ENGR 4421: Robotics II

ROS Tutorial: Client Libraries

02/02/2023

Outline

- Using `colcon` to build packages
- Creating a workspace
- Creating a package
- Writing a simple publisher and subscriber (Python)

Using `colcon` to Build Packages

- Tutorial page:

  https://docs.ros.org/en/humble/Tutorials/Beginner-Client-Libraries/Colcon-Tutorial.html

- Install `colcon`:

```
sudo apt install python3-colcon-common-extensions
```

- Setup `colcon`:

```
echo "source /usr/share/colcon_cd/function/colcon_cd.sh" >> ~/.bashrc
echo "export _colcon_cd_root=/opt/ros/humble/" >> ~/.bashrc
pip install colcon-argcomplete
echo "source /usr/share/colcon_argcomplete/hook/colcon-argcomplete.bash" >> ~/.bashrc
echo 'eval "$(register-python-argcomplete3 ros2)"' >> ~/.bashrc
echo 'eval "$(register-python-argcomplete3 colcon)"' >> ~/.bashrc
```

Creating A Workspace

- Tutorial page:

  https://docs.ros.org/en/humble/Tutorials/Beginner-Client-Libraries/Creating-A-Workspace/Creating-A-Workspace.html

- Create workspace directory:
  ```
  mkdir -p ~/ros2_ws/src
  ```

- Clone a few ROS packages:
  ```
  cd ~/ros2_ws/src
  git clone https://github.com/ros/ros_tutorials.git -b humble-devel
  ```

- Resolve dependencies:
  ```
  cd ~/ros2_ws/
  rosdep install -i --from-path src --rosdistro humble -y
  ```

- Build workspace:
  ```
  cd ~/ros2_ws/
  colcon build
  ```

- Source the overlay:
  ```
  echo "source ~/ros2_ws/install/local_setup.bash" >> ~/.bashrc
  ```

Creating A Package

- Tutorial page:

  https://docs.ros.org/en/humble/Tutorials/Beginner-Client-Libraries/Creating-Your-First-ROS2-Package.html

- Create a package:

  ```
  ros2 pkg create --build-type ament_python <package_name>
  ```

- Package contents:

  ```
  <package_name>/  package.xml  resource/  setup.cfg  setup.py  test/
  ```

Publisher (Python) with gpiozero

```python
import rclpy
from rclpy.node import Node
from std_msgs.msg import String
from gpiozero import Button

class MinimalPublisher(Node):
    def __init__(self):
        super().__init__('minimal_publisher')
        self.button = Button(26)
        self.publisher_ = self.create_publisher(String, 'button_topic', 10)
        timer_period = 0.5  # seconds
        self.timer = self.create_timer(timer_period, self.timer_callback)
    def timer_callback(self):
        msg = String()
        if self.button.is_pressed:
            msg.data = "pressed"
        else:
            msg.data = "unpressed"
        self.publisher_.publish(msg)
        self.get_logger().info('Publishing: "%s"' % msg.data)
        # self.i += 1

def main(args=None):
    rclpy.init(args=args)
    minimal_publisher = MinimalPublisher()
    rclpy.spin(minimal_publisher)
    minimal_publisher.destroy_node()
    rclpy.shutdown()

if __name__ == '__main__':
    main()
```

Subscriber (Python) with gpiozero

```python
import rclpy
from rclpy.node import Node
from std_msgs.msg import String
from gpiozero import LED

class MinimalSubscriber(Node):
    def __init__(self):
        super().__init__('minimal_subscriber')
        self.led = LED(19)
        self.led.off()
        self.subscription = self.create_subscription(String, 'button_topic', self.listener_callback, 1)
        self.subscription  # prevent unused variable warning

    def listener_callback(self, msg):
        if msg.data == "pressed":
            self.led.on()
        else:
            self.led.off()

def main(args=None):
    rclpy.init(args=args)
    minimal_subscriber = MinimalSubscriber()
    rclpy.spin(minimal_subscriber)
    minimal_subscriber.destroy_node()
    rclpy.shutdown()

if __name__ == '__main__':
    main()
```