethodIdType HttpMethod, rschar *PathPtr, rschar
*QueryPtr, rsuint32 Instance)
{
    if (HttpMethod != AHM_GET)
    {
        return RSS_NOT_SUPPORTED;
    }
}

SendApiHttpServerSendResponseReq(COLA_TASK, Instance, 200, "OK",

```

```

        Info(NULL, 0, 0, NULL, 0), // Calc the size of the main page
        NULL,                    // Static body not used
        AddHeader,               // Call back used to add HTTP headers
        Info);                  // Callback used to generate the main page
    return RSS_SUCCESS;
}
//Resource Callback of SoftAPPAGE. Generates the Conf Page resource.
static RsStatusType ConfSoftAP(AhHttpMethodIdType HttpMethod, rschar *PathPtr, rschar
*QueryPtr, rsuint32 Instance)
{
    if (HttpMethod != AHM_GET)
    {
        return RSS_NOT_SUPPORTED;
    }

    SendApiHttpServerSendResponseReq(COLA_TASK, Instance, 200, "OK",
        SOFTAPPage(NULL, 0, 0, NULL, 0), // Calc the size of the main page
        NULL,                    // Static body not used
        AddHeader,               // Call back used to add HTTP headers
        SOFTAPPage);            // Callback used to generate the main page
    return RSS_SUCCESS;
}
/*// I try to implement application form and implement NVS system but I don't have time :(//
static RsStatusType ConfSoftAP(AhHttpMethodIdType HttpMethod, rschar *PathPtr, rschar
*QueryPtr, rsuint32 Instance)
{
    static const char* const okPage =
        "<!DOCTYPE html>"
        "<html>"
        "<body>"
        "<h1>Se ha modificado los parametros correctament. Reinicia el modulo</h1>"
        "<form method=\"link\" action=\"/\">"
        "<input type=\"submit\" value=\"OK\">"
        "</form>"
        "</body>"
        "</html>";
    static const char* const failPage =
        "<!DOCTYPE html>"
        "<html>"
        "<body>"
        "<h1>No has introducido la contraseña!</h1>"
        "<form method=\"link\" action=\"/SOFTAPPage\">"

```

```

        "<input type=\"submit\" value=\"OK\">"
        "</form>"
        "</body>"
        "</html>";
    AhQueryDataType query[3];
    rsuint8 queryCount = HttpSplitQueryString(3, query, QueryPtr);

    if (HttpMethod != AHM_GET)
    {
        return RSS_NOT_SUPPORTED;
    }

    if (queryCount == 3)
    {
        rschar ssid [25];
        rschar SecurityType [25];
        rschar Key [25];

        // Static ip ?
        if (query[0].ValuePtr[0] == '1')
        {
            ok = FALSE;
            if ((query[0].ValuePtr)==NULL)
            {
                if ((query[1].ValuePtr=="NONE")
                {
                    if (query[2].ValuePtr==NULL)
                    {
                        ok = TRUE;
                    }
                }
            }
        }
        else
        {
            SendApiHttpServerSendResponseReq(COLA_TASK, Instance, 200, "OK", strlen(failPage),
            (rsuint8*)failPage, NULL, NULL);
            return RSS_SUCCESS;
        }
    }
    else
    {

```

```

    SendApiHttpServerSendResponseReq(COLA_TASK, Instance, 200, "OK", SOFTAPPage(NULL, 0, 0,
NULL, 0), NULL, NULL, SOFTAPPage);
    return RSS_SUCCESS;
}
return RSS_SUCCESS;
}

}
*/

/**
 * @brief : It controls the SPI communication. This implementation reads commands, but the main
protothread only reads temperatures
    In future works, the implementation of command SPI, will activate SOFT AP and Web Server
modes and read temperatures.
 * @param Pt : current protothread pointer
 * @param Mail : protothread mail
 **/
static PT_THREAD(SPI_temperatures(struct pt *Pt, const RosMailType *Mail)) {
    static struct pt childPt;
    PT_BEGIN(Pt);

    //SPI SLAVE
    //define baud rate
    const rsuint32 baud_rate = 9600;

    // starts SPI slave driver
    PT_SPAWN(Pt, &childPt, PtDrvSpiSlaveInit(&childPt, Mail, baud_rate));
    // Defines a loop while command is equal command 16 or 17
    //if(app_data.SPI1[2]==16 || 17){
    //while (app_data.SPI1[2]==16 || 17) {
        while(1){
            // waits to received a SPI data
            PT_WAIT_UNTIL(Pt, IS_RECEIVED(SPI_SLAVE_RX_DATA));
            RosTimerStart(APP_PACKET_DELAY_TIMER, (3000 * RS_T1MS), &PacketDelayTimer);
            PT_YIELD_UNTIL(Pt, IS_RECEIVED(APP_PACKET_DELAY_TIMEOUT));

            //reads and put data in buffer
            rsuint8 recibidospi[DrvSpiSlaveRxGetSize()];
            DrvSpiSlaveRx(&recibidospi[0], sizeof(recibidospi));

            app_data.SPI1[0] = recibidospi[0];

```

```

    app_data.SPI1[1] = recibidospi[1];
    app_data.SPI1[2] = recibidospi[2];

    // sends the same data to Arduino
    DrvSpiSlaveTxStart(&app_data.SPI1[0], 3);
        // empty the buffer
        DrvSpiSlaveRxFlush();
    }
/*Protothread exit if command is different that 16 or 17
PT_EXIT(Pt);
}

        // Defines a loop while command is different 16 or 17
while (app_data.SPI1[2]!=16 || 17) {

        PT_WAIT_UNTIL(Pt, IS_RECEIVED(SPI_SLAVE_RX_DATA));
    RosTimerStart(APP_PACKET_DELAY_TIMER, (3000 * RS_T1MS), &PacketDelayTimer);
        PT_YIELD_UNTIL(Pt, IS_RECEIVED(APP_PACKET_DELAY_TIMEOUT));

    printf ("Intentando enviar datos...\n");

    rsuint8 recibidospi[DrvSpiSlaveRxGetSize()];
    DrvSpiSlaveRx(&recibidospi[0], sizeof(recibidospi));

    app_data.SPI1[0] = recibidospi[0];
    app_data.SPI1[1] = recibidospi[1];
    app_data.SPI1[2] = recibidospi[2];

    DrvSpiSlaveTxStart(&app_data.SPI1[0], 3);
    printf("ENVIADO\n");
        DrvSpiSlaveRxFlush();
    }
*/
    PT_END(Pt);
}

/**
 * @brief Main implementation protothread. Starts Soft AP, Intern Driver Temperature of RTX,
Reads Wifi HW&SW and platform version,
    Starts HTTP server, add pages and starts the SPI communication
        In future works, when command SPI works, this protothread only starts Http
Server and add pages.
 * @param Pt : current protothread pointer

```

```
* @param Mail : protothread mail
**/
```

```
static PT_THREAD(WebServer(struct pt *Pt, const RosMailType* Mail))
{
    static struct pt SensorUpdatePt;
    static struct pt childPt;
    PT_BEGIN(Pt);
    // Starts Soft Ap MODE
    PT_SPAWN(Pt, &childPt, SoftAPMode(&childPt, Mail));
    // Measure internal temperature inside the EFM32 chip
    PT_SPAWN(Pt, &SensorUpdatePt, PtDrvIntTempMeasure(&SensorUpdatePt, Mail,
    &app_data.Temperature));
    //Reads the Wifi Hardware and Software Version
    SendApiWifiGetVersionReq(COLA_TASK);
    PT_WAIT_UNTIL(Pt, IS_RECEIVED(API_WIFI_GET_VERSION_CFM));
    sprintf(app_data.SWVERSION, "%d.%d.%d.%d.%d",
        (int)(((ApiWifiGetVersionCfmType *)Mail)->SwVersion & 0xF0000000) >> 28,
        (int)(((ApiWifiGetVersionCfmType *)Mail)->SwVersion & 0x0F000000) >> 24,
        (int)(((ApiWifiGetVersionCfmType *)Mail)->SwVersion & 0x00FC0000) >> 18,
        (int)(((ApiWifiGetVersionCfmType *)Mail)->SwVersion & 0x0003FF00) >> 8,
        (int)(((ApiWifiGetVersionCfmType *)Mail)->SwVersion & 0x000000FF));
    sprintf(app_data.HWVERSION, "%d.%d.%d.%d.%d",
        (int)(((ApiWifiGetVersionCfmType *)Mail)->HwVersion & 0xF0000000) >> 28,
        (int)(((ApiWifiGetVersionCfmType *)Mail)->HwVersion & 0x0F000000) >> 24,
        (int)(((ApiWifiGetVersionCfmType *)Mail)->HwVersion & 0x00FC0000) >> 18,
        (int)(((ApiWifiGetVersionCfmType *)Mail)->HwVersion & 0x0003FF00) >> 8,
        (int)(((ApiWifiGetVersionCfmType *)Mail)->HwVersion & 0x000000FF));
    //Reads the Platform Version
    SendApiGetPlatformVersionReq(COLA_TASK);
    PT_WAIT_UNTIL(Pt, IS_RECEIVED(API_GET_PLATFORM_VERSION_CFM));
    sprintf(app_data.platform, "%X.%X.%X.%X (20%02X-%02X-%02X %02X:%02X)",
        ((ApiGetPlatformVersionCfmType *)Mail)->Version >> 8,
        (rsuint8)((ApiGetPlatformVersionCfmType *)Mail)->Version,
        ((ApiGetPlatformVersionCfmType *)Mail)->SubVersion,
        ((ApiGetPlatformVersionCfmType *)Mail)->BuildNumber,
        ((ApiGetPlatformVersionCfmType *)Mail)->LinkDate[0],
        ((ApiGetPlatformVersionCfmType *)Mail)->LinkDate[1],
        ((ApiGetPlatformVersionCfmType *)Mail)->LinkDate[2],
        ((ApiGetPlatformVersionCfmType *)Mail)->LinkDate[3],
        ((ApiGetPlatformVersionCfmType *)Mail)->LinkDate[4]);
    // Start HTTP server
```



```

SendApiHttpServerInitReq(COLA_TASK, 80, NULL);
PT_WAIT_UNTIL(Pt, IS_RECEIVED(API_HTTP_SERVER_INIT_CFM) &&
((ApiHttpServerInitCfmType*)Mail)->Port == 80);
if (((ApiHttpServerInitCfmType*)Mail)->Status == RSS_SUCCESS)
{
    // Server started
    app_data.HttpServerStarted = TRUE;
    app_data.HttpInstance = ((ApiHttpServerInitCfmType*)Mail)->Instance;

    // Add main page
    SendApiHttpServerAddResourceReq(COLA_TASK, app_data.HttpInstance, "/", OnMainPage);
    SendApiHttpServerAddResourceReq(COLA_TASK, app_data.HttpInstance, "/Info",
InfoPage);
    SendApiHttpServerAddResourceReq(COLA_TASK, app_data.HttpInstance, "/SoftAP",
ConfSoftAP);
}
PT_SPAWN(Pt, &childPt, SPI_temperatures(&childPt, Mail));
PT_END(Pt);
}

```

//////////////////////////////////*****END IMPLEMENTATION*****//////////////////////////////////

Linies 1765-1954 Implementacio commandes:

```

//////////////////////////////////***** ALEJANDRO COMMAND IMPLEMENTATION*****//////////////////////////////////
// Whent Initialize SOFT AP or close HTTP server, the SPI communication close too...No sense!!! >o<'
// Actually the application starts directly in Webserver Protothread!! Sorry guys! :(
while (1) {

    // Wait until SPI data is received

    PT_SPAWN(Pt, &childPt, SPI_temperatures(&childPt, Mail));

    switch (app_data.SPI1[2]) {
    case 1: { // get status
        rsuint8 status = Wifi_get_status();
        DrvSpiTxStart(&status, 1);
        PT_WAIT_UNTIL(Pt, IS_RECEIVED(SPI_TX_DONE));
        break;
    }
    case 2: { // DNS resolve
        // Read name to resolve (ex: "www.example.com")

```

```

// First read the size of the name
rsuint8 name_size;
DrvSpiRx(&name_size, sizeof(name_size));
PT_WAIT_UNTIL(Pt, IS_RECEIVED(SPI_RX_DATA));

// Second, read the name
rsuint8 name[100];
DrvSpiRx((rsuint8*)&name, name_size);
PT_WAIT_UNTIL(Pt, IS_RECEIVED(SPI_RX_DATA));
name[name_size] = 0; // put trailing zero

// Resolve
rsuint32 response;
PT_SPAWN(Pt, &childPt, PtWifi_DNS_resolve(&childPt, Mail, name, &response));

// Send response
DrvSpiTxStart((rsuint8*)&response, sizeof(response));
PT_WAIT_UNTIL(Pt, IS_RECEIVED(SPI_TX_DONE));
break;
}
case 3: { // IP config
// First read the size of the config
rsuint8 config_size;
DrvSpiRx(&config_size, sizeof(config_size));
PT_WAIT_UNTIL(Pt, IS_RECEIVED(SPI_RX_DATA));

// Second, read the config
rsuint8 config[100];
if (config_size > 0) {
    DrvSpiRx((rsuint8*)&config, config_size);
    PT_WAIT_UNTIL(Pt, IS_RECEIVED(SPI_RX_DATA));
}

// Do IP config
PT_SPAWN(Pt, &childPt, PtWifi_IP_config(&childPt, Mail,
    config_size > 0 ? config : NULL));
break;
}
case 4: { // TCP start
// Given the IP address of the server (rsuint32), start the connection.
// The upper layer must poll in order to check when the connection

```

```

// has been established.
ApiSocketAddrType addr;
addr.Domain = ASD_AF_INET;

// Read the IP of the TCP server (rsuint32)
DrvSpiRx((rsuint8*)&addr.Ip.V4.Addr, sizeof(addr.Ip.V4.Addr));
PT_WAIT_UNTIL(Pt, IS_RECEIVED(SPI_RX_DATA));

// Read the port of the TCP server
DrvSpiRx((rsuint8*)&addr.Port, sizeof(addr.Port));
PT_WAIT_UNTIL(Pt, IS_RECEIVED(SPI_RX_DATA));

// Start TCP connection
PT_SPAWN(Pt, &childPt, PtWifi_TCP_start(&childPt, Mail, addr));
break;
}
case 5: { // Associate & connect to the WiFi AP
    PT_SPAWN(Pt, &childPt, PtWifi_connect(&childPt, Mail));
    break;
}
case 6: { // WiFi AP deassociate & disconnect
    PT_SPAWN(Pt, &childPt, PtWifi_disconnect(&childPt, Mail));
    break;
}
case 7: { // setup AP
    // Read ap_data size
    rsuint8 ap_data_size;
    DrvSpiRx(&ap_data_size, sizeof(ap_data_size));
    PT_WAIT_UNTIL(Pt, IS_RECEIVED(SPI_RX_DATA));

    // Read ap_data
    rsuint8 ap_data[100];
    if (ap_data_size > 0) {
        DrvSpiRx((rsuint8*)&ap_data, ap_data_size);
        PT_WAIT_UNTIL(Pt, IS_RECEIVED(SPI_RX_DATA));
    }

    PT_SPAWN(Pt, &childPt, PtWifi_setup_AP(&childPt, Mail,
        ap_data_size > 0 ? ap_data : NULL));
    break;
}
case 8: { // TCP socket close

```

```

Wifi_TCP_close();
break;
}
case 9: { // TCP receive
// The TCP data is already in rx_buffer
// Number of bytes: TCP_Rx_bufferLength
DrvSpiTxStart(rx_buffer, TCP_Rx_bufferLength);
PT_WAIT_UNTIL(Pt, IS_RECEIVED(SPI_TX_DONE));
break;
}
case 10: { // TCP send
// Read the number of bytes to send (rsuint16)
rsuint16 len;
DrvSpiRx((rsuint8*)&len, sizeof(len));
PT_WAIT_UNTIL(Pt, IS_RECEIVED(SPI_RX_DATA));

if (len > TX_BUFFER_LENGTH)
    len = TX_BUFFER_LENGTH;

// Read data to send into tx_buffer
DrvSpiRx(tx_buffer, len);
PT_WAIT_UNTIL(Pt, IS_RECEIVED(SPI_RX_DATA));

// Send data using the TCP socket
SendApiSocketSendReq(COLA_TASK, socketHandle, tx_buffer, len, 0);
break;
}
case 11: { // Wifi chip power on/off
// Read parameter (0=off, 1=on)
rsuint8 param;
DrvSpiRx(&param, sizeof(param));
PT_WAIT_UNTIL(Pt, IS_RECEIVED(SPI_RX_DATA));

PT_SPAWN(Pt, &childPt, PtWifi_power_on_off(&childPt, Mail,
                                             param));

break;
}
case 12: { // Wifi set powersave profile
// Read parameter
// 0: low power, 1: medium power, 2: high power, 3: max power
rsuint8 param;
DrvSpiRx(&param, sizeof(param));

```

```

PT_WAIT_UNTIL(Pt, IS_RECEIVED(SPI_RX_DATA));

// Set powersave profile
Wifi_set_power_save_profile(param);
break;
}
case 13: { // Wifi set transmit power
// Read parameter
rsuint8 param;
DrvSpiRx(&param, sizeof(param));
PT_WAIT_UNTIL(Pt, IS_RECEIVED(SPI_RX_DATA));

// Set transmit power
Wifi_set_tx_power(param);
break;
}
case 14: { // Wifi chip suspend
PT_SPAWN(Pt, &childPt, PtWifi_suspend(&childPt, Mail));
break;
}
case 15: { // Wifi chip resume
PT_SPAWN(Pt, &childPt, PtWifi_resume(&childPt, Mail));
break;
}
case 16: { // Mode Soft AP . This mode init SoftAp mode, starts DHCP server,
//PT_SPAWN(Pt, &childPt, SoftAPMode(&childPt, Mail));
break;
}
case 17: { // Mode Webserver. Initialize the WEB and read the temperatures
/*PT_SPAWN(Pt, &childPt, WebServer(&childPt, Mail));
PT_SPAWN(Pt, &childPt, SPI_temperatures(&childPt, Mail));
if (app_data.HttpServerStarted)
{
SendApiHttpTerminateReq(COLA_TASK, 80);
PT_WAIT_UNTIL(Pt, IS_RECEIVED(API_HTTP_TERMINATE_CFM) &&
((ApiHttpTerminateCfmType*)Mail)->Instance == app_data.HttpInstance);
app_data.HttpServerStarted = FALSE;
}*/
break;
}
}
}
}

```

```
#endif
```

```
PT_END(Pt);  
}
```

Linies 1981-1984 L'aplicacio inicia el protothread del Webserver

```
// Start the Main protothread  
//PtStart(&PtList, PtMain, NULL, NULL);  
PtStart(&PtList, WebServer, NULL, NULL);  
break;
```

9.5 CODI DE PROVA SERVER WEB AMB EASYWEB

Aquest codi mostra que es pot realitzar també el server web amb el mètode EASYWEB. Unicament es mostrarà el codi del protothread principal, ja que la resta es pot definir igual com el servidor web.

La resta del codi es pot trobar dins del directori **EASYWEB** a l'arxiu comprimit del projecte:

```
static PT_THREAD(PtMain(struct pt *Pt, const RosMailType* Mail))  
{  
static struct pt childPt;  
static struct pt SensorUpdatePt;  
PT_BEGIN(Pt);  
// Build ID string  
PT_SPAWN(Pt, &childPt, PtAppWifiReset(&childPt, Mail));  
PT_SPAWN(Pt, &SensorUpdatePt, PtDrvIntTempMeasure(&SensorUpdatePt, Mail,  
&AppData.Temperature));  
const ApiWifiMacAddrType *mac_addr = AppWifiGetMacAddr();  
snprintf(IdStr, sizeof(IdStr), "SCK_AP_%02X%02X%02X", (*mac_addr)[3], (*mac_addr)[4],  
(*mac_addr)[5]);  
snprintf (TitleStr, sizeof(TitleStr), "RTX Web Server");  
snprintf (InfoStr, sizeof (InfoStr), "El SSID es: %s \n La temperatura es:%d,%d", IdStr,  
AppData.Temperature/10, AppData.Temperature%10);  
  
// Measure internal temperature inside the EFM32 chip.  
PT_SPAWN(Pt, &SensorUpdatePt, PtDrvIntTempMeasure(&SensorUpdatePt, Mail,  
&AppData.Temperature));  
// Sends the configuration to AppWebconfig Page  
AppWebConfigSetSsid((rsuint8 *)IdStr);  
AppWebConfigSetTitle((rschar *)TitleStr);  
AppWebConfigSetInfoText((rschar *)InfoStr);
```

```
// Sends the page to AppWebconfig
SendApiWebConfigAddPageReq(COLA_TASK, "/prueba", "PRUEBA", OnMainPage);

// Do WEB config (the module must be reset to exit WEB config mode)
PT_SPAWN(Pt, &AppData.ChildPt, PtAppWebConfig(&AppData.ChildPt, Mail, &PtList));

PT_END(Pt);
}
```