GESTION D'UN PARKING DE 10 PLACES

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 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).

Ébauche du projet de maquette

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 le feu rouge situé en haut et à droite de l'entrée extérieure s'allume. 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 le feu restera rouge.

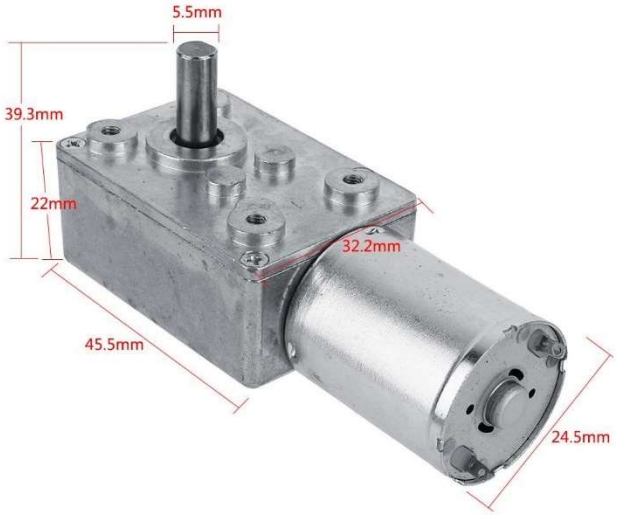
En revanche, si un véhicule sort du parking en libérant une place, le feu rouge sera éteint et une diode verte correspondante à la place libre s'allumera au-dessus de portail et donc il sera possible d'ouvrir à nouveau la barrière pour l'entrée d'un véhicule.

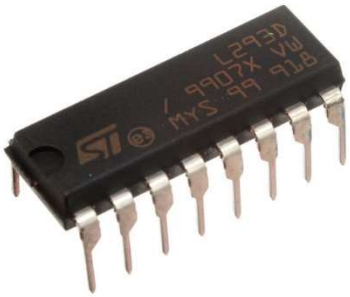
Dix diodes vertes situées au-dessus du portail d'entrée correspondent chacune à une place disponible lorsqu'elles sont allumées. Si le parking est vide, les dix diodes seront allumées comme au démarrage du programme par défaut.

Lorsque le bouton d'entrée est pressé, un buzzer sonne pendant quelques secondes, la barrière s'ouvre s'il reste une ou plusieurs places, une diode verte s'éteint car une place disponible devient occupée.

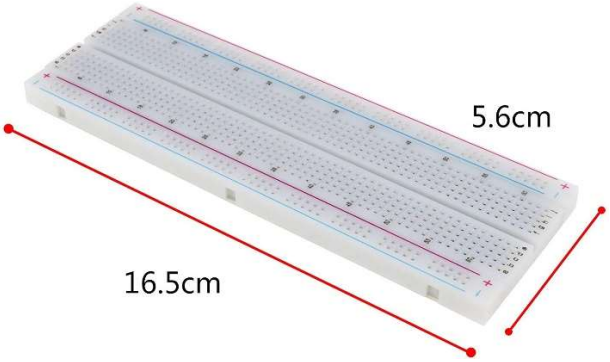
Le bouton de sortie est situé à l'intérieur du parking à gauche du portail de sortie. Ce bouton ne déclenchera l'ouverture de la barrière que s'il reste des véhicules sur le parking. Si le parking est vide, la barrière ne s'ouvrira pas.

Matériel utilisé





- 1 moteur FTVOGUE 12 V avec réducteur (engrenages métalliques)
- 10 LEDs vertes précâblées avec résistance incorporée
- 1 LED rouge précâblée avec résistance incorporée
- 2 gros boutons-poussoir
- 1 buzzer (Brève sonnerie à l'entrée)
- 1 circuit intégré L293D
- 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:

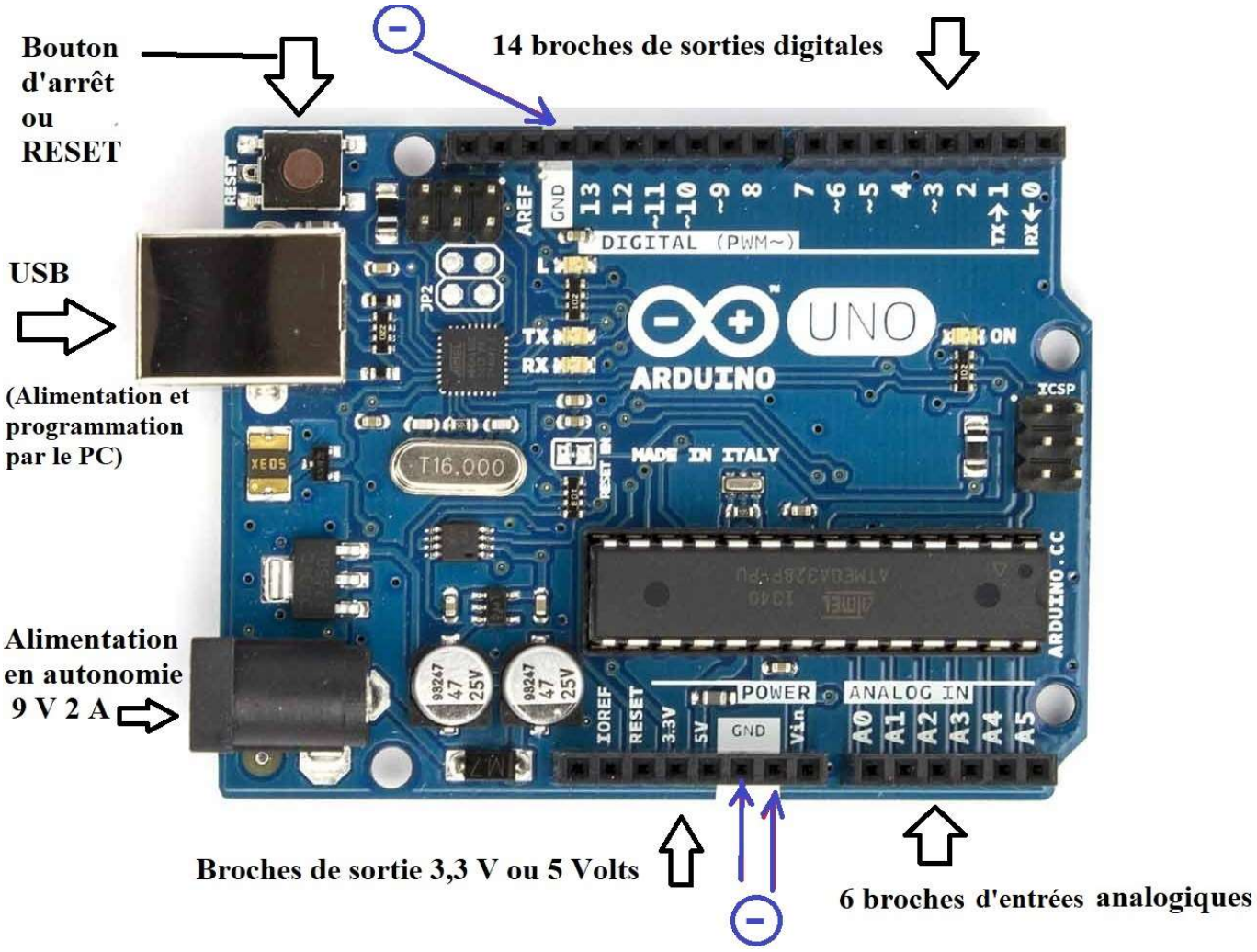


Schéma d'une carte ARDUINO UNO REV 3 avec ses broches de connexion

La carte ARDUINO UNO peut être alimentée elle-même soit par un câble USB branché sur un ordinateur ou un chargeur de téléphone (Entrée USB) soit par un transformateur 9V 2A sur la seconde entrée.

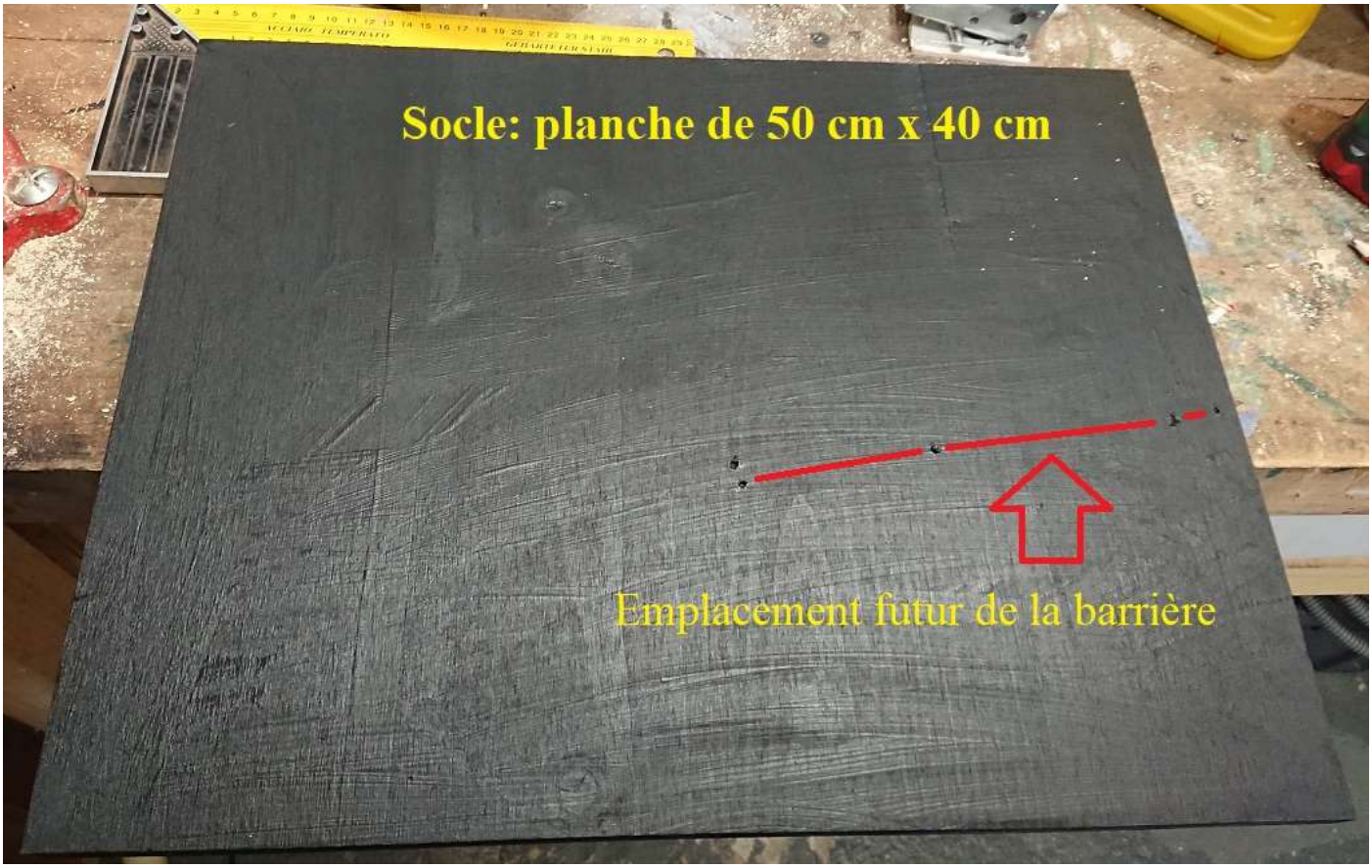
La programmation de la carte se fait exclusivement par l'entrée USB connectée à l'ordinateur sur lequel le programme ARDUINO ou B4R (Visual Basic pour Arduino) est développé.

Il existe trois bornes de terre GND (de l'anglais Ground) correspondantes au pôle négatif.

Il existe deux bornes de pôle positif, l'une de 5 Volts que nous utilisons pour alimenter le composant L293D et l'autre de 3,3 Volts que nous n'utilisons pas dans ce programme.

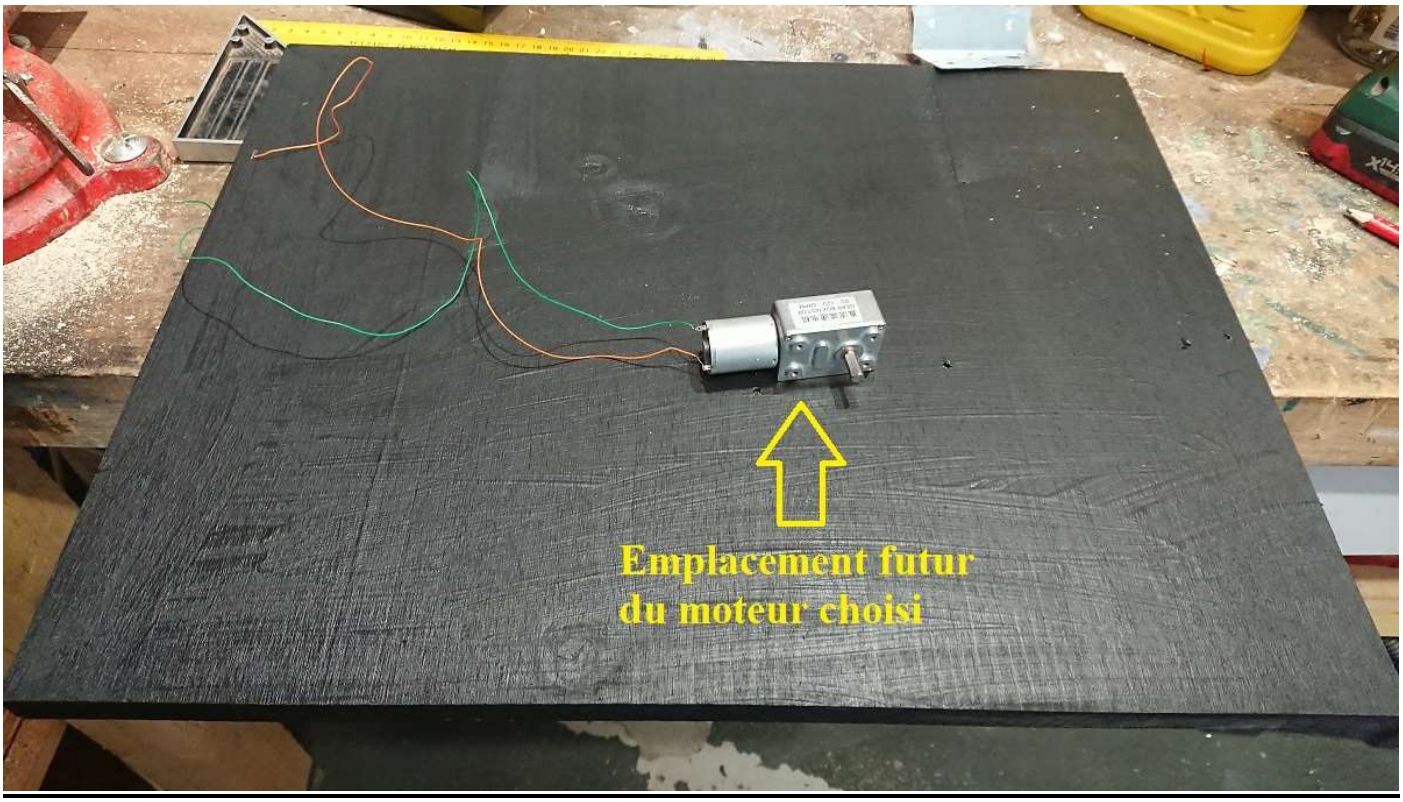
La maquette est constituée d'une planche en bois de 50 cm de longueur sur 40 cm de largeur avec une épaisseur de 2 cm. La clôture a été coupée à la pince coupante dans du grillage métallique laqué blanc et fixée à la planche support par des petits cavaliers enfoncés au marteau.
Le moteur est fixé sur une petite équerre métallique elle-même vissée sur le socle en bois d'une part et sur la partie gauche du portail d'entrée également découpé dans une planche de 2 cm d'épaisseur.
La barrière est une branche de bambou de 12 cm de longueur, vissée sur le filetage d'un serre-tringle pour garde boue vélo qui s'adapte exactement sur l'axe de sortie du moteur avec une vis de blocage perpendiculaire.

Construction de la maquette de parking



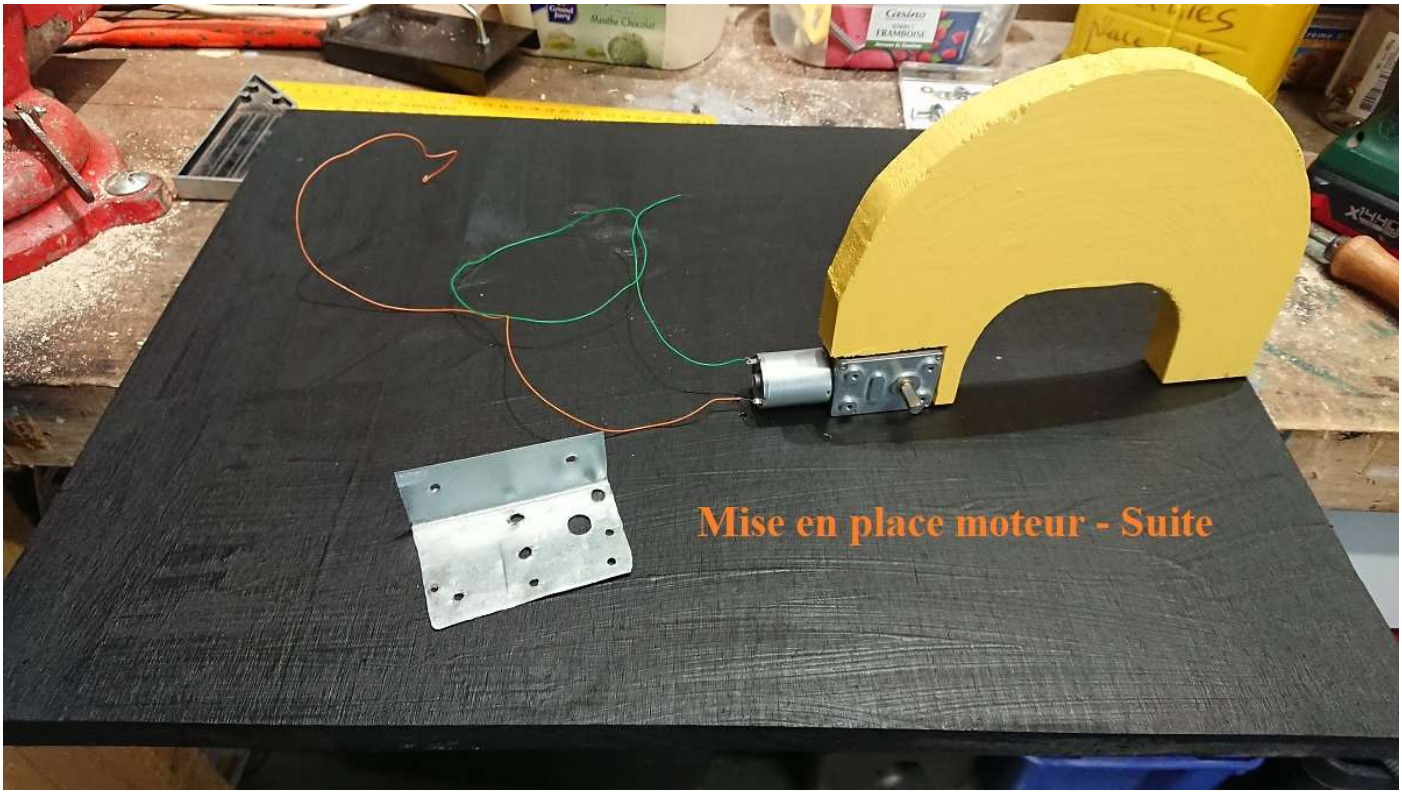
Socle de la maquette de parking

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



Position du moteur choisi

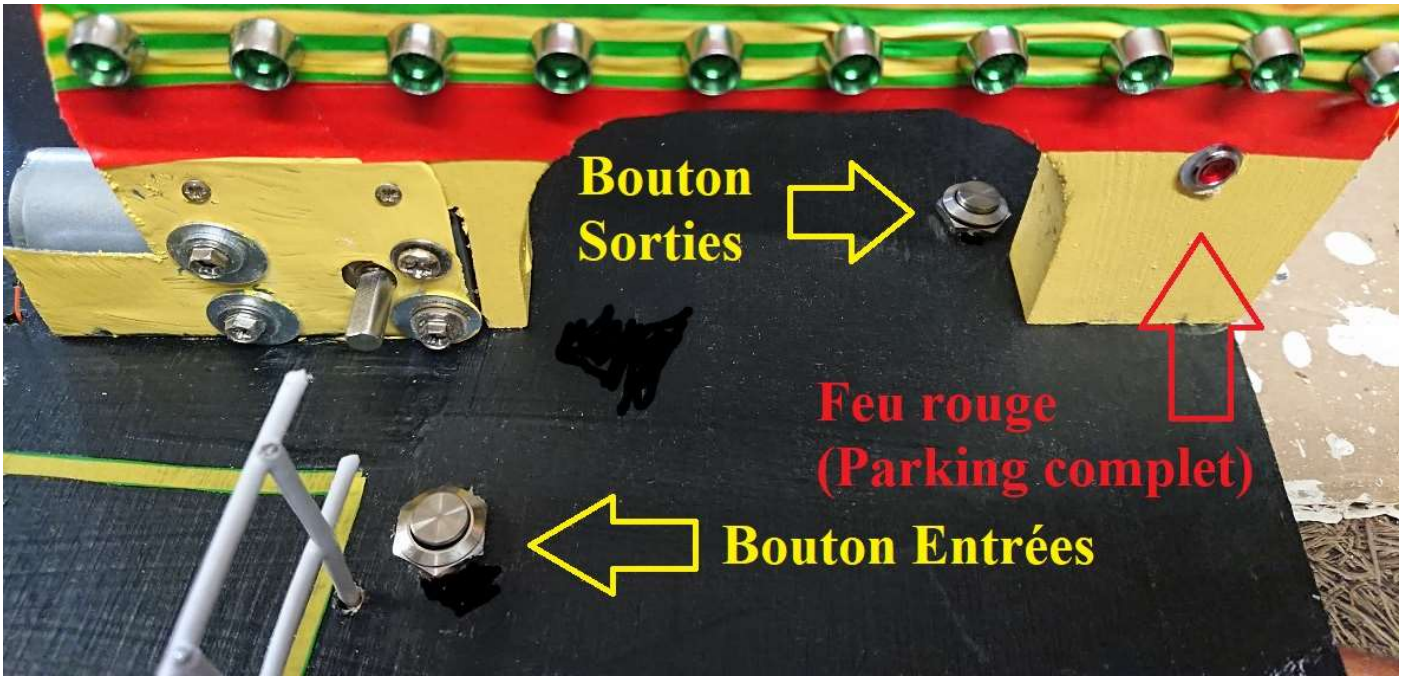
Le moteur sera positionné ainsi sur le socle de la maquette, dans le prolongement du futur portail dont on aperçoit les trous de fixation au sol.



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 78 à 83 et 94 à 98 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 boutons-poussoirs et du feu rouge

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 diodes vertes et le feu rouge, le buzzer, 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 78 à 83 (entrées) et 94 à 98 (sorties) dans le programme B4R enregistré dans la carte ARDUINO UNO.

(Voir le détail du programme **GestionParking3.B4R** à la fin de ce 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.

Récapitulatif des branchements et connexions

Connexions entre la maquette et la carte ARDUINO UNO

Feu rouge (fil rouge) >>> Broche 19 = broche analogique A5 de la carte Arduino Uno
Diode verte n°1 (fil vert) >>> Broche 10 de la carte Arduino Uno
Diode verte n°2 (fil jaune) >>> Broche 2 de la carte Arduino Uno
Diode verte n°3 (fil bleu) >>> Broche 3 de la carte Arduino Uno
Diode verte n°4 (fil orangé) >>> Broche 4 de la carte Arduino Uno

Diode verte n°5 (fil violet) >>> Broche 5 de la carte Arduino Uno
Diode verte n°6 (fil marron) >>> Broche 6 de la carte Arduino Uno
Diode verte n°7 (fil vert) >>> Broche 7 de la carte Arduino Uno
Diode verte n°8 (fil rouge) >>> Broche 8 de la carte Arduino Uno
Diode verte n°9 (fil jaune) >>> Broche 9 de la carte Arduino Uno
Diode verte n°10 (fil bleu) >>> Broche 11 de la carte Arduino Uno
Bouton Entrée (fil bleu) >>> Broche d'entrée analogique A0 de la carte Arduino Uno
Bouton Sortie (fil rouge) >>> Broche d'entrée analogique A1 de la carte Arduino Uno
BUZZER (fil rouge) >>> Broche 18 = broche analogique A4 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 10 diodes vertes, de la LED rouge, des boutons d'entrée et de sortie, du buzzer (fil noir) sont toutes reliées à ce fil noir par des connexions situées sous le socle de la maquette - Soudures)

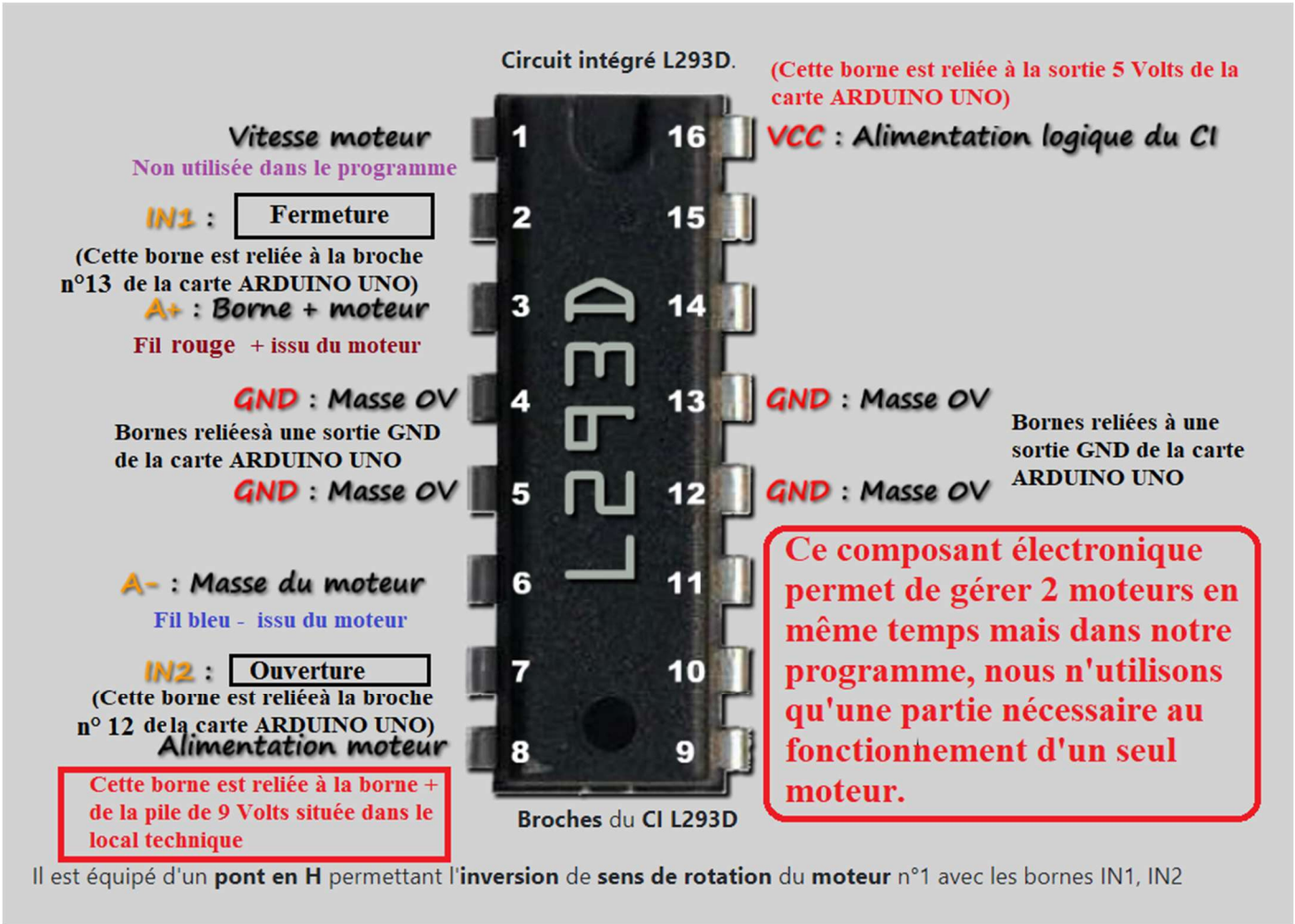
Connexion entre le moteur et le circuit L293D

Fil rouge (+) >>> Borne 3 du circuit intégré L293D (fil marron sur la maquette)
Fil bleu (-) >>> Borne 6 du circuit intégré L293D (fil vert sur la maquette)

Connexions entre le circuit L293D et la carte ARDUINO UNO

Sortie 5 Volts de la carte reliée à la borne 16 de L293D
Broche n° 12 de la carte reliée à la borne 2 de L293 (Ouverture de la barrière)
Broche n° 13 de la carte reliée à la borne 7 de L293D (Fermeture de la barrière)
*(NB – La broche n° 13 allumera **la diode n° 13 intégrée à la carte Arduino Uno** pendant la fermeture…)*
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 la ligne GND – de la plaque de connexions et les bornes 4 et 5 et 12 et 13 de L293D



Détail des connexions du circuit intégré « L293D » avec « GestionParking3.B4R»

Pour illustrer toutes ces connexions, vous trouverez ci-dessous un schéma dessiné grâce à l'application « FRITZING » permettant de comprendre le fonctionnement du programme simulé ici sur une plaque de connexions sans soudures.

Les connexions de la maquette sont basées sur ce schéma sachant que les LEDS, les boutons, le buzzer et le moteur se trouvent sur la maquette elle-même. Il ne reste que le composant L293D sur la plaque, rappelons que c'est ce circuit intégré qui permet l'inversion du sens de rotation du moteur.

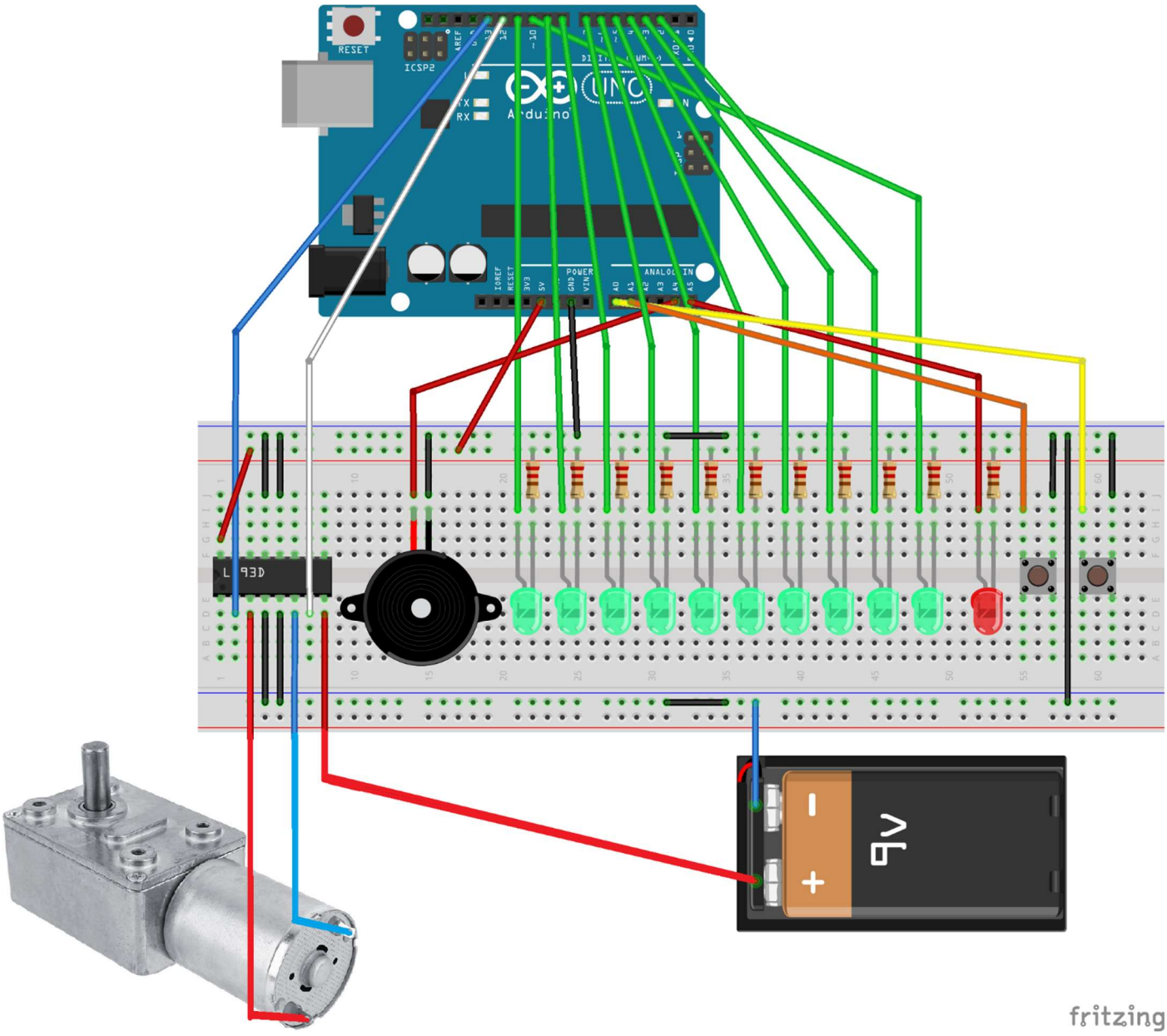
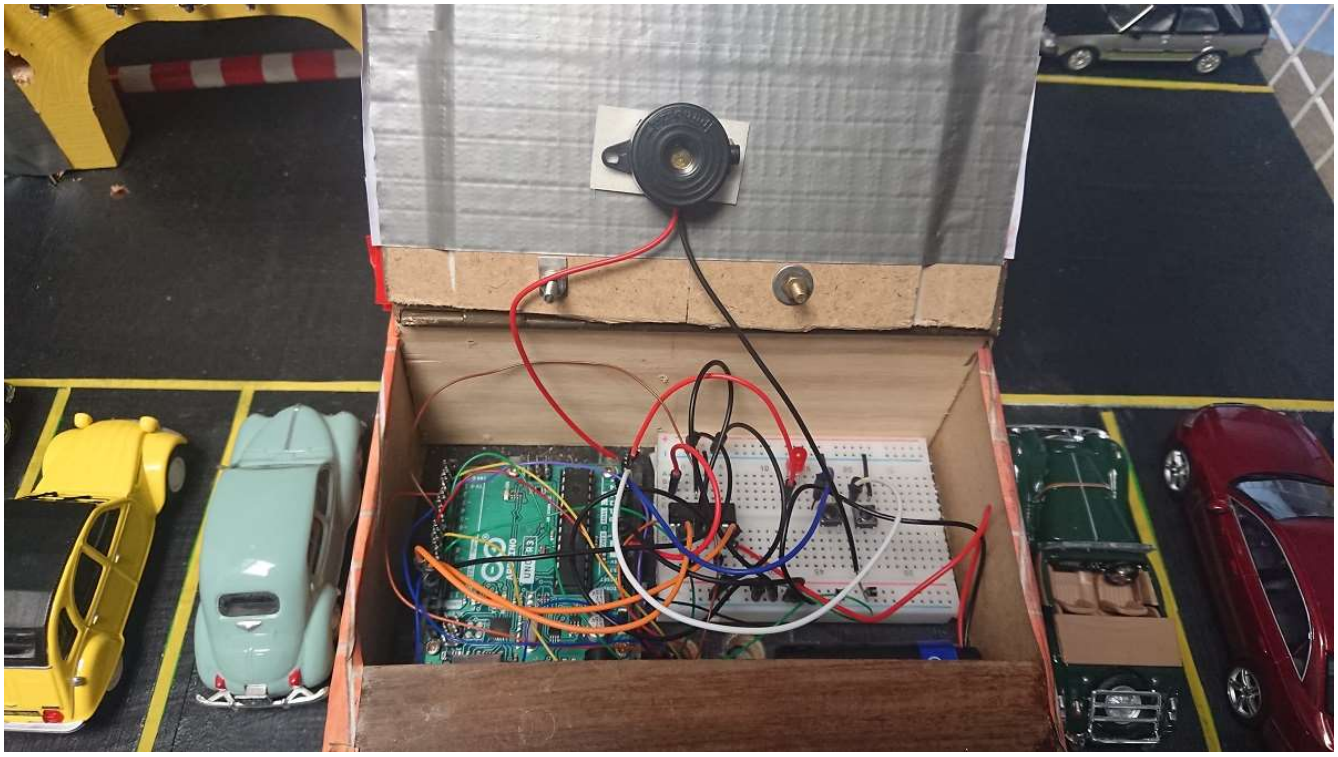


Schéma des connexions correspondantes au parking virtuel géré par le programme B4R



Connexions installées dans le local technique sur le parking

Parking vide >>> Toutes les LEDs vertes sont allumées

Maquette vue de dessus et vue latérale

Programme GestionParking3.B4R

```
' GestionParking3.B4R
#Region Project Attributes
 #AutoFlushLogs: True
#CheckArrayBounds: True
 #StackBufferSize: 300
#End Region

' GESTION D'UN PARKING VIRTUEL DE 10 PLACES VIDE AU DÉPART >>> 10 places sont libres >>> 10 LEDs vertes sont allumées
' Marc DANIEL - via CARTE ARDUINO UNO complétée par l'utilisation du composant L293D - Mars 2021
' Maquette de Parking avec moteur + réducteur et ouverture directe de la barrière

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 pinBuzzer As Pin 'broche pour le buzzer
Private pinLED1, pinLED2, pinLED3, pinLED4, pinLED5, pinLED6, pinLED7, pinLED8,pinLED9, pinLED10, pinLEDrouge As Pin
                'broches pour les LEDs
Public LED1 = False As Boolean
Public LED2 = False As Boolean
Public LED3 = False As Boolean
Public LED4 = False As Boolean
Public LED5 = False As Boolean
Public LED6 = False As Boolean
Public LED7 = False As Boolean
Public LED8 = False As Boolean
Public LED9 = False As Boolean
Public LED10 = False As Boolean
Public LEDrouge = False As Boolean
Private pinOuverture, pinFermeture As Pin 'broches de sorties pour les connexions motorisation barrière
Public Places As UInt
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")
pinLED1.Initialize(10, pinLED10.MODE_OUTPUT)
pinLED2.Initialize(2, pinLED2.MODE_OUTPUT)
pinLED3.Initialize(3, pinLED3.MODE_OUTPUT)
pinLED4.Initialize(4, pinLED4.MODE_OUTPUT)
pinLED5.Initialize (5, pinLED5.MODE_OUTPUT)
pinLED6.Initialize(6, pinLED6.MODE_OUTPUT)
pinLED7.Initialize(7, pinLED7.MODE_OUTPUT)
pinLED8.Initialize(8, pinLED8.MODE_OUTPUT)
pinLED9.Initialize (9, pinLED9.MODE_OUTPUT)
pinLED10.Initialize(11, pinLED10.MODE_OUTPUT)
pinLEDrouge.Initialize(19, pinLEDrouge.MODE_OUTPUT) 'Broche analogique A5
pinBuzzer.Initialize(18, pinBuzzer.MODE_OUTPUT) 'Broche analogique A4
pinOuverture.Initialize(12, pinOuverture.MODE_OUTPUT)
        'Connexion au composant L293D borne "IN2" pour l'ouverture de la barrière
pinFermeture.Initialize (13, pinFermeture.MODE_OUTPUT)
         'Connexion au composant L293D borne "IN1" pour la fermeture de la barrière (et allumage de la diode 13 Arduino)
Places=10 '10 places de parking sont libres au démarrage - Le parking est vide
CallSubPlus("Depart", 0,0)

End Sub

Private Sub Depart ' ILy a 10 places de libres - les 10 LEDs vertes sont allumées au départ
pinLED1.DigitalWrite(True)
pinLED2.DigitalWrite(True)
pinLED3.DigitalWrite(True)
pinLED4.DigitalWrite(True)
```

```
 64 pinLED5.DigitalWrite(True)
 65 pinLED6.DigitalWrite(True)
 66 pinLED7.DigitalWrite(True)
 67 pinLED8.DigitalWrite(True)
 68 pinLED9.DigitalWrite(True)
 69 pinLED10.DigitalWrite(True)
 70 End Sub
 71
 72 Private Sub pinButtonEntree_StateChanged(State1 As Boolean)
 73 Log("État: ", State1) 'Log la valeur de State1
 74 If State1 = False Then
 75 If Places = 0 Then CallSubPlus("GestionPlaces", 0, 0)
 76 If Places > 0 Then
 77 Places = Places -1
 78 CallSubPlus("Buzzer",0,0)
 79 CallSubPlus("GestionPlaces",0,0)
 80 CallSubPlus("Ouverture",500,0) ' Ouverture de la barrière
 81 CallSubPlus("Pause",5150,0) 'Mouvement du véhicule
 82 CallSubPlus("Fermeture", 10300, 0) ' Fermeture de la barrière
 83 CallSubPlus("FinEntree",15000,0) ' Fin de la fermeture de la barrière
 84 End If
 85 End If
 86 End Sub
 87
 88 Private Sub pinButtonSortie_StateChanged(State2 As Boolean)
 89 Log("État: ", State2) 'Log la valeur de State2
 90 If State2 = False Then
 91 If Places = 10 Then CallSubPlus("GestionPlaces",0, 0)
 92 If Places < 10 Then
 93 Places=Places + 1
 94 CallSubPlus("GestionPlaces",0,0)
 95 CallSubPlus("Ouverture",200,0) ' Ouverture de la barrière
 96 CallSubPlus("Pause",4850,0) 'Mouvement du véhicule
 97 CallSubPlus("Fermeture", 10000, 0) ' Fermeture de la barrière
 98 CallSubPlus("FinSortie",14700,0) ' Fin de la fermeture de la barrière
 99 End If
100 End If
101 End Sub
102
103 Private Sub Ouverture(Tag As Byte)
104 pinBuzzer.DigitalWrite(False)
105 pinOuverture.DigitalWrite(True) ' ouvre la barrière pour entrée ou sortie d'un véhicule
106 End Sub
107
108 Private Sub Pause(Tag As Byte)
109 pinOuverture.DigitalWrite(False) 'La barrière reste ouverte - Mouvement du véhicule entrant ou sortant
110 End Sub
111
112 Private Sub Fermeture(Tag As Byte)
113 pinFermeture.DigitalWrite(True) 'fermeture de la barrière
114 End Sub
115
116 Private Sub FinEntree(Tag As Byte)
117 pinFermeture.DigitalWrite(False) 'Arrêt de la fermeture de la barrière
118
119 End Sub
120
121 Private Sub FinSortie(Tag As Byte)
122 pinFermeture.DigitalWrite(False) 'Arrêt de la fermeture de la barrière
123 End Sub
124
125 Private Sub Buzzer(Tag As Byte)
126 pinBuzzer.DigitalWrite(True)
127 End Sub
128
129
130 Private Sub GestionPlaces
131 Select Places
132 Case 0
133 pinLED1.DigitalWrite(False)
```

```
134 pinLED2.DigitalWrite(False)
135 pinLED3.DigitalWrite(False)
136 pinLED4.DigitalWrite(False)
137 pinLED5.DigitalWrite(False)
138 pinLED6.DigitalWrite(False)
139 pinLED7.DigitalWrite(False)
140 pinLED8.DigitalWrite(False)
141 pinLED9.DigitalWrite(False)
142 pinLED10.DigitalWrite(False)
143 pinLEDrouge.DigitalWrite(True) 'allume la LED rouge (plus de places)
144 Case 1
145 pinLEDrouge.DigitalWrite(False) 'éteint la LED Rouge
146 pinLED1.DigitalWrite(True) ' Une place est libre
147 pinLED2.DigitalWrite(False)
148 pinLED3.DigitalWrite(False)
149 pinLED4.DigitalWrite(False)
150 pinLED5.DigitalWrite(False)
151 pinLED6.DigitalWrite(False)
152 pinLED7.DigitalWrite(False)
153 pinLED8.DigitalWrite(False)
154 pinLED9.DigitalWrite(False)
155 pinLED10.DigitalWrite(False)
156 Case 2
157 pinLED1.DigitalWrite(True)
158 pinLED2.DigitalWrite(True)
159 pinLED3.DigitalWrite(False)
160 pinLED4.DigitalWrite(False)
161 pinLED5.DigitalWrite(False)
162 pinLED6.DigitalWrite(False)
163 pinLED7.DigitalWrite(False)
164 pinLED8.DigitalWrite(False)
165 pinLED9.DigitalWrite(False)
166 pinLED10.DigitalWrite(False)
167 Case 3
168 pinLED1.DigitalWrite(True)
169 pinLED2.DigitalWrite(True)
170 pinLED3.DigitalWrite(True)
171 pinLED4.DigitalWrite(False)
172 pinLED5.DigitalWrite(False)
173 pinLED6.DigitalWrite(False)
174 pinLED7.DigitalWrite(False)
175 pinLED8.DigitalWrite(False)
176 pinLED9.DigitalWrite(False)
177 pinLED10.DigitalWrite(False)
178 Case 4
179 pinLED1.DigitalWrite(True)
180 pinLED2.DigitalWrite(True)
181 pinLED3.DigitalWrite(True)
182 pinLED4.DigitalWrite(True)
183 pinLED5.DigitalWrite(False)
184 pinLED6.DigitalWrite(False)
185 pinLED7.DigitalWrite(False)
186 pinLED8.DigitalWrite(False)
187 pinLED9.DigitalWrite(False)
188 pinLED10.DigitalWrite(False)
189 Case 5
190 pinLED1.DigitalWrite(True)
191 pinLED2.DigitalWrite(True)
192 pinLED3.DigitalWrite(True)
193 pinLED4.DigitalWrite(True)
194 pinLED5.DigitalWrite(True)
195 pinLED6.DigitalWrite(False)
196 pinLED7.DigitalWrite(False)
197 pinLED8.DigitalWrite(False)
198 pinLED9.DigitalWrite(False)
199 pinLED10.DigitalWrite(False)
200 Case 6
201 pinLED1.DigitalWrite(True)
202 pinLED2.DigitalWrite(True)
203 pinLED3.DigitalWrite(True)
```

```
pinLED4.DigitalWrite(True)
pinLED5.DigitalWrite(True)
pinLED6.DigitalWrite(True)
pinLED7.DigitalWrite(False)
pinLED8.DigitalWrite(False)
pinLED9.DigitalWrite(False)
pinLED10.DigitalWrite(False)
Case 7
pinLED1.DigitalWrite(True)
pinLED2.DigitalWrite(True)
pinLED3.DigitalWrite(True)
pinLED4.DigitalWrite(True)
pinLED5.DigitalWrite(True)
pinLED6.DigitalWrite(True)
pinLED7.DigitalWrite(True)
pinLED8.DigitalWrite(False)
pinLED9.DigitalWrite(False)
pinLED10.DigitalWrite(False)
Case 8
pinLED1.DigitalWrite(True)
pinLED2.DigitalWrite(True)
pinLED3.DigitalWrite(True)
pinLED4.DigitalWrite(True)
pinLED5.DigitalWrite(True)
pinLED6.DigitalWrite(True)
pinLED7.DigitalWrite(True)
pinLED8.DigitalWrite(True)
pinLED9.DigitalWrite(False)
pinLED10.DigitalWrite(False)
Case 9
pinLED1.DigitalWrite(True)
pinLED2.DigitalWrite(True)
pinLED3.DigitalWrite(True)
pinLED4.DigitalWrite(True)
pinLED5.DigitalWrite(True)
pinLED6.DigitalWrite(True)
pinLED7.DigitalWrite(True)
pinLED8.DigitalWrite(True)
pinLED9.DigitalWrite(True)
pinLED10.DigitalWrite(False)
Case 10
pinLED1.DigitalWrite(True)
pinLED2.DigitalWrite(True)
pinLED3.DigitalWrite(True)
pinLED4.DigitalWrite(True)
pinLED5.DigitalWrite(True)
pinLED6.DigitalWrite(True)
pinLED7.DigitalWrite(True)
pinLED8.DigitalWrite(True)
pinLED9.DigitalWrite(True)
pinLED10.DigitalWrite(True)
End Select
End Sub
```

-:-

Marc DANIEL – Mars 2021

Ce programme GestionParking3.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