AUDITIVASSIST

Sistema de detecció i notificació de sons quotidians dins la vivenda pensat per persones amb discapacitat auditiva.

Eudald Ferré Baró
Grau d'Enginyeria Informàtica
TFG de l'àrea d'Arduino

Consultor: Antoni Morell Pérez
Professor: Pere Tuset Peiró

8 de Gener de 2019

Llicència

FITXA DEL TREBALL FINAL

Títol del treball:	*AUDITIVASSIST*
Nom de l'autor:	*Eudald Ferré Baró*
Nom del consultor/a:	*Antoni Morell Pérez*
Nom del PRA:	*Pere Tuset Peiró*
Data de lliurament (mm/aaaa):	*01/2019*
Titulació o programa:	*Grau Enginyeria Informàtica*
Àrea del Treball Final:	*Arduino*
Idioma del treball:	*Català*
Paraules clau	*Arduino, ESP8266, MQTT*

Resum del Treball

El present document te per objectiu descriure totes les fases de desenvolupament del producte AUDITIVASSIST, que s'han definit com, estudi del mercat i de productes existents, viabilitat del producte, selecció de tecnologies a utilitzar, desenvolupament i proves del prototip.

AUDITIVASSIST és un sistema de codi obert i de baix cost, per detectar i notificar sons típics en una vivenda, amb l'objectiu d'ajudar a persones amb discapacitat auditiva.

Aquest sistema es basa en tecnologies IoT i té tres tipus de nodes, el node detector de so, el node d'actuador i el node coordinador. El nombre de nodes que té un sistema concret és variable, i depèn de les fonts de so que es volen detectar, així com del nombre de llums amb les que es volen fer les notificacions.

El funcionament bàsic del sistema, es el de mitjançant el node detector de so, detectar sons a partir d'un determinat llindar (sons molt propers), i notificar-los al sistema mitjançant comunicacions MQTT. El node actuador, en rebre aquestes notificacions activa i desactiva les llums en base a uns patrons per notificar l'esdeveniment a l'usuari.

A la conclusió del treball s'han assolit tots els objectius tant tècnics com de planificació i econòmics fixats a l'inici del projecte, obtenint així, un prototip totalment operatiu.

Abstract (in English, 250 words or less):

The purpose of this document is to describe all the development phases of the AUDITIVASSIST product, which have been defined as, market research and existing products, product viability, selection of technologies to be used, development and testing of the prototype.

AUDITIVASSIST is a system of open source and low cost, to detect and notify typical sounds in a house, with the aim of helping people with hearing impairments.

This system is based on IoT technologies and has three types of nodes, the sound detector node, the actuator node and the coordinating node. The number of nodes that have a specific system is variable, and it depends on the sources of sound that you want to detect, as well as the number of lights with which you want to make the notifications.

The basic operation of the system is that of using the sound detector node, detecting sounds from a certain threshold (sounds very close), and notifying them to the system through MQTT communications. The actuator node, upon receiving these notifications, activates and deactivates the lights based on patterns to notify the event to the user.

Finally, all the technical, planning and economic goals set at the beginning of the project have been achieved, thus obtaining a fully operational prototype.

Contingut

Índex de figures

1. Introducció

1.1 Context i justificació

El terme domòtica fa referència als sistemes capaços d'automatitzar una vivenda aportant serveis que tenen per objectiu els següents àmbits d'aplicació:

- Estalvi energètic
- Millora del confort de les persones
- Millora de la seguretat de les persones
- Comunicacions
- Accessibilitat

Amb el desenvolupament d'aquest projecte es pretén fer ús de la domòtica per millorar l'autonomia de les persones amb discapacitat auditiva dins la seva llar.

Avui en dia, amb l'explosió de les tecnologies IoT i l'aparició d'estàndards que milloren la interoperabilitat entre sistemes i fabricants, es poden desenvolupar sistemes més simples accessibles a tothom i adaptables a la gran varietat de necessitats concretes dels usuaris.

Tot i aquest context favorable, la realitat es que hi ha una manca real de sistemes que permetin aquest col·lectiu de persones millorar la seva vida quotidiana dins la seva llar, col·lectiu, que per altra banda, representa, segons l'informe EDAD (INE, 2008), el 2,5% de la població espanyola, mentre que les estimacions fetes per la OMS, preveuen que al 2050, una de cada deu persones patiran discapacitat auditiva.

Per tant, aquest projecte pretén desenvolupar un producte plenament funcional amb el menor cost econòmic possible que sigui integrable amb altres sistemes i que millori l'accessibilitat a persones amb discapacitat auditiva

1.2 Objectius

L'objectiu del projecte es desenvolupar un sistema simple i autoconfigurable que assisteixi a persones amb discapacitat auditiva dins la seva llar. Per això, el sistema ha de ser capaç de detectar sons quotidians d'una vivenda i transformar-los en senyals lumínics i/o notificacions al telèfon mòbil.

Actualment ja existeixen petits sistemes que porten a terme aquestes funcions, però o son sistemes antics, centralitzats i principalment cablejats, el que complica i encareix la seva instal·lació; o bé, s'han de comprar diversos *gadgets*, cadascun amb la seva pròpia funció. Aquest projecte, pretén unificar totes les necessitats d'aquest col·lectiu dins la llar, en un sol sistema fent ús de les darreres tecnologies en IoT, i fent-lo accessible a tot tipus de persones, simplificant i abaratint la seva instal·lació.

Per tal d'assolir aquests dos principals objectius, simplicitat i cost reduït, s'ha optat per utilitzar una plataforma de codi obert com és Arduino, per:

- Cost
- Estàndard
- Multiplataforma
- Entorn de programació simple i clar
- Codi obert i, programari i maquinari extensible

Els objectius concrets a que s'han fixat per al desenvolupament d'aquest projecte són:

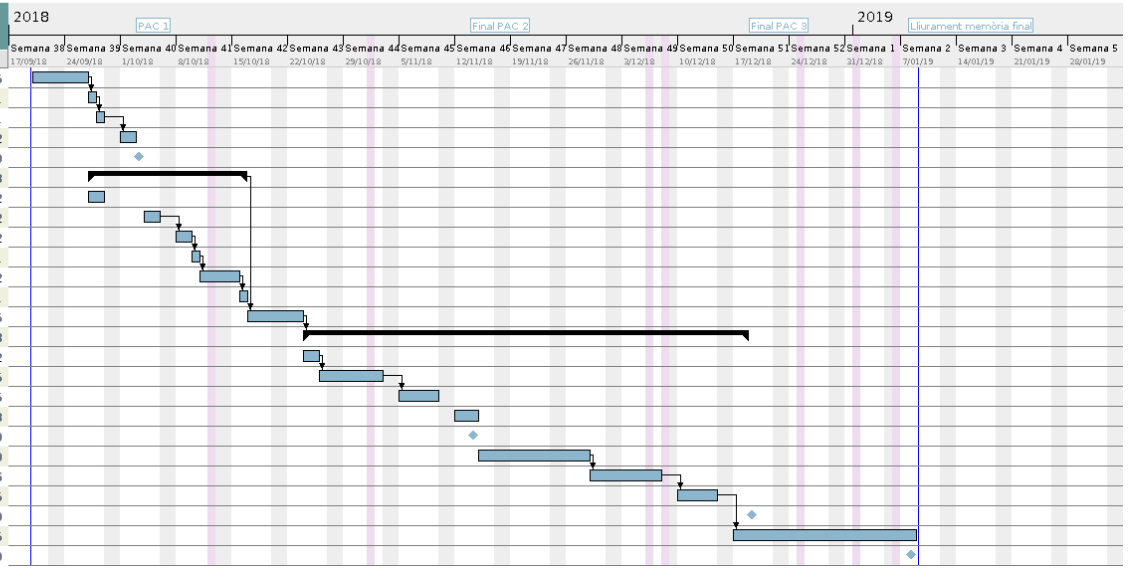
- Desenvolupament d'un node captador de sons amb taxes de detecció superiors als 98%, per a sons provinents de fonts a una distància inferior als 50 cm.
- Desenvolupament d'un node actuador amb capacitat per controlar càrregues, tant resistives com inductives, de fins a 2 Ampers.
- Desenvolupament d'un mòdul de notificacions a telèfons mòbils.
- La latència total de qualsevol notificació no ha de superar en cap cas els 5 segons de retard, entre l'esdeveniment i la darrera notificació a l'usuari.
- L'usuari, a través del portal web del sistema, ha de poder activar alarmes que se li notifiquin a través de les llums, a mode de despertador.
- El cost final del *pack* estàndard, format per 2 nodes detectors, 4 nodes actuadors i 1 node coordinador ha de ser inferior als 300€/*pack*.

Així doncs, per tal d'assolir l'objectiu principal del projecte, caldrà al llarg del desenvolupament d'aquest anar assolint fites concretes:

- Selecció de la metodologia de captació i reconeixement d'un so.
- Selecció del maquinari necessari, per als tres tipus d'elements previstos al sistema.
- Selecció de l'arquitectura del sistema.
- Familiaritzar-se amb la tecnologia escollida.
- Desenvolupament d'una interfície web a Raspberry Pi.
- Desenvolupament d'un mòdul d'enviament de notificacions a telèfons mòbils a Raspberry Pi.

1.3 Planificació

Nombre	Fecha de inicio	Fecha de fin	Duración
Selecció temàtica del projecte	20/09/18	26/09/18	5
Definició del projecte	27/09/18	27/09/18	1
Estudi de viabilitat	28/09/18	28/09/18	1
Planificació	1/10/18	2/10/18	2
PAC 1	3/10/18	3/10/18	0
Fase d'estudi	27/09/18	16/10/18	13
Tecnologies disponibles	27/09/18	28/09/18	2
Plataforma Arduino	4/10/18	5/10/18	2
Plataforma Raspberry Pi	8/10/18	9/10/18	2
Captació i reconeixement de sons	10/10/18	10/10/18	1
Arquitectura MQTT	11/10/18	15/10/18	2
Selecció final d'equipament a utilitzar	16/10/18	16/10/18	1
Acopi de material	17/10/18	23/10/18	5
Fase de desenvolupament	24/10/18	18/12/18	38
Instal·lació del servidor MQTT a Raspberry Pi	24/10/18	25/10/18	2
Implementació comunicacions MQTT a Arduino	26/10/18	2/11/18	5
Implementació detecció i reconeixement de sons	5/11/18	9/11/18	5
Proves parcials	12/11/18	14/11/18	3
Final PAC 2	14/11/18	14/11/18	0
Implementació servidor Web a Raspberry Pi	15/11/18	28/11/18	10
Implementació notificacions mòbils	29/11/18	7/12/18	6
Proves finals	10/12/18	14/12/18	5
Final PAC 3	19/12/18	19/12/18	0
Documentació final	17/12/18	8/01/19	15
Lliurament memòria final	8/01/19	8/01/19	0



Il·lustració 1: Diagrama de Gantt amb la planificació del projecte

2. Estudi de productes semblants

2.1 Visualfy

Il·lustració 2: Visualfy Home

Visualfy, podem dir que és l'únic producte que hem localitzat al mercat amb la mateixa funció que la que es busca en aquest projecte, i que integra les darreres tecnologies disponibles al mercat, sense cap mena de dubte, Visualfy és un producte molt innovador.

Tanmateix, fruit d'aquesta innovació, detectem algunes limitacions per a segons quin tipus d'usuari, a continuació les detallem.

Fa us de les darreres tecnologies sense integrar-ne d'altres de més simples, el que centra aquest producte a un tipus de públic més jove. Això es un problema, més si tenim en compte que segons dades de l'estudi EDAD 2008 (INE, 2008), la major concentració de discapacitats es troba en edats superiors als 65 anys. Val a dir, que no tenim la dada concreta per a discapacitats auditives, però entenem que la distribució ha de ser molt semblant.

Així doncs, tenint en compte, que el principal mercat objectiu d'aquest producte seran persones amb edats superiors als 65 anys, trobem a faltar sistemes més simples. Es a dir, que l'usuari no hagi de dependre només del telèfon mòbil o de televisors, rellotges o bombetes intel·ligents, per rebre les notificacions, sinó que també disposi de sistemes més quotidians, mitjançant els quals, pugui rebre les notificacions, com podria ser, l'ús de l'enllumenat de la mateixa vivenda.

Un altre punt negatiu que trobem es el **cost**. El producte bàsic, que inclou 3 sensors de sons, té un cost superior als 500€, a banda, s'ha de comptar amb una quota anual de 24€ per la APP, sense la qual, no es poden rebre notificacions. A banda, s'ha de comptar amb el cost de tots aquells elements intel·ligents on es vulgui rebre notificacions, com son rellotges, televisors o llums intel·ligents.

Per això, entenent la tecnologia com un mecanisme d'integració, no pas d'exclusió, en aquest projecte es pretén, fent ús de les darreres tecnologies, desenvolupar un producte innovador però accessible a tothom, tant des de la vessant tècnica com de l'econòmica.

Ara bé, això no treu que ens trobem davant d'un producte de referència, exemple d'innovació i d'integració de les darreres tecnologies al servei de les persones amb discapacitat, i podem afirmar quasi amb rotunditat que l'únic producte d'aquestes característiques que existeix a nivell mundial.

3. Viabilitat

El mercat objectiu d'aquest tipus de producte són totes les famílies amb algun dels seus integrants amb discapacitat auditiva. A continuació posem algunes dades de fonts oficials que ens poden ajudar a determinar la grandària actual i futura del mercat objectiu.

Segons dades de l'estudi EDAD 2008 (INE, 2008), dins l'estat espanyol hi ha més d'un milió de persones majors de sis anys amb discapacitat auditiva, el que representa el 2,5% de la població espanyola.

Per altra banda, si ens fixem en un context mundial, segons dades de la Organització Mundial de la salut (OMS), 466 milions de persones a tot el món pateixen discapacitat auditiva. I es preveu que al 2050 més de 900 milions de persones – es a dir, una de cada 10 persones - també patiran discapacitat auditiva.

3.1 Viabilitat tècnica

El fet de basar el desenvolupament sobre dues plataformes com Arduino i Raspberry Pi, ambdues àmpliament emprades a nivell mundial per desenvolupar tot tipus de projectes, això ens atorga un nivell de viabilitat tècnica contrastat.

Per altra banda, a nivell de comunicacions entre nodes, per tal de donar robustesa al sistema i simplificar el seu desenvolupament, s'utilitzarà un protocol estandarditzat molt utilitzat a l'entorn IoT, com és MQTT (*Message Queuing Telemetry Transport*). A nivell de xarxes, inicialment només es preveu l'ús de xarxes WiFi, donat que a l'actualitat tots els habitatges disposen o tenen possibilitat de disposaré de xarxes d'aquest tipus.

3.2 Viabilitat econòmica

Per fer l'estudi de la viabilitat econòmica del producte, farem un anàlisi de cost i benefici, que ens indiqui la viabilitat o no del projecte, fent una estimació del cost del desenvolupament del producte i del llançament d'aquest al mercat. Un cop acotat el cost, estimarem les vendes que es poden produir, per finalment, obtenir l'indicador ROI (Retorn d'inversió).

COST DESENVOLUPAMENT I ADEQUACIÓ DEL PROTOTIP

Concepte	Quantitat	Preu	Total
Raspberry Pi Model A+	1	31,34€	31,34€
Font d'alimentació	2	12,00€	24,00€
Caixa Raspberry Pi	1	9,95€	9,95€
Sensor de so	1	13,92€	13,92€
Sparkfun ESP8266 WiFi Shield	1	18,00€	18,00€
WiFi ESP8266 SSR Board	1	20,00€	20,00€
Altres materials	1	20.00€	20,00€
Tests i certificacions del prototip	1	2.000,00€	2.000,00€
Hores d'enginyeria de desenvolupament	300	25,00€	6.250,00€
		TOTAL	**8.387,21€**

COST DE PROMOCIÓ I LLENÇAMENT

Concepte	Quantitat	Preu	Total
Events i fires	1	4.000€	4.000€
Publicitat online	1	3.000€	3.000€

Pàgina web	1	2.000€	2.000€
Generació de contingut audiovisual	1	1.000€	1.000€
		TOTAL	**10.000€**

Així doncs, tal com podem veure a la taula anterior, el cost total de desenvolupament, adequació, promoció i llançament està proper als 20.000€. A continuació passem a calcular els costos indirectes generats per la venda del producte, aquí hi inclourem el cost de distribució, del suport tècnic i de la garantia dels equips.

COSTOS INDIRECTES

Concepte	Cost per *pack* o equip
Distribució (empaquetatge, transport i altres)	5,00€
Suport tècnic (personal de suport, material d'ajuda, etc.)	2,50€
Garantia	2,50€
TOTAL	10,00€

Ara passem a calcular el cost directe de cadascun dels nodes:

COST NODE COORDINADOR

Concepte	Quantitat	Preu	Total
Raspberry Pi Model A+	1	31,34€	31,34€
Font d'alimentació	2	12,00€	24,00€
Caixa Raspberry Pi	1	9,95€	9,95€
		TOTAL PVP	**53,29€**

COST NODE CAPTADOR DE SONS

Concepte	Quantitat	Preu	Total
Sensor de so	1	13,92€	13,92€
Sparkfun ESP8266 WiFi Shield	1	18,00€	18,00€
		TOTAL PVP	**31,92€**

COST NODE ACTUADOR

Concepte	Quantitat	Preu	Total
WiFi ESP8266 SSR Board	1	20,00€	20,00€
		TOTAL PVP	**20,00€**

El càlcul dels costos directes de cada actuador s'han fet utilitzant preus PVP, tanmateix la política de compres obligarà a no treballar amb cap proveïdor que no ofereixi un mínim d'un 30% de descompte, amb el que el producte es vendria amb un marge de venda mínim del 30%. Tanmateix, per tal d'amortitzar la inversió inicial, es pretén aplicar un 10% addicional per aquest concepte.

Ara, fixem uns objectius de vendes realistes durant els primers 5 anys que van des de la venda de 50 *packs* el primer any, amb increments del 100% any a any fins arribar a la venda de 800 *packs* el cinquè any. A continuació, calcularem quant de temps es trigarà a recuperar la inversió inicial, estimant la venda d'un *pack* estàndard que contindrà dos nodes captadors – 1 per captar trucades de telèfon i un altre per al timbre de casa – 4 nodes actuadors, per les diferents estances i 1 node coordinador. El preu final d'aquest *pack* seria de 197,13€ + 10% d'amortització + 10€ de costos indirectes → 226,85€ IVA inclòs.

OBJECTIUS DE VENDES I D'AMORTITZACIÓ

Any	Objectius de vendes	Increment	Amortització inversió inicial
1	50	-	8,25%
2	100	100%	24,75%
3	200	100%	57,75%
4	400	100%	>100%
5	800	100%	

Així doncs, tal com es pot veure a la darrera taula, amb les previsions de vendes fetes, amb 4 anys es podria recuperar la inversió feta per al desenvolupament i llançament del producte. A partir d'aquí, o es poden amortitzar altres inversions fetes per la millora, o es podria suprimir aquest 10% addicional, abaratint el producte, tenint en compte que ja disposem d'un marge de venda ampli d'un mínim del 30%.

4. Tecnologies

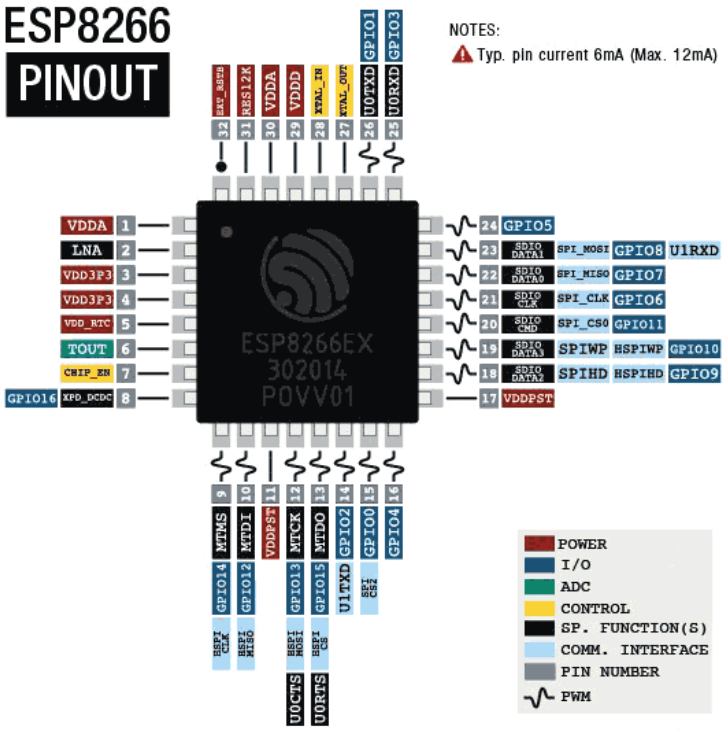
4.1 ESP8266

El ESP8266 és un SoC (*System on Chip*) que incorpora un processador de baix consum amb una arquitectura RISC de 32 bits i un xip WiFi amb gestió de pila TCP/IP. L'aparició d'aquest microcontrolador l'any 2014 va revolucionar el sector IoT, per el seu cost tant baix, actualment es troba al voltant dels 3€, i per el ventall d'oportunitats que obria. El fet de tractar-se d'un microcontrolador de baix consum amb connectivitat WiFi incorporada i a un preu tant baix, el fa el producte ideal per a dispositius IoT.

El processador que integra és un Tensilica L106 de 32 bits amb arquitectura RISC que funciona a una velocitat de 80MHz, i amb una velocitat màxima de 160MHz.

A nivell de connectivitat, podem dir que suporta IPv4 i protocols com TCP, UDP, HTTP i FTP. Així mateix, també hi ha suport per al protocol HTTPS, tot i que no de forma nativa, sinó mitjançant llibreries externes. Per altra banda, també disposa de busos de comunicació SPI, I2C i UART.

I ja per acabar, les tres darreres propietats que el fan especialment atractiu per al projecte que estem desenvolupant, es que incorpora memòria flaix, no al propi xip, però si en un xip a banda, i la seva integració es totalment transparent; incorpora ports d'E/S i un controlador ADC; i finalment, es programable des de l'IDE d'Arduino.



Il·lustració 1: Pinout ESP8266

Així doncs, tal i com veurem al llarg de la memòria, basarem tots els nostres nodes sobre aquest xip, ja que ens permet, programar-lo des del propi entorn de desenvolupament d'Arduino, incorpora comunicació WiFi, disposa de memòria flash, gràcies a la qual podrem emmagatzemar fitxers de configuració i incorpora ports d'E/S suficients per a la nostra aplicació. Per tant, si executéssim aquest projecte amb Arduino, necessitaríem una placa Arduino que incorporés

Wifi, de cost més elevat o una placa Arduino + una *Shield*. Per tant, podem afirmar que la diferència de preu entre executar el projecte amb Arduino o basat en ESP8266 duplicaria el cost.

4.2 Arduino

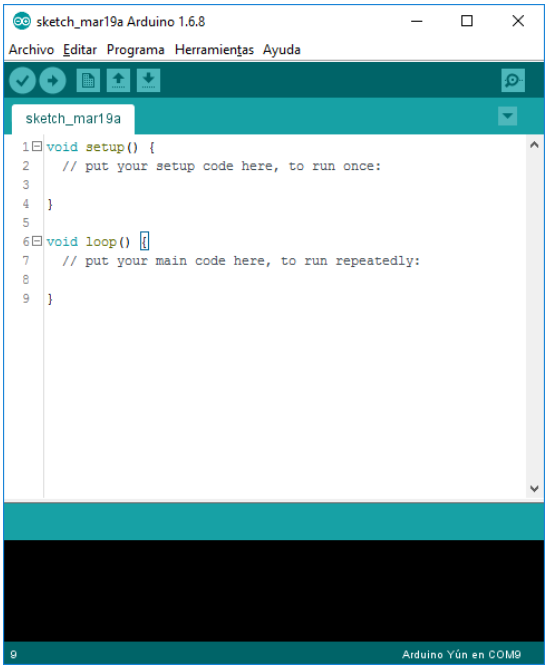
Arduino es un projecte que neix l'any 2005, com una evolució de Wiring, un projecte fruit d'una tesi d'Hernando Barragán a l'institut IVREA d'Ivrea (Itàlia) (Barragán, s.f.). L'objectiu principal del projecte es desenvolupar un maquinari amb un microcontrolador i una interfície per programar-lo, simple i de baix cost que permeti a persones sense coneixements tècnics avançats desenvolupar projectes de diversa naturalesa.

Els dissenys de maquinari d'Arduino estan formats per un microcontrolador i ports d'entrada/sortida, tant digitals com analògics, així com ports de comunicació que permeten ampliar les funcionalitats de la placa base amb targetes *shield*. Tots aquests dissenys són de codi obert i es pot descarregar tota la informació del seu disseny.

En aquest projecte, NO farem servir maquinari d'Arduino, ja que existeixen tecnologies i plataformes, també de codi obert, que s'adapten molt millor al projecte, tal i com ja hem explicat anteriorment. Tanmateix, si que farem servir el seu entorn de programació, ja que ens ofereix un entorn i un codi molt simple i fàcil d'utilitzar i el maquinari utilitzat permet ser programat des d'aquest entorn.

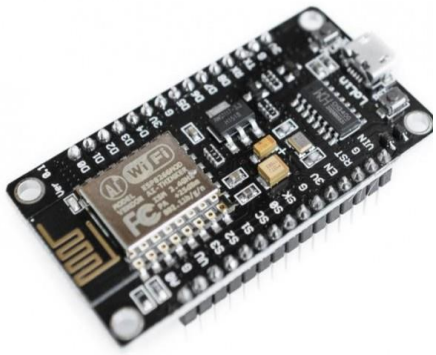
L'entorn de programació d'Arduino és una aplicació multiplataforma escrita amb llenguatge Java, i de codi obert. Aquest entorn permet escriure programes a les plaques d'Arduino utilitzant llenguatge *scketch*. *Scketch* és un llenguatge semblant a C que ha de contenir obligatòriament dues funcions:

- *Setup()*: Dins d'aquesta funció s'ha de inicialitzar tots els components que es volen utilitzar a la funció *loop()*.
- *Loop()*: Aquesta funció s'executa cíclicament i és la funció que contindrà la lògica del programa.

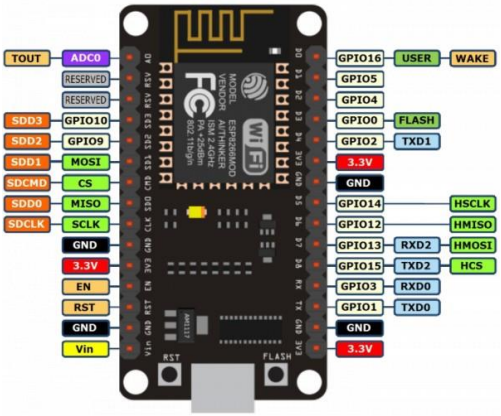


Il·lustració 3: IDE d'Arduino

4.3 NodeMCU



Il·lustració 4: Placa NodeMCU



Il·lustració 5: Pinout NodeMCU

NodeMCU és una placa de desenvolupament que integra el ESP8286. NodeMCU és una plataforma de codi obert molt orientada al desenvolupament IoT, que inclou, tant el maquinari basat en el mòdul ESP-12 (ESP8266) i un port micro-USB per descarregar programes, com el programari que permet controlar el ESP8286.

A nivell de programari, ens trobem davant d'una plataforma de desenvolupament amb Lua, que a priori, no encaixa massa amb el projecte que estem desenvolupant, amb el que només ens afegiria complexitat a la solució final. Tanmateix, l'autèntic potencial que ens ofereix es que també pot ser programada des del mateix IDE d'Arduino com si d'una placa d'Arduino es tractés.

Per tant, en resum, la tria d'aquesta tecnologia per al desenvolupament d'aquest projecte rau en el fet de disposar d'una placa amb la comunicació Wi-Fi integrada a un preu molt inferior al d'una placa Arduino + *shield* i que es programa amb el mateix IDE que la resta del projecte.

I per si això no fos prou, fent ús d'aquesta tecnologia hem trobat un fabricant de la Índia (ARMTRONIX) que desenvolupa plaques basades en nodeMCU alimentades a 220VAC i que integren un relé d'estat sòlid, el que ens dona tot el maquinari necessari per al desenvolupament del node actuador del nostre projecte, i en una sola placa.

A demés, el fet d'incorporar un relé d'estat sòlid, augmenta la fiabilitat del sistema al hora de connectar carregues ja que no tenen contactes mecànics que es desgastin ni genera arcs elèctrics a la connexió o desconnexió de la càrrega que disminueixin la vida útil dels contactes.

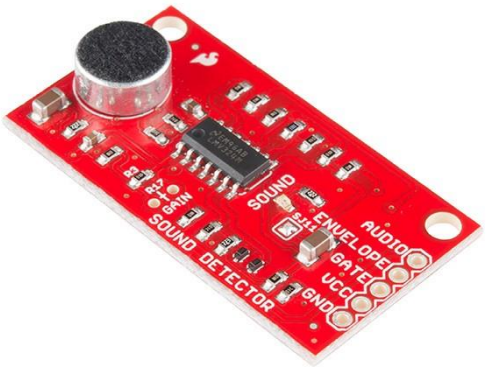
Altrament, aquests fabricant també subministra el codi Arduino necessari per crear un AP (*Access Point Wifi*) i una web de configuració, quan la placa arrenca sense informació de la xarxa a la que s'ha de connectar o quan aquesta no està disponible, així com un codi bàsic de comunicació mitjançant MQTT per encendre i apagar el relé d'estat sòlid (ARMTRONIX, 2018). Per tant, donat que aquest codi ja ens aporta una part de la funcionalitat que volem donar al sistema, l'utilitzarem com a base, sobre la que treballar adaptant-lo en tot allò que necessitem.

Il·lustració 6: WIFI ESP8266 SSR Board

4.4 Detecció de sons

Per a la detecció de sons, utilitzarem un sensor desenvolupat dins la comunitat de codi obert, que podem emmarcar dins la comunitat Arduino.

Aquest sensor disposa d'un micròfon, un amplificador i tres sorties – 2 analògiques i 1 digital. La sortida digital s'activa quan el so detectat supera un llindar que es pot configurar modificant la ganacia del circuit, tanmateix, això obliga a configurant-la mitjançant resistències. Per evitar això, en aquest projecte, utilitzarem la sortida analògica que ens dona l'amplitud del so detectat, gracies a la qual podrem detectar els sons buscats i descartar aquells que no són d'interès.



Il·lustració 7: Sensor de sons

4.5 Raspberry Pi

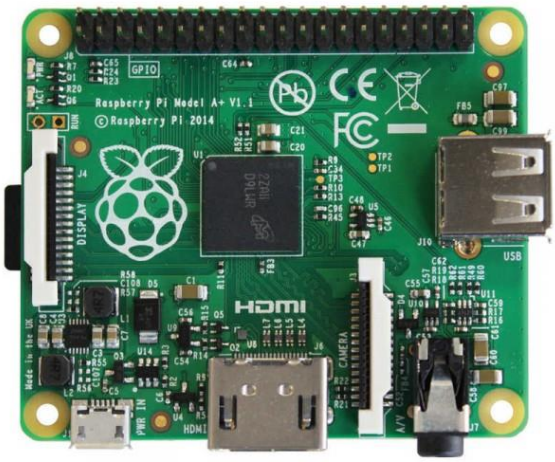
Raspberry Pi és un ordinador de dimensions molt reduïdes en una sola placa i de baix cost, desenvolupat per la fundació Raspberry Pi, amb l'objectiu d'estimular l'aprenentatge de ciències de la computació a les escoles.

A diferència del que trobàvem amb Arduino, Raspberry Pi no es de codi obert en quant a plataforma de maquinari, ara bé, si que ho és la plataforma de programari oficial, la qual es una versió adaptada de Debian, anomenada Raspbian.

Si ens fixem en l'arquitectura, també trobem diferències importants amb Arduino, ja que mentre aquest darrer integrava un microcontrolador, Raspberry Pi funciona amb un microprocessador, el que li permet disposar d'una major potència de càlcul i de memòria.

Per aquest projecte, donat que el node coordinador no requerirà de grans necessitats de memòria, ni necessitats addicionals de ports per a perifèrics, s'ha optat per la versió reduïda i econòmica de Raspberry Pi, el model A+, a continuació detallem les característiques d'aquest model:

- Processador: Broadcom BCM2835 SoC Full HD
- GPU: Co-processador multimèdia Dual Core Videocore IV
- RAM: 256MB SDRAM 700MHZ
- HDD: MicroSD
- 1 port USB
- Sortida HDMI



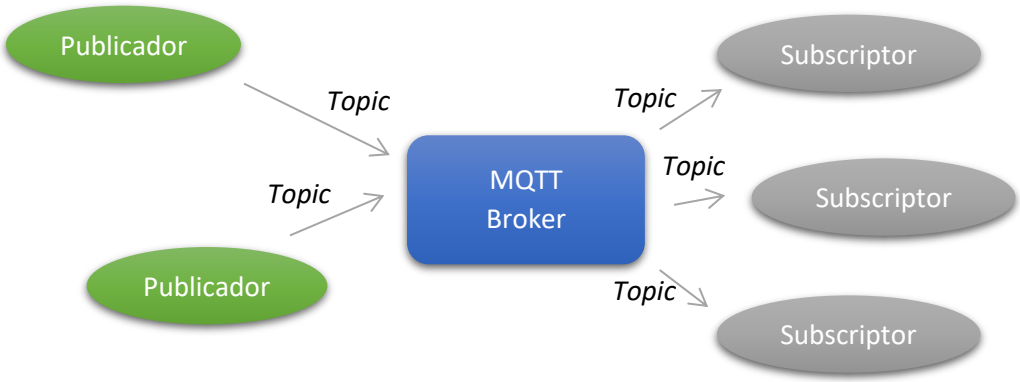
Il·lustració 8: Raspberry Pi model A+

En quant a programari, Raspberry Pi es compatible amb un gran nombre de sistemes operatius, la majoria basats en nucli Linux, però fins i tot hi ha versions de Windows compatibles. Tanmateix, en aquest projecte utilitzarem el sistema operatiu oficial, que com ja hem comentat anteriorment, és Raspbian.

4.6 MQTT

MQTT (*Message Queue Telemetry Transport*) és un protocol especialment simple i lleuger, dissenyat per a sistemes que disposen de molt poc ample de banda, que requereixen un consum energètic molt baix i amb pocs recursos disponibles. Per totes aquestes característiques, aquest protocol és especialment idoni per a sistemes M2M o IoT.

L'arquitectura d'aquest protocol segueix un model Publicador/Subscritptor, amb un node central anomenat *broker* que es qui coordina les comunicacions entre subscriptors i publicadors.



El funcionament es força simple i gira al voltant del concepte de *Topic*, mitjançant els quals el *broker* coordina les comunicacions entre publicadors i subscriptors. Els subscriptors es subscriuen a un tema del que volen rebre actualitzacions i els publicadors publiquen les actualitzacions d'un tema que rebran tots els seus subscriptors.

Addicionalment, MQTT incorpora una característica molt interessant per aquest tipus de sistemes, l'estructura jeràrquica dels *Topics*, és a dir, els clients subscrits a un tema, rebran totes les seves actualitzacions, així com, totes les actualitzacions dels temes que jeràrquicament són dependents del tema subscrit.

En el camp del programari de codi obert, podem trobar diversos servidors MQTT que podríem fer servir, tanmateix, el que trobem que té una millor acceptació, amb un ventall més ampli de documentació a la xarxa, és Mosquitto (Eclipse Foundation, s.f.), raó per la qual, ens decidim a utilitzar-lo per aquest projecte. A nivell de clients MQTT per arduino, també trobem diverses opcions, però finalment es decantem per utilitzar la llibreria *Arduino Client for MQTT* (O'Leary, s.f.), ja que és un projecte de codi obert molt actiu i amb molta documentació disponible a la xarxa.

5. Desenvolupament del projecte

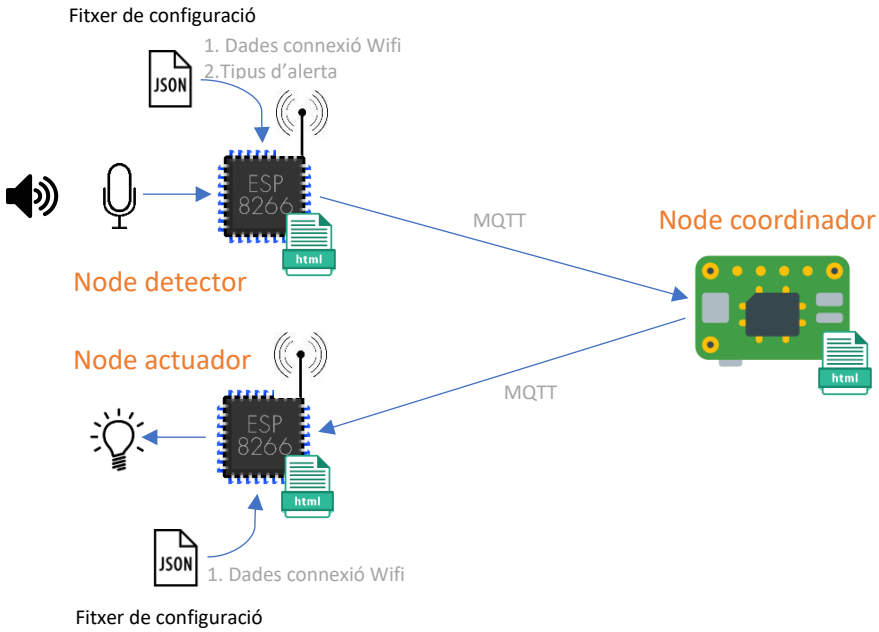
Abans d'entrar a explicar les funcionalitats concretes de cada node, a la següent figura es pot veure de forma general el sistema a desenvolupar.

El principi bàsic de funcionament, està basat en la detecció de sons a partir d'un llindar mitjançant un micròfon i un amplificador de senyal. La sortida d'aquest amplificador està connectada a l'entrada ADC d'un microcontrolador ESP8266 connectat a una xarxa WiFi. Quan el senyal provinent del amplificador supera un llindar, el microcontrolador envia un missatge MQTT al sistema.

El node actuador, igual que el node detector, està basat en un microcontrolador ESP8266 connectat a la mateixa xarxa WiFi. En rebre una notificació MQTT de so detectat, el node executa un patró d'activació i desactivació de la seva sortida digital, que varia en funció de l'esdeveniment rebut.

Els nodes basats en ESP8266, tenen un fitxer JSON emmagatzemat a la seva memòria ROM on es guarden els paràmetres de connexió a la xarxa WiFi, així com, en el cas del node detector, el tipus d'alerta que ha d'enviar al sistema en detectar un so. Tots aquests paràmetres son modificables mitjançant el portal web del propi node.

Finalment, el node coordinador, basat en una Raspberry Pi connectada a la mateixa xarxa WiFi, executa el servidor MQTT, s'encarrega de notificar via telèfon mòbil els diferents esdeveniments del sistema i disposa d'un portal web per configurar, testejar o supervisar el sistema.



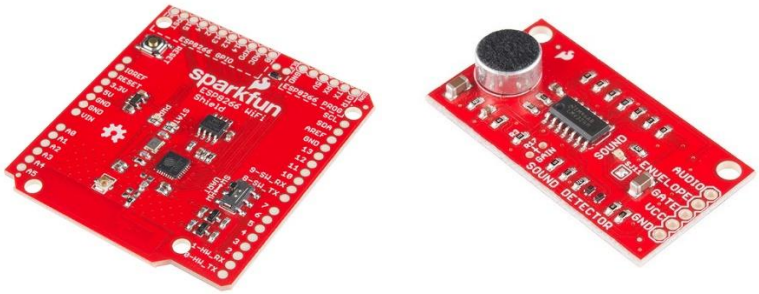
Il·lustració 9: Visió general del sistema a desenvolupar

5.1 Node captador de sons

El node captador de sons serà l'encarregat de detectar sons a partir d'un llindar i notificar-ho al sistema.

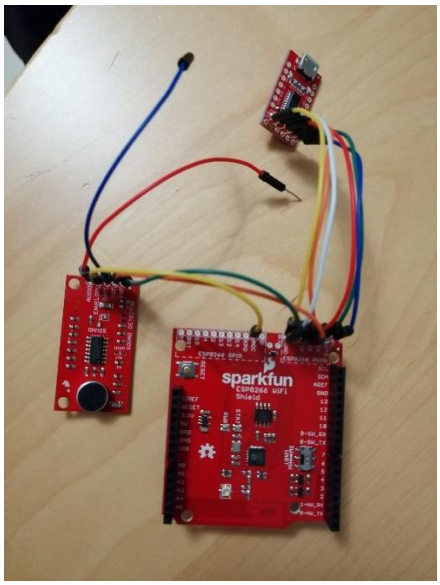
El maquinari d'aquest node, tal i com ja avançàvem a l'apartat de tecnologies, està basat en ESP8266 amb el sensor de sons connectat a la seva entrada ADC.

A nivell de programari, per tal d'estandaritzar els programes dels diferents nodes, en aquest node utilitzarem com a base, el codi subministrat amb la placa *WiFi ESP8266 SSR Board*, ja que aquest ja inicia la xarxa WiFi com un AP, si no s'ha configurat cap xarxa per connectar o la que s'ha configurat no està disponible. I addicionalment, disposa d'un portal Web que permet configurar els paràmetres de la xarxa i els emmagatzema en un fitxer amb format JSON, i incorpora les comunicacions MQTT amb la llibreria *Arduino Client for MQTT*.



Il·lustració 10: Maquinari node captador de sons

El primer pas és el muntatge del maquinari, realitzant les connexions entre la placa ESP8266 i el sensor de sons, que les farem mitjançant cablejat i soldadura. El resultat final del muntatge és el que es mostra a la següent figura:



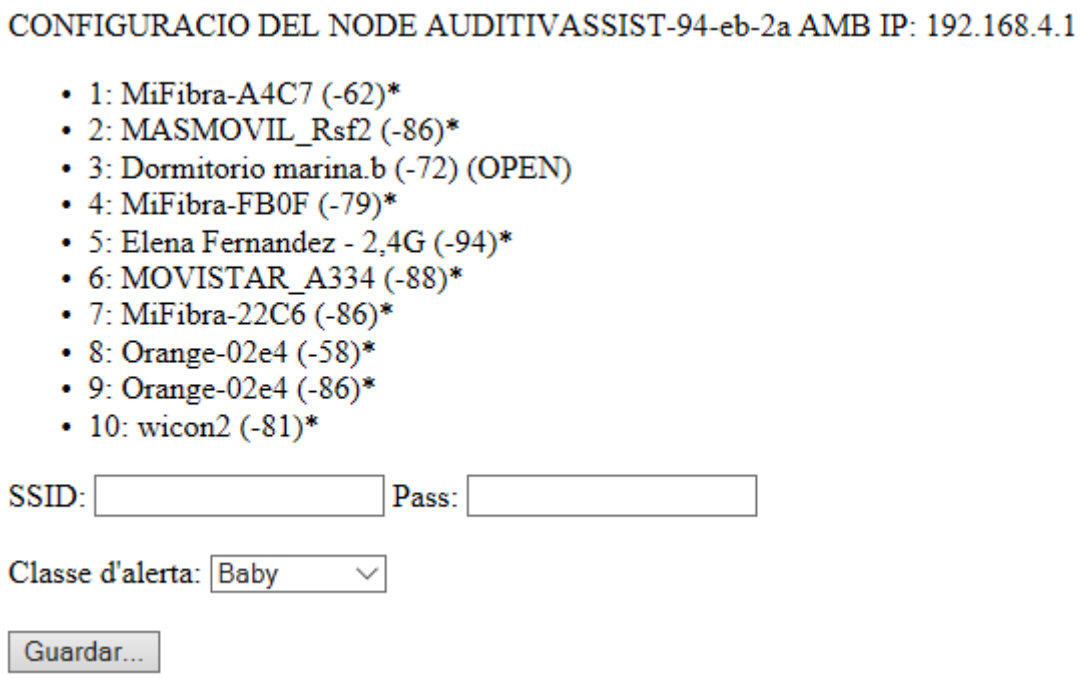
Il·lustració 11: Muntatge node captador de sons

En quant a codi, el node captador configurarà la seva entrada ADC i llegirà el seu valor constantment, perquè quan aquest superi el llindar configurat, el node enviï el missatge MQTT corresponent, en base a la seva configuració.

El missatge MQTT enviat, dependrà de la configuració feta al portal web del propi node. Tal i com es pot veure a la Il·lustració 12, els nodes actuadors tenen un camp de tipus d'alerta, on l'usuari haurà de configurar quin tipus de so detectarà el node. En base a aquesta configuració el node en detectar un so enviarà un missatge o un altre. A la següent taula es poden veure els diferents missatges que pot enviar un node actuador:

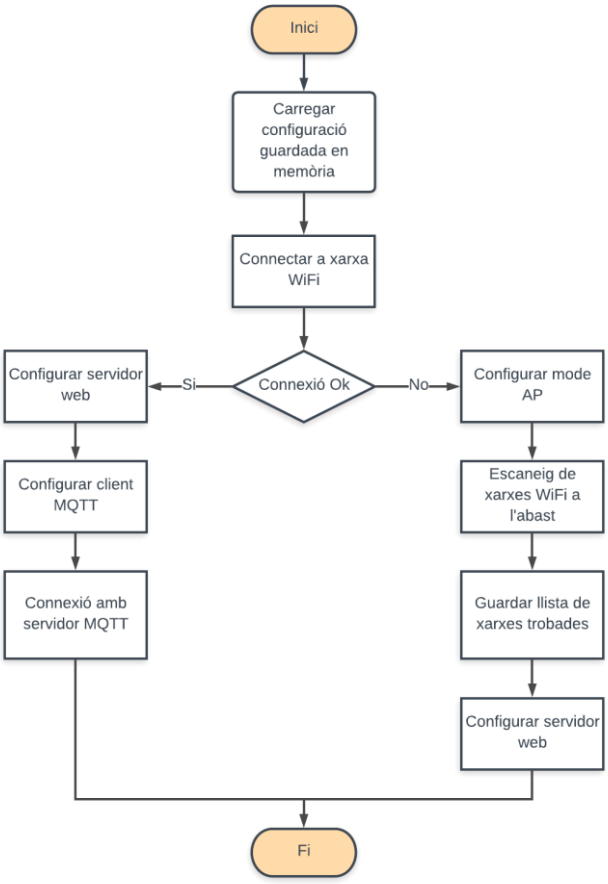
Missatge	Descripció
Baby	El node configurat per detectar el plor d'un nadó ha detectat un so
Door ring 1	El node configurat per detectar el timbre de la porta 1 ha detectat un so
Door ring 2	El node configurat per detectar el timbre de la porta 2 ha detectat un so
Telephone	El node configurat per detectar el so del telèfon ha detectat un so

La configuració del node es portarà a terme a través de la pàgina de configuració, que està basada en la subministrada amb la placa *WiFi ESP8266 SSR Board*, personalitzant els paràmetres necessaris. Així doncs, la web de configuració final d'aquest node es la que es mostra a la següent figura:

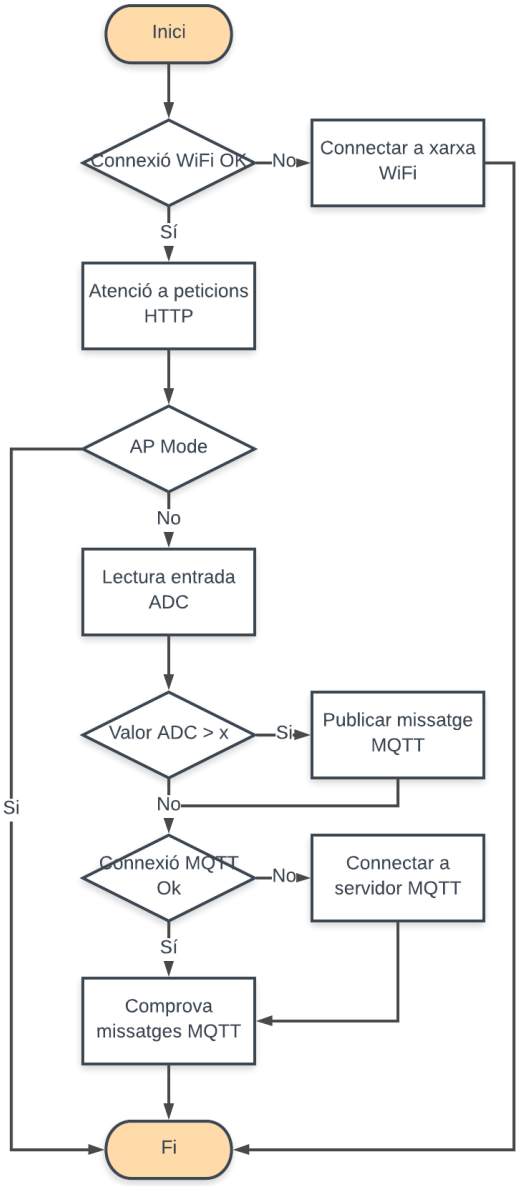


Il·lustració 12: Web de configuració node detector

A continuació presentem els diagrames de flux corresponents a les dues funcions *setup* i *loop:*



Il·lustració 13: Diagrama de flux "setup" nodes detector i actuador

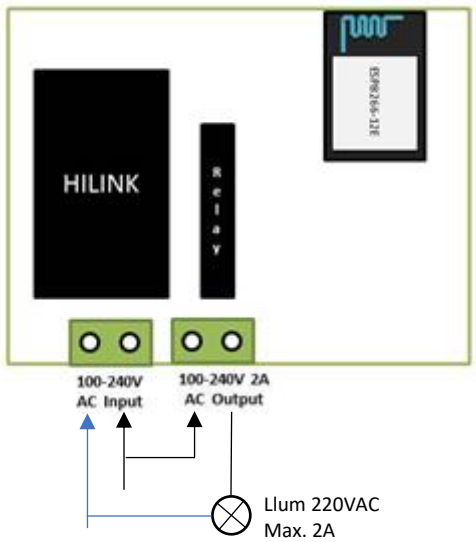


Il·lustració 14: Diagrama de flux "loop" node detector de sons

5.2 Node actuador

El node actuador serà l'encarregat de fer les senyals lumíniques que correspongui en base a l'alerta rebuda dels nodes captadors de sons.

Tal i com ja avançàvem a l'apartat de tecnologies, el maquinari triat per aquest node, ha estat una placa del fabricant Indi ARMTRONIX que compleix el 100% dels requisits que necessitàvem.



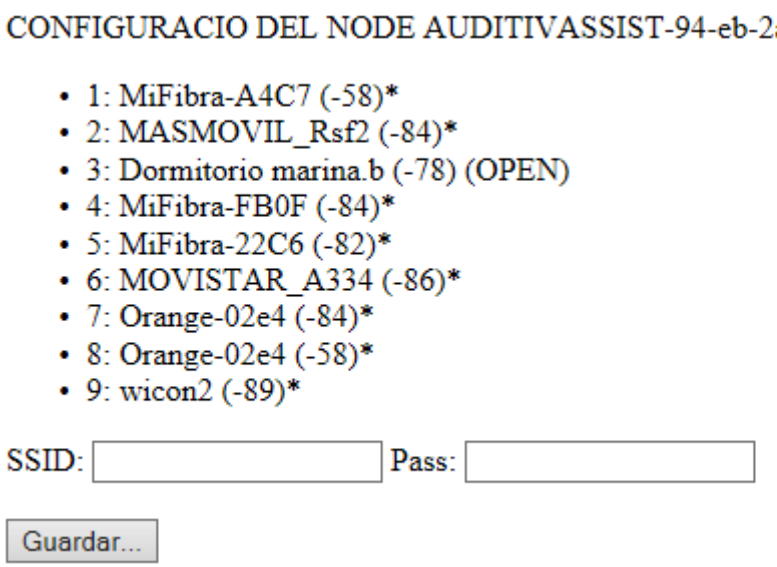
Il·lustració 15: Connexió node actuador

A nivell de programari, utilitzarem, com a base, el mateix programa que entrega el fabricant, ja que aquest ja inicia la xarxa WiFi com un AP, si no s'ha configurat cap xarxa per connectar o la que s'ha configurat no està disponible. I addicionalment, disposa d'un portal Web molt senzill que permet configurar els paràmetres de la xarxa i del servidor MQTT, i incorpora les comunicacions MQTT amb la llibreria *Arduino Client for MQTT*.

A nivell de modificacions que s'han de fer en aquest codi, per una banda, modificarem la funció que es crida en rebre dades al buffer de comunicacions MQTT. En aquesta funció indicarem els diferents missatges que es poden rebre i com ha d'actuar la sortida en base al missatge rebut. A la següent taula indiquem els diferents estats i els patrons d'activació de la sortida:

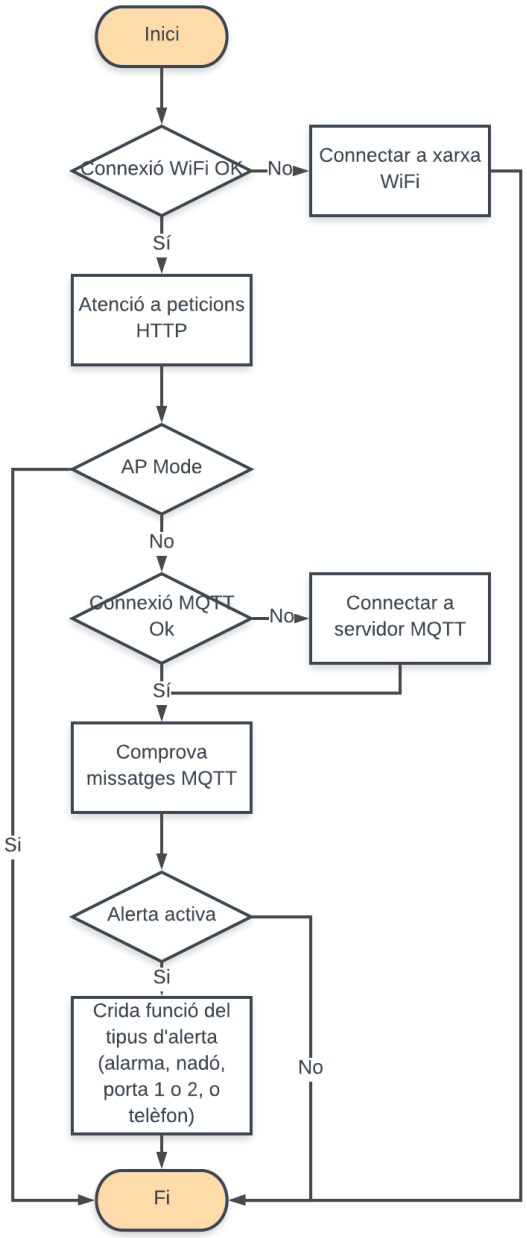
Missatge	Descripció	Patró d'activació de llums
Baby	S'ha detectat el plor d'un nen	Actiu 1s – Inactiu 1s
Door ring 1	S'ha detectat el timbre de la porta 1	Actiu 2s – Inactiu 1s
Door ring 2	S'ha detectat el timbre de la porta 2	Actiu 1s – Inactiu 2s
Telephone	S'ha detectat el timbre del telèfon	Actiu 0.5s – Inactiu 0.5s – Actiu 1s – Inactiu 0.5s – Actiu 0.5s – Inactiu 1s
Alarm	Activació d'una alarma definda per l'usuari	Actiu 2s – Inactiu 2s

I per últim, modificarem la pagina de configuració, ja que la web que subministra el fabricant, també permet configurar els paràmetres de comunicacions MQTT, tanmateix, per nosaltres, aquesta configuració és dona de fabrica i no es modificable, per tant traiem els paràmetres de la web de configuració i els fixem en el propi codi.



Il·lustració 16: Web de configuració node

A continuació presentem el diagrama de flux corresponent a la funció *loop*, el diagrama de flux corresponent a la funció *setup* es el mateix que per al node detector de sons*:*



Il·lustració 17:Diagrama de flux "loop" node actuador

5.3 Programari nodes detector i actuador

Tal i com ja hem comentat anteriorment, donat que la operativa de ambdós nodes es molt similar, per tal d'estandarditzar el codi, s'ha optat per desenvolupar un únic programa que amb una directiva de compilador, li diem a aquest, si el node que estem descarregant es un node actuador o un node detector, i el compilador genera només aquell codi que es necessari per al node en qüestió.

El desenvolupament d'aquests nodes, s'ha basat el codi d'Arduino per a ESP8266 desenvolupat per ARMTRONIX (ARMTRONIX, 2018), que ja integra quasi tota la funcionalitat que necessitem (comunicació MQTT, configuració de xarxa WiFi, inicialització en mode AP, servidor HTTP). Sobre el seu esquelet hem desenvolupat el programari dels nodes detector i actuador, adaptant el codi a les necessitats del projecte. A continuació enumerem les principals modificacions que ha calgut fer:

1. Incorporar un fitxer nou al programa amb les funcions de control de la sortida digital del node actuador. Aquest fitxer, incorpora les funcions de cada esdeveniment que es pot produir al sistema amb el seu patró d'activació i desactivació de la sortida digital.
2. Modificació de les funcions de comunicació MQTT, adaptant-les a les necessitats del projecte, incorporant els tipus de missatges que es poden donar.
3. Incorporació d'un missatge MQTT amb text "DEVICES" que utilitzarem per descobrir el nodes AUDITIVASSIST existents a la xarxa. En rebre aquest missatge, tots els nodes existents contestaran amb la seva identificació.
4. Modificació de les pàgines HTML per adaptar els paràmetres a les necessitats del projecte.
5. Simplificació dels processos d'arrencada i configuració, eliminant funcionalitats que no eren d'interès per al projecte.
6. Adaptació dels paràmetres que s'emmagatzemen en memòria no volàtil.

Per al desenvolupament d'aquest programari, s'han utilitzat les següents llibreries:

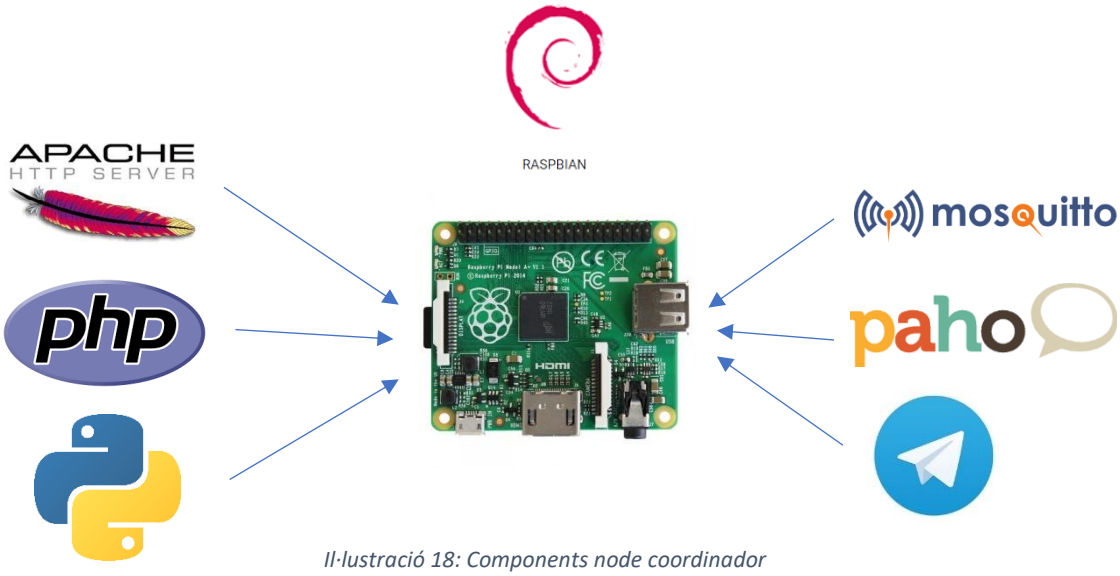
- ESP8266WiFi: Aquesta llibreria ens permet gestionar la connexió Wifi, permetent-nos connectar els nodes a una xarxa existent o posar-los en mode AP.
- ESP8266mDNS: Aquesta llibreria implementa el protocol mDNS en Arduino, gracies a la qual podem definir noms als nodes i localitzar-los per aquest nom sense necessitat d'instal·lar un servidor DNS.
- WiFiClient: Es la llibreria que ens permet crear clients WiFi, i enviar o rebre dades a través d'ells.
- PubSubClient: En permet de forma senzilla la implementació d'un client MQTT en Arduino.
- ArduinoJson: Aquesta llibreria ens permet emmagatzemar les dades que necessitem emmagatzemar en memòria no volàtil, dins d'un fitxer estructurades en format JSON.
- FS: La llibreria *File System* es la que ens permet emmagatzemar dades a la memòria *Flash* que conté el ESP8266, gracies a la qual emmagatzemem les dades de la connexió WiFi, entre altres.

5.4 Node coordinador

El node coordinador serà el node capçalera del sistema, des d'on podrem configurar alertes, notificacions i supervisar el sistema al complert. Així mateix, aquest node també actuarà de coordinador de les comunicacions entre sodes captadors i nodes actuadors, es a dir, serà el *broker* de l'arquitectura de comunicació MQTT.

Aquest rol el desenvoluparà la targeta Raspberry Pi, ja que pot muntar sistema operatiu i ens permetrà instal·lar, els servidors tant MQTT com Web. El sistema operatiu que utilitzarem per la Raspberry Pi, serà la darrera versió de Raspbian.

Per poder desenvolupar totes les funcionalitats requerides per aquest node, caldrà instal·lar els següents aplicatius i/o llibreries:



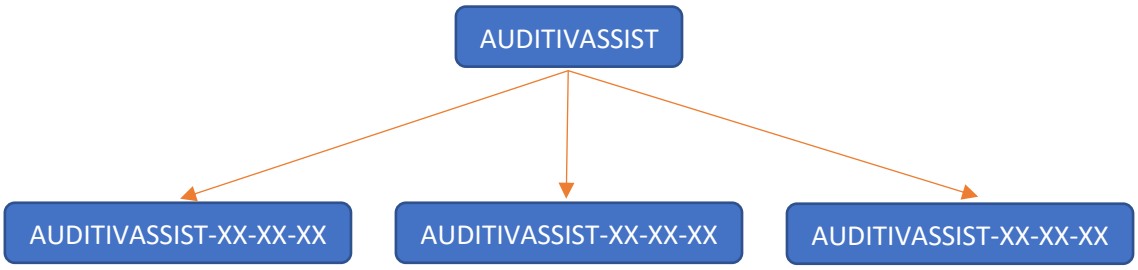
Il·lustració 18: Components node coordinador

5.4.1 Comunicacions MQTT

Tal i com ja hem avançat a l'apartat de tecnologies, el servidor MQTT triat per aquest projecte ha estat Mosquitto. Aquest servidor ens permet l'ús d'una jerarquia de *topics*, a través de la qual podem donar funcionalitats addicionals als sistema.

En concret, al sistema tindrem dos tipus d'alertes, aquelles que es reproduiran a tots els nodes actuadors de la vivenda, i aquelles que podrem configurar perquè només siguin notificades a través dels nodes actuadors que l'usuari triï. Per poder incorporar aquesta funcionalitat els nodes actuadors es subscriuran al *topic* AUDITIVASSIST on es publicaran totes les notificacions a reproduir a tots els nodes actuadors del sistema, així com, al *topic* AUDITIVASSIST/AUDITIVASSIST-XX-XX-XX, concret per a cada node actuador, on es publicaran aquelles notificacions que només s'han de reproduir en el node concret.

Així doncs, la jerarquia de *topics* resultant del sistema serà:



Il·lustració 19: Jerarquia de topics

D'aquesta forma, aquells nodes del sistema amb funcions especials, subscrivint-se al *topic* AUDITIVASSIST/# podran rebre totes les notificacions del sistema, mentre que la resta de nodes subscrivint-se al *topic* AUDITIVASSIST o AUDITIVASSIST-XX-XX-XX rebran els diferents tipus de notificacions.

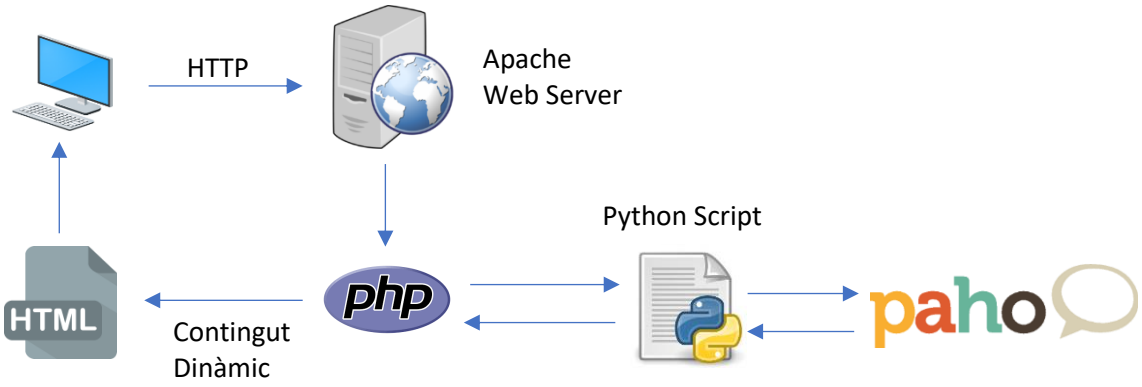
5.4.2 Servidor Web

Per al desenvolupament del servidor web encastat al node coordinador, s'utilitza un servidor HTTP APACHE, que ens proporcionarà tots els serveis HTTP requerits.

Addicionalment, i amb l'objectiu de donar major interactivitat al portal web, també utilitzarem un servidor PHP per tal d'incrustar codi entre les etiquetes HTML, permetent-nos carregar contingut dinàmic a les diferents pàgines.

I finalment, donat que el portal web haurà d'interactuar amb el servidor MQTT, enviant o llegint missatges, utilitzarem la llibreria PAHO, de codi obert, que implementa un client MQTT.

Aquesta llibreria està desenvolupada en diferents codis, en aquest cas, per coneixements del codi, utilitzarem la implementació en Python, tot i que no és la millor integració entre MQTT i un servidor web. Ja que la funcionalitat MQTT la desenvoluparem mitjançant scripts en Python i els cridarem des del servidor web amb codi PHP, el qual serà l'encarregat de presentar les dades a l'usuari. Donat que Paho també disposa de clients implementats en Javascript o PHP la integració amb un entorn web seria força més eficient.



Il·lustració 20: Esquema de peticions desencadenades al servidor web

5.4.3 Mòdul de notificacions mòbils

El mòdul de notificacions a mòbils, també s'implementarà mitjançant un script en Python i la llibreria *pyTg* (luckydonald, s.f.), que ens permetrà comunicar amb el client de Telegram (vysheng, s.f.) instal·lat a la Raspberry Pi.

El mòdul de notificacions, bàsicament, estarà format per un *script* Python que s'executarà com un servei del S.O. Aquest *script* obrirà un client MQTT fent ús de la llibreria Paho, mitjançant el qual, rebrà tots els missatges que es produeixen al sistema. En produir-se qualsevol missatge dins el sistema, el mòdul de notificacions el rebrà, el processarà i si és un missatge que requereix de notificació, l'enviarà de forma instantània.

El desenvolupament d'aquest mòdul utilitza les llibreries:

- Paho MQTT Client
- Pytg

En primer lloc, amb la segona llibreria, s'iniciarà un client de telegram, el qual utilitzarem per enviar missatges quan sigui necessari.

A continuació, mitjançant la primera de les llibreries s'inicia un client MQTT que executarà el llaç de comprovació (*loop()*) de forma continua. Quan es rebin notificacions MQTT el client executarà la funció donada, que comprovarà quin tipus d'esdeveniment s'ha produït i enviarà el missatge corresponent.

```
#!/usr/bin/env python
print('start telegram script execution')

import paho.mqtt.client as mqtt
import time
from pytg import Telegram

def on_message(client, userdata, message) :
  if str(message.payload) == 'TELEPHONE' :
   sender.send_msg(u"eudald", u"HI HA UNA TRUCADA DE TELEFON")
  elif str(message.payload) == 'DOOR1' :
    sender.send_msg(u"eudald", u"HI HA UNA TRUCADA A LA PORTA 1")
  elif str(message.payload) == 'DOOR2' :
    sender.send_msg(u"eudald", u"HI HA UNA TRUCADA A LA PORTA 2")
  elif str(message.payload) == 'BABY' :
    sender.send_msg(u"eudald", u"EL NADO ESTA PLORANT")
tg = Telegram(
        telegram="/home/pi/tg/bin/telegram-cli",
        pubkey_file="/home/pi/tg/tg-server.pub")

receiver = tg.receiver
sender = tg.sender

#creates mqtt client
client = mqtt.Client('RASPBERRY')
#connect to broker
client.connect('AUDITIVASSIST-PI')
#subscribe to topic
client.subscribe('AUDITIVASSIST/#')
#configure function callback
client.on_message=on_message
#call devices
client.publish('AUDITIVASSIST','DEVICES')
#loop
while 1:
  client.loop()
  time.sleep(1)
```

Il·lustració 21: Codi mòdul de notificacions mòbils

Finalment, per instal·lar l'*script* com un servei del sistema operatiu, es crearà un dimoni a *systemd*. Per fer-ho, s'ha de crear un fitxer amb extensió .service al directori /etc/systemd/System. En aquest cas, el servei l'anomenarem auditivassist_tg.service.

```
        Fichero: /etc/systemd/system/auditivassist_tg.service

[Unit]
Description=AUDITIVASSIST TELEGRAM SERVICE
After=network.target
StartLimitIntervalSec=0

[Service]
Type=simple
Restart=always
RestartSec=1
User=pi
ExecStart=/home/pi/Desktop/python/telegram.py

[Install]
WantedBy=multi-user.target
```

Il·lustració 22: Codi configuració dimoni

Un cop generat el fitxer, habilitem el servei amb la següent comanda:

```
$ sudo systemctl enable auditivassist_tg
```

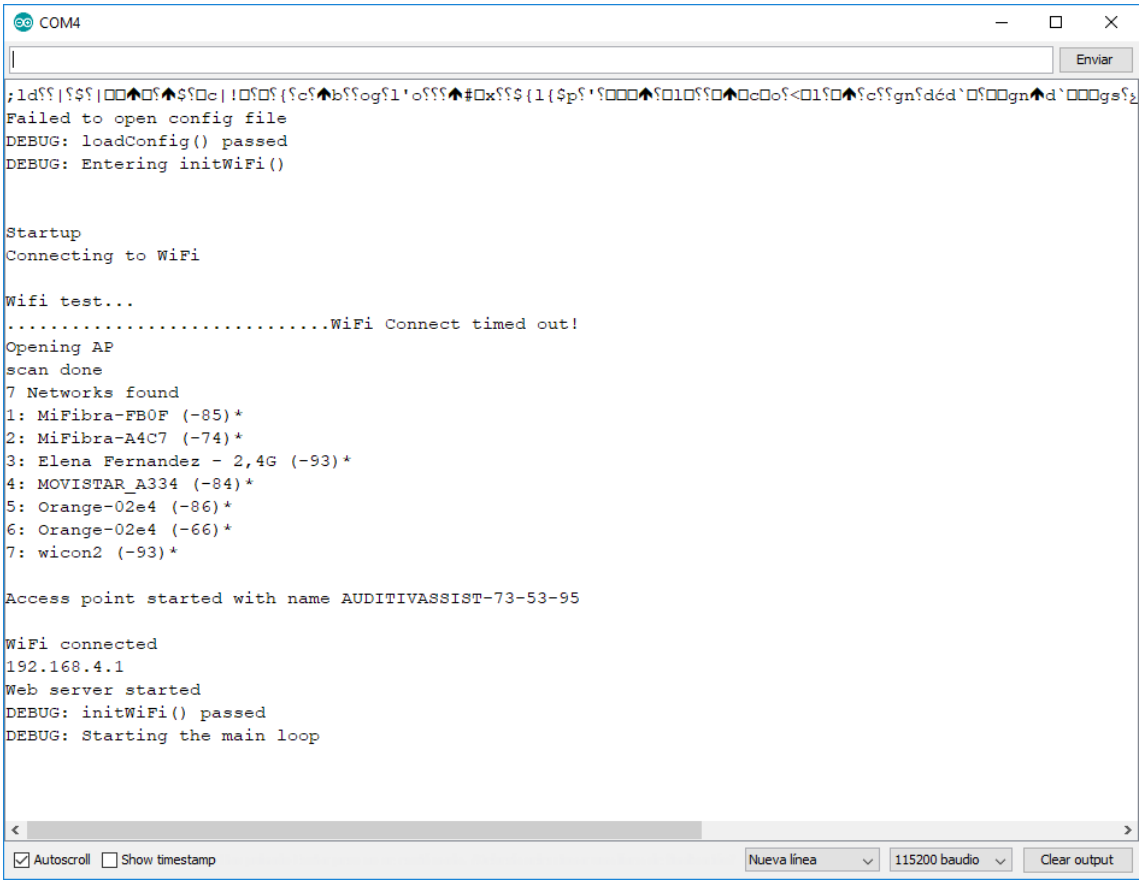
Ara ja es pot reiniciar el sistema i el servei s'inicia de forma automàtica.

6. Prototip i proves

6.1 Node captador de sons

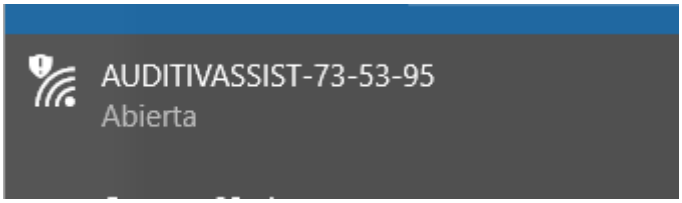
Un cop finalitzat el muntatge del maquinari del node i desenvolupat tot el codi necessari per aquest, iniciem les proves.

Començarem provant la configuració del node, quan aquest és nou i no te dades de cap connexió Wifi. Si connectem el node al PC i veiem mitjançant el terminal les sortides de text per debugar, on anirem seguint el procés que va fent el programa. Tal i com es pot veure a la següent figura, el node llegeix les dades de configuració guardades a memòria i intenta connectar-se a la xarxa WiFi configurada, com aquesta no hi és, el procés de connexió falla i automàticament s'inicia la configuració del node en mode AP (*Access Point*). Un cop acabada la configuració del mode AP, configura el servidor web i inicia l'execució normal del programa, a l'espera de sol·licituds de connexió i de recursos web.



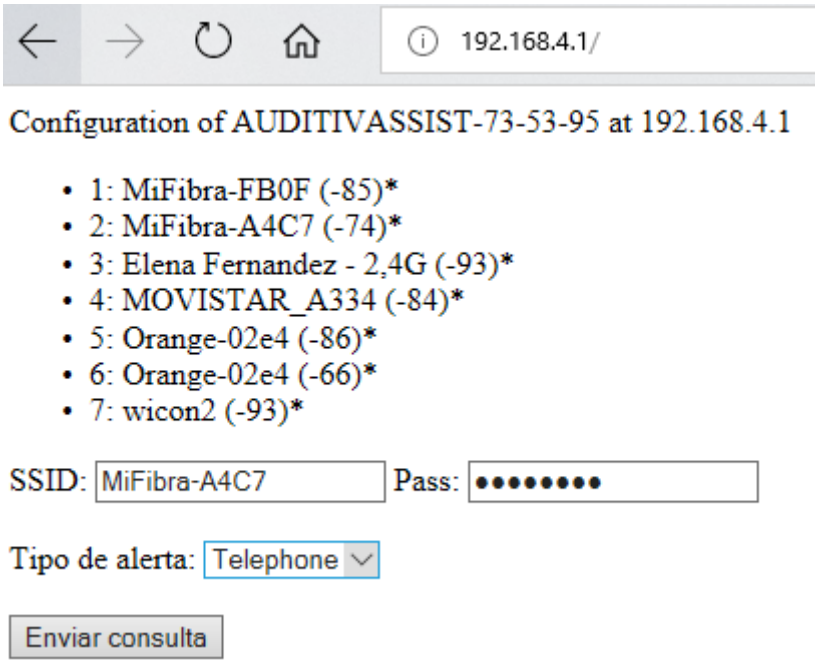
Il·lustració 23: Sortida terminal node sense configurar

Ara si obrim la llista de xarxes disponibles des de l'ordinador, veurem que hi ha una xarxa disponible nova:



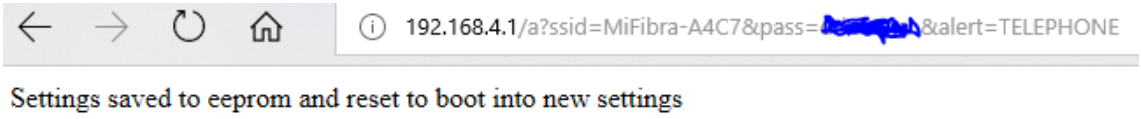
Il·lustració 24: Xarxa WiFi node en mode AP

Ens hi connectem i obrim l'explorador, obrint la IP que porten els nodes per defecte (192.168.4.1):

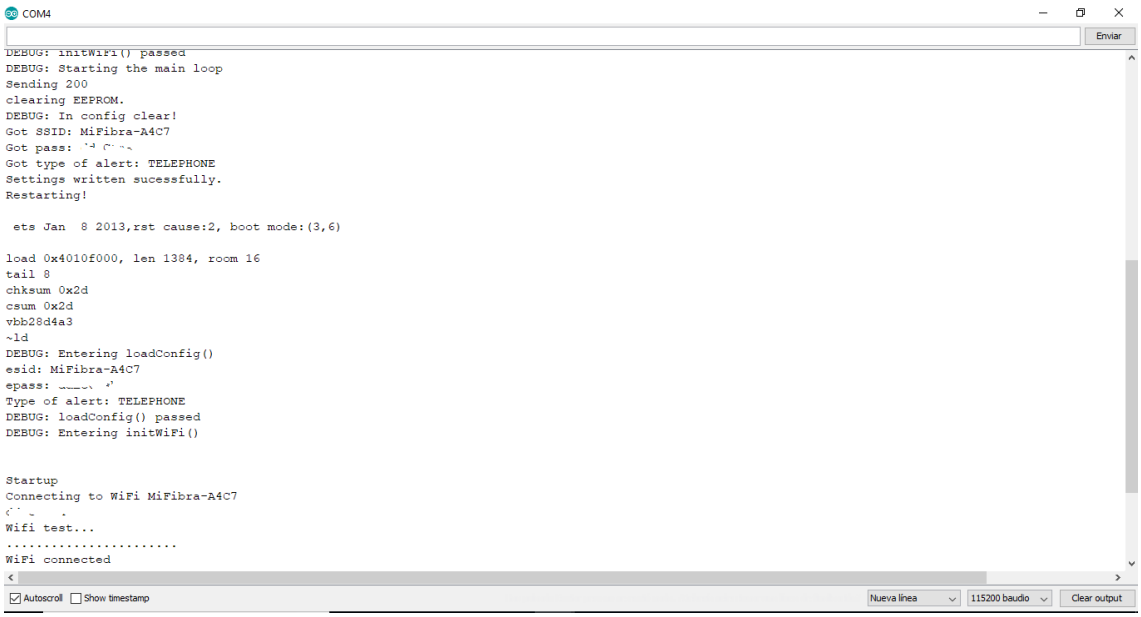


Il·lustració 25: Pàgina de configuració node

Guardem les dades, i obtenim la següent sortida per el terminal:



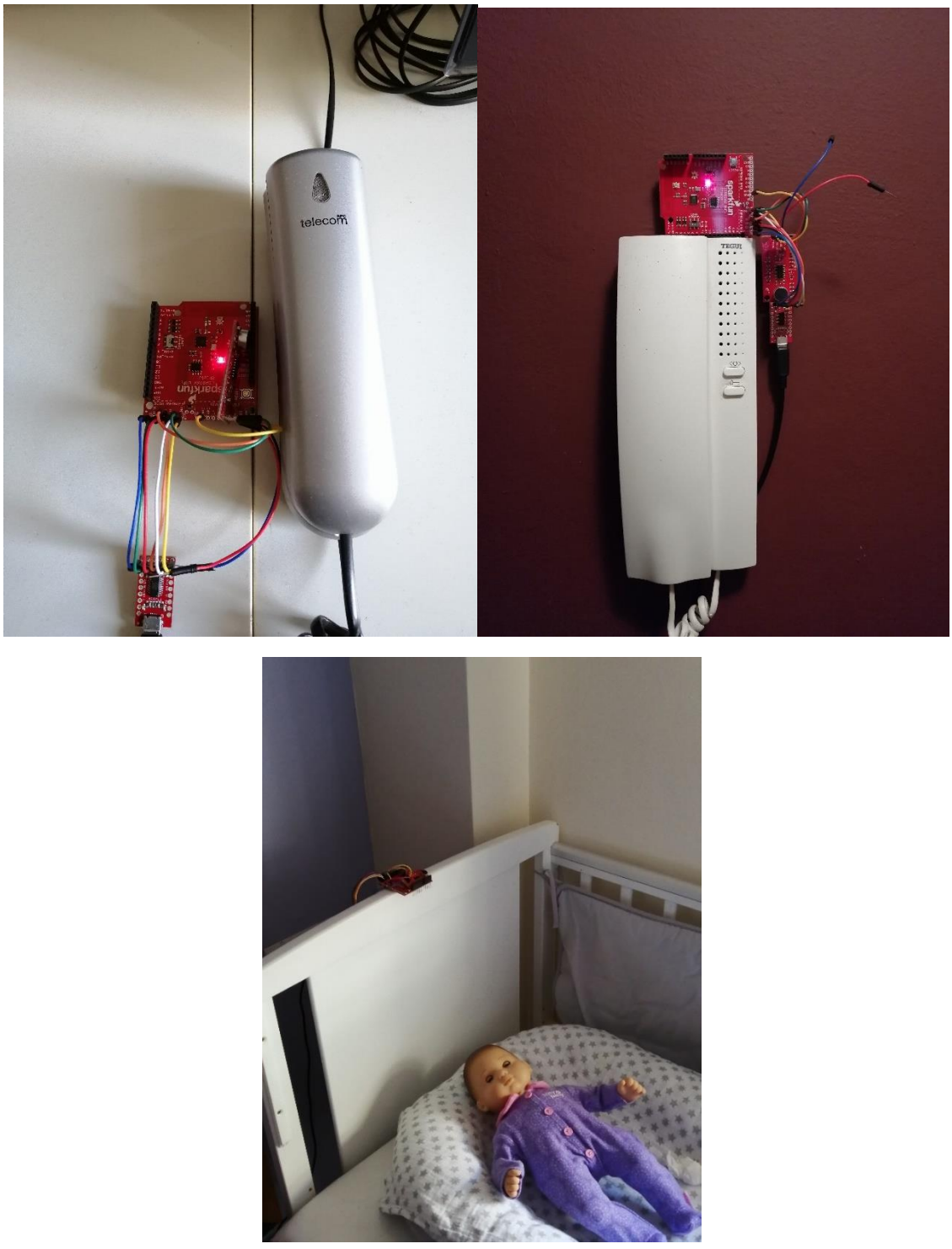
Il·lustració 26: Confirmació de dades emmagatzemades



Il·lustració 27:Sortida terminal connexió WiFi i MQTT satisfactòria

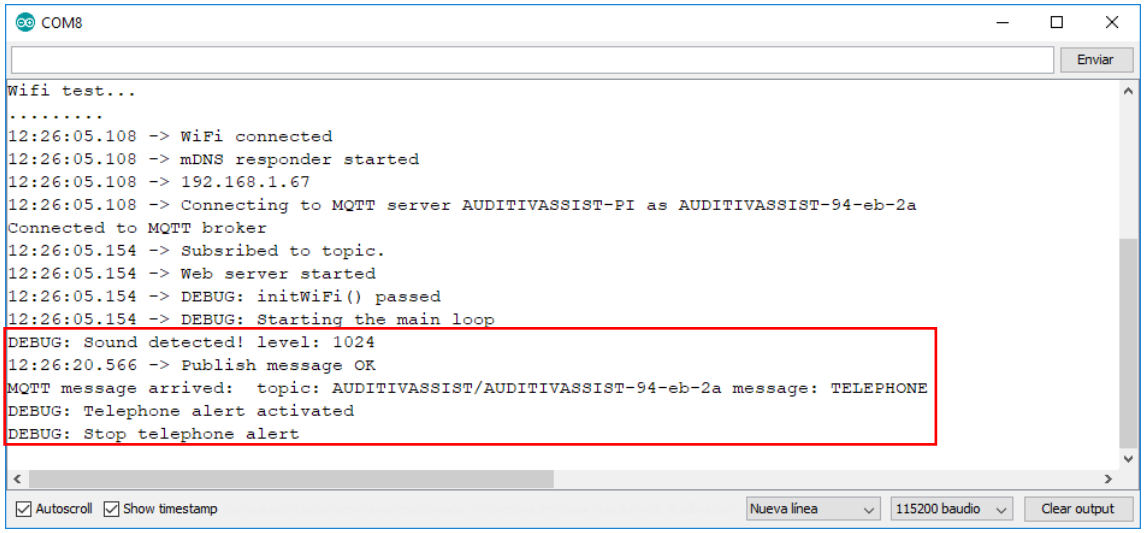
En aquest punt, ja tenim el node correctament configurat, amb les dades de la xarxa WiFi a la que s'ha de connectar, i el tipus d'alerta que emetrà en detectar algun so. Aquestes dades s'emmagatzemen a la memòria no volàtil del node, de manera que no es perdin mai.

Ara passem a provar la detecció de sons i la retransmissió de avisos a través del client MQTT. En primer lloc, situem el node a un radi inferior a 1 metre de la font de sons a detectar, tal i com es pot veure a les següents figures:



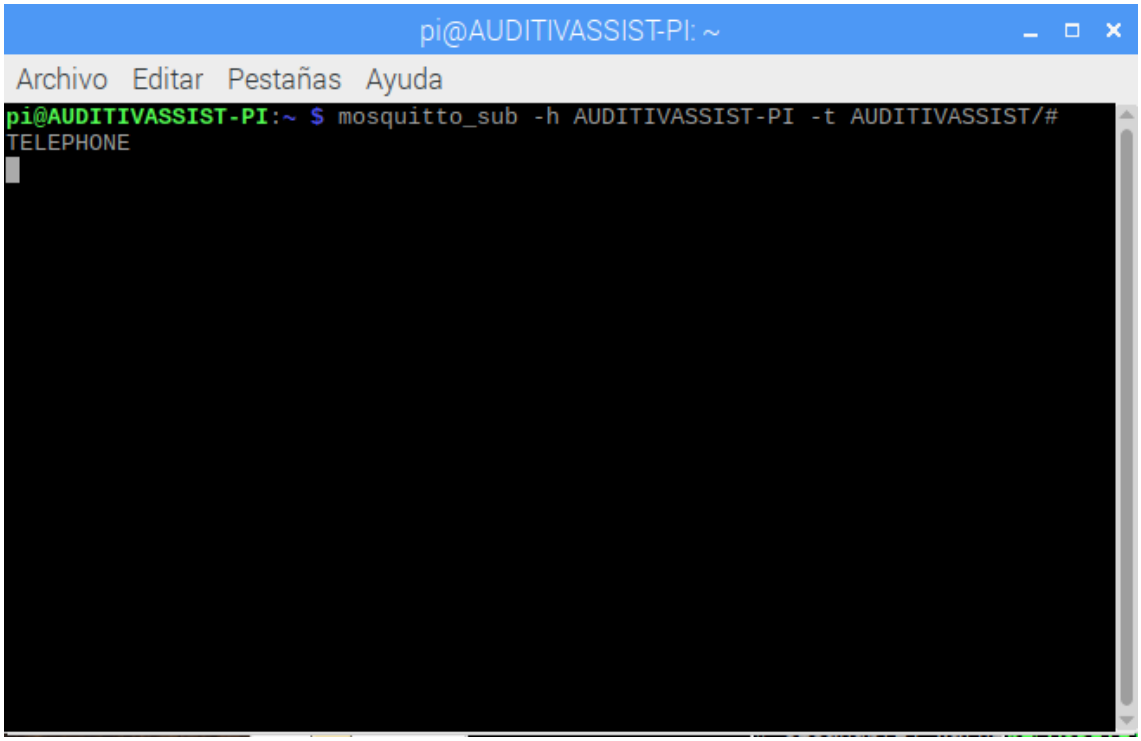
Il·lustració 28: Muntatge del node captador de sons respecte les diferents fonts de sons a detectar

Un cop realitzat el muntatge del node a una distància no superior a 50 centímetres de les diferents fonts de so, activem el so i comprovem la detecció i l'enviament de missatges MQTT.



Il·lustració 29:Sortida terminal detecció de so i notificació MQTT

Comprovem des del node coordinador que el missatge MQTT s'ha retransmès correctament:



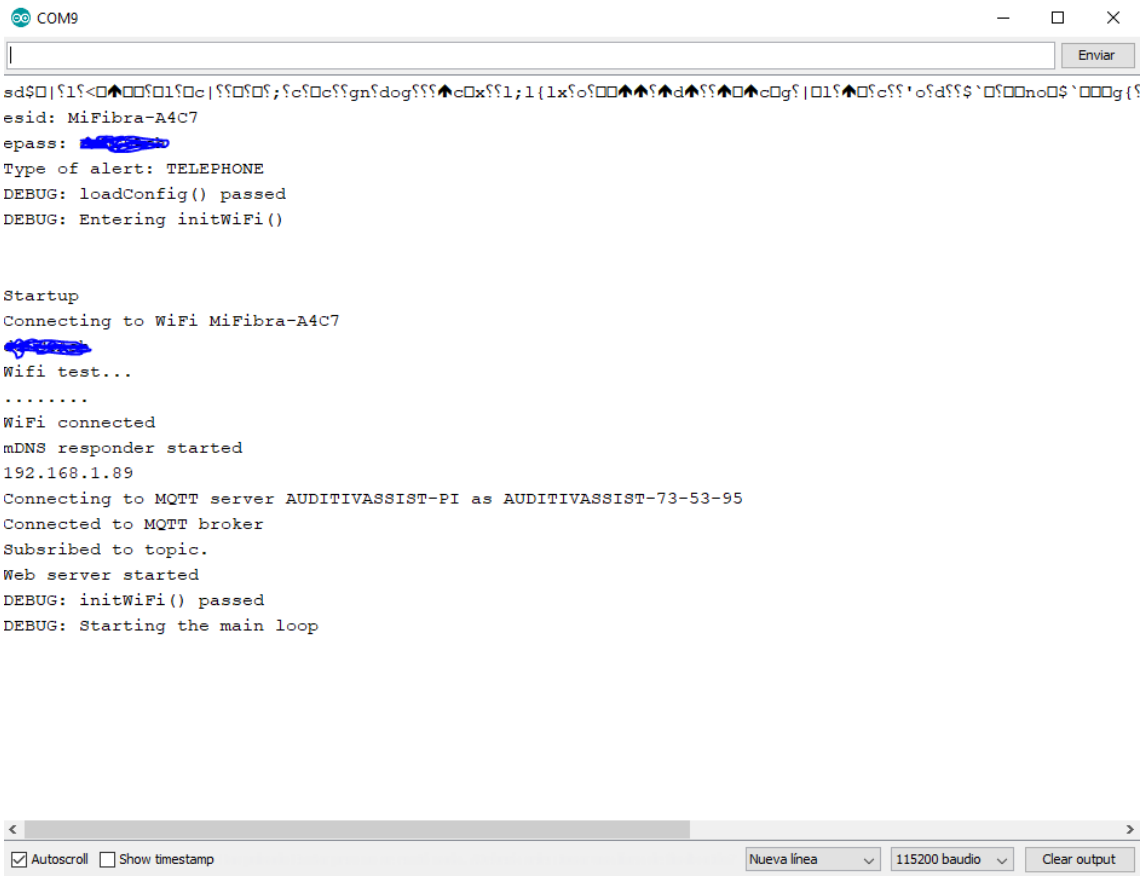
Il·lustració 30: Recepció de notificacions MQTT al servidor Mosquitto

6.2 Node actuador

Tal i com ja hem avançat anteriorment, el procés de configuració del node es totalment idèntic tant per al node detector de sons com per al node actuador, per això, en aquest apartat, donarem per vàlides les proves de configuració fetes amb el node captador de sons. Per tant, passem directament a l'apartat de funcionalitat.

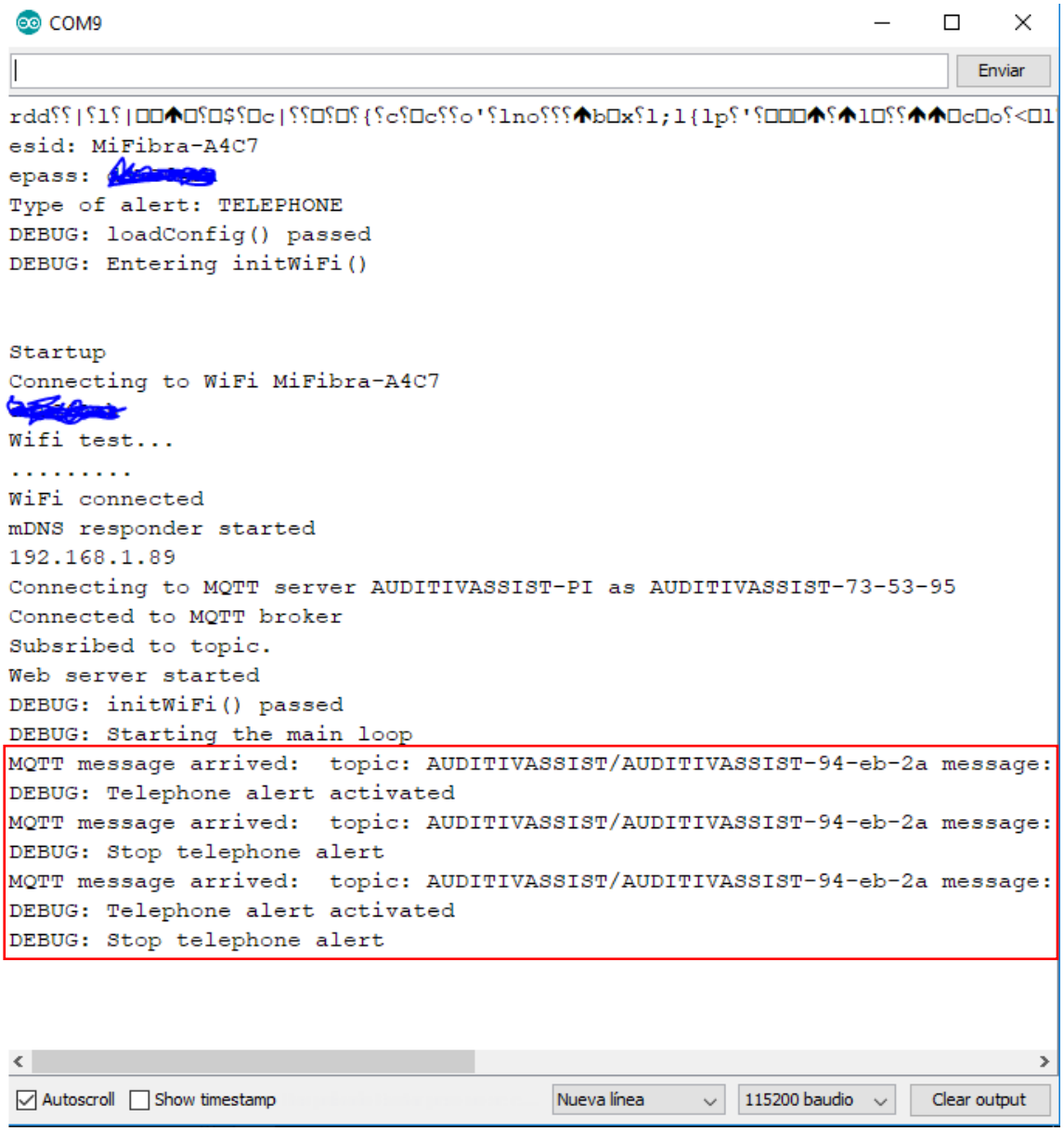
Per fer les proves, tot i que aquesta targeta pot anar alimentada a 220VAC i es pot connectar a la seva sortida una llum de 220VAC amb una càrrega màxima de 2A, les proves les farem alimentant la targeta mitjançant un cable USB i supervisant l'actuació de la sortida mitjançant un led que integra la mateixa placa, d'aquesta forma podrem rebre dades de la targeta al terminal.

Així doncs, connectem la placa al PC i obtenim el següent registre:



Il·lustració 31:Sortida terminal connexió WiFi i MQTT satisfactòria

Ara passem a reproduir un so, perquè el detecti el node detector i retransmeti el missatge pertinent, que rebrà el node actuador i realitzarà la seqüencia corresponent, d'activació i desactivació de la seva sortida.



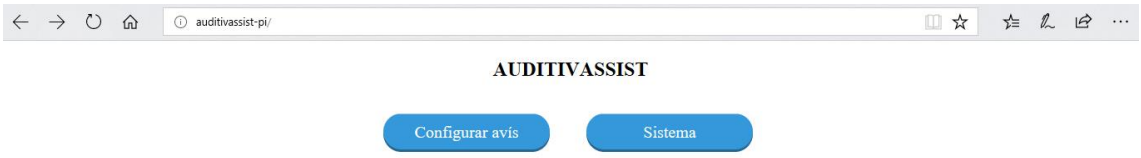
Il·lustració 32: Sortida terminal recepció notificacions MQTT

6.3 Node coordinador

El node coordinador té tres tasques importants, en primer lloc, fer de *broker* de les comunicacions MQTT, tasca que al llarg de les proves fetes amb els dos nodes anteriors, hem pogut veure en diverses ocasions com funciona correctament. En segon lloc, aquest node disposa d'un portal web, des d'on poder configurar avisos o supervisar o provar el sistema. I finalment, la darrera tasca és la d'enviar notificacions a telèfons mòbils.

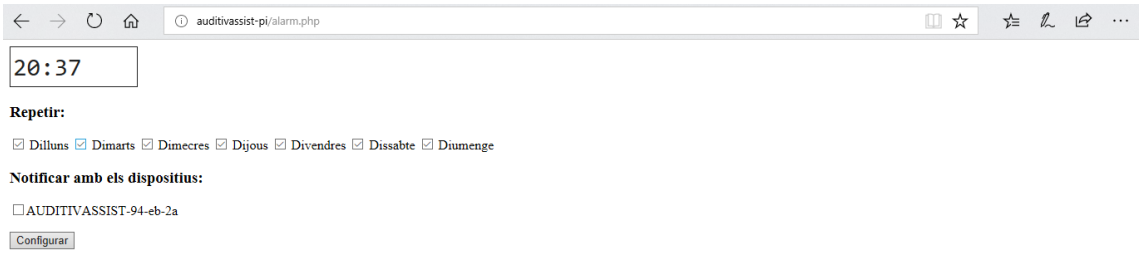
Donat que la primera de les tasques, ja ha estat provada intrínsecament a les proves dels nodes anteriors, ara passem a provar el portal web, des d'on configurar avisos, a mode de despertador, per exemple, o supervisar o provar el sistema.

Per accedir al portal web del sistema, des de qualsevol dispositiu connectat a la mateixa xarxa, i mitjançant un explorador web, s'accedeix a la direcció http://auditivassist-pi.



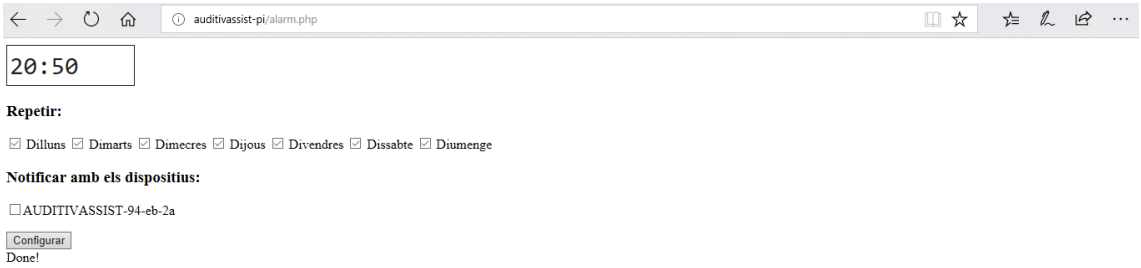
Il·lustració 33: Pàgina inicial del sistema

Des de la pàgina inicial, si s'accedeix a la configuració d'un avís trobem la següent pàgina.



Il·lustració 34: Pàgina de configuració d'alarmes

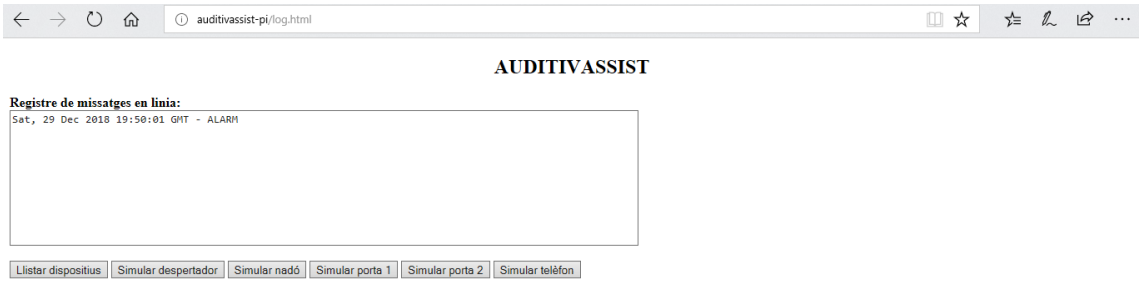
Des d'aquí es pot configurar una alarma seleccionant els dies de la setmana que es vol reproduir i l'hora a la que s'ha de fer l'avís així com els dispositius actuadors mitjançant els quals es vol reproduir l'alarma, ja que en aquest cas no tindria sentit reproduir-ho a totes les estances. Un cop feta la configuració, mitjançant el botó *Configurar* s'envia la configuració, i si aquesta es produeix de forma satisfactòria, apareixerà el missatge *Done!*.



Il·lustració 35: Configuració d'alarma satisfactòria

Per comprovar que el sistema, efectivament reprodueix l'alarma, utilitzarem la segona pàgina, la de Sistema. Des d'aquesta pàgina, es pot veure un registre, en línia, dels missatges que es produeixen al sistema, així com simular els diferents esdeveniments que es poden produir dins el sistema.

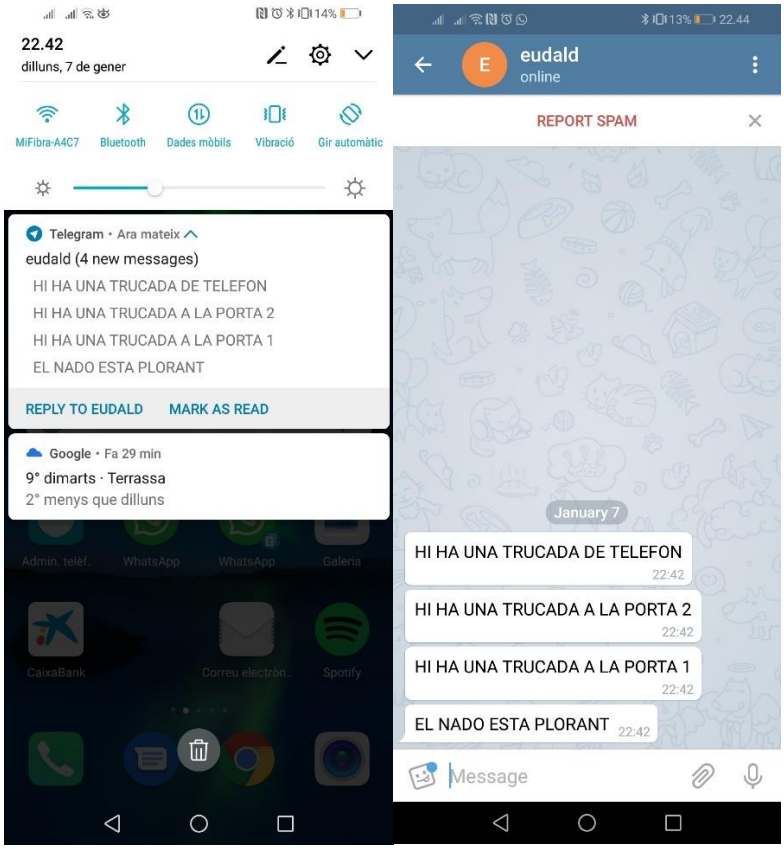
Tornant a la prova anterior, tal i com es pot veure a la següent figura, al registre de missatges podem veure el missatge MQTT enviat produït per l'esdeveniment programat anteriorment. Val a dir que la diferència horària entre l'esdeveniment programat i el registrat es deu únicament a la configuració de la zona horària.



Il·lustració 36: Pàgina de sistema

Finalment, la darrera prova a fer es la del mòdul de notificacions mòbils. Per fer la prova, utilitzarem els botons de prova que tenim a la pàgina de sistema, amb ells generarem els quatre esdeveniments que es notifiquen a través del mòbil (trucada de telèfon, trucada a la porta 1 i 2, i nadó plorant), tal i com es mostra a la següent captura:

Il·lustració 37: Esdeveniments generats per al test de notificacions al mòbil



Il·lustració 38: Notificació d'esdeveniments al telèfon mòbil

7. Conclusions

Un cop finalitzat el projecte, cal fer un anàlisi del que ha estat l'execució, els resultats obtinguts i l'assoliment o no dels objectius fixats a l'inici d'aquest.

En primer lloc, si ens fixem en la planificació, val a dir que el compliment d'aquesta ha estat parcial, ja que sobretot a la segona fase, la de desenvolupament, hi ha hagut falta de recursos que ha originat que tant les proves parcials com les finals s'hagin vist endarrerides. Tanmateix, val a dir que l'objectiu final s'ha complert.

Durant l'execució del projecte també hem patit algunes modificacions respecte la idea original, degut principalment al descobriment i l'adquisició de coneixement del microprocessador ESP8266 en la fase d'estudi, ja que originalment el projecte estava pensat per desenvolupar íntegrament amb la plataforma d'Arduino. Això, ens ha permès minimitzar de forma considerable els costos del prototip i del desenvolupament d'aquest, ja que hem pogut treballar amb un maquinari amb totes les funcionalitats integrades, evitant així, la implementació de comunicacions entre targetes, i que a demés també disposa de ports d'E/S, permetent-nos connectar també el sensor de so.

En quant a objectius, a l'inici del projecte es van definir els següents objectius:

- Desenvolupament d'un node captador de sons amb taxes de detecció superiors als 98%, per a sons provinents de fonts a una distància inferior als 50 cm.
- Desenvolupament d'un node actuador amb capacitat per controlar càrregues, tant resistives com inductives, de fins a 2 Ampers.
- Desenvolupament d'un mòdul de notificacions a telèfons mòbils.
- La latència total de qualsevol notificació no ha de superar en cap cas els 5 segons de retard, entre l'esdeveniment i la darrera notificació a l'usuari.
- L'usuari, a través del portal web del sistema, ha de poder activar alarmes que se li notifiquin a través de les llums, a mode de despertador.
- El cost final del *pack* estàndard, format per 2 nodes detectors, 4 nodes actuadors i 1 node coordinador ha de ser inferior als 300€/*pack*.

Tal i com s'ha anat demostrant al llarg d'aquesta memòria, podem afirmar que s'han assolit tots els objectius marcats.

Dins d'aquest objectius, en cap cas, era el desenvolupament d'un producte comercial, sinó el prototip d'un producte que podria arribar a ser comercialitzat, ja que s'ha demostrat que tant econòmicament com tècnicament es viable.

Finalment, el projecte m'ha permès descobrir un microcontrolador nou, molt més enfocat al desenvolupament de tecnologies IoT, com és el ESP8266, que sense dubte presenta grans virtuts respecte Arduino, que encara té molta feina per fer en aquest sentit. Així com endinsar-me en el sector de les tecnologies IoT i aprendre tecnologies de desenvolupament d'interfícies Web, com pot ser els webservices, php o html.

Per tot això, crec que tot el procés de desenvolupament del projecte ha estat un procés molt enriquidor i apassionant, que m'ha permès posar en pràctica els coneixements adquirits al llarg de tots aquests estudis i adquirir coneixements nous.

7.1 Possibles millores

Al llarg del desenvolupament del projecte, a mesura que el producte va agafant forma, van apareixent idees noves sobre funcionalitats o millores que es podrien incorporar el producte a continuació n'enumerem algunes:

- Desenvolupament del producte comercial, fent el disseny industrial del producte.
- Millora de les interfícies web.
- Incorporació d'una pàgina de configuració per la configuració complerta del mòdul de notificacions mòbils. Això inclou que s'han de poder crear, esborrar, editar i llistar els contactes als qui notificar-los les alertes.
- A nivell de detecció de sons, es podria explorar la idoneïtat d'utilitzar sistemes no tant de detecció de so, sinó de reconeixement d'aquest. De fet, es una via molt interessant que es va explorar, fent ús d'un motor d'IA sobre la Raspberry Pi, però el projecte marxava massa de l'àrea en que l'estàvem desenvolupant.
- Desenvolupament d'una APP per dispositius mòbils des d'on l'usuari pugui interactuar de forma més còmode amb el sistema, podent configurar-lo o rebre les seves alertes, sense dependre d'aplicacions de tercers. Això molt probablement milloraria la usabilitat del producte.

8. Bibliografia

Anònim. (6 / 11 / 2018). *Wikipedia*. Consultat el 7 / 11 / 2018, a https://es.wikipedia.org/wiki/Arduino

ARMTRONIX. (13 / 11 / 2018). *Repositori WiFi ESP8266 SSR Board*. Recollit de https://github.com/armtronix/Wifi-Triac-SSR

ARMTRONIX. (sense data). *ARMTRONIX*. Consultat el 15 / 11 / 2018, a http://armtronix.net/

Barragán, H. (sense data). *https://arduinohistory.github.io/*. Consultat el 9 / 11 / 2018, a https://arduinohistory.github.io/

Eclipse Foundation. (sense data). *Repositori Mosquitto*. Consultat el 04 / 11 / 2018, a https://github.com/eclipse/mosquitto

Eclipse.org. (sense data). *Paho*. Consultat el 2 / 12 / 2018, a https://www.eclipse.org/paho/

GEEKY THEORY. (sense data). *Tutorial raspberry Pi - Crear un servidor Web*. Consultat el 7 / 12 / 2018, a https://geekytheory.com/tutorial-raspberry-pi-crear-servidor-web

INE. (2008). *Encuesta de Discapacidad, Autonomía personal y situaciones de Dependencia (EDAD).* Instituto Nacional de Estadistica. Recollit de https://www.ine.es/prensa/np524.pdf

luckydonald. (sense data). *pytg - Python package that wraps around Telegram messenger CLI*. Recollit de https://github.com/luckydonald/pytg

Morel, B. (5 / 9 / 2017). *Creating a Linux service with systemd*. Consultat el 6 / 1 / 2019, a https://medium.com/@benmorel/creating-a-linux-service-with-systemd-611b5c8b91d6

Mosquitto.org. (sense data). *mosquitto.conf — the configuration file for mosquitto*. Consultat el 1 / 12 / 2018, a https://mosquitto.org/man/mosquitto-conf-5.html

MQTT.org. (sense data). *Frequently Asked Questions*. Consultat el 11 / 14 / 2018, a http://mqtt.org/faq

O'Leary, N. (sense data). *Repositori Client Arduino per MQTT*. Consultat el 05 / 11 / 2018, a https://github.com/knolleary/pubsubclient

Organització Mundial de la salut. (15 / 3 / 2018). *Sordera y pérdida de la audición.* Consultat el 29 / 9 / 2018, a http://www.who.int/es/news-room/fact-sheets/detail/deafness-and-hearing-loss

Raspberry Pi Foundation. (sense data). *Raspberry Pi*. Recollit de https://www.raspberrypi.org/

Visualfy. (14 / 11 / 2018). *Visualfy*. Recollit de https://www.visualfy.com/es/

vysheng. (sense data). *Repositori telegram-cli*. Recollit de https://github.com/vysheng/tg

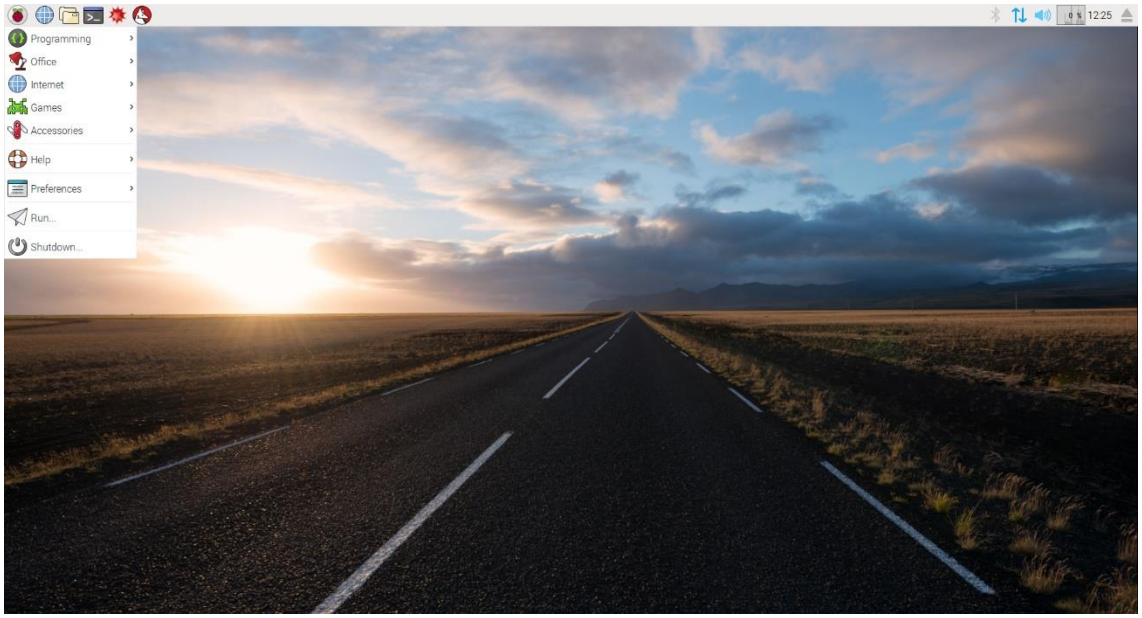
ANNEX 1. Instal·lació i configuració del node coordinador

Tal i com ja hem avançat anteriorment, el node coordinador és un node format per una Raspberry Pi i tot un seguit de programari que permet desenvolupar la tasca encomanda a aquest node.

Per instal·lar i configurar el node, en primer lloc hem de instal·lar el sistema operatiu a la Raspberry, en aquest cas, s'ha utilitzat el sistema operatiu oficial de Raspberry, Raspbian basat en una distribució de Linux.

Per instal·lar-lo, en primer lloc, necessitem disposar d'una targeta SD de, com a mínim, 8GB. Un cop tenim la targeta, podem descarregar-nos Raspbian de la pàgina oficial de Raspberry (Raspberry Pi Foundation, s.f.). En aquest cas, hem utilitzat la versió d'escriptori amb el programari recomanat. Un cop descarregat el SO en format *.zip, el descomprimim i mitjançat alguna eina d'escriptura d'imatges, gravem la imatge a la targeta SD.

Un cop tenim la imatge gravada a la targeta SD, la introduïm a la Raspberry Pi, i la iniciem.



Amb el sistema operatiu instal·lat i funcionant, passem a descriure la instal·lació de tot el programari necessari perquè funcioni el node coordinador. Comencem descrivint els passos seguits per configurar la Raspberry Pi perquè actuï com a servidor MQTT.

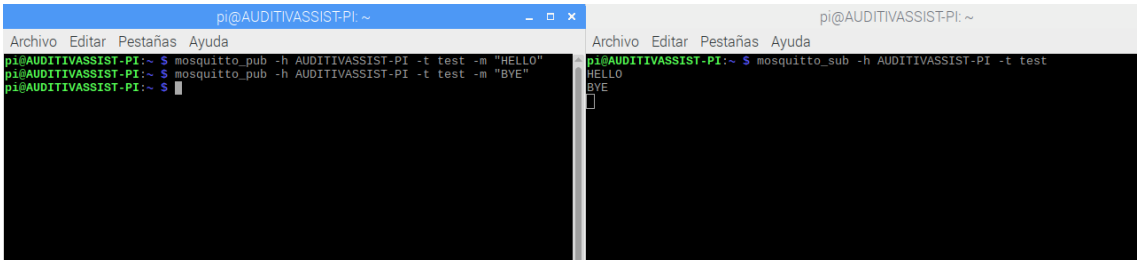
En primer lloc, obrim un consola i la primera comanda que farem serà per actualitzar el repositori.

```
$ sudo apt-get update
```

A continuació instal·lem el servidor Mosquitto amb la següent comanda:

```
$ sudo apt-get install mosquitto
```

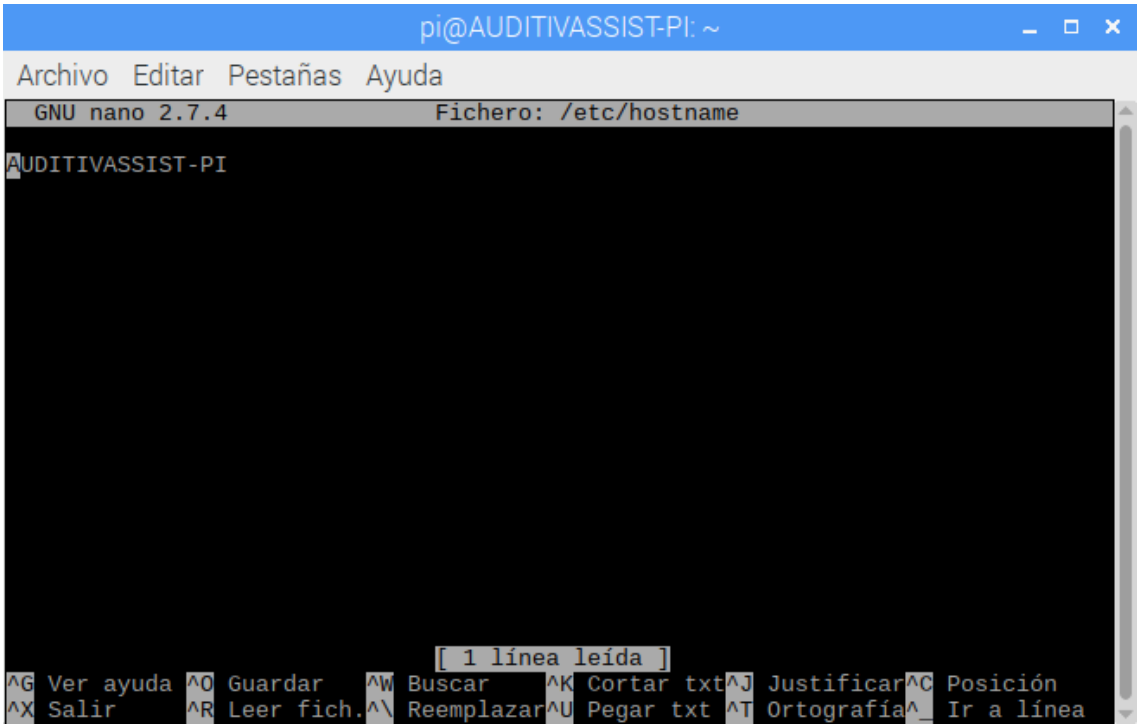
Un cop instal·lat el servidor, comprovem que està funcionant obrint dos terminals i utilitzant-ne un com a subscriptor al *topic* que publicarem des de l'altre terminal.



Així doncs, ja tenim el servidor MQTT instal·lat i funcionant de forma automàtica a la Raspberry Pi. Ara cal configurar un nom de *host* a la Raspberry Pi, perquè la resta de nodes no hagin de conèixer la direcció IP que s'ha assignat al node coordinador, per defecte, Raspbian porta el nom de Raspbery i el podem modificar amb la següent comanda:

```
$ sudo nano /etc/hostname
```

Modifiquem el nom i posem el *hostname* de AUDITIVASSIST-PI, d'aquesta forma, tots els nodes sempre buscaran aquest *host* a la xarxa, independentment de la IP que el servidor DHCP li assigni.



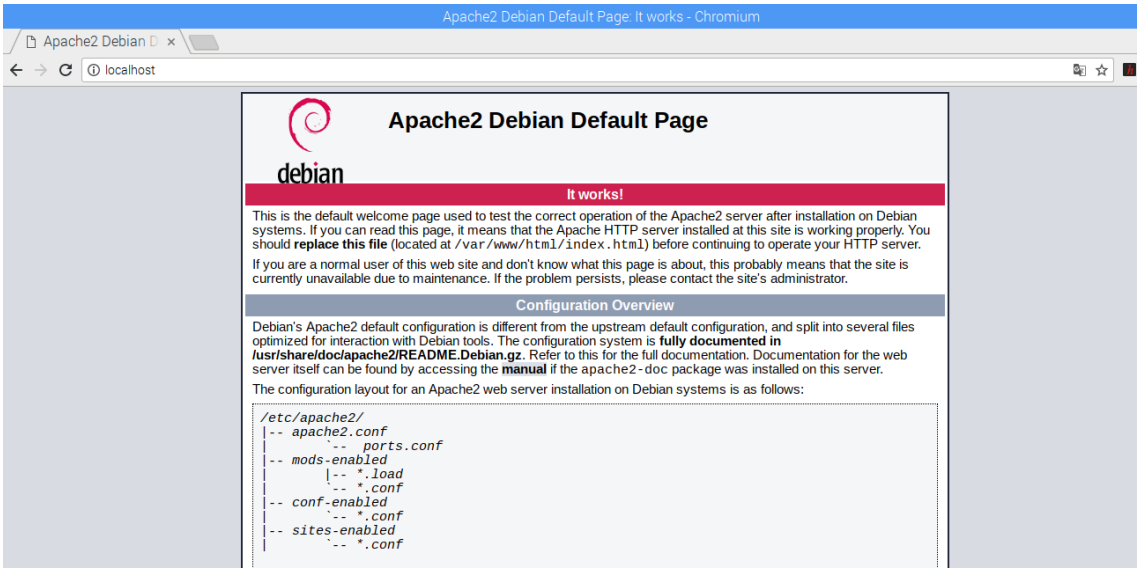
Per acabar amb la instal·lació del servidor de Mosquitto, s'ha de fer alguns ajustos a la configuració. Aquesta configuració es pot portar a terme mitjançant l'edició del fitxer Mosquitto.conf, existent a /etc/Mosquitto. Essencialment, aquesta configuració s'ha de duu a terme perquè el servidor accepti connexions mitjançant Websockets, tecnologia que utilitzarem per a la interacció entre el portal web i el sistema. A continuació mostrem el contingut del fitxer utilitzat al sistema:

```
pi@AUDITIVASSIST-PI: ~
Archivo  Editar  Pestañas  Ayuda
 GNU nano 2.7.4                    Fichero: /etc/mosquitto/mosquitto.conf

# Config file for mosquitto
#
# See mosquitto.conf(5) for more information.


max_queued_messages 200

message_size_limit 0

allow_zero_length_clientid true

allow_duplicate_messages false

#log_dest file /var/log/mosquitto/mosquitto.log

listener 1883

listener 3001 0.0.0.0
protocol websockets

autosave_interval 900

autosave_on_changes false

persistence true
persistence_location /var/lib/mosquitto
```

Ara passem a la instal·lació del servidor HTTP, per poder allotjar el portal web dins la Raspberry. Per fer-ho utilitzarem la següent comanda:

```
$ sudo apt-get install apache2
```

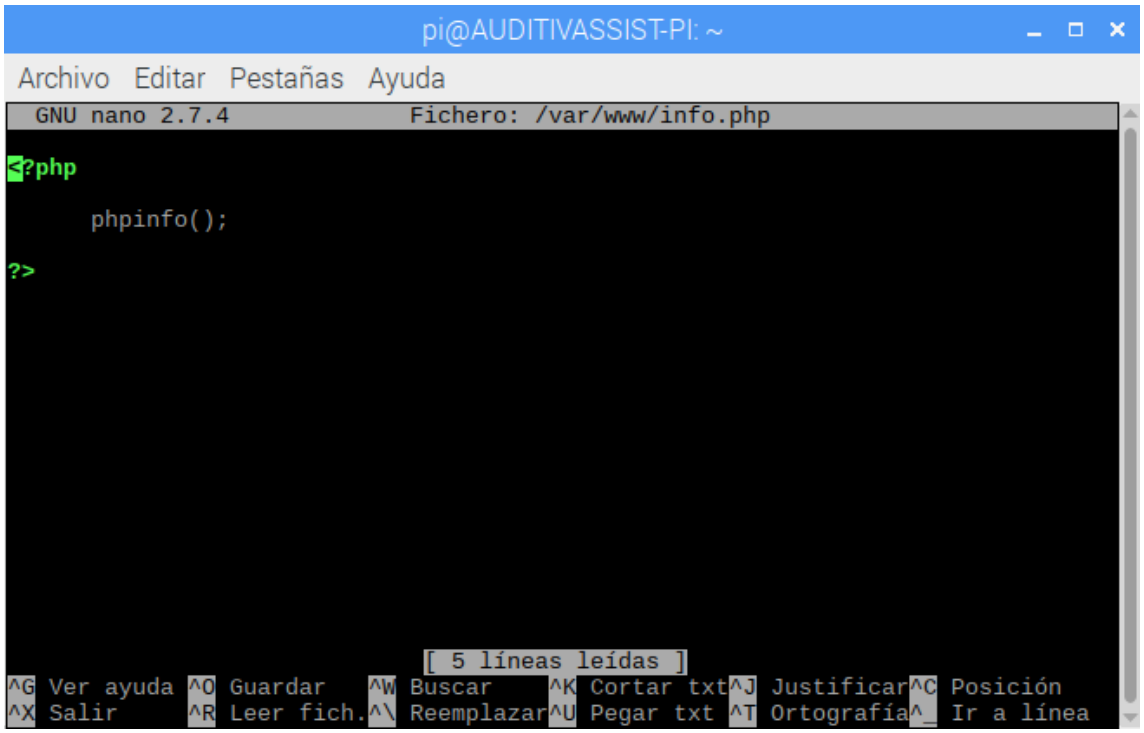
Un cop instal·lat el servidor, comprovem que funciona correctament obrint el navegador i connectant-nos contra el *host* local i ens apareix la següent pàgina confirmant que el servidor està correctament instal·lat i funcionant.



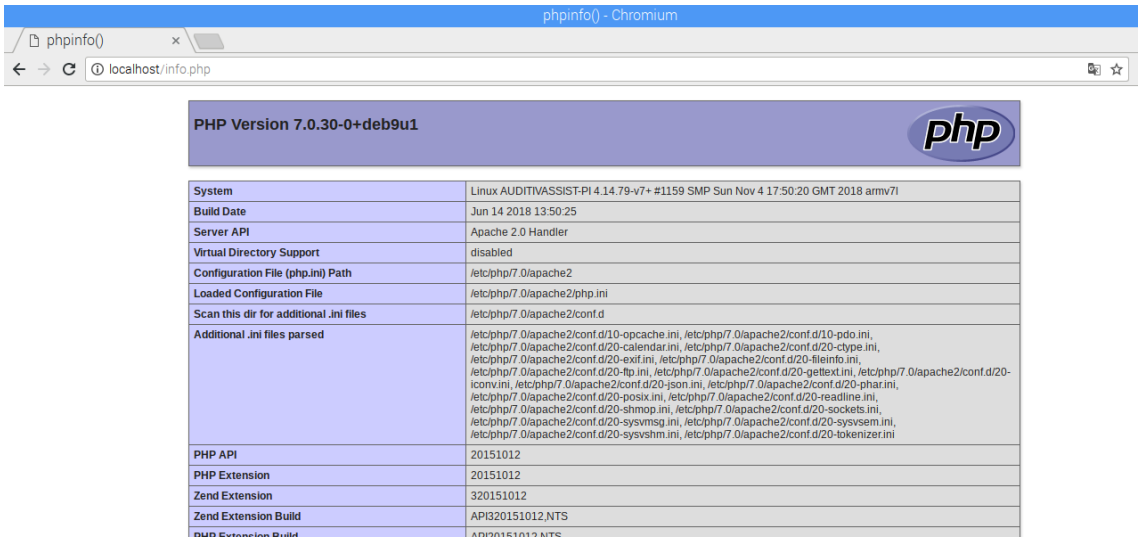
Tot seguit, procedim a instal·lar PHP per poder crear contingut dinàmic dins el nostre portal web, per fer-ho utilitzarem la següent comanda:

```
$ sudo apt-get install php5
```

Un cop instal·lat, per comprovar que s'ha instal·lat correctament es pot crear un fitxer *.php dins el directori del servidor web (/var/www/html), per exemple el podem anomenar info.php. Al fitxer hi podem escriure el següent codi:



I a través del navegador web, ens connectem al *host* local al fitxer que acabem de crear i si la instal·lació ha estat correcte hem d'obtenir la següent sortida:



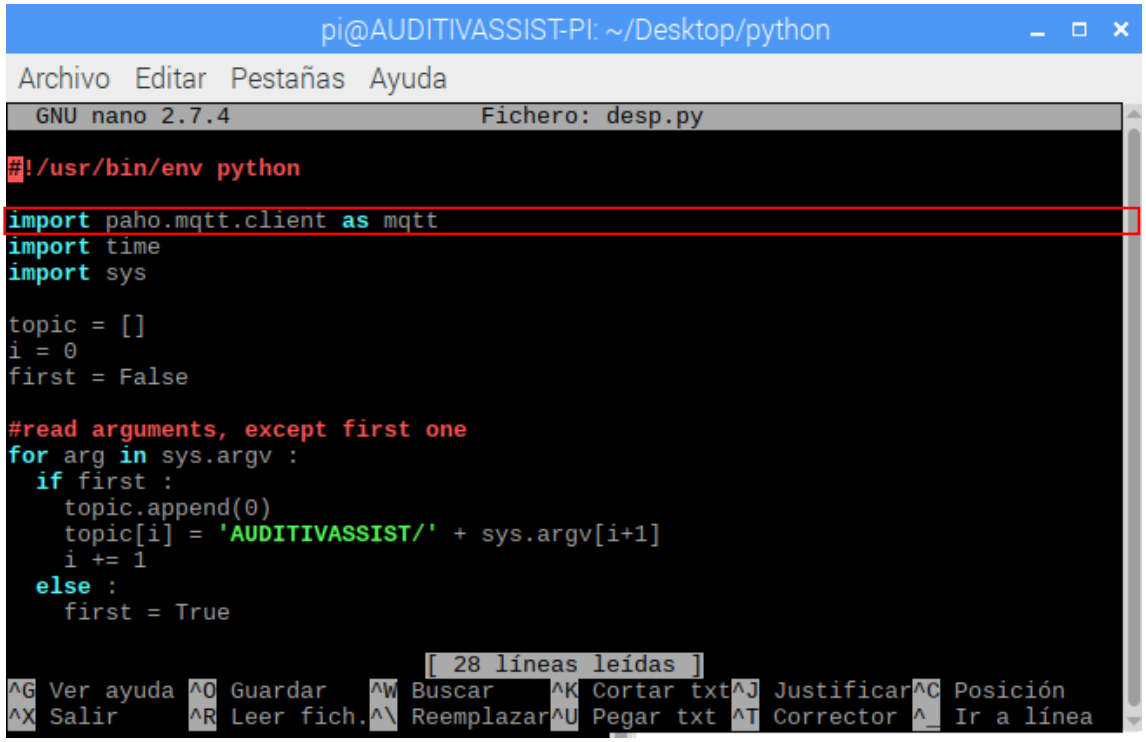
System	Linux AUDITIVASSIST-PI 4.14.79-v7+ #1159 SMP Sun Nov 4 17:50:20 GMT 2018 armv7l
Build Date	Jun 14 2018 13:50:25
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.0/apache2
Loaded Configuration File	/etc/php/7.0/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.0/apache2/conf.d
Additional .ini files parsed	/etc/php/7.0/apache2/conf.d/10-opcache.ini, /etc/php/7.0/apache2/conf.d/10-pdo.ini, /etc/php/7.0/apache2/conf.d/20-calendar.ini, /etc/php/7.0/apache2/conf.d/20-ctype.ini, /etc/php/7.0/apache2/conf.d/20-exif.ini, /etc/php/7.0/apache2/conf.d/20-fileinfo.ini, /etc/php/7.0/apache2/conf.d/20-ftp.ini, /etc/php/7.0/apache2/conf.d/20-gettext.ini, /etc/php/7.0/apache2/conf.d/20-iconv.ini, /etc/php/7.0/apache2/conf.d/20-json.ini, /etc/php/7.0/apache2/conf.d/20-phar.ini, /etc/php/7.0/apache2/conf.d/20-posix.ini, /etc/php/7.0/apache2/conf.d/20-readline.ini, /etc/php/7.0/apache2/conf.d/20-shmop.ini, /etc/php/7.0/apache2/conf.d/20-sockets.ini, /etc/php/7.0/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.0/apache2/conf.d/20-sysvsem.ini, /etc/php/7.0/apache2/conf.d/20-sysvshm.ini, /etc/php/7.0/apache2/conf.d/20-tokenizer.ini
PHP API	20151012
PHP Extension	20151012
Zend Extension	320151012
Zend Extension Build	API320151012,NTS

A continuació caldria instal·lar Python, però en aquest cas no cal, ja que el SO que s'ha utilitzat ja el conté, per el que no explicarem la seva instal·lació.

Ara passem a descriure la instal·lació de la llibreria Paho-MQTT, que utilitzarem als *scripts* Python per interactuar amb el servidor MQTT. En primer lloc, s'ha de fer la instal·lació mitjançant la següent comanda:

```
$ sudo pip install paho-mqtt
```

Amb aquesta comanda ja tenim la llibreria instal·lada, i per fer-ne ús, únicament hem d'incloure als *scripts* Python la següent línia:



I ja per acabar, falta instal·lar el client de Telegram per poder enviar notificacions a telèfons mòbils, així com la llibreria de Python que ens permet interactuar amb el client des d'*scripts* Python.

Per la instal·lació del client de Telegram, en primer lloc, clonem el repositori dins la Raspberry Pi, mitjançant la següent comanda, havent-nos situat, prèviament, dins la carpeta on volem clonar el repositori, en el nostre cas dins del mateix *home* de l'usuari "pi":

```
$ git clone --recursive https://github.com/vysheng/tg.git && cd tg
```

A continuació, abans de passar a la compilació, s'han d'instal·lar totes les llibreries amb dependències mitjançant la següent comanda:

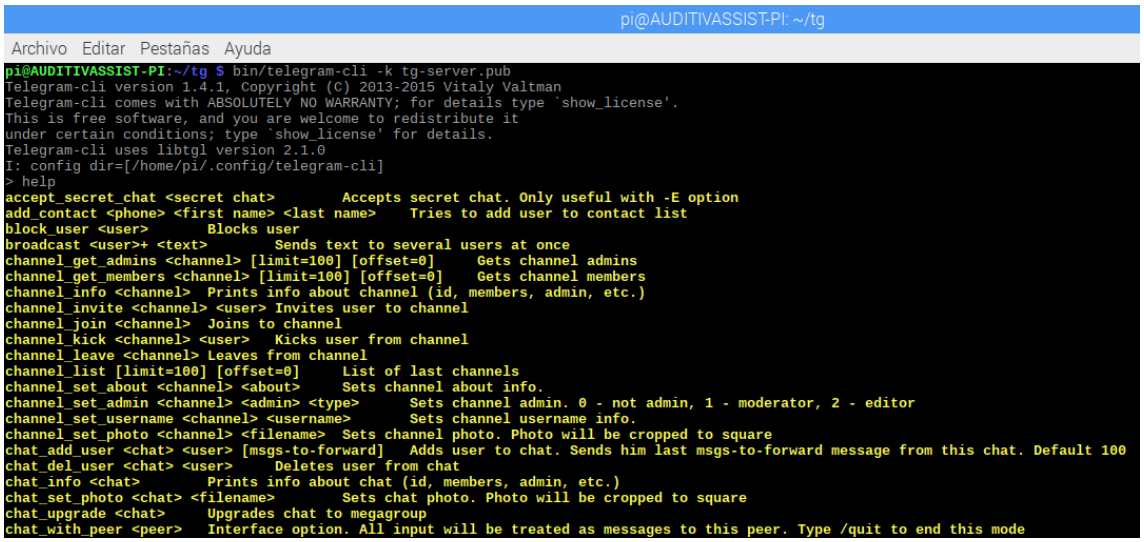
```
$ sudo apt-get install libreadline-dev libconfig-dev libssl-dev lua5.2
liblua5.2-dev libevent-dev libjansson-dev libpython-dev make
```

Ara si, passem a la compilació:

```
$ ./configure
$ make
```

Finalment, si el procés s'ha produït de forma satisfactòria, es pot executar el client amb la següent comanda:

```
$ bin/telegram-cli -k tg-server.pub
```



Per la instal·lació de la llibreria Python *pytg* que permetrà interactuar amb el client de telegram des dels *scripts* Python, utilitzarem la següent comanda:

```
$ sudo pip install pytg
```

Amb aquesta comanda ja tenim la llibreria instal·lada, i per fer-ne ús, únicament hem d'incloure als *scripts* Python la següent línia:

pi@AUDITIVASSIST-PI: ~

Archivo Editar Pestañas Ayuda

GNU nano 2.7.4 Fichero: /home/pi/Desktop/python/telegram.py

```
#!/usr/bin/env python
print('start telegram script execution')

import paho.mqtt.client as mqtt
import time
from pytg import Telegram

def on_message(client, userdata, message) :
  if str(message.payload) == 'TELEPHONE' :
   sender.send_msg(u"personal", u"HI HA UNA TRUCADA DE TELEFON")
  elif str(message.payload) == 'DOOR1' :
    sender.send_msg(u"personal", u"HI HA UNA TRUCADA A LA PORTA 1")
  elif str(message.payload) == 'DOOR2' :
    sender.send_msg(u"personal", u"HI HA UNA TRUCADA A LA PORTA 2")
  elif str(message.payload) == 'BABY' :
    sender.send_msg(u"personal", u"EL NADO ESTA PLORANT")

tg = Telegram(
        telegram="/home/pi/tg/bin/telegram-cli",
        pubkey_file="/home/pi/tg/tg-server.pub")

receiver = tg.receiver
sender = tg.sender

#creates mqtt client
client = mqtt.Client('RASPBERRY')
```

^G Ver ayuda ^O Guardar ^W Buscar ^K Cortar txt ^J Justificar ^C Posición
^X Salir ^R Leer fich. ^\ Reemplazar ^U Pegar txt ^T Corrector ^_ Ir a línea