Étude de cas	SIMULER, MESURER UN COMPORTEMENT	**Durée : 1 h 30**
	Support : SERRURE BIOMÉTRIQUE	

Nom de l'élève :	**Classe :**	**Date :**

Matériel ressource :
- **Maquette Serrure Biométrique.**
- **Cordon USB.**
- **Ordinateur équipé du logiciel BascomAVR.**
- **Chronomètre.**

Documents ressources :
- **Dossier technique de la serrure**

Compétences abordées :
- Simuler le comportement d'un système technique à partir de l'évolution d'un paramètre d'entrée ou de sortie.

Objectifs de l'activité

La partie commande de la chaîne d'information de la serrure biométrique est assurée par un composant nommé « micro contrôleur ». Il a été programmé par le constructeur de la serrure, afin de remplir la fonction principale, en respectant les contraintes imposées par le cahier des charges.
Dans cette activité, après avoir observé le comportement initial de la serrure, il vous sera demandé de modifier le programme initial.

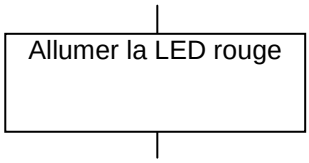
Comportement de la serrure en mode autonome.

Les algorigrammes

Pour décrire les différentes étapes du fonctionnement de la serrure, et sa logique de fonctionnement, on utilise un outil de présentation graphique appelé algorigramme. Son formalisme est le suivant :

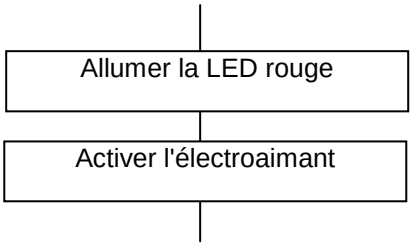
Pour réaliser une action élémentaire, un calcul ou tout autre traitement simple, il suffit de mettre un verbe d'action suivi d'un complément dans un rectangle. Ce rectangle possède une entrée, et une sortie.

Exemple :



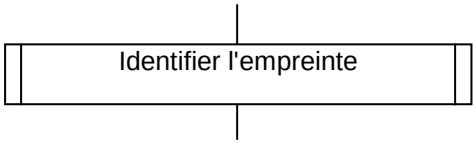
Pour décrire une suite d'actions, il suffit d'enchaîner les rectangles dans l'ordre dans lequel ils doivent être exécuté, on appel cela une ***Séquence*** :

exemple de séquence :



Pour décrire des actions plus complexes, c'est à dire faisant appel à plusieurs traitements élémentaires, il suffit de mettre un cadre double :

exemple : On utilise ici un traitement mathématique qui permet d'identifier l'empreinte, mais on ne donne pas le détail de comment cela est fait. Cela permet d'alléger la représentation de traitements plus complexes.

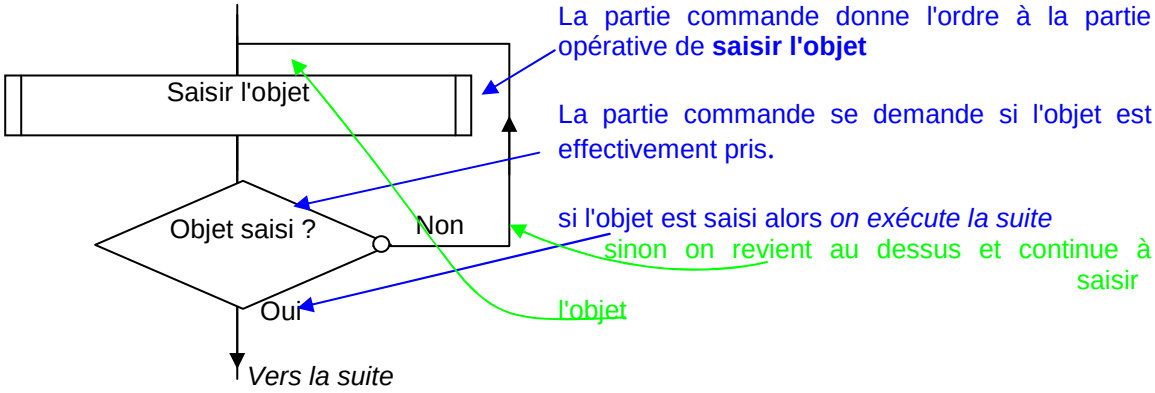


On pourra par la suite sur un autre document, donner un algorigramme séparé représentant ce que fait réellement la partie du programme qui permet d'obtenir le résultat du traitement « identifier l'empreinte ». Cette représentation correspond à ce que l'on appel un ***sous-programme***.

L'autre cas de figure très important dans les algorigrammes est le ***test.***

On le représente par un losange qui possède 1 entrée, et 2 sorties. On l'utilise chaque fois que le programme doit s'interroger sur le résultat d'un traitement.

Exemple : supposons que l'on ait sur un système automatisé un objet à saisir, et qu'il existe un capteur permettant de savoir si l'objet a effectivement été pris, on décrira par l'algorigramme suivant



Cela revient à formuler le fonctionnement comme ceci :

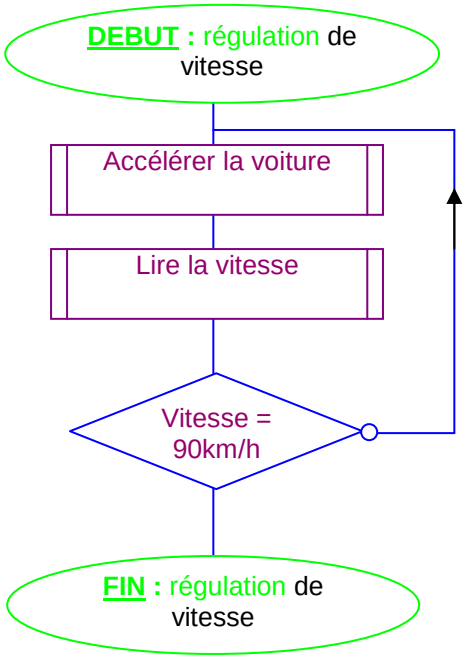
FAIRE : saisir l'objet

TANT QUE : l'objet n'est pas saisi.

La description de ce que doit faire le système en fonction des événements extérieurs, ce fait donc par l'intermédiaire de cet outil graphique. Il permettra par la suite, l'écriture d'un programme dans un langage donné, car tous les langages de programmation intègre ces structures de bases, à savoir

- la séquence
- le sous-programme (ou sous-traitement)
- le test

Exemple : réguler la vitesse de la voiture à 90 Km/h.



Le programme correspondant pourrait s'écrire :

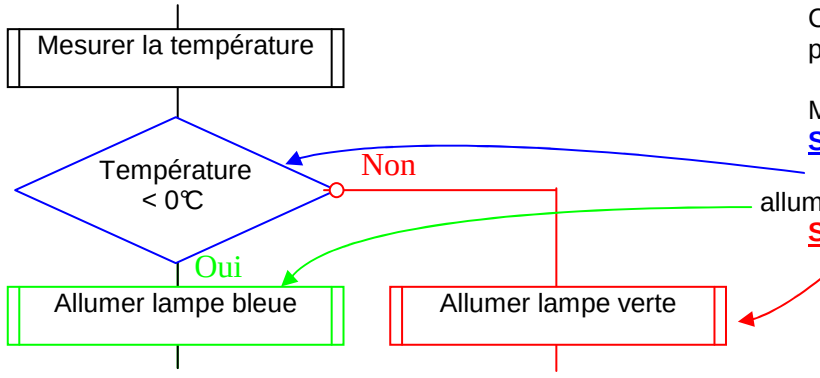
DEBUT: régulation de vitesse
FAIRE
    Accélérer_la_voiture( );
    Lire_la_vitesse ( );
TANT QUE vitesse ≠ 90 km/h
FIN : régulation de vitesse

Les mots soulignés en gras correspondent à des indicateurs de la structure logique du programme. Ils indiquent le début et la fin du programme, ainsi qu'une boucle : en effet, tant que la vitesse de consigne n'a pas été atteinte le programme exécute en boucle les instructions d'accélération et de lecture de la vitesse.

La structure qui apparaît en couleur bleue est présente dans la quasi-totalité des langages de programmation, sauf que la plupart d'entre eux utilise la langue anglaise ce qui donne : **DO (faire) WHILE (tant que).**

On peut également utiliser la même représentation, lors d'une *alternative*. Elle s'exprime en français par l'expression « SI une condition est remplie ALORS faire quelque-chose SINON faire autre chose ».

Exemple : Si la température est inférieure à 0°C, a llumer la lampe bleue, sinon allumer la lampe verte.



Ce qui donne en langage de programmation :

Mesurer_température( );
SI Température < 10
    ALORS
allumer_lampe(bleue);
SINON allumer_lampe(verte);

De même que précédemment, les langages utilisent cette structure en anglais IF (SI), THEN (ALORS), ELSE (SINON).

Application à la serrure biométrique.

Commencer par faire l'acquisition d'une empreinte valide en suivant la procédure décrite dans le dossier technique.

Première observation : Empreinte valide

La glissière étant fermée au départ : relever la glissière et présenter une empreinte valide.
Observer la réaction du système : relever notamment les durées d'allumage de l'éclairage du lecteur d'empreinte, le nombre de bip émis et leur longueur, le temps d'alimentation de l'électroaimant...

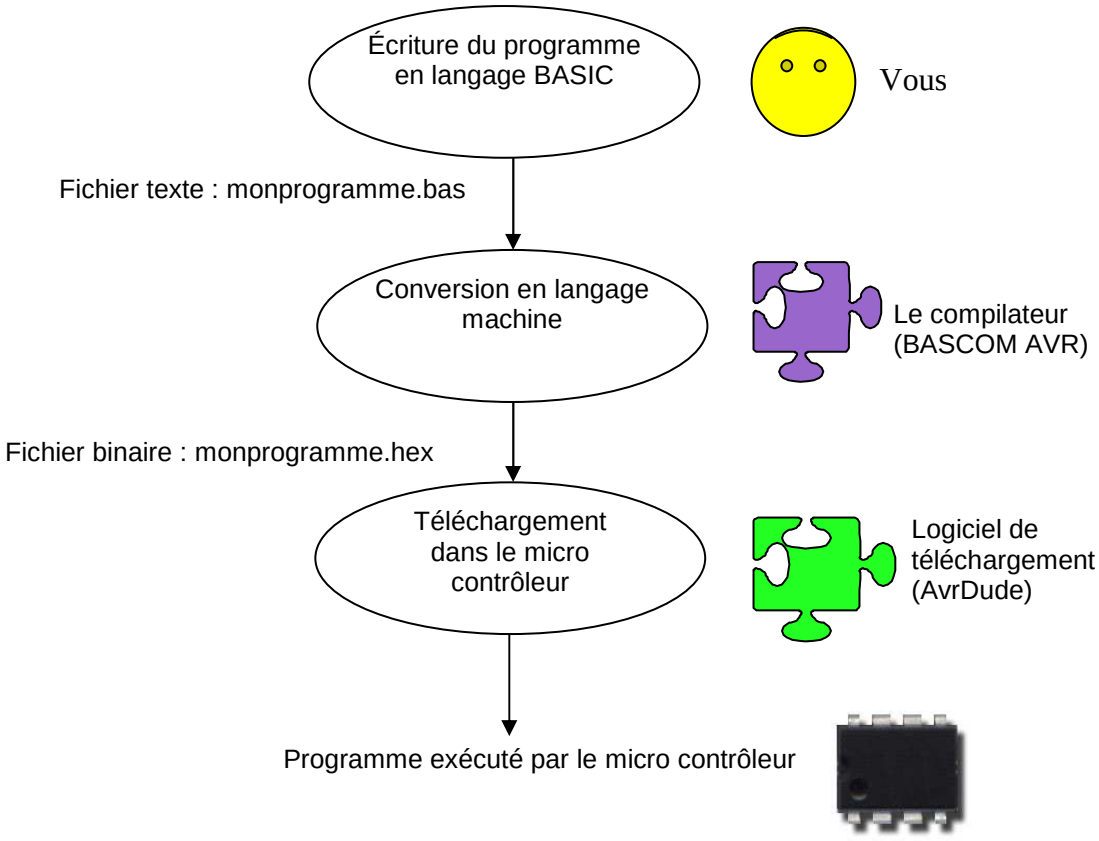
Deuxième observation : Empreinte non valide

La glissière étant fermée au départ : relever la glissière et présenter une empreinte non valide.
Observer la réaction du système.

En tenant compte de vos observations, Compléter l'organigramme en page 2 du document réponse

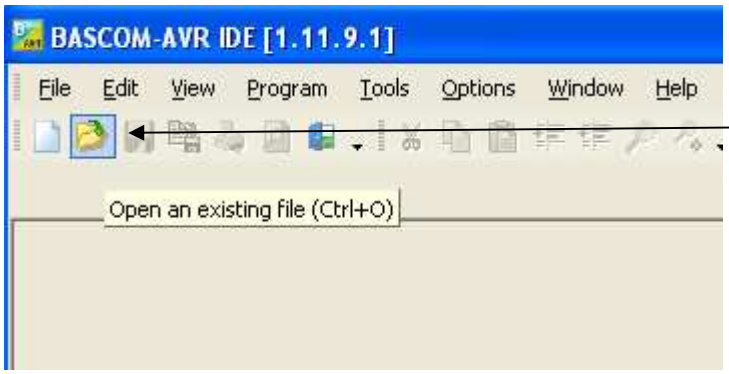
Mise en œuvre de la chaîne de développement du programme

Dans cette partie, vous allez découvrir et mettre en œuvre la chaîne de développement du logiciel de la serrure biométrique. A partir d'un programme écrit dans un langage de programmation (le BASIC), le fichier est converti en instructions compréhensibles par le micro contrôleur (un fichier binaire) : on appel cette phase ***la compilation***, elle est réalisée par un programme qui s'appelle *le compilateur.*
Ensuite le fichier binaire est transféré dans le micro contrôleur par un logiciel spécial ceci par l'intermédiaire d'un câble reliant la carte électronique à l'ordinateur.



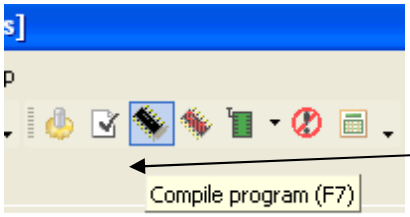
Copier **le répertoire programmes_serrure** dans votre répertoire de travail.

Lancer le logiciel BASCOM AVR 

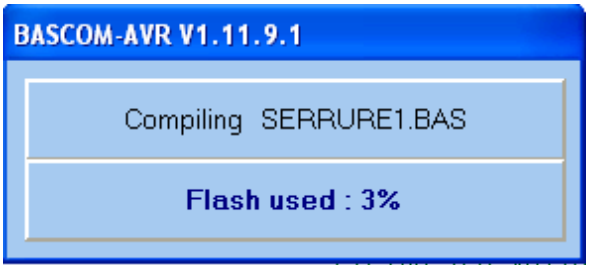


Faire « Ouvrir un fichier existant » pour aller chercher le fichier **serrure1.bas**

Le programme étant déjà écrit, on va demander la traduction en langage machine (compilation)



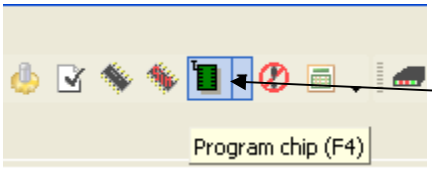
Cliquer sur l'icône de compilation  *ou la touche F7*



Une fenêtre s'ouvre vous indiquant que la compilation s'est bien passée, et le taux d'occupation de votre programme dans la mémoire

La dernière phase est le transfert du programme dans la mémoire du micro contrôleur.

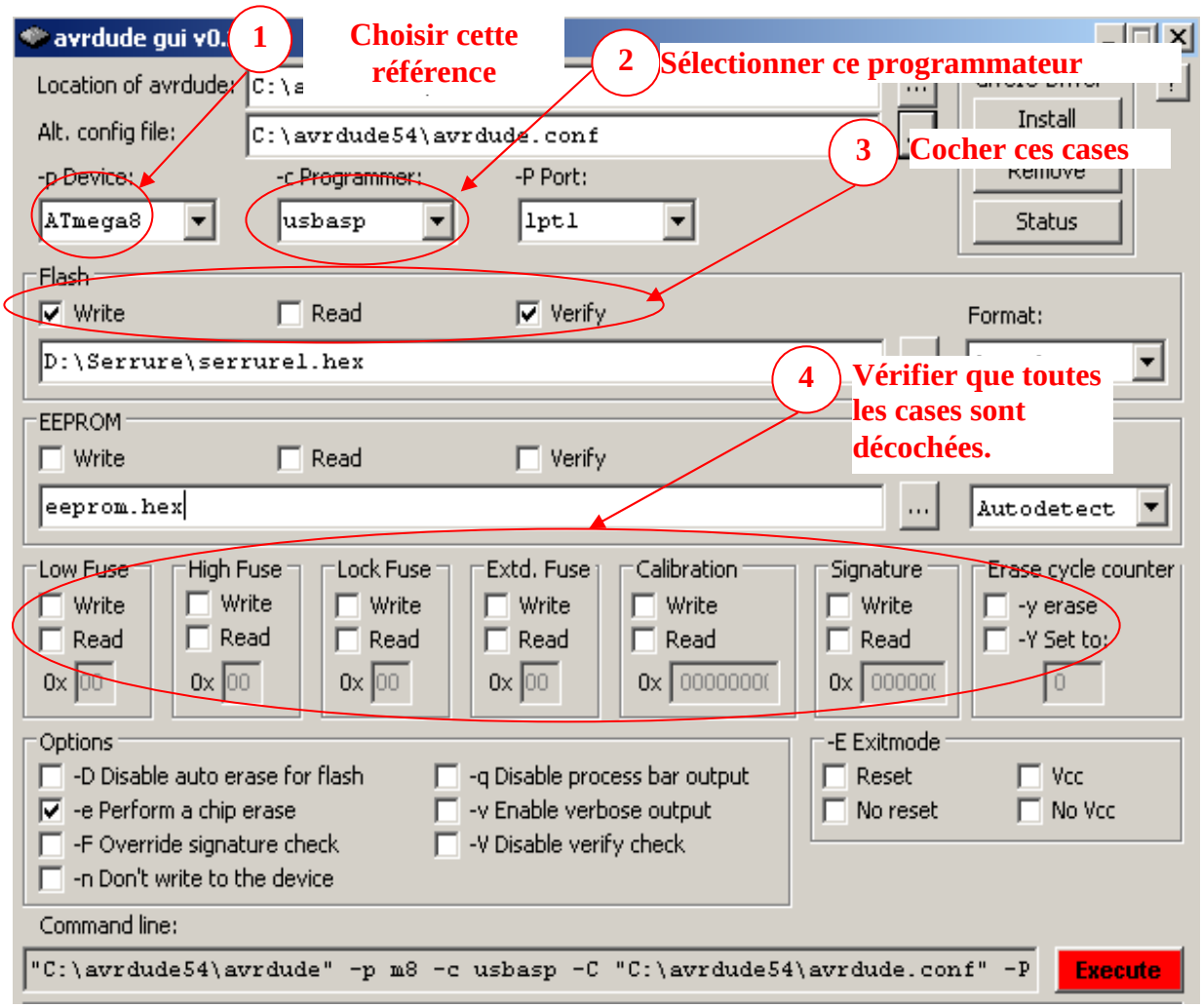
- Sur la maquette de la serrure biométrique, basculer le commutateur en position SIMULATION.
- Connecter le câble USB entre l'ordinateur et la maquette.



Cliquer sur l'icône de programmation  *ou la touche F4*

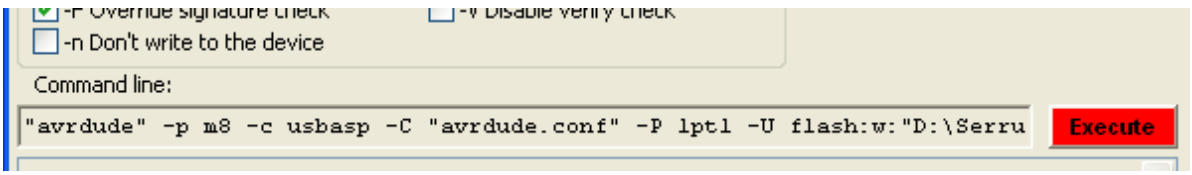
Une nouvelle fenêtre s'ouvre : c'est le logiciel de téléchargement (Avrdude).

Avant de télécharger le programme, il faut vérifier certains points de configuration du logiciel, en suivant les instructions ci-dessous :



5 Charger votre programme le fichier serrure1.hex en appuyant sur ce bouton

Appuyer sur **Execute** pour lancer le téléchargement.



Un texte s'affiche dans le bas de la fenêtre vous indiquant le bon téléchargement.
Observer le fonctionnement ainsi obtenu.

Mise au point d'un programme.

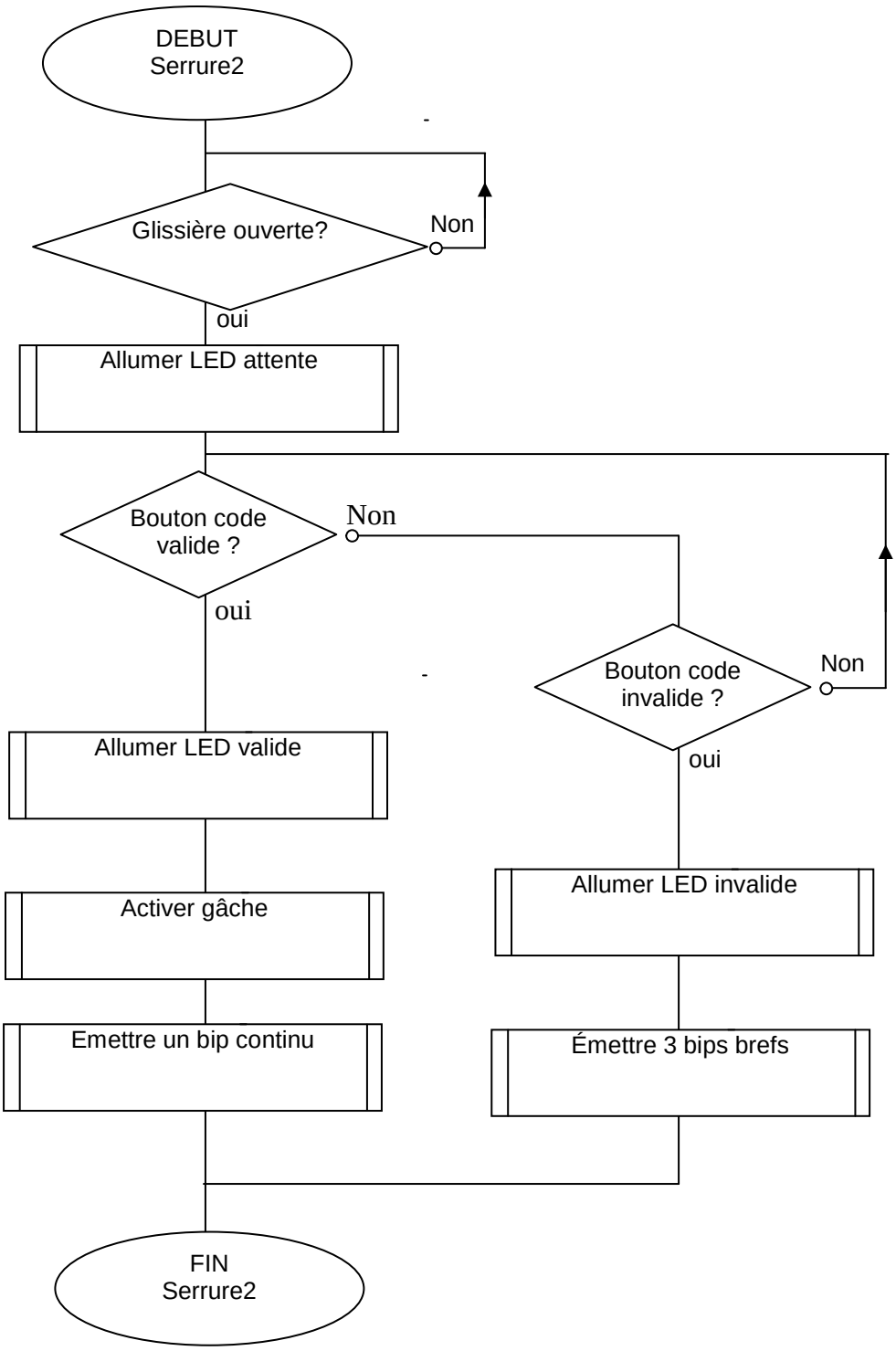
Dans cette partie, vous devrez à partir d'un algorigramme donné, compléter un programme, le transférer dans le micro contrôleur, et vérifier son fonctionnement sur le système réel.

L'algorigramme correspond au fonctionnement réel mais avec une simulation de la prise d'empreinte qui est un processus trop complexe à votre niveau.

Un appui sur le bouton rouge simule une mauvaise empreinte.
Un appui sur le bouton vert simule une bonne empreinte.

La LED orange est la LED d'attente, elle est allumée en attente d'une identification d'empreinte.
La LED verte correspond à une prise d'empreinte valide.
La LED rouge correspond à une prise d'empreinte invalide.

L'algorigramme correspondant au fonctionnement voulu est le suivant :



Codage du programme

Lancer le logiciel bascom AVR BASCOM-AVR

Ouvrir le fichier serrure2_eleve.bas en suivant la même procédure que précédemment. Le début du programme comprend les instructions de configuration du matériel, vous n'avez pas à les modifier.

L'activation d'un élément se fait par la commande **Set**, ainsi on écrira :

Instruction à écrire	Effet sur la maquette
Set Led_valide	Allume la led d'empreinte valide (verte)
Set Led_invalide	Allume la led d'empreinte invalide (rouge)
Set gache	Active la gâche

La désactivation d'un élément se fait par la commande **Reset**, ainsi on écrira :

Instruction à écrire	Effet sur la maquette
Reset Led_valide	Éteint la led d'empreinte valide (verte)
Reset Led_invalide	Éteint la led d'empreinte invalide (rouge)
Reset gache	désactive la gâche

L'émission d'un bip se fait par l'intermédiaire de la fonction **bip(nombre,type)** où

- **nombre** est le nombre entier de bip à émettre.
- **type** peut prendre la valeur bref, ou continu.

On écrira donc :

Instruction à écrire	Effet sur la maquette
Call bip(3,bref)	Émet 3 bips courts
Call bip(1,continu)	Émet 1 bip long

Compléter les instructions manquantes en respectant parfaitement la syntaxe des instructions indiquée dans les tableaux ci-dessus.

Compiler, et transférer votre programme dans la maquette en procédant comme précédemment.

Vérifier le fonctionnement réel.

Si vous constatez des erreurs, corriger le programme, et relancer un processus compilation / téléchargement après chaque modification.

Compléter la page 1 du document réponse avec reportant les instructions saisies dans le programme.