Arduino et Python

Disons le d'emblée ce billet ne vise qu'à montrer comment Arduino et Python peuvent communiquer. Il s'adresse aux plus téméraires et aux curieux! Vous trouverez sur le Web de multiples ressources à ce sujet. L'idée est simple: vous avez une version de Python disponible sur le réseau et vous désirez piloter l'Arduino. Voici comment faire. [Source: http://www.f-legrand.fr/scidoc/docmml/sciphys/arduino/python/python.html] Notons que ce code peut fonctionner à partir d'une clé USB! Seuls le driver Arduino et le logiciel Arduino (quoiqu'il existe un plan B ;)) devront être installés sur le poste.

Le code Arduino

```
#define PIN_MODE 100
#define DIGITAL_WRITE 101
#define DIGITAL_READ 102
#define ANALOG_WRITE 103
#define ANALOG_READ 104

void commande_pin_mode() {
    char pin,mode;
    while (Serial.available()<2);
    pin = Serial.read(); // pin number
    mode = Serial.read(); // 0 = INPUT, 1 = OUTPUT
    pinMode(pin,mode);
}

void commande_digital_write() {
   char pin,output;
   while (Serial.available()<2);
   pin = Serial.read(); // pin number
   output = Serial.read(); // 0 = LOW, 1 = HIGH
   digitalWrite(pin,output);
}

void commande_digital_read() {
   char pin,input;
   while (Serial.available()<1);
   pin = Serial.read(); // pin number
   input = digitalRead(pin);
   Serial.write(input);
}

void commande_analog_write() {
   char pin,output;
   while (Serial.available()<2);
   pin = Serial.read(); // pin number
   output = Serial.read(); // PWM value between 0 and 255
   analogWrite(pin,output);
}
```

```
void commande_analog_read() {
   char pin;
   int value;
   while (Serial.available()<1);
   pin = Serial.read(); // pin number
   value = analogRead(pin);
   Serial.write((value>>8)&0xFF); // 8 bits de poids fort
   Serial.write(value & 0xFF); // 8 bits de poids faible
}

void setup() {
  char c;
  Serial.begin(500000);
  Serial.flush();
  c = 0;
  Serial.write(c);
  c = 255;
  Serial.write(c);
  c = 0;
  Serial.write(c);
}

void loop() {
  char commande;
  if (Serial.available()>0) {
     commande = Serial.read();
     if (commande==PIN_MODE) commande_pin_mode();
     else if (commande==DIGITAL_WRITE) commande_digital_write();
     else if (commande==DIGITAL_READ) commande_digital_read();
     else if (commande==ANALOG_WRITE) commande_analog_write();
     else if (commande==ANALOG_READ) commande_analog_read();
  }
  // autres actions à placer ici
}
```

Le protocole d'échange est basé sur l'envoi de caractères en 8 bits sur la liaison série. Une séquence d'initialisation 0-255-0 permet d'indiquer que l'arduino est prête.

Il y a un code pour les différentes commandes.

```
#define PIN_MODE 100
#define DIGITAL_WRITE 101
#define DIGITAL_READ 102
#define ANALOG_WRITE 103
#define ANALOG_READ 104
```

A chaque commande un certain nombre de paramètres est requis. Pour écrire sur une sortie digitale on enverra trois caractères: 101 (pin) (output). Actionner la sortie 13: 101-13-1

Ce code vous permet d'ajouter des fonctionnalités comme des composants branchés en I2C avec des codes de commandes que vous pourrez ajouter facilement. L'on pourrait également ajouter la lecture

d'un capteur température/humidité DHT11/DHT22 (ce sera votre devoir de vacances ;)).

Une classe python est créée pour permettre un interfaçage facile.

commandesPython.py

```
# -*- coding: utf-8 -*-
import serial
class Arduino():
    def __init__(self,port):
        self.ser = serial.Serial(port,baudrate=500000)
        c_recu = self.ser.read(1)
        while ord(c_recu)!=0:
            c_recu = self.ser.read(1)
        c_recu = self.ser.read(1)
        while ord(c_recu)!=255:
            c_recu = self.ser.read(1)
        c_recu = self.ser.read(1)
        while ord(c_recu)!=0:
            c_recu = self.ser.read(1)
        self.PIN_MODE = 100
        self.DIGITAL_WRITE = 101
        self.DIGITAL_READ = 102
        self.ANALOG_WRITE = 103
        self.ANALOG_READ = 104
        self.INPUT = 0
        self.OUTPUT = 1
        self.LOW = 0
        self.HIGH = 1
    def close(self):
        self.ser.close()

    def pinMode(self,pin,mode):
        self.ser.write(chr(self.PIN_MODE).encode())
        self.ser.write(chr(pin).encode())
        self.ser.write(chr(mode).encode())

    def digitalWrite(self,pin,output):
        self.ser.write(chr(self.DIGITAL_WRITE).encode())
        self.ser.write(chr(pin).encode())
        self.ser.write(chr(output).encode())

    def digitalRead(self,pin):
        self.ser.write(chr(self.DIGITAL_READ).encode())
        self.ser.write(chr(pin).encode())
        x = self.ser.read(1)
        return ord(x)

    def analogWrite(self,pin,output):
        self.ser.write(chr(self.ANALOG_WRITE).encode())
        self.ser.write(chr(pin).encode())
```

```
        self.ser.write(chr(output).encode())

    def analogRead(self,pin):
        self.ser.write(chr(self.ANALOG_READ).encode())
        self.ser.write(chr(pin).encode())
        c1 = ord(self.ser.read(1))
        c2 = ord(self.ser.read(1))
        return c1*0x100+c2
```

fichier test.py

```
# -*- coding: utf-8 -*-
import time
from commandesPython import Arduino

port = "COM8"
ard = Arduino(port)
ard.pinMode(13,ard.OUTPUT)



for i in range(10):
    print(ard.analogRead(0))
    print(i)
    ard.digitalWrite(13,ard.HIGH)
    time.sleep(0.5)
    ard.digitalWrite(13,ard.LOW)
    time.sleep(0.5)
ard.close()
```

Si la commande en ligne de commande vous semble trop austère, il est possible d'ajouter un contexte fenêtré facilement.

```
# -*- coding: utf-8 -*-
import time
from commandesPython import Arduino
from tkinter import *

class App(Arduino):
    def __init__(self):
        self.port = "COM8"
        self.ard = Arduino(self.port)
        self.W=200
        self.H=200
        self.root = Tk()
        self.create_interface()
        self.update_clock()
        self.configure()
        self.root.mainloop()
```

```
    def send_arduino(self):
        self.ard.digitalWrite(13,self.ard.HIGH)
        time.sleep(0.25)
        self.ard.digitalWrite(13,self.ard.LOW)
        time.sleep(0.25)
    def create_interface(self):
        can = Canvas(self.root, width=self.W, height=self.H, bg='ivory')
        can.pack(side=TOP, padx= 5, pady= 5)
        btn1 = Button(self.root, text="Arduino", command =
self.send_arduino)
        btn1.pack(side=LEFT)
        self.text1=Label(self.root, text="A0: ", fg="red")
        self.text1.pack()

    def configure(self):
        self.ard.pinMode(13,self.ard.OUTPUT)
    def update_clock(self):
        now = time.strftime("%H:%M:%S")
        self.text1.configure(text= self.ard.analogRead(0))
        self.root.after(1000, self.update_clock)

app=App()
```