ScanParking

Projet BTS SN

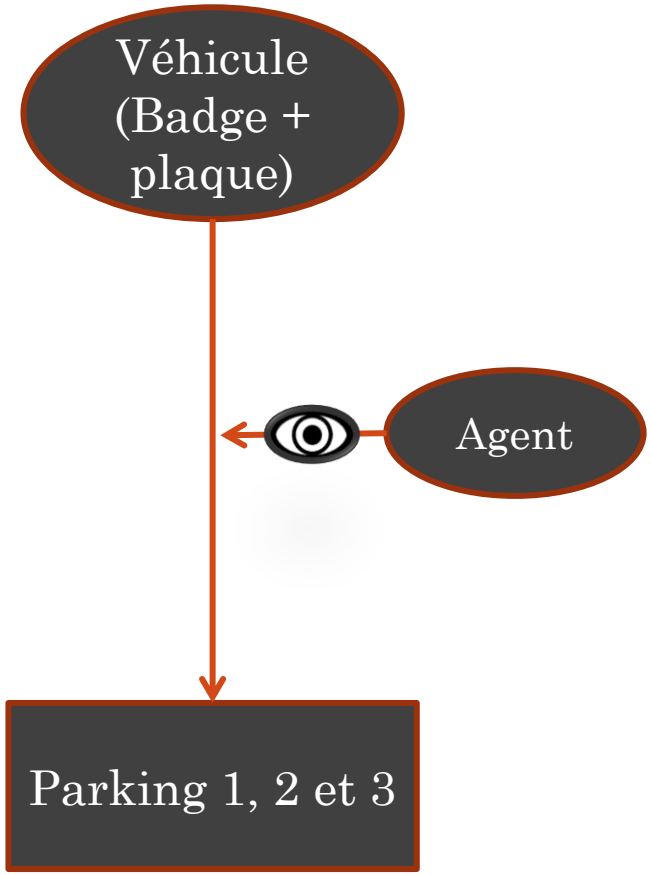
2018-2019

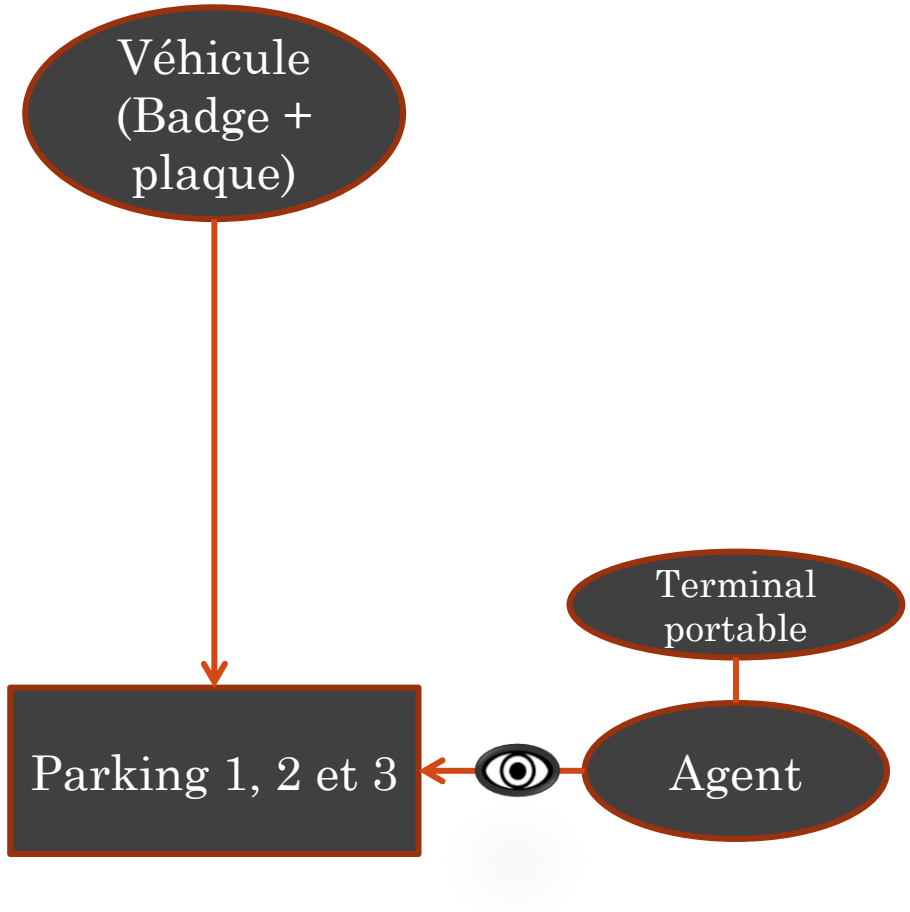
Ruiz Guillaume - Ballanger Alan - Fabris Damien - Maillet Bastien

SOMMAIRE

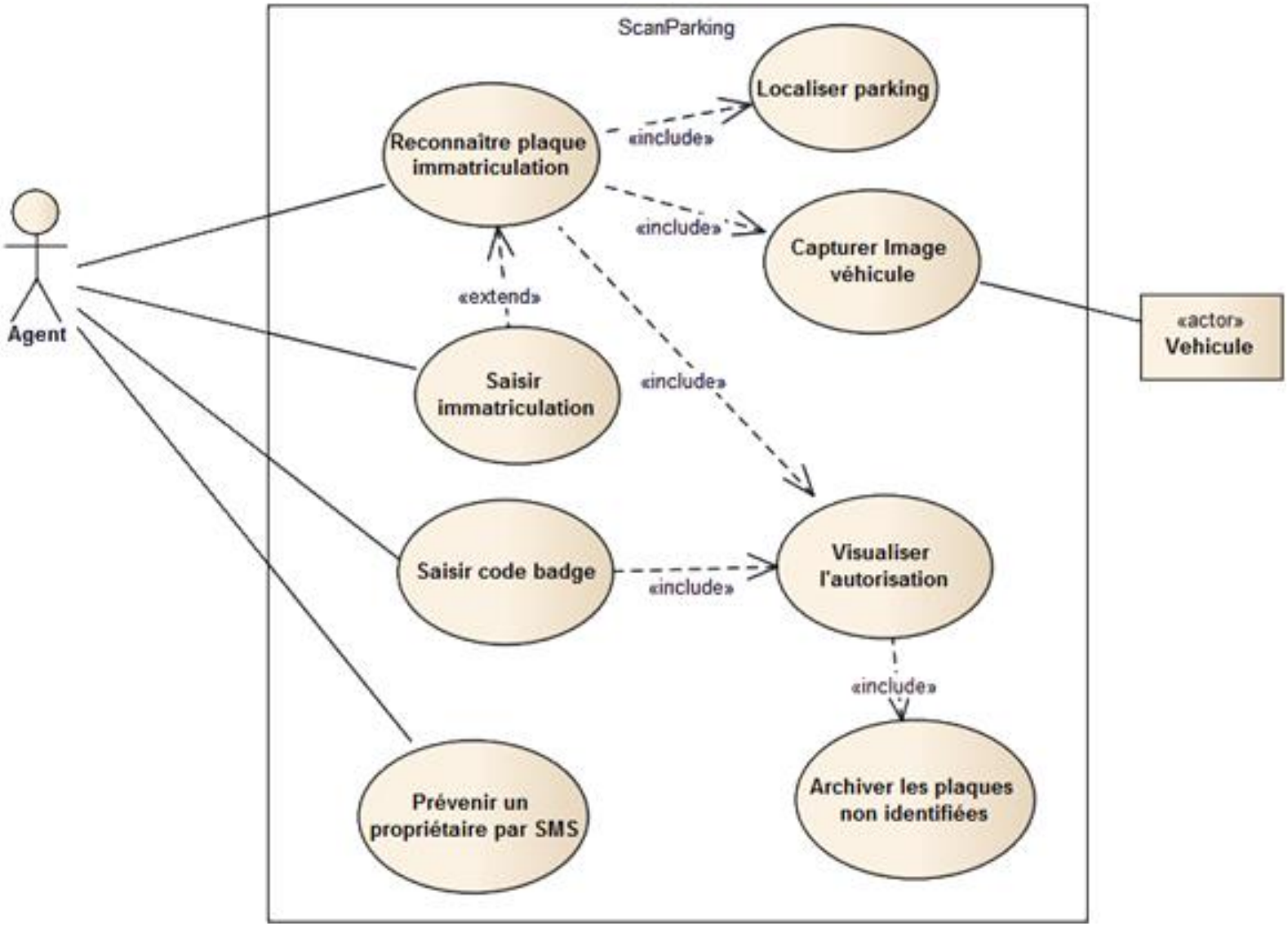
- Présentation du projet
- Cas d’utilisation
- Exigences
- Matériel
- Architecture logicielle
- Fonctionnement global
- Partie personnelle

Présentation du projet

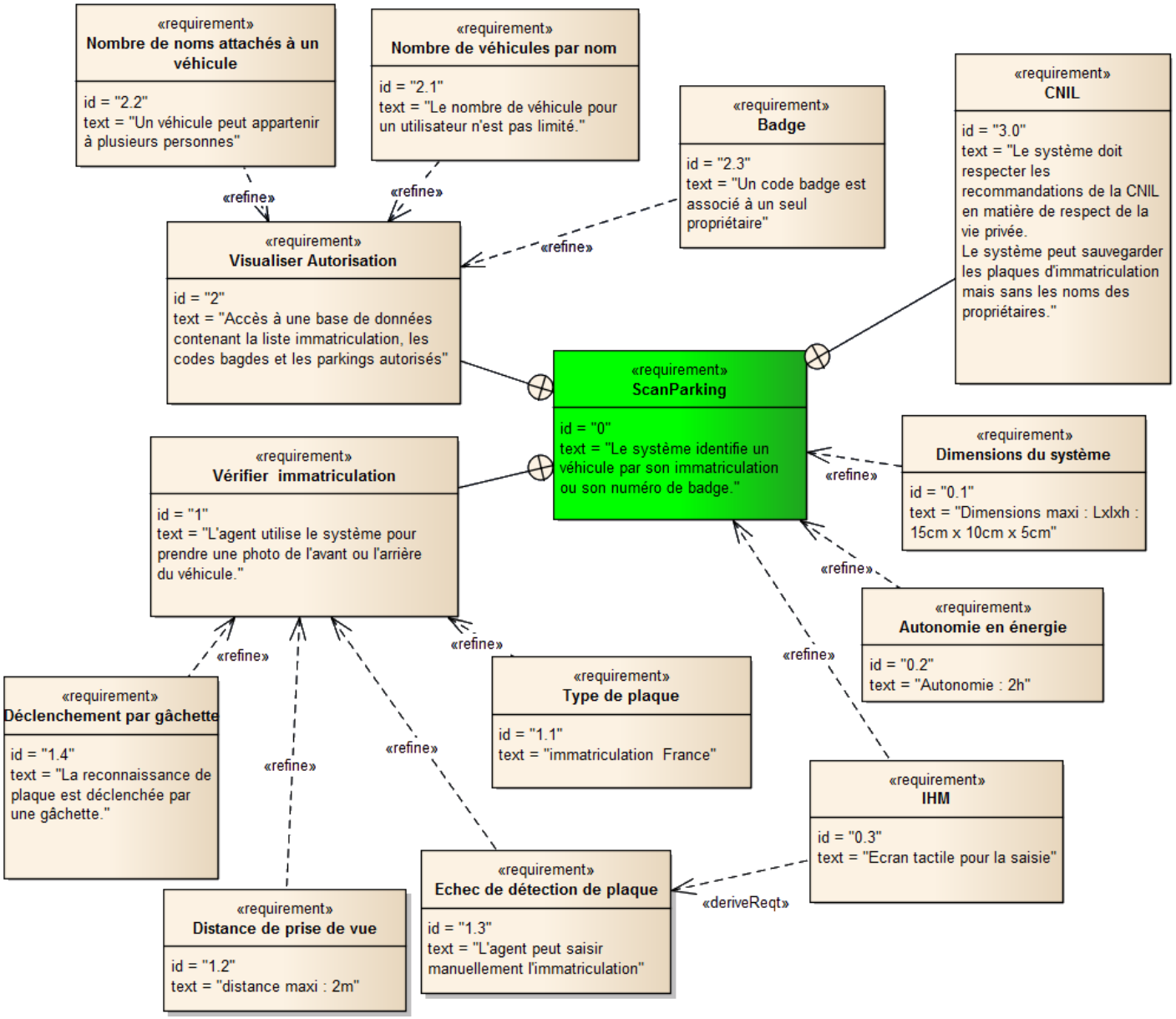




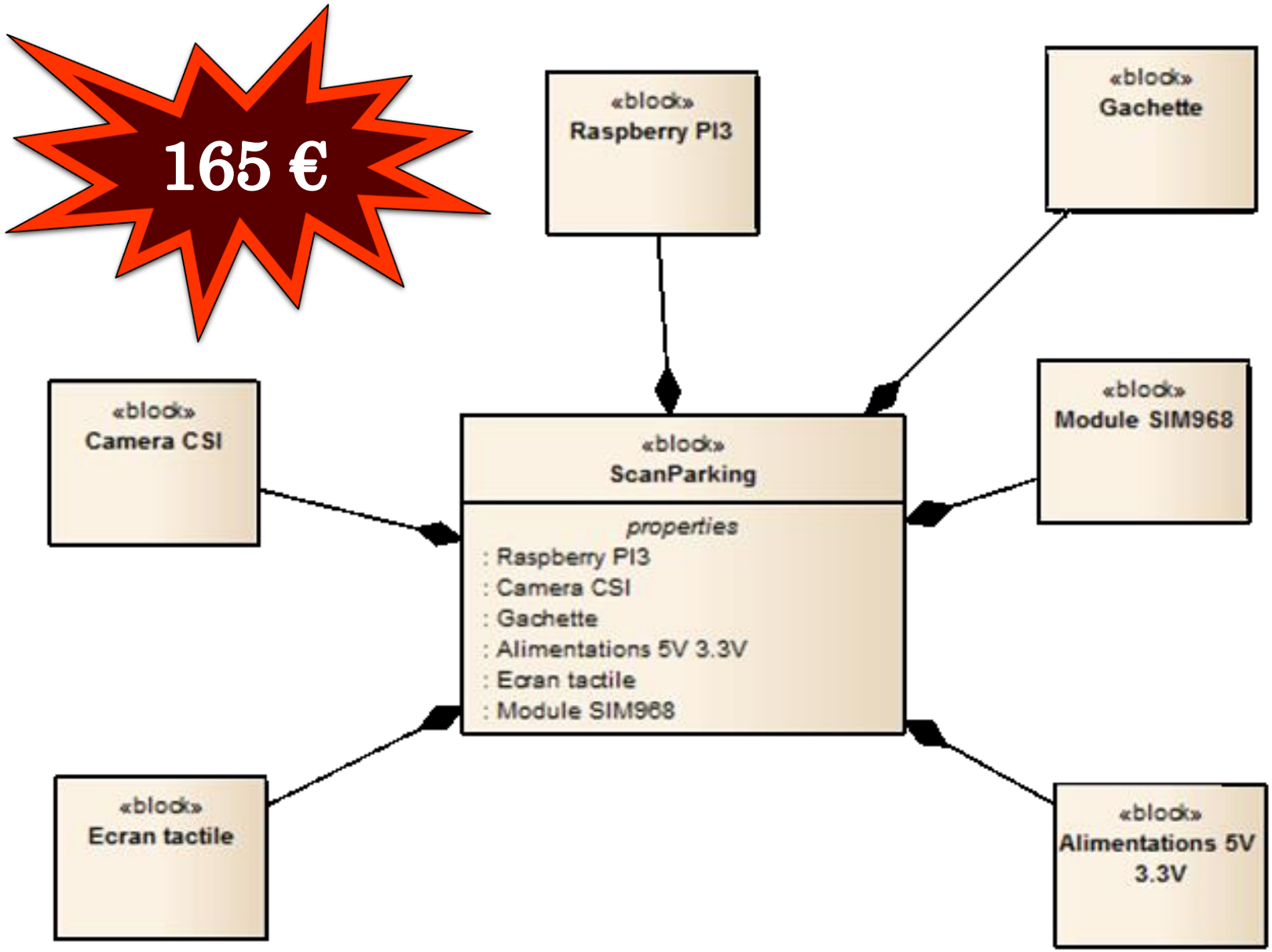
Cas d'utilisation



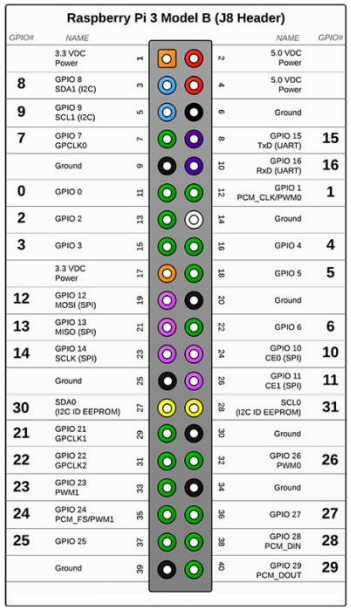
Exigences

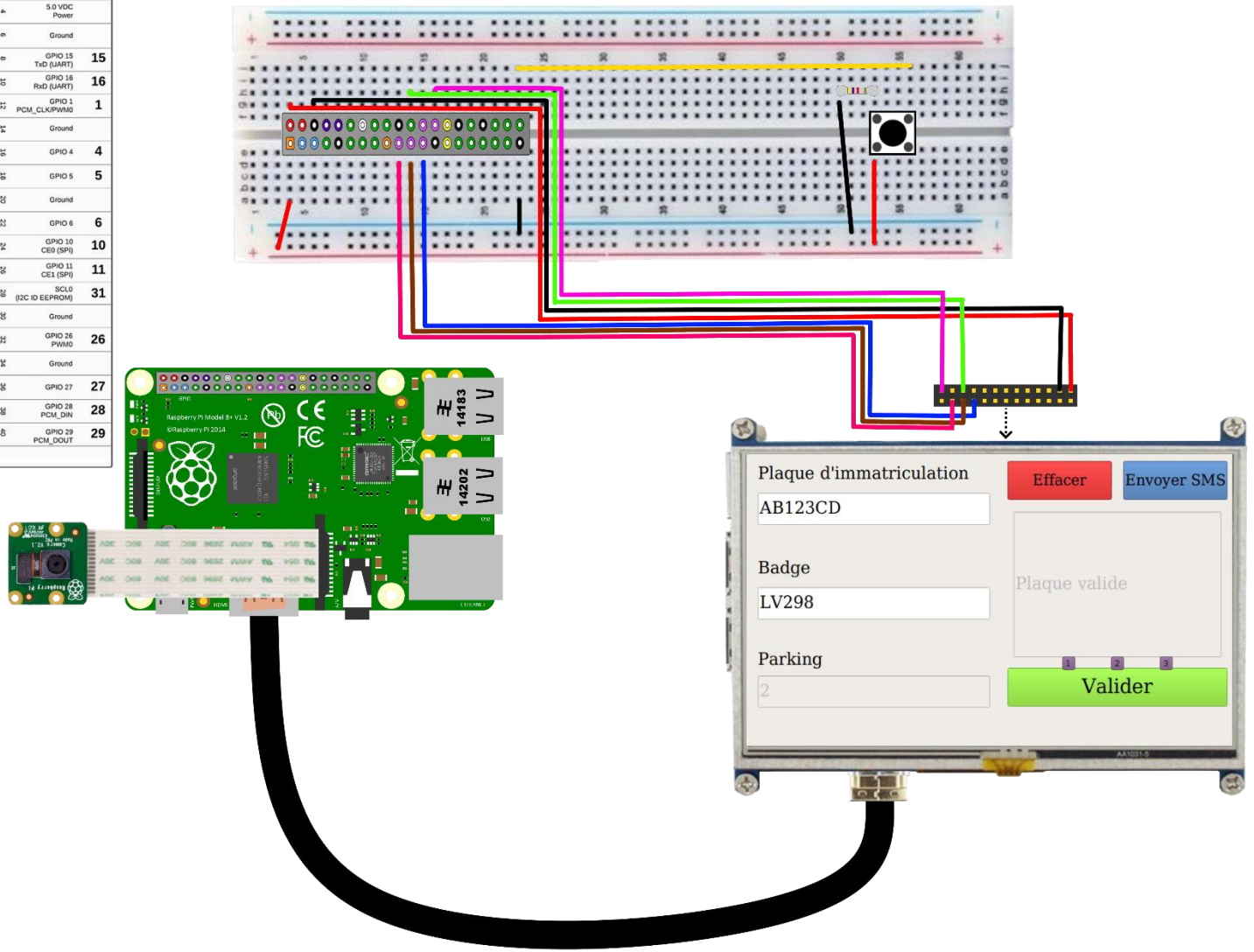

«requirement»
Nombre de noms attachés à un véhicule
id = "2.2"
text = "Un véhicule peut appartenir à plusieurs personnes"
«requirement»
Nombre de véhicules par nom
id = "2.1"
text = "Le nombre de véhicule pour un utilisateur n'est pas limité."
«requirement»
Badge
id = "2.3"
text = "Un code badge est associé à un seul propriétaire"
«requirement»
CNIL
id = "3.0"
text = "Le système doit respecter les recommandations de la CNIL en matière de respect de la vie privée.
Le système peut sauvegarder les plaques d'immatriculation mais sans les noms des propriétaires."
«refine»
«refine»
«refine»
«requirement»
Visualiser Autorisation
id = "2"
text = "Accès à une base de données contenant la liste immatriculation, les codes bagdes et les parkings autorisés"
«requirement»
ScanParking
id = "0"
text = "Le système identifie un véhicule par son immatriculation ou son numéro de badge."
«requirement»
Vérifier immatriculation
id = "1"
text = "L'agent utilise le système pour prendre une photo de l'avant ou l'arrière du véhicule."
«requirement»
Dimensions du système
id = "0.1"
text = "Dimensions maxi : Lxlxh : 15cm x 10cm x 5cm"
«refine»
«refine»
«requirement»
Autonomie en énergie
id = "0.2"
text = "Autonomie : 2h"
«refine»
«refine»
«refine»
«refine»
«refine»
«requirement»
Déclenchement par gâchette
id = "1.4"
text = "La reconnaissance de plaque est déclenchée par une gâchette."
«requirement»
Type de plaque
id = "1.1"
text = "immatriculation France"
«requirement»
IHM
id = "0.3"
text = "Ecran tactile pour la saisie"
«requirement»
Distance de prise de vue
id = "1.2"
text = "distance maxi : 2m"
«requirement»
Echec de détection de plaque
id = "1.3"
text = "L'agent peut saisir manuellement l'immatriculation"
«deriveReqt»

Architecture matérielle

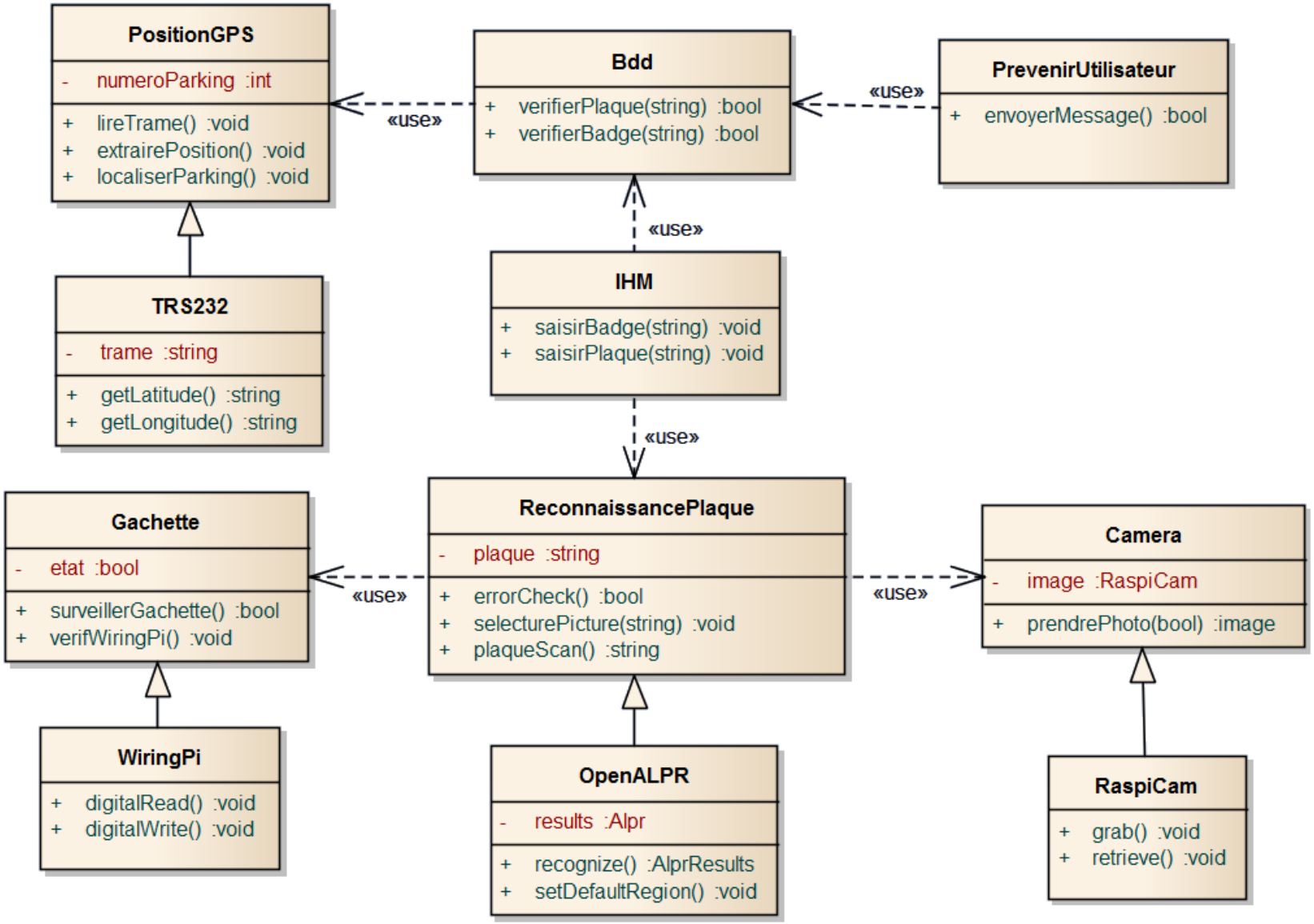


Architecture matérielle

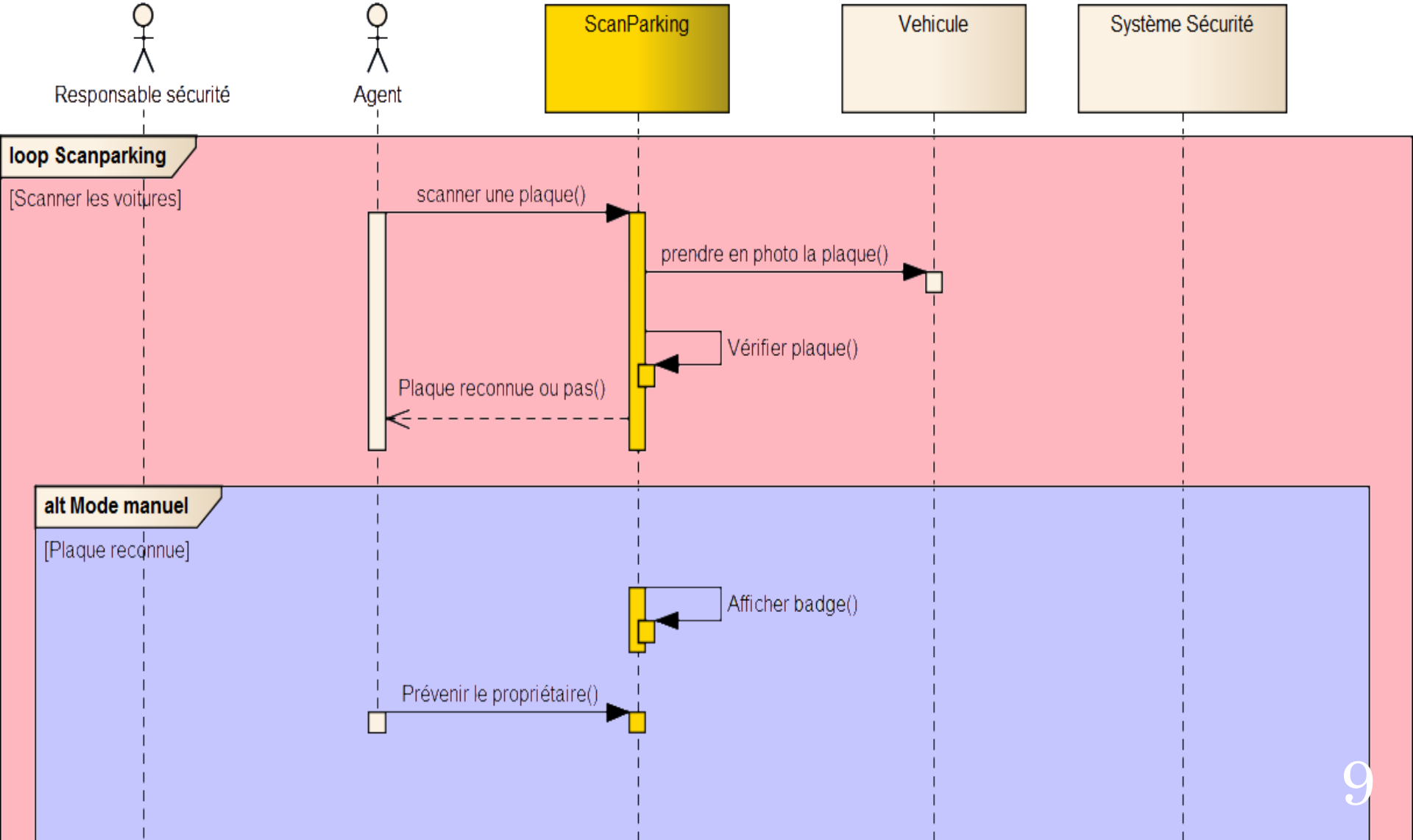


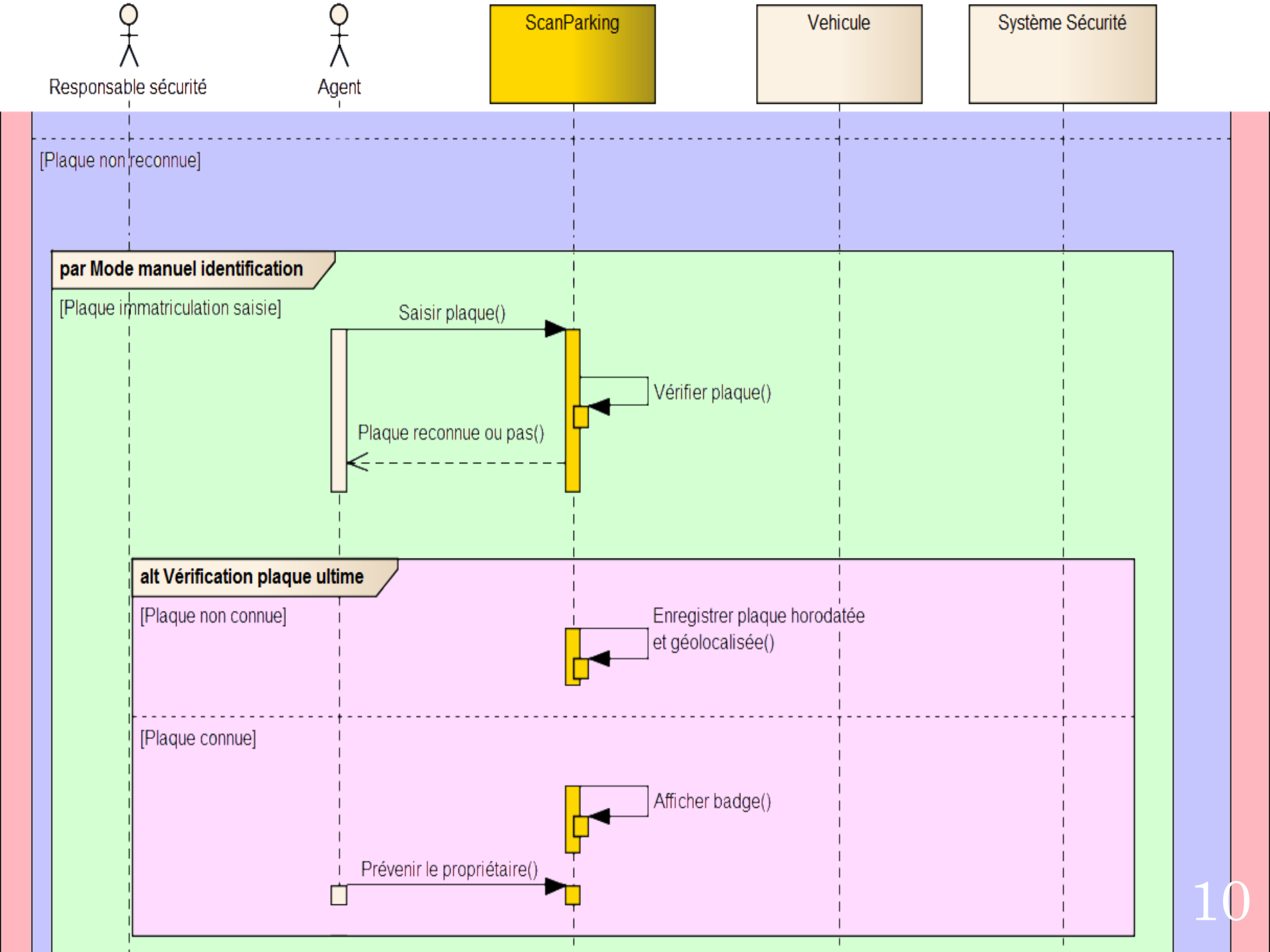


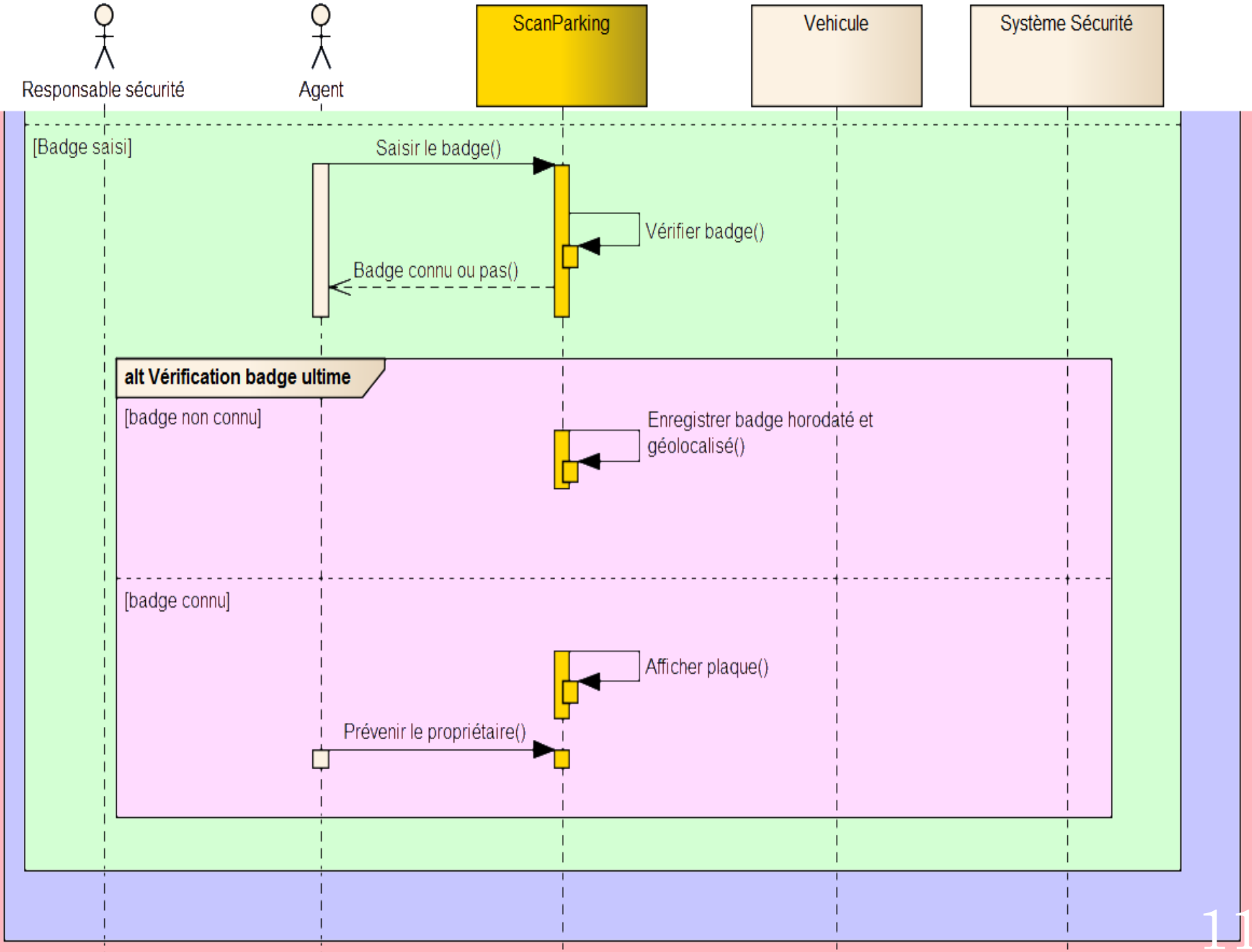
Architecture logicielle



Fonctionnement global




Responsable sécurité
Agent
ScanParking
Vehicule
Système Sécurité
[Plaque non reconnue]
par Mode manuel identification
[Plaque immatriculation saisie]
Saisir plaque()
Vérifier plaque()
Plaque reconnue ou pas()
alt Vérification plaque ultime
[Plaque non connue]
Enregistrer plaque horodatée
et géolocalisée()
[Plaque connue]
Afficher badge()
Prévenir le propriétaire()


Responsable sécurité
Agent
ScanParking
Vehicule
Système Sécurité
[Badge saisi]
Saisir le badge()
Vérifier badge()
Badge connu ou pas()
alt Vérification badge ultime
[badge non connu]
Enregistrer badge horodaté et géolocalisé()
[badge connu]
Afficher plaque()
Prévenir le propriétaire()

Partie personnelle

Partie 1 : Capture et analyse de plaque

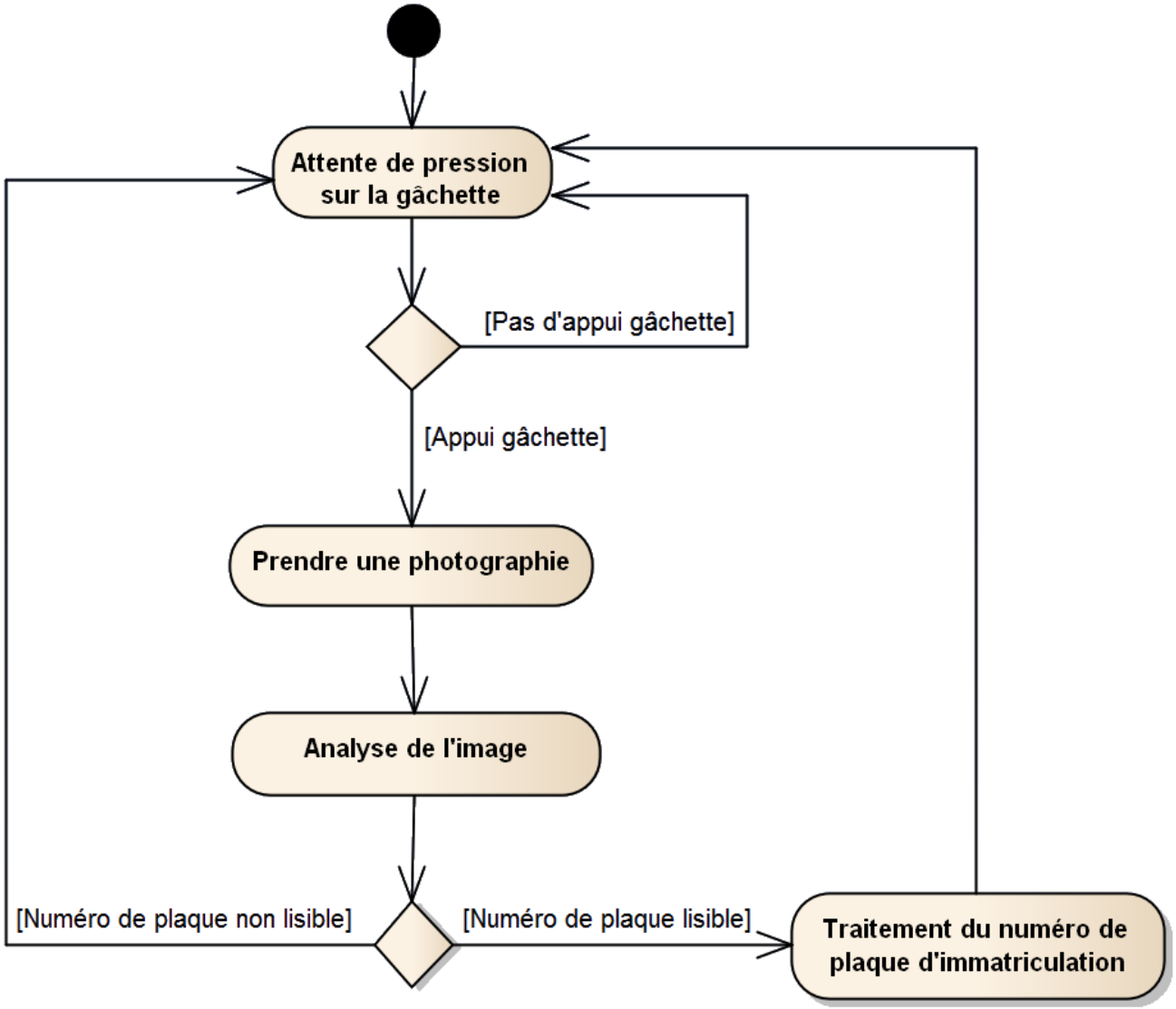
Partie 2 : IHM + Bases de données

Partie 3 : Géolocalisation + SMS

Partie 4 : Connectique et gestion du matériel

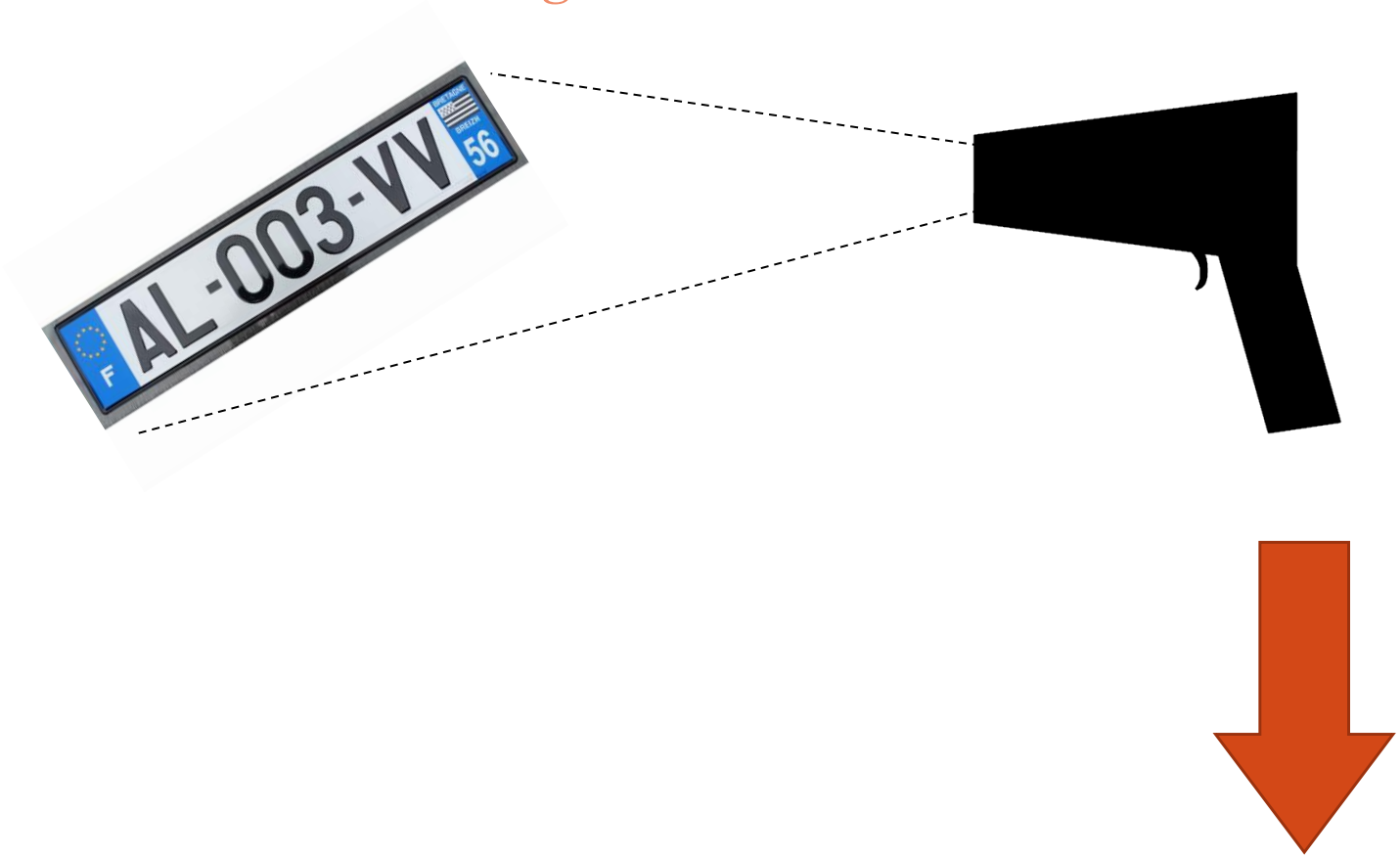
Partie personnelle

Fonctionnement global :



Partie personnelle

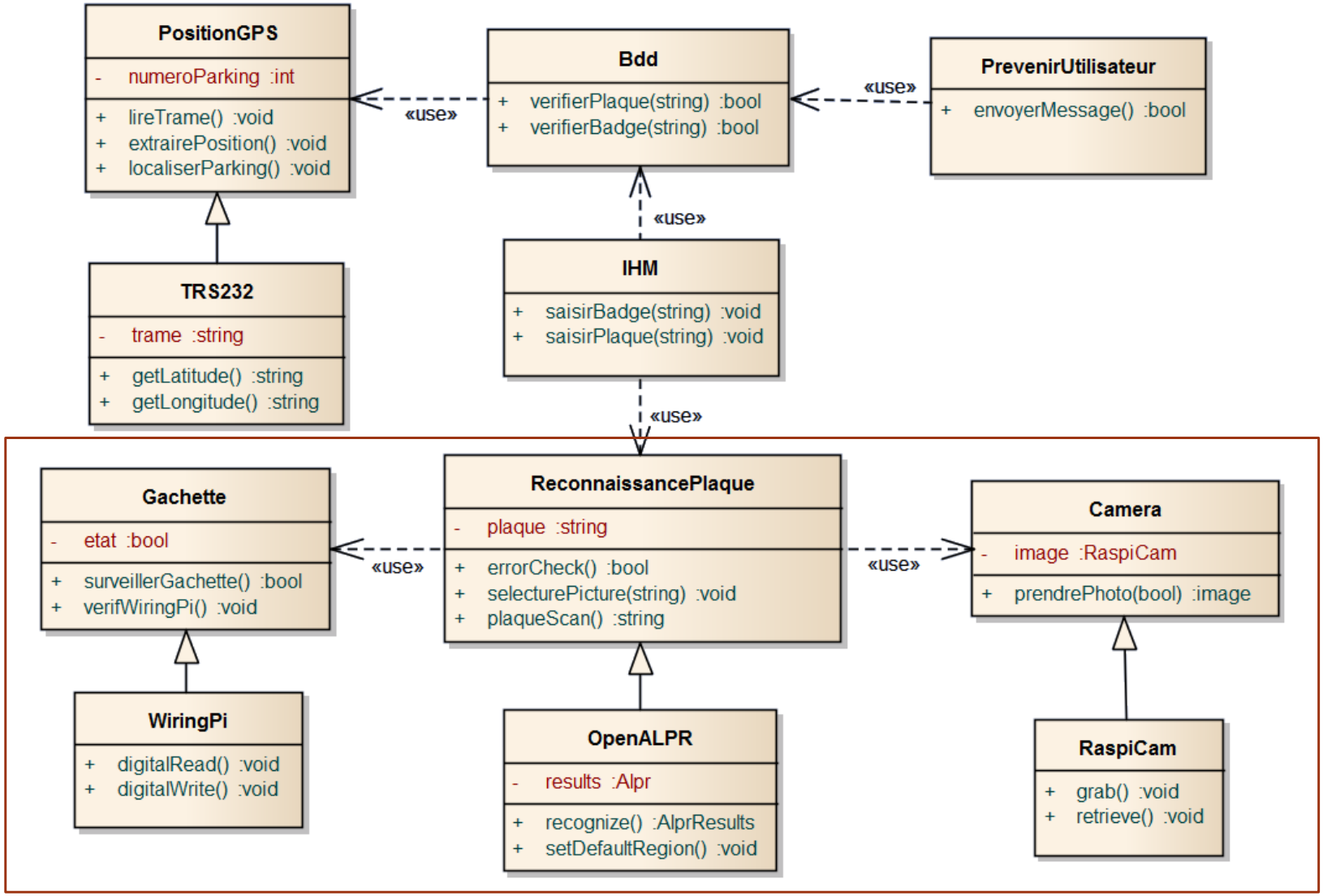
Fonctionnement global :




```
pi@raspberryGuillaume:~/ScanParking $ ./ScanParking
AL003VV
```

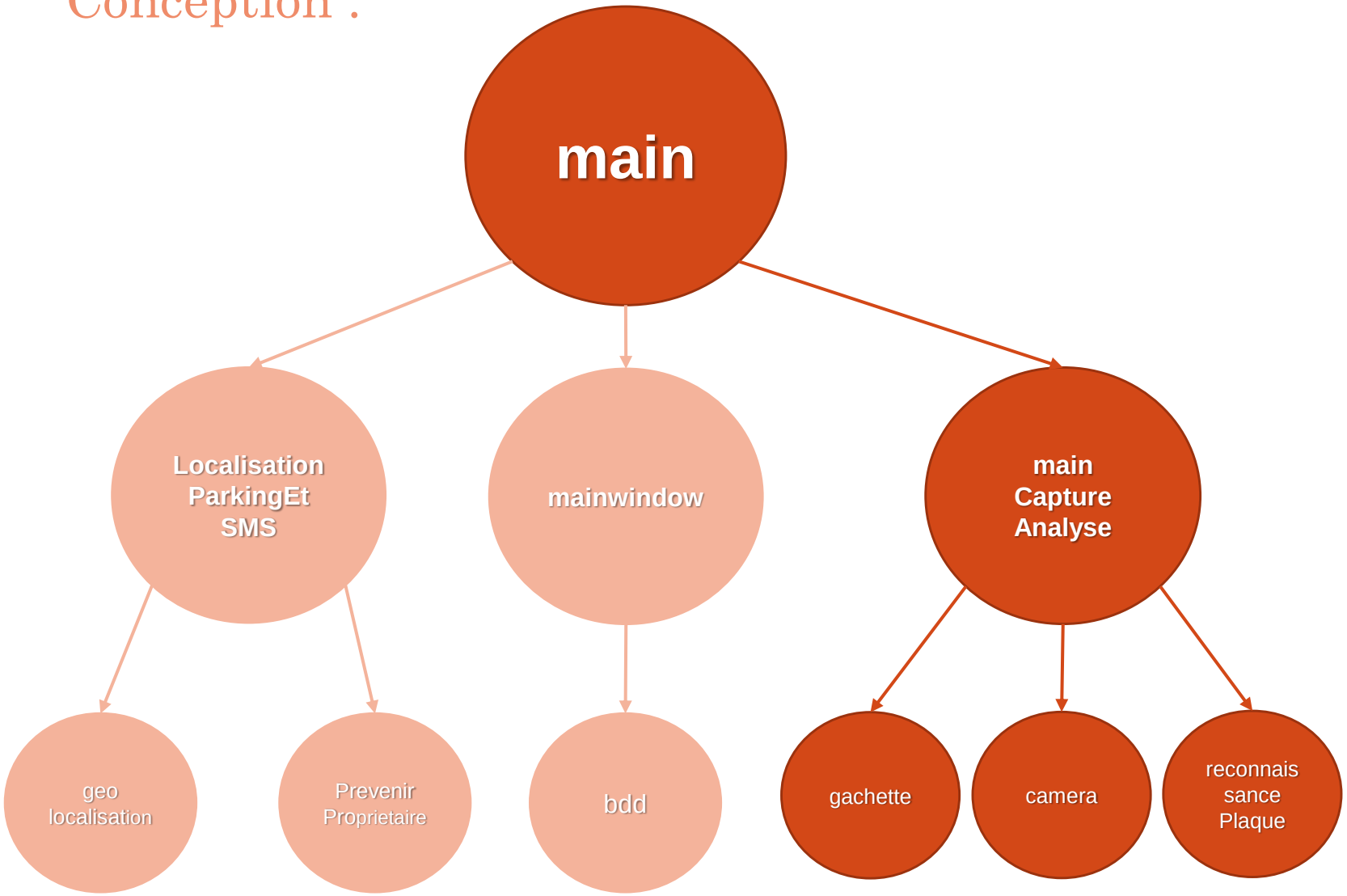
Partie personnelle

Architecture logicielle :



Partie personnelle

Conception :



Partie personnelle

Conception :

mainCaptureAnalyse.cpp :

```
void CMainScanParking::run()
{
    QString numPlaque;
    Gachette gachette;
    Camera camera;
    ReconnaissancePlaque reconnaissancePlaque;

    gachette.errorCheck();
    camera.errorCheck();
    reconnaissancePlaque.errorCheck();
    while(1)
    {

        gachette.surveillerGachette(); //Surveillance de la gachette est bloquante
        camera.prendrePhoto();
        numPlaque = QString::fromStdString(reconnaissancePlaque.scanPlaque("/home/pi/imagePlaque.jpg"));

        emit MonSignal(numPlaque); //Envoi du signal avec le numero de plaque
    }

    camera.release(); // Cette methode n'est jamais appelée dans cette configuration
}
```

Partie personnelle

Conception :

gachette.cpp :

```
Gachette::Gachette()
{
    wiringPiSetup();
    portGachette = 29; // configuration du port GPIO d'ecoute du signal gachette
    gachettePression = 0;
    pinMode(portGachette, INPUT);
}

void Gachette::surveillerGachette()
{
    while(1)
    {
        if (digitalRead(portGachette) == LOW && gachettePression == 0)
        {
            std::cout << "Pression gachette" << std::endl;
            gachettePression = 1;
            break;
        }

        if (digitalRead(portGachette) == HIGH)
        {
            gachettePression = 0;
        }
        delay(20);
    }
}
```

Partie personnelle

Conception :

camera.cpp :

```
Camera::Camera()
{
    //set camera parameters
    m_camera.set(CV_CAP_PROP_FORMAT, CV_8UC1); //Format of the Mat objects returned by retrieve()
    nbImage = 7;
}
bool Camera::errorCheck()
{
    //Check if camera is opened
    if (!m_camera.open())
    {
        std::cerr << "Error opening the camera" << std::endl;
        return 1;
    }
}
void Camera::prendrePhoto()
{
    std::cout << "Prise de la photo" << std::endl;
    for ( int i=0; i<nbImage; i++)
    {
        m_camera.grab();            // Grabs the next frame from capturing device
        m_camera.retrieve(image);   // Save the grabbed frame
    }
    cv::imwrite("/home/pi/imagePlaque.jpg",image);
}
```

Partie personnelle

Conception :

reconnaissancePlaque.cpp :

```
std::string ReconnaissancePlaque::scanPlaque(std::string image)
{
    std::cout << "Analyse de l'image" << std::endl;
    alpr::AlprResults results = openalpr.recognize(image); // Recognize an image file.

    if (results.plates.size() == 0)
    {
        return "Plaque non lisible";
    }

    alpr::AlprPlateResult plate = results.plates[0];
    alpr::AlprPlate candidate = plate.bestPlate;
    numeroPlaque = candidate.characters;

    for(unsigned int i = 0; i < numeroPlaque.size())
    {
        if (numeroPlaque[i] == 'O')numeroPlaque[i] = '0';
        if (numeroPlaque[i] == 'U')numeroPlaque[i] = 'V';
        if (numeroPlaque[i] == 'I')numeroPlaque[i] = '1';
        i++;
    }

    std::cout << "Numero de plaque : " << numeroPlaque << std::endl;

    return numeroPlaque;
}
```