Practica final del curs plaques robòtiques

Domotització de residència amb plaques robòtiques

INDEX

Barcelona, 26 de novembre de 2020

Joan Faig Martí

Descripció

La idea és desenvolupar un sistema IOT (Internet of Things) de domotització domèstica per tal de controlar una residencia a distancia. El control implica tant la monitorització de l'entorn (com la temperatura o diverses alarmes com fugues o nivell del dipòsit d'aigua o la intrusió) com una certa actuació, com encendre la calefacció a distància o encendre algun llum com a mesura de seguretat preventiva.

El sistema reb dades de sensors d'àmbit domèstic i en funció de cert procés de càlcul, pot actuar sobre uns actuadors i enviar les dades a través de internet als usuaris a través d'un dispositiu (ordinador, tablet, mòbil)

Disseny

Es preveu un disseny modular per disposar d'uns quants sensors i actuadors i fer-los actuar de la forma més similar possible (sistematitzar i estructurar els processos) de forma que l'afegit o eliminació de un dispositiu sigui el més simple possible.

Per anar emmagatzemant les dades que es generin i es disposi d'accés des de l'exterior, es connecta un Arduino a un ordinador Raspberry Pi, on hi haurà instal·lada una base de dades i un servidor web que donarà accés a través de una pàgina web als dispositius externs, com ordinadors, tablets i mòbils.

Seqüència de monitorització d'un sensor:
Sensor 1 -->Arduino --->RaspberryPi--->Client

Seqüència de actuació sobre un actuador:
Client -->RaspberryPi --->Arduino--->Sensor 1

Connexió Arduino--Raspberry: Port Serie
Connexió RaspberryPi --Clients: Wifi

Serveis a Arduino:

1. Recollida de dades de sensors i enviament a la Raspberry Pi pel port serie
2. Actuació sobre els actuadors en funcio de certs càlculs inters i/o dades rebudes pel port serie desde la Raspberry Pi

Serveis a Raspberry Pi:

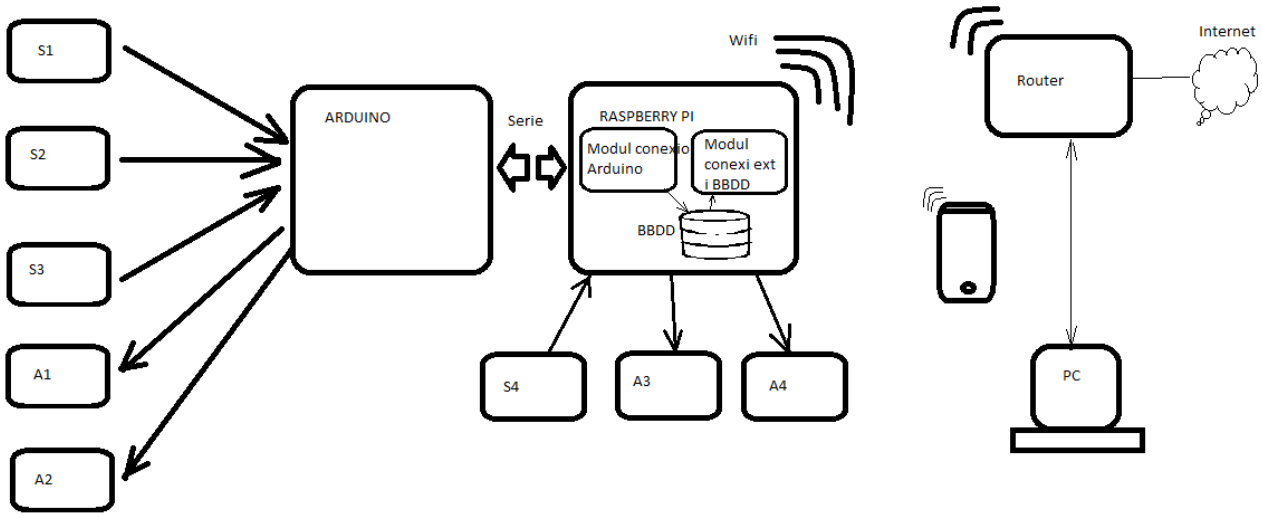
1. Servei de recollida de dades de Arduino a través del port serie i altres sensors pel GPIO
2. Servei de enviament de dades a Sistema de Gestió de Base de Dades (SGBD)

3. Servei web per enviar dades a clients: Apache o NGINX desenvolupament amb Python i modul Flask, Pg Web a desenvolupar
4. Servei de Gestió de base de dades SGBD: MySQL, SQLite, etc

SW a clients:

1. Accés per web browser: Chrome, IE, Edge, etc

Esquema General:



Desenvolupament

De cara a desenvolupar el projecte, es comença amb un cas d'ús ben senzill; es tracta de connectar un detector de temperatura i humitat al sistema i es pugui observar des d'un client extern a la xarxa (accés des de browser web).

Tasques unitàries bàsiques:

1. Es connecta el sensor DHT11 a l'Arduino i es programa l'Arduino per tal de enviar per port sèrie a la RaspPi les dades de temperatura i humitat cada 10 minuts (configurable).
   Components:
   1. Arduino Uno
   2. Sensor DHT11
   3. Cables i connectors
   4. Programa a Arduino: **ProjecteFinal_[x].ino** (essent [x] un número de versió), última versió a l'annex 1

Per independitzar cada sensor/actuador de la resta, cal fer un **multithreading**.
Cada Sensor o actuador disposa de un temps d'actuació timeSx (per sensors) o timeAx (per actuadors) (essent x el numero de dispositiu corresponent) i el loop anirà actuant a cada dispositiu en funció de la seguent peça de codi:

```
void loop() {
 temps=millis();
 if((temps-anteriorS1) > timeS1){
   anteriorS1= temps;
   actuaS1();                          // funció de l'actuació sobre dispositu S1
 }
 if((temps-anteriorA1) > timeA1){
   anteriorA1= temps;
   actuaA1();                          // funció de l'actuació sobre dispositu A1
 }
 if((temps-anteriorA2) > timeA2){
   anteriorA2= temps;
   actuaA2();                          // funció de l'actuació sobre dispositu A2
 }
.....                                  // altres threads pe A3, A4, A5, S2, S3, S4...
}
```

2. Es programa la RaspPi de forma que rebi les dades del port sèrie i les emmagatzemi a una BBDD sqlite en una taula del sensor
   Components:
   1. Elements punt 1.
   2. Raspberri Pi
   3. Cables de connexió
   4. Programa de recepció de dades i introducció a BBDD
   5. BBDD SQLite

   Pas 1: Volcat Raspberri Pi OS Lite (de https://howtoraspberrypi.com/downloads/) a RPi mitjançant aplicació Balena Etcher
   Pas 2: configuració de Wifi a la xarxa del CIFO
   Pas 3: Habilitació del servidor SSH a la raspberry
   Pas4: connexió de l'arduino a RPI per port sèrie: Primer cal baixar-se el mòdul de Serie a python de la RPI mitjançant:

   sudo apt-get update

   sudo apt-get upgrade

   sudo apt-get install python-pip

   sudo python -m pip install pyserial

   Pas 5: Instal·lació de SQLite a RPi: **sudo apt-get install sqlite3**

Programa base a Rasberry Pi: **projecteFinal_[x].py** (essent [x] un numero de versió)

EL programa obre primer una base de dades per tal d'emmagatzemar temperatures, humitats, posicions , nivells, estats de actuadors.

En la situació actual conté dues taules:

DHT_11: taula que emmagatzema

- la temperatura de la casa, rebuda del sensor de l'arduino
- la seva humitat, rebuda del sensor de l'arduino
- data i hora de cada mesura

CALEF: Taula que emmagatzema

- Estat de la calefacció ; si té l'ordre d'encendre o apagar-se. Rebut dels botons d'activació/apagament dins la pàgina web
- Si esta en mode manual o mode automàtic (encara no implementat) Rebut de la casella dins la pàgina web
- data i hora de cada mesura

A continuació, el programa entra en un bucle en que va rebent dades del port serie i les va emmagatzemant a la taula SHT_11 de la base de dades.

```
data=arduino.readline()
if b"Temperature: " in data:
    temps = datetime.datetime.now()
     valueTemp=float(regex.findall(str(data))[0])
if b"Humidity: " in data:
    temps= datetime.datetime.now()
     valueHum=float(regex.findall(str(data))[0])
    query="INSERT INTO DHT_11( ID, TEMP, HUM, TEMPS) " "VALUES(:ID, :TEMP, :HUM, :TEMPS)"
    params={'ID':contador, 'TEMP':valueTemp, 'HUM':valueHum, 'TEMPS':temps}
     conn.execute(query, params)
    conn.commit()
```

A més, un cop ha rebut dades, envia a l'Arduino el estat de la calefacció (per encendre-la o apagar-la) extret de la tabla CALEF de la base de dades

```
 queryCalef="""select ENCES from CALEF order by rowid desc limit 1"""
cursor4= conn.execute(queryCalef)
ences=cursor4.fetchall()[0][0]

if ences == 0:
    arduino.write(b"0")
if ences == 1:
    arduino.write(b"1")
```

3. S’activa un servidor web. En comptes de un servidor Apache o NGINX, es crea una aplicació python amb els mòduls Flask i una pàgina web en Html de forma que es llegeixin les mesures del sensor DHT11. S’ha de obrir ports al router per tal de tenir accés des de l’exterior [TBD] Al tanto amb la seguretat!

   Components

   1. Servidor Web
   2. Accés al router
   3. Pàgina HTML

Programa base a Raspberry Pi: **servidorDomotic_[x].py** (essent [x] un número de versió)

Programa desenvolupat en python amb modul Flask , per tal de actuar com a servidor web.

La funció principal (hello()) llegeix els valor de temperatura, humitat, temps i estat de la calefacció i els envia a la pàgina web index.html

```
queryTemp="""select TEMP from DHT_11 order by rowid desc limit 1"""
 cursor = conn.execute(queryTemp)
  temperatura=cursor.fetchall()[0][0]
.....
  if ences==0:
     estatCalef="APAGAT"
  else:
     estatCalef="ENCES"
  conn.close()

  now = datetime.datetime.now()
  templateData = {
    'title' : 'DOMUS!',
    'time': temps,
    'temper': temperatura,
    'hum': humitat,
    'encenCalef': estatCalef
    }
  return render_template('index.html', **templateData)
```

La funció encendreCalef i apagaCalef fasn les mateixes funcions de llegeix que hello, però a més, comproba l'estat del boto d'encendre i apagar, i en cas de ser diferent a l'ultim valor enregistrat a la base de dades, escriu un nou registre amb el nou estat

Bloc de codi de encendreCalef (per apagaCalef, la variable encenCalef te els valors canviats)

```
if encenCalef==0:                                   # Si troba un estat anterior apagat i s'ha pres botó d'encndre
   encenCalef=1                                     # Activem la calefacció
  estatCalef="ENCES"                                # i ho imprimim per pantalla
   automCalef=0
   now = datetime.datetime.now()
   query="INSERT INTO CALEF( ID, ENCES, AUTOMATIC, TEMPS) " "VALUES(:ID, :ENCES, :AUTOMATIC, :TEMPS)"
   params={'ID':contador+1, 'ENCES':encenCalef, 'AUTOMATIC':automCalef, 'TEMPS':now}
   conn.execute(query, params)                      # i ho desem com a registre a la BBDD
   conn.commit()
conn.close()
```

Arxiu html: *templates/*index.html

A la pàgina web es veuran els valor de temps i data de l'ultim valor enregistrat, la seva temperatura, humitat i si la calefacció està encesa o apagada.

```
<h2> Temps i hora de l'ultima dada: {{ time }}</h2>
```

```
<h3> Temperatura: {{ temper }}ºC i Humitat: {{ hum }}%</h3>
<h3> Calefacció: {{ encenCalef }}</h3>
```

A més disposa de dos botons per encendre i apagar la calefacció

```
<h3> Prem ON o OFF per apagar/encendre la calefacció </h3>
<form action="/calefON/" method="post">
        <button name="calefON" type="submit">ON</button>
</form>
<form action="/calefOFF/" method="post">
        <button name="calefOFF" type="submit">OFF</button>
</form>
```

4. Es comprova accés des de l'exterior a través d'un browser Web d'un telèfon mòbil i d'un ordinador.

Tasques següents:

1. Es connecta un altre sensor [TBD], configurant tot el procés bàsic descrit anteriorment
2. Es connecta un actuador [TBD]

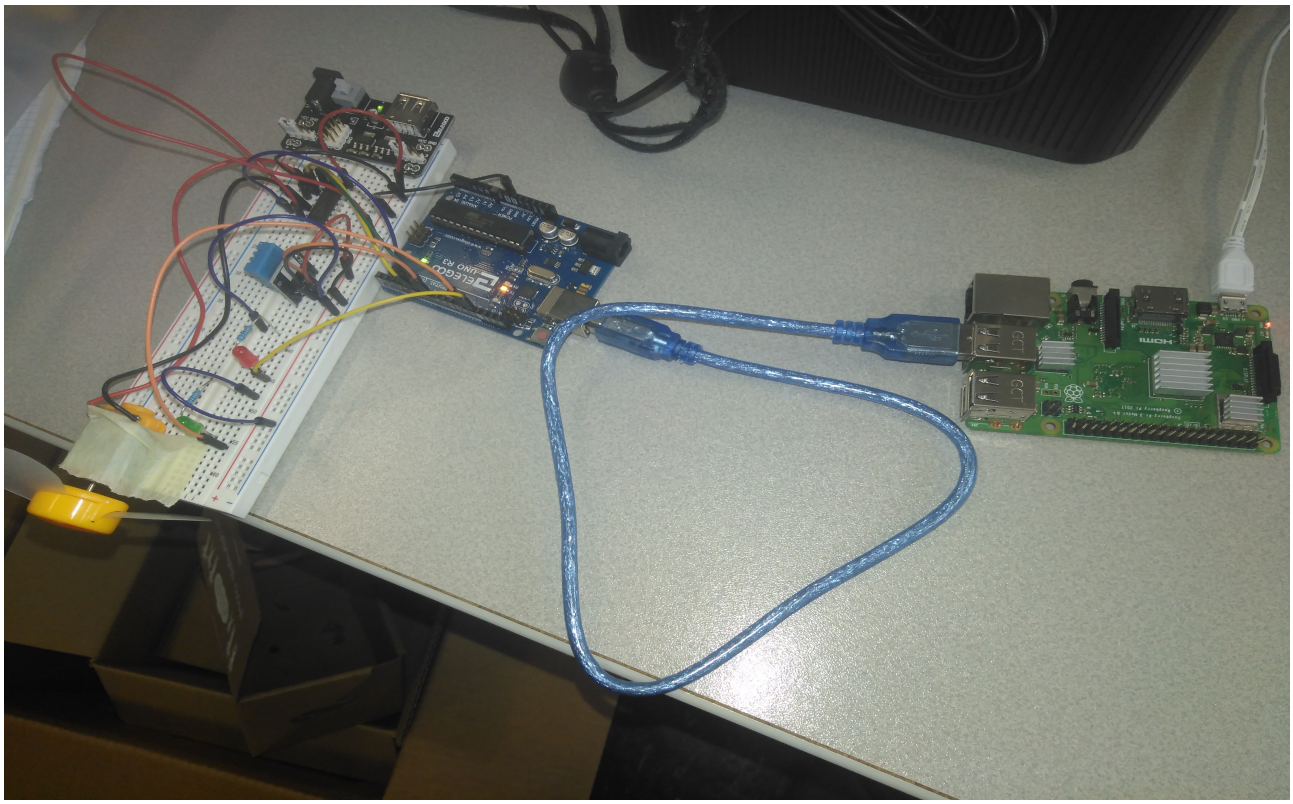
Bibliografia

https://randomnerdtutorials.com/esp8266-publishing-dht22-readings-to-sqlite-database/

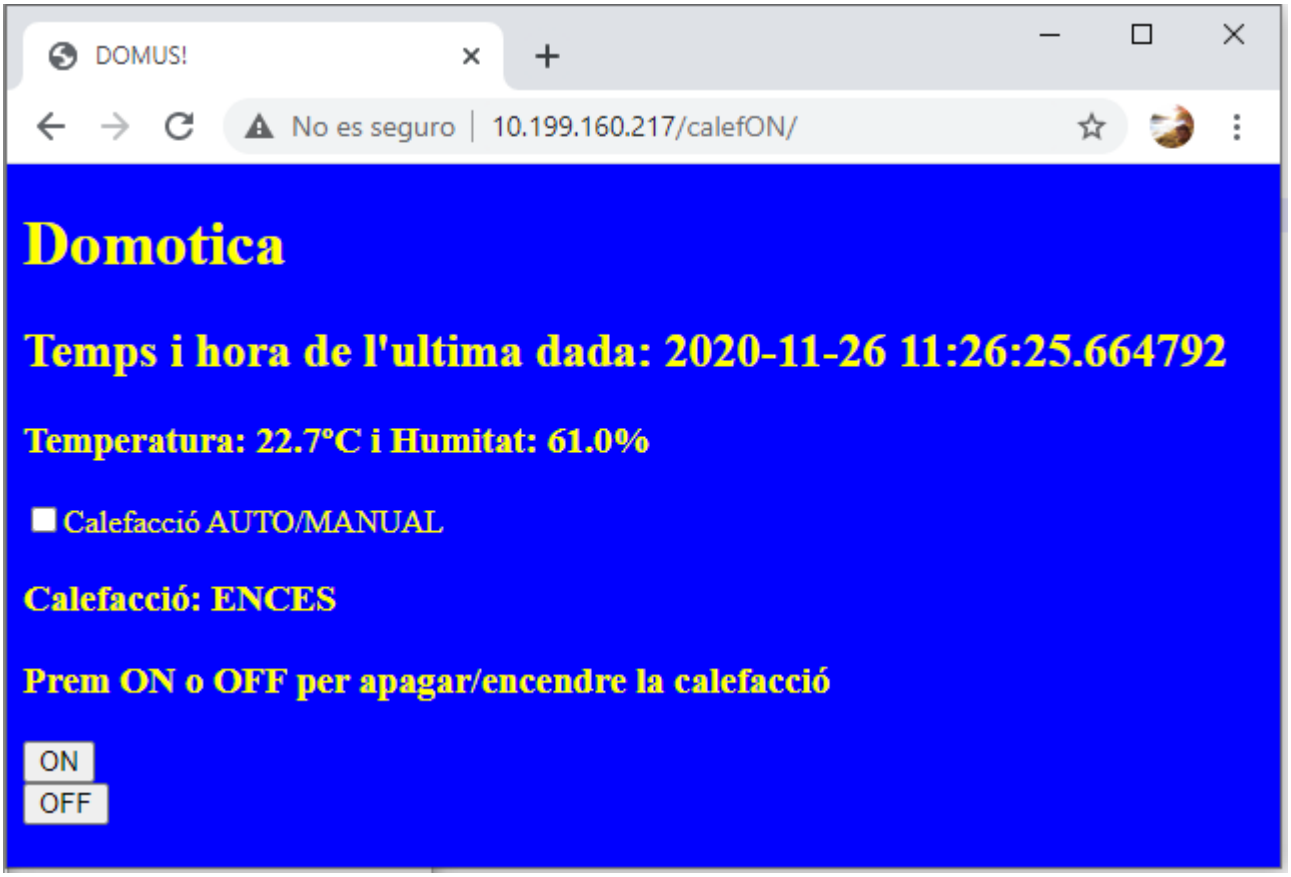
Resultats i conclusió

A data d'avui, s'ha pogut fer una prova de concepte, al haver desenvolupat un prototipus que permet llegir dades d'un Arduino des de qualsevol pàgina web de la xarxa (o exterior si es pot obrir ports) i commandar la calefacció. La pàgina web ofereix els últims valors de Temperatura i Humitat trobats a la Base de Dades del servidor i la disponibilitat d'encendre i apagar manualment la calefacció

Muntatge:



Execució de la pàgina Web:



A partir d'aquí, es pot anar millorant amb les següents funcionalitats:

a) Afegir una connexió automàtica a un calefactor, de forma que es pugui connectar en manual o automàtic: Si està en mode manual, serà necessària l'engegat o apagat des de la pàgina web (ja implementat). Si està en mode automàtic, l'engegat i apagat es farà en funció de la temperatura mesurada; i.e: Si la $T^a>20^a$ romandrà apagat. Si la $T^a<18^o$, s'encendrà
b) Instal·lar una càmera amb infrarojos a la Raspberry Pi per a que actui com a detector de seguretat. Si s'activa començarà a gravar durant un minut i s'enviarà un correu electrònic o un SMS a la persona interessada.
c) Instalar un detector de fugues d'aigua, de forma que si s'activa, s'envii un email o un SMS al telefon programat
d) Instalar un detector de nivell al disposit d'aigua de reg, per veure si es disposa d'aigua pel rec automatic. Si el nivell es troba per sota del minim, s'envia un email o un SMS.

ANNEX 1: Llistat Programa Arduino

```
//Projecte Final del Curs Plaques Robòtiques
// Joan Faig Martí
// Novembre 2020

#include <DHT.h>
#define DHTPIN 7 // Si el pin digital per les dades del sensor DHT11 es el 7.
#define DHTTYPE DHT11 // Especifiquem que el sensor es el DHT11
DHT dht(DHTPIN, DHTTYPE); // Inicialitzacio 1

#define CALEF_ON 12
#define ERROR_CALEF 13
#define ENABLE 6
#define DIRA 5
#define DIRB 4

unsigned long temps, timeS1=100;  //Sensor S1 cada 10s
unsigned long timeA1=120;         //Actuador A1 cada 12s
unsigned long anteriorS1=0,anteriorA1=0;

void setup() {
 Serial.begin(9600); // Inicialitzacio Comunicació Sèrie i Velocitat.
 dht.begin(); // Inicialitzacio
 pinMode(CALEF_ON, OUTPUT);
 pinMode(ERROR_CALEF, OUTPUT);
 pinMode(ENABLE,OUTPUT);
 pinMode(DIRA,OUTPUT);
 pinMode(DIRB,OUTPUT);
 digitalWrite(ERROR_CALEF, LOW);
 digitalWrite(CALEF_ON, LOW);
}

void loop() {
```

```
  temps=millis();
  if((temps-anteriorS1) > timeS1){
    anteriorS1= temps;
    actuaS1();
  }
  if((temps-anteriorA1) > timeA1){
    anteriorA1= temps;
    actuaA1();
  }
}

void actuaS1(){
 float h = dht.readHumidity(); // lectura de la Humitat com a decimal a la variable h
 float t = dht.readTemperature(); // lectura de la Temperatura com a decimal a la variable t.
 // Comprova si alguna lectura falla, surt per provar de nou.
 if (isnan(h) || isnan(t)) {
 Serial.println("Errada en llegir el sensor DHT11 !"); // imprimeix i nova linia
 return;
 }
 Serial.print("Temperature: "); //imprimeix paraula i mateixa linia
 Serial.print(t); // imprimeix valor de temperatura
 Serial.print("\n"); // Canvi de linia
 Serial.print("Humidity: "); // imprimeix paraula i mateixa linia
 Serial.print(h); // imprimeix valor de humitat
 Serial.print("\n"); // canvi de linia
}

void actuaA1(){
//   while (!Serial.available()) {} // wait for data to arrive
  // serial read section
  //while (Serial.available()){
  //  delay(30);  //delay to allow buffer to fill
    if (Serial.available() >0){
      int c = Serial.read()-'0';  //gets one byte from serial buffer
      //readString += c; //makes the string readString
      if (c==0){
```

```
      digitalWrite(CALEF_ON, LOW);
      digitalWrite(ERROR_CALEF, LOW);
      digitalWrite(ENABLE, LOW);
      Serial.write("C igual a 0");
    }else if(c==1){
      digitalWrite(CALEF_ON, HIGH);
      digitalWrite(ERROR_CALEF, LOW);
      digitalWrite(ENABLE,HIGH);
      digitalWrite(DIRA,LOW);
      digitalWrite(DIRB,HIGH);
      Serial.write("C igual a 1");
    } else {digitalWrite(ERROR_CALEF, HIGH); digitalWrite(CALEF_ON, LOW); Serial.write("C
indeterminat");}
   }
  //}
}
```

ANNEX 2: Llistat programa Raspberry

```
# Variant 3: Introducció de  programació SQLite
# segons  https://medium.com/analytics-vidhya/sqlite-database-crud-operations-using-python-3774929eb799
# 18-11-2020
# Variant 4: Botons de calefacció Manual/automatic i ON /OFF
# segons
# 23-11-2020

import serial
import datetime
import logging
import sys
import io
import re
import sqlite3


arduino= serial.Serial(
    port='/dev/ttyACM0',
    baudrate=9600,
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS,
    timeout=1)

conn=sqlite3.connect('dadesDomotiques.db')
cursor=conn.cursor()
cursor.execute("DROP TABLE IF EXISTS DHT_11")
query="""CREATE TABLE DHT_11(
    ID INT PRIMARY KEY NOT NULL,
    TEMP FLOAT,
    HUM FLOAT,
    TEMPS TIME
     )"""
cursor.execute(query)

cursor.execute("DROP TABLE IF EXISTS CALEF")
query="""CREATE TABLE CALEF(
    ID INT PRIMARY KEY NOT NULL,
    ENCES INT,
    AUTOMATIC INT,
    TEMPS TIME
     )"""
cursor.execute(query)
query="INSERT INTO CALEF( ID, ENCES, AUTOMATIC, TEMPS) " "VALUES(:ID, :ENCES,
:AUTOMATIC, :TEMPS)"
params={'ID':'1', 'ENCES':'0', 'AUTOMATIC':'0', 'TEMPS':datetime.datetime.now()}
cursor.execute(query, params)

regex =re.compile("[0-9]{2}\.[0-9]{2}")
valueTemp=0.
valueHum=0.

print("Inici lectura")
contador=1
while (contador <20000):
    try:
        data=arduino.readline()
```

```
        if b"Temperature: " in data:
            temps = datetime.datetime.now()
            #print ("data:"+str(data))
            #print (regex.findall(str(data))[0]+"\n")
            valueTemp=float(regex.findall(str(data))[0])
            #fitxer.write(str(temps)+';')
            #fitxer.write(regex.findall(str(data))[0]+"\n")
        if b"Humidity: " in data:
            temps= datetime.datetime.now()
            #print (data)
            valueHum=float(regex.findall(str(data))[0])
            query="INSERT INTO DHT_11( ID, TEMP, HUM, TEMPS) " "VALUES(:ID, :TEMP, :HUM, :TEMPS)"
            params={'ID':contador, 'TEMP':valueTemp, 'HUM':valueHum, 'TEMPS':temps}
            print ("Contador:"+str(contador))
            conn.execute(query, params)
            conn.commit()
            contador=contador+1

            queryCalef="""select ENCES from CALEF order by rowid desc limit 1"""
            cursor4= conn.execute(queryCalef)
            ences=cursor4.fetchall()[0][0]
            print ("Calefacció està {}".format(ences))
            if ences == 0:
                arduino.write(b"0")
            if ences == 1:
                arduino.write(b"1")

            print(arduino.readline())
    except KeyboardInterrupt:
        break

conn.close()
arduino.close()
sys.exit()
```

ANNEX 3: Llistat programa Raspberry Pi: servidorDomotic_[x].py

```
# Versio 2
# Incloure 3 botons per activar calefacció
# 25-11-2020
# Versio 3
# Posar les accions de la funcio hello a les dues fucnions encendreCalef i apagarCalef
# Reduir les connexions a BBDD fent inser i select a la matiexa connexió
from flask import Flask, render_template
import datetime
import sqlite3

temperatura=21.70
humitat=56.00

app = Flask(__name__)
@app.route("/")
def hello():
   conn=sqlite3.connect('../dadesDomotiques.db')
   cursor=conn.cursor()

   queryTemp="""select TEMP from DHT_11 order by rowid desc limit 1"""
   queryHum="""select HUM from DHT_11 order by rowid desc limit 1"""
   queryTemps="""select TEMPS from DHT_11 order by rowid desc limit 1"""
   queryCalef="""select ENCES from CALEF order by rowid desc limit 1"""

   cursor = conn.execute(queryTemp)
   temperatura=cursor.fetchall()[0][0]
   cursor2 = conn.execute(queryHum)
   humitat=cursor2.fetchall()[0][0]
   cursor3= conn.execute(queryTemps)
   temps=cursor3.fetchall()[0][0]
   cursor4= conn.execute(queryCalef)
   ences=cursor4.fetchall()[0][0]
   if ences==0:
      estatCalef="APAGAT"
   else:
      estatCalef="ENCES"
   conn.close()

   now = datetime.datetime.now()
   templateData = {
     'title' : 'DOMUS!',
     'time': temps,
     'temper': temperatura,
     'hum': humitat,
     'encenCalef': estatCalef
     }
   return render_template('index.html', **templateData)

#@app.route("/", methods=['POST'])
@app.route("/calefON/", methods=['POST'])
def encendreCalef():
   conn=sqlite3.connect('../dadesDomotiques.db')
   cursor=conn.cursor()
```

```
    queryTemp="""select TEMP from DHT_11 order by rowid desc limit 1"""
    queryHum="""select HUM from DHT_11 order by rowid desc limit 1"""
    queryTemps="""select TEMPS from DHT_11 order by rowid desc limit 1"""
    queryCalefID="""select ID from CALEF order by rowid desc limit 1"""
    queryCalefEnces="""select ENCES from CALEF order by rowid desc limit 1"""
    cursor = conn.execute(queryTemp)
    temperatura=cursor.fetchall()[0][0]
    cursor2 = conn.execute(queryHum)
    humitat=cursor2.fetchall()[0][0]
    cursor3= conn.execute(queryTemps)
    temps=cursor3.fetchall()[0][0]
    cursor4=conn.execute(queryCalefID)
    contador=cursor4.fetchall()[0][0]
    cursor5=conn.execute(queryCalefEnces)
    encenCalef=cursor5.fetchall()[0][0]

    now = datetime.datetime.now()
    print("Encen")
    print("La variable {} es {}".format(encenCalef,type(encenCalef)))
    if encenCalef==0:
        encenCalef=1
        estatCalef="ENCES"
        automCalef=0
        now = datetime.datetime.now()
        query="INSERT INTO CALEF( ID, ENCES, AUTOMATIC, TEMPS) " "VALUES(:ID, :ENCES,
:AUTOMATIC, :TEMPS)"
        params={'ID':contador+1, 'ENCES':encenCalef, 'AUTOMATIC':automCalef, 'TEMPS':now}
        print ("Contador:"+str(contador))
        #print("Temps:"+str(temps)+"ENCES:"+str(encenCalef)+"Automatic:"+str(automCalef))
        conn.execute(query, params)
        conn.commit()
    conn.close()

    templateData = {
      'title' : 'DOMUS!',
      'time': temps,
      'temper': temperatura,
      'hum': humitat,
      'encenCalef': estatCalef
      }
    forward_message = "Encenent Calefaccio..."
    #hello()
    return render_template('index.html', **templateData);


#@app.route("/", methods=['POST'])
@app.route("/calefOFF/", methods=['POST'])
def apagarCalef():
    conn=sqlite3.connect('../dadesDomotiques.db')
    cursor=conn.cursor()

    queryTemp="""select TEMP from DHT_11 order by rowid desc limit 1"""
    queryHum="""select HUM from DHT_11 order by rowid desc limit 1"""
    queryTemps="""select TEMPS from DHT_11 order by rowid desc limit 1"""
    queryCalefID="""select ID from CALEF order by rowid desc limit 1"""
    queryCalefEnces="""select ENCES from CALEF order by rowid desc limit 1"""
    cursor = conn.execute(queryTemp)
    temperatura=cursor.fetchall()[0][0]
    cursor2 = conn.execute(queryHum)
    humitat=cursor2.fetchall()[0][0]
    cursor3= conn.execute(queryTemps)
    temps=cursor3.fetchall()[0][0]
```

```
    cursor4=conn.execute(queryCalefID)
    contador=cursor4.fetchall()[0][0]
    cursor5=conn.execute(queryCalefEnces)
    encenCalef=cursor5.fetchall()[0][0]

    now = datetime.datetime.now()
    print("Apaga")
    print("La variable {} es {}".format(encenCalef,type(encenCalef)))
    if encenCalef==1:
        encenCalef=0
        estatCalef="APAGAT"
        automCalef=0
        now = datetime.datetime.now()
        query="INSERT INTO CALEF( ID, ENCES, AUTOMATIC, TEMPS) " "VALUES(:ID, :ENCES,
:AUTOMATIC, :TEMPS)"
        params={'ID':contador+1, 'ENCES':encenCalef, 'AUTOMATIC':automCalef, 'TEMPS':now}
        print ("Contador:"+str(contador))
       # print("Temps:"+str(temps)+"ENCES:"+str(encenCalef)+"Automatic:"+str(automCalef))
        conn.execute(query, params)
        conn.commit()
    conn.close()

    templateData = {
      'title' : 'DOMUS!',
      'time': temps,
      'temper': temperatura,
      'hum': humitat,
      'encenCalef': estatCalef
      }
    forward_message = "Apagant Calefaccio..."
    #hello()
    return render_template('index.html', **templateData);

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80, debug=True)
```

ANNEX 4: Llistat arxiu : index.html

```
<!DOCTYPE html>
	<head>
		<title>{{ title }}</title>
		<link rel="stylesheet" href="../static/style.css/">
	</head>
	<body>
		<h1> Domotica </h1>
		<h2> Temps i hora de l'ultima dada: {{ time }}</h2>
		<h3> Temperatura: {{ temper }}ºC i Humitat: {{ hum }}%</h3>

		<input ID="ckeckboxAUTO" name="calef" type="checkbox" />Calefacció AUTO/MANUAL
		<h3> Calefacció: {{ encenCalef }}</h3>
		<h3> Prem ON o OFF per apagar/encendre la calefacció </h3>
		<form action="/calefON/" method="post">
			<button name="calefON" type="submit">ON</button>
		</form>
		<form action="/calefOFF/" method="post">
			<button name="calefOFF" type="submit">OFF</button>
		</form>
	</body>
</html>
```