Universitat de Girona
Escola Politècnica Superior

Treball final de grau

Estudi: Grau en Enginyeria Electrònica Industrial i Automàtica

Títol: Desenvolupament i implementació d'un sistema encastat per a la monitorització de dades essencials de sis ruscs a través de ThingSpeak

Document: 1. Memòria

Alumne: Antoni Capó Barceló

Tutor: Lluís Pacheco Valls
Departament: Arquitectura i tecnologia de computadors
Àrea: Arquitectura i tecnologia de computadors

Convocatòria (mes/any): Setembre 2023

ÍNDEX

1. INTRODUCCIÓ

1.1. Antecedents

L'apicultura, pràctica mil·lenària, té les seves arrels en la prehistòria quan els humans caçaven la mel d'abelles salvatges. A mesura que les societats evolucionaven, la relació amb les abelles es va aprofundir. Les civilitzacions antigues, com els egipcis, babilonis i grecs, van desenvolupar mètodes per a la recol·lecció de mel i cera, fins i tot atribuint simbolismes com la immortalitat a les abelles.

L'apogeu de l'apicultura va ser a Roma, on es van refinar les tècniques d'obtenció de mel i es van implementar maneres de manejar les colònies d'abelles. A l'Edat Mitjana, els monestirs van mantenir vives aquestes pràctiques i van afegir coneixements nous. A mesura que el món es globalitzava, les diferents regions desenvolupaven les seves pròpies tècniques apícoles, adaptades als seus entorns únics.

L'apicultura ha tingut un paper essencial en la pol·linització i la producció d'aliments, contribuint a la biodiversitat i a l'agricultura sostenible. Avui dia, a mesura que s'afronten desafiaments com la disminució de les poblacions d'abelles i el canvi climàtic, l'apicultura continua sent d'importància crítica.

1.2. Objecte

Els aficionats no tenen l'apicultura com l'activitat principal productiva en la seva vida, i aquesta sol ser més un passatemps que realitzen amb passió. Per a aquest motiu, es comú que no puguin gaudir del temps suficient per a tenir tots els ruscs controlats, i així detectar problemes, malalties a temps. Com a conseqüència d'aquesta falta de control, pot haver una disminució de la població del rusc (o la mort del mateix), propagació de paràsits a altres ruscs i una eficiència en la producció menor de l'esperada.

Avui en dia tenim la tecnologia suficient com per a monitoritzar diferents paràmetres i determinar l'estat de salut a distància. Sense la necessitat de fer un diagnosis presencial, es redueix considerablement el temps de demanda i així tenir una atenció adequada a les abelles per a detectar a temps qualsevol deficiència. Malgrat existeixen opcions en el mercat d'aparells que poden realitzar aquesta monitorització, no són opcions que qualsevol aficionat a l'apicultura podria permetre's.

Per a aquest motiu, els objectius d'aquest treball són proporcionar un sistema encastat que, a un cost baix, pugui permetre una monitorització a distància de les dades de ruscs més significatives, que puguin permetre avaluar l'estat de salut i l'època adequada per a l'extracció de mel. Ja que els ruscs solen posicionar-se en zones remotes, el dispositiu també ha de comptar amb un sistema d'alimentació aïllada que pugui enfrontar un funcionament permanent.

En el mercat actual han sorgit algunes empreses, com Anel o 3Bee, que han començat a comercialitzar productes semblants, però el seu cost no es assequible per al pressupost d'un apicultor aficionat. Aquestes empreses ofereixen productes que oscil·len des dels 285€ fins als 770€ (preus sense IVA). Per a les versions més bàsiques tan sols permet la monitorització d'un sol rusc, mentre que en les versions més costoses s'hi poden connectar fins a 3 ruscs. El sistema d'aquest projecte ha de poder reduir el cost de monitorització per rusc.

1.3. Especificacions i abast

El projecte està enfocat en el disseny i programació dels dispositius electrònics necessaris per a dur a terme la monitorització dels ruscs. Per a determinar l'estat de salut d'un rusc son essencials la temperatura i humitat de l'interior del rusc, i mitjançant l'evolució del pes es pot determinar el moment idoni per a la recol·lecció de mel, per tant aquestes seran les mesures que es tindran en compte en la monitorització. Un cop realitzades les mesures, aquestes s'enviaran a través d'internet, mitjançant l'ús d'una targeta SIM, a la plataforma ThingSpeak.

ThingSpeak és una plataforma d'Internet de les Coses (IoT) que permet als usuaris recopilar, analitzar i visualitzar dades en temps real generades per sensors, dispositius i altres fonts de dades. Amb una capacitat de dades molt alta en el seu pla gratuït, presenta un escenari perfecte pel dispositiu que es desenvoluparà en aquest projecte.

El sistema tindrà la capacitat de funcionar a través d'una bateria, que serà carregada a través d'una placa solar. Per a assegurar un funcionament permanent del sistema i preveure dies de baixa o nul·la producció solar, la bateria ha de poder oferir una autonomia de 4 dies. Es buscarà minimitzar al màxim l'ús innecessari d'energia.

El sistema encastat podrà emetre una xarxa Wi-Fi amb la qual, des d'un ordinador o telèfon amb la capacitat d'enllaçar-se a aquestes xarxes, permetrà calibrar els diferents sensors,

ajustar els paràmetres del nostre servidor de ThingSpeak i actualitzar altre informació que es consideri interessant la seva modificació.

Aquest projecte es tracta d'un prototipus que podrà ser instal·lat, en el futur, en un entorn real i verificar la viabilitat dels sensors i components seleccionats.

2. AVALUACIÓ DE L'ESTAT D'UN RUSC

Hi ha diferents paràmetres en un rusc que poden ser mesurats i, un cop analitzats, podrem tenir coneixement de l'estat de salut del mateix o si es troba en una època òptima per a l'extracció de mel o altres productes. En la monitorització d'apiaris, els paràmetres més deterministes per a aconseguir el punt anterior són la humitat interior, la temperatura interior i el pes del rusc.

Existeixen altres mesures que també ens donen diferents indicatius, com son la qualitat de l'aire i el soroll (les dues mesures a l'interior del rusc), però la lectura de resultats d'aquests darrers paràmetres encara es troben en estudi i no permeten l'avaluació directe del rusc. També cal mencionar que per a visualitzar aquests valors son necessaris uns sensors més delicats i una instal·lació més precisa.

Figura 1. Abella infectada de varroa, un dels paràsits més comuns

La temperatura ideal d'un rusc ha d'oscil·lar entre els 32ºC i els 36ºC en la zona de cria (normalment la part més interna del rusc). Amb la temperatura podem saber si les abelles poden termoregular el rusc, en cas contrari ho podríem identificar perquè el sensor marcaria una temperatura aproximada a l'exterior.

Per a la humitat ens basarem en un estudi realitzat per la Universitat de Córdoba, que ens indica que els valors òptims són entre un 50% i un 60% en la càmera de cria, però que en la resta del rusc es pot oscil·lar entre un 40% i un 70%. Un cop superat aquests llindars, poden proliferar fongs que podrien acabar amb l'eixam.

Figura 2. Quadre afectat per fongs degut a l'excessiva humitat

Finalment amb el pes podrem detectar diferents situacions. En un mateix dia podrem veure el moment en que l'eixam surt cada dia del rusc, i amb la disminució que ens mostrarà el sensor podrem saber la magnitud de l'eixam. Si detectem que al final del dia hi ha hagut una pèrdua important, això voldrà dir que l'eixam ha eixamat, amb lo que havia sorgit una altre reina al rusc i, la vella o la nova ha abandonat el rusc amb aproximadament la meitat de l'eixam. Amb el pas dels dies hauríem de poder veure l'augment gradual del pes del rusc, per tant ens podríem fer una idea de l'evolució de la recol·lecta de mel que realitzen les abelles. Quan estam en època de campanya i veiem que el l'augment ha deixat de produir-se, o inclòs comença a minvar, les mesures ens diran que aquest es el moment idoni per a l'extracció de mel. El pes aproximat d'un rusc artificial (sense cera ni eixam a dintre) és de 20 kg.

Figura 3. Eixam recent format que cerca lloc per a establir-se

3. HARDWARE

Un sistema encastat és un tipus de sistema informàtic dissenyat per a una funció específica o un conjunt específic de funcions. Aquests sistemes solen estar incrustats en altres dispositius o maquinària més grans i tenen l'objectiu de controlar i supervisar les operacions d'aquests dispositius. Són dissenyats per a un propòsit particular i no estan destinats a ser reprogramats per a altres funcions com ho podria ser una computadora general.

Exemples comuns de sistemes encastats inclouen microcontroladors en electrodomèstics, sistemes de navegació en vehicles, sistemes de control en maquinària industrial i fins i tot en aparells mèdics. Aquests sistemes estan optimitzats per a l'eficiència i el rendiment en la seva tasca específica, i sovint funcionen amb recursos limitats com memòria i potència de càlcul.

En general els sistemes encastats es poden programar directament amb el llenguatge assemblador incorporat en el mateix microprocessador o amb compiladors específics que poden utilitzar llenguatges com C o C++. Tanmateix, també existeixen diverses plataformes desenvolupades per diversos fabricants les quals proporciones diverses eines pel desenvolupament del disseny d'aplicacions i prototips com són, Arduino, Raspberry Pi, etc.

En el present capítol es descriurà quines són les característiques i funcionalitats principals, per tal de desenvolupar les tasques necessàries per a la monitorització dels ruscs. També es descriuran tots els elements escollits pel disseny del hardware del dispositiu i quins han sigut els criteris alhora de l'elecció.

3.1. Controlador

Per a estalviar temps de disseny i treballar en un entorn robust i econòmic, he escollit utilitzar una de les plataformes de desenvolupament que he mencionat en l'apartat anterior. La plataforma per la qual m'he decantat ha estat Raspberry Pi Zero WH. Es tracta d'un ordinador de placa única (SBC) que, encara que el seu format es petit, compta amb nombrosos ports d'entrada i sortida i connectivitat Wi-Fi integrada. Aquest controlador, juntament amb altres de la família Raspberry Pi, son àmpliament utilitzats en projectes d'IoT degut a la seva capacitat de processament en comparació al seu baix consum i cost.

El model Zero WH és el menys potent dels models que tenen connectivitat Wi-Fi (característica que volem tenir per a calibrar els sensors i modificar paràmetres de forma remota), però degut

a que les tasques que es realitzaran són poc demandants, el Zero WH és més que suficient, permetent-nos estalviar diners, espai i consum.

Figura 4. Raspberry Pi Zero WH

3.2. Sistema d'alimentació

Actualment l'únic sistema viable per a alimentar el sistema en un lloc remot on no disposem de connexió a la xarxa és mitjançant l'ús d'una placa solar i una bateria. Aquest sistema s'ha fet molt popular i assequible aquests darrers anys per a alimentar petits dispositius en situacions similars a la d'aquest projecte. El més important és dimensionar la placa i la bateria per a que puguem tenir energia suficient per a mantenir el sistema en funcionament durant 4 dies com a mínim sense cap tipus de càrrega, i que la placa solar pugui aportar suficient energia com per a suplir el consum del sistema i mantenir la càrrega de la bateria.

En l'annex B.2 s'ha calculat que la capacitat de la bateria de 12V ha de ser mínim de 17,67Ah per a complir amb les especificacions del projecte. En el mercat la capacitat normal immediatament superior és de 18Ah. En el mateix annex sabem que el sistema consumirà com a màxim 53Wh cada dia. Per norma general es considera que una placa solar pot proporcionar una energia igual a la potència màxima de la placa durant 5 hores en un dia normal de sol. D'aquesta forma, una placa de 20W pot proporcionar 100Wh en un dia normal, suficient per a cobrir el consum del sistema i amb els excedents carregar la bateria en cas que fos necessari.

La placa solar, per a carregar la bateria i subministrar alimentació al sistema precisa d'un regulador de càrrega. Un regulador de càrrega, també conegut com a controlador de càrrega, és un dispositiu utilitzat en sistemes d'energia solar o eòlica per controlar i regular la càrrega de les bateries que emmagatzemen l'energia generada per fonts renovables, com ara els panells solars o els aerogeneradors. La seva funció principal és protegir les bateries i optimitzar la càrrega i la descàrrega d'energia per garantir un rendiment eficient i una vida útil prolongada de les bateries. Per sort, actualment aquets panells solar de tan baixa potència solen vendre's amb un kit que ja incorpora aquest regulador de càrrega.

Figura 5. Kit solar que conté un panell solar policristal·lí de 20W i un controlador de càrrega de 10A

3.3. Sensors

Els sensors són aquells elements que transformen les magnituds físiques o químiques de l'entorn en elèctriques per tal de poder llegir-les i d'aquesta forma posteriorment actuar d'acord al seu estat. Es disposarà de diversos tipus de sensors: temperatura, humitat i pes.

3.3.1. Temperatura

El sensor de temperatura que s'utilitzarà és el DS18B20. Aquest sensor és una opció popular i versàtil per a la mesura precisa de la temperatura en diferents aplicacions, gràcies a la seva facilitat d'ús, precisió, immunitat al soroll i compatibilitat amb diferents plataformes de

desenvolupament. El DS18B20 és un sensor de temperatura digital que utilitza una interfície de comunicació unidireccional, anomenada 1-Wire.

1-Wire és una interfície de comunicació unidireccional desenvolupada per la companyia Maxim Integrated. Aquesta tecnologia permet connectar múltiples dispositius (fins a 30) a un únic fil de dades i utilitzar aquest fil per transmetre tant les dades com l'alimentació elèctrica als dispositius connectats.

Figura 6. Sensor temperatura DS18B20

Existeixen molts altres models de sensors de temperatura en el mercat, pensats per a ser integrats en sistemes d'IoT, amb diferents tecnologies, com el PT100. El que ha provocat l'elecció del DS18B20 ha estat la interfície 1-Wire, ja que simplifica la connexió i facilita la lectura de múltiples sensors amb un sol cable, reduint així el nombre de ports de la Raspberry Pi ocupats. El DS18B20 ofereix un rang de temperatura entre -55ºC i 125ºC, rang que compleix de sobre amb les exigències d'aquest projecte.

3.3.2. Humitat

En el camp dels sensors d'humitat en sistemes d'IoT hi ha destaquen els sensors capacitius. Aquests sensors utilitzen la variació de la capacitància per a determinar la humitat. Són precisos i tenen una resposta ràpida. Les sèries més utilitzades són la DHT (DHT11 i DHT22) i la SHT (SHT11, SHT21 i SHT30). D'aquestes dues sèries la que més ens interessa és la DHT, ja que cada sensor que connectem tan sols ocuparia un port de la Raspberry Pi, mentre que els sensors de la sèrie SHT necessiten 2 ports per a transmetre la informació.

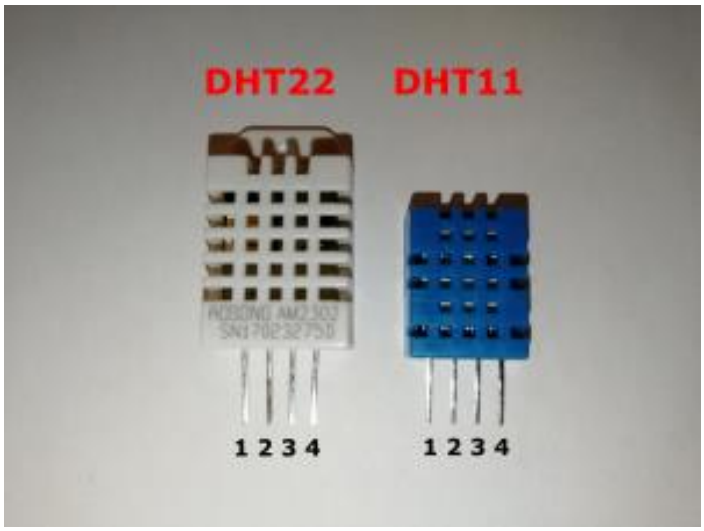

Figura 7. Sensors de la sèrie DHT

Entre el DHT11 i el DHT22 s'escollirà el DHT22 degut a que, mentre el DHT11 ofereix una precisió del ±5% en un rang del 20% 80%, el DHT22 ens proporciona una precisió d'entre el ±2-5% en un rang del 0% al 100%. La diferència de cost entre els dos sensors és molt baixa, i gràcies a les millors prestacions del DHT22 provoca que la seva elecció sigui indubtable.

Sensor	DHT22	DHT11
Rang de la temperatura	-40ºC - +125ºC	0ºC – 50ºC
Precisió de la temperatura	±0,5ºC	±2ºC
Rang de la humitat	0% - 100%	20% - 80%
Precisió de la humitat	±2-5%	±5%
Dimensions	15,1mm x 25mm x 7,7mm	15,5mm x 12mm x 5,5mm
Voltatge d'operació	3V - 5V	3V -5V
Consum	2,5mA	2,5mA

Taula 1. Especificacions dels sensors de la sèrie DHT

Com es pot veure a la taula anterior, aquest sensors també incorporen un sensor de temperatura que pot servir perfectament per a la monitorització dels ruscs, per tant no tindria sentit l'ús del sensor DS18B20. S'ha decidit conservar el DS18B20 degut a que tan sols perdríem la connectivitat d'un port de la Raspberry Pi i, a canvi, tenim la possibilitat de monitoritzar la temperatura interior dels ruscs en dos punts diferents, sense haver de modificar el hardware, si en un futur ho trobem adient.

3.3.3. Pes

Per al pes és important tenir en compte que hi ha diferents models de ruscs, cada un d'ells amb diferents bases. Això implica que s'haurà de buscar una solució que pugui ser el més universal possible.

Per a mesurar el pes en aquests tipus de sistemes és bastant comú l'ús de cèl·lules de càrrega. Les cèl·lules de càrrega funcionen sobre la base d'un principi físic conegut com l'efecte piezoelèctric o l'efecte elàstic. Quan una força o una càrrega es transmet a través de la cèl·lula, aquesta experimenta una deformació mecànica que provoca canvis en les seves propietats elèctriques, com la resistència o la capacitat. Aquesta variació es mesura per determinar la força o la càrrega aplicada. Les cèl·lules de càrrega ofereixen una alta precisió i resolució en la mesura de càrregues. La resolució es pot ajustar mitjançant l'elecció de la cèl·lula de càrrega adequada i la calibració.

Figura 8. Cèl·lula de càrrega de 50kg

En les bàscules d'ús domèstic s'utilitzen 4 cèl·lules de càrrega que solen suportar 50kg cadascuna, proporcionant una lectura de fins a 200kg, i solen mostrar una precisió de 0,1kg. El rang de treball i precisió d'aquestes bàscules són ideals per a mesurar el pes dels ruscs, per tant s'ha decidit copiar el sistema. A part de les 4 cèl·lules, la bàscula compta amb un circuit electrònic de condicionament per a processar la senyal i convertir-la en una lectura de pes comprensible. El circuit que s'utilitzarà en aquest projecte serà el HX711, un amplificador i convertidor analògic-digital (ADC) d'alt rendiment dissenyat específicament per a la mesura de càrregues o pes a través de cèl·lules de càrrega. És un component electrònic molt utilitzat en aplicacions on es requereix la mesura precisa de pes. Aquest component, a diferència que els altres sensors, necessita 2 ports de la Raspberry Pi per a transmetre la informació.

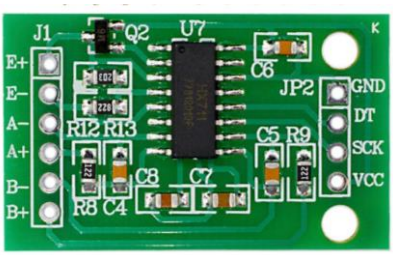

Figura 9. Amplificador i convertidor ADC HX711

3.4. Targeta SIM

El dispositiu que s'està desenvolupant en aquest treball només té sentit si la informació que mesurem pot ser consultada i analitzada de forma remota, ja que de forma presencial tan sols fa falta un anàlisi visual de l'interior del rusc per a l'avaluació de l'estat del mateix. Per a realitzar aquesta monitorització remota l'ús d'internet serà essencial, i ja que en meitat de la naturalesa no comptem amb cap connexió Wi-Fi o Ethernet que ens pugui proporcionar internet a la nostra Raspberry Pi, l'única connectivitat possible serà a través d'una targeta SIM.

Una targeta SIM, que significa "Subscriber Identity Module" o "Mòdul d'Identitat del Subscriptor" en anglès, és una petita targeta de plàstic que s'insereix en un telèfon mòbil o en un dispositiu compatible amb comunicacions mòbils per a emmagatzemar la informació d'identitat d'un subscriptor i habilitar la comunicació amb la xarxa de telefonia mòbil.

Per a obtenir una d'aquestes targetes s'ha de realitzar un contracte amb una companyia telefònica. La solució més econòmica que s'ha pogut trobar ha estat a través de la companyia HITS Mobile (degut a la variabilitat del sector de la telefonia, i a les possibles avantatges que l'apicultor pugui tenir amb el seu actual operador, l'elecció de la companyia i contracte més econòmic pot ser diferent).

HITS Mobile és una empresa de telecomunicacions que opera en el mercat espanyol. Ofereix serveis de telefonia mòbil, internet i altres serveis de comunicació als seus clients, però la tarifa que ens interessa es la de "Pagament per ús". Aquesta tarifa, com el seu nom indica, el client paga per l'ús que realitzi. En aquest cas, HITS Mobile cobra de forma mensual 1€ pel manteniment de la targeta i ens permet utilitzar 500MB sense cap sobrecost. Excedir-nos d'aquets límit pot elevar considerablement la factura, per tant el nostre sistema haurà de consumir menys de 500MB casa mes, més que suficient per a l'enviament de multitud d'informació.

3.5. Mòdem Stick

Un "mòdem stick" o "dongle de mòdem" és un petit dispositiu USB que incorpora les funcions d'un mòdem i es connecta a una computadora o un dispositiu compatible mitjançant un port USB. Aquests dispositius es fan servir per a proporcionar connexió a Internet mòbil mitjançant xarxes de telefonia mòbil, com 3G, 4G o 5G, i solen ser una opció de connexió a Internet portàtil que pot ser útil quan no hi ha una connexió Wi-Fi disponible. El mòdem stick està dissenyat per a utilitzar-se amb targetes SIM de proveïdors de serveis de telefonia mòbil. Un cop s'insereix una targeta SIM compatible al dispositiu, aquest es pot connectar a la xarxa de telefonia mòbil per a proporcionar connexió a Internet mòbil.

Figura 10. Huawei E3531

Aquests dispositius solen ser proporcionats per les mateixes operadores i, per tant, limitats a les targetes SIM de la seva empresa. Una de les poques marques reconegudes que proporciona mòdem sticks lliures és Huawei. Dintre dels models que Huawei té al mercat s'escollirà el model més econòmic, que accepti com a mínim connexió 4G i que sigui els controladors de Raspberry Pi siguin compatibles. El motiu és principal del 4G és degut a que les antenes que proporcionen senyals 3G tenen els dies comptats, i segons la localització de l'apiari no seria estrany que no hi hagi connexió 3G en un futur pròxim. El model que compleix les nostres necessitats és el Huawei E3531.

3.6. Witty Pi

Els paràmetres seleccionats en el ruscs pel seu anàlisis tenen una variació gradual, per tant la monitorització continua és un desaprofitament d'energia i de dades. Per a solucionar això, el sistema s'iniciarà automàticament quan sigui necessari realitzar noves mesures i s'apagarà un cop la informació hagi estat llegida i enviada. D'aquesta forma també s'allarga la vida útil del sistema, ja que tenir-lo en funcionament les 24 hores del dia podria perjudicar la vida del components electrònics.

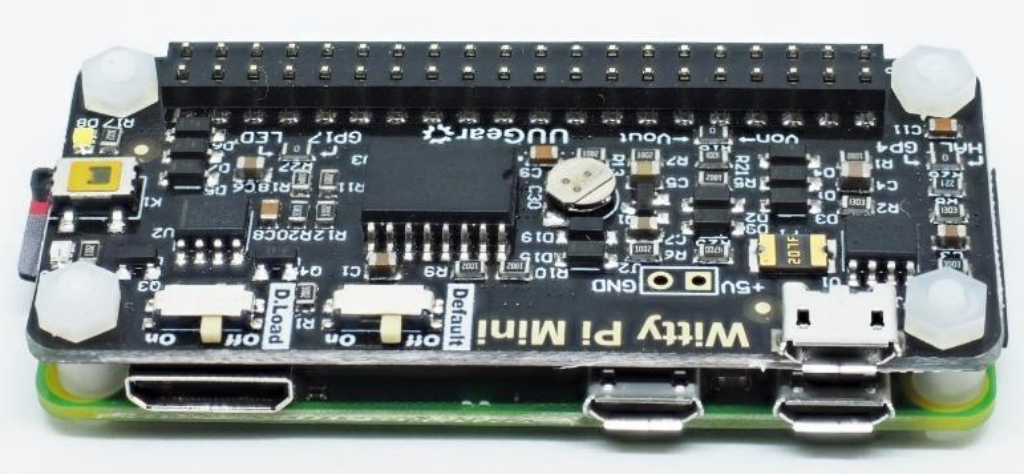

Figura 11. Witty Pi Mini 3 acoblada a una Raspberry Pi Zero W

La Raspberry Pi pot ser programada per a ser apagada en un moment en concret, però després ella mateixa no podria ser iniciada. Per a aquest motiu s'ha decidit utilitzar el mòdul Witty Pi 3 Mini. Witty Pi és un dispositiu electrònic creat per Pi Supply, una empresa especialitzada en productes per a Raspberry Pi, que s'afegeix a una Raspberry Pi i permet controlar i gestionar l'apagada (shutdown) i l'arrencada (boot) programades de la Raspberry Pi.

El funcionament d'aquest dispositiu és ser l'intermediari entre l'alimentació i la Raspberry Pi Zero, d'aquesta forma quan es vol iniciar la Raspberry Pi el mòdul Witty Pi fa de pont i deixa passar l'alimentació, però en el moment en que es desitja realitzar una apagada, aquest envia l'ordre d'apagada a la Raspberry Pi, i un cop finalitzat tan sols la Witty Pi en mode repòs tindria un consum d'energia. La Witty Pi 3 Mini és el model més recent dissenyat per a la Raspberry Pi Zero W.

4. DISSENY

En el present capítol es descriurà el disseny del sistema encastat i els criteris alhora de l'elecció. En l'apartat anterior ja hem fet menció dels principals elements que ha de tenir el sistema, però ara es el moment de visualitzar la forma en la que han d'estar connectats pel seu correcte funcionament i l'aprofitament màxim de les prestacions.

4.1. Ports d'entrada i sortida de la Raspberry Pi Zero W

La Raspberry Pi és el cervell del sistema, així que realitzarem el disseny del sistema al seu voltant. La Raspberry Pi Zero WH compta amb 40 pins i 5 ports d'entrades i sortides. L'ús d'aquests ports d'E/S es troba especificat en la següent taula.

Port d'E/S	Utilitat pel projecte
MicroSD	Contindrà la targeta microSD que emmagtzema tot el programari que controlarà la Raspberry Pi
Mini HDMI	No necessari pel funcionament del sistema
MicroUSB (funció OTG)	Connexió del mòdem stick per a la connexió 4G
MicroUSB (alimentació)	No necessari, ja que l'alimentació es realitzarà a través del mòdul Witty Pi 3 Mini
Camera	No necessari pel funcionament del sistema

Taula 2. Ús dels ports d'E/S de la Raspberry Pi Zero W

Dels 40 pins d'entrada i sortida que té la Raspberry Pi, 28 son entrades i sortides d'ús general (GPIO) i podrem definir la funció de cada un d'ells a conveniència. Els 12 pins que perdem són utilitzats per a proporcionar punts d'alimentació a 5V i 3,3V i massa (GND). En la resta de pins ha d'estar connectat el mòdul Witty Pi 3 Mini, els sensors, un botó (per a canviar el mode de funcionament del sistema entre mesura i configuració) i un LED que ens indiqui en tot moment en quin estat es troba el sistema. Dels 28 GPIO que proporciona Raspberry Pi, els pins 27 i 28 són utilitzats per la pròpia Raspberry Pi per a funcions internes i es recomana deixar-los lliure. Segons el datasheet del Witty Pi, aquest mòdul necessita fer ús dels pins 3, 5, 7, 11 i 8. Pels sensors de temperatura necessitem tan sols 1 pin qualsevol, independentment del nombre de ruscs que vulguem analitzar. Necessitarem també un pin per a connectar-hi un botó i un altre per a col·locar el LED. Amb aquests requisits, ens queden 18 pins que podrem utilitzar amb total llibertat per a connectar la resta de sensors.

Per a cada rusc, sense comptar el sensor de temperatura, haurem d'utilitzar 3 canals (1 pel sensor d'humitat i 2 pel convertidor ADC que ens proporcionarà el pes). Això vol dir que per a

cada rusc que vulguem afegir necessitarem utilitzar 3 pins de la Raspberry Pi. Si ens queden 18 pins això significa que tenim els pins justos com per a poder monitoritzar 6 ruscs amb una mateixa Raspberry Pi.

GPIO	Ús	Ubicació física		Ús	GPIO
	3,3V	1	2	5V	
2	Witty Pi	3	4	5V	
3	Witty Pi	5	6	GND	
4	Witty Pi	7	8	Witty Pi	14
	GND	9	10	HX711 [SCK] nº6	15
17	Witty Pi	11	12	HX711 [DT] nº6	18
27	DS18B20 (nº1 - nº6)	13	14	GND	
22	DHT22 nº1	15	16	DHT22 nº6	23
	3,3V	17	18	HX711 [SCK] nº5	24
10	HX711 [DT] nº1	19	20	GND	
9	HX711 [SCK] nº1	21	22	HX711 [DT] nº5	25
11	DHT22 nº2	23	24	Botó	8
	GND	25	26	Base transistor NPN (LED)	7
0	Raspberry Pi	27	28	Raspberry Pi	1
5	HX711 [DT] nº2	29	30	GND	
6	HX711 [SCK] nº2	31	32	DHT22 nº5	12
13	DHT22 nº3	33	34	GND	
19	HX711 [DT] nº3	35	36	HX711 [SCK] nº4	16
26	HX711 [SCK] nº3	37	38	HX711 [DT] nº4	20
	GND	39	40	DHT22 nº4	21

Taula 3. Distribució de les entrades i sortides generals de la Raspberry Pi Zero W

4.2. LED

El LED que ens indica l'estat del sistema té poca lluminositat (difícil de detectar amb la llum de sol) si l'alimentació es realitza a través d'un pin GPIO, ja que aquests funcionen amb un voltatge màxim de 3,3V. Per a aquest motiu s'ha decidit utilitzar l'alimentació de 5V i, per controlar l'activació del LED, s'utilitzarà un transistor NPN (en aquest projecte s'ha utilitzat el PN2222A, però qualsevol transistor NPN genèric que pugui suportar 5V pot servir igual) com a interruptor. Aquest transistor tindrà el col·lector connectat a l'alimentació de 5V, l'emissor al positiu del LED (el negatiu del LED ha d'estar connectat a massa) i la base a un pin GPIO de la Raspberry Pi. D'aquesta forma, sempre que el pin estigui configurat com a HIGH (3,3V), tindrem intensitat circulant per la base i el transistor estarà en saturació, llavors el LED podrà rebre alimentació i il·luminar-se a través dels 5V. Quan el GPIO estigui configurat com a LOW (0V), per la base no podrà circular intensitat i el transistor es trobarà en curt, provocant que el LED romaní apagat.

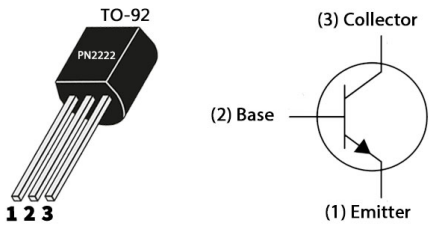

Figura 12. Transistor PN2222A

Per al LED cal una resistència que limiti la intensitat que circularà per ell i evitar que s'espatlli (en aquest cas, el LED seleccionat ja porta aquesta resistència, però en altres casos el valor comú d'aquestes resistències sol ser de 200Ω). Per a la base del transistor també fa falta que la intensitat estigui limitada, però la Raspberry Pi compta amb unes resistència push-up i push-down (en funció de l'ús del pin s'activa una o l'altre) a cada un dels pins GPIO. Aquesta resistència interna de la Raspberry Pi ja ens proporciona la limitació necessària.

4.3. Mòdem Stick

Per a poder tenir connexió a internet a través del Huawei E3531 primer ha de passar per a una petita configuració a qualsevol computador. Un cop introduïda la targeta SIM al mòdem, s'ha hagut de connectar al port USB tipus A i esperar que el navegador predeterminat del sistema ens obri la finestra de configuració (en cas de que no s'hagi obert la finestra, es pot obrir en el navegador en la direcció "http://192.168.8.1"). Un cop aquí, demana introduir el PIN de la targeta SIM i una casella indica si es vol deshabilitar l'ús del PIN en les properes connexions. S'ha d'introduir el PIN i marcar la casella, ja que quan es trobi en la Raspberry Pi no tindrem la capacitat d'indicar-lo. Un cop inicialitzat el mòdem ja es pot desconnectar del computador.

Ara la Raspberry Pi ja pot connectar-se a internet a través del mòdem, sempre i quan s'utilitzi un cable OTG connectat al port microUSB de la Raspberry Pi (el que es troba en la part més pròxima del centre). Un cable OTG (On-The-Go) és un tipus especial de cable USB que permet connectar dispositius electrònics com telèfons mòbils, tauletes, càmeres digitals, controladors de jocs i altres perifèrics USB a altres dispositius electrònics com ara telèfons mòbils o tauletes sense necessitat de tenir un host USB tradicional. Aquest tipus de cable és especialment útil per a establir connexions directes entre dispositius sense la necessitat d'un ordinador intermediari. El mòdem activa automàticament la targeta SIM un cop té alimentació i els controladors natius de la Raspberry Pi detecten aquesta connexió com un accés ethernet.

4.4. Cèl·lules de càrrega i convertidor HX711

Les cèl·lules de càrrega han de connectar-se seguint l'esquema d'un pont de Wheatstone. Aquesta configuració es pot veure en la figura 13, i es basa en alimentar el sistema a un voltatge constant i conegut (Vin), amb 1 o més resistències variables i mesurar la diferencia de voltatge als punts entremitjos de cada branca (Vout). En el nostre cas les 4 resistències són variables i idèntiques (si no tenim en compte les toleràncies de fabricació). Això vol dir que si no es connecten adequadament, tots els punts del sistema variaran de la mateixa forma i la diferencia de potencial a Vout mai podria variar.

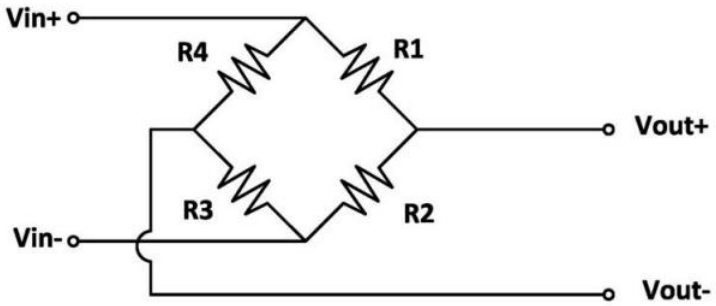

Figura 13. Pont de Wheatstone

Amb el sistema en repòs, totes les resistències haurien de tenir quasi el mateix valor, amb lo que Vout hauria de ser pràcticament zero. Per a poder mesurar variacions, les resistències R1 i R3 han de tenir un comportament contrari a les R2 i R4, es a dir, si un conjunt augmenta la seva resistència en front a una càrrega, l'altre conjunt ha de disminuir. Això provocarà que la tensió als punts Vout+ i Vout- variïn de forma proporcional a la càrrega, i així establir una relació directa entre Vout i el pes. Si el conjunt R1 i R3 és el que disminueix, tindrem a cada un dels seus extrems una caiguda de tensió menor que en l'estat anterior, amb lo que el voltatge a Vout+ haurà incrementat i a Vout- haurà disminuït. Per a realitzar la correcta connexió tan sols serà necessari seguir el següent esquema i respectar els codis de color dels cables de les cèl·lules de càrrega.

S'ha de tenir en compte que aquestes cèl·lules també sofreixen una variació de la seva resistència degut a la temperatura, per a aquest motiu s'instal·larà un DS18B20 per a mesurar la temperatura exterior i realitzar una correcció dels resultats obtinguts en la mesura del pes.

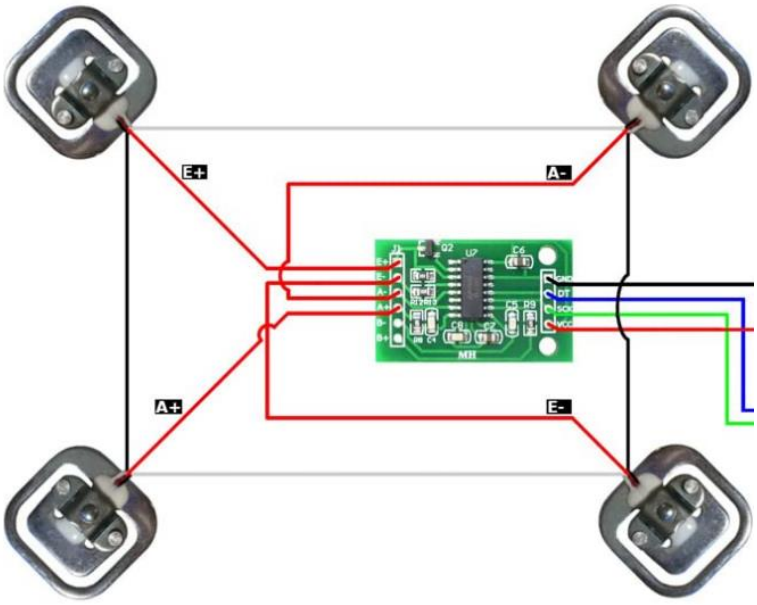

Figura 14. Esquema de connexió de 4 cèl·lules de càrrega

Existeixen diferents models de ruscs que existeixen, i és sovint que la part inferior disposi de ventilació i per tant no pugui superfície suport. Per a aquest motiu, es disposaran de dues barres d'alumini quadrades de 50cm de longitud i en cada una d'elles es muntaran dues cèl·lules amb una separació de 35cm entre elles. D'aquesta forma cada parell de cèl·lules es poden ajustar a l'amplada del rusc i es pot assegurar que aquest reposi de forma consistent als 4 punts de suport. Per a muntar les cèl·lules en les barres d'alumini s'han imprès 4 suports creats per l'usuari "patrick3345" al portal web "Thingiverse".

Figura 15. Barres d'alumini amb les cèl·lules de càrrega muntades

Al mercat hi ha moltes marques que ofereixen mòduls HX711 i alguns d'aquests poden tenir problemes de fàbrica. Idealment la resistència entre E- i GND ha de ser zero, si no es el cas es pot connectar un cable entre aquests dos punts. Hi ha alguns mòduls que presenten amb l'ús del canal B present més soroll de l'habitual. I per finalitzar, hi ha errors sovint de sincronització entre la Raspberry Pi Zero i el mòdul HX711 quan aquest funciona a 10Hz (freqüència predeterminada amb la que es distribueixen), però si s'activa el mode a 80Hz aquest problemes haurien de desaparèixer. Per a activar els 80Hz s'ha d'anular la connexió del pin 15 de l'integrat HX711 amb GND i connectar el pin 15 a VCC.

4.5. Mòduls dels sistema encastat

El sistema ha d'estar muntat a l'exterior, però aquest es troba exposat als agents meteorològics i ambientals de l'exterior. Per a simplificar el muntatge es dividirà el sistema en 4 mòduls diferents, cada un d'ells a dintre d'una caixa de connexió estanca.

El primer dels mòduls contindrà únicament la bateria, i d'aquesta caixa tan sols sortiran els cables d'alimentació d'aquesta. El següent mòdul protegirà el regulador de càrrega, i en aquest cas tindrem els cables provinents de la placa solar, els cables que alimenten la bateria i un connector USB tipus A de caravana (l'alimentació d'aquest es realitza a 12V, però el mateix endoll transforma aquest voltatge als 5V del port USB). Aquest segon mòdul s'encarrega de la distribució de l'energia de tot el sistema.

Figura 16. Interior del mòdul de la bateria (esquerra) i el mòdul de distribució de l'energia (dreta)

El mòdul més important és el cervell del sistema, el seu interior contindrà la Raspberry Pi Zero W (amb el mòdul Witty Pi 3 Mini connectat a sobre), el mòdem Huawei E3531, el botó per a controlar l'estat del sistema, el LED per saber en quin estat ens trobem i un sensor DS18B20 per a mesurar la temperatura ambiental. Los connexions exteriors que tindrà aquest mòdul és un cable d'alimentació connectat al mòdul Witty Pi 3 Mini, i que rebrà energia a través del connector USB del mòdul que conté el regulador de càrrega. Les altre connexions externes del mòdul principal són 6 connectors port sèrie. Cada un d'aquests connectors està pensat per comunicar-se amb el darrer mòdul.

Figura 17. Interior del mòdul principal

Figura 18. Laterals del mòdul principal

El darrer mòdul és l'únic que pot ser replicat, ja que és el que protegeix les connexions dels sensors de cada rusc, per tant podem tenir des d'un fins a sis d'aquests mòduls. Al igual que el mòdul principal, té un connector extern port sèrie que servirà per a establir la comunicació dels sensors amb el mòdul principal. Al seu interior tan sols tindrem la connexió dels diferents sensors i el convertidor HX711, ja que aquest no fa falta que estigui en l'exterior. La comunicació entre el mòdul principal i el mòdul de sensors s'ha decidit realitzar a través de port sèrie ja que és el connector amb almenys 6 pins (en aquest cas tenim 9,) on tant el connector com el cable solen ser bastant comuns i econòmics. Malgrat s'utilitzin connector i cable port sèrie, la connexió serà directa amb la Raspberry Pi, no hi haurà cap protocol sèrie.

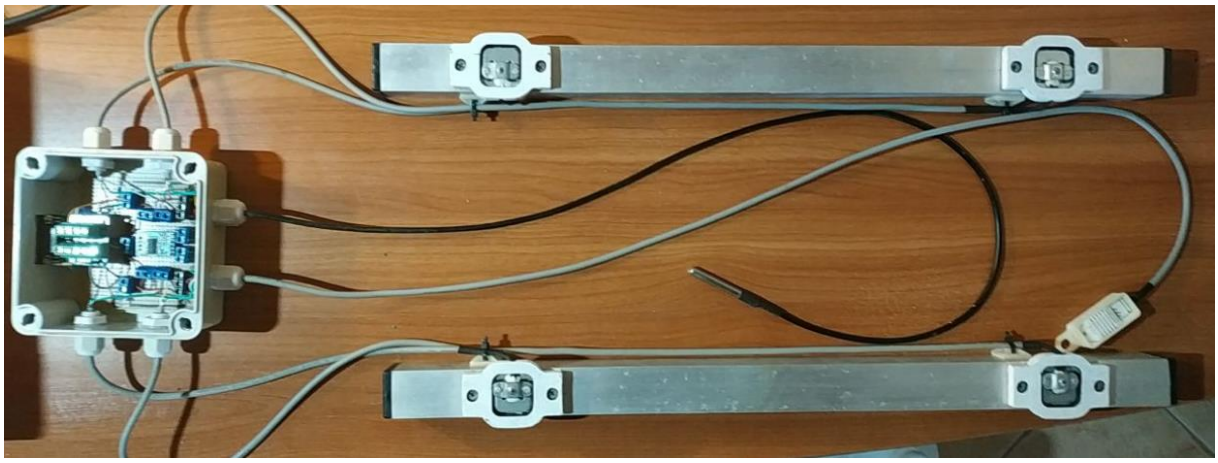

Figura 19. Mòdul de sensors per a un rusc

4.6. Altres aspectes importants dels sensors

Com es pot observar en la figura 19, els cablejat que comunica les cèl·lules de càrrega amb la caixa del mòdul no es l'original, això es degut a que s'han substituït per un cable tripolar 0,22mm^2 preparat per a suportar l'ambient exterior. Aprofitant el mateix cable, s'ha connectat el DHT22 mitjançant un connector JST-XH de 3 pins.

Per a les futures proves en un rusc real, s'ha protegit el sensor DHT22 mitjançant la impressió d'un encapsulat dissenyat per l'usuari "AlexBinary" i publicat a la plataforma "Thingverse". La raó és degut a que les abelles immobilitzen amb una pasta tot el que pugui sofrir moviment a dintre del rusc. Pel sensor de temperatura això és indiferent, però el DHT22 necessita estar en contacte directe amb l'aire, i les abelles podrien tapar amb aquesta pasta les ventilacions del sensor. La coberta tan sols s'utilitzarà en el prototip per verificar el comportament de les

abelles i, en cas des produir-se, evitar que el sensor es pugui fer malbé i poder ser reutilitzat. Si es produeix aquest escenari, s'estudiaran alternatives per a evitar-ho.

Pels sensors DS18B20 cal remarcar que utilitzen la comunicació 1-Wire. Això vol dir que el cable d'informació de tots ells es comú.

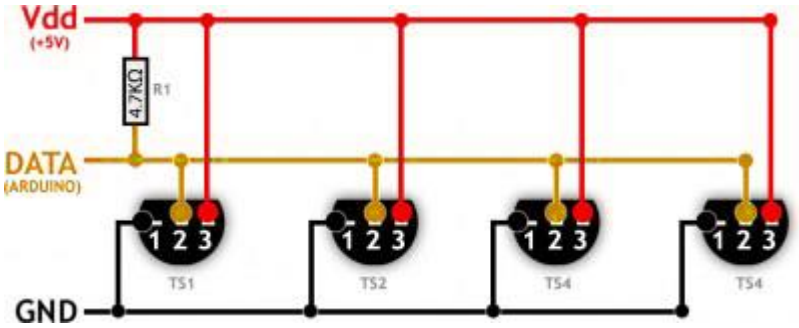

Figura 20. Exemple de la connexió de tres DS18B20 utilitzant la configuració 1-Wire

Per últim, tant els fabricants del sensor DS18B20 com del DHT22 recomanen utilitzar una resistència push-up (connexió entre VCC i el pin d'informació) per a estabilitzar les lectures i augmentar la fiabilitat. Aquesta resistència assegura que la tensió sigui alta (en l'estat lògic 1, ja que son sensors digitals) quan el sensor no està transmetent dades. Es recomana una resistència d'un valor de 4,7kΩ. Pels sensors DS18B20 tan sols farà falta l'ús d'una única resistència per a tots ells, per lo que es pot instal·lar al mòdul principal del sistema, però pels DHT22 farà falta instal·lar-ne una en cada mòdul que conté la connexió del sensors per a cada rusc.

5. THINGSPEAK

ThingSpeak és una plataforma d'Internet de les Coses (IoT) que permet als usuaris recopilar, analitzar i visualitzar dades de sensors i dispositius connectats. És un servei en línia que facilita la creació d'aplicacions i projectes IoT, especialment quan es tracta de monitorització i control de dades a través de la web. Algunes de les funcions principals de ThingSpeak inclouen la recopilació de dades, emmagatzemament i processament de dades a través de MATLAB, visualització de dades mitjançant gràfics i taules en temps real, accions automàtiques i regles basades en les dades i integració amb altres serveis i plataformes IoT.

Per a tenir un compte a ThingSpeak tan sols ens caldrà un correu electrònic operatiu. La llicència gratuïta permet recopilar tres milions de dades de forma anual. Per a arribar a aquest límit absurd, cada dia hauríem de recopilar 8.220 dades, en aquest cas la nostra monitorització no es tan abusiva i no s'assolirà aquest límit. Una altre limitació és la creació de 4 canals. Un canal a ThingSpeak és com un contenidor que permet agrupar dades relacionades i ofereix eines per gestionar i presentar aquestes dades de manera efectiva. Cada un dels canals pot contenir fins a 8 camps de dades. L'opció més lògica amb aquestes eines i el nostre dispositiu és crear 3 canals (temperatura, humitat i pes) i en cada un dels canals habilitar 6 camps per a poder emmagatzemar la mesura de cada un dels ruscs.

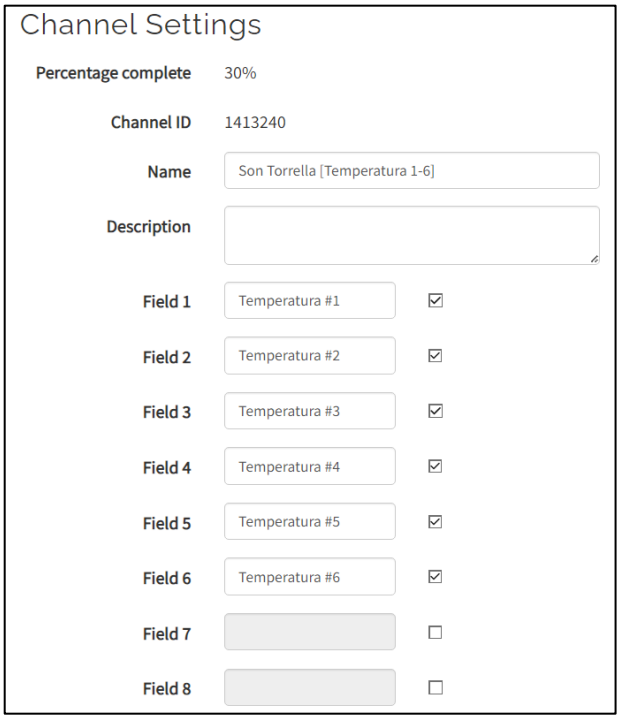

Figura 21. Exemple de creació d'un canal a ThingSpeak per a la temperatura

Un cop creat el canal, podrem entrar a la vista privada de cada un d'ells i configurar com volem que es visualitzin les nostres dades. Per a poder visualitzar una gràfica amb l'evolució de les mesures, haurem d'afegir una "Visualització". Un cop creat, podem editar-lo clicant sobre el llapis, que es troba a la cantonada dreta superior, i configurar-lo a gust de l'usuari. Cal remarcar que tan sols pot haver una gràfica per a cada camp ("Field").

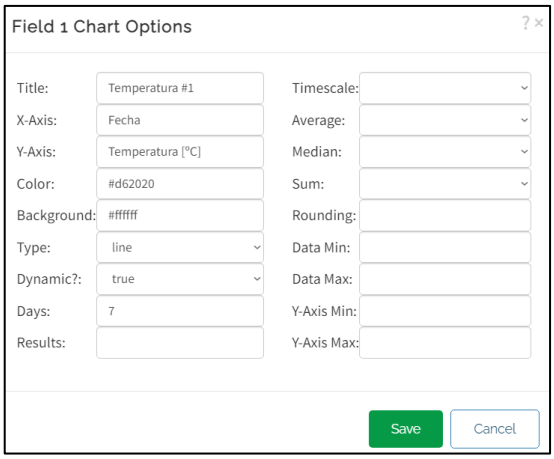

Figura 22. Exemple de configuració d'una gràfica per a la temperatura

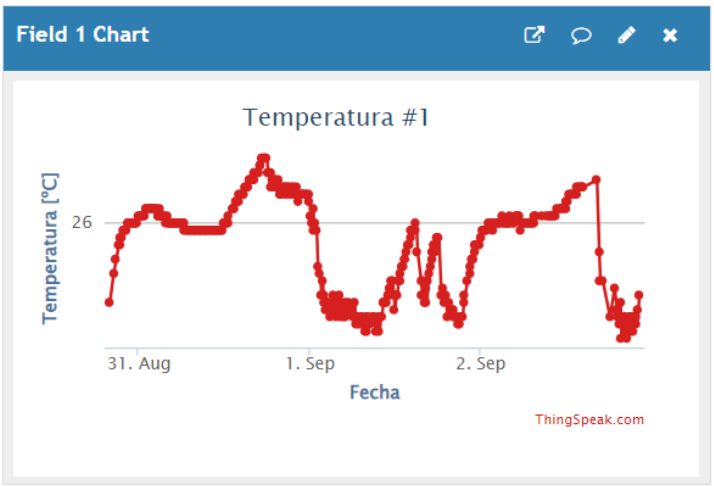

Figura 23. Exemple de visualització de les dades

Les gràfiques són útils per a poder visualitzar l'evolució al llarga del temps d'una magnitud, però a vegades ens pot ser interessant conèixer tan sols la darrera lectura, i ThingSpeak ens proporciona unes quantes eines per a realitzar-ho de forma atractiva visualment. Aquestes eines són els "Widgets" i poden seleccionar entre diferents dissenys. Per a la temperatura i la humitat el disseny més interessant és el "Gauge", ja que, com es pot veure en la següent

figura, podrem definir diferents colors per a diferents marges de lectura. Així, segons els valors límits de l'estat de salut dels ruscs que hem vist en l'apartat 2 d'aquest projecte, podem configurar-ho per no haver de recordar aquests valors i poder guiar-nos pel codi de color. Pel "Widget" es obligatori seleccionar el camp que es vol representar.

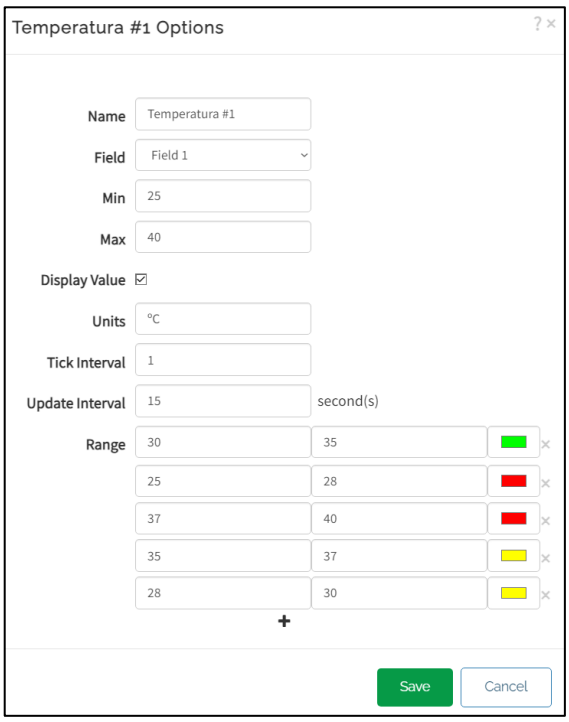

Figura 24. Exemple de configuració d'un "Widget" per a la temperatura

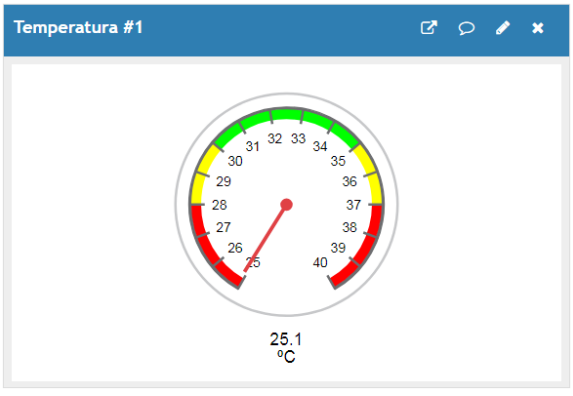

Figura 25. Exemple de visualització del "Widget" per a la temperatura

El "Widget" ideal per a la humitat segueix la mateixa configuració que el de temperatura, però modificar els valors i marges admissibles. Per el pes el més representatiu seria el display numèric.

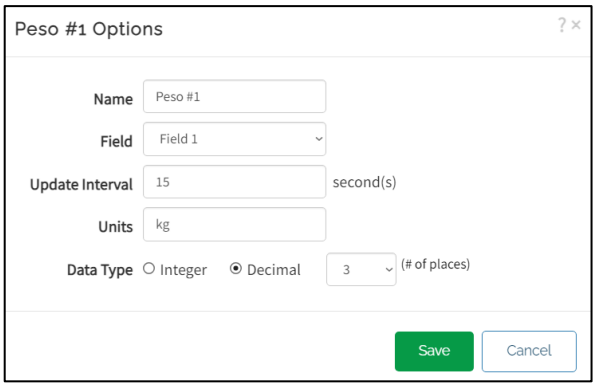

Figura 26. Exemple de configuració del display numèric per al pes

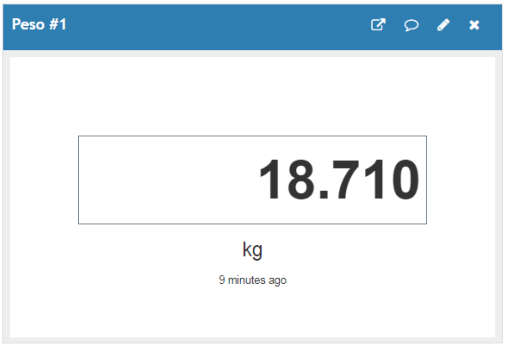

Figura 27. Visualització del display numèric per al pes

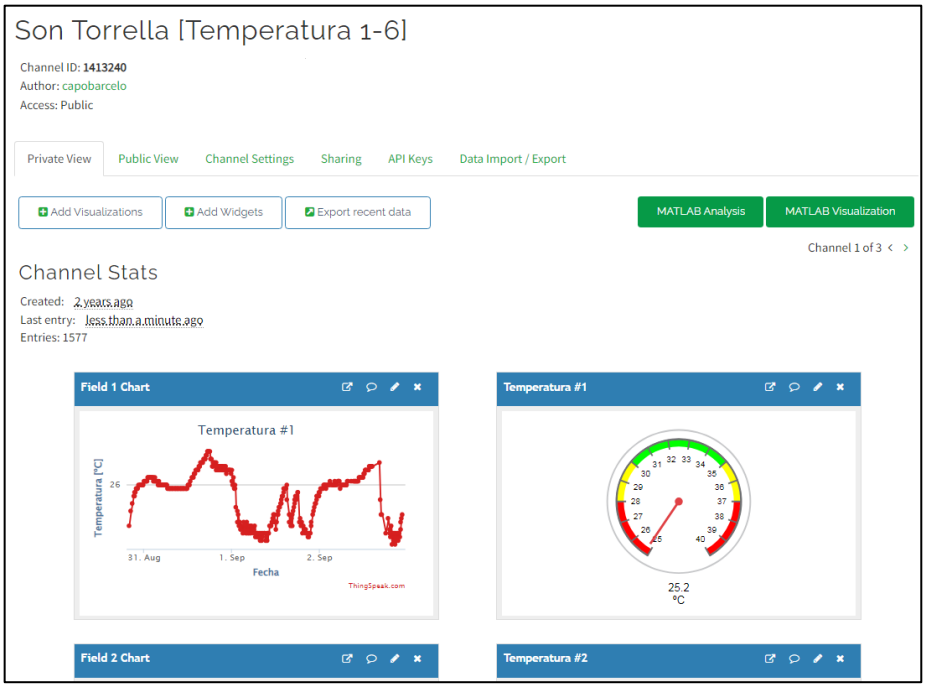

Figura 28. Exemple de pantalla d'inic per a un canal de temperatura

En aquest punt ja tenim tots els nostres canals perfectament configurats i a punt per a començar a rebre informació. Per a que les dades puguin ser enviades correctament als nostres canals, ens farà falta saber dos codis per a cada un d'ells. El primer codi és el número identificador del canal, aquest es troba situat a la part superior un cop estam a dintre d'un (a la Figura 28 es pot identificar com a "Channel ID"). L'altre codi és la clau API d'escriptura, que es pot visualitzar en l'accés directe "API Keys" que hi ha a dintre de cada canal. La clau API d'escriptura, o "Write API Key", és un codi d'autentificació que s'utilitza per a permetre l'accés i la comunicació segura entre una aplicació o dispositiu i la plataforma ThingSpeak. Aquestes claus s'utilitzen per a garantir que tan sols usuaris autoritzats i dispositius tinguin permís per a escriure valors als diferents canals.

6. PROGRAMARI

El sistema operatiu en el qual s'executa tot el programari del dispositiu és Raspberry Pi OS Lite. Aquest sistema operatiu és una versió més lleugera que el sistema operatiu principal de Raspberry Pi, ja que no compta amb un entorn d'escriptori, però això no ens afecta perquè en cap moment s'ha d'interactuar amb el dispositiu a través d'un monitor.

Amb el sistema operatiu totalment net, s'han instal·lat diferents programaris essencials pel funcionament de les diferents tasques Aquests programaris són "dnsmasq", "hostapd", "python3-pip", "sqlite3", "gunicorn", "nginx", "flask", "flask-sqalchemy", "Adafruit_DHT", "HX711", "thingspeak" i el programari de Witty Pi per a poder controlar el seu mòdul (les instruccions d'instal·lació es troben en el full tècnic de la Witty Pi 3 Mini).

La interfície 1-Wire es troba de forma predeterminada en el GPIO-4, pin que ocupa el mòdul Witty Pi 3 Mini i pot crear conflicte si es deixa d'aquesta forma. A part, aquesta interfície ha d'estar ocupada pels sensors DS18B20. Per a aquest motiu, en l'arxiu "config.txt" de la carpeta "/etc" s'ha d'afegir la comanda "dtoverlay=w1-gpio-pullup,gpiopin=27,pullup=on" per a poder definir i activar la interfície 1-Wire al GPIO-27, on es troben connectats tots els sensors de temperatura.

6.1. Funcionament bàsic del sistema

La Raspberry Pi s'engega automàticament quan el mòdul Witty Pi 3 Mini s'alimenta per primer cop o quan aquest li dona l'ordre a la Raspberry Pi en base a l'horari amb el qual s'ha programat. Durant la inicialització, el LED romana encès de forma constant. Quan la Raspberry Pi ha acabat el procés d'inicialització, el servici principal que controla el funcionament del sistema s'anomena "ToggleMode". Aquest servici ha estat creat per a que funcioni de la següent forma. El primer pas essencial que realitza és esborrar els registres de les mesures realitzades en l'anterior engegada (si n'hi ha). Un cop finalitzada la neteja, crida al servici "MeasurementMode" per a que s'activi. Seguidament s'afegeix una detecció per flanc de baixada al pin on hi tenim connectat el botó. Finalment amb el LED es crea una intermitència ràpida de 0,2 segons al 50% per indicar-nos que el sistema ha acabat de posar-se en punt, que s'ha iniciat el procés de lectura dels sensors i que si es vol canviar a l'estat de configuració es pot detectar amb l'activació del botó.

Si el botó no es pren, amb el temps tindrem una ordre de suspensió per part del servici "MeasurementMode", indicant al sistema que ja s'han realitzat totes les mesures i no cal seguir gastant recursos, o per part del mòdul Witty Pi, fent saber al sistema que l'usuari, amb l'horari que ha configurat, vol que el sistema comenci el procés de suspensió independentment de l'estat de les mesures. Si el botó es pren en qualque moment del mode de mesura, automàticament el LED torna a estar il·luminat de forma constant, es dona l'ordre d'aturar el servici "MeasurementMode" i iniciar el servicis "hostapd", "dnsmasq", "nginx" i "SetupInterface". Un cop aquests servicis han estat completament inicialitzat, el LED passarà a tenir una intermitència lenta de 0,8 segons al 50% per indicar-nos que el canvi de mode s'ha completat i ara ens trobem en el mode configuració. En el mode configuració no hi haurà cap ordre automàtica de suspensió, per tant podem estar tot el temps que vulguem en aquest estat.

Si es torna a prémer el botó, el LED tornarà a estar il·luminat de forma constant, es dona l'ordre d'aturar els servicis "hostapd", "dnsmasq", "nginx" i "SetupInterface" i inicialitzar el servici "MeasurementMode". Un cop "MeasurementMode" ja es troba en execució, el LED tornarà a emetre una intermitència ràpida, indicant-nos que el sistema es troba en mode de mesura i que, un cop finalitzades o per degut a l'horari, el sistema s'apagarà automàticament.

El servici "MeasurementMode" revisa una base de dades per a comprovar quins sensors es volen mesurar, i en cas de ser afirmatiu, executa l'script adequat al sensor ("MeasureAM2302.py", "MeasureDS18B20.py" o "MeasureHX711.py") per a realitzar una lectura correcta. Seguidament guarda la mesura realitzada en la base de dades, i al final envia el valor al canal i camp de ThingSpeak que s'hagi configurat per a aquest sensor. Un cop està tot revisat, envia una ordre de suspensió amb un retard d'un minut i seguidament actualitza l'horari del mòdul Witty Pi.

Tota la informació variable s'emmagatzema en una base de dades SQL que conté 6 taules. Aquesta base de dades pot ser modificada tan en el mode de mesura com en el mode de configuració. La següent taula mostra l'estructura i la informació que s'emmagatzema en aquesta base de dades.

Taula	Dada	Tipus
ThingSpeak	Número del canal	Integer
	Nom del canal	Text
	Identificador del canal	Text
	Clau d'escriptura del canal	Text
Pins GPIO disponibles	GPIO	Integer
	Disponibilitat?	Integer
Humitat	Rusc	Integer
	Habilitat?	Integer
	GPIO	Integer
	Número del canal de ThingSpeak	Integer
	Camp del canal de ThingSpeak	Integer
	Offset	Real
	Darrera mesura	Real
	Mesura enviada?	Integer
Temperatura	Rusc	Integer
	Habilitat?	Integer
	Identificador	Text
	Número del canal de ThingSpeak	Integer
	Camp del canal de ThingSpeak	Integer
	Offset	Real
	Darrera mesura	Real
	Mesura enviada?	Integer
1-Wire	Primary Key	Integer
	Identificador del sensor	Text
Pes	Rusc	Integer
	Habilitat?	Integer
	GPIO DT	Integer
	GPIO SCK	Integer
	Canal HX711	Text
	Número del canal de ThingSpeak	Integer
	Camp del canal de ThingSpeak	Integer
	Offset	Real
	Escala	Real
	Identificador temperatura exterior	Text
	Offset temperatura	Real
	Temperatura referència	Real
	Factor de correcció per temperatura	Real
	Darrera mesura	Real
	Mesura enviada?	Integer

Taula 4. Configuració de la base de dades

El servici "hostapd" s'utilitza per a crear un punt d'accés Wi-Fi amb la Raspberry Pi, que en aquest cas tindrà el nom de "SweetHive" i una contrasenya igual a "SweetHive!". El servici "dnsmasq" és un servidor DNS lleuger i un servidor de servei DHCP que ens serveix per a

assignar direccions IP als dispositius que es connectin al punt d'accés (en aquest cas en un rang de 192.168.4.2 a 192.168.4.20 amb un temps de concessió de 1 hora) i per a assignar la direcció 192.168.4.1 a la pròpia Raspberry Pi. El servici "SetupInterface" genera un servidor HTTP lligat a una aplicació mitjançant el protocol "socket Unix" mitjançant "gunicorn". I finalment el servici "ngnix" en serveix per a redirigir les sol·licituds HTTP que arriben al port 80 del nostre servidor (192.168.4.1) al l'aplicació web que està executant "gunicorn" a través de "socket Unix". D'aquesta forma tenim una aplicació web allotjada a la direcció IP desitjada fixe.

6.2. Aplicació Web

Un cop que ens hem connectat al punt d'accés "SweetHive" i en una finestra de qualsevol navegador ens hem dirigit a la direcció "192.168.4.1" hauríem de poder visualitzar l'aplicació Web. Aquesta aplicació web ha estat creada a partir de "Flask", una eina que serveix per a desenvolupar aplicacions web ràpides i lleugeres amb Python. És un framework simple, flexible i fàcil d'aprendre que té integració amb bases de dades SQL, funció que ens interessa perquè és aquí on es guarden tots els paràmetres que es poden modificar en la web i que després serviran per a ajustar el comportament dels scripts en el mode de mesura.

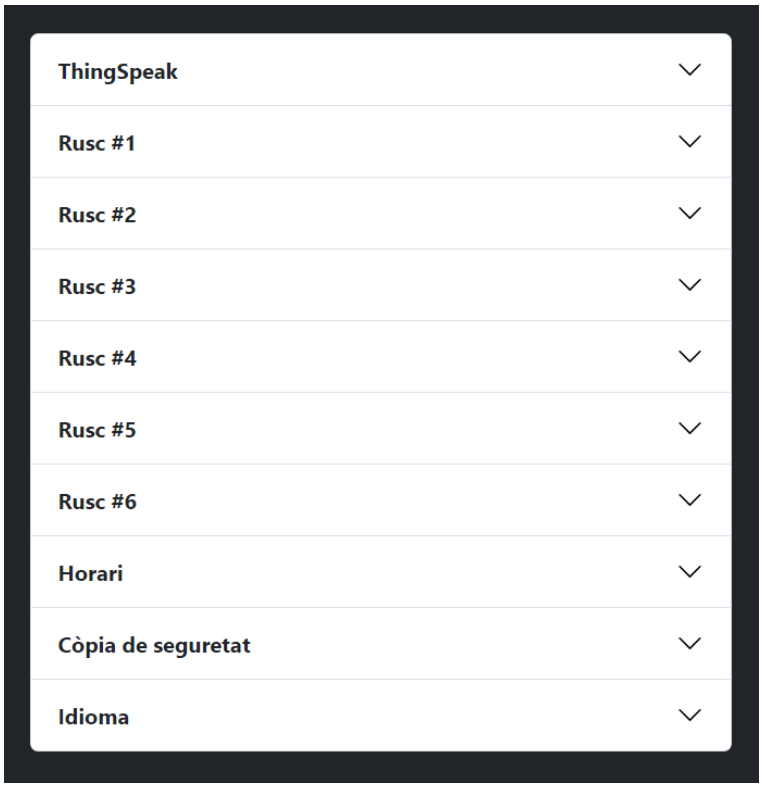

Figura 29. Vista principal de l'aplicació web

Per a agilitzar el disseny de la web i tenir una web receptiva (ajust automàtic del disseny i text en funció de la mida de la pantalla) s'ha utilitzat "Bootstrap", un framework de disseny de webs de codi obert que proporciona compatibilitat amb múltiples navegadors

6.2.1. ThingSpeak

La vista principal de la web és una llista de desplegables que ens ofereixen una visió ràpida de tot al que tenim accés. En el primer desplegable titulat "ThingSpeak" podrem afegir o eliminar els canals a on volem que vagin les nostres lectures. Es poden tenir fins a 20 canals emmagatzemats, i per a cada un cal indicar el nom del mateix (no té perquè coincidir amb el nom real del canal, però ajuda a identificar l'ús del canal), l'identificador del canal i la clau API d'escriptura. Per a l'identificador i la clau és necessari introduir la informació amb el format correcte, sinó la web no deixa guardar el canal.

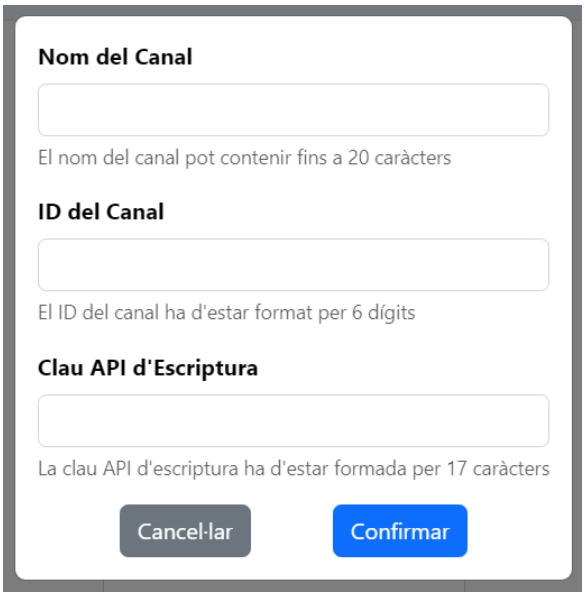

Figura 30. Desplegable per a introduir la informació d'un canal de ThingSpeak

6.2.2. Ruscs

Els sis següents desplegables corresponen a cada un dels ruscs que es poden monitoritzar. A dintre de cada un d'ells tindrem una secció per a habilitar, configurar i realitzar una mesura per cada un dels sensors. En el sensor de temperatura DS18B20 podrem buscar i seleccionar l'identificador del sensor, ajustar l'offset i indicar el canal i camp a on han de ser enviades les lectures d'aquest sensor. En el sensor d'humitat DHT22 podrem seleccionar el seu pin GPIO,

ajustar l'offset i indicar el canal i camp a on han de ser enviades les lectures realitzades. Finalment amb el convertidor HX711 és on tenim més paràmetres de configuració. Podem indicar el pin GPIO per als ports DT i SCK, el canal del que fa ús el convertidor (A o B), evidentment el canal i camp que volem a "ThingSpeak", l'offset (que es igual al valor que mesuren les cèl·lules de càrrega quan es troben sense càrrega), l'escala (que es calcula realitzant una mesura amb un pes conegut i dividint el valor mesurat entre el pes real) i diferents paràmetres per a compensar l'efecte de la temperatura en les cèl·lules de càrrega. En aquests paràmetres es selecciona l'identificador del DS18B20 que es troba en l'exterior, després s'indica la compensació del DS18B20 i la temperatura de referència en la qual s'ha calibrat el HX711 i l'efecte de la temperatura s'anul·la. Per a saber la correcta correcció g/ºC s'han de realitzar diferents lectures amb un pes conegut a diferents temperatures exteriors.

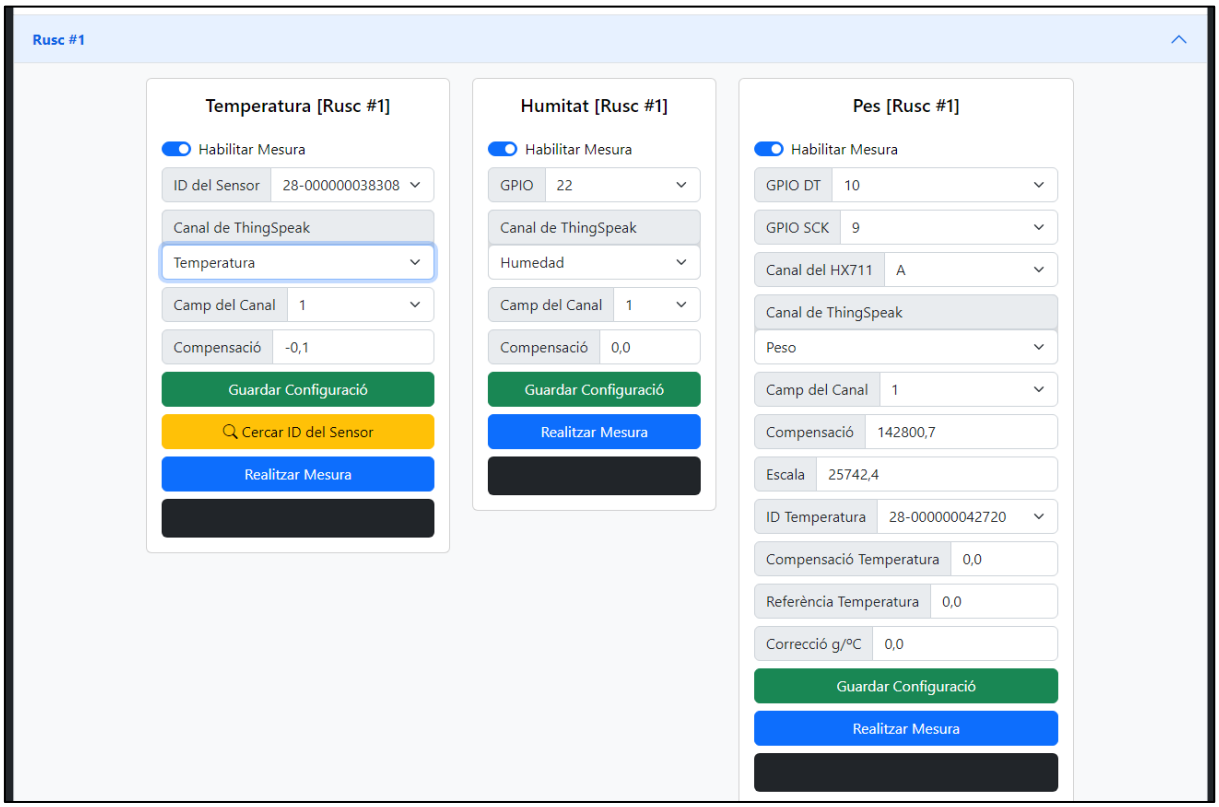

Figura 31. Exemple del desplegable Rusc #1 amb tots els sensors configurats

Tots els camps es troben limitats, tant amb longitud com en format, per a que l'usuari no pugui introduir cap valor ocasioni un malbé a l'aplicació.

6.2.3. Horari, còpia de seguretat i idioma

En el antepenúltim desplegable podrem modificar l'horari d'inici i suspensió del sistema. Hi ha un horari predeterminat d'un cicle de 10 minuts amb 5 minuts encès i els altres 5 apagat. Aquest horari es pot configurar seguint les instruccions del full tècnic del Witty Pi 3 Mini o dirigint-nos a la web "http://www.uugear.com/app/wittypi-scriptgen/" que ens proporciona el fabricant i es un editor d'horaris amb més exemples predeterminats. Una restricció que no s'indica en la documentació és que no pot haver una diferència major de 8 anys entre la data d'inici i de finalització de l'horari. Un cop modificat, es pot guardar, però si abans de tocar res es vol tenir una còpia de l'horari antic, el polsador blau permet descarregar l'horari com un fitxer. Els polsadors negres permet cercar un arxiu compatible al nostre dispositiu i carregar-lo a l'editor. Per motius de seguretat, tan sols es permeten pujar arxius amb format "wpi" i que no siguin excessivament pesats.

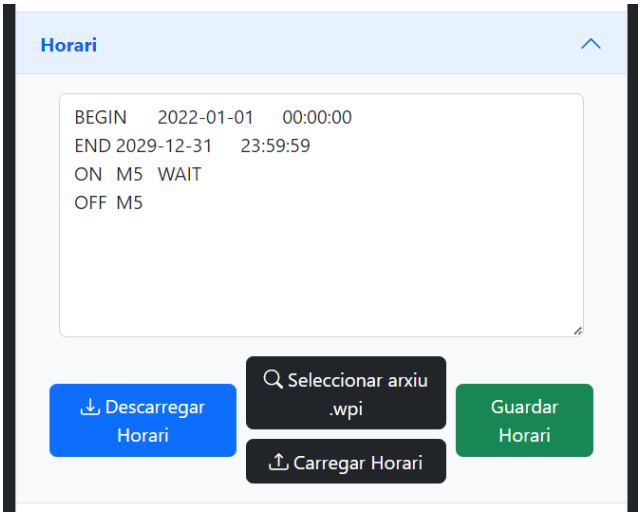

Figura 32. Desplegable amb l'horari programat del sistema

En el desplegable de la còpia de seguretat simplement podem descarregar la base de dades que conté de tots els paràmetres que es troben en l'aplicació web, d'aquesta forma podem tenir una còpia de seguretat si en qualque moment s'han de fer canvis que podríem voler revertir tan sols seleccionant el mateix arxiu i carregant-lo a l'aplicació. El cercador tan sols deixarà seleccionar arxius en format "db" i que no siguin excessivament pesats.

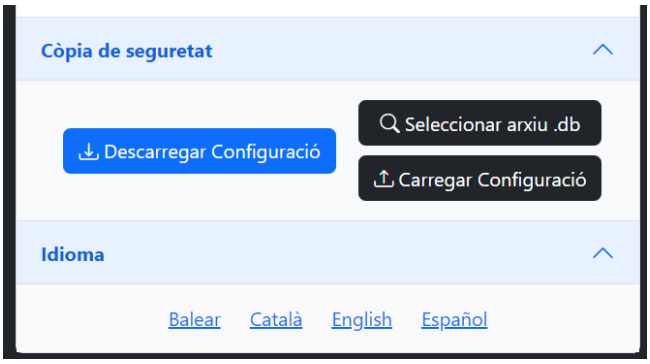

Figura 33. Darrers desplegables de l'aplicació

Finalment al darrer desplegable podrem seleccionar l'idioma de la interfície polsant sobre l'idioma desitjat i amb la recàrrega automàtica de la web aquesta ja es podrà llegir en l'idioma seleccionat.

7. RESUM DEL PRESSUPOST

Els costos totals per l'adquisició dels elements que componen el sistema de monitorització així com les hores de treball destinades al mateix, ascendeixen a un import de sis-cents quaranta-dos euros amb onze cèntims , sense IVA.

8. CONCLUSIONS

Aquest projecte té la funció de projecte tècnic, en el qual es presenta la informació referent a les característiques tècniques de disseny del hardware i programari d'un sistema de monitorització de ruscs.

Un cop finalitzat el projecte es pot dir que s'han pogut complir tots els objectius exposats en la introducció. S'ha aconseguit monitoritzar, amb un mateix sistema, la temperatura, humitat i pes de 6 ruscs diferents. L'històric d'aquestes dades es troben emmagatzemades a la plataforma ThingSpeak, on poden ser consultades i analitzades en qualsevol moment de forma remota. El sistema d'alimentació es totalment independent a la xarxa elèctrica gràcies a l'energia solar, i en el cas d'absència total de llum, permetrà mantenir el funcionament durant 4 dies en les condicions més demandants. El sistema per a 6 ruscs suposa un cost aproximat de 600€, lo que es tradueix amb en un cost aproximat de 100€ per a la monitorització de cada rusc de l'apiari. La relació més econòmica dels productes que es troben en el mercat es troba entre els 250 i 300€ per a cada rusc, i tan sols oferint la monitorització del pes. Per tant, podem assegurar que s'ha pogut crear un dispositiu més econòmic que les solucions actuals i amb millors prestacions.

El prototip encara no s'ha provat en un rusc real, degut a això el dispositiu podria sofrir canvis un cop hagi superat el temps de prova i s'analitzin els aspectes que requereixen ser millorats.

CAPO BARCELO, ANTONI (AUTENTICACIÓN)

Firmado digitalmente por CAPO BARCELO, ANTONI (AUTENTICACIÓN)
Fecha: 2023.09.04 15:02:15 +02'00'

Antoni Capó Barceló

Graduat en Enginyeria Electrònica, Industrial i Automàtica

Santa Maria del Camí, 4 de setembre de 2023

9. RELACIÓ DE DOCUMENTS

Aquest projecte està constituït pels cinc documents següents: memòria, plànols, plec de condicions, estat d'amidaments i pressupost.

10. BIBLIOGRAFIA

ANEL. Tenda especialitzada en la venta de productes apícoles (https://www.anel.gr, 10 d'agost de 2023)

BOOTSTRAP. Framework per a disseny de pàgines webs responsives (https://getbootstrap.com/, 3 d'abril de 2023)

LA TIENDA DEL APICULTOR. Tenda especialitzada en la venta de productes apícoles (https://www.latiendadelapicultor.com/ , 10 d'agost de 2023)

RASPBERRY PI. Empresa fabricant dels microcomputadors Raspberry Pi . (https://www.raspberrypi.com/, 15 d'agost de 2023)

THINGSPEAK. Plataforma oberta que pensada per a emmagatzemar i analitzar dades de sistemes d'IoT (https://thingspeak.com/, 19 d'agost de 2023)

THINGVERSE. Plataforma web on usuaris poden compartir models 3d per a poder ser impresos (https://www.thingiverse.com/, 12 d'agost de 2023)

UUGEAR. Empresa que desenvolupa dispositius per a les plataformes Raspberry Pi i Arduino (https://www.uugear.com/, 20 d'agost de 2023)

11. GLOSSARI

AP: Punt d'accés

API: Interfície de programació d'aplicacions

DHCP: Protocol de configuració dinàmica de host

GND: Terra

GPIO: Entrades i sortides de propòsit general

IoT: Internet de les coses

IP: Protocol d'Internet

LED: Díode emissor de llum

OTG: Sobre la marxa

SIM: Mòdul d'identificació de l'abonat

USB: Bus sèrie universal

VCC: Voltatge comú col·lector

A. CODI INFORMÀTIC

En aquest annex trobarem el codi informàtic que s'ha utilitzat en el sistema. Hi ha alguns arxius que són generats pel sistema operatiu y les llibreries als quals ha estat necessari modificar algun paràmetre, per a bloquejar o desbloquejar alguna funcionalitat. Aquests arxius també es troben inclosos en aquest annex, si hi qualque arxiu que no està reflectit en aquest projecte significa que s'ha conservat el format original.

A.1. Configuració del sistema

A la carpeta "boot" de la Raspberry Pi, s'emmagatzema un arxiu anomenat "config.txt". Aquest arxiu és una part important de la configuració del sistema i es fa servir per definir paràmetres i opcions que afecten el comportament de la Raspberry Pi durant el procés d'inicialització.

```
# For more options and information see

# http://rpf.io/configtxt

# Some settings may impact device functionality. See link above for details



# uncomment if you get no picture on HDMI for a default "safe" mode

#hdmi_safe=1



# uncomment the following to adjust overscan. Use positive numbers if console

# goes off screen, and negative if there is too much border

#overscan_left=16

#overscan_right=16

#overscan_top=16

#overscan_bottom=16
```

```
# uncomment to force a console size. By default it will be display's size minus

# overscan.

#framebuffer_width=1280

#framebuffer_height=720



# uncomment if hdmi display is not detected and composite is being output

#hdmi_force_hotplug=1



# uncomment to force a specific HDMI mode (this will force VGA)

#hdmi_group=1

#hdmi_mode=1



# uncomment to force a HDMI mode rather than DVI. This can make audio work in

# DMT (computer monitor) modes

#hdmi_drive=2



# uncomment to increase signal to HDMI, if you have interference, blanking, or

# no display

#config_hdmi_boost=4



# uncomment for composite PAL
```

```
#sdtv_mode=2



#uncomment to overclock the arm. 700 MHz is the default.

#arm_freq=800



# Uncomment some or all of these to enable the optional hardware interfaces

#dtparam=i2c_arm=on

#dtparam=i2s=on

#dtparam=spi=on



# Uncomment this to enable infrared communication.

#dtoverlay=gpio-ir,gpio_pin=17

#dtoverlay=gpio-ir-tx,gpio_pin=18



# Additional overlays and parameters are documented /boot/overlays/README



# Enable audio (loads snd_bcm2835)

dtparam=audio=on



# Automatically load overlays for detected cameras

camera_auto_detect=1
```

```
# Automatically load overlays for detected DSI displays

display_auto_detect=1



# Enable DRM VC4 V3D driver

dtoverlay=vc4-kms-v3d

max_framebuffers=2



# Disable compensation for displays with overscan

disable_overscan=1



[cm4]

# Enable host mode on the 2711 built-in XHCI USB controller.

# This line should be removed if the legacy DWC2 controller is required

# (e.g. for USB device mode) or if USB support is not required.

otg_mode=1



[all]



[pi4]

# Run as fast as firmware / board allows

arm_boost=1
```

```
[all]

dtparam=i2c1=on

dtparam=i2c_arm=on

dtoverlay=pi3-miniuart-bt

dtoverlay=miniuart-bt

core_freq=250

dtparam=i2c_arm=on

dtoverlay=w1-gpio-pullup,gpiopin=27,pullup=on

enable_uart=1
```

A.2. Carpeta '/etc'

La carpeta /etc en una Raspberry Pi (o en cualquier sistema Linux) és una ubicació crucial que conté arxius de configuració que defineixen el comportament i la configuració de diversos aspectes del sistema operatiu. Aquesta carpeta emmagatzema una gran varietat d'arxius de configuració per a diferents programes i serveis instal·lats en la Raspberry Pi.

A.2.1. Arxiu '/etc/default/hostapd'

```
# Defaults for hostapd initscript

#

# WARNING: The DAEMON_CONF setting has been deprecated and will be removed

#          in future package releases.

#

# See /usr/share/doc/hostapd/README.Debian for information about alternative

# methods of managing hostapd.
```

```
#

# Uncomment and set DAEMON_CONF to the absolute path of a hostapd configuration

# file and hostapd will be started during system boot. An example configuration

# file can be found at /usr/share/doc/hostapd/examples/hostapd.conf.gz

#

DAEMON_CONF="/etc/hostapd/hostapd.conf"



# Additional daemon options to be appended to hostapd command:-

#      -d   show more debug messages (-dd for even more)

#      -K   include key data in debug messages

#      -t   include timestamps in some debug messages

#

# Note that -B (daemon mode) and -P (pidfile) options are automatically

# configured by the init.d script and must not be added to DAEMON_OPTS.

#

#DAEMON_OPTS=""
```

A.2.2. Arxiu ‘/etc/hostapd/hostapd.conf’

```
interface=wlan0

hw_mode=g

channel=10

ieee80211d=1

country_code=ES
```

```
ieee80211n=1

wmm_enabled=1

ssid=SweetHive

auth_algs=1

wpa=2

wpa_key_mgmt=WPA-PSK

rsn_pairwise=CCMP

wpa_passphrase=SweetHive!
```

A.2.3. Arxiu '/etc/nginx/sites-available/SetupInterface'

```
server {

       listen 80;



       location / {

              include proxy_params;

              proxy_pass
http://unix:/home/SweetHive/SetupInterface/SetupInterface.sock;

       }

}
```

A.2.4. Arxiu '/etc/systemd/system/MeasurementMode.service'

```
[Unit]

Description = Check all enabled sensors, take a measure and upload them to ThingSpeak

After = multi-user.target
```

```
[Service]

User = root

ExecStart = /usr/bin/python3 /home/SweetHive/Services/MeasurementMode.py



[Install]

WantedBy = multi-user.target
```

A.2.5. Arxiu '/etc/systemd/system/SetupInterface.service'

```
[Unit]

Description = Gunicorn instance to serve Setup Interface

After = network.target



[Service]

User = root

WorkingDirectory = /home/SweetHive/SetupInterface

ExecStart = /usr/bin/gunicorn --workers 1 --bind unix:SetupInterface.sock
SetupInterface:app



[Install]

WantedBy = multi-user.target
```

A.2.6. Arxiu '/etc/systemd/system/ToggleMode.service'

```
[Unit]
```

```
Description = Toggle between Measurement Mode and Setup Mode

After = multi-user.target



[Service]

User = root

ExecStart = /usr/bin/python3 /home/SweetHive/Services/ToggleMode.py



[Install]

WantedBy = multi-user.target
```

A.2.7. Arxiu '/etc/dhcpcd.conf'

```
# A sample configuration for dhcpcd.

# See dhcpcd.conf(5) for details.



# Allow users of this group to interact with dhcpcd via the control socket.

#controlgroup wheel



# Inform the DHCP server of our hostname for DDNS.

hostname



# Use the hardware address of the interface for the Client ID.

clientid

# or
```

```
# Use the same DUID + IAID as set in DHCPv6 for DHCPv4 ClientID as per RFC4361.

# Some non-RFC compliant DHCP servers do not reply with this set.

# In this case, comment out duid and enable clientid above.

#duid



# Persist interface configuration when dhcpcd exits.

persistent



# Rapid commit support.

# Safe to enable by default because it requires the equivalent option set

# on the server to actually work.

option rapid_commit



# A list of options to request from the DHCP server.

option domain_name_servers, domain_name, domain_search, host_name

option classless_static_routes

# Respect the network MTU. This is applied to DHCP routes.

option interface_mtu



# Most distributions have NTP support.

#option ntp_servers
```

```
# A ServerID is required by RFC2131.

require dhcp_server_identifier



# Generate SLAAC address using the Hardware Address of the interface

#slaac hwaddr

# OR generate Stable Private IPv6 Addresses based from the DUID

slaac private



# Example static IP configuration:

#interface eth0

#static ip_address=192.168.0.10/24

#static ip6_address=fd51:42f8:caae:d92e::ff/64

#static routers=192.168.0.1

#static domain_name_servers=192.168.0.1 8.8.8.8 fd51:42f8:caae:d92e::1



# It is possible to fall back to a static IP if DHCP fails:

# define static profile

#profile static_eth0

#static ip_address=192.168.1.23/24

#static routers=192.168.1.1

#static domain_name_servers=192.168.1.1
```

```
# fallback to static profile on eth0

#interface eth0

#fallback static_eth0



interface wlan0

       static ip_address=192.168.4.1/24
```

A.2.8. Arxiu '/etc/default/dnsmasq.conf'

```
interface=wlan0

dhcp-range=192.168.4.2,192.168.4.20,255.255.255.0,1h

dhcp-option=3,192.168.4.1

dhcp-option=6,192.168.4.1
```

A.3. Carpeta '/home/SweetHive/MeasurementScripts'

Aquesta carpeta conté cada un dels scripts que s'utilitzen per a comunicar-se amb els sensors del sistema, realitzar la mesura i enviar la informació a ThingSpeak

A.3.1. Arxiu '/home/SweetHive/MeasurementScripts/MeasureAM2302.py'

```
#!usr/bin/env python3



import Adafruit_DHT

import sqlite3

import sys

from os.path import dirname
```

```
DbDirectory = dirname(__file__).split('/M')[0] +
'/SetupInterface/database/Settings.db'

AM2302Sensor = Adafruit_DHT.AM2302

db = sqlite3.connect(DbDirectory)

Hive = int(sys.argv[1])

GPIO = db.execute("SELECT GPIO FROM Humidity WHERE Hive == ?", (Hive, )).fetchone()

if GPIO[0]:

	try:

		Humidity, Temperature = Adafruit_DHT.read_retry(AM2302Sensor, GPIO[0])

		ReadingError = 0

	except:

		print("Reading error for AM2302 from hive number", Hive)

		ReadingError = 1

	if Humidity:

		Offset = db.execute("SELECT Offset FROM Humidity WHERE Hive == ?",

							(Hive, )).fetchone()

		if Offset[0]:

			Humidity = Humidity - float(Offset[0])

		Humidity = round(Humidity, 1)

		db.execute("UPDATE Humidity SET Measurement = ? WHERE Hive = ?",

							(Humidity, Hive))

		db.commit()
```

```
            print("Humidity inside hive number", str(Hive) + ":", Humidity, "%")
      elif not ReadingError:
            print("Unable to read AM2302 from hive number", Hive)
else:
      print("AM2302 from hive number", Hive, "has not assigned GPIO")
db.close()
```

A.3.2. Arxiu '/home/SweetHive/MeasurementScripts/MeasureDS18B20.py'

```
#!usr/bin/env python3


import sqlite3
import sys
from os.path import dirname


DbDirectory             =             dirname(__file__).split('/M')[0]             +
'/SetupInterface/database/Settings.db'
OneWireDirectory = '/sys/bus/w1/devices/'
db = sqlite3.connect(DbDirectory)
Hive = int(sys.argv[1])
ID = db.execute("SELECT ID FROM Temperature WHERE Hive == ?", (Hive, )).fetchone()
if ID[0]:
      try:
            Status = 'Reading'
```

```
        SensorFile = open(OneWireDirectory + ID[0] + '/w1_slave', 'r')
        SensorLines = SensorFile.readlines()
        SensorFile.close()
        Status = SensorLines[0].strip()[-3:]
    except:
        print("DS18B20 from hive number", Hive, "was not found")
    if Status == 'YES':
        MeasurePosition = SensorLines[1].find('t=')
        Temp = SensorLines[1][MeasurePosition + 2:]
        Temperature = float(Temp) / 1000
        if Temperature != 85:
            Offset = db.execute("SELECT Offset FROM Temperature WHERE " \
                                "Hive == ?",
                                (Hive, )).fetchone()
            if Offset[0]:
                Temperature = Temperature - float(Offset[0])
            Temperature = round(Temperature, 1)
            db.execute("UPDATE Temperature SET Measurement = ? " \
                                "WHERE Hive = ?",
                                (Temperature, Hive))
            db.commit()
            DegreeSign = u"\N{DEGREE SIGN}"
```

```
                    print("Temperature inside hive number", str(Hive) + ":",
                                                        Temperature,
                                                        DegreeSign + "C")
              else:
                    print("Recieved error 85 for DS18B20 from hive number", Hive)
       elif Status == 'NO':
              print("DS18B20 reading from hive number", Hive, "is not a valid value")
else:
       print("A DS18B20 ID from hive number", Hive, "was not selected")
db.close()
```

A.3.3. Arxiu '/home/SweetHive/MeasurementScripts/MeasureHX711.py'

```
#!usr/bin/env python3



from hx711 import HX711
import sqlite3
import RPi.GPIO as GPIO
import sys
from os.path import dirname


AdmissibleError = 0.05
```

```
DbDirectory = dirname(__file__).split('/M')[0] +
'/SetupInterface/database/Settings.db'

db = sqlite3.connect(DbDirectory)

Hive = int(sys.argv[1])

GPIODT = db.execute("SELECT GPIODT FROM Weight WHERE Hive == ?", (Hive, )).fetchone()

GPIOSCK = db.execute("SELECT GPIOSCK FROM Weight WHERE Hive == ?", (Hive,
)).fetchone()

Channel = db.execute("SELECT Channel FROM Weight WHERE Hive == ?", (Hive,
)).fetchone()

if GPIODT[0] and GPIOSCK[0] and GPIODT[0] != GPIOSCK[0]:

	OneWireDirectory = '/sys/bus/w1/devices/'

	ID = db.execute("SELECT TemperatureID FROM Weight WHERE Hive == ?",

                                                (Hive, )).fetchone()

	if ID[0]:

		try:

			Status = 'Reading'

			SensorFile = open(OneWireDirectory + ID[0] + '/w1_slave', 'r')

			SensorLines = SensorFile.readlines()

			SensorFile.close()

			Status = SensorLines[0].strip()[-3:]

		except:

			Temperature = 85

			print("Temperature Sensor for weight from hive number", Hive,

                                                "was not found")
```

```
if Status == 'YES':
        MeasurePosition = SensorLines[1].find('t=')
        Temp = SensorLines[1][MeasurePosition + 2:]
        Temperature = float(Temp) / 1000
        if Temperature != 85:
                Offset = db.execute("SELECT TemperatureOffset FROM " \
                                    "Weight WHERE " \
                                    "Hive == ?",
                                    (Hive, )).fetchone()
                Ref = db.execute("SELECT TemperatureReference FROM " \
                                    "Weight WHERE " \
                                    "Hive == ?",
                                    (Hive, )).fetchone()
                Corr = db.execute("SELECT TemperatureCorrection FROM " \
                                    "Weight WHERE " \
                                    "Hive == ?",
                                    (Hive, )).fetchone()
                if Offset[0]:
                        Temperature = Temperature - float(Offset[0])
                if Ref[0]:
                        Temperature = Temperature - float(Ref[0])
                if Corr[0]:
```

```
                        Correction = (Temperature * float(Corr[0]))
                    else:
                        Correction = 0
                else:
                    print("Recieved error 85 for weight from hive number",
                                                Hive)
            elif Status == 'NO':
                Temperature = 85
                print("Temperature reading for weight from hive number", Hive,
                                                "is not a valid value")
        else:
            Temperature = 0
            Correction = 0
            print("A Temperature Sensor for weight from hive number", Hive,
                                                "was not selected")
        try:
            Ready = 0
            ResetSuccess = 0
            hx = HX711(dout_pin = GPIODT[0],
                       pd_sck_pin = GPIOSCK[0],
                       channel = Channel[0])
            hx.power_up()
```

```
        Ready = hx._ready()
        if Ready:
                ResetSuccess = hx.reset()
        ConnectionError = 0
    except:
        print("Connection error for HX711 from hive number", Hive)
        ConnectionError = 1
    if ResetSuccess and Temperature != 85:
        WeightMeasures = hx.get_raw_data(100)
        Readings = len(WeightMeasures)
        SortedMeasures = sorted(WeightMeasures)
        ReliableWeight = SortedMeasures[int(Readings/2)]
        ReliableMeasures = []
        AdmissibleError = ReliableWeight * AdmissibleError
        for Measure in range(Readings):
                Error = SortedMeasures[Measure] - ReliableWeight
                if abs(Error) < abs(AdmissibleError):
                        ReliableMeasures.append(SortedMeasures[Measure])
        if len(ReliableMeasures) > 50:
                Weight = sum(ReliableMeasures)/len(ReliableMeasures)
                Offset = db.execute("SELECT Offset FROM Weight WHERE Hive == ?",
                                                    (Hive, )).fetchone()
```

```
                Scale = db.execute("SELECT Scale FROM Weight WHERE Hive == ?",
                                                (Hive, )).fetchone()
                if Offset[0]:
                        Weight = Weight - float(Offset[0])
                if Scale[0]:
                        Weight = Weight / float(Scale[0])
                Weight = Weight + (Correction / 1000)
                Weight = round(Weight, 2)
                db.execute("UPDATE Weight SET Measurement = ? WHERE Hive = ?",
                                                (Weight, Hive))
                db.commit()
                print("Weight of hive number", str(Hive) + ":", Weight,"kg")
        elif len(ReliableMeasures) < 10:
                print("HX711 from hive number", Hive, "was not found")
        else:
                print("Not enough acceptable HX711 readings from hive number",
                                                Hive)
        hx.power_down()
elif Temperature == 85:
        print("A Temperature Sensor for weight from hive number", Hive,
                                                "was selected, but",
                                                "ended with error")
```

```
        elif not Ready and not ConnectionError:
                print("HX711 from hive number", Hive, "was not found")
                hx.power_down()
        elif not ConnectionError:
                print("HX711 from hive number", Hive, "unable to reset")
                hx.power_down()
        GPIO.cleanup()
elif not GPIODT[0]:
        print("HX711 from hive number", Hive, "has not assigned DT GPIO")
elif not GPIOSCK[0]:
        print("HX711 from hive number", Hive, "has not assigned SCK GPIO")
else:
        print("Hx711 from hive number", Hive, "has DT and SCK assigned on the same
GPIO")
db.close()
```

A.3.4. Arxiu '/home/SweetHive/MeasurementScripts/SearchDS18B20ID.py'

```
#!usr/bin/env python3



import sqlite3
from glob import glob
from os.path import dirname
```

```
DbDirectory             =             dirname(__file__).split('/M')[0]             +
'/SetupInterface/database/Settings.db'

OneWireDirectory = '/sys/bus/w1/devices/'

DevicesDirectories = glob(OneWireDirectory + '28*')

IDs = [ID.split('devices/')[1] for ID in DevicesDirectories]

db = sqlite3.connect(DbDirectory)

db.execute("DELETE FROM W1")

for Sensor in range(len(IDs)):

      db.execute("INSERT INTO W1 VALUES (?, ?)", (Sensor + 1, IDs[Sensor]))

db.commit()

db.close()

print("List of ID from DS18B20 successfully updated")
```

A.4. Carpeta '/home/SweetHive/Services'

Aquesta carpeta conté els executables dels servicis "MeasurementMode" i "ToggleMode".

A.4.1. Arxiu '/home/SweetHive/Services/MeasurementMode.py'

```
#!usr/bin/env python3



import thingspeak

import sqlite3

from os.path import dirname, exists

from subprocess import run
```

```
def ToThingSpeak(Measure, ChannelNumber, ChannelField):
	ID = db.execute("SELECT ChannelID FROM TS WHERE ChannelNumber == ?",
                                                (ChannelNumber, )).fetchone()
	Key = db.execute("SELECT ChannelWriteAPIKey FROM TS WHERE ChannelNumber == ?",
                                                (ChannelNumber, )).fetchone()
	Channel = thingspeak.Channel(id = ID, api_key = Key)
	for Attempt in range(5):
		try:
			Channel.update({ChannelField:Measure})
			print("Measurement uploaded successfully")
			return 0
		except:
			print("Connection failed on attempt", Attempt + 1)
	return 1



if __name__ == '__main__':
	DbDirectory = dirname(__file__).split('/Services')[0] + \
                                        '/SetupInterface/database/Settings.db'
	ScriptsDirectory = dirname(__file__).split('/Services')[0] +
'/MeasurementScripts'
	if exists(DbDirectory):
		db = sqlite3.connect(DbDirectory)
		Hives = db.execute("SELECT Hive FROM Humidity").fetchall()
```

```
Script = ScriptsDirectory + '/MeasureAM2302.py'

for Hive in range(len(Hives)):

        Enabled = db.execute("SELECT Enabled FROM Humidity " \

                                "WHERE Hive == ?",

                                (Hive + 1, )).fetchone()

        Sent = db.execute("SELECT Sent FROM Humidity WHERE Hive == ?",

                                 (Hive + 1, )).fetchone()

        if Enabled[0] and not Sent[0]:

                run(["python", Script, str(Hive + 1)])

                Measure = db.execute("SELECT Measurement FROM " \

                                "Humidity WHERE Hive == ?",

                                 (Hive + 1, )).fetchone()

                if Measure[0]:

                        Channel = db.execute("SELECT " \

                                "ThingSpeakChannelNumber FROM " \

                                "Humidity WHERE Hive == ?",

                                 (Hive + 1, )).fetchone()

                        Field = db.execute("SELECT " \

                                "ThingSpeakChannelField FROM " \

                                "Humidity WHERE Hive == ?",

                                 (Hive + 1, )).fetchone()

                        if Channel[0] and Field[0]:
```

```
                Upload = ToThingSpeak(Measure[0],
                                      Channel[0],
                                      Field[0])
                if Upload == 0:
                        db.execute("UPDATE Humidity " \
                                "SET Sent = ? " \
                                "WHERE Hive = ?",
                                (1, Hive + 1))
                        db.commit()
                        print("Humidity measurement " \
                                "from hive number",
                                Hive + 1, "has been sent")
        elif not Channel[0]:
                print("Humidity sensor " \
                        "from hive number", Hive + 1,
                        "has not ThingSpeak " \
                        "Channel assigned")
        else:
                print("Humidity sensor " \
                        "from hive number", Hive + 1,
                        "has not ThingSpeak " \
                        "Channel Field assigned")
```

```
            elif not Enabled[0]:
                  print("Humidity sensor from hive number",
                                    Hive + 1, "is not enabled")
            else:
                  print("Humidity measurement from hive number",
                                    Hive + 1, "was already sent")
Hives = db.execute("SELECT Hive FROM Temperature").fetchall()
Script = ScriptsDirectory + '/MeasureDS18B20.py'
for Hive in range(len(Hives)):
      Enabled = db.execute("SELECT Enabled FROM Temperature " \
                                    "WHERE Hive == ?",
                                     (Hive + 1, )).fetchone()
      Sent = db.execute("SELECT Sent FROM Temperature WHERE Hive == ?",
                                     (Hive + 1, )).fetchone()
      if Enabled[0] and not Sent[0]:
            run(["python", Script, str(Hive + 1)])
            Measure = db.execute("SELECT Measurement FROM " \
                                    "Temperature WHERE Hive == ?",
                                     (Hive + 1, )).fetchone()
            if Measure[0]:
                  Channel = db.execute("SELECT " \
                                    "ThingSpeakChannelNumber FROM " \
```

```
                "Temperature WHERE Hive == ?",
                 (Hive + 1, )).fetchone()
        Field = db.execute("SELECT " \
                "ThingSpeakChannelField FROM " \
                "Temperature WHERE Hive == ?",
                 (Hive + 1, )).fetchone()
        if Channel[0] and Field[0]:
            Upload = ToThingSpeak(Measure[0],
                                  Channel[0],
                                  Field[0])
            if Upload == 0:
                db.execute("UPDATE Temperature " \
                       "SET Sent = ? " \
                       "WHERE Hive = ?",
                       (1, Hive + 1))
                db.commit()
                print("Temperature measurement " \
                       "from hive number",
                       Hive + 1, "has been sent")
        elif not Channel[0]:
            print("Temperature sensor " \
                  "from hive number", Hive + 1,
```

```
                                        "has not ThingSpeak " \
                                        "Channel assigned")
                            else:
                                    print("Temperature sensor " \
                                        "from hive number", Hive + 1,
                                        "has not ThingSpeak " \
                                        "Channel Field assigned")
                    elif not Enabled[0]:
                            print("Temperature sensor from hive number",
                                        Hive + 1, "is not enabled")
                    else:
                            print("Temperature measurement from hive number",
                                        Hive + 1, "was already sent")
            Hives = db.execute("SELECT Hive FROM Weight").fetchall()
            Script = ScriptsDirectory + '/MeasureHX711.py'
            for Hive in range(len(Hives)):
                    Enabled = db.execute("SELECT Enabled FROM Weight " \
                                        "WHERE Hive == ?",
                                        (Hive + 1, )).fetchone()
                    Sent = db.execute("SELECT Sent FROM Weight WHERE Hive == ?",
                                        (Hive + 1, )).fetchone()
                    if Enabled[0] and not Sent[0]:
```

```
run(["python", Script, str(Hive + 1)])
Measure = db.execute("SELECT Measurement FROM " \
                     "Weight WHERE Hive == ?",
                      (Hive + 1, )).fetchone()
if Measure[0]:
      Channel = db.execute("SELECT " \
                     "ThingSpeakChannelNumber FROM " \
                     "Weight WHERE Hive == ?",
                      (Hive + 1, )).fetchone()
      Field = db.execute("SELECT " \
                     "ThingSpeakChannelField FROM " \
                     "Weight WHERE Hive == ?",
                      (Hive + 1, )).fetchone()
      if Channel[0] and Field[0]:
            Upload = ToThingSpeak(Measure[0],
                                  Channel[0],
                                  Field[0])
            if Upload == 0:
                  db.execute("UPDATE Weight " \
                        "SET Sent = ? " \
                        "WHERE Hive = ?",
                        (1, Hive + 1))
```

```
                                        db.commit()
                                        print("Weight measurement " \
                                              "from hive number ",
                                              Hive + 1, "has been sent")
                                elif not Channel[0]:
                                      print("Weight sensor " \
                                              "from hive number", Hive + 1,
                                              "has not ThingSpeak " \
                                              "Channel assigned")
                                else:
                                      print("Weight sensor " \
                                              "from hive number", Hive + 1,
                                              "has not ThingSpeak " \
                                              "Channel Field assigned")
                        elif not Enabled[0]:
                              print("Weight sensor from hive number",
                                              Hive + 1, "is not enabled")
                        else:
                              print("Weight measurement from hive number",
                                              Hive + 1, "was already sent")
                db.close()
        else:
```

```
            print("Settings database was not found")

      run(["shutdown", "-h", "1"])

      UpdateWittyPi = dirname(__file__).split('Services')[0] +
'/wittypi/runScript.sh'

      run([UpdateWittyPi])
```

A.4.2. Arxiu '/home/SweetHive/Services/ToggleMode.py'

```
#!/usr/bin/env python3



import RPi.GPIO as GPIO

import signal

import sqlite3

from os.path import dirname, exists

from time import sleep

from subprocess import run



LEDGPIO = 7

ButtonGPIO = 8

BlinkTimeMeasurementMode = 0.1

BlinkTimeSetupMode = 0.4

TimeON = BlinkTimeMeasurementMode

TimeOFF = TimeON

BounceTime = 12000
```

```
def CleanExit(sig, frame):
	GPIO.setup(LEDGPIO, GPIO.IN, pull_up_down=GPIO.PUD_UP)
	#run(["ip", "link", "set", "wlan0", "down"])
	#MeasurementMode = run(["systemctl", "is-active", "hostapd"])
	#if MeasurementMode.returncode != 0:
		#run(["wpa_cli", "-p", "/var/run/wpa_supplicant", "terminate"])
	#run(["dhclient", "-r", "wlan0"])


def ButtonPressedCallback(channel):
	global TimeON, TimeOFF
	TimeOFF = 0
	MeasurementMode = run(["systemctl", "is-active", "hostapd"])
	if MeasurementMode.returncode == 0:
		TimeON = BlinkTimeMeasurementMode
		StopSetupMode()
		StartMeasurementMode()
		TimeOFF = TimeON
		print("SweetHive is on Measurement Mode now")
	else:
		TimeON = BlinkTimeSetupMode
		print("Switching to Setup Mode...")
```

```
            StopMeasurementMode()

            StartSetupMode()

            TimeOFF = TimeON

            print("SweetHive is on Setup Mode now")



def StartMeasurementMode():

      #run(["wpa_supplicant", "-i", "wlan0", "-B",

                        #"-c", "/etc/wpa_supplicant/wpa_supplicant.conf"])

      #run(["dhclient", "wlan0"])

      #run(["dhcpcd", "-n", "wlan0"])

      run(["systemctl", "start", "MeasurementMode"])



def StopMeasurementMode():

      run(["systemctl", "stop", "MeasurementMode"])

      run(["shutdown", "-c"])

      #run(["ip", "link", "set", "wlan0", "down"])

      #run(["wpa_cli", "-p", "/var/run/wpa_supplicant", "terminate"])

      #run(["systemctl", "stop", "wpa_supplicant"])

      #run(["dhclient", "-r", "wlan0"])



def StartSetupMode():

      run(["systemctl", "start", "SetupInterface"])
```

```
        run(["systemctl", "start", "hostapd"])
        run(["systemctl", "start", "dnsmasq"])
        run(["systemctl", "start", "nginx"])


def StopSetupMode():
        run(["ip", "link", "set", "wlan0", "down"])
        run(["systemctl", "stop", "nginx"])
        run(["systemctl", "stop", "SetupInterface"])
        run(["systemctl", "stop", "hostapd"])
        run(["systemctl", "stop", "dnsmasq"])


def ResetSensors():
        DbDirectory = dirname(__file__).split('/Services')[0] + \
                                        '/SetupInterface/database/Settings.db'
        if exists(DbDirectory):
                db = sqlite3.connect(DbDirectory)
                Hives = db.execute("SELECT Hive FROM Temperature").fetchall()
                for Hive in range(len(Hives)):
                        db.execute("UPDATE Temperature SET Measurement = ? " \
                                        "WHERE Hive = ?", (None, Hive + 1))
                        db.execute("UPDATE Temperature SET Sent = ? " \
                                        "WHERE Hive = ?", (0, Hive + 1))
```

```
                db.commit()
            Hives = db.execute("SELECT Hive FROM Humidity").fetchall()
            for Hive in range(len(Hives)):
                db.execute("UPDATE Humidity SET Measurement = ? " \
                                "WHERE Hive = ?", (None, Hive + 1))
                db.execute("UPDATE Humidity SET Sent = ? " \
                                "WHERE Hive = ?", (0, Hive + 1))
                db.commit()
            Hives = db.execute("SELECT Hive FROM Weight").fetchall()
            for Hive in range(len(Hives)):
                db.execute("UPDATE Weight SET Measurement = ? " \
                                "WHERE Hive = ?", (None, Hive + 1))
                db.execute("UPDATE Weight SET Sent = ? " \
                                "WHERE Hive = ?", (0, Hive + 1))
                db.commit()
            db.close()
        else:
            print("Settings database was not found")


if __name__ == '__main__':
        for Signal in [signal.SIGTERM, signal.SIGINT]:
            signal.signal(Signal, CleanExit)
```

```
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(ButtonGPIO, GPIO.IN, pull_up_down=GPIO.PUD_UP)
        GPIO.setup(LEDGPIO, GPIO.OUT, initial=GPIO.HIGH)
        StopSetupMode()
        ResetSensors()
        StartMeasurementMode()
        GPIO.add_event_detect(ButtonGPIO, GPIO.FALLING,
                    callback=ButtonPressedCallback, bouncetime=BounceTime)
        print("SweetHive has been fully loaded and ready to work")
        while True:
               GPIO.output(LEDGPIO, GPIO.LOW)
               sleep(TimeOFF)
               GPIO.output(LEDGPIO, GPIO.HIGH)
               sleep(TimeON)
```

A.5. Carpeta '/home/SweetHive/SetupInterface'

Aquesta carpeta conté tot el necessari per a poder recrear la pàgina web, amb la qual, l'usuari pot interactuar i modificar diferents paràmetres del sistema a través d'un dispositiu amb connexió Wi-Fi.

A.5.1. Arxiu '/home/SweetHive/SetupInterface/Language/Balear.json'

```
{
    "Title": {
        "Index": "Configuració"
```

```
    },

    "AccordionHeader": {
        "ThingSpeak": "ThingSpeak",
        "Hive": "Caera #",
        "Schedule": "Horari",
        "SettingsBackup": "Còpia de seguretat",
        "Language": "Idioma"
    },

    "ThingSpeakChannelRegister": {
        "Name": "Nom des Canal",
        "NameHelp": "Es nom des canal pot contenir fins a 20 caràcters",
        "ID": "ID des Canal",
        "IDHelp": "S'ID des canal ha d'estar format per 6 dígits",
        "WriteAPIKey": "Clau API d'Escriptura",
        "WriteAPIKeyHelp": "Sa clau API d'escriptura ha d'estar formada per 17
caràcters"
    },

    "HiveSetupInterface": {
        "Temperature": "Temperatura ",
        "Humidity": "Humitat ",
```

```
        "Weight": "Pes ",
        "SensorEnable": "Habilitar Mesura",
        "GPIO": "GPIO",
        "ThingSpeakChannel": "Canal de ThingSpeak",
        "ThingSpeakField": "Camp des Canal",
        "Offset": "Compensació",
        "Save": "Guardar Configuració",
        "Measure": "Realitzar Mesura",
        "DS18B20ID": "ID des Sensor",
        "DS18B20SearchID": "Cercar ID des Sensor",
        "HX711DT": "GPIO DT",
        "HX711SCK": "GPIO SCK",
        "HX711Channel": "Canal des HX711",
        "HX711Scale": "Escala",
        "HX711TemperatureID": "ID Temperatura",
        "HX711TemperatureOffset": "Compensació Temperatura",
        "HX711TemperatureReference": "Referència Temperatura",
        "HX711TemperatureCorrection": "Correcció g/°C"
    },

    "Schedule": {
        "Save": "Guardar Horari",
```

```
        "Download": "Descarregar Horari",
        "Upload": "Carregar Horari",
        "SelectFile": "Seleccionar arxiu .wpi"
    },

    "SettingsBackup": {
        "Download": "Descarregar Configuració",
        "Upload": "Carregar Configuració",
        "SelectFile": "Seleccionar arxiu .db"
    },

    "Alert": {
        "ThingSpeakChannelRegisterName": {
            "Missing": "És necessari incloure un nom pes canal",
            "TooLong": "Es nom des canal és massa llarg"
        },
        "ThingSpeakChannelRegisterID": {
            "Missing": "És necessari incloure s'ID des canal",
            "TooShort": "S'ID des canal és massa curt",
            "TooLong": "S'ID des canal és massa llarg"
        },
        "ThingSpeakChannelRegisterWriteAPIKey": {
```

```
            "Missing": "És necessari incloure sa clau API d'escriptura",
            "TooShort": "Sa clau API d'escriptura és massa curta",
            "TooLong": "Sa clau API d'escriptura és massa llarga"
        },
        "TemperatureOffset": {
            "BadInput": "Sa compensació ha de ser un nombre real",
            "RangeUnderflow": "Sa compensació mínima permesa per sa temperatura és -
100",
            "RangeOverflow": "Sa compensació màxima permesa per sa temperatura és
100",
            "StepMismatch": "Només es permet un decimal"
        },
        "TemperatureReference": {
            "BadInput": "Sa referència ha de ser un nombre real",
            "RangeUnderflow": "Sa referència mínima permesa per sa temperatura és -
100",
            "RangeOverflow": "Sa referència màxima permesa per sa temperatura és 100",
            "StepMismatch": "Només es permet un decimal"
        },
        "HumidityOffset": {
            "BadInput": "Sa compensació ha de ser un nombre real",
            "RangeUnderflow": "Sa compensació mínima permesa per sa humitat és 0",
            "RangeOverflow": "Sa compensació màxima permesa per sa humitat és 100",
            "StepMismatch": "Només es permet un decimal"
```

```
        },

        "WeightOffset": {

            "BadInput": "Sa compensació ha de ser un nombre real",

            "RangeUnderflow": "Sa compensació mínima permesa per es pes és -
1.000.000",

            "RangeOverflow": "Sa compensació màxima permesa per es pes és 1.000.000",

            "StepMismatch": "Només es permet un decimal"

        },

        "WeightScale": {

            "BadInput": "S'escala ha de ser un nombre real",

            "RangeUnderflow": "S'escala mínima permesa per es pes és -100.000",

            "RangeOverflow": "S'escala màxima permesa per es pes és 100.000",

            "StepMismatch": "Només es permet un decimal"

        },

        "WeightCorrection": {

            "BadInput": "Sa correcció de pes per temperatura ha de ser un nombre
real",

            "RangeUnderflow": "Sa correcció de pes per temperatura mínima permesa és
-1000",

            "RangeOverflow": "Sa correcció de pes per temperatura màxima permesa és
1000",

            "StepMismatch": "Només es permet un decimal"

        },

        "MaxThingSpeakChannelsRegistered" : "S'ha assolit es nombre màxim de canals
de ThingSpeak que poden ser registrats. Un canal ha de ser eliminat abans d'afegir un
altre.",
```

```
        "DeleteThingSpeakChannel": "Es següent canal de ThingSpeak serà eliminat:",
        "Cancel": "Cancel·lar",
        "Confirm": "Confirmar",
        "ScheduleUpload": {
            "WrongFileType": "S'arxiu ha de ser .wpi",
            "FileSizeTooBig": "S'arxiu supera es 10MB"
        },
        "SettingsBackupUpload": {
            "WrongFileType": "S'arxiu ha de ser .db",
            "FileSizeTooBig": "S'arxiu supera es 10MB"
        }
    }
}
```

A.5.2. Arxiu '/home/SweetHive/SetupInterface/Language/Català.json'

```
{
    "Title": {
        "Index": "Configuració"
    },

    "AccordionHeader": {
        "ThingSpeak": "ThingSpeak",
        "Hive": "Rusc #",
```

```
        "Schedule": "Horari",

        "SettingsBackup": "Còpia de seguretat",

        "Language": "Idioma"

    },



    "ThingSpeakChannelRegister": {

        "Name": "Nom del Canal",

        "NameHelp": "El nom del canal pot contenir fins a 20 caràcters",

        "ID": "ID del Canal",

        "IDHelp": "El ID del canal ha d'estar format per 6 dígits",

        "WriteAPIKey": "Clau API d'Escriptura",

        "WriteAPIKeyHelp": "La clau API d'escriptura ha d'estar formada per 17
caràcters"

    },



    "HiveSetupInterface": {

        "Temperature": "Temperatura ",

        "Humidity": "Humitat ",

        "Weight": "Pes ",

        "SensorEnable": "Habilitar Mesura",

        "GPIO": "GPIO",

        "ThingSpeakChannel": "Canal de ThingSpeak",

        "ThingSpeakField": "Camp del Canal",
```

```
        "Offset": "Compensació",
        "Save": "Guardar Configuració",
        "Measure": "Realitzar Mesura",
        "DS18B20ID": "ID del Sensor",
        "DS18B20SearchID": "Cercar ID del Sensor",
        "HX711DT": "GPIO DT",
        "HX711SCK": "GPIO SCK",
        "HX711Channel": "Canal del HX711",
        "HX711Scale": "Escala",
        "HX711TemperatureID": "ID Temperatura",
        "HX711TemperatureOffset": "Compensació Temperatura",
        "HX711TemperatureReference": "Referència Temperatura",
        "HX711TemperatureCorrection": "Correcció g/°C"
    },

    "Schedule": {
        "Save": "Guardar Horari",
        "Download": "Descarregar Horari",
        "Upload": "Carregar Horari",
        "SelectFile": "Seleccionar arxiu .wpi"
    },
```

```
"SettingsBackup": {
    "Download": "Descarregar Configuració",
    "Upload": "Carregar Configuració",
    "SelectFile": "Seleccionar arxiu .db"
},

"Alert": {
    "ThingSpeakChannelRegisterName": {
        "Missing": "És necessari incloure un nom pel canal",
        "TooLong": "El nom del canal és massa llarg"
    },
    "ThingSpeakChannelRegisterID": {
        "Missing": "És necessari incloure el ID del canal",
        "TooShort": "El ID del canal és massa curt",
        "TooLong": "El ID del canal és massa llarg"
    },
    "ThingSpeakChannelRegisterWriteAPIKey": {
        "Missing": "És necessari incloure la clau API d'escriptura",
        "TooShort": "La clau API d'escriptura és massa curta",
        "TooLong": "La clau API d'escriptura és massa llarga"
    },
    "TemperatureOffset": {
```

```
            "BadInput": "La compensació ha de ser un nombre real",

            "RangeUnderflow": "La compensació mínima permesa per a la temperatura és
-100",

            "RangeOverflow": "La compensació màxima permesa per a la temperatura és
100",

            "StepMismatch": "Tan sols es permet un decimal"

        },

        "TemperatureReference": {

            "BadInput": "La referència ha de ser un nombre real",

            "RangeUnderflow": "La referència mínima permesa per a la temperatura és
-100",

            "RangeOverflow": "La referència màxima permesa per a la temperatura és
100",

            "StepMismatch": "Tan sols es permet un decimal"

        },

        "HumidityOffset": {

            "BadInput": "La compensació ha de ser un nombre real",

            "RangeUnderflow": "La compensació mínima permesa per a la humitat és 0",

            "RangeOverflow": "La compensació màxima permesa per a la humitat és 100",

            "StepMismatch": "Tan sols es permet un decimal"

        },

        "WeightOffset": {

            "BadInput": "La compensació ha de ser un nombre real",

            "RangeUnderflow": "La compensació mínima permesa per al pes és -
1.000.000",
```

```
            "RangeOverflow": "La compensació màxima permesa per al pes és 1.000.000",

            "StepMismatch": "Tan sols es permet un decimal"

        },

        "WeightScale": {

            "BadInput": "L'escala ha de ser un nombre real",

            "RangeUnderflow": "L'escala mínima permesa per al pes és -100.000",

            "RangeOverflow": "L'escala màxima permesa per al pes és 100.000",

            "StepMismatch": "Tan sols es permet un decimal"

        },

        "WeightCorrection": {

            "BadInput": "La correcció de pes per temperatura ha de ser un nombre
real",

            "RangeUnderflow": "La correcció de pes per temperatura mínima permesa és
-1000",

            "RangeOverflow": "La correcció de pes per temperatura màxima permesa és
1000",

            "StepMismatch": "Tan sols es permet un decimal"

        },

        "MaxThingSpeakChannelsRegistered" : "S'ha assolit el nombre màxim de canals
de ThingSpeak que poden ser registrats. Un canal ha de ser eliminat abans d'afegir un
altre.",

        "DeleteThingSpeakChannel": "El següent canal de ThingSpeak serà eliminat:",

        "Cancel": "Cancel·lar",

        "Confirm": "Confirmar",

        "ScheduleUpload": {
```

```
            "WrongFileType": "L'arxiu ha de ser .wpi",
            "FileSizeTooBig": "L'arxiu supera els 10MB"
        },
        "SettingsBackupUpload": {
            "WrongFileType": "L'arxiu ha de ser .db",
            "FileSizeTooBig": "L'arxiu supera els 10MB"
        }
    }
}
```

A.5.3. Arxiu '/home/SweetHive/SetupInterface/Language/DefaultLanguage.json'

```
{
    "Title": {
        "Index": "Configuración"
    },


    "AccordionHeader": {
        "ThingSpeak": "ThingSpeak",
        "Hive": "Colmena #",
        "Schedule": "Horario",
        "SettingsBackup": "Copia de seguridad",
        "Language": "Idioma"
    },
```

```
    "ThingSpeakChannelRegister": {

        "Name": "Nombre del Canal",

        "NameHelp": "El nombre del canal puede contener hasta 20 carácteres",

        "ID": "ID del Canal",

        "IDHelp": "El ID del canal debe estar formado por 6 dígitos",

        "WriteAPIKey": "Clave API de Escritura",

        "WriteAPIKeyHelp": "La clave API de escritura debe estar formada por 17
carácteres"

    },



    "HiveSetupInterface": {

        "Temperature": "Temperatura ",

        "Humidity": "Humedad ",

        "Weight": "Peso ",

        "SensorEnable": "Habilitar Medición",

        "GPIO": "GPIO",

        "ThingSpeakChannel": "Canal de ThingSpeak",

        "ThingSpeakField": "Campo del Canal",

        "Offset": "Compensación",

        "Save": "Guardar Configuración",

        "Measure": "Realizar Medición",

        "DS18B20ID": "ID del Sensor",
```

```
        "DS18B20SearchID": "Buscar ID del Sensor",
        "HX711DT": "GPIO DT",
        "HX711SCK": "GPIO SCK",
        "HX711Channel": "Canal del HX711",
        "HX711Scale": "Escala",
        "HX711TemperatureID": "ID Temperatura",
        "HX711TemperatureOffset": "Compensación Temperatura",
        "HX711TemperatureReference": "Referencia Temperatura",
        "HX711TemperatureCorrection": "Corrección g/°C"
    },

    "Schedule": {
        "Save": "Guardar Horario",
        "Download": "Descargar Horario",
        "Upload": "Cargar Horario",
        "SelectFile": "Seleccionar archivo .wpi"
    },

    "SettingsBackup": {
        "Download": "Descargar Configuración",
        "Upload": "Cargar Configuración",
        "SelectFile": "Seleccionar archivo .db"
```

```
    },



    "Alert": {

        "ThingSpeakChannelRegisterName": {

            "Missing": "Es necesario incluir un nombre para el canal",

            "TooLong": "El nombre del canal es demasiado largo"

        },

        "ThingSpeakChannelRegisterID": {

            "Missing": "Es necesario incluir el ID del canal",

            "TooShort": "El ID del canal es demasiado corto",

            "TooLong": "El ID del canal es demasiado largo"

        },

        "ThingSpeakChannelRegisterWriteAPIKey": {

            "Missing": "Es necesario incluir la clave API de escritura",

            "TooShort": "La clave API de escritura es demasiada corta",

            "TooLong": "La clave API de escritura es demasiada larga"

        },

        "TemperatureOffset": {

            "BadInput": "La compensación debe ser un número real",

            "RangeUnderflow": "La compensación mínima permitida para la temperatura
es -100",

            "RangeOverflow": "La compensación máxima permitida para la temperatura es
100",
```

```
            "StepMismatch": "Solamente se permite un decimal"

        },

        "TemperatureReference": {

            "BadInput": "La referencia debe ser un número real",

            "RangeUnderflow": "La referencia mínima permitida para la temperatura es
-100",

            "RangeOverflow": "La referencia máxima permitida para la temperatura es
100",

            "StepMismatch": "Solamente se permite un decimal"

        },

        "HumidityOffset": {

            "BadInput": "La compensación debe ser un número real",

            "RangeUnderflow": "La compensación mínima permitida para la humedad es
0",

            "RangeOverflow": "La compensación máxima permitida para la humedad es
100",

            "StepMismatch": "Solamente se permite un decimal"

        },

        "WeightOffset": {

            "BadInput": "La compensación debe ser un número real",

            "RangeUnderflow": "La compensación mínima permitida para el peso es -
1.000.000",

            "RangeOverflow": "La compensación máxima permitida para el peso es
1.000.000",

            "StepMismatch": "Solamente se permite un decimal"

        },
```

```
        "WeightScale": {

            "BadInput": "La escala debe ser un número real",

            "RangeUnderflow": "La escala mínima permitida para el peso es -100.000",

            "RangeOverflow": "La escala máxima permitida para el peso es 100.000",

            "StepMismatch": "Solamente se permite un decimal"

        },

        "WeightCorrection": {

            "BadInput": "La corrección de peso por temperatura debe ser un número
real",

            "RangeUnderflow": "La corrección de peso por temperatura mínima permitida
es -1000",

            "RangeOverflow": "La corrección de peso por temperatura máxima permitida
es 1000",

            "StepMismatch": "Solamente se permite un decimal"

        },

        "MaxThingSpeakChannelsRegistered" : "Se ha alcanzado el número máximo de
canales de ThingSpeak que pueden ser registrados. Un canal debe ser eliminado antes
de agregar otro.",

        "DeleteThingSpeakChannel": "El siguiente canal de ThingSpeak va a ser
eliminado:",

        "Cancel": "Cancelar",

        "Confirm": "Confirmar",

        "ScheduleUpload": {

            "WrongFileType": "El archivo debe ser .wpi",

            "FileSizeTooBig": "El archivo sobrepasa los 10MB"

        },
```

```
        "SettingsBackupUpload": {
            "WrongFileType": "El archivo debe ser .db",
            "FileSizeTooBig": "El archivo sobrepasa los 10MB"
        }
    }
}
```

A.5.4. Arxiu '/home/SweetHive/SetupInterface/Language/English.json'

```
{
    "Title": {
        "Index": "Setup"
    },

    "AccordionHeader": {
        "ThingSpeak": "ThingSpeak",
        "Hive": "Hive #",
        "Schedule": "Schedule",
        "SettingsBackup": "Settings Backup",
        "Language": "Language"
    },

    "ThingSpeakChannelRegister": {
        "Name": "Channel Name",
```

```
        "NameHelp": "The channel name can contain up to 20 characters",
        "ID": "Channel ID",
        "IDHelp": "The channel ID must be made up of 6 digits",
        "WriteAPIKey": "Write API Key",
        "WriteAPIKeyHelp": "The write API key must be made up of 17 characters"
    },

    "HiveSetupInterface": {
        "Temperature": "Temperature ",
        "Humidity": "Humidity ",
        "Weight": "Weight ",
        "SensorEnable": "Enable Measurement",
        "GPIO": "GPIO",
        "ThingSpeakChannel": "ThingSpeak Channel",
        "ThingSpeakField": "Channel Field",
        "Offset": "Offset",
        "Save": "Save Setup",
        "Measure": "Perform Measurement",
        "DS18B20ID": "Sensor ID",
        "DS18B20SearchID": "Search Sensor ID",
        "HX711DT": "GPIO DT",
        "HX711SCK": "GPIO SCK",
```

```
        "HX711Channel": "HX711 Channel",
        "HX711Scale": "Scale",
        "HX711TemperatureID": "Temperature ID",
        "HX711TemperatureOffset": "Temperature Offset",
        "HX711TemperatureReference": "Temperature Reference",
        "HX711TemperatureCorrection": "Correction g/°C"
    },

    "Schedule": {
        "Save": "Save Schedule",
        "Download": "Download Schedule",
        "Upload": "Upload Schedule",
        "SelectFile": "Select file .wpi"
    },

    "SettingsBackup": {
        "Download": "Download Settings",
        "Upload": "Upload Settings",
        "SelectFile": "Select file .db"
    },

    "Alert": {
```

```
"ThingSpeakChannelRegisterName": {
    "Missing": "A channel name is required",
    "TooLong": "The channel name is too long"
},
"ThingSpeakChannelRegisterID": {
    "Missing": "The channel ID is required",
    "TooShort": "The channel ID is too short",
    "TooLong": "The channel ID is too long"
},
"ThingSpeakChannelRegisterWriteAPIKey": {
    "Missing": "The write API key is required",
    "TooShort": "The write API key is too short",
    "TooLong": "The write API key is too long"
},
"TemperatureOffset": {
    "BadInput": "Offset must be a real number",
    "RangeUnderflow": "Minimum offset allowed for temperature is -100",
    "RangeOverflow": "Maximum offset allowed for temperature is 100",
    "StepMismatch": "Only one decimal is allowed"
},
"TemperatureReference": {
    "BadInput": "Reference must be a real number",
```

```
        "RangeUnderflow": "Minimum reference allowed for temperature is -100",
        "RangeOverflow": "Maximum reference allowed for temperature is 100",
        "StepMismatch": "Only one decimal is allowed"
    },
    "HumidityOffset": {
        "BadInput": "Offset must be a real number",
        "RangeUnderflow": "Minimum offset allowed for humidity is 0",
        "RangeOverflow": "Maximum offset allowed for humidity is 100",
        "StepMismatch": "Only one decimal is allowed"
    },
    "WeightOffset": {
        "BadInput": "Offset must be a real number",
        "RangeUnderflow": "Minimum offset allowed for weight is -1.000.000",
        "RangeOverflow": "Maximum offset allowed for weight is 1.000.000",
        "StepMismatch": "Only one decimal is allowed"
    },
    "WeightScale": {
        "BadInput": "Scale must be a real number",
        "RangeUnderflow": "Minimum scale allowed for weight is -100.000",
        "RangeOverflow": "Maximum scale allowed for weight is 100.000",
        "StepMismatch": "Only one decimal is allowed"
    },
```

```
        "WeightCorrection": {

            "BadInput": "Weight/temperature correction must be a real number",

            "RangeUnderflow": "Minimum weight/temperature correction allowed is -
1000",

            "RangeOverflow": "Maximum weight/temperature correction allowed is 1000",

            "StepMismatch": "Only one decimal is allowed"

        },

        "MaxThingSpeakChannelsRegistered" : "The maximum number of ThingSpeak
channels that can be registered has been reached. A channel must be removed before
another one can be added.",

        "DeleteThingSpeakChannel": "The following ThingSpeak channel is going to be
removed:",

        "Cancel": "Cancel",

        "Confirm": "Confirm",

        "ScheduleUpload": {

            "WrongFileType": "File must be .wpi",

            "FileSizeTooBig": "File exceeds 10MB"

        },

        "SettingsBackupUpload": {

            "WrongFileType": "File must be .db",

            "FileSizeTooBig": "File exceeds 10MB"

        }

    }

}
```

A.5.5. Arxiu '/home/SweetHive/SetupInterface/Language/Español.json'

```
{

    "Title": {

        "Index": "Configuración"

    },



    "AccordionHeader": {

        "ThingSpeak": "ThingSpeak",

        "Hive": "Colmena #",

        "Schedule": "Horario",

        "SettingsBackup": "Copia de seguridad",

        "Language": "Idioma"

    },



    "ThingSpeakChannelRegister": {

        "Name": "Nombre del Canal",

        "NameHelp": "El nombre del canal puede contener hasta 20 carácteres",

        "ID": "ID del Canal",

        "IDHelp": "El ID del canal debe estar formado por 6 dígitos",

        "WriteAPIKey": "Clave API de Escritura",

        "WriteAPIKeyHelp": "La clave API de escritura debe estar formada por 17
carácteres"
```

```
},

"HiveSetupInterface": {
    "Temperature": "Temperatura ",
    "Humidity": "Humedad ",
    "Weight": "Peso ",
    "SensorEnable": "Habilitar Medición",
    "GPIO": "GPIO",
    "ThingSpeakChannel": "Canal de ThingSpeak",
    "ThingSpeakField": "Campo del Canal",
    "Offset": "Compensación",
    "Save": "Guardar Configuración",
    "Measure": "Realizar Medición",
    "DS18B20ID": "ID del Sensor",
    "DS18B20SearchID": "Buscar ID del Sensor",
    "HX711DT": "GPIO DT",
    "HX711SCK": "GPIO SCK",
    "HX711Channel": "Canal del HX711",
    "HX711Scale": "Escala",
    "HX711TemperatureID": "ID Temperatura",
    "HX711TemperatureOffset": "Compensación Temperatura",
    "HX711TemperatureReference": "Referencia Temperatura",
```

```
        "HX711TemperatureCorrection": "Corrección g/°C"
    },

    "Schedule": {
        "Save": "Guardar Horario",
        "Download": "Descargar Horario",
        "Upload": "Cargar Horario",
        "SelectFile": "Seleccionar archivo .wpi"
    },

    "SettingsBackup": {
        "Download": "Descargar Configuración",
        "Upload": "Cargar Configuración",
        "SelectFile": "Seleccionar archivo .db"
    },

    "Alert": {
        "ThingSpeakChannelRegisterName": {
            "Missing": "Es necesario incluir un nombre para el canal",
            "TooLong": "El nombre del canal es demasiado largo"
        },
        "ThingSpeakChannelRegisterID": {
```

```
            "Missing": "Es necesario incluir el ID del canal",

            "TooShort": "El ID del canal es demasiado corto",

            "TooLong": "El ID del canal es demasiado largo"

        },

        "ThingSpeakChannelRegisterWriteAPIKey": {

            "Missing": "Es necesario incluir la clave API de escritura",

            "TooShort": "La clave API de escritura es demasiada corta",

            "TooLong": "La clave API de escritura es demasiada larga"

        },

        "TemperatureOffset": {

            "BadInput": "La compensación debe ser un número real",

            "RangeUnderflow": "La compensación mínima permitida para la temperatura
es -100",

            "RangeOverflow": "La compensación máxima permitida para la temperatura es
100",

            "StepMismatch": "Solamente se permite un decimal"

        },

        "TemperatureReference": {

            "BadInput": "La referencia debe ser un número real",

            "RangeUnderflow": "La referencia mínima permitida para la temperatura es
-100",

            "RangeOverflow": "La referencia máxima permitida para la temperatura es
100",

            "StepMismatch": "Solamente se permite un decimal"

        },
```

```
        "HumidityOffset": {
            "BadInput": "La compensación debe ser un número real",
            "RangeUnderflow": "La compensación mínima permitida para la humedad es 0",
            "RangeOverflow": "La compensación máxima permitida para la humedad es 100",
            "StepMismatch": "Solamente se permite un decimal"
        },
        "WeightOffset": {
            "BadInput": "La compensación debe ser un número real",
            "RangeUnderflow": "La compensación mínima permitida para el peso es -1.000.000",
            "RangeOverflow": "La compensación máxima permitida para el peso es 1.000.000",
            "StepMismatch": "Solamente se permite un decimal"
        },
        "WeightScale": {
            "BadInput": "La escala debe ser un número real",
            "RangeUnderflow": "La escala mínima permitida para el peso es -100.000",
            "RangeOverflow": "La escala máxima permitida para el peso es 100.000",
            "StepMismatch": "Solamente se permite un decimal"
        },
        "WeightCorrection": {
            "BadInput": "La corrección de peso por temperatura debe ser un número real",
```

```
            "RangeUnderflow": "La corrección de peso por temperatura mínima permitida
es -1000",

            "RangeOverflow": "La corrección de peso por temperatura máxima permitida
es 1000",

            "StepMismatch": "Solamente se permite un decimal"

        },

        "MaxThingSpeakChannelsRegistered" : "Se ha alcanzado el número máximo de
canales de ThingSpeak que pueden ser registrados. Un canal debe ser eliminado antes
de agregar otro.",

        "DeleteThingSpeakChannel": "El siguiente canal de ThingSpeak va a ser
eliminado:",

        "Cancel": "Cancelar",

        "Confirm": "Confirmar",

        "ScheduleUpload": {

            "WrongFileType": "El archivo debe ser .wpi",

            "FileSizeTooBig": "El archivo sobrepasa los 10MB"

        },

        "SettingsBackupUpload": {

            "WrongFileType": "El archivo debe ser .db",

            "FileSizeTooBig": "El archivo sobrepasa los 10MB"

        }

    }

}
```

A.5.6. Arxiu '/home/SweetHive/SetupInterface/static/js/funcions.js'

```
function InvalidThingSpeakName(TextBox, Missing, TooLong) {
```

```
    if (TextBox.validity.valueMissing) {
        TextBox.setCustomValidity(Missing);
    } else if (TextBox.validity.tooLong) {
        TextBox.setCustomValidity(TooLong);
    } else {
        TextBox.setCustomValidity('');
    }

    input.reportValidity();
}

function InvalidThingSpeakID(TextBox, Missing, TooShort, TooLong) {

    if (TextBox.validity.valueMissing) {
        TextBox.setCustomValidity(Missing);
    } else if (TextBox.validity.tooShort) {
        TextBox.setCustomValidity(TooShort);
    } else if (TextBox.validity.tooLong) {
        TextBox.setCustomValidity(TooLong);
    } else {
        TextBox.setCustomValidity('');
```

```
    }


    input.reportValidity();
}


function InvalidThingSpeakWriteAPIKey(TextBox, Missing, TooShort, TooLong) {


    if (TextBox.validity.valueMissing) {
        TextBox.setCustomValidity(Missing);
    } else if (TextBox.validity.tooShort) {
        TextBox.setCustomValidity(TooShort);
    } else if (TextBox.validity.tooLong) {
        TextBox.setCustomValidity(TooLong);
    } else {
        TextBox.setCustomValidity('');
    }


    input.reportValidity();
}


function InvalidOffset(TextBox, BadInput, RangeUnderflow, RangeOverflow,
StepMismatch) {
```

```
    if (TextBox.validity.badInput) {

        TextBox.setCustomValidity(BadInput);

    } else if (TextBox.validity.rangeUnderflow) {

        TextBox.setCustomValidity(RangeUnderflow);

    } else if (TextBox.validity.rangeOverflow) {

        TextBox.setCustomValidity(RangeOverflow);

    } else if (TextBox.validity.stepMismatch) {

        TextBox.setCustomValidity(StepMismatch);

    } else {

        TextBox.setCustomValidity('');

    }



    input.reportValidity();

}



function InvalidScale(TextBox, BadInput, RangeUnderflow, RangeOverflow, StepMismatch)
{



    if (TextBox.validity.badInput) {

        TextBox.setCustomValidity(BadInput);

    } else if (TextBox.validity.rangeUnderflow) {

        TextBox.setCustomValidity(RangeUnderflow);

    } else if (TextBox.validity.rangeOverflow) {
```

```
        TextBox.setCustomValidity(RangeOverflow);
    } else if (TextBox.validity.stepMismatch) {
        TextBox.setCustomValidity(StepMismatch);
    } else {
        TextBox.setCustomValidity('');
    }

    input.reportValidity();
}

function InvalidReference(TextBox, BadInput, RangeUnderflow, RangeOverflow,
StepMismatch) {

    if (TextBox.validity.badInput) {
        TextBox.setCustomValidity(BadInput);
    } else if (TextBox.validity.rangeUnderflow) {
        TextBox.setCustomValidity(RangeUnderflow);
    } else if (TextBox.validity.rangeOverflow) {
        TextBox.setCustomValidity(RangeOverflow);
    } else if (TextBox.validity.stepMismatch) {
        TextBox.setCustomValidity(StepMismatch);
    } else {
        TextBox.setCustomValidity('');
```

```
    }



    input.reportValidity();

}



function InvalidCorrection(TextBox, BadInput, RangeUnderflow, RangeOverflow,
StepMismatch) {



    if (TextBox.validity.badInput) {

        TextBox.setCustomValidity(BadInput);

    } else if (TextBox.validity.rangeUnderflow) {

        TextBox.setCustomValidity(RangeUnderflow);

    } else if (TextBox.validity.rangeOverflow) {

        TextBox.setCustomValidity(RangeOverflow);

    } else if (TextBox.validity.stepMismatch) {

        TextBox.setCustomValidity(StepMismatch);

    } else {

        TextBox.setCustomValidity('');

    }



    input.reportValidity();

}
```

```
function ValidateFile(File, Type, AlertID, LabelID, WrongType, TooBig) {



    if (File.name.split(".").slice(-1) != Type) {

        alertPlaceholder = document.getElementById(AlertID)

        wrapper = document.createElement('div')

        wrapper.innerHTML = [

            `<div class="alert alert-danger alert-dismissible" role="alert">`,

            `   <div>'${WrongType}'</div>`,

            '   <button type="button" class="btn-close" data-bs-dismiss="alert" aria-
label="Close"></button>',

            '</div>'

        ].join('')

        alertPlaceholder.append(wrapper)

    } else if (File.size > 10 * 1048576) {

        alertPlaceholder = document.getElementById(AlertID)

        wrapper = document.createElement('div')

        wrapper.innerHTML = [

            `<div class="alert alert-danger alert-dismissible" role="alert">`,

            `   <div>'${TooBig}'</div>`,

            '   <button type="button" class="btn-close" data-bs-dismiss="alert" aria-
label="Close"></button>',

            '</div>'

        ].join('')
```

```
        alertPlaceholder.append(wrapper)

    } else {

        label = document.getElementById(LabelID);

        label.textContent = File.name;

    }

}
```

A.5.7. Arxiu '/home/SweetHive/SetupInterface/templates/index.html'

```
{% extends 'layout.html' %}



{%    from    'macros.html'    import    OpenAccordionItem,    CloseAccordionItem,
FormThingSpeakChannel, FormTemperature, FormHumidity, FormWeight %}



{% block Title %}{{ language.Title.Index }}{% endblock %}



{% block Content %}

<div class="accordion" id="accordionPanelsStayOpen">

      {{    OpenAccordionItem(language.AccordionHeader.ThingSpeak,    'ThingSpeak',
Accordion) }}

      {% if ThingSpeakChannels|length > 0 %}

      <div class="row justify-content-center">

            {% for ThingSpeak in ThingSpeakChannels %}

            <div class="col-auto">

                  <div class="card mb-3" id="ThingSpeak{{ ThingSpeak.ChannelNumber
}}">
```

```
                        <div class="card-body">
                            <div class="table-responsive">
                                <table class="table">
                                    <thead>
                                        <tr>
                                            <th scope="col" class="text-center" colspan="2">
                                                {{ ThingSpeak.ChannelName }}
                                            </th>
                                        </tr>
                                    </thead>

                                    <body>
                                        <tr>
                                            <td scope="row" class="align-bottom text-center">
                                                <p>{{ ThingSpeak.ChannelID }}</p>
                                                <p>{{ ThingSpeak.ChannelWriteAPIKey }}</p>
                                            </td>
                                            <td class="align-middle">
                                                <div class="d-flex flex-column">
                                                    <button type="button" class="btn btn-primary m-1" data-bs-toggle="modal"
```

```
                                                                    data-
bs-target="#EditThingSpeakChannel{{ ThingSpeak.ChannelNumber }}Modal">

                                                                    <i
class="bi bi-pencil-fill"></i>

                                                                </button>

                                                                <div
class="modal fade" tabindex="-1" aria-hidden="true"

      id="EditThingSpeakChannel{{ ThingSpeak.ChannelNumber }}Modal"

                                                                    aria-
labelledby="EditThingSpeakChannel{{ ThingSpeak.ChannelNumber }}ModalLabel">

                                                                    <div
class="row justify-content-center">

      <div class="col-auto">

            <div class="modal-dialog mt-5">

                  <div class="modal-content">

                        <div class="modal-body">

                              <form method="POST"

                                    action="/EditThingSpeakChannel/{{
ThingSpeak.ChannelNumber }}">

                                    {{ FormThingSpeakChannel(language,

ThingSpeak.ChannelName,

ThingSpeak.ChannelID,
```

```
ThingSpeak.ChannelWriteAPIKey) }}

                                    </form>

                                </div>

                            </div>

                        </div>

                    </div>

                    </div>

                                                                    </div>

                                                                    <button
type="button" class="btn btn-danger m-1" data-bs-toggle="modal"

                                                                            data-
bs-target="#DeleteThingSpeakChannel{{ ThingSpeak.ChannelNumber }}Modal">

                                                                            <i
class="bi bi-trash3-fill"></i>

                                                                    </button>

                                                                    <div
class="modal fade" tabindex="-1" aria-hidden="true"

                    id="DeleteThingSpeakChannel{{ ThingSpeak.ChannelNumber }}Modal"

                                                                            aria-
labelledby="DeleteThingSpeakChannel{{ ThingSpeak.ChannelNumber }}ModalLabel">

                                                                            <div
class="row justify-content-center">

                    <div class="col-auto">
```

```
            <div class="modal-dialog mt-5">

                  <div class="modal-content">

                        <div class="modal-body">

                              <b>{{ language.Alert.DeleteThingSpeakChannel }}</b>

                              <p class="text-center">

                                    {{ ThingSpeak.ChannelName }}

                              </p>

                              <div class="d-flex justify-content-evenly">

                                    <button type="button" data-bs-dismiss="modal"

                                          class="btn btn-secondary">

                                          {{ language.Alert.Cancel }}

                                    </button>

                                    <a          href="/DeleteThingSpeakChannel/{{
ThingSpeak.ChannelNumber }}"

                                          type="button" class="btn btn-primary">

                                          {{ language.Alert.Confirm }}

                                    </a>
```

```
                                </div>

                            </div>

                        </div>

                    </div>

                </div>

                                                                    </div>

                                                                </div>

                                                            </td>

                                                        </tr>

                                                    </body>

                                                </table>

                                            </div>

                                        </div>

                                    </div>

                                </div>

                                {% endfor %}

                            </div>

                            {% endif %}

                            <div class="row justify-content-center">
```

```
            <div class="col-auto">
                <button type="button" class="btn btn-success" data-bs-
toggle="modal"
                        data-bs-target="#AddThingSpeakChannelModal">
                        <i class="bi bi-plus-lg"></i>
                </button>
                <div class="modal fade" id="AddThingSpeakChannelModal"
tabindex="-1"
                        aria-labelledby="AddThingSpeakChannelModalLabel" aria-
hidden="true">
                        <div class="row justify-content-center">
                              <div class="col-auto">
                                    <div class="modal-dialog mt-5">
                                          <div class="modal-content">
                                                <div class="modal-body">
                                                      {% if
ThingSpeakChannels|length >= MaxThingSpeakChannels %}
                                                      <p><b>{{
language.Alert.MaxThingSpeakChannelsRegistered }}</b></p>
                                                      <div class="d-flex
justify-content-evenly">
                                                            <button
type="button" class="btn btn-primary " data-bs-dismiss="modal">
                                                                  {{
language.Alert.Confirm }}
                                                            </button>
                                                      </div>
```

```
                                                {% else %}
                                                <form
action="/AddThingSpeakChannel" method="POST">
                                                    {{
FormThingSpeakChannel(language, "", "", "") }}
                                                </form>
                                                {% endif %}
                                            </div>
                                        </div>
                                    </div>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
                {{ CloseAccordionItem() }}

                {% for HiveNumber in range(MaxHives) %}
                {{ OpenAccordionItem(language.AccordionHeader.Hive ~ (HiveNumber + 1), 'Hive'
~ (HiveNumber + 1), Accordion) }}
                <div class="row justify-content-center">
                    {{ FormTemperature(language, HiveNumber + 1,
TemperatureSensors[HiveNumber], ThingSpeakChannels, DS18B20SensorID) }}
                    {{ FormHumidity(language, HiveNumber + 1, HumiditySensors[HiveNumber],
ThingSpeakChannels, GPIO) }}
```

```
            {{ FormWeight(language, HiveNumber + 1, WeightSensors[HiveNumber],
ThingSpeakChannels, GPIO, DS18B20SensorID) }}

      </div>

      {{ CloseAccordionItem() }}

      {% endfor %}



      {{      OpenAccordionItem(language.AccordionHeader.Schedule,      'Schedule',
Accordion)}}

      <div class="row justify-content-center">

            <form method="POST" action="/ChangeSchedule" id="SaveSchedule">

                  <div class="form-group mx-3">

                        <textarea class="form-control mb-3" id="ScheduleRegister"
name="FormScheduleRegister" rows="8"

                              oninput="this.value = this.value.toUpperCase()">{{
Schedule }}</textarea>

                  </div>

            </form>

      </div>

      <div class="row justify-content-center">

            <div class="d-flex justify-content-evenly align-items-center">

                  <a href="/Download/schedule.wpi" type="button" class="btn btn-
primary mx-2">

                        <i class="bi bi-download"></i>

                        {{ language.Schedule.Download }}

                  </a>
```

```
                    <form           method="POST"           action="/UploadSchedule"
enctype="multipart/form-data">

                        <div class="form-group d-flex flex-column">

                            <div id="ScheduleUploadAlert"></div>

                            <label  for="ScheduleUpload"  class="btn  btn-dark"
id="ScheduleUploadLabel">

                                <i class="bi bi-search"></i>

                                {{ language.Schedule.SelectFile }}

                            </label>

                            <input       class="form-control"       type="file"
id="ScheduleUpload" name="FormScheduleUpload" accept=".wpi"

                                style="display:none"
onchange="ValidateFile(this.files[0],          'wpi',          'ScheduleUploadAlert',
'ScheduleUploadLabel',

                                '{{     language.Alert.ScheduleUpload.WrongFileType
}}',

                                '{{     language.Alert.ScheduleUpload.FileSizeTooBig
}}')">

                            <button type="submit" class="btn btn-dark mt-2">

                                <i class="bi bi-upload"></i>

                                {{ language.Schedule.Upload }}

                            </button>

                        </div>

                    </form>

                    <button type="submit" form="SaveSchedule" class="btn btn-success
mx-2">

                        {{ language.Schedule.Save }}
```

```
                    </button>

                </div>

        </div>

        {{ CloseAccordionItem() }}



        {{                  OpenAccordionItem(language.AccordionHeader.SettingsBackup,
'SettingsBackup', Accordion)}}

        <div class="row justify-content-center">

                <div class="d-flex justify-content-evenly align-items-center">

                    <a  href="/Download/Settings.db"  type="button"  class="btn  btn-
primary mx-2">

                            <i class="bi bi-download"></i>

                            {{ language.SettingsBackup.Download }}

                    </a>

                    <form        method="POST"        action="/UploadSettingsBackup"
enctype="multipart/form-data">

                            <div class="form-group d-flex flex-column">

                                <div id="SettingsBackupUploadAlert"></div>

                                <label  for="SettingsBackupUpload"  class="btn  btn-
dark" id="SettingsBackupUploadLabel">

                                        <i class="bi bi-search"></i>

                                        {{ language.SettingsBackup.SelectFile }}

                                </label>

                                <input  onchange="ValidateFile(this.files[0],  'db',
'SettingsBackupUploadAlert', 'SettingsBackupUploadLabel',
```

```
                                                                                    '{{
language.Alert.SettingsBackupUpload.WrongFileType }}',

                                                                                    '{{
language.Alert.SettingsBackupUpload.FileSizeTooBig }}')" id="SettingsBackupUpload"

                                        name="FormSettingsBackupUpload"  class="form-
control" type="file" accept=".db"

                                        style="display:none">

                                <button type="submit" class="btn btn-dark mt-2">

                                        <i class="bi bi-upload"></i>

                                        {{ language.SettingsBackup.Upload }}

                                </button>

                        </div>

                </form>

        </div>

    </div>

    {{ CloseAccordionItem() }}



    {{      OpenAccordionItem(language.AccordionHeader.Language,      'Language',
Accordion)}}

    <div class="row justify-content-center">

        {% for AvailableLanguage in Languages %}

        {% if AvailableLanguage != 'DefaultLanguage' %}

        <div class="col-auto">

            <a href="/ChangeLanguage/{{ AvailableLanguage }}">

                {{ AvailableLanguage }}
```

```
                    </a>

                </div>

                {% endif %}

                {% endfor %}

        </div>

        {{ CloseAccordionItem() }}

</div>

{% endblock %}



{% block Scroll %}{{ ScrollingTo }}{% endblock %}
```

A.5.8. Arxiu '/home/SweetHive/SetupInterface/templates/layout.html'

```
<!DOCTYPE html>



<head>

    <meta charset="UTF-8">

    <meta http-equiv="Cache-control" content="no-cache, no-store, must-revalidate">

    <meta http-equiv="Pragma" content="no-cache">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>{% block Title %}{% endblock %}</title>

    <link rel="icon" href="{{ url_for('static', filename='ico/favicon.ico') }}"
type="image/x-icon">
```

```
    <link rel="stylesheet" href="{{ url_for('static', filename='css/bootstrap.min.css') }}">

    <link rel="stylesheet" href="{{ url_for('static', filename='css/bootstrap-icons.css') }}">

</head>



<body class="bg-dark">

    <div class="container my-5">

        {% block Content %}{% endblock %}

    </div>

    <script src="{{ url_for('static', filename='js/bootstrap.bundle.min.js') }}"></script>

    <script src="{{ url_for('static', filename='js/functions.js') }}"></script>

    <script>

        location.hash = "#{% block Scroll %}{% endblock %}"

    </script>

</body>



</html>
```

A.5.9. Arxiu ‘/home/SweetHive/SetupInterface/templates/macros.html’

```
{% macro OpenAccordionItem(Header, Id, Panels) %}

<div class="accordion-item">

    <h2 class="accordion-header" id="Heading{{ Id }}">

        <a href="/UpdateAccordionPanels/{{ Id }}" style="text-decoration: none">
```

```
            <button class="accordion-button {% if Panels[Id] == false %}collapsed{%
endif %}" type="button"

                data-bs-toggle="collapse"  data-bs-target="#Panel{{  Id  }}"  aria-
controls="Panel{{ Id }}">

                <b>{{ Header }}</b>

            </button>

        </a>

    </h2>

    <div id="Panel{{ Id }}" class="accordion-collapse collapse {% if Panels[Id] ==
true %}show{% endif %}"

        aria-labelledby="Heading{{ Id }}">

        <div class="accordion-body bg-light">

            {% endmacro %}




            {% macro CloseAccordionItem() %}

        </div>

    </div>

</div>

{% endmacro %}




{% macro FormThingSpeakChannel(language, ThingSpeakChannelName, ThingSpeakChannelID,
ThingSpeakChannelWriteAPIKey) %}

<div class="form-group">

    <label for="ThingSpeakChannelNameRegister" class="form-label">

        <b>{{ language.ThingSpeakChannelRegister.Name }}</b>
```

```
    </label>

    <input oninvalid="InvalidThingSpeakName(this,

                                            '{{
language.Alert.ThingSpeakChannelRegisterName.Missing }}',

                                            '{{
language.Alert.ThingSpeakChannelRegisterName.TooLong }}')"

        id="ThingSpeakChannelNameRegister"          name="FormThingSpeakChannelName"
value="{{ ThingSpeakChannelName }}" required

        oninput="InvalidThingSpeakName(this,

                                        '{{
language.Alert.ThingSpeakChannelRegisterName.Missing }}',

                                        '{{
language.Alert.ThingSpeakChannelRegisterName.TooLong }}')" minlength="1"

        maxlength="20" type="text" class="form-control">

    <div id="ThingSpeakChannelNameHelp" class="form-text">

        {{ language.ThingSpeakChannelRegister.NameHelp }}

    </div>

</div>

<div class="form-group mt-3">

    <label for="ThingSpeakChannelIDRegister" class="form-label">

        <b>{{ language.ThingSpeakChannelRegister.ID }}</b>

    </label>

    <input oninvalid="InvalidThingSpeakID(this,

                                        '{{
language.Alert.ThingSpeakChannelRegisterID.Missing }}',

                                        '{{
language.Alert.ThingSpeakChannelRegisterID.TooShort }}',
```

```
                                          '{{
language.Alert.ThingSpeakChannelRegisterID.TooLong }}')"

        id="ThingSpeakChannelIDRegister"  name="FormThingSpeakChannelID"  value="{{
ThingSpeakChannelID }}" required

        oninput="InvalidThingSpeakID(this,

                                        '{{
language.Alert.ThingSpeakChannelRegisterID.Missing }}',

                                        '{{
language.Alert.ThingSpeakChannelRegisterID.TooShort }}',

                                        '{{
language.Alert.ThingSpeakChannelRegisterID.TooLong }}')" minlength="7"

        maxlength="7" type="text" class="form-control">

    <div id="ThingSpeakChannelIDHelp" class="form-text">

        {{ language.ThingSpeakChannelRegister.IDHelp }}

    </div>

</div>

<div class="form-group mt-3">

    <label for="ThingSpeakChannelWriteAPIKeyRegister" class="form-label">

        <b>{{ language.ThingSpeakChannelRegister.WriteAPIKey }}</b>

    </label>

    <input oninput="InvalidThingSpeakWriteAPIKey(this,

                                                '{{
language.Alert.ThingSpeakChannelRegisterWriteAPIKey.Missing }}',

                                                '{{
language.Alert.ThingSpeakChannelRegisterWriteAPIKey.TooShort }}',

                                                '{{
language.Alert.ThingSpeakChannelRegisterWriteAPIKey.TooLong }}')"
```

```
        oninvalid="InvalidThingSpeakWriteAPIKey(this,

                                                '{{
language.Alert.ThingSpeakChannelRegisterWriteAPIKey.Missing }}',

                                                '{{
language.Alert.ThingSpeakChannelRegisterWriteAPIKey.TooShort }}',

                                                '{{
language.Alert.ThingSpeakChannelRegisterWriteAPIKey.TooLong }}')"

        id="ThingSpeakChannelWriteAPIKeyRegister"
name="FormThingSpeakChannelWriteAPIKey" type="text"

        class="form-control"  value="{{  ThingSpeakChannelWriteAPIKey  }}"  required
minlength="16" maxlength="16">

    <div id="ThingSpeakChannelWriteAPIKeyHelp" class="form-text">

        {{ language.ThingSpeakChannelRegister.WriteAPIKeyHelp }}

    </div>

</div>

<div class="d-flex justify-content-evenly mt-3">

    <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">

        {{ language.Alert.Cancel }}

    </button>

    <button type="submit" class="btn btn-primary">

        {{ language.Alert.Confirm }}

    </button>

</div>

{% endmacro %}
```

```
{% macro FormTemperature(language, HiveNumber, TemperatureSensor, ThingSpeakChannels,
ID) %}

<div class="col-auto">

    <div class="card mb-3" id="Hive{{ HiveNumber }}Temperature">

        <h5 class="card-tittle text-center mt-3">

            {{             language.HiveSetupInterface.Temperature             }}[{{
language.AccordionHeader.Hive }}{{ HiveNumber}}]

        </h5>

        <div class="card-body">

            <form action="/EditTemperatureSensor/{{ HiveNumber }}" method="POST">

                <div class="form-check form-switch">

                    <input  class="form-check-input"  type="checkbox"  role="switch"
id="TemperatureSensorEnabled"

                        value="1"    name="FormTemperatureSensorEnabled"    {%    if
TemperatureSensor.Enabled %} checked

                        {% endif %}>

                    <input     type="hidden"     name="FormTemperatureSensorEnabled"
value="0">

                    <label class="form-check-label" for="TemperatureSensorEnabled">

                        {{ language.HiveSetupInterface.SensorEnable }}

                    </label>

                </div>

                <div class="input-group mt-2">

                    <label class="input-group-text" for="TemperatureSensorID">

                        {{ language.HiveSetupInterface.DS18B20ID }}
```

```
                    </label>

                    <select       class="form-select"       id="TemperatureSensorID"
name="FormTemperatureSensorID">

                        <option></option>

                        {% if TemperatureSensor.ID %}

                        <option selected>{{ TemperatureSensor.ID }}</option>

                        {% endif %}

                        {% for DS18B20 in ID %}

                        {% if TemperatureSensor.ID != DS18B20.SensorID %}

                        <option>{{ DS18B20.SensorID }}</option>

                        {% endif %}

                        {% endfor %}

                    </select>

                </div>

                <div class="form-group mt-2 bg-secondary bg-opacity-25 rounded-2">

                    <label                                    class="input-group-text"
for="TemperatureSensorThingSpeakChannel">

                        {{ language.HiveSetupInterface.ThingSpeakChannel }}

                    </label>

                    <select                                          class="form-select"
id="TemperatureSensorThingSpeakChannel"

                        name="FormTemperatureSensorThingSpeakChannel">

                        <option></option>

                        {% for ThingSpeak in ThingSpeakChannels %}
```

```
                        <option value="{{ ThingSpeak.ChannelNumber }}"

                            {%   if   TemperatureSensor.ThingSpeakChannelNumber   ==
ThingSpeak.ChannelNumber %} selected

                            {% endif %}>

                            {{ ThingSpeak.ChannelName }}

                        </option>

                        {% endfor %}

                    </select>

                </div>

                <div class="input-group mt-2">

                    <label                                    class="input-group-text"
for="TemperatureSensorThingSpeakChannelField">

                        {{ language.HiveSetupInterface.ThingSpeakField }}

                    </label>

                    <select                                        class="form-select"
id="TemperatureSensorThingSpeakChannelField"

                        name="FormTemperatureSensorThingSpeakChannelField">

                        <option></option>

                        {% for Field in range(10) %}

                        <option  {%  if  TemperatureSensor.ThingSpeakChannelField  ==
Field + 1 %} selected {% endif %}>

                            {{ Field + 1 }}

                        </option>

                        {% endfor %}

                    </select>
```

```
                </div>

                <div class="input-group mt-2">

                    <label class="input-group-text" for="TemperatureSensorOffset">

                        {{ language.HiveSetupInterface.Offset }}

                    </label>

                    <input             oninput="InvalidOffset(this,             '{{
language.Alert.TemperatureOffset.BadInput }}',

                                                            '{{
language.Alert.TemperatureOffset.RangeUnderflow }}',

                                                            '{{
language.Alert.TemperatureOffset.RangeOverflow }}',

                                                            '{{
language.Alert.TemperatureOffset.StepMismatch }}')"

                        oninvalid="InvalidOffset(this,                          '{{
language.Alert.TemperatureOffset.BadInput }}',

                                                           '{{
language.Alert.TemperatureOffset.RangeUnderflow }}',

                                                           '{{
language.Alert.TemperatureOffset.RangeOverflow }}',

                                                           '{{
language.Alert.TemperatureOffset.StepMismatch }}')"

                        type="number"                   id="TemperatureSensorOffset"
name="FormTemperatureSensorOffset"

                        class="form-control" {% if not TemperatureSensor.Offset %}
value="0.0" {% else %}

                        value="{{ TemperatureSensor.Offset }}" {% endif %} min="-100"
max="100" step="0.1">

                </div>

                <div class="form-group d-grid mt-2">
```

```
                    <button type="submit" class="btn btn-success">
                        {{ language.HiveSetupInterface.Save }}
                    </button>
                </div>
            </form>
            <div class="d-flex flex-column d-grid mt-2">
                <a  href="/Execute/Temperature/{{  HiveNumber  }}/SearchDS18B20ID.py"
type="button"
                    class="btn btn-warning">
                    <i class="bi bi-search"></i>
                    {{ language.HiveSetupInterface.DS18B20SearchID }}
                </a>
                <a  href="/Execute/Temperature/{{  HiveNumber  }}/MeasureDS18B20.py"
type="submit"
                    class="btn btn-primary mt-2">
                    {{ language.HiveSetupInterface.Measure }}
                </a>
            </div>
            <div class="card mt-2 bg-dark">
                <div class="card-body">
                    <h4 class="text-center text-white">
                        {% if TemperatureSensor.Measurement %}
                        {{ TemperatureSensor.Measurement }} °C
```

```
                        {% endif %}

                    </h4>

                </div>

            </div>

        </div>

    </div>

</div>

{% endmacro %}



{% macro FormHumidity(language, HiveNumber, HumiditySensor, ThingSpeakChannels, GPIO)
%}

<div class="col-auto">

    <div class="card mb-3" id="Hive{{ HiveNumber }}Humidity">

        <h5 class="card-tittle text-center mt-3">

            {{              language.HiveSetupInterface.Humidity              }}[{{
language.AccordionHeader.Hive }}{{ HiveNumber }}]

        </h5>

        <div class="card-body">

            <form action="/EditHumiditySensor/{{ HiveNumber }}" method="POST">

                <div class="form-check form-switch">

                    <input  class="form-check-input"  type="checkbox"  role="switch"
id="HumiditySensorEnabled" value="1"

                        name="FormHumiditySensorEnabled"           {%           if
HumiditySensor.Enabled %} checked {% endif %}>

                    <input type="hidden" name="FormHumiditySensorEnabled" value="0">
```

```
                    <label class="form-check-label" for="HumiditySensorEnabled">
                        {{ language.HiveSetupInterface.SensorEnable }}
                    </label>
                </div>
                <div class="input-group mt-2">
                    <label class="input-group-text" for="HumiditySensorGPIO">
                        {{ language.HiveSetupInterface.GPIO }}
                    </label>
                    <select class="form-select" id="HumiditySensorGPIO"
name="FormHumiditySensorGPIO">
                        <option></option>
                        {% if HumiditySensor.GPIO %}
                        <option selected>{{ HumiditySensor.GPIO }}</option>
                        {% endif %}
                        {% for CheckGPIO in GPIO %}
                        {% if CheckGPIO.GPIO != HumiditySensor.GPIO %}
                        <option>{{ CheckGPIO.GPIO }}</option>
                        {% endif %}
                        {% endfor %}
                    </select>
                </div>
                <div class="form-group mt-2 bg-secondary bg-opacity-25 rounded-2">
```

```
                    <label                                    class="input-group-text"
for="HumiditySensorThingSpeakChannel">

                        {{ language.HiveSetupInterface.ThingSpeakChannel }}

                    </label>

                    <select class="form-select" id="HumiditySensorThingSpeakChannel"

                        name="FormHumiditySensorThingSpeakChannel">

                        <option></option>

                        {% for ThingSpeak in ThingSpeakChannels %}

                        <option value="{{ ThingSpeak.ChannelNumber }}"

                            {%    if    HumiditySensor.ThingSpeakChannelNumber    ==
ThingSpeak.ChannelNumber %} selected

                            {% endif %}>

                            {{ ThingSpeak.ChannelName }}

                        </option>

                        {% endfor %}

                    </select>

                </div>

                <div class="input-group mt-2">

                    <label                                    class="input-group-text"
for="HumiditySensorThingSpeakChannelField">

                        {{ language.HiveSetupInterface.ThingSpeakField }}

                    </label>

                    <select                                        class="form-select"
id="HumiditySensorThingSpeakChannelField"

                        name="FormHumiditySensorThingSpeakChannelField">
```

```
                        <option></option>

                        {% for Field in range(10) %}

                        <option {% if HumiditySensor.ThingSpeakChannelField == Field
+ 1 %} selected {% endif %}>

                            {{ Field + 1 }}

                        </option>

                        {% endfor %}

                    </select>

                </div>

                <div class="input-group mt-2">

                    <label class="input-group-text" for="HumiditySensorOffset">

                        {{ language.HiveSetupInterface.Offset }}

                    </label>

                    <input             oninput="InvalidOffset(this,             '{{
language.Alert.HumidityOffset.BadInput }}',

                                                          '{{
language.Alert.HumidityOffset.RangeUnderflow }}',

                                                          '{{
language.Alert.HumidityOffset.RangeOverflow }}',

                                                          '{{
language.Alert.HumidityOffset.StepMismatch }}')" step="0.1"

                        oninvalid="InvalidOffset(this,                           '{{
language.Alert.HumidityOffset.BadInput }}',

                                                         '{{
language.Alert.HumidityOffset.RangeUnderflow }}',

                                                         '{{
language.Alert.HumidityOffset.RangeOverflow }}',
```

```
                                                    '{{
language.Alert.HumidityOffset.StepMismatch }}')"

                        type="number"                       id="HumiditySensorOffset"
name="FormHumiditySensorOffset" class="form-control"

                        {% if not HumiditySensor.Offset %} value="0.0" {% else %}
value="{{ HumiditySensor.Offset }}"

                        {% endif %} min="0" max="100">

                </div>

                <div class="form-group d-grid mt-2">

                    <button type="submit" class="btn btn-success">

                        {{ language.HiveSetupInterface.Save }}

                    </button>

                </div>

            </form>

            <div class="d-grid mt-2">

                <a   href="/Execute/Humidity/{{   HiveNumber   }}/MeasureAM2302.py"
type="submit" class="btn btn-primary">

                    {{ language.HiveSetupInterface.Measure }}

                </a>

            </div>

            <div class="card mt-2 bg-dark">

                <div class="card-body">

                    <h4 class="text-center text-white">

                        {% if HumiditySensor.Measurement %}

                        {{ HumiditySensor.Measurement }} %
```

```
                        {% endif %}

                    </h4>

                </div>

            </div>

        </div>

    </div>

</div>

{% endmacro %}



{% macro FormWeight(language, HiveNumber, WeightSensor, ThingSpeakChannels, GPIO, ID)
%}

<div class="col-auto">

    <div class="card" id="Hive{{ HiveNumber }}Weight">

        <h5 class="card-tittle text-center mt-3">

            {{                language.HiveSetupInterface.Weight                }}[{{
language.AccordionHeader.Hive }}{{ HiveNumber }}]

        </h5>

        <div class="card-body">

            <form action="/EditWeightSensor/{{ HiveNumber }}" method="POST">

                <div class="form-check form-switch">

                    <input  class="form-check-input"  type="checkbox"  role="switch"
id="WeightSensorEnabled" value="1"

                        name="FormWeightSensorEnabled" {% if WeightSensor.Enabled %}
checked {% endif %}>

                    <input type="hidden" name="FormWeightSensorEnabled" value="0">
```

```
                    <label class="form-check-label" for="WeightSensorEnabled">
                        {{ language.HiveSetupInterface.SensorEnable }}
                    </label>
                </div>
                <div class="input-group mt-2">
                    <label class="input-group-text" for="WeightSensorGPIODT">
                        {{ language.HiveSetupInterface.HX711DT }}
                    </label>
                    <select class="form-select" id="WeightSensorGPIODT"
name="FormWeightSensorGPIODT">
                        <option></option>
                        {% if WeightSensor.GPIODT %}
                        <option selected>{{ WeightSensor.GPIODT }}</option>
                        {% endif %}
                        {% for CheckGPIO in GPIO %}
                        {% if CheckGPIO.GPIO != WeightSensor.GPIODT %}
                        <option>{{ CheckGPIO.GPIO }}</option>
                        {% endif %}
                        {% endfor %}
                    </select>
                </div>
                <div class="input-group mt-2">
                    <label class="input-group-text" for="WeightSensorGPIOSCK">
```

```
                        {{ language.HiveSetupInterface.HX711SCK }}

                    </label>

                    <select       class="form-select"       id="WeightSensorGPIOSCK"
name="FormWeightSensorGPIOSCK">

                        <option></option>

                        {% if WeightSensor.GPIOSCK %}

                        <option selected>{{ WeightSensor.GPIOSCK }}</option>

                        {% endif %}

                        {% for CheckGPIO in GPIO %}

                        {% if CheckGPIO.GPIO != WeightSensor.GPIOSCK %}

                        <option>{{ CheckGPIO.GPIO }}</option>

                        {% endif %}

                        {% endfor %}

                    </select>

                </div>

                <div class="input-group mt-2">

                    <label class="input-group-text" for="WeightSensorChannel">

                        {{ language.HiveSetupInterface.HX711Channel }}

                    </label>

                    <select       class="form-select"       id="WeightSensorChannel"
name="FormWeightSensorChannel">

                        <option>A</option>

                        <option {% if WeightSensor.Channel == 'B' %} selected {% endif
%}>
```

```
                            B

                        </option>

                    </select>

                </div>

                <div class="form-group mt-2 bg-secondary bg-opacity-25 rounded-2">

                    <label                                    class="input-group-text"
for="WeightSensorThingSpeakChannel">

                        {{ language.HiveSetupInterface.ThingSpeakChannel }}

                    </label>

                    <select class="form-select" id="WeightSensorThingSpeakChannel"

                        name="FormWeightSensorThingSpeakChannel">

                        <option></option>

                        {% for ThingSpeak in ThingSpeakChannels %}

                        <option value="{{ ThingSpeak.ChannelNumber }}"

                            {%     if     WeightSensor.ThingSpeakChannelNumber     ==
ThingSpeak.ChannelNumber %} selected

                            {% endif %}>

                            {{ ThingSpeak.ChannelName }}

                        </option>

                        {% endfor %}

                    </select>

                </div>

                <div class="input-group mt-2">
```

```
                    <label                                    class="input-group-text"
for="WeightSensorThingSpeakChannelField">

                        {{ language.HiveSetupInterface.ThingSpeakField }}

                    </label>

                    <select                                   class="form-select"
id="WeightSensorThingSpeakChannelField"

                        name="FormWeightSensorThingSpeakChannelField">

                        <option></option>

                        {% for Field in range(10) %}

                        <option {% if WeightSensor.ThingSpeakChannelField == Field +
1 %} selected {% endif %}>

                            {{ Field + 1 }}

                        </option>

                        {% endfor %}

                    </select>

                </div>

                <div class="input-group mt-2">

                    <label class="input-group-text" for="WeightSensorOffset">

                        {{ language.HiveSetupInterface.Offset }}

                    </label>

                    <input             oninput="InvalidOffset(this,             '{{
language.Alert.WeightOffset.BadInput }}',

                                                          '{{
language.Alert.WeightOffset.RangeUnderflow }}',

                                                          '{{
language.Alert.WeightOffset.RangeOverflow }}',
```

```
                                                    '{{
language.Alert.WeightOffset.StepMismatch }}')" step="0.1"

                        oninvalid="InvalidOffset(this,                          '{{
language.Alert.WeightOffset.BadInput }}',

                                                    '{{
language.Alert.WeightOffset.RangeUnderflow }}',

                                                    '{{
language.Alert.WeightOffset.RangeOverflow }}',

                                                    '{{
language.Alert.WeightOffset.StepMismatch }}')" type="number"

                        id="WeightSensorOffset"       name="FormWeightSensorOffset"
class="form-control"

                        {% if not WeightSensor.Offset %} value="0.0" {% else %}
value="{{ WeightSensor.Offset }}"

                        {% endif %} min="-8388608" max="8388607">

                </div>

                <div class="input-group mt-2">

                    <label class="input-group-text" for="WeightSensorScale">

                        {{ language.HiveSetupInterface.HX711Scale }}

                    </label>

                    <input             oninput="InvalidScale(this,             '{{
language.Alert.WeightScale.BadInput }}',

                                                    '{{
language.Alert.WeightScale.RangeUnderflow }}',

                                                    '{{
language.Alert.WeightScale.RangeOverflow }}',

                                                    '{{
language.Alert.WeightScale.StepMismatch }}')" step="0.1"

                        oninvalid="InvalidScale(this,                           '{{
language.Alert.WeightScale.BadInput }}',
```

```
                                                    '{{
language.Alert.WeightScale.RangeUnderflow }}',

                                                    '{{
language.Alert.WeightScale.RangeOverflow }}',

                                                    '{{
language.Alert.WeightScale.StepMismatch }}')" type="number"

                        id="WeightSensorScale"          name="FormWeightSensorScale"
class="form-control"

                        {% if not WeightSensor.Scale %} value="1.0" {% else %}
value="{{ WeightSensor.Scale }}"

                        {% endif %} min="-100000" max="100000">

                </div>

                <div class="input-group mt-2">

                    <label class="input-group-text" for="WeightTemperatureID">

                        {{ language.HiveSetupInterface.HX711TemperatureID }}

                    </label>

                    <select       class="form-select"       id="WeightTemperatureID"
name="FormWeightTemperatureID">

                        <option></option>

                        {% if WeightSensor.TemperatureID %}

                        <option selected>{{ WeightSensor.TemperatureID }}</option>

                        {% endif %}

                        {% for DS18B20 in ID %}

                        {% if WeightSensor.TemperatureID != DS18B20.SensorID %}

                        <option>{{ DS18B20.SensorID }}</option>

                        {% endif %}
```

```
                    {% endfor %}

                </select>

            </div>

            <div class="input-group mt-2">

                <label class="input-group-text" for="WeightTemperatureOffset">

                    {{ language.HiveSetupInterface.HX711TemperatureOffset }}

                </label>

                <input             oninput="InvalidOffset(this,             '{{
language.Alert.TemperatureOffset.BadInput }}',

                                                          '{{
language.Alert.TemperatureOffset.RangeUnderflow }}',

                                                          '{{
language.Alert.TemperatureOffset.RangeOverflow }}',

                                                          '{{
language.Alert.TemperatureOffset.StepMismatch }}')"

                    oninvalid="InvalidOffset(this,                            '{{
language.Alert.TemperatureOffset.BadInput }}',

                                                         '{{
language.Alert.TemperatureOffset.RangeUnderflow }}',

                                                         '{{
language.Alert.TemperatureOffset.RangeOverflow }}',

                                                         '{{
language.Alert.TemperatureOffset.StepMismatch }}')"

                    type="number"                  id="WeightTemperatureOffset"
name="FormWeightTemperatureOffset"

                    class="form-control"           {%          if          not
WeightSensor.TemperatureOffset %} value="0.0" {% else %}

                    value="{{  WeightSensor.TemperatureOffset  }}"  {%  endif  %}
min="-100" max="100" step="0.1">
```

```
                </div>

                <div class="input-group mt-2">

                    <label                                    class="input-group-text"
for="WeightTemperatureReference">

                        {{ language.HiveSetupInterface.HX711TemperatureReference }}

                    </label>

                    <input            oninput="InvalidReference(this,            '{{
language.Alert.TemperatureReference.BadInput }}',

                                                                '{{
language.Alert.TemperatureReference.RangeUnderflow }}',

                                                                '{{
language.Alert.TemperatureReference.RangeOverflow }}',

                                                                '{{
language.Alert.TemperatureReference.StepMismatch }}')"

                        oninvalid="InvalidReference(this,                          '{{
language.Alert.TemperatureReference.BadInput }}',

                                                               '{{
language.Alert.TemperatureReference.RangeUnderflow }}',

                                                               '{{
language.Alert.TemperatureReference.RangeOverflow }}',

                                                               '{{
language.Alert.TemperatureReference.StepMismatch }}')"

                        type="number"               id="WeightTemperatureReference"
name="FormWeightTemperatureReference"

                        class="form-control"          {%          if          not
WeightSensor.TemperatureReference %} value="0.0" {% else %}

                        value="{{ WeightSensor.TemperatureReference }}" {% endif %}
min="-100" max="100" step="0.1">

                </div>

                <div class="input-group mt-2">
```

```
                    <label                                    class="input-group-text"
for="WeightTemperatureCorrection">

                        {{ language.HiveSetupInterface.HX711TemperatureCorrection }}

                    </label>

                    <input           oninput="InvalidCorrection(this,           '{{
language.Alert.WeightCorrection.BadInput }}',

                                                                '{{
language.Alert.WeightCorrection.RangeUnderflow }}',

                                                                '{{
language.Alert.WeightCorrection.RangeOverflow }}',

                                                                '{{
language.Alert.WeightCorrection.StepMismatch }}')"

                        oninvalid="InvalidCorrection(this,                      '{{
language.Alert.WeightCorrection.BadInput }}',

                                                               '{{
language.Alert.WeightCorrection.RangeUnderflow }}',

                                                               '{{
language.Alert.WeightCorrection.RangeOverflow }}',

                                                               '{{
language.Alert.WeightCorrection.StepMismatch }}')"

                        type="number"               id="WeightTemperatureCorrection"
name="FormWeightTemperatureCorrection"

                        class="form-control"          {%          if          not
WeightSensor.TemperatureCorrection %} value="0.0" {% else %}

                        value="{{ WeightSensor.TemperatureCorrection }}" {% endif %}
min="-1000" max="1000" step="0.1">

                </div>

                <div class="form-group d-grid mt-2">

                    <button type="submit" class="btn btn-success">

                        {{ language.HiveSetupInterface.Save }}
```

```
                    </button>

                </div>

            </form>

            <div class="d-grid mt-2">

                <a    href="/Execute/Weight/{{    HiveNumber    }}/MeasureHX711.py"
type="submit" class="btn btn-primary">

                    {{ language.HiveSetupInterface.Measure }}

                </a>

            </div>

            <div class="card mt-2 bg-dark">

                <div class="card-body">

                    <h4 class="text-center text-white">

                        {% if WeightSensor.Measurement %}

                        {{ WeightSensor.Measurement }} kg

                        {% endif %}

                    </h4>

                </div>

            </div>

        </div>

    </div>

</div>

{% endmacro %}
```

A.5.10. Arxiu '/home/SweetHive/SetupInterface/SetupInterface.py'

```
#!/usr/bin/env python3



from flask import Flask, redirect, render_template, request, json, send_file

from flask_sqlalchemy import SQLAlchemy

from os.path import dirname, join

from os import listdir

from subprocess import run



UPLOAD_FOLDER = dirname(__file__) + '/database'



app = Flask(__name__)

app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///database/Settings.db'

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

db = SQLAlchemy(app)

MaxHives = 6

MaxThingSpeakChannels = 20

ScrollingTo = ''



class TS(db.Model):

    ChannelNumber = db.Column(db.INTEGER, primary_key = True)
```

```
    ChannelName = db.Column(db.TEXT)

    ChannelID = db.Column(db.TEXT)

    ChannelWriteAPIKey = db.Column(db.TEXT)



class Temperature(db.Model):

    Hive = db.Column(db.INTEGER, primary_key = True)

    Enabled = db.Column(db.INTEGER)

    ID = db.Column(db.TEXT)

    ThingSpeakChannelNumber = db.Column(db.INTEGER)

    ThingSpeakChannelField = db.Column(db.INTEGER)

    Offset = db.Column(db.REAL)

    Measurement = db.Column(db.REAL)

    Sent = db.Column(db.INTEGER)



class Humidity(db.Model):

    Hive = db.Column(db.INTEGER, primary_key = True)

    Enabled = db.Column(db.INTEGER)

    GPIO = db.Column(db.INTEGER)

    ThingSpeakChannelNumber = db.Column(db.INTEGER)

    ThingSpeakChannelField = db.Column(db.INTEGER)

    Offset = db.Column(db.REAL)

    Measurement = db.Column(db.REAL)
```

```
    Sent = db.Column(db.INTEGER)



class Weight(db.Model):

    Hive = db.Column(db.INTEGER, primary_key = True)

    Enabled = db.Column(db.INTEGER)

    GPIODT = db.Column(db.INTEGER)

    GPIOSCK = db.Column(db.INTEGER)

    Channel = db.Column(db.TEXT)

    ThingSpeakChannelNumber = db.Column(db.INTEGER)

    ThingSpeakChannelField = db.Column(db.INTEGER)

    Offset = db.Column(db.REAL)

    Scale = db.Column(db.REAL)

    TemperatureID = db.Column(db.TEXT)

    TemperatureOffset = db.Column(db.REAL)

    TemperatureReference = db.Column(db.REAL)

    TemperatureCorrection = db.Column(db.REAL)

    Measurement = db.Column(db.REAL)

    Sent = db.Column(db.INTEGER)



class W1(db.Model):

    PrimaryKey = db.Column(db.INTEGER, primary_key = True)

    SensorID = db.Column(db.TEXT)
```

```
class AvailableGPIO(db.Model):
    GPIO = db.Column(db.INTEGER, primary_key = True)
    Available = db.Column(db.INTEGER)

db.create_all()

def FillExpectedHives():
    for Hives in range(MaxHives):
        if not Temperature.query.get(Hives + 1):
            Sensor = Temperature(Hive = Hives + 1, Enabled = 0, Sent = 0)
            db.session.add(Sensor)
        if not Humidity.query.get(Hives + 1):
            Sensor = Humidity(Hive = Hives + 1, Enabled = 0, Sent = 0)
            db.session.add(Sensor)
        if not Weight.query.get(Hives + 1):
            Sensor = Weight(Hive = Hives + 1, Enabled = 0, Sent = 0)
            db.session.add(Sensor)
    db.session.commit()

def CheckAvailableGPIO():
    for GPIO in range(28):
```

```
        if GPIO not in [0, 1, 2, 3, 4, 7, 8, 14, 17, 27]:

            if not AvailableGPIO.query.get(GPIO):

                IO = AvailableGPIO(GPIO = GPIO)

                db.session.add(IO)

            IO = AvailableGPIO.query.get(GPIO)

            if Humidity.query.filter_by(Enabled = 1, GPIO = GPIO).first():

                IO.Available = 0

            elif Weight.query.filter_by(Enabled = 1, GPIODT = GPIO).first():

                IO.Available = 0

            elif Weight.query.filter_by(Enabled = 1, GPIOSCK = GPIO).first():

                IO.Available = 0

            else:

                IO.Available = 1

    db.session.commit()



LanguagesFiles = listdir(dirname(__file__) + '/language')

Languages = [Files.split('.')[0] for Files in LanguagesFiles]



try:

    with open(dirname(__file__) + '/database/SelectedLanguage.json', 'r', encoding =
'utf8') as language_file:

        language = json.load(language_file)

        language_file.close()
```

```
except:

    with open(dirname(__file__) + '/language/DefaultLanguage.json', 'r', encoding =
'utf8') as language_file:

        language = json.load(language_file)

        language_file.close()



Accordion = '{ "ThingSpeak": false, "Language": false, "Schedule": false,
"SettingsBackup": false'

for Heading in range(MaxHives):

    Accordion = Accordion + ', "Hive' + str(Heading + 1) + '": false'

Accordion = Accordion + ' }'

Accordion = json.loads(Accordion)



with open(dirname(__file__) + '/database/AccordionState.json', 'w') as
accordion_file:

    json.dump(Accordion, accordion_file)

    accordion_file.close()



try:

    Schedule = open(dirname(__file__).split('/Setup')[0] + '/wittypi/schedule.wpi',
'r').read()

    open(dirname(__file__) + '/database/schedule.wpi', 'w').write(Schedule)

except:

    try:
```

```
        ScheduleFile    =    open(dirname(__file__)    +    '/database/schedule.wpi',
'r').read().split()

        for Word in range(len(ScheduleFile)):

            if Word == 100:

                break

            elif Word == 0:

                Schedule = ScheduleFile[Word]

            elif ScheduleFile[Word] in ['END', 'ON', 'OFF', 'BEGIN']:

                Schedule = Schedule + '\n' + ScheduleFile[Word]

            else:

                Schedule = Schedule + '\t' + ScheduleFile[Word]

        open(dirname(__file__).split('/Setup')[0]     +     '/wittypi/schedule.wpi',
'w').write(Schedule)

    except:

        Schedule = ''

        open(dirname(__file__) + '/database/schedule.wpi', 'w').write(Schedule)



FillExpectedHives()

CheckAvailableGPIO()

ThingSpeakChannels = TS.query.all()

TemperatureSensors = Temperature.query.all()

HumiditySensors = Humidity.query.all()

WeightSensors = Weight.query.all()
```

```
GPIO = AvailableGPIO.query.all()

DS18B20SensorID = W1.query.all()



@app.route('/')

def index():

    return render_template('index.html',

                           MaxHives = MaxHives,

                           MaxThingSpeakChannels = MaxThingSpeakChannels,

                           ScrollingTo = ScrollingTo,

                           Languages = sorted(Languages),

                           language = language,

                           Accordion = Accordion,

                           ThingSpeakChannels = ThingSpeakChannels,

                           TemperatureSensors = TemperatureSensors,

                           HumiditySensors = HumiditySensors,

                           WeightSensors = WeightSensors,

                           GPIO = GPIO,

                           DS18B20SensorID = DS18B20SensorID,

                           Schedule = Schedule)



@app.route('/AddThingSpeakChannel', methods = ['POST'])

def AddThingSpeakChannel():
```

```
    global ScrollingTo, ThingSpeakChannels

    for LowestAvailablePrimaryKey in range(MaxThingSpeakChannels):

        if not TS.query.get(LowestAvailablePrimaryKey + 1):

            LowestAvailablePrimaryKey += 1

            break

    ScrollingTo = 'ThingSpeak' + str(LowestAvailablePrimaryKey)

    ThingSpeakChannel = TS(ChannelNumber = LowestAvailablePrimaryKey,

                           ChannelName = request.form['FormThingSpeakChannelName'],

                           ChannelID = request.form['FormThingSpeakChannelID'],

                           ChannelWriteAPIKey                                    =
request.form['FormThingSpeakChannelWriteAPIKey'])

    db.session.add(ThingSpeakChannel)

    db.session.commit()

    ThingSpeakChannels = TS.query.all()

    return redirect('/')



@app.route('/EditThingSpeakChannel/<ThingSpeakChannelNumber>', methods = ['POST'])

def EditThingSpeakChannel(ThingSpeakChannelNumber):

    global  ScrollingTo,  ThingSpeakChannels,  TemperatureSensors,  HumiditySensors,
WeightSensors

    ScrollingTo = 'ThingSpeak' + str(ThingSpeakChannelNumber)

    ThingSpeakChannel = TS.query.get(ThingSpeakChannelNumber)

    ThingSpeakChannel.ChannelName = request.form['FormThingSpeakChannelName']
```

```
    ThingSpeakChannel.ChannelID = request.form['FormThingSpeakChannelID']

    ThingSpeakChannel.ChannelWriteAPIKey = request.form['FormThingSpeakChannelWriteAPIKey']

    for Hives in range(MaxHives):

        TemperatureSensor = Temperature.query.get(Hives + 1)

        if TemperatureSensor.ThingSpeakChannelNumber == ThingSpeakChannel.ChannelNumber:

            TemperatureSensor.Sent = 0

        HumiditySensor = Humidity.query.get(Hives + 1)

        if HumiditySensor.ThingSpeakChannelNumber == ThingSpeakChannel.ChannelNumber:

            HumiditySensor.Sent = 0

        WeightSensor = Weight.query.get(Hives + 1)

        if WeightSensor.ThingSpeakChannelNumber == ThingSpeakChannel.ChannelNumber:

            WeightSensor.Sent = 0

    db.session.commit()

    ThingSpeakChannels = TS.query.all()

    TemperatureSensors = Temperature.query.all()

    HumiditySensors = Humidity.query.all()

    WeightSensors = Weight.query.all()

    return redirect('/')



@app.route('/DeleteThingSpeakChannel/<ThingSpeakChannelNumber>')

def DeleteThingSpeakChannel(ThingSpeakChannelNumber):
```

```
    global ScrollingTo, ThingSpeakChannels, TemperatureSensors, HumiditySensors,
WeightSensors, GPIO

    ScrollingTo = 'HeadingThingSpeak'

    ThingSpeakChannel = TS.query.get(ThingSpeakChannelNumber)

    for Hives in range(MaxHives):

        TemperatureSensor = Temperature.query.get(Hives + 1)

        if TemperatureSensor.ThingSpeakChannelNumber ==
ThingSpeakChannel.ChannelNumber:

            TemperatureSensor.Enabled = 0

            TemperatureSensor.ThingSpeakChannelNumber = None

            TemperatureSensor.Sent = 0

        HumiditySensor = Humidity.query.get(Hives + 1)

        if HumiditySensor.ThingSpeakChannelNumber ==
ThingSpeakChannel.ChannelNumber:

            HumiditySensor.Enabled = 0

            HumiditySensor.ThingSpeakChannelNumber = None

            HumiditySensor.Sent = 0

        WeightSensor = Weight.query.get(Hives + 1)

        if WeightSensor.ThingSpeakChannelNumber == ThingSpeakChannel.ChannelNumber:

            WeightSensor.Enabled = 0

            WeightSensor.ThingSpeakChannelNumber = None

            WeightSensor.Sent = 0

    db.session.delete(ThingSpeakChannel)

    db.session.commit()
```

```
    CheckAvailableGPIO()

    ThingSpeakChannels = TS.query.all()

    TemperatureSensors = Temperature.query.all()

    HumiditySensors = Humidity.query.all()

    WeightSensors = Weight.query.all()

    GPIO = AvailableGPIO.query.all()

    return redirect('/')



@app.route('/EditTemperatureSensor/<Hive>', methods = ['POST'])

def EditTemperatureSensor(Hive):

    global ScrollingTo, TemperatureSensors

    ScrollingTo = 'Hive' + str(Hive) + 'Temperature'

    TemperatureSensor = Temperature.query.get(Hive)

    TemperatureSensor.Enabled = request.form['FormTemperatureSensorEnabled']

    TemperatureSensor.ID = request.form['FormTemperatureSensorID']

    TemperatureSensor.ThingSpeakChannelNumber                                    =
request.form['FormTemperatureSensorThingSpeakChannel']

    TemperatureSensor.ThingSpeakChannelField                                     =
request.form['FormTemperatureSensorThingSpeakChannelField']

    if request.form['FormTemperatureSensorOffset']:

        TemperatureSensor.Offset = request.form['FormTemperatureSensorOffset']

    else:

        TemperatureSensor.Offset = 0.0
```

```
    TemperatureSensor.Measurement = None

    TemperatureSensor.Sent = 0

    db.session.commit()

    TemperatureSensors = Temperature.query.all()

    return redirect('/')



@app.route('/EditHumiditySensor/<Hive>', methods = ['POST'])

def EditHumiditySensor(Hive):

    global ScrollingTo, HumiditySensors, WeightSensors, GPIO

    ScrollingTo = 'Hive' + str(Hive) + 'Humidity'

    HumiditySensor = Humidity.query.get(Hive)

    HumiditySensor.Enabled = request.form['FormHumiditySensorEnabled']

    HumiditySensor.GPIO = request.form['FormHumiditySensorGPIO']

    HumiditySensor.ThingSpeakChannelNumber                                    =
request.form['FormHumiditySensorThingSpeakChannel']

    HumiditySensor.ThingSpeakChannelField                                     =
request.form['FormHumiditySensorThingSpeakChannelField']

    if request.form['FormHumiditySensorOffset']:

        HumiditySensor.Offset = request.form['FormHumiditySensorOffset']

    else:

        HumiditySensor.Offset = 0.0

    HumiditySensor.Measurement = None

    HumiditySensor.Sent = 0
```

```
    db.session.commit()
    if HumiditySensor.Enabled:
        GPIO = HumiditySensor.GPIO
        Hive = HumiditySensor.Hive
        for Hives in range(MaxHives):
            if Hives + 1 != Hive:
                HumiditySensor = Humidity.query.get(Hives + 1)
                if HumiditySensor.GPIO == GPIO:
                    HumiditySensor.GPIO = None
            WeightSensor = Weight.query.get(Hives + 1)
            if WeightSensor.GPIODT == GPIO:
                WeightSensor.GPIODT = None
            if WeightSensor.GPIOSCK == GPIO:
                WeightSensor.GPIOSCK = None
        db.session.commit()
    CheckAvailableGPIO()
    HumiditySensors = Humidity.query.all()
    WeightSensors = Weight.query.all()
    GPIO = AvailableGPIO.query.all()
    return redirect('/')


@app.route('/EditWeightSensor/<Hive>', methods = ['POST'])
```

```
def EditWeightSensor(Hive):

    global ScrollingTo, HumiditySensors, WeightSensors, GPIO

    ScrollingTo = 'Hive' + str(Hive) + 'Weight'

    WeightSensor = Weight.query.get(Hive)

    WeightSensor.Enabled = request.form['FormWeightSensorEnabled']

    WeightSensor.GPIODT = request.form['FormWeightSensorGPIODT']

    WeightSensor.GPIOSCK = request.form['FormWeightSensorGPIOSCK']

    WeightSensor.Channel = request.form['FormWeightSensorChannel']

    WeightSensor.ThingSpeakChannelNumber                                    =
request.form['FormWeightSensorThingSpeakChannel']

    WeightSensor.ThingSpeakChannelField                                     =
request.form['FormWeightSensorThingSpeakChannelField']

    if request.form['FormWeightSensorOffset']:

        WeightSensor.Offset = request.form['FormWeightSensorOffset']

    else:

        WeightSensor.Offset = 0.0

    if request.form['FormWeightSensorScale']:

        WeightSensor.Scale = request.form['FormWeightSensorScale']

    else:

        WeightSensor.Scale = 1.0

    WeightSensor.TemperatureID = request.form['FormWeightTemperatureID']

    if request.form['FormWeightTemperatureOffset']:

        WeightSensor.TemperatureOffset = request.form['FormWeightTemperatureOffset']
```

```
    else:

        WeightSensor.TemperatureOffset = 0.0

    if request.form['FormWeightTemperatureReference']:

        WeightSensor.TemperatureReference                                    =
request.form['FormWeightTemperatureReference']

    else:

        WeightSensor.TemperatureReference = 0.0

    if request.form['FormWeightTemperatureCorrection']:

        WeightSensor.TemperatureCorrection                                   =
request.form['FormWeightTemperatureCorrection']

    else:

        WeightSensor.TemperatureCorrection = 0.0

    WeightSensor.Measurement = None

    WeightSensor.Sent = 0

    db.session.commit()

    if WeightSensor.Enabled:

        GPIODT = WeightSensor.GPIODT

        GPIOSCK = WeightSensor.GPIOSCK

        Hive = WeightSensor.Hive

        for Hives in range(MaxHives):

            HumiditySensor = Humidity.query.get(Hives + 1)

            if HumiditySensor.GPIO == GPIODT:

                HumiditySensor.GPIO = None
```

```
            elif HumiditySensor.GPIO == GPIOSCK:
                HumiditySensor.GPIO = None
            if Hives + 1 != Hive:
                WeightSensor = Weight.query.get(Hives + 1)
                if WeightSensor.GPIODT == GPIODT:
                    WeightSensor.GPIODT = None
                elif WeightSensor.GPIODT == GPIOSCK:
                    WeightSensor.GPIODT = None
                if WeightSensor.GPIOSCK == GPIODT:
                    WeightSensor.GPIOSCK = None
                elif WeightSensor.GPIOSCK == GPIOSCK:
                    WeightSensor.GPIOSCK = None
        db.session.commit()
    CheckAvailableGPIO()
    HumiditySensors = Humidity.query.all()
    WeightSensors = Weight.query.all()
    GPIO = AvailableGPIO.query.all()
    return redirect('/')


@app.route('/UpdateAccordionPanels/<Panel>')
def UpdateAccordionPanels(Panel):
    global ScrollingTo, Accordion
```

```
    ScrollingTo = 'Heading' + str(Panel)

    with open(dirname(__file__) + '/database/AccordionState.json', 'r+') as
accordion_file:

        Accordion = json.load(accordion_file)

        Accordion[Panel] = not Accordion[Panel]

        accordion_file.seek(0)

        json.dump(Accordion, accordion_file)

        accordion_file.truncate()

        accordion_file.close()

    return redirect('/')



@app.route('/ChangeLanguage/<Language>')

def ChangeLanguage(Language):

    global ScrollingTo, language, Accordion

    ScrollingTo = ''

    with open(dirname(__file__) + '/language/' + Language + '.json', 'r', encoding =
'utf8') as language_file:

        language = json.load(language_file)

        language_file.close()

    with open(dirname(__file__) + '/database/SelectedLanguage.json', 'w', encoding =
'utf8') as language_file:

        json.dump(language, language_file)

        language_file.close()

    with open(dirname(__file__) + '/database/AccordionState.json', 'r+') as
accordion_file:
```

```
        Accordion = json.load(accordion_file)

        Accordion['Language'] = not Accordion['Language']

        accordion_file.seek(0)

        json.dump(Accordion, accordion_file)

        accordion_file.truncate()

        accordion_file.close()

    return redirect('/')



@app.route('/Execute/<Variable>/<Hive>/<Script>')

def Execute(Variable, Hive, Script):

    global ScrollingTo, DS18B20SensorID, TemperatureSensors, HumiditySensors,
WeightSensors

    ScrollingTo = 'Hive' + str(Hive) + Variable

    ScriptDirectory = dirname(__file__).split('/Setup')[0] + '/MeasurementScripts/'
+ Script

    run(["sudo", "python", ScriptDirectory, Hive])

    if Variable == 'Temperature':

        if Script == 'SearchDS18B20ID.py':

            DS18B20SensorID = W1.query.all()

        else:

            TemperatureSensors = Temperature.query.all()

    if Variable == 'Humidity':

        HumiditySensors = Humidity.query.all()
```

```
    if Variable == 'Weight':

        WeightSensors = Weight.query.all()

    return redirect('/')



@app.route('/ChangeSchedule', methods = ['POST'])

def ChangeSchedule():

    global ScrollingTo, Schedule

    ScrollingTo = 'HeadingSchedule'

    ScheduleFile = request.form['FormScheduleRegister'].split()

    for Word in range(len(ScheduleFile)):

        if Word == 100:

            break

        elif Word == 0:

            Schedule = ScheduleFile[Word]

        elif ScheduleFile[Word] in ['END', 'ON', 'OFF', 'BEGIN']:

            Schedule = Schedule + '\n' + ScheduleFile[Word]

        else:

            Schedule = Schedule + '\t' + ScheduleFile[Word]

    open(dirname(__file__) + '/database/schedule.wpi', 'w').write(Schedule)

    open(dirname(__file__).split('/Setup')[0]       +       '/wittypi/schedule.wpi',
'w').write(Schedule)

    return redirect('/')
```

```
@app.route('/Download/<File>')

def Download(File):

    Path = dirname(__file__) + '/database/' + File

    return send_file(Path, as_attachment=True)



@app.route('/UploadSchedule', methods = ['POST'])

def UploadSchedule():

    global ScrollingTo, Schedule

    ScrollingTo = 'HeadingSchedule'

    if 'FormScheduleUpload' not in request.files:

        return redirect('/')

    ScheduleFile = request.files['FormScheduleUpload']

    if ScheduleFile:

        ScheduleFile.save(join(app.config['UPLOAD_FOLDER'], 'schedule.wpi'))

        ScheduleFile    =    open(dirname(__file__)    +    '/database/schedule.wpi',
'r').read().split()

        for Word in range(len(ScheduleFile)):

            if Word == 100:

                break

            elif Word == 0:

                Schedule = ScheduleFile[Word]

            elif ScheduleFile[Word] in ['END', 'ON', 'OFF', 'BEGIN']:

                Schedule = Schedule + '\n' + ScheduleFile[Word]
```

```
            else:

                Schedule = Schedule + '\t' + ScheduleFile[Word]

    return redirect('/')



@app.route('/UploadSettingsBackup', methods = ['POST'])

def UploadSettingsBackup():

    global ScrollingTo, ThingSpeakChannels, TemperatureSensors, HumiditySensors,
WeightSensors, GPIO, DS18B20SensorID, Accordion

    ScrollingTo = 'HeadingSettingsBackup'

    if 'FormSettingsBackupUpload' not in request.files:

        return redirect('/')

    SettingsBackup = request.files['FormSettingsBackupUpload']

    if SettingsBackup:

        SettingsBackup.save(join(app.config['UPLOAD_FOLDER'], 'Settings.db'))

        db.create_all()

        FillExpectedHives()

        CheckAvailableGPIO()

        ThingSpeakChannels = TS.query.all()

        TemperatureSensors = Temperature.query.all()

        HumiditySensors = Humidity.query.all()

        WeightSensors = Weight.query.all()

        GPIO = AvailableGPIO.query.all()

        DS18B20SensorID = W1.query.all()
```

```
        with open(dirname(__file__) + '/database/AccordionState.json', 'r+') as
accordion_file:

            Accordion = json.load(accordion_file)

            Accordion['SettingsBackup'] = not Accordion['SettingsBackup']

            accordion_file.seek(0)

            json.dump(Accordion, accordion_file)

            accordion_file.truncate()

            accordion_file.close()

        ScrollingTo = ''

    return redirect('/')



if __name__ == '__main__':

    app.run(debug = True)
```

A.6. Carpeta '/home/SweetHive/wittypi'

En aquesta carpeta s'emmagatzema l'arxiu "schedule.wpi" que conté la seqüencia d'inici i aturada del sistema. La programació següent és la predeterminada per a la monitorització d'un sol rusc.

```
BEGIN  2022-01-01   00:00:00

END    2029-12-31   23:59:59

ON     M5     WAIT

OFF    M5
```

B. CÀLCULS

B.1. Càlcul de la bateria

En aquest apartat es farà el càlcul de la capacitat adient de la bateria per a que amb una sola càrrega pugui mantenir el sistema encastat en funcionament durant 4 dies sense cap problema. En la següent taula s'organitzen tots els elements del sistema que tenen qualque consum d'energia, la seva alimentació, el seu consum màxim, el nombre d'elements que poden ser instal·lats com a màxim i la potència màxima total del conjunt de cada element.

Element	Alimentació (V)	Intensitat màxima (mA)	Nombre d'elements	Potència màxima total (mW)
Regulador de càrrega solar	12	13,0	1	156,0
Raspberry Pi Zero WH	5	500,0	1	2.500,0
Witty Pi 3 Mini	5	6,0	1	30,0
Huawei E3531	5	600,0	1	3.000,0
LED 5V	5	20,0	1	100,0
DS18B20	5	1,5	7	52,5
DHT22	5	1,5	6	45,0
HX711	5	1,5	6	45,0
Pont de Wheatstone	5	2,5	6	75,0

Taula 5. Elements amb consum energètic del sistema

Per a conèixer la seva intensitat màxima s'ha consultat el datasheet de cada un ells, i la potència s'ha calculat en funció de la següent equació, que es basa amb la llei de Watt.

$$P = \sum V * I = V * I * nº \text{ d'elements} \qquad \text{(Eq. 1)}$$

Dels elements a la taula, els únic que tindran un consum les 24 hores del dia són el regulador de càrrega solar i el mòdul Witty Pi 3 Mini. La particularitat del mòdul Witty Pi és que quan el sistema romaní apagat tan sols s'ha de contemplar un intensitat en repòs de 1mA, que amb els 5V d'alimentació signifiquen 5mW durant tot el dia per la parts seva. Amb aquests dos elements tenim una potència màxima de 161mW que haurem de preveure com a constant. Per a obtenir l'energia consumida tan sols fa falta multiplicar la potència per les hores de consum. En aquest, per a 24 hores es consumiran 3,86Wh al final de cada dia aproximadament.

Pel conjunt d'elements que no romandran en funcionament les 24 hores del dia tenim una potència total màxima aproximada de 5,85W (la suma de totes les potències màximes, incloent la del Witty Pi 3 Mini). Aquesta potència pot ser demandada des de que el sistema es posa en marxa fins que entra en suspensió. En l'apartat B.2 d'aquest projecte s'aconsella que amb la monitorització de 6 ruscs no es sobrepassin les 2 mesures per hora, degut a la optimització de l'ús de dades d'internet. El sistema tarda 1 minut en inicialitzar-se, cada mesura completa d'un rusc triga aproximadament 1,5 minuts i finalment s'utilitzen 30 segons per a la suspensió. Per a 6 ruscs això ens dona un total de 10,5 minuts per a un cicle complert, que serien 21 minuts cada hora (dues mesures cada hora), i 8,4 hores al final del dia (24 mesures). Amb la potència de 5,85W que s'ha calculat abans i aquestes 8,4 hores d'ús al final del dia, tindríem un consum de 49,14Wh. Juntament amb el consum constant calculat en l'anterior paràgraf, el sistema en la seva totalitat tindria un consum màxim diari de 53Wh.

Situació	Potència màxima (W)	Hores de consum diàries	Consum diari màxim (Wh)
Sistema en repòs	0,161	24,0	3,86
Sistema en mesura	5,850	8,4	49,14
		Total	53,00

Taula 6. Resum dels consums del sistema

La bateria, per a poder mantenir el sistema en funcionament 4 dies, sense rebre cap càrrega, haurà de ser capaç d'emmagatzemar 212Wh d'energia. Les bateries que es poden connectar al regulador de càrrega han de ser de 12V. Amb la potència i el voltatge podem calcular la capacitat en Ah de la bateria utilitzant l'equació 2. El resultat és que serà necessària una bateria de 12V amb una capacitat mínima de 17,67Ah.

$$C\ (Ah) = E\ (Wh) / V\ (V) \qquad \text{(Eq. 2)}$$

B.2. Càlcul del consum de dades d'internet

Per a saber el consum de dades d'internet del sistema s'ha fet ús de les característiques del mòdem Huawei E3531. Aquest mòdem, com la majoria, permeten emmagatzemar la quantitat de dades consumes en un període de temps. S'ha utilitzat el mòdem en un ordinador i s'ha entrat a la seva finestra de configuració, s'ha netejat el comptador de dades i després s'ha tornat a connectar al sistema encastat.

El sistema ha realitzat 6 cicles complerts en els quals ha llegit els sensors per a un rusc i ha enviat la informació a ThingSpeak. Després d'aquest procés, s'ha tornat a connectar el mòdem al computador, on en la seva finestra de configuració se'ns indica que hem consumit 337kB. Si aquest consum el dividim entre 6, tenim que en cada cicle per a un únic rusc tindrem un consum mitjà de 56,16kB (d'aquí en endavant s'arrodonirà a 60kB).

Un cop coneguda la informació anterior, s'ha netejat el comptador un altre cop i s'han realitzat uns altres 6 cicles del sistema, però aquest cop s'han desconnectat els sensors. Aquests darrers 6 cicles s'han realitzat per a saber el consum que té el mòdem per a establir connexió amb la xarxa telefònica i la sincronització de l'hora del mòdul Witty Pi 3 Mini. El resultat que ens proporciona el mòdem és de 103kB en total, que es tradueix a un consum mitjà de 17,16kB (d'aquí en endavant s'arrodonirà a 20kB) del propi sistema, sense l'enviament de mesures.

Com a resultat, tenim que dels 60kB inicials, 20kB són un consum base en un cicle i els 40kB és el resultat directe d'haver monitoritzat un rusc en el mateix cicle. Pel consum de dades tenim un límit mensual de 500MB, per tant seria interessant construir una equació que ens permeti saber el consum que realitzarem a final de mes en funció del nombre de ruscs a monitoritzar i el cicles que es vulguin realitzar cada hora.

$$\text{Dades mensuals (MB)} = (0{,}02 + 0{,}04 * \text{nº ruscs}) * \text{nº cicles/h} * 24 * 31 \qquad \text{(Eq. 3)}$$

Amb l'equació anterior tenim una forma senzilla de preveure el consum mensual d'internet que tindrà el sistema i saber si excedirem el límit de la tarifa amb la configuració que s'ha desitjat que tingui el dispositiu. Però hi ha un problema, el límit de dades mensuals es un valor que sempre coneixerem degut a la nostra tarifa, i el nombre de ruscs tampoc serà una incògnita al moment d'aplicar aquesta fórmula. El que realment ens interessa saber és el nombre màxim de cicles que podrem realitzar cada hora sense superar el límit de dades. Per a aconseguir això, aïllarem nº cicles/h de l'equació 3.

$$\text{nº}\frac{\text{cicles}}{\text{h}} < \frac{\text{Límit mensual dades (MB)}}{(0{,}02 + 0{,}04 * \text{nº ruscs}) * 24 * 31} \qquad \text{(Eq. 4)}$$

Si acomodem l'equació, el resultat final és el següent:

$$\text{nº}\frac{\text{cicles}}{\text{h}} < \frac{\text{Límit mensual dades (MB)}}{14{,}88 + 29{,}76 * \text{nº ruscs}} \qquad \text{(Eq. 5)}$$

Si ens posicionem en l'escenari amb el consum més alt (6 ruscs) i tenim com a límit els 500MB mensuals, utilitzant l'equació 5 sabrem que hauríem de realitzar menys de 2,58 cicles cada hora. Això vol dir que amb 6 ruscs haurem de programar el dispositiu per a que realitzi com a molt dues monitoritzacions cada hora (si volem realitzar les mesures de forma simètrica, seria una cada 30 minuts). Si amb aquesta configuració volem saber el consum aproximat a final de mes, podem reutilitzar l'equació 3 amb sis ruscs als sistema i dos cicles cada hora. Com a resultat final consumirem aproximadament 386,88MB cada mes, marge suficient que ens assegura no tenir sobrecost en la factura de l'operadora.