Dossier technique : parking intelligent

1. Découverte du produit et de la problématique technique

Afin de permettre aux automobilistes souhaitant se garer à proximité d'une gare, un parking souterrain sur 3 niveaux a été réalisé en 2007. Chaque niveau comporte 132 places dont 10 pour les personnes à mobilité réduite (PMR).

Le système de paiement par bornes nécessite d'être modernisé. Il est parfois difficile de se rappeler où se trouve sa voiture. C'est pourquoi le gestionnaire du parking a décidé de mettre en place une nouvelle solution permettant aux automobilistes de réserver le parking à distance via une application WEB. À l'issue de cette réservation, un code unique à quatre chiffres est fourni au propriétaire de la voiture qui en arrivant sur la borne peut saisir ce code afin d'ouvrir la barrière. En complément l'automobiliste peut connaître son emplacement dans le parking (niveau et numéro de place) une fois le code saisi en recevant cette information sur son smartphone (exemple : niveau -2, place 25).

Ce parking propose une autre particularité non étudiée ici, de limiter la consommation énergétique de la borne et de l'affichage avec un système de détection à distance de l'arrivée d'un véhicule permettant de « réveiller » le système.

Description fonctionnelle du système simplifié du parking intelligent

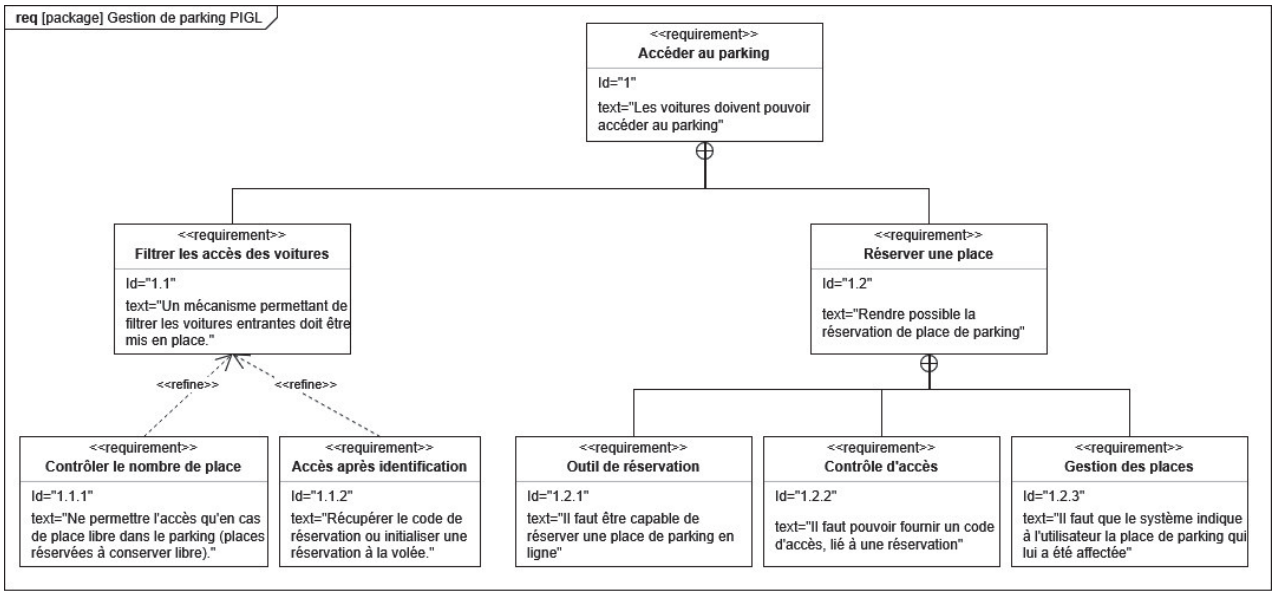


Figure 1 : Diagramme d'exigence du parking

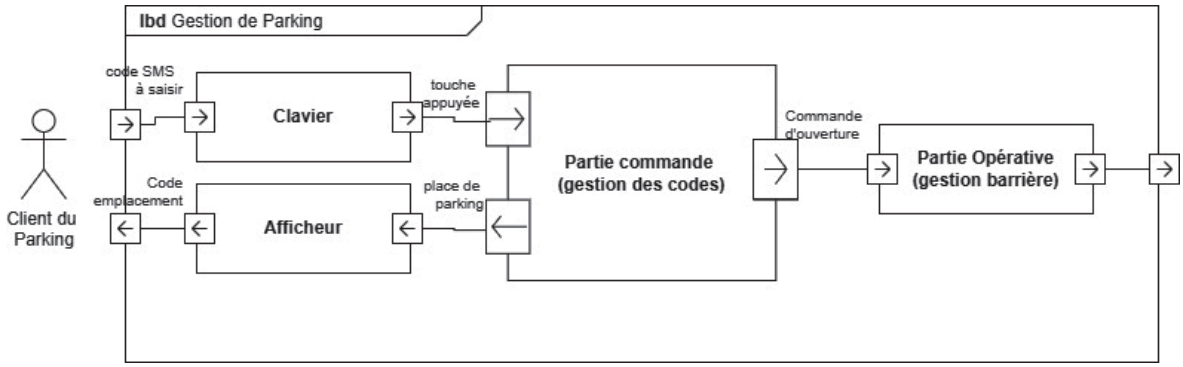


Figure 2 : Diagramme des blocs internes du parking

Caractéristiques du parking

- Dimensionnement du parking : 3 niveaux de 132 places.
- Acquisition du code par un clavier matricé.
- Affichage des instructions sur un afficheur LCD.

Cahier des charges

- Affichage du code et des instructions sur 2 lignes de 16 caractères.
- Le code doit être à 4 chiffres.
- Clavier souple 12 touches.

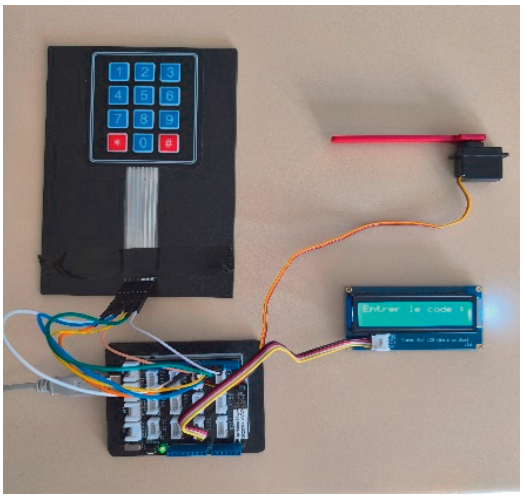


Schéma structurel existant

- Le système de gestion du code pour l'ouverture et la fermeture de la barrière est géré par un clavier matricé et une carte Arduino.
- L'affichage des instructions se fait par un afficheur LCD 16x2.
- La barrière est ouverte grâce un moteur à courant continu. Un servomoteur représente l'ensemble {moteur + barrière} dans l'application.

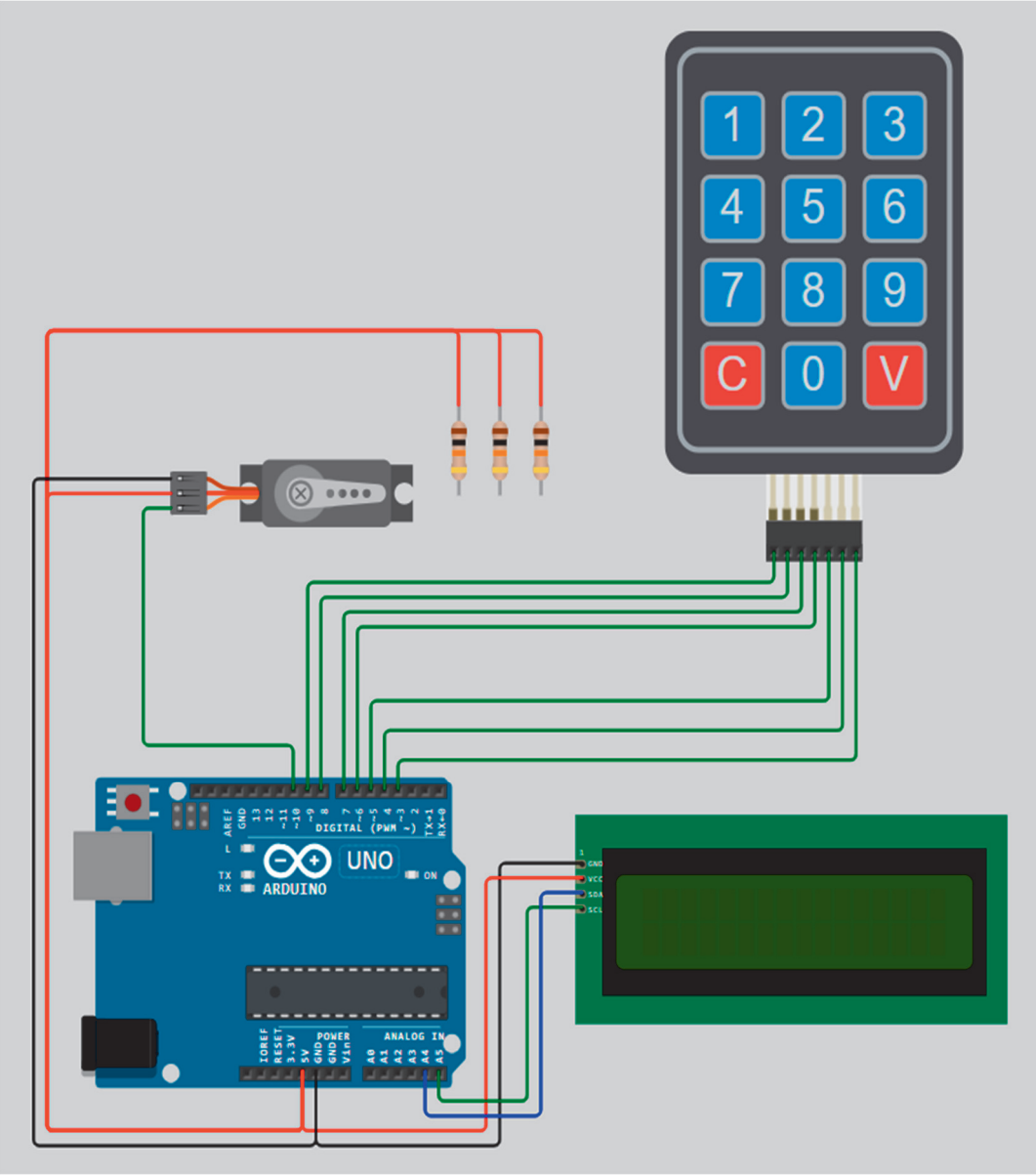


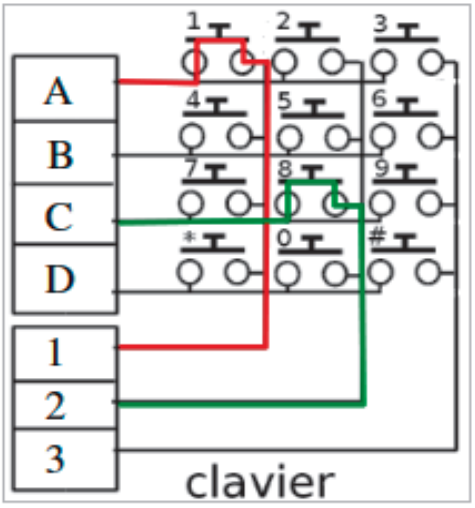
Figure 3 : Schéma de câblage à compléter sur Wokwi

2. Conception

Principe du clavier matricé

Tableau de correspondance ligne/colonne :

	1	2	3
A	1	2	3
B	4	5	6
C	7	8	9
D	*	0	#

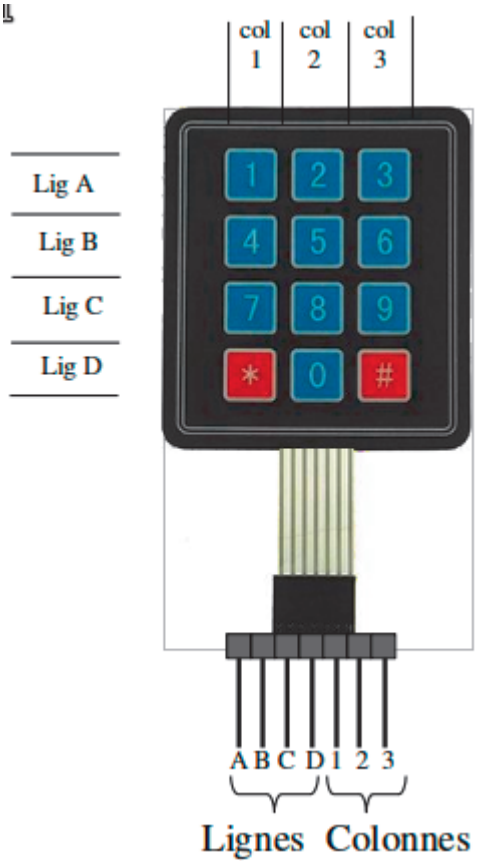


Exemple :

- Si on appuie sur la touche " 1 ", on va relier " A " à " 1 "
- Si on appuie sur la touche " 8 ", on va relier " C " à " 2 "

Pour un clavier 12 touches, on a besoin de 7 broches :

- 3 broches pour les 3 colonnes
- 4 broches pour les 4 lignes



À chaque fois que l'on appuie sur une touche, un contact à l'intérieur du clavier vient relier les broches correspondant à la colonne et la ligne de la touche appuyée.

Le principe pour détecter qu'une touche est appuyée est le suivant :

1- Appliquer un niveau logique 0 sur la broche d'une des 4 lignes. Les 3 autres lignes sont positionnées au niveau logique 1.

2- Regarder sur chacune des broches correspondant aux colonnes si on retrouve ce niveau logique 0. Si on retrouve ce niveau logique 0 sur une colonne, cela signifie que la touche située à l'intersection de cette colonne et de la ligne au niveau logique 0, est enfoncée.

3- On passe à la ligne suivante et on recommence.

4- On réalise ces opérations suffisamment rapidement pour détecter même de brefs appuis sur le clavier.

Connexions sur simulateur

Pour gérer/programmer le clavier sous Arduino, on utilise ici le logiciel en ligne WOKWI à l'adresse : https://wokwi.com/projects/new/arduino-uno.

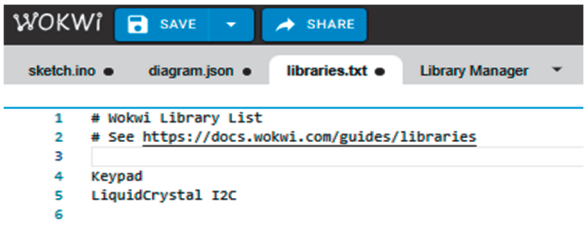
La mise en place de l'environnement de simulation se fait en 4 étapes :

1. Le système est initialement démarré avec 3 onglets (`sketch.ino`, `diagram.json`, `Library Manager`).

2. Insérer le schéma de démarrage donné par le fichier `diagram.json` en réalisant un copier-coller à partir d'un éditeur de texte (Notepad++, bloc-note, nano, etc.). Cela permettra d'obtenir le diagramme présenté en page 2 (pour le premier programme).
   - Pour ajouter une connexion (un câble), cliquer sur une broche à connecter, puis cliquer sur la broche cible (la broche à connecter à l'aide du câble).
   - Pour ajouter un composant, appuyer sur la touche « a » du clavier puis sélectionner le composant voulu.

3. Pour ajouter les différents programmes à compléter dans la partie « conception », sur le même principe que pour les diagrammes, il suffit de copier le contenu du fichier programme dans l'onglet `sketch.ino`.

4. Il reste enfin à ajouter les librairies requises par le système. Pour cela, dans l'onglet `Library Manager`, en utilisant le bouton + visible, ajouter les librairies suivantes :
   - Keypad
   - LiquidCrystal_I2C



Dès lors, un nouvel onglet sera visible (`librairies.txt`), récapitulant les bibliothèques utilisées.

3. Simulation

Les objets créés

`Reservation {`
`  String code;`
`  int etage;`
`  int place;`
`}`
	L'objet `Reservation` contient trois paramètres : le code de réservation, l'étage de la place réservée, ainsi que le numéro de la place réservée. Pour accéder aux différents éléments :
	• `objet.code`
	• `objet.etage`
	• `objet.place`

Les fonctions

`AfficherPlace(Reservation data)`	Affiche la place affectée à la réservation fournie en paramètre.
`AfficherCodeErrone()`	Affiche un message précisant que le code fourni est faux.
`getCode()`	Récupère le code saisi par l'utilisateur. Cette fonction est à coder par vos soins.
`controlCode(String code)`	Vérifie la validité du code et enclenche l'affichage de la place le cas échéant. Cette fonction est à coder par vos soins.

Structure de la fonction `getCode()`

La structure de la fonction `getCode()` est la suivante.
Une boucle attend la validation de l'utilisateur (appui de la touche V) pour renvoyer le code.
Dans cette boucle, il est attendu l'appui d'une touche, puis, en fonction de la touche appuyée, les actions suivantes sont exécutées :

1. Touche `C` : retirer le dernier caractère du code (corriger)
2. Touche `V` : renvoie le code saisi par l'utilisateur (valider)
3. Autre touche (0 à 9) : ajout du chiffre choisi au code à 4 chiffres

Les variables disponibles

`NUM_ROWS`	Nombre de lignes sur le clavier
`NUM_COLS`	Nombre de colonnes sur le clavier
`RESERVATION_COUNT`	Nombre de réservations enregistrées dans le système
`RESERVATION_DATA`	Un tableau contenant l'ensemble des réservations
`keys`	Un tableau à 2 dimensions contenant les caractères du clavier
`rowsPin`	Broche de connexion des lignes du clavier matricé
`colPins`	Broche de connexion des colonnes du clavier matricé
`kp`	L'objet représentant le clavier
`lcd`	L'objet représentant l'écran LCD

Les bibliothèques disponibles

L'objet String

`String message = String("Coucou")`	Créer un objet String contenant le message coucou
`message.length()`	Renvoie la longueur de la String (6 dans ce cas)
`message.concat("un truc")`	Ajoute "un truc" à la fin du message existant
`message.remove(index)`	Retire le caractère n°index du message

La librairie Keypad

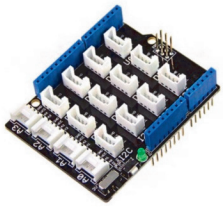
`Keypad kp = Keypad(`
`  makeKeymap(keys),`
`  rowPins, colPins,`
`  NUM_ROWS, NUM_COLS);`
	Initialise une variable (`kp`) représentant le clavier matricé. Il prend en paramètre la liste des touches, les pins des lignes et des colonnes, le nombre de lignes et de colonnes.

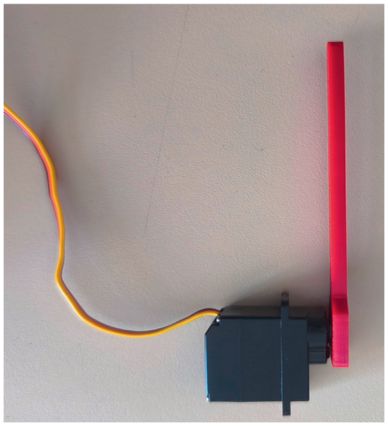
`kp.getKey()`	Récupère la touche appuyée. Renvoie `NO_KEY` en l'absence de touche appuyée.

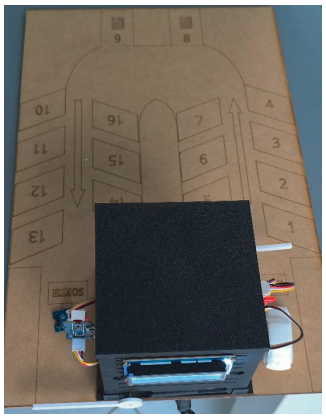
4. Expérimentation

Pour effectuer l'expérimentation, le matériel suivant est à disposition :

- Une carte de développement Arduino ;
- Un shield de connexion "Base" ;
- Un clavier souple 12 touches ;
- Un écran à cristaux liquides ;
- Un servo-moteur pour valider visuellement l'ouverture du parking (optionnel) ;

 Carte Arduino Uno R3	 Shield base Grove	 Clavier souple	 Afficheur LCD 2x16 Grove





Servo moteur et barrière pour parking ou maquette parking