TREBALL FINAL DE GRAU:

DISSENY I IMPLEMENTACIÓ D'UN SERVIDOR WEB QUE PERMET LA CONFIGURACIÓ DE PARÀMETRES EN EL PROCESSADOR HOST PER AL MÒDUL Wi-Fi RTX4100

GRAU DE TECNOLOGIA DE TELECOMUNICACIÓ

Carlos Saleta Pereda

Consultor: Pere Tuset Peiró

RESUM

En aquest projecte hem treballat sobre la placa Smart Citizen Kit 1.5, que està formada per Arduino Due amb un Shield propi de l'empresa Smart Citizen anomenat SCDVK (Smart Citizen Development Kit), que incorpora un mòdul Wi-Fi de baix consum que rep el nom de RTX4100.

El que busquem és ampliar la funcionalitat del RTX4100 per tal de poder alliberar de tasques la placa SCDVK, que està basada en Arduino. Gràcies a aquesta ampliació de firmware, podrem donar més possibilitats a aquesta tecnologia, tenint en compte les bases donades pel model anterior Wi-Fi RN131 de l'empresa Microchip, que ha servit de precedent per a tenir una guia per treballar amb el RTX4100.

Ampliant les COLApps generades a partir del treball "Analysis, Improvement and Development of New Firmware for the Smart Citizen Kit Ambient Board", realitzat per Miguel Colom, volem aconseguir dues funcions:

- Realitzar un WebServer que permeti la configuració i visualització de paràmetres de la Databoard, com poden ser els valors de la EPROOM, dades capturades pel sensor, etc. Aquest mode s'haurà de configurar dins del RTX4100, donant una porta d'accés al Kit, per configurar-lo de manera personalitzada.

- Elaborar un mode SoftAp oferint un punt d'accés on, mitjançant dispositius mòbils, puguem accedir a la plataforma on s'assignarà una IP estàtica i, una vegada connectat al dispositiu, podrem accedir a la pàgina del WebServer que ens donarà les funcionalitats ressenyades abans.

A més, s'haurà de configurar un mode de comunicació mitjançant SPI (Serial Peripheral Interface) per a que la placa SCDVK es comuniqui amb el seu mòdul Wi-Fi i, d'aquesta manera, pugui enviar dades com, per exemple, les del sensor de temperatura i humitat del sensor C10 (el qual disposarà d'un fil d'execució per recollir les dades rebudes) i aquest les mostrarà als usuaris. Gràcies al SPI, les dades del Shield de Smart Citizen es transmetran al RTX4100 i aquest les gestionarà i exposarà de manera fluida per a l'usuari mitjançant les funcions noves abans comentades.

Per tant, tenim dues qüestions importants: la primera és la **creació d'un AP** per tal de connectar-se al mòdul i, una vegada s'hagi establert la connexió, la segona és poder

accedir al **Servidor Web** on es podran consultar i/o modificar diferents aspectes de la placa gràcies a la **comunicació SPI** per tal d'oferir opcions als diferents usuaris que entrin a la plataforma.

Paraules Clau: Smart Citizen Kit, Arduino Due, SCDVK, RTX4100, Wi-Fi, SoftAp, WebServer, SPI, EPROOM.

ABSTRACT

In this project we have worked with a Smart Citizen Kit 1.5 board. It consists of an Arduino Due that holds an actual Shield from SmartCitizen Company called SCDVK (Smart Citizen Development Kit) which incorporates a low-energy Wi-Fi module called RTX4100.

We seek to expand the functionality of the RTX4100 to liberate tasks of the SCDVK board, which is based on Arduino. With the firmware expansion, we will be able to provide more opportunities for this technology, following the rules set by the previous Wi-Fi model RN131 from Microchip Company which served as precedent to have a guide to work with RTX4100.

Extending the COLApps generated from "Analysis, Improvement and Development of New Firmware for the Smart Citizen Kit Environment Board," realized by Miguel Colom, we want to achieve two functions:

- Build up a Web Server that allows to configure and view Databoard settings, such as EPROOM values, data captured by a sensor, etc. This mode should be set in RTX4100, providing a gateway to the Kit to configure it personally.

- Develop a SoftAp mode by offering a gateway where we can access the platform through a mobile device to assign a static IP and, once connected to the desired device, being able to access the Web Server page that will give us the capabilities outlined earlier.

In addition, we ought to set up a communication mode with an SPI (Serial Peripheral Interface) in order for the SCDVK board to communicate with its own Wi-Fi module and thus data like temperature and humidity of the C10 sensor (which has an execution thread to collect the received data) can be sent and seen by the users afterwards. Due to SPI, the data from Smart Citizen's Shield will be transferred to RTX4100 and this will be managed and exposed to the user through the new features discussed earlier.

Therefore we have two important issues: the first one is to create a gateway to connect to the module and, once the connection is established, the second one is accessing the Web Server where we will be able to view and / or modify board parameters through SPI communication to provide options for different users who are using the platform.

KeyWords: Paraules Clau: Smart Citizen Kit, Arduino Due, SCDVK, RTX4100, Wi-Fi, SoftAp, WebServer, SPI, EPROOM.

Agraïments

Gràcies a tota la meva família per tot el suport que he rebut, sobretot dels meus pares i germanes. A la meva parella, l'Eva, per ser la millor amiga i companya que tinc; ha estat una font d'inspiració, lluita i esforç per a mi. Als meus avis, padrines i tiets per confiar sempre en mi (es sentirien orgullosos de mi).

Al Xavi i Pedro, els meus millors amics, per tenir sempre confiança en mi. Als meus amics i companys de feina Miguel Angel, Jose R. , Francisco O., Josep i Dídac per facilitar-me els canvis de torn per poder-ho combinar amb la Universitat.

Als meus companys d'Universitat, Alex i Alejandro, per les contínues converses i el suport moral rebut.

“Els programes han de ser escrits perquè els llegeixin les persones, i només incidentalment, perquè ho executin les màquines”

Abelson and Sussman

INDÈX DE CONTINGUTS

Taula de il·lustracions

Referencia de taules

Introducció

Text d'introducció

Arduino és un component relativament jove que va néixer a Itàlia.

Aquest petit dispositiu dóna la possibilitat de fer infinitat de projectes i de desenvolupar diferents funcions; les seves principals avantatges son la seva mida reduïda, el poc consum que requereix i la quantitat de mòduls Shield per fer diferents funcions.

A més, cal saber que estem tractant amb una tecnologia en evolució contínua i que cada vegada van sortint al mercat plaques més potents. En el nostre cas, Arduino Due, és un dels importants junt amb Arduino Mega.

Amb el temps, aquesta tecnologia s'està imposant en els Instituts Tecnològics per a que la gent tingui una primera presa de contacte amb aquest nou tipus de programació i electrònica per a realitzar diferents aplicacions en tots els camps tecnològics, com poden ser Internet, domòtica, etc.

En aquest treball, el mòdul Arduino Due ve acompanyat de la placa SCDKV amb el mòdul RTX4100. Aquest mòdul s'ha construït per l'empresa Smart Citizen la qual, gestiona diferents aspectes ambientals d'una ciutat, com poden ser la temperatura, la humitat, la il·luminació, etc. L'empresa vol que la seva placa, i en concret el seu mòdul RTX4100, faci diferents tasques de gestió i monitorització ja que afegint aquestes funcions alliberem una mica el pes de tot el sistema per a la placa SCDVK.

El mòdul RTX4100 és un dispositiu Wi-Fi que gràcies al seu poc consum i la seva gestió energètica, amb petites aplicacions fa que sigui una eina eficient i, per tant, es vol ampliar el ventall d'aplicacions en aquest mòdul i així oferir més serveis als usuaris.

Per tant, en aquest treball, tractarem tant la programació com la teoria de les COLApps del propi RTX4100 i del Firmware per el Kit de Smart Citizen, per tal d'oferir comunicació entre el RTX4100 i el SCDVK i oferir funcionalitats diverses que en un futur podrien esdevenir importants per gestions en l'entorn de les Smart Cities.

Il·lustració 1 – Logos de esquerra a dreta: Arduino, RTX A/S, Smart Citizen

Justificació del projecte

En l'actualitat, les Telecomunicacions s'estan fent servir cada vegada més en les ciutats, ja que son una eina que vol donar avantatges, tant per a la vida quotidiana de les persones, com per a les grans, mitjanes i petites empreses, per a desenvolupar els seus negocis i fer créixer la seva activitat empresarial.

Les avantatges que donen les xarxes, sobretot Wi-Fi, per recollir informació que servirà en diferents àmbits de la ciutat és el tema que tractarem en aquest projecte.

Smart Citizen, vol realitzar un projecte d'actuació directament al host, mitjançant el mòdul Wi-FI RTX4100, que ajudarà a compartir dades i a configurar el mòdul de manera remota. Per tant, estem davant d'una eina que pot oferir moltes possibilitats en l'actualitat i en un futur proper.

Motivació

En un principi, la implementació de noves funcionalitats en el mòdul RTX4100 no era el meu objectiu ja que em volia centrar únicament en Arduino, però veient les possibilitats que pot oferir, vaig decidir continuar amb aquest projecte a fi de donar-li més funcions i entendre molt més el funcionament dels components que formen la placa SCDVK i la seva programació.

Es tracta d'una tasca difícil però, a la vegada, enriquidora ja que t'aporta experiència i coneixements nous, en l'àmbit de la programació, que resulten molt interessants.

A més, el mòdul RTX4100 crida l'atenció pel seu sistema de funcionament amb l'enviament de "mails" dins del seu Sistema Operatiu i les seves COLApps. Gràcies a la seva reduïda mida i al seu consum energètic mínim, pot derivar en aplicacions per a la ciutat molt importants i de molt de servei.

Objectius

Els objectius que es volen assolir en aquest projecte son diversos i de diferents àmbits. Es volen implementar noves funcionalitats al Smart Citizen Kit 1.5 junt amb el RTX4100, per fer oferir una fàcil configuració i visualització dels paràmetres de la placa, sense fer servir recursos externs i, per tant, millorar les prestacions del Kit.

A continuació es destaquen els tres principals punts:

1. Servidor Web : Es vol desenvolupar un servidor web encastat dins del RTX4100 que doni comunicació amb els components que hi son dins de la placa SCDVK, ja siguin sensors, EPROOM, etc. També es volen inclouré paràmetres de informació per a la connectivitat. A més, caldrà que tingui un ús fàcil i simple per tal d'oferir una bona experiència a l'usuari.

2. SoftAP: Donar la possibilitat d'una connexió mitjançant Wi-Fi perquè es pugui accedir a la placa directament i, d'aquesta manera, oferir una connexió directa al servidor web de dades i configuració.

3. Documentació: Establir unes pautes de programació i elaborar un full de ruta clar i estricte per a assolir noves funcions al mòdul RTX4100 mitjançant les seves COLApps; i així, en un futur pròxim, poder afegir cada vegada més funcions per augmentar les possibilitat d'aquest mòdul tant eficient gràcies a la implementació i acceptació de les modificacions en el seu Firmware mitjançant les COLApps.

Enfoc del treball

El treball constarà de cinc grans camps que s'han desenvolupat en la realització d'aquest projecte i que han ajudat a arribar a entendre el funcionament del mòdul RTX4100 i la placa SCDVK, juntament amb la seva programació. Dins de cada camp hi ha sub-camps que cobriran tots els aspectes importants en cada àrea.

Hardware

El primer aspecte bàsic a destacar és la introducció dels elements de hardware amb els que s'ha de treballar per saber com funcionen, com es gestionen i, per tant, aprendre sobre les eines que utilitzen i com, aquestes, és poden modificar per a obtenir coneixements teòrics suficients per aplicar-los a la segona part del projecte.

Configuració d'entorn

Aquest apartat està dividit en dos sub-apartats:

El primer tracta de fer una introducció teòrica dels elements més importants per a conèixer-los per implantar les bases del desenvolupament, perquè si no es coneixen, resultarà molt difícil realitzar alguna cosa positiva en el següent apartat.

El segon ofereix les bases per a la configuració dels dos entorns amb els que es treballarà; és a dir, l'entorn Arduino per la càrrega i modificació del Firmware del mòdul SCDVK Shield (per editar i afegir codi per configurar diferents funcionalitats com pot ser la comunicació SPI) i l'entorn de càrrega de COLApps, per assolir coneixement de com funciona la càrrega d'aquestes aplicacions. A més, s'intentarà afegir informació complementària per a la càrrega del Firmware del RTX4100, perquè durant el projecte, s'ha necessitat realitzar aquest procés varies vegades al produir-se problemes amb el mòdul i el seu Firmware (càrrega errònia, Firmware corrupte, etc) que algunes vegades ha deixat de funcionar i l'única solució ha estat recarregar el seu Firmware amb les eines ofertes pel distribuïdor.

Desenvolupament

Aquest apartat es centra en la realització de l'edició dels codis tant pel Kit SCDVK com per al mòdul RTX4100, fent servir els processos apresos en l'anterior part i d'aquesta manera programar millores en el seu funcionament . Es destacarà també parts essencials del codi per al funcionament de les noves funcionalitats i, així, aprendre una mica sobre com treballa i quines avantatges pot donar. El codi serà lliurat en format "zip" amb les noves funcionalitats.

Diagrames de flux

En aquesta secció hi són tots el diagrames de flux dels processos comentats en la memòria i detallats en com avancen amb el seu procés de creació

Resultats finals

Resum de les millores incorporades i dels resultats aconseguits en el procés de desenvolupament fent especial èmfasis en els objectius marcats en aquests TFG.

Futurs desenvolupaments

Aquí s'exposaran els camins a seguir i les funcionalitats que manquen per implementar dins del Kit. Tanmateix, s'explicarà en quin punt es troben les funcions generades i el codi implementat per donar una ajuda a futures ampliacions en les aplicacions que es volen oferir.

Informació i conclusions

Per finalitzar, s'exposaran aspectes de caire informatiu juntament amb les conclusions i els problemes generats durant el desenvolupament de les noves aplicacions per poder oferir informació de valor per a futures aplicacions o ampliacions en el Kit.

Resultats potencials

Els resultats que es poden extraure quan estiguin totes les funcionalitats requerides per Smart Citizen podria ser, per exemple, una xarxa d'estacions meteorològiques on, amb un consum mínim, es pot oferir informació de temperatura, humitat, etc. A més podria ser modificable i canviar paràmetres d'us mitjançant una pàgina de configuració.

Tenir una xarxa d'aquests dispositius per Catalunya amb un servei centralitzat de gestió on cadascun ocupi una direcció i, d'aquesta manera, consultar en temps real totes i cadascuna de les estacions, que estaran col·locades per la ciutat o territori al qual es vol destinar.

Il·lustració 2- Exemple de funcionalitat.

ESTAT DE L'ART

Antecedents

Els seus antecedents son la placa SCK 1.1, anterior a la 1.5 que és amb la que s'està treballant en aquest projecte. El kit anterior portava incorporat un mòdul Wi-Fi RN131, anomenat WiFly.

La versió actual està construïda basant-se en un Atmel ATMEGA32U4 (8bits). Una de les principals millores a versions anteriors són que incorpora un port USB per la comunicació i d'aquesta manera estalviem la conversió d'aquest.

Les principals característiques d'aquest micro-controlador son:

- Consum 200mA.
- 16/32Kbytes de EEPROM disponible per emmagatzemar el Firmware.
- 1.25/2.5 Kbytes de SRAM interna.
- 512Kbytes / 1 Kbytes EEPROM interna.
- Interfície JTAG.
- Suport USB 2.0 de manera nativa.
- Temperatures entre -40º i 85º.
- Voltatges 2.7 fins a 5.5 Volts.
- Freqüències entre 8Mhz a 2.7 Volts i 16Mhz a 4.5 Volts.
- Modes dormitori: Idle, Power-save, Power-down, ADC Noise Reduction, Standby, and Extended Standby.

Les conclusions que es poden extreure de les principals característiques som que es tracta d'un sistema de molt baix consum, amb molt poca memòria de EEPROM, que en un futur podria comportar problemes per a carregar el Firmware. Té modes d'estalvi d'energia molt sofisticats, que ajuden per generar un mode de funcionament més eficient amb el consum i ofereix robustesa davant condicions climatològiques dures, per tant, és un candidat excepcional per a funcionar en entorns exteriors.

Per últim, s'ha de saber que es tracta d'un sistema ràpid gràcies al marge de freqüències en les que treballa i s'ha de tenir en compte que doblant la freqüència, el consum no es dobla de manera proporcional, per tant l'escalabilitat en la rapidesa també es eficient.

Il·lustració 3- Smart Citizen Kit 1.1 Ambient Board

El mòdul Wi-Fi RN131

El mòdul Wi-Fi que incorpora la placa SCDVK, el RN131, anomenat WiFly, es comunica mitjançant comandes per sèrie amb la placa, per a transmetre les dades dels diferents sensors que té a la plataforma web.

Com ja hem comentat, el mòdul Wi-Fi és el RN131, està format per un *SoC* (System on Chip) que integra les modalitats de radiofreqüència 802.11b/g i el Stack/Pila TCP/IP. Només necessita 4 connexions (TX, RX, PWR, GND) per a crear una connexió simple sense fils. A part, aquest mòdul proporciona diferents aspectes com són el so, el moviment i l'acceleració, que són dades analògiques sense necessitat de maquinaria addicional.

Il·lustració 4- RN131

Tanmateix, aquest mòdul té limitacions i impediments, perquè ofereixen un marge i un ventall de funcionalitats i aplicacions finit, és a dir, l'empresa Microchip (creadora del RN131), va limitar el seu Firmware i només accepta les ordres que estan referenciades en el document explicatiu del mòdul, ofert per la mateixa empresa, i aquesta limitació impedeix un creixement de funcions i un problema molt gran en l'àmbit tecnològic.

Les ordres suportades per aquest mòdul nomes inclouen aquestes:

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

Com veiem, la taula d'ordres és molt limitada, per això Smart Citizen va dissenyar el Kit 1.5 amb el RTX4100, que ofereix un sistema àmpliament modificable i amb una maniobrabilitat molt gran i així pal·lia les mancances del RN131 i ofereix als programadors, una eina més variable i que suporta més canvis gràcies a la seva edició de funcionalitats.

Hardware

Smart Citizen Kit 1.5

Arduino Due

Arduino és una plataforma electrònica oberta per crear prototips basats en hardware i software de manera flexible i fàcils d'utilitzar. Està basada en una placa amb un micro-controlador i un entorn de desenvolupament.

Aquesta placa inclou el micro-controlador Atmel SAM3X8E 10 ARM CortexM3, amb una arquitectura de 32 bits, a més dels ports d'entrada i sortida tant digitals com analògics i sortides PWM i de comunicacions, per al control d'objectes físics com poden ser LEDs, servos, botons, etc.

És una plataforma lliure i modificable que pretén promoure diferents principis per a ser considerats elements open hardware:

- Publicació de la documentació, incloent-hi els arxius dels dissenys, que permeten la seva modificació i distribució.

- Especificar quina porció del disseny és oberta en cas de que no s'alliberin tots els seus components.

- Oferir el software per al visionat dels arxius de disseny i de la documentació, perquè es pugui escriure el codi open-source de manera fàcil.

- La llicència no ha de restringir que es vengui o comparteixi la documentació necessària, per tant no demana una tarifa per la seva venta o la dels seus derivats.

- La llicència no discrimina a cap grup o persona.

- La llicència no fa restriccions en cap camp o activitat d'ús de l'obra, és a dir, no es pot limitar la seva utilització únicament per a negocis o prohibir la seva utilització.

- La llicència de l'obra no pot dependre d'un producte particular.

- La llicència no ha de restringir altre hardware o software, és a dir, no pot insistir en que altres components de hardware o software externs als dispositius siguin també open-source.

- La llicència pot ser neutral, cap disposició de la mateixa s'ha de basar en una tecnologia especifica, part o component, material o interfície per al seu ús.

Complint amb aquestes premisses, els dissenys i esquemes d'Arduino es distribueixen sota la llicència "Creative Commons Attribution-ShareAlike 2.5"[5].

Avantatges de Arduino

Els avantatges d'aquesta plataforma, a part de la facilitat que donen per a treballar-hi amb la programació del micro-controlador, són:

- Assequible: Molt més econòmiques que altres plataformes de micro-controladors. La més elevada de preu en el mercat pot arribar a uns 60 euros.

- Multi-plataforma: Accepta treballar amb Windows, Mac OSX i Linux. En canvi, la majoria dels altres micro-controladors estan limitats nomes a Windows.

- Entorn de programació: Es fàcil d'assimilar, incloent-hi a principiants que comencen a treballar amb electrònica, perquè es una plataforma simple i que per a usuaris avançats ofereix una flexibilitat molt gran.

- Software ampliable i codi obert: es tracta d'una plataforma open-source "Wiring", és a dir, està publicat sobre una llicència lliure i preparat per a ser ampliat per programadors de tots els nivells.

Models Arduino

Existeixen gran nombre de models Arduino de diferents mides i potencies, oferint més o menys entrades, tant analògiques com digitals, segons el projecte al que estigui destinat.

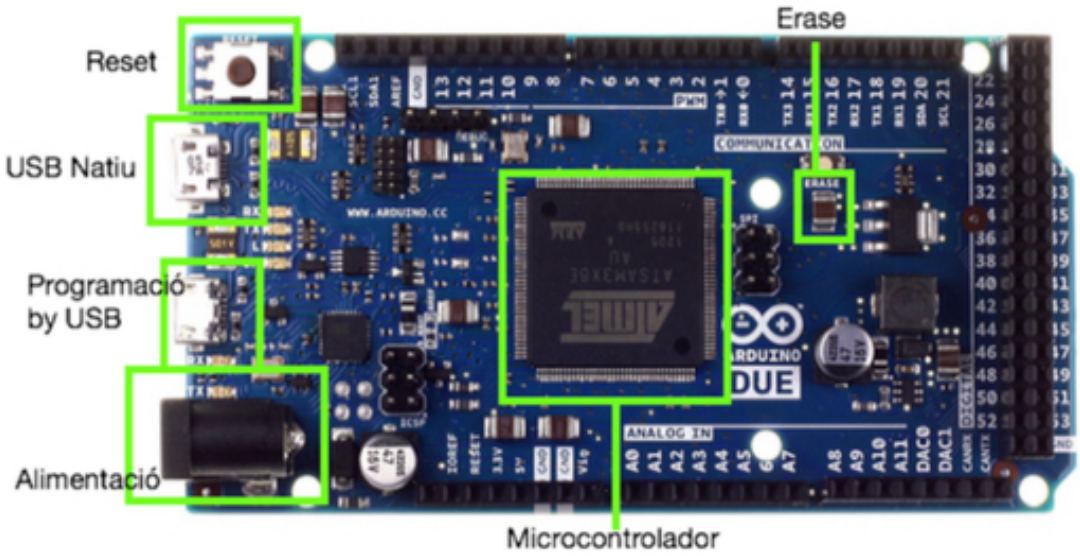
Els models mes emblemàtics són:

- Arduino UNO
- Arduino YUN
- Arduino MEGA
- Arduino Mini
- Arduino Micro
- Arduino DUE

En aquest projecte es treballarà amb Arduino Due, que ofereix les següents característiques:

- *Micro-controlador ATMEL SAM3X8E 10 ARM CortexM3.*
- *32 bits.*
- *512Kbytes de memòria flash per a programes i aplicacions*
- *96 Kbytes de SRAM.*
- *Controlador DMA per compensar la càrrega de CPU de tasques que executin processos de manera molt intensa.*
- *54 pins digitals per a connexions que seran I/O.*
- *12 entrades digitals per a connexions analògiques.*
- *4 ports sèrie per a connexió.*
- *Rellotges CPU de 84 Mhz.*
- *Dos connectors USB.*
- *2 DAC (digital – analògic).*
- *2 connectors TWI.*
- *Un connector d'alimentació Jack estàndard.*
- *Connectors SPI.*
- *JTAG.*
- *Boto d'alliberament de memòria flash i de reset del sistema.*
- *Alimentació de la placa 3,3 Volts.*

- *Límits d'alimentació entre 6-16 Volts.*
- *Voltatge recomanable 7-12 Volts.*



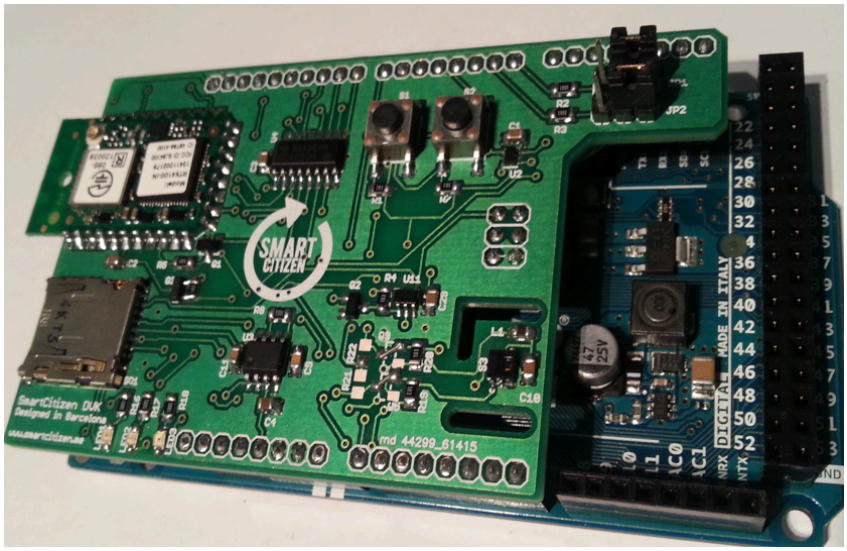
Il·lustració 5- Arduino Due.

SCDVK SHIELD

A mode d'introducció, la placa en funcionament per Smart Citizen és la versió 1.1, anterior a la que s'està treballant en aquest projecte. Cal conèixer una mica la seva antecessora per a tenir un punt de partida sobre aquest dispositius hardware. La documentació es pot trobar en l'apartat d'antecedents per tal de tenir un punt de partida prou ampli.

El nou SCDVK Shield que va crear Smart Citizen, tracta de donar més avantatges als desenvolupadors gràcies a la seva potència superior i manipulació que ofereix el seu mòdul Wi-Fi RTX4100. El Firmware encara està en procés de desenvolupament i s'ha portat a l'arquitectura ARM Cortex M3 d'Arduino Due i, en un futur, es vol portar també al ARM Corte M0+ (SAMD21).

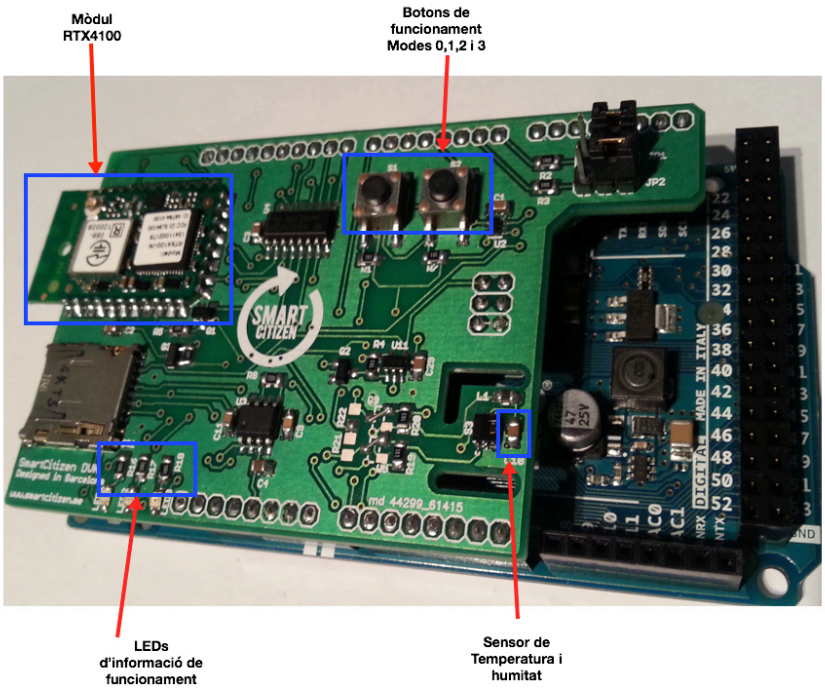
Les variacions amb el seu antecessor son el canvi de MCU, perquè té implementat el nou ATMEL ARM Cortex M0+ i que té una arquitectura similar a la del Arduino DUE (Atmel SAM3X8E ARM Cortex-M3).



Il·lustració 6- Smart Citizen Kit 1.5.

L'empresa Smart Citizen està buscant el desenvolupament de funcions SPI per la comunicació, encara que ja s'han fet proves a nivell de hardware mitjançant comunicació SPI manual gràcies al projecte de Miguel Colom "Analysis, Improvement, and Development of New Firmware for the Smart Citizen Kit Ambient Board".

La placa amb la que treballem en aquest projecte permet, mitjançant el seu sensor de temperatura i humitat, oferir dades reals de l'entorn en el que està. A continuació es mostraran els punts importants a conéixer per saber com està estructurada.



Il·lustració 7- Smart Citizen Kit 1.5 , Components

En la imatge podem veure que diposem de diferents parts ben diferenciades, que s'han de saber per tal d'establir un vincle de comprensió amb el producte per utilitzar-lo. Les parts més importants són:

- Mòdul Wi-Fi RTX4100.
- LEDs de informació del mode de funcionament.
- Sensor de temperatura i humitat ubicat al component C10.
- Botons de funcionament per pasar d'un mode al altre.

L'empresa Smart Citizen ens posa a disposició el Firmware del kit 1.5, que es composa dels següents arxius que s'han de modificar (concretament el SCDVK.cpp) per a donar-li funcionalitats SPI per la comunicació de les diferents dades al mòdul RTX4100:

Il·lustració 8- Arxius Firmware SDVK 1.5

Les característiques d'aquesta placa són mostrades a continuació:

- Microcontrolador EFM32G230F128 (RTX4100).
- Processador de 32Mhz ARM Cortex- M3.
- Memòria Flash Interna de 128Kbytes.
- 16 Kbytes de memòria RAM útils per l'execució d'aplicacions e interacció.
- Alimentació mitjançant LDO.
- Control d'alimentació mitjançant dos LDOS adicionals que alimenten el mòdul Wi-Fi.

Modes de funcionament de la Placa SCDVK

Com s'ha vist a les imatges anteriors, disposa de dos botons (S1 i S2) per a passar d'un mode a un altre i els LEDs ens informarán en quin mode està el Kit.

Il·lustració 9- Botons S1i S2 i LEDs d'informació de mode.

Els diferents modes de funcionament son els següents:

MODE 1

Aquest mode és el definit per defecte i al connectar la placa els LEDs verd i taronja estaràn encesos. Si el taronja no s'encén, indica que el RTX4100 està buit o té algun problema amb el seu Firmware, per tant, s'haurà de fer una neteja de Firmware o carregar una COLApp per activar-lo. Aquest mode envia, mitjançant el USB Serial a una velocitat de 9600 bauds, els valors capturats pel sensor d'humitat i temperatura. També informa de si hi ha connectada una tarjeta SD al socket. Aquests valors es poden trobar al Firmware, que més endavant en aquesta memòria es podràn saber.

MODE 2

En el moment que es polsi el boto S1, el LED verd començará a parpellejar i ens indicarà que està en mode terminal i que es podrà carregar COLApps mitjançant la comunicació per USB. Una vegada carregada la COLApp el LED groc s'encendrà. Si es tornés a prémer el boto S1, tornaria al MODE 1.

MODE 3

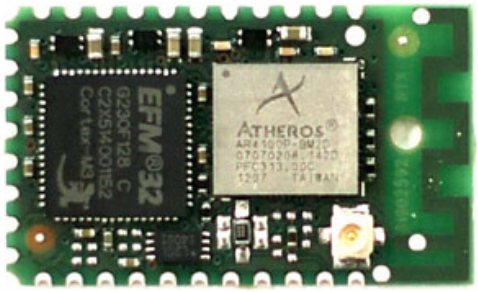
Al prémer el boto S2, el LED blau s'activarà indicant que ja es poden pujar aplicacions mitjançant les eines d'AmelieSDK fent servir el COLAController i pujant COLApps al mòdul Wi-Fi RTX4100. Farem un analisis més complet en l'apartat de preparació de l'entorn.

MODE 4

Al tornar a prémer el boto S2, els LEDs verd i blau s'activaran i es podrà accedir al Firmware del RTX i ens permetran l'actualització del Firmware del RTX4100, ja que en alguns moments del desenvolupament, s'ha hagut de carregar de nou el Firmware per errades en el mòdul o errors de càrrega de les COLApps. Cal saber que durant la càrrega del Firmware, ja sigui modificant o no al RTX4100, no es pot desconectar. Es pot saber quan està carregant en el moment en que la llum blava està parpadejant i la llum taronja es queda fixa de manera molt dèbil.

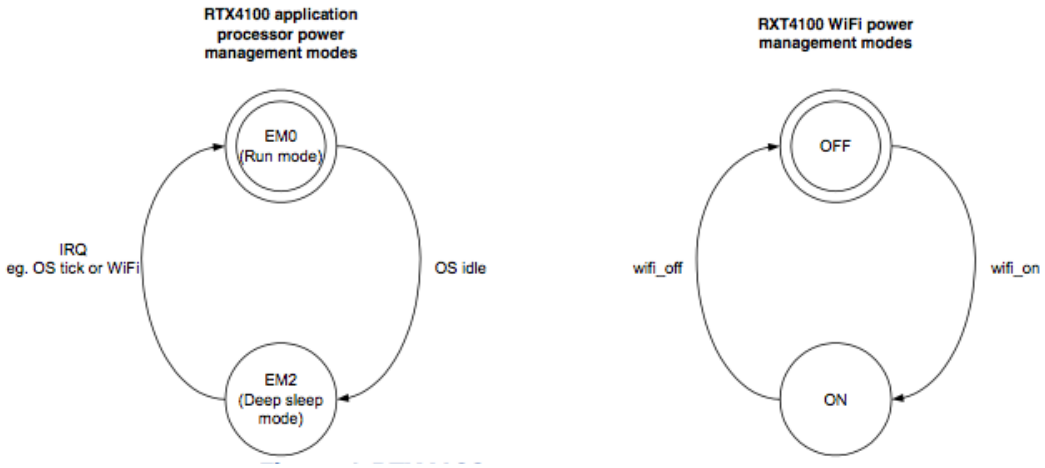
RTX4100

Tenint present com està distribuida la placa SCDVK, s'ha de conéixer com és el mòdul Wi-Fi que incorpora aquesta plataforma.



Il·lustració 10- Mòdul Wi-Fi RTX4100

Com s'ha dit abans, el RTX4100 té avantatges respecte al seu predecessor per ser una eina completament configurable i amb un entorn simple, en el que el desenvolupador pot crear aplicacions de manera lliure per oferir més funcionalitats.
A més, té unes gràfiques de consum que són molt reduides gràcies als seus modes d'espera per inactivitat i també per ser eficient amb la seva feina.



Il·lustració 11- RTX4100, modes eficients de consum energètic.

Gràcies a les funcions de posar en espera o desactivar el Wi-Fi per inactivitat o per no se necessari en certs moments, aconsegueix reduir el seu consum i augmentar de manera molt significativa la seva eficiència respecte a altres mòduls Wi-Fi. Aquest disposa de mode Run i mode Sleep, apart de tenir l'opció d'encendre i apagar la part Wi-Fi del mòdul.

A continuació és mostrarà una taula de consum, on s'hi descriuen diferents modes de funcionament, on la configuració: connexió Wi-Fi en mode 802.11n, el mòdul amb un voltatge de 3.5 volts i que el terminal de referència de configuració de l'aplicació és "Wi-Fi PsProfile Low". D'aquesta manera és té una idea del consum segons el mode en el qual està funcionant el RTX4100.

Descripció	Condició	Consum típic
Standby Wifi off	WiFi off	2.7μA
Standby Wifi on	WiFi on	0.5mA
WiFi activat i associat, IEEE PS latencia alta	DTIM=1000ms Psprofile=low	0.8mA
WiFi activat i associat, IEEE PS latencia alta	DTIM=300ms Psprofile=low	0.5mA
WiFi activat i associat, IEEE PS latencia alta	DTIM=100ms Psprofile=low	2.8mA

Taula 1- Consum energia segons característiques de funcionament

Les característiques principals que és poden extrauré de la documentació del RTX4100 són les següents:

- Mode Standby per a estalvi en el consum d'energia.
- Chip Wi-Fi de baix consum (85mA).
- Suport per a IPv4 i IPv6.
- Modes de funcionament per estalvi d'energia.
- Comunicació amb altres mòduls mitjançant diferents tipus de comunicació: UART, SPI, I2C.
- Ports GPIO.
- Timers.
- ADC i DAC ports.
- Límit de temperatures entre -40º i 85º (suporta entorn exterior).

Els avantatges indiquen que és un mòdul preparat per condicions extremes, doncs suporta condicions exteriors i a més té un consum inferior al RN131 , ja que el seu

consum és de 210mA i el RTX4100 només en consumeix 85mA. A part de tot el que s'ha pogut veure, la gran avantatge respecte al seu predecesor, l'anterior Kit de Smart Citizen, és la possibilitat de desenvolupar aplicacions, gràcies a AmelieSDK, de manera lliure i per tant, donant opció a que els desenvolupadors puguin donar-li més funcionalitats al hardware.

Entorn de Carrega i edició del Firmware

Teòrica de funcionament

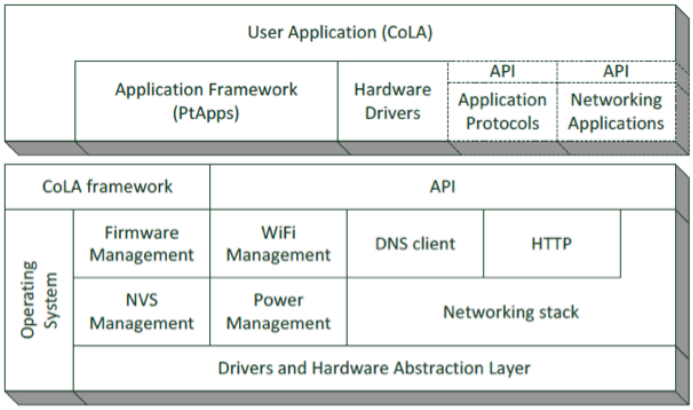
Abans d'entrar en matèria dins l'entorn de càrrega, tant per la plataforma SCDVK com per al mòdul RTX4100, s'ha de fer una breu explicació dels punts més importants en l'apartat teòric, ja que influeix de manera molt significativa en els següents punts i en el desenvolupament de les funcionalitats demanades, això afavoreix a adquirir una comprensió nítida i de manera estructurada dels punts teòrics més importants en el projecte.

Teòrica software RTX4100

Aquests paquet de API's relacionades amb aquest mòdul estan facilitades pel mateix proveïdor i ajuda a entendre el funcionament i com desenvolupar-hi noves eines. El Firmware està dividit en les COLApps i el mòdul Firmware que dóna suport a aquestes.

La plataforma AmelieSDK ajuda a programar les noves COLApps amb ajuda de les API's i el framework COLApps. Aquestes COLApps, que són aplicacions d'usuari, es poden carregar al mòdul sense incidir en el seu Firmware, de manera que el programador pot errar amb la programació sense afectar al mòdul, aquest procés evita problemes perquè que queden separats per dos grans blocs d'utilització.

L'arquitectura Software del RTX4100 consta de les següents parts:



Il·lustració 12- Arquitectura RTX4100

Es pot veure que hi han dos grans blocs, un dedicat a les Co-Located Application (COLApps) i l'altre encarat al sistema firmware del mòdul.

Primer bloc

Co-Located Application(CoLA)

User Aplication	Implementació de funcionalitats
Application Framework	Conjunt d'aplicacions(PtApps) que funcionen a partir de Protothreads i que donen funcionalitats de manera comuna, es a dir, implementa aplicacions comuns reutilitzades per totes les aplicacions de referencia.
Hardware Drives	Els drivers comuns que s'han fet servir en aplicacions de referència que donen l'accés al hardware.
APIs –Networking Applications	Component opcional. Implementa aplicacions de xarxa com FTP, TFTP, HTTP, WebServer.
APIs-Applications protocols	Implementa un nombre de traducció de protocols com podria ser la codificació i descodificació de XML per a missatges Payloads.

Taula 2- Bloc Co-Located Application (CoLA)

Segon Bloc

Plataforma Firmware

CoLA Framework	Dona servei per a implementar un programa model on l'aplicació es connecta de manera dinàmica amb els proveïdors de serveis en les capes baixes del sistema
API	Mostra totes les funcionalitats necessàries per l'aplicació per a implementar el dispositiu Wi-Fi, com podria ser un sensor de humitat o un actuador que connecti algun tipus de dispositiu extern.
Sistema Operatiu	Dona funcionalitats per a realitzar tasques amb un baix cost energètic com ara les COLApps
Networking Stack	Dona la funcionalitat per implementar la IP de la xarxa per a IPv4 i IPv6
DNS Client	Ofereix la traducció de dominis a adreces IP mitjançant les crides al servidor DNS
HTTP	Configuracions comuns per a implementar HTTP Server i client
Wi-Fi Managment	Component encarregat de tots els aspectes de la connexió Wi-Fi, com poden ser els AP. També ofereix ajuda a tasques d'estalvi d'energia.
Power Managment	Gestió de rellotges interns de la MCU i d'aquesta manera poder controlar l'energia que es consumeix. Al tenir un control temporal es poden controlar de manera més exhaustiva el consum energètic dels components
Firmware Managment	Ofereix la possibilitat d'actualitzar de manera remota la aplicació, es a dir, mitjançant les COLApps
NVS Managment	Implementació d'un sistema d'emmagatzematge no volàtil en la flash interna de la MCU
Drivers	Dona funcionalitat per a un cert nombre d'aplicacions amb una sèrie de controladors de hardware per a perifèrics MUC i el mòdul Wi-Fi

Taula 3- Bloc plataforma Firmware.

Protothreads

Els Framework de les aplicacions estan basats en Protothreads, perquè el sistema té un recursos finits i limitats i per tant ofereix una càrrega d'aquests mes lleugera. Aquests Protothreads faciliten el codi i donen més agilitat al sistema, perquè la càrrega de treball es molt menor.

A més ofereixen l'execució de fils d'execució (Threads) per separat i que poden estar actius fins que uns altres s'activin, i això pot succeir per una pausa en el fil d'execució fins que succeeixi una condició i dóna facilitats per a gestionar-los de manera més fàcil i ràpida. Aquests específics no necessiten reserva de memòria i utilitzen menys RAM per als processos.

En un entorn com el RTX4100 és fa essencial disposar d'aquest tipus de recursos que ajuden al seguiment, ampliació, manteniment i consum del sistema de manera eficient i econòmica.

Aquests Protothreads, ajuden a establir feines en cascada sincronitzada, és a dir, quan un acaba, comença l'altre i així successivament, o fer una execució per passos relacionats, per exemple, en aquest projecte, s'han fet crides de manera ordenada per crear un servidor web, ja que el primer que tindrem que fer es crear el AP, després generar el servidor i per últim generar la seva pàgina web. Es pot veure que ha d'existir una correlació i un ordre lògic dels successos, per mantenir un control en el desenvolupament. Més endavant, quan aprofundim en el codi, es veurà més clar aquest tipus de conceptes.

Les macros definides estudiades dins del AmelieSDK tenen sis tasques ben diferenciades i que a continuació es farà una breu explicació:

Per inicialitzar:

PT_INIT(pt) : Primer és tenen que inicialitzar i a continuació és podran cridar a partir d'un altre fil d'execució. "Pt" es refereix a un punter a la estructura de control del Protothread.

Per finalitzar:

PT_EXIT(pt): Executa la sortida del Protothread tenint en compte, que si es cridat per un altre, el fil d'execució "pare" podrà continuar funcionant sense problemes

PT_RESTART(pt): Realitza un reinici del protothread, per tant executa un bloqueig i inicia des del principi.

Jeràrquics:

PT_WAIT_THREAD(pt, thread) : realitza una pausa fins que el fill indicat amb Thread no finalitzi la seva execució.

PT_SPAWN(pt, child, thread): Fa una crida al fill i espera fins que aquest finalitzi. És un bon sistema per un procés d'execució esglaonada, perquè ajuda a controlar l'execució per parts del programa.

Cessió de fils d'execució (Threads):

PT_YIELD(pt): Executa un bloqueig al Protothread i cedeix la execució a la resta de processos i aquests continuaràn des del punt on va patir la aturada.

PT_YIELD_UNTIL(pt, condition): Inicia un bloqueig de un Protothread i en cedeix l'execució a la resta, tenint en compte que per retornar necessitarà complir la condició. Com succeeix amb PT_YIELD, el procés continuarà des del punt on es va aturar.

Bloqueig:

PT_WAIT_UNTIL(pt, condition) : realitza un bloqueig i esperarà fins que la condició sigui certa.

PT_WAIT_WHILE(pt, condition): realitza la mateixa tasca que l'anterior amb la diferència que si es compleix la condició, quedarà en estat d'espera.

RTX Operating System (ROS)

El ROS és, de manera resumida, el sistema operatiu del RTX, que està basat en missatges anomenats “Mails”, que donen les funcionalitats necessàries per a realitzar tasques individuals i que aquestes es puguin comunicar-se entre elles mitjançant el sistema.

En el sistema es crearà un fil d’execució principal on generarà tots els fils secundaris que necessiti per a realitzar les tasques, podem veure que alguns aspectes estan basats en sistemes centralitzats.

Els “Mails” que s’envien contenen etiquetes amb identificadors i altres dades que al arribar a al gestor els posarà en cua conforme vagi rebent-los i indicarà el camí que agafarà i quin fil d’execució el rebrà.

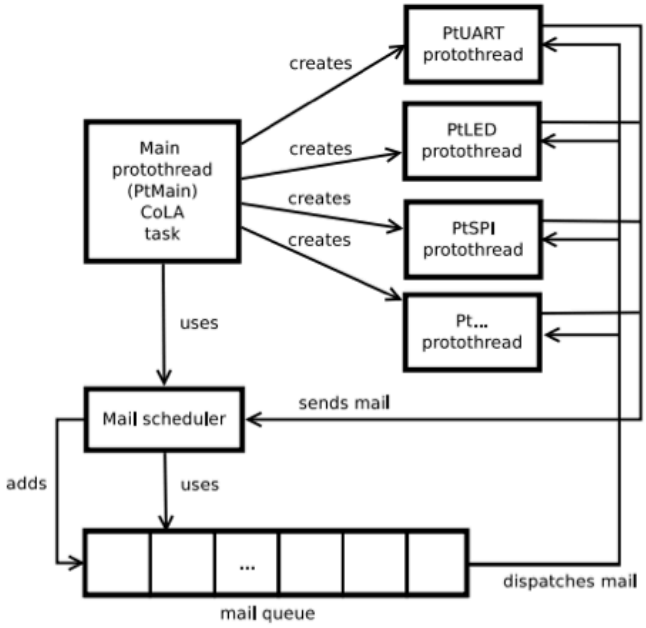
A més de les diferents avantatges, disposa d’altres funcions, com un sistema temporitzador que executa subscripcions a “mails” mitjançant el S.O. i que dóna molt joc per aplicacions que necessitin d’un temporitzador.

Per emmagatzemar les dades generades o configuracions del sistema, disposa d’una memòria no volàtil que dóna la possibilitat d’emmagatzemar les dades importants. Recordem que aquesta memòria té dos grans blocs, una part reservada pel Firmware (512bytes) i la resta serà per poder carregar COLApps d’usuari.

Per accedir-hi a la NVS en tenim dues funcions especifiques:

NvsRead(): Realitzarà una lectura de la NVS i d’aquesta manera sabrem les dades contingudes.

NvsWrite(): Es tracta d’una escriptura en la NVS, per tant, es farà servir quan es vulgui guardar alguna dada important, com poden ser dades de configuració.



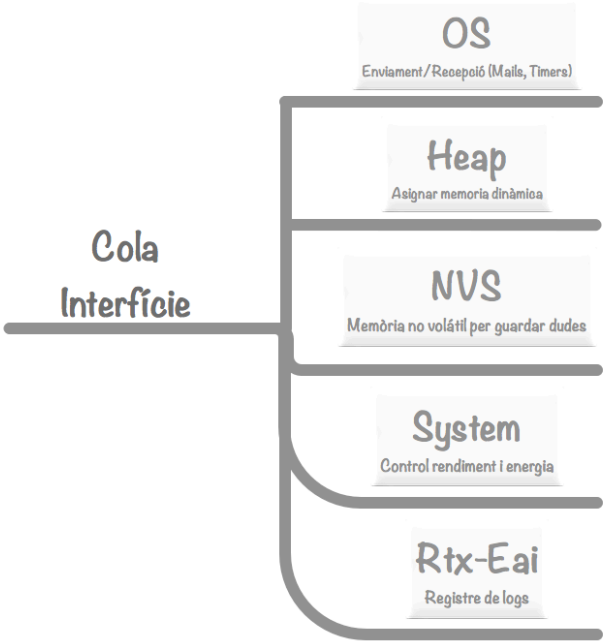
Il·lustració 13- Esquema funcionament ROS mitjançant "Mails".

Podem veure que CoLAtask serà necessari en totes les aplicacions CoLA, ja que serà el encarregat d'enviar els "Mails" al sistema.

AmelieSDK

Es tracta del programa utilitzat per les COLApps. Aquesta fa servir funcions de diferents camps i un conjunt de "Mails", d'aquesta manera poder accedir a la pròpia plataforma, sent de diferent tipus, com pot ser el Socket API, IpConfig Api, etc.

La Interfície de CoLA exporta totes les funcions del ROS a un bloc especial de dades, on estan emmagatzemades en memòria flash. Aquesta interfície dóna funcions d'accés a diferents aspectes.



Il·lustració 14- Cola Interfície

En Amelie SDK s'inclouen diferents programes de configuració i comunicació per a pujar les COLApps al mòdul RTX. A més, es pot veure exemples de COLApps i un glossari de llibreries i APIs que podem fer servir en el nostre disseny. Alguns estan en carpetes com Components o Include.

Cal tenir en compte aquests arxius, que facilitaran la feina de programar, perquè moltes macros i rutines estan descrites dins d'ells. El temps de dedicació en fer un anàlisis de les llibreries importants i de les macros i Protothreads que s'hagi de fer servir, pot ser una tasca llarga, ja que hi ha molta informació, pel que cal tenir un full d'apunts per fer anotacions de tot el que s'estigui inspeccionant.

Comunicació SPI

La comunicació per SPI caldrà tenir-la present ja que es farà servir per a comunicar de manera automàtica les dades dels sensors al RTX4100 i d'aquesta manera oferir les dades mitjançant el Servidor Web.

El protocol SPI funciona en full-duplex i treballa de manera síncrona, d'aquesta manera els dispositius es comuniquen entre ells al mateix temps. El funcionament és simple, en tenim dos dispositius A i B, un realitzarà les tasques com a "Master" i l'altre com "Slave". "Slave" funcionarà de manera que enviaran i rebran informació del "Master" i aquest farà el mateix enviant informació al "Slave". Caldrà tenir un registre de desplaçament diferent per a cadascun d'ells, ja que emmagatzemen els bits rebuts

de manera paral·lela i els registres faran la funció de fer una conversió per a la transmissió d'informació.

La transmissió i sincronització es realitza mitjançant quatre senyals bàsiques descrites a continuació:

- *SCLK (rellotge)*: El pols que marca la sincronia de la comunicació. A cada pols s'envia o es rep un bit que també s'anomena TAKT.
- *MOSI (Master Output Slave Input):* Sortida de dades de "Master" i entrada de dades a "Slave".
- *MISO (Master Input Slave Output):* Sortida de dades de "Slave" i entrada de dades a "Master".
- *SS (Slave Select):* Selecciona un "Slave" o per activar mitjançant el "Master" un "Slave, també anomenat SSTE.

De manera resumida, el que s'espera és enviar les dades des del SCK Shield al mòdul (RTX4100) i esperar ordres del canal SPI, una vegada rebudes, les executa i torna a enviar la informació al mòdul.

El funcionament per passes seria d'aquesta manera:

1. La topologia feta servir per la comunicació es Anell.
2. "Master" configura rellotge tenint en compte les limitacions de "Slave".
3. "Master" selecciona el "Slave" amb nivell lògic 0.
4. Cada cicle de rellotge SPI s'executen transaccions en full-duplex .
5. "Master" envia un bit per la línia "MOSI", "Slave" fa la lectura i envia per línia "MISO".
6. "Master i "Slave" tenen 8 bits (8 posicions, de 0 a 7) de registre. Master rep a la posició 0 via MISO de "Slave" i envia la posició 7 via MOSI a "Slave", d'aquesta manera va desplaçant-se el registre, és a dir, els bits rebuts aniran al bit de menys pes i el enviat serà el de més pes, fent la funció de cua FIFO (First Input First Output) tenint en compte la capacitat del registre de 8 bits.

En el treball realitzat per Miguel Colom " Analysis, Improvement, and Development of New Firmware for the Smart Citizen Kit Ambient Board" es van implementar 15 "Commands reference" explicats en la memòria de Colom.

Teòrica Programació HTML

Les bases que es tenen que assolir en l'entorn de programació Web per a generar la pàgina dins del servidor són les següents:

- Creació d'etiquetes per els identificadors.
- Afegir etiquetes de classes per a configurar l'aspecte de la Web.
- Aprendre diferents atributs: width, height, margin, padding, etc.
- Utilitzar correctament la capçalera de la Web.
- Generació del codi HTML5.
- Control en el tancament d'etiquetes per el correcte funcionament.
- Investigar la manera d'incrustar la fulla d'estils dins del codi HTML.
- Saber diferents aspectes de creació de tabs, posició i transacció.
- Tenir en compte la combinació entre els colors de Background i de la font.

S'ha de remarcar, que els problemes generats degut a la mida del Buffer han sigut diversos, perquè el disseny de la pàgina web, contemplava diferents pestanyes de configuració.

Programari utilitzat

El programari utilitzat per a desenvolupar les diferents pars del projectes són:

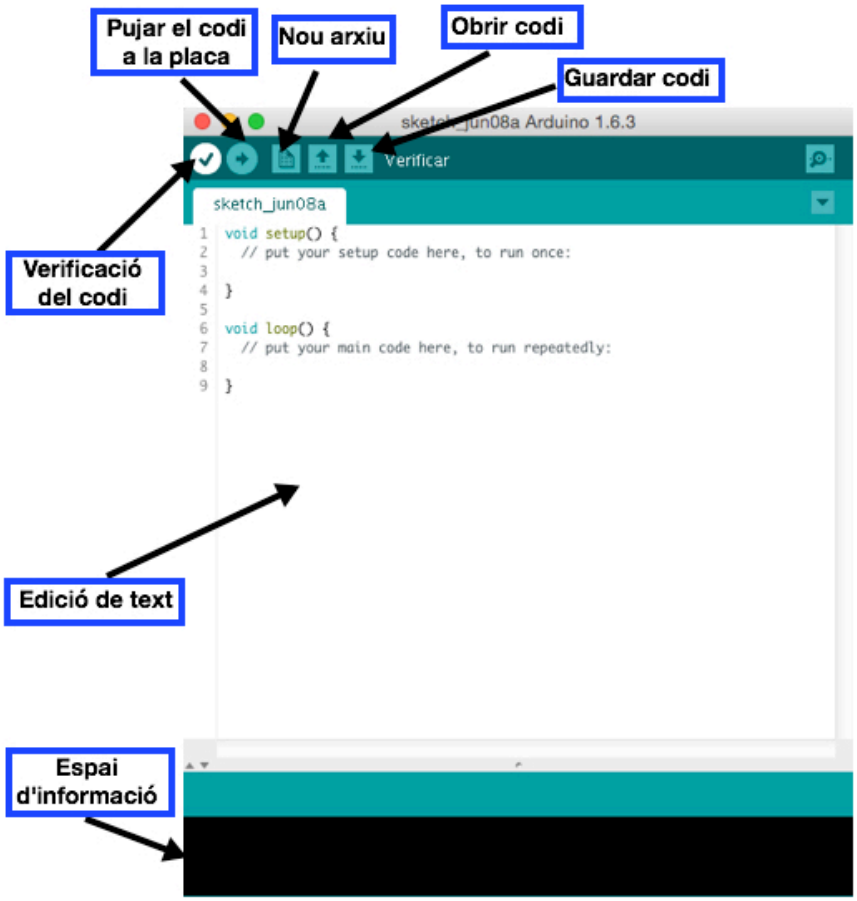
- Per la creació de continguts Web→ Brackets
- Per la creació a generar el codi de la COLApp-→Sublime
- Per la creació de codi per Arduino → Arduino IDE

Entorn Arduino SCDVK Shield

L'eina feta servir per a treballar amb els mòduls Arduino, és Arduino IDE, que és un entorn de desenvolupament que implementa el llenguatge Processing de Wiring i que és molt semblant al codi C++.

L'eina IDE es gratuïta i distribuïda per la seva pàgina web on està disponible per als diferents S.O. de l'actualitat.

En la següent imatge podem veure l'edició per al Firmware Arduino, que es composa del editor de text, la compilació de codi i la càrrega per a pujar el codi a la placa o descarregar-lo d'aquesta



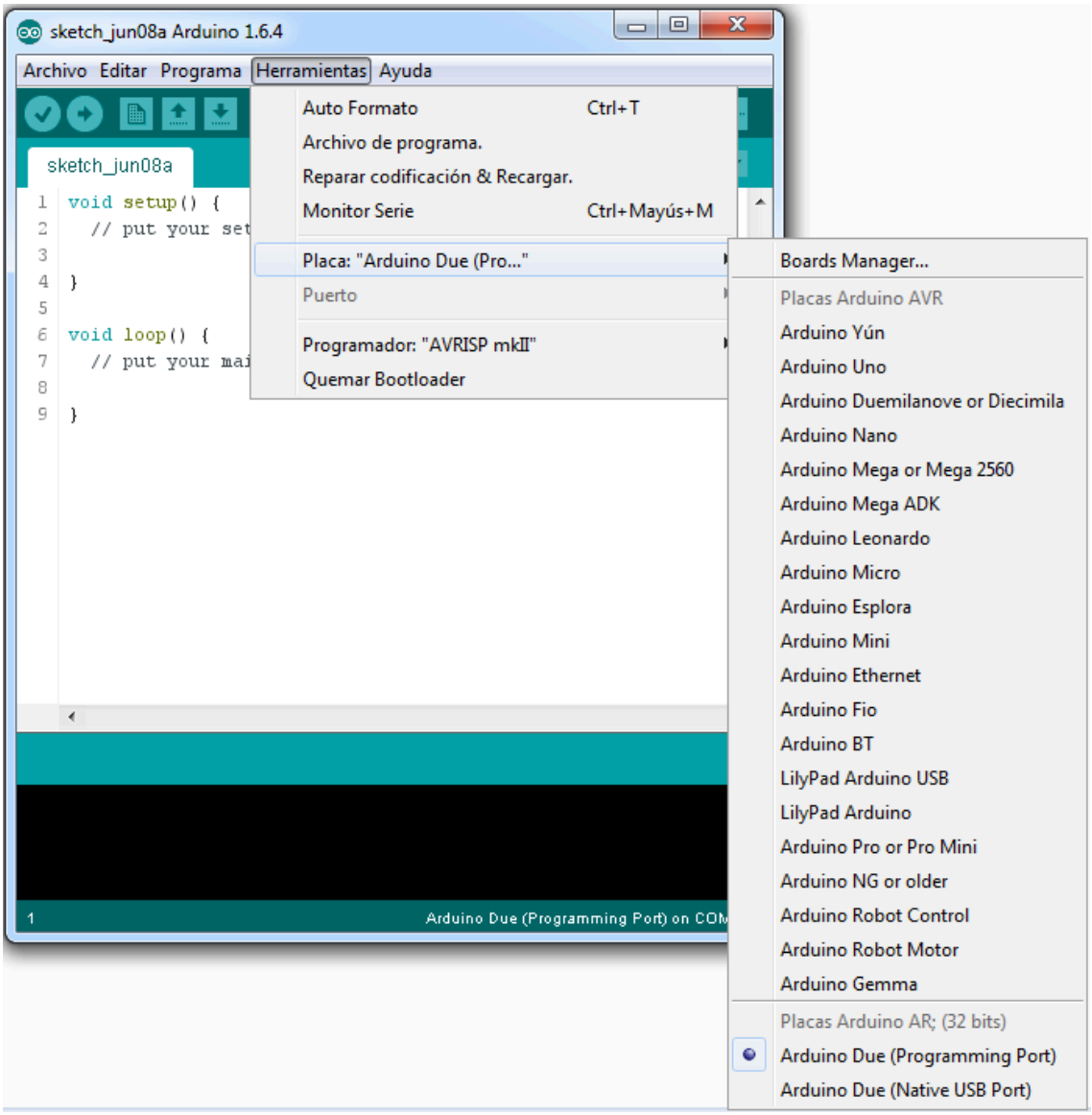
Il·lustració 15-Entorn d'edició codi Arduino

L'entorn segueix una línia principal anomenada "Sketch" on executarà unes tasques de definició posades dins de "void setup()" i després de definir-les, treballarà amb elles en una espècie de cercle d'execució anomenat "void loop()".

Configuració entorn Arduino Due

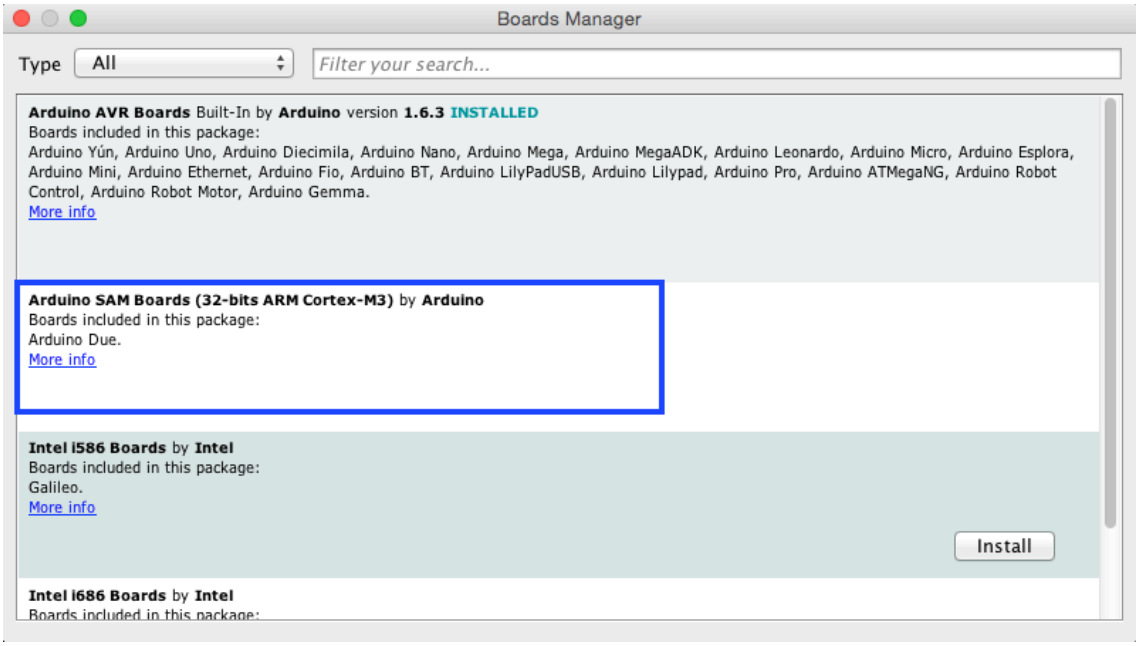
Una vegada instal·lat la IDE de Arduino, ens farà falta configurar el seu entorn per tal de que la nostra placa Smart Citizen Kit 1.5 sigui identificada per l'entorn software.

El primer que tindrà que fer es obrir el IDE de Arduino i anar a Board Manager.



Il·lustració 16- configuració entorn Arduino IDE.

Seguidament el que es farà es instal·lar els components necessaris per a poder treballar amb la placa. A continuació dins de Boards Manager es descarregarà el paquet de “Arduino SAM Boards (32-bits ARM Cortex-M3) by Arduino” com es mostra a la següent imatge.

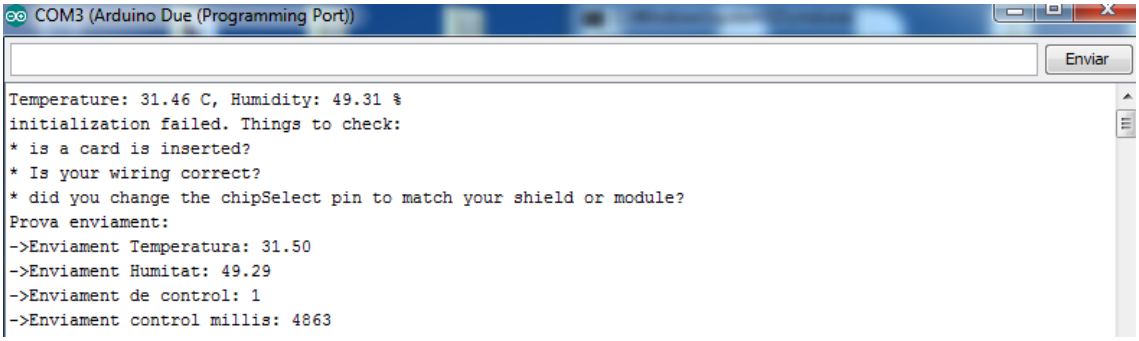


Il·lustració 17- Instal·lació llibreries per a Boards SAM de Arduino

Una vegada feta la instal·lació, nomes caldrà saber per quin port COM està connectada la placa i així es connecti de manera correcta amb el nostre entorn de treball.

Una vegada instal·lat tot el necessari, per comprovar que la placa Arduino funciona correctament, podem accedir al monitor sèrie per veure si hi ha comunicació entre l'ordinador i el Kit, si no mostres cap dada, la comunicació fallaria i s'hauria de revisar tant els drivers que hi ha instal·lats al equip per a reconèixer la placa Arduino, com el port de comunicacions que assigna el sistema a la placa.

En el cas present, no és van tenir problemes amb aquesta qüestió i al posar en marxa el monitor sèrie funcionava de manera correcte, es pot veure en la següent il·lustració:



Il·lustració 18- Monitor Serie, Arduino Due.

Podem veure a la il·lustració que el Kit ens dóna dades i per tant es comunica correctament.

Els arxius que composen el Firmware base facilitat per Smart Citizen en la seva pàgina https://github.com/fablabcn/SmartCitizenDVK/tree/master/Firmware són:

- SCDVK.cpp : És el codi on s'implementaran les tasques i les funcions que s'hauran de realitzar.
- SCDVK.h: Arxiu Header de SCDVK.
- SmartCitizen_DVK.ino: Arxiu que executa les funcions que és defineixen en el SCDVK.cpp.
- Constants.h: Definició de constants per a poder facilitar la programació del arxiu SCDVK.cpp.

Apart, Arduino té la possibilitat de incloure llibreries de diferents tipus on s'implementen funcions típiques per a la interconnexió de dispositius o mòduls electrònics que poden ser de la mateixa pàgina de Arduino o realitzats per terceres persones.

Anàlisis de programació en Arduino/SCDVK Shield

El llenguatge de programació de les plaques Arduino té com a base C++ amb el seu entorn IDE. El programa pròpiament d'Arduino es coneix amb el nom de "Sketch", abans esmentat, i el seu sistema de programació es una implementació de Wiring, amb unes similituds físiques amb Processing, ja que es la seva base.

Cal saber que la programació en Arduino té tres temes principals que es tenen que conèixer:

- La seva estructura
- Les seves variables
- Les funcions definides

La seva execució té dos principals punts els quals es tenen que conèixer el seu funcionament:

Void setup(): Només es carrega una vegada quan s'inicia el programa i s'encarrega de definir les seves variables i les declaracions. Es una part important a conèixer, ja que aquí es on farem el procés "START" del codi.

Void loop(): Aquesta part s'executarà contínuament i aquí estarà el desenvolupament del codi principal i s'executarà una vegada el "SETUP" hagi acabat de iniciar les variables i funcions

Funcions a tenir en compte

Les funcions que és fan servir son importants per realitzar el codi, doncs s'hauran de conèixer, perquè sinó la tasca de programar en Arduino resultarà més complicada i tenint-les en compte ajuden a crear el codi de manera correcta.

Les mes rellevants son:

pinMode(pin,mode)	En el primer argument s'indicarà el pin digital al qual és vol afectar i el mode podrà ser per configurar-ho com a entrada amb el valor INPUT o com a sortida , amb el valor OUTPUT
digitalRead(pin)	Es fa una lectura del pin digital indicat i retornarà un valor "HIGH" o "LOW"
digitalWrite(pin,value)	S'escull el pin digital en el qual es vol escriure el valor que podrà ser "HIGH" o "LOW"
analogRead(pin)	Fa una lectura del pin analògic indicat amb una resolució

	predeterminada de 10 bits. El funcionament esta nomes encarat als pins que van des del A0 al A11. El resultat pot ser un enter que pot prendre el valor entre 0 i 1023.
AnalogWrite(pin,valor)	Dona la possibilitat d'escriure un valor entre 0 i 255 (8bits) en el pin que indiquem.
analogReference(type)	S'utilitza per a configurar el valor de voltatge que es farà servir en l'entrada analògica, que normalment per defecte, són 3,3 volts.
analogReadResolution(bits)	Amb els bits indicats establirem la resolució amb la que es pot llegir una entrada analògica. El predeterminat són 10 bits i pot arribar a suportar 12bits de resolució , es a dir, de 0 a 4095.
analogWriteResolution(bits)	De manera anàloga a la anterior però ara per a definir la resolució d'escriptura.
Delay(valor)	Estableix una pausa amb el valor indicat en l'argument, la unitat d'aquest valor són milisegons.
Millis()	Dona la quantitat de mili segons que la placa porta en funcionament amb el programa que estigui executant.

Taula 4- Instruccions rellevants Firmware Arduino Due.

Llibreries Arduino

Les llibreries donen funcionalitats i al ser una plataforma lliure es poden crear unes de noves seguint uns estàndards marcats.

Les llibreries més importants i de rellevància en el projecte son:

Standard Llibreries

SD

Per a llegir i escriure en una targeta SD. Ens donarà la possibilitat de comunicar-nos amb la SD per a guardar arxius e imatges e inclús emmagatzemar una Web per no sobrecarregar el mòdul RTX4100.

SPI

Per a la comunicació amb dispositius, fent servir el Bus Serial Peripheral Interface (SPI). Dona la possibilitat d'establir comunicació amb el RTX4100 i d'aquesta manera intercanviar dades a més de tenir opció a configurar diferents aspectes.

Wi-Fi

Per a connectar a internet fent servir el mòdul Arduino Wi-Fi Shield. Encara que no s'utilitzi, es millor donar-li una ullada per saber certs aspectes de funcionament.

EEPROM

Escriure i llegir a la memòria no volàtil.

Timing

DateTime

Una llibreria per a mantenir la data i el temps actual en el software, d'aquesta manera podríem enviar aquestes dades per SPI al mòdul RTX4100 per a informar del temps real.

Utilitats

PString

Es tracta d'una classe de poc pes per a imprimir als buffers.

Streaming

Un mètode per a simplificar les declaracions de mostreig o impressió

Entorn RTX4100 Firmware

En el proces de desenvolupament algunes vegades s'ha hagut de restablir el Firmware del mòdul RTX4100 per errades en la càrrega de COLApps o per causes externes al software, generades pel hardware de la placa, ja que hi ha hagut problemes amb ella.

Els passos a seguir per a restablir el mòdul seràn els següents:

1. Entrarem a la pàgina de RTX i ens registrarem per accedir a la base de dades.
2. Anirem a la secció de descàrregues i descarregarem el pack anomenat RTX41xx_Platform_Firmware_Update_Pack_v1.6.0.52.zip.
3. El descomprimim en una carpeta en C:, per exemple C:\rtx4100.
4. Activarem el mode 4 de la placa, és a dir, dos pressions al boto S2, per més informació dels modes, aneu a "Modes de funcionament de la placa SCDKV".
5. Accedim a la carpeta descomprimida i executem el packet FwuRTX41xx_V0106_N0060.exe i ens demanarà el port COM al qual està conectada la placa.

Il·lustració 19- Inici carrega Firmware RTX4100.

6. Li diem el port i ens demanarà que activem el mode pulsant dues vegades el S2, una vegada executat aquesta instrucció començarà la càrrega del Firmware.

Il·lustració 20-Establiment de connexió per carrega Firmware RTX.

Il·lustració 21-Procés de carrega del Firmware RTX.

Al instal·lar de nou el Firmware, la COLapp carregada s'esborrarà i es tindrà que tornar a pujar-la al mòdul.

Entorn RTX COLapps (AmelieSDK)

Carpetes del paquet Amelie SDK

Abans de tot, per treballar amb Amelie SDK i la càrrega de COLApps, el primer que s'haurà de fer es descarregar el paquet AmelieSDK (AmelieSDK_v1.6.0.58.exe) de la pàgina www.rtx.dk/LPW/RTX4100, on es realitza el registre per accedir a tot un seguit de documents i software per treballar amb el mòdul.

Una vegada descarregat el paquet i instal·lat al ordinador i és crea un conjunt de diferents carpetes dins de la carpeta C:\AmelieSDK que a continuació les desglosa de manera resumida:

Ruta: C:\AmelieSDK\v1.6.0.58

3Party	Inclou fitxers de tercers amb importants arxius d'acces de periferics.
ColaController	Aplicació per la carrega de COLApps via UART al mòdul RTX4100.
Components	Essencials per a desenvolupar noves COLApps.
Documents	Documentació per l'usuari i programador.
Include	Headers més utils i comuns per la creació de COLApps.
Projects	COLApps d'exemples.
Tools	Eines utilitzades per el desenvolupament de les COLApps. DoxyGen,RTxbuild.

Taula 5- Ruta C:\AmelieSDK\v1.6.0.58.

Seguidament es realitza una inspecció dins de Projects, on s'ha de fer un especial èmfasi en la carpeta "Components" dins de Amelie, que ajuda de manerea molt important al desenvolupament de les COLApps gràcies a les carpetes PtApp, Drivers, Api, ApiMps i WebConfig.

Ruta: C:\AmelieSDK\v1.6.0.58\Projects\Amelie\Components\

ApiMps	Arxius Heades
Api	Api pels mails en qüestións d'enviament i conformador del paquet "Mail".
Drivers	Aquí estan implementats diferents drivers de Hardware.
PtApps	Important carpeta on destaca: AppWifi.c/h, AppWebConfig.c/h, AppSocketc./h i AppDhcpd.c/h.
WebConfig	Arxius Header per la definició del Api Webconfig interfície.

Taula 6- Ruta C:\AmelieSDK\v1.6.0.58\Projects\Amelie\Components.

Dins de COLApps s'accedeix a la carpeta on hi han exemples de COLApps i hi han els següents directoris.

Ruta: C:\AmelieSDK\v1.6.0.58\Projects\Amelie\COLApps\

Apps	Aplicacions de guia i on tindrem que crear el projecte.
Config	Arxius de configuració.
Lib	Arxius dels mòduls RTX4100 i 4140.
Rsx	Utilitat per API de enviament de "Mails" mitjançant via UART.
Scripts	Petits programes que es fan servir amb el Linker.

Taula 7- Ruta C:\AmelieSDK\v1.6.0.58\Projects\Amelie\COLApps.

Dins de Apps trobarem exemples de COLApps que es poden consultar per a obtenir una idea del desenvolupament.

Ruta: C:\AmelieSDK\v1.6.0.58\Projects\Amelie\COLApps\Apps

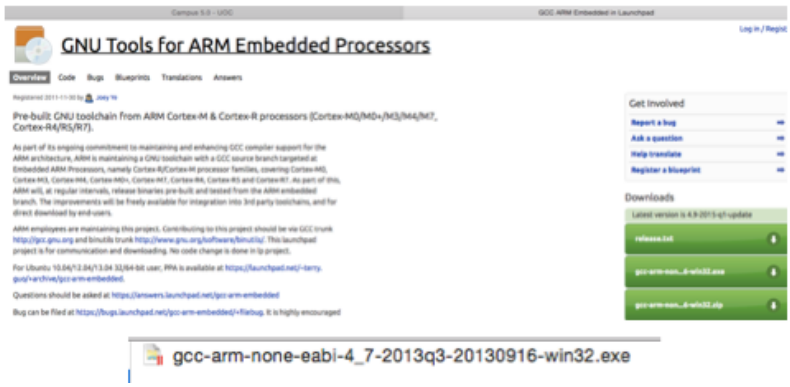
Blinky	Aplicació on es tracta d'encendre Led del SCDVK Shield.
Servidor/Client TCP i UDP	Es tracta de COLApps on es generen servidor i clients per als dos protocols.
Aplicació Terminal	COLApp per la funcionalitat de terminal.
Sensors de temperatura	COLApps que treballen amb els sensors.
SoftApTcpServer	Exemple de una implementació d'un servidor TCP en mode SOFTAP.
WebServer	Exemple d'un servidor WebServer i que ajudarà molt al desenvolupament en aquest projecte.

Taula 8- Ruta C:\AmelieSDK\v1.6.0.58\Projects\Amelie\COLApps\Apps.

Instal·lació d'eines per RTX4100

Una vegada instal·lat el paquet AmelieSDK, es farà necessari un altre programa per poder compilar el codi, es tracta del paquet "toolchain", que inclou el compilador i creació d'arxius perquè despres en el procés de Upload amb el COLAcontroller es puguin pujar al mòdul RTX4100.

L'arxiu es pot descarregar de https://launchpad.net/gcc-arm-embedded/4.9/4.9-2015-q1-update/+download/gcc-arm-none-eabi-4_9-2015q1-20150306-win32.exe per a Windows.



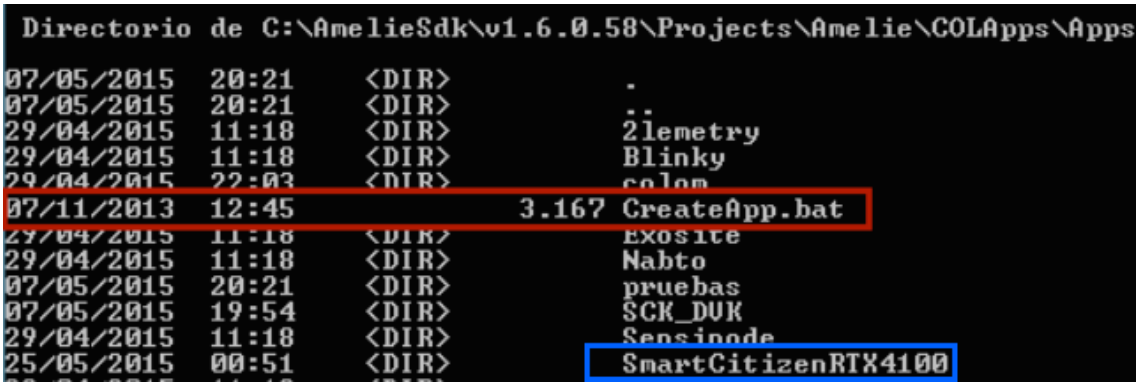
Il·lustració 22- GNN Tools per la compilació del codi per RTX4100.

Una vegada descarregada e instal·lada aquesta eina ja es té tot el necessari per a començar a treballar.

Creació del projecte

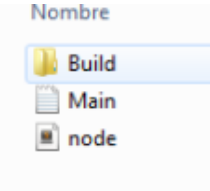
Per a crear el projecte anirem a la Ruta: C:\AmelieSDK\v1.6.0.58\Projects\Amelie\COLApps\Apps

Aquí trobarem l'arxiu "CreateApp.bat" , que es l'encarregat de crear el nou projecte COLApp.



Il·lustració 23- Creació projecte dins de COLApps.

Per a creació després de la instrucció .bat s'assigna un nom al projecte. Una vegada creat, sortirà un missatge que indicarà que ha estat creat satisfactòriament. Dins del projecte es genera l'arxiu node, on indicarà nodes d'unió amb altres arxius. L'arxiu "Main.c" , on hi estarà el codi de la COLApp.

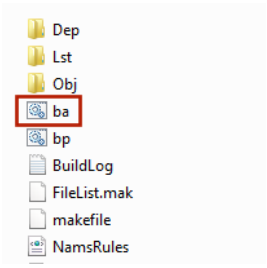


Il·lustració 24- Arxius creats dins del projecte.

La carpeta Build en parlarem seguidament al apartat de “Compilació d’un projecte”.
El projecte d’aquesta memòria el podem veure indicat amb un reacuadre blau.

Compilació d’un projecte

Per la compilació del codi escrit en el “Main.c” s’accedeix a la carpeta Build/RTX4100_WSAB/ on es troven diferents arxius, el “ba” executa la compilació del codi del “Main.c”.

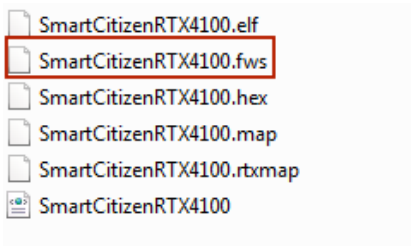


Il·lustració 25- Arxius creats dins de la carpeta Build\RTX4100_WSAB.

Una vegada compilat es creen diferents arxius, el que té l’extensió “fws” és el que es farà servir per a pujar-lo al mòdul RTX4100.

```
C:\AmelieSdk\v1.6.0.58\Projects\Amelie\COLApps\Apps\SmartCitizenRTX4100\Build\RTX4100_WSAB>ba
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\SmartCitizenRTX4100\Build\RTX4100_WSAB


Finished: Projects\Amelie\COLApps\Apps\SmartCitizenRTX4100\Build\RTX4100_WSAB
Time: 0:0
Updating C:\AmelieSdk\v1.6.0.58\Projects\Amelie\COLApps\Config\BuildInfo.inc


AppCommon.c
AppSocket.c
AppSntp.c
Main.c
AppLed.c
AppWiFi.c
AppShell.c
AppDhcpd.c
AppWebConfig.c
DrvAdc.c
DrvButtons.c
DrvLeuart.c
DrvLsm303dlhc.c
DrvI2cIntf.c
DrvLps331ap.c
DrvApds990x.c
DrvGpioInt.c
DrvHlH613x.c
DrvHoneywellAirflow.c
DrvHoneywellPressure.c
DrvMvs0409.c
DrvSi1353lt.c
DrvIntTemp.c
DrvNtcTemp.c
DrvPir.c
DrvUsart.c
BuildInfo.c
DrvSpi.c
DrvSpiSlave.c
ImageHeader.c
FwsInfo.c

Linking SmartCitizenRTX4100.hex

   10398 bytes code
     552 bytes data
    2036 bytes const
   12434 bytes flash

Creating intel-hex file
Creating binary file
Creating FW update file
Time: 00:04:44b
```

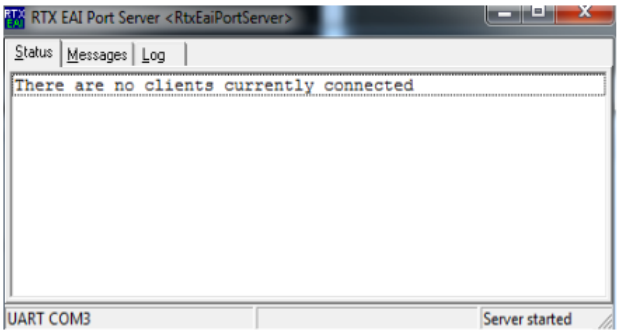
Il·lustració 26- Compilació del codi.



Il·lustració 27- Creació d'arxius després de la compilació.

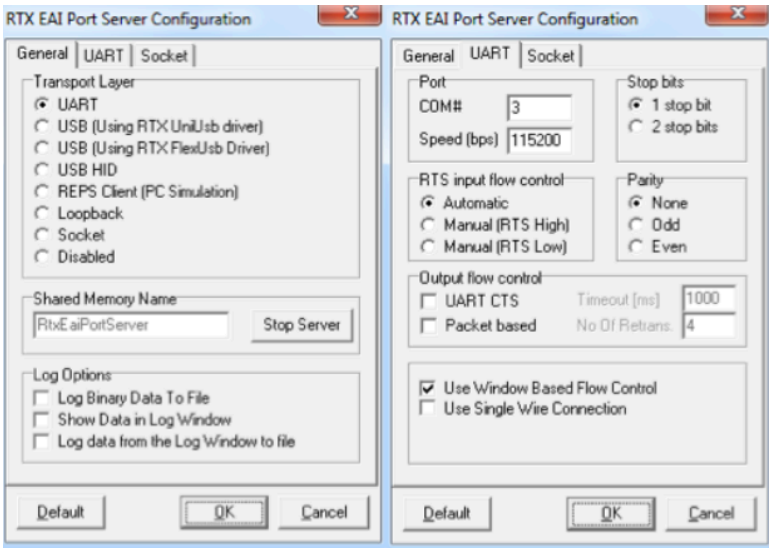
Carrega de la COLApp creada al mòdul RTX4100

Una vegada compilat, es connecta la placa al PC s'ha d'executar el programa que es va instal·lar amb el paquet "AmelieSDK" anomenat RTX EAI Port Server.



Il·lustració 28- RTX EAI Port Server Interfície.

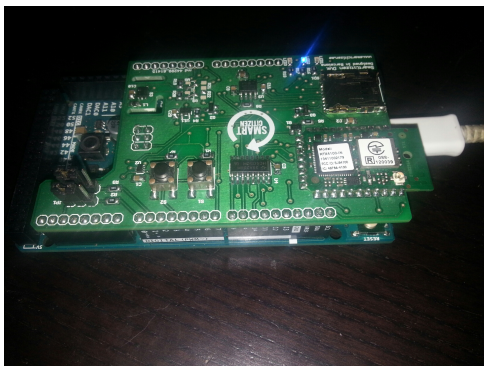
Aquest programa s'ha de configurar entrant amb el boto esquerra del ratolí dins de les opcions i indicant el COM al qual està connectat la placa i desmarcant UART CTS, tal i com es pot veure a la imatge següent on es pot veure la configuració utilitzada.



Il·lustració 29- Paràmetres RTX4100.

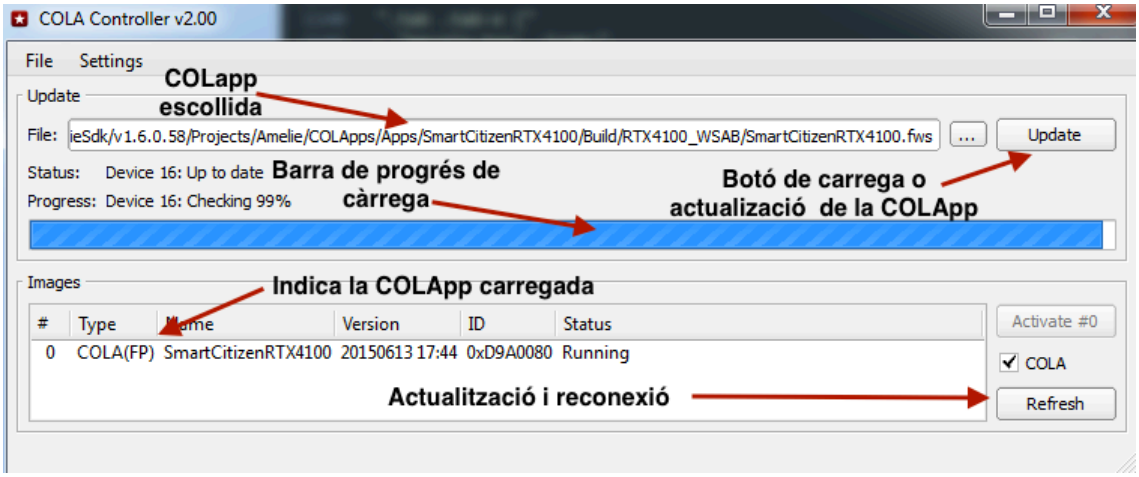
Una vegada configurat i tenint compilat el codi, s'executara un altre programa del paquet "AmelieSDK" anomenat COLAcontroller, que es l'aplicació per a pujar el codi compilat al mòdul.

Una vegada obert el programa és polsa una vegada el boto S2 per entrar en mode 3 i d'aquesta manera ja es pugui actualitzar o pujar una nova COLApp.



Il·lustració 30- Mode 3 de la placa per la carrega de COLApps.

Amb el COLAcontroller carregarem una vegada connectada la placa la nova COLApp o la modificació de la mateixa. A continuació s'indica alguns aspectes importants dins del entorn del programa.



Il·lustració 31- ColaController Interfície.

En aquesta part s'ha après com funciona cada entorn dedicat a la càrrega. Primer s'ha vist la càrrega de Firmware per la SCDVK Shield com per la càrrega del Firmware del RTX i la pujada de COLApps al mòdul. Per tant es un aspecte important per a establir una agilitat amb els diferents camps del projecte.

Mode de depuració

A mode informatiu, s'informa de que "AmelieSDK" incorpora una eina de seguiment dels elements mitjançant els "Mails" de sortida i entrada. Aquesta eina es l'aplicació "Mail SD Tracer" i resulta una eina de depuració on es poden definir de quin dels elements es vol fer un seguiment dins del menú Task Options.

Desenvolupament del projecte

En el disseny i desenvolupament del projecte, es tenen que definir diferents aspectes en el codi, com la inclusió dels arxius necessaris (#include) que es troben dins del paquet "AmelieSDK" per poder realitzar les tasques que s'explicaran a continuació. A més es tenen que definir (#define) diversos paràmetres del SoftAP i de la configuració de la IP estàtica. És defineixen també diferents variables d'emmagatzematge per la NVS dins de la "struct", com son la "Temperature", "TemperatureRTX", HttpServerStarted, ap_info, etc.

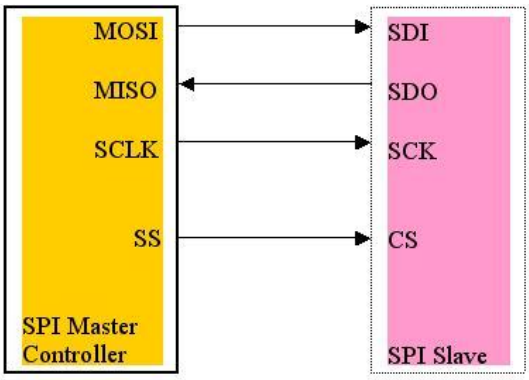
Comunicació SPI SCDVK Shield a RTX4100

Instruccions rellevants

Per fer la tramesa de dades entre el Arduino Due, amb el seu Shield SCDVK, al RTX4100 es desenvoluparà el protocol de comunicació SPI tant en el Firmware Arduino com en la COLApp generada per aquest projecte.

S'ha de tenir en compte que SPI funciona de manera diferent a qualsevol placa Arduino i per tant s'ha de fer una implementació de manera que aconsegueixi transmetre dades del seu sensor col·locat en el SCDVK Shield.

Per establir una comunicació SPI, cal que entre els dos punts de connexió, un adopti el paper de "Master" i l'altre de "Slave". El "Master" serà el SCDVK Shield, que és el que enviarà les dades al RTX4100, que farà el paper de "Slave" i aquest serà el que mostri aquestes dades rebudes mitjançant la comunicació SPI.



Il·lustració 32- Comunicació SPI.

Abans de passar al desenvolupament del codi cal conèixer algunes instruccions essencials per a desenvolupar el codi dins de l'entorn Arduino.

SPI.Begin()
Inicia el bus de comunicació SPI. Cal

SPI.Begin()
Tancament del bus de comunicació SPI

SPI.beginTransaction(SPISettings(speed,dataOrder,dataMode))
Inici del bus de comunicació per la transferència de dades amb les propietats dels arguments: - Speed: Velocitat de transacció - dataOrdre= hi ha de dos tipus, bit més pes (MSBFIRST) o bit de menys pes (LSBFIRST) - dataMode: Tenim quatre tipus -> SPI_MODE0, SPI_MODE1,SPI_MODE2 i SPI_MODE3

SPI.endTransaction()
Finalitza la comunicació i la tramesa de dades.

El mode de funcionament per la transacció de dades en la instrucció SPI.BeginTransaction ha de ser SPI_MODE0, sent el Clock Polarity (CPOL) = 0 i Clock Phase (CPHA) = 0. RTX4100 dóna molts problemes amb els altres modes de

funcionament i no estableixen comunicació, l'únic mode funcional i que accepta el mòdul Wi-Fi es el MODE0.

Desenvolupament

El desenvolupament consta de dues parts: Enviament de Temperatura i Humitat i enviament de paràmetres i modes de funcionament mitjançant els polsadors S1 i S2 de la placa SCDVK Shield.

La primera part ha estat assolida per aquest projecte amb alguns problemes de comunicació, quedant-se fora la segona part per falta de temps.

Establiment de comunicació i enviament de dades del sensor

En aquesta fase es vol incidir en diferents aspectes del disseny del codi del Firmware de Arduino DUE/SCDVK Shield.

Per començar s'obre l'arxiu SCDVK.cpp i es defineix el "Header" de la comunicació SPI al inici del codi per tal de utilitzar les instruccions per posar en marxa el bus de comunicació.

```
#include <SPI.h>
```

Una vegada fet la crida del "Header", crearem tres bytes per contenir dades i un per al control de les voltes al bucle que es col·locarà al "Main". El primer contenidor serà "temperatura", el segon serà "humitat", el tercer serà "control" igualat a cero perquè farà la funció de comptador de voltes i per últim "volta_numero", que s'utilitza de comparador entre el que s'envia i el que es rep.

Per començar en SCDVK::begin(), on s'inicien totes les interfícies, farem la crida corresponent per iniciar el bus SPI, sent aquesta crida la següent:

```
SPI.begin();
```

S'ha pogut comprovar que si en SCDVK::Begin no es posa el nivell actiu (HIGH), aquest no s'activa per defecte i per tant no funciona, llavors amb la següent instrucció dins de void "SCDVK::begin()" arreglarem aquest problema:

digitalWrite(RTX_CS, HIGH)

Després es defineixen la velocitat, el ordre dels bytes i el mode (que com hem dit abans MODE0, ja que altres modes no funcionen) al SCDVK::Main(), on definirem les propietats principals del bus de comunicació SPI com la velocitat a 10000, ja que es suficient per a transmetre aquestes dades. A més es defineixen el "dataOrder" al bit de més pes "MSBFIRST" i el mode "SPI_MODE0" amb la ordre:

SPI.beginTransaction(SPISettings(10000,MSBFIRST,SPI_MODE0));

Una vegada s'ha definit les propietats i definit els bytes de contenció (temperatura, humitat, control i volta_numero), nomes caldrà enviar les dades de manera que cadascun de les següents instruccions enviarà cadascuna de les dades explicades:

- Per la Temperatura

  Temperatura=SPI.transfer(getTemperature());

- Per la Humitat

  Humitat = SPI.transfer(getHumidity());

- Per al control de que la informació s'envia

  Volta_numero=SPI.transfer(control));

Aquestes tres instruccions enviaran tres paràmetres al RTX4100. Es va intentar crear un quart per definir el mode de funcionament amb una comanda SPI, però no s'ha aconseguit interpretar l'ordre rebuda en el RTX i per tant el desenvolupament d'aquest últim paràmetre ha quedat sense solució i no s'inclou al codi final.

Una part també important, es fer una crida "get" per obtenir temperatura i humitat dins del Header de SCDVK, perquè si no està definit el codi no compilarà al fer servir els paràmetres getTemperature() i getHumidity().

Dins de SCDVK::Main es defineix un bucle, perquè SCDVK Shield estigui contínuament enviant les dades al RTX_4100. Els primers intents en el desenvolupament van ser definir un "while(1)", però l'enviament de dades no funcionava correctament i es va decidir crear un byte "t" que forces el bucle dins d'un

"if" i d'aquesta manera solucionar els problemes que ocasionava fer-ho amb la instrucció abans esmentada. A més s'ha disposat de "delays" per establir pauses entre processos per fer-ho mes fàcil i més entenedor.
En l'arxiu zip entregat amb la memòria, és pot comprovar com són tots els passos definits al Firmware perquè SCDVK Shield perquè faci la seva part en la tramesa de dades.

Definició SPI al mòdul RTX4100

Instruccions rellevants

En la definició de la comunicació SPI al mòdul RTX però van sorgir problemes de diferents tipus, ja que al desenvolupar el protocol SPI es va detectar que en la placa SCDVK Shield hi ha un error en el disseny, degut a aquest error, RTX es comporta com "Master" també, per tant, es impossible establir connexió, ja que els dos estan en mode "Master" i una de les exigències en el bus de comunicació SPI es que un adopti la funció de "Master" i l'altre de "Slave".

Intentant trobar una solució per no deteriorar en cap component de la placa, es va realitzar una modificació en una de les llibreries del AmelieSDK, concretament en :

C:\AmelieSDK\v1.6.0.58\Projects\Amelie\Components\Drivers\DrvSpiSlave.c

Aquesta modificació fa un canvi en les línies que van de la 274 a la 276 , concretament es col·loquen aquestes instruccions:

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

Il·lustració 33- Modificació en DrvSpiSlave.c

D'aquesta manera, es soluciona un problema de disseny en la placa SCDVK Shield de Smart Citizen. Smart Citizen va facilitar un codi d'exemple per RTX, el qual s'ha modificat perquè funcioni la comunicació.

Les instruccions mes destacades que es fan servir en el disseny de la funcionalitat son:

PtDrvSpiSlaveInit(Struct pt Pt, const RosMailType* Mail, rsuint32, BaudRate);
Aquest ProtoThread inicia la comunicació SPI i en defineixen els arguments que necessitaran per formalitzar el Bus.

DrvSpiSlaveInit(rsuint32 BaudRate);
Aquesta es una funció d'inici del drive SPI

DrvSpiSlaveRxGetSize
S'obtenen el nombre de bytes que hi son al buffer

DrvSpiSlaveRx(rsuint8 *RxBufferPtr, rsuint16 RxBufferLength);
Es fa una lectura de les dades contingudes al buffer

DrvSpiSlaveRxFlush()
Buidat del buffer de RTX

RsStatusType DrvSpiSlaveTxStart(rsuint8 *TxDataPtr, rsuint16 TxDataLenght)
Comença la transmissió de dades, on el primer argument serà la dada o dades i el segon el nombre de dades.

Desenvolupament

El primer que es fa es crear un fil d'execució anomenat "Pt_softAp" per la tasca de SPI dins del mòdul RTX, amb els arguments definits en el codi. Una vegada creat es defineixen diferents aspectes d'especial rellevància com la velocitat de transmissió que s'ha definit a 50000, sent suficient per la rebuda i tramesa d'aquestes dades.

Const rsuint32 baud_rate = 50000;

El que s'ha volgut fer es que al rebre les dades del sensor via SPI del SCDVK Shield, el RTX apart de publicar-les a la web les retorni i d'aquesta manera, mitjançant el monitor sèrie del IDE de Arduino, poder veure si el procés funciona bé.

Una vegada es té l'inici s'haurà d'esperar les dades que han d'arribar del RTX, per tant es fa servir el protothread WAIT_UNTIL dins d'un bucle infinit, amb la instrucció IS_RECEIVED, que mantindrà el bloqueig en el "While" infinit fins que no rebi cap data.

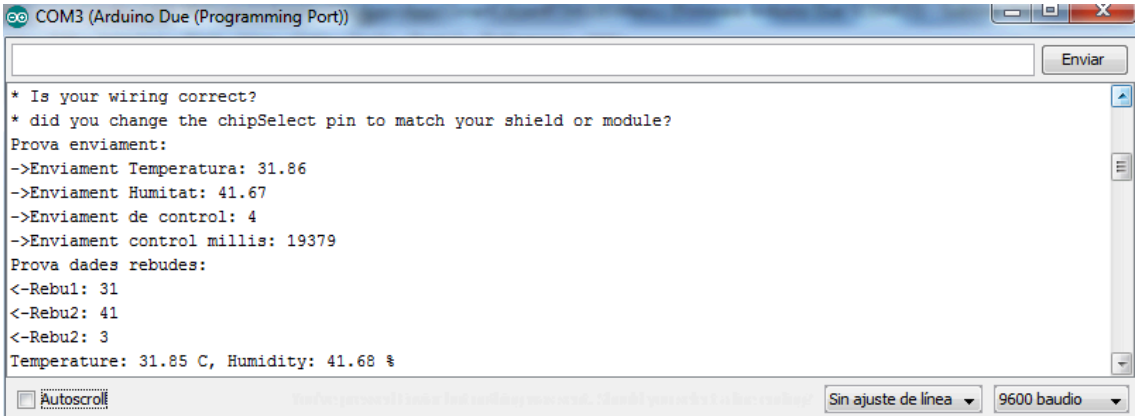
El següent pas es donar un numero de posicions a la variable creada segons el nombre de dades obtingudes gràcies al DrvSpiSlaveRxGetSize, és a dir, si en rep tres, la variable tindrà tres posicions de memòria [0], [1] i [2].

A continuació el que es fa es que en cada posició guardar la lectura que s'extrau de la funció DrvSpiSlaveRx. Una vegada està feta aquest a lectura, es guarda les dades a la NVS mitjançant la "Struct" definida en l'aparat del codi "Enumerations/Type definitions/Structs", que aquests ens serviran per portar-los a la plana web per fer la publicació de les dades rebudes.

Per finalitzar els següents passos son de comprovació de que tot funciona bé i per tant les dades rebudes es reenvien al SCDVK Shield per veure que tot funciona correctament i que RTX les rep, per fer aquest enviament tenim la funció abans esmentada "DrvSpiSlaveStart", que utilitzarà la variable amb la seva matriu de posicions i l'enviarà, tenint en compte que es tindrà que indicar quantes dades envia.

Per últim bes buidarà el buffer de RTX i tornarà a començar el bucle infinit, repetint els passos descrits en aquesta explicació.

La següent il·lustració demostra que la comunicació SPI funciona correctament, enviant les dades del sensor C10 de SCDVK Shield al mòdul RTX4100:



Il·lustració 34- Visionat de la comunicació SPI mitjançant el Monitor d'Arduino IDE.

Notes del procés

Cal indicar que per l'enviament de temperatures i humitat s'ha descobert que el sensor C10 de la SCDVK Shield envia les dades amb el format "00.00" , on els dos primers son enters i els altres decimals. Conforme s'ha avançat en el disseny i en l'estudi del funcionament de la comunicació SPI, es generà un problema amb el format de dades al comprovar que el bus de comunicació d'aquest protocol els decimals no s'envien i per tant, nomes es rep la part entera de la part enviada del Shield de Smart Citizen.

Mode SoftAp

Instruccions rellevants

PtAppWifiReset(&childPt, Mail)
Tracta de fer un reinici al Wi-Fi del mòdul RTX

AppWifiGetMacAddr();
D'aquesta funció podem obtenir la Mac Address de mòdul

AppWiFiIpv4Config()
Configuració dels aspectes de direcció i mascara

IS_RECIEVED (API_WIFI_CONNECT_IND)
El Fil d'execució esperarà fins que el Wi-Fi rebi un client que es connecti

Desenvolupament

En aquesta part es pretén realitzar la creació d'un SoftAp, tenint en compte que al principi del codi tindrem que definir les seves característiques com son, la IP , Subnet, Gateway, ESSID, SECURITY, etc.

Totes aquestes característiques estan configurades dins del SoftAp i dins del seu fil d'execució s'ha fet servir paràmetres de IP estàtics.

Un dels requisits era que no tingués cap tipus de seguretat ni password, això s'aconsegueix definint-ho en la configuració amb el següent paràmetre:

SOFT_AP_SECURITY_TYPE AWST_NONE/ NULL

Una vegada definits aquests paràmetres, el que es realitza en el codi es primer crear una SSID amb la combinació de diferents valors (SSID i MacAddr) per obtenir un nou SSID. Una vegada tenim tots els paràmetres definits, fem una crida amb PT_SPWAN per iniciar amb l'enviament d'un "Mail" el mode SoftAp.

Després es defineixen de manera estàtica la IP i la màscara, configurant els paràmetres amb la funció abans esmentada AppWifiIpv4Config. Després, s'envien les dades definides de la piscina creada dins dels paràmetres inicials en el programa del codi referent al SOFT_AP, on es marcarà l'allargada de la piscina i paràmetres com el temps de concessió.

Es van intentar fer proves per a mostrar el SSID en la web, però no s'ha aconseguit i hi ha funcions que tenen a veure amb el intent d'aquesta tasca que en un futur es pot millorar.

WebServer

Aquesta tasca es divideix en diferents parts de configuració i que ha estat de molta ajuda la COLApp de mostra "WebServer" i "AppWebConfig" per realitzar aquesta part. Degut a problemes en el disseny, no s'ha pogut establir una separació lògica entre WebServer i SoftAp i per tant, les dues van lligades com es podrà veure a continuació.

En aquesta secció al tenir tantes parts no es definiran els aspectes rellevants, sinó una explicació detallada del procés de construcció del WebServer i com s'inicia.

Generació de la pàgina Web del servidor

En primer lloc s'ha de generar la Pàgina Web fent servir el GenerateMainPage que es cridat amb el #include AppWebConfig.h. Aquesta part definirà el codi del Web, sabent que la plana no pot contenir una gran quantitat de format HTML, perquè genera al buffer problemes de capacitat, si sobrecarreguem amb moltes propietats i classes dins del codi es pot tenir l'inconvenient de que la pàgina no es mostri o nomes es mostri una part, a més s'han conèixer aspectes en la creació Web dins del codi de programació C per la correcta interpretació del codi HTML amb dos aspectes fonamentals.

1. Les classes creades en HTML amb l'etiqueta DIV, si estan marcades amb cometes es realitzarà una modificació per que s'interpreti be, un exemple :

div class="server" -> div class=\"server\"

2. Els símbols de "%" no son correctament interpretats i per tant es busquen diferents solucions, un exemple seria:

Width: 100 % -> width: 600px (tenint en compte la proporció de la plana)

Com a element important s'ha de destacar snprintf, on hi seran les dades que mostrarem dins de la web i que s'han recollit tant pel bus de comunicació entre el mòdul Wi-Fi el Shield, com per la funció per la obtenció de dades del RTX4100.

Els aspectes que es podran visionar al WebServer seran la temperatura i humitat rebuda mitjançant SPI del sensor C10 del SCDVK Shield i la temperatura interna del mòdul RTX4100. Apart, es mostrarà la data, encara que la configuració del codi no ha permès configurar una data verídica.

Un altre aspecte a incloure es el "AddHeader" també extret del AppwebConfig, on s'inclourà els elements de capçalera com son la data, abans esmentada.

Per ultimar les funcions extretes de AppWebConfig, s'utilitzarà "OnMainPage", on es definirà la resposta de la HTTP del Servidor i que retornarà RSS_SUCCES, en cas afirmatiu o RSS_NOT_SUPPORTED en el cas contrari. Cal saber que si es afirmativa, es fa servir la funció "SendApiHttpServerSendResponeReq" que tracta d'enviar una instància creant i enviant la pàgina Web on s'inclouran els arguments per el càlcul de la mida i trucades a "AddHeader" i "GenerateMainPage" que retornaran el resultat generant la capçalera del Web i la generació de la pàgina principal.

Una vegada configurat el apartat HTTP del Servidor Web, la comunicació SPI i la correcta obtenció de les dades mitjançant aquest bus, podrem definir el últim Thread (connect_wifi) que englobarà tots els processos descrits amb anterioritat i seguint un ordre d'execució lògic i ordenat.

Creació esglaonada dels diferents apartats

La creació de tot el procés serà de manera esglaonada, sent el millor mètode trobat perquè la generació del servidor, plana principal del servidor i obtenció de dades sigui i es faci de manera correcta, perquè durant les diverses proves realitzades es comprova que les crides als "Threads" i funcions s'han de fer de manera ordenada i esglaonada.

Per començar es crea el fil d'execució anomenat "connect_wifi" on es definiran el childPt i UpdateSensor, per l'actualització de dades del sensor i apart la creació d'una variable per a guardar el SSID encara que no es mostri de manera correcta al servidor Web, tot i així, es marca la pauta per a implementacions futures.

A continuació, es seguirà un ordre lògic d'execució per tal de generar totes les funcions de manera correcta, llavors, partint d'aquesta premissa, el inici de tot és fer la crida amb "PT_SPAWN" per iniciar el "Thread" de "Pt_softAP" per a crear el AP. Una vegada executada aquesta tasca, es realitza un altre crida per obtenir la temperatura interna del mòdul RTX4100, per a mostrar-la al servidor Web de manera informativa utilitzant la funció PtDrvIntTempMesure. Una vegada obtinguda aquesta dada del mòdul es fan crides per la obtenció del SSID "AppWifiGetSsid".

De moment, està iniciat el SoftAP i el drive de Temperatura del RTX, a continuació es crearà el servidor (SendApiHttpServerInitReq) i esperarà fins a rebre resposta (PT_WAIT_UNTIL). Una vegada realitzada aquesta acció, es comprovarà si realment

està iniciat amb el que s'ha descrit en "OnMainPage" amb el "RSS_SUCCES" mitjançant un "if".

Si s'ha complert la instrucció, la creació del servidor haurà estat satisfactòria i funcionarà de manera correcta. El següent pas serà la creació de la pàgina Web fent una instància amb "SendApiHttpServerAddResourceReq" i per tant, farà la creació de la pàgina que farem servir per a posar-li el contingut de "GenerateMainPage".

Si fem una aturada en el procés, tenim que de manera esglaonada hem fet crides i per tant la creació dels següents elements:

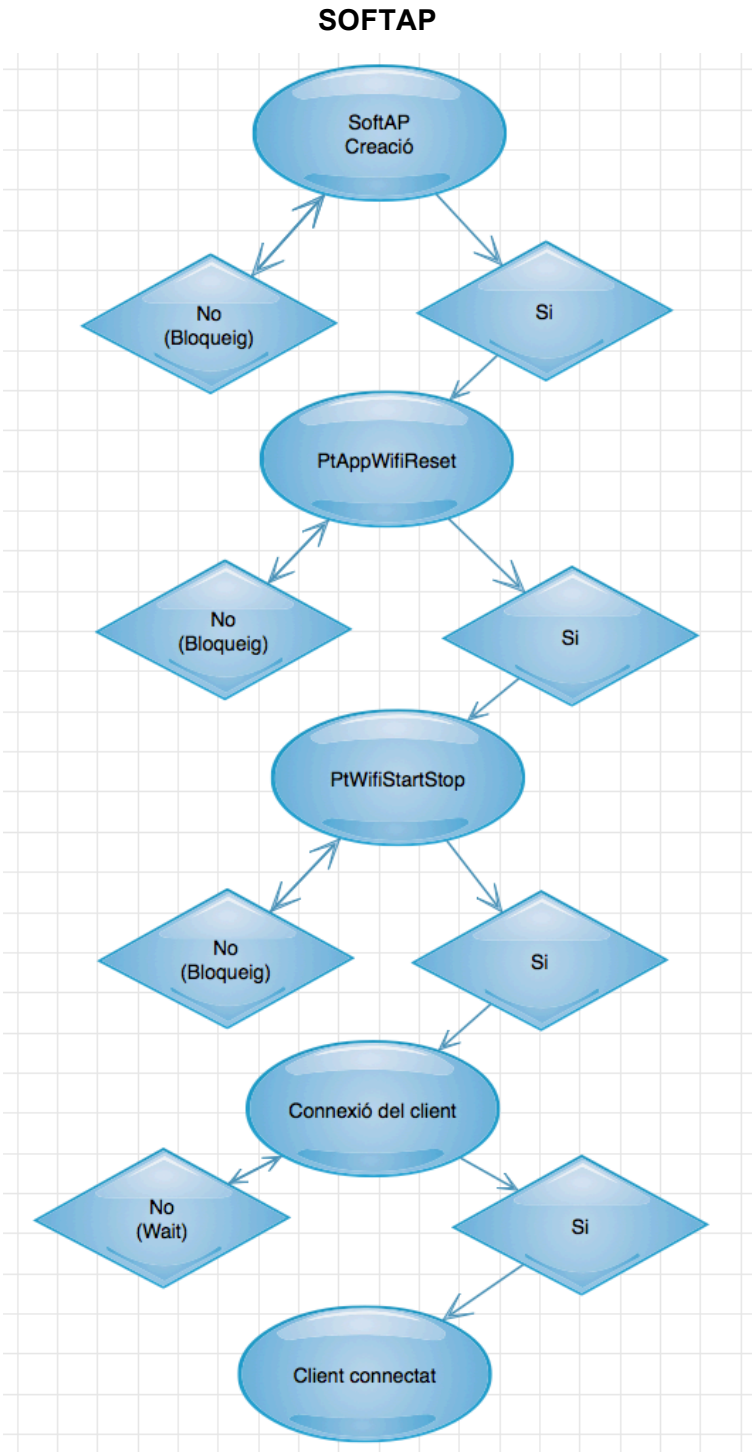
1. Creació SoftAp
2. Instancia per comprovar temperatura UpdateSensor
3. Trucada per l'inici del WebServer
4. Espera fins l'inici d'aquest
5. Comprovació de l'inici del servidor
6. Creació de la Plana Web on entrarà GenerateMainPage

Veien tot el que s'ha fet, només queda fer una crida al Thread SPI "Pt_spi" per obtenir les dades de temperatura i humitat que farem servir en GenerateMainPage, perquè ja estarà creada dins del procés esglaonat.

D'aquesta manera, s'ha creat un SoftAP que donarà el punt d'accés a la plataforma del servidor Web, on es generarà una plana que mostrarà les dades de temperatura i humitat del sensor de temperatura C10 situat en la SCDVK Shield mitjançant SPI i la temperatura interna del RTX, juntament amb la data, encara que de manera incorrecta.

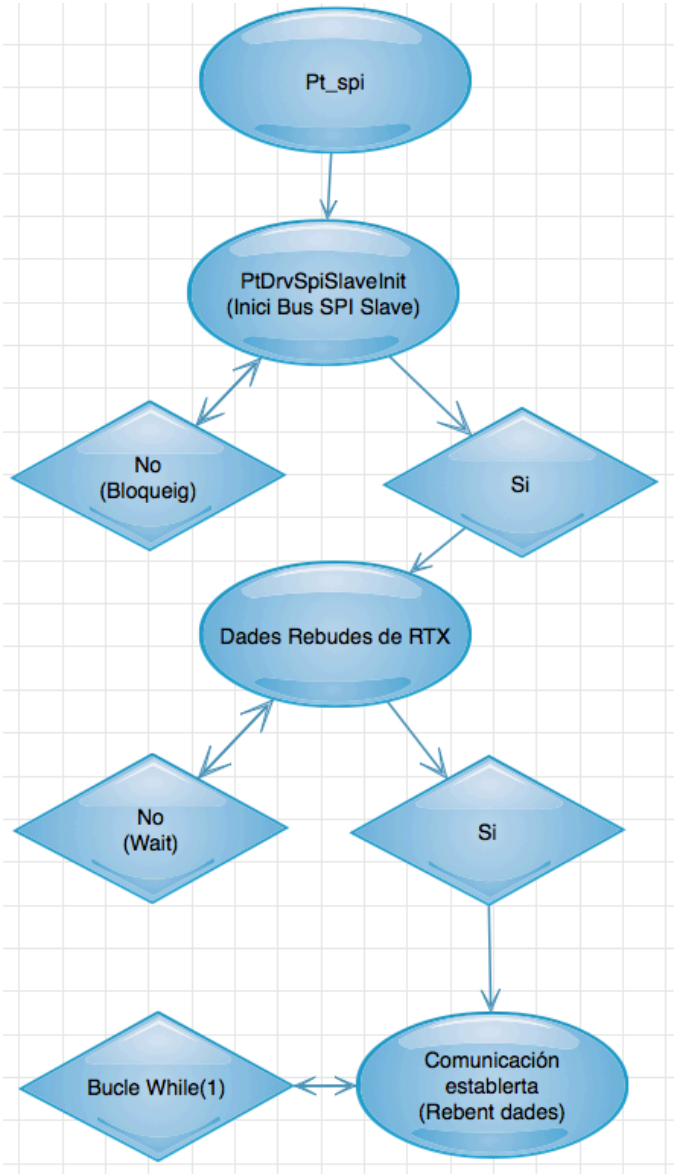
Diagrama de flux

En aquesta secció es presenten els diagrames de flux dels Threads programats:



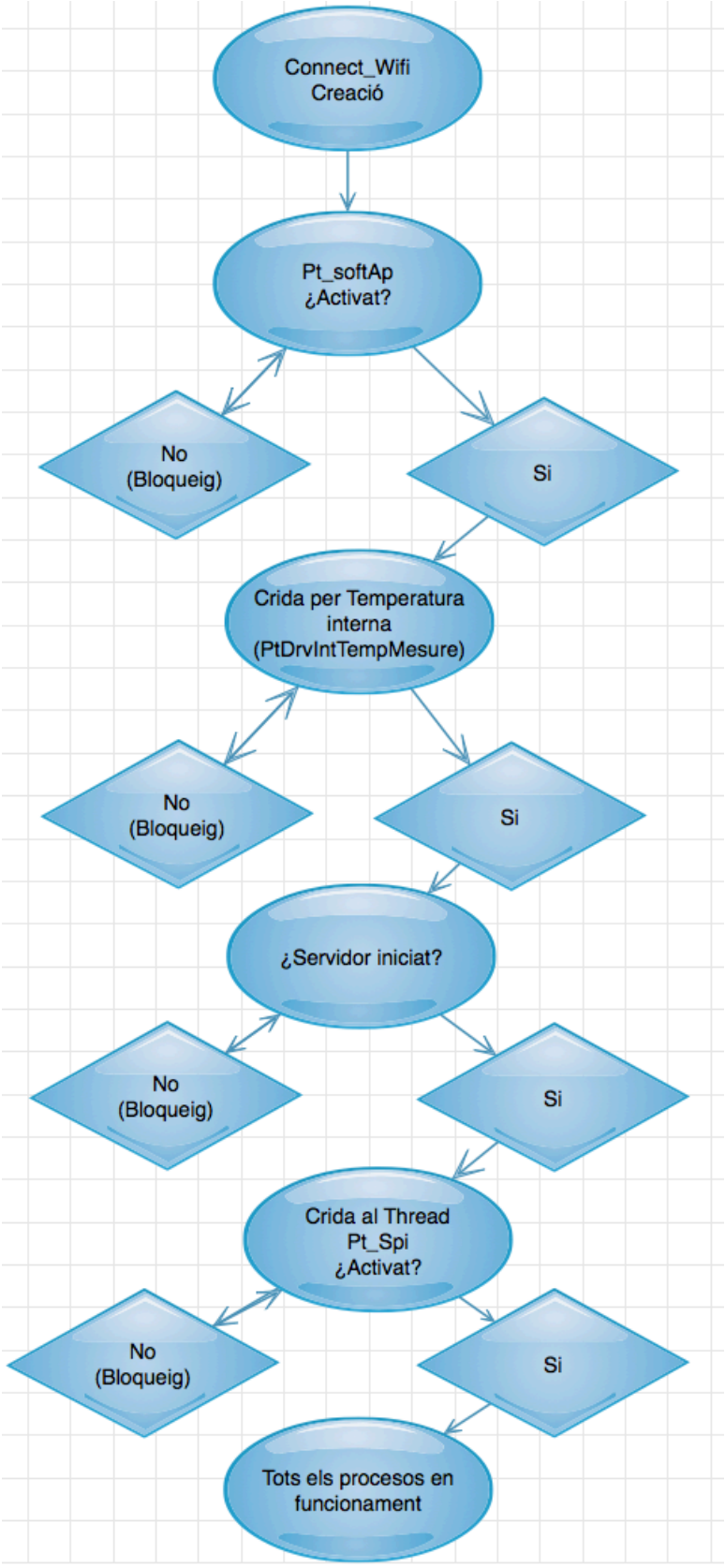
Il·lustració 35-Diagrama de flux SOFTAP.

SPI



Il·lustració 36-Diagrama de flux SPI.

Unió de tots els processos des del procés “Pt_connect_wifi”



Il·lustració 37-Diagrama de flux de tots els processos.

Resultats Finals

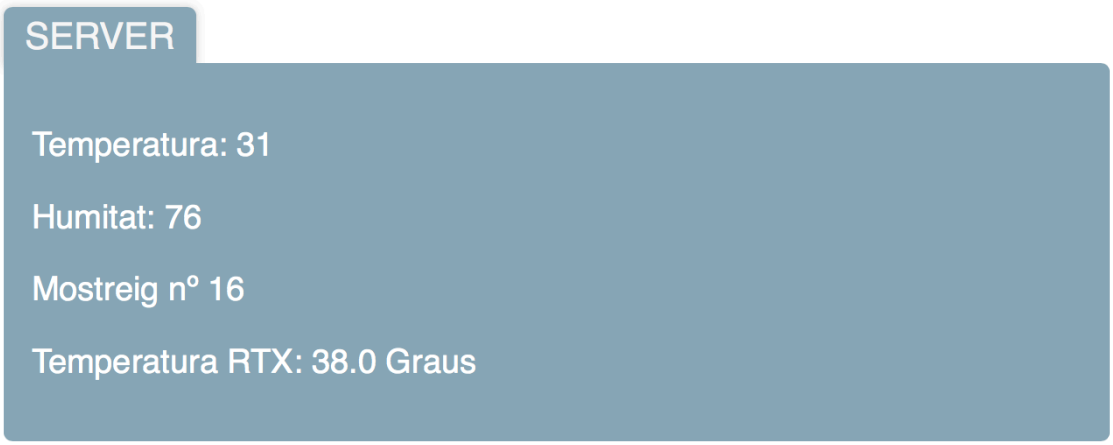
En aquest procés de construcció s'ha dissenyat de manera correcta el punt d'acces per a la connexió Wi-Fi mitjançant un dispositiu mòbil. Connectat al punt d'acces, es genera correctament el servidor que es qui guardarà la Web de control.

Una vegada s'han creat els processos anteriors, es genera la Web que contindrà els aspectes informatius comentats en el projecte i per finalitzar es crida al SPI per obtenir les dades de temperatura, humitat i mostreig assignar-los als paràmetres definits dins de la Struct generada amb el sobrenom "app_data". A més s'inclou la temperatura interna del mòdul RTX4100, per a tenir informació del mòdul.

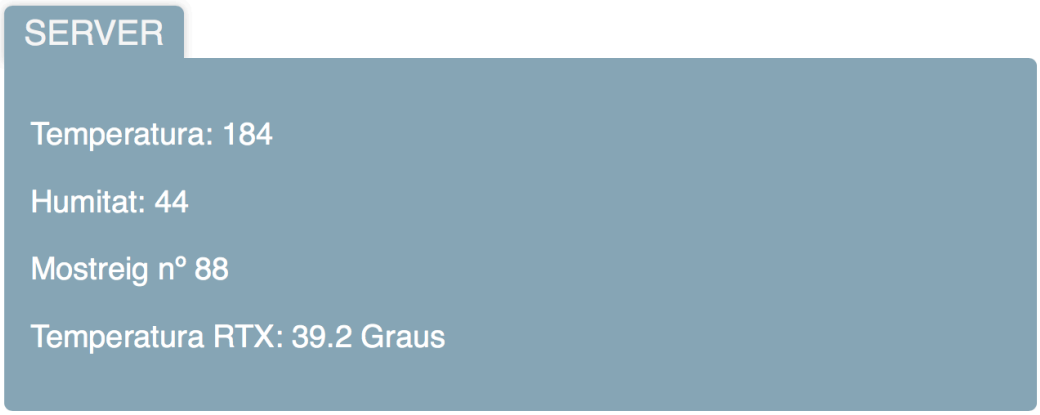
Els resultats aconseguits es mostren en la següent imatge, on es pot veure que la humitat no mostra bé el resultat degut a algun problema amb la Web. La temperatura a vegades també dona errors, encara que d'altres funciona bé.

Es vol recalcar, que la comunicació SPI s'ha comprovat varies vegades i s'han fet diverses proves en la transferència de "Master" a "Slave" i els problemes generats no sorgeixen d'aquesta part. Destacaria algun problema amb al n=snprintf.

A continuació es mostra una il·lustració del resultat final:



Il·lustració 38- Captura de l'entorn Web amb les mesures de temperatures donades correctament.



Il·lustració 39-Captura de l'entorn Web amb la mesura de temperatura errònia.

INFORMACIÓ I CONCLUSIÓ

En el present projecte, s'han establert les bases necessàries per a continuar amb el desenvolupament de les funcions demanades per Smart Citizen, de igual manera es defineixen també els paràmetres per incorporar més contingut a la pàgina web mitjançant la incursió de etiquetes identificatives "id".

Els problemes generats durant aquest projecte han sigut:

- Entrega de Kit amb poc marge d'actuació.
- Problemes amb el mòdul RTX4100 al quedar-se sense resposta i trobar-li una solució per que torni a funcionar.
- El Firmware carregat a Arduino Due ha donat alguns problemes, quedant el kit inservible i cercant respostes per solucionar els problemes.
- A vegades les respostes de Smart Citizen han estat amb dies de diferencia.

Degut a aquests problemes i la realització de totes les funcionalitats ha sigut impossible d'implementar i a més amb la correcció amb la espera de correcció del disseny per que la comunicació SPI funciones correctament.

Bibliografia

Noms de les Fonts d'informació, enllaç i data de consulta:

1. Equip Smart Citizen, "The Smart Citizen Platform". www.smartcitizen.me, març del 2015.

2. Colom, M. A*nalysis, 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*, Març del 2015*

3. Wiring Platform Web, Small Team of Volunteers. http://wiring.org.co , febrer del 2015.

4. Processing Platform Web, Small Team of Volunteers, http://processing.org, març del 2015

5. SCK CoLApp for RTX41000. https://github.com/fablabbcn/SmartCitizenRTX4100, abril del 2015

6. Guia d'usuari Smart Citizen. https://github.com/fablabbcn/SmartCitizenDVK/blob/master/Documentation/RTX4100/RTX4100_User_Guide_Module_Evaluation_UG1.pdf , març del 2015

7. Arduino Platform Web. www.arduino.cc/es/Tutorial/Sketch, març del 2015.

8. Arduino Platform Web. http://arduino.cc/en/Hacking/LibraryTutorial, març del 2015

9. Smart Citizen Platform, SCK COLApp for RTX41000. https://github.com/fablabbcn/SmartCitizenRTX4100 , marc del 2015

10. Microchip, mòdul RN313 documentació. http://ww1.microchip.com/downloads/en/DeviceDoc/rn-131-ds-v3.2r.pdf, març del 2015

11. Smart Citizen Platform, Firmware SCK 1.1. https://github.com/fablabbcn/Smart-Citizen-Kit/, abril del 2015.

12. Atmel Portal Web, Atmel Cortex M0+ SAMD21 datasheet. http://www.atmel.com/Images/Atmel-42181-SAM-D21_Datasheet.pdf , març 2015.

13. RTX A/S API Specification Amelie Platform.
www.rtx.dk/LPW/RTX4100, març del 2015.

14. Exemples: http://notepad-plus-plus.org o http://sublime.com
Descarregar de https://launchpad.net/gcc-arm-embedded/+milestone/4.7-2013-q3-update

15. RTX Platform Web, Documentation RTX4100. RTX4100_Application_Note_SoftAP_AN8.pdf , març del 2015.

16. Llenguatges i estàndars Web, UOC.
http://www.uoc.cat, abril del 2015.