* roslaunch aikit\_nav amcl 開地圖
* roslaunch aikit\_rviz amcl\_view\_dwa.launch 開rviz
* roslaunch aikit\_nav aikit\_nav\_dwa\_demo.launch map:=/home/robot/maps/demo 地圖快捷
* 1.開掃地圖和,儲存地圖,開gmapping
* 1.roslaunch aikit\_chassis drive\_start.launch //開始一些東東
* 2.roslaunch aikit\_rviz gmapping\_view.launch //開gmapping 的介面
* 3.roslaunch aikit\_nav gmapping\_demo.launch //開始掃地圖(drive\_start需開啟)
* 4.掃完後，儲存地圖rosrun map\_server map\_saver -f /demo/robot/maps/[新的檔案名]
* 圖片:
* O—> Topic —>O
* 快點，因為没回應
* O—> servicr —>O
* ←—----------------
* param—>O
* (action)
* server
* | |
* | |
* V V
* server client
* self.move\_base = actionlib.SimpleActionClient("move\_base", MoveBaseAction)
  + roslaunch turtlebot\_navigation amcl\_demo.launch
    - 三个功能正在使用
    - amcl demo 地圖，雷達，底盤
* rospy.loginfo("Waiting for move\_base action server...")
  + print sth
  + plan 路線
* self.move\_base.wait\_for\_server(rospy.Duration(120))
  + 上限120秒，等待server連結
  + 等待
    - self.move\_base.cancel\_goal()
    - rospy.loginfo("waiting for /amcl\_pose")
    - rospy.wait\_for\_message("/amcl\_pose", GoalStatusArray)
    - rospy.loginfo("waiting for /move\_base/status")
    - rospy.wait\_for\_message("/move\_base/status", GoalStatusArray)
    - rospy.loginfo("RoboChassis OK"
* frame\_id == map
* self.initial\_pose = PoseWithCovarianceStamped()
  + from geometry\_msgs.msg import Pose, PoseWithCovarianceStamped, Point, Quaternion, PointStamped, PoseStamped
  + 初如pose/路徑
* self.current\_pose = PoseWithCovarianceStamped()
* self.goal = MoveBaseGoal()
  + 目標
  + from move\_base\_msgs.msg import MoveBaseAction, MoveBaseGoal
* rospy.wait\_for\_message("initialpose", PoseWithCovarianceStamped)
  + 存到server
* 實行
  + def move\_to(self, x, y, theta):
  + location = self.point\_to\_pose(x, y, theta)
  + self.goal = MoveBaseGoal()
  + self.goal.target\_pose.header.frame\_id = self.frame\_id
  + self.goal.target\_pose.header.stamp = rospy.Time.now()
  + self.goal.target\_pose.pose = location
  + self.move\_base.send\_goal(self.goal)
  + time.sleep(1)
  + success = self.move\_base.wait\_for\_result(rospy.Duration(300))
  + if success == 1:
  + rospy.loginfo("Reached point.")
  + else:
  + rospy.loginfo("Failed to reach point.")
  + return success
* 狀態間的不同處理方法
  + if code == 0: #無指令
  + pass
  + elif code == 1: # 行進
  + pass
  + elif code == 3: # 到達地點
  + rospy.loginfo("3. Move to %.2f, %.2f, %.2f" % (P[0], P[1], P[2]))
  + G = chassis.get\_current\_pose()
  + chassis.move\_to(P[0], P[1], P[2]) #起如和目標不停地相对行走
  + P = G #初始點改為G
  + elif code == 4: # 錯誤
  + chassis.set\_goal\_in\_rviz()
  + G = chassis.get\_goal\_pose()
  + rospy.loginfo("To %.2f, %.2f, %.2f" % (G[0], G[1], G[2]))
  + else: #行進時有物體欄著
  + rospy.loginfo("%d, %s" % (code, text))
* class RobotChassis:
  + 物件導向
  + 30-179 初始化
  + chassis = RobotChassis()
* chassis.set\_initial\_pose\_in\_rviz()
  + 等rviz的數據
* P = chassis.get\_current\_pose()
  + 初始點
* G = chassis.get\_goal\_pose()
  + 目標點
* if \_\_name\_\_ == "\_\_main\_\_":
* rospy.init\_node("home\_edu\_robot\_chassis")
* rate = rospy.Rate(20)
* # 1. Create a RobotChassis object.
* chassis = RobotChassis()
* # 2. Set current pose at the first time.
* chassis.set\_initial\_pose\_in\_rviz()
* P = chassis.get\_current\_pose()
* rospy.loginfo("From %.2f, %.2f, %.2f" % (P[0], P[1], P[2]))
* # 3. Set the target pose at the first time.
* chassis.set\_goal\_in\_rviz()
* G = chassis.get\_goal\_pose()
* rospy.loginfo("To %.2f, %.2f, %.2f" % (G[0], G[1], G[2]))
* while not rospy.is\_shutdown():
* # 4. Get the chassis status.
* code = chassis.status\_code
* text = chassis.status\_text
* # 5. From P to G, then from G to P.
* if code == 0: # No plan.
* pass
* elif code == 1: # Processing.
* pass
* elif code == 3: # Reach point.
* rospy.loginfo("3. Move to %.2f, %.2f, %.2f" % (P[0], P[1], P[2]))
* G = chassis.get\_current\_pose()
* chassis.move\_to(P[0], P[1], P[2])
* P = G
* elif code == 4: # No solution.
* chassis.set\_goal\_in\_rviz()
* G = chassis.get\_goal\_pose()
* rospy.loginfo("To %.2f, %.2f, %.2f" % (G[0], G[1], G[2]))
* else:
* rospy.loginfo("%d, %s" % (code, text))
* rate.sleep()
* rospy.loginfo("END")