Université de Vincennes Saint-Denis
Master 1 Informatique

Evaluation en :

Micro-Controlleurs et micro-Processeurs

Projet réalisé par :

M^{elle}.Liza BELKACEM

Année 2019-2020

6 mai 2020

0.1 Idée

J'ai eu l'idée de mettre en place un système de supervision, comme on pourrait en retrouver dans un milieu industriel ou dans les maisons intelligentes. Le but sera d'afficher des informations sur l'écran LCD en fonction d'évènements qui se passent dans le milieu extérieur.

0.2 Représentation

-Deux boutons, qui pourraient représenter par exemple deux barrières infrarouges et dont le signal reçu passe de 1 à 0 lorsqu'un objet passe devant.

-Deux potentiomètres. Le premier sert de "consigne" et est réglé par l'utilisateur. Le second représentera un capteur. À titre d'exemple, sur la vidéo à la suite vous verrez un potentiomètre rotatif qui représentera la consigne et un autre sous forme de glissière qui sera le capteur.

-Une LED rouge, nous permettra de faire une alarme visuelle. Elle sera normalement éteinte mais si la valeur du capteur dépasse celle de la consigne alors elle s'allumera.

Ce tableau résume la liste des composants :

Nom	Quantité	Composant
U1	1	Arduino Uno R3
U2	1	Écran LCD 16x2
Rpot1 Rpot2 Rpot3	3	250 kΩ, Potentiomètre
R1	1	220 Ω Résistance
D1	1	Rouge LED
R2	1	1 kΩ Résistance
S1 S2	2	Bouton poussoir

0.3 Comportement de l'écran

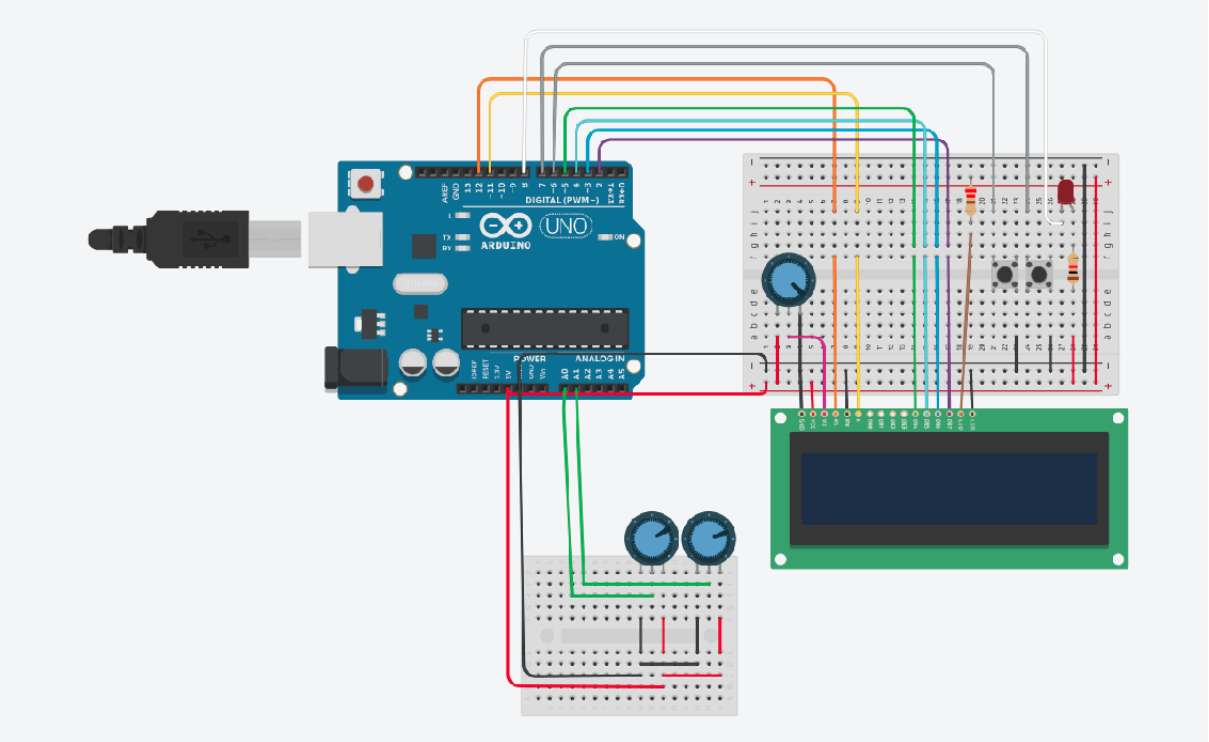
Chaque interface sera affichée pendant cinq secondes à tour de rôle. La première affichera l'état des boutons. Enfin, bien que l'information "consigne/capteur" ne s'affiche que toutes les 5 secondes, l'alarme (la LED rouge), elle, est visible à tout moment si la valeur du capteur dépasse celle de la consigne.

0.4 Montage

J'ai vaillé à satisfaire ces trois conditions :
-Des condensateurs de filtrage pour éviter les rebonds parasites créés par les boutons.
-Mettre les potentiomètres sur des entrées analogiques.
-Brancher la LED dans le bon sens et ne pas oublier sa résistance de limitation de courant.

J'ai fais la simulation sur tinkercad voici le lien :
https ://www.tinkercad.com/things/krCowrN62yz-mon-projet
Voici le lien du montage que j'ai effectué sur Tincerkard :

Et voici une image pour récapituler :



0.5 Code

0.5.1 Les variables

-5 pour les entrées/sorties (2 boutons, 2 potentiomètres, 1 LED)

-2 tableaux pour contenir et préparer les messages à afficher sur la première et deuxième ligne.

-4 pour contenir les mesures faites et 4 autres servant de mémoire pour ces mesures.

-1variable contenant le temps écoulé et une servant à savoir sur quelle "interface" nous sommes en train d'écrire.
Ce tableau résume toutes les variables avec leurs types :

Nom	Type	Description
boutonGauche	`const int`	Broche du bouton de gauche
boutonDroite	`const int`	Broche du bouton de droite
potentiometreGauche	`const int`	Broche du potar "consigne"
potentiometreDroite	`const int`	Broche du potar "alarme"
ledAlarme	`const int`	Broche de la LED d'alarme
messageHaut[16]	`char`	Tableau représentant la ligne du haut
messageBas[16]	`char`	Tableau représentant la ligne du bas
etatGauche	`int`	État du bouton de gauche
etatDroite	`int`	État du bouton de droite
niveauGauche	`int`	Conversion du potar de gauche
niveauDroite	`int`	Conversion du potar de droite
etatGauche_old	`int`	Mémoire de l'état du bouton de gauche
etatDroite_old	`int`	Mémoire de l'état du bouton de droite
niveauGauche_old	`int`	Mémoire de la conversion du potar de gauche
niveauDroite_old	`int`	Mémoire de la conversion du potar de droite
temps	`unsigned long`	Pour mémoriser le temps écoulé
ecran	`boolean`	Pour savoir sur quelle interface on écrit

0.5.2 Setup

Le setup n'aura que peu de choses à faire puisqu'il suffira de régler les broches en entrées/-sorties et de mettre en marche l'écran LCD.
Code :

```
1  void setup() {
2     // règlage des entrées/sorties
3     pinMode(boutonGauche, INPUT);
4     pinMode(boutonDroite, INPUT);
5     digitalWrite(boutonGauche, HIGH);
6     digitalWrite(boutonDroite, HIGH);
7     pinMode(ledAlarme, OUTPUT);
8     digitalWrite(ledAlarme, HIGH);
9
10    // réglage du LCD
11    lcd.begin(16, 2); // règle la  taille du LCD
12    lcd.noBlink(); // pas de clignotement
13    lcd.noCursor(); // pas de curseur
14    lcd.noAutoscroll(); // pas de défilement
15 }
```

Listing 1 – code Setup

0.5.3 recupererDonnees

elles sera chargée de faire le relevé des valeurs. Son objectif sera de faire les conversions analogiques et de regarder l'état des entrées numériques. Elle stockera bien entendu chacune des mesures dans la variable concernée.
Code :

```
  void recupererDonnees()
{
    // efface les anciens avec les "nouveaux anciens"
    etatGauche_old = etatGauche;
    etatDroite_old = etatDroite;
    niveauGauche_old = niveauGauche;
    niveauDroite_old = niveauDroite;

    // effectue les mesures
    etatGauche = digitalRead(boutonGauche);
    etatDroite = digitalRead(boutonDroite);
    niveauGauche = analogRead(potentiometreGauche);
    niveauDroite = analogRead(potentiometreDroite);

    // pour s'assurer que les conversions analogiques sont terminées
    // avant de passer à la suite on fait une petite pause
    delay(2);
}
```

Listing 2 – code recupererDonnes()

0.5.4 boutonsChanged() et potarChanged()

Ces deux fonctions vont nous permettre de déterminer si oui ou non il faut mettre à jour l'écran. En effet, afin d'éviter un phénomène de scintillement qui se produit si on envoi des données sans arrêt, on préfère écrire sur l'écran que si nécessaire. Pour décider si l'on doit mettre à jour les "phrases" concernant les boutons, il suffit de vérifier l'état "ancien" et l'état courant de chaque bouton. Si l'état est différent, notre fonction renvoie true, sinon elle renvoie false. Une même fonction sera codée pour les valeurs analogiques. Cependant, comme les valeurs lues par le convertisseur de la carte Arduino ne sont pas toujours très stables (je rappelle que le convertisseur offre plus ou moins deux bits de précision, soit 20mV de précision totale), on va faire une petite opération. Cette opération consiste à regarder si la valeur absolue de la différence entre la valeur courante et la valeur ancienne est supérieure à deux unités. Si c'est le cas, on renvoi true, sinon false.
Code :

```
boolean boutonsChanged()
{
    // si un bouton à changé d'état
    if(etatGauche_old != etatGauche || etatDroite_old != etatDroite)
        return true;
    else
        return false;
}

boolean potarChanged()
{
```

```
12      // si un potentiomètre affiche une différence de plus de 2 unités
13      // entre ces deux valeurs, alors on met à jour
14      if(abs(niveauGauche_old-niveauGauche) > 2 ||
15         abs(niveauDroite_old-niveauDroite) > 2)
16      {
17          return true;
18      }
19      else
20      {
21          return false;
22      }
23  }
```

Listing 3 – code boutonsChanged() et portarChanged

0.5.5 updateEcran()

Une dernière fonction nous servira à faire la mise à jour de l'écran. Elle va préparer les deux chaines de caractères (celle du haut et celle du bas) et va ensuite les envoyer successivement sur l'écran. Pour écrire dans les chaines, on vérifiera la valeur de la variable ecran pour savoir si on doit écrire les valeurs des potentiomètres ou celles des boutons. L'envoi à l'écran se fait simplement avec print(). On notera le clear() de l'écran avant de faire les mises à jour. En effet, sans cela les valeurs pourraient se chevaucher (si on écrit un on OFF puis un ON sans clear() cela affichera un ONF)
Code :

```
1   void updateEcran()
2   {
3       if(ecran)
4       {
5           // prépare les chaines à mettre sur l'écran : boutons
6           if(etatGauche)
7               sprintf(messageHaut,"Bouton G : ON");
8           else
9               sprintf(messageHaut,"Bouton G : OFF");
10          if(etatDroite)
11              sprintf(messageBas,"Bouton D : ON");
12          else
13              sprintf(messageBas,"Bouton D : OFF");
14          }
15          else
16          {
17              // prépare les chaines à mettre sur l'écran : potentiomètres
18              sprintf(messageHaut,"gauche = %4d", niveauGauche);
19              sprintf(messageBas,"droite = %4d", niveauDroite);
20      }
21
22      // on envoie le texte
23      lcd.clear();
24      lcd.setCursor(0,0);
25      lcd.print(messageHaut);
26      lcd.setCursor(0,1);
27      lcd.print(messageBas);
28  }
```

Listing 4 – code updateEcran()

0.5.6 La boucle principale

elle est relativement légère, grâce aux fonctions permettant de repartir le code en unité logique. La boucle principale n'a plus qu'à les utiliser à bon escient et dans le bon ordre pour faire son travail. Dans l'ordre il nous faudra donc :

-Récupérer toutes les données (faire les conversions, etc.).

-Selon l'interface courante, afficher soit les états des boutons soit les valeurs des potentiomètres si ils/elles ont changé(e)s.

-Tester les valeurs des potentiomètres pour déclencher l'alarme ou non.

-si 5 secondes se sont écoulées, changer d'interface et mettre à jour l'écran.

code :

```
1  void loop() {
2
3      // commence par récupérer les données des boutons et capteurs
4      recupererDonnees();
5
6      if(ecran) // quel écran affiche t'on ? (bouton ou potentiomètre ?)
7      {
8          if(boutonsChanged()) // si un bouton a changé d'état
9          updateEcran();
10     }
11     else
12     {
13         if(potarChanged()) // si un potentiomètre a changé d'état
14         updateEcran();
15     }
16
17     if(niveauDroite > niveauGauche)
18         // RAPPEL : piloté à l'état bas donc on allume !
19         digitalWrite(ledAlarme, LOW);
20     else
21         digitalWrite(ledAlarme, HIGH);
22
23     // si ça fait 5s qu'on affiche la même donnée
24     if(millis() - temps > 5000)
25     {
26         ecran = ~ecran;
27         lcd.clear();
28         updateEcran();
29         temps = millis();
30     }
31 }
```

Listing 5 – code de la boucle principale

0.6 code complet

code complet

```
1
2  #include "LiquidCrystal.h" // on inclut la librairie
3
4  // les branchements
5  const int boutonGauche = 11; // le bouton de gauche
6  const int boutonDroite = 12; // le bouton de droite
7  const int potentiometreGauche = 0; // le potentiomètre de gauche (analogique 0)
8  const int potentiometreDroite = 1; // le potentiomètre de droite (analogique 1)
9  const int ledAlarme = 2; // la LED est branché sur la sortie 2
10
11 // initialise l'écran avec les bonnes broches
12 // ATTENTION, REMPLACER LES NOMBRES PAR VOS BRANCHEMENTS    VOUS !
13 LiquidCrystal lcd(11,10,5,4,3,2);
14
15 char messageHaut[16] = ""; // Message sur la ligne du dessus
16 char messageBas[16] = ""; // Message sur la ligne du dessous
17
18 unsigned long temps = 0; // pour garder une trace du temps qui s'écoule
19 boolean ecran = LOW; // savoir si on affiche les boutons ou les conversions
20
21 int etatGauche = LOW; // état du bouton de gauche
22 int etatDroite = LOW; // état du bouton de droite
23 int niveauGauche = 0; // conversion du potentiomètre de gauche
24 int niveauDroite = 0; // conversion du potentiomètre de droite
25
26 // les mêmes variables mais "old"
27 // servant de mémoire pour constater un changement
28 int etatGauche_old = LOW; // état du bouton de gauche
29 int etatDroite_old = LOW; // état du bouton de droite
30 int niveauGauche_old = 0; // conversion du potentiomètre de gauche
31 int niveauDroite_old = 0; // conversion du potentiomètre de droite
32
33 // ----------------------------------------------------------------------
34
35 void setup() {
36     // réglage des entrées/sorties
37     pinMode(boutonGauche, INPUT);
38     pinMode(boutonDroite, INPUT);
39     digitalWrite(boutonGauche, HIGH);
40     digitalWrite(boutonDroite, HIGH);
41     pinMode(ledAlarme, OUTPUT);
42     digitalWrite(ledAlarme, HIGH);
43
44     // paramétrage du LCD
45     lcd.begin(16, 2); // règle la  taille du LCD
46     lcd.noBlink(); // pas de clignotement
47     lcd.noCursor(); // pas de curseur
48     lcd.noAutoscroll(); // pas de défilement
49 }
50
51 void loop() {
52
53     // commence par récupérer les données des boutons et capteurs
54     recupererDonnees();
55
56     if(ecran) // quel écran affiche-t'on ? (bouton ou potentiomètre ?)
57     {
58         if(boutonsChanged()) // si un bouton a changé d'état
59             updateEcran();
```

```
60      }
61      else
62      {
63          if(potarChanged()) // si un potentiomètre a changé d'état
64              updateEcran();
65      }
66
67      if(niveauDroite > niveauGauche)
68          // RAPPEL : piloté à l'état bas donc on allume !
69          digitalWrite(ledAlarme, LOW);
70      else
71          digitalWrite(ledAlarme, HIGH);
72
73      // si ça fait 5s qu'on affiche la même donnée
74      if(millis() - temps > 5000)
75      {
76          ecran = ~ecran;
77          lcd.clear();
78          updateEcran();
79          temps = millis();
80      }
81  }
82
83  // ----------------------------------------------------------------------
84
85  void recupererDonnees()
86  {
87      // efface les anciens avec les "nouveaux anciens"
88      etatGauche_old = etatGauche;
89      etatDroite_old = etatDroite;
90      niveauGauche_old = niveauGauche;
91      niveauDroite_old = niveauDroite;
92
93      etatGauche = digitalRead(boutonGauche);
94      etatDroite = digitalRead(boutonDroite);
95      niveauGauche = analogRead(potentiometreGauche);
96      niveauDroite = analogRead(potentiometreDroite);
97
98      // pour s'assurer que les conversions analogiques sont terminées
99      // on fait une petite pause avant de passer à la suite
100     delay(1);
101 }
102
103 boolean boutonsChanged()
104 {
105     if(etatGauche_old != etatGauche || etatDroite_old != etatDroite)
106         return true;
107     else
108         return false;
109 }
110
111 boolean potarChanged()
112 {
113     // si un potentiomètre affiche une différence de plus de 2 unités
114     // entre ces deux valeurs, alors on met à jour
115     if(abs(niveauGauche_old-niveauGauche) > 2 ||
116        abs(niveauDroite_old-niveauDroite) > 2)
117     {
118         return true;
119     }
```

```
120     else
121     {
122         return false;
123     }
124 }
125
126 void updateEcran()
127 {
128     if(ecran)
129     {
130         // prépare les chaines à mettre sur l'écran
131         if(etatGauche)
132             sprintf(messageHaut,"Bouton G : ON");
133         else
134             sprintf(messageHaut,"Bouton G : OFF");
135         if(etatDroite)
136             sprintf(messageBas,"Bouton D : ON");
137         else
138             sprintf(messageBas,"Bouton D : OFF");
139     }
140     else
141     {
142         // prépare les chaines à mettre sur l'écran
143         sprintf(messageHaut,"gauche = %4d", niveauGauche);
144         sprintf(messageBas,"droite = %4d", niveauDroite);
145     }
146
147     // on envoie le texte
148     lcd.clear();
149     lcd.setCursor(0,0);
150     lcd.print(messageHaut);
151     lcd.setCursor(0,1);
152     lcd.print(messageBas);
153 }
```

Listing 6 – code source complet