Maxime Carrillo

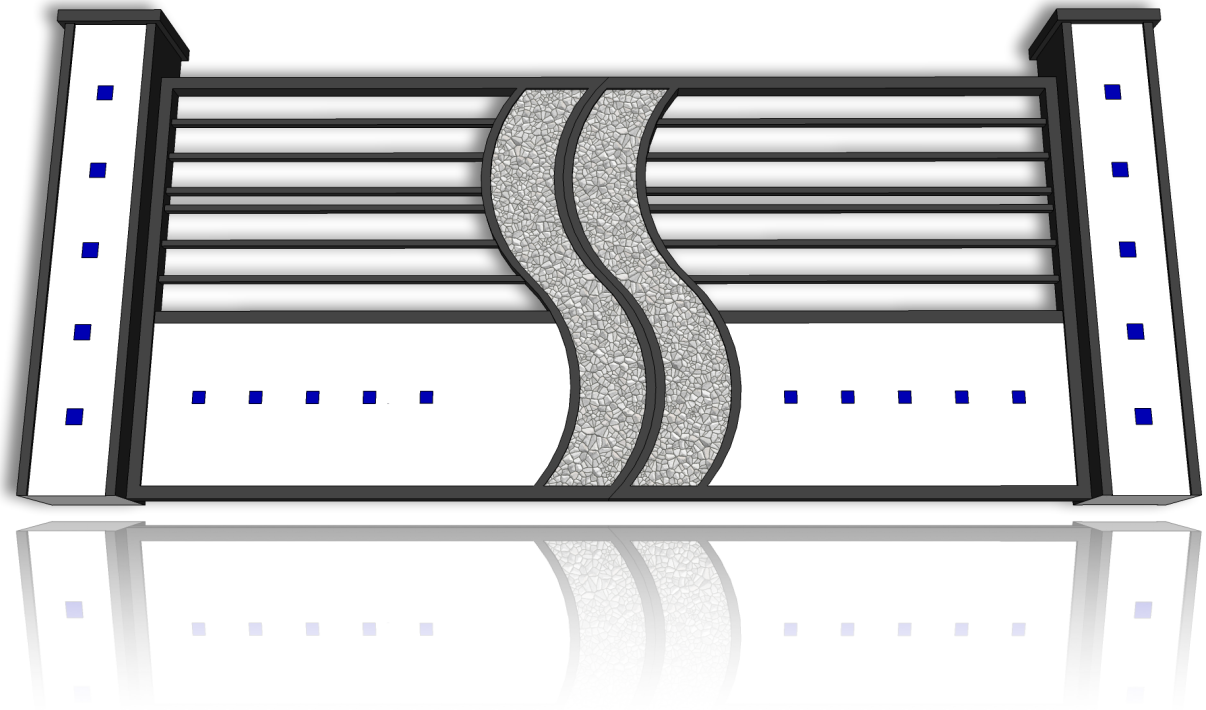
Baccalaureate : Science and Technology of Industry and Sustainable Development

Specialty : Digital Information Systems

Project :

iGate

The ultra-automatic gate



Discovery Project

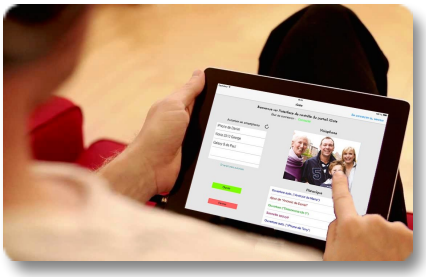
Introduction

iGate is a full gate system at home. It lets you to enter and leave your home without caring a remote control, regardless going to see who is ringing at the entrance, and even knowing when and who to open the gate.

When you arrive home, by approaching by car to the entrance of your home, the gate detects your smartphone and opens automatically, only for you.
Once installed on your couch, watch what is happening around your home with outdoor surveillance, or check who open the gate during the day.

The gate detect your smartphone…

…then opens.



Control everything from your sofa.

Preview

The system is distributed across multiple platforms: a tablet application, smartphone application, a website and an opening manager, located near the gate.

Tablet application

Smartphone application

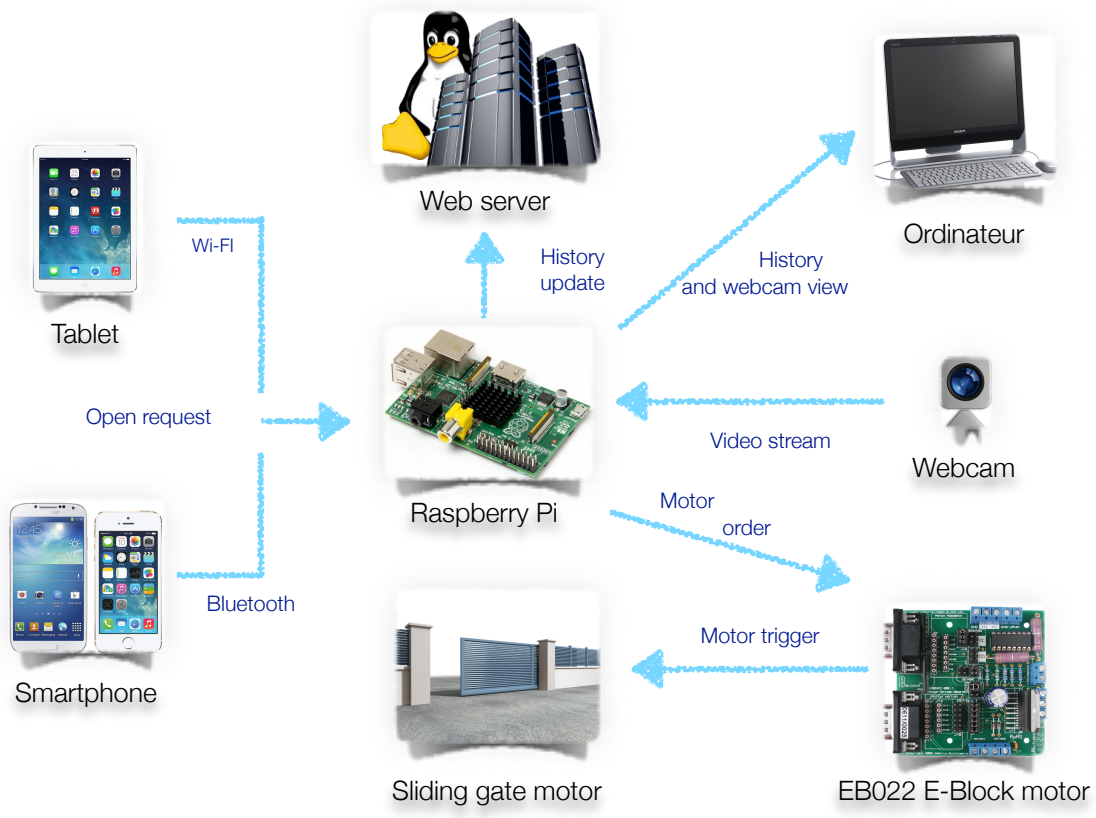
Opening manager

Web site

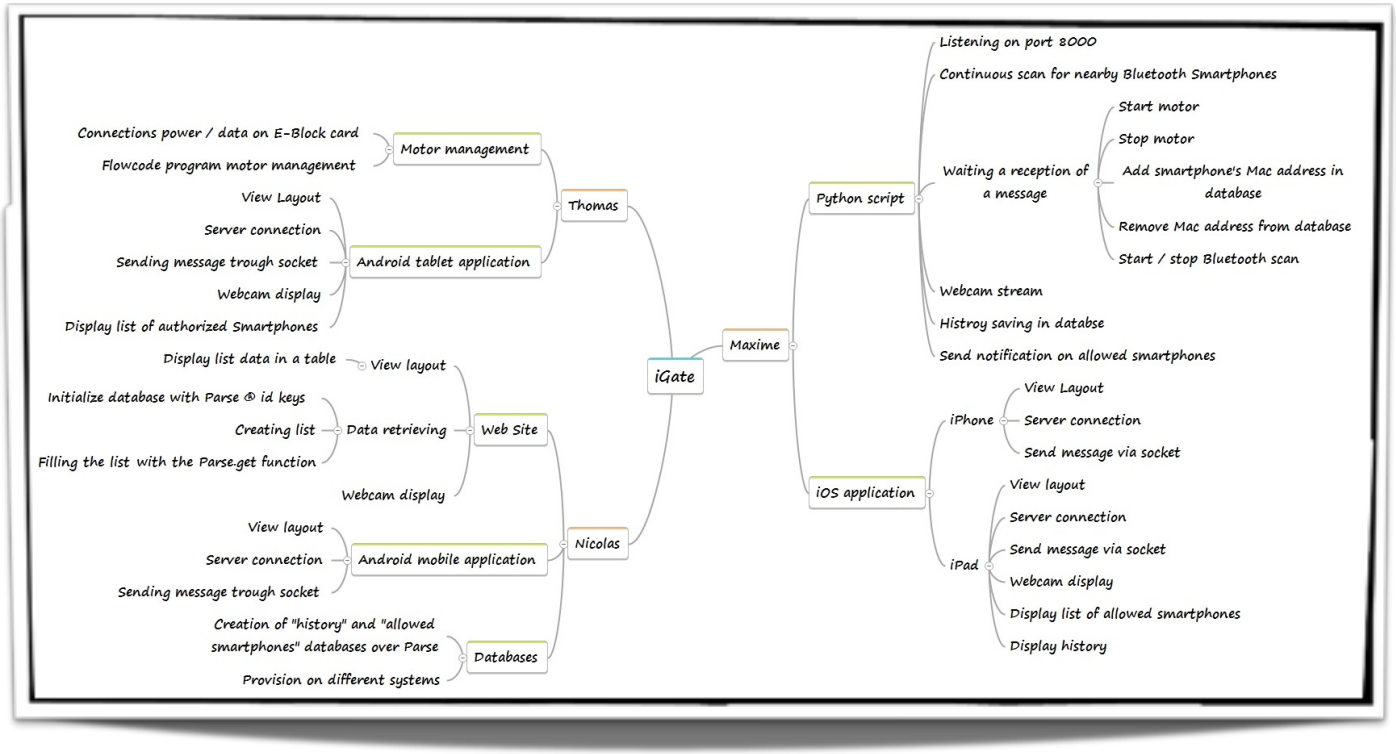
Technical documentation

Schema and MindView

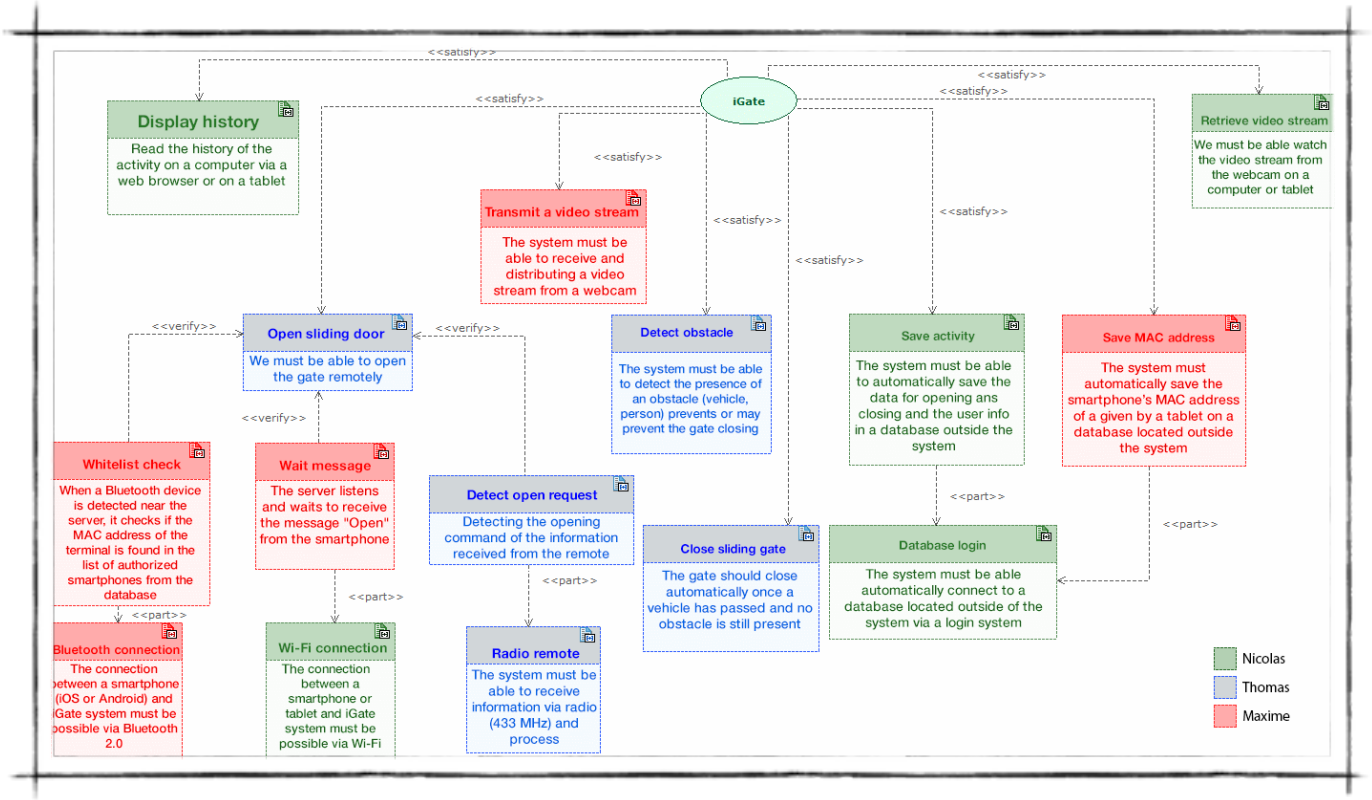
Here is a diagram representing the overall system :



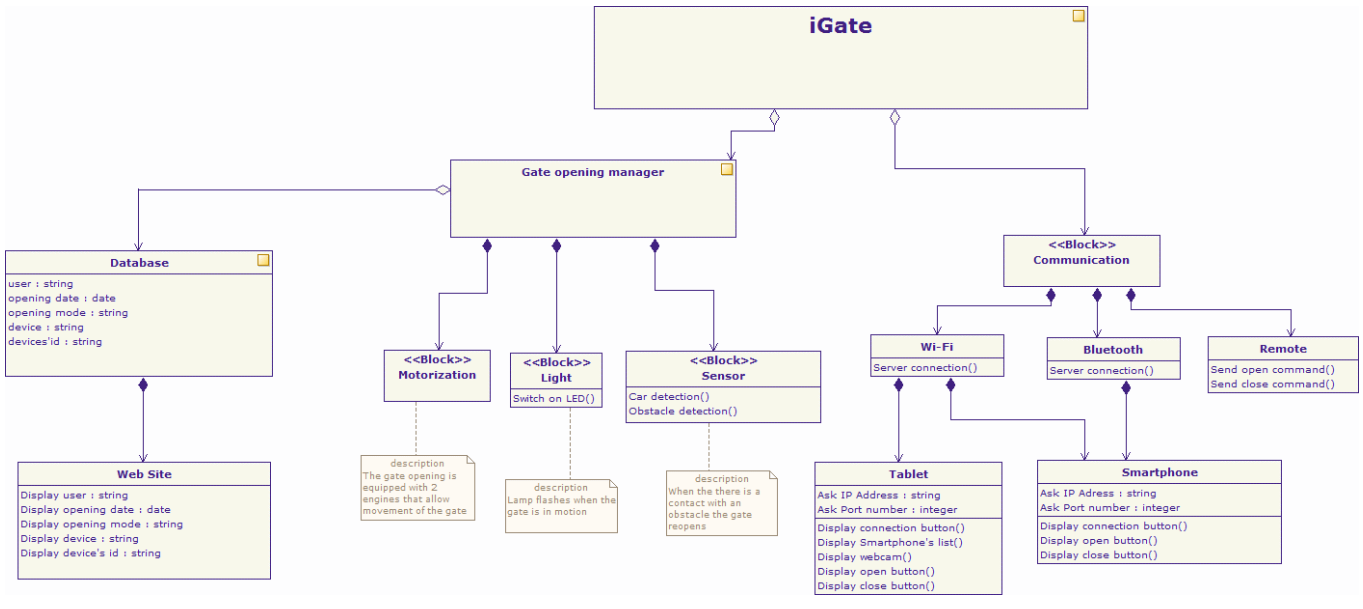
Here is the mind map, providing information on the various tasks. For my part, I made the Python script (program running on the Raspberry Pi), and iOS tablet and smartphone applications, ie iPad and iPhone.



Requirements diagram



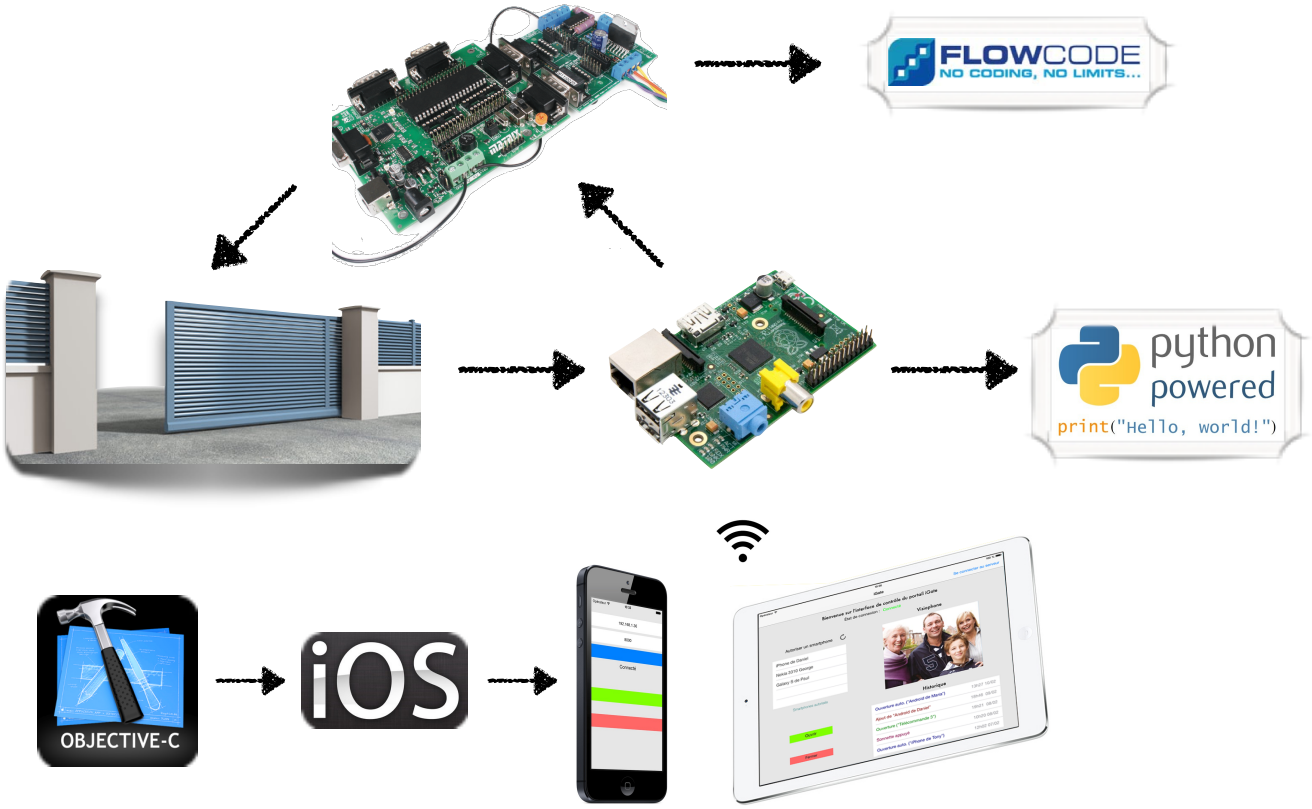
Blocks diagram



Development

Preface

Simulation and hardware available forced us to use a motor that simulates the operation of a sliding gate, but the overall operation is exactly the same. Programming is the most important part of the project, it makes live the heart of the system.
We find in several places of the system : in the smartphone and tablet applications in the Raspberry Pi server program and the E-Block.
To recap, I realized applications for the iPhone and iPad and the server Raspberry Pi For the remainder of this file program, we look at the iPhone application and the server program.
Below is a diagram showing the relationship between programming and gate.

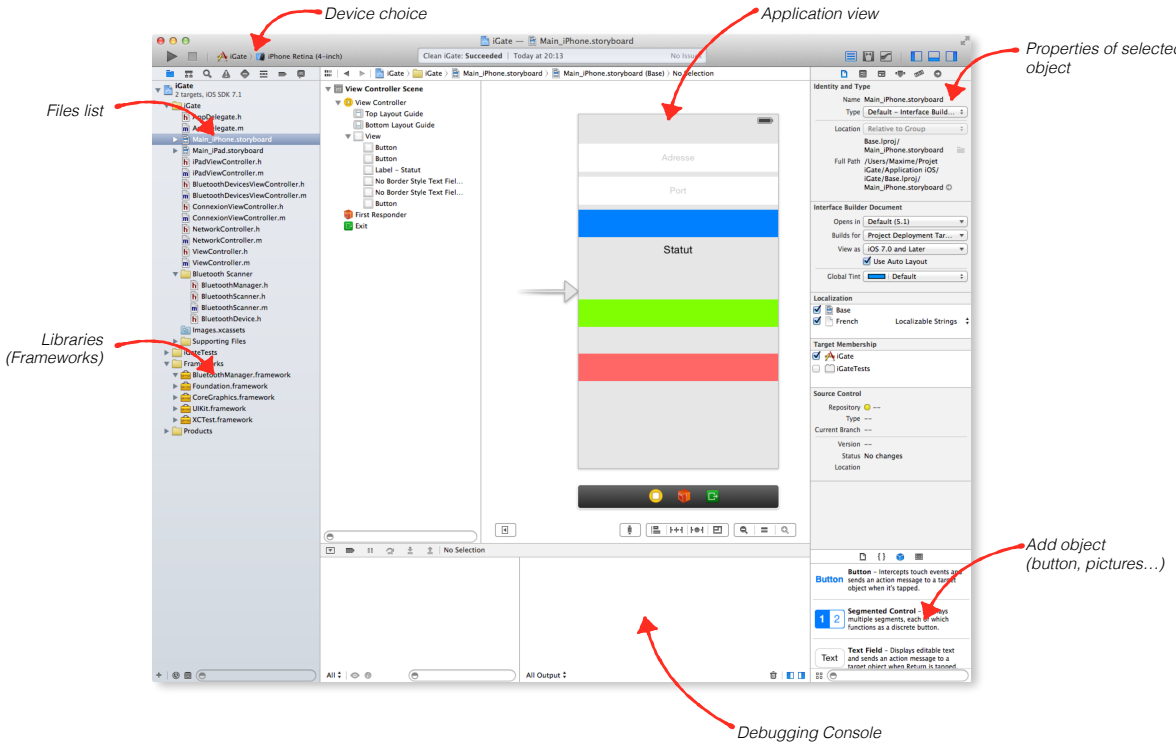


The opening manager (server) is a program written in Python, a programming language multi-platform created in 1990. Has the particularity of having a clean and simple syntax for easy readability. In addition, it's a open-source language and we can easily find functions or software libraries with an active community on the Internet.

The iPhone and iPad are devices running on the iOS operating system, which is based on Unix (same basis as Linux and Android) system, but it accepts only one language: Objective-C.
A language object oriented from 1983: each function is independent and can be used as desired. There are two extensive libraries : Coca and GNUStep that avoids rewriting many basic functions.

Interface

To develop an iOS application, one software exist : Xcode, a software development kit (SDK). The interface stands is in one file: the Storyboard.



Programming

Objective-C is a derivative of C + +, the syntax is similar. The code consists of two files: the Header and Main.

The header file :

- calls required libraries (« ready-to-write-code » to avoid rewrite basic functions)
- announces objects interface (to exploit them such as detect support, change a displayed value...)
- announces functions.

The main file includes the following functions : each piece of code is called by an event (launching the application, pressing a button …).

```objc
// Import the library
#import <UIKit/UIKit.h>

// Definition of class type
@interface ViewController : UIViewController

// Creation and properties of each object
// nonatomic : used for many threads at the same time
// managed by the ARC (automatic reference counting)
@property(nonatomic, strong) IBOutlet UITextField *host;
@property(nonatomic, strong) IBOutlet UITextField *port;

@property(nonatomic, strong) IBOutlet UIButton *connect;
@property(nonatomic, strong) IBOutlet UIButton *openButton;
@property(nonatomic, strong) IBOutlet UIButton *closeButton;

@property(nonatomic, strong) IBOutlet UILabel *statut;

// Creation of functions (instance methods)
- (IBAction)doConnect:(id)sender;
- (IBAction)openGate:(id)sender;
- (IBAction)closeGate:(id)sender;

@end
```

Header file

```objc
// Function performed when the application is launched
- (void)viewDidLoad
{
    [super viewDidLoad];

    // Update statut
    statut.text = @"Not connected";

    // Load saved IP address and port from User Defaults
    NSUserDefaults *defaults = [NSUserDefaults
standardUserDefaults];

    if ([defaults objectForKey:@"iGate.host"] != nil)
        host.text = [defaults objectForKey:@"iGate.host"];

    if ([defaults objectForKey:@"iGate.port"] != nil)
        port.text = [defaults objectForKey:@"iGate.port"];

    // Hide button because we are disconnected
    openButton.hidden = YES;
    closeButton.hidden = YES;

}
```

Main file

Main file (end)

```
// Function when you press the open button
- (IBAction)openGate:(id)sender {

    // Message to gate opening
    [[NetworkController sharedInstance] sendMessage:@"open_gate"];

}

// Function when you press the close button
- (IBAction)closeGate:(id)sender {

    // Message to close opening
    [[NetworkController sharedInstance] sendMessage:@"close"];

}

- (IBAction)doConnect:(id)sender {

    // Saving IP address and port
    NSUserDefaults* defaults = [NSUserDefaults standardUserDefaults];
    [defaults setObject:[host text] forKey:@"iGate.host"];
    [defaults setObject:[port text] forKey:@"iGate.port"];
    [defaults setBool:NO forKey:@"iGate.auto"];
    [defaults synchronize];

    // Hide keyboard
    [[self host] resignFirstResponder];
    [[self port] resignFirstResponder];

    // Statut update
    statut.text = @"Connecting…";

    // Starting NetworkController
    [[NetworkController sharedInstance] updateIP];
    [[NetworkController sharedInstance] connect];

    [NetworkController sharedInstance].connectionOpenedBlock = ^(NetworkController* connection){

        // Connection done, updating statut
        statut.text = @"Connecté";

        // Display open and close button
        openButton.hidden = NO;
        closeButton.hidden = NO;

    };
}
```

The client is ready ! Remains to write the source code of the server.
Python use a different syntax, although approaching the Objective-C, a single file is needed. The code is unified :

- libraries :

```
# Import libraries
from twisted.internet.protocol import Factory, Protocol
from twisted.internet import reactor
from twisted.internet import task
from bluetooth import *
```

- global variables :

```
# Timeout and loop for Bluetooth scan
bluetoothLoop = task.LoopingCall(scanBluetoothDevice)
timeout = 15.0

# Creating pickle (similar to NSUserDefaults)
pickleBT = pickle

# Creating Bluetooth address list
btAdressList = list();

# Prowl object for notification
apikey = '37ffac54cd7dc51ce06e55ec2c77eb9db386783d'
prowlNotif = prowlpy.Prowl(apikey)

# Variable for actual device and MAC address
actualDevice = "device"
actualMac = "mac"
# Parse ID
APPLICATION_ID = "Uq7t2TwJZItGuq1Ga7ivUxpdQ1K9KJ8Js33LRdkv"
```

- functions :

```
# Initialisation and starting server
factory = Factory()
factory.protocol = iGateServer
reactor.listenTCP(PORT, factory, 50, IFACE)
reactor.run()

# Function for receiving a message
class iGateServer(Protocol):
  def connectionMade(self):

    print("A device is connected")

  def dataReceived(self, data):

    # Message reçu
    if data == "open_gate":
      print("Opening gate…")
      motor.open
      try:
        # Envoi d'une notification
        prowlNotif.sendMessage("Gate open !")
        # Mise à jour de l'historique
        histoy = []
        histoy.append(createBTList(name=actualDevice, address=actualMac))
        batcher = ParseBatcher()
        batcher.batch_save(histoy)
      except Exception,msg:
        print("Unable to open gate")
```

Conclusion

Our project was more interesting. Designing a home automation system is a challenge that must comply with the guidelines to avoid surprises. In addition, a collaborative team committed the deadlines over several months, a further constraint ignored so far. A challenge achieved improving an existing need: a secure automatic gate controlled by external device.
Imagine technological solutions is informative and gives us to think logically. This project also allowed us to learn how to find solutions when unexpected problems occur: dysfunction, incorrect code, abnormal interruption... By the exact location of the problem, it becomes possible to solve through documentation, the team members or a teacher, or even an additional reflection.

Amount of the project

iGate at a reasonable cost, although there have no gate itself. The cost of smartphone or tablet is not accounted for, it's assumed that you are already in possession of both.
The total price is $240 allocated as follows :
1 complete Raspberry Pi : $80
1 E-Block set : $130
1 servomotor : $30

Thanks

I extend my thanks to Mr. S. Siot Taillefer and Mr. JL. Tayan for all help provided during this 28 sessions.