Tutorial

Open Source List

ROS

Use the publish subscribe communication mechanism of ROS master

- The master of ROS provides the communication between the nodes and the controlling agent as one of the node to control other nodes
- The nodes are the application software(like position estimation, navigation), the driver of the AGV and the controlling agent

ROS master introduction

- http://wiki.ros.org/Master

Fluentd

- To serve as the server which provides the routing mechanism for the logging data

- Tutorial
  - slide
    - https://www.slideshare.net/treasure-data/fluentd-meetup-in-japan-11410514
  -

- Plugin
  - fluent-logger-python
    - 提供 python 可用 api，並使用 ip / tcp 傳輸資料到 fluentd 上
    - https://github.com/fluent/fluent-logger-python
  - exec output plugin
    - 將環境變量中的 stdout 重定向到程式執行時的輸入，因此可以支援各種語言的 output
    - fluentd 內建 output plugin
    - https://docs.fluentd.org/v0.12/articles/out_exec
    - https://github.com/fluent/data-collection

OSbrain

- The multi-agent library for the safety monitoring agents

- The OSbrain depends on Pyro4 and ZeroMQ
  - Pyro4

official website: https://pythonhosted.org/Pyro4/index.html
example: http://www.voidcn.com/blog/xiaolewennofollow/article/p-6147887.html

OpenCV

- Use to detect variation of the frame when any object appears on the course

安裝

安裝 osBrain

```
pip3 install osbrain
```

the command would automatically install the osbrain package and PyZmq library

安裝 Fluentd and plugin

Fluentd 安裝

由於 Fluentd 是要用來當做裝置本身以及裝置和 edge 間的 routing server，因此要安裝在幾個不同的裝置上。以下列出在 ubuntu 14.04、Raspbian Jessie、ubuntu Mate 上的安裝。

Ubuntu 14.04 上安裝

1. 首先要安裝 ruby 的開發環境

ruby 開發環境安裝
https://www.ruby-lang.org/zh_tw/downloads/
下載 Ruby 2.3.4 解壓縮後進入安裝目錄執行如下：

```
$ ./configure
$ make
$ sudo make install
```

2. 之後使用 gem 安裝 Fluentd

參考官網(https://docs.fluentd.org/v0.12/articles/install-by-gem)

```
gem install fluentd -v "~> 0.12.0" --no-ri --no-rdoc
```

如果在 gem install 時出現如下問題

```
$ gem install 安裝程式
ERROR: Loading command: install (LoadError)
    cannot load such file -- zlib
ERROR: While executing gem ... (NameError)
    uninitialized constant Gem::Commands::InstallComman
```

移除 ruby 相關套件、安裝缺少的相依套件後重新安裝 ruby

```
$ sudo apt-get install zlib1g-dev
$ sudo apt-get install libssl-dev
```

Install on Raspbian (Raspberry Pi3)

官網有直接的 solution

參考官網(https://docs.fluentd.org/v0.12/articles/raspberrypi-cloud-data-logger)

Install on ubuntu mate(raspberry pi3)

1. install rvm

```
$ gpg --keyserver hkp://keys.gnupg.net --recv-keys
409B6B1796C275462A1703113804BB82D39DC0E3
7D2BAF1CF37B13E2069D6956105BD0E739499BDB
$ \curl -sSL https://get.rvm.io | bash -s stable
$ source ~/.bashrc
$ source ~/.bash_profile
```

2. install Ruby 2.3.4

```
$ rvm install 2.3.4
```

3. fluentd 安裝(gem 安裝版)

   安裝說明網址

   https://docs.fluentd.org/v0.12/articles/install-by-gem

Reference

RVM 安裝與使用教學
http://railsfun.tw/t/rvm/28
RVM official
https://rvm.io/

Plugin 安裝

fluent-logger-python
https://github.com/fluent/fluent-logger-python
```
pip install fluent-logger
```

安裝 ROS

Raspberry Pi 上安裝 ROS install

- 使用的 Raspberry Pi 配置
  - 硬體：Raspberry Pi 3
  - 系統：Ubuntu Mate 16.04
  - ROS 版本: Kinetic

- Raspberry Pi 上安裝 ROS 可以參考官網
  - Installing ROS Kinetic on the Raspberry (Official)
    http://wiki.ros.org/ROSberryPi/Installing%20ROS%20Kinetic%20on%20the%20Raspberry%20Pi

  - SD card install
    http://www.german-robot.com/2016/05/26/raspberry-pi-sd-card-image/
  -

- Require to install python 3 dependence when writing in python 3

```
pip3 install catkin-tools
pip3 install rospkg
```

開啟步驟

Step 1 啟動 edge 端上的 agent

進入到 safety_monitoring/src 資料夾下開啟 run_remote_agent.py

```
sudo python3 run_remote_agent.py
```

顯示如下代表成功開啟

```
Broadcast server running on 0.0.0.0:9091
NS running on 127.0.0.1:210 (127.0.0.1)
URI = PYRO:Pyro.NameServer@127.0.0.1:210
INFO [2017-08-08 05:55:10.443527] (monitoring_agent_edge): Monitoring agent
is running
```

Step 2 啟動裝置端上的 agent

開啟 run_local_agent.py

```
sudo python3 run_remote_agent.py
```

顯示如下代表成功開啟

```
Broadcast server running on 0.0.0.0:9091
NS running on 127.0.0.1:200 (127.0.0.1)
URI = PYRO:Pyro.NameServer@127.0.0.1:200
INFO [2017-08-08 07:55:40.918743] (monitoring_agent_local): Waiting Fluentd
connecting...
```

Step 3 啟動 Fluentd

step 3 in the official website

https://docs.fluentd.org/v0.12/articles/install-by-gem

使用方法

Monitor

介紹如何在 application 的程式中使用 monitor。我們將使用實際感測器例子做說明，此感測器是中線感測器，可以感測到 AGV 移動時在路中間的比例，在此以 ratio 變數表示。

創建 monitor 物件

創建時可以同時指定 tag 的名稱，下面例子是使用"data_infrared_line"作為監控用時的 tag 名。創建完後使用方法 setup()建立與 fluentd 間的連線設定。

```
from instrumentation.monitor import Monitor
monitor = Monitor("data_infrared_line")
monitor.setup()
```

當添加新的 tag 名稱時，fluentd 的 conf 裡也要添加相對應新的 tag 值

safety_monitoring/conf/fluentd/fluent.conf

```
<match monitor.data_infrared_line>
@type copy
  <store>
   @type exec
   buffer_type memory
   command python3 /home/pi/monitoring/conf/fluent/plugin
plugin/exec_plugin.py
   format json
   flush_interval 1s # for debugging/checking
  </store>
  <store>
    @type stdout
  </store>
</match>
```

log

log 的資料格式是使用 json 並且有格式上面的規範如下：

```
log =  {
        "tags":{"host": "string_only", "type": "string_only",
"device":"string_only"},
        "fields": {"name_value1": value1, "name_value2": value2 ...]
        }
```

實際的使用例子

```
log_exapmple = {
```

```
        "tags": {"host": "alphabot","type": "position",
"device":"infrared_light"},
        "fields" : ["center_ratio": ratio]
        }
```

在程式中使用 monitor.send()方法送出 log 資料

```
#The log data
#ratio 是我們收集到的數據
log_exapmple={
          "tags": {"host": "alphabot","type": "position",
"device":"infrared_light"},
          "fields" : ["center_ratio": ratio]
        }
#送出資料
monitor.send(inputData)
```

Agent

裝置端創建 agent

在裝置端上需要創建至少兩個 agent，分別是 monitoring agent 以及 controlling agent，創建方式如下：

```
#第三方的 multi-agent library
from osbrain import SocketAddress
from osbrain import run_nameserver
from osbrain import run_agent
from osbrain import proxy
```

```
#我們的函示庫，用於提供監控用的 reactive agent，分別繼承於 Agent class 的兩種 class
from safetyagent.agent import MonitoringAgent, ControllingAgent
```

```
# host, port for local and edge
local_socket_address = SocketAddress('127.0.0.1', 200)
edge_socket_address = SocketAddress('127.0.0.1', 210)
```

```
# start the name server for multi-agent system
ns_local = run_nameserver(addr = local_socket_address)


# run the two agents, monitoring agent (local) and controlling agent(local)
agent_1 = run_agent('monitoring_agent_local', base = MonitoringAgent)
agent_2 = run_agent('controlling_agent_local', base = ControllingAgent)


# connect to the remote agent
ns_remote = proxy.NSProxy(nsaddr=edge_socket_address, timeout=10)
agent_3 = ns_remote.proxy('monitoring_agent_edge')
agent_3 = run_agent('monitoring_agent_local_2', base = MonitoringAgent)
```

Edge 端創建 agent

在 edge 端創建 agent 與 device 端創建的方法類似，但只有 monitoring agent 需要創建

Controller 的設定及使用

設定

controller 是用來對 application 進行控制的物件，當需要撰寫提供給 controlling agent 使用的 controller 時，必須按使用下面的寫法。必須具有 run()、display()、update(msg)三種方法。

```
class Controller:
  def __init__(self):
    pass
  def display(self):
    pass
  def update(self, msg):
    pass
  def run(self,msg):
    pass
```

display 方法用來提供 controller 當前狀態；update 方法和 run 方法是用來給 controlling agent 使用，當有 message 從 monitoring agent 送來時，controlling agent 會更新 controller 的狀態。run 方法則是用來開啟 controller。

使用

```
import safetyagent.messagechannel as messagechannel
from safetyagent.controllerset import CloseDistanceController


def controller_handeler(agent, message):
    agent.update_controller(message)
    agent.display_controller()


'''
set the Push-Pull communication pattern. The input of parameters are local
monitoring_agent, controlling agent, remote monitoring agent, channel_local,
channel_remote, handler of controlling agent
'''
messagechannel.set(agent_1,agent_2,agent_3,'local_channel','edge_channel',
controller_handeler)


# set the controller used by the controlling agent
controller = CloseDistanceController()
agent_2.set_controller(controller)
```

Rule 設定及使用

設定

tag 是用來標示 log 裡資料的種類，當 rule 要被觸發時，必須要有正確對應的 tag，因此只有當 rule 使用的 tag 與 monitor 中設定的 log 型式一致時 rule 才有意義。

tag 是為具有{”tags”:{“host”: “s”, “type”: “s” , “device”: “s”}, “field” : [“s”,”s”,”s”]}pattern 的字典(dictionary)，其中”s”是可以替換的 string

註: Tag 在 API documentation 中又成 log mark

rule 可以在 ruleset.py(src/safetyagent/ruleset.py)中設定。首先在撰寫 rule 時會用到的 function 和 tag，其表示如下：

#我們使用前面的中線感測器以及距離感測器做例子，分別為 `tag_1`、`tag_2`

```
tag_1 ={
          "tags": {"host": "alphabot","type": "position",
"device":"infrared_light"},
          "fields" : ["center_ratio"]
        }
tag_2 =  {
        "tags":{"host": "alphabot","type": "distance",
"device":"utrasonic_sensor"},
        "fields": ["front"]
        }


def ca_1(fields):
        if fields[0]["center_ratio"] < 50:
            return 'Left'
        if fields[1]["front"] < 30
        else:
            return 'Stop'
```

由於 ca_1 function 同時使用了 tag_1 和 tag_2，需使用 fields[0]和 fields[1]用以區別 tag_1 和 tag_2。

當 ruleset.py 設定完後，需要創建相應的 rule 物件。

```
from safetyagent.rule import Rule
from safetyagent.ruleset import tag_1, tag_2, ca_1, ca_2


rule1 = Rule(tag_1, tag_2)
rule1.add_condition_action(ca_1)
```

Rule 的參數的順序必須按照 ruleset.py 中的定義來添加。由上面的例子可以知道 rule1 物件是同時擁有 tag_1, tag_2 以及 ca_1 的 rule

使用

agent 中添加 rule

```
agent_1.set_rule(rule1)
```

執行

```
# set the port for the monitoring agent to listen the fluentd log message
```

```
Host = 'localhost'
Port = '8087'
agent_1.monitor(Host, Port)


# running the setting


agent_1.trigger_rule('local_channel')
```