

Enseignes et afficheurs à LED

Langages interprétés spécialisés

Langages interprétés spécialisés

Pierre-Yves Rochat

- Motivation pour inventer un langage
- Instructions dans un tableau
- Interpréteur du langage
- Langages graphiques

Motivation pour inventer un langage

- Animer une enseigne à LED = suite d'opérations sur les groupes des LED
- Animer un afficheur matriciel = envoyer des séquences graphiques
- Les animations deviennent souvent longues
- Difficile à lire
- Beaucoup de place en mémoire
- **Inventer un langage**
- Décrire les animations dans ce langage

En Arduino

```

1 loop() {
2   digitalWrite (P2_0, 1); delay (100);
3   digitalWrite (P2_1, 1); delay (100);
4   digitalWrite (P2_2, 1); delay (100);
5   digitalWrite (P2_3, 1); delay (100);
6   digitalWrite (P2_4, 1); delay (100);
7   digitalWrite (P2_5, 1); delay (100);
8   digitalWrite (P2_6, 1); delay (100);
9   digitalWrite (P2_7, 1); delay (200);
10  digitalWrite (P2_7, 0); delay (100);
11  digitalWrite (P2_6, 0); delay (100);
12  digitalWrite (P2_5, 0); delay (100);
13  digitalWrite (P2_4, 0); delay (100);
14  digitalWrite (P2_3, 0); delay (100);
15  digitalWrite (P2_2, 0); delay (100);
16  digitalWrite (P2_1, 0); delay (100);
17  digitalWrite (P2_0, 0); delay (300);
18 }

```

- digitalWrite() : 8 octets
- delay() : 10 octets
- P10UT |= (1<<0); : 4 octets

Instructions dans un tableau

```

1 uint8_t Animation[] = {
2   Sortie0+0n, Attente+10,
3   Sortie1+0n, Attente+10,
4   Sortie2+0n, Attente+10,
5   Sortie3+0n, Attente+10,
6   Sortie4+0n, Attente+10,
7   Sortie5+0n, Attente+10,
8   Sortie6+0n, Attente+10,
9   Sortie7+0n, Attente+20,
10  Sortie7+0ff, Attente+10,
11  Sortie6+0ff, Attente+10,
12  Sortie5+0ff, Attente+10,
13  Sortie4+0ff, Attente+10,
14  Sortie3+0ff, Attente+10,
15  Sortie2+0ff, Attente+10,
16  Sortie1+0ff, Attente+10,
17  Sortie0+0ff, Attente+30,
18  Fin
19 }

```

- Mettre une *intentité*
- sur une *sortie*
- Attendre une certaine *durée*

```

1 #define On 0b01000000
2 #define Off 0b00000000
3 #define Sortie0 0
4 #define Sortie1 1
5 #define Sortie2 2
6 #define Sortie3 3
7 #define Sortie4 4
8 #define Sortie5 5
9 #define Sortie6 6
10 #define Sortie7 7
11 #define Attente 0b10000000
12 #define Fin 0b111111

```

Langage binaire

```

1 // Description des instructions :
2 // b7 b6 b5 b4 b3 b2 b1 b0 : instructions sur 8 bits
3 // -----
4 // 0 i0 s5-s4-s3-s2-s1-s0 : met une intensité sur une sortie
5 // 1 d6-d5-d4-d3-d2-d1-d0 : attente
6 // -----
7 //
8 // Sorties sur 6 bits (maximum 64 sorties)
9 // Intensité sur 1 bit (On ou OFF)
10
11 // Durée sur 7 bits, exprimée en dixième de seconde (0 à 12.6 secondes)
    }
1 #define On 0b01000000
2 #define Off 0b00000000
3 #define Sortie0 0
4 #define ...
5 #define Attente 0b10000000
6 #define Fin 0b111111
  
```

Interpréteur

```
1 void Exec () {
2     uint8_t instr = Programme[pc++]; // lit l'instruction
3     if (instr==Fin) { // gère la fin du programme
4         pc = 0;
5     } else {
6         if (instr & 0x8000) { // attente
7             AttenteMs(10 * (instr & 0x7F));
8         } else { // set intensité
9             if (instr & 0x40) {
10                Allume(instr & 0x3F);
11            } else {
12                Eteint(instr & 0x3F);
13            }
14        }
15    }
16 }
```

Langages plus complexes

- Gestion de l'intensité des LED par BCM (*Binary Coded Modulation*)
- Agir sur des groupes de LED
- Programmes en parallèle

Langage graphique

```

1  #define DrH 0x30 // + dx (4 bits) : droite horizontale depuis le curseur
2  #define DrV 0x40 // + dy (4 bits) : droite verticale, depuis le curseur
3  #define PlusX 0x50 // + dx (4 bits) : avance le curseur en X
4  #define PlusY 0x60 // + dy (4 bits) : avance le curseur en Y
5  #define MoinsX 0x70 // + dx (4 bits) : recule le curseur en X
6  #define MoinsY 0x80 // + dy (4 bits) : recule le curseur en Y
7  #define Repete 0x90 // + 4 bits : préfixe de répétition pour l'instr. suivante
8  #define Delai 0xA0 // + 4 bits : Attente, valeur exposant de 2
9  #define SetAccu 0xB0 // + 4 bits : Charge l'accumulateur (utilisé pour Intens)
10 #define Label 0xC0 // + 5 bits (32 routines max)
11 #define Call 0xE0 // + 5 bits
12 #define Fin 0 // fin du programme
  
```

Langage graphique

```

1 #define Vide 1 // efface l'écran
2 #define Ret 2 // retour de sous-routine (saut à l'adresse sur la pile)
3 #define Origine 3 // place le curseur à 0,0
4 #define ZeroX 4 // met à zéro X
5 #define Intens 5 // détermine l'intensité selon la valeur de l'accumulateur
6 #define Mask 0x9 //
7 #define InvMask 0xA // inverse le masque courant
8 #define SetDel 0xB // définit délai utilisé entre l'affichage de chaque point
9 #define SetDel0 0xC // définit la valeur du délai 0
10 #define Effet 0xD // Exécute un effet programmé
11 #define Libre1 0xE // instructions non utilisées
12 #define Libre2 0xF
  
```

Langages interprétés spécialisés

- Motivation pour inventer un langage
- Instructions dans un tableau
- Interpréteur du langage
- Langages graphiques