Gestion informatisée d'un parking

Cette maquette de parking de 10 places est pilotée par un programme écrit en B4R (Visual Basic pour cartes ARDUINO) et enregistré sur la carte ARDUINO UNO à laquelle la maquette est connectée.

Tant qu'un nouveau programme n'est pas sauvegardé sur cette carte, elle garde le programme « GestionParking3.B4R » en mémoire.

Le logiciel est conçu pour un parking de 10 places qui sont disponibles au démarrage du programme ce qui signifie que le parking est vide et qu'il y a 10 places de libres.

Sur la carte ARDUINO UNO, il y a un bouton « RESET ». Si un appui est réalisé sur ce bouton, le programme est réinitialisé au départ (10 places libres).

Première ébauche du projet de maquette *(Les 10 LEDs vertes ont ensuite été remplacées par un écran LCD)*

Au fur et à mesure des entrées et sorties de véhicules, le programme calcule le nombre de places disponibles. Si les 10 places sont occupées, le parking est complet et les deux feux rouges extérieurs s'allumeront. Dans ce cas, si une pression a lieu sur le bouton d'entrée situé à gauche du portail, la barrière du parking ne s'ouvrira pas et les feux resteront rouges.

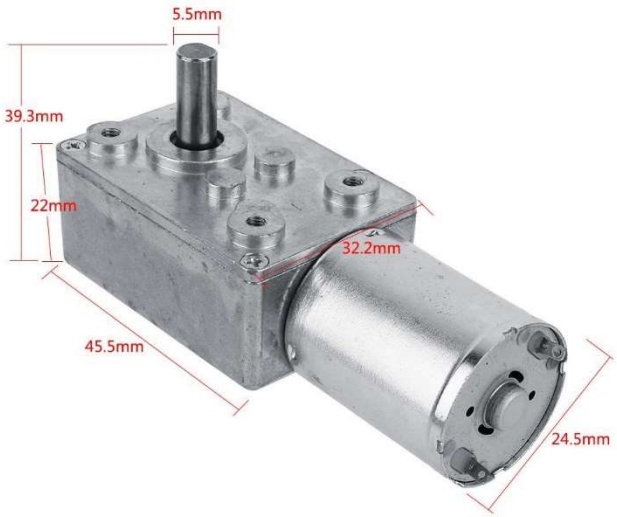
En revanche, si un véhicule sort du parking en libérant une place, les feux rouges s'éteindront et il sera possible d'ouvrir à nouveau la barrière pour l'entrée d'un véhicule.

S'il ne reste qu'une seule place libre sur le parking, les feux passeront à l'orange. De deux à dix places, les feux seront verts.

Lorsque le bouton d'entrée est pressé, un buzzer sonne pendant quelques secondes, la barrière s'ouvre s'il reste une ou plusieurs places. Un écran LCD placé au-dessus du portail d'entrée affiche en temps réel les disponibilités, c'est-à-dire le nombre de places encore libres sur le parking.

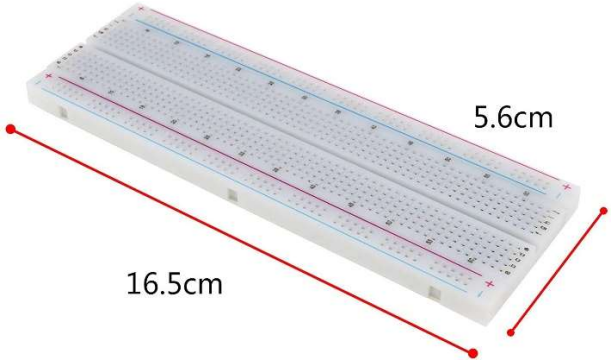
Le bouton de sortie est situé à l'intérieur du parking à gauche du portail de sortie. Ce bouton ne déclenchera l'ouverture de la barrière que s'il reste des véhicules sur le parking. Si le parking est vide, la barrière ne s'ouvrira pas.

Matériel utilisé



- 1 moteur FTVOGUE 12 V avec réducteur (engrenages métalliques)
- Deux feux tricolores précâblés avec des petites résistances intégrées
- 2 gros boutons-poussoirs encastrés dans le socle
- 1 buzzer (Brève sonnerie à l'appui sur le bouton « Entrée»)
- 1 circuit intégré L293D (Il permet l'inversion du sens de rotation du moteur)
- 1 écran LCD de 2 lignes de 16 caractères
- 1 interface I2C permettant de connecter facilement l'écran LCD à la carte Arduino Uno

- 1 demi-plaque de connexions sans soudures dite « breadboard »



Plaque de connexions sans soudures entière

(Cette plaque sert à connecter correctement le circuit intégré L293D)

- Fils avec broches mâles pour les connexions diverses
- Un serre-tringle pour garde boue vélo (s'adapte exactement sur l'axe de sortie du moteur)
- Une tige de bambou non fissurée de 12 cm de longueur qui sera vissée sur le filetage
- 1 carte ARDUINO UNO:

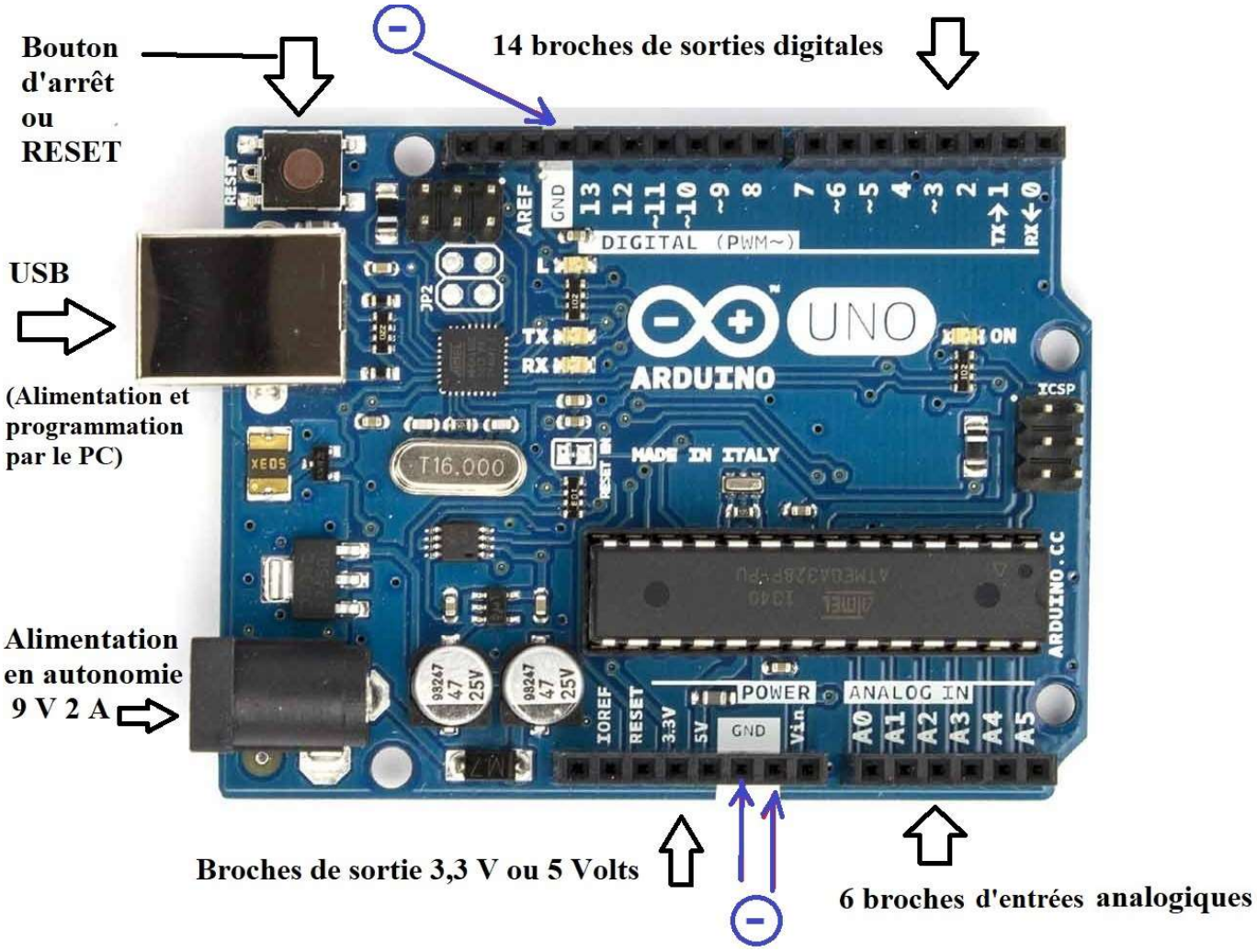
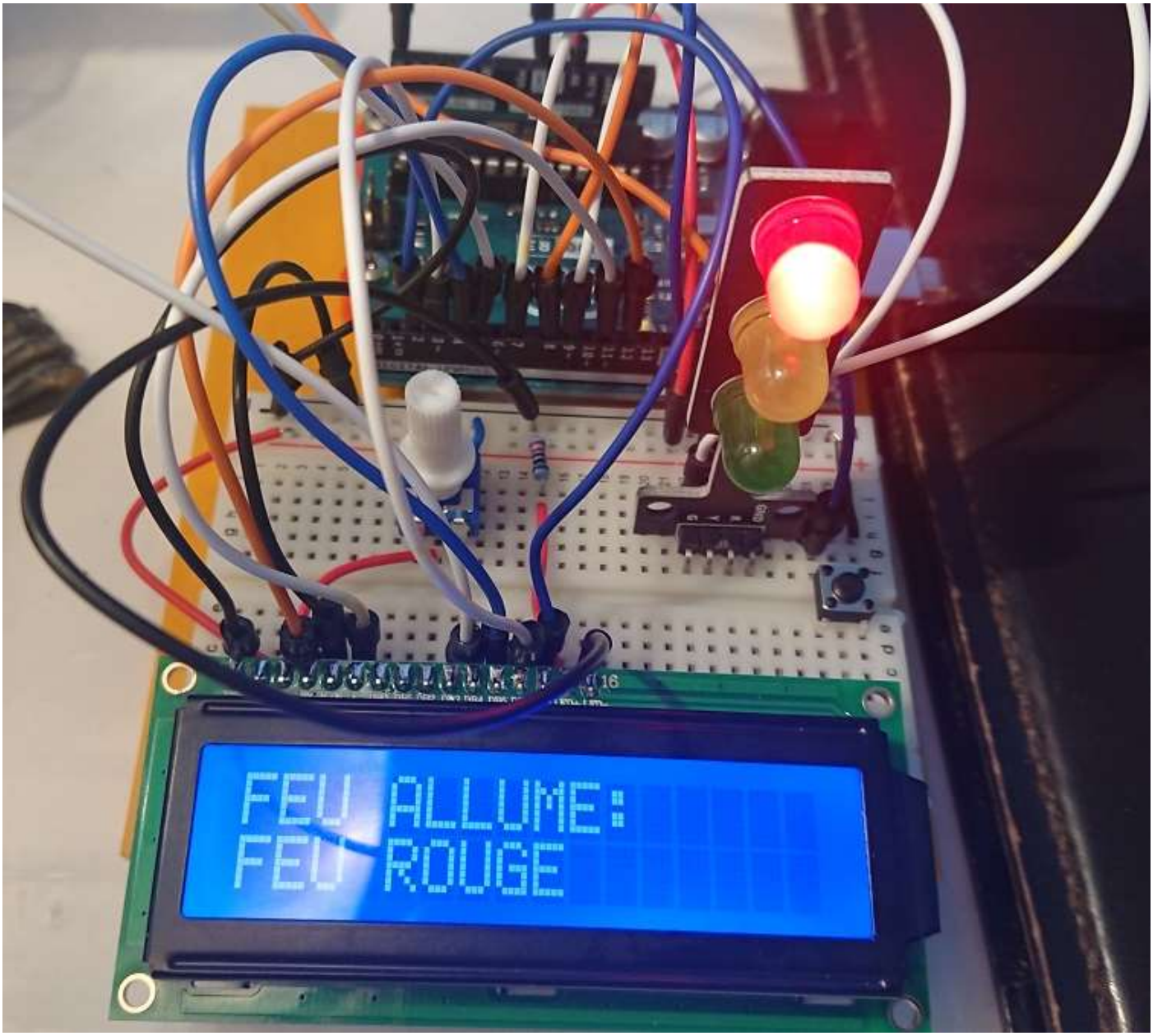


Schéma d'une carte ARDUINO UNO REV 3 avec ses broches de connexion



Ecran LCD utilisé sans la petite interface I2C

Comme on peut le voir, il est possible de se passer de l'interface I2C mais sur les 16 broches de connexion de l'écran LCD, douze doivent être raccordées, ce qui pose des problèmes techniques en situation réelle sur la maquette.

Par ailleurs, ce système de connexions nécessite l'utilisation de 6 broches digitales de la carte Arduino Uno et le recours à un potentiomètre nécessaire pour régler le contraste de l'affichage lumineux de l'écran LCD

Page suivante, un schéma « Fritzing » permet de visualiser la complexité de toutes ces connexions (sans la présence de l'interface I2C)..

Nous avons donc choisi d'utiliser la petite interface I2C qui nous permettra de connecter l'écran LCD en utilisant seulement 2 bornes analogiques A4 et A5 de la carte Arduino Uno.

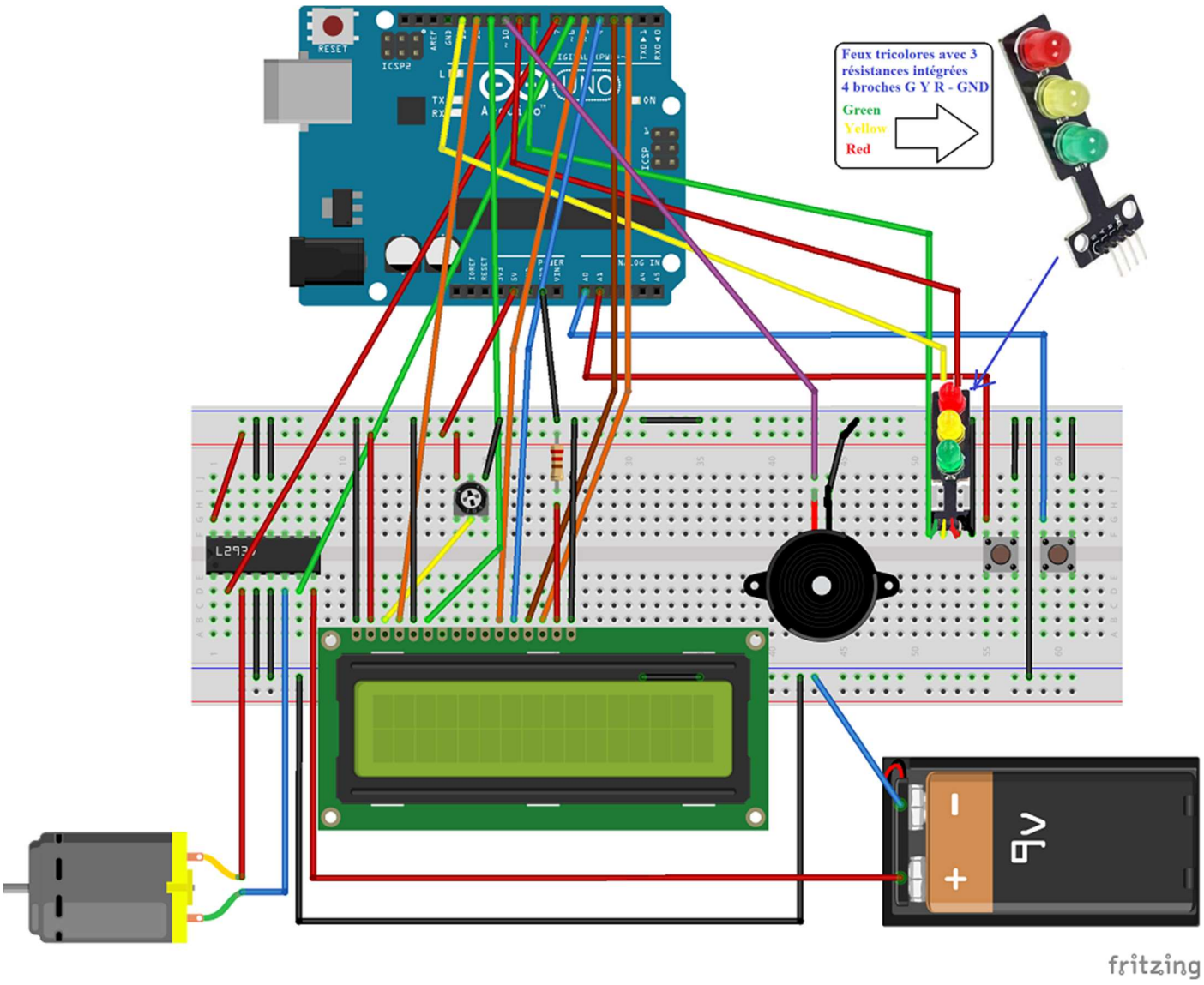
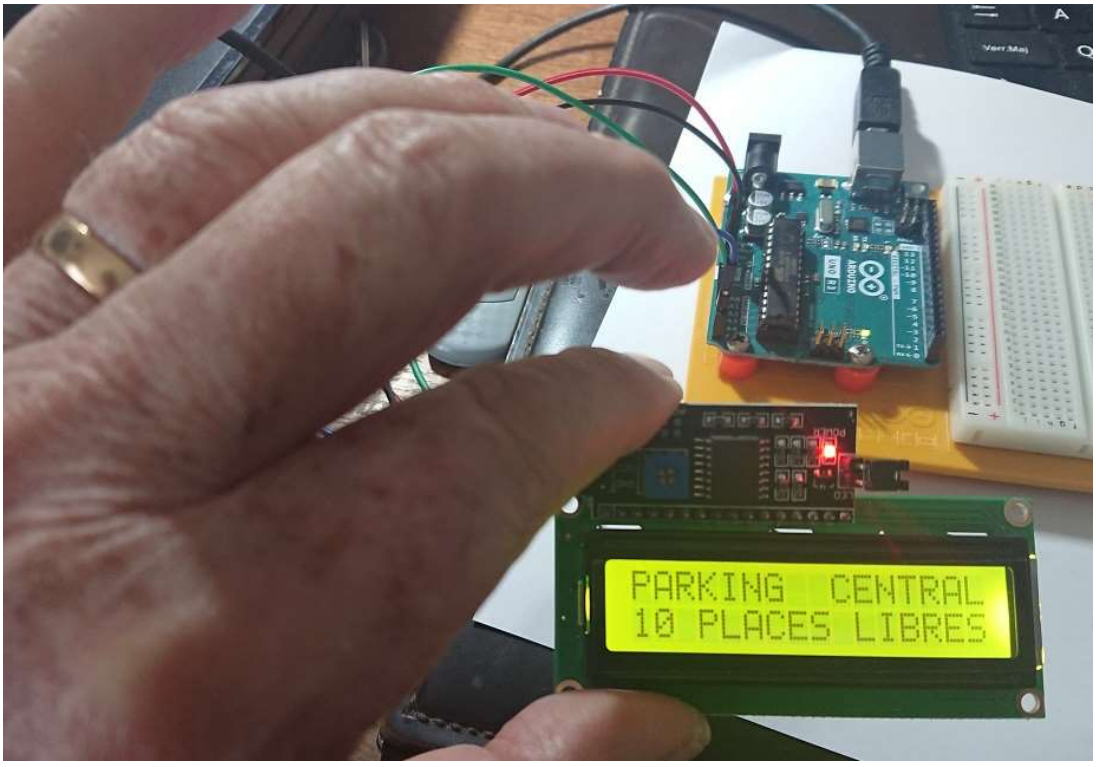


Schéma de connexions d'un écran LCD utilisé sans la petite interface I2C



Ecran LCD utilisé avec son interface I2C

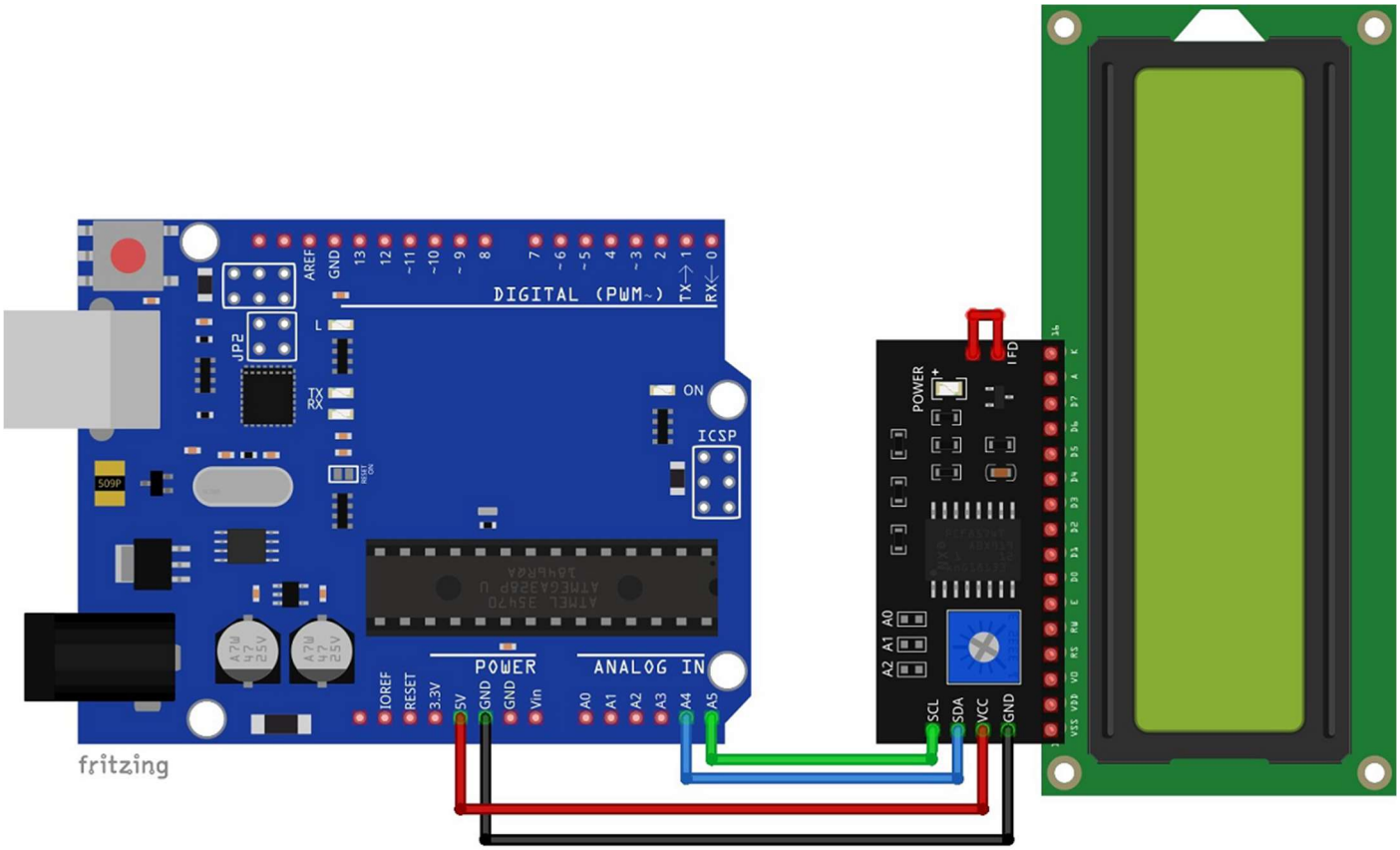
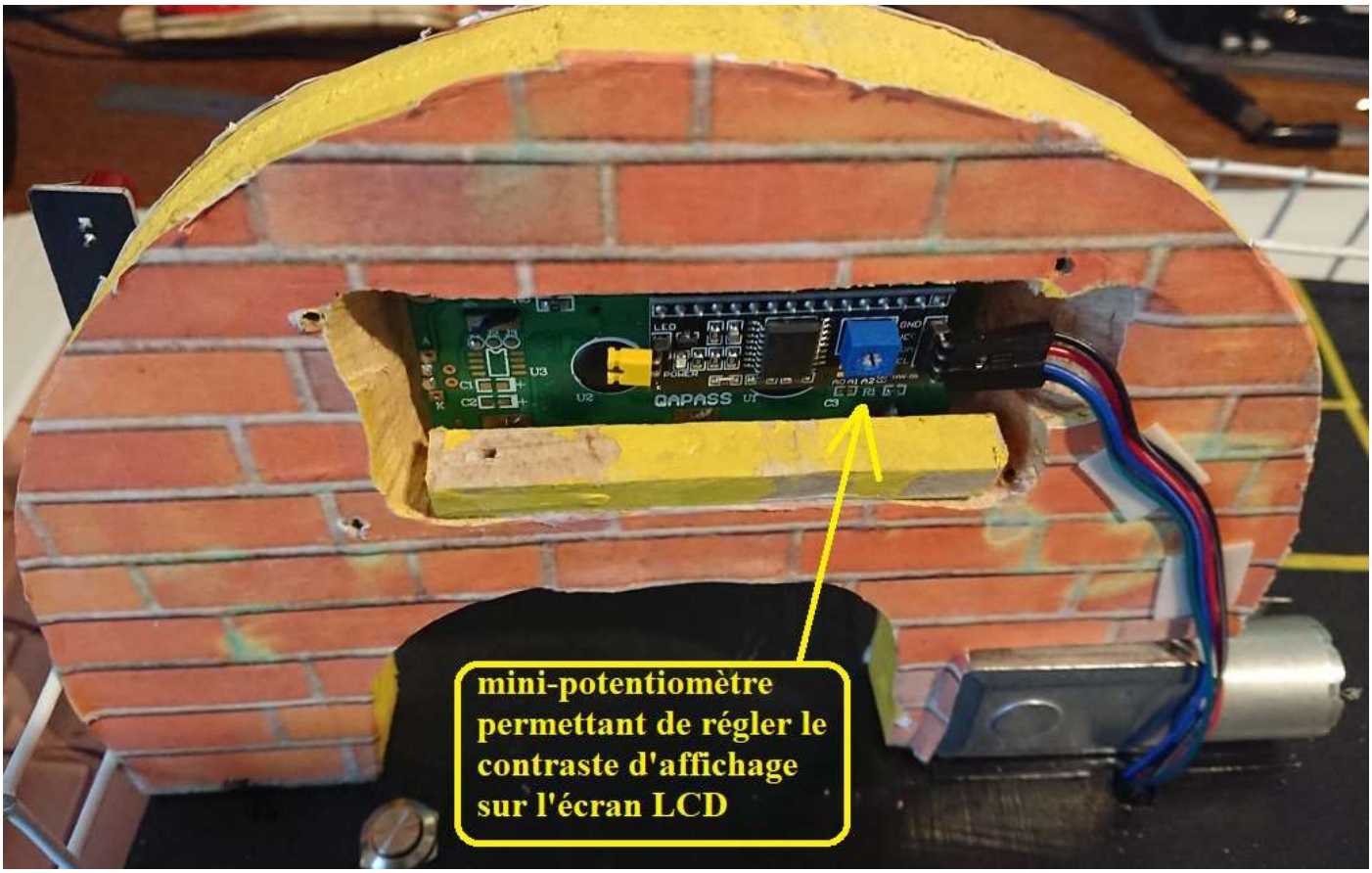


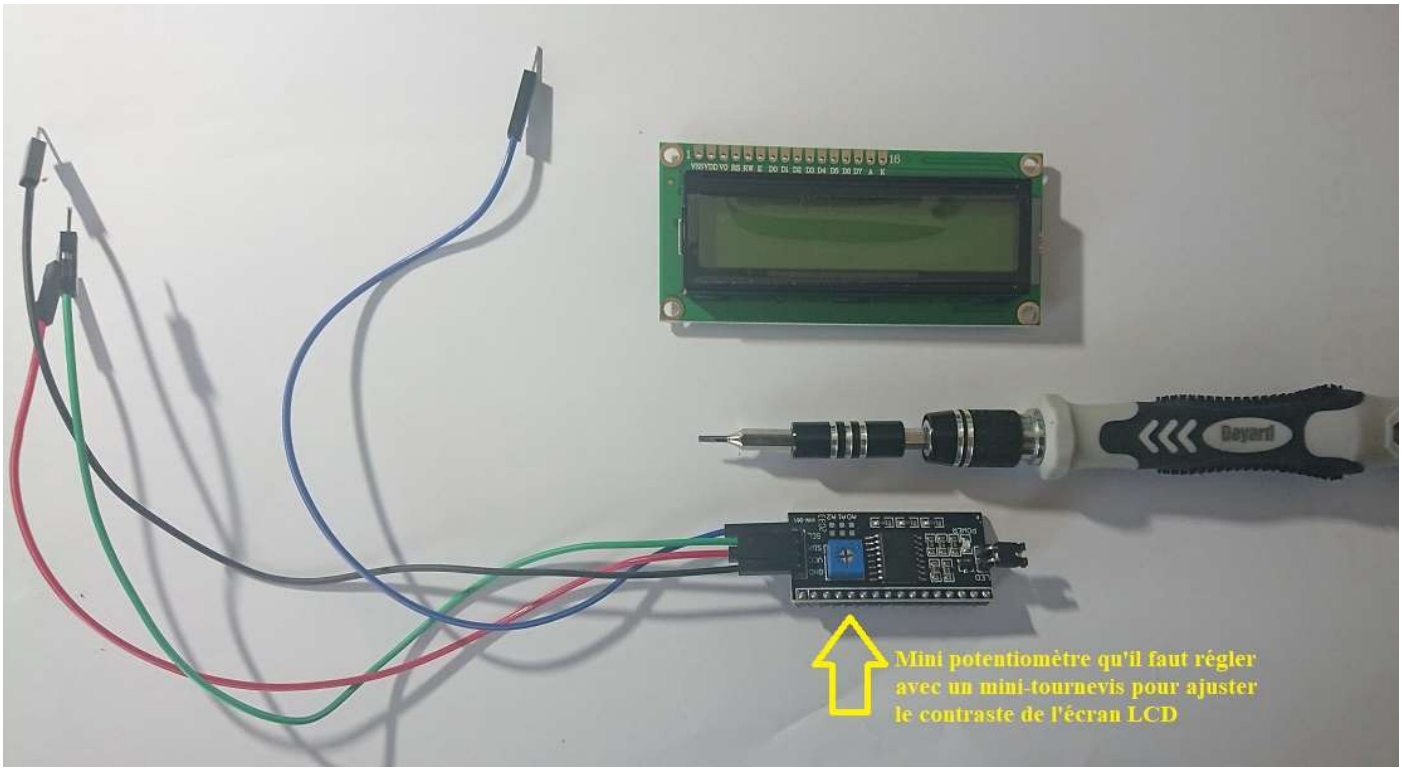
Schéma simplifié de connexion d'un écran LCD utilisé avec son interface I2C

Câble noir >>> Masse (-) GND de la carte Arduino ou sur une ligne GND de la breadboard
Câble rouge >>> Borne de sortie 5 Volts de la carte ou ligne rouge de la breadboard
Câble bleu (Serial Data Line) Données >>> Borne analogique A4
Câble vert (Serial Clock Line) Temporisations >>> Borne analogique A5

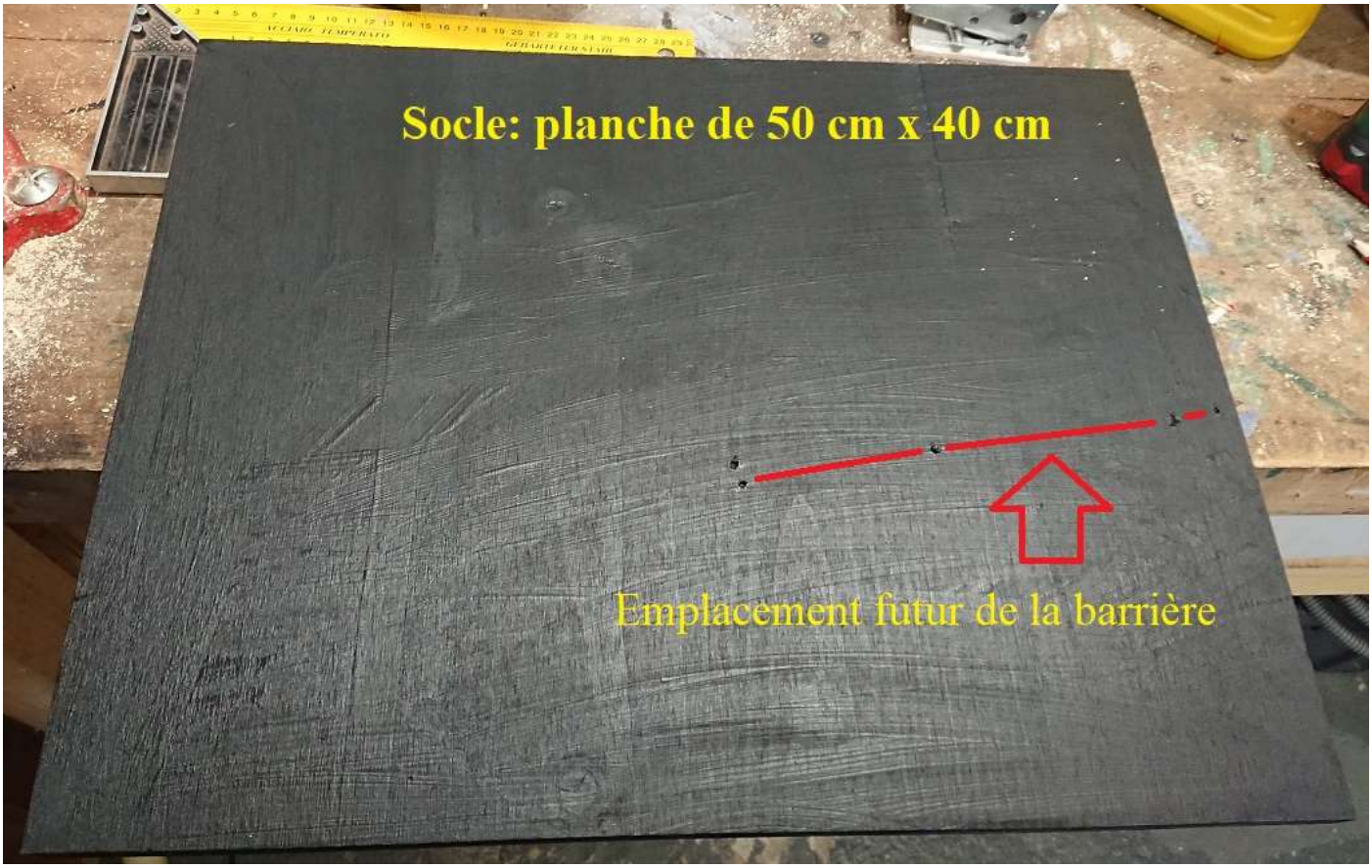
Avec ce système, deux bornes analogiques (A4 et A5) suffisent pour gérer l'écran LCD. Dans l'exemple de notre maquette, nous avons soudé les 16 broches de l'interface derrière l'écran LCD afin de limiter la surface occupée par l'ensemble et pouvoir encastrer le tout au-dessus du portail d'entrée du parking.



<u>Ecran LCD encastré au dessus du portail d'entrée du parking</u> (avec son interface I2C)

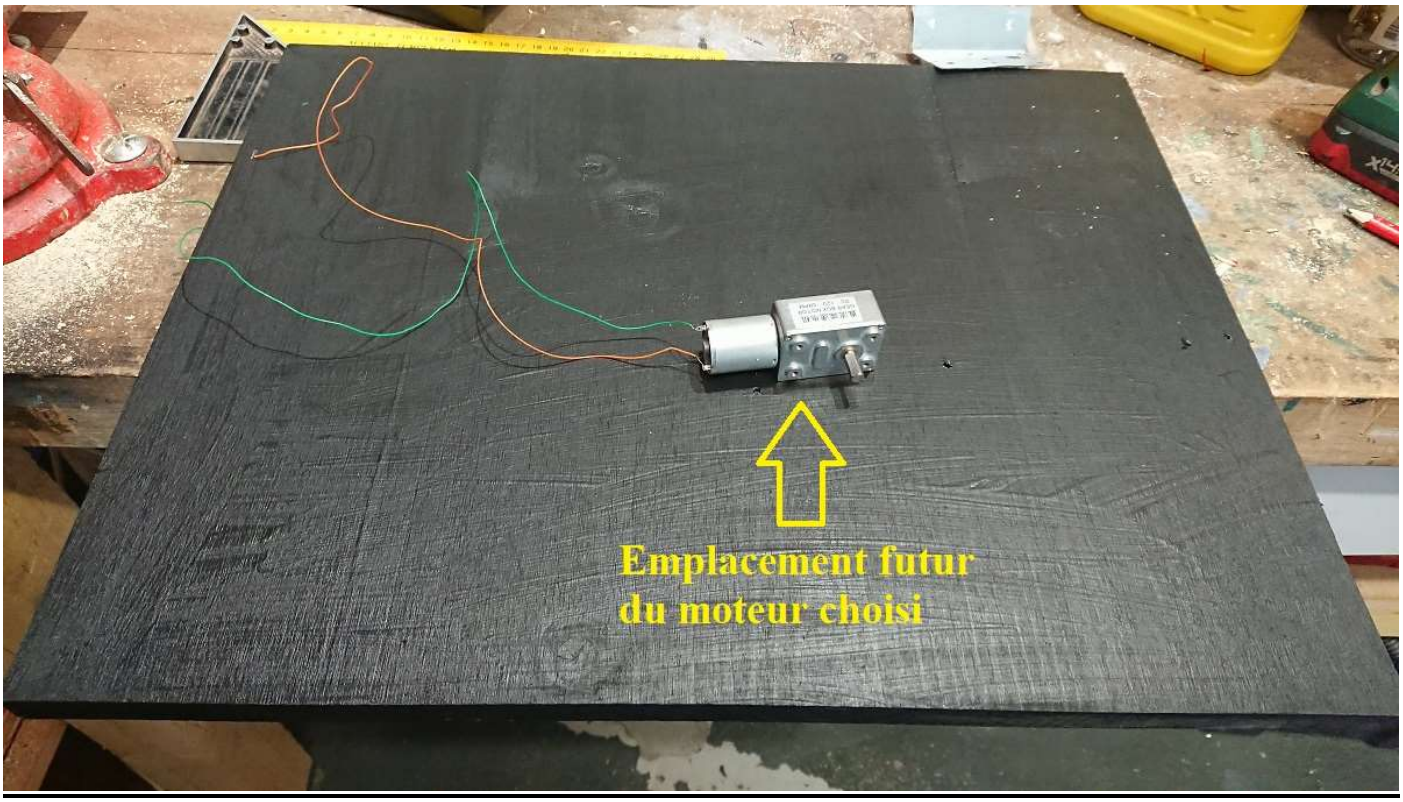


Construction de la maquette de parking



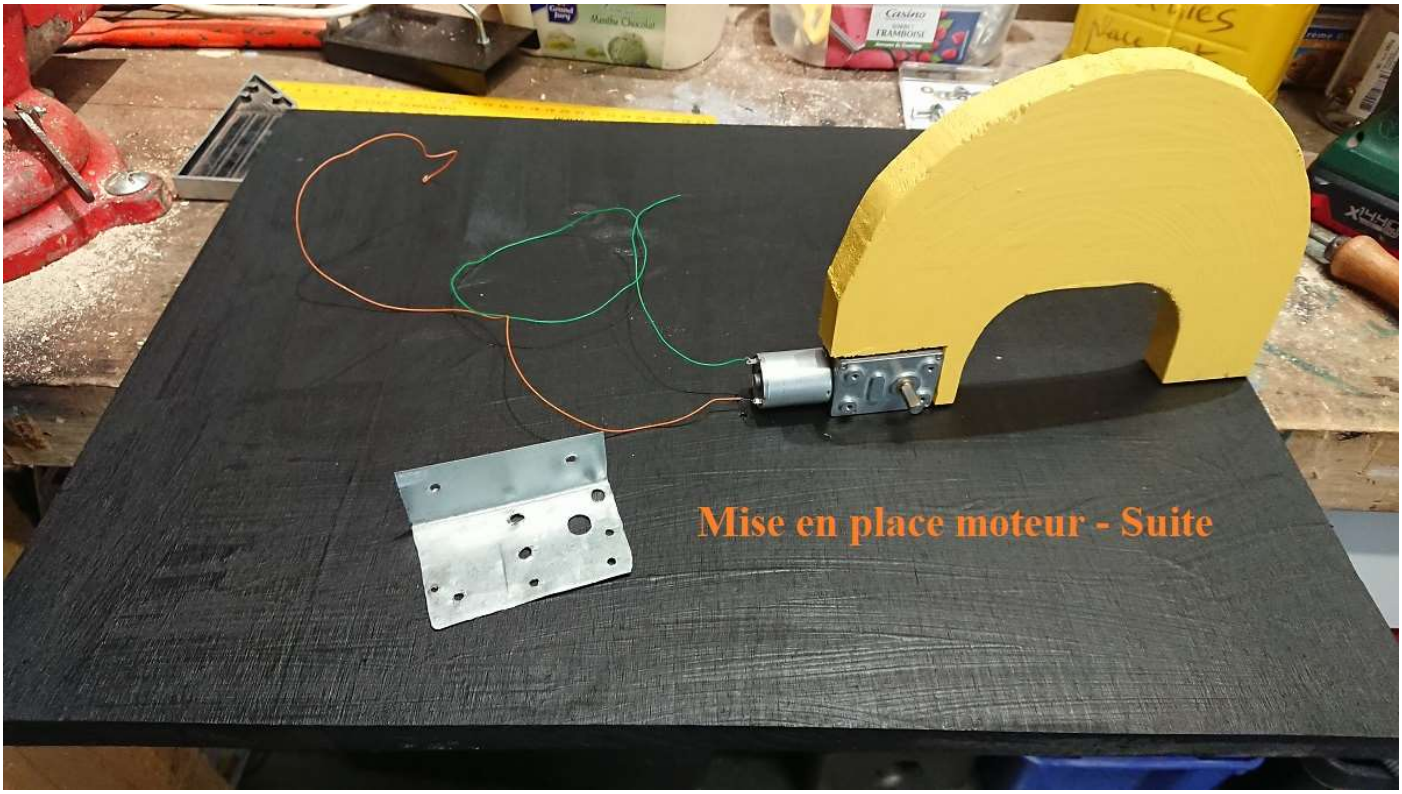
Socle de la maquette de parking

(Planche de 2 cm d'épaisseur enduite d'une peinture acrylique noire)



Position du moteur choisi

Le moteur sera positionné ainsi sur le socle de la maquette, dans le prolongement du futur portail dont on aperçoit les trous de fixation au sol.

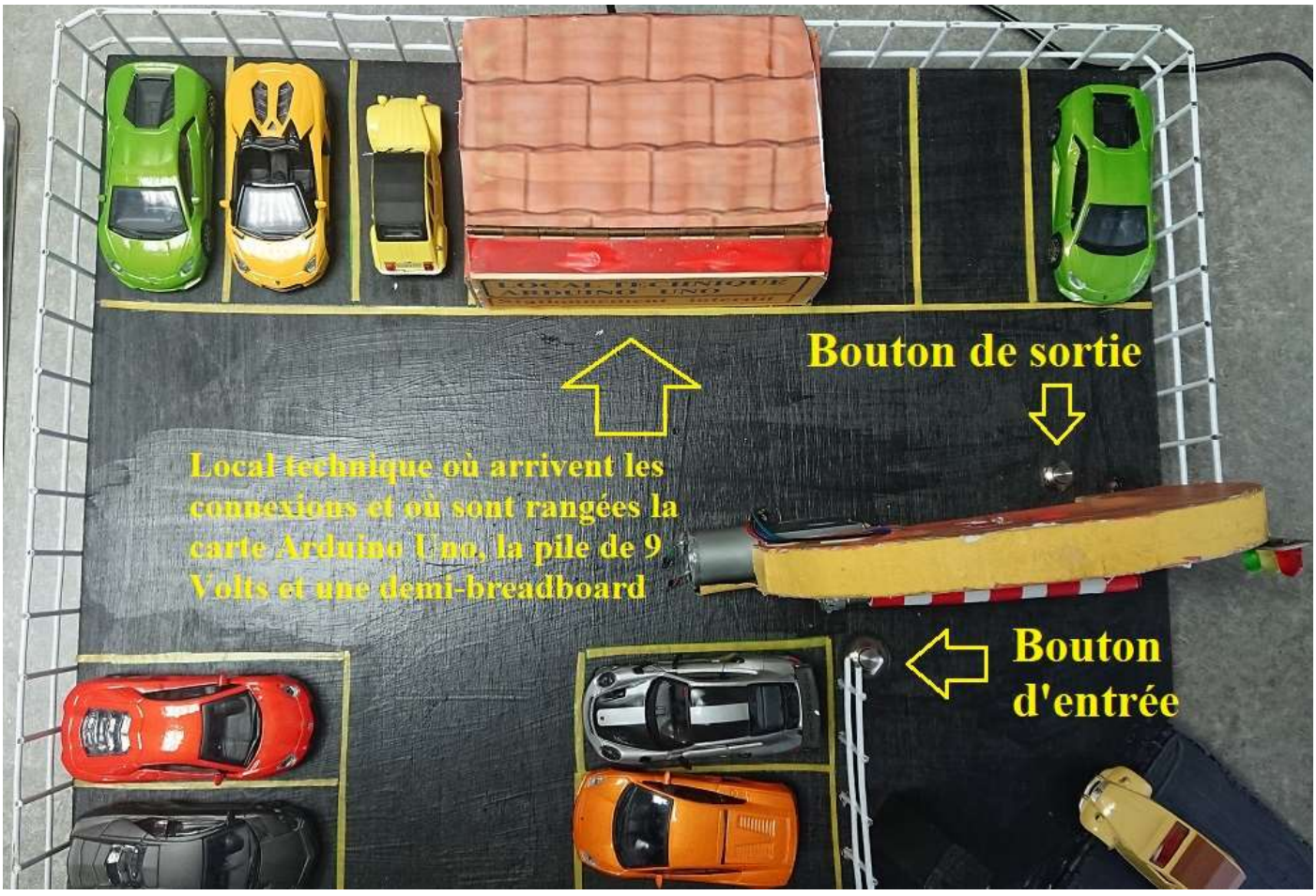


Le moteur sera vissé sur cette équerre métallique elle-même fixée sur le socle par deux vis à bois et ensuite également vissée sur le « mur d'entrée » par deux petites vis à bois.

Le moteur a été vissé derrière l'équerre métallique par 4 petites vis adaptées aux filetages pré-existants sur la partie réducteur de l'ensemble de motorisation. Un ««serre-tringle» pour garde-boue vélo est fixé sur l'axe de sortie du moteur sur lequel il s'adapte à la perfection. La «barrière» constituée par une tige de bambou de 12 cm de longueur est vissée sur le filetage du serre-tringle.

NB – Il est conseillé d'effectuer les essais et les réglages des temporisations avant de fixer définitivement la barrière car le couple obtenu à la sortie du réducteur est très puissant et des risques d'autodestruction de la barrière sont très possibles si la barrière heurte le sol et que le moteur continue à fonctionner (Voir lignes 63 à 68 et 79 à 83 du programme B4R). J'ai établi les temporisations pour ce type de moteur alimenté par une pile à pressions de 9 Volts. Si la tension de l'alimentation est inférieure (piles de 3 Volts ou 4,5 Volts), la vitesse de rotation du moteur sera plus lente et donc nécessitera plus de temps. Si la tension est supérieure (12 Volts par exemple), la vitesse de rotation du moteur sera plus rapide et nécessitera moins de temps.

Emplacements des deux feux tricolores et de l'écran LCD récapitulatif

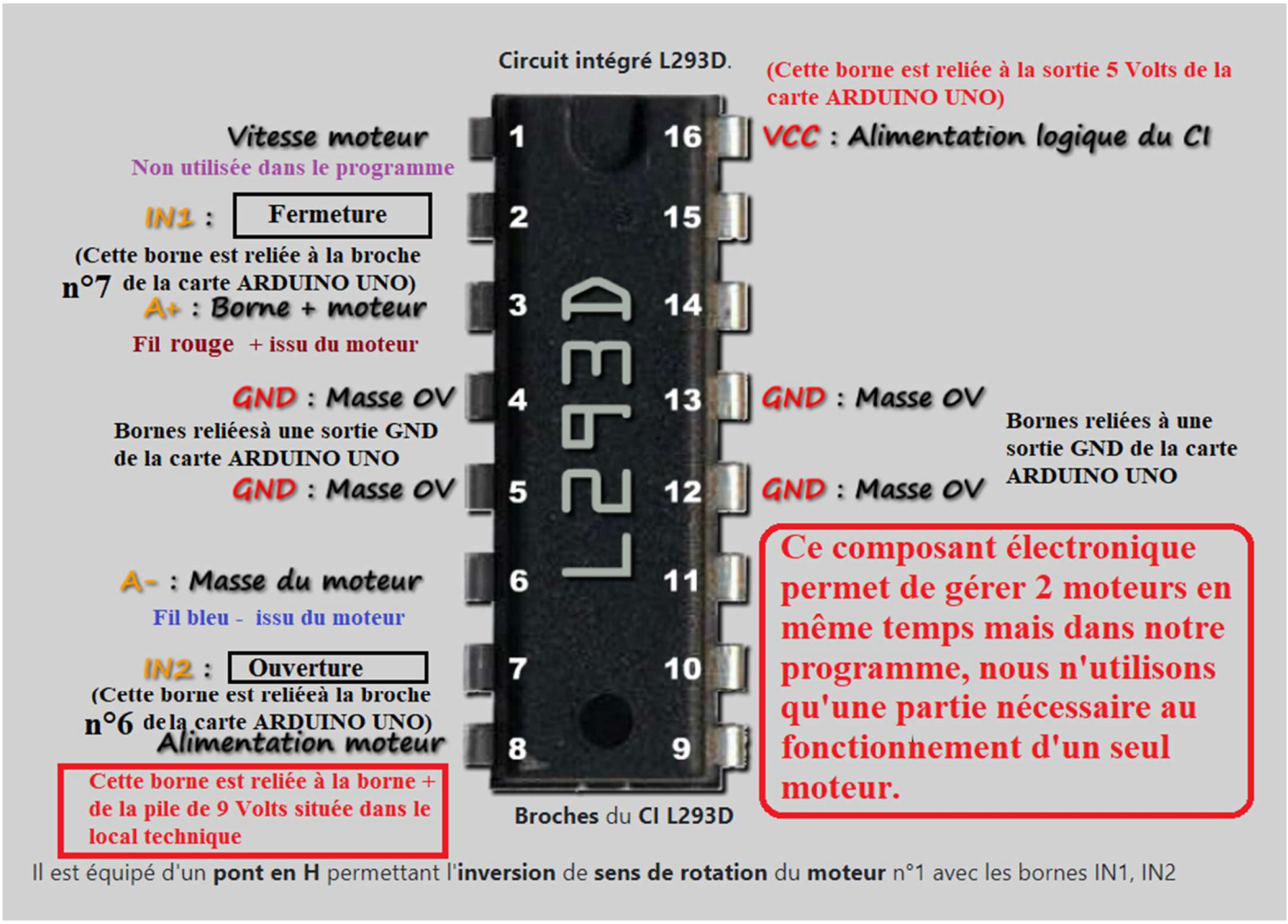


Maquette terminée vue par dessus

Pour fonctionner correctement, cette maquette doit être connectée et alimentée via la carte ARDUINO UNO fixée au fond du «local technique» du parking. Cette carte va gérer les boutons d'entrée et de sortie, les feux tricolores, le buzzer, l'affichage sur l'écran LCD, l'ouverture et la fermeture de la barrière et donc le sens de rotation du moteur qui sera inversé pour passer de l'ouverture à la fermeture grâce à l'utilisation du circuit intégré L293D fixé sur la demi-plaque de connexions dans le même local technique du parking. La carte ARDUINO UNO gère les temporisations : durée de l'ouverture, durée pendant laquelle la barrière reste ouverte pour permettre l'entrée ou la sortie d'un véhicule, durée de la fermeture. Ces temporisations ont été programmées dans les lignes 63 à 68 (entrées) et 79 à 83 (sorties) dans le programme B4R enregistré dans la carte ARDUINO UNO. (Voir le programme B4R complet en fin de document)

Le moteur lui-même est alimenté en courant continu par une pile à pressions de 9 Volts, stockée dans le même local technique.

Le circuit intégré L293D accroché sur la demi-plaque de connexions (« breadboard ») est alimenté en courant continu de 5 Volts fourni par la carte ARDUINO UNO.



Détail des connexions du circuit intégré « L293D » avec « GestionParking4.B4R»

Récapitulatif des branchements et connexions

Connexions entre la maquette et la carte ARDUINO UNO

Feux tricolores (vert, jaune, rouge) en double exemplaire
Feux verts (fils verts) >>> Broche n°8 de la carte Arduino Uno
Feux jaunes (fils jaunes) >>> Broche 13 de la carte Arduino Uno
(NB – La broche n° 13 allumera également la petite diode n° 13 intégrée à la carte Arduino Uno)

Feux rouges (fils rouges) >>> Broche 9 de la carte Arduino Uno
Bouton Entrée >>> Broche d'entrée analogique A0 de la carte Arduino Uno
Bouton Sortie >>> Broche d'entrée analogique A1 de la carte Arduino Uno
BUZZER >>> Broche n°2 de la carte Arduino Uno
(Le buzzer est fixé au plafond-couvercle du « local technique »)
Masse de la maquette (fil noir) >>> **Une broche GND** de la carte Arduino Uno
(Les masses des feux tricolores, des boutons d'entrée et de sortie, du buzzer sont toutes reliées à la masse de la carte Arduino Uno par l'intermédiaire des lignes GND de la plaque de connexions)

Connexion entre le moteur et le circuit L293D

Fil marron (+) >>> Borne 3 du circuit intégré L293D
Fil vert (-) >>> Borne 6 du circuit intégré L293D

Connexion entre la pile de 9 Volts et le circuit L293D

Fil rouge (+) >>> Borne 8 du circuit intégré L293D (Alimentation du moteur)
Fil bleu (-) >>> Masse GND de la plaque de connexions

Connexions entre le circuit L293D et la carte ARDUINO UNO

Sortie 5 Volts de la carte reliée à la borne 16 de L293D (Alimentation du circuit intégré L293D)
Broche n° 6 de la carte reliée à la borne 7 de L293D (Ouverture de la barrière)
Broche n° 7 de la carte reliée à la borne 2 de L293 (Fermeture de la barrière)
Autre broche GND (fil noir) de la carte reliée à la ligne GND – de la plaque de connexions sans soudures
Notons pour terminer la nécessité de **cavaliers (noirs)** entre les lignes GND de la plaque de connexions et les bornes 4 et 5 et 12 et 13 de L293D

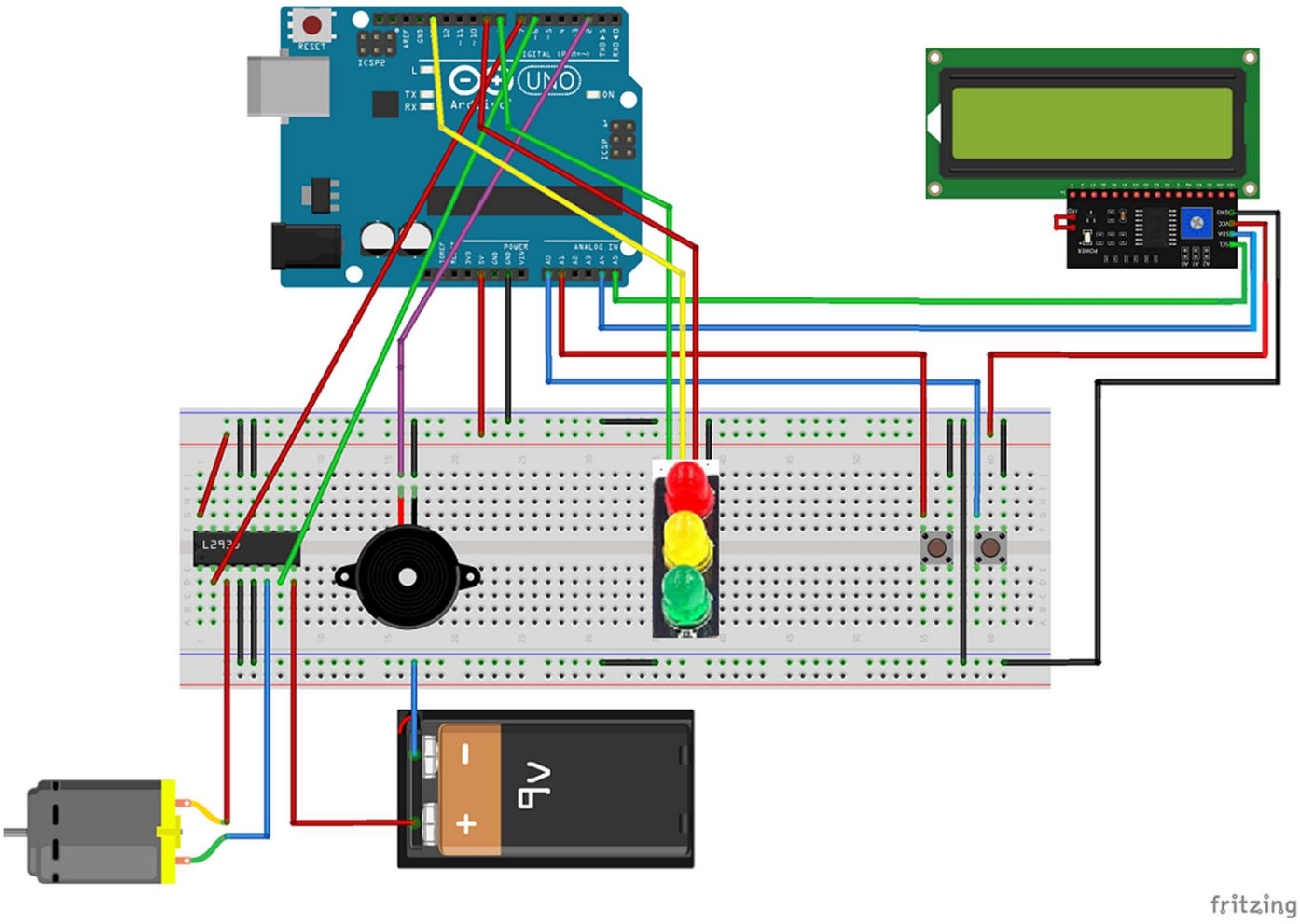
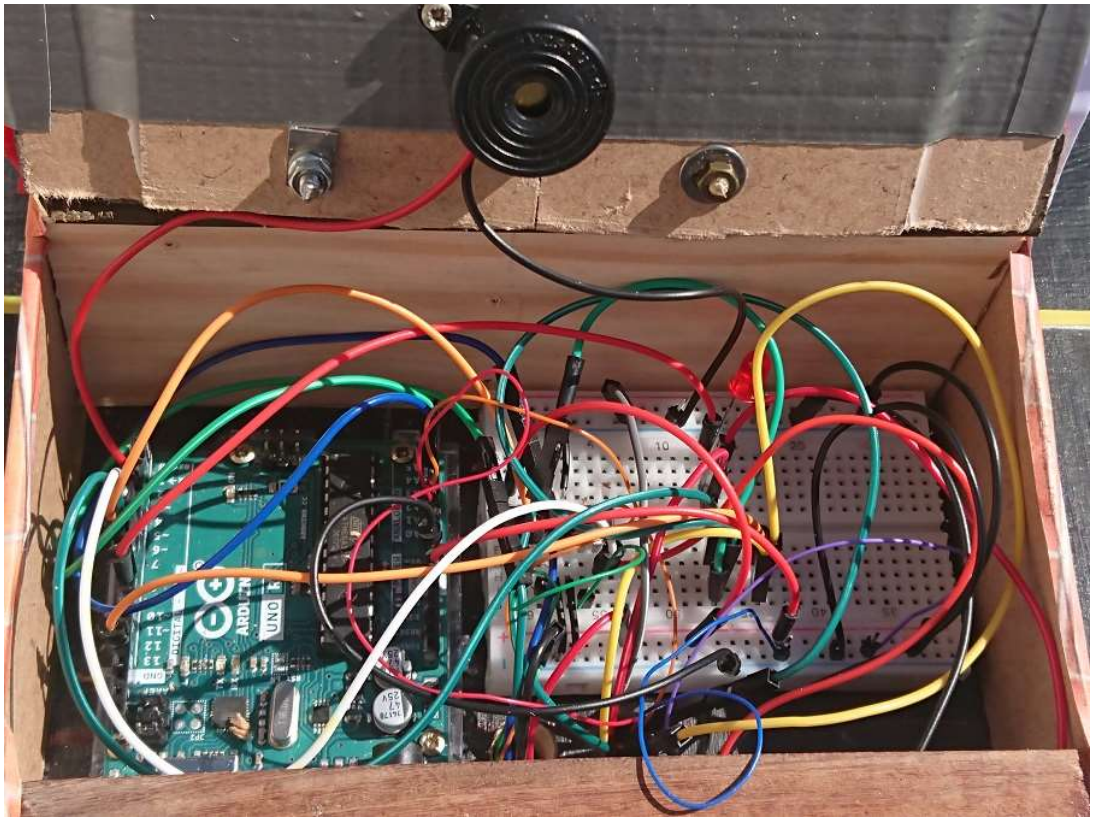
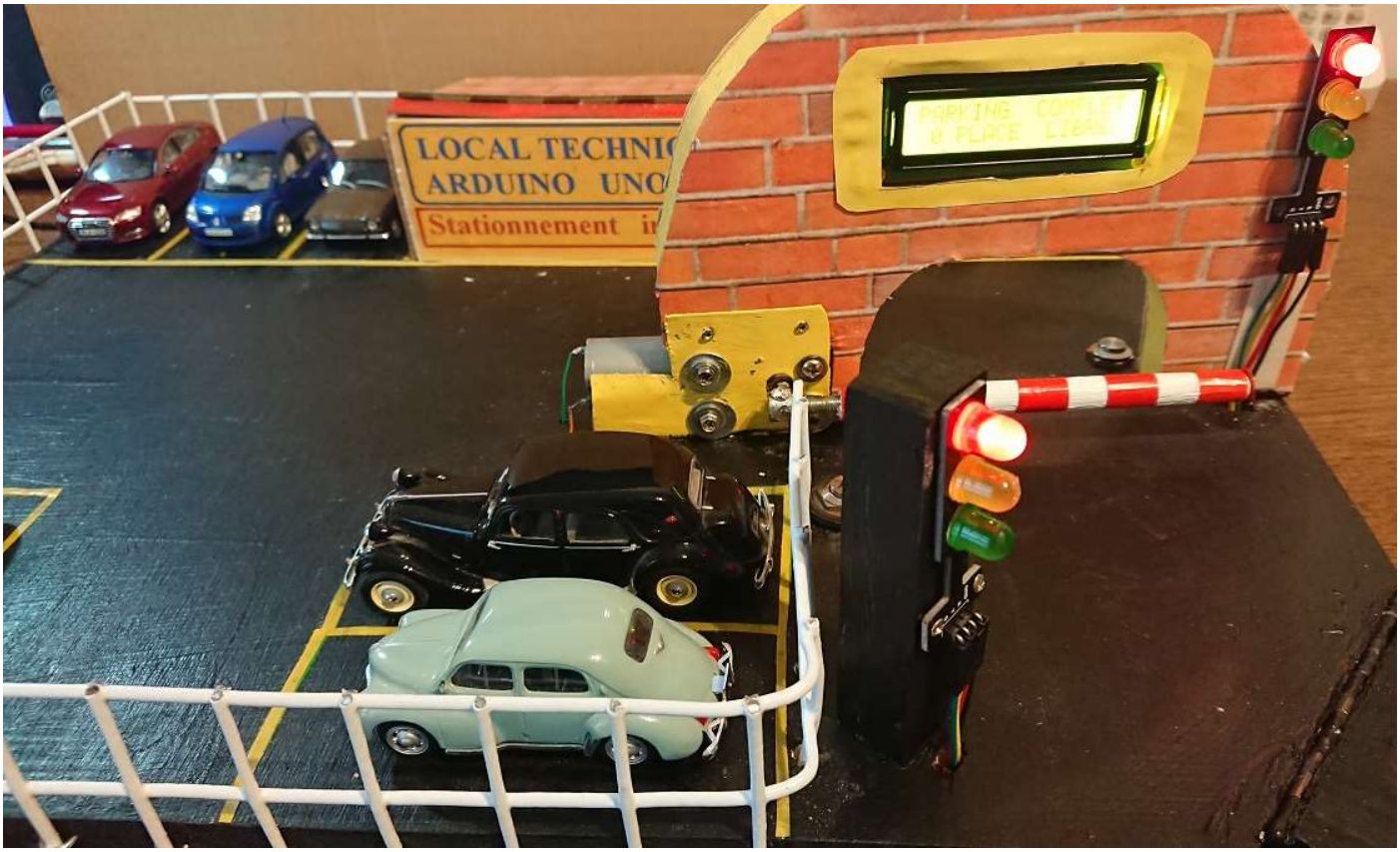


Schéma général des connexions utilisées dans ce projet



Connexions installées dans le local technique sur le parking



Parking complet – Les feux rouges sont allumés

Programme GestionParking4.B4R

```
' GestionParking4.B4R - Utilisation d'un écran LCD
#Region Project Attributes
#AutoFlushLogs: True
#CheckArrayBounds: True
#StackBufferSize: 300
#End Region
' GESTION D'UN PARKING VIRTUEL DE 10 PLACES (Maquette construite) DONT 10 sont LIBRES au démarrage du programme
' -:-:-:-:-:-:-:-:- Marc DANIEL - Avril 2021 -:-:-:-:-:-:-:-:-:-
' CARTE ARDUINO UNO - Circuit Intégré L293D - Ecran LCD et son interface I2C (2 lignes de 16 caractères)

Sub Process_Globals
Public Serial1 As Serial
Private pinButtonEntree As Pin 'broche pour le bouton d'entrée du parking
Private pinButtonSortie As Pin 'broche pour le bouton de sortie du parking
Private pinLEDVerte, pinLEDRouge, pinLEDJaune As Pin 'broches pour les feux tricolores
Private PinBuzzer As Pin ' broche pour le buzzer (Brève sonnerie à la pression du bouton d'entrée)
Private pinOuverture, pinFermeture As Pin 'broches pour les connexions motorisation barrière
Public Places As UInt
Private LCD As LiquidCrystal_I2C 'Bibliothèque « rLiquidCrystal_I2C» à charger et à utiliser
End Sub

Private Sub AppStart
Serial1.Initialize(115200)
pinButtonEntree.Initialize(pinButtonEntree.A0, pinButtonEntree.MODE_INPUT_PULLUP)
pinButtonEntree.AddListener("pinButtonEntree_StateChanged")
pinButtonSortie.Initialize(pinButtonSortie.A1, pinButtonSortie.MODE_INPUT_PULLUP)
pinButtonSortie.AddListener("pinButtonSortie_StateChanged")
pinLEDVerte.Initialize(8, pinLEDVerte.MODE_OUTPUT)
pinLEDRouge.Initialize(9, pinLEDRouge.MODE_OUTPUT)
pinLEDJaune.Initialize(13,pinLEDJaune.MODE_OUTPUT) ' (+ LED 13 intégrée aux cartes Arduino)
PinBuzzer.Initialize(2, PinBuzzer.MODE_OUTPUT)
pinOuverture.Initialize(6, pinOuverture.MODE_OUTPUT) 'Connexion au composant L293D borne n°2  (pour l'ouverture de la barrière)
pinFermeture.Initialize (7, pinFermeture.MODE_OUTPUT) 'Connexion au composant L293D borne n°7  (pour la fermeture de la barrière)
Places=10
'10 places de parking sont disponibles au démarrage et donc le parking est vide
CallSubPlus("GestionPlaces", 0, 0)
CallSubPlus("Depart", 0,0)
'Broches analogiques A4 et A5 réservées pour la connexion de l'écran LCD à l'adresse «0x27»
LCD.Initialize(0x27, 16, 2) ' Initialisation de l'écran LCD avec 2 lignes de 16 caractères
End Sub

Private Sub Depart
PinBuzzer.DigitalWrite(False)
CallSubPlus("GestionPlaces", 0, 0)
'LCD.Clear
LCD.Backlight = True
LCD.SetCursor(0,0)
LCD.Write("PARKING CENTRAL")
LCD.SetCursor(0,1) 'Place le curseur au début de la seconde ligne de l'écran LCD
LCD.Write("10 PLACES LIBRES")
pinLEDVerte.DigitalWrite(True)
End Sub

Private Sub pinButtonEntree_StateChanged(State1 As Boolean)
Log("État: ", State1) 'Log la valeur de State1
If State1 = False Then
If Places=0 Then CallSubPlus("GestionPlaces",0,0)
If Places > 0 Then
Places=Places-1
(Ligne 63) CallSubPlus("Buzzer",0,0)
(Ligne 64) CallSubPlus("GestionPlaces",0,0)
(Ligne 65) CallSubPlus("Ouverture",500,0) ' Ouverture de la barrière
(Ligne 66) CallSubPlus("Pause",5150,0) 'Mouvement du véhicule
(Ligne 67) CallSubPlus("Fermeture", 10300, 0) ' Fermeture de la barrière
(Ligne 68)CallSubPlus("FinEntree",15000,0) ' Fin de la fermeture de la barrière
End If
```

```
End If
End Sub

Private Sub pinButtonSortie_StateChanged(State2 As Boolean)
Log("État: ", State2) 'Log la valeur de State2
If State2 = False Then
If Places=10 Then CallSubPlus("GestionPLaces",0,0)
If Places < 10 Then
Places=Places+1
(Ligne 79) CallSubPlus("GestionPlaces",0,0)
(Ligne 80) CallSubPlus("Ouverture",200,0) ' Ouverture de la barrière
(Ligne 81) CallSubPlus("Pause",4850,0) 'Mouvement du véhicule
(Ligne 82) CallSubPlus("Fermeture", 10000, 0) ' Fermeture de la barrière
Ligne 83) CallSubPlus("FinSortie",14700,0) ' Fin de la fermeture de la barrière
End If
End If
End Sub

Private Sub Ouverture(Tag As Byte)
pinOuverture.DigitalWrite(True) ' ouvre la barrière pour entrée ou sortie d'un véhicule
End Sub

Private Sub Pause(Tag As Byte)
pinOuverture.DigitalWrite(False) 'La barrière reste ouverte - Mouvement du véhicule entrant ou sortant
End Sub

Private Sub Fermeture(Tag As Byte)
pinFermeture.DigitalWrite(True) 'fermeture de la barrière
End Sub


Private Sub FinEntree(Tag As Byte)
pinFermeture.DigitalWrite(False) 'Arrêt de la fermeture de la barrière
End Sub


Private Sub FinSortie(Tag As Byte)
pinFermeture.DigitalWrite(False) 'Arrêt de la fermeture de la barrière
End Sub


Private Sub Buzzer(Tag As Byte)
PinBuzzer.DigitalWrite(True)
Delay (500)
PinBuzzer.DigitalWrite(False)
End Sub


Private Sub GestionPlaces
Select Places
Case 0
pinLEDVerte.DigitalWrite(False) ' éteint le feu vert
pinLEDJaune.DigitalWrite(False) ' éteint le feu orange
pinLEDRouge.DigitalWrite(True) 'allume le feu rouge (plus de places)
LCD.Clear
LCD.Write("PARKING COMPLET")
LCD.SetCursor(0, 1) 'Place le curseur au début de la seconde ligne de l'écran LCD
LCD.Write(" 0 PLACE LIBRE")
Case 1
pinLEDRouge.DigitalWrite(False) 'éteint le feu rouge
pinLEDJaune.DigitalWrite(True) ' allume le feu orange (Indique qu'il ne reste plus qu'une seule place de libre)
LCD.Clear
LCD.Write("PARKING CENTRAL")
LCD.SetCursor(0, 1) 'Place le curseur au début de la seconde ligne de l'écran LCD
LCD.Write(" 1 PLACE LIBRE")
Case 2
pinLEDRouge.DigitalWrite(False) ' éteint le feu rouge
pinLEDJaune.DigitalWrite(False) ' éteint le feu orange
pinLEDVerte.DigitalWrite(True) ' allume le feu vert
LCD.Clear
```

```
LCD.Write("PARKING CENTRAL")
LCD.SetCursor(0, 1) 'Place le curseur au début de la seconde ligne del'écran LCD
LCD.Write("2 PLACES LIBRES")
Case 3
pinLEDRouge.DigitalWrite(False) 'éteint la LED Rouge
pinLEDJaune.DigitalWrite(False) ' éteint le feu orange
pinLEDVerte.DigitalWrite(True)
LCD.Clear
LCD.Write("PARKING CENTRAL")
LCD.SetCursor(0, 1) 'Place le curseur au début de la seconde ligne de l'écran LCD
LCD.Write("3 PLACES LIBRES")
Case 4
pinLEDRouge.DigitalWrite(False) 'éteint la LED Rouge
pinLEDJaune.DigitalWrite(False) ' éteint le feu orange
pinLEDVerte.DigitalWrite(True)
LCD.Clear
LCD.Write("PARKING CENTRAL")
LCD.SetCursor(0, 1) 'Place le curseur au début de la seconde ligne de l'écran LCD
LCD.Write("4 PLACES LIBRES")
Case 5
pinLEDRouge.DigitalWrite(False) 'éteint la LED Rouge
pinLEDJaune.DigitalWrite(False) ' éteint le feu orange
pinLEDVerte.DigitalWrite(True)
LCD.Clear
LCD.Write("PARKING CENTRAL")
LCD.SetCursor(0, 1) 'Place le curseur au début de la seconde ligne de l'écran LCD
LCD.Write("5 PLACES LIBRES")
Case 6
pinLEDRouge.DigitalWrite(False) 'éteint la LED Rouge
pinLEDJaune.DigitalWrite(False) ' éteint le feu orange
pinLEDVerte.DigitalWrite(True)
LCD.Clear
LCD.Write("PARKING CENTRAL")
LCD.SetCursor(0, 1) 'Place le curseur au début de la seconde ligne de l'écran LCD
LCD.Write("6 PLACES LIBRES")
Case 7
pinLEDRouge.DigitalWrite(False) 'éteint la LED Rouge
pinLEDJaune.DigitalWrite(False) ' éteint le feu orange
pinLEDVerte.DigitalWrite(True)
LCD.Clear
LCD.Write("PARKING CENTRAL")
LCD.SetCursor(0, 1) 'Place le curseur au début de la seconde ligne de l'écran LCD
LCD.Write("7 PLACES LIBRES")
Case 8
pinLEDRouge.DigitalWrite(False) 'éteint la LED Rouge
pinLEDJaune.DigitalWrite(False) ' éteint le feu orange
pinLEDVerte.DigitalWrite(True)
LCD.Clear
LCD.Write("PARKING CENTRAL")
LCD.SetCursor(0, 1) 'Place le curseur au début de la seconde ligne de l'écran LCD
LCD.Write("8 PLACES LIBRES")
Case 9
pinLEDRouge.DigitalWrite(False) 'éteint la LED Rouge
pinLEDJaune.DigitalWrite(False) ' éteint le feu orange
pinLEDVerte.DigitalWrite(True)
LCD.Clear
LCD.Write("PARKING CENTRAL")
LCD.SetCursor(0, 1) 'Place le curseur au début de la seconde ligne de l'écran LCD
LCD.Write("9 PLACES LIBRES")
Case 10
pinLEDRouge.DigitalWrite(False) 'éteint la LED Rouge
pinLEDJaune.DigitalWrite(False) ' éteint le feu orange
pinLEDVerte.DigitalWrite(True)
LCD.Clear
LCD.Write("PARKING CENTRAL")
LCD.SetCursor(0, 1) 'Place le curseur au début de la seconde ligne de l'écran LCD
LCD.Write("10 PLACES LIBRES")
End Select
End Sub
```

Marc DANIEL – Avril 2021

Ce programme GestionParking4.B4R a été développé grâce à la plate-forme : https://www.b4x.com/

Et plus particulièrement B4R >>> Learn Arduino ESP8266 & ESP32 Basic Programming | B4R (b4x.com)

Installation des logiciels nécessaires ici >>> https://www.b4x.com/b4r.html#installation

Maquette de parking terminée