Checkpoint # Report

[EECN30169] Mobile Robot 2024

Student ID: 313605019 **Name:** 方敏 **Date:** **2024/10/04**

1. Purpose:

Checkpoint 1 目的是設置機器人的軟體開發環境，在 Raspberry Pi 上安裝 Ubuntu Mate 18.04 和 ROS Melodic，並透過 ROS（Robot Operating System）系統將 Raspberry Pi 與 Arduino Uno 連線；還有學習 ROS 的檔案系統和圖層架構、建立 package file，以及通過 Publisher 發送數據、Subscriber 接收數據的過程學習 ROS 節點間通訊。

2. Description of Design:

Raspberry Pi 向 Arduino 發送數據，Arduino 處理該數據，Task 為將其乘以 2 後將結果傳回；這個設計利用 ROS 框架，通過 rosserial 套件來實現這兩個設備之間的實時通訊。

實驗分為以下幾個步驟 :

A. Environment setting on Raspberry Pi
   1. Install Ubuntu mate 18.04
   2. Install ROS Melodic
   3. Setting SSH between Raspberry Pi and PC

B. ROS
   1. ROS file system level structure
   2. ROS computation graph level
   3. Create package file
   4. ROS nodes communication
   5. Publisher and Subscriber
   6. CMakeLists.txt
   7. roslaunch

C. Rosserial package
   1. Arduino IDE setup
   2. Install the Software
   3. Create a publisher and subscriber by using rosserial

不同於 Note 中建議的將 publisher 及 subscriber 程式碼分開檔案並用 launch 檔案連接，我們選擇將 publisher 及 sublisher 寫在同一份檔案裡並根據參數名稱修改 CMakeLists，以下為程式碼：

```cpp
#include "ros/ros.h"
#include "std_msgs/Int32.h"
#include <iostream>
#include <stdio.h>

int flag = 1;

void number_callback(const std_msgs::Int32::ConstPtr& msg) {
    printf("message from Arduino is %d\n", msg->data);
    flag = 1;
}

int main(int argc, char **argv)
{

    ros::init(argc, argv, "rpi_node");

    ros::NodeHandle node_obj;

    ros::AsyncSpinner spinner(0);
    spinner.start();

    ros::Publisher rpi_publisher = node_obj.advertise<std_msgs::Int32>("topic_rpisend", 10);
    ros::Subscriber rpi_subscriber = node_obj.subscribe("topic_arduinosend", 10, number_callback);
    ros::Rate loop_rate(1);


    while (ros::ok()) {
        std_msgs::Int32 msg;

        if (flag == 1){
            std::cout << "user's input is :";
            std::cin >> msg.data;
            rpi_publisher.publish(msg);
            flag = 0;
        }


        ros::spinOnce();
        loop_rate.sleep();

    }
    return 0;
}
```

Arduino 程式碼：

```cpp
//run on Arduino
#include <ros.h>
#include <std_msgs/Int32.h>

std_msgs::Int32 msg;
std_msgs::Int32 return_msg;
ros::NodeHandle nh;


ros::Publisher arduino_pub("topic_arduinosend", &return_msg);


void messageCb(const std_msgs::Int32 &msg){
  return_msg.data = msg.data * 2;
  arduino_pub.publish(&return_msg);
}

ros::Subscriber<std_msgs::Int32> arduino_sub("topic_rpisend", messageCb);


void setup() {
  //Serial.begin(57600);
  nh.initNode();
  nh.advertise(arduino_pub);
  nh.subscribe(arduino_sub);
}
```

```
void loop() {
  //Serial.print(msg.data, return_msg.data);
  //ROS_INFO("arduino get [%d]", msg.data);
  //ROS_INFO("return_msg.data  [%d]", return_msg.data);

  nh.spinOnce();
}
```

3. Result

Task1：

```
lab639@lab639-desktop:~$ rosversion -d
melodic
```

在 Task1 我們會確認 ROS 的安裝版本，在命令行輸入"rosversion -d"，可以獲得當前機器上 ROS 的版本訊息。

Task2：

```
^Clab639@lab639-desktop:~$ rosrun lab1_pkg demo_topic_pubsub
user's input is :5
message from Arduino is 10
user's input is :20
message from Arduino is 40
user's input is :
```

在 Task2 中，Raspberry Pi 同時執行 publisher 和 subscriber 的功能，Publisher 的主要作用是向名為 "topic_rpisend" 的主題發布數據，而 Subscriber 則監聽來自 Arduino 的 "topic_arduinosend" 主題，並在接收到新訊息時觸發相應的回調函數；Arduino Uno 將使用 ros_serial 通訊協議來監聽" topic_rpisend "主題，一旦監聽到新的資訊，就會將其乘以 2，然後將結果發布到" topic_arduinosend "主題上。這個過程展示了 Raspberry Pi 作為通訊節點，負責數據的發布與接收，Arduino Uno 作為轉換器的角色，將它收到的資訊乘二增加後回傳給 Raspberry Pi，為 Raspberry Pi 與 Arduino 進行數據交互的具體實現。

4. Discussion

這次的 Checkpoint 1 讓我從硬體到軟體的層面都有了全面的學習與提升，特別是對 ROS 系統的運作有了更深的理解，並且為後續的實驗打下了堅實的基礎。我逐步掌握了 Publisher 和 Subscriber 節點的運作方式，Publisher 節點將數據發佈到主題，Arduino 接收到後進行處理，再將結果通過新的主題傳回 Raspberry Pi 的 Subscriber 節點，這種雙向的數據傳輸讓我親身體驗了 ROS 的靈活性和強大的分佈式處理能力。

其中在 Task2 中，我們原本需要先執行 echo $ROS_PACKAGE_PATH 與 source ~/catkin_ws/devel/setup.bash 才能成功 rosrun，如下圖：

```
lab639@lab639-desktop:~$ rosrun lab1_pkg demo_topic_pubsub
[rospack] Error: package 'lab1_pkg' not found
lab639@lab639-desktop:~$ echo $ROS_PACKAGE_PATH
/opt/ros/melodic/share
lab639@lab639-desktop:~$ source ~/catkin_ws/devel/setup.bash
lab639@lab639-desktop:~$ rosrun lab1_pkg demo_topic_pubsub
user's input is :6
```

這兩串指令功能分別為檢查 ROS 的包路徑以及更新當前終端的環境變數，確保 ROS 能夠從工作空間中查找節點、執行檔案、消息類型；如果在 .bashrc 文件中加入這句 "source ~/catkin_ws/devel/setup.bash" 後，就不需要執行這兩句就可以直接 rosrun。

通過這次實驗，我不僅學習到許多 ROS 與 Arduino 的知識，也更加理解其在物聯網中適用的實際場景，謝謝老師的指導以及助教們不厭其煩的幫我們解決問題。