UNIVERSITAT POLITÈCNICA DE CATALUNYA

Control i monitorització d'una peixera.

14 d'octubre de 2016

Memòria del projecte que presenta Jordi Torres Farràs
sota la direcció del Dr. Eng. Pere Palà Schönwälder
i co-direcció d'Alexis López Riera
per assolir el grau d'Enginyer en Sistemes TIC.

Índex

Índex de figures

Abstract

This project describes and shows a system that monitors and controls a fish tank. I've been eager for this project for many years and The Bachelor Final Project has been the perfect excuse to carry it out without buying a specific commercial product for it. The goals of the project are to improve water quality and reduce the maintenance works. The project has been carried out while the fish tank was in operation. This fact hindered the works at some points since every work had to be revised for the proper functioning of the fish tank. Eventually, I've been able to monitor the fish tank and control it effectively, however, it still can be improved: the system could be expanded by either devices or software improvements.

Resum

Aquest projecte descriu i mostra un sistema que monitoritza i controla una peixera. Des de fa anys que tenia ganes de fer aquest projecte i el treball final de grau ha sigut l'excusa per poder-lo dur a terme sense haver de comprar un producte comercial específic per això. Els objectius marcats són millorar la qualitat de l'aigua i reduir la feina de manteniment. El projecte s'ha dut a terme mentre l'aquari estava en funcionament, cosa que ha dificultat en alguns moments la feina ja que en tot moment s'havia de revisar la feina feta perquè no interferís en el funcionament adequat de la peixera. Finalment he aconseguit monitoritzar-la i controlar-la adequadament tot i que queda marge per la millora, es pot ampliar tan amb instruments com amb millores de software.

Part I.

Memòria

1. Introducció

Des de petit que tinc aquari a casa, vaig començar amb el típic aquari rodó amb peixos taronges i a poc a poc, amb l'edat, m'hi he anat engrescant. Ara fa 8 anys que tinc la peixera actual i any rere any he anat fent-la més autònoma i més estètica visualment. Al principi comprava components comercials ja preparats per peixera i darrerament ja m'ho he començat a fer jo. El fet de confeccionar-me jo mateix els components té doble recompensa, ja que m'agrada molt fer-los i a més, econòmicament surt més barat. El projecte de fi de grau ha sigut l'excusa perfecte per poder començar un projecte que no tindrà fi per automatitzar la peixera, ja que pràcticament no hi ha opcions comercials o econòmicament no m'ho puc permetre.

1.1. Objectius

Amb aquest treball em proposo assolir els següents objectius:

- Fer la peixera més autònoma: Que la peixera sigui capaç d'autoregular els paràmetres sense necessitar una atenció constant.
- Reduir la feina de manteniment diari: L'aquari necessita un manteniment constant ja que es vital per a la supervivència dels individus que hi viuen.
- Anticipar-me als canvis de paràmetres i controlar la peixera a distància: Algun cop que no era a casa, quan he arribat m'he trobat la peixera amb els paràmetres malament o sense llum. Vull poder saber que passa en tot moment i poder fer algun canvi al respecte sense necessitat d'estar a casa per fer-ho.
- Aconseguir un sistema versàtil i que es pugui ampliar fàcilment.
- Aplicar tot el que he aprés durant el grau per fer un únic projecte.

1.2. L'aquariofília

L'aquariofília moderna és l'afició a la cria de peixos i altres organismes aquàtics en un aquari, sota condicions controlades. Ha evolucionat molt al llarg dels segles, des del manteniment de carpes daurades amb fins ornamentals en recipients i estanys, des de fa 2000 anys. Existeixen referències en l'antiga Xina sobre la cria de peixos daurats en dipòsits ceràmics, a temperatura ambient. El nivell de l'aquariofília era totalment bàsic, sense sistemes de suport de vida per als peixos, que requerien espècies resistents i constants canvis d'aigua. Aquest sistema arcaic es va perpetuar fins als nostres dies a través de les peixeres, i es va superar amb el desenvolupament dels aquaris moderns. Actualment l'Aquariofília és una afició que pot arribar a alts nivells de coneixement i sofisticació, que traspassen la frontera d'afició per esdevenir una veritable ciència.

Segons els peixos es poden classificar en varis tipus d'aquariofília:

- Els carpins daurats i les carpes koi[1.1]: ve dels inicis de l'aquariofília però encara es practica en molts estanys particulars arreu del mon.

Figura 1.1.: Estany de carpes Koi (foto pròpia).

- Els peixos tropicals[1.2]: són peixos i plantes importats dels països tropicals, són d'aigua dolça i calenta (24-28^{o}C).
- L'aquariofília Marina[1.3]: són peixos i corals marins els quals és mantenen amb aigua salada i calenta.

El principi bàsic de l'aquariofília moderna és la recreació d'un ecosistema aquàtic artificial en el qual puguin desenvolupar un comportament natural tot tipus d'espècies aquàtiques, i estabilitzat a través de sistemes tècnics auxiliars. Ja no és una afició centrada en el manteniment exclusiu de peixos, sinó una afició basada en una ciència, l'aquariologia. Existeixen molts aquaris sense peixos i creats específicament per a plantes aquàtiques, invertebrats, amfibis i rèptils aquàtics. Als Estats Units l'aquariofília ocupa el tercer lloc en l'ordre dels passatemps més practicats (el primer és la fotografia i el segon la filatèlia). Al Japó hi ha prop d'1,2 milions de aquaristes (els japonesos creuen que els aquaris "porten sort"). Al Brasil hi ha més de 500.000 aquaris. Extret de Viquipèdia [Viq16a]

El projecte que desenvolupo està implementat en un aquari tropical d'aigua dolça[1.2].

1.2.1. El funcionament general d'una peixera

Un aquari ha de tenir uns elements bàsics per funcionar, que són els següents:

- El filtre [1.4]: Ha de recollir l'aigua de la peixera, filtrar-la i tornar-la a tirar de tal manera que circuli i no deixi cap zona del aquari sense circular, per evitar que es faci mal bé i no es renovi.

Figura 1.2.: Aquari tropical (foto pròpia).

- El radiador [1.5]: Escalfa l'aigua del aquari i s'ha de posar de tal manera que l'aigua que mou el filtre també passi pel radiador, ja que així té un millor rendiment i no queden zones amb molta diferència de temperatura. El mateix escalfador té un regulador en el qual pots aconseguir la temperatura desitjada.
- La il·luminació [1.6]: L'aquari ha d'estar il·luminat perquè les plantes puguin sobreviure però a la vegada que no surtin algues. Per això no es bo que hi toqui la llum del sol ja que fa que hi creixin molt fàcilment les algues. Els peixos també necessiten llum però amb l'ambiental en tenen suficient.

Tot i que amb aquests elements ja hi poden viure les plantes i els peixos, també n'hi han d'altres que ajuden a mantenir els paràmetres de l'aigua i aconsegueixen una qualitat que necessitaria molta dedicació. Els paràmetres més importants són el pH, la duresa, gH i el manteniment del nivell de l'aigua ja que s'evapora i se'n hi ha d'anar afegint. Per regular la duresa es qüestió de barrejar aigua destil·lada o d'osmosis, amb gH=0 i aigua de l'aixeta, que té dureses superiors a 12. En canvi el pH es pot regular tirant productes reguladors i/o CO2. En el meu cas, utilitzo CO2 ja que és molt més senzill de regular i el pots mantenir més o menys constant, a la vegada que també és adob per les plantes. Els elements secundaris que utilitzo són:

- Equip osmosis [1.7]: Consta de varis filtres que acaben deixant l'aigua sense pràcticament sals minerals. És ideal per barrejar amb aigua de l'aixeta per tal d'aconseguir el nivell de gH que es vulgui.
- Sonda de pH [1.8] i equip de CO2 [1.9][1.10]: Amb la sonda sabem el pH que hi ha a l'aigua i segons si el volem baixar o pujar, afegim més o menys CO2. Un augment de CO2 fa baixar el pH. S'ha de tenir en compte que el CO2 en grans quantitats és nociu pels peixos ja que no podrien respirar. Per dispersar el CO2 per la peixera hi ha varies possibilitats, jo utilitzo un atomitzador col·locat a la sortia del filtre.
- Detector de nivell i bomba per mantenir-lo: Consta d'una boia que acciona una bomba dintre d'una garrafa de 25 litres prèviament omplerta amb aigua d'osmosis.
- Termòmetre o sensor de temperatura [1.11]: Serveix per saber la temperatura de l'aigua. Amb el sensor connectat al sistema podem anar enregistrant els canvis que fa. També es

Figura 1.3.: Aquari marí (Imatges de google).

pot connectar el radiador al sistema i regular la temperatura sense fer servir el regulador del propi radiador.

1.3. El mercat actual

Al mercat hi ha múltiples varietats de productes per regular els paràmetres d'un aquari. Alguns dels que poden servir d'exemple són els següents:

- Kit Profilux 3.IT Profiset PB [Aqu16], inclou: ProfiLux Plus III, adaptador europeu de corrent AC/DC, base d'endolls analògica de 6 canals (tipus Schuko), vàlvula electromagnètica paer CO2, elèctrode de pH, sensor de temperatura, Cable RS- 232 para la connexió al PC. Aquest seria semblant al meu però amb un preu de 270 euros.
- Regulador medidor de pH [Ser16], inclou: regulador, sonda, controlador de sortida d'energia. Aquest només permet regular el pH per 89,90 euros.
- AT Control System Aqua Medic [Tro16], inclou: l'ordinador i un sensor de temperatura, però a base de components es pot ampliar. A part de sondes de pH, conductivitat, densitat... també, entre altres perifèrics, hi ha una interfície per internet on permet consultar els valors i també una altre per SMS on et pot enviar missatges de les alertes. El pack bàsic val 593,5 euros.

1.4. L'estructura del document

En aquest document explico que he fet servir i com ho he fet per poder monitoritzar i controlar la meva peixera. Començo explicant en el segon capítol els components utilitzats, després l'organització d'aquests en el tercer i seguidament en el quart, com estan programats. En la darrera part del projecte, al cinquè capítol, explico com està integrat el sistema i al sisè, quina

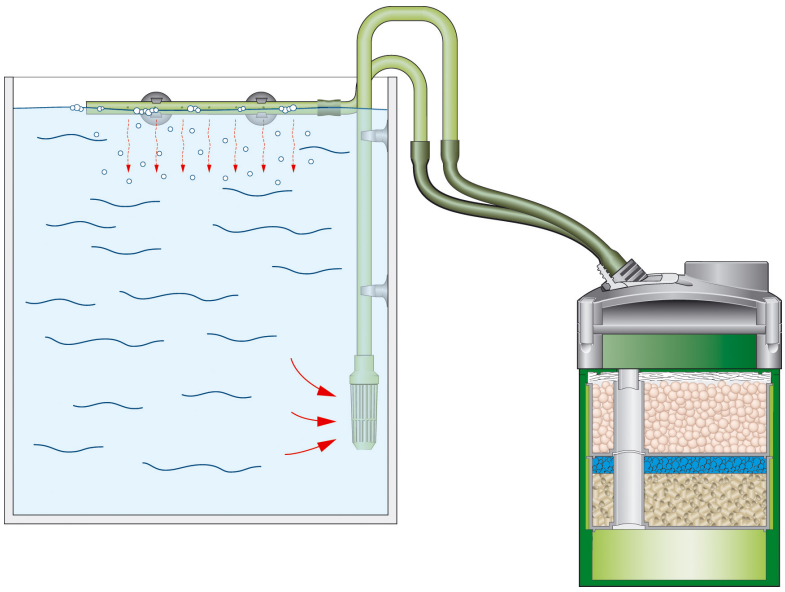

Figura 1.4.: Funcionament filtre exterior (Imatges de google).

Figura 1.5.: Radiador per Aquari (Imatges de google).

interacció té amb l'usuari. Per acabar en el vuitè hi han les conclusions que n'he tret i les possibles continuacions que tindria el projecte.

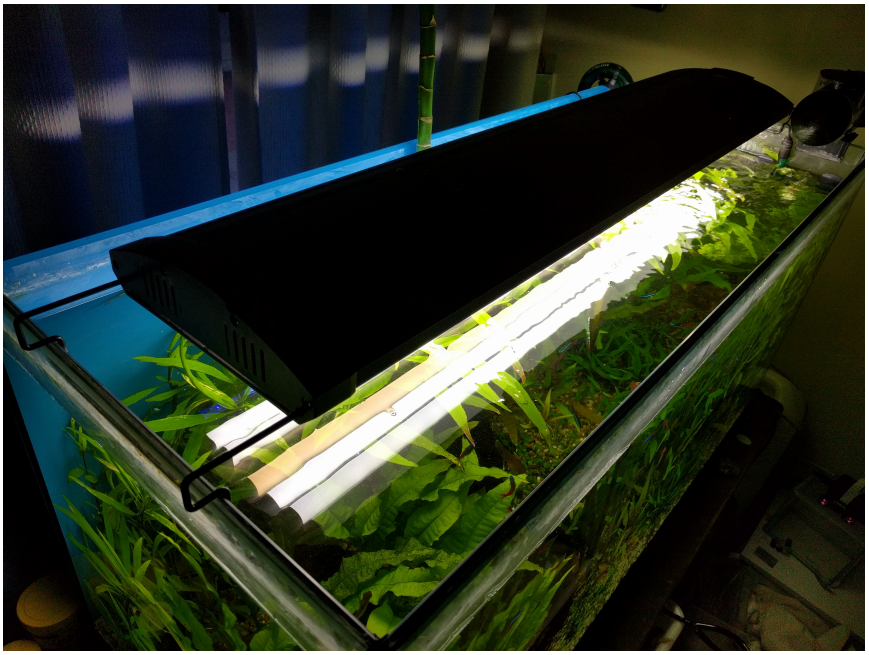

Figura 1.6.: Il·luminació (Imatge pròpia).

Figura 1.7.: Equip osmosis (Imatge pròpia).

Figura 1.8.: Sonda de pH (Imatge pròpia).

Figura 1.9.: Bombona de CO2 amb manòmetre (Imatge pròpia).

Figura 1.10.: Atomitzador (Imatge pròpia).

Figura 1.11.: Sensor de temperatura ds18b20(Imatges de google).

2. Els components electrònics utilitzats

Utilitzo un microordinador que s'encarrega de gestionar els paràmetres i que s'hi pugui accedir des de Internet. Com a perifèrics consta d'una càmera i un microcontrolador. El microcontrolador també té altres perifèrics, una placa de relés i un conversor que activa i desactiva components de la peixera i llegeix dades.

2.1. La Raspberry Pi 3

El Raspberry Pi és un ordinador monoplaca o SBC (acrònim en anglès de Single-Board Computer) de baix cost desenvolupat en el Regne Unit per la Fundació Raspberry Pi. L'objectiu principal d'aquest disseny és estimular l'ensenyança de les ciències de la computació, però també s'ha popularitzat com a plataforma per a dissenys d'aficionats i per a usos informàtics generals. Els primers models es van començar a comercialitzar el febrer de 2012. La segona evolució, Raspberry B+, es va anunciar el juliol de 2014 tot mantenint el mateix processador però amb diverses millores en els seus connectors i un menor consum energètic. El darrer model, Raspberry Pi 3[2.1], es va anunciar el 2 de febrer de 2016 i compta amb un processador molt millorat amb fins a 10 vegades més ràpid que la Raspberry Pi original, connectivitat sense fils wireless i Bluetooth 4.1 a més un GB de memòria RAM. La fundació dóna suport

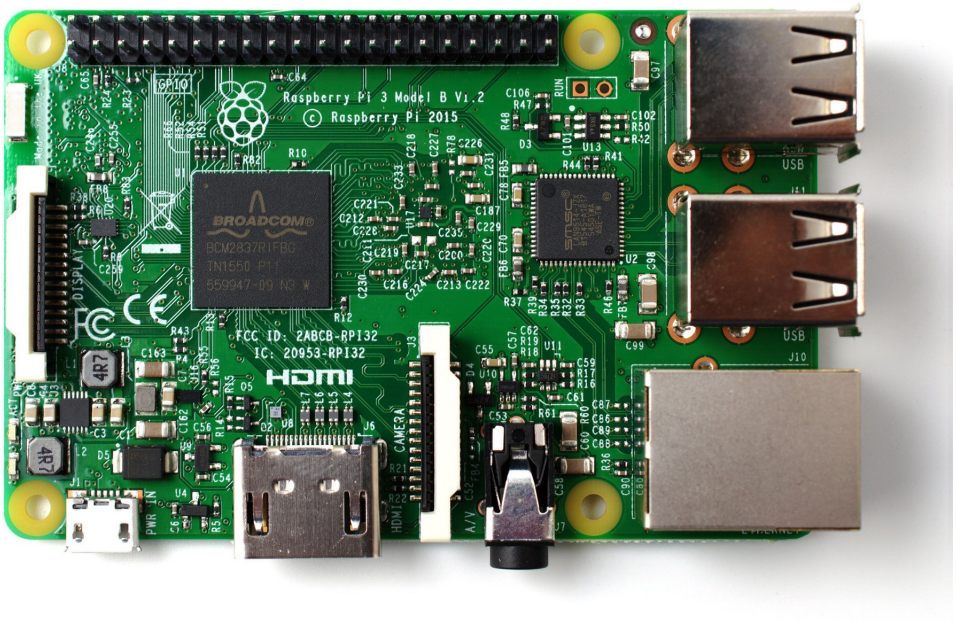

Figura 2.1.: Raspberry Pi 3(Imatges de google).

per a descàrregues de diferents distribucions Linux adaptades a l'arquitectura ARM: Raspbian (derivada de Debian), RISC OS 5, Arch Linux ARM (derivat d'Arch Linux) i Pidora (derivada de Fedora). Com a eina d'ensenyament de programació promouen principalment el llenguatge de programació Python, però també altres llenguatges com ara Tiny BASIC, C, Perl i Ruby. Extret de Viquipèdia [Viq16c]

En el projecte faig servir la Raspberry Pi 3[2.1].

2.1.1. La Càmera Pi NOIR

La Càmera Pi [2.2] és la càmera que proporciona Raspberry: és de 8 megapixels i pot fer vídeo a 1080p. A part d'això aquesta no té filtre d'infraroig per així poder gravar de nit. Porta dos leds infraroig, un a cada banda, per poder gravar sempre i aparentment no fer llum.

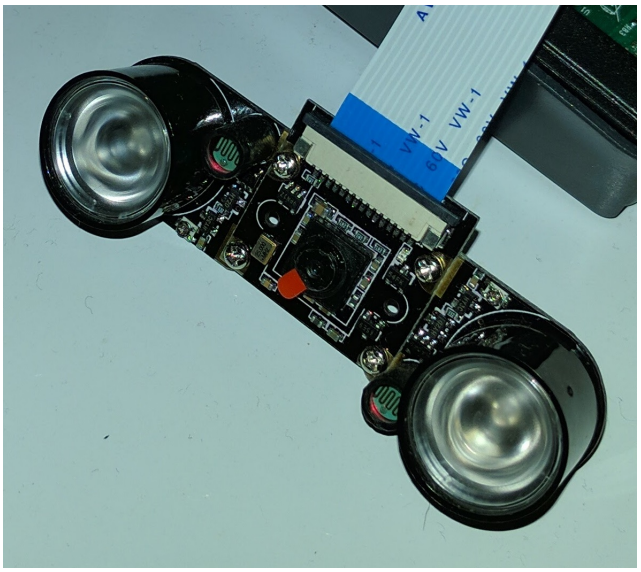

Figura 2.2.: Càmera Pi NOIR amb leds infraroig(Imatge pròpia).

2.2. L'Arduino MEGA2560

Arduino és un nom propi masculí italià que significa "gran amic". Arduino és una placa de circuit imprès simple basada en el microcontrolador de codi obert provinent de la plataforma de codi obert Wiring amb l'objectiu de fer més simple i accessible el disseny de circuits electrònics amb microcontroladors. El maquinari consisteix en dissenys simples de maquinari lliure amb processadors Atmel AVR en una placa amb pins E/S. L'entorn de desenvolupament implementa el llenguatge Processing de Wiring, molt semblant a C++. Arduino es pot utilitzar per desenvolupar objectes interactius autònoms o pot ser connectat a programari de l'ordinador (p. ex. Macromedia Flash, Processing, Max/MSP, Pure Data). Les plaques es poden muntar a mà o adquirir-se i els IDE de font oberta es poden descarregar de franc. Extret de Viquipèdia [Viq16b]

Jo he escollit l'Arduino Mega [2.3] ja que té més entrades i sortides i en un futur, no hi hauria problema per afegir components.

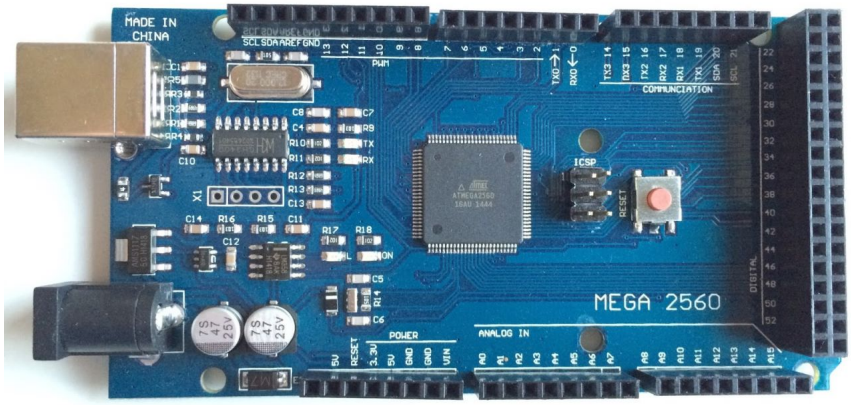

Figura 2.3.: Arduino MEGA2560 (Imatges de google).

2.2.1. El shield de relés

Un relé (del francès relais, relleu) és un mecanisme elèctric, inventat per Joseph Henry el 1835, que permet modificar l'estat d'un commutador elèctric gràcies a l'electricitat. A grans trets és un commutador elèctric que és accionat per un electroimant que obre o tanca un o diversos contactes. Una característica important d'aquest component és que permet controlar circuits elèctrics de voltatge o intensitat molt superior al d'entrada. Extret de Viquipèdia [Viq16d]

El shield de relés [2.4] en té 8 i pot ser utilitzat amb l'Arduino o la Raspberry que van a 5V.

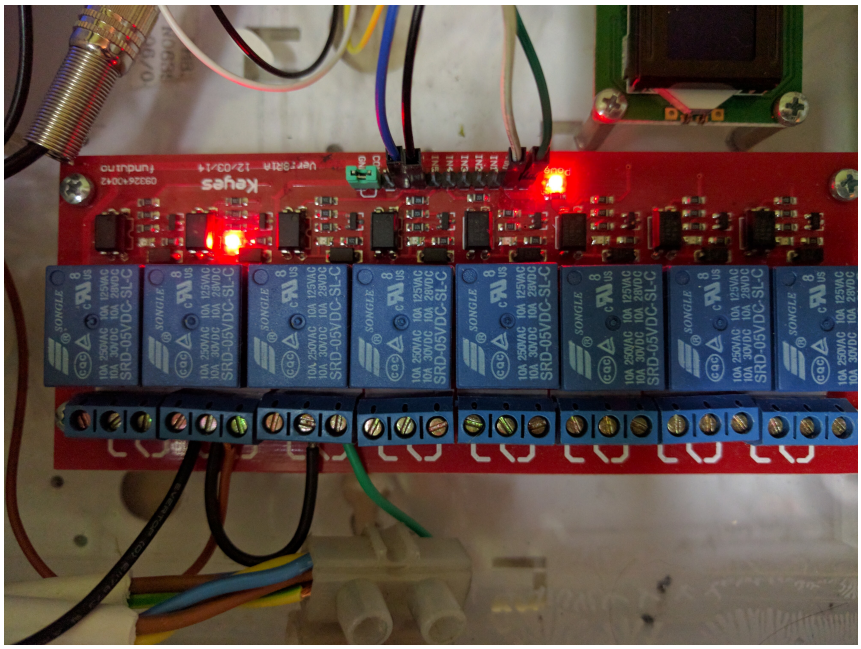

Figura 2.4.: Shield de relés (Imatges de google).

2.2.2. El conversor

La sonda de pH es llegeix a través d'aquest aparell. La sonda envia la tensió en mV i a través d'aquest conversor [2.5] que el transforma en valor analògic de 0 a 5 volts. És una mica delicat ja que si es mou o si rep interferències, el valor de pH varia considerablement.

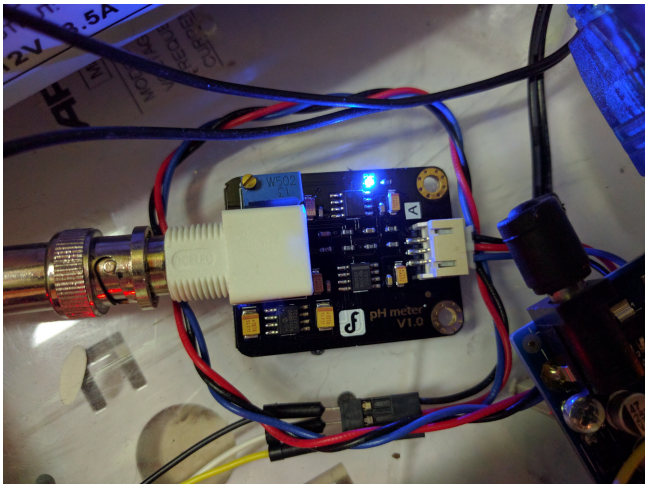

Figura 2.5.: Conversor (Imatge pròpia).

3. El disseny del sistema

El sistema està adaptat a l'aquari que ja tenia en propietat per tal de donar-li intel·ligència i autonomia. L'objectiu del sistema és millorar la qualitat de la peixera i a la vegada reduir la feina de manteniment. Hi ha feines que evidentment no les podrà dur a terme però les més importants de cara a la qualitat de vida dels individus sí .

3.1. Els conceptes previs

Abans d'exposar el disseny cal conèixer algunes aplicacions que utilitzo i alguns conceptes que crec que són importants per entendre bé el sistema. Les aplicacions són: Telegram, Servidor Python, Apache http server. Els conceptes són: la gestió d'usuaris i sistema robust.

3.1.1. El Telegram

Telegram és una aplicació de missatgeria instantània gratuïta i feta amb programari lliure que permet enviar i rebre missatges a través d'Internet. Permet crear grups i enviar imatges i vídeos. Un dels seus objectius és proveir una major privadesa i seguretat en comparació amb altres aplicacions similars. Ha estat desenvolupat pels germans Nikolai i Pavel Durov, els creadors de la xarxa social VK.

Amb cent milions d'usuaris, actualment és el tercer client de missatgeria mòbil pel que fa a nombre d'usuaris al món. Telegram és una aplicació de programari lliure, que pot ser modificada per la comunitat, pel que existeixen clients oficials per als sistemes operatius mòbils Android i iOS, i també per a ordinadors, així com versions per executar en el navegador. Extret de Viquipèdia [Viq16e].

He triat aquesta aplicació ja que permet crear Bots (programa informàtic que imita el comportament humà)i això em permet interactuar amb la Raspberry com si fos un 'humà'. És a dir, que si jo li faig una petició a través d'una comanda, em contesta amb la resposta adient. També dir que hi ha varies llibreries de Python per programar el bot, entre elles algunes recomanades pel propi Telegram com són:twx.botapi, Telepot i Telegram Bot Service. Jo he escollit Telepot ja que he trobat alguns exemples que m'han ajudat a entendre com funciona.

3.1.2. El servidor Python

Un servidor en informàtica és un maquinari equipat d'un programari capaç de rebre peticions d'altres màquines i donar-li les respostes adequades. El fet de ser en Python em facilita la feina ja que és un llenguatge de programació que conec. També dir que és una pràctica ja feta del grau adaptada al projecte [iTI16].

3.1.3. L'Apache http server

L'Apache és un servidor http de fàcil configuració i conegut arreu. Durant el grau hi hem treballat i també ja l'utilitzava a casa meva abans de fer el projecte. Per tant ara, l'he aprofitat

per col·locar-hi la web de la peixera.

3.1.4. La gestió d'usuaris

Com que el bot el programo jo, implemento una gestió d'usuaris on hi distingeixo 3 tipus:

- Administrador: Rep totes les consultes que es fan al bot sigui qui sigui l'usuari. A part també té algunes funcions exclusives que els altres usuaris no tenen.
- Alertats: Aquests usuaris només tenen d'extra que reben notificacions d'alertes i no cal que siguin administradors.
- Usuari base: Aquests usuaris poden interactuar amb el bot però només amb les funcions bàsiques.

El fet de ser d'un grup no n'exclou cap altre, només aporta privilegis respecte l'usuari base.

3.1.5. El sistema robust

Un sistema robust és aquell que està a prova d'error. Això és necessari ja que en un aquari hi han éssers vius i depenen del sistema per viure en bones condicions. En cas d'algun error que faci variar els paràmetres pot causar la mort de tots el individus. Per això molt important que tots els canvis que és van fent duran el projecte i les possibles ampliacions, siguin vigilats durant un període prudent. Per això en el projecte implemento un vigilant que controla la majoria dels processos i els reactiva si s'aturen.

3.2. El sistema de monitoratge

El sistema de monitoratge està centrat en la Raspberry, on es gestionen les dades i es distribueixen cap als serveis, alguns en la pròpia Raspberry i altres en diferents components. La idea és saber en tot moment com està la peixera i per això he escollit una aplicació de missatgeria instantània(Telegram) i també una web. Seguint l'esquema [3.1], tot comença amb la peixera, els sensors llegeixen l'estat de la peixera i a través del Arduino, la Raspberry en fa l'anàlisi. Un cop analitzades fa activar o desactivar els actuadors a través del Arduino. Per altre banda, la Raspberry emet les dades quan es demanen, des de Internet quan és per Telegram i pel Servidor Python cap al Apache per la web. A través de Telegram també es pot actuar, en aquest cas la Raspberry activa o desactiva els actuadors a través del Arduino. El sistema actual també es podria dur a terme sense l'Arduino, però en cas d'ampliar el sistema hi ha més versatilitat d'aquesta manera. Els sensors llegeixen el pH i la temperatura de l'aigua. Els actuadors activen/desactiven la vàlvula de CO2, la llum(encendre i apagar en manual i automàtic) i la càmera.

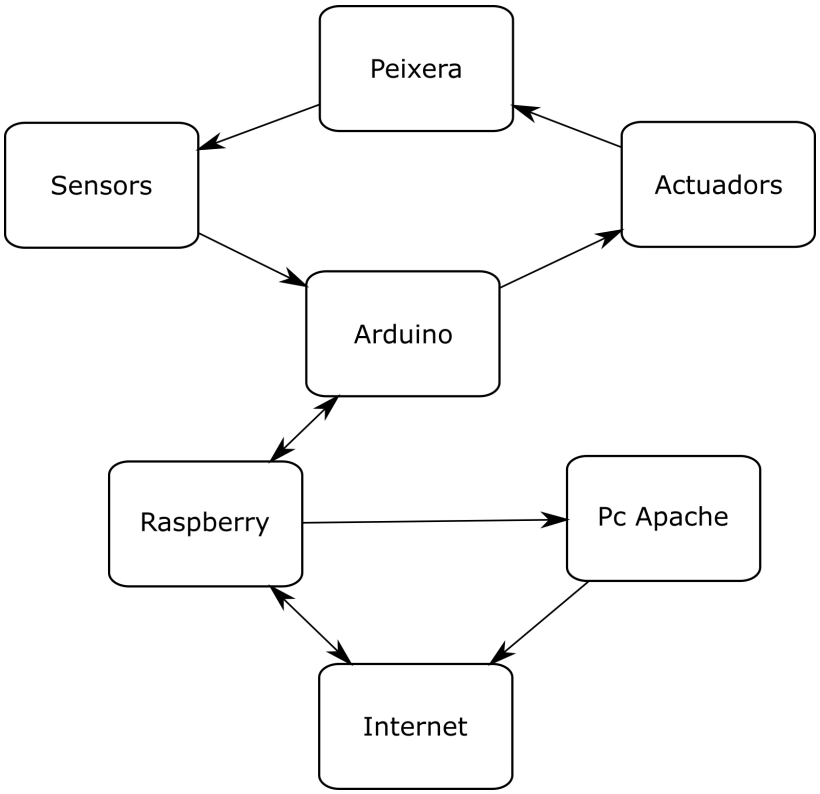

Figura 3.1.: Esquema del sistema.

4. Les comunicacions utilitzades

La comunicació Raspberry-Arduino, es fa pel port sèrie mitjançant USB i la comunicació de l'Arduino amb els altres components es fa a través d'entrades i sortides digitals o entrades analògiques. També existeixen altres comunicacions com per exemple la web amb el servidor Python i els processos entre sí.

4.1. La comunicació sèrie

La comunicació sèrie o comunicació seqüencial, en telecomunicacions i informàtica, és el procés d'enviar dades bit a bit de forma seqüencial, sobre un canal de comunicació o un bus. L'avantatge de la comunicació sèrie és que necessita un nombre més petit de línies de transmissió que en una comunicació paral·lela que transmet a la mateixa informació. Per altre banda, també és més lent i ha de fer servir freqüències més elevades.

Aquest protocol s'utilitza per la comunicació entre la Raspberry i l'Arduino.

4.2. Les entrades analògiques

L'entrada analògica com el seu nom diu, permet llegir un senyal analògic entre 0 i 5 volts. Aquest valor acaba estant codificat en bits segons la programació del Arduino. Es pot afinar més o menys en un tram de la tensió segons convingui a l'aplicació.

4.3. Les entrades/sortides digitals

El sensor de temperatura es comunica d'aquesta forma; l'Arduino li demana les dades i el sensor respon en conseqüència. És útil per que no fan falta gaires cables ja que amb l'alimentació, el GND i el de comunicació ja queda cobert. No obstant, com que només hi ha un cable, la comunicació no pot ser molt ràpida però per el cas que ens ocupa no hi haurà cap mena de problema.

4.4. Les altres comunicacions

La resta de comunicacions no necessiten un medi propi si no que s'aprofiten del mateix Internet o del sistema operatiu. La comunicació entre el servidor i la web es realitza per http, aprofitant la instal·lació d'Internet de casa. Al ser en la xarxa privada, al servidor Python no s'hi pot accedir des de Internet. El pc on està allotjada utilitza proxy per fer la connexió.

En la comunicació entre processos aprofito la llibreria multiprocessing que proporciona una cua on els processos poden agafar i posar valors. D'aquesta manera utilitzo dues cues, una on el procés de gestió anirà posant les dades per tal de que la resta els pugi llegir i una altre cua on els processos capacitats per modificar paràmetres, com per exemple la llum, puguin posar els valors allí per tal de que el sistema de gestió els pugui recollir.

5. El software dels dispositius

Per programar utilitzo llenguatges ja utilitzats al grau, en l'Arduino ArduinoIDE [Ard16], per la Raspberry Python i per la web HTML i JavaScript. En el cas del servidor Python i la web adapto una pràctica feta durant el grau.

5.1. L'Arduino

L'Arduino només va seguint el bucle principal esperant lectures del port sèrie i a la vegada va recollint valors de pH i temperatura, per quan es vulguin tenir no haver de buscar-los. Com que no varien molt bruscament si no passa cap desgràcia, no importa si són de fa uns segons. Pel port serie l'hi arribaran ordres de la Raspberry, que poden ser ordres per actuar en la llum, CO2... o perquè demana un valor de pH o temperatura. El mòdul principal és el següent:

```
#include <DallasTemperature.h>
#include <OneWire.h>
#define CALENTADOR_IN 25
#define LG1 43
#define autoled 0
#define onled 1
#define offled 2
int intled_a = autoled;
int intled_b = autoled;
int botled_a = autoled;
int botled_b = autoled;
float p = false;
float c = false;
float l = false;
int r = 'U';
#define SensorPin 0       //pH meter Analog output to Arduino Analog Input 0
unsigned long int avgValue; //Store the average value of the sensor feedback
float b;
int buf[30],temph;
int DS18S20_Pin = 25; //DS18S20 Signal pin on digital 22
//Temperature chip i/o
OneWire ds(DS18S20_Pin); // on digital pin 22
DallasTemperature sensors(&ds);
bool e_bomba, llum, e_calentador, e_nivell;
float temperatura, phValue;
int nivell=0;
int e;
void setup () {
  Serial.begin(9600);
  pinMode(CO2, OUTPUT);
  sensors.begin();
  llums_init();
```

```
}
void loop () {
    Serial.flush();
    if (Serial.available()){
      r = Serial.read();
      if (r=='P'){
        Serial.println(phValue);
      }
      else if (r=='T'){
        Serial.println(temperatura);
        //Serial.println(26.0);
      }
      else if (r=='N'){
        //Serial.println(nivell);
        Serial.println(false);
      }
      else if (r=='C'){
        while (!Serial.available()){}
        e=Serial.read();
        if (e=='I'){
          co2(true);
          //Serial.println("OK");
        }
        else if (e=='O'){
          co2(false);
          //Serial.PRINTLN("OK");
        }
      }
      else if (r=='E'){
        while (!Serial.available()){}
        e=Serial.read();
        if (e=='I'){
          //co2(true);
          //Serial.PRINTLN("OK");
        }
        else if (e=='O'){
          //co2(false);
          //Serial.PRINTLN("OK");
        }
      }
      else if (r=='V'){
        while (!Serial.available()){}
        e=Serial.read();
        if (e=='I'){
          //co2(true);
          //Serial.PRINTLN("OK");
        }
        else if (e=='O'){
          //co2(false);
          //Serial.PRINTLN("OK");
        }
      }
      else if (r=='B'){
```

```
      while (!Serial.available()){}
      e=Serial.read();
      if (e=='I'){
        //co2(true);
        //Serial.PRINTLN("OK");
      }
      else if (e=='O'){
        //co2(false);
        //Serial.PRINTLN("OK");
      }
    }
    else if (r=='L'){
      while (!Serial.available()){}
      e=Serial.read();
      if (e=='I'){
        llum_auto(true);
        //Serial.PRINTLN("OK");
      }
      else if (e=='O'){
        llum_auto(false);
        //Serial.PRINTLN("OK");
      }
    }
  }

  /*
  if(!digitalRead(automatic)){
    //llum_auto(true);
  }
  else if(!digitalRead(manual)){
    llum_auto(true);
  }
  else if (digitalRead(automatic) && digitalRead(manual)){
    llum_auto(false);
  }
  */
  temp();
  ph();
  Serial.flush();
}
```

La resta de mòduls on s'activen i es desactiven els components es troben a l'annex[1].

5.2. La Raspberry

La Raspberry és la que s'encarrega de gestionar les dades i distribuir-les cap als serveis, alguns en la pròpia Raspberry i altres en diferents components. Els serveis que hi allotja són: el bot de Telegram i el servidor Python. Per això hi treballen varis processos a la vegada i es comuniquen entre sí [5.1].

- El Bot de Telegram: Aquest procés s'encarrega de gestionar els missatges que es reben i contestar adientment. S'espera que els missatges siguin per demanar dades de la pei-

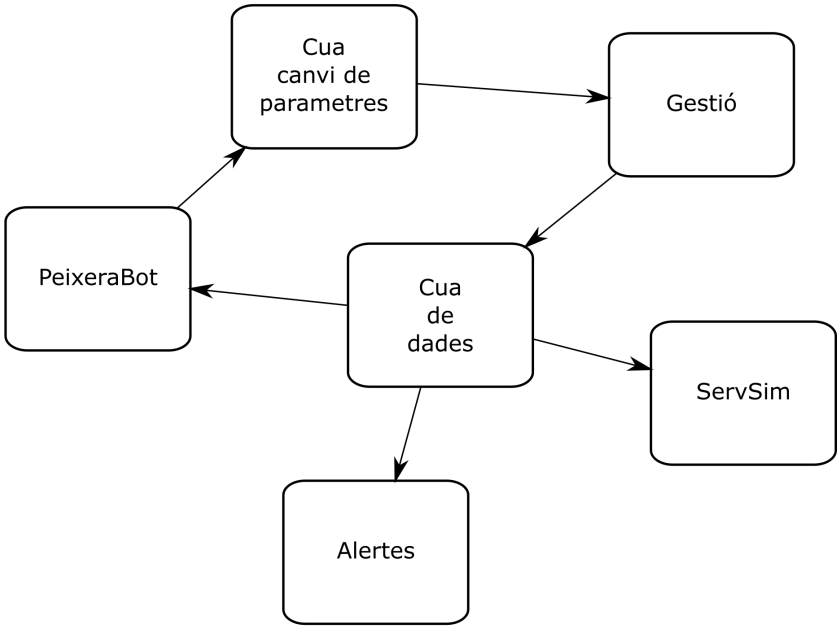

Figura 5.1.: Relacions entre processos de Python.

xera o per accionar o aturar alguns components com per exemple la llum. A la vegada també avisa als administradors de tots els missatges que rep el bot i també en deixa constància a un fitxer. Les dades les recull d'una cua on el procés de gestió les va deixant periòdicament. A la vegada en cas d'accionar algun component, es deixa a una altre cua que el procés de gestió recollirà.

- Les alertes: És un derivat del bot ja que quan detecta que el ph o la temperatura estan fora d'uns límits, envia un missatge de Telegram a totes les persones que estan registrades.
- La gestió de la peixera: És el procés més important de tots ja que s'encarrega de que la peixera estigui en bones condicions. Va comprovant que els paràmetres estiguin dins dels límits i en cas de no estar-ho actua en conseqüència. Un cop fet això posa el valor a la cua per tal de que la resta de processos se'n puguin servir.
- El servidor Python: Aquest procés proveeix les dades a la pàgina web. La pàgina web demana al servidor les dades i ell les recull de la cua per servir-les.
- El vigilant: Per últim tenim el vigilant de tot el sistema, que s'encarrega d'activar tots els processos i de que es mantinguin en marxa. En cas de no estar-ho els reactiva.

5.2.1. Els mòduls de Python

He separat les tasques per mòduls ja que així és fa més fàcil [5.2].

La comunicació amb l'Arduino

Aquest mòdul només serveix per transmetre dades per el port sèrie a l'Arduino [2.2]. Només es pot utilitzar per el mòdul de gestió.

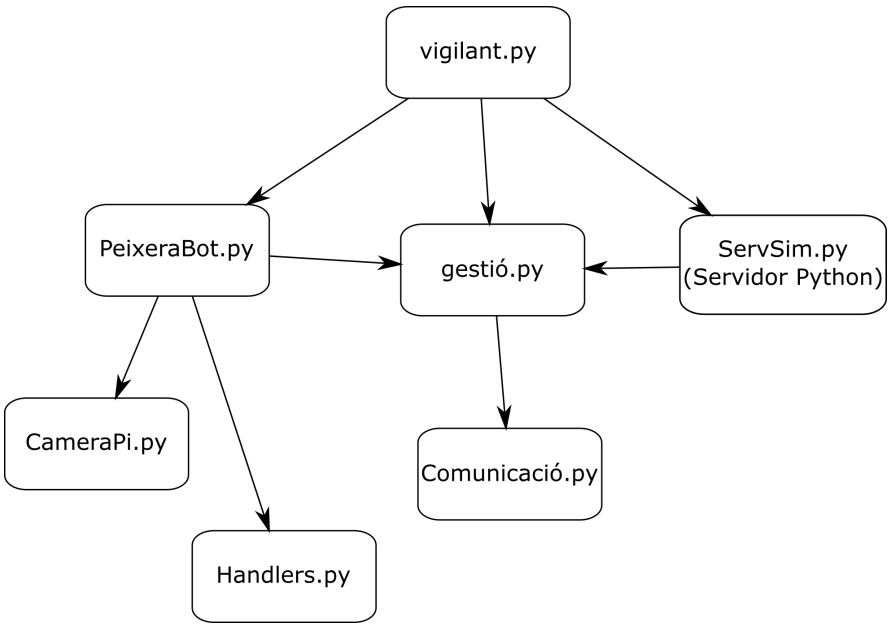

Figura 5.2.: Relacions entre mòduls de Python.

```
import serial
import time
class Arduino(object):
   def __init__(self):
      self.port = serial.Serial('/dev/ttyACM0', 9600)
      time.sleep(2)
   def demana(self, dada):
      while True:
         try:
            time.sleep(0.5)
            self.port.flush()
            #print 'demana', dada
            self.port.write(dada)
            #time.sleep(1)
            b = self.port.readline()
            #print b
            return float(b)
         except:
            print 'ERROR demana '+dada
            pass
   def escriu(self, dada, estat):
      while True:
         try:
            time.sleep(0.5)
            self.port.flush()
            #print 'escriu '+dada
            self.port.write(dada)
            #print estat
            if estat:
```

```
            self.port.write('I')
        else:
            self.port.write('O')
        #print dada+' escrit'
        return estat
    except:
        print 'ERROR escriu '+dada+str(estat)
        pass
```

La gestió de la peixera

Aquest és el mòdul [2.1] que s'encarrega de gestionar la peixera a través del mòdul de comunicació. Les feines que fa de gestió són les següents:

- Recull les dades i les afegeix a la cua per tal de que la resta de processos les puguin llegir.
- Registra tots els estats de cada component en un fitxer de text per després poder saber el que ha passat en cas d'algun incident.
- Amb els valors de pH i temperatura en fa gràfiques perquè després la resta de processos se'n puguin servir.
- Regula el pH segons uns marges preestablerts que depenen dels peixos que hi hagi (en aquest moment entre 6.7 i 6.8).
- Encén i apaga la llum. L'encén entre les 14h i les 21h i l'apaga la resta d'hores.
- Està preparat per si en un futur s'hi afegeix un sensor de nivell i poder encendre i apagar una bomba per omplir la peixera.
- També està preparat per si es connecta el radiador i/o ventiladors, és puguin regular segons el sensor de temperatura.

Aquest és el codi d'aquest mòdul, on podem veure la funció 'actualitza' que serà la que es cridarà com a procés. També hi ha la funció 'actua llum' que serà cridada per els processos per encendre o apagar la llum.

```
from datetime import *
from comunicacio import Arduino
import time as timer
import Gnuplot, sys, os
NA_BAIX = True
NA_ALT = False
ON = True
OFF = False
AUTO = True
MANUAL = False
PH = 'P'
TEMP = 'T'
NIV = 'N'
CO2 = 'C'
CALENTADOR = 'E'
```

```
VENTILADOR = 'V'
BOMBA = 'B'
LLUM = 'L'
class Dades(object):
```

Aquesta funció inicialitza la peixera de tal manera que queda tot parat i els paràmetres a 0. Així, quan faci un actualització activarà el que convingui.

```
def __init__(self,
    nom="/home/pi/peixera2/p_casa"+datetime.now().strftime("%Y%b%d")+".dat"):
   self.ph = 0.0
   self.temp = 0.0
   self.llum = OFF
   self.llum_mode = AUTO
   self.co2 = OFF
   self.calentador = OFF
   self.ventilador = OFF
   self.bomba_aigua = OFF
   self.nivell_aigua = NA_BAIX
   self.arduino = Arduino()
   self.nom = nom
```

Aquesta funció guarda les dades actuals en el fitxer de text, que és self.nom.

```
def guarda_dades(self):
   f = open(self.nom, 'a')
   f.write(' \n' + datetime.now().strftime("%Y-%b-%d-%H:%M:%S") + ' ')
   f.write(str(self.ph)+' '+str(self.temp)+' nivell-'+str(self.nivell_aigua)+'
       llum-'+str(self.llum)+' co2-'+str(self.co2)+'
       calentador-'+str(self.calentador)+' ventiladors-'+str(self.ventilador)+'
       bomba-'+str(self.bomba_aigua)+ ' ' + str(self.llum_mode))
   f.close()
```

Les funcions següents fan les gràfiques del pH i temperatura. Pot semblar que estan repetides però la veritat és que unes fan una imatge per una futura interfície gràfica i les altres per ser enviades per Telegram o per la web.

```
def grafica_ph(self):
   gp = Gnuplot.Gnuplot()
   gp('set timefmt "%Y-%b-%d-%H:%M:%S"')
   gp('set yrange [6:8]')
   gp('set xdata time')
   gp('set format x "%d %H:%M"')
   gp('set xtics font ", 12"')
   s1 = 'set xrange ["'
   s3 = '":"'
   s5 = '"]'
   ara = datetime.now()
   DD = timedelta(days=3)
   ahir = ara - DD
   s2 = ahir.strftime("%Y-%b-%d-%H:%M:%S")
   s4 = ara.strftime("%Y-%b-%d-%H:%M:%S")
   #  gp('set xrange ["Mar-15-21:00:00":"Mar-17-00:00:00"]')
```

```
    gp(s1+s2+s3+s4+s5)
    gp('set nokey')
    gp('set terminal png size 1920,1080 enhanced font "Helvetica,12"')
    gp('set output "/home/pi/peixera2/ph.png"')
    gp("plot '/home/pi/peixera2/p_casa.dat' using 1:2 title 'ph'")
  def grafica_int_ph(self):
    gp = Gnuplot.Gnuplot()
    gp('set timefmt "%Y-%b-%d-%H:%M:%S"')
    gp('set yrange [6:8]')
    gp('set xdata time')
    gp('set format x "%d %H:%M"')
    gp('set xtics font ", 8"')
    s1 = 'set xrange ["'
    s3 = '":"'
    s5 = '"]'
    ara = datetime.now()
    DD = timedelta(days=3)
    ahir = ara - DD
    s2 = ahir.strftime("%Y-%b-%d-%H:%M:%S")
    s4 = ara.strftime("%Y-%b-%d-%H:%M:%S")
    #  gp('set xrange ["Mar-15-21:00:00":"Mar-17-00:00:00"]')
    gp(s1+s2+s3+s4+s5)
    gp('set nokey')
    gp('set lmargin at screen 0')
    gp('set rmargin at screen 1')
    gp('set tmargin at screen 1')
    gp('set bmargin at screen 0')
    gp('set xtics offset 0, screen 0.1')
    gp('set ytics offset screen 0.1, 0')
    gp('set terminal png size 320,144 enhanced font "Helvetica,8"')
    gp('set output "/home/pi/peixera2/ph_int.png"')
    gp("plot '/home/pi/peixera2/p_casa.dat' using 1:2 title 'ph'")
  def grafica_temp(self):
    gp = Gnuplot.Gnuplot()
    gp('set timefmt "%Y-%b-%d-%H:%M:%S"')
    gp('set yrange [20:32]')
    gp('set xdata time')
    gp('set format x "%d %H:%M"')
    gp('set xtics font ", 12"')
    s1 = 'set xrange ["'
    s3 = '":"'
    s5 = '"]'
    ara = datetime.now()
    DD = timedelta(days=3)
    ahir = ara - DD
    s2 = ahir.strftime("%Y-%b-%d-%H:%M:%S")
    s4 = ara.strftime("%Y-%b-%d-%H:%M:%S")
    #  gp('set xrange ["Mar-15-21:00:00":"Mar-17-00:00:00"]')
    gp(s1+s2+s3+s4+s5)
    gp('set nokey')
    gp('set terminal png size 1920,1080 enhanced font "Helvetica,12"')
    gp('set output "/home/pi/peixera2/temp.png"')
    gp("plot '/home/pi/peixera2/p_casa.dat' using 1:3 title 'Temperatura'")
```

```
def grafica_int_temp(self):
    gp = Gnuplot.Gnuplot()
    gp('set timefmt "%Y-%b-%d-%H:%M:%S"')
    gp('set yrange [20:32]')
    gp('set xdata time')
    gp('set format x "%d %H:%M"')
    gp('set xtics font ", 8"')
    s1 = 'set xrange ["'
    s3 = '":"'
    s5 = '"]'
    ara = datetime.now()
    DD = timedelta(days=3)
    ahir = ara - DD
    s2 = ahir.strftime("%Y-%b-%d-%H:%M:%S")
    s4 = ara.strftime("%Y-%b-%d-%H:%M:%S")
    #  gp('set xrange ["Mar-15-21:00:00":"Mar-17-00:00:00"]')
    gp(s1+s2+s3+s4+s5)
    gp('set nokey')
    gp('set lmargin at screen 0')
    gp('set rmargin at screen 1')
    gp('set tmargin at screen 1')
    gp('set bmargin at screen 0')
    gp('set xtics offset 0, screen 0.1')
    gp('set ytics offset screen 0.1, 0')
    gp('set terminal png size 320,144 enhanced font "Helvetica,8"')
    gp('set output "/home/pi/peixera2/temp_int.png"')
    gp("plot '/home/pi/peixera2/p_casa.dat' using 1:3 title 'Temperatura'")
```

La següent funció és la que envia les comandes a l'Arduino per actuar sobre els dispositius. Es pot veure com està preparat per possibles ampliacions.

```
def actua(self):
    self.arduino.escriu(CO2, self.co2)
    self.arduino.escriu(CALENTADOR, self.calentador)
    self.arduino.escriu(VENTILADOR, self.ventilador)
    self.arduino.escriu(BOMBA, self.bomba_aigua)
    self.arduino.escriu(LLUM, self.llum)
```

Consulta és la funció per obtenir els paràmetres de la peixera, que també està preparada per possibles ampliacions.

```
def consulta(self):
    self.ph = self.arduino.demana(PH)
    #print self.ph
    self.temp = self.arduino.demana(TEMP)
    #print self.temp
    #self.nivell_aigua = self.arduino.demana(NIV)
```

Actua llum és una funció cridada pel mòdul de Telegram per actuar sobre la peixera només en la llum.

```
def actua_llum(self, onoff, mode, q):
    self.llum_mode = mode
```

```
self.llum = onoff
q.put_nowait(self)
```

Actualitza és el procés que manté la peixera en bon estat i el vigilant la controla per tal de que no s'aturi mai. Al principi mira la cua on els serveis deixen les seves dades en cas que vulguin actuar en algun component. Després fa una consulta i gestiona les dades. Per acabar omple la cua on la resta de processos recullen les dades i guarda les dades i en fa gràfiques.

```
def actualitza(self, q=None, l=None):
   try:
      if not l.empty():
         while not l.empty():
            p = l.get_nowait()
         if p.llum_mode:
            self.llum_mode = AUTO
         else:
            self.llum_mode = MANUAL
         if p.llum:
            self.llum = ON
         else:
            self.llum = OFF
   except:
      pass
   self.consulta()
   if self.ph > 6.8:
      self.co2 = ON
   elif self.ph < 6.7:
      self.co2 = OFF

   if self.temp < 24.0:
      self.calentador = ON
      self.ventilador = OFF
   elif self.temp > 28.0:
      self.ventilador = ON
      self.calentador = OFF

   if self.nivell_aigua == NA_BAIX:
      self.bomba_aigua = ON
   else:
      self.bomba_aigia = OFF

   now = datetime.now()
   now_time = now.time()
   if self.llum_mode == AUTO:
      if now_time >= time(12,00) and now_time <= time(19,00):
         self.llum = ON
      else:
         self.llum = OFF
   try:
      while not q.empty():
         p = q.get_nowait()
      while not q.full():
```

```
            q.put_nowait(self)
        except:
           pass
        self.actua()
        self.guarda_dades()
        self.grafica_ph()
        self.grafica_int_ph()
        self.grafica_temp()
        self.grafica_int_temp()
        timer.sleep(5)
```

El Bot de Telegram

El Bot de Telegram [2.3] està implementat amb la llibreria Telepot [nic16]. Està preparat perquè algunes funcions siguin exclusives per administradors i la resta de clients no les puguin usar, com per exemple fer fotos o encendre i apagar la llum. Per altre banda també conté una funció que servirà per alertar en cas de que el pH o la temperatura sobre passin uns limits. Aquestes alertes arribaran als clients als quals prèviament s'hagin afegit.També he creat uns teclats per tal de que sigui més fàcil interactuar amb el bot. N'hi ha dos; el teclat general [5.3] i el teclat específic per interactuar amb la llum[5.4]. Hi ha un altre mòdul molt lligat aquest que conté les ordres que pot servir el bot.

Figura 5.3.: Teclat general.

Aquests dos mòduls tenen el següent codi:

- Mòdul principal, la primera funció inicialitza el bot i el deixa en marxa.

  També hi ha funcions com check admin, difon admin.... que s'utilitzen en la gestió d'usuaris per tal de difondre un missatge a tots els administradors o alertats i per comprovar que siguin d'aquest grup.

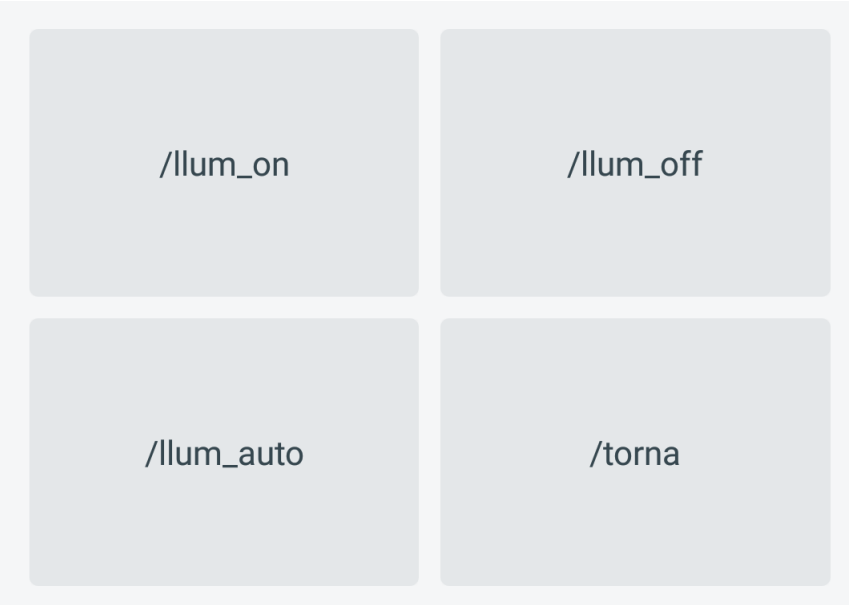

Figura 5.4.: Teclat llum.

```
# -*- coding: utf-8 -*-
import telepot
import time
from handlers import *
from telepot import namedtuple as types
from camerapi import Camera
class PeixeraBot:
    def __init__(self, p, q, l):
        self.users = []
        self.usuaris = [line.rstrip('\n') for line in
            open('/home/pi/peixera2/usuaris.txt')]
        self.administradors = [7267782, 4375388]
        self.alertats = [7267782, 4375388]
        BOT_TOKEN = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
        self.bot = telepot.Bot(BOT_TOKEN)
        self.peixera = p
        self.q = q
        self.l = l
        self.cam = Camera()
        self.handlers = {'/start': Start(self.bot, self.users),
            '/stop': Stop(self.bot, self.users),
            '/help': Help(self.bot),
            '/torna/': Torna(self.bot, self.teclat_general),
            '/llum':Llum(self.bot,self.peixera, self.q, self.teclat_llum),
            '/co2': Co2(self.bot,self.peixera, self.q),
            '/ph' : Ph(self.bot,self.peixera, self.q),
            '/temp' :Temp(self.bot,self.peixera, self.q),
            '/hora' : Hora(self.bot),
            '/grafica_ph' : Grafica_ph(self.bot),
            '/grafica_temp' : Grafica_temp(self.bot),
```

```
            '/foto' : Foto(self.bot, self.cam, self.administradors,
                self.check_admin, self.difon_admin),
            '/video' : Video(self.bot, self.cam, self.administradors,
                self.check_admin, self.difon_admin),
            '/llum_on' : Llum_on(self.bot,self.peixera, self.l,
                self.administradors, self.check_admin, self.difon_admin),
            '/llum_off' : Llum_off(self.bot,self.peixera, self.l,
                self.administradors, self.check_admin, self.difon_admin),
            '/llum_auto' : Llum_auto(self.bot,self.peixera, self.l,
                self.administradors, self.check_admin, self.difon_admin),
            }
        self.bot.message_loop(self.handle_message)
```

Aquestes funcions s'utilitzen en la gestió d'usuaris per tal de difondre un missatge a tots els administradors o alertats i per comprovar que siguin d'aquest grup.

```
def check_admin(self, cid):
    for id in self.administradors:
        if cid == id:
            return True
    return False
def difon_admin(self, m):
    for id in self.administradors:
        self.bot.sendMessage(id, m)
#Difon un missatge d'alerta
def difon_alerta(self, m):
    for id in self.alertats:
        self.bot.sendMessage(id, m)
```

Aquí faig els teclats general i l'especific per llum.

```
#Teclat general
def teclat_general(self, cid):
    #markup = types.ReplyKeyboardMarkup(row_width=3)
    markup=types.ReplyKeyboardMarkup(keyboard=[['/llum', '/co2',
        '/ph'],['/temp', '/hora', '/grafica_ph'],['/grafica_temp', '/foto',
        '/video']])
    self.bot.sendMessage(cid, 'Tria una opcio:', reply_markup=markup)
#Teclat llum
def teclat_llum(self, cid):
    #markup = types.ReplyKeyboardMarkup(row_width=2)
    markup=types.ReplyKeyboardMarkup(keyboard=[['/llum_on',
        '/llum_off'],['/llum_auto', '/torna']])
    self.bot.sendMessage(cid, 'Tria una opcio:', reply_markup=markup)
```

La funció alerta que avisa quan algun paràmetre no està correcte. Avisa al grup d'usuaris alertats.

```
def genera_alertes(self):
    while True:
        while self.q.empty():
            pass
```

```
        """
        self.peixera = self.q.get_nowait()
        if (self.peixera.ph < 6.6 or self.peixera.ph > 6.9):
         print 'alerta'
         self.difon_alerta('PH: '+ str(self.peixera.ph))
        if (self.peixera.temp < 23.0 or self.peixera.temp > 29.0):
         print 'alerta'
         self.difon_alerta('Temperatura: '+ str(self.peixera.temp))
        """
        time.sleep(60)
```

Aquesta última funció és la que gestiona tots els missatges que van arribant i els deriva segons la comanda.

```
def handle_message(self, message):
    print message
    cid = message['from']['id']
    if cid > 0:
        missatge = str(message['chat']['first_name']) + " [" + str(cid) +
            "]: " + message['text']
    else:
        missatge = str(message['from']['first_name']) + "[" + str(cid) + "]:
            " + message['text']
    f = open( '/home/pi/peixera2/log.txt', 'a')
    f.write(missatge + "\n")
    f.close()
    self.difon_admin(missatge)
    print 'ok'
    responses = []
    try:
        responses.append(self.handlers[message['text']].processMessage(message))
    except:
        for handler in self.handlers:
            responses.append(self.handlers[handler].processMessage(message))
    userId = message['from']['id']
    for response in responses:
        if response:
            self.bot.sendMessage(userId, response)
            self.teclat_general(message['from']['id'])
```

- Mòdul d'accions: Aquest mòdul conté les accions que a part de les normals per un bot de Telegram com /start, /stop i /help, també n'hi ha d'altres:
  - start: Inicialitza el bot.

```
import telepot
from datetime import datetime
class Start:
    def __init__(self, bot, users):
        self.bot = bot
        self.users = users
        self.usuaris = [line.rstrip('\n') for line in
```

```
            open('/home/pi/peixera2/usuaris.txt')]
    def processMessage(self, message):
        content_type, chat_type, chat_id = telepot.glance(message)
        cid = message['from']['id']
        if not str(cid) in self.usuaris: bot por primera vez.
            self.usuaris.append(str(cid)).
            aux = open( '/home/pi/peixera2/usuaris.txt', 'a')
            aux.write( str(cid) + "\n")
            aux.close()
        if content_type == 'text' and message['text'] == '/start':
            actualUser = message['from']['username']
            actualId = message['from']['id']
            self.users.append({actualId: actualUser})
            return "Bienvingut al @PeixeraBot!!!!"
```

– stop: Atura el bot.

```
class Stop:
    def __init__(self, bot, users):
        self.bot = bot
        self.users = users
    def processMessage(self, message):
        content_type, chat_type, chat_id = telepot.glance(message)
        actualUser = ''
        if content_type == 'text' and message['text'] == '/stop':
            actualUser = message['from']['username']
            actualId = message['from']['id']
            self.users.remove({actualId: actualUser})
            return "Fins aviat " + actualUser
```

– help: Torna un missatge d'ajuda.

```
class Help:
    def __init__(self, bot):
        self.bot = bot
    def processMessage(self, message):
        content_type, chat_type, chat_id = telepot.glance(message)

        if content_type == 'text' and message['text'] == '/help':
            the_help = 'Estas al PeixeraBot. Al teclat veuras les opcions
                que tens. '
            print the_help
            userId = message['from']['id']
            return the_help
```

– llum: Retorna l'estat de la llum i entres en el menú per gestionar-la.

```
class Llum:
    def __init__(self, bot, peixera, q, teclat):
        self.bot = bot
        self.p = peixera
        self.q = q
```

```
            self.teclat = teclat
            try:
                while self.q.empty():
                    pass
                self.p = self.q.get_nowait()
            except:
                pass
        def processMessage(self, message):
            try:
                while self.q.empty():
                    pass
                self.p = self.q.get_nowait()
            except:
                pass
            content_type, chat_type, chat_id = telepot.glance(message)
            if content_type == 'text' and message['text'] == '/llum':
                self.bot.sendMessage(message['from']['id'], 'Llum:
                    '+str(self.p.llum))
                self.teclat(message['from']['id'])
                #return 'Llum: '+str(self.p.llum)
```

– torna: Torna al teclat general en cas d'estar al teclat de llum.

```
class Torna:
    def __init__(self, bot, teclat):
        self.bot = bot
        self.teclat = teclat
    def processMessage(self, message):
        content_type, chat_type, chat_id = telepot.glance(message)
        if content_type == 'text' and message['text'] == '/torna':
            self.teclat(message['from']['id'])
```

– co2: Retorna l'estat del CO2

```
class Co2:
    def __init__(self, bot, peixera, q):
        self.bot = bot
        self.p = peixera
        self.q = q
        try:
            while self.q.empty():
                pass
            self.p = self.q.get_nowait()
        except:
            pass
    def processMessage(self, message):
        try:
            while self.q.empty():
                pass
            self.p = self.q.get_nowait()
        except:
            pass
```

```
        content_type, chat_type, chat_id = telepot.glance(message)

        if content_type == 'text' and message['text'] == '/co2':
            return 'CO2: '+str(self.p.co2)
```

– ph: Retorna el valor de pH

```
class Ph:
    def __init__(self, bot, peixera, q):
        self.bot = bot
        self.p = peixera
        self.q = q
        try:
            while self.q.empty():
                pass
            self.p = self.q.get_nowait()
        except:
            pass
    def processMessage(self, message):
        try:
            while self.q.empty():
                pass
            self.p = self.q.get_nowait()
        except:
            pass
        content_type, chat_type, chat_id = telepot.glance(message)

        if content_type == 'text' and message['text'] == '/ph':
            return 'Ph: '+str(self.p.ph)
```

– temp: Retorna el valor de temperatura.

```
class Temp:
    def __init__(self, bot, peixera, q):
        self.bot = bot
        self.p = peixera
        self.q = q
        try:
            while self.q.empty():
                pass
            self.p = self.q.get_nowait()
        except:
            pass
    def processMessage(self, message):
        try:
            while self.q.empty():
                pass
            self.p = self.q.get_nowait()
        except:
            pass
        content_type, chat_type, chat_id = telepot.glance(message)
        if content_type == 'text' and message['text'] == '/temp':
```

```
            return 'Temperatura: '+str(self.p.temp)
```

– hora: Retorna la hora a la qual està la Raspberry i per tant funciona el sistema.

```
class Hora:
    def __init__(self, bot):
        self.bot = bot
    def processMessage(self, message):
        content_type, chat_type, chat_id = telepot.glance(message)
        if content_type == 'text' and message['text'] == '/hora':
            return 'Hora: '+datetime.now().strftime("%Y-%b-%d-%H:%M:%S")
```

– gràfica ph: Retorna la gràfica de pH dels valors dels últims 3 dies.

```
class Grafica_ph:
    def __init__(self, bot):
        self.bot = bot
    def processMessage(self, message):
        content_type, chat_type, chat_id = telepot.glance(message)
        if content_type == 'text' and message['text'] == '/grafica_ph':
            self.bot.sendPhoto( message['from']['id'], open(
                '/home/pi/peixera2/ph.png', 'rb'))
```

– gràfica temp: Retorna la gràfica de temperatura dels valors dels últims 3 dies

```
class Grafica_temp:
    def __init__(self, bot):
        self.bot = bot
    def processMessage(self, message):
        content_type, chat_type, chat_id = telepot.glance(message)
        if content_type == 'text' and message['text'] == '/grafica_temp':
            self.bot.sendPhoto( message['from']['id'], open(
                '/home/pi/peixera2/temp.png', 'rb'))
```

– foto: Fa una foto i la retorna.

```
class Foto:
    def __init__(self, bot, cam, admins, check, difon):
        self.bot = bot
        self.cam = cam
        self.admins = admins
        self.check = check
        self.difon = difon
    def processMessage(self, message):
        content_type, chat_type, chat_id = telepot.glance(message)
        if content_type == 'text' and message['text'] == '/foto':
            if self.check(message['from']['id']):
                self.bot.sendPhoto( message['from']['id'], open(
                    self.cam.foto(), 'rb'))
            else:
                self.bot.sendMessage( message['from']['id'], 'Aquesta funcio
                    no et pertoca, no ets administrador')
```

```
                if cid > 0:
                    self.difon("El usuari " + str(m.chat.first_name) + " amb
                        ID: " + " [" + str(cid) + "]: " + " ha intentat fer
                        una foto") # Si 'cid' es positivo, usaremos
                        'm.chat.first_name' para el nombre.
                else:
                    self.difon("El usuari " + str(m.from_user.first_name) +
                        " amb ID: " + " [" + str(cid) + "]: " + " ha
                        intentat fer una foto")
```

– vídeo: Fa un vídeo de 20 segons i el retorna.

```
class Video:
    def __init__(self, bot, cam, admins, check, difon):
        self.bot = bot
        self.cam = cam
        self.admins = admins
        self.check = check
        self.difon = difon

    def processMessage(self, message):
        content_type, chat_type, chat_id = telepot.glance(message)

        if content_type == 'text' and message['text'] == '/Video':
            if self.check(message['from']['id']):
                self.bot.sendVideo(message['from']['id'], open(
                    self.cam.foto(), 'rb'))
            else:
                self.bot.sendMessage( message['from']['id'], 'Aquesta funcio
                    no et pertoca, no ets administrador')
                if message['from']['id'] > 0:
                    self.difon("El usuari " + str(m.chat.first_name) + " amb
                        ID: " + " [" + str(cid) + "]: " + " ha intentat fer
                        un video") # Si 'cid' es positivo, usaremos
                        'm.chat.first_name' para el nombre.
                else:
                    self.difon("El usuari " + str(m.from_user.first_name) +
                        " amb ID: " + " [" + str(cid) + "]: " + " ha
                        intentat fer un video")
```

– llum on: Encén la llum i la posa en mode manual.

```
class Llum_on:
    def __init__(self, bot, peixera, l, admins, check, difon):
        self.bot = bot
        self.p = peixera
        self.l = l
        self.admins = admins
        self.check = check
        self.difon = difon
    def processMessage(self, message):
        content_type, chat_type, chat_id = telepot.glance(message)
```

```
        if content_type == 'text' and message['text'] == '/llum_on':
            if self.check(message['from']['id']):
                self.p.llum = True
                self.p.llum_mode = False
                self.l.put_nowait(self.p)
                return 'Llum: '+str(self.p.llum)
            else:
                self.bot.sendMessage( message['from']['id'], 'Aquesta funcio
                    no et pertoca, no ets administrador')
                if cid > 0:
                    self.difon("El usuari " + str(m.chat.first_name) + " amb
                        ID: " + " [" + str(cid) + "]: " + " ha intentat
                        encendre la llum") # Si 'cid' es positivo, usaremos
                        'm.chat.first_name' para el nombre.
                else:
                    self.difon("El usuari " + str(m.from_user.first_name) +
                        " amb ID: " + " [" + str(cid) + "]: " + " ha
                        intentat encendre la llum")
```

– llum off: Apaga la llum i la posa en mode manual.

```
class Llum_off:
    def __init__(self, bot, peixera, l, admins, check, difon):
        self.bot = bot
        self.p = peixera
        self.l = l
        self.admins = admins
        self.check = check
        self.difon = difon
    def processMessage(self, message):
        content_type, chat_type, chat_id = telepot.glance(message)

        if content_type == 'text' and message['text'] == '/llum_off':
            if self.check(message['from']['id']):
                self.p.llum = False
                self.p.llum_mode = False
                self.l.put_nowait(self.p)
                return 'Llum: '+str(self.p.llum)
            else:
                self.bot.sendMessage( message['from']['id'], 'Aquesta funcio
                    no et pertoca, no ets administrador')
                if cid > 0:
                    self.difon("El usuari " + str(m.chat.first_name) + " amb
                        ID: " + " [" + str(cid) + "]: " + " ha intentat
                        apagar la llum") # Si 'cid' es positivo, usaremos
                        'm.chat.first_name' para el nombre.
                else:
                    self.difon("El usuari " + str(m.from_user.first_name) +
                        " amb ID: " + " [" + str(cid) + "]: " + " ha
                        intentat apagar la llum")
```

– llum auto: Deixa la llum en mode automàtic, segons l'horari esmentat abans.

```
class Llum_auto:
    def __init__(self, bot, peixera, l, admins, check, difon):
        self.bot = bot
        self.p = peixera
        self.l = l
        self.admins = admins
        self.check = check
        self.difon = difon

    def processMessage(self, message):
        content_type, chat_type, chat_id = telepot.glance(message)

        if content_type == 'text' and message['text'] == '/llum_auto':
            if self.check(message['from']['id']):
                self.p.llum = False
                self.p.llum_mode = True
                self.l.put_nowait(self.p)
                return 'Llum: '+str(self.p.llum)
            else:
                self.bot.sendMessage( message['from']['id'], 'Aquesta funcio
                    no et pertoca, no ets administrador')
                if cid > 0:
                    self.difon("El usuari " + str(m.chat.first_name) + " amb
                        ID: " + " [" + str(cid) + "]: " + " ha intentat
                        automatizar la llum") # Si 'cid' es positivo,
                        usaremos 'm.chat.first_name' para el nombre.
                else:
                    self.difon("El usuari " + str(m.from_user.first_name) +
                        " amb ID: " + " [" + str(cid) + "]: " + " ha
                        intentat automatizar la llum")
```

La càmera

És una classe de Phyton on conté les funcions per fer fotos i vídeos [2.5].

```
import datetime
import time
from picamera import PiCamera

class Camera(object):
    def foto(self):
        fname = '/home/pi/peixera2/fotos/foto_' +
            datetime.datetime.now().strftime("%Y%m%d-%H%M%S")+'.png'
        self = PiCamera()
        self.resolution = (1024, 768)
        self.rotation = 180
        self.capture(fname)
        self.close()
        return fname

    def video(self, temps):
```

```
        fname = '/home/pi/peixera2/videos/video_' +
            datetime.datetime.now().strftime("%Y%m%d-%H%M%S")+'.h264'
        if temps > 20:
            temps = 20
        self = PiCamera()
        self.resolution = (1024, 768)
        self.rotation = 180
        self.start_recording(fname, resize=(1024, 768), format='h264')
        time.sleep(temps)
        self.stop_recording()
        self.close()
        return fname
```

El servidor Python

És un servidor simple que només serveix per enviar les dades de la peixera. Fins ara només és utlitizat per la web informativa però pot ser usat per qualsevol altre. Recordar que és una pràctica del grau adaptada [iTI16] [2.7].

```
import SimpleHTTPServer
import BaseHTTPServer
import CGIHTTPServer
import sys, os
import time
import datetime
from gestio import Dades
from multiprocessing import Process, Queue
class BondiaHTTPRequestHandler(BaseHTTPServer.BaseHTTPRequestHandler):
    def _head_html(self):
        self.send_response(200)
        self.send_header("Content-type", "text/html")
        self.end_headers()
    def _head_html_img(self):
        self.send_response(200)
        self.send_header("Content-type", "image/png")
        self.end_headers()
    def _head_error_404(self):
        self.send_error(404, "File not found")
        resposta = """
        <html>
        <head>
        <title> Error 404 </title>
        </head>
        <body>
        <p>No s'ha pogut trobar la pagina</p>
        </body>
        </html>
        """
        self.wfile.write(resposta)
    def _estat_ph(self):
        while q.empty():
            pass
```

```
        p = q.get_nowait()
        self._head_html()
        self.wfile.write(str(p.ph))
    def _estat_temp(self):
        while q.empty():
            pass
        p = q.get_nowait()
        self._head_html()
        self.wfile.write(str(p.temp))
    def _grafica_ph(self):
        self._head_html_img()
        resposta = file(r"/home/pi/peixera2/ph.png", "rb").read()
        self.wfile.write(resposta)
    def _grafica_temp(self):
        self._head_html_img()
        resposta = file(r"/home/pi/peixera2/temp.png", "rb").read()
        self.wfile.write(resposta)
    def _estat_hora(self):
        self._head_html()
        self.wfile.write(datetime.datetime.now().strftime("%Y-%b-%d %H:%M:%S"))
    def _estat_co2(self):
        while q.empty():
            pass
        p = q.get_nowait()
        self._head_html()
        if p.co2:
            self.wfile.write('ON')
        else:
            self.wfile.write('OFF')
    def _estat_led(self):
        while q.empty():
            pass
        p = q.get_nowait()
        if p.llum:
            self._head_html()
            self.wfile.write('on')
        else:
            self._head_html()
            self.wfile.write('off')
    def do_GET(self):
        path = self.path
        host = self.headers['Host']
        print 'URL:{0} al servidor {1}'.format(path,host)
        if path == '/request_state':
            print "ES DEMANA L'ESTAT DEL LED"
            self._estat_led()
        elif path == '/':
            self._estat_led()
        elif path == '/ph':
            self._estat_ph()
        elif path == '/temp':
            self._estat_temp()
        elif path == '/grafica_ph':
```

```
            self._grafica_ph()
        elif path == '/grafica_temp':
            self._grafica_temp()
        elif path == '/CO2':
            self._estat_co2()
        elif path == '/hora':
            self._estat_hora()
        else:
            self._head_error_404()
def run(cua, cua_l):
    global p
    p = Dades("/home/pi/peixera2/p_casa.dat")
    PORT = 8008
    global q
    q = cua
    Handler = BondiaHTTPRequestHandler
    httpd = BaseHTTPServer.HTTPServer(("", PORT), Handler)
    print "serving at port", PORT
    httpd.serve_forever()
```

El vigilant

El vigilant[2.6] controla que tot estigui en funcionament. Reactiva els processos que queden parats perquè tot segueixi en ordre.

```
from gestio import Dades
from multiprocessing import Process, Queue
from datetime import datetime
import subprocess
import time
import ServSim2
from PeixeraBot2 import *
if __name__ == '__main__':
    peixera = Dades("/home/pi/peixera2/p_casa.dat")
    q = Queue(20)
    l = Queue(10)
    peixera.actualitza(q,l)
    bot=PeixeraBot(peixera,q,l)
    print 'ff'
    #Process(target=bot.run).start()
    feines={'alertes':[bot.genera_alertes,(''), 0], 'servidor':[ServSim2.run,
        (q,l,),0], 'actualiza':[peixera.actualitza, (q,l,),0]}
    for feina in feines:
        if feina == 'server':
            feines[feina][2]=subprocess.call(feines[feina][0])
        else:
            feines[feina][2]=Process(target=feines[feina][0], args=feines[feina][1])
            feines[feina][2].start()
    while True:
        time.sleep(5)
        print 'b'
        #peixera.actualitza(q,l)
```

```
for feina in feines:
    try:
        if feines[feina][2].is_alive():
            pass
        else:
            if feina == 'server':
                feines[feina][2]=subprocess.call(feines[feina][0])
            else:
                feines[feina][2]=Process(target=feines[feina][0],
                    args=feines[feina][1])
                feines[feina][2].start()
    except:
        if feina == 'server':
            feines[feina][2]=subprocess.call(feines[feina][0])
        else:
            feines[feina][2]=Process(target=feines[feina][0],
                args=feines[feina][1])
            feines[feina][2].start()
```

5.3. La web informativa

La web es troba en un pc de la casa el qual també allotja altres serveis, com l'owncloud, Transmission i Plex. La web està programada amb HTML i la funció és mostrar en la pantalla les dades en temps real de la peixera amb una actualització cada 5 segons. Recordar que es una pràctica del grau adaptada [iTI16]. El codi es pot trobar als apèndix[**??**].

6. La integració del sistema

La idea és col·locar tots els dispositius dins d'una capsa dissenyada[6.1] per quedar penjant de l'aquari en un lateral. Actualment però, estan en una caixa provisional[6.2]. La Raspberry està connectada directament amb un cable USB a l'Arduino. També hi ha una font d'alimentació de 12V que alimenta l'Arduino ja que s'ha d'alimentar la placa de relés i amb l'alimentació USB de la Raspberry no n'hi ha prou. El sensor de temperatura necessita un pull-up i hi he posat una resistència de 4,7 Kohms i he fet una placa per la connexió[6.4]. La Raspberry ja inclou un connector per la càmera[6.3].

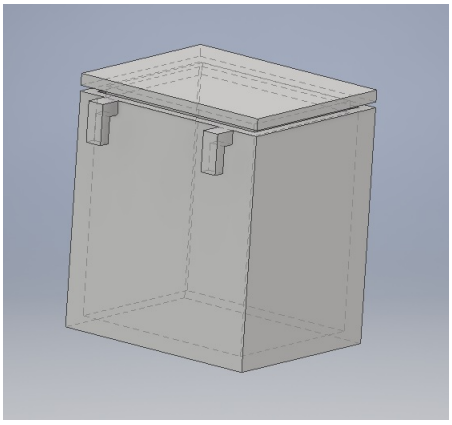

Figura 6.1.: Prototip de caixa per allotjar l'electrònica.

6.1. La interacció amb l'usuari

He intentat que la interacció sigui senzilla i fàcil a l'abast de tothom. Hi ha dues maneres de saber l'estat de la peixera, a través de l'aplicació de Telegram i a través de la web. En canvi per modificar estats només n'hi ha una, a través de l'aplicació. Ho he fet així per seguretat ja que no era l'objectiu establir un sistema de seguretat a la pagina web i de totes maneres, és una possible ampliació a fer.

6.1.1. El Bot de Telegram

A través del bot tenim totes les eines per saber en cada moment què està passant. Ja sigui consultant els paràmetres o fent fotos i vídeos. A les imatges [6.5][6.6][6.7] es veu com funciona i el comportament que té. L'exemple 1 i 2 són des d'un administrador i la tercera des d'un usuari base.

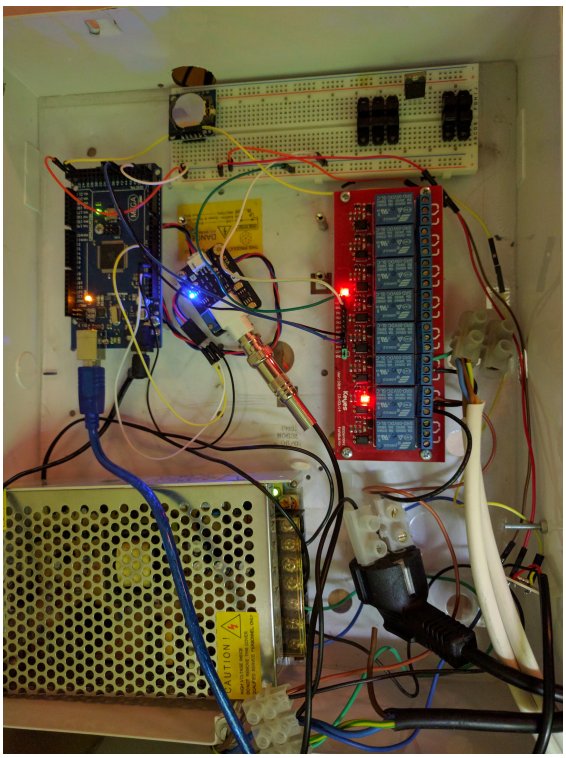

Figura 6.2.: Caixa provisional.

6.1.2. La web informativa

La web informativa, preparada per pantalles petites, és la forma més ràpida(en cas de no disposar de Telegram al dispositiu) per veure tots els paràmetres. Tot i que no pots actuar en el sistema, pots adonar-te de possibles desajustos. Primer ens trobem al menú de la pàgina principal de casa [6.8]. En el menú trobarem 'Peixera' que ens porta a la web informativa[6.9]. Aquesta web trobem un altre menú on podem anar a veure les gràfiques dels paràmetres[6.10][6.11].

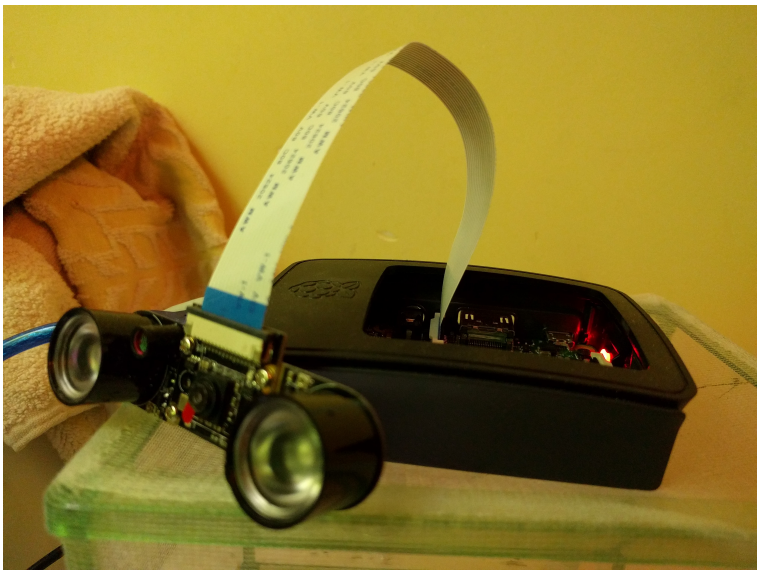

Figura 6.3.: Raspberry Pi 3 amb la càmera.

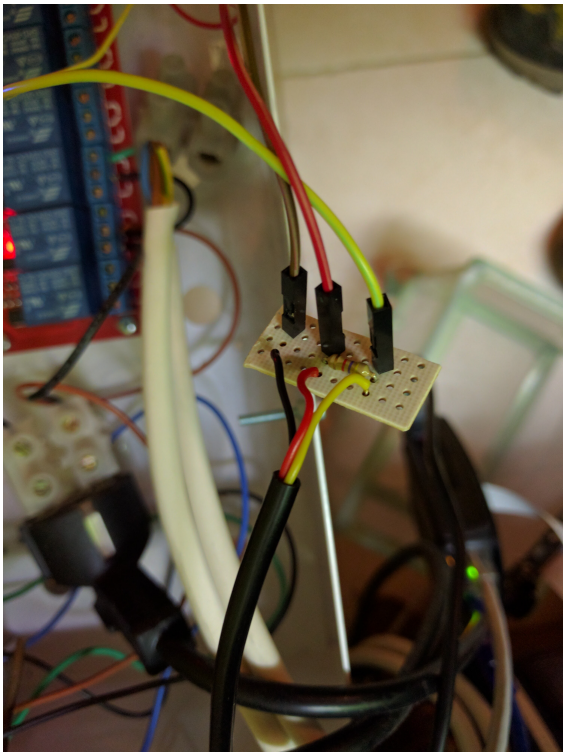

Figura 6.4.: Connexió del sensor de temperatura.

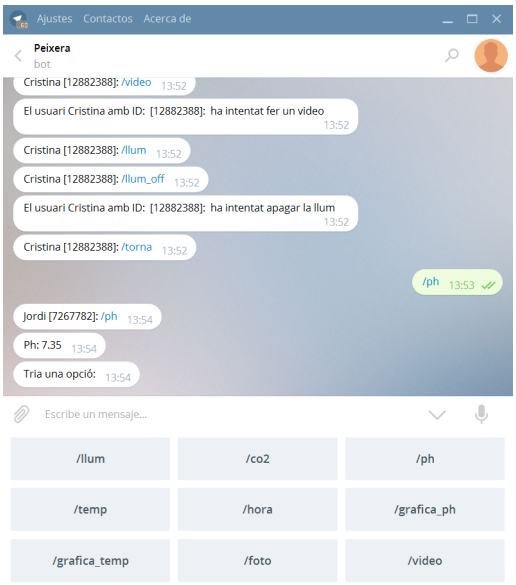

Figura 6.5.: Exemple del bot de Telegram 1.

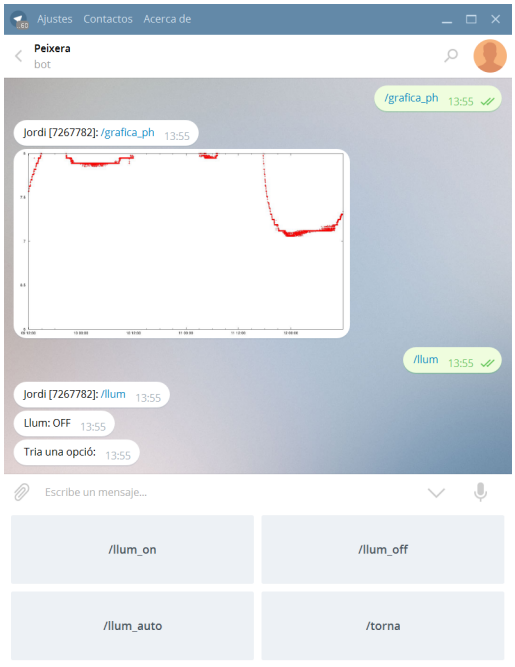

Figura 6.6.: Exemple del bot de Telegram 2.

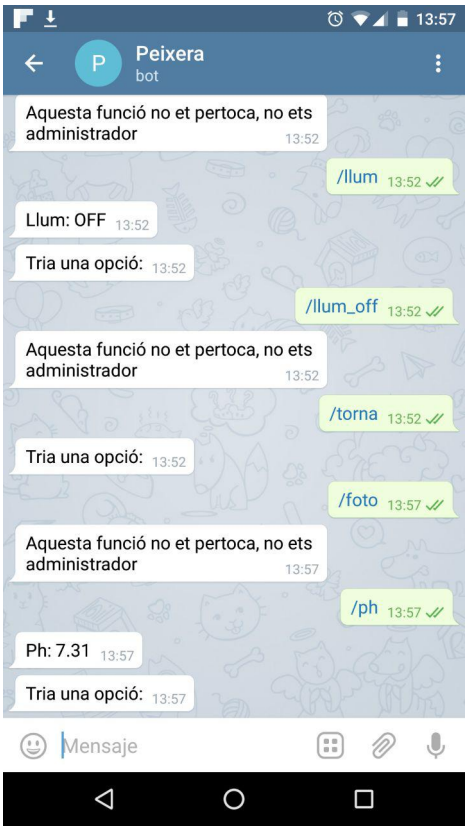

Figura 6.7.: Exemple del bot de Telegram 3.

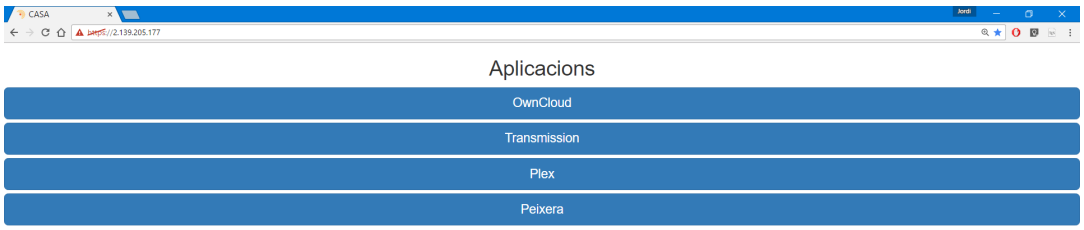

Figura 6.8.: Web inicial.

Figura 6.9.: Web informativa.

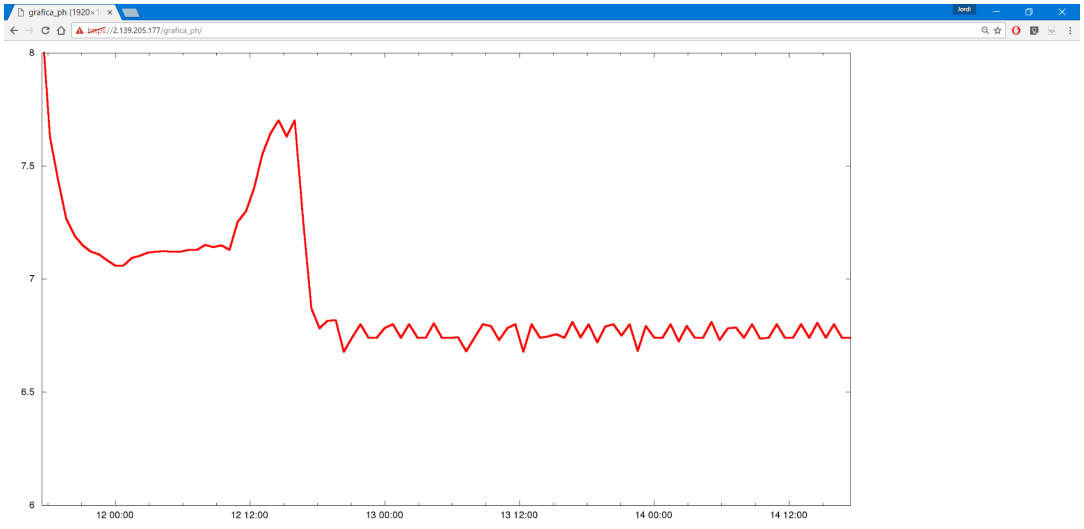

Figura 6.10.: Gràfica de pH.

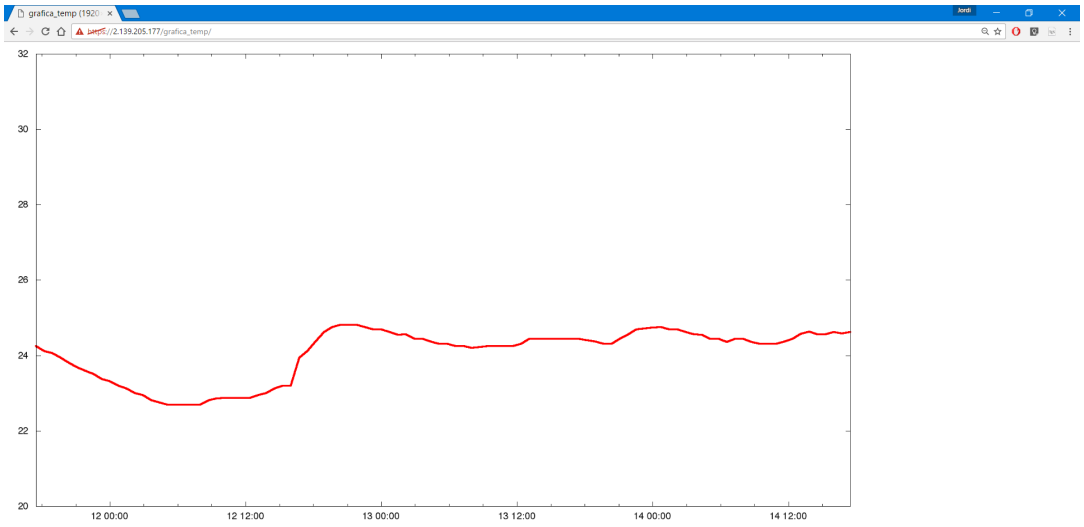

Figura 6.11.: Gràfica de temperatura.

7. Conclusions

Amb aquest projecte he aconseguit bona part dels objectius. En primer lloc he aconseguit que la qualitat de l'aigua millorés moltíssim, ja que manté els paràmetres molt més constants. En conseqüència, em lliura de prendre mesures de pH constantment, així com d'estar pendent contínuament del CO2. Seguidament relaciono els objectius plantejats inicialment amb els resultats:

- Fer la peixera més autònoma: Puc dir que la peixera es capaç de regular adientment els nivells de pH sense necessitat d'estar-hi present. M'hauria agradat poder dir que també hi ha més paràmetres que controlo, però no ha sigut possible per alguns problemes que he tingut durant el projecte.

- Reduir la feina de manteniment diari: el manteniment s'ha reduït de pràcticament cada dia a un cop per setmana. Amb la qual cosa he aconseguit millor qualitat amb menys feina.

- Anticipar-me als canvis de paràmetres i controlar la peixera a distància: Actualment puc marxar sabent que si els paràmetres es descontrolen, el sistema m'avisa a l'instant. He tingut l'oportunitat de comprovar-ho durant els darrers dies de projecte quan s'ha espatllat l'atomitzador de CO2 i ha pujat el pH.

- Aconseguir un sistema versàtil i que es pugui ampliar fàcilment: Crec que ho he aconseguit tot i que ho corroboraré més endavant quan segueixi amb el projecte.

- Aplicar tot el que he aprés durant el grau per fer un únic projecte: Objectiu assolit. Com es pot veure he aplicat els coneixements de moltes assignatures del grau.

Estic satisfet ja que he complert gran part dels objectius proposats tot i que encara tinc moltes idees per desenvolupar, que m'hauria agradat implementar-les.

El sistema aconseguit és molt més econòmic que els comercials. El cost total està entorn als 150 euros mentre que el preu de mercat per un producte similar és més del doble.

Malgrat tot, he de reconèixer que la feina no ha estat fàcil ja que hi han hagut algunes baixes i alguns components s'han avariat. El principal problema amb el que m'he trobat a l'hora de desenvolupar aquest projecte ha sigut el fet d'haver de mantenir sempre la peixera en bones condicions, cosa que no he pogut fer sempre, ja sigui per errors en el codi i altres per no haver previst un sistema paral·lel per gestionar-ho durant les aturades del sistema per fer modificacions.

Amb el projecte també he pogut posar en acció tot el que he aprés durant el grau i també he aprés coses noves com utilitzar LaTeX o fer un bot de Telegram. Crec que he aconseguit un bon sistema tot i que com ja he dit abans no te fi i per tant encara queda molt per desenvolupar.

7.1. Línies futures

Com ja he explicat anteriorment el projecte no té un final, ja que sempre es podran afegir components i millorar els serveis. per això proposo millores, tot i que algunes ja estan mig començades i pensades.

- Interfície gràfica amb una pantalla tàctil.
- Fer el manteniment del nivell de l'aigua des del sistema.
- Fer el control de temperatura també des del sistema.
- Fer un control del consum, tan elèctric com de CO2 o d'aigua.
- Adquirir sondes per altres paràmetres i integrar-les.

Bibliografia

[Aqu16] Aquamail. *Llista de preus aquamail*. 2016. URL: https://www.aquamail.com/?accion=seccion&id=142.

[Ard16] Arduino. *IDE. Arduino Software*. 2016. URL: https://www.arduino.cc/en/Main/Software.

[iTI16] OCW iTIC. *Servidor Python*. 2016. URL: http://ocwitic.epsem.upc.edu/assignatures/asi/practiques/asi.practica-7/view.

[nic16] nickoala. *Llibreria python telepot*. 2016. URL: https://github.com/nickoala/telepot.

[Ser16] Servovendi. *Regulador de pH comenrcial*. 2016. URL: http://www.servovendi.com/es/controlador-regulador-medidor-de-ph-para-acuario-piscina-ph-301.html.

[Tro16] Tropiacuariumbilbao. *Ordinador comercial per aquari*. 2016. URL: http://www.tropiacuariumbilbao.com/contents/es/p632_at_control_system.html.

[Viq16a] Viquipèdia. *Aquariofília*. 2016. URL: https://ca.wikipedia.org/wiki/Aquariof%C3%ADlia.

[Viq16b] Viquipèdia. *Arduino MEGA*. 2016. URL: https://ca.wikipedia.org/wiki/Arduino.

[Viq16c] Viquipèdia. *Raspberry Pi*. 2016. URL: https://ca.wikipedia.org/wiki/Raspberry_Pi.

[Viq16d] Viquipèdia. *Relé*. 2016. URL: https://ca.wikipedia.org/wiki/Rel%C3%A9.

[Viq16e] Viquipèdia. *Telegram*. 2016. URL: https://ca.wikipedia.org/wiki/Telegram.

Part II.

Apèndixs

Apèndixs del document

1. El software del Arduino

```
#include <DallasTemperature.h>
#include <OneWire.h>
#define CALENTADOR_IN 25
#define LG1 43
#define autoled 0
#define onled 1
#define offled 2
int intled_a = autoled;
int intled_b = autoled;
int botled_a = autoled;
int botled_b = autoled;
float p = false;
float c = false;
float l = false;
int r = 'U';
#define SensorPin 0       //pH meter Analog output to Arduino Analog Input 0
unsigned long int avgValue; //Store the average value of the sensor feedback
float b;
int buf[30],temph;
int DS18S20_Pin = 25; //DS18S20 Signal pin on digital 22
//Temperature chip i/o
OneWire ds(DS18S20_Pin); // on digital pin 22
DallasTemperature sensors(&ds);
bool e_bomba, llum, e_calentador, e_nivell;
float temperatura, phValue;
int nivell=0;
int e;
void setup () {
  Serial.begin(9600);
  pinMode(CO2, OUTPUT);
  sensors.begin();
  llums_init();
}
void loop () {
    Serial.flush();
    if (Serial.available()){
      r = Serial.read();
      if (r=='P'){
        Serial.println(phValue);
      }
      else if (r=='T'){
        Serial.println(temperatura);
```

```
  //Serial.println(26.0);
}
else if (r=='N'){
  //Serial.println(nivell);
  Serial.println(false);
}
else if (r=='C'){
  while (!Serial.available()){}
  e=Serial.read();
  if (e=='I'){
    co2(true);
    //Serial.println("OK");
  }
  else if (e=='O'){
    co2(false);
    //Serial.PRINTLN("OK");
  }
}
else if (r=='E'){
  while (!Serial.available()){}
  e=Serial.read();
  if (e=='I'){
    //co2(true);
    //Serial.PRINTLN("OK");
  }
  else if (e=='O'){
    //co2(false);
    //Serial.PRINTLN("OK");
  }
}
else if (r=='V'){
  while (!Serial.available()){}
  e=Serial.read();
  if (e=='I'){
    //co2(true);
    //Serial.PRINTLN("OK");
  }
  else if (e=='O'){
    //co2(false);
    //Serial.PRINTLN("OK");
  }
}
else if (r=='B'){
  while (!Serial.available()){}
  e=Serial.read();
  if (e=='I'){
    //co2(true);
    //Serial.PRINTLN("OK");
  }
  else if (e=='O'){
    //co2(false);
    //Serial.PRINTLN("OK");
  }
```

```
      }
      else if (r=='L'){
        while (!Serial.available()){}
        e=Serial.read();
        if (e=='I'){
          llum_auto(true);
          //Serial.PRINTLN("OK");
        }
        else if (e=='O'){
          llum_auto(false);
          //Serial.PRINTLN("OK");
        }
      }
    }

  /*
  if(!digitalRead(automatic)){
    //llum_auto(true);
  }
  else if(!digitalRead(manual)){
    llum_auto(true);
  }
  else if (digitalRead(automatic) && digitalRead(manual)){
    llum_auto(false);
  }
  */
  temp();
  ph();
  Serial.flush();
}


void co2(bool estat_co2){
  if (estat_co2){
    digitalWrite(CO2,HIGH);
    c = true;
  }
  else if (!estat_co2){
    digitalWrite(CO2,LOW);
    c = false;
  }
}


void llums_init(void){
  pinMode(LG1, OUTPUT);
  llum = true;
}

void llum_auto(bool estat){ //control de les llums
  //es fa de dia
  if (estat){
    digitalWrite(LG1, LOW);
```

```
    llum = true;
  }
  //es fa de nit
  else{
     digitalWrite(LG1,HIGH);
     llum = false;
  }
  //Serial.println("auto");
}


void ph(void){
  for(int i=0;i<30;i++)     //Get 10 sample value from the sensor for smooth the value
  {
    buf[i]=analogRead(SensorPin);
    delay(10);
  }
  for(int i=0;i<29;i++)      //sort the analog from small to large
  {
    for(int j=i+1;j<30;j++)
    {
      if(buf[i]>buf[j])
      {
        temph=buf[i];
        buf[i]=buf[j];
        buf[j]=temph;
      }
    }
  }
  avgValue=0;
  for(int i=13;i<19;i++)                   //take the average value of 6 center sample
  {
    avgValue+=buf[i];
  }
  //Serial.print(avgValue);
  phValue=map(avgValue,2148,2430,400,700);
  phValue=phValue/100;
  //Serial.println(" ");
}


void temp(void){
  sensors.requestTemperatures();
  temperatura = sensors.getTempCByIndex(0);
}
```

2. Els mòduls Python

2.1. gestió.py

```
# -*- coding: utf-8 -*-
#!usr/bin/python
```

```
from datetime import *
from comunicacio import Arduino
import time as timer
import Gnuplot, sys, os

NA_BAIX = True
NA_ALT = False
ON = True
OFF = False
AUTO = True
MANUAL = False

PH = 'P'
TEMP = 'T'
NIV = 'N'

CO2 = 'C'
CALENTADOR = 'E'
VENTILADOR = 'V'
BOMBA = 'B'
LLUM = 'L'

class Dades(object):

   def __init__(self,
       nom="/home/pi/peixera2/p_casa"+datetime.now().strftime("%Y%b%d")+".dat"):
      self.ph = 0.0
      self.temp = 0.0
      self.llum = OFF
      self.llum_mode = AUTO
      self.co2 = OFF
      self.calentador = OFF
      self.ventilador = OFF
      self.bomba_aigua = OFF
      self.nivell_aigua = NA_BAIX
      self.arduino = Arduino()
      self.nom = nom

   def guarda_dades(self):
      f = open(self.nom, 'a')
      f.write(' \n' + datetime.now().strftime("%Y-%b-%d-%H:%M:%S") + ' ')
      f.write(str(self.ph)+' '+str(self.temp)+' nivell-'+str(self.nivell_aigua)+'
          llum-'+str(self.llum)+' co2-'+str(self.co2)+'
          calentador-'+str(self.calentador)+' ventiladors-'+str(self.ventilador)+'
          bomba-'+str(self.bomba_aigua)+ ' ' + str(self.llum_mode))
      f.close()

   def grafica_ph(self):
      gp = Gnuplot.Gnuplot()
      gp('set timefmt "%Y-%b-%d-%H:%M:%S"')
      gp('set yrange [6:8]')
      gp('set xdata time')
```

```
        gp('set format x "%d %H:%M"')
        gp('set xtics font ", 16"')
        s1 = 'set xrange ["'
        s3 = '":"'
        s5 = '"]'
        ara = datetime.now()
        DD = timedelta(days=3)
        ahir = ara - DD
        s2 = ahir.strftime("%Y-%b-%d-%H:%M:%S")
        s4 = ara.strftime("%Y-%b-%d-%H:%M:%S")
        #  gp('set xrange ["Mar-15-21:00:00":"Mar-17-00:00:00"]')
        gp(s1+s2+s3+s4+s5)
        gp('set nokey')
        gp('set terminal png size 1920,1080 enhanced font "Helvetica,16"')
        gp('set output "/home/pi/peixera2/ph.png"')
        gp("plot '/home/pi/peixera2/p_casa.dat' using 1:2 title 'ph' with lines lw 4
            smooth csplines")

    def grafica_int_ph(self):
        gp = Gnuplot.Gnuplot()
        gp('set timefmt "%Y-%b-%d-%H:%M:%S"')
        gp('set yrange [6:8]')
        gp('set xdata time')
        gp('set format x "%d %H:%M"')
        gp('set xtics font ", 8"')
        s1 = 'set xrange ["'
        s3 = '":"'
        s5 = '"]'
        ara = datetime.now()
        DD = timedelta(days=3)
        ahir = ara - DD
        s2 = ahir.strftime("%Y-%b-%d-%H:%M:%S")
        s4 = ara.strftime("%Y-%b-%d-%H:%M:%S")
        #  gp('set xrange ["Mar-15-21:00:00":"Mar-17-00:00:00"]')
        gp(s1+s2+s3+s4+s5)
        gp('set nokey')
        gp('set lmargin at screen 0')
        gp('set rmargin at screen 1')
        gp('set tmargin at screen 1')
        gp('set bmargin at screen 0')
        gp('set xtics offset 0, screen 0.1')
        gp('set ytics offset screen 0.1, 0')
        gp('set terminal png size 320,144 enhanced font "Helvetica,8"')
        gp('set output "/home/pi/peixera2/ph_int.png"')
        gp("plot '/home/pi/peixera2/p_casa.dat' using 1:2 title 'ph' with lines lw 4
            smooth csplines")

    def grafica_temp(self):
        gp = Gnuplot.Gnuplot()
        gp('set timefmt "%Y-%b-%d-%H:%M:%S"')
        gp('set yrange [20:32]')
        gp('set xdata time')
        gp('set format x "%d %H:%M"')
```

```
        gp('set xtics font ", 16"')
        s1 = 'set xrange ["'
        s3 = '":"'
        s5 = '"]'
        ara = datetime.now()
        DD = timedelta(days=3)
        ahir = ara - DD
        s2 = ahir.strftime("%Y-%b-%d-%H:%M:%S")
        s4 = ara.strftime("%Y-%b-%d-%H:%M:%S")
        #  gp('set xrange ["Mar-15-21:00:00":"Mar-17-00:00:00"]')
        gp(s1+s2+s3+s4+s5)
        gp('set nokey')
        gp('set terminal png size 1920,1080 enhanced font "Helvetica,16"')
        gp('set output "/home/pi/peixera2/temp.png"')
        gp("plot '/home/pi/peixera2/p_casa.dat' using 1:3 title 'Temperatura' with
            lines lw 4 smooth csplines")

    def grafica_int_temp(self):
        gp = Gnuplot.Gnuplot()
        gp('set timefmt "%Y-%b-%d-%H:%M:%S"')
        gp('set yrange [20:32]')
        gp('set xdata time')
        gp('set format x "%d %H:%M"')
        gp('set xtics font ", 8"')
        s1 = 'set xrange ["'
        s3 = '":"'
        s5 = '"]'
        ara = datetime.now()
        DD = timedelta(days=3)
        ahir = ara - DD
        s2 = ahir.strftime("%Y-%b-%d-%H:%M:%S")
        s4 = ara.strftime("%Y-%b-%d-%H:%M:%S")
        #  gp('set xrange ["Mar-15-21:00:00":"Mar-17-00:00:00"]')
        gp(s1+s2+s3+s4+s5)
        gp('set nokey')
        gp('set lmargin at screen 0')
        gp('set rmargin at screen 1')
        gp('set tmargin at screen 1')
        gp('set bmargin at screen 0')
        gp('set xtics offset 0, screen 0.1')
        gp('set ytics offset screen 0.1, 0')
        gp('set terminal png size 320,144 enhanced font "Helvetica,8"')
        gp('set output "/home/pi/peixera2/temp_int.png"')
        gp("plot '/home/pi/peixera2/p_casa.dat' using 1:3 title 'Temperatura' with
            lines lw 4 smooth csplines")

    def actua(self):
        self.arduino.escriu(CO2, self.co2)
        self.arduino.escriu(CALENTADOR, self.calentador)
        self.arduino.escriu(VENTILADOR, self.ventilador)
        self.arduino.escriu(BOMBA, self.bomba_aigua)
        self.arduino.escriu(LLUM, self.llum)
```

```
def consulta(self):
   self.ph = self.arduino.demana(PH)
   #print self.ph
   self.temp = self.arduino.demana(TEMP)
   #print self.temp
   #self.nivell_aigua = self.arduino.demana(NIV)

def actua_llum(self, onoff, mode, q):
   self.llum_mode = mode
   self.llum = onoff
   q.put_nowait(self)

def actualitza(self, q=None, l=None):
   try:
      if not l.empty():
         while not l.empty():
            p = l.get_nowait()
         if p.llum_mode:
            self.llum_mode = AUTO
         else:
            self.llum_mode = MANUAL
         if p.llum:
            self.llum = ON
         else:
            self.llum = OFF
   except:
      pass

   self.consulta()
   if self.ph > 6.8:
      self.co2 = ON
   elif self.ph < 6.7:
      self.co2 = OFF

   if self.temp < 24.0:
      self.calentador = ON
      self.ventilador = OFF
   elif self.temp > 28.0:
      self.ventilador = ON
      self.calentador = OFF

   if self.nivell_aigua == NA_BAIX:
      self.bomba_aigua = ON
   else:
      self.bomba_aigua = OFF

   now = datetime.now()
   now_time = now.time()
   if self.llum_mode == AUTO:
      if now_time >= time(12,00) and now_time <= time(19,00):
         self.llum = ON
      else:
         self.llum = OFF
```

```
      try:
         while not q.empty():
            p = q.get_nowait()
         while not q.full():
            q.put_nowait(self)
      except:
         pass
      self.actua()
      self.guarda_dades()
      self.grafica_ph()
      self.grafica_int_ph()
      self.grafica_temp()
      self.grafica_int_temp()

   def actualitza_indef(self, q=None, l=None):
      while True:
         self.actualitza(q,l)
         timer.sleep(5)
```

2.2. comunicació.py

```
# -*- coding: utf-8 -*-
#!usr/bin/python

import serial
import time

class Arduino(object):

   def __init__(self):
      self.port = serial.Serial('/dev/ttyACM0', 9600)
      time.sleep(2)

   def demana(self, dada):
      while True:
         try:
            time.sleep(0.5)
            self.port.flush()
            #print 'demana', dada
            self.port.write(dada)
            #time.sleep(1)
            b = self.port.readline()
            #print b
            return float(b)
         except:
            print 'ERROR demana '+dada
            pass


   def escriu(self, dada, estat):
      while True:
         try:
```

```
            time.sleep(0.5)
            self.port.flush()
            #print 'escriu '+dada
            self.port.write(dada)
            #print estat
            if estat:
               self.port.write('I')
            else:
               self.port.write('O')
            #print dada+' escrit'
            return estat
         except:
            print 'ERROR escriu '+dada+str(estat)
            pass
```

2.3. PeixeraBot.py

```
# -*- coding: utf-8 -*-
#!usr/bin/python

import telepot
import time
from handlers import *
from telepot import namedtuple as types
from camerapi import Camera

class PeixeraBot:
    def __init__(self, p, q, l):
        self.users = []
        self.usuaris = [line.rstrip('\n') for line in
            open('/home/pi/peixera2/usuaris.txt')]
        self.administradors = [7267782, 4375388]
        self.alertats = [7267782, 4375388]
        BOT_TOKEN = '268635964:AAEbth9eb89eZMAFVQNsI5TpccqzvTaNxeA'
        self.bot = telepot.Bot(BOT_TOKEN)
        self.peixera = p
        self.q = q
        self.l = l
        self.cam = Camera()
        self.handlers = {'/start': Start(self.bot, self.users),
            '/stop': Stop(self.bot, self.users),
            '/help': Help(self.bot),
            '/torna/': Torna(self.bot, self.teclat_general),
            '/llum':Llum(self.bot,self.peixera, self.q, self.teclat_llum),
            '/co2': Co2(self.bot,self.peixera, self.q),
            '/ph' : Ph(self.bot,self.peixera, self.q),
            '/temp' :Temp(self.bot,self.peixera, self.q),
            '/hora' : Hora(self.bot),
            '/grafica_ph' : Grafica_ph(self.bot),
            '/grafica_temp' : Grafica_temp(self.bot),
            '/foto' : Foto(self.bot, self.cam, self.administradors, self.check_admin,
                self.difon_admin),
```

```
        '/video' : Video(self.bot, self.cam, self.administradors,
            self.check_admin, self.difon_admin),
        '/llum_on' : Llum_on(self.bot,self.peixera, self.l, self.administradors,
            self.check_admin, self.difon_admin),
        '/llum_off' : Llum_off(self.bot,self.peixera, self.l,
            self.administradors, self.check_admin, self.difon_admin),
        '/llum_auto' : Llum_auto(self.bot,self.peixera, self.l,
            self.administradors, self.check_admin, self.difon_admin),
        }

    self.bot.message_loop(self.handle_message)


def check_admin(self, cid):
    for id in self.administradors:
        if cid == id:
            return True
    return False

def difon_admin(self, m):
    for id in self.administradors:
        self.bot.sendMessage(id, m)

#Difon un missatge d'alerta
def difon_alerta(self, m):
    for id in self.alertats:
        self.bot.sendMessage(id, m)

#Teclat general
def teclat_general(self, cid):
    #markup = types.ReplyKeyboardMarkup(row_width=3)
    markup=types.ReplyKeyboardMarkup(keyboard=[['/llum', '/co2', '/ph'],['/temp',
        '/hora', '/grafica_ph'],['/grafica_temp', '/foto', '/video']])
    self.bot.sendMessage(cid, "Tria una opcio:", reply_markup=markup)

#Teclat llum
def teclat_llum(self, cid):
    #markup = types.ReplyKeyboardMarkup(row_width=2)
    markup=types.ReplyKeyboardMarkup(keyboard=[['/llum_on',
        '/llum_off'],['/llum_auto', '/torna']])
    self.bot.sendMessage(cid, "Tria una opcio:", reply_markup=markup)

def genera_alertes(self):
    while True:
        while self.q.empty():
            pass
        self.peixera = self.q.get_nowait()
        if (self.peixera.ph < 6.6 or self.peixera.ph > 6.9):
         print 'alerta'
         self.difon_alerta('PH: '+ str(self.peixera.ph))
        if (self.peixera.temp < 23.0 or self.peixera.temp > 29.0):
         print 'alerta'
         self.difon_alerta('Temperatura: '+ str(self.peixera.temp))
```

```
            time.sleep(60)

    def handle_message(self, message):
        print message
        cid = message['from']['id']
        if cid > 0:
            missatge = str(message['chat']['first_name']) + " [" + str(cid) + "]: " +
                message['text']
        else:
            missatge = str(message['from']['first_name']) + "[" + str(cid) + "]: " +
                message['text']
        f = open( '/home/pi/peixera2/log.txt', 'a')
        f.write(missatge + "\n")
        f.close()
        self.difon_admin(missatge)
        print 'ok'
        responses = []
        try:
            responses.append(self.handlers[message['text']].processMessage(message))
        except:
            for handler in self.handlers:
                responses.append(self.handlers[handler].processMessage(message))
        userId = message['from']['id']
        for response in responses:
            if response:
                self.bot.sendMessage(userId, response)
                self.teclat_general(message['from']['id'])
```

2.4. handlers.py

```
# -*- coding: utf-8 -*-
#!usr/bin/python

import telepot
from datetime import datetime

class Start:
    def __init__(self, bot, users):
        self.bot = bot
        self.users = users
        self.usuaris = [line.rstrip('\n') for line in
            open('/home/pi/peixera2/usuaris.txt')]

    def processMessage(self, message):
        content_type, chat_type, chat_id = telepot.glance(message)
        cid = message['from']['id']
        if not str(cid) in self.usuaris:
            self.usuaris.append(str(cid))
            aux = open( '/home/pi/peixera2/usuaris.txt', 'a')
            aux.write( str(cid) + "\n")
            aux.close()
```

```
        if content_type == 'text' and message['text'] == '/start':
            actualUser = message['from']['username']
            actualId = message['from']['id']
            self.users.append({actualId: actualUser})
            return "Bienvingut al @PeixeraBot!!!!"

class Stop:
    def __init__(self, bot, users):
        self.bot = bot
        self.users = users

    def processMessage(self, message):
        content_type, chat_type, chat_id = telepot.glance(message)

        actualUser = ''
        if content_type == 'text' and message['text'] == '/stop':
            actualUser = message['from']['username']
            actualId = message['from']['id']
            self.users.remove({actualId: actualUser})
            return "Fins aviat " + actualUser

class Help:
    def __init__(self, bot):
        self.bot = bot

    def processMessage(self, message):
        content_type, chat_type, chat_id = telepot.glance(message)

        if content_type == 'text' and message['text'] == '/help':
            the_help = 'Estas al PeixeraBot. Al teclat veuras les opcions que tens. '
            print the_help
            userId = message['from']['id']
            return the_help

class Torna:
    def __init__(self, bot, teclat):
        self.bot = bot
        self.teclat = teclat

    def processMessage(self, message):
        content_type, chat_type, chat_id = telepot.glance(message)

        if content_type == 'text' and message['text'] == '/torna':
            self.teclat(message['from']['id'])

class Llum:
    def __init__(self, bot, peixera, q, teclat):
        self.bot = bot
        self.p = peixera
        self.q = q
        self.teclat = teclat
        try:
            while self.q.empty():
```

```
                pass
            self.p = self.q.get_nowait()
        except:
            pass

    def processMessage(self, message):
        try:
            while self.q.empty():
                pass
            self.p = self.q.get_nowait()
        except:
            pass
        content_type, chat_type, chat_id = telepot.glance(message)

        if content_type == 'text' and message['text'] == '/llum':
            if self.p.llum:
                self.bot.sendMessage(message['from']['id'], 'Llum: ON')
            else:
                self.bot.sendMessage(message['from']['id'], 'Llum: OFF')

            self.teclat(message['from']['id'])
            #return 'Llum: '+str(self.p.llum)

class Co2:
    def __init__(self, bot, peixera, q):
        self.bot = bot
        self.p = peixera
        self.q = q
        try:
            while self.q.empty():
                pass
            self.p = self.q.get_nowait()
        except:
            pass

    def processMessage(self, message):
        try:
            while self.q.empty():
                pass
            self.p = self.q.get_nowait()
        except:
            pass
        content_type, chat_type, chat_id = telepot.glance(message)

        if content_type == 'text' and message['text'] == '/co2':
            if self.p.co2:
                return 'CO2: ON'
            else:
                return 'CO2: OFF'

class Ph:
    def __init__(self, bot, peixera, q):
        self.bot = bot
```

```
        self.p = peixera
        self.q = q
        try:
            while self.q.empty():
                pass
            self.p = self.q.get_nowait()
        except:
            pass

    def processMessage(self, message):
        try:
            while self.q.empty():
                pass
            self.p = self.q.get_nowait()
        except:
            pass
        content_type, chat_type, chat_id = telepot.glance(message)

        if content_type == 'text' and message['text'] == '/ph':
            return 'Ph: '+str(self.p.ph)

class Temp:
    def __init__(self, bot, peixera, q):
        self.bot = bot
        self.p = peixera
        self.q = q
        try:
            while self.q.empty():
                pass
            self.p = self.q.get_nowait()
        except:
            pass

    def processMessage(self, message):
        try:
            while self.q.empty():
                pass
            self.p = self.q.get_nowait()
        except:
            pass
        content_type, chat_type, chat_id = telepot.glance(message)

        if content_type == 'text' and message['text'] == '/temp':
            return 'Temperatura: '+str(self.p.temp)

class Hora:
    def __init__(self, bot):
        self.bot = bot

    def processMessage(self, message):

        content_type, chat_type, chat_id = telepot.glance(message)
```

```
        if content_type == 'text' and message['text'] == '/hora':
            return 'Hora: '+datetime.now().strftime("%Y-%b-%d-%H:%M:%S")

class Grafica_ph:
    def __init__(self, bot):
        self.bot = bot

    def processMessage(self, message):
        content_type, chat_type, chat_id = telepot.glance(message)
        if content_type == 'text' and message['text'] == '/grafica_ph':
            self.bot.sendPhoto( message['from']['id'], open(
                '/home/pi/peixera2/ph.png', 'rb'))

class Grafica_temp:
    def __init__(self, bot):
        self.bot = bot

    def processMessage(self, message):
        content_type, chat_type, chat_id = telepot.glance(message)
        if content_type == 'text' and message['text'] == '/grafica_temp':
            self.bot.sendPhoto( message['from']['id'], open(
                '/home/pi/peixera2/temp.png', 'rb'))

class Foto:
    def __init__(self, bot, cam, admins, check, difon):
        self.bot = bot
        self.cam = cam
        self.admins = admins
        self.check = check
        self.difon = difon

    def processMessage(self, message):
        content_type, chat_type, chat_id = telepot.glance(message)


        if content_type == 'text' and message['text'] == '/foto':
            if self.check(message['from']['id']):
                self.bot.sendPhoto( message['from']['id'], open( self.cam.foto(),
                    'rb'))
            else:
                self.bot.sendMessage( message['from']['id'], 'Aquesta funcio no et
                    pertoca, no ets administrador')
                if message['from']['id'] > 0:
                    self.difon("El usuari " + str(message['from']['first_name']) + "
                        amb ID: " + " [" + str(message['from']['id']) + "]: " + " ha
                        intentat fer una foto")
                else:
                    self.difon("El usuari " + str(message['from']['first_name']) + "
                        amb ID: " + " [" + str(message['from']['id']) + "]: " + " ha
                        intentat fer una foto")

class Video:
    def __init__(self, bot, cam, admins, check, difon):
```

```
        self.bot = bot
        self.cam = cam
        self.admins = admins
        self.check = check
        self.difon = difon

    def processMessage(self, message):
        content_type, chat_type, chat_id = telepot.glance(message)

        if content_type == 'text' and message['text'] == '/video':
            if self.check(message['from']['id']):
                self.bot.sendVideo(message['from']['id'], open( self.cam.video(10),
                    'rb'))
            else:
                self.bot.sendMessage( message['from']['id'], 'Aquesta funcio no et
                    pertoca, no ets administrador')
                if message['from']['id'] > 0:
                    self.difon("El usuari " + str(message['from']['first_name']) + "
                        amb ID: " + " [" + str(message['from']['id']) + "]: " + " ha
                        intentat fer un video")
                else:
                    self.difon("El usuari " + str(message['from']['first_name']) + "
                        amb ID: " + " [" + str(message['from']['id']) + "]: " + " ha
                        intentat fer un video")

class Llum_on:
    def __init__(self, bot, peixera, l, admins, check, difon):
        self.bot = bot
        self.p = peixera
        self.l = l
        self.admins = admins
        self.check = check
        self.difon = difon

    def processMessage(self, message):
        content_type, chat_type, chat_id = telepot.glance(message)

        if content_type == 'text' and message['text'] == '/llum_on':
            if self.check(message['from']['id']):
                self.p.llum = True
                self.p.llum_mode = False
                self.l.put_nowait(self.p)
                return 'Llum: '+str(self.p.llum)
            else:
                self.bot.sendMessage( message['from']['id'], 'Aquesta funcio no et
                    pertoca, no ets administrador')
                if message['from']['id'] > 0:
                    self.difon("El usuari " + str(mmessage['from']['first_name']) + "
                        amb ID: " + " [" + str(message['from']['id']) + "]: " + " ha
                        intentat encendre la llum")
                else:
                    self.difon("El usuari " + str(message['from']['first_name']) + "
                        amb ID: " + " [" + str(message['from']['id']) + "]: " + " ha
```

```
                    intentat encendre la llum")

class Llum_off:
    def __init__(self, bot, peixera, l, admins, check, difon):
        self.bot = bot
        self.p = peixera
        self.l = l
        self.admins = admins
        self.check = check
        self.difon = difon

    def processMessage(self, message):
        content_type, chat_type, chat_id = telepot.glance(message)

        if content_type == 'text' and message['text'] == '/llum_off':
            if self.check(message['from']['id']):
                self.p.llum = False
                self.p.llum_mode = False
                self.l.put_nowait(self.p)
                return 'Llum: '+str(self.p.llum)
            else:
                self.bot.sendMessage( message['from']['id'], 'Aquesta funcio no et
                    pertoca, no ets administrador')
                if message['from']['id'] > 0:
                    self.difon("El usuari " + str(message['from']['first_name']) + "
                        amb ID: " + " [" + str(message['from']['id']) + "]: " + " ha
                        intentat apagar la llum")
                else:
                    self.difon("El usuari " + str(message['from']['first_name']) + "
                        amb ID: " + " [" + str(message['from']['id']) + "]: " + " ha
                        intentat apagar la llum")


class Llum_auto:
    def __init__(self, bot, peixera, l, admins, check, difon):
        self.bot = bot
        self.p = peixera
        self.l = l
        self.admins = admins
        self.check = check
        self.difon = difon

    def processMessage(self, message):
        content_type, chat_type, chat_id = telepot.glance(message)

        if content_type == 'text' and message['text'] == '/llum_auto':
            if self.check(message['from']['id']):
                self.p.llum = False
                self.p.llum_mode = True
                self.l.put_nowait(self.p)
                return 'Llum: '+str(self.p.llum)
            else:
                self.bot.sendMessage( message['from']['id'], 'Aquesta funcio no et
```

```
                pertoca, no ets administrador')
            if message['from']['id'] > 0:
                self.difon("El usuari " + str(message['from']['first_name']) + "
                    amb ID: " + " [" + str(message['from']['id']) + "]: " + " ha
                    intentat automatizar la llum")
            else:
                self.difon("El usuari " + str(message['from']['first_name']) + "
                    amb ID: " + " [" + str(message['from']['id']) + "]: " + " ha
                    intentat automatizar la llum")
```

2.5. camerapi.py

```
# -*- coding: utf-8 -*-
#!usr/bin/python

import datetime
import time
from picamera import PiCamera

class Camera(object):
    def foto(self):
        fname = '/home/pi/peixera2/fotos/foto_' +
            datetime.datetime.now().strftime("%Y%m%d-%H%M%S")+'.png'
        self = PiCamera()
        self.resolution = (1024, 768)
        self.rotation = 180
        self.capture(fname)
        self.close()
        return fname

    def video(self, temps):
        fname = '/home/pi/peixera2/videos/video_' +
            datetime.datetime.now().strftime("%Y%m%d-%H%M%S")+'.h264'
        if temps > 20:
            temps = 20
        self = PiCamera()
        self.resolution = (1024, 768)
        self.rotation = 180
        self.start_recording(fname, resize=(1024, 768), format='h264')
        time.sleep(temps)
        self.stop_recording()
        self.close()
        return fname
```

2.6. vigilant.py

```
# -*- coding: utf-8 -*-
#!usr/bin/python

from gestio import Dades
from multiprocessing import Process, Queue
from datetime import datetime
```

```
import subprocess
import time
import ServSim2
from PeixeraBot2 import *

if __name__ == '__main__':
    peixera = Dades("/home/pi/peixera2/p_casa.dat")
    q = Queue(20)
    l = Queue(10)
    peixera.actualitza(q,l)
    bot=PeixeraBot(peixera,q,l)
    print 'ff'
    #Process(target=bot.run).start()

    feines={'alertes':[bot.genera_alertes,('')], 0], 'servidor':[ServSim2.run,
        (q,l,),0], 'actualiza':[peixera.actualitza_indef, (q,l,),0]}
    for feina in feines:
        if feina == 'server':
            feines[feina][2]=subprocess.call(feines[feina][0])
        else:
            feines[feina][2]=Process(target=feines[feina][0], args=feines[feina][1])
            feines[feina][2].start()
    while True:
        time.sleep(5)
        print 'b'
        #peixera.actualitza(q,l)
        for feina in feines:
            try:
                if feines[feina][2].is_alive():
                    pass
                else:
                    if feina == 'server':
                        feines[feina][2]=subprocess.call(feines[feina][0])
                    else:
                        feines[feina][2]=Process(target=feines[feina][0],
                            args=feines[feina][1])
                        feines[feina][2].start()
            except:
                if feina == 'server':
                    feines[feina][2]=subprocess.call(feines[feina][0])
                else:
                    feines[feina][2]=Process(target=feines[feina][0],
                        args=feines[feina][1])
                    feines[feina][2].start()
```

2.7. ServSim.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import SimpleHTTPServer
import BaseHTTPServer
```

```
import CGIHTTPServer
import sys, os
import time
import datetime
from gestio import Dades
from multiprocessing import Process, Queue


class BondiaHTTPRequestHandler(BaseHTTPServer.BaseHTTPRequestHandler):

    def _head_html(self):
        self.send_response(200)
        self.send_header("Content-type", "text/html")
        self.end_headers()

    def _head_html_img(self):
        self.send_response(200)
        self.send_header("Content-type", "image/png")
        self.end_headers()

    def _head_error_404(self):
        self.send_error(404, "File not found")
        resposta = """
        <html>
        <head>
        <title> Error 404 </title>
        </head>
        <body>
        <p>No s'ha pogut trobar la pagina</p>
        </body>
        </html>
        """
        self.wfile.write(resposta)

    def _bon_dia(self):

        self._head_html()
        resposta ="""<html>
      <head>
        <title>Bon dia</title>
        </head>
        <body>
        <p>Bon dia</p>
        </body>
        </html>
        """
        self.wfile.write(resposta)
    """
    def _encen_led(self):
        arduinoPort.write('E')
        getSerialValue = arduinoPort.readline()
        print "S'ENTRA A ENCEN LED"
   print getSerialValue
```

```
        self._head_html()
        resposta = "on"
        self.wfile.write(resposta)

    def _apaga_led(self):
        arduinoPort.write('A')
        getSerialValue = arduinoPort.readline()
        print "S'ENTRA A APAGA LED"
    print getSerialValue
        self._head_html()
        resposta = "off"
        self.wfile.write(resposta)

    def _auto_led(self):
        arduinoPort.write('U')
        getSerialValue = arduinoPort.readline()
        print "S'ENTRA A auto LED"
    print getSerialValue
        self._head_html()
        resposta = "auto"
        self.wfile.write(resposta)
    """
    def _estat_ph(self):
        while q.empty():
            pass
        p = q.get_nowait()
        self._head_html()
        self.wfile.write(str(p.ph))

    def _estat_temp(self):
        while q.empty():
            pass
        p = q.get_nowait()
        self._head_html()
        self.wfile.write(str(p.temp))

    def _grafica_ph(self):
        self._head_html_img()
        resposta = file(r"/home/pi/peixera2/ph.png", "rb").read()
        self.wfile.write(resposta)

    def _grafica_temp(self):
        self._head_html_img()
        resposta = file(r"/home/pi/peixera2/temp.png", "rb").read()
        self.wfile.write(resposta)

    def _estat_hora(self):
        self._head_html()
        self.wfile.write(datetime.datetime.now().strftime("%Y-%b-%d %H:%M:%S"))

    """
    def _posa_hora(self):
        arduinoPort.write('h')
```

```
        getSerialValue = arduinoPort.readline()
        print "S'ENTRA A EDITA HORA"
        print getSerialValue
        self._head_html()
    if getSerialValue == 'OK\n\r':
       arduinoPort.write(time.strftime("%b %d %y"))
       arduinoPort.write(time.strftime("%H %M %S"))
            getSerialValue = arduinoPort.readline()
       self.wfile.write(getSerialValue)
    else:
            getSerialValue = arduinoPort.readline()
            self.wfile.write(getSerialValue)
     """

     def _estat_co2(self):
         while q.empty():
             pass
         p = q.get_nowait()
         self._head_html()
         if p.co2:
             self.wfile.write('ON')
         else:
             self.wfile.write('OFF')

     def _estat_led(self):
         while q.empty():
             pass
         p = q.get_nowait()
         if p.llum:
             self._head_html()
             self.wfile.write('on')
         else:
             self._head_html()
             self.wfile.write('off')

     def do_GET(self):
         path = self.path
         host = self.headers['Host']
         print 'URL:{0} al servidor {1}'.format(path,host)
         if path == '/request_state':
             print "ES DEMANA L'ESTAT DEL LED"
             self._estat_led()
         elif path == '/':
             self._estat_led()
         elif path == '/ph':
             self._estat_ph()
         elif path == '/temp':
             self._estat_temp()
         elif path == '/grafica_ph':
             self._grafica_ph()
         elif path == '/grafica_temp':
             self._grafica_temp()
         elif path == '/CO2':
```

```
            self._estat_co2()
        elif path == '/hora':
            self._estat_hora()
        else:
            self._head_error_404()
        """
        elif path == '/posahora':
            self._posa_hora()
        elif path == '/on':
            print "ES VOL ENCENDRE EL LED"
            self._encen_led()
        elif (path == '/off'):
            print "ES VOL APAGAR EL LED"
            self._apaga_led()
       elif (path == '/auto'):
            print "ES VOL AUTOMATITZAR EL LED"
            self._auto_led()
        """

def run(cua, cua_l):
    global p
    p = Dades("/home/pi/peixera2/p_casa.dat")
    PORT = 8008
    global q
    q = cua
    global l
    l = cua_l
    Handler = BondiaHTTPRequestHandler
    httpd = BaseHTTPServer.HTTPServer(("", PORT), Handler)
    print "serving at port", PORT
    httpd.serve_forever()



\section{El codi font de la web}
\label{web}
\begin{lstlisting}[style=HTML]
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Peixera</title>
    <link rel="shorcut icon" href="logo_koi.jpg">
    <link rel="stylesheet" type="text/css" href="styles.css">
    <script src="jquery-1.11.2.min.js"></script>
    <script
        src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet"
        href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css">
    <script
        src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min.js"></script>
    <script
```

```
    src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>
<script header="Access-Control-Allow-Origin: *"></script>
<script>
  window.onload = function updateLed() {
  var xmlhttp;
  var obj_div;
  obj_div = document.getElementById("estatLed");
  xmlhttp=new XMLHttpRequest();
  xmlhttp.onreadystatechange = function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
  {
  if (xmlhttp.responseText=="off")
  {
  obj_div.style.color="red";
  obj_div.innerHTML="OFF";
  console.log("updateLed OFF");
  }
  if (xmlhttp.responseText=="on")
  {
  obj_div.style.color="green";
  obj_div.innerHTML="ON";
  }
  }
  }
  xmlhttp.open("GET","/estat/", true);
  xmlhttp.send();
  }
  function loadXMLDoc() {
  var xmlhttp;
  var obj_div;
  obj_div = document.getElementById("estatLed");
  xmlhttp=new XMLHttpRequest();
  xmlhttp.onreadystatechange = function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
  {
  console.log(xmlhttp.responseText);
  if (xmlhttp.responseText=="off")
  {
  obj_div.style.color="red";
  obj_div.innerHTML="OFF";
  console.log("load OFF");
  }
  if (xmlhttp.responseText=="on")
  {
  obj_div.style.color="green";
  obj_div.innerHTML="ON";
  }
  }
  }
  xmlhttp.open("GET","/estat/", true);
  xmlhttp.send();
```

```
}
function grafica_ph(){
var xmlhttp;
var obj_div;
obj_div = document.getElementById("grafica_ph");
xmlhttp=new XMLHttpRequest();
xmlhttp.onreadystatechange=function()
{
if (xmlhttp.readyState==4 && xmlhttp.status==200)
{
obj_div.style.color="red";
obj_div.innerHTML=xmlhttp.responseText;
}
}
xmlhttp.open("GET", "/grafica_ph/", true);
xmlhttp.send();
}
function grafica_temp(){
var xmlhttp;
var ogj_div;
obj_div = document.getElemntById("grafica_temp");
xmlhttp=new XMLHttpRequest();
xmlhttp.onreadystatechange=function()
{
if (xmlhttp.readyState==4 && xmlhttp.status==200)
{
obj_div.style.color="red";
obj_div.innerHTML=xmlhttp.responseText;
}
}
xmlhttp.open("GET", "/grafica_temp/", true);
xmlhttp.send();
}
function estatPh(){
var xmlhttp;
var obj_div;
obj_div = document.getElementById("estatPh");
xmlhttp=new XMLHttpRequest();
xmlhttp.onreadystatechange=function()
{
if (xmlhttp.readyState==4 && xmlhttp.status==200)
{
obj_div.style.color="black";
obj_div.innerHTML=xmlhttp.responseText;
}
}
xmlhttp.open("GET", "/ph/", true);
xmlhttp.send();
}
function estatTemp(){
var xmlhttp;
var obj_div;
obj_div = document.getElementById("estatTemp");
```

```
xmlhttp=new XMLHttpRequest();
xmlhttp.onreadystatechange=function()
{
if (xmlhttp.readyState==4 && xmlhttp.status==200)
{
obj_div.innerHTML=xmlhttp.responseText;
}
}
xmlhttp.open("GET", "/temp/", true);
xmlhttp.send();
}
function estatCO2(){
var xmlhttp;
var obj_div;
obj_div = document.getElementById("estatCO2");
xmlhttp=new XMLHttpRequest();
xmlhttp.onreadystatechange = function()
{
if (xmlhttp.readyState==4 && xmlhttp.status==200)
{
console.log(xmlhttp.responseText);
obj_div.style.color="black";
obj_div.innerHTML=xmlhttp.responseText;

if (xmlhttp.responseText=="OFF")
{
obj_div.style.color="red";
obj_div.innerHTML=xmlhttp.responseText;
console.log("load OFF");
}
if (xmlhttp.responseText=="ON")
{
obj_div.style.color="green";
obj_div.innerHTML=xmlhttp.responseText;;
}
}
}
xmlhttp.open("GET","/CO2/", true);
xmlhttp.send();
}
function estatHora(){
var xmlhttp;
var obj_div;
obj_div = document.getElementById("estatHora");
xmlhttp=new XMLHttpRequest();
xmlhttp.onreadystatechange=function()
{
if (xmlhttp.readyState==4 && xmlhttp.status==200)
{
obj_div.style.color="black";
obj_div.innerHTML=xmlhttp.responseText;
}
}
```

```
    xmlhttp.open("GET", "/hora/", true);
    xmlhttp.send();
    }
  </script>
</head>
<body>
  <div align="center" class="header">
    <h1>Estat de la peixera</h1>
  </div>
  <center class="container">
    <p>Llum: <div id="estatLed" style="font-weight: bold; font-size:
        xx-large"></div></p>
    <p>CO2: <div id="estatCO2" style="font-weight: bold; font-size:
        xx-large"></div></p>
    <p>PH: <div id="estatPh" style="font-weight: bold; font-size:
        xx-large"></div></p>
    <p>Temperatura: <div id="estatTemp" style="font-weight: bold; font-size:
        xx-large"></div></p>
    <p>Hora: <div id="estatHora" style="font-weight: bold; font-size:
        xx-large"></div></p>
  </center>
  <div class="buttons">
    <input class="btn btn-primary btn-lg btn-block" style="float: left;
        background-color: blue" type="submit" value="Grafica ph" id="boto_auto"
        onClick="window.location='/grafica_ph/';"/>
    <input class="btn btn-primary btn-lg btn-block" style="float: left;
        bachground-color: blue" type="submit" value="Grafica Temperatura"
        id="boto_auto" onClick="window.location='/grafica_temp/';"/>
    <input class="btn btn-primary btn-lg btn-block" style="float: left"
        type="submit" value="Tornar" id="boto_tornar"
        onClick="window.location='https://2.139.205.177/index.html';"/>
  </div>
  <script>
    setInterval("loadXMLDoc()",5000);
  </script>
  <script>
    setInterval("estatPh()",5000);
  </script>
  <script>
    setInterval("estatTemp()",5000);
  </script>
  <script>
    setInterval("estatHora()",5000);
  </script>
  <script>
    setInterval("estatCO2()",5000);
  </script>
</body>
</html>
```