Hack the World

Carles Mateos Pi

Resum– Aquest projecte ha consistit a crear un sistema de posicionament indoor 3D mitjançant diversos Arduino que actuen com a nodes fixes i una Raspberry Pi que és l'element a posicionar. Per dur-ho a terme, inicialment s'ha intentat crear un sistema el qual, a través de la intensitat de senyal rebuda entre transmissors de radiofreqüència, es pugui calcular la distància de la Raspberry a cadascun dels Arduino, amb la finalitat de poder triangularitzar l'element dins duna sala. Posteriorment, en vista a certs problemes amb el hardware transmissor de ràdio, s'ha decidit crear un sistema basat en enviament i recepció de so per poder calcular distàncies. Finalment perquè l'usuari d'aquest sistema pugui veure les dades, s'ha creat un servidor web amb el Backend format per NodeJS i el Frontend creat en ReactJS.

Paraules clau– IPS, posicionament indoor, Arduino, MySensors, Raspberry Pi, comunicació NRF, sistema basat en so, NodeJS, ReactJS.

Abstract– This project has been to create a 3D indoor positioning system using several Arduino which act as fixed nodes and a Raspberry Pi which is the element to position. In order to carry it out, initially, it was attempted to create a system which, through the intensity of signal received between radio frequency transmitters, could calculate the distance of the Raspberry to each of the Arduino, in order to be able to triangulate the element within a room. Later, in view of certain problems with the radio transmitter hardware, it has been decided to create a system based on sending and receiving sound in order to calculate distances. Finally, so that the user of this system can see the data, a web server has been created with the Backend formed by NodeJS and the Frontend created in ReactJS.

Keywords– IPS, indoor positioning, Arduino, MySensors, Raspberry Pi, NRF communication, sound based system, NodeJS, ReactJS.

1 Introducció

DURANT la dècada dels 70 es va crear el Sistema de Posicionament Global, que coneixem avui en dia com a GPS. Aquest sistema permet determinar la posició d'un element a escala terrestre mitjançant uns satèl·lits repartits al voltant del globus terraqüi. El sistema GPS consta de 24 satèl·lits que orbiten a uns 20.180 km d'altura amb trajectòries fortament sincronitzades per tal de cobrir la totalitat de la superfície terrestre.

Com podem veure, s'han necessitat una gran quantitat de recursos i esforç per tal d'aconseguir el nostre sistema de posicionament global actual. Tot i els esforços aquest sistema té un marge d'error d'uns certs metres 1-3 m.

En els últims anys s'ha començat a popularitzar una altra mena de sistema de posicionament, el Sistema de Posicionament a Interiors o de l'anglès Indoor System Positioning (IPS). Com diu el nom, aquest sistema serveix per localitzar elements o entitats en espais més reduïts, com podrien ser: centres comercials, magatzems, pàrquings, campus universitaris, entre molts d'altres. Existeixen múltiples tècniques per crear sistemes IPS, on les més conegudes o usuals són aquelles que fan ús d'ones de ràdio, com per exemple sistemes basats en tecnologia Wi-Fi o Bluetooth. D'altra banda tenim addicionalment sistemes menys tradicionals com ara sistemes basats en marcadors visuals o inclús sistemes basats en el magnetisme dels edificis i dels elements magnètics a posicionar.

La motivació d'aquest projecte és la d'intentar construir un sistema de posicionament indoor 3D de baix cost, format per un sistema basat en intensitat de senyal de les ones electromagnètiques que viatjaran des de nodes fixes posicionats en una sala fins a l'element a posicionar en l'espai. Els elements fixes són les plaques Arduino Nano, les quals tenen una mida reduïda i ofereixen multitud d'usos en el món de l'automatització domèstica, per altra banda l'element a ser posicionat és en aquest cas una Raspberry Pi 3 model B+.

Per a la comunicació entre els nodes s'ha fet servir el framework anomenat MySensors. Aquest framework és open-

• E-mail de contacte: carles.mateos@e-campus.uab.cat
• Menció realitzada: Enginyeria de Tecnologies de la Informació
• Treball tutoritzat per: Oriol Parera Fiestas (dEIC)
• Curs 2018/19

source i està dedicat a facilitar les comunicacions entre diferents elements hardware fent servir unes llibreries escrites en llenguatge C, les quals implementen un protocol de comunicació compatible amb diferents hardwares transmissors de ràdio.

Durant l'elaboració del projecte es va trobar un problema relacionat amb el hardware de l'antena de comunicació model NRF24L01, la qual no permet de cap manera saber la intensitat de senyal entre dos transmissors. A causa d'aquest problema es va replantejar el projecte per tal de trobar una alternativa que fos capaç d'implementar aquest sistema de posicionament. L'opció que finalment es va escollir va ser la de crear un sistema basat en emissió i recepció de so, on gràcies a la velocitat del so podríem arribar a determinar la distància entre dos elements.

El document llavors queda estructurat de la següent manera, a la secció 2 trobem l'estat de l'art on es contextualitza aquest projecte, a la secció 3 trobem els objectius de la primera fase del projecte, seguidament a la secció 4 trobem la metodologia i planificació. Trobem tota l'etapa de desenvolupament inicial a la secció 5, la redefinició dels requisits del projecte a la secció 6 i l'etapa final del projecte a la secció 7. Finalment podem veure els resultats obtinguts a la secció 8, juntament amb les línies de futur que li augura a aquest projecte, ja per acabar trobem les conclusions a la secció 10 i finalment una secció d'agraïments i de referències bibliogràfiques.

2 ESTAT DE L'ART

En l'actualitat la tecnologia i el món de la informàtica està cada vegada més integrat en el nostre dia a dia. És per això que la domòtica s'està fent més popular en les últimes dècades i podem trobar multitud d'objectes domotitzats que abans no ho eren, com ara televisors amb connexió a Internet que permeten gaudir d'infinitat de continguts multimèdia o neveres intel·ligents que són capaces de determinar l'estat dels aliments, entre moltes altres coses.

Actualment els costos d'aquests productes són relativament elevats, en comparació amb els mateixos productes tradicionals. Tenim per exemple l'empresa Localino, la qual ofereix el seu producte de sistema de posicionament en 3D a qui ho vulgui adquirir[1], això sí, a un preu que no tothom es pot permetre (al voltant de 500€). És en aquest punt on les persones amb certs coneixements en el món de la informàtica van començar a crear els components necessaris i desenvolupar el software adient per tal de crear els seus propis sistemes domòtics.

Gràcies a aquests avenços podrem dur a terme aquest projecte de posicionament indoor tan ambiciós a un preu molt reduït (menys de 100€) a causa dels baixos costos dels materials i el software obert gratuït.

3 OBJECTIUS

Una vegada analitzat el context en el qual es desenvoluparà aquest projecte s'han determinat una sèrie d'objectius necessaris perquè el producte final sigui un èxit.

- Crear infraestructura hardware per a la comunicació.

  Per fer-ho, primerament s'han adquirit tots els elements hardware necessaris i posteriorment s'ha procedit a la interconnexió d'aquests.

- Establir comunicació entre dos nodes.

  Aquest objectiu ha consistit en primer lloc, a configurar el Gateway[2] de MySensors dins de la Raspberry i el software dels elements. Per a l'enviament de missatges per part de la Raspberry s'ha codificat un programa en Java que obre un socket contra el Gateway local, creant i enviant un missatge estructurat[3] de manera determinada cap al socket. En segon lloc, per part de l'Arduino s'ha inclòs la llibreria de MySensors i s'han definit mètodes que són cridats quan l'Arduino rep un missatge, per tal de retornar una resposta.

- Transformar les intensitats de senyals en distància.

  Per assolir l'objectiu, s'ha de crear una fórmula matemàtica a partir de l'experimentació, on a través de la intensitat de senyal entre Raspberry i Arduino, es pugui determinar la distància que els separa.

- Tractament i processament de dades.

  La finalitat d'emmagatzemar i tractar les dades ,es la d'obtenir informació útil. El fet d'emmagatzemar les dades ens serveix per mantenir un registre de les diferents intensitats rebudes per part dels Arduino, així com guardar l'històric de posicions on ha estat la Raspberry en l'espai. Gràcies a això podrem tractar les dades i obtenir per exemple la distància recorreguda durant cert temps o en quina posició s'ha quedat més estona, entre molts d'altres exemples.

- Accedir remotament al sistema.

  Aquest objectiu consisteix a crear un servidor web el qual sigui accessible via Internet, per poder per exemple, veure des del nostre telèfon mòbil a la feina on es troba cert objecte a la nostra casa en aquell moment.

  Per fer això s'ha configurat l'encaminador de casa, obrint el port de la Raspberry on es troba el servidor web. Això és possible i té sentit gràcies a la tecnologia Wi-Fi que incorpora el model de Raspberry 3 B+.

4 METODOLOGIA I PLANIFICACIÓ

Una vegada definit els objectius del projecte s'ha procedit a definir i elaborar la metodologia i la planificació d'aquest.

La metodologia d'aquest projecte s'ha basat en gran part del temps a treballar de forma iterativa. Inicialment les feines a fer eren purament lineals (investigar components, adquirir-los, muntar el hardware...) però a mesura que anava avançant els dies es va transformar en un procés iteratiu. Això és així donat que aquest projecte té un pes enorme en l'experimentació, cosa que comporta un temps molt gran dedicat a provar i a fallar constantment, aprenentatge per prova-error.

Durant el desenvolupament del projecte es van trobar certs inconvenients amb el hardware d'antena que van comportar adquirir nous components i canviar l'arquitectura hardware i software. L'impacte que va tenir aquest canvi no va ser tan gran degut, en part, a la bona metodologia que

s'estava seguint, ja que en ser iterativa permet una flexibilitat davant de nous requisits i evita que el projecte sigui estàtic.

Pel que fa a la planificació, s'han definit diverses feines ordenades de forma prioritària per al correcte desenvolupament del treball:

1. Identificació dels components hardware necessaris.
   1.1. Anàlisi i recollida de requeriments.
   Aquesta fase consisteix a recollir els diferents requeriments del nostre projecte.
   1.2. Investigació i cerca a Internet.
   Inicialment s'ha hagut de cercar a les diferents fonts d'informació, que principalment ha sigut Internet, sobre diferents projectes similars que facin ús d'aquest tipus de components. Aquesta cerca ha permès juntament amb els requeriments del projecte, definir quines versions dels components són compatibles per adquirir per a l'elaboració del projecte.
   1.3. Elaboració de la llista de components.
   Una vegada s'ha efectuat la recollida de requeriments i la investigació dels components s'ha fet un llistat per comprar el necessari.
2. Creació de la infraestructura de xarxa per a la comunicació.
   2.1. Muntatge dels elements principals.
   Consisteix a muntar els components hardware que queden fixes en l'espai on es vol determinar la posició d'un element, això ve a ser els Arduino juntament amb els components d'antena. De la mateixa manera tenim doncs la Raspberry amb un altre element d'antena que actuen juntament com a element mòbil a ser posicionat dins d'aquest espai tridimensional.
   2.2. Preparació Software dels elements.
   Consisteix a posar a punt el software, tant els Arduino com la Raspberry, instal·lant la llibreria de mysensors per fer de Gateway (software encarregat de comunicar-se amb l'element de ràdio i enviar/rebre dades als/dels altres Arduino).
3. Procés d'investigació per intercanviar dades d'intensitats de senyals.
   Això significa portar a terme un procés d'investigació, el qual s'ha d'enfocar en saber de quina manera es pot determinar el nivell d'intensitat produït per un Arduino i captat a la Raspberry.
4. Codificació del Controller i Arduino
   4.1. Codificació del Controller.
   Consisteix a crear un software el qual ha de ser capaç de comunicar-se amb el Gateway de la Raspberry, per tal de rebre/enviar missatges i poder processar la informació que contenen.
   4.2. Codificació d'Arduino.
   Es tracta de crear un sketch (programa d'Arduino) el qual ha de tenir la capacitat d'enviar/rebre missatges cap al Gateway allotjat a la Raspberry.
5. Experimentació en diferents entorns.
   Consisteix a provar constantment la funcionalitat actual del sistema. Això es fa de manera iterativa per tal de treure conclusions a mesura que es va avançant i fent canvis, tant en el hardware com en el software.
6. Tractament i processament de dades.
   S'ha de muntar a la Raspberry una base de dades. Aquesta base de dades ens serveix per mantenir un registre de les diferents intensitats rebudes per part dels Arduino i per mantenir un registre de les posicions de la Raspberry.
7. Connectivitat Wi-Fi del Controller per accedir remotament a les dades.
   Consisteix a configurar l'encaminador de casa amb l'objectiu de poder accedir remotament als arxius, servidor web, al Controller i altres, a través d'Internet. S'ha de connectar la Raspberry a l'encaminador i obrir el port des d'aquest últim.

5 Desenvolupament Inicial

En aquest apartat es troba tota la feina que s'ha fet durant els primers mesos i on s'explica detalladament com s'ha anat desenvolupant la part inicial del projecte.

5.1 Arquitectura Hardware

Tal com s'ha explicat anteriorment, aquest sistema de posicionament consta de diversos components Arduino que actuen com a nodes fixes i s'encarreguen de delimitar la sala real que volem mapejar. Cada un dels Arduino està connectat de manera específica amb uns components d'antena de radiofreqüència (model NRF24L01) per poder comunicar-se amb la Raspberry. La Raspberry conseqüentment també té connectat el mateix component d'antena. Addicionalment, per fer tasques remotament i accedir a les dades, s'ha connectat la Raspberry a un encaminador amb connexió a Internet.

Inicialment per treballar amb la Raspberry es va instal·lar un monitor, un teclat i un ratolí per dur a terme les primeres configuracions, com la instal·lació del sistema operatiu Raspbian o més endavant establir la connexió cap a l'encaminador via Wi-fi. Finalment aquests elements connectats a la Raspberry han estat retirats, ja que gràcies a la connexió en remot via SSH es poden fer les tasques que resten.

A la Fig.1 podem veure el resultat de la infraestructura inicial.

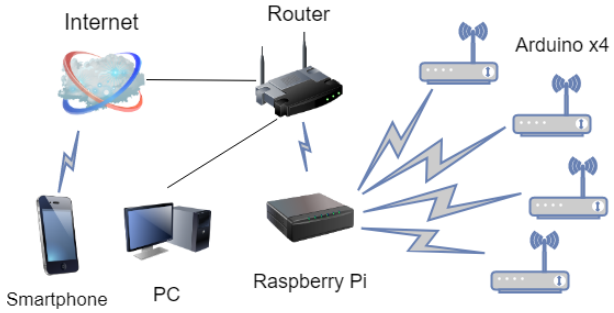

Fig. 1: Arquitectura inicial

5.2 Software dels elements

Primerament, una vegada instal·lat el sistema operatiu Raspbian a la Raspberry, s'ha procedit a instal·lar el Gateway de MySensors. Per fer això s'ha seguit la guia que trobem al site web oficial. Aquest software s'encarrega de controlar el protocol de missatges que fan servir tots els components Arduino amb la llibreria de MySensors, així com serveix com a intermediari entre el controlador i cadascun dels nodes. A l'executar el Gateway aquest deixa obert un terminal on podem observar tots i cadascuns dels missatges que passen pel transmissor de la Raspberry, tenint a més disponible la possibilitat veure com s'autoconfigura o busca certs elements hardware enviant missatges de tipus ping quan aquests deixen de respondre al cap d'un temps. En addició, el Gateway obre un port a la Raspberry per poder parlar-hi a través d'un socket si així es considera.

Una vegada dut a terme les configuracions a la Raspberry, s'ha continuat desenvolupant una primera versió d'un sketch d'Arduino, on la funcionalitat inicial ha consistit a incloure la llibreria C de MySensors, configurant els paràmetres perquè els missatges poguessin ser enviats i rebuts pel transmissor. Una vegada ajustats els paràmetres inicials s'ha procedit a enviar un missatge simple cada 3 segons per veure si la connexió realment funcionava i es veia reflectida al terminal de la Raspberry corresponent al Gateway.

En veure que efectivament els missatges arriben, s'ha començat a pensar de quina forma es pot extreu-re aquesta informació i quina és la forma per la qual podem enviar missatges cap als nostres Arduino. Com s'ha comentat anteriorment, el Gateway deixa obert un port a la Raspberry que per defecte és el 5003, això permet que es pugui obrir un socket contra aquest port. Arran d'això s'ha començat a desenvolupar un programa escrit en Java el qual finalment ha sigut el Controller d'aquest sistema. Per llegir dades provinents del Gateway simplement s'ha obert un socket contra el port esmenat amb anterioritat des d'un ordinador connectat a la mateixa xarxa que la Raspberry, on s'ha pogut observar uns missatges en un format molt concret:

node-id;child-sensor-id;command;ack;type;payload

- *node-id*: correspon al id de l'Arduino.
- *child-sensor-id*: correspon al id d'un dels sensors adherits a l'Arduino
- *command*: aquest paràmetre identifica el tipus de comanda del missatge. Tenim per exemple missatges de tiput Set o de tipus Request.
- *ack*: Indica si el missatge conté un ack, en funció del seu valor 0 ó 1 i si és un missatge d'anada o tornada.
- *type*: indica el tipus de missatge i significa diferents coses en funció del paràmetre *command*.
- *payload*: És la part del missatge en si. En aquest projecte correspon al strings que s'intercanvien els nodes.

Tots els elements del missatge a excepció del *payload* corresponen a valors enters positius. Per a més informació es recomana visitar el site web on trobem l'API de MySensors[3].

Un cop après quina és la manera en què s'han de crear els missatges per tal que aquests siguin rebuts correctament, s'ha creat un missatge específic: 2;11;3;1;24;contesta . Aquest missatge va dirigit al sensor 11 del node 2 amb una comanda de tipus *internal* i efecte ping el qual s'especifica que se'n vol l'ack i a més porta una string anomenada çontesta". A continuació s'ha creat al sketch d'un dels nodes, un mètode que quan el transmissor rep un missatge dirigit a ell, l'Arduino contesta amb un missatge que conté el payload "hola".

Arribats a aquest punt ja tenim per una banda, poder rebre missatges i tractar-los des d'un programa escrit en Java i per altra banda poder enviar missatges dirigits específicament a un node en concret.

5.3 Investigació

En aquesta etapa del projecte es comença un procés d'investigació el qual té com a objectiu trobar i identificar de quina forma podem determinar la intensitat de senyal que es dóna entre dos elements transmissors de ràdio.

Investigant a diverses fonts d'informació, essencialment Internet, s'ha arribat a la conclusió que el hardware de ràdio adquirit per als nodes i la Raspberry, no té cap mode de determinar quina és la intensitat de senyal que hi ha entre dos transmissors.

Cercant a la web oficial[4] on es troba l'API d'aquest hardware, es pot veure que únicament tenim un registre que indica amb valors 1 ó 0 si l'emissor respecte al receptor té una intensitat de senyal major o menor a -64dBm. Aquest registre és concretament el registre CD que es troba en posició de memòria 09 en hexadecimal segons la documentació del hardware[5].

Per afrontar aquest inconvenient s'han buscat alternatives, una de les quals ha sigut intentar treure el Round Trip Time (RTT) quan es fa un ping entre la Raspberry i un dels Arduino. S'han aprofitat tots els coneixements apresos amb anterioritat respecte a la creació i enviament de missatges, afegint puntualment al codi en Java la lògica de comptar el temps. Consisteix a establir unes marques de temps just el moment abans d'enviar el missatge i una altra just al moment en què rebem la resposta, on finalment en fer la resta obtenim el RTT. Desafortunadament aquest mètode no serveix per a aquest projecte, ja que la velocitat a la qual viatgen les ones de ràdio electromagnètiques és molt alta, concretament viatgen a la velocitat de la llum (300.000 km/s), fent que una variació de només 1 ms impliqui un desplaçament teòric de 300 km. Aquesta opció s'ha descartat rotundament.

Una altra alternativa que ha sorgit durant el procés d'investigació ha sigut la de pensar si hi havia alguna altre tipus d'ona que viatgés més lentament, cosa que possiblement ens podria servir per a complir els objectius. Efectivament, tenim les ones de so les quals viatgen només a 343 m/s aproximadament i que en 1 ms recorren només 34 centímetres (1).

$$\triangle x = \triangle v \cdot \triangle t = 343,2m/s \cdot 1ms = 0,34m \quad (1)$$

Donada la viabilitat teòrica de l'alternativa, s'ha procedit a redefinir els requisits del projecte tenint en compte les noves necessitats, així com els nous objectius.

6 Actualització de Requisits

El projecte s'ha d'adaptar a la nova alternativa que ha estat escollida: un IPS basat en emissió i recepció de so. Per tal que la continuació del projecte i el final d'aquest sigui un èxit s'han redefinit els següents objectius:

- Crear infraestructura hardware per a la comunicació.

  Haurem d'actualitzar la nostra infraestructura incorporant els nous elements necessaris per a la comunicació entre els nodes. Aquests elements seran: 1 emissor de so, que anirà connectat a la Raspberry i 4 receptors de so connectat als nodes, evidentment 1 receptor per cada Arduino.

- Establir comunicació entre dos nodes.

  A partir d'ara la comunicació d'anada (Raspberry ⇒ Arduino) es farà a través d'ones de so i la comunicació de tornada (Arduino ⇒ Raspberry) es farà via el sistema actual de radiofreqüència. S'han de buscar doncs mètodes per poder reproduir sons des del Controller i mètodes per captar sons des dels Arduino.

- Transformar el RTT en distància.

  Substitueix l'objectiu inicial de determinar distàncies a través d'intensitats de senyal, té com a finalitat transformar una vegada obtingut el temps que triga un so a ser captat per un Arduino a unitats de distància. S'hauran d'aplicar fórmules matemàtiques així com tècniques estadístiques com mitjanes i desviacions.

7 Desenvolupament Final

En aquest apartat es troba tota la feina que s'ha fet durant els últims mesos i on s'explica detalladament com s'ha anat desenvolupant la part final del projecte.

7.1 Arquitectura Hardware

Tota la infraestructura que tenim fins al moment ens és útil i només farà falta uns determinats components per tenir-la llesta definitivament.

Després d'investigar a Internet diversos projectes relacionats amb la domòtica i MySensors, s'ha vist que gran part dels projectes que fan servir micròfons utilitzen components molt similars. Aquests micròfons tenen 4 pins, 1 dedicat a l'alimentació (5V), 1 per la presa de terra (GND), 1 de sortida digital i finalment 1 de sortida analògica. La majoria tenen un element que permet graduar l'offset del condensador electret [6]. Finalment, perquè el projecte es pogués dur a terme durant el temps assignat, s'ha adquirit a través d'Amazon 4 components detectors de so com els que s'acaben de descriure. Realment d'aquest hardware només ens interessarà la sortida analògica, ja que dóna més oportunitat a ser més precís a l'hora del calibrat.

Per a l'emissió de so, inicialment s'ha fet servir uns altaveus grans que donen suport a un ordinador de sobretaula, però finalment, tenint en compte la prova tècnica que es vol dur a terme el dia de la defensa, farem servir un altaveu portàtil amb entrada auxiliar de so.

A la Fig.2 podem veure el resultat de la infraestructura final.

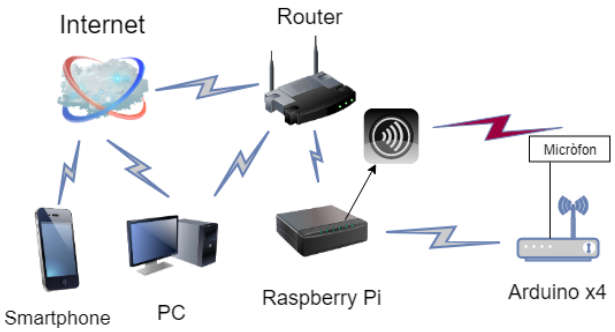

Fig. 2: Arquitectura inicial

7.2 Software dels elements

Un cop tenim llesta tota la infraestructura hardware final, comencem a incloure els nous desenvolupaments que seran necessaris en tots i cadascun dels softwares creats fins al moment. Això inclou el Controller escrit en Java i els sketchs dels Arduino. Finalment acabem amb la creació d'un servidor web el qual té com a Backend un servidor escrit en NodeJS[7] i un Frontend fet amb tecnologia ReactJS[8].

Primerament s'ha començat a investigar com es pot obtenir el senyal d'àudio del micròfon. Per fer això s'ha connectat un dels cables a la sortida analògica del micròfon i a una de les entrades analògiques de les quals disposa Arduino Nano, en aquest cas el pin A7. Seguidament al loop del sketch simplement s'ha posat que anés mostrant per pantalla quin era el valor analògic que estava captant. Els valors que sortien per pantalla eren tots relativament similars i a primera vista no semblava que el micròfon funcionés. Amb l'ajuda d'un tornavís es va rotar l'element que fa variar l'offset del component i d'aquesta forma es va veure que el valor mostrat per pantalla no podia ser major a 1023. Això és degut al fet que el registre màxim que té Arduino són 10 bits[9].

En aquest punt s'ha de fer una transformació de les dades, ja que el sensor ens està donant una sortida de fins a 5000mV. El factor de conversió consisteix a determinar quants mV correspon 1 unitat dins del registre, efectuant la divisió de 5000mV entre 1023 valors, obtenim que cada unitat correspon a 4.88 mV. Un cop la conversió s'ha efectuat, se segueix sense veure una resposta visual clara sobre si el micròfon funciona correctament o no. Per acabar de resoldre aquest dubte s'ha optat per fer ús de l'estadística. Això ha consistit a aplicar la fórmula de la desviació típica sobre un conjunt d'elements d'entrada. El codi desenvolupat permet inicialment establir una mida fixa d'un buffer i anar guardant les dades en aquest cada cert període (1 ms), llavors quan el buffer ha estat complet es calcula la desviació típica i es mostra per pantalla. Posant a prova el desenvolupat fins al moment resulta que finalment ha funcionat, si en un entorn amb silenci es fa un soroll donant un cop a una taula per exemple, podem observar com afirmativament la desviació típica augmenta relativament respecte als estats anteriors i següents.

Una vegada resolt el dubte de si el receptor d'àudio funcionava o no s'ha procedit a experimentar amb aquest, amb l'objectiu d'establir un threshold o llindar pel qual es pugui determinar si l'Arduino ha rebut un so o no. Això encara que comporta moltes hores d'experimentació és molt important per al bon funcionament del sistema de posicionament. La calibració d'aquests sensors es fa perquè volem evitar en un alt percentatge el nombre de falsos positius, valors que l'Arduino donaria com a valors vàlids i implicarien un càlcul erroni en la distància que separa la Raspberry d'un dels nodes, fent que el càlcul de la posició final també sigui errònia.

Per dur a terme aquest calibratge s'ha començat inicialment amb uns altaveus de sobretaula que fan el paper temporal d'emissors de so en aquest sistema. A més a més s'ha fet servir un site web[10], el qual permet crear ones sinusoidals en un rang de freqüència comprès entre 20Hz-20KHz, que correspon justament al rang de freqüència que pot escoltar un ésser humà. Per establir el llindar primerament s'ha observat quina és la mitjana dels valors que van sortint per pantalla visualment de manera aproximada, seguidament s'estableix el llindar amb aquest valor i si el valor llegit està per sobre del llindar, llavors s'imprimeix per pantalla "So detectat". D'aquesta forma provant diverses hores amb certes freqüències i llindars s'ha pogut determinar quin rang de freqüències el sensor és més propens a detectar.

Els resultats de les proves conclouen que aquest hardware detecta millor freqüències compreses entre 330Hz-430Hz, fet que ha produït que es determini un so de 350Hz de freqüència i 1 segon de duració per a totes les proves següents. Com tots els micròfons que s'han adquirit tenen l'offset a diferents posicions i és pràcticament impossible sincronitzar-los tots a la mateixa posició, ja que és un procés manual, el calibratge del llindar s'ha de fer a nivell de software i és independent per a cada Arduino.

Un cop ja s'ha adaptat el software del sketch als nous requisits del micròfon i s'ha definit un so que maximitza la possibilitat de ser escoltat pel sensor, és hora de configurar el Controller de la Raspberry per tal d'incloure la funcionalitat de poder emetre un so mitjançant un altaveu. Aquest desenvolupament no ha sigut gaire difícil però sí que s'ha vist afectat per una refactorització del codi. Inicialment el programa Java ha de definir la ruta on es troba el recurs referent al fitxer d'àudio de freqüència 350Hz, seguidament es carrega en memòria les dades i finalment es reprodueix el so. Tot seguit el programa entra en bucle on es queda llegint tot el que rep pel socket obert contra el Gateway, quan el programa llegeix una línia que té com a payload el valor "So detectat"que correspon al missatge enviat per part de l'Arduino quan aquest detecta un so, llavors es fa una segona marca de temps. Es calcula la diferència entre les marques de temps i s'obté el temps que ha trigat el so a anar i ser detectat per l'Arduino, que correspon al RTT. Com s'ha comentat amb anterioritat es consideren insignificant els retards en la comunicació de tornada per radiofreqüència, ja que només impliquen 1 ms en el càlcul del RTT. El programa mostra per pantalla el RTT d'aquella iteració i torna a començar el cicle, emetent un so i esperant que el Gateway porti el missatge amb el payload corresponent. En aquest moment es consideren certs punts del codi que necessiten ser refactoritzats:

- Necessitat d'un timeout.

  Aquest és un dels punts més urgents, ja que com que no hi ha un timeout definit, el programa es queda esperant indefinidament fins que l'Arduino contesti. Com durant les proves és molt probable que s'allunyi massa l'emissor del receptor o hi hagi alguna mena de problema en la comunicació per radiofreqüència i l'Arduino no arribi mai a captar cap so o a enviar la confirmació, fent que el programa es quedi esperant. Finalment ha consistit a establir un timeout de 5 segons, condicionat per cada lectura que es fa del socket, és a dir, si després de rebre una línia i processar-la ja han passat 5 segons llavors deixem d'esperar i tornem a començar el cicle.

- Desacoblament de la reproducció del so.

  Fins al moment tots els mètodes de posada a punt de l'àudio i reproducció estan fortament acoblats al bucle principal del programa, fet que es pot veure al terminal, ja que la primera iteració sempre triga més de 500 ms en ser detectat. Això és degut al fet que inicialment l'àudio no es troba carregat en memòria i el programa estableix una marca de temps just després de cridar al mètode que reprodueix el so. El temps que triga el programa a carregar per primer cop l'àudio en memòria feia que el primer RTT sempre fos molt elevat. Per resoldre això s'ha dividit en dos aquesta funcionalitat, primerament es posa a punt l'arxiu i el programa fa un sleep de 700 ms, seguidament fem la primera marca de temps i immediatament cridem al mètode que reprodueix el so. D'aquesta forma s'ha aconseguit aïllar aquests mètodes tan acoblats i que el primer RTT que rebi el programa no estigui afectat per temps de càrrega a memòria.

- Netejat del buffer del socket.

  Durant el desenvolupament del Controller no es va contemplar certa característica sobre el buffer que llegeix la sortida del socket obert contra el Gateway. Aquesta característica fa referència al fet que el buffer funciona com una pila o stack, cosa que feia que en certes ocasions el RTT que s'obtenia era d'1 ms. Per entendre aquesta situació es considera el següent exemple: Imaginem que tenim un àudio amb una duració exacta d'un segon i que quan l'Arduino detecta aquest so llavors fa un sleep de també un segon. Resulta que l'Arduino està col·locat en una posició en la qual les ones reboten molt contra els objectes i tot i haver estat esperant 1 segon sense detectar res, resulta que l'ona rebota i l'Arduino ho detecta just al final. El que produeix això és que l'Arduino envii dues vegades en un interval d'1 segon que ha detectat un so, fent que el buffer guardi dues vegades la petició de rebuda. El que això implica és que en la següent iteració el programa llegirà del buffer i veurà la segona afirmació de la iteració anterior, donant-la com a vàlida i conseqüentment donant un valor del RTT no fidel amb la realitat.

  Per arreglar aquesta problemàtica hi ha dues opcions: per una banda augmentar el temps d'espera que fa l'Arduino entre recepció de so o per altra banda buidar el buffer entre cada una de les iteracions. Finalment s'ha considerat només buidar el buffer entre iteracions,

ja que augmentar el sleep del sketch faria augmentar el temps global de les iteracions.

Després d'haver aplicat aquestes refactoritzacions del codi, el Controller ja és completament capaç de calcular el RTT sense cap mena de contaminació. En aquest punt del projecte s'ha plantejat de crear la funcionalitat de distingir pel que fa al software quin Arduino ens està donant cert valor de RTT. El software creat fins al moment només espera la cadena "So detectatï no fa diferenciació de quin és el component hardware que hi ha al darrere.

Per complir aquest objectiu s'han hagut de fer dues coses, per una banda definir quin hardware seria el Node 1 i quin el Node 2 i així successivament, per altra banda s'han hagut d'agregar noves lògiques al Controller per tal de poder fer la distinció entre els Arduino. El que s'ha decidit és fer una diferenciació pel que fa al payload de resposta. Tot i tenir un camp dins del missatge que identifica quin és el node que ho ha enviat, tal com s'ha explicat a la secció 5.2, aquest valor és establert de forma arbitrària pel Gateway i pot anar canviant durant el temps. Tenint en compte que això podria portar molts problemes de cara al desenvolupament, finalment s'ha establert que els nodes tinguin un identificador intern i que el payload sigui estructurat de la següent manera: "Node "+ node-id.

D'aquesta manera al Controller s'ha hagut d'adaptar tot el codi que fa la detecció del missatge i configurar-lo perquè tingui en compte la part final del payload, que correspon a l'identificador del node. Amb aquest canvi inicialment només es pot veure quin dels nodes respon més ràpidament, ja que encara el codi no té la lògica associada d'esperar múltiples respostes, ja que com s'ha comentat abans s'efectua una neteja del bucle en cada iteració, fent que només es pugui veure el primer node que ha contestat. Per aplicar la lògica de múltiple espera s'ha hagut de definir, un nombre fixe de nodes que esperarem que contestin, i fins que no contestin tots o no s'hagi acabat el timeout, el programa seguirà escoltant en bucle. El nombre de nodes que volem que contestin s'ha parametritzat, de tal forma que sigui més còmoda l'experimentació constant que implica aquest projecte.

7.3 Servidor web

Un dels objectius d'aquest projecte és crear un servidor web per tal de mostrar de forma adequada totes les dades relacionades amb els Arduino i la distància a la qual es troben de la Raspberry.

Per tal de satisfer aquest objectiu s'ha procedit a crear un servidor web, el qual tindrà com a Backend un servidor escrit en NodeJS i un Frontend escrit en ReactJS. El fet d'haver seleccionat aquests frameworks és degut a la popularitat que han anat agafant durant aquests últims anys i que ha fet que en el món laboral estiguin tan sol·licitats, motivant-me a mi per intentar aplicar els coneixements sobre aquestes tecnologies en projectes reals.

En aquest projecte el Backend i el Frontend són totalment independents i no necessiten l'un de l'altre per a funcionar. Inicialment s'ha començat a desenvolupar el Frontend, ja que és aquí on es determina quines són les dades que volem mostrar i en quin format, d'aquesta manera haurem de crear l'API del Backend en funció de les dades que es precisin. En una iteració inicial, s'ha decidit implementar una taula simple, la qual contindrà l'identificador de l'Arduino, el RTT i la distància a la qual es troba de la Raspberry en funció del RTT. Aquesta taula s'ha d'anar refrescant de forma dinàmica cada cert temps, per tal d'estar sempre actualitzada i mostrar les dades més recents. Aquest tipus de refrescs es poden fer de forma automàtica i sense haver de recarregar la pàgina gràcies a la tecnologia React, la qual té un cicle de vida que permet renderitzar de nou els components (la taula és un component) si el *state* d'aquests s'ha modificat. Llavors s'ha establert una funció que cada 500 ms fa una crida al Backend i actualitza el *state* del component amb les noves dades.

De cara al Frontend, l'estructura de dades que precisa el component de la taula, és un array que contingui objectes, cadascun amb els paràmetres *node*, *time* i *dist*. El Backend llavors ha de retornar al Frontend les dades en aquesta estructura que acabem de definir. Després de meditar sobre aquest control i transformació de les dades, s'ha decidit que serà el Controller l'encarregat d'escriure en un fitxer les dades en el format que ho vol el frontal, de mode que el servidor de NodeJS només haurà de llegir un fitxer de text i enviar la resposta cap al frontal cada vegada que rebi una petició.

Finalment s'ha implementat la lògica al Controller de transformar el valor del RTT en unitats de distància, aplicant la fórmula del càlcul de distàncies:

$$\triangle x = \triangle v \cdot RTT(ms) = 0.34m/ms \cdot RTT(ms) \quad (2)$$

Aquest càlcul s'efectua posteriorment un cop ja tenim tots els RTT de la iteració del bucle, per tal de no incrementar els diferents RTT si el programa es queda fent càlculs. Un cop obtingudes les distàncies llavors s'estructuren els objectes, s'inclouen en un array i s'escriuen en un fitxer, per tal de posteriorment ser llegit per NodeJS i enviat al Frontend que renderitza aquesta taula.

8 RESULTATS

En aquesta secció es comparen cadascun dels objectius conjuntament amb la feina feta per assolir-los.

Tal com s'ha vist en la secció del desenvolupament, l'objectiu de crear una infraestructura hardware que permeti una comunicació punt a punt entre Raspberry i Arduino ha sigut totalment satisfactòria, havent canviat fins i tot de requeriments a la meitat del projecte. Respecte a l'intercanvi d'informació entre nodes també ha estat un èxit, havent pogut implementar dos variants en el tipus de comunicació: comunicació per radiofreqüència i comunicació basada en enviament i recepció de so. Aquesta implementació ha fet que finalment es pugui obtenir un valor de RTT entre Raspberry i Arduino.

Respecte a la transformació del RTT en unitats de distància s'ha seguit la fórmula matemàtica (2) exposada anteriorment. Existeix una problemàtica en el càlcul del RTT, ja que per diversos factors s'obté una variació aproximada de 10 ms que implica un error en el càlcul final de 3,4 metres.

S'ha pogut distingir satisfactòriament a quina distància es troben els diferents Arduino, sense tenir encreuaments de dades, gràcies a la lògica del sketch en contenir l'id del

node en el payload del missatge i a la lògica del Controller per poder diferenciar-los.

Pel que fa al tractament i processament de dades s'ha aconseguit guardar en un fitxer l'històric de distàncies i RTT que s'han obtingut per a cadascun dels Arduino durant el temps. Fent que sigui possible una futura consulta a aquest arxiu per tal d'extreure més estadístiques i conclusions

L'objectiu d'accedir remotament al sistema s'ha pogut satisfer gràcies a la creació del servidor web basat en NodeJS i ReactJS, addicionalment configurant l'encaminador de casa per tal d'obrir els ports i fent ús d'una eina que permet associar la IP dinàmica de l'encaminador de casa a un nom de domini de forma gratuïta. Gràcies a això podem accedir des de qualsevol navegador web de forma remota per veure a quina distància es troba la Raspberry de cadascun dels nodes.

Conforme als resultats exposats, es vol destacar, la casi completesa del 100% dels objectius a excepció del procés de mapejat 3D i triangulació de l'element a ser posicionat. Amb la variació actual que es té sobre el RTT, un sistema de posicionament indoor per a cases per exemple no seria del tot adequat. En l'estat actual del projecte, es podria aplicar aquest sistema en llocs on aquest marge d'error no fos tan significatiu, per exemple, identificar en quina sala o botiga d'un centre comercial es troba cert element que volem tenir monitorat. Tot i la falta del modelatge 3D i un càlcul del RTT més aproximat, es considera que el projecte ha arribat a un estat força madur per al temps que tenen associats els projectes de fi de grau.

9 Linies de Futur

En aquesta secció s'exposen tots els punts del projecte que es considera que necessiten alguna refactorització o alguna nova implementació de cara al futur.

Un dels punts que més mal de caps ha portat en aquest projecte ha sigut el component de hardware del micròfon. Sense cap mena de dubte una de les primeres coses que s'haurien de fer per continuar endavant amb aquest projecte és substituir aquest component per un amb més sensibilitat, ja que perquè aquest detecti un so a més de 4 metres és molt complicat i implica fer un soroll massa fort.

El càlcul de les distàncies està molt condicionat per certes variacions que impliquen un càlcul erroni del RTT, es té la sospita que el component hardware del micròfon sigui el causant d'aquestes variacions i és per això que es prioritza el canvi d'aquest. A més a més s'hauria de valorar el fet de mostrar distàncies més acurades aplicant per exemple una mitja d'un conjunt de distàncies obtingudes durant els últims 10 segons. Fet que milloraria la precisió a canvi d'actualitzar les dades de manera menys freqüent.

Un altre tema relacionat ve a ser intentar veure si seria possible poder adquirir un component que fos capaç de detectar sons els quals no fossin escoltats per les persones, és a dir per sobre del rang dels 20KHz, d'aquesta manera el càlcul de distàncies no implicaria haver d'escoltar un so molest cada vegada.

D'altre manera, es podria intentar implementar un sistema que mesurés distàncies mitjançant tècniques d'aprenentatge supervisat, per tal de veure si millora la precisió.

Addicionalment de cara al futur s'hauria d'implementar una base de dades per tal de guardar de forma més ordenada totes les dades que es van generant. Això permetria fer estudis de manera més còmoda sobre les dades, podent obtenir per exemple recorreguts dins d'una sala, quanta distància s'ha recorregut en un cert temps, etcètera.

Finalment quedaria el tema del mapejat 3D de l'objecte en la sala, havent de crear un espai tridimensional tant per la part de càlculs que hauria de fer el Backend com la presentació que s'hauria de mostrar a l'usuari des del Frontend.

10 Conclusions

En aquest apartat s'exposen les conclusions que han sigut fruit de la realització d'aquest projecte de final de grau.

Durant l'elaboració d'aquest, he après i reforçats molts coneixements en diferents àmbits i tecnologies. Per una banda he après que gràcies al software obert de MySensors es poden fer projectes realment útils per a la vida domèstica amb molt pocs diners, ja que els components són molt econòmics i permeten molta flexibilitat pel que fa a l'expansió de funcionalitats que poden oferir. A més, encara que existeixen ja plataformes de control de sensors domòtics de software obert, com per exemple Domoticz[11], és possible escriure el teu propi controlador de forma relativament senzilla si tens coneixements en programació.

Per altra banda també he reforçat coneixements referents a la comunicació via sockets i a la integració d'aquesta comunicació en entorns Java, així com el reforçament de la programació en aquest llenguatge.

Finalment he reforçat coneixements en creació de servidors web fent servir NodeJS com a Backend i a més l'ús de la tecnologia React per a frontals web, que no havia tocat mai durant el transcurs del grau i recentment ho estic aplicant a més projectes personals.

Gràcies a aquests tipus de projectes la domòtica a baix cost és cada dia més accessible per a tothom i en un futur farà que les automatitzacions i petits sistemes domòtics es trobin a totes les cases.

Agraïments

En primer lloc, agraeixo sincerament al meu tutor de projecte Oriol Parera per ajudar-me des de l'inici, aportant noves idees en els moments que més encallat m'he trobat, sense ell aquest treball no hagués pogut arribar tan lluny com ho ha fet.

En segon lloc m'agradaria agrair a la meva família, en especial a la meva parella Cristina, per tots aquells moments que aquest treball m'ha portat tant a la frustració de no saber com avançar com per a l'alegria de poder continuar cap endavant.

Finalment agraeixo als meus companys i companyes d'universitat per escoltar-me i aconsellar-me durant aquesta última etapa del grau.

Referències

[1] Localino - Solutions for Indoor Localization — RTLS. [En línia]. Disponible a: https://www.localino.net/en Consulta: 31 de Maig del 2019.

[2] Building a Raspberry Pi Gateway — MySensors - Create your own Connected Home Experience. [En línia]. Disponible a: https://www.mysensors.org/build/raspberry Consulta: 4 de Juny del 2019.

[3] Serial Protocol - 2.x — MySensors - Create your own Connected Home Experience. [En línia]. Disponible a: https://www.mysensors.org/download/serial_api_20 Consulta: 4 de Juny del 2019.

[4] Maniacbug.github.io. RF24: RF24 Class Reference. [En línia]. Disponible a: https://maniacbug.github.io/RF24/classRF24.html Consulta: 6 de Juny del 2019.

[5] NRF24L01 Documentation [En línia]. Disponible a: http://forum.arduino.cc/index.php?action=dlattach;topic=225113.0;attach=72943 Consulta: 7 de Juny del 2019.

[6] En.wikipedia.org. Electret microphone. [En línia]. Disponible a: https://en.wikipedia.org/wiki/Electret_microphone Consulta: 8 de Juny del 2019.

[7] Acerca — Node.js. [En línia]. Disponible a: https://nodejs.org/es/about Consulta: 9 de Juny del 2019.

[8] React – Una biblioteca de JavaScript para construir interfaces de usuario. [En línia]. Disponible a: https://es.reactjs.org Consulta: 13 de Juny del 2019.

[9] Arduino.cc. Arduino Reference. [En línia]. Disponible a: https://www.arduino.cc/reference/en/language/functions/analog-io/analogread Consulta: 8 de Juny del 2019.

[10] Szynalski.com. Online Tone Generator - generate pure tones of any frequency. [En línia]. Disponible a: https://www.szynalski.com/tone-generator Consulta: 13 de Juny del 2019.

[11] Domoticz.com. - Domoticz. [En línia]. Disponible a: http://www.domoticz.com Consulta: 21 de Juny del 2019.

APÈNDIX

A continuació es presenten algunes de les imatges relacionades amb aquest projecte, així com part de la planificació i certes interfícies d'usuari.

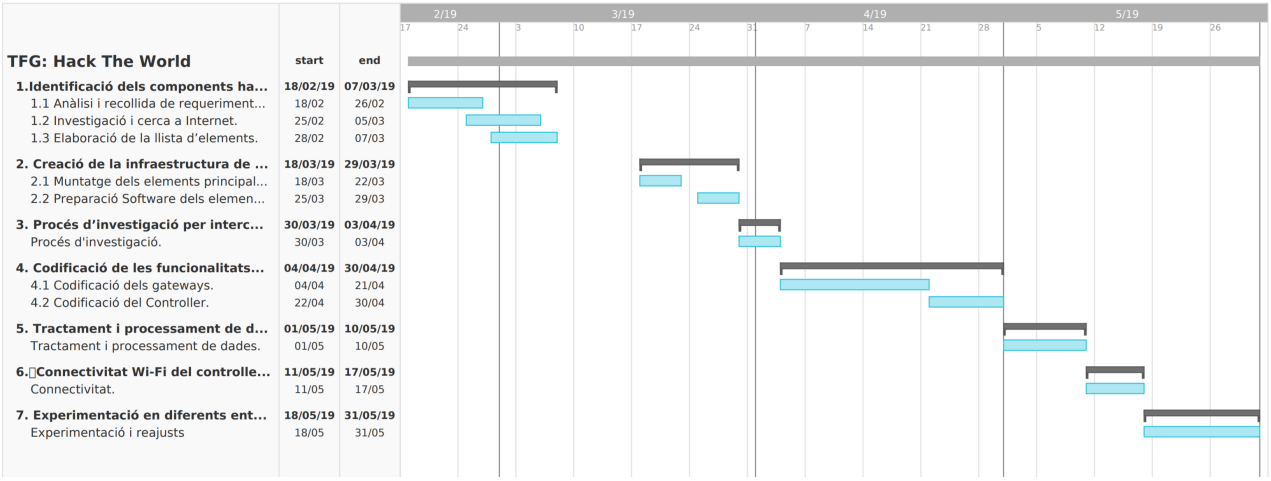

Fig. 3: Diagrama de Gantt del projecte

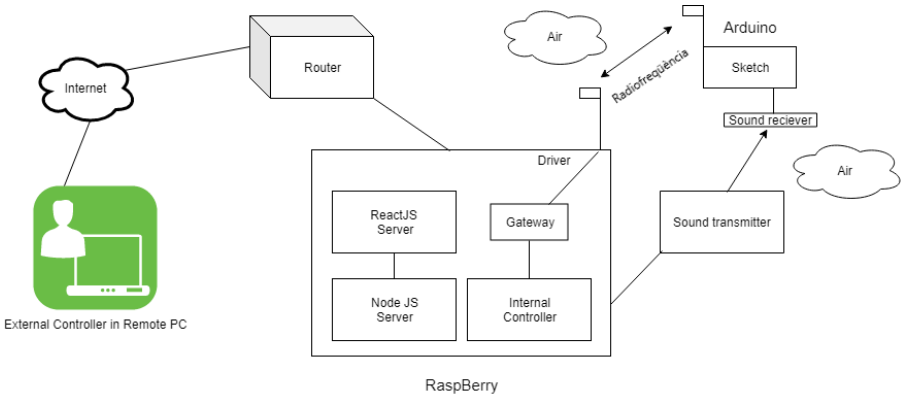

Fig. 4: Infraestructura final del Software

Node	Temps(ms)	Distància(m)
Nodo 1	34	11.67
Nodo 2	37	12.58

Fig. 5: Aplicació web en versió Desktop

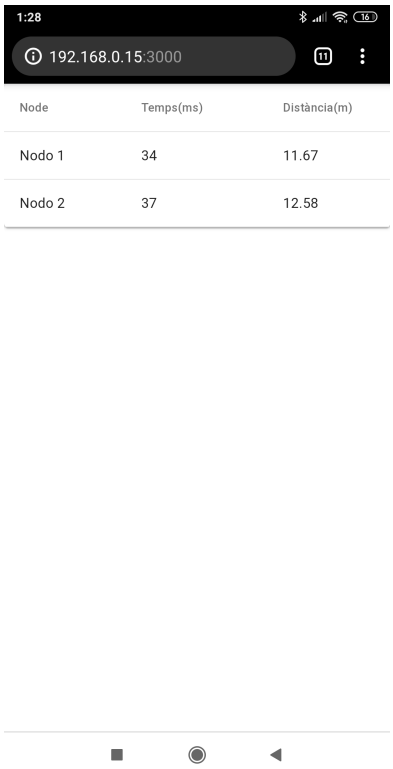

Fig. 6: Aplicació web en versió Mobile

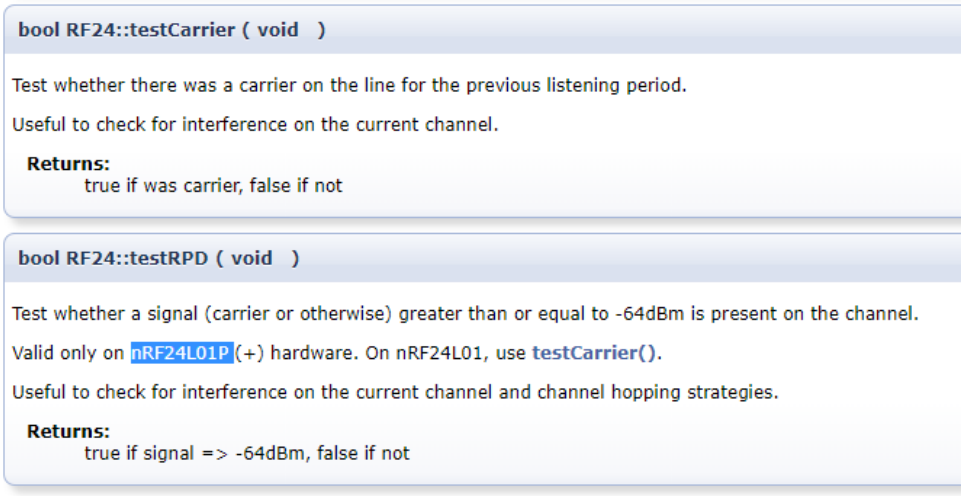

Fig. 7: Descripció del mètode que indica el portador en el hardware del component d'antena

```
/****************************************************************************/

bool RF24::testCarrier(void)
{
  return ( read_register(CD) & 1 );
}

/****************************************************************************/

bool RF24::testRPD(void)
{
  return ( read_register(RPD) & 1 ) ;
}

/****************************************************************************/
```

Fig. 8: Codi del mètode que indica el portador en el hardware del component d'antena