

# STRONG BOX 3000

## ETAPE 2

```
3 require File.expand_path("../config/initializers", __FILE__)
4 # Prevent database truncation if the environment is production
5 abort("The Rails environment is running in production mode!")
6 require "rails_helper"
7 require "spec/rails"
8
9 require "capybara/rspec"
10 require "capybara/rspec"
11
12 Capybara.javascript_driver = :webkit
13 Category.delete_all; Category.create
14 Shoulda::Matchers.configure do |config|
15   config.integrate do |with|
16     with.test_framework :rspec
17     with.library :rails
18   end
19 end
20
21 # Add additional requires below this line. Make sure to require any file
22 # Requires supporting ruby files with support for Capybara, e.g.
23 # spec/support/ and its subdirectories. These files will be loaded by
24 # run as spec files by default. This way the support files can be
25 # in _spec.rb will both be required and loaded. This way the support
26 # run twice. It is recommended that you do not use the support
27 # end with _spec.rb. You can configure the support files to be
28 # loaded on the command line by the command line.
29
30 No results found for 'mongoid'
```

---

## GROUPE 6

# 1. INTRODUCTION

---

Ce document présente l'étape 2 du projet intitulé STRONGBOX 3000 qui entame la partie de conception du code arduino par des logigrammes.

## 2. CONTEXTE

---

La célèbre agence d'espionnage MI7 a vécu des heures sombres ces dernières semaines. À plusieurs reprises, le matériel mis à disposition sur le terrain pour leurs agents a été détourné. Nous sommes positionnés en tant qu'une équipe de conception en ingénierie qui a été désignée pour travailler sur un projet qui pourrait régler une bonne fois pour toutes le problème que l'MI7 a rencontré : un coffre-fort qui requiert plusieurs touches de sécurité pour s'ouvrir.

## 3. DÉMARCHE

---

Sur le logiciel flowgorithm, nous concevons les logigrammes du code dans le but de bien comprendre son fonctionnement et pouvoir réaliser plus tard sa version définitive avec le langage C arduino.

## 4. TRAVAIL À FAIRE

---

1/-les algorithmes (sous forme de logigrammes) de chaque mécanisme d'authentification

+

2/-l'algorithme principal (sous forme de logigramme) du système d'authentification du coffre décrivant le processus complet : depuis l'identification du modèle de carte, la détermination du niveau de sécurité, l'appel aux différents mécanismes d'authentification et jusqu'à l'ouverture du coffre

Dans notre logigramme chaque **MA** est une *fonction*.

*télécharger le logigramme:* [STRONGBOX\\_3000\\_LOGIGRAMME](#)

*ouvrir avec le logiciel flowgorithm:* [flowgorithm download](#)

3/-Une liste des variables utilisées avec leur type. Vous préciserez également leur portée

Column1 ▾	Main() ▾	Column3 ▾
Boolean	Loop	local
Real	InputVoltage	local

Column1	DetermineMas()	Column2
Integer	AllModels[]	Global
Integer	SecurityLVLS[]	Global
Integer	CurrentSecurityLVL	Global
Integer	i	Local
Boolean	MA	Local
Column1	MA1()	Column2
Boolean	Status	Local
Integer	Chances	Global
Integer	i	Local
Integer	v	Local
Boolean	S	Local
String	Question	Local
Integer	InputAnswer	Local
String	Answers[]	Local
Integer	Correct	Local

Column1 MA2() Column2		
Boolean	Status	Global
Boolean	s	Local
Integer	i	Local
Integer	Chances	Global
Integer	C[]	Local
Integer	curentCode	Global
Integer	in	Local
Column1 MA(3) Column2		
Boolean	Status	Local
Integer	E	Local
Boolean	in	Local
Boolean	A	Local
Integer	Chances	Global
Column1 MA(4) Column2		
Boolean	Status	Local
Integer	E	Local
Boolean	in	Local
Boolean	A	Local
Integer	chances	Global

Column1	MAB(5)	Column2
Boolean	Status	Local
Integer	Chances	Global
Integer	i	Local
Integer	z	Local
Boolean	S	Local
String	LetterInput	Local
Integer	CodeInput	Local
String	Agents[]	Global
Integer	Codes[]	Global

## 5. CONCLUSION

---

Nous sommes désormais proches de l'aboutissement de ce projet.  
Il reste encore à transformer les logigrammes en code Arduino.

**FIN DU DOCUMENT.**