README.md

Wilson Coding Battlespace - Challenge Roverity

Présentation générale

Relire les slides

Plus de détails

Composants

- Raspberry Pi :
  - communique avec l'Arduino via la liaison série
  - commande le capteur de photo
  - connecté sur le réseau et accessible via SSH
- Arduino :
  - pilote les moteurs des roues, le capteur de distance et le servo-moteur permettant d'actionner le godet
  - interprète les informations reçues par la liaison série
  - renvoie les informations à propos de son état via la liaison série
- Plate-forme à quatre roues motrices
- Capteur photo
- Capteur de distance ultrasonic avec une précision de 1 cm

Liaison série

L'Arduino et le Raspberry Pi communiquent via cette liaison série.

- Vitesse de la liaison : **9600 bauds**
- Port: **/dev/ttyACM0**

L'ouverture de la liaison série redémarre automatiquement l'Aruidno ! Ce reboot peut durer 2 secondes. **Il est nécessaire d'attendre le READY avant tout envoi de nouvelle instruction**.

Trame envoyée

Les ordres sont à envoyer sous forme d'une suite de caractères terminée par un caractère de nouvelle ligne "\n";

Exemple :

```
f10\n
```

Chaque caractère correspond à un ordre pour un élément décrit dans le tableau suivant :

Position de l'élément	Définition	Valeur possible	Traduction de l'exemple
1	Direction du robot	f = avance, b = recule, g = godet en bas, u = godet en haut, m = faire mesure, s = arrêt	avance
2	Nombre de pas à faire	entier de 0 à 1000	10 pas

Les actions **g**, **u**, **m** et **s** doivent avoir un nombre de pas à **zéro**. Exemple : `g0\n` , `u0\n` , `m0\n` et `s0\n`

Trame reçue

Les éléments de la trame sont séparés par des virgules "**,**" et chaque trame se termine par un point-virgule "**;**".

Exemple :

```
102,40,5;
```

Chaque élément de cette chaine, entre les séparateurs, correspond à une des définitions suivantes.

Position de l'élément	Définition	Valeur possible	Traduction de l'exemple
1	Direction du robot en code ASCII	102 = f = avance, 98 = b = recule, 103 = g = godet en bas, 117 = u = godet en haut, 109 = m = faire mesure, 115 = s = arrêt	avance
2	Nombre de pas **restant** à faire	entier de 0 à 1000	40 pas restants
3	distance jusqu'à l'obstacle capté par le capteur de distance, en cm	0 - 400	obstacle à 5cm

Tant que le compteur de pas n'a pas atteint zéro, **aucune nouvelle instruction** ne sera acceptée par le robot.

La mesure de distance est mise à jour après l'envoi de l'action **m**.

Accusé de réception et état des instructions

Au démarrage de la liaison série, elle envoie une trame `READY` pour indiquer que la liaison série est démarrée et prête à recevoir une instruction.

Lorsqu'une instruction a été envoyée depuis la liaison série, elle envoie un `ACK` pour indiquer que l'instruction a bien été reçue. Dans ce cas, la liaison série ne peut plus recevoir de nouvelles instructions tant que le compteur de pas n'a pas atteint zéro.

Une fois que le compteur de pas a atteint zéro, un `READY` est envoyé depuis la liaison série. Vous pouvez alors envoyer une nouvelle instruction.

Prendre une photo avec le Raspberry Pi

Pour prendre une photo avec le raspberry pi, vous devez utiliser la commande rapistill

La photo à envoyer doit être d'une résolution de **1024x576** avec une compression de **50%**.

Commande pour prendre une photo :

```
raspistill -w 1024 -h 576 -q 50 -o photo.jpg
```

Envoi des données à l'API

Au cours de sa mission, le robot devra envoyer quelques informations à la base.

Adresse de la base : **http://192.168.1.254:8000/**

Un token de sécurité vous sera communiqué. Il faudra l'insérer dans le header **x-token**.

Envoi d'une mesure de distance

POST /robots/:id/mesure

Headers :

```
content-type: application/json
x-token: token-team
```

Body :

```
{
    mesure: n
}
```

n est un entier positif tel qu'il a été reçu depuis la liaison série.

Envoi de la photo

POST /robots/:id/photo

Headers :

```
content-type: application/json
x-token: token-team
```

Body :

```
{
    photo: base64_encoded
}
```

base64_encoded correspond à la photo prise avec la commande `raspistill` . La photo doit être encodée en base64.

Librairies

En fonction de votre langage de programmation, des librairies sont fournies pour simplifier les échanges avec la liaison série ou l'API.

Le Raspberry Pi fonctionne sous raspbian. Tous les outils nécessaires sont déjà installés.

NodeJS

Les librairies NodeJS request et serialport sont installées en global.

- NodeJS v5
- serialport v2.1.2
- request

Ouverture d'une liaison série :

```
var serialport      = require("serialport");
var SerialPort      = serialport.SerialPort;
var sp = new SerialPort("/dev/ttyACM0", {
    parser: serialport.parsers.readline(";"),
    baudrate: 9600
}, true);

sp.on('error', function(data){
    console.log("[serialport] " + data);
});

sp.on('open', function(){
    sp.on('data', function(data) {
        console.log(data); // exemple de valeur : "102,2,0"
    });
//    sp.write(order, function(err, res) {});
});
```

PHP

- PHP 5.6
- serialport
- php5-curl et nategood/httpful

Les librairies PHP httpful et serialport sont installées en global.

Ouverture d'une liaison série :

```
<?php
set_time_limit(0);

require_once '/home/pi/.composer/vendor/autoload.php ';

use lepiaf\SerialPort\SerialPort ;
use lepiaf\SerialPort\Parser\SeparatorParser ;
use lepiaf\SerialPort\Configure\TTYConfigure ;
use lepiaf\SerialPort\Exception\LogicException ;

try {
    $serialPort = new SerialPort(new SeparatorParser(";"), new TTYConfigure());
    $serialPort->open('/dev/ttyACM0');
} catch (LogicException $e) {
    echo $e->getMessage();
    die();
}

while ($data = $serialPort->read()) {
    echo $data."\n"; // exemple de valeur : "102,2,0"
}

// $serialPort->write($order);

$serialPort->close();
```

Note

Pour les développements en php, **apache/nginx n'est pas installé**. Les scripts doivent être lancés via le cli php.

```
php -S 0.0.0.0:8080 index.php
```