platformIO, l'Arduino sans Arduino

- Objet : Installation et utilisation de PlatformIO, un IDE pour développer l'embarqué
- Niveau requis :
  débutant
- Commentaires : Avec ça, on peu plus où moins choisir son éditeur de texte pour programmer l'Arduino. Il n'est plus nécessaire[1)] d'installer l'IDE Arduino, qui commence à dater sous Debian pour des histoires de licences.
- Débutant, à savoir : Utiliser GNU/Linux en ligne de commande, tout commence là !.

Introduction

platerformIO est un IDE[2)] qui permet de programmer de nombreuses cartes à microcontrôleurs et autres bêtes à poils raz.

On peut simplement installer le *coeur* du système *platformIOCore*, qui fournit des outils en ligne de commande pour simplifier les transferts et autres opérations vers les microcontôleurs.

C'est un programme en Python 2.7, non packagé pour Debian. Comme il utilise quelques modules dans des versions différentes de celles de Debian Stretch, on va utiliser un *Environnement virtuel* pour bien séparer tout ça, et l'installer via `pip`.

On verra ensuite[3)] pour l'intégration dans ce script dans un éditeur de texte, vim par exemple, puisque il fait parti de ceux supportés.

Installation

Installer virtualenv

Si vous ne l'avez pas encore, installons `virtualenv`

```
apt install python-virtualenv
```

Création et activation de l'environnement

On peut ensuite créer l'environnement virtuel dans l'emplacement de votre choix (`~/prog/arduino` par exemple), en demandant l'utilisation des bibliothèques systèmes avec l'option `–system-site-packages`. Ainsi, seront installées par `pip` seulement celles qui sont nécessaires. Ce n'est pas un obligation ceci-dit.

```
mkdir -p ~/prog/arduino
cd ~/prog/arduino
virtualenv platformio --system-site-packages
```

On peut ensuite activer l'environnement virtuel, *avant d'installer `platformIO`*.

Il faudra activer cet environnement avant chaque utilisation

```
source platformio/bin/activate
cd platformio
```

Installation avec pip

Bien vérifier que l'environnement virtuel est activé. Sinon, le programme va s'installer dans votre système, et risque d'entrer en conflit avec des bibliothèques python qui lui sont nécessaire.

On voit apparaître le nom de l'environnement avant le prompt lorsqu'il est activé. Dans notre cas, le prompt va ressembler à ça

```
(platformio) user@host:~/prog/arduino/platformio
```

```
pip-install platformio
```

Mettre en place les règles UDEV

Afin que le système sache quoi faire avec les cartes lorsqu'elles sont branchées en USB, il faut installer des règles UDEV. Pour ce faire, il suffit de télécharger ce fichier

/etc/udev/rules.d/99-platformio-udev.rules

```
# Copyright (c) 2014-present PlatformIO <contact@platformio.org>
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

```
#
# INSTALLATION
#

# UDEV Rules for PlatformIO supported boards,
http://platformio.org/boards
#
# The latest version of this file may be found at:
#
https://github.com/platformio/platformio-core/blob/develop/scripts/99-p
latformio-udev.rules
#
# This file must be placed at:
#   /etc/udev/rules.d/99-platformio-udev.rules    (preferred location)
#       or
#   /lib/udev/rules.d/99-platformio-udev.rules    (req'd on some broken
systems)
#
# To install, type this command in a terminal:
#   sudo cp 99-platformio-udev.rules /etc/udev/rules.d/99-platformio-
udev.rules
#
# Restart "udev" management tool:
#   sudo service udev restart
#       or
#   sudo udevadm control --reload-rules
#   sudo udevadm trigger
#
# Ubuntu/Debian users may need to add own “username” to the “dialout”
group if
# they are not “root”, doing this issuing a
#   sudo usermod -a -G dialout $USER
#   sudo usermod -a -G plugdev $USER
#
# After this file is installed, physically unplug and reconnect your
board.

# CP210X USB UART
SUBSYSTEMS=="usb", ATTRS{idVendor}=="10c4", ATTRS{idProduct}=="ea60",
MODE:="0666"

# FT232R USB UART
SUBSYSTEMS=="usb", ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6001",
MODE:="0666"

# Prolific Technology, Inc. PL2303 Serial Port
SUBSYSTEMS=="usb", ATTRS{idVendor}=="067b", ATTRS{idProduct}=="2303",
MODE:="0666"

# QinHeng Electronics HL-340 USB-Serial adapter
SUBSYSTEMS=="usb", ATTRS{idVendor}=="1a86", ATTRS{idProduct}=="7523",
```

```
MODE:="0666"

# Arduino boards
SUBSYSTEMS=="usb", ATTRS{idVendor}=="2341",
ATTRS{idProduct}=="[08][02]*", MODE:="0666"
SUBSYSTEMS=="usb", ATTRS{idVendor}=="2a03",
ATTRS{idProduct}=="[08][02]*", MODE:="0666"

# Arduino SAM-BA
ATTRS{idVendor}=="03eb", ATTRS{idProduct}=="6124",
ENV{ID_MM_DEVICE_IGNORE}="1"
ATTRS{idVendor}=="03eb", ATTRS{idProduct}=="6124",
ENV{MTP_NO_PROBE}="1"
SUBSYSTEMS=="usb", ATTRS{idVendor}=="03eb", ATTRS{idProduct}=="6124",
MODE:="0666"
KERNEL=="ttyACM*", ATTRS{idVendor}=="03eb", ATTRS{idProduct}=="6124",
MODE:="0666"

# Digistump boards
SUBSYSTEMS=="usb", ATTRS{idVendor}=="16d0", ATTRS{idProduct}=="0753",
MODE:="0666"
KERNEL=="ttyACM*", ATTRS{idVendor}=="16d0", ATTRS{idProduct}=="0753",
MODE:="0666", ENV{ID_MM_DEVICE_IGNORE}="1"

# STM32 discovery boards, with onboard st/linkv2
SUBSYSTEMS=="usb", ATTRS{idVendor}=="0483", ATTRS{idProduct}=="374?",
MODE:="0666"

# USBtiny
SUBSYSTEMS=="usb", ATTRS{idProduct}=="0c9f", ATTRS{idVendor}=="1781",
MODE="0666"

# USBasp V2.0
SUBSYSTEMS=="usb", ATTRS{idVendor}=="16c0", ATTRS{idProduct}=="05dc",
MODE:="0666"

# Teensy boards
ATTRS{idVendor}=="16c0", ATTRS{idProduct}=="04[789]?",
ENV{ID_MM_DEVICE_IGNORE}="1"
ATTRS{idVendor}=="16c0", ATTRS{idProduct}=="04[789]?",
ENV{MTP_NO_PROBE}="1"
SUBSYSTEMS=="usb", ATTRS{idVendor}=="16c0",
ATTRS{idProduct}=="04[789]?", MODE:="0666"
KERNEL=="ttyACM*", ATTRS{idVendor}=="16c0",
ATTRS{idProduct}=="04[789]?", MODE:="0666"

#TI Stellaris Launchpad
SUBSYSTEMS=="usb", ATTRS{idVendor}=="1cbe", ATTRS{idProduct}=="00fd",
MODE="0666"
```

```
#TI MSP430 Launchpad
SUBSYSTEMS=="usb", ATTRS{idVendor}=="0451", ATTRS{idProduct}=="f432",
MODE="0666"

# CMSIS-DAP compatible adapters
ATTRS{product}=="*CMSIS-DAP*", MODE="664", GROUP="plugdev"

# Black Magic Probe
SUBSYSTEM=="tty", ATTRS{interface}=="Black Magic GDB Server"
SUBSYSTEM=="tty", ATTRS{interface}=="Black Magic UART Port"
```

Recharger ensuite les règles

```
service udev restart
#       ou
udevadm control --reload-rules
udevadm trigger
```

Ajouter son utilisateur au groupe ''dialout''

Pour qu'il soit possible d'utiliser les ports série sans être *root*.

```
adduser $USER dialout
```

Utilisation

On peut faire beaucoup de chose avec ce programme. On va prendre un simple petit exemple : charger l'exemple `blink` sur un *Arduino Uno* (c'est largement inspiré de la doc officielle)

Dans notre cas, il va toujours falloir se trouver dans notre *environnement virtuel*. Un petit rappel pour l'activer

```
source ~/prog/arduino/platformio/bin/activate
```

Créer un projet

Commençons par créer un dossier qui va accueillir notre projet

```
mkdir -p ~/prog/arduino/platformio/blink
cd blink
```

On lance ensuite une commande qui va créer la structure du projet

```
platformio init --board uno
```

Copions simplement l'exemple `blink` qui va faire clignoter la LED intégrée raccordée au pin 13 dans le dossier `src` créé par la commande précédente

src/blink.ino

```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the
voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the
voltage LOW
  delay(1000);                       // wait for a second
}
```

```
cd src
```

Téléverser le projet

Cas général

Il ne reste plus qu'a compiler et téléverser dans l'Arduino raccordé en USB

```
platformio run --target upload
```

Lors de la première utilisation, `platformio` télécharge les différents outils nécessaire (`avrdude` par exemple). Cela ne sera plus le cas ensuite.

Utiliser un programmateur externe (pour les plateforme AVR seuelement)

Certains microcontrolleurs AVR nécessite l'utilisation d'un programmateur spécial. C'est intégré dans la définition des cartes, mais il faut lancer la commande avec l'option indiquant d'utiliser ce programmateur. Ce n'est donc plus `upload` mais `program`

```
platformio run --target program
```

Le moniteur Série

On peut ouvrir un moniteur série avec la commande `platformio device monitor`. Cette commande dispose de différentes options. Par exemple, pour obtenir un comportement se rapprochant de la console série de l'IDE Arduino

```
platformio device monitor --echo --eol LF
```

```
--- Miniterm on /dev/ttyACM0  9600,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
```

Et voilà, vous pouvez utiliser un Arduino sans l'IDE Arduino 😀

1)

a priori

2)

Environnement de Développement Intégré

3)

Si j'y arrive un jour