TÉCNICO LISBOA

ATHENS NETWORK
ADVANCED TECHNOLOGY
HIGHER EDUCATION NETWORK

The doorman of the Future

A new approach to managing studying rooms

UPM115 - Physical Computing based on Open Software and Hardware Platforms [EN]

Autores:

António Vidais (IST - Lisboa) antonio.vidais@tecnico.ulisboa.pt
Tomaz Silva (IST - Lisboa) tomaz.goncalves-silva@tecnico.ulisboa.pt

Athens – Winter of 2022

1 Introduction

The need for this system comes from the fact that every day, students trying to study on their college have to scour their entire college for a room where they can study. The *Future Doorman* solves this by sharing data from open rooms to everyone who needs it.

It is composed of 2 systems, a client that is attached to the door of each room and a server connected to a computer that can share the information either online or on a screen.

2 Choice of Hardware

2.1 Client

After some tries, we decided that the best platform available was the ESP32, because of the integrated Wifi module, without compromising on the number of GPIO pins and memory.

However, choosing an ESP-WROOM-32 meant that 5v power wouldn't be available. That meant that we couldn't use ultrasound sensors. While searching for a 3.3v compatible distance sensor. We settled on LIDAR sensor because it provided the precision that the application requires and works on both 3.3v and 5v power.

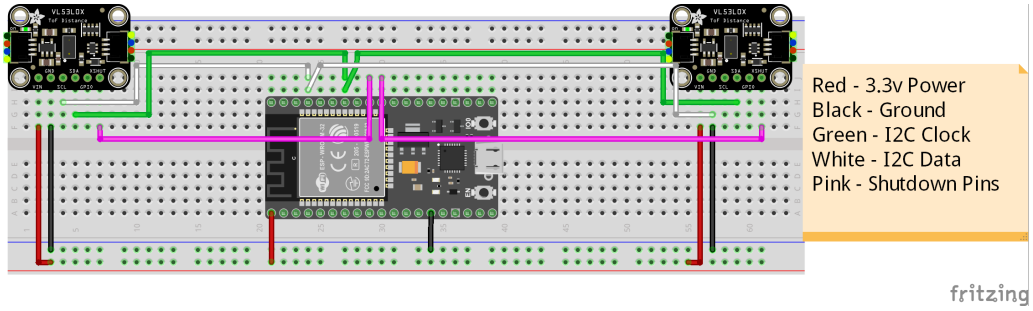

Figura 1: Client Schematic

2.2 Server

For the server we also wanted to use an ESP-WROOM-32 because of the compact size and, once again, the superior wifi module. The server doesn't need any additional modules as it´s only function is to connect the server computer to the clients.

3 Hardware Limitations

As we tested the hardware, we realized that there was a limitation in the refresh rate of the LIDAR Sensors, as they would sometimes miss count if people are fast enough. This situation only happened rarely so it isn't too much of a problem.

One Limitation that we couldn't solve is that LIDAR sensors are dependent on colors, so we predict some unexpected behaviour if the background isn't mono color.

We also noticed that the server and client would often have different refresh rates and getting them to match would be hard, so we had to separate the sensor reading from the data

sending. We settled on using an HTTP server between the 2 modules, where the server requests data from the client and the client answers by sending both room and occupation data. This way both modules are independent and can work at different refresh rates.

4 Displaying data

To get the data from clients, the server sends multiple HTTP requests and then analyses the responses to get the data for each room. Responses give the server information on both room and occupation.

The data is displayed in a window as follows.

On the left side, the classrooms' name surrounded by a color representing it's percentage of occupation.

- Green: 0 to 30 percent of room is free;
- Yellow: 30 to 75 percent of the room is occupied;
- Red: More than 75 percent of the room is occupied;

  (Note that it is possible to have more than 100 percent occupation and in this case it will also display red);

On the right of the classrooms' name you can see how many people are inside the room and it's maximum capacity.

On the right side of the window, 2 buttons that let the user sort the rooms in 2 ways that make it more convenient. One is alphabetically accordingly to the room's name, and the other sorts based on the percentage of occupation of each room (from low to high).

In the case, where there are too many rooms and they don't all fit in the windows' resolution, a sliding animation begins, showing all the hidden rooms. Once it reaches the end, resets to the first room.

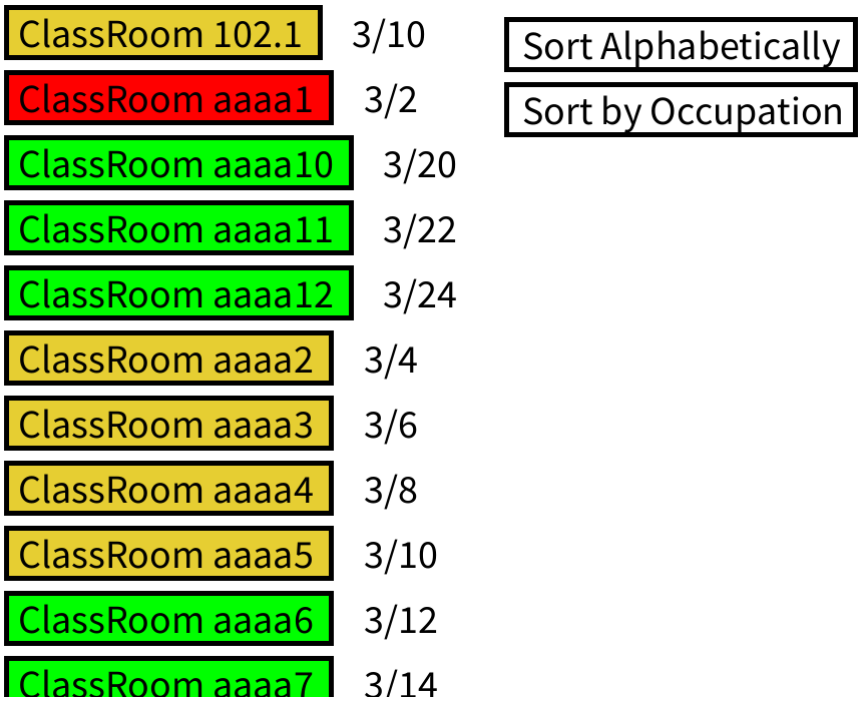

Figura 2: Display schematic

5 Source Code

Most of the Arduino source code is copied from examples or publicly available snippets. One snippet that must highlighted is where data from both sensors is taken and used to perceive if a person is either entering or leaving the room:

```
read_dual_sensors();
if( (measure1.RangeStatus!=4)&&(measure2.RangeStatus==4)){
  read_dual_sensors();
  while (!((measure1.RangeStatus==4)&&(measure2.RangeStatus==4))){
    read_dual_sensors();
  }
      people++;

} else if ((measure1.RangeStatus==4)&&(measure2.RangeStatus!=4)){
  read_dual_sensors();
 while (!((measure1.RangeStatus==4)&&(measure2.RangeStatus==4))){
    read_dual_sensors();
  }
  if (people>0)  people--;
  }
```

Figura 3: Code Snippet showing people count

The full source code can be found on GitHub

6 Self-evaluation

6.1 What went wrong

There were some things that we intended to do extra but, unfortunately, due to issues with both breadboards and Arduino's lost a lot of valuable time.

6.2 What went well

We were able to divide the work very efficiently with António being responsible for designing and assembling the hardware (both client and server and the interaction between them) and Tomaz making sure the data gets treated, and then displaying it in an efficient and visually attractive User Interface.

By making the correct hardware choices, we were able to have a system that runs fully on 3.3v power, which reduces power consumption substantially.

6.3 The next steps

One important aspect this system lacks is the ability to calibrate the sensors. By having an initial measure of the background the system wouldn't be as dependent on the LIDAR sensors accurate measuring.

This system has potential to grow if we wanted to. We can scale to it almost indefinitely because there isn't a necessity for high refresh rates. One server can have many clients without problems which makes integration on big universities simpler than other systems.

The object of this system is to help students, so our initial idea was to have occupation data shown on an app (or a website), where students could quick and easily find a room to study in.

7 Conclusion

During this project we were challenged many times, with problems that an engineer designing a similar system only gets if he test real hardware. This is an important lesson in a world where everything is virtual. Our initial *virtual* designs appeared to work well but, as we found out, that wasn't the case when it came to the real deal.