Treball final de màster

Estudi: Màster en Enginyeria Informàtica

Títol: Disseny i implementació d'un sistema de monitorització de partícules contaminants en ciutats

Document: Memòria

Alumne: Miquel Farreras Casamort

Tutor: Lluís Pacheco Valls
Departament: Arquitectura i Tecnologia de Computadors
Àrea: Arquitectura i Tecnologia de Computadors

Convocatòria (mes/any): Juny/2019

Índex

1. Introducció i objectius

1.1. Introducció

El canvi climàtic ha passat de ser una idea llunyana a convertir-se en un fet, cada vegada tenim menys pluges i les mitjanes de temperatura són més elevades que en èpoques anteriors. Gran part de la culpa la tenen les activitats humanes que generen gasos contaminants i d'efecte hivernacle.

Les ciutats són uns grans focus de contaminació que cal mesurar, per tal de determinar, en temps real, quina contaminació estan respirant els seus habitants. Aquestes dades haurien de ser públiques i accessibles per a qualsevol ciutadà, ja que tothom té dret a conèixer la qualitat de l'aire que respira. A més, en l'actualitat i en un futur molt pròxim, les administracions hauran d'actuar, segons decrets de la UE, en contra d'aquests episodis de contaminació, i per això és indispensable tenir aquesta informació en temps real.

Per tant, a causa de l'impacte que té en la nostra salut i en el futur del nostre planeta, la contaminació està en el punt de mira de les administracions. Així entre les necessitats actuals de les ciutats, la monitorització de les partícules i gasos contaminants esdevé cabdal per tal possibilitar una millor qualitat de vida per els seus ciutadans.

1.2. Motivacions

Els sistemes encastats són uns elements cada vegada més utilitzats en la nostra vida quotidiana i ens poden ajudar a controlar el nostre entorn per viure millor. A més, si es seleccionen bé, es tracta d'elements de relativament baix cost i amb facilitat d'implantació en l'entorn.

La contaminació és un tema molt important sobretot en les grans ciutats i és necessari que els habitants puguin conèixer l'aire que respiren. Aquesta possibilitat ens permetria evitar, per exemple, la pràctica d'esports a l'aire lliure en episodis d'alta contaminació, activitats a l'exterior amb col·lectius vulnerables (infants, persones d'edat avançada o embarassades entre altres) o permetre a les autoritats limitar l'accés o velocitat dels vehicles que accedeixin a l'àrea urbana per tal de mitigar aquests episodis d'alta contaminació a temps. La xarxa actual de

sensors professionals situats pel territori té un cost molt elevat, dóna poques dades significatives i no són en temps real.

La recepció de les dades recollides per aquest projecte en un servidor al cloud i el seu tractament es portarà a terme en el treball de final de Màster en Enginyeria Informàtica de l'alumne Robert Garcia Ventura.

A més, es va presentar la nostra idea al SmartCAT Challenge 2018 [1] i vam aconseguir el 3r premi, tenint en compte que competíem amb candidats d'start-ups molt més desenvolupades.

1.3. Propòsit

La idea és dissenyar una xarxa de sensors que permeti realitzar aquestes mesures en una ciutat, de manera que, un cop creada la infraestructura de software (configuració dels dispositius de lectura, creació del servidor, etc.), faci possible que un usuari es fabriqui el seu hardware amb el conjunt de sensors tenint els coneixements mínims. Es realitzaran diverses versions amb més o menys sensors per a diferents pressupostos/propòsits.

1.4. Antecedents

A l'assignatura de Projecte de Sistemes Encastats i Ubicus es va començar un projecte on es va connectar un sensor de partícules amb un computador IoT, que permetia l'enviament de dades a un servidor. No era un sistema escalable ni de fàcil muntatge (necessari soldar elements), no resistent a la intempèrie i amb la necessitat de ser més econòmic. Necessitava una millora en el procés de muntatge, configuració i eficiència en l'enviament de dades, a part de tenir registrats quins punts de lectura recollien dades per a monitoritzar el seu funcionament. Va servir, però, com a al·licient per a continuar amb aquesta idea.

Amb aquest projecte es pretén realitzar l'arquitectura òptima per portar a terme la creació de la xarxa de sensors desitjada amb un cost mínim i una fiabilitat més elevada.

1.5. Objectius

Dissenyar i implementar un entorn anomenat RoMi per mesurar la contaminació de l'aire de les ciutats a partir d'una xarxa de sensors. Donada la complexitat del projecte es dividirà en

dos. El primer (aquí presentat) centrat en el disseny dels sensors, la configuració i la seva instal·lació. El segon centrat en la recopilació i anàlisi de les dades obtingudes de la xarxa de sensors.

1.6. Abast

Per assolir el nostre objectiu caldrà:

- Dissenyar i implementar un dispositiu a partir de sensors, ordinadors encastats i altres components capaç de recollir dades de contaminació amb un cost relativament baix. Es faran prototips resistents i s'afegiran els sensors que puguin ser necessaris, com ara d'altres gasos, meteorologia, etc. Fins ara només tenim sensor GPS i sensor de partícules PM2.5 i PM10.
- Programar el dispositiu de forma que reculli la informació desitjada per poder ser processada posteriorment. Programació des de zero per a eliminar errors del inici del projecte, tant de l'enviament de dades i com l'execució de les lectures.
- Preparar l'enviament de dades mitjançant Wi-Fi, 4G o altres tecnologies.
- Determinar les ubicacions dels punts de lectura. Aquestes poden ser fixes, com ara edificis, o mòbils amb un sensor GPS, situats en vehicles.
- Analitzar la precisió dels sensors respecte un sensor de partícules professional.
- Crear una aplicació per Android on poder configurar cada sensor i consultar lectures.
- Integrar el treball realitzat amb el projecte que recopila i analitza les dades.

2. Estudi de viabilitat

2.1. Recursos necessaris

Necessitem muntar una infraestructura amb el hardware bàsic per a tenir un punt fix o mòbil amb lectures dels gasos contaminants i qualitat de l'aire. Això inclou:

- Computador per a obtenir les dades i enviar-les a un servidor on tractar-les.
- Conjunt de sensors ambientals per a mesurar els paràmetres més rellevants de la qualitat de l'aire.
- Sensor GPS en el cas del sensor mòbil per tal de recollir els punts del recorregut.
- Recipient/protecció dels components respecte les inclemències meteorològiques.

També és necessari poder configurar el computador segons els temps de lectura, els sensors connectats, les coordenades de la ubicació, etc., per això crearem una aplicació bàsica per Android on passar aquests paràmetres per Bluetooth o Wi-Fi al computador del punt de lectura.

A part del hardware instal·lat en cada punt, també serà necessari un servidor on dipositar les dades i poder-les consultar des de qualsevol punt. En aquest aspecte, i com que es tracta d'un projecte on les dades les tractarà l'alumne Robert Garcia Ventura en el seu Treball de Final de Màster en Enginyeria Informàtica, la API es farà en el seu treball i per tant la configuració del servidor on es pujaran les dades també serà feta en aquest altre treball.

També serà necessària una certa mà d'obra per a muntar els sensors, l'objectiu seria poder fer que aquest sistema sigui de fàcil muntatge per als usuaris, com ara escoles, instituts o aficionats. Això seria possible una vegada estigui preparada la configuració de tot el software relacionat amb la lectura i enviament de dades dels sensors, sempre que busquem solucions que permetin el muntatge sense soldar (reduint molt el grau de complexitat).

2.2. Viabilitat econòmica

Econòmicament hem de tenir en compte el cost dels components de hardware per a cada punt de lectura (es detallen les decisions tècniques i econòmiques en l'apartat *Estudis i decisions*):

Component	Preu per comanda	Enviament	Unitats per comanda	Unitats per caixa	Cost per caixa
Sensor MQ135	0,93 €	1,29 €	1	1	2,22 €
Sensor MQ7	0,93 €	1,28 €	1	1	2,21 €
Sensor DHT11	0,56 €	0,53 €	1	1	1,09 €
Sensor HPMA115S0	17,07 €	4,24 €	1	1	21,31 €
Conversor ADS1115	2,59 €	0,35 €	1	1	2,94 €
Raspberry Pi Zero WH	14,60 €	0 €	1	1	14,60 €
Targeta microSD 8GB	10,10 €	1,10 €	1	1	11,20 €
Acoblaments ràpids de cables	0,68 €	0 €	10	2	0,14 €
Cable Molex Picoblade 8 pins 1.25mm separació (½ per caixa)	1,72 €	1,15 €	10	0,5	0,14 €
Cables per a protoboard mascle-femella i femella-femella	1,77 €	0 €	40	20	0,89 €
Cinta de doble cara 3m + 10 mm (màx 0,25 m per caixa)	0,86 €	0 €	1	0,25	0,22 €
Caixa 20x10x7 cm	5,33 €	0 €	1	1	5,33 €
Resistència 10k Ohm	0,48 €	1,25 €	100	1	0,02 €
Protoboard 400 pin	0,86 €	1,16 €	1	1	2,02 €
Alimentador micro USB	4,08 €	0 €	1	1	4,08 €
TOTAL					**68,40 €**

Per tant, vist el cost de cada sensor, és factible fer-ne una xarxa per les ciutats, sobretot si es compara amb el cost d'altres opcions com Bettair (https://bettaircities.com/) on cada sensor fix (amb bateries) costa aproximadament 4000 € i s'ha de reemplaçar cada any.

Les compres les realitzem finalment a Aliexpress, ja que després de mirar diverses webs on comprar els sensors eren exactament iguals, i bastant més cars. A Amazon, per exemple, ens oferien un sensor MQ135 per 6,95 € i a més l'enviament trigava el mateix que per Aliexpress.

Això és degut a que hi ha venedors intermediaris que també els venen des de Xina. La Raspberry Pi sí que va ser comprada a Kubii.es, al ser una entitat acceptada com a proveïdor de la UdG i tenir disponibilitat en aquell moment.

2.2.1. Recursos humans

Inicialment gran part de les hores es dediquen al muntatge del hardware, la configuració del sistema, lectura dels sensors, desar i enviar de les dades, en l'embolcall que permeti situar-los a l'exterior.

Una vegada depurada la solució, si es vol crear una xarxa de sensors en una ciutat, seria necessària la mà d'obra en els següents aspectes:

1. Muntatge dels components dins l'embolcall (2h)
2. Gravació del software a la targeta SD (10 minuts)
3. Configuració amb la app des de l'smartphone (5 minuts)
4. Col·locació del sensor a l'exterior connectat a un endoll de 220 V (variable, depenent de la ubicació i accés a font d'energia, cal comptar desplaçament i temps d'instal·lació).

Per al finançament s'obté una beca per part de la Facultat d'Econòmiques de la UdG, amb un import total de 985,64€ durant el període 15/9/2018 fins al 31/01/2019 per al disseny i configuració de la xarxa de sensors i preparació dels prototips.

2.2.2. Llicències de programari

Totes les llicències de programari utilitzades són open source, bé sota llicència GPL o LGPL, o bé sota llicència MIT [2], les quals són molt permissives i ens permeten modificar el software segons les nostres necessitats, mentre que el codi que es generarà per aquest projecte inicialment és privat.

Un cop ben depurada la idea s'haurà de decidir si es fa públic el codi generat, per exemple sota llicència GPL o MIT, i així fer una xarxa oberta de sensors on els centres educatius o qualsevol persona que ho desitgi pugui muntar-se un punt de lectura propi, compartint les lectures a la xarxa comuna de sensors.

2.3. Viabilitat tecnològica

Actualment aquest projecte està encarat en la connexió Wi-Fi i alimentació elèctrica fixa des d'endolls de 220 V o bé bateries externes de 5v USB. Hi ha diverses tecnologies que ens podrien permetre un treball futur bastant extens:

- Connexió 4G o LoRaWAN: En cas de que no hi hagi un punt d'accés Wi-Fi, ja sigui per estar fora de la ciutat o bé perquè no es tingui permís per accedir a cap punt d'accés, serà necessària alguna tecnologia per comunicar amb el servidor. Es podria utilitzar 4G però el cost mensual és elevat. També seria factible un mòdul WAN, com LoRaWAN, que permeti comunicar-se amb un gateway més o menys proper. Un exemple és la xarxa IoT The Things Network que s'està muntant a Girona [3].
- Punt de lectura fix i amb bateria (per exemple amb Arduino). Seria necessari adquirir l'Arduino de més baix consum per a aquest objectiu. En l'apartat Estudis i decisions es detalla una estimació d'un sistema autònom amb bateries. Segons la quantitat de sensors, el computador utilitzat, i la tecnologia de la freqüència de lectura, la duració de les bateries pot variar dràsticament.

2.3. Conclusió de viabilitat

Vistos tots els punts anteriors podem dir que es tracta d'un projecte factible tant a nivell econòmic com tecnològic. Es tracta d'uns sensors bastant econòmics que podrien interessar a organitzacions públiques o privades sempre que el seu grau de precisió i durabilitat siguin acceptables. Caldrà controlar el tema de l'exposició a la intempèrie per maximitzar la durada del sistema.

3. Metodologia

Utilitzarem una metodologia de tipus àgil, permetent així un desenvolupament dinàmic de la solució i amb una posada en producció relativament ràpida. El manifest àgil té 11 principis [4]:

1. La nostra principal és satisfer al client a través de l'entrega ràpida i contínua de software i hardware de valor. El client en aquest cas seríem de moment els integrants del grup de desenvolupament però més endavant podrien ser organismes públics, empreses o aficionats que ens facin les seves peticions o suggerències.
2. Els requisits canviants són benvinguts, així es té un avantatge competitiu superior per al client.
3. Entregar freqüentment i periòdicament un software que funcioni.
4. Les persones del negoci i els desenvolupadors han de treballar de manera quotidiana a través del projecte.
5. Construir els projectes amb individus motivats.
6. La manera més eficient i efectiva de comunicar-se és cara a cara.
7. El producte que funcioni és la principal mesura de progrés.
8. Els processos àgils promouen el desenvolupament sostingut, cal tenir un ritme constant i indefinit. En el cas del nostre projecte seguirem treballant una vegada entregat aquest document, ja que la idea és portar a terme la distribució d'aquests punts, ja sigui alliberant el codi i esquemes perquè tothom els pugui instal·lar o bé com a negoci futur.
9. Cal tenir sempre atenció a l'excel·lència tècnica, així també es millora l'agilitat.
10. La simplicitat és essencial.
11. Les millors arquitectures, requisits i dissenys surten d'equips que s'autoorganitzen.

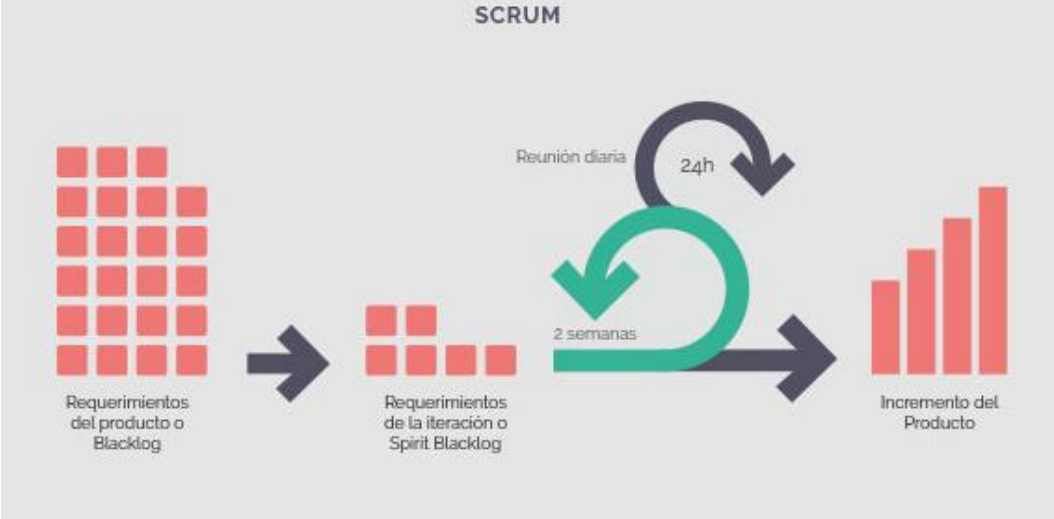


Figura 1: Esquema de la metodologia Scrum

Més en concret, farem servir la metodologia Scrum, amb les següents peculiaritats:

1. Desenvolupament incremental, en comptes de planificar i executar tot el producte de cop.
2. La qualitat s'analitza amb el mateix coneixement de les persones de l'equip.
3. Solapament de fases de desenvolupament.
4. Seguir els passos de desenvolupament àgil, amb iteracions (també anomenades "sprints"), on cada vegada s'incrementa la funcionalitat. Cada inici d'sprint té la seva reunió de planificació, i cada final té la seva revisió per a veure si s'assoleix l'objectiu. La comunicació fluida i recurrent és molt necessària en aquesta metodologia de treball.

En el cas del nostre projecte tindrem diversos sprints que ens permetran assolir els nostres objectius de manera ràpida i organitzada, a l'apartat planificació es detallen els que s'han assolit fins la data de finalització d'aquest document.

Durant el projecte s'utilitzaran eines de comunicació més fluides que els correus electrònics, com per exemple un grup de Telegram on hi ha els integrants de l'equip i que permet una comunicació instantània dels problemes o idees que vagin sorgint.

A més, per tal d'organitzar les tasques s'utilitzarà Trello per assignar-les i veure en quin estat progressen.

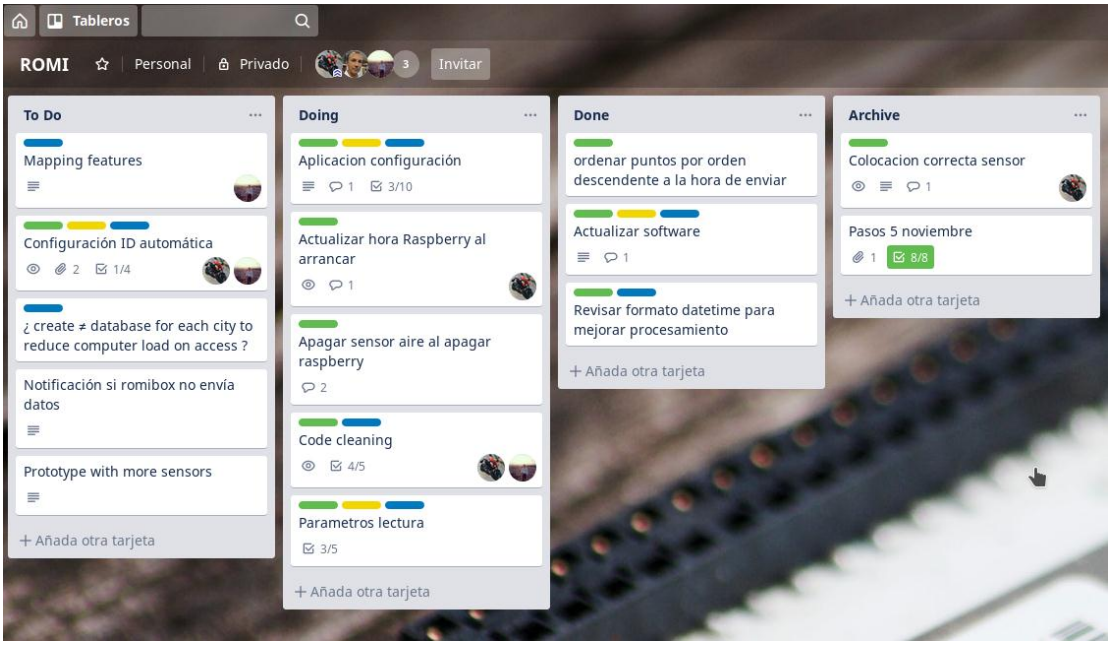


Figura 2: Exemple de tauler de Trello amb l'estat de les tasques

També s'utilitzaran dos repositoris a GitHub per a pujar el codi i veure els canvis que es van realitzant. Inicialment són privats però en un futur quan la solució sigui més definida segurament seran públics. Un dels repositoris es diu RoMiBox i es tracta del codi per la Raspberry Pi en els punts de lectura i l'altre repositori es diu RoMiApp i serveix per a crear l'aplicació de configuració. Més endavant podria servir per tenir la funció de consultes de les dades recollides.

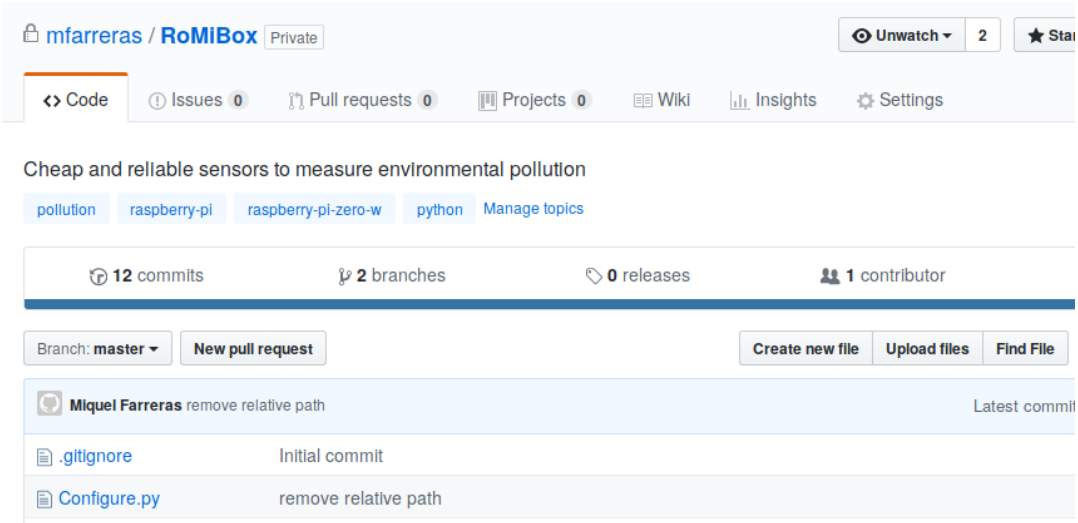


Figura 3: Repositori de GitHub de la RomiBox

4. Planificació

L'inici del projecte actual se situa a finals d'octubre, quan es fa el full de projecte i aprovació del mateix i es comença a analitzar la solució a partir dels mateixos dies.

Un cop analitzat el context del projecte, es realitza el disseny i es prenen totes les decisions necessàries.

El document del treball s'ha de presentar com a molt tard el 10 de juny de 2019, tenint en compte que és necessari imprimir i entregar la documentació presencialment marquem com a data límit el 03/06/2019 per a realitzar aquesta part.

Com que es tracta d'una metodologia Scrum, planificarem amb un diagrama de Gantt el pla inicial que volem seguir per cada sprint, tot i que acaben variant lleugerament les assignacions de tasques per cadascun d'ells.

La idea és analitzar com satisfer millor les necessitats del projecte i després començar un desenvolupament de tota la solució, aplicant millores sobre les idees o incidències que anem trobant.

Les tasques realitzades s'han distribuït en el temps segons el següent diagrama, com es pot veure es superposen moltes tasques relacionades entre si ja que la implementació i les proves van lligades:

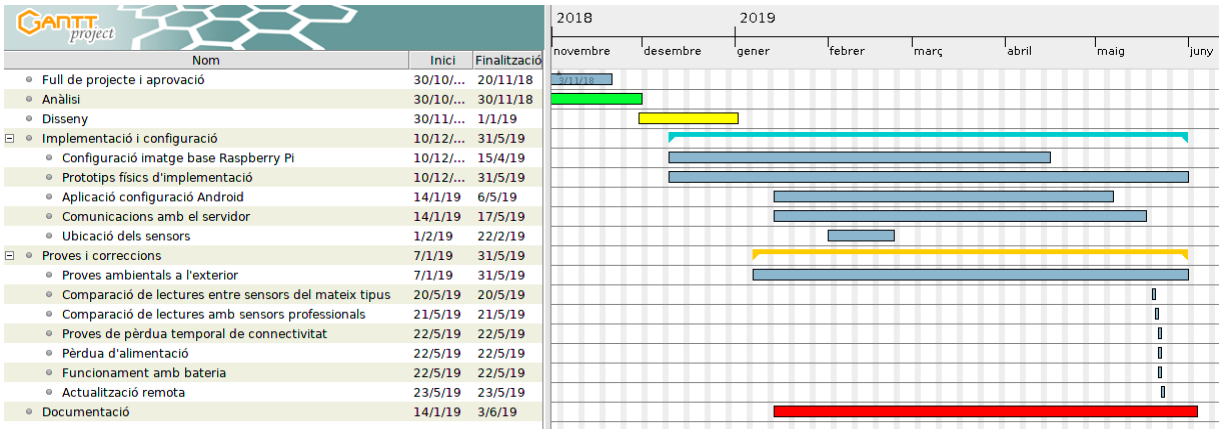


Figura 4: Planificació del projecte durant el curs

Les hores dedicades al treball des de l'inici d'aquest projecte (30 d'octubre) fins al 3 de juny són un total de 400 h aproximadament, tenint en compte les reunions i jornades de prova conjuntes entre els participants en el projecte, desenvolupament individual de la solució, proves i documentació. En aquest sentit es superen àmpliament els 9 crèdits ECTS estimats per a la realització d'aquest TFM.

5. Conceptes previs

A continuació es defineixen els termes més utilitzats i importants per a la redacció i comprensió d'aquest projecte.

5.1. Sistema incrustat

Sistema informàtic concebut per a realitzar una tasca determinada i integrat en un sol dispositiu. A diferència d'un ordinador personal, que realitza diverses tasques, aquests dispositius només solen realitzar una tasca determinada.

El nostre projecte és clarament centrat en realitzar sistemes encastats per a llegir la contaminació ambiental, i amb comunicacions sense fils. Les principals característiques que s'han de complir són les següents:

- **Cost**: el mínim possible.
- **Memòria**: Espai mínim necessari. GB relativament barats.
- **Potència de càlcul:** Potència justa per a la funcionalitat, per evitar consumir en excés i realitzar la tasca sense problemes.
- **Autonomia**: Consum mínim necessari, tant si es disposa de bateries com d'energia permanent. S'ha d'optimitzar el consum elèctric.
- **Temps**: Temps delimitats d'execució i expiració de les tasques,
- **Fiabilitat**: El nostre projecte no té riscos per les vides ni per grans inversions, per tant no es tracta d'un sistema crític. De totes maneres el percentatge de fallades ha de ser baix i les lectures han de ser precises dins un rang determinat.
- **Seguretat**: S'han de realitzar les comunicacions de manera privada, encara que el nostre sistema no tindrà dades personals de cap usuari. Les ubicacions dels sensors seran desades però seran ofuscades (lleugerament modificades).

5.2. IoT

La internet de les coses (Internet of Things) [5] és la integració de connectivitat a Internet d'objectes utilitzats diariament. S'integren amb electrònica, connectivitat i hardware com sensors, i es poden comunicar entre ells i/o ser monitoritzats i controlats remotament.

Bàsicament, aquest projecte és un sistema encastat aplicant la política d'IoT, on tots els requisits es compleixen, ja que té un protocol senzill, computació al núvol, comunicació sense cables, un sistema lleuger i en temps real i al darrera hi ha una recerca activa per a obtenir els millors resultats. La integració en l'entorn es pot a portar a terme amb la instal·lació en llocs públics com l'enllumenat o estacions d'altres tipus ja existents, com meteorològiques, entre moltes altres possibilitats.

Figura 5: Camps de l'aplicació de Internet of Things

5.3. API

Una Interfície de Programació d'Aplicacions (Application Programming Interface, API, en anglès)[6], és una interfície que defineix com han de comunicar-se diversos programes entre si.

Més en concret una API REST (Representational State Transfer) serà el que s'utilitzarà en aquest projecte si el que volem és comunicar el nostre sistema encastat amb un servidor al núvol. Un REST és una arquitectura pensada per a sistemes distribuïts, que tindrà les següents característiques[7]:

- **Uniform interface**.
- **Client-Server**. Dades al servidor, codi client més simple.

- **Stateless**. El servidor no ha de desar informació entre cada petició seguida.
- **Cacheable**. Les obtencions d'informació es poden desar per no haver de repetir les peticions al servidor i estalviar ample de banda.
- **Layered system**. Hi poden haver servidors caché o balanceig entre el/s servidor/s final/s i el client, entre d'altres solucions d'alta disponibilitat.
- **Code on demand (opcional)**. Es pot oferir codi des del servidor per incrementar la funcionalitat dels clients temporalment.

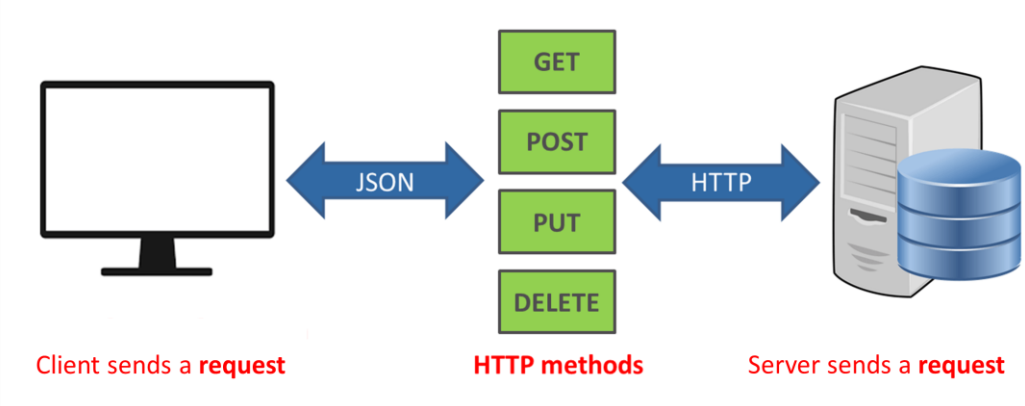


Figura 6: Comunicació per API Rest entre client i servidor/s

5.4. Sistema Operatiu

Conjunt de programes que fan funcionar un computador. Gestionen la informació tractada, les comunicacions amb l'exterior i els perifèrics, controlar els errors, optimitzar els recursos, etc.

Els més comuns són Microsoft Windows, Linux o Mac OS X, entre d'altres.

La estructura d'un computador (i on s'ubica el sistema operatiu) es representa en el següent esquema:

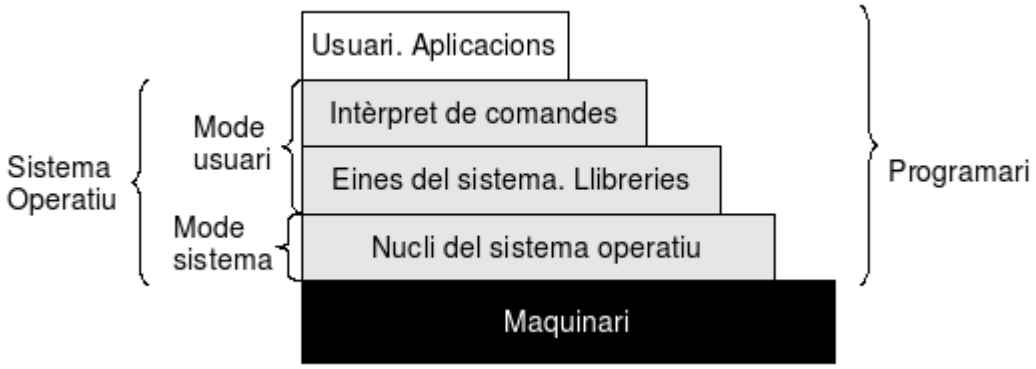


Figura 7: Capes de hardware i software que composen un ordinador

En el nostre cas en la part de programari utilitzarem Linux com a Sistema Operatiu, les aplicacions seran realitzades en Python i SQL i tindrem llibreries diverses per a poder llegir els sensors, que seran el maquinari.

5.5. Contaminació atmosfèrica

La contaminació atmosfèrica és la contaminació present en l'aire, amb components presents en quantitats superiors a les naturals que suposen un risc per a la salut o una molèstia per les persones, animals i/o plantes.

Els principals contaminants atmosfèrics són [8]:

- Òxids de sofre (SO_2 i SO_3), combustió de carbons i petrolis, produeix irritació a les vies respiratòries.
- Sulfur d'hidrogen (SH_2), processos industrials, fan pudor i són tòxics.
- Monòxid de Carboni (CO), combustió incompleta de petrolis, produeix mal de cap degut a la toxicitat, que interfereix en el transport d'oxigen per la sang.
- Diòxid de Carboni (CO_2), produït en la combustió completa de petrolis, augmenta l'escalfament global del planeta degut a l'efecte hivernacle.
- Hidrocarburs, emesos per motors de combustió, mostrats com a cancerígens.
- Ozó, produït per trànsit de vehicles intens, efectes no demostrats.
- Òxids de Nitrogen (NO i NO_2), produïts per vehicles sobretot dièsel, tòxics i irritants.
- Partícules o pols, les partícules més petites produïdes per indústries o erosió causen malalties respiratòries, que es poden introduir als sistemes respiratoris si es tracta de partícules entre 1 µm i 10 µm.

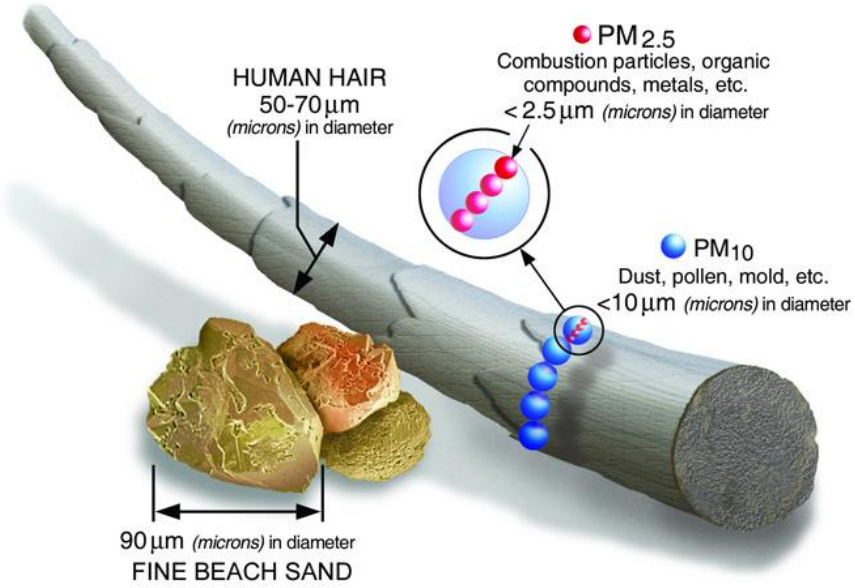


Figura 8: Comparació entre el cabell humà i els PM 2.5 i PM 10

Per tant és molt important mantenir controlats aquests contaminants a les ciutats per tal de mantenir una bona qualitat de vida per als seus habitants i per al planeta en general, podent reduir els efectes del canvi climàtic si s'apliquen polítiques més restrictives.

5.6. Comunicacions sense fils

5.6.1. Wi-Fi

Tecnologia de comunicació sense fils que típicament fa servir les freqüències 2.4 i 5 GHz, i que s'utilitza per a establir connexions a internet d'alta velocitat i sense cables en distàncies curtes, de l'ordre de desenes de metres com a màxim. L'SSID és el nom del punt d'accés.

Figura 9: Exemple de distribució de les ones Wi-Fi en una llar

5.6.2. 4G

Tecnologia de telecomunicacions sense fils que es refereix a la quarta generació de la telefonia mòbil (també anomenada Long Term Evolution o LTE). Està basada en un protocol IP i una xarxa d'antenes que formen cel·les, les quals permeten que els dispositius mòbils es connectin a la xarxa i puguin comunicar-se amb la resta d'Internet, a part de poder realitzar trucades i enviar SMS.

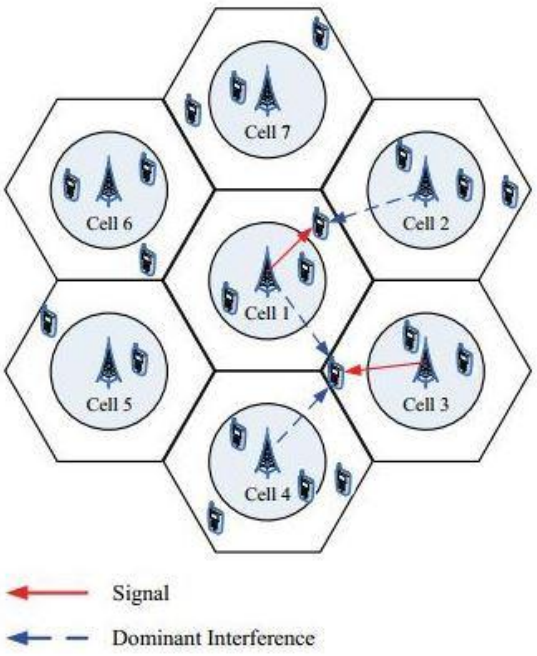


Figura 10: Distribució de cel·les de telefonia 4G

5.6.3. LoRaWAN

El nom deriva de l'abreviació Long Range [9]. Tecnologia en expansió que està ideada per a petits dispositius IoT que han de realitzar un enviament o recepció de dades cap a servidors al núvol. Aquesta connexió es realitza mitjançant unes freqüències baixes (baix consum però també baixa velocitat), que permeten comunicar els clients amb els gateway. Els gateway acumulen el trànsit de peticions i els envien cap a Internet per dirigir-les al servidor corresponent.

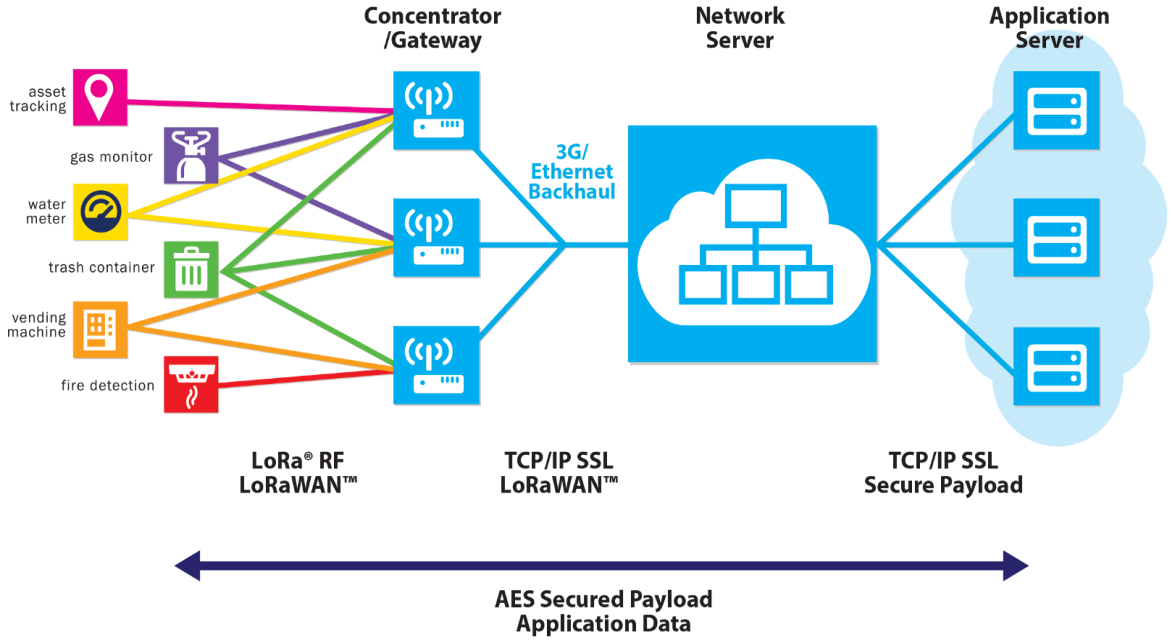


Figura 11: Comunicació entre elements IoT i els servidors mitjançant LoRaWAN

5.7. Equip de persones

Dr. Lluís Pacheco Valls

Tutor del Treball de Final de Màster

Departament: Arquitectura i Tecnologia de computadors

Grup de recerca: Visió per computador i robòtica (VICOROB)

Dr. Nicolas Boccard

Participant i dinamitzador del projecte

Departament: Economia

Grup de recerca: Grup de recerca en anàlisi econòmica

Robert Garcia Ventura

Realitzarà projecte vinculat a aquest on es reben i analitzen les dades

Estudiant del Màster en Enginyeria Informàtica de la UdG

Miquel Farreras Casamort

Participant principal en la part de hardware i enviament de dades del projecte

Estudiant del Màster en Enginyeria Informàtica de la UdG

6. Requisits del sistema

6.1. Requisits bàsics

Vistes les necessitats i conceptes necessaris, la solució de hardware i software desenvolupada ha de permetre:

- Recollir dades dels contaminants que més varien i que siguin els més perjudicials per la salut dels habitants, a un servidor al núvol.
- Construir un prototip de punt de lectura amb un cost baix i que proporcioni una fiabilitat mínima de lectura i de funcionament a mig/llarg termini.
- Que aquest hardware pugui resistir a un ús a la intempèrie sense deixar de funcionar.
- Que la construcció del hardware es pugui fer per part d'un usuari sense requerir un nivell tècnic excessiu.
- Que els costos de comunicacions i consum energètic siguin el més baixos possible.
- Idear diversos tipus de prototips, mòbils o fixes

6.2. Requisits específics

- Necessitem un microcomputador que tingui un consum baix i la potència suficient per a realitzar les tasques de lectura de sensors i enviament de les dades al servidor al cloud, mitjançant una API REST. També ha de tenir emmagatzematge en cas de pèrdua de connexió.
- Hem de seleccionar els sensors més importants i la manera de connectar-los al microcomputador (GPIO, I2C, SPI…).
- Hem d'idear o cercar un recipient de fàcil obtenció i que protegeixi el conjunt de components de la intempèrie.
- Hem de simplificar el procés de muntatge eliminant la necessitat de soldar.
- Dissenyar els diversos tipus de prototips, segons siguin mòbils o fixes, contenint GPS en el cas dels mòbils.
- El prototip del hardware de cada punt de lectura s'anomenarà RoMiBox.

7. Estudis i decisions

7.1. Ordinador IoT

Els requisits són:

- Connexions amb pins GPIO per a llegir sensors i alimentar-los
- Emmagatzematge intern mínim per desar les dades en cas de pèrdua de connexió o d'alimentació (volem tenir un historial)
- Connexió Wi-Fi
- Possibilitat de connexió 4G o LORA/Zigbee/similars
- Econòmic (menys de 50 €)
- No és necessària una potència de càlcul gaire elevada, només cal llegir sensors, desar i enviar dades

7.1.2. Raspberry Pi 3 B+

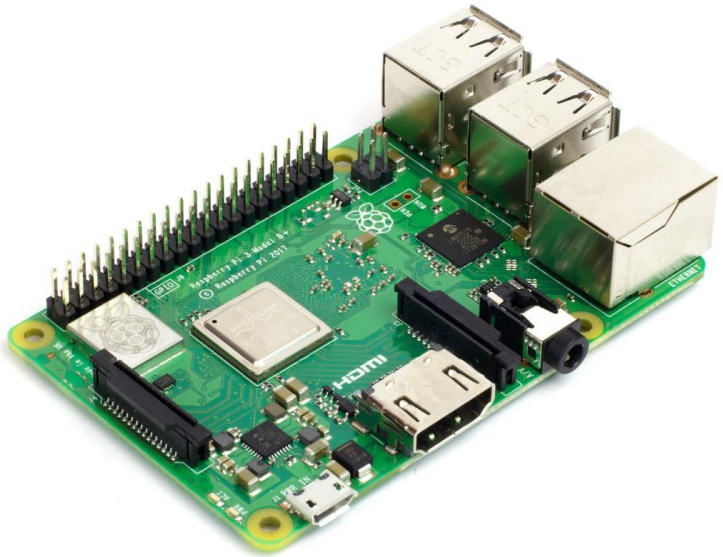


Figura 12: Raspberry Pi 3B+

Última versió del computador monoplaca més famós, utilitzat en els passos anteriors al projecte actual. Per les tasques que es realitzen és massa potent, ja que la idea és mantenir la simplicitat de només llegir els sensors i enviar les dades. Per a una tasca com aquesta no cal tenir una CPU ARM amb 4 cores i 1 GB de memòria RAM. A més, el consum en mA és bastant elevat (400 mA en repòs) i això ens limitaria molt en cas de tenir una versió mòbil.

Les especificacions tècniques són les següents [10]:

- CPU + GPU: Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- RAM: 1GB LPDDR2 SDRAM
- Wi-Fi + Bluetooth: 2.4GHz y 5GHz IEEE 802.11.b/g/n/ac, Bluetooth 4.2, BLE
- Ethernet: Gigabit Ethernet sobre USB 2.0 (300 Mbps)
- GPIO de 40 pins
- HDMI
- 4 ports USB 2.0
- Port CSI per a connectar una càmera
- Port DSI per a connectar una pantalla tàctil
- Sortida d'àudio estèreo i vídeo compost
- Micro-SD
- Power-over-Ethernet (PoE)
- Cost: 35,99 €

Vist el gràfic següent, passarem a analitzar la Raspberry Pi Zero WH (Wireless with Headers, Wifi i Bluetooth + GPIO soldats) per veure si compleix els nostres requisits.

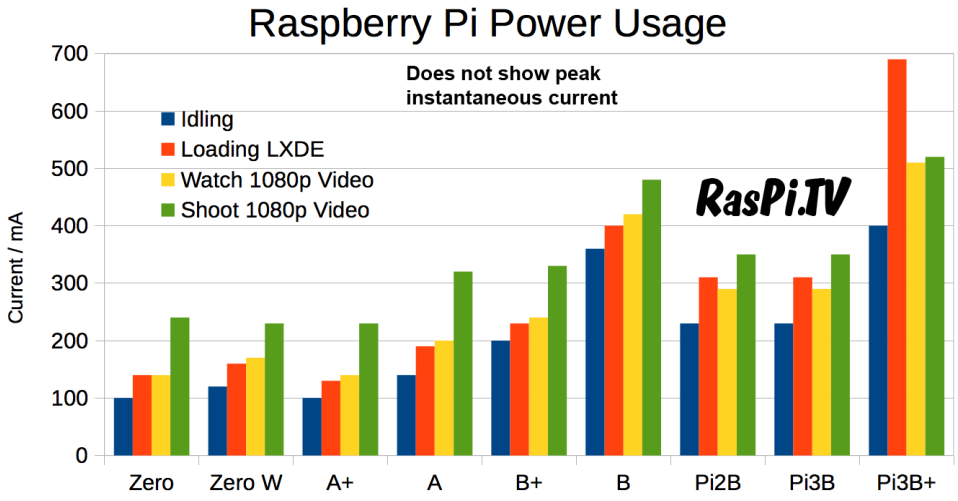


Figura 13: Consum dels diferents models de Raspberry en mA

7.1.2. Raspberry Pi Zero WH

Figura 14: Raspberry Pi Zero WH

Anem a analitzar la versió retallada de la Raspberry Pi, que ve amb 1 sol core de CPU i 512 MB de RAM [11] El cost enviat a Espanya són al voltant de 10-15 € depenent del proveïdor i per tant es tracta d'un estalvi important. No necessitem el connector Ethernet ni els USB, a més de l'HDMI de mida completa, la qual cosa permet una reducció de costos i del consum elèctric molt importants. Sí que té connectivitat Wi-Fi i Bluetooth Low Energy (LE) que ens poden permetre molta flexibilitat. Aquesta versió de la Raspberry Pi WH consumeix 120 mA en repòs, mentre que si desactivem el LED consumeix 80 mA.

Les especificacions tècniques són les següents:

- CPU + GPU: Broadcom BCM2835, single core ARM11 @ 1 GHz
- RAM: 512MB
- Wi-Fi + Bluetooth: 2.4GHz IEEE 802.11.b/g/n, Bluetooth 4.1, BLE
- GPIO de 40 pins
- Mini-HDMI (sortida vídeo 1080p 60 Hz)
- Port CSI per a connectar una càmera
- Micro-SD
- Micro-USB OTG
- Alimentació Micro USB
- Cost: 14,60 €

7.1.3. Arduino Uno

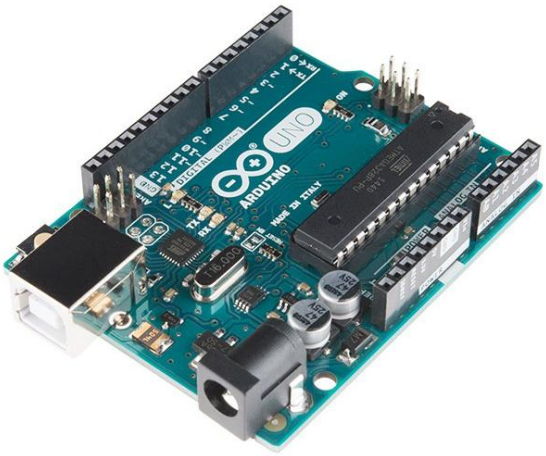


Figura 15: Arduino Uno [12]

Encara que més limitada, perquè només permet carregar un fitxer de codi a executar en bucle, es tracta d'una opció també apta per al muntatge d'aquest projecte. No té connexió Wi-Fi integrada i això la limita una mica, tot i que es pot afegir amb un accessori comprat a part. El consum en repòs varia entre 46.5mA a 34.4mA, si es desactiven els LED. Tampoc té emmagatzematge intern suficient per emmagatzemar gaires dades si es perd la connexió o l'alimentació, es tracta d'un buffer de només 512 bytes. Sí que té, en canvi, entrades analògiques de les que no disposa cap Raspberry Pi i que ens poden ser útils per a alguns tipus de sensors. Descartem els clons xinesos per a tenir una dubtosa qualitat, tractant-se del component principal del nostre projecte.

Especificacions:

- Microcontrolador ATmega328P
- Voltatge de treball: 5V
- Voltatge d'entrada (recomanat): 7-12V
- Voltatge d'entrada (límit): 6-20V
- Digital I/O Pins: 14 (6 per a output PWM)
- PWM Digital I/O Pins: 6
- Pins d'entrada analògica: 6
- Corrent DC per cada pin I/O: 20 mA
- Corrent DC per cada pin 3.3V: 50 mA
- Memòria flash: 32 KB (ATmega328P)
- SRAM: 2 KB (ATmega328P)

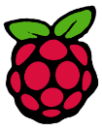
- EEPROM: 1 KB (ATmega328P)
- Velocitat de rellotge: 16 MHz
- LED_BUILTIN: 13
- Cost (sense Wi-Fi): 20,48 €
- Cost (Wi-Fi integrat): 47,07 €

Finalment, vistes les possibilitats i cost, ens decidim per la Raspberry Pi Zero WH, que ens ofereix un cost baix, moltes opcions de connectivitat, permet executar codi en Python, integra connectivitat Wi-Fi i el consum és bastant contingut.

No es descarta en un treball futur que es realitzin prototips amb un Arduino però el cost d'aquesta placa més el mòdul de comunicació Wi-Fi o LoRa queda una mica per sobre del cost inicial pensat per al projecte.

7.2. Sistema operatiu

Com que finalment utilitzem una Raspberry Pi Zero WH, instal·larem un sistema operatiu basat en GNU/Linux. Les opcions disponibles són [13]:

Nom distribució	Icona	Característica principal	Apte
Raspbian		Sistema més utilitzat sobre Raspberry Pi, basat en Debian i amb una gran comunitat de suport.	
Ubuntu MATE		Entorn d'escriptori MATE.	
Ubuntu Core		Ubuntu optimitzat per a Raspberry Pi, únic dels aquí comparats que podria servir a part de Raspbian. Encara no té tanta comunitat com Raspbian.	
Ubuntu Server		Poc optimitzat per a ser utilitzat sobre un microcomputador com el nostre.	
Windows 10 IoT Core		Entorn gràfic o sense, però sense tant suport i encara en fases inicials.	

OSMC		Media Center.	
LibreELEC		Orientat a reproducció multimèdia.	
PiNET		Serveix per a muntar una aula amb usuaris i filesystems centralitzats, no aplica.	
RISC OS	RISC OS Pi	No basat en Linux, poc suportat respecte Raspbian per a un projecte com el nostre	
Weather Station		Projecte d'Oracle per muntar estacions meteorològiques.	
IchigoJam	IchigoJam	Per als primers passos en la programació.	

Finalment gravarem Raspbian Stretch Lite a la targeta SD, ja que és el SO més utilitzat en aquesta plataforma de hardware i és derivat de Debian, una garantia d'estabilitat. A més aquesta versió no porta entorn d'escriptori ni programari innecessari (com el paquet ofimàtic).

El procediment per a instal·lar un sistema operatiu sobre la Raspberry Pi és gravar el fitxer .img a la targeta SD, descarregat des del web del sistema operatiu seleccionat. Per a facilitar i accelerar la gravació de targetes SD, hem de seleccionar un programari que faciliti aquesta tasca.

7.3. Solucions de gravació d'imatges de disc

Per facilitar la gravació dels fitxers .img, hem buscat un software que les faci ràpidament i qualsevol usuari/plataforma els pugui utilitzar. Després de fer una cerca, utilitzarem el Balena Etcher, ja que és multi plataforma i bastant reconegut:

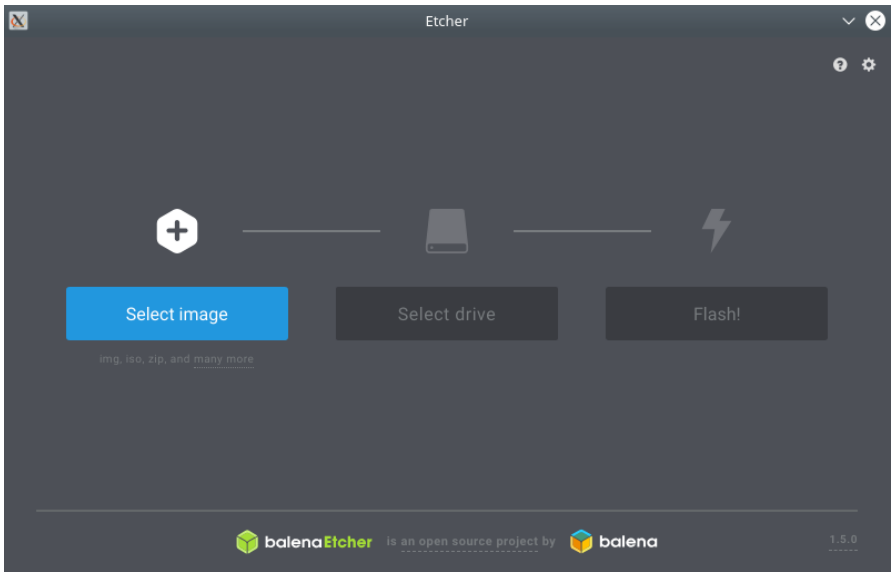


Figura 16: Execució del Balena Etcher, programa de gravació de fitxers .img

7.4. Sistemes de generació d'imatges de disc personalitzades

Inicialment utilitzem DD per generar un .img de les imatges, aquesta és una tasca que típicament no hauran de realitzar els usuaris, per tant es pot realitzar per línia de comandes. Consultem la web de Raspbian ja que és una font confiable d'informació sobre aquests temes [14].

Linux:

Llistar els dispositius disponibles amb la comanda lsblk:

```
mfarreras@mfarreras-Z97P-D3:~$ lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
loop0    7:0    0  89,3M  1 loop /snap/core/6673
loop2    7:2    0  89,4M  1 loop /snap/core/6818
loop3    7:3    0 180,2M  1 loop /snap/spotify/35
loop4    7:4    0   174M  1 loop /snap/spotify/34
loop5    7:5    0 174,5M  1 loop /snap/spotify/32
loop6    7:6    0  88,4M  1 loop /snap/core/6964
sda      8:0    0 111,8G  0 disk
├─sda1   8:1    0   549M  0 part
├─sda2   8:2    0  71,3G  0 part
├─sda3   8:3    0     1K  0 part
└─sda5   8:5    0    40G  0 part /
sdb      8:16   0 931,5G  0 disk
└─sdb1   8:17   0 931,5G  0 part
sdc      8:32   1  14,5G  0 disk
├─sdc1   8:33   1  42,9M  0 part
└─sdc2   8:34   1  14,4G  0 part
sr0     11:0    1  1024M  0 rom
```

Figura 17: Execució de lsblk a Ubuntu

Identificar l'ID i substituir-lo a la comanda següent (en aquest cas /dev/sdc), vista la mida total del disc (targeta SD de 16 GB):

```
mfarreras@mfarreras-Z97P-D3:~$ sudo dd bs=4M if=/dev/sdb | gzip > raspbian.img.gz
```

Mac OS:

Executar diskutil list:

```
ConsoleAppDemo — test — bash — 80×17
CRUMPMAC:ConsoleAppDemo crump$ diskutil list
/dev/disk0
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:      GUID_partition_scheme                        *251.0 GB   disk0
   1:                        EFI EFI                     209.7 MB   disk0s1
   2:          Apple_CoreStorage                        250.1 GB   disk0s2
   3:                 Apple_Boot Recovery HD            650.1 MB   disk0s3
/dev/disk1
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:                  Apple_HFS Macintosh HD           *249.8 GB   disk1
                                 Logical Volume on disk0s2

                                 Unlocked Encrypted
/dev/disk2
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:     FDisk_partition_scheme                        *15.5 GB    disk2
   1:             Windows_FAT_32 RASPBIAN                15.5 GB    disk2s1
```

Figura 18:Exemple d'execució de diskutil list a Mac OS

Identificar l'ID i substituir-lo a la comanda següent (en aquest cas /dev/rdisk2):

sudo dd bs=4m if=/dev/rdisk2 of=raspbian.img

Windows:

Crear la imatge amb el programa Win32 Disk Imager, seleccionant el dispositiu i el nom de fitxer: https://sourceforge.net/projects/win32diskimager/

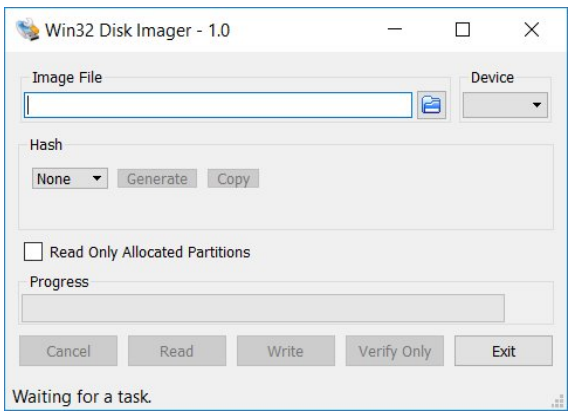


Figura 19: Win32 Disk Imager, programa de gravació i backup amb fitxers .img

7.5. Emmagatzematge

Seguint amb les despeses contingudes, seleccionem la targeta SD més econòmica disponible en el mercat que ens permeti instal·lar el sistema operatiu Raspbian. Decidim utilitzar les targetes micro SDHC de 8 GB classe 10 que ens permetran tenir un rendiment bo i el cost mínim, ja que amb menys capacitat no es pot gravar ja el fitxer .img de Raspbian [15]. De qualsevol marca amb una mínima reputació són vàlides. Es poden utilitzar de més capacitat sense problemes.

Figura 20: Targeta SD Verbatim utilitzada en el nostre projecte

7.6. Sensors

Ens interessaria mesurar els següents components de l'ambient, però per a crear un prototip barat i significatiu hem d'escollir quins sensors utilitzem segons el seu cost i prestacions. Tindrem en compte els components més comuns i més variables segons el trànsit a les ciutats, a més de les condicions climatològiques, en negreta els més desitjables:

- **Partícules PM2.5 i PM10**
- **CO2**
- **CO**
- **NOx**
- **Humitat**
- **Temperatura**
- Ozó
- Metà
- SO2
- Soroll
- Vent i direcció

Finalment no incloem un sensor GPS pel seu cost elevat (més de 35 €) i perquè l'interès del projecte de moment està en els punts estàtics de lectura.

Comparem en la següent taula els diferents sensors analitzats, segons les dades proporcionades i el seu preu:

Tipus de gas	**MQ7**	**DHT11**	**WSP2110**	**SCD30**	**MQ4**	**BME680**	**MQ135**	**HPMA115S0**
CO2			X	X			X	
CO	X							
NOx							X	
Humitat		X		X		X		
Temperatura		X		X		X		
SO2								
Ozó								
Metà					X			
Partícules PM 2,5								X
Partícules PM 10								X
Preu	2,21 €	1,09 €	11,66 €	72,52 €	1,47€	12,22 €	2,22 €	16,66€

Pel seu cost i característiques els sensors seleccionats **(en negreta)** són:

- **Partícules PM2.5 i PM10 → HPMA115S0**
- **CO2 → MQ135**
- **CO → MQ7**
- **NOx → MQ135**
- **Humitat → DHT11**
- **Temperatura → DHT11**
- Ozó → MQ131. Massa car tot i que hi ha unitats per uns 20 €. Seria un candidat per a l'ampliació del projecte.

- Metà → MQ4. És poc rellevant en les ciutats. També seria un candidat per ampliar el projecte.
- SO2 → SO2-20. Massa car, els més barats sense placa ja valen uns 110 €.
- Soroll → Muntatge complex i car. La majoria són per a un ús independent de qualsevol computador i això comportaria haver d'adaptar-los al nostre projecte. A part que els més barats valen a partir de 30 €.
- Vent i direcció → Muntatge complex, voluminós i car. Masses peces mòbils a tenir en compte, a part que deixaria de ser un dispositiu discret per a l'ús en llocs públics.

7.6.1. NOx i CO2→ MQ-135

El sensor utilitzat per a mesurar NOx i CO2 serà el MQ135, molt conegut en projectes per Arduino i Raspberry pel seu funcionament simple, baix cost i precisió acceptable. Consta d'una resistència que s'ha d'escalfar abans de la lectura (per donar valors estables) i el seu output és analògic, amb la qual cosa ens caldrà un conversor Analògic/Digital. L'output digital només dóna 1 si detecta el cas en qüestió.

Figura 21: Sensor MQ-135

7.6.2. CO → MQ7

Tipus de sensor molt similar al MQ135 i que permet obtenir els valors en ppm de CO en l'aire. També ofereix les lectures analògiques la qual cosa fa que necessitem el conversor A/D.

Figura 22: Sensor MQ-7

7.6.3. Humitat i temperatura → DHT11

Sensor de temperatura i humitat molt utilitzat i bastant precís, que ofereix les lectures en digital, llegibles pels pins GPIO de la Raspberry directament. Per error s'ha acabat comprant el que no porta resistència integrada, però de cares a nous punts de lectura és més recomanable amb la placa integrada, ja que els pins de connexió són més gruixuts i menys punts de fallada durant el muntatge. A més es perd el quart pin que no té cap funció.

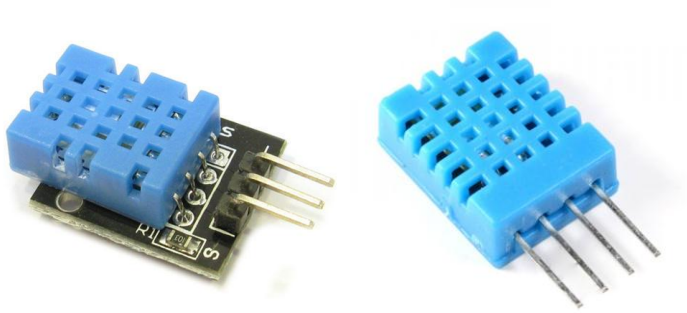


Figura 23: Diferència entre el sensor DHT11 amb i sense placa amb resistència integrada

7.6.4. Partícules → HPMA115S0

Sensor làser de partícules PM 2.5 i PM10, connectat per un port sèrie UART, un port ja disponible en la nostra Raspberry Pi. Ha sigut utilitzat des de l'inici del projecte i la seva relació precisió/durabilitat/preu és molt bona.

Figura 24: Sensor Honeywell HPMA115S0

7.7. Conversor A/D

Hem decidit utilitzar una Raspberry Pi i aquesta només té entrades digitals. Tots els sensors econòmics que hem trobat per a la lectura dels gasos tenen una sortida analògica per llegir la quantitat (la digital és 0 o 1 indicant si detecten el gas o no).

Després de veure diferents opcions, n'hi ha dues que ens poden servir:

7.7.1 Conversor A/D ADS1115

Es tracta d'un conversor analògic/digital, la versió d'Adafruit costa uns 15 € però acabem trobant un versió retallada amb els pins ja soldats per 2,95 € incloent enviament, que té 4 entrades analògiques i comunicació amb la Raspberry pels ports SDA i SCL (I2C), veure annex *Datasheet ADS1115* per a més detalls.

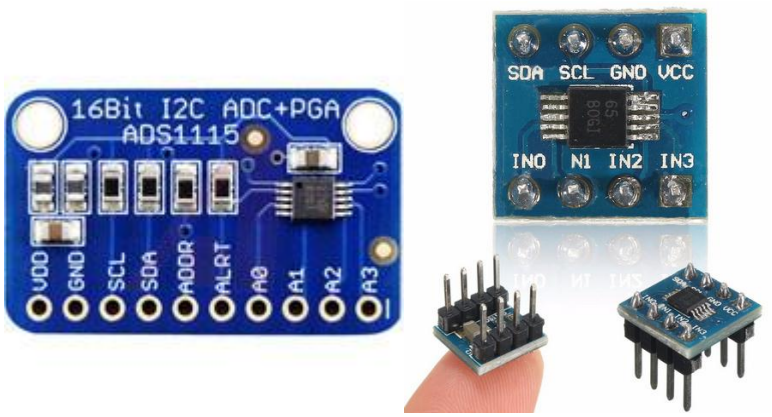


Figura 25: Diferència entre ADS1115 d'Adafruit i el model retallat

7.7.2 Conversor A/D MCP3008

En aquest cas es tracta d'un conversor més complex en el nombre de connexions a realitzar i s'ha de situar necessàriament sobre una protoboard, el cost d'aquest xip és de 3,08 € en el moment de la consulta tenint en compte l'enviament, amb la qual cosa queda descartat per major complexitat i major cost. El protocol utilitzat és SPI. Veure annex *Datasheet MCP3008* per a més informació.

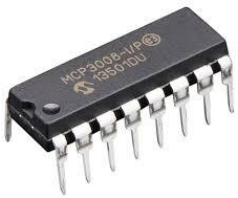


Figura 26: Xip conversor A/D MCP3008

7.8. Comunicacions

Com que el projecte està inicialment pensat per estar distribuït per l'interior de ciutats i moltes ja ofereixen la seva xarxa Wi-Fi, optem per utilitzar el mòdul Wi-Fi integrat en la nostra

Raspberry Pi Zero WH. Sí que pot ser desitjable més endavant afegir una comunicació 4G/LoRa però en ambdós casos s'augmenta molt el cost de la solució, tant pel mòdul a afegir, com per possibles pagaments, ja siguin mensuals o anuals per cada sensor, per la qual cosa de moment es descarten tot i que en un treball futur podrien aplicar-se.

7.9. Embolcall

Després de diverses proves i idees tenim tres tipus d'embolcalls candidats:

7.9.1. Maletí

El maletí de plàstic va ser utilitzat durant el primer prototip a l'assignatura de Projecte de Sistemes Encastats i Ubicus, on es va fer un prototip mòbil amb un sensor de partícules HPMA115S0 i GPS. Com que de moment volem fer punts fixes de lectura, aquest queda descartat. Si acabem fent un prototip mòbil seria el candidat principal per la seva facilitat de ser transportat.

Figura 27: Maletí de plàstic utilitzat en fases anteriors del projecte

7.9.2. Caixa acrílica

Un prototip utilitzat en aquest projecte és el de la caixa acrílica transparent, donat el seu baix cost. Aquest format és molt compacte i no permet la instal·lació de tots els sensors escollits, però s'ha utilitzat per al desplegament de sensors de només partícules en suspensió. L'avantatge és que també permet la mobilitat connectant-ho amb una bateria externa, encara que cal fer tots els forats necessaris per a passar els cables o permetre la lectura dels sensors cap a l'exterior.

Gràcies a la col·laboració del Dr. Nicolas Boccard es van distribuir diverses unitats del sensor Honeywell HPMA115S0 amb la Raspberry Pi Zero WH per diverses facultats de la UdG. Aquests prototips segueixen en funcionament després d'uns 10 mesos a la fi de redacció d'aquest projecte, la qual cosa demostra la resistència a la intempèrie del sistema, i la majoria dels sensors no han donat problemes. També es van instal·lar a Bonmatí però un patia moltes interferències en un entorn industrial i un altre va ser destruït per l'entrada d'aigua després d'una pluja torrencial.

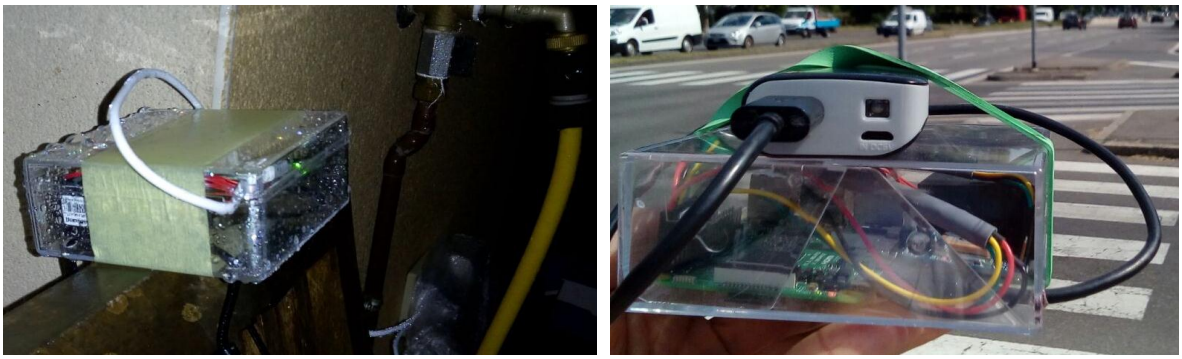


Figura 28: Prototips amb caixa acrílica

7.9.3. Caixa de connexions

Teòricament aquesta és la proposta més adequada per als sensors fixes, ja que permet una protecció superior a la pluja i és més dissimulada en cas d'estar a la vista de persones alienes. Realitzarem un prototip amb aquest tipus de caixa ja que els forats utilitzats per passar cables ens serviran com a orificis per obtenir l'aire pels sensors, traient els sensors MQ circulars per la part de sota. A més permeten múltiples possibilitats de subjecció com brides o cargols.

Figura 29: Caixa de connexions elèctriques per a exterior

8. Anàlisi i disseny del sistema

Un cop escollits els components del nostre sistema anem a definir l'esquema de connexió del hardware i també les comunicacions entre els punts de lectura i el servidor. Per a fer el següent esquema s'utilitza el programa de disseny Fritzing (https://fritzing.org), que a més permetria en un futur realitzar la impressió del pcb personalitzat i optimitzat segons les nostres necessitats.

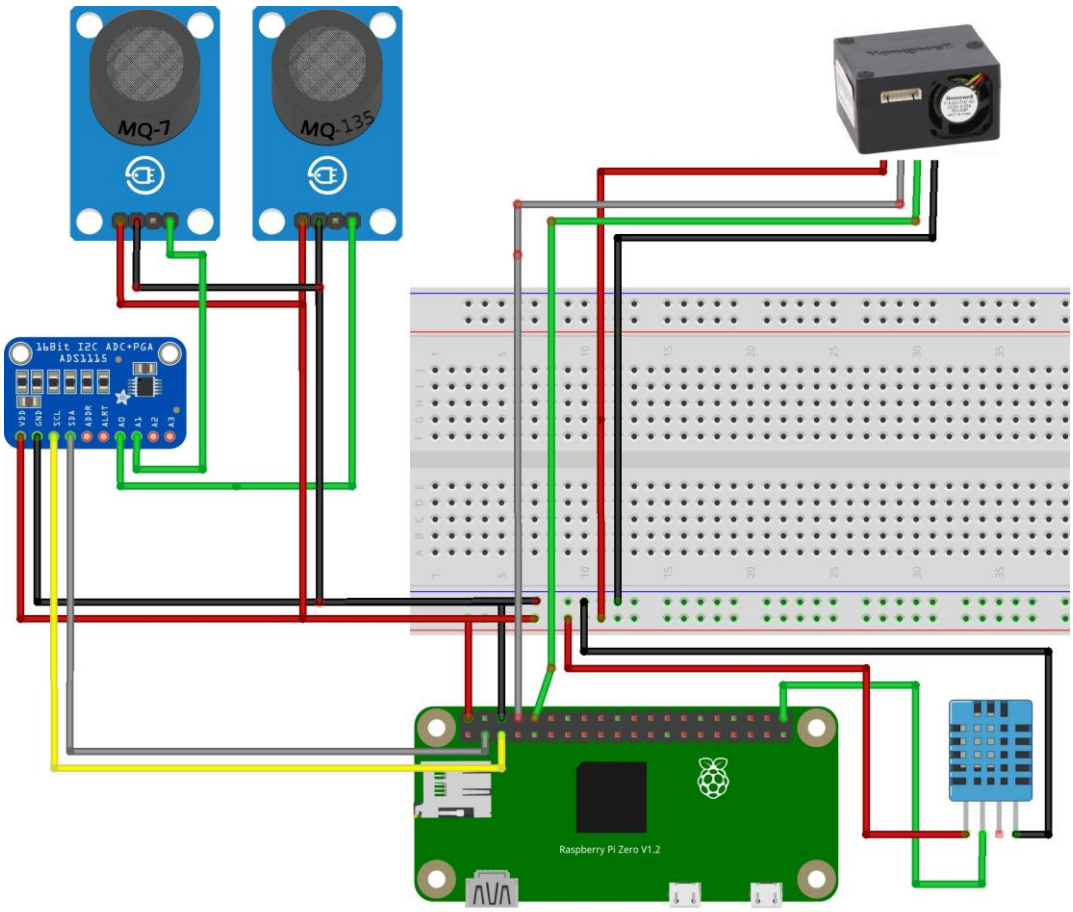


Figura 30: Esquema inicial de les connexions entre la Raspberry Pi i els sensors

Inicialment les comunicacions es realitzen amb el servidor ubicat al cloud en el servei de 1and1 contractat per l'alumne Robert Garcia Ventura, on es pugen les dades en JSON a través d'una petició amb PHP.

A finals de la realització d'aquest treball, dins el marc d'inici de la realització del projecte del TFM d'en Robert, s'ha portat el servidor a Amazon Web Services i per tant la comunicació es realitza amb una API molt més eficient i ben estructurada.

A continuació es mostra l'esquema inicial de comunicació amb PHP:

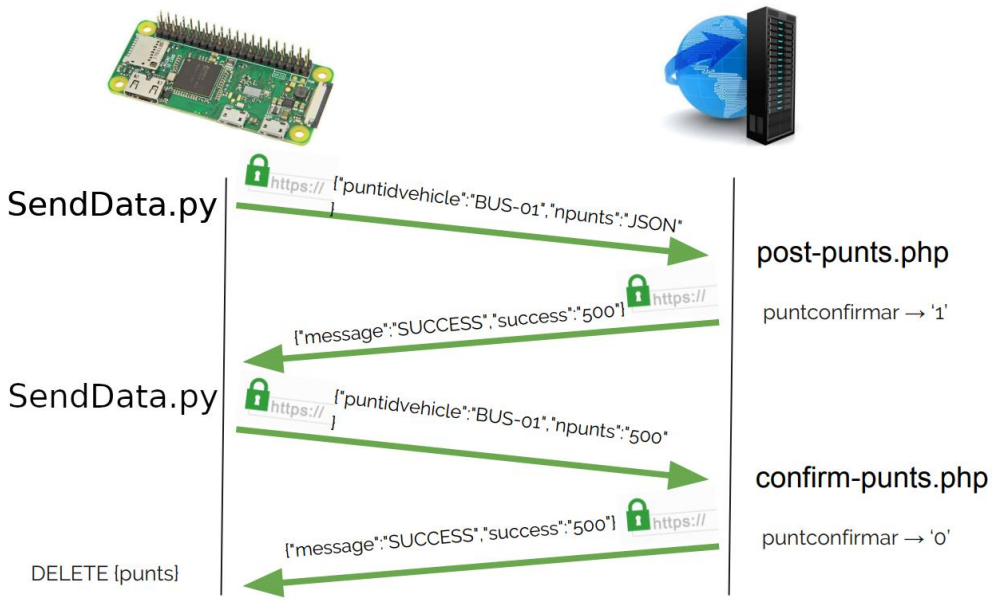


Figura 31: Esquema de connexió mitjançant PHP i les confirmacions de punts enviats

La comunicació amb la nova API és d'una sola anada i tornada, on la resposta del servidor ja és la confirmació de rebuda dels punts si rebem el codi HTTP OK 200. L'esquema de l'API és el següent (disseny i implementació fets per l'alumne Robert Garcia Ventura):

PATH	GET	POST	PUT	DELETE
/{api-version}/users/	Get list of users	Create new user		
/{api-version}/users/{user-id}/	Get info of the user		Update info of the user	Delete user
/{api-version}/users/{user-id}/stations/	Get list of stations of the user			
/{api-version}/stations/	Get list of stations	Create new station		
/{api-version}/stations/{station-id}/	Get info of the station		Update info of the station	Delete station
/{api-version}/stations/{station-id}/measures/	Get list of measures from the station	Create new measure		
/{api-version}/stations/{station-id}/measures/{measure-id}/	Get info of the measure		Update info of measure	Delete measure

Figura 32:Disseny API servidor RoMi

8.1. Apagat de Sensors

Per a preservar la duració dels sensors es proposa l'apagat d'aquests mitjançant un transistor NPN, controlat per un pin de la Raspberry Pi i aplicant una resistència entre el transistor i la Raspberry per evitar malmetre-la.

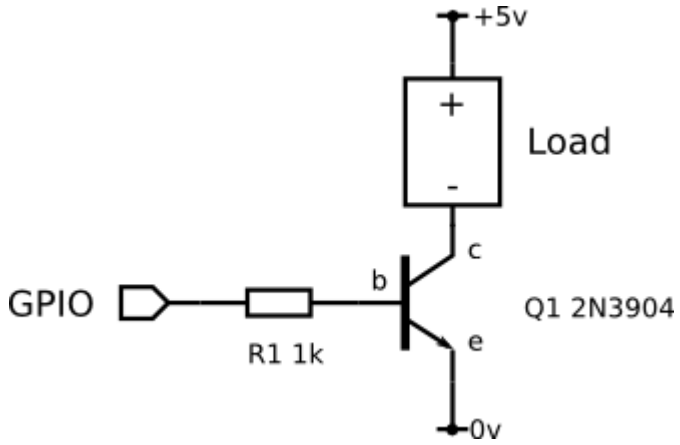


Figura 33: Mètode de connexió del transistor per apagar els sensors

Comprem una bossa de 100 transistors NPN 2n3904 per 1.74 €. Un cop provat s'acaba descartant ja que al tallar el corrent als 4 sensors alhora (que consumeixen bastant corrent) es reinicia la Raspberry.

9. Implementació i proves

9.1. Configuració imatge base Raspberry

Creem la targeta micro SD amb l'última imatge de Raspbian de Novembre de 2018, baixant-la del web oficial (https://www.raspberrypi.org/downloads/raspbian/). Només necessitem la versió Stretch Lite sense entorn d'escriptori, estalviant recursos (recordem que en la Raspberry Pi Zero WH són bastant limitats).

Utilitzem el Balena Etcher per a gravar el fitxer .img a la targeta SD, en aquest cas hem utilitzat una targeta SD de 8 GB Verbatim com les que hem escollit a l'apartat Estudis i decisions.

Una vegada gravada la targeta SD, muntarem el sistema de fitxers i podem observar la partició /boot. Allà hi posem dos fitxers per tal de poder-nos connectar remotament a la Raspberry Pi sense necessitat de cap monitor ni teclat, només per SSH. Els fitxers a crear són:

- SSH: fitxer en blanc anomenat així, serveix per activar el SSH
- wpa_supplicant.conf: serveix per a aplicar la configuració de la xarxa Wi-Fi on es connectarà la Raspberry per realitzar la configuració d'aquesta imatge base, a continuació es mostra el format que ha de tenir:

```
update_config=1
network={
    ssid="Nom_WIFI"
    psk="CONTRASENYA
}
```

Un cop arrencada la Raspberry ens connectem per SSH a la seva IP. Per a trobar la IP escanegem la xarxa amb una aplicació com Fing, que detecta quins dispositius estan connectats a la xarxa Wi-Fi on està connectat el nostre smartphone. En aquest cas s'està utilitzant l'usuari pi amb la IP 192.168.0.199, la contrasenya per defecte és Raspberry:

```
mfarreras@ideapad320:~$ ssh pi@192.168.0.199
```

Canviem la contrasenya de l'usuari pi amb la comanda passwd:

```
pi@raspberrypi ~ $ passwd
Changing password for pi.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
pi@raspberrypi ~ $
```

Figura 34: Canvi de contrasenya a Linux

Cal configurar la resolució de noms DNS, apuntem als DNS de Google ja que sinó no ens podríem connectar per nom al domini https://romi.gq ni actualitzar els paquets. Per a fer-ho modifiquem el fitxer /etc/dhcpcd.conf i afegim la següent línia:

```
static domain_name_servers=8.8.4.4 8.8.8.8
```

Amb això ja tenim accés a Internet per noms DNS i podrem procedir a actualitzar els paquets des dels repositoris de Raspbian, per a fer-ho executem les següents comandes:

```
pi@raspberrypi:~ $ sudo apt update && sudo apt upgrade
```

Després de bastanta estona, uns 30 minuts (recordem que la nostra Raspberry Pi Zero només té 1 core de CPU i 512 MB de ram), ja tenim el sistema actualitzat. De cara a mantenir una seguretat acceptable, i com que es tracta d'un sistema que s'instal·larà una vegada i no es farà un manteniment periòdic és important automatitzar les actualitzacions crítiques de seguretat:

```
pi@raspberrypi:~ $ sudo apt install unattended-upgrades
```

9.2. Desar dades

Per tal de desar les dades recollides pels sensors, creem una base de dades MYSQL en local a la Raspberry que ens permetrà després realitzar l'enviament de les dades, i en cas de pèrdua de la connexió, mantenir les lectures per enviar-les al servidor. Ho fem així ja que la gestió d'una BD és molt més neta que un fitxer i evitem possibles deadlocks en cas de fer lectures i/o enviaments simultanis. Des del codi Python del fitxer readSensors.py es desen les lectures a la BD segons els sensors disponibles. Només tenim una taula dins la base de dades anomenada punts, i a continuació es llista la taula i els seus camps:

Per a crear la base de dades executem la següent comanda:

```
pi@raspberrypi:~/RoMiBox $ echo "create database `readings`" | mysql -u username -p
```

```
MariaDB [readings]> use readings;
Database changed
MariaDB [readings]> show tables;
+--------------------+
| Tables_in_readings |
+--------------------+
| readings           |
+--------------------+
1 row in set (0.00 sec)

MariaDB [readings]> select * from readings;
Empty set (0.00 sec)
```

Figura 35: Taula de la base de dades SQL a la Raspberry Pi

Llavors creem la taula amb el fitxer SQL, que contindrà les columnes de cada lectura:

```
pi@raspberrypi:~/RoMiBox $ sudo mysql -u root readings < readings.sql
```

```
MariaDB [readings]> show columns from readings;
+-------------+------------+------+-----+---------+----------------+
| Field       | Type       | Null | Key | Default | Extra          |
+-------------+------------+------+-----+---------+----------------+
| id          | int(11)    | NO   | PRI | NULL    | auto_increment |
| romiboxid   | varchar(6) | NO   |     | NULL    |                |
| date        | varchar(8) | NO   |     | NULL    |                |
| time        | varchar(6) | NO   |     | NULL    |                |
| longitude   | double     | NO   |     | NULL    |                |
| latitude    | double     | NO   |     | NULL    |                |
| PM10        | double     | NO   |     | NULL    |                |
| PM25        | double     | NO   |     | NULL    |                |
| temperature | double     | NO   |     | NULL    |                |
| humidity    | double     | NO   |     | NULL    |                |
| CO          | double     | NO   |     | NULL    |                |
| CO2         | double     | NO   |     | NULL    |                |
| NOx         | double     | NO   |     | NULL    |                |
| confirm     | varchar(1) | NO   |     | NULL    |                |
+-------------+------------+------+-----+---------+----------------+
14 rows in set (0.02 sec)
```

Figura 36: Columnes de la taula readings de la BD SQL

9.3. Llegir sensors

El següent pas és llegir els sensors per tal de recollir les dades i desar-les localment. Posteriorment les enviarem al servidor per HTTPS. Utilitzarem Python 3.6 ja que el Python 2.7 inicialment utilitzat en l'altre projecte l'any que ve ja estarà descontinuat [16]. Això suposa que el codi existent serà convenientment traduït perquè continuï funcionant, ja que directament donava errors sintàctics.

9.3.1 HPMA115S0

Sensor ja utilitzat i llegit amb Python 2.6 amb anterioritat [17], durant l'inici del projecte, per tant hem de repassar el codi ja existent per utilitzar-lo amb Python 3.6. Hem d'activar el port sèrie amb la comanda raspi-config de la Raspberry Pi per a poder llegir i controlar el sensor a través dels ports TX i RX:

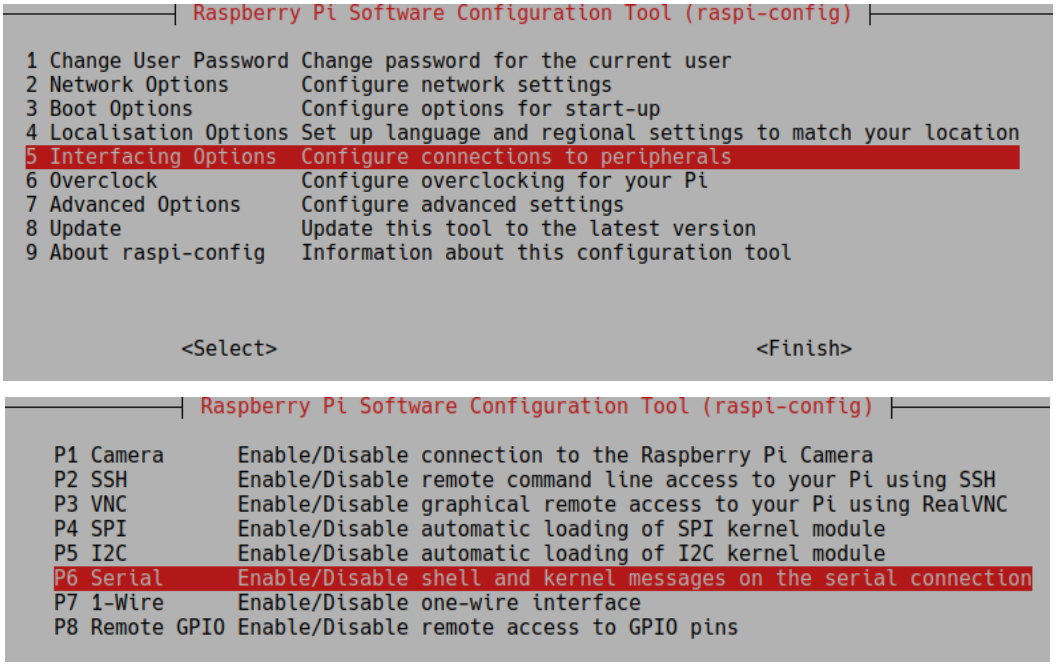


Figura 37: Configuració del port sèrie mitjançant raspi-config

Seleccionem que no volem un login shell a través del port sèrie. Una vegada activat el serial i reiniciada la Raspberry, i executat el codi, obtenim les lectures al cap d'uns 6 segons després de l'activació del codi Python. La connexió al sensor és Molex Picoblade de 8 pins i 1,25 mm entre pins, i la connexió de la Raspberry és amb pins GPIO. Vista la incompatibilitat i que volem evitar soldar per facilitar el muntatge, utilitzem els connectors ràpids pelant les puntes dels cables Molex i els de protoboard per a unir els pins corresponents:

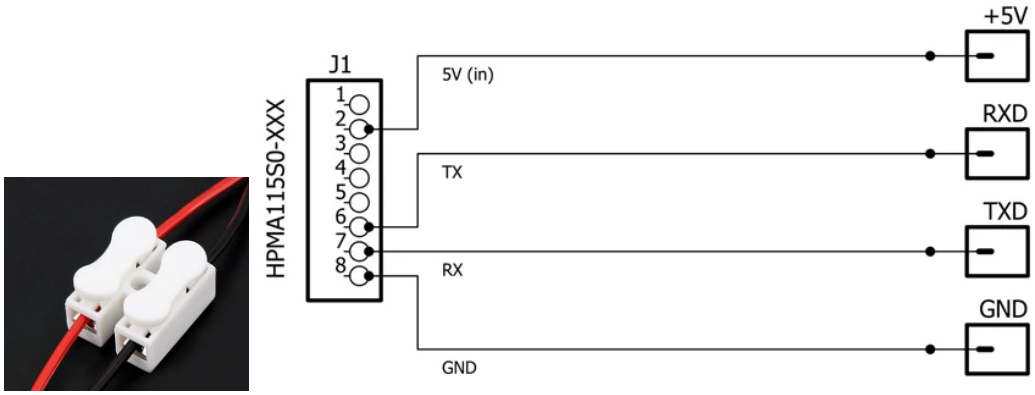


Figura 38: Connectors ràpids i esquema de connexió del sensor HPMA115S0

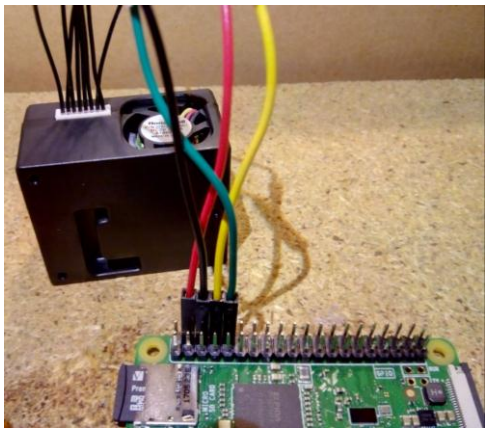


Figura 39: Connexió entre el sensor HPMA115S0 i la Raspberry Pi Zero

Vista la datasheet d'aquest sensor, és important remarcar que té dues posicions de treball correctes i dues incorrectes, ja que sinó s'acumularia pols durant el seu ús dins l'embolcall:

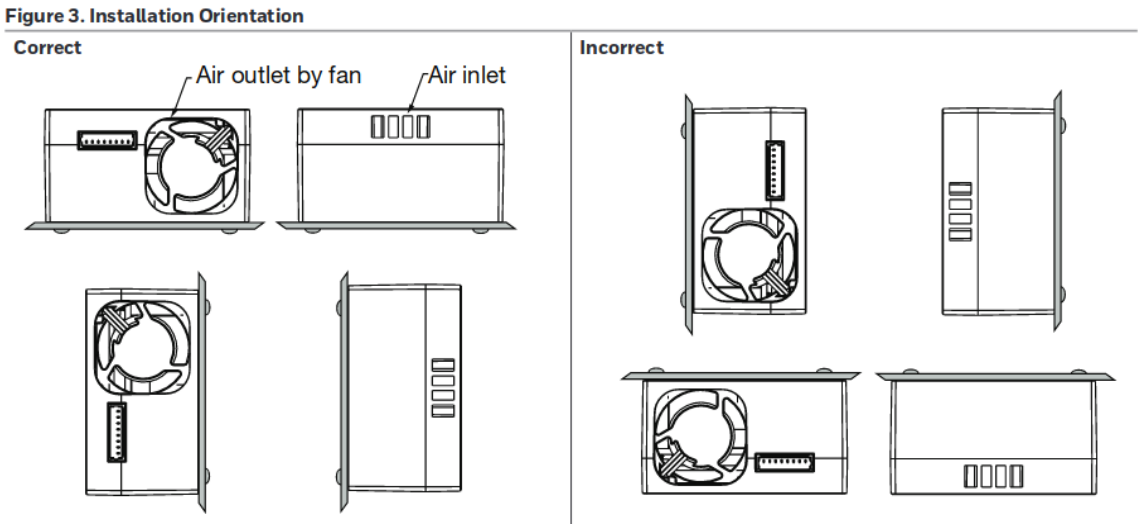


Figura 40: Posicions correctes i incorrectes del sensor HPMA115S0

9.3.2. DHT11

El sensor DHT11 té l'alimentació i una sortida digital llegible directament des dels GPIO de la Raspberry Pi, amb la qual cosa és el sensor més simple de connectar. L'esquema de connexió és el següent (aquí es mostra una Raspberry Pi 3 però els pins són els mateixos):

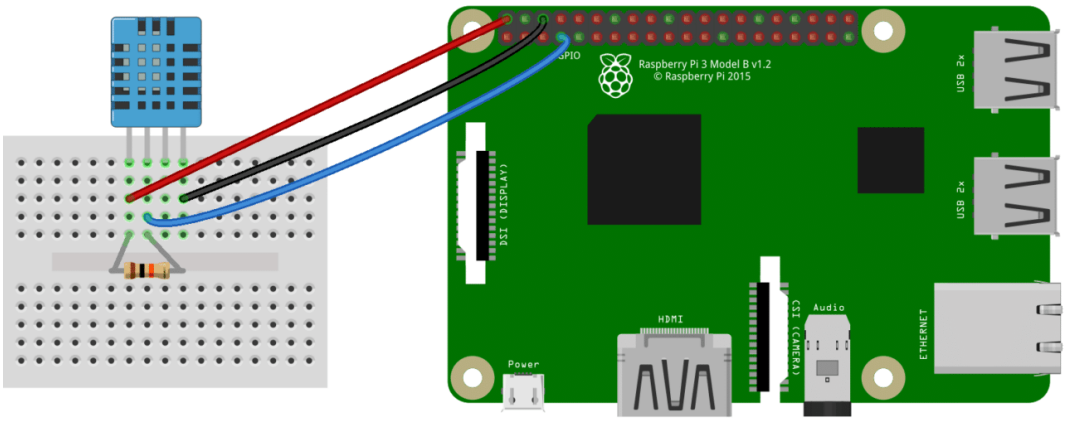


Figura 41: Connexió DHT11 amb resistència a part [18]

En el primer prototip vam escollir el sensor que no porta placa adjacent i per tant hem de connectar una resistència de 10K Ohm per a llegir correctament les dades. En futures unitats seria més aconsellable utilitzar el que ja porta la resistència integrada a part d'un forat a la placa per a fixar-lo millor. A continuació es pot veure la diferència entre ambdues versions (la nostra és la de la dreta):

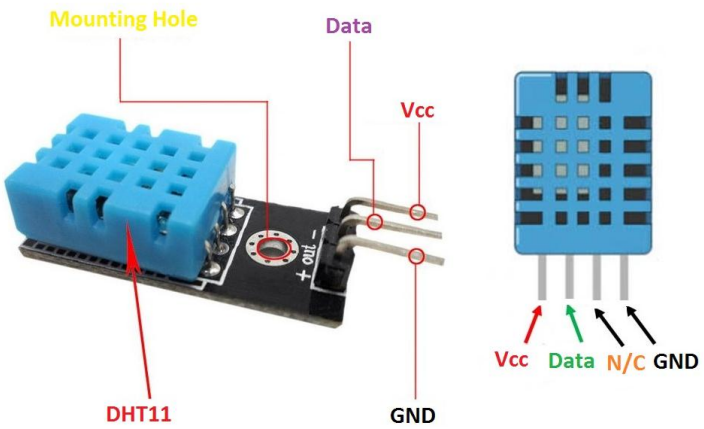


Figura 42: Pins de connexió sensor DHT11

Cal instal·lar la llibreria DHT11 per tal de llegir el sensor:

```
sudo pip3 install Adafruit_DHT
```

Executem l'script TestDHT11_v1.py i obtenim les lectures correctament (comparat amb una estació meteorològica interior):

```
pi@raspberrypi:~/RoMiBox/test $ python3 TestDHT11_v1.py
Temp=26.0*C  Humidity=56.0%
```

9.3.3. Preparació ADS1115

Com ja es comenta en l'apartat *Estudis i Decisions*, cal tenir en compte que per llegir els sensors MQ-135 i MQ-7 necessitem un conversor d'analògic a digital. Una vegada escollida aquesta opció, connectem els cables de la següent manera:

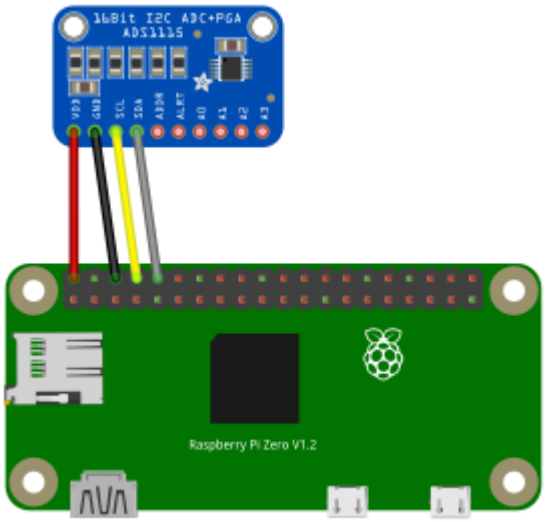


Figura 43: Connexió conversor A/D ADS1115 amb la Raspberry Pi

L'esquema mostra el xip d'Adafruit que també té les entrades ADDR i ALRT, en el nostre cas no són necessàries en el nostre ja que se seleccionen pels pins SCL i SDA.

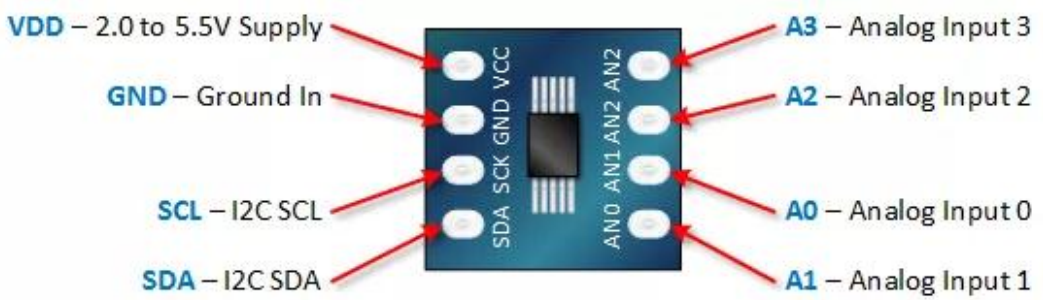


Figura 44: Pins ADS1115 simplificat

Una vegada connectat per I2C a la Raspberry Pi, hem d'habilitar-lo a través de la comanda raspi-config:

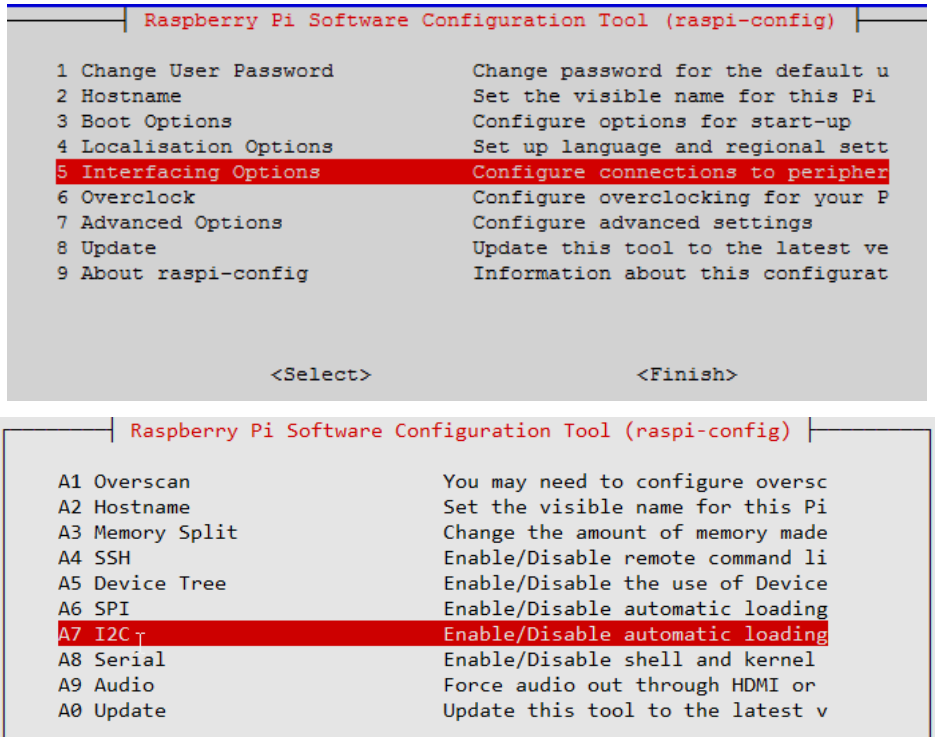


Figura 45: Habilitant I2C amb el raspi-config

Reiniciem la Raspberry i comprovem (instal·lant el paquet i2c-tools) que ja ens detecta el conversor. La opció -y 1 és el lloc on es munta el conversor, en el cas de Raspberry més antigues s'utilitzaria el port 0:

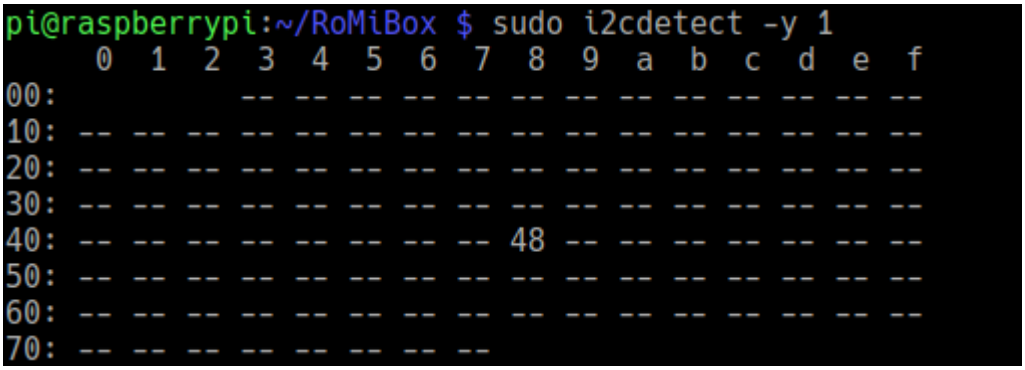


Figura 46: Prova de lectura del port del conversor per I2C

Un cop habilitat passem a connectar els 2 sensors analògics al conversor i a llegir-los.

9.3.4. MQ-135

La lectura del sensor MQ-135 serà molt similar a la del MQ-7, ja que els dos treballen amb el mateix principi d'una resistència de SnO2, que permet obtenir la quantitat en parts per milió en l'aire del component que mesuren. Aquesta resistència s'escalfa lleugerament per a realitzar les lectures, per això necessita un preescalfament abans de poder obtenir lectures precises. El que sí canviarà entre els dos és que els càlculs a aplicar segons la el voltatge que passa per la resistència seran diferents, la qual cosa es pot solucionar amb codi d'usuaris de GitHub que ja s'han trobat amb el mateix problema i han realitzat una calibració segons cada tipus de sensor. A continuació es pot veure un esquema simplificat per la connexió del sensor MQ-7 que serà la mateixa pel MQ-135, on s'usen

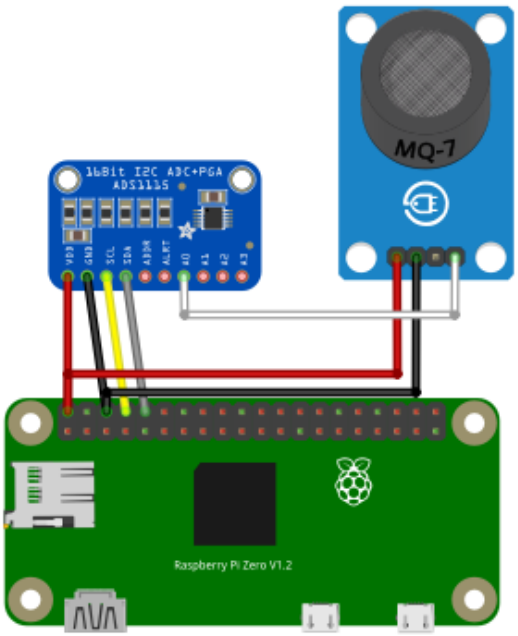


Figura 47: Esquema de connexió entre el sensor MQ-7, conversor ADS1115 i Raspberry Pi

Utilitzarem el port A0 (adreça 0) per al sensor MQ-135 i el A1 per al sensor MQ-7. Això ens deixa possibilitat per ampliar les lectures en un futur per a dos sensors analògics més, com podria ser el sensor d'O3 (MQ-131), entre d'altres.

Cal escalfar el sensor durant 24h continuades (igual que el MQ-7 i tots els sensors basats en SnO2, veure *Datasheet MQ-135* i *Datasheet MQ-7*). Aquest procés en diuen "burn" o cremat i es fa per estabilitzar la resistència i les seves posteriors lectures. Després d'aquest procés que realitzem amb una Raspberry apagada connectada a la corrent (segueix alimentant), ja podem realitzar lectures amb fiabilitat a partir d'entre 1 i 3 minuts d'escalfament.

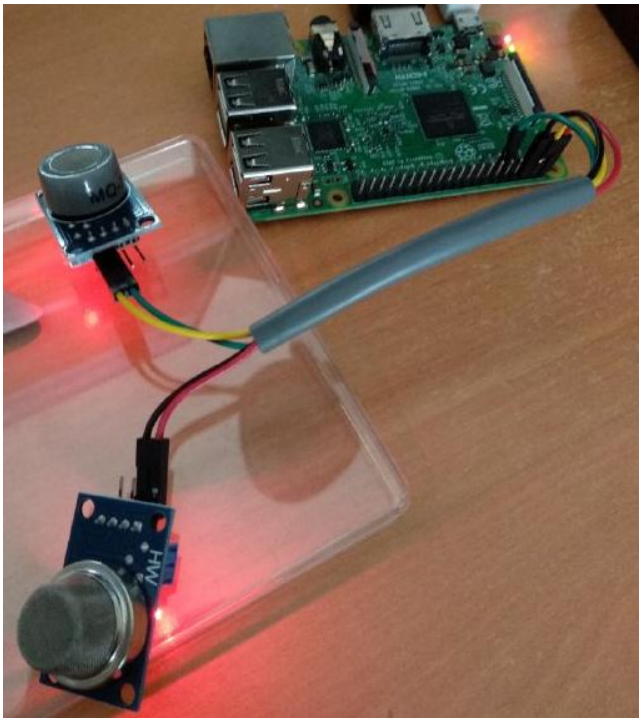


Figura 48: Cremat dels dos sensors durant 24h alimentats amb una Raspberry Pi 3B

Per tal de llegir del sensor a través del ADS1115, hem d'instal·lar la llibreria actual d'Adafruit que ens permetrà obtenir els voltatges que arriben de cada sensor. Executem les següents comandes amb l'instal·lador de paquets de Python3, pip3:

```
sudo pip3 install adafruit-circuitpython-lis3dh
sudo pip3 install adafruit-circuitpython-ads1x15
```

Un cop instal·lats els dos paquets (triguen bastant), podem provar l'script de lectura del MQ135 (veure codi en els annexos). Aquest retorna un valor en volts i un altre per la lectura obtinguda. Encara que podem llegir aquests sensors directament, ens trobem que cal calibrar-los comparant-los amb un sensor ja provat, la qual cosa dificultarà lleugerament la implementació. Veure fitxa *Technical data MQ-135 Gas Sensor* per a més informació. La idea és calibrar el primer amb un de professional que publiqui les dades i ajustar la resistència variable d'aquests sensors. En el cas del MQ7 serà necessari el mateix tipus de calibració.

A continuació es mostra la gràfica de correlacions entre gasos, on gairebé tots tenen una pendent semblant en la relació entre resistència oposada i ppm reals:

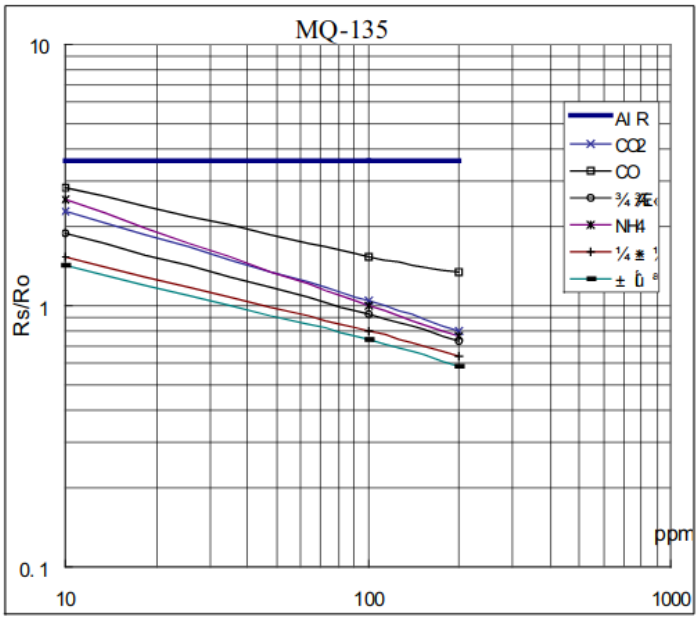


Fig.3 is shows the typical sensitivity characteristics of the MQ-135 for several gases.
in their: Temp: 20
Humidity: 65%
O_2 concentration 21%
RL=20kΩ
Ro: sensor resistance at 100ppm of NH_3 in the clean air.
Rs:sensor resistance at various concentrations of gases.

Figura 49: Correlacions entre els diferents gasos que pot detectar el sensor MQ-135

Si executem l'script TestMQ135.py (veure annex), veiem que ja mesura els PPM de CO2 amb dades bastant coherents (pendent de cal·libració):

```
MQ135 Gas Sensor:

        MQ135 RZero: 67
        Corrected RZero: 72
        Resistance: 43
        PPM: 567
        Corrected PPM: 466 ppm
```

Finalment, vist que el sensor només es pot mesurar per un tipus de gas i la resta serien càlculs estimats, escollim el CO2, ja que fer aquests càlculs sobre un sensor que també s'ha de calibrar creiem que generaria una possible desviació massa gran.

Una vegada llegit el sensor MQ-135 i vist que el conversor ADS1115 funciona, passem a llegir el sensor MQ-7.

9.3.5. MQ-7

Vista la lectura del MQ-135, i un cop fet el "cremat" de 24h de la resistència, ara caldrà obtenir el codi corresponent a aquest sensor. Les llibreries utilitzades per a l'ADS1115 són les mateixes

que per al sensor anterior, l'únic que canvia són els càlculs per a obtenir els valors de ppm per al CO en comptes del CO2 o el NOx.

L'esquema de connexió és el mateix mostrat en el cas del sensor MQ-135, connectant-lo al port A1 del conversor. La gràfica de correlacions per al MQ-7 és la següent:

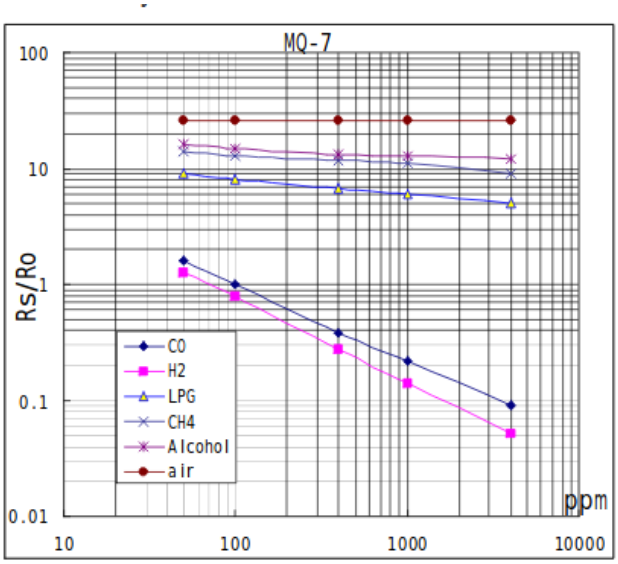


Fig.3 is shows the typical sensitivity characteristics of the MQ-7 for several gases.
in their: Temp: 20℃、
Humidity: 65%、
O_2 concentration 21%
RL=10kΩ
Ro: sensor resistance at 100ppm CO in the clean air.
Rs: sensor resistance at various concentrations of gases.

Figura 50: Correlacions entre els diferents gasos que pot detectar el sensor MQ-7

Una vegada configurats tots els sensors, ajuntem el codi per tal de realitzar les lectures des d'una sola crida i desar les dades a la BD. Veure annex ReadSensors.py per consultar el codi.

9.4. Enviar dades

L'enviament de dades es realitza de forma periòdica segons el que s'indiqui al fitxer XML de configuració. En l'enviament es limita a un màxim de 200 lectures per enviament per tal d'evitar un trànsit i posterior ocupació excessiva del servidor receptor de les dades.

L'enviament es fa periòdicament gràcies al crontab, on s'executa l'script de configuració a l'inici i aquest va realitzant l'enviament cada X segons tenint en compte el contingut del fitxer de configuració.

```
@reboot sudo python /home/pi/RoMiBox/Configure.py >/dev/null 2>&1
```

En el fitxer Configure.py es crida un while true amb sleep X segons, on X és el temps d'espera entre cada enviament.

```
command = "while true; do sudo python /home/pi/RoMiBox/SendData.py; sleep " + str(sendinginterval) + "; done"
process = subprocess.Popen(command, shell=True, stdout=subprocess.PIPE)
```

Després de realitzar diversos enviaments a la API, detectem el següent error:

```
requests.exceptions.SSLError: ("bad handshake: Error([('', 'osrandom_rand_bytes', 'getrandom() initialization failed.')],)",)
```

Una vegada tenim la base de la nostra solució preparada podem començar a configurar el Bluetooth per a poder enviar les dades de configuració des de l'aplicació, com intervals de lectura i enviament, ID del punt de lectura, i ubicació amb coordenades GPS del punt d'instal·lació.

9.5. Script Bluetooth si no hi ha Wi-Fi disponible

Inicialment la idea era fer un punt d'accés Wi-Fi des de la Raspberry en cas de pèrdua de connexió al Wi-Fi definitiu, i així poder configurar la RoMiBox des d'una app sense haver-nos de connectar amb el PC per SSH. Després de diversos scripts provats, s'acaba descartant l'ús del Wi-Fi com a mitjà de configuració ja que se succeïen molts problemes amb la connexió, degut a la falta de recursos de CPU observada des del visualitzador de tasques htop. Vist això, i que tenim connectivitat Bluetooth també integrada a la placa, decidim fer un script senzill per tal de mostrar-se disponible per Bluetooth si la RoMiBox no està connectada a cap Wi-Fi coneguda.

Cas 1: No té cap Wi-Fi desada coneguda, activa Bluetooth per realitzar configuració amb l'smartphone i va cercant xarxes Wi-Fi dins les conegudes periòdicament:

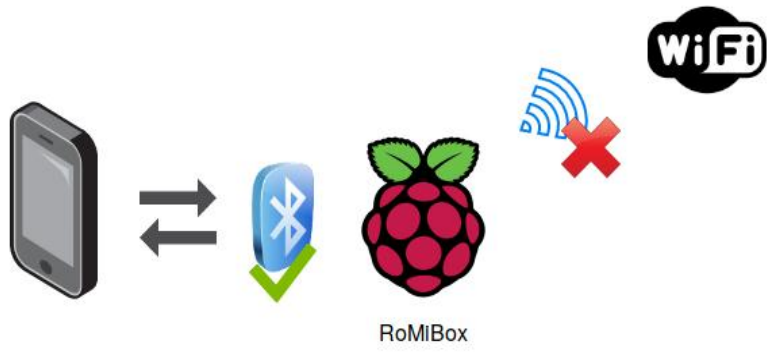


Figura 51: Comunicacions en funcionament durant la configuració de la RoMiBox

Cas 2: Connectat al Wi-Fi, cas desitjat com a definitiu durant el funcionament del punt de lectura. Periòdicament es consulta que l'estat segueixi sent el mateix, sinó tornem a l'escenari del cas 1:

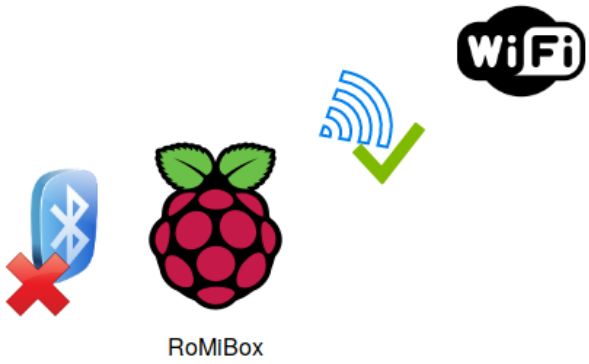


Figura 52: Comunicacions en funcionament durant les lectures de la RoMiBox

Com que en la primera arrancada no hi haurà cap Wi-Fi configurat, hem de tenir en compte els següents punts:

- Activar el Bluetooth de manera visible i protegit per PIN per a poder connectar l'app.
- Un cop es rebi la configuració de la RoMiBox (intervals de lectura/enviament, ID i model), caldrà activar les lectures i actualitzar el codi descarregant el disponible del repositori de GitHub.
- Mantenir la lectura de dades i enviaments segons els paràmetres de l'XML rebut.
- Actualitzar periòdicament els fitxers connectant-se al GitHub de RoMiBox a través d'un token dedicat, només de consulta, per tots els punts de lectura. Amb aquest token aconseguim no tenir el repositori obert i evitem manipulacions no desitjades del repositori.
- Una vegada actualitzats els fitxers, cal reiniciar serveis o bé tota la Raspberry Pi (això últim és bo per aplicar les actualitzacions de seguretat).

9.6. Programació de les lectures i actualitzacions

Les lectures i actualitzacions es realitzen periòdicament, executant-se el codi Python pertinent i obtenint la configuració segons un fitxer XML que conté els temps dels intervals d'execució, entre d'altres paràmetres:

```
▼<root>
   <id>TEST01</id>
   <reading_interval>30</reading_interval>
   <sending_interval>60</sending_interval>
   <latitude>3</latitude>
   <longitude>42.5</longitude>
   <romibox_type>STATIC_PRO</romibox_type>
 </root>
```

Figura 53: Fitxer XML de configuració de la RoMiBox de proves

Per tant, el que acabem fent és un script en Python amb el nom Configure.py (veure annex) que realitza dues tasques:

- Aplicar la configuració del Wi-Fi si encara no està feta. Un cop feta eliminem els SSID i password del fitxer XML.
- Iniciar els dos processos de lectura i escriptura amb els seus intervals corresponents, per a fer-ho executem un while true amb sleep entre cada execució:

```
command = "while true; do sudo python3 /home/pi/RoMiBox/ReadSensors.py; sleep
" + str(readinginterval) + "; done"
process = subprocess.Popen(command, shell=True, stdout=subprocess.PIPE)

#Set Execution time for sending

command = "while true; do sudo python3 /home/pi/RoMiBox/SendDataAPI.py; sleep
" + str(sendinginterval) + "; done"
process = subprocess.Popen(command, shell=True, stdout=subprocess.PIPE)
```

Figura 54: Execució dels script de lectura i enviament de dades al fitxer Configure.py

Les actualitzacions del codi es faran a partir d'un pull des de la branca master del projecte RoMiBox ubicat a GitHub, així assegurem tenir la disponibilitat d'aquest servei. S'aplica un token de només lectura per a totes les RoMiBox per a evitar modificacions malintencionades, i s'aplica al crontab una entrada @daily que farà una petició a una hora determinada del dia i s'aplicaran els canvis. El fitxer Update.py és buit i es pot utilitzar per a cridar scripts que s'afegeixin posteriorment i així poder realitzar multitud de tasques de manteniment com modificació de la base de dades, instal·lar paquets amb un fitxer .sh, etc.

```
@daily cd /home/pi/RoMiBox && git pull https://
oauth2:4b3a05ba15734829352e92616ae32cb365f63d97@github.com/mfarreras/RoMiBox.git
&& sudo python3 /home/pi/RoMiBox/Update.py >/dev/null 2>&1
```

Figura 55: Comanda utilitzada per l'actualització diària del codi

9.7. Aplicació Android

9.7.1. Configuració via fitxer XML enviat amb Bluetooth

Com es comenta a l'apartat *9.6. Programació de les lectures i actualitzacions*, la idea és tenir la configuració amb un fitxer de configuració XML que ens permetrà llegir-la posteriorment des dels script en Python. Per a configurar amb l'smartphone Android la nostra RoMiBox haurem de generar aquest fitxer des del mòbil i enviar-lo per Wi-Fi o Bluetooth a la Raspberry Pi. Es tractarà d'una aplicació molt simple on introduir els següents paràmetres:

- SSID del Wi-Fi on es connectarà la RoMiBox.
- Contrasenya Wi-Fi
- ID de la RoMiBox
- Interval de lectura (s) de tots els sensors
- Interval d'enviament (s) de les dades al servidor al cloud
- Latitud en coordenades GPS del punt de lectura
- Longitud en coordenades GPS del punt de lectura
- Model de caixa. Static o Static Pro segons si és un punt de lectura només de partícules o el complet.

Crearem un aplicació bàsica amb Android Studio i Java, on es podran introduir aquestes dades amb un botó d'enviament per enviar-les. Si falla l'enviament haurà de mostrar l'error. Creem un nou projecte i escollim una Activity amb un panell de navegació a la dreta per a futures implementacions de la consulta de dades:

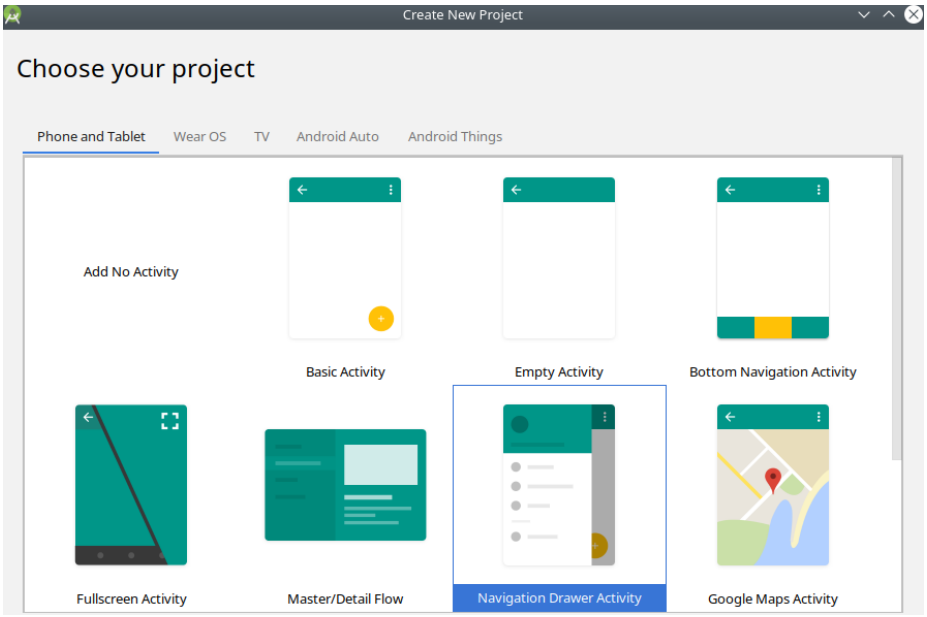


Figura 56: Tipus de layouts disponibles per a iniciar un projecte a Android Studio

Utilitzem la API 19 d'Android ja que és ens ofereix les funcionalitats que necessitem, té millores de l'emmagatzematge (ideal per desar el fitxer XML generat), millores en les notificacions, per si s'utilitzés aquesta app per consultar les dades en temps real i generar alertes, i dóna compatibilitat amb la gran majoria de dispositius recents (95.3 % de dispositius compatibles en funcionament) [20].

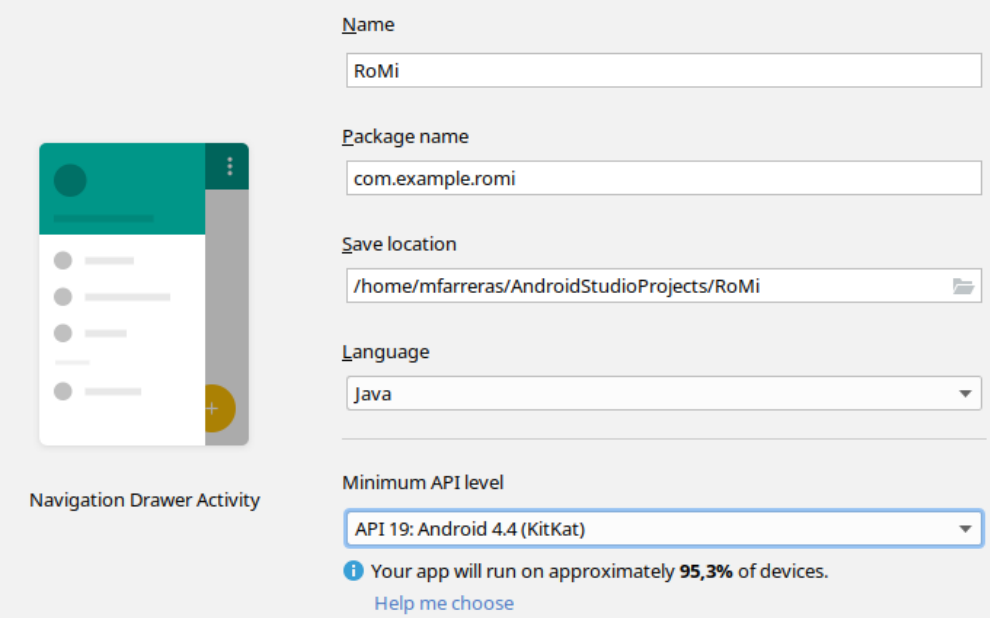


Figura 57: Configuració del nostre projecte RoMiApp

9.7.2. Configuració Wi-Fi

El problema d'utilitzar un fitxer de configuració en XML és que la contrasenya del Wi-Fi pot contenir caràcters, per tant finalment optem per un fitxer JSON. Aquest contindrà totes les dades de configuració abans esmentades i s'esborraran les configuracions del Wi-Fi al desar hashejada la contrasenya al fitxer wpa_supplicant.conf. Es genera també un script configureWifi.sh on s'edita el fitxer de configuració del wpa_supplicant.conf i s'elimina la configuració del fitxer JSON. És necessari canviar al fitxer JSON (encara que després s'ha hagut de canviar bastant de codi), ja que la codificació de cada caràcter d'una contrasenya ens portaria massa complicació innecessària.

Per tant el fitxer de configuració resultant de la nostra RoMiBox de proves tindria el següent aspecte:

```
{
    wifi_ssid: "TEST"
    wifi_password: "Prova1234%&"
    id: "TEST01",
    reading_interval: 30,
    sending_interval: 60,
    latitude: 3,
    longitude: 42.5,
    romibox_type: "STATIC_PRO"
}
```

I la interfície gràfica de l'aplicació tindrà una activity amb els paràmetres necessaris per la configuració de la RoMiBox:

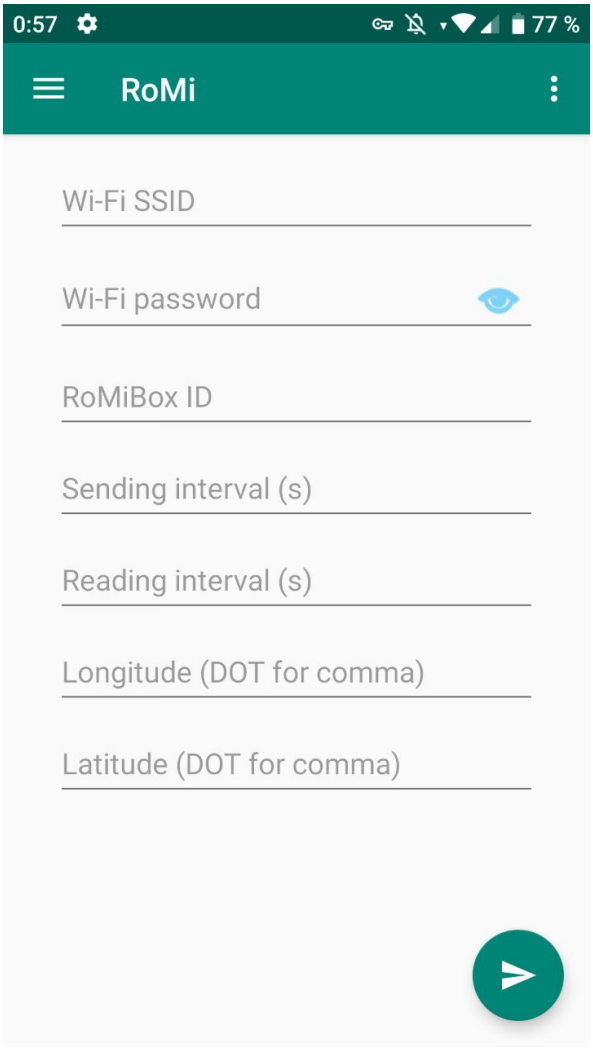


Figura 58: Única pantalla de l'aplicació RoMiApp

9.8. Prototip físic d'implementació

Com es comenta en l'apartat Estudis i decisions, la solució candidata a ser definitiva i que sembla més adequada per les seves qualitats és la caixa d'unions elèctriques. La mida final és de 200 x 100 x 70. A continuació es descriu la disposició dels components dins la caixa:

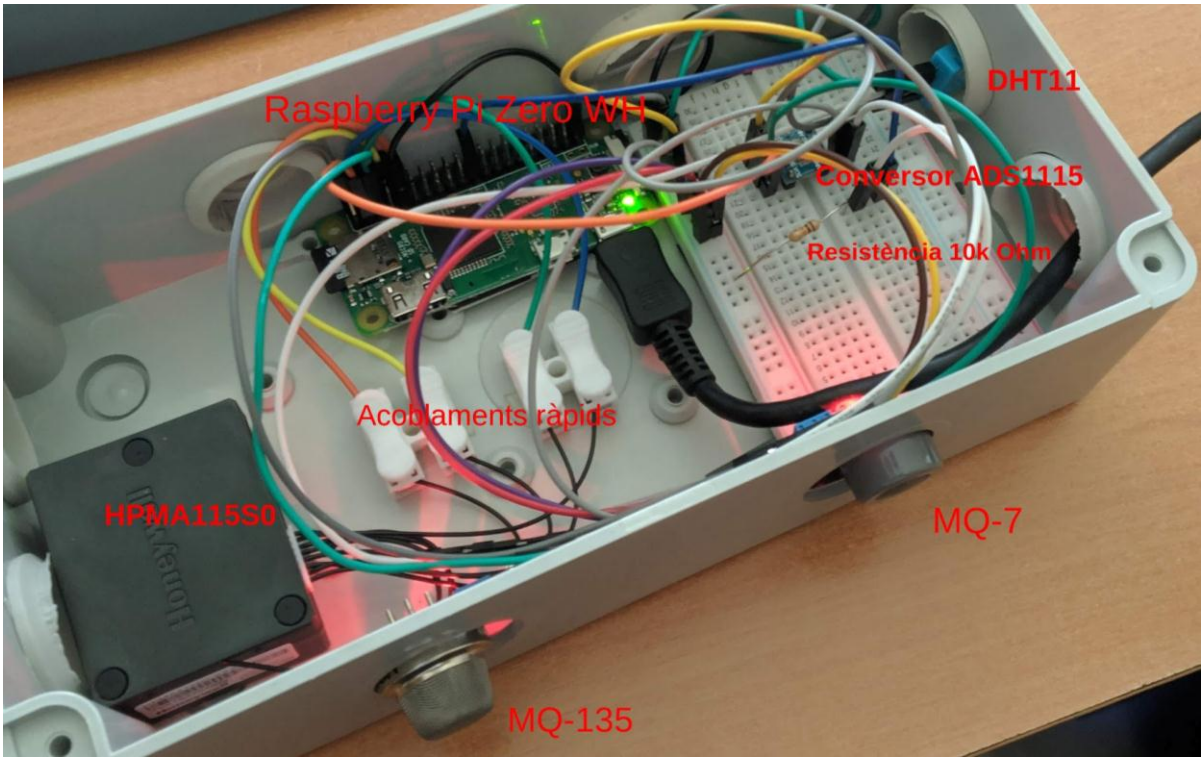


Figura 59: Components muntats dins la RoMiBox

Tots els components s'han fixat amb la cinta de doble cara, que proporciona una subjecció molt elevada.

L'entrada del cable es fa a través d'un petit forat a una de les tapes de goma. La tapa davantera té una goma incorporada amb la qual cosa es tracta d'un embolcall molt resistent a l'aigua. Hi ha molt poques probabilitats de que entri aigua ja que els únics orificis queden a la part de baix i per tant l'aigua per gravetat no hauria d'accedir a la caixa. A part, es col·loquen els sensors que necessiten contacte amb l'exterior a la part de baix i els laterals, de manera que els dos laterals tenen un tall de la tapa només per la part de baix per actuar com a barrera.

Aquest disseny ens permetria afegir còmodament un altre sensor al costat del HPMA115S0, a més el conversor ADS1115 té disponibles dues entrades més per convertir dos sensors analògics extra. També hi hauria la possibilitat de foradar la part inferior de la caixa per afegir els sensors extra necessaris o incrementar la protecció contra la pluja si no es volen utilitzar els forats laterals, però en el nostre cas no es considera necessari.

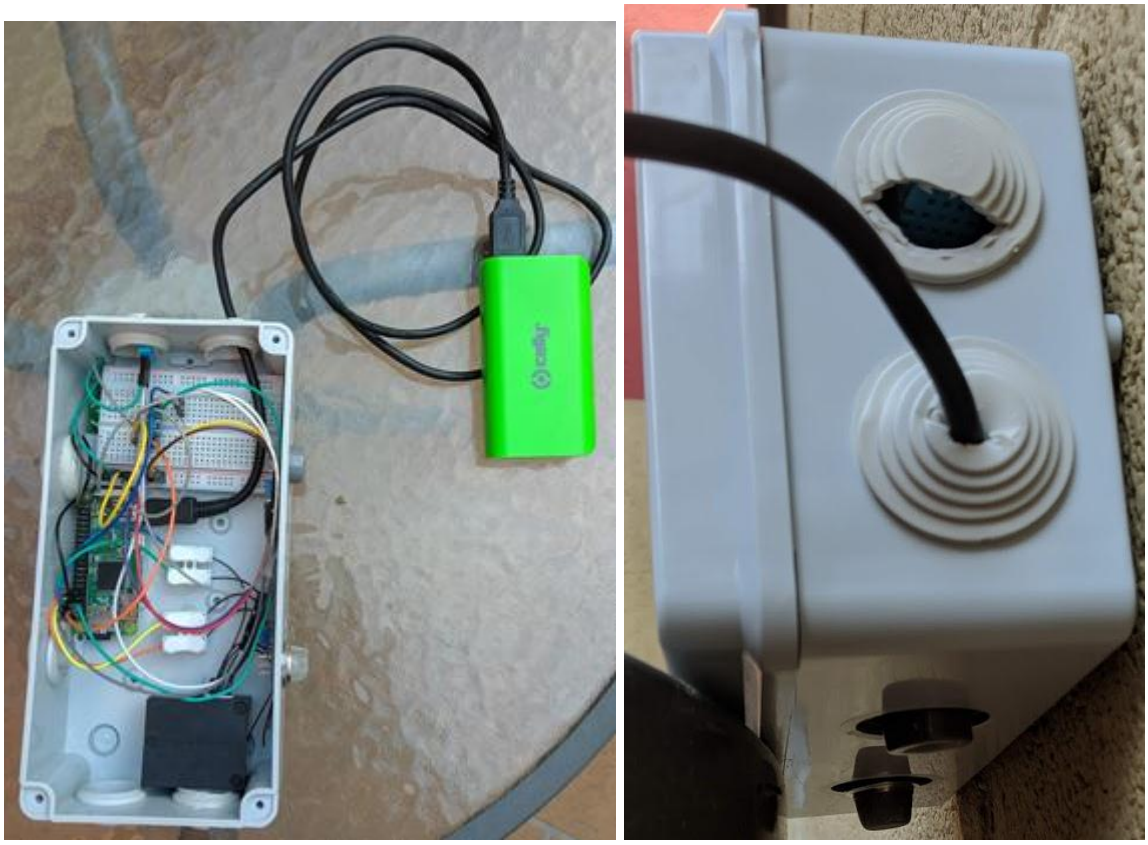


Figura 60: Prova a l'exterior amb bateria i detall dels orificis de la caixa

Una curiositat és l'aspecte d'il·luminació vermella que asoleix l'interior de la caixa quan està en funcionament en l'obscuritat i sense tapa (amb la tapa instal·lada no s'observa cap tipus d'il·luminació a l'exterior). Això és degut a que els dos sensors MQ tenen un LED vermell que indica el seu estat de funcionament i il·luminen tot l'interior de la caixa. El LED verd de la Raspberry indica la seva activitat també:

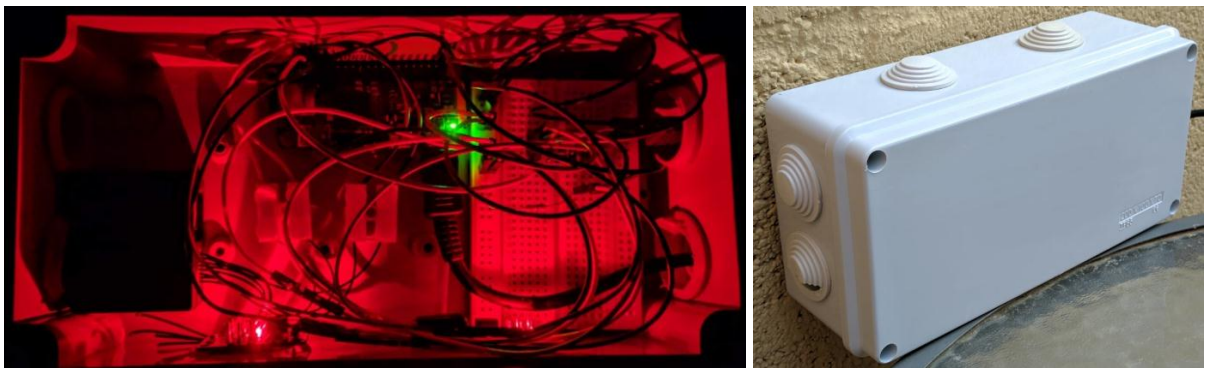


Figura 61: Il·luminació nocturna i aspecte amb la tapa tancada

10. Implantació i resultats

A continuació es descriuen els resultats i proves realitzats sobre el prototip muntat.

10.1. Proves ambientals a l'exterior/intempèrie

Les proves a la intempèrie ja van resultar bastant satisfactòries amb les caixes acríliques, encara que un dels 10 sensors instal·lats va resultar afectat per una pluja torrencial pocs dies després de ser instal·lat. Com que no ha plogut els dies de la prova a l'exterior amb la caixa de connexions o bé ha sigut pluja molt superficial, es decideix ruixar amb una mànega la caixa des de diferents angles, de manera que simulem una pluja forta (sense aplicar molta pressió però amb quantitat d'aigua). El resultat és que no ha entrat ni una gota d'aigua dins la caixa, amb la qual cosa segueix funcionant sense problemes.

Figura 62: Caixa després de la prova de mullat i caragols utilitzats pel tancament, ja inclosos en la compra de la caixa

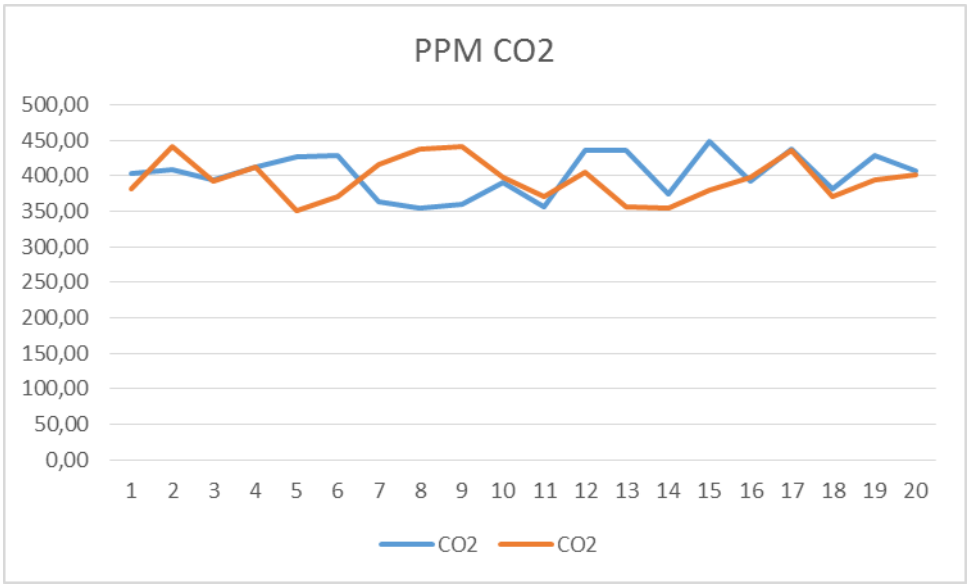
Caldria provar en ambient fred si es mantenen les lectures correctes dels sensors però en aquesta època de l'any (maig) no es produeixen glaçades.

Per assegurar l'estanqueïtat es podria aplicar silicona al voltant de cada goma, i afegir uns tubs al voltant dels sensors MQ que surten a fora, perquè no els toqués cap gota. Es tracta de dues millores senzilles i que deixarien la caixa molt protegida davant de qualsevol inclemència meteorològica.

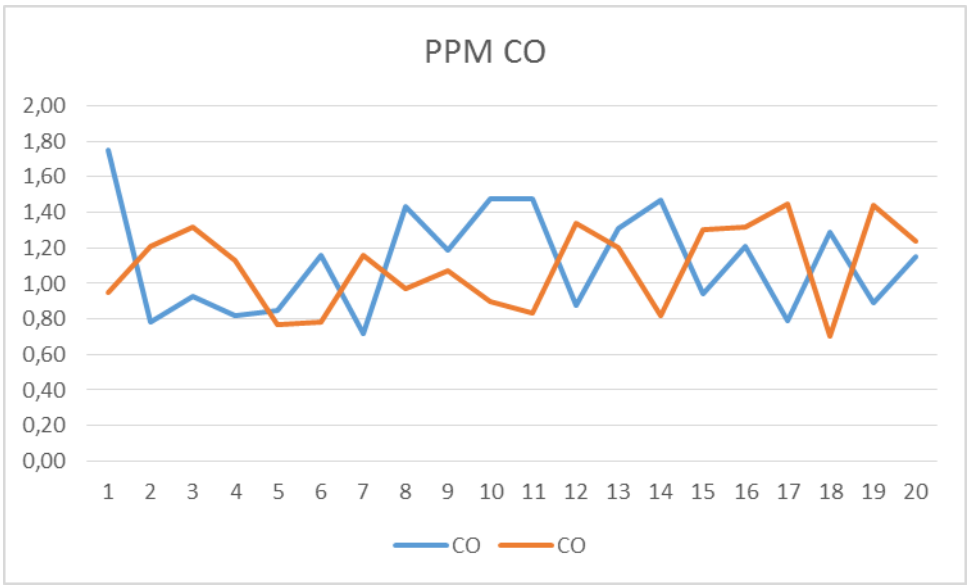
10.2. Comparació de lectures entre sensors del mateix tipus

Després del cremat dels sensors MQ que es van comprar a part (dues unitats de cada), es realitza una prova de 10 minuts amb lectures cada 30 segons, des de dos Raspberry Pi diferents i s'obtenen les següents lectures:

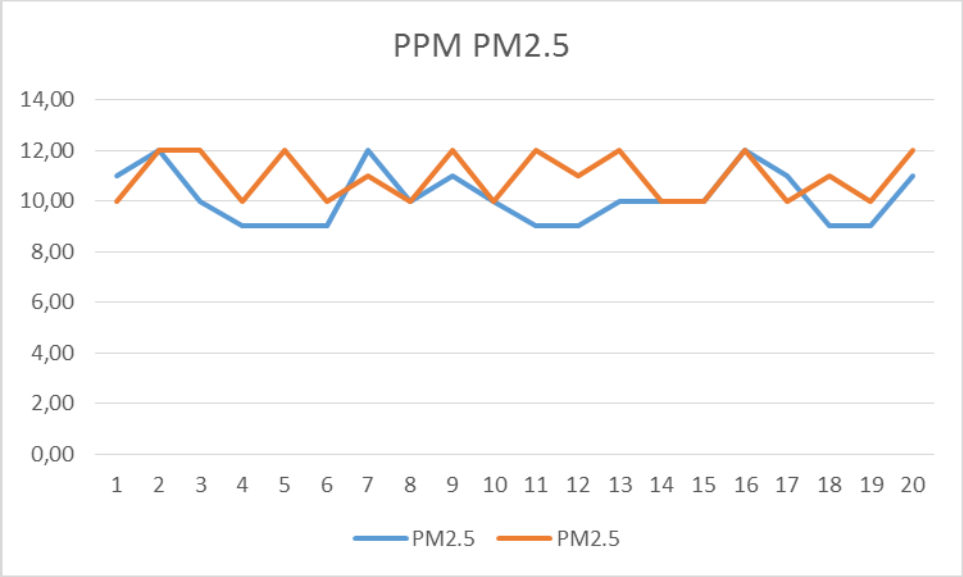
MQ-135: Presenta bastanta estabilitat en les lectures de PPM de CO2.

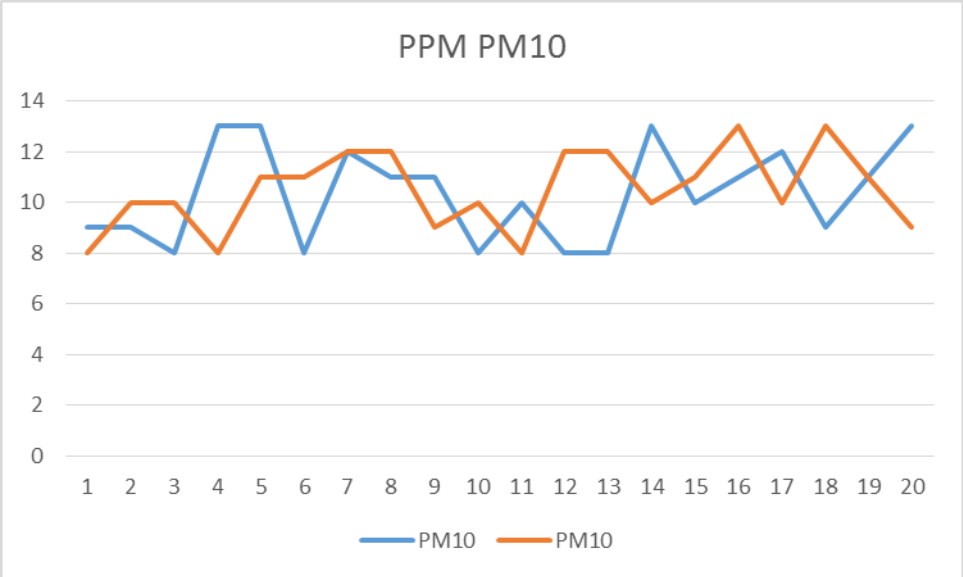


MQ-7: Té un pic puntual en la primera lectura i després s'estabilitza lleugerament.



HPMA115S0: Lectura de PM 2.5 més estable que la de PM10, que presenta més variació.





Amb el DHT11 no hem pogut realitzar aquestes proves, perquè es va haver de canviar per problemes amb el primer sensor i no teníem més disponibilitat d'aquest.

10.3. Comparació de lectures amb sensors professionals

Aquest apartat no ha donat temps a portar-lo a terme ja que fins a finals de maig encara ens falta el permís per instal·lar un sensor just al costat del de la Generalitat. De totes maneres s'instal·larà segurament durant el mes de Juny que és el que ens van prometre.

També ens hem posat en contacte amb el departament de Física Ambiental de la UdG, amb l'Institut de Medi Ambient i amb el grup de recerca Lequia i cap tenia a la seva disposició

aparells per a la mesura de la qualitat de l'aire, amb la qual cosa haurem d'esperar a tenir disponible la instal·lació a l'Escola de Música per a poder comparar correctament les dades.

10.4. Proves de pèrdua temporal de connectivitat

La pèrdua de connectivitat fa que s'acumulin els punts a la base de dades. Això no suposa cap problema ja que tenim diversos gigues lliures d'emmagatzematge. Quan es recupera la connectivitat Wi-Fi s'envien els punts de 200 en 200 per a no col·lapsar el servidor com va passar amb els punts de lectura anteriors. Als log del servidor es pot observar com s'envien de 500 en 500 si falla l'enviament (paràmetre anterior a 500 que s'ha reduït a 200):

```
[2019-05-22 07:55:19] UDG005 -> Total de punts GPS: 500
[2019-05-22 07:56:11] UDG006 -> Total de punts GPS: 500
[2019-05-22 07:56:11] UDG001 -> Total de punts GPS: 500
[2019-05-22 07:56:11] UDG004 -> Total de punts GPS: 500
```

10.5. Pèrdua d'alimentació

La pèrdua d'alimentació fa que es reinicii la Raspberry i que per tant hi hagi una interrupció en les lectures del punt de lectura afectat. Sense una bateria no podem canviar aquest comportament, però la fiabilitat de les elèctriques en les grans ciutats és bastant elevada i no solen afectar més que uns pocs minuts, amb la qual cosa no seria un punt crític.

Existeix la possibilitat remota que es corrompi el sistema operatiu però és un cas molt remot, les nostres caixes de test s'han apagat multitud de vegades sense donar cap problema d'aquest tipus. Segons l'article [22], és sobretot quan s'engeguen i apaguen que pot passar.

10.7. Actualització remota

L'actualització remota també funciona correctament, per a provar-la comprovem que el fitxer Update.py està buit. El modifiquem a GitHub (branca master) i esperem un dia a comprovar que realment s'ha actualitzat:

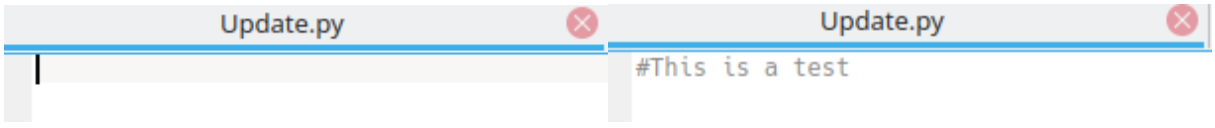


Figura 63: Abans i després de l'actualització

10.8. Problemes durant el desenvolupament

10.8.1. Configuració via hotspot Wi-Fi

La configuració des d'un smartphone inicialment es va fer a través d'un hotspot Wi-Fi que es creava quan no hi havia cap connexió disponible coneguda. Aquest mètode ens causava talls molt llargs en connexions ja establertes degut a l'alt consum de CPU en cada comprovació. Finalment s'ha optat per utilitzar el mateix mètode amb un script senzill que permet comprovar si tenim connexió Wi-Fi i sinó s'activa el Bluetooth, d'aquesta manera el consum de CPU sembla ser que és més baix i per tant deixen d'existir els talls.

10.8.2. Talls en la connexió per SSH

Els talls en la connexió per SSH via Wi-Fi van ser deguts precisament al punt anterior, per tant ja estan solucionats. Sí que quan es realitzen lectures dels sensors i enviaments es produeix un petit col·lapse d'1-2 segons que fa que deixi de respondre momentàniament a les pulsacions de teclat però és normal ja que tenim els recursos molt limitats. A més només ens connectarem per SSH si es realitza una configuració via app o bé estem realitzant proves de desenvolupament, no durant la posada en producció del punt de lectura.

10.8.3. Dificultats ADS1115

El sensor ADS1115 és bastant utilitzat tot i que el MCP3008 (que és més complex en quant a connexions) és més popular i té més suport de la comunitat d'usuaris. L'avantatge del nostre és que ja ve amb els pins soldats i per tant facilita el muntatge del punt de lectura, a part que s'han publicat noves llibreries en Python3 per a poder seguir utilitzant-lo (recordem que Python 2 acaba el seu suport el 1 de gener del 2020).

10.8.4. Traducció Python 2.7 a 3.6

La resta del codi que s'havia fet s'ha traduït de Python 2.7 a 3.6, de manera que evitem que quedi també fora de suport. Això ha provocat que haguéssim de canviar diverses llibreries utilitzades durant el principi del projecte però no ha suposat gaire més problema ja que el codi seguia els estàndards de gramàtica del Python actual. Sí que ha calgut modificar el codi del sensor HPMA115S0 ja que estava fet per Python 2 i encara no hi havia cap equivalent disponible per al Python 3. També s'han hagut de reinstal·lar els paquets amb pip3.

10.8.5. Col·lapses del servidor al enviar dades

Durant les proves amb els punts de lectura que només contenien un sensor Honeywell HPMA115S0 (de PM 2.5 i PM 10), es van instal·lar 10 punts que enviaven periòdicament dades al servidor romi.gq amb PHP sense la nova API. La confirmació dels enviaments es realitzava amb una altra petició i aquesta va deixar de respondre, la qual cosa feia que s'acumulessin les lectures i cada vegada s'enviessin paquets més grans (limitat a 500 paquets per enviament). Amb la creació de la nova API per part d'en Robert i l'enviament més eficient es va solucionar l'inconvenient.

10.8.6. Eliminació accidental de punts no confirmats

El punt anterior va servir també per observar que els punts no confirmats s'acabaven perdent després del primer enviament. Calia eliminar només els ja enviats (i s'estaven eliminant tots). Es tractava d'un error de disseny en l'enviament des de la Raspberry, que ha estat corregit marcant com a pendents només els últims que s'han intentat.

Anteriorment es marcava com a pendent un grup de lectures, s'enviava i si fallava l'enviament quedaven marcats com a pendents de confirmar. Si es feia un altre intent amb un altre grup de punts, aquests també quedaven pendents de confirmar. Si l'enviament era correcte, aquests i el grup anterior s'eliminaven de la base de dades de la Raspberry provocant aquesta pèrdua de lectures. La solució és desmarcar qualsevol punt pendent abans de fer un intent. A continuació es pot veure l'UPDATE amb SQL per evitar aquesta casuística:

```
comanda = "UPDATE readings SET confirm = 2 WHERE confirm=1 LIMIT 200;"
db.insert(comanda)
select_query = """SELECT * FROM readings WHERE confirm = 2 ORDER BY date DESC, time DESC;"""
```

En les comandes anteriors s'ha aplicat la correcció, on el LIMIT 200 s'aplica al fer l'UPDATE de les línies. Si falla l'enviament desmarquem qualsevol línia que estigui pendent de confirmar, aquí encara s'enviava per PHP:

```
    if int(exit) == int(nPunts) and message == "SUCCESS":
        data = {'puntidvehicle':id_bus,
            'npunts':nPunts}
        r = requests.post(url = URL2, data = data)
        comanda = "DELETE FROM readings WHERE confirm=2;"
        db.insert(comanda)
    else:
        comanda = "UPDATE readings SET confirm = 2 WHERE confirm=1;"
        db.insert(comanda)
```

10.8.7. Subministrament lent de components

Per la compra de sensors a un cost baix, teníem diverses opcions, com Amazon, Aliexpress o altres distribuïdors més dedicats a aquesta tasca. Aquests últims tenien un preu normalment superior a la resta i les existències o bé eren molt limitades o l'enviament també tardava molt. Des d'Amazon l'enviament dels de cost més baix trigava com els d'Aliexpress (15 dies aproximadament), mentre que el preu de compra d'aquest últim era més baix. Per això s'ha utilitzat com a proveïdor principal. Des d'Amazon hi havia disponibilitat en 2-4 dies però el preu es multiplicava per 4 en moltes ocasions. Per exemple per comprar el sensor MQ-135 de la mateixa qualitat:

Elenxs MQ135 Módulo Sensor de Calidad de Aire nocivo detección de Gases - Negro
de Elenxs

Sé el primero en opinar sobre este producto

Precio: EUR 2,20 Envío GRATIS.
Precio final del producto

Nuevos: 1 desde EUR 2,20

- Gran aplicación para proyectos de bricolaje.
- Otro de salida: 4 actual; de salida de pico: 3A.
- Tensión de alimentación: 4.8V

EUR 2,20
Envío GRATIS.
Recíbelo entre 21 jun. - 1 jul. al elegir Entrega estándar durante la tramitación del pedido. Ver detalles

En stock.

Cantidad: 1

EUR 2,20 + Envío GRATIS

Añadir a la cesta

Comprar ya

Vendido y enviado por ESElenxs.

Figura 64: Cost barat des d'Amazon, però enviament lent

1 unidad ShengYang MQ135 MQ-135 Sensor de calidad del aire Módulo de detección de Gas peligroso

Ver nombre original del producto en inglés

4.8 (91 votos) | 189 vendidos

Precio: € 1,09 / unidad

Oferta: € 1,02 / unidad -7% Termina en 2 días

Envío: € 0,40 a Spain vía AliExpress Saver Shipping

Tiempo de entrega: 17 días

Cantidad: – 1 + unidad (1266 unidades disponible)

Comprar ahora | Añadir a la cesta | 171

Figura 65: Cost més barat des d'Aliexpress, enviament igual de lent

Figura 66: Cost relativament car des d'Amazon, amb enviament ràpid

10.8.8. Fallada sensor DHT11

Després de la realització de les proves bàsiques de funcionament, el sensor de temperatura i humitat va començar a donar 28 ºC tota l'estona. Sospitem que va ser degut al connectar aquest sensor directament sense resistència durant les primeres lectures. Finalment canviant el sensor es va solucionar (surten 33 i 32 ºC degut al contacte amb les mans), a part que les lectures van passar a ser molt més ràpides:

```
pi@raspberrypi:~/RoMiBox $ python3 test/TestDHT11.py
Temp=28.0*C  Humidity=47.0%
pi@raspberrypi:~/RoMiBox $ python3 test/TestDHT11.py
Temp=28.0*C  Humidity=49.0%
pi@raspberrypi:~/RoMiBox $ python3 test/TestDHT11.py
Temp=28.0*C  Humidity=50.0%
pi@raspberrypi:~/RoMiBox $ python3 test/TestDHT11.py
Temp=28.0*C  Humidity=49.0%
pi@raspberrypi:~/RoMiBox $ python3 test/TestDHT11.py
Temp=28.0*C  Humidity=73.0%
pi@raspberrypi:~/RoMiBox $ python3 test/TestDHT11.py
Temp=33.0*C  Humidity=89.0%
pi@raspberrypi:~/RoMiBox $ python3 test/TestDHT11.py
Temp=32.0*C  Humidity=65.0%
```

Figura 67: Lectures del sensor defectuós sense canvis i posterior lectura més elevada del nou al haver-lo tocat amb les mans

10.8.9. Contrasenya amb símbols especials en XML

Després d'haver implementat el fitxer de configuració, ens vam adonar que al enviar la contrasenya amb símbols especials a través d'un fitxer XML ens estava donant certs problemes. La causa era que XML codifica els caràcters especials d'una altra manera [22], i per tant la Raspberry rebia un codi XML incorrecte. Per exemple, el caràcter "&" es codifica com a

“&”, amb la qual cosa quedava un fitxer incorrecte d’XML i no es podia llegir per part del codi Python que el necessitava. La solució final i més simple ha sigut modificar el tipus de fitxer a JSON per a poder inserir qualsevol string en el camp de contrasenya, SSID o també l’ID de la RoMiBox.

11. Conclusions

Amb aquest projecte hem pogut desenvolupar una solució fixa per a la lectura dels principals contaminants que es poden trobar en l'aire de les ciutats amb un cost baix i un muntatge relativament fàcil.

La configuració remota amb l'aplicació per Android també facilita molt la instal·lació, a més d'aprofitar moltes de les capacitats de la Raspberry Pi Zero WH fent servir el Bluetooth. Aquesta placa és una plataforma molt modular i ideal per aquest tipus de projectes, ja que es pot consultar un gran ventall de sensors.

Hi ha hagut alguns punts tècnics que han donat certa lentitud al desenvolupament, com és el cas del pas a Python 3.6, les pèrdues de connexió, el fitxer XML, etc., però s'han solucionat a temps. També ha sigut un impediment no trobar sensors professionals amb els que comparar resultats o bé no obtenir els permisos a temps.

Alguns punts encara es poden millorar, com es pot veure en l'apartat de treball futur, però creiem que en relació qualitat preu és un sensor molt complet i amb bona capacitat per al seu desplegament en moltes ciutats, gràcies a la facilitat d'instal·lació i el preu.

De cara al treball de final de màster de l'alumne Robert Garcia Ventura pensem que les dades seran de prou qualitat com per poder analitzar-les un cop desplegats els sensors, i la posterior consulta en directe serà d'un gran interès, entre d'altres tasques que podrà realitzar.

Aquest projecte ens ha permès aplicar diversos coneixements obtinguts durant el grau i el màster, com ara de l'assignatura "Projecte de sistemes encastats i Ubicus" per a tota la part de hardware i comunicació amb els sensors, mentre que també es tenen en compte bastants conceptes de l'assignatura "Disseny i implementació de xarxes i sistemes distribuïts" per a les comunicacions entre l'smartphone i la RoMiBox, i l'enviament de dades al núvol.

Personalment m'ha agradat molt la realització d'aquest treball i la nostra intenció és continuar desenvolupant la solució per a oferir la possibilitat de tenir una xarxa de sensors barats i precisos. Es continuarà amb el desplegament de sensors per diverses institucions de la ciutat de Girona, començant per l'Escola de Música on hi ha un sensor professional de la Generalitat.

12. Treball futur

12.1 Versió amb sensor GPS

Finalment aquest projecte s'ha encarat per a versions fixes dels punts de lectura, degut a la variabilitat de lectures d'un punt mòbil que ens podria proporcionar dades errònies. De totes maneres seria interessant desenvolupar la solució amb GPS (com es va fer en l'assignatura de Projecte de Sistemes Encastats i Ubicus) per tal de poder disposar d'un sensor personal, fins i tot wearable o ubicat en vehicles.

12.2. LoRaWAN

Amb LoRaWAN podríem comunicar-nos sense dependre del Wi-Fi, encara que també cal que en un radi mínim entre 2 i 10 km hi hagi una gateway (conversor de LoRa a Internet). A Girona s'està implantant amb la qual cosa seria viable desenvolupar-ho.

12.3. Arduino amb bateries

Vista la mobilitat d'una versió amb GPS, també podríem optimitzar el consum energètic (tenint menys funcions) amb una Arduino Uno o Nano, i així fer una versió mòbil amb bateries. També serviria per instal·lar-la independentment de qualsevol endoll.

13. Bibliografia

[1] SmartCAT Challenge 2018.
http://smartcatalonia.gencat.cat/ca/projectes/ciutats_i_regions/smartcat-challenge/edicions-anteriors/smartcat-challenge-2018/
[2] Diferència entre llicències GPL i MIT.
https://stackoverflow.com/questions/3902754/mit-vs-gpl-license
[3]The Things Network Girona.
https://www.thethingsnetwork.org/community/girona/
[4] Metodologies de desenvolupament àgil.
https://blog.conectart.com/metodologias-agiles/
[5]Internet of Things segons Deloitte.
https://www2.deloitte.com/es/es/pages/technology/articles/IoT-internet-of-things.html
[6] Conceptos sobre APIs Rest.
https://asiermarques.com/2013/conceptos-sobre-apis-rest/
[7] ¿Qué es una API REST?.
https://www.idento.es/blog/desarrollo-web/que-es-una-api-rest/
[8] Major air pollutants.
https://www.infoplease.com/math-science/earth-environment/major-air-pollutants
[9] Haciendo IoT con LoRa: Capítulo 1.- ¿Qué es LoRa y LoRaWAN?.
https://medium.com/beelan/haciendo-iot-con-lora-cap%C3%ADtulo-1-qu%C3%A9-es-lora-y-lorawan-8c08d44208e8
[10] Raspberry Pi Model 3 B+.
https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/
[11] Raspberry Pi Zero WH
https://www.adafruit.com/product/3708
[12] Arduino Uno REV3.
https://store.arduino.cc/arduino-uno-rev3
[13] Downloads - Raspberry Pi. Sistemes operatius disponibles.
https://www.raspberrypi.org/downloads/
[14] Backups Raspberry Pi SD.
https://www.raspberrypi.org/documentation/linux/filesystem/backup.md
[15] SD cards. Raspberry Pi.
https://www.raspberrypi.org/documentation/installation/sd-cards.md

[16] Retirada suport Python 2.7
https://pythonclock.org/
[17] Llibreria HPMA115S0 ThingType.
https://github.com/ThingType/HPMA115S0_Python_library
[18] DHT11 Raspberry Pi
http://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-the-raspberry-pi/
[19] ADS1115 per a Arduino
http://henrysbench.capnfatz.com/henrys-bench/arduino-voltage-measurements/arduino-ads1115-module-getting-started-tutorial/
[20] API 19 Android 4.4
https://developer.android.com/about/versions/android-4.4.html
[21] Prevent SD card corruption.
https://www.raspberrypi.org/forums/viewtopic.php?t=36533
[22] Problemes amb símbols i XML
https://stackoverflow.com/questions/1328538/how-do-i-escape-ampersands-in-xml-so-they-are-rendered-as-entities-in-html

14. Annexos

14.1. Codi

A continuació es poden consultar els diferents fitxers de codi utilitzats en el projecte:

```
import os
import subprocess
import json

configfile = '/home/pi/configuration.xml'

with open(configfile) as json_file:
    #Read XML configuration file
    config = json.load(json_file)

    #Configure Wi-Fi
    if config["wifi_ssid"]:
        ssid = config["wifi_ssid"]
        password = config["wifi_password"]
        command = "sudo /home/pi/RoMiBox/configureWifi.sh " + ssid + " " + password
        subprocess.call(command, shell=True)
        subprocess.call("service networking restart", shell=True)

    #Configure read and sending intervals
    readinginterval = config["reading_interval"]
    sendinginterval = config["sending_interval"]

    #Set Execution time for reading

    command = "while true; do sudo python3 /home/pi/RoMiBox/ReadSensors.py; sleep " + str(readinginterval)
    + "; done"
    process = subprocess.Popen(command, shell=True, stdout=subprocess.PIPE)

    #Set Execution time for sending

    command = "while true; do sudo python3 /home/pi/RoMiBox/SendDataAPI.py; sleep " + str(sendinginterval)
    + "; done"
    process = subprocess.Popen(command, shell=True, stdout=subprocess.PIPE)
```

```
#!/bin/sh

file="/etc/wpa_supplicant/wpa_supplicant.conf"

passwd=$(wpa_passphrase $1 $2 | sed -n 4p)

nssid=$(egrep -nrw $1 $file | cut -f1 -d:)
npasswd=$(egrep -nrw $passwd $file | head -1 | cut -f1 -d:)

if [ \( ! -z "$npasswd" \) ]
then
    npasswd=$(($npasswd-1))
    exists=$(echo $nssid | egrep "$npasswd")
fi

if [ \( -z "$nssid" \) -o \( -z "$exists" \) ]
then
    #Empty, so wifi does not exist on configuration file
    wpa_passphrase $1 $2 | sudo tee -a /etc/wpa_supplicant/wpa_supplicant.conf 2> /dev/null
fi

sudo sed -i '/^[[:blank:]]*#/d;s/#.*//' $file

#remove password from configuration.xml if exists

file="/home/pi/configuration.json"
existspass=$(egrep -nrw "wifi_password" $file | head -1 | cut -f1 -d:)
if [ \( ! -z "$existspass" \) ]
then
    sudo sed -i '2,3d' /home/pi/configuration.json
fi
```

```
import time
import serial
from serial import Serial

READ_PARTICLE_MEASUREMENT = 4
HPM_READ_PARTICLE_MEASUREMENT_LEN = 5
HPM_MAX_RESP_SIZE = 8
HPM_CMD_RESP_HEAD = 0x40


class HPMA115S0:
    _serial = None
    _pm2_5 = None
    _pm10 = None
    _dataBuf = [None] * (HPM_READ_PARTICLE_MEASUREMENT_LEN - 1)

    def __init__(self, ser):
        """
        Constructor for the HPMA115S0 class
        """
        self._serial = serial.Serial()
        self._serial.port = ser
        self._serial.baudrate = 9600
        self._serial.stopbits = serial.STOPBITS_ONE
        self._serial.bytesize = serial.EIGHTBITS
        self._serial.timeout = 1
        self._serial.open()

    def init(self):
        """
        Function which initializes the sensor.
        """
        time.sleep(0.1)
        self.startParticleMeasurement()
        time.sleep(0.1)
        self.disableAutoSend()

    def sendCmd(self, cmdBuf):
        """
        Function which sends a serial command to the sensor (cf datasheet)

        Params:
                cmdBuf: the array containing the datas to be sent
        """
        self._serial.write(bytearray(cmdBuf))

    def readCmdResp(self, cmdType):
        """
        Function which reads command response from the sensor

        Params:
                cmdType : Expected command type
        """
        respBuf = [None] * HPM_MAX_RESP_SIZE
        respIdx = 0
        calChecksum = 0
        for i in range(0, len(respBuf)):
            respBuf[i] = 0

        if (self.readStringUntil(HPM_CMD_RESP_HEAD)):
            respBuf[0] = HPM_CMD_RESP_HEAD
            respBuf[1] = ord(self._serial.read())

            if (respBuf[1] and ((respBuf[1] + 1) <= len(respBuf) - 2) and (respBuf[1] - 1) <=
            len(self._dataBuf)):
                resp = self.readBytes(respBuf, respBuf[1] + 1, 2)
                respBuf = resp[0]
                respCount = resp[1]
                if (respCount == respBuf[1] + 1):
                    if (respBuf[2] == cmdType):
                        while (respIdx < (2 + respBuf[1])):
                            calChecksum += respBuf[respIdx]
                            respIdx += 1
                        calChecksum = (65536 - calChecksum) % 256
                        if (calChecksum == respBuf[2 + respBuf[1]]):
                            for i in range(0, len(self._dataBuf)):
```

```
                        self._dataBuf[i] = 0
                    j = 0
                    for i in range(0, 4):
                        self._dataBuf[j] = respBuf[i + 3]
                        j += 1
                    return (respBuf[1] - 1)
        return False

    def startParticleMeasurement(self):
        """
        Function which starts sensor measurement
        """
        cmd = [0x68, 0x01, 0x01, 0x96]
        self.sendCmd(cmd)

    def stopParticleMeasurement(self):
        """
        Function which stops sensor measurement
        """
        cmd = [0x68, 0x01, 0x02, 0x95]
        self.sendCmd(cmd)

    def disableAutoSend(self):
        """
        Function which stops auto send by the sensor
        """
        cmd = [0x68, 0x01, 0x20, 0x77]
        self.sendCmd(cmd)

    def readParticleMeasurement(self):
        """
        Function which sends a read command to sensor to get retrieve datas
        """
        cmdBuf = [0x68, 0x01, 0x04, 0x93]

        self.sendCmd(cmdBuf)

        if (self.readCmdResp(READ_PARTICLE_MEASUREMENT) == (HPM_READ_PARTICLE_MEASUREMENT_LEN - 1)):
            self._pm2_5 = self._dataBuf[0] * 256 + self._dataBuf[1]
            self._pm10 = self._dataBuf[2] * 256 + self._dataBuf[3]

            return True

        return False

    def readStringUntil(self, terminator):
        """
        Function to start reading when the sensor is ready to transmit datas

        """
        c = self._serial.read()
        if (ord(c) == terminator):
            return True

    def readBytes(self, buffer, length, index):
        count = 0
        while (count < length):
            c = self._serial.read()
            for ch in c:
                ch = ord(c)
                if (ch < 0):
                    break
                buffer[index] = ch
                count += 1
                index += 1
        return [buffer, count]
```

```sql
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

CREATE TABLE `readings` (
  `id` int(11) NOT NULL,
  `romiboxid` varchar(6) NOT NULL,
  `date` varchar(8) NOT NULL,
  `time` varchar(6) NOT NULL,
  `longitude` double NOT NULL,
  `latitude` double NOT NULL,
  `PM10` double NOT NULL,
  `PM25` double NOT NULL,
  `temperature` double NOT NULL,
  `humidity`double NOT NULL,
  `CO` double NOT NULL,
  `CO2` double NOT NULL,
  `confirm` varchar(1) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;


ALTER TABLE `readings`
  ADD PRIMARY KEY (`id`);

ALTER TABLE `readings`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=114;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

```
import time
import board
import busio
import adafruit_ads1x15.ads1115 as ADS
from adafruit_ads1x15.analog_in import AnalogIn
import requests
import MySQLdb
import HPMA115S0
import os
import threading
import datetime
from xml.dom import minidom
import sys
import math
import operator
import Adafruit_DHT

sensor = Adafruit_DHT.DHT11


configfile = '/home/pi/configuration.json'

# The load resistance on the board
RLOAD = 10.0
# Calibration resistance at atmospheric CO2 level
RZERO = 76.63
# Parameters for calculating ppm of CO2 from sensor resistance
PARA = 116.6020682
PARB = 2.769034857
# Parameters to model temperature and humidity dependence
CORA = 0.00035
CORB = 0.02718
CORC = 1.39538
CORD = 0.0018
CORE = -0.003333333
CORF = -0.001923077
CORG = 1.130128205
# Atmospheric CO2 level for calibration purposes
ATMOCO2 = 397.13

coefficientAco = 19.32
coefficientBco = -0.64


# Create the I2C bus
i2c = busio.I2C(board.SCL, board.SDA)

# Create the ADC object using the I2C bus
ads = ADS.ADS1115(i2c)

# Create single-ended input on channel 0
chan = AnalogIn(ads, ADS.P0)
chan2 = AnalogIn(ads, ADS.P1)

def getCorrectionFactor(t,h,CORA,CORB,CORC,CORD,CORE,CORF,CORG):
    # Linearization of the temperature dependency curve under and above 20 degree C
    # below 20degC: fact = a * t * t - b * t - (h - 33) * d
    # above 20degC: fact = a * t + b * h + c
    # this assumes a linear dependency on humidity
    if t < 20:
        return CORA * t * t - CORB * t + CORC - (h-33.)*CORD
    else:
        return CORE * t + CORF * h + CORG

def getResistance(value_pin,RLOAD):
    return ((1023./value_pin) - 1.)*RLOAD

def getCorrectedResistance(t,h,CORA,CORB,CORC,CORD,CORE,CORF,CORG,value_pin,RLOAD):
    return getResistance(value_pin,RLOAD) / getCorrectionFactor(t,h,CORA,CORB,CORC,CORD,CORE,CORF,CORG)

def getPPM(PARA,RZERO,PARB,value_pin,RLOAD):
    return PARA * math.pow((getResistance(value_pin,RLOAD)/RZERO), -PARB)

def getCorrectedPPM(t,h,CORA,CORB,CORC,CORD,CORE,CORF,CORG,value_pin,RLOAD,PARA,RZERO,PARB):
    return PARA *
    math.pow((getCorrectedResistance(t,h,CORA,CORB,CORC,CORD,CORE,CORF,CORG,value_pin,RLOAD)/RZERO), -
```

```
    return PARA *
    math.pow((getCorrectedResistance(t,h,CORA,CORB,CORC,CORD,CORE,CORF,CORG,value_pin,RLOAD)/RZERO), -
    PARB)

def getRZero(value_pin,RLOAD,ATMOCO2,PARA,PARB):
    return getResistance(value_pin,RLOAD) * math.pow((ATMOCO2/PARA), (1./PARB))

def getCorrectedRZero(t,h,CORA,CORB,CORC,CORD,CORE,CORF,CORG,value_pin,RLOAD,ATMOCO2,PARA,PARB):
    return getCorrectedResistance(t,h,CORA,CORB,CORC,CORD,CORE,CORF,CORG,value_pin,RLOAD) *
    math.pow((ATMOCO2/PARA), (1./PARB))

def map(x,in_min,in_max,out_min,out_max):
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min

def main():
    #start the port ttyS0 (serial) to read the sensor
    hpma115S0 = HPMA115S0.HPMA115S0("/dev/ttyS0")

    hpma115S0.init()
    hpma115S0.startParticleMeasurement()

    #connect to database
    class Database:
        host = 'localhost'
        user = 'root'
        password = ''
        db = 'readings'

        def __init__(self):
            self.connection = MySQLdb.connect(self.host, self.user, self.password, self.db)
            self.cursor = self.connection.cursor()

        def insert(self, query):
            try:
                self.cursor.execute(query)
                self.connection.commit()
            except:
                self.connection.rollback()

        def query(self, query):
            cursor = self.connection.cursor( MySQLdb.cursors.DictCursor )
            cursor.execute(query)
            return cursor.fetchall()

        def __del__(self):
            self.connection.close()

    db = Database()

    #Read configuration file
    open(configfile) as json_file
    #Read configuration file
    config = json.load(json_file)
    romiboxid =  config["id"]
    longitude = config["longitude"]
    latitude = config["latitude"]
    romiboxtype = config["romibox_type"

    while True: #while sensor has not started
        if (hpma115S0.readParticleMeasurement()): #if able to read the sensor
            PM25 = hpma115S0._pm2_5
            PM10 = hpma115S0._pm10
            readingdate = time.strftime('%Y%m%d')
            readingtime = time.strftime('%H%M%S')
            if romiboxtype == "STATIC_PRO":
                chan = AnalogIn(ads, ADS.P0)
                value_ads = chan.value # value obtained by ADS1115
                value_pin = map((value_ads-150), 0, 1023, 0, 1023)
                rzero = getRZero(value_pin,RLOAD,ATMOCO2,PARA,PARB)

                h, t = Adafruit_DHT.read_retry(sensor, 21)

                correctedRZero =
                getCorrectedRZero(t,h,CORA,CORB,CORC,CORD,CORE,CORF,CORG,value_pin,RLOAD,ATMOCO2,PARA,PARB
                )
                resistance = getResistance(value_pin,RLOAD)
```

```
                ppm = getPPM(PARA,RZERO,PARB,value_pin,RLOAD)
                correctedPPMco2 =
                getCorrectedPPM(t,h,CORA,CORB,CORC,CORD,CORE,CORF,CORG,value_pin,RLOAD,PARA,RZERO,PARB)

                data = chan2.value # value obtained by ADS1115
                v_out = chan2.voltage

                correctedPPMco = coefficientAco*pow((5-v_out)/v_out, coefficientBco);
            else:
                t = 0
                h = 0
                correctedPPMco = 0
                correctedPPMco2 = 0
            # Data Insert into the table
            query = "INSERT INTO readings (romiboxid, date, time, longitude, latitude, PM25, PM10,
            temperature, humidity, CO, CO2, confirm) VALUES (\'" + romiboxid + "\', \'" + readingdate +
            "\', \'" + readingtime + "\', \'" + longitude + "\', \'" + latitude + "\', \'" + str(PM25) +
            "\', \'" + str(PM10) + "\', \'" + str(t) + "\', \'" +  str(h) + " \', \'" +
            str(correctedPPMco) +"\', \'" + str(correctedPPMco2) +"\', 1);"
            db.insert(query)
            break #exit loop if able to read the value

if __name__ == "__main__":
    main()
```

```
# importing the requests library
import requests
import MySQLdb
import json
from time import *
import time

class Database:

    host = 'localhost'
    user = 'root'
    password = ''
    db = 'readings'

    def __init__(self):
        self.connection = MySQLdb.connect(self.host, self.user, self.password, self.db)
        self.cursor = self.connection.cursor()

    def insert(self, query):
        try:
            self.cursor.execute(query)
            self.connection.commit()
        except:
            self.connection.rollback()

    def query(self, query):
        cursor = self.connection.cursor( MySQLdb.cursors.DictCursor )
        cursor.execute(query)
        return cursor.fetchall()

    def __del__(self):
        self.connection.close()


if __name__ == "__main__":

    db = Database()

    # Data retrieved from the table
    comanda = "UPDATE readings SET confirm = 2 WHERE confirm=1 LIMIT 200;"
    db.insert(comanda)
    select_query = """SELECT * FROM readings WHERE confirm = 2 ORDER BY date DESC, time DESC"""
    punts = db.query(select_query)
    puntArray = ""
    nPunts = 0
    id_bus = ""
    romiboxid = ""
    for punt in punts:
        id = str(punt['id'])
        romiboxid = str(punt['romiboxid']) # romibox id
        date = str(punt['date']) # date AAAAMMDD
        date = date[:4] + '-' + date[4:]
        date = date[:7] + '-' + date[7:]
        ptime = str(punt['time']) # time HHMMSS
        ptime = ptime[:2] + ":" + ptime[2:]
        ptime = ptime[:5] + ":" + ptime[5:]
        longitude = str(punt['longitude']) # longitude GPS
        latitude = str(punt['latitude']) # latitude GPS
        pm10 = str(punt['PM10']) # optic sensor pm10
        pm25 = str(punt['PM25']) # optic sensor pm2.5
        t = str(punt['temperature']) # optic sensor pm2.5
        h = str(punt['humidity']) # optic sensor pm2.5
        co = str(punt['CO']) # optic sensor pm2.5
        co2 = str(punt['CO2']) # optic sensor pm2.5

        # JSON format
        jsonpm10 = "{\"type_id\": \"PM10\",\"datetime\": \"" + date + " " + ptime + "\",\"value\": " +
        pm10 + "}"
        jsonpm25 = "{\"type_id\": \"PM25\",\"datetime\": \"" + date + " " + ptime + "\",\"value\": " +
        pm25 + "}"
        jsont = "{\"type_id\": \"T\",\"datetime\": \"" + date + " " + ptime + "\",\"value\": " + t + "}"
        jsonh = "{\"type_id\": \"H\",\"datetime\": \"" + date + " " + ptime + "\",\"value\": " + h + "}"
        jsonco = "{\"type_id\": \"CO\",\"datetime\": \"" + date + " " + ptime + "\",\"value\": " + co +
        "}"
        jsonco2 = "{\"type_id\": \"CO2\",\"datetime\": \"" + date + " " + ptime + "\",\"value\": " + co2 +
        "}"
```

```
        if nPunts == 0:
            puntArray = jsonpm10 + "," + jsonpm25 + "," + jsont + "," + jsonh + "," + jsonco + "," +
            jsonco2
        else:
            puntArray = puntArray + "," + jsonpm10 + "," + jsonpm25 + "," + jsont + "," + jsonh + "," +
            jsonco + "," + jsonco2
    nPunts = nPunts + 1
    data = "{\"station_id\": \"" + romiboxid + "\", \"measures\": [" + puntArray + "]}"
    URL = "https://x5bmgaql9d.execute-api.eu-west-1.amazonaws.com/prod/v1/stations/" + romiboxid + "/
    measures"
    # send post request and remove confirmed data
    r = requests.post(url = URL, data = data, headers={"accept":"application/json", "content-
    type":"application/json", "x-api-key":"aSSx5EJlIN8dNyirhRB8X1x3nFzHJ0pYaPY2wO6C"})
    if r.status_code == 200:
        comanda = "DELETE FROM readings WHERE confirm=2;"
        db.insert(comanda)
    else:
        comanda = "UPDATE readings SET confirm = 1 WHERE confirm=2;"
        db.insert(comanda)
```

14.3. Datasheets

A continuació s'inclouen les datasheet dels sensors utilitzats en aquest projecte:

Honeywell

THE POWER OF **CONNECTED**

HPM Series
Particle Sensor

32322550
Issue E

DESCRIPTION

The Honeywell HPM Series Particle Sensor is a laser-based sensor which uses the light scattering method to detect and count particles in the concentration range of 0 µg/m^3 to 1,000 µg/m^3 in a given environment. A laser light source illuminates a particle as it is pulled through the detection chamber. As particles pass through the laser beam, the light source becomes obscured and is recorded on the photo or light detector. The light is then analyzed and converted to an electrical signal to calculate concentrations in real time. The Honeywell particle sensor provides information on the particle concentration for given particle concentration range.

VALUE TO CUSTOMERS

- Enables the ability to more accurately and cost-effectively monitor or control environmental particulate
- Industry-leading long life of 20,000 hours of continuous use essentially equates to seven years of product life (based on up to eight hours of operation per day)
- Proven EMC performance enables the ability to perform more accurately in a variety of tough industrial environments
- Ultra-fast response time of <6 s allows the HPM Series to respond to environmental conditions in real time
- Enhanced reliability allows for use in harsh environments

DIFFERENTIATION

- Industry-leading long life of 20,000 hours provides stable operation and continuous use
- Proven EMC performance, based on IEC61000 stable operation, ±15% accuracy

FEATURES

- Laser-based light scattering particle sensing
- Concentration range: 0 µg/m^3 to 1,000 µg/m^3
- Fully calibrated
- EMC: Heavy industrial level IEC61000
- Response time: <6 s
- Supply current: 80 mA max.
- Output signal: UART (Universal Asynchronous Receiver/ Transmitter)
- PM2.5 output (PM10 output optional)
- RoHS compliant
- REACH compliant

POTENTIAL INDUSTRIAL APPLICATIONS

- HVAC:
  - Air conditioners
  - Air quality monitors
  - Environmental monitoring
- Consumer products:
  - Air cleaners
  - Air conditioners
  - Air purifiers
  - Car air cleaners
  - Handheld air quality detectors

Particle Sensor
HPM Series

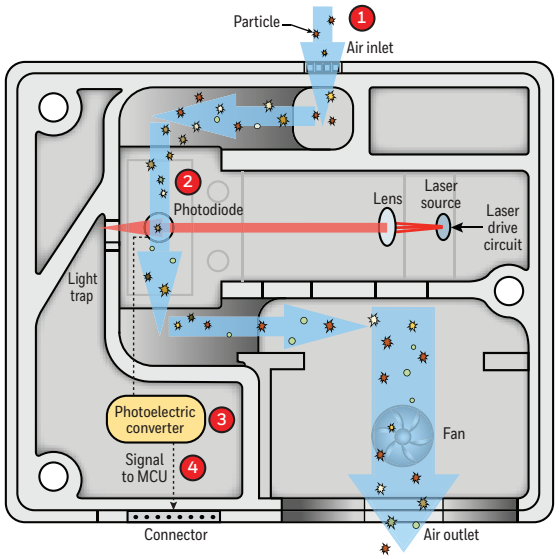
Table 1. Specifications

Characteristic	Parameter
Operating principle	laser scattering
Detection[1,2]	PM2.5 and PM10
Output data[1, 2]	PM2.5 in µg/m³ and PM10 in µg/m³
Concentration range	0 µg/m³ to 1,000 µg/m³
Accuracy (at 25°C ±5°C): 0 µg/m³ to 100 µg/m³ 100 µg/m³ to 1000 µg/m³	 ±15 µg/m³ ±15 %
Response time	<6 s
Supply voltage	5 V ±0.2 V
Standby current (at 25°C ±5°C)	<20 mA
Supply current (at 25°C ±5°C)	<80 mA
Temperature: operating storage	 -10°C to 50°C [-14°F to 122°F] -30°C to 65°C [-22°F to 149°F]
Humidity (operating and storage)	0 %RH to 95 %RH non-condensing
Output protocol[3]	UART; baud rate: 9600, databits: 8, stopbits: 1, parity: no
Operating time: continuous mode intermittent mode	 20,000 hr depends on duty cycle
Laser class	Laser Class 1: IEC/EN 60825-1: 650 nm
ESD	±4 kV contact, ±8 kV air per IEC 61000-4-2
Radiated immunity	1 V/m (80 MHz to 1000 MHz) per IEC 61000-4-3
Fast transient burst	±0.5 kV per IEC61000-4-4
Immunity to conducted disturbances radiated emissions	3 V per IEC61000-4-6
Radiated emissions	40 dB 30 MHz to 230 MHz; 47 dB 230 MHz to 1000 MHz per CISPR 14
Conducted emissions	0.15 MHz to 30 MHz in compliance with CISPR 14

[1] PM2.5 is particulate matter ≤2.5 µm in diameter; PM10 is particulate matter ≤10 µm in diameter.
[2] PM10 in µg/m³ is calculated from PM 2.5 readings.
[3] Contact Honeywell for other output options.

CLASS 1 LASER PRODUCT

Figure 1. HPM Series Operation (Top Down View)



1. Fan draws in air through inlet.
2. Air passes through the laser where the light reflected off the particles is captured by the photodiode.
3. The photodiode passes information to the photoelectric converter
   The photoelectric converter processes the signal from the particles into density.
4. Signal is transmitted to micro control unit where a proprietary algorithm processes the data and supplies outputs for the density of the particulate (µg/m³).

Figure 2. Mounting Dimensions (mm/[in] For reference only.)

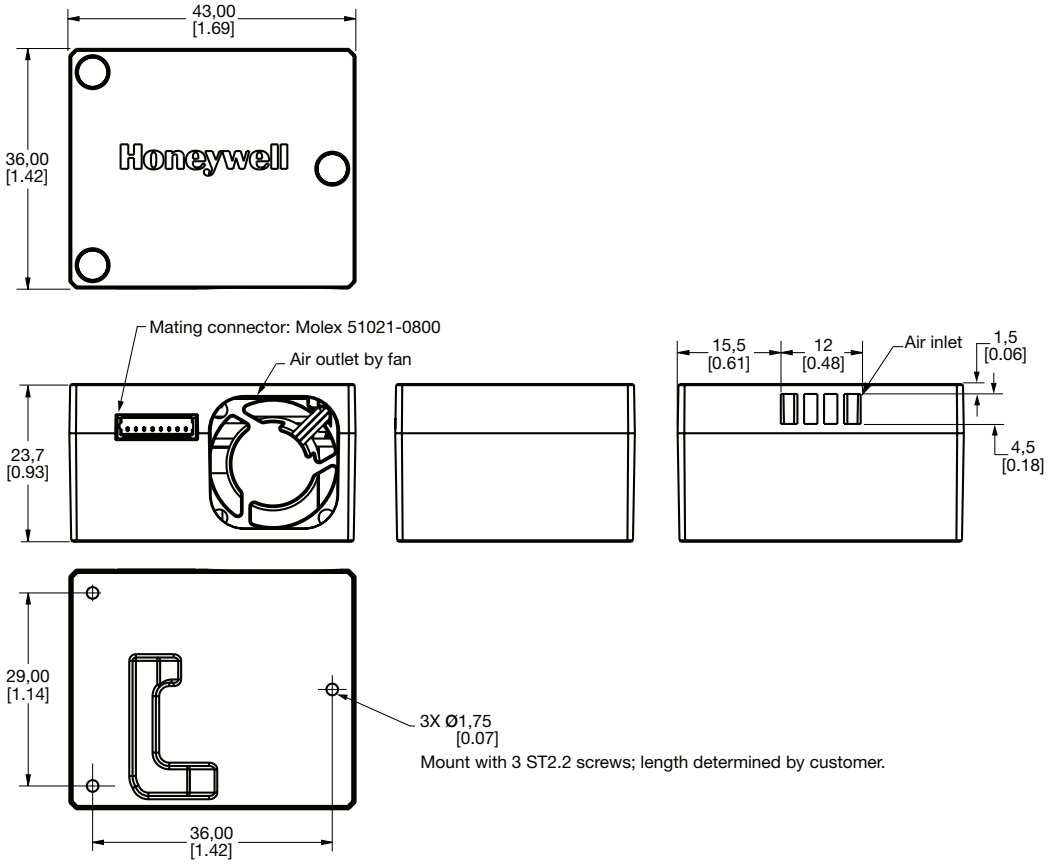
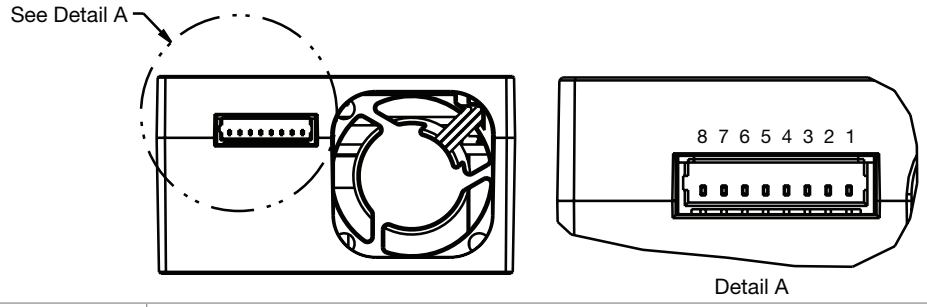


Table 2. Connector Pinout



Pin	Name	Description
1	+3.3 V	power output (+3.3 V/100 mA)
2	5 V	power input (5 V)
3	N/A	N/A
4	N/A	N/A
5	TEST	used for testing (NA)
6	TX	UART TX output (0 - 3.3 V)
7	RX	UART RX input (0 - 3.3 V)
8	GND	power input (ground terminal)

Product Installation

NOTICE

IMPROPER INSTALLATION

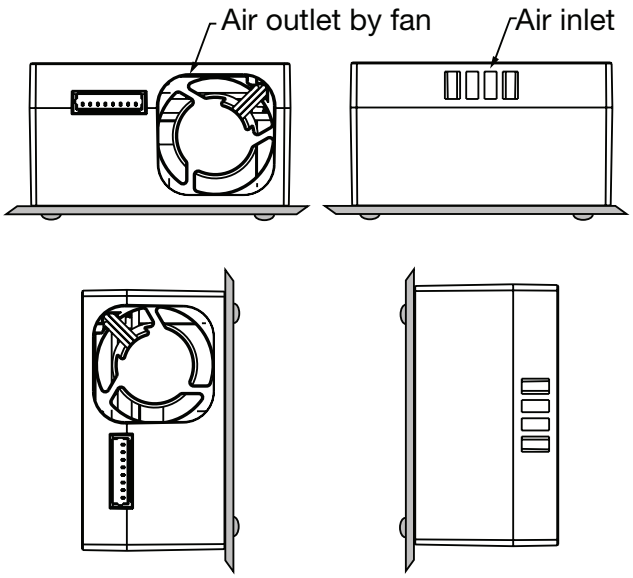
To avoid particulate settling or accumulation at the air outlet or air inlet, which may affect product sensitivity and accuracy, ensure that the HPM Series Particle Sensor:

- Is installed correctly according to Figure 2.
- Is installed such that the air inlet and air outlets are not blocked and that the flow of air through the sensor is neither restricted nor reduced.

Install the product to the desired surface using the screw size shown in Figure 1.

Figure 3. Installation Orientation

Correct



Incorrect

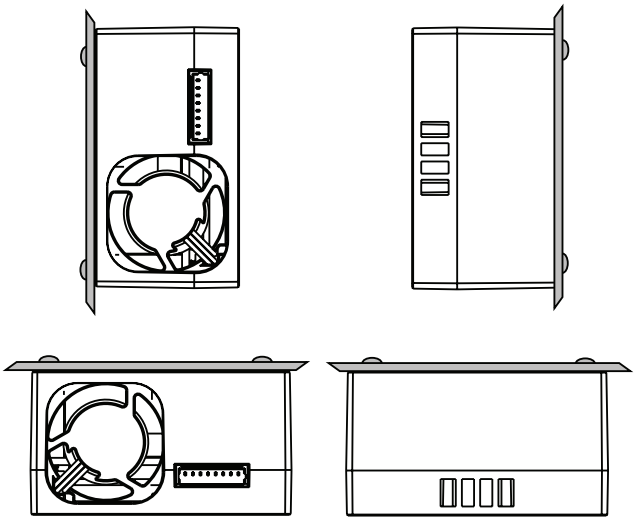


Table 3. Order Guide

Catalog Listing	Description
HPMA115S0-XXX	HPM Series PM2.5 Particle Sensor with UART output

Table 4. Customer Use Protocol[1]

Command Length (Bytes)	HEAD	LEN	CMD	Data	CS	Example
Read Particle Measuring Results						
Send	0x68	0x01	0x04	NA	CS = MOD ((65536-(HEAD+LEN+CMD+DATA)), 256)	68 01 04 93
Response, Pos ACK	0x40	0x05	0X04	"DF1, DF2, DF3, DF4 PM2.5 = DF1 * 256 + DF2 PM10 = DF3 * 256 + DF4"	CS = MOD ((65536-(HEAD+LEN+CMD+DATA)), 256)	40 05 04 00 30 00 31 56
Response, Neg ACK	0X9696					
Start Particle Measurement						
Send	0x68	0x01	0x01	NA	CS = MOD ((65536-(HEAD+LEN+CMD+DATA)), 256)	68 01 01 96
Response, Pos ACK	0xA5A5					
Response, Neg ACK	0x9696					
Stop Particle Measurement[2]						
Send	0x68	0x01	0x02	NA	CS = MOD ((65536-(HEAD+LEN+CMD+DATA)), 256)	68 01 02 95
Response, Pos ACK	0xA5A5					
Response, Neg ACK	0x9696					
Set Customer Adjustment Coefficient						
Send	0x68	0x02	0x08	DF1: 30 ~ 200 (Default, 100)	CS = MOD ((65536-(HEAD+LEN+CMD+DATA)), 256)	68 02 08 64 2A
Response, Pos ACK	0xA5A5					
Response, Neg ACK	0x9696					
Read Customer Adjustment Coefficient						
Send	0x68	0x01	0x10	NA	CS = MOD ((65536-(HEAD+LEN+CMD+DATA)), 256)	68 01 10 87
Response, Pos ACK	0X40	0X02	0X10	DF1: 30 ~ 200 (Default, 100)	CS = MOD ((65536-(HEAD+LEN+CMD+DATA)), 256)	40 02 10 64 4A
Response, Neg ACK	0x9696					
Stop Auto Send						
Send	0x68	0x01	0x20	NA	CS = MOD ((65536-(HEAD+LEN+CMD+DATA)), 256)	68 01 20 77
Response, Pos ACK	0xA5A5					
Response, Neg ACK	0x9696					
Enable Auto Send[3]						
Send	0x68	0x01	0x40	NA	CS = MOD ((65536-(HEAD+LEN+CMD+DATA)), 256)	68 01 40 57
Response, Pos ACK	0xA5A5					
Response, Neg ACK	0x9696					

[1]Product l life may vary depending on the specific application in which the sensor is utilized.

[2]Shuts down the fan, helping to extend the life of the product.

[3]See Table 5 for data format.

Table 5. Data Format (Protocol Length: 32 Bytes)

Byte Number	Head0	Head0	Head0
Byte0	Head0	0x42	fixed
Byte1	Head1	0x4d	
Byte2	Len_H	...	Frame Length = 2x13+2(data length + checksum length)
Byte3	Len_L	...	
Byte4	Data0_H	...	reserve
Byte5	Data0_L	...	
Byte6	Data1_H	...	PM2.5 concentration (standard particulate matter)
Byte7	Data1_L	...	
Byte8	Data2_H	...	PM10 concentration (standard particulate matter)
Byte9	Data2_L	...	
Byte10	Data3_H	...	reserve
Byte11	Data3_L	...	
Byte12	Data4_H	...	reserve
Byte13	Data4_L	...	
Byte14	Data5_H	...	reserve
Byte15	Data5_L	...	
Byte16	Data6_H	...	reserve
Byte17	Data6_L	...	
Byte18	Data7_H	...	reserve
Byte19	Data7_L	...	
Byte20	Data8_H	...	reserve
Byte21	Data8_L	...	
Byte22	Data9_H	...	reserve
Byte23	Data9_L	...	
Byte24	Data10_H	...	reserve
Byte25	Data10_L	...	
Byte26	Data11_H	...	reserve
Byte27	Data11_L	...	
Byte28	Data12_H	...	reserve
Byte29	Data12_L	...	
Byte30	CheckSum_H	...	Checksum = Head0+Head1+Len_H+Len_L+Data0_H+...+Data12_L
Byte31	CheckSum_H	...	

ADDITIONAL INFORMATION

The following associated literature is available on the Honeywell web site at sensing.honeywell.com:

- Sell sheet
- Frequently Asked Questions (FAQs)

For more information

Honeywell Sensing and Internet of Things services its customers through a worldwide network of sales offices and distributors. For application assistance, current specifications, pricing or the nearest Authorized Distributor, visit sensing.honeywell.com or call:

Asia Pacific	+65 6355-2828
Europe	+44 (0) 1698 481481
USA/Canada	+1-800-537-6945

Honeywell Sensing and Internet of Things
9680 Old Bailes Road
Fort Mill, SC 29707
www. honeywell.com

⚠ WARNING
PERSONAL INJURY

DO NOT USE these products as safety or emergency stop devices or in any other application where failure of the product could result in personal injury.

Failure to comply with these instructions could result in death or serious injury.

⚠ WARNING
MISUSE OF DOCUMENTATION

- The information presented in this datasheet is for reference only. Do not use this document as a product installation guide.
- Complete installation, operation, and maintenance information is provided in the instructions supplied with each product.

Failure to comply with these instructions could result in death or serious injury.

Warranty/Remedy

Honeywell warrants goods of its manufacture as being free of defective materials and faulty workmanship during the applicable warranty period. Honeywell's standard product warranty applies unless agreed to otherwise by Honeywell in writing; please refer to your order acknowledgment or consult your local sales office for specific warranty details. If warranted goods are returned to Honeywell during the period of coverage, Honeywell will repair or replace, at its option, without charge those items that Honeywell, in its sole discretion, finds defective. **The foregoing is buyer's sole remedy and is in lieu of all other warranties, expressed or implied, including those of merchantability and fitness for a particular purpose. In no event shall Honeywell be liable for consequential, special, or indirect damages.**

While Honeywell may provide application assistance personally, through our literature and the Honeywell web site, it is buyer's sole responsibility to determine the suitability of the product in the application.

Specifications may change without notice. The information we supply is believed to be accurate and reliable as of this writing. However, Honeywell assumes no responsibility for its use.

Honeywell

32322552-E-EN | E | 05/18

TECHNICAL DATA MQ-135 GAS SENSOR

FEATURES

Wide detecting scope — Fast response and High sensitivity
Stable and long life — Simple drive circuit

APPLICATION

They are used in air quality control equipments for buildings/offices, are suitable for detecting of NH3,NOx, alcohol, Benzene, smoke,CO_2,etc.

SPECIFICATIONS

A. Standard work condition

Symbol	Parameter name	Technical condition	Remarks
Vc	Circuit voltage	5V±0.1	AC OR DC
V_H	Heating voltage	5V±0.1	ACOR DC
R_L	Load resistance	can adjust	
R_H	Heater resistance	33Ω ±5%	Room Tem
P_H	Heating consumption	less than 800mw	

B. Environment condition

Symbol	Parameter name	Technical condition	Remarks
Tao	Using Tem	-10℃-45℃	
Tas	Storage Tem	-20℃-70℃	
R_H	Related humidity	less than 95%Rh	
O_2	Oxygen concentration	21%(standard condition)Oxygen concentration can affect sensitivity	minimum value is over 2%

C. Sensitivity characteristic

Symbol	Parameter name	Technical parameter	Ramark 2
Rs	Sensing Resistance	30KΩ-200KΩ (100ppm NH_3)	Detecting concentration scope: 10ppm-300ppm NH_3 10ppm-1000ppm Benzene 10ppm-300ppm Alcohol
α (200/50) NH_3	Concentration Slope rate	≤0.65	
Standard Detecting Condition	Temp: 20℃±2℃ Humidity: 65%±5%	Vc:5V±0.1 Vh: 5V±0.1	
Preheat time	Over 24 hour		

D. Structure and configuration, basic measuring circuit

	Parts	Materials
1	Gas sensing layer	SnO_2
2	Electrode	Au
3	Electrode line	Pt
4	Heater coil	Ni-Cr alloy
5	Tubular ceramic	Al_2O_3
6	Anti-explosion network	Stainless steel gauze (SUS316 100-mesh)
7	Clamp ring	Copper plating Ni
8	Resin base	Bakelite
9	Tube Pin	Copper plating Ni

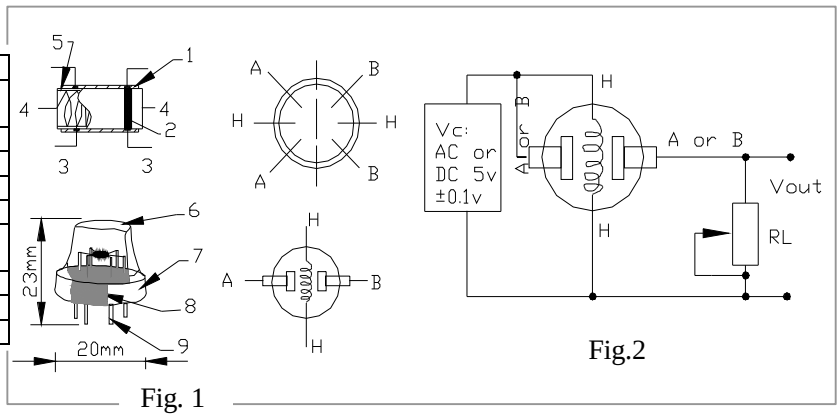
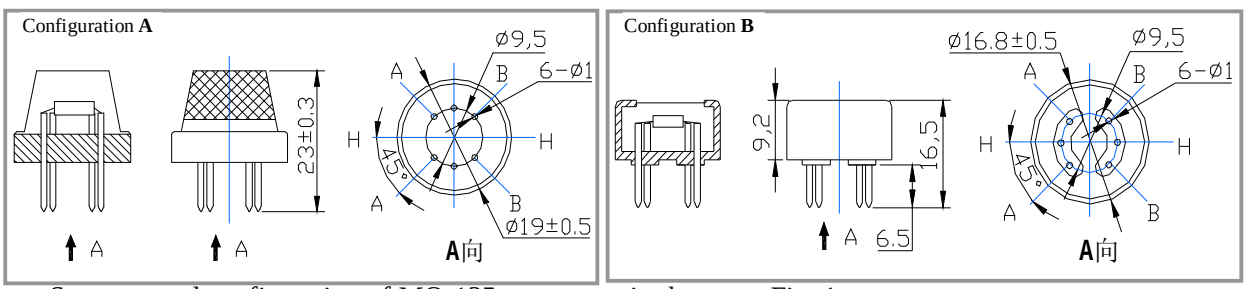


Fig. 1 Fig.2



Structure and configuration of MQ-135 gas sensor is shown as Fig. 1 (Configuration **A or B),** sensor composed by micro AL_2O_3 ceramic tube, Tin Dioxide (SnO_2) sensitive layer, measuring electrode and heater are fixed into a crust made by plastic and stainless steel net. The heater provides necessary work conditions for work of

sensitive components. The enveloped MQ-135 have 6 pin ,4 of them are used to fetch signals, and other 2 are used for providing heating current.

Electric parameter measurement circuit is shown as Fig.2

E. Sensitivity characteristic curve

Fig.2 sensitivity characteristics of the MQ-135

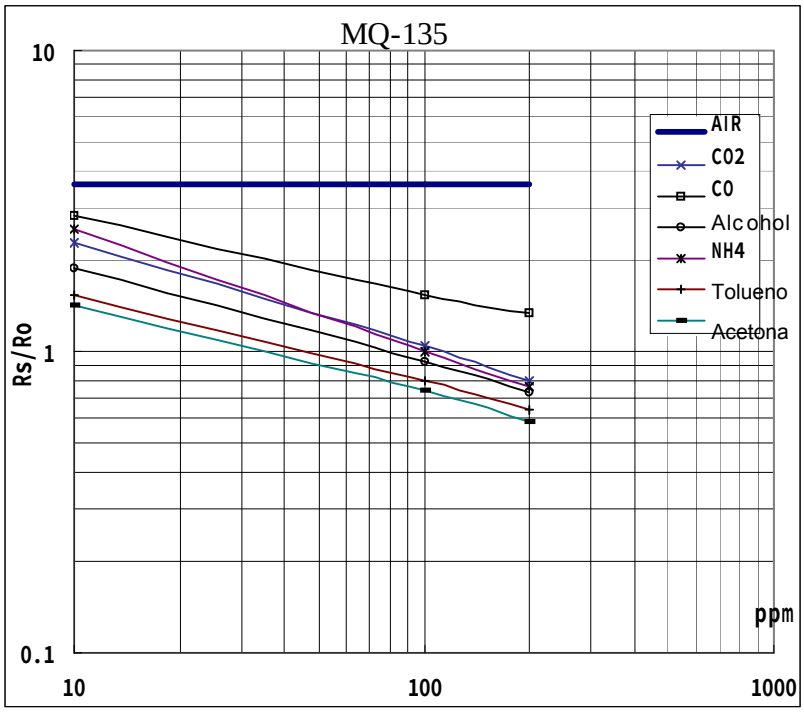


Fig.3 is shows the typical sensitivity characteristics of the MQ-135 for several gases.
in their: Temp: 20℃、
Humidity: 65%、
O_2 concentration 21%
RL=20kΩ
Ro: sensor resistance at 100ppm of NH_3 in the clean air.
Rs:sensor resistance at various concentrations of gases.

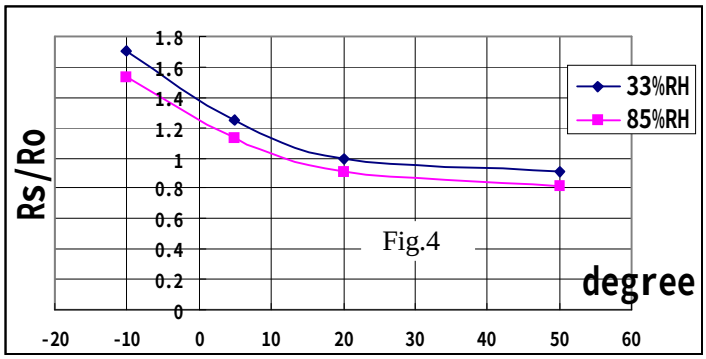


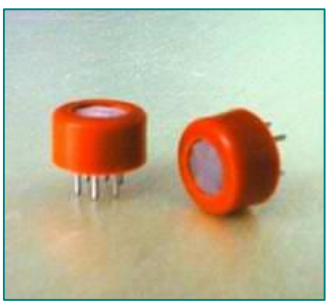
Fig.4

Fig.4 is shows the typical dependence of the MQ-135 on temperature and humidity.
Ro: sensor resistance at 100ppm of NH_3 in air at 33%RH and 20 degree.
Rs: sensor resistance at 100ppm of NH_3 at different temperatures and humidities.

SENSITIVITY ADJUSTMENT

Resistance value of MQ-135 is difference to various kinds and various concentration gases. So,When using this components, sensitivity adjustment is very necessary. we recommend that you calibrate the detector for 100ppm NH_3 or 50ppm Alcohol concentration in air and use value of Load resistancethat(R_L) about 20 KΩ (10KΩ to 47 KΩ).

When accurately measuring, the proper alarm point for the gas detector should be determined after considering the temperature and humidity influence.



Notification

1 Following conditions must be prohibited

1.1 Exposed to organic silicon steam

Organic silicon steam cause sensors invalid, sensors must be avoid exposing to silicon bond, fixature, silicon latex, putty or plastic contain silicon environment

1.2 High Corrosive gas

If the sensors exposed to high concentration corrosive gas (such as H_2Sz, SO_X, Cl_2, HCl etc), it will not only result in corrosion of sensors structure, also it cause sincere sensitivity attenuation.

1.3 Alkali, Alkali metals salt, halogen pollution

The sensors performance will be changed badly if sensors be sprayed polluted by alkali metals salt especially brine, or be exposed to halogen such as fluorin.

1.4 Touch water

Sensitivity of the sensors will be reduced when spattered or dipped in water.

1.5 Freezing

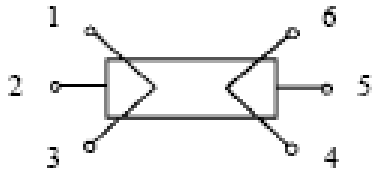
Do avoid icing on sensor'surface, otherwise sensor would lose sensitivity.

1.6 Applied voltage higher

Applied voltage on sensor should not be higher than stipulated value, otherwise it cause down-line or heater damaged, and bring on sensors' sensitivity characteristic changed badly.

1.7 Voltage on wrong pins

For 6 pins sensor, if apply voltage on 1、3 pins or 4、6 pins, it will make lead broken, and without signal when apply on 2、4 pins



2 Following conditions must be avoided

2.1 Water Condensation

Indoor conditions, slight water condensation will effect sensors performance lightly. However, if water condensation on sensors surface and keep a certain period, sensor' sensitivity will be decreased.

2.2 Used in high gas concentration

No matter the sensor is electrified or not, if long time placed in high gas concentration, if will affect sensors characteristic.

2.3 Long time storage

The sensors resistance produce reversible drift if it's stored for long time without electrify, this drift is related with storage conditions. Sensors should be stored in airproof without silicon gel bag with clean air. For the sensors with long time storage but no electrify, they need long aging time for stbility before using.

2.4 Long time exposed to adverse environment

No matter the sensors electrified or not, if exposed to adverse environment for long time, such as high humidity, high temperature, or high pollution etc, it will effect the sensors performance badly.

2.5 Vibration

Continual vibration will result in sensors down-lead response then repture. In transportation or assembling line, pneumatic screwdriver/ultrasonic welding machine can lead this vibration.

2.6 Concussion

If sensors meet strong concussion, it may lead its lead wire disconnected.

2.7 Usage

For sensor, handmade welding is optimal way. If use wave crest welding should meet the following conditions:

2.7.1 Soldering flux: Rosin soldering flux contains least chlorine

2.7.2 Speed: 1-2 Meter/ Minute

2.7.3 Warm-up temperature：100±20℃

2.7.4 Welding temperature：250±10℃

2.7.5 1 time pass wave crest welding machine

If disobey the above using terms, sensors sensitivity will be reduced.

TECHNICAL DATA MQ-7 GAS SENSOR

FEATURES

* High sensitivity to carbon monoxide
* Stable and long life

APPLICATION

They are used in gas detecting equipment for carbon monoxide(CO) in family and industry or car.

SPECIFICATIONS

A. Standard work condition

Symbol	Parameter name	Technical condition	Remark
Vc	circuit voltage	5V±0.1	Ac or Dc
V_H (H)	Heating voltage (high)	5V±0.1	Ac or Dc
V_H (L)	Heating voltage (low)	1.4V±0.1	Ac or Dc
R_L	Load resistance	Can adjust	
R_H	Heating resistance	33Ω ±5%	Room temperature
$T_{H(H)}$	Heating time (high)	60±1 seconds	
$T_{H(L)}$	Heating time (low)	90±1 seconds	
PH	Heating consumption	About 350mW	

b. Environment conditions

Symbol	Parameters	Technical conditions	Remark
Tao	Using temperature	-20℃-50℃	
Tas	Storage temperature	-20℃-50℃	Advice using scope
RH	Relative humidity	Less than 95%RH	
O_2	Oxygen concentration	21%(stand condition) the oxygen concentration can affect the sensitivity characteristic	Minimum value is over 2%

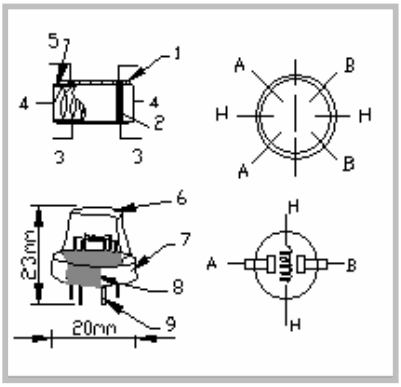
c. Sensitivity characteristic

symbol	Parameters	Technical parameters	Remark
Rs	Surface resistance Of sensitive body	2-20k	In 100ppm Carbon Monoxide
a (300/100ppm)	Concentration slope rate	Less than 0.5	Rs (300ppm)/Rs(100ppm)
Standard working condition	Temperature -20℃±2℃ relative humidity 65%±5% RL:10KΩ ±5%		
	Vc:5V±0.1V VH:5V±0.1V VH:1.4V±0.1V		
Preheat time	No less than 48 hours	Detecting range: 20ppm-2000ppm carbon monoxide	

D. Structure and configuration, basic measuring circuit

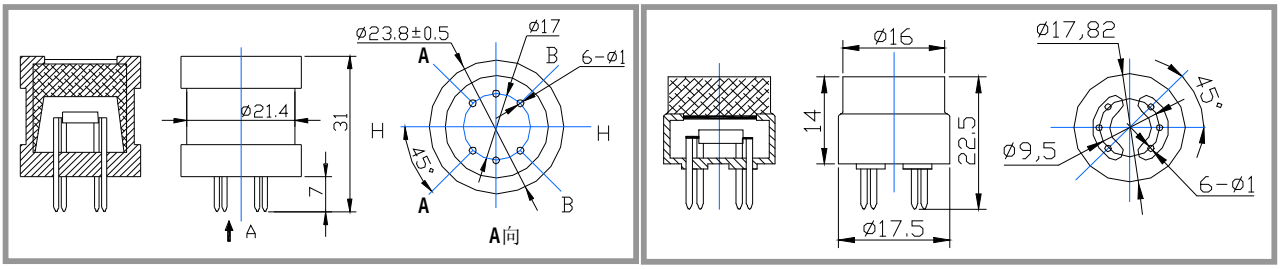
Structure and configuration of MQ-7 gas sensor is shown as Fig. 1 (Configuration A or B), sensor composed by micro AL_2O_3 ceramic tube, Tin Dioxide (SnO_2) sensitive layer, measuring electrode and heater are fixed into a crust made by plastic and stainless steel net. The heater provides necessary work conditions for work of sensitive components. The enveloped MQ-7 have

6 pin ,4 of them are used to fetch signals, and other 2 are used for providing heating current.



	Parts	Materials
1	Gas sensing layer	SnO_2
2	Electrode	Au
3	Electrode line	Pt
4	Heater coil	Ni-Cr alloy
5	Tubular ceramic	Al_2O_3
6	Anti-explosion network	Stainless steel gauze (SUS316 100-mesh)
7	Clamp ring	Copper plating Ni
8	Resin base	Bakelite
9	Tube Pin	Copper plating Ni

Fig.1



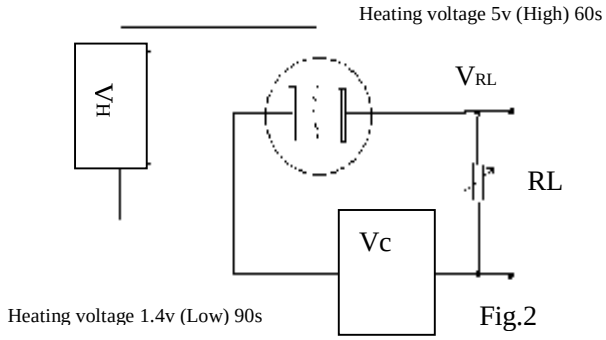


Fig.2

Standard circuit:

As shown in Fig 2, standard measuring circuit of MQ-7 sensitive components consists of 2 parts. one is heating circuit having time control function (the high voltage and the low voltage work circularly). The second is the signal output circuit, it can accurately respond changes of surface resistance of the sensor.

Electric parameter measurement circuit is shown as Fig.2

E. Sensitivity characteristic curve

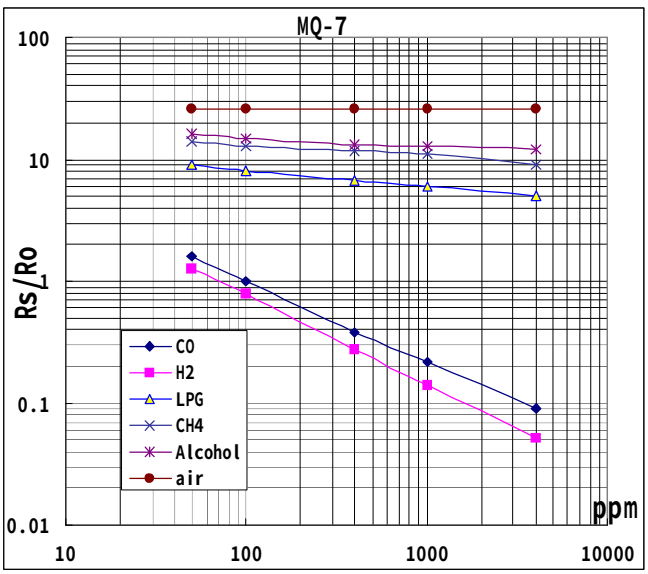


Fig.3 is shows the typical sensitivity characteristics of the MQ-7 for several gases.

in their: Temp: 20℃、

Humidity: 65%、

O_2 concentration 21%

RL=10kΩ

Ro: sensor resistance at 100ppm CO in the clean air.

Rs: sensor resistance at various concentrations of gases.

Fig.3 sensitivity characteristics of the MQ-7

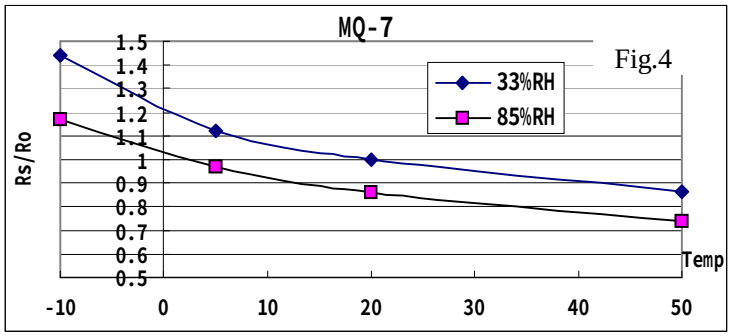


Fig.4

Fig.4 is shows the typical dependence of the MQ-7 on temperature and humidity.

Ro: sensor resistance at 100ppm CO in air at 33%RH and 20degree.

Rs: sensor resistance at 100ppm CO at different temperatures and humidities.

OPERATION PRINCIPLE

. The surface resistance of the sensor Rs is obtained through effected voltage signal output of the load resistance RL which series-wound. The relationship between them is described:

Rs\RL = (Vc-VRL) / VRL

Fig. 5 shows alterable situation of RL signal output measured by using Fig. 2 circuit output

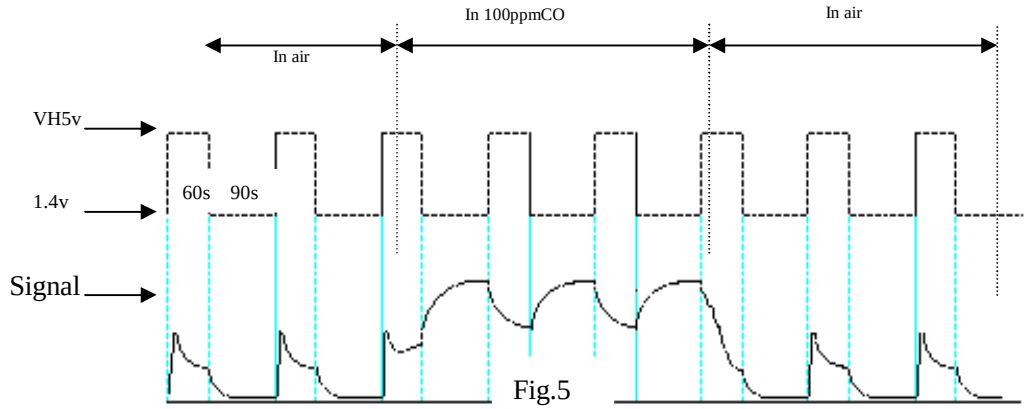


Fig.5

signal when the sensor is shifted from clean air to carbon monoxide (CO) , output signal measurement is made within one or two complete heating period (2.5 minute from high voltage to low voltage).

Sensitive layer of MQ-7 gas sensitive components is made of SnO_2 with stability, So, it has excellent long term stability. Its service life can reach 5 years under using condition.

SENSITVITY ADJUSTMENT

Resistance value of MQ-7 is difference to various kinds and various concentration gases. So, When using this components, sensitivity adjustment is very necessary. we recommend that you calibrate the detector for 200ppm CO in air and use value of Load resistance that(R_L) about 10 KΩ (5KΩ to 47 KΩ).

When accurately measuring, the proper alarm point for the gas detector should be determined after considering the temperature and humidity influence. The sensitivity adjusting program:

a. Connect the sensor to the application circuit.

b. Turn on the power, keep preheating through electricity over 48 hours.

c. Adjust the load resistance RL until you get a signal value which is respond to a certain carbon monoxide concentration at the end point of 90 seconds.

d. Adjust the another load resistance RL until you get a signal value which is respond to a CO concentration at the end point of 60 seconds .