Internet of Things

Project

Pellier Bastien & Antoine Dombrecht
Pierre-Marie frerot & Maxime danguin

Contribution:

-Antoine DOMBRECHT: Search for labs and for the magic mirror, purchase of materials, installation of the magic mirror with his different steps, program implementation for the magic mirror, Camera and motion sensor test.

-Bastien PELLIER: Search and implementation for labs and for the magic mirror, purchase of materials, program creation for the magic mirror, filming, Camera and motion sensor test.

-Pierre-Marie FREROT: Search and implementation for labs and for the magic mirror, filming, editing video / post-production, writing of the report.

-Maxime DANGUIN: Search for labs and for the magic mirror.

Project – Lab 1 & 2 :

--- Getting started with the ESP32 ---

In order to getting us familiarized with aspects related to the ESP32, we initially need install the development environment and then design small programs that we can flash into the development board in order to run them. We will install the environment on our machine on Linux.

The first step to install the development environment is to check the dependencies and installing the esp toolchain (xtensa)

The first hard thing is the modification of the PATH that need to be permanent. (This is a modification of the. /bashrc instead of a simple command line

The next step is to install the framework from a git repository. Once again, we need to setup the framework path into. /bachrc

The right for the user to use the development board is the final step of implementing the development interface, by using the command line (sudo usermod –a –G dialout $antoine)

Finally, we need to prepare the bootloader (command: pip install esptool) and change the parameter on the menu configuration (command: make menuconfig) because of problem due to the compilator

Program Hello World:

The goal of this program is to test if we compile and if we can flash without any problem

As this program is just a looped menu (restart every 10 second) and we have already our first problem. The make config didn't find the necessary gcc to compile and we had to modify our menuconfig by changing where the flash seeks his data.

Program Blink (led):

This program was divided into many step:

- Understanding the blink.c from the get-started
- Modify the program to use an "external" led
- Modify how the led blink
- Recreate a program from scratch to see how it compile

On the program from "get-started" repository, we needed to understand two main point:

```
#define BLINK_GPIO CONFIG_BLINK_GPIO

void blink_task(void *pvParameter)
{
    /* Configure the IOMUX register for pad BLINK_GPIO (some pads are
       muxed to GPIO on reset already, but some default to other
       functions and need to be switched to GPIO. Consult the
       Technical Reference for a list of pads and their default
       functions.)
    */
    gpio_pad_select_gpio(BLINK_GPIO);
    /* Set the GPIO as a push/pull output */
    gpio_set_direction(BLINK_GPIO, GPIO_MODE_OUTPUT);
    while(1) {
        /* Blink off (output low) */
        gpio_set_level(BLINK_GPIO, 0);
        vTaskDelay(1000 / portTICK_PERIOD_MS);
        /* Blink on (output high) */
        gpio_set_level(BLINK_GPIO, 1);
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```

First, and the most important is the "#define" where you indicate every "pin".

In this case we use the built in led to check if the processor has no problem. The main default of the use of this test is that it takes time to understand how the code is processed.

It's a very simple code that make the built in led blink every second, the hardest thing to do was to understand how to make the definition.

For example, in the code we use, we need to" cast" 4 pin slots (2 to give the instruction and 2 to give the energy for the led)

In this part of our work, we had a lot of trial and error. For example, we tried to follow the instruction from {http://www.instructables.com/id/Blinking-an-LED-With-ESP32/} but the program send many error as it didn't understand the ledPin variable.

We also tried to just copy the code part of {https://techtutorialsx.com/2017/06/15/firebeetle-esp32-blinking-the-on-board-led/}. This also didn't work well as the setup function resulted as a error, we didn't found why.

Finally, we worked on already programed code that we modified to change the used led from the built-in to an external and configure a timer to change how the led blink.

With our first program, we're able to change the blinking rhythm of a led. We changed that by making the program being able to take charge of a second led.

Once on program is fully functionally, we tried to recreate the code outside, by creating a new repository

Our Program: Magic mirror

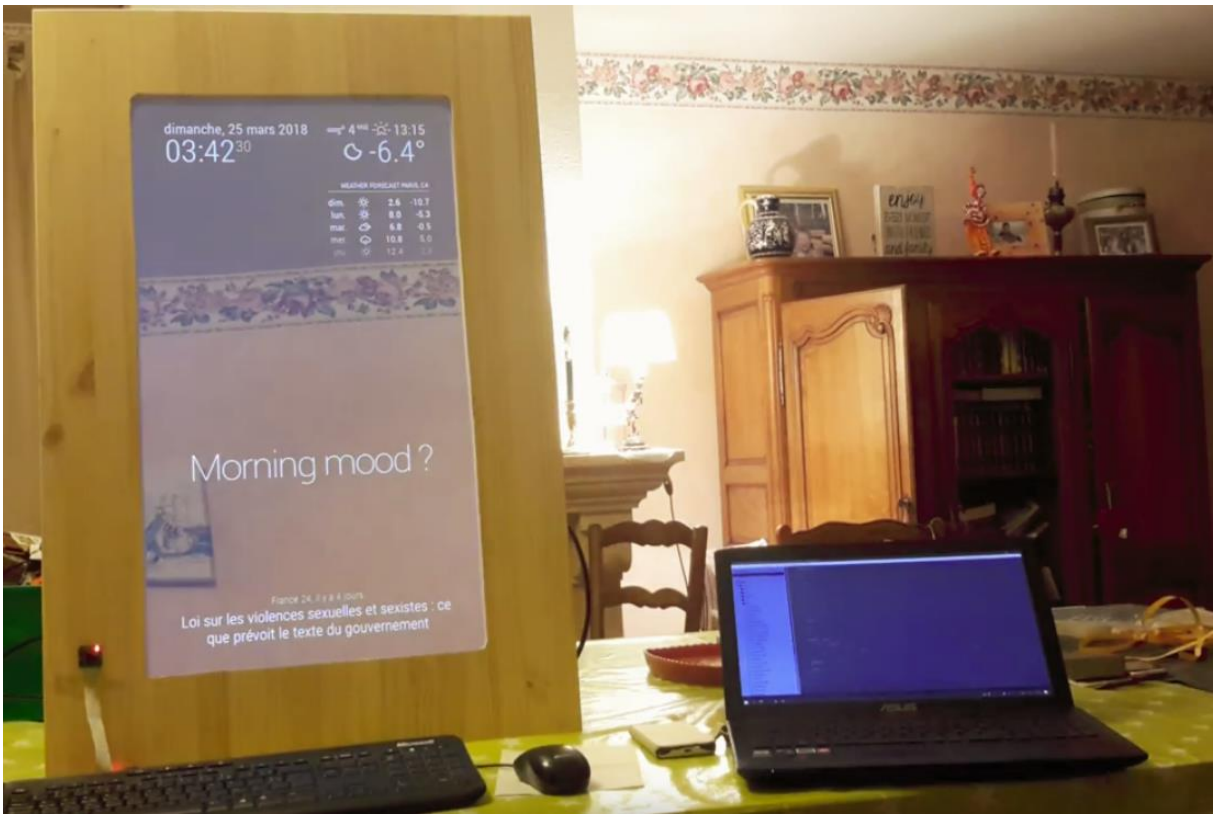
Our final goal is to create a "magic mirror" that we've called Sapio-mirror.

To create such a thing, we need some material:

- A raspberryPi (model b - 1Gb ram) (34,00 euros)
- A specific alimentation for Raspberry Pi (16,50 euros)
- A 32Gb SD card (17,90 euros)
- A monitor or a hdmi cable (old material)
- One-way mirror (179,34 euros)
- Wood formwork (75,36 euros)
- Some software (ex: an SD card formatter)
- Sensor (8.99 euros)
- Camera (12,37 euros)

Total: 357,63 euros

Here an image of the result:



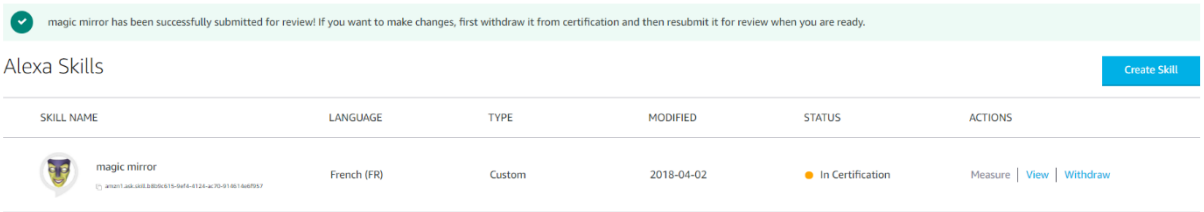
It exist many tutorial on the internet, the one we've followed is http://emmanuel-vergne.fr/reflexions-autour-dun-appareil-connecte/

The tutorial gives us a basic interface with only the time/date, temperature, event, the latest news and a little text.

It need some change, for example we will add a facial recognition and word recognition.

word recognition

The word recognition needs certification, as seen in the image below:



We also need to create a script that use Alexa (it's amazon version of Cortana or Siri)

First you need to install Alexa from https://alexa.amazon.com

Once it's certified, it's possible to say "magic mirror, turn on" or "magic mirror, turn off" to enable/desable the screen with the following script:
https://gist.github.com/benrules2/c6ab906d94e4988c92b1596d20b5f7a2#file-check_queue-py

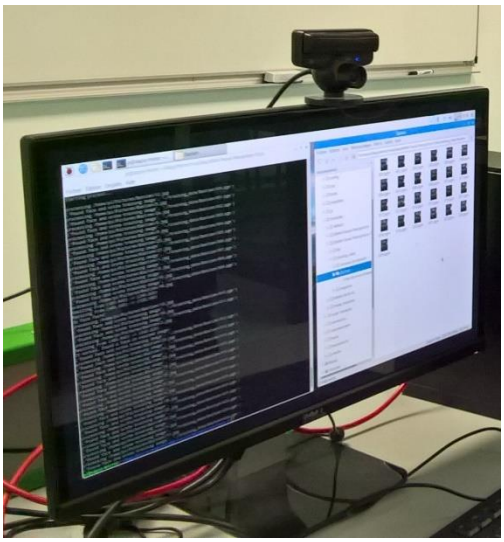
Facial recognition

Another module we've planned to install on the magic mirror device is a camera that will be able to do a facial recognition.

This little upgrade can be useful after as it can help the mirror "select" the events or information if he recognizes a peculiar face.

We will follow this tutorial:

https://www.magicmirrorcentral.com/pir-sensor-raspberry-pi-magic-mirror/



As you can see on the image, this software need to have a "library" of persons head to work, and one image doesn't give enough data.

Once the library is complete enough, the magic mirror can detect a specific head.

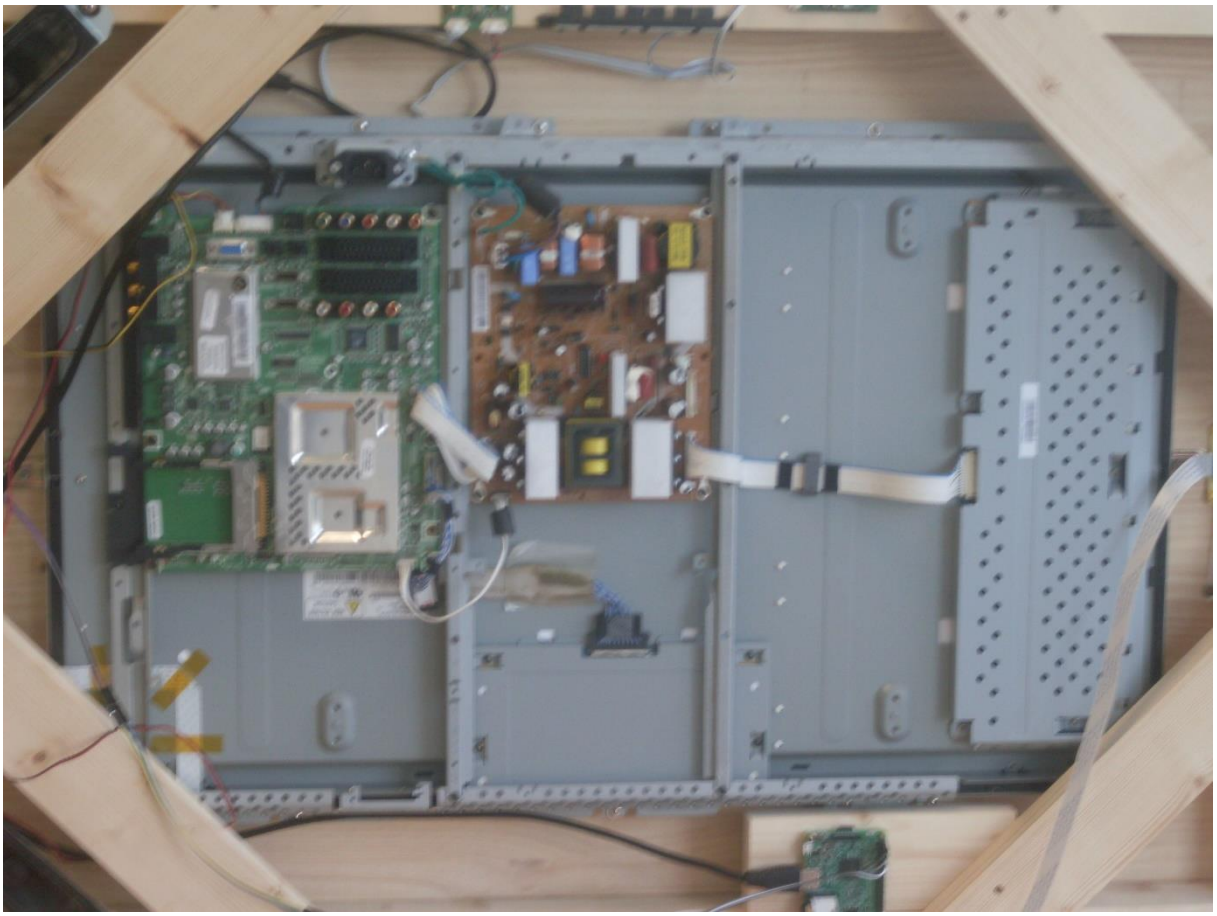
Movement sensor

Another thing we've added is a moving sensor, it permits to switch on and off the mirror after a certain cooldown period without anybody in front of the mirror.

Formwork

One last important thing the formwork.

Maybe we can consider it as something illogical to talk, it's also very important as we need to dismount a television and remount it in the wood formwork we have bought.



It needs caution as the whole material is very fragile (sensor/camera, raspberry and tv buttons). The total weight of the mirror approaches 16 kilograms.

Finally, we have created a hole on the top for the camera and one on the bottom for the sensor, on the side we have pasted all the buttons and the raspberry as we need an easy access to it.

Sources:

Magic mirror install

http://emmanuel-vergne.fr/installation-du-magic-mirror-partie-1/

https://forum.magicmirror.builders/topic/236/complete-setup-tutorial/2

move sensor install

https://www.magicmirrorcentral.com/pir-sensor-raspberry-pi-magic-mirror/

facial recognition installs

https://blog.fossasia.org/adding-face-recognition-based-authentication-to-susi-magicmirror-module/

Alexa installs

http://www.cyber-omelette.com/2017/01/alexa-run-script.html