UNIVERSITÉ TOULOUSE III PAUL SABATIER

Département EEA - Faculté Sciences et Ingénierie

Master 1 SME

UE Conception Systèmes

Année 2023-2024

Compte rendu de travaux pratiques

-

Borne de recharge

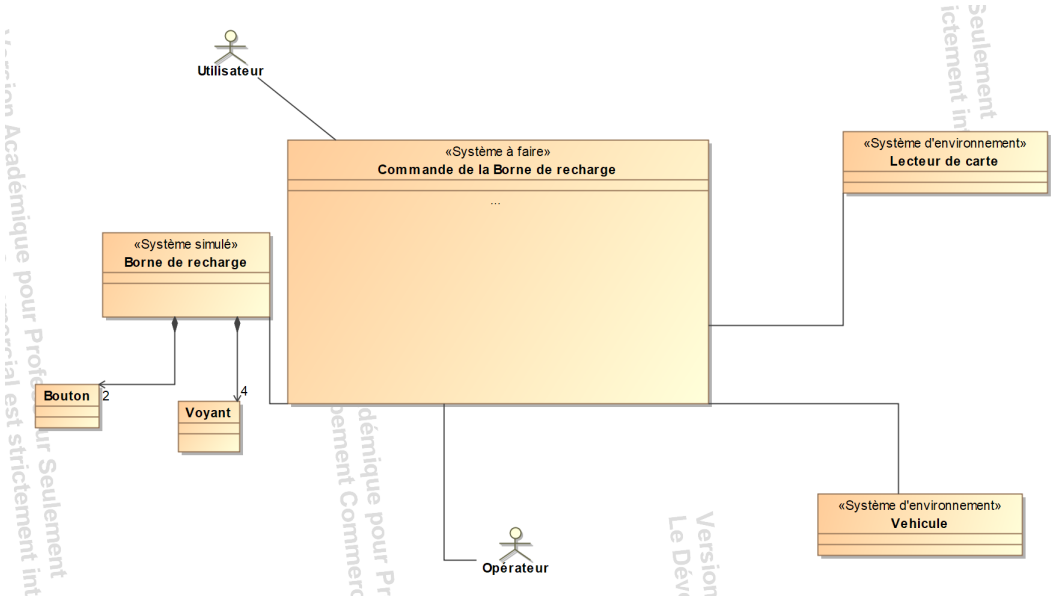
Rédigé par
Erwann Jamin | 22000256
Christopher Rakotondratsima | 22017131

27 mars 2024

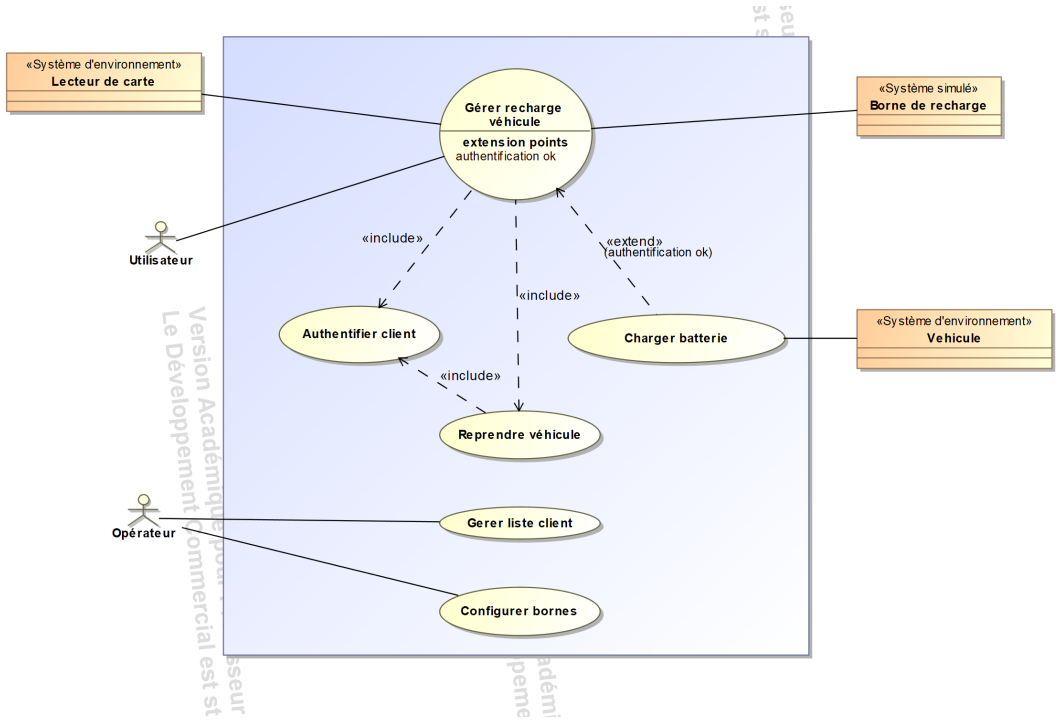
Table des matières

1 Analyse

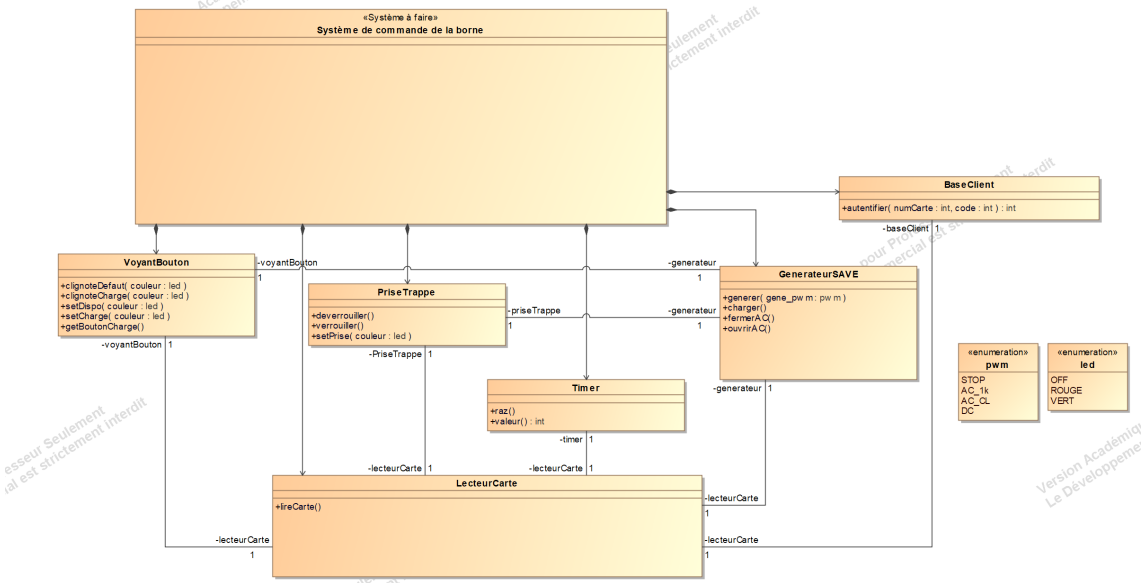
1.1 Diagramme de contexte



1.2 Diagramme des cas d'utilisation



1.3 Ébauche de diagramme de classes



1.4 Ébauche de diagrammes de séquence

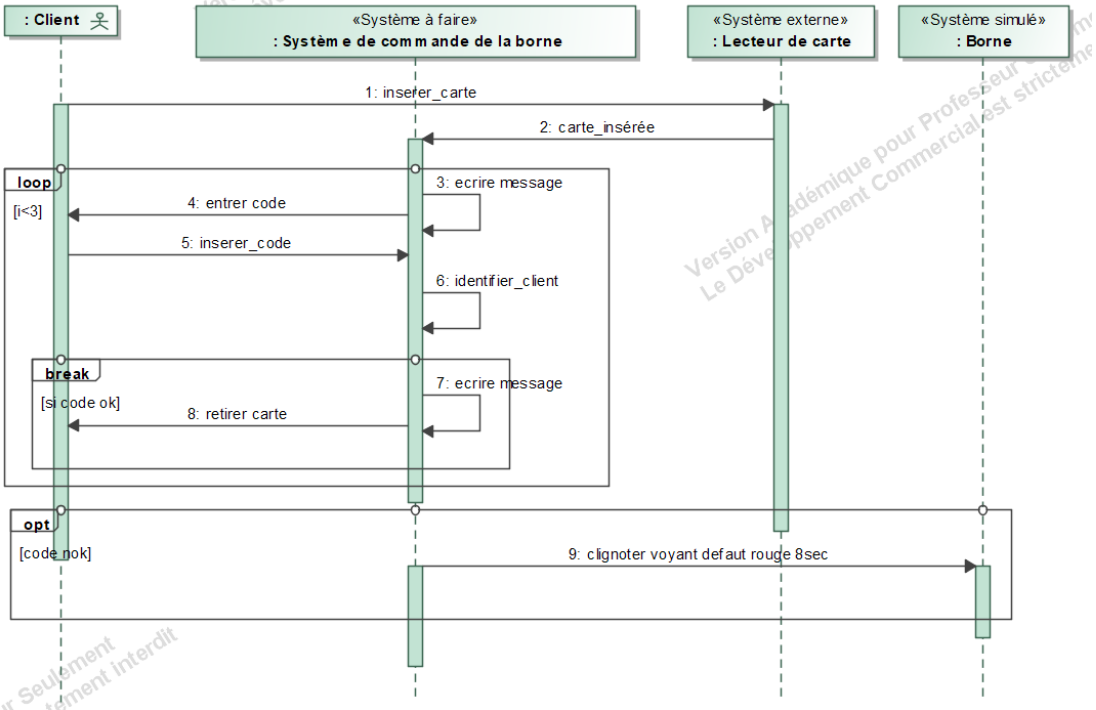


FIGURE 1 – Authentification client

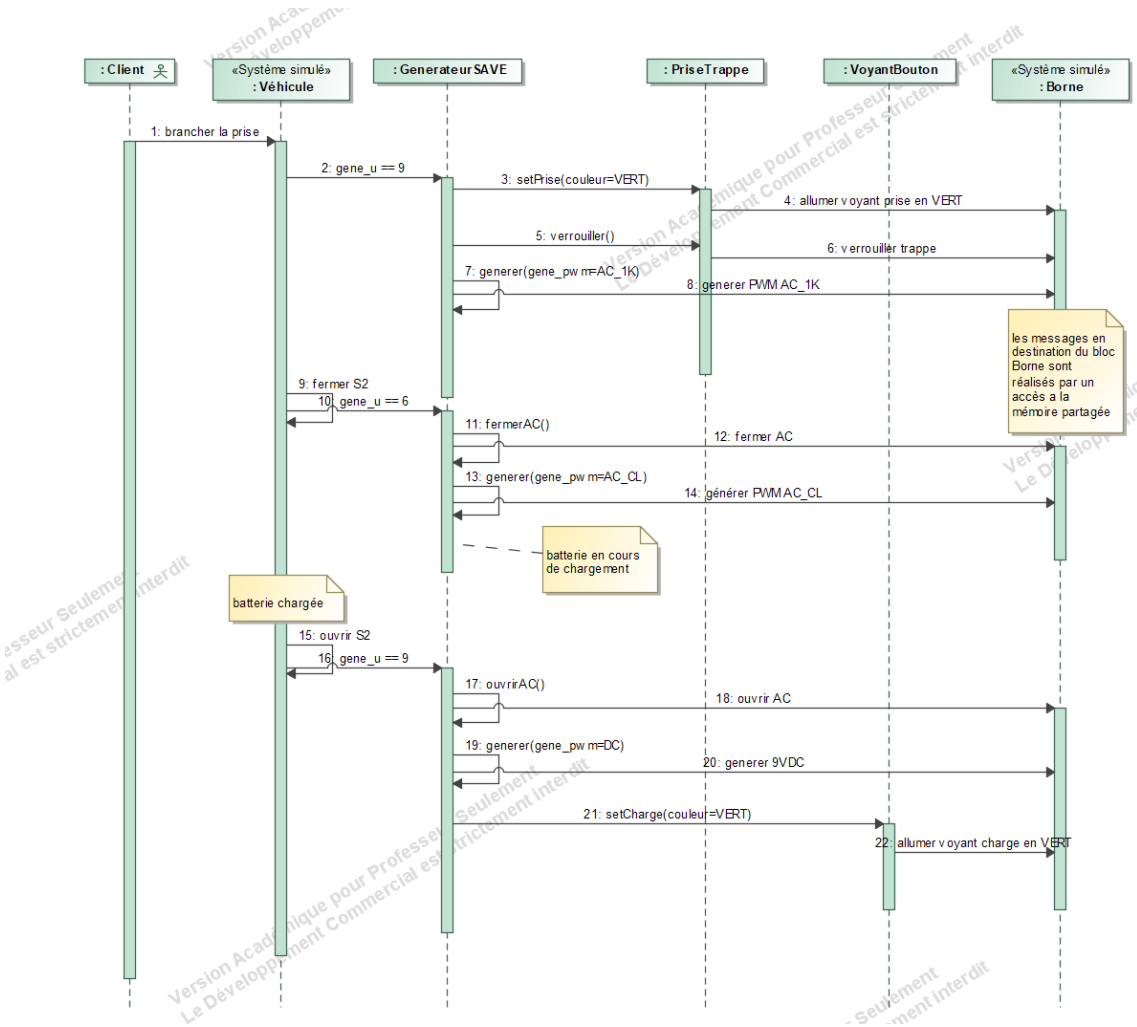


FIGURE 2 – Chargement de la batterie

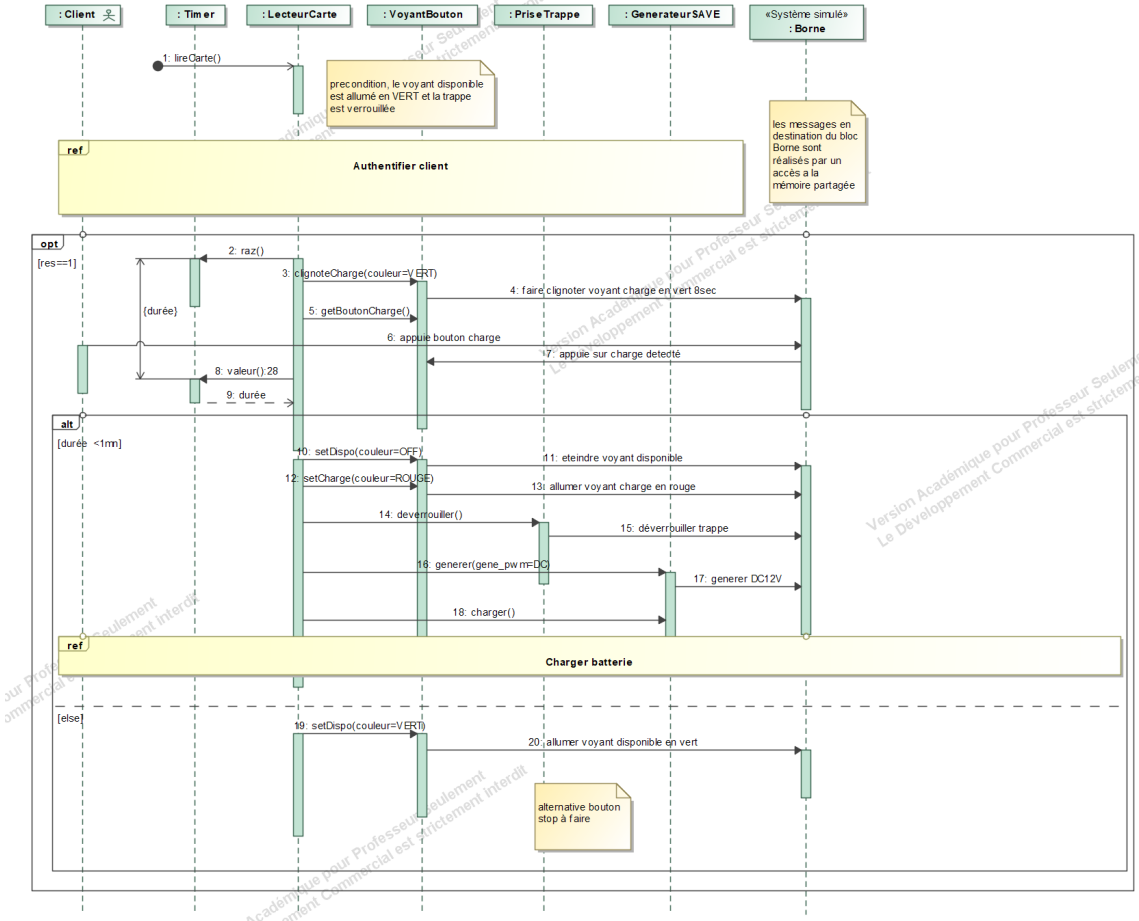


FIGURE 3 – Gestion de la charge du véhicule

1.5 Diagrammes de collaborations

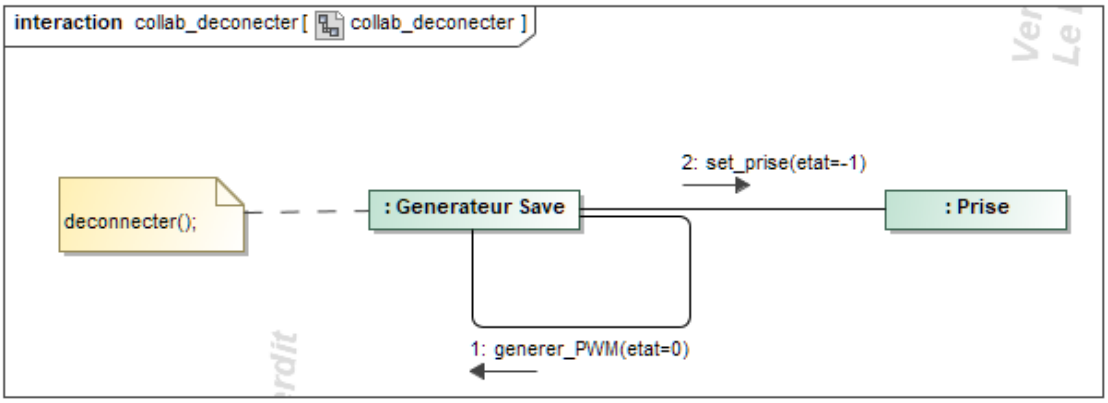
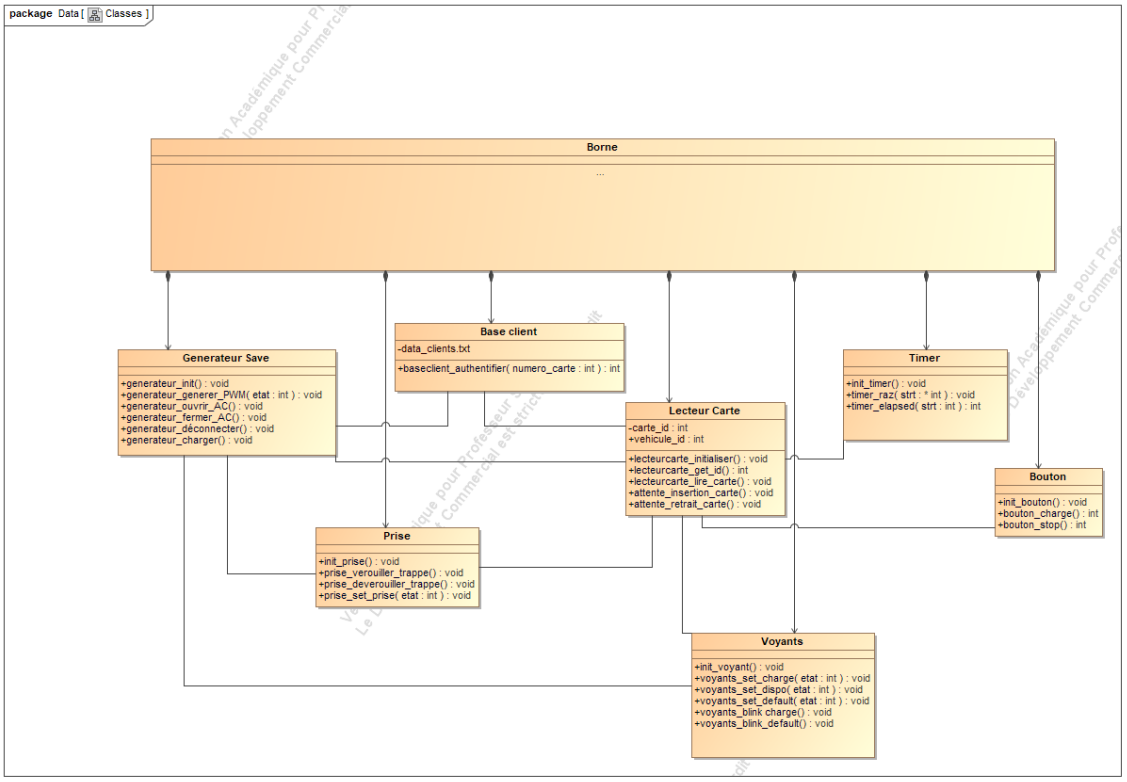


FIGURE 4 – Déconnections

2 Conception

2.1 Diagramme de classes final



2.2 Description des contrats types

Se référer à la documentation en annexe.

2.3 Use case principal : gérer recharge véhicule

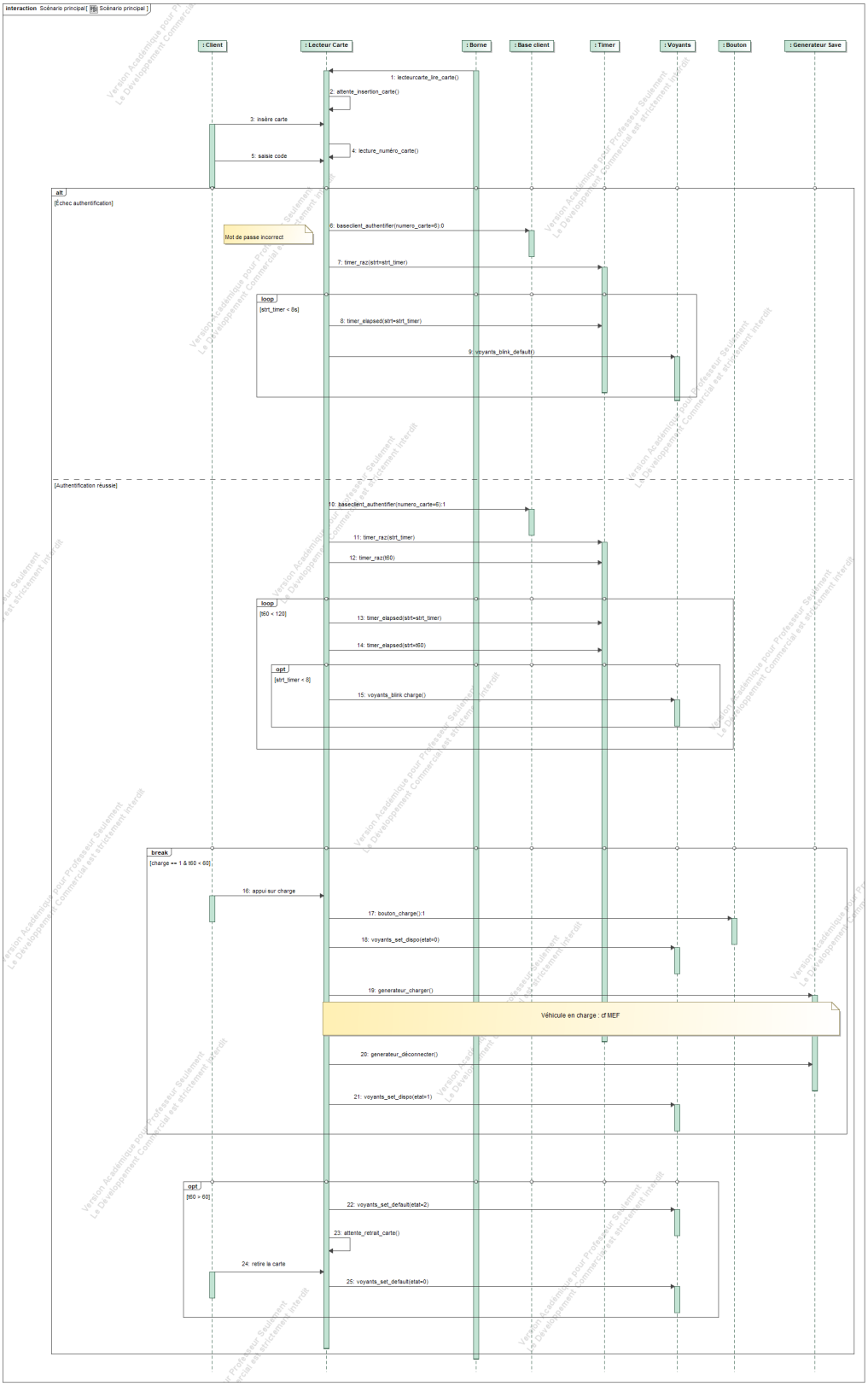


FIGURE 5 – Diagramme de séquence principal

2.4 Use case : charger batterie

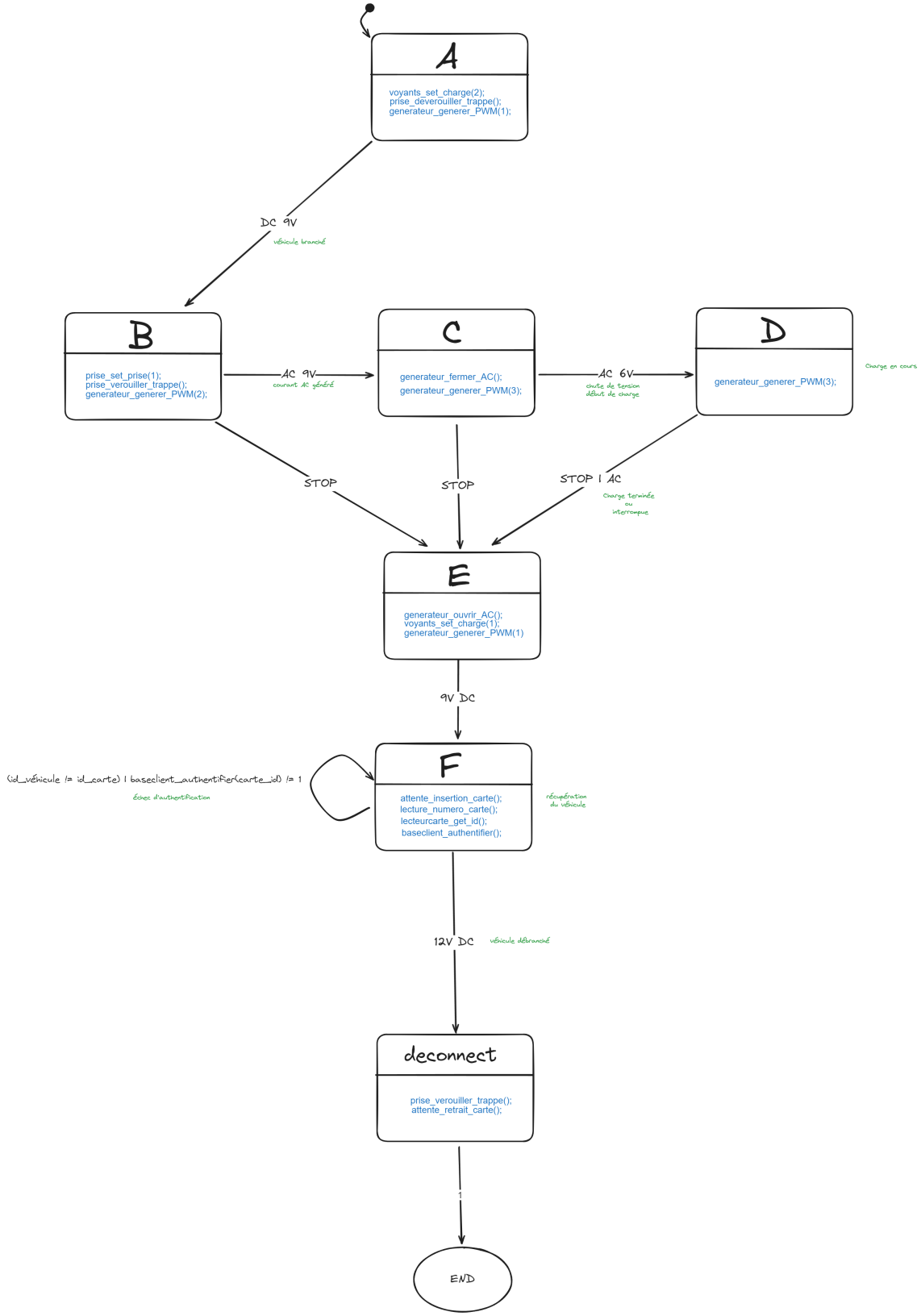


FIGURE 6 – Machine à états descriptive

2.5 Use case : reprendre véhicule

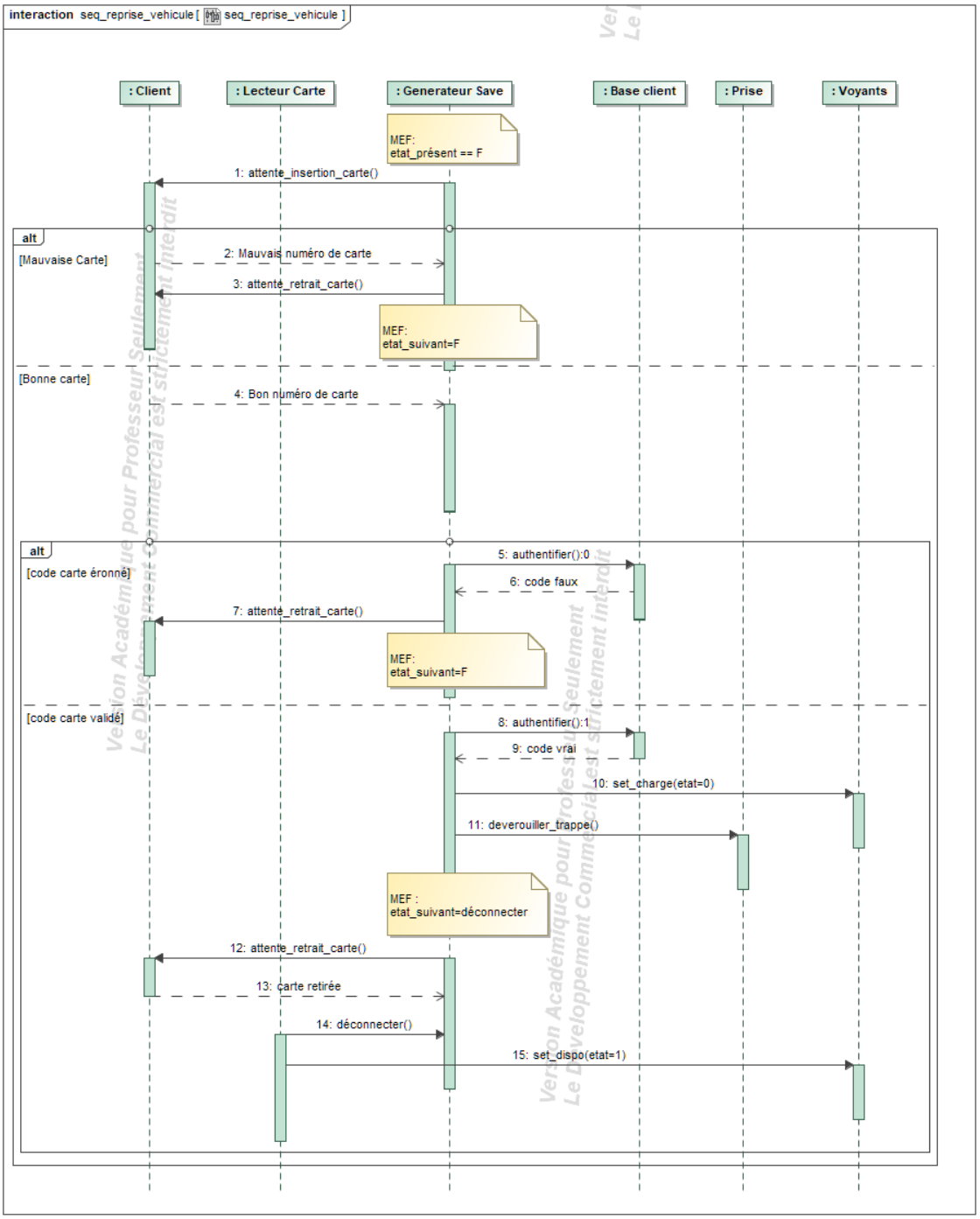


FIGURE 7 – Diagramme de séquence de la reprise du véhicule

3 Implémentation en C

Vous trouverez ci-après les codes c respectifs de chaque classes :

Listing 1 – borne.c

```c
#include <stdio.h>
#include <memoire_borne.h>
#include <donnees_borne.h>

#include "lecteurcarte.h"

int main()
{

    lecteurcarte_initialiser();

    while (1)
    {
        lecteurcarte_lire_carte();
    }

}
```

Listing 2 – lecteurcarte.c

```
#include<stdio.h>
#include <unistd.h>

#include "lecteurcarte.h"
#include "baseclient.h"

#include "voyants.h"
#include "bouton.h"
#include "prise.h"
#include "timer.h"
#include "generateur_save.h"

//=============================================================

// attribut priv   s
int carte_id ;

// attribut public
int vehicule_id ;

//=============================================================

// m   thodes
void lecteurcarte_initialiser()
{
        initialisations_ports();
        init_voyant();
        init_bouton();
        init_timer();
        generateur_init();
}

int lecteurcarte_get_id()
{
        return vehicule_id ;
}

void lecteurcarte_lire_carte()
{
        int strt_timer, t60;     // timers
        printf("\nVeuillez inserer votre carte. \n");
        attente_insertion_carte();

        carte_id = lecture_numero_carte();

        printf(">ID Carte : %d.\n", carte_id );

// ------------------------------------------------------------------
        if ( baseclient_authentifier( carte_id ) == 1 )
        {
                // SI AUTHENTIFICATION REUSSIE
                vehicule_id = carte_id ;
                printf(">Authentification r   ussie \n");
                timer_raz(&strt_timer);                         //mise a zero du timer
                timer_raz(&t60);

                while(timer_elapsed(t60) < 120)          // Buffer test 2mn
                {
                        if(timer_elapsed(strt_timer)<8) // Blink charge 8s
                        {
                                voyants_blink_charge();
                                //printf("delai : %ds\n",8-timer_elapsed(strt_timer) );
                        }

                        if (bouton_charge() ==1)                // si bouton appuy         temps
                        {
                                voyants_set_dispo(0);

                                printf("\nVeuillez retirer votre carte.\n");
                                attente_retrait_carte();

                                generateur_charger(); // chargement
```

```
                        break ;
                }

                if(timer_elapsed(t60)>=60)      // temps imparti      coul      60s
                {
                        printf("\n>Temps ecoule\n");
                        voyants_set_default(2);
                        sleep(1);

                        printf("\n>Veuillez retirer votre carte.\n");
                        attente_retrait_carte();
                        printf(">carte retiree \n");

                        sleep(1);
                        voyants_set_default(0);
                        return ;
                }
        }



    }
// ------------------------------------------------------------------------
    else // SI ECHEC AUTHENTIFICATION
    {
        printf(">Authentification echec \n");
        timer_raz(&strt_timer);
        while(timer_elapsed(strt_timer)<8)      // default => clignote 8s
        {
                voyants_blink_default();
        }

        printf("\n>Veuillez retirer votre carte.\n");
        attente_retrait_carte();
        printf(">carte retiree \n");
        voyants_set_default(0);
        return ;
    }

// ------------------------------------------------------------------------
    // SI r   cup   ration v   hicule succ   s
    generateur_deconnecter();
    voyants_set_dispo(1);

    printf("\n----\nFIN\n----\n");
}
```

Listing 3 – bouton.c

```c
#include<stdio.h>

#include "bouton.h"

#include<donnees_borne.h>
#include<memoire_borne.h>

entrees* io ;
int shmid ;

void init_bouton()
{
        io=acces_memoire(&shmid);
}


int bouton_charge()
{
        int state = io->bouton_charge ;
        return state ;

}

int bouton_stop()
{
        int state = io->bouton_stop ;
        return state ;
}
```

Listing 4 – voyants.c

```
#include<stdio.h>
#include <unistd.h>

#include "voyants.h"
#include "timer.h"

#include<donnees_borne.h>
#include<memoire_borne.h>

entrees* io ;
int shmid ;
int timer_start   ;
int timer_sec = 0;

void init_voyant()
{
        io=acces_memoire(&shmid);
}

void voyants_blink_charge()
{
                io->led_charge = VERT ;
                usleep( 500000 ); // attente 0.5 s
                io->led_charge = OFF ;
                usleep( 500000 ); // attente 0.5 s
}

void voyants_set_dispo(int etat)
{

        switch(etat)
        {       case 0:
                        io->led_dispo = OFF ;
                        break ;
                case 1:
                        io->led_dispo = VERT;
                        break ;
                default:
                        io->led_dispo = OFF;
                        break;
        }

}

void voyants_set_default(int etat)
{

        switch(etat)
        {       case 0:
                        io->led_defaut = OFF ;
                        break ;
                case 1:
                        io->led_defaut = VERT;
                        break ;
                case 2:
                        io->led_defaut = ROUGE;
                        break ;
                default:
                        io->led_defaut = OFF;
                        break;
        }

}
void voyants_blink_default()
{
                io->led_defaut = ROUGE ;
                usleep( 500000 ); // attente 0.5 s
                io->led_defaut = OFF ;
                usleep( 500000 ); // attente 0.5 s
}
```

```
void voyants_set_charge(int etat)
{

        switch(etat)
        {       case 0:
                        io->led_charge = OFF ;
                        break ;
                case 1:
                        io->led_charge = VERT;
                        break ;
                case 2:
                        io->led_charge = ROUGE;
                        break ;
                default:
                        io->led_charge = OFF;
                        break;
        }

}
```

Listing 5 – prise.c

```
#include "prise.h"
#include<stdio.h>
#include<donnees_borne.h>
#include<memoire_borne.h>

entrees* io ;
int shmid ;

void init_prise()
{
        io=acces_memoire(&shmid);
}
void prise_verouiller_trappe()
{
        io->led_trappe = OFF ;
}

void prise_deverouiller_trappe()
{
        io->led_trappe = VERT;
}

void prise_set_prise(int etat)
{
        switch(etat)
        {       case 0:
                        io->led_prise = ROUGE ;
                        break ;
                case 1: // cas prise branch   e
                        io->led_prise = VERT;
                        break ;
                default:        // cas prise rang   e
                        io->led_prise = OFF;
                        break;
        }
}
```

Listing 6 – timer.c

```c
#include<stdio.h>
#include <unistd.h>

#include "timer.h"

#include<donnees_borne.h>
#include<memoire_borne.h>

entrees* io ;

int shmid ;


time_t now ; // temps actuel
time_t start_time ;
//=============================================================

void init_timer()
{
        io=acces_memoire(&shmid);

}

void timer_raz(int* strt)
{
        *strt = io->timer_sec;
}

int timer_elapsed( int strt)
{

        int delta = io->timer_sec-strt;
        //printf("delt_time %d\n", io->timer_sec);
        return delta;
}
```

Listing 7 – generateur_save.c

```
//generateur_save.c

#include<stdio.h>
#include<donnees_borne.h>
#include<memoire_borne.h>
#include <unistd.h>
#include<time.h>

#include "baseclient.h"
#include "lecteurcarte.h"
#include "generateur_save.h"
#include "voyants.h"
#include "prise.h"

entrees* io ;
int shmid ;

// ================================================================
// attribut priv
int carte_id ;

// ================================================================
// m   thodes
void generateur_init()
{
        io=acces_memoire(&shmid);
}

void generateur_charger()
{
        voyants_set_charge(2); // Voyant charge => ROUGE
        prise_deverouiller_trappe();

// ----------------------------------------------------------------
// MEF
        typedef enum _etat {A,B,C,D,E,F, pre_deconnect,deconnect, END} etat;
        etat etat_present, etat_suivant ;

        // initialisation
        etat_present = A ;
        etat_suivant=A;

        printf("Veuillez brancher la prise\n");
        while(etat_present!=END)
        {
                //Bloc F : transitions
                if ((etat_present == A) && (io->gene_u == 9) && (io->gene_pwm == DC) )
                {
                        etat_suivant = B ;
                        printf(">prise branchee\n");
                }
                else if (etat_present == B)
                {
                        if ( (io->gene_u == 9) && (io->gene_pwm == AC_1K) )
                                etat_suivant = C;
                                printf(">connexion a la voiture\n");
                        }
                        else if (io->bouton_stop ==1)
                        {       etat_suivant = E;
                                printf(">demande stop\n");
                        }
                }

                else if (etat_present == C)
                {
                        if(io->gene_u == 6)
                        {
                                etat_suivant = D;
                                printf(">chargement...\n");
                        }
                        else if(io->bouton_stop==1)
                        {       etat_suivant=E;
```

```
                        printf(">demande stop\n");
                }

        }

        else if (etat_present == D)
        {
                if( (io->bouton_stop==1) || ( (io->gene_u == 9) && (io->gene_pwm !=DC) ) )
                {
                        etat_suivant=E;
                        printf(">fin de charge\n");
                }
        }
        else if(etat_present == E)
        {
                if( /*(io->gene_u == 12) &&*/ (io->gene_pwm =DC) )
                {       etat_suivant = F;
                        printf(">r   cup   rer vehicule\n");
                }
        }
// ------------------------------------------------------------------
// blocs hybrides
        else if(etat_present == F)
        {
                attente_insertion_carte();
                carte_id = lecture_numero_carte();

                if (lecteurcarte_get_id() == carte_id)
                {
                        printf("\n--\n>ID CORRECT\n--\n");

                        if ( baseclient_authentifier( carte_id ) == 1 )
                        {
                                // authentification succ   s

                                printf(">Veuillez debrancher le vehicule\n");
                                voyants_set_charge(0); // Voyant Charge => OFF
                                prise_deverouiller_trappe(); // Prise      d   brancher
                                etat_suivant = deconnect;
                        }
                        else
                        {
                                printf(">mauvais mot de passe\n");
                                printf(">retirer carte et reessayer\n");
                                attente_retrait_carte();
                                printf(">carte retiree\n\n");
                                etat_suivant = F;
                        }
                }
                else
                {
                        printf("\n--\nCECI N'EST PAS VOTRE VEHICULE\n--\n");
                        printf(">RETIREZ CARTE\n");
                        attente_retrait_carte();
                        printf(">carte retiree\n\n");
                        etat_suivant = F;
                }
        }

        else if (etat_present == deconnect)
        {
                if( (io->gene_u == 12) && (io->gene_pwm =DC) )
                {
                        prise_verouiller_trappe();
                        printf("\n>Veuillez retirer votre carte.\n");
                        attente_retrait_carte();
                        printf(">carte retiree \n");
                        etat_suivant = END ;
                }
        }

// ------------------------------------------------------------------
        //Block M : m   moire
```

```
                etat_present = etat_suivant;
                sleep(1);
// ------------------------------------------------------------------------
                //Block G : sorties
                if(etat_present == A)
                {
                        voyants_set_charge(2); // Voyant charge => Rouge
                        prise_deverouiller_trappe();
                        generateur_generer_PWM(1); // 12V DC output

                }

                if(etat_present == B)
                {
                        prise_set_prise(1); // Voyant prise => Vert
                        prise_verouiller_trappe();
                        generateur_generer_PWM(2); // AC_1K output
                }

                if(etat_present == C)
                {
                        generateur_fermer_AC();
                        generateur_generer_PWM(3); // AC_CL output
                }

                if(etat_present == D)
                {
                        generateur_generer_PWM(3); // AC_CL output
                }
                if(etat_present == E)
                {
                        generateur_ouvrir_AC();
                        voyants_set_charge(1); // Voyant charge => Vert
                        generateur_generer_PWM(1); // DC output
                }
        }
}

void generateur_ouvrir_AC()
{
        io->contacteur_AC = 0;
}

void generateur_fermer_AC()
{
        io->contacteur_AC = 1;
}



void generateur_generer_PWM(int etat)
{
        switch(etat)
        {
                case 0:
                        io->gene_pwm = STOP;
                        break;
                case 1 :
                        io->gene_pwm = DC;
                        break;
                case 2 :
                        io->gene_pwm = AC_1K;
                        break;
                case 3 :
                        io->gene_pwm =  AC_CL;
                        break;
                default :
                        io->gene_pwm = STOP;
                        break;
        }
}

void generateur_deconnecter()
```

```
{
                generateur_generer_PWM(0);          // Generateur coup
                prise_set_prise(-1);                // Voyant Prise => OFF
}
```

4 Annexe

UE Conception systèmes
Borne de recharge 1.0
Documentation

Généré par Doxygen 1.10.0

Chapitre 1

Index des fichiers

1.1 Liste des fichiers

Liste de tous les fichiers documentés avec une brève description :

Chapitre 2

Documentation des fichiers

2.1 baseclient.h

```
00001 #ifndef BASECLIENT_H
00002 #define BASECLIENT_H
00003
00004 int baseclient_authentifier( int numero_carte ) ;
00005
00006 int baseclient_reprise();
00007
00008
00009 #endif // BASECLIENT_H
```

2.2 Référence du fichier bouton.c

Bibliothèque des fonctions pour la gestion des boutons.

```
#include <stdio.h>
#include "bouton.h"
#include <donnees_borne.h>
#include <memoire_borne.h>
```

Fonctions

- void **init_bouton** ()

  Initialisation des ports.

- int bouton_charge ()

  Lecture de l'état du bouton charge.

- int bouton_stop ()

  Lecture de l'état du bouton stop.

Variables

- entrees * **io**
- int **shmid**

2.2.1 Description détaillée

Bibliothèque des fonctions pour la gestion des boutons.

Auteur

E.JAMIN & C.RAKOTONDRATSIMA

Version

1.0

Date

05 jan.2024

2.2.2 Documentation des fonctions

2.2.2.1 bouton_charge()

```
int bouton_charge ( )
```

Lecture de l'état du bouton charge.

Renvoie

{1} si bouton appuyé

2.2.2.2 bouton_stop()

```
int bouton_stop ( )
```

Lecture de l'état du bouton stop.

Renvoie

{1} si bouton appuyé

2.3 bouton.h

```
00001 #ifndef BOUTON_H
00002 #define BOUTON_H
00003
00004 void init_bouton();
00005
00006 int bouton_charge();
00007
00008 int bouton_stop();
00009
00010 #endif
```

2.4 Référence du fichier generateur_save.c

Bibliothèque des fonctions associées au générateur de tension.

```
#include <stdio.h>
#include <donnees_borne.h>
#include <memoire_borne.h>
#include <unistd.h>
#include <time.h>
#include "baseclient.h"
#include "lecteurcarte.h"
#include "generateur_save.h"
#include "voyants.h"
#include "prise.h"
```

Fonctions

- void **generateur_init** ()

  Initialisation des ports.

- void **generateur_charger** ()

  séquences d'instructions lors de la charge du véhicule, voir Machine à états

- void **generateur_ouvrir_AC** ()

  Fonction pour ouverture du contacteur AC.

- void **generateur_fermer_AC** ()

  Fonction pour fermeture du contacteur AC.

- void generateur_generer_PWM (int etat)

  0 : arrêt du générateur
  1 : générateur en fonctionnement DC
  2 : générateur en fonctionnement PWM 1Khz
  3 : générateur en fonctionnement PWM à fréquence variable

  autre : arrêt du générateur par défaut

- void **generateur_deconnecter** ()

  Fonction pour déconnecter le générateur et éteindre le voyant prise.

Variables

- entrees * **io**
- int **shmid**
- int **carte_id**

2.4.1 Description détaillée

Bibliothèque des fonctions associées au générateur de tension.

Auteur

E.JAMIN & C.RAKOTONDRATSIMA

Version

5.0

Date

05 jan.2024

2.4.2 Documentation des fonctions

2.4.2.1 generateur_generer_PWM()

```
void generateur_generer_PWM (
            int etat )
```

0 : arrêt du générateur
1 : générateur en fonctionnement DC
2 : générateur en fonctionnement PWM 1Khz
3 : générateur en fonctionnement PWM à fréquence variable

autre : arrêt du générateur par défaut

Paramètres

etat	entier pour commander le fonctionnement du générateur

2.5 generateur_save.h

```
00001 #ifndef GENERATEUR_SAVE_H
00002 #define GENERATEUR_SAVE_H
00003
00004 void generateur_init();
00005
00006 void generateur_generer_PWM(int etat);
00007
00008 void generateur_ouvrir_AC();
00009 void generateur_fermer_AC();
00010
00011 void generateur_charger();
00012
00013 void generateur_deconnecter();
00014
00015 #endif
```

2.6 Référence du fichier lecteurcarte.c

Bibliothèque des fonctions associées au lecteur de cartes.

```
#include <stdio.h>
#include <unistd.h>
#include "lecteurcarte.h"
#include "baseclient.h"
#include "voyants.h"
#include "bouton.h"
#include "prise.h"
#include "timer.h"
#include "generateur_save.h"
```

Fonctions

- void **lecteurcarte_initialiser** ()

  Initialisation des ports partagés.

- void lecteurcarte_lire_carte ()

  Fonction éxécutée en boucle dans le programme principal
  Séquence d'instructions à faire à partir d'une première insertion de carte.

Variables

- int **carte_id**
- int **vehicule_id**

2.6.1 Description détaillée

Bibliothèque des fonctions associées au lecteur de cartes.

Auteur

E.JAMIN & C.RAKOTONDRATSIMA

Version

3.0

Date

05 jan.2024

2.6.2 Documentation des fonctions

2.6.2.1 lecteurcarte_lire_carte()

```
void lecteurcarte_lire_carte ( )
```

Fonction éxécutée en boucle dans le programme principal
Séquence d'instructions à faire à partir d'une première insertion de carte.

```
\fn int lecteurcarte_get_id()
\brief Fonction pour récupérer l'id_du véhicule
\return {vehicule id}
```

int lecteurcarte_get_id() { return vehicule_id ; } /**

2.7 lecteurcarte.h

```
00001 #ifndef LECTEURCARTE_H
00002 #define LECTEURCARTE_H
00003 #include <lcarte.h>
00004
00005 void lecteurcarte_initialiser();
00006 void lecteurcarte_lire_carte();
00007 int lecteurcarte_get_id();
00008
00009 #endif // LECTEURCARTE_H
```

2.8 Référence du fichier prise.c

Bibliothèque des fonctions gérant la trappe et l'état de la prise.

```
#include "prise.h"
#include <stdio.h>
#include <donnees_borne.h>
#include <memoire_borne.h>
```

Fonctions

- void **init_prise** ()

  Initialisation des ports.

- void **prise_verouiller_trappe** ()

  Fonction verrouillant la trappe et éteignant le voyant de trappe.

- void **prise_deverouiller_trappe** ()

  Fonction déverrouillant la trappe et allumant le voyant de trappe en vert.

- void **prise_set_prise** (int etat)

Variables

- entrees * **io**
- int **shmid**

2.8.1 Description détaillée

Bibliothèque des fonctions gérant la trappe et l'état de la prise.

Auteur

E.JAMIN & C.RAKOTONDRATSIMA

Version

1.0

Date

05 jan.2024

2.9 prise.h

```
00001 #ifndef PRISE_H
00002 #define PRISE_H
00003
00004 void init_prise();
00005
00006 void prise_verouiller_trappe();
00007
00008 void prise_deverouiller_trappe();
00009
00010 void prise_set_prise();
00011
00012 #endif
```

2.10 Référence du fichier timer.c

Bibliothèque des fonctions pour la gestion du temps.

```
#include <stdio.h>
#include <unistd.h>
#include "timer.h"
#include <donnees_borne.h>
#include <memoire_borne.h>
```

Fonctions

- void **init_timer** ()

  Initialisation des ports.

- void timer_raz (int *strt)

  fonction d'initialisation du timer
  récupère l'heure actuelle du système

- int timer_elapsed (int strt)

  fonction qui compare l'heure d'initialisation du timer avec l'heure actuelle du système

Variables

- entrees * **io**
- int **shmid**

2.10.1 Description détaillée

Bibliothèque des fonctions pour la gestion du temps.

Auteur

E.JAMIN & C.RAKOTONDRATSIMA

Version

1.0

Date

05 jan.2024

2.10.2 Documentation des fonctions

2.10.2.1 timer_elapsed()

```
int timer_elapsed (
            int strt )
```

fonction qui compare l'heure d'initialisation du timer avec l'heure actuelle du système

Paramètres

<strt>	pointeur sur un entier de référence au timer

Renvoie

{temps écoulé en secondes}

2.10.2.2 timer_raz()

```
void timer_raz (
            int * strt )
```

fonction d'initialisation du timer
récupère l'heure actuelle du système

Paramètres

<strt>	pointeur sur un entier de référence au timer

2.11 timer.h

```
00001 #ifndef TIMER_H
00002 #define TIMER_H
00003
00004 void init_timer();
00005
00006 void timer_raz(int * strt);
00007
00008 int timer_elapsed(int strt);
00009
00010
00011 #endif
```

2.12 Référence du fichier voyants.c

Bibliothèque des fonctions pour la gestion des voyants.

```
#include <stdio.h>
#include <unistd.h>
#include "voyants.h"
#include "timer.h"
#include <donnees_borne.h>
#include <memoire_borne.h>
```

Fonctions

- void **init_voyant** ()

  Initialisation des ports.

- void **voyants_blink_charge** ()

  fait clignoter le voyant charge en vert par intervalles de 500 ms

- void voyants_set_dispo (int etat)

  suivant la valeur de etat :
  1 : voyant éteint
  2 : voyant allumé en vert
  autre : voyant éteint par défaut

- void voyants_set_default (int etat)

  suivant la valeur de etat :
  0 : voyant éteint
  1 : voyant allumé en vert
  2 : voyant allumé en rouge
  autre : voyant éteint par défaut

- void **voyants_blink_default** ()

  fait clignoter le voyant default en rouge par intervalles de 500 ms

- void voyants_set_charge (int etat)

  suivant la valeur de etat :
  0 : voyant éteint
  1 : voyant allumé en vert
  2 : voyant allumé en rouge
  autre : voyant éteint par défaut

Variables

- entrees * **io**
- int **shmid**

2.12.1 Description détaillée

Bibliothèque des fonctions pour la gestion des voyants.

Auteur

E.JAMIN & C.RAKOTONDRATSIMA

Version

1.0

Date

05 jan.2024

2.12.2 Documentation des fonctions

2.12.2.1 voyants_set_charge()

```
void voyants_set_charge (
            int etat )
```

suivant la valeur de etat :
0 : voyant éteint
1 : voyant allumé en vert
2 : voyant allumé en rouge
autre : voyant éteint par défaut

Paramètres

etat	entier pour commander l'état du voyant charge

2.12.2.2 voyants_set_default()

```
void voyants_set_default (
            int etat )
```

suivant la valeur de etat :
0 : voyant éteint
1 : voyant allumé en vert
2 : voyant allumé en rouge
autre : voyant éteint par défaut

Paramètres

etat	entier pour commander l'état du voyant default

2.12.2.3 voyants_set_dispo()

```
void voyants_set_dispo (
            int etat )
```

suivant la valeur de etat :
1 : voyant éteint
2 : voyant allumé en vert
autre : voyant éteint par défaut

Paramètres

etat	entier pour commander l'état du voyant dispo

2.13 voyants.h

```
00001 #ifndef VOYANTS_H
00002 #define VOYANTS_H
00003
00004 void init_voyant();
00005
00006 void voyants_set_dispo(int etat);
00007 void voyants_set_default(int etat);
00008 void voyants_set_charge(int etat);
00009
00010 void voyants_blink_charge();
00011 void voyants_blink_default();
00012
00013 int voyants_dispo();
00014
00015
00016
00017 #endif
```

Index