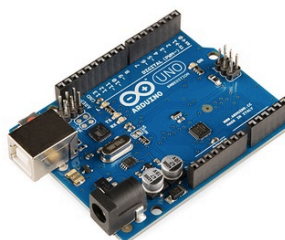


(<http://sciences-physiques.ac-besancon.fr>)
Jan 09 2020

Contrôler un microcontrôleur grâce à Python.

- Par Frédéric Guérinet (<http://sciences-physiques.ac-besancon.fr/author/fredadmin/>)

Ressource proposée par : JF. AMIOTTE-SUCHET (<mailto:jean-francois.amiotte-suchet@ac-besancon.fr>)



Il est également possible de

piloter la carte à microcontrôleur ArduinoTM en langage Python. On évite ainsi l'apprentissage d'un nouveau langage de programmation mais les fonctionnalités offertes sont réduites aux capacités proposées par la bibliothèque python.

Etape n°1

Copier le code ci-dessous dans un nouveau programme Python et nommer-le "*microcontrôleurs.py*". Ce programme doit être enregistré soit **dans le répertoire de travail**, soit dans le répertoire contenant les bibliothèques de la distribution Python utilisée (Ex : « lib » pour Edupython). Vous trouverez quelques explications sur les fonctions dans ce document (http://sciences-physiques.ac-besancon.fr/wp-content/uploads/sites/3/Ressources_pédagogiques/Numérique_Arduino/Controle_Python/memento-microcontrôleurs.pdf).

```

1.  #! encoding: utf-8
2.  import serial
3.
4.  class arduino():
5.      def __init__(self,port):
6.          self.serie = serial.Serial(port,baudrate=9600)
7.          synchro = ord(self.serie.read())
8.          while synchro != 0:
9.              synchro = ord(self.serie.read())
10.
11.
12.      def sortie_numerique(self,pin,etat):
13.          self.serie.write(chr(1).encode('latin-1'))
14.          self.serie.write(chr(pin).encode('latin-1'))
15.          self.serie.write(chr(etat).encode('latin-1'))
16.
17.      def entree_numerique(self,pin):
18.          self.serie.write(chr(2).encode('latin-1'))
19.          self.serie.write(chr(pin).encode('latin-1'))
20.          val=ord(self.serie.read())
21.          return val
22.
23.      def sortie_analogique(self,pin,val):
24.          self.serie.write(chr(3).encode('latin-1'))
25.          self.serie.write(chr(pin).encode('latin-1'))
26.          self.serie.write(chr(val).encode('latin-1'))
27.
28.      def entree_analogique(self,pin):
29.          self.serie.write(chr(4).encode('latin-1'))
30.          self.serie.write(chr(pin).encode('latin-1'))
31.          val1=ord(self.serie.read())
32.          val2=ord(self.serie.read())
33.          return val1*256 + val2
34.
35.      def son(self,pin,freq,duree=0):

```

```

36.         self.serie.write(chr(5).encode('latin-1'))
37.         self.serie.write(chr(pin).encode('latin-1'))
38.         self.serie.write(chr(freq>>8 & 255).encode('latin-1'))
39.         self.serie.write(chr(freq & 255).encode('latin-1'))
40.         self.serie.write(chr(int(duree*1000)>>8 & 255).encode('latin-1'))
41.         self.serie.write(chr(int(duree*1000) & 255).encode('latin-1'))
42.
43.     def module_us(self,echo,trig):
44.         self.serie.write(chr(6).encode('latin-1'))
45.         self.serie.write(chr(echo).encode('latin-1'))
46.         self.serie.write(chr(trig).encode('latin-1'))
47.         val1=ord(self.serie.read())
48.         val2=ord(self.serie.read())
49.         return val1*256 + val2
50.
51.     def resistance_pt100(self,cs,di,do,clk):
52.         self.serie.write(chr(7).encode('latin-1'))
53.         self.serie.write(chr(cs).encode('latin-1'))
54.         self.serie.write(chr(di).encode('latin-1'))
55.         self.serie.write(chr(do).encode('latin-1'))
56.         self.serie.write(chr(clk).encode('latin-1'))
57.         val1=ord(self.serie.read())
58.         val2=ord(self.serie.read())
59.         return 430*(val1*256 + val2)/32768
60.
61.     def fermer(self):
62.         self.serie.close()

```

Étape n°2

Copier le programme ci-dessous dans une nouveau programme Arduino, nommer le fichier “microcontrôleur.ino”.

Important!

La première ligne permet de charger la bibliothèque Adafruit_MAX31865. Rappelons que cette bibliothèque a été modifiée (téléchargement de la bibliothèque modifiée) (http://sciences-physiques.ac-besancon.fr/wp-content/uploads/sites/3/Ressources_pédagogiques/Numérique_Arduino/2nde_Pt100_2019/Adafruit_MAX31865_BEsançon.zip) pour inclure une procédure permettant d’obtenir aisément la valeur de la résistance de la sonde Pt100. L’ajout de cette procédure ne modifie en rien le fonctionnement classique de cette bibliothèque.

Notice

Pour l’installation de cette bibliothèque modifiée, il faut suivre la procédure classique : *Menu croquis / inclure une bibliothèque / ajouter la bibliothèque .ZIP*

Une fois la bibliothèque installée (à faire la première fois), téléverser le programme “microcontrôleur.ino” dans la carte avec le logiciel Arduino™ IDE.

```

1.     #include <Adafruit_MAX31865.h>
2.
3.     void sortie_numerique() {
4.         while (Serial.available() < 2) {
5.             }
6.         int pin = Serial.read();
7.         int etat = Serial.read();
8.         pinMode(pin, OUTPUT);
9.         if (etat == 1) {
10.            digitalWrite(pin, HIGH);
11.        } else {
12.            digitalWrite(pin, LOW);
13.        }
14.    }
15.
16.    void entree_numerique() {
17.        while (Serial.available() < 1) {
18.            }
19.        int pin = Serial.read();
20.        pinMode(pin, INPUT);
21.        Serial.write(digitalRead(pin));
22.    }

```

```

23.
24. void sortie_analogique() {
25.     while (Serial.available() < 2) {
26.     }
27.     int pin = Serial.read();
28.     int val = Serial.read();
29.     pinMode(pin, OUTPUT);
30.     analogWrite(pin, val);
31. }
32.
33. void entree_analogique() {
34.     while (Serial.available() < 1) {
35.     }
36.     int pin = Serial.read();
37.     int val = analogRead(pin);
38.     Serial.write((val>>8)&0xFF);
39.     Serial.write(val & 0xFF);
40. }
41.
42. void son() {
43.     while (Serial.available() < 5) {
44.     }
45.     int pin = Serial.read();
46.     int freq1 = Serial.read();
47.     int freq2 = Serial.read();
48.     int duree1 = Serial.read();
49.     int duree2 = Serial.read();
50.     unsigned int freq = 256*freq1 + freq2;
51.     unsigned int duree = 256*duree1 + duree2;
52.     if (freq == 0) {noTone(pin);}
53.     else if (duree == 0) {tone(pin,freq);}
54.     else {tone(pin,freq,duree);}
55. }
56.
57. void module_us() {
58.     while (Serial.available() < 2) {
59.     }
60.     int echo = Serial.read();
61.     int trig = Serial.read();
62.     // envoie une impulsion
63.     pinMode(trig, OUTPUT);
64.     digitalWrite(trig, LOW);
65.     delayMicroseconds(2);
66.     digitalWrite(trig, HIGH);
67.     delayMicroseconds(5);
68.     digitalWrite(trig, LOW);
69.
70.     // écoute de l'écho
71.     pinMode(echo, INPUT);
72.     unsigned int duree = pulseIn(echo, HIGH);
73.     Serial.write((duree>>8)&0xFF);
74.     Serial.write(duree & 0xFF);
75. }
76.
77. void pt100() {
78.     while (Serial.available() < 4) {
79.     }
80.     int cs = Serial.read();
81.     int di = Serial.read();
82.     int doo = Serial.read();
83.     int clk = Serial.read();
84.     Adafruit_MAX31865 max = Adafruit_MAX31865(cs,di,doo,clk);
85.     max.begin(MAX31865_2WIRE);
86.     unsigned int rtd = max.readRTD();
87.     Serial.write((rtd>>8)&0xFF);
88.     Serial.write(rtd & 0xFF);
89. }
90.
91. void setup() {
92.     Serial.begin(9600);
93.     Serial.flush();
94.     Serial.write(0);
95. }
96.
97. void loop() {
98.     if (Serial.available()) {
99.         int index = Serial.read();
100.        if (index == 1) {
101.            sortie_numerique();
102.        }

```

```

103.         else if (index == 2) {
104.             entree_numerique();
105.         }
106.         else if (index == 3) {
107.             sortie_analogique();
108.         }
109.         else if (index == 4) {
110.             entree_analogique();
111.         }
112.         else if (index == 5) {
113.             son();
114.         }
115.         else if (index == 6) {
116.             module_us();
117.         }
118.         else if (index == 7) {
119.             pt100();
120.         }
121.     }
122.
123. }

```

Étape n°3 :

Au début du programme Python, il faut importer la bibliothèque “microcontrolleurs” :

```
from microcontrolleurs import arduino
```

Puis il faut indiquer le port COM (le format dépend du système d’exploitation) sur lequel est reliée la carte à microcontrôleur Arduino™. Ce port est accessible depuis l’IDE Arduino ou le gestionnaire de périphérique.

Exemples :

```

macarte = arduino("COM3") ,

macarte = arduino("/dev/ttyACM1")

macarte = arduino("/dev/cu.usbmodem1421")

```