IoT Project:
Smart House

François Bidet

February 9, 2018

1 Presentation

My project consists in the simulation of a smart house.

Some sensors are in the house and send data to a server, in or out the house. It can be a computer with a specific software or a specific device as for thermostats fixed on a wall. Those sensors can be temperature sensors, lumen sensors, gas sensors, et caetera.

Some actuators are in the house and receive commands from the server. They can be heaters, lights, electric shutters, windows, et caetera.

For the project, I have used:

- humiture sensor
- lumen sensor
- button
- dual color led (for heater and air-conditioner representation)
- RGB led

2 Behavior

The raspberry plays the role of gateway. It sends sensors data second to the server.

A computer plays the role of the server. It receives sensors data and it sends commands to the actuators via the raspberry every 5 seconds.

The delays of the sender and receiver are arbitrary fixed. For a real implementation, we need to configure those delays: we need a faster control of lumen than of the temperature.

2.1 Temperature management

The server can fixe a reference temperature. When it receives temperature data, it compares to the reference:

- if the temperature is lower than the reference, it sends command to turn on the heaters.
- if the temperature is higher than the reference, it sends command to turn on the air-conditioner (cooler).
- else it sends command to turn off both.

When the rapsberry receives command:

- if it is to turn on the heaters, it fixes the color of the dual-color led to red.
- if it is to turn on the air-conditioner, it fixes the color of the dual-color led to green.
- else it turns off the dual-color led.

An hysteresis is implemented in the server program but due to the temperature sensor precision (I got entire values like 20.0°C and never 20.3°C), this behavior does not appear.

Figure 1: Server interface to control the reference temperature

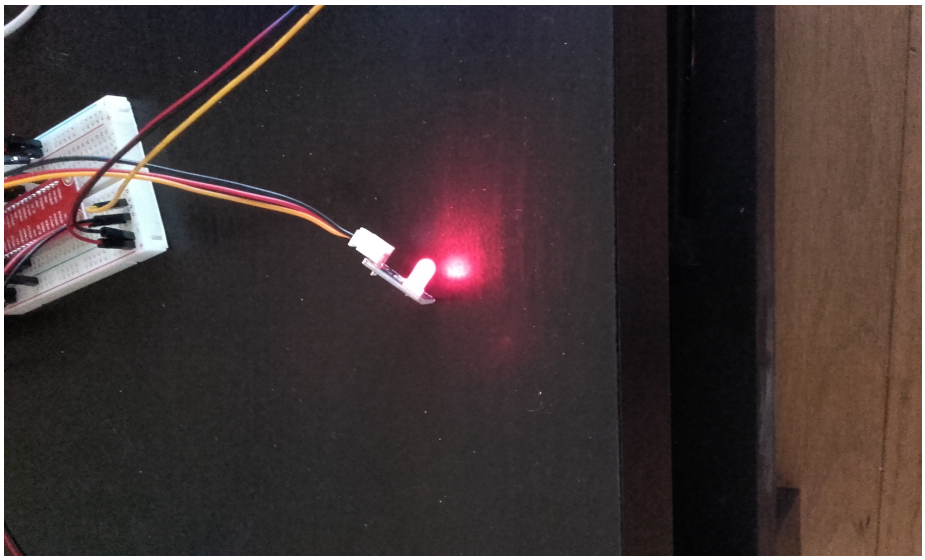

Figure 2: Temperature lower than the reference

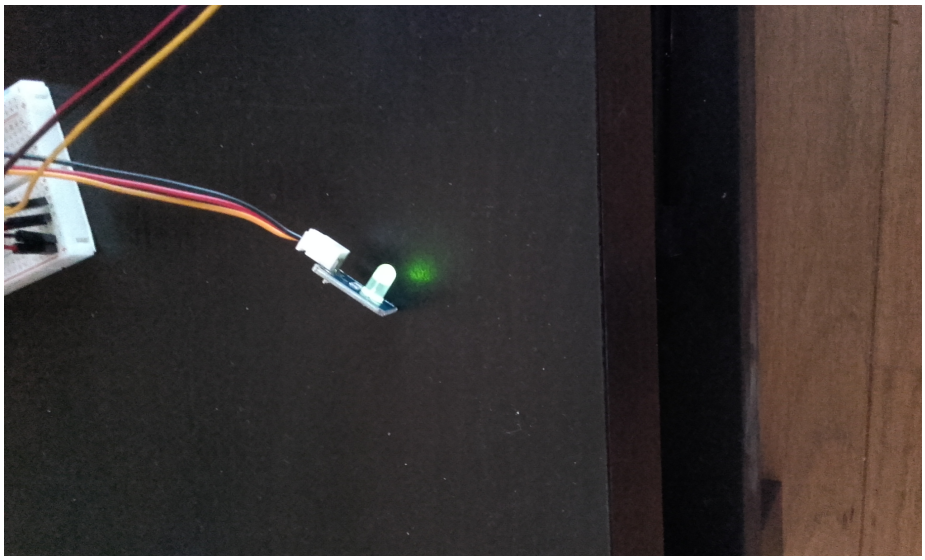

Figure 3: Temperature higher than the reference

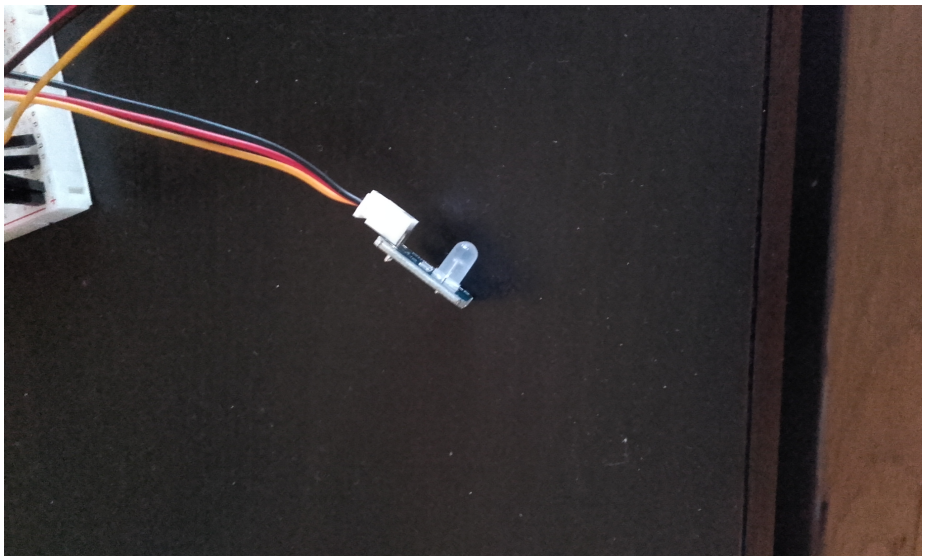

Figure 4: Temperature equal to the reference

2.2 Lumen management

A button in the house (on the raspberry) can turn on or turn off the lights.

If lights are turned on, the server fixes the light power according to the lumen sensor data. Max and min limits are hardcoded for the lumen sensor data values (variable resistance of the sensor).

- If the lumen sensor data value is higher than the max limit, the lights are turned off.
- If the lumen sensor data value is lower than the min limit, the lights are turn on at maximum power.
- If the lumen sensor data value is between the limits, the lights are turn on at power linearly depending of the value.

The raspberry which control the lights create a smooth transition between the previous light power and the new ordered by the server.

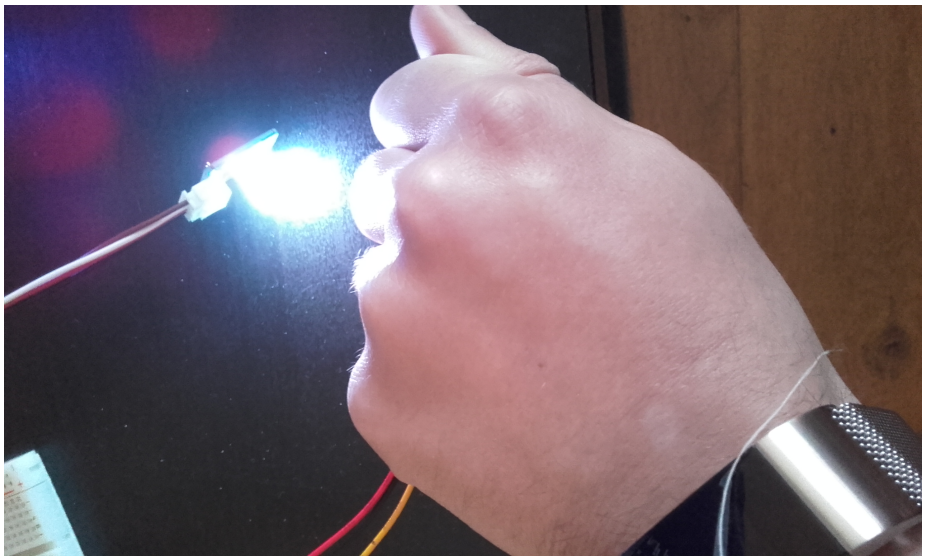

Figure 5: Little brightness (lumen sensor in the closed hand)

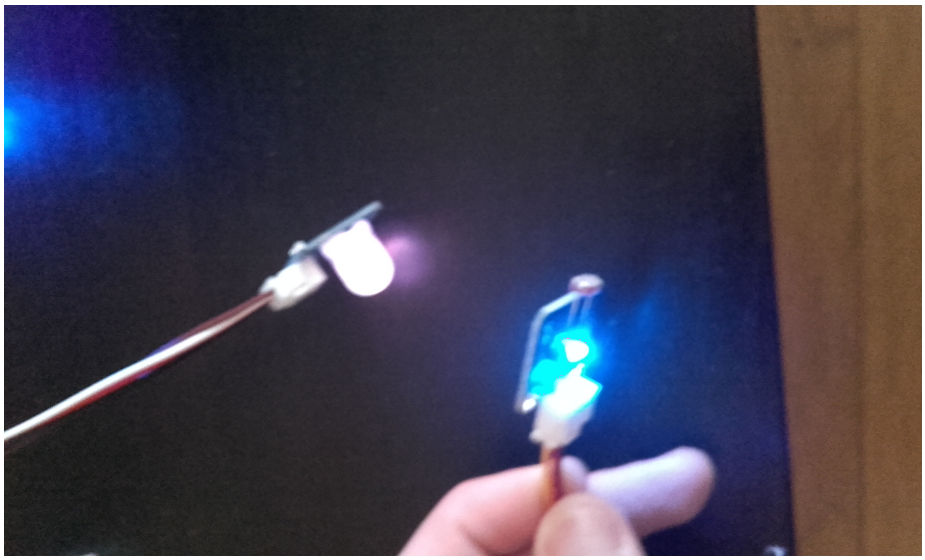

Figure 6: A lot of brightness (lumen sensor facing to the window)

3 Architecture

3.1 Hardware

All sensors and actuators are fixed on the raspberry board.

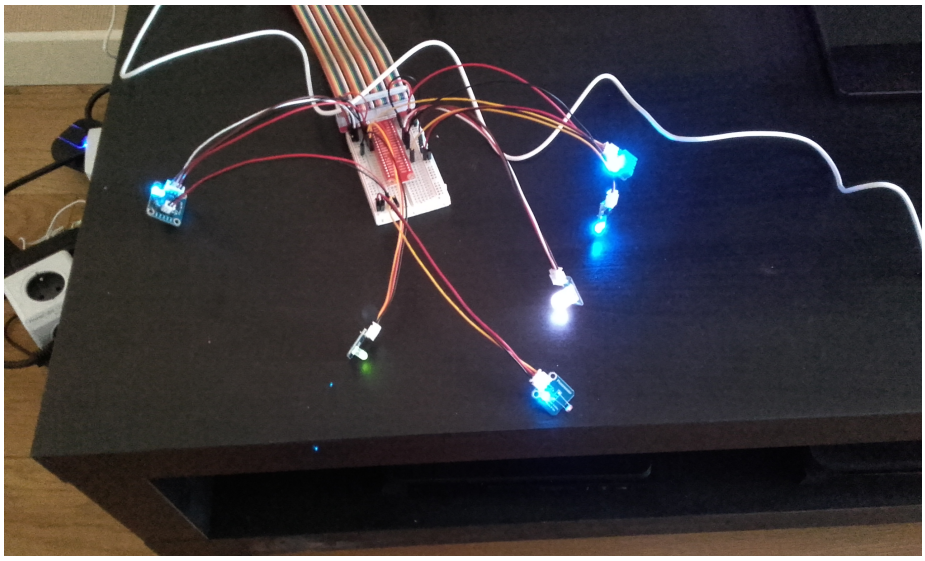

Figure 7: The entire system

3.1.1 Assembly

PCF8591 and Photoresistor

	PCF8591	(remove the jumper)
VCC	->	3.3V
GND	->	GND
SCL	->	SCL
SDA	->	SDA

	Photoresistor	
VCC	->	3.3V
GND	->	GND
SIG	->	[PCF8591]AIN0

Humiture Sensor

GND	->	GND
VCC	->	5V
SIG	->	GPIO17

Dual-Color LED

GND	->	GND
G	->	GPIO27
R	->	GPIO22

RGB LED

VCC	->	5V
R	->	GPIO23
G	->	GPIO24
B	->	GPIO25

Button

V	->	5V
G	->	GND
S	->	GPIO12

3.2 Software

On the raspberry, a main thread is running and can display sensors information when running in a debug mode (Figure 8). A thread is running for getting regularly sensors values (one subthread for each sensor) and store them in static variables. A thread is running to manage the connection: one subthread to send sensors data and one other subthread to receive server commands.

On the server, one thread is running to receive raspberry sensors data, one other thread is running to send regularly commands and a third thread is running to get the reference temperature from the user.

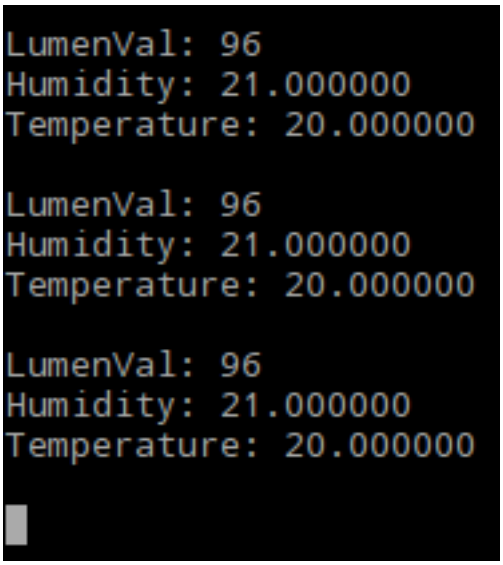

Figure 8: Debug mode running on the raspberry

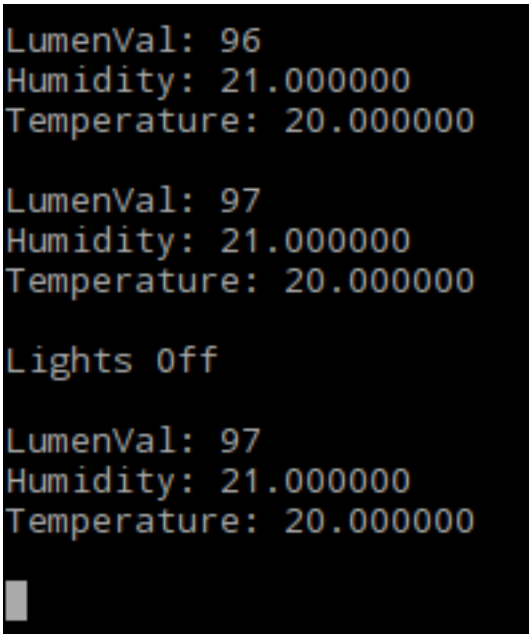

Figure 9: Debug mode running on the raspberry with lights off notification (button pressed)

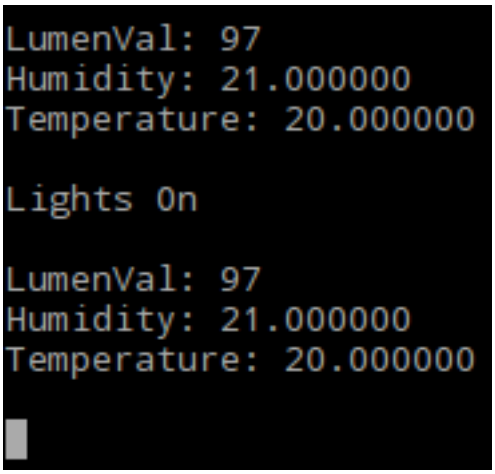

Figure 10: Debug mode running on the raspberry with lights on notification (button pressed)

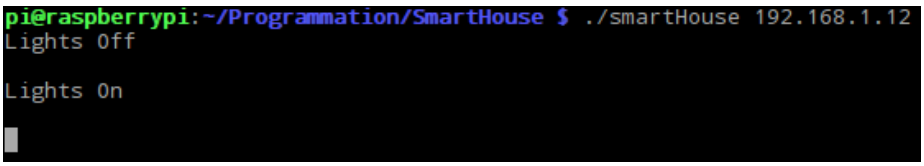

Figure 11: Normal mode running on the raspberry with lights notifications