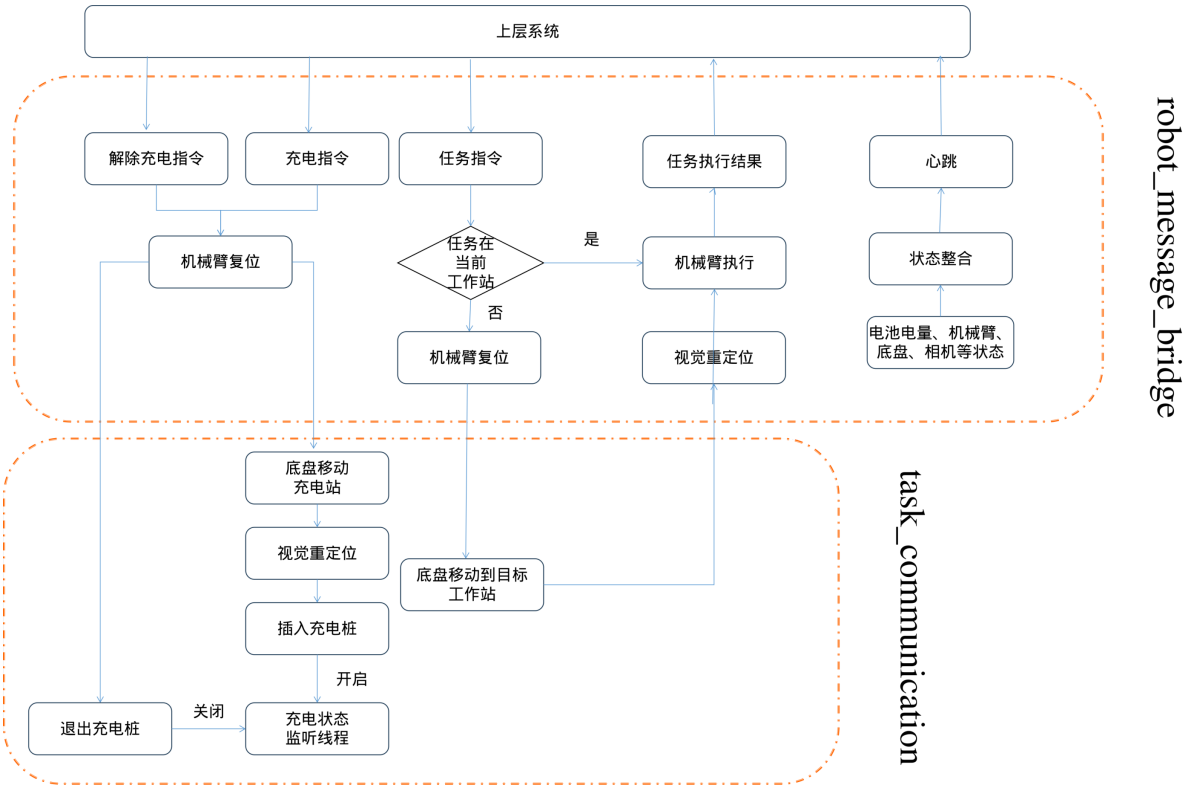
机器化学家机器人说明文档

1.robot_message_bridge

robot_message_bridge是联系上层调度和机器人各个模块的枢纽，主要联系流程如下图所示：



上层系统通过http下发指令，包括任务指令、充电指令、解除充电指令，robot_message_bridge异步接收指令，通过回调函数具体执行指令。指令形式如下，一般包括指令的id，时间戳stamp，工作站名称destination，具体的操作。接收到指令后通过上述流程进行执行不同的任务。

```
{
  'id': 1683372407232012288,
 'stamp': None,
  'destination': 'charge_station',
 'operations': [{'operation': 'charge'}]
}
```

机械臂复位和重定位消息格式分别为

```
{
  "id": '',
  "destination": "",
  "operations": [{"operation": "reset"}]
}

{
  "id": '',
 "destination": "",
 "operations": [{"operation": "relocation"}]
}
```

底盘的移动、充电、解除充电指令为

```
{
  "id": '',
 "stamp": '',
 "destination": "",
  "action": "move"
}

{
  "id": '',
 "stamp": '',
 "destination": "",
  "action": "charge"/"uncharge"
}
```

执行结果通过mqtt返回给上层服务id表示执行指令的id，时间戳stamp，status表示机器人状态，current_workstation当前机器人所在的工作站，state指令执行的结果包括done和error，detail表示具体的细节。

```
{
    "id": '',
    "stamp": '',
    "status": '',
    "current_workstation": '',
    "state": '',
    "detail": {"navi": '', "oper": ''}
}
```

心跳通过http发送给上层系统，具体的格式如下，detailMsg为具体的信息，currentStation为当前工作站的名称，electricityQuantity为当前电池的电量，machineCode为机器人的id用于区分不同的机器人，detail为工作站使用，机器人不使用，心跳的状态通过ros服务消息进行查询。

```
{
    'stamp': "",
    'status': '',
    'detailMsg': '',
    'currentStation': '',
    'electricityQuantity': 80.0,
    'machineCode': 'robot',  # 机器人不用，为固定值
    'detail': None,  # 机器人不用，为固定值
}
```

节点参数主要包括机器人服务器的地址(告知上层)、发送mqtt的地址、与发送心跳的地址，相关参数在robot_message_bridge/config/param.json

```
{
    "dms_url":"http://xxx.xxx.x.xxx/apixxxx",
    "mqtt_ip":"xxx.xxx.x.xxx",
    mqtt_port:12345
}
```

2.task_communication

task_communication节点接收move、charge、uncharge等指令，move指令直接进行路径规划后进行移动，charge指令首先移动到充电站（距离充电桩10cm左右），然后进行视觉的精确定位，精确定位后插入充电桩，开启充电监听线程，充电监听线程主要应对航发底盘各种充电问题设置的，具体功能包括充电到100后断电，直到下降到90后继续充电，充电异常的时候把充电继电器的开关开关一次等操作。

节点运行的参数通过roslaunch以及gflag传入,主要参数aruco_id为充电桩重定位的aruco码id，log_dir_path日志的保存地址，station_pose_path站点的文件地址，camera_pose_path充电桩相机的位姿文件保存地址。

节点启动

```
roslaunch task_communication task_communication.launch
```

```
<launch>

  <node name="platform_communication_node" pkg="task_communication"
type="platform_communication_node" args=
          "-aruco_id 47
    -log_dir_path $(find task_communication)/logfiles
    -station_pose_path $(find
task_communication)/station_cfg/hefei_xuexiao/station_english_new.json
    -camera_pose_path $(find
task_communication)/station_cfg/hefei_xuexiao/camera_pose.json"
        output="screen">
  </node>
  <node name="record_platform_pose_node" pkg="task_communication"
type="record_platform_pose_carto.py" output="screen" />
</launch>
```

工作站标点节点，主要参数包括保存站点位姿的文件地址以及相机位姿的文件地址，节点启动后按提示标点即可

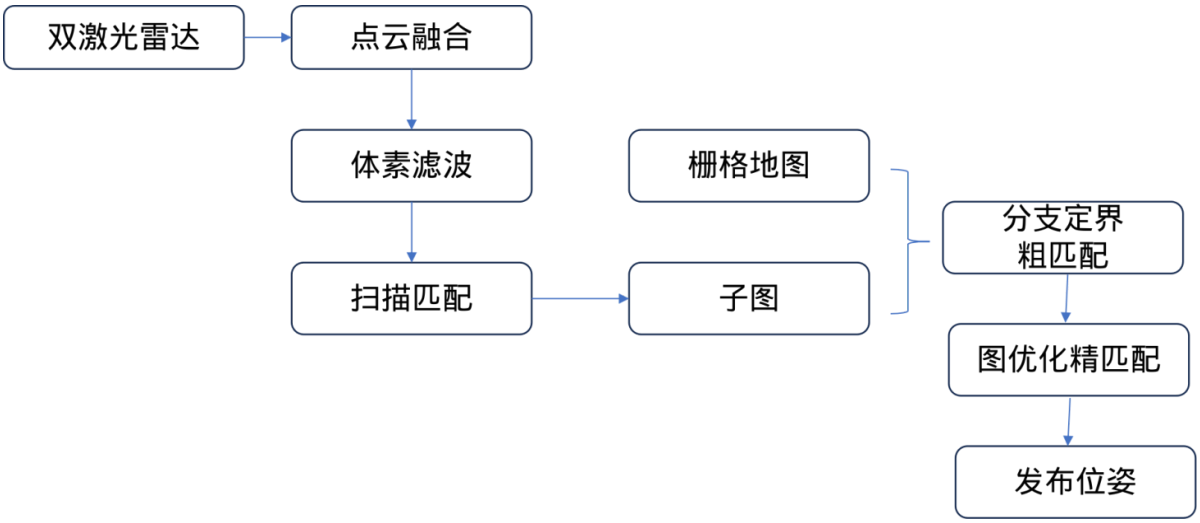
```
roslaunch task_communication record_pose.launch
```

```
<launch>


    <node name="hf_record_pose_node" pkg="task_communication"
type="hf_record_pose_node" args=
    "
    -stationJsonName $(find
task_communication)/station_cfg/hefei_xuexiao/station_english_new.json
    -cameraJsonName $(find
task_communication)/station_cfg/hefei_xuexiao/camera_pose.json"
          output="screen">
    </node>
</launch>
```

3.cartographer建图和定位

建图和定位都是基于cartographer改的，具体的流程代码中有注释。主要更改了两个地方，一个是建图时ros格式地图的发布格式，使得栅格数值和move_base使用的格式一致，路径规划的静态地图使用此生成的地图。二是纯定位时增加了提供初始位姿的功能，使得机器人开机后能够快速准确的定位。只使用了激光雷达（使用里程计效果差，cartographer太相信松耦合的结果），双激光雷达进行icp融合后送入框架。



建图流程

```
roslaunch communication_rs485 platform_init.launch #启动底盘

roslaunch cartographer_ros aichem_carto.launch #建图

rosservice call /finish_trajectory 0 #终止建图

rosservice call /write_state "{filename:
'${HOME}/speed/aichem_cartographer/src/cartographer_ros/cartographer_ros/configur
ation_files/aichem_map1.pbstream'}"#保存pbstream地图

rosrun map_server map_saver -f aichem_map1 rosrun cartographer_ros
cartographer_pbstream_to_ros_map -
pbstream_filename=${HOME}/speed/aichem_cartographer/src/cartographer_ros/cartogra
pher_ros/configuration_files/aichem_map1.pbstream -
map_filestem=${HOME}/speed/aichem_cartographer/src/cartographer_ros/cartographer_
ros/configuration_files/aichem_map1 -resolution=0.02 #把pbstream地图转换成ros格式地图
以便路径规划使用
```

aichem_carto.launch主要包括是不是重定位参数localization，scan消息的名称，其他参数放在aichem_carto.lua(参数已经调好，**注意和定位的参数不完全相同**)中

```
<launch>
    <param name="/use_sim_time" value="false" />

    <!--    是否重定位-->
    <param name="/localization" type="bool" value = "false"/>

    <node name="cartographer_node" pkg="cartographer_ros"
          type="cartographer_node" args="
          -configuration_directory $(find cartographer_ros)/configuration_files
```

```
          -configuration_basename aichem_carto.lua"
          output="screen">
        <remap from="scan" to="/scan_full_filtered" />
        <remap from="odom" to="/odom" />

    </node>

    <node name="cartographer_occupancy_grid_node" pkg="cartographer_ros"
          type="cartographer_occupancy_grid_node" args="-resolution 0.02" />

    <node name="rviz" pkg="rviz" type="rviz" required="true"
          args="-d $(find
cartographer_ros)/configuration_files/aichem_carto.rviz" />
</launch>
```

```
include "map_builder.lua"
include "trajectory_builder.lua"

options = {
  map_builder = MAP_BUILDER,                -- map_builder.lua的配置信息
  trajectory_builder = TRAJECTORY_BUILDER,  -- trajectory_builder.lua的配置信息

  map_frame = "map",                        -- 地图坐标系的名字
  tracking_frame = "hf_base_link",              -- 将所有传感器数据转换到这个坐标系下
  published_frame = "odom",            -- tf: map -> footprint,    指定你要把map连接
到哪个坐标系
  odom_frame = "odom",                      -- 如果provide_odom_frame为true，carto
发布里程计的坐标系名字
  provide_odom_frame = false,               -- 是否需要carto提供map->odom的连接，是否
提供odom的tf，如果为true,则tf树为map->odom->footprint,
                                            -- 如果为false tf树为map->footprint
  publish_frame_projected_to_2d = false,    -- 是否将坐标系投影到平面上
  use_pose_extrapolator = false,            -- 发布tf时是使用pose_extrapolator的位姿
还是前端计算出来的位姿

  use_odometry = false,                     -- 是否使用里程计,如果使用要求一定要有odom
的tf
  use_nav_sat = false,                      -- 是否使用gps
  use_landmarks = false,                    -- 是否使用landmark
  num_laser_scans = 1,                      -- 是否使用单线激光数据
  num_multi_echo_laser_scans = 0,           -- 是否使用multi_echo_laser_scans数据
  num_subdivisions_per_laser_scan = 1,      -- 1帧数据被分成几次处理,一般为1
  num_point_clouds = 0,                     -- 是否使用点云数据

  lookup_transform_timeout_sec = 0.2,       -- 查找tf时的超时时间
  submap_publish_period_sec = 0.3,          -- 发布数据的时间间隔
  pose_publish_period_sec = 5e-3,
  trajectory_publish_period_sec = 30e-3,

  rangefinder_sampling_ratio = 1.,          -- 传感器数据的采样频率
  odometry_sampling_ratio = 1.,
  fixed_frame_pose_sampling_ratio = 1.,
  imu_sampling_ratio = 1.,
  landmarks_sampling_ratio = 1.,
}
```

```
MAP_BUILDER.use_trajectory_builder_2d = true

TRAJECTORY_BUILDER_2D.use_imu_data = false
TRAJECTORY_BUILDER_2D.min_range = 0.3
TRAJECTORY_BUILDER_2D.max_range = 100.
TRAJECTORY_BUILDER_2D.min_z = -0.2
TRAJECTORY_BUILDER_2D.max_z = 1.4
TRAJECTORY_BUILDER_2D.min_range=0
TRAJECTORY_BUILDER_2D.max_range=30
TRAJECTORY_BUILDER_2D.voxel_filter_size = 0.02

TRAJECTORY_BUILDER_2D.use_online_correlative_scan_matching = true
TRAJECTORY_BUILDER_2D.ceres_scan_matcher.occupied_space_weight = 10.
TRAJECTORY_BUILDER_2D.ceres_scan_matcher.translation_weight = 1.
TRAJECTORY_BUILDER_2D.ceres_scan_matcher.rotation_weight = 1.

TRAJECTORY_BUILDER_2D.submaps.num_range_data = 40.
TRAJECTORY_BUILDER_2D.submaps.grid_options_2d.resolution = 0.02


TRAJECTORY_BUILDER_2D.submaps.range_data_inserter.probability_grid_range_data_ins
erter.hit_probability = 0.6
TRAJECTORY_BUILDER_2D.submaps.range_data_inserter.probability_grid_range_data_ins
erter.miss_probability = 0.49

POSE_GRAPH.optimize_every_n_nodes = 80.
POSE_GRAPH.constraint_builder.sampling_ratio = 0.3
POSE_GRAPH.constraint_builder.max_constraint_distance = 15.
POSE_GRAPH.constraint_builder.min_score = 0.48
POSE_GRAPH.constraint_builder.global_localization_min_score = 0.60

return options
```

定位流程，launch文件中包含ros格式的地图地址，pbstream地图的地址，初始的位姿等，纯定位的参数包含在aichem_localization.lua下

```
roslaunch communication_rs485 platform_init.launch #启动底盘

roslaunch cartographer_ros aichem_localization.launch #定位
```

```
<launch>

  <arg name="map_file" default="$(find
cartographer_ros)/configuration_files/aichem_map1.yaml"/>
  <node name="map_server" pkg="map_server" type="map_server" args="$(arg
map_file)" />

  <!-- pbstream的地址与名称 -->
  <arg name="load_state_filename" default="$(find
cartographer_ros)/configuration_files/aichem_map1.pbstream"/>

  <!-- 不使用bag的时间戳 -->
  <param name="/use_sim_time" value="false" />
```

```
  <!--  重定位初始位姿-->
  <param name="/localization" type="bool" value = "true"/>
  <rosparam command="load" file="$(find
task_communication)/station_cfg/current_pose.yaml"></rosparam>

  <!-- 启动cartographer -->
  <node name="cartographer_node" pkg="cartographer_ros"
      type="cartographer_node" args="
          -configuration_directory $(find cartographer_ros)/configuration_files
          -configuration_basename aichem_localization.lua
          -load_state_filename $(arg load_state_filename)"
      output="screen">
    <remap from="scan" to="/scan_full_filtered" />
    <remap from="odom" to="/odom" />
  </node>


  <!-- 启动rviz -->
  <node name="rviz" pkg="rviz" type="rviz" required="true"
      args="-d $(find cartographer_ros)/configuration_files/aichem_carto.rviz" />


</launch>
```

4.路径规划及其他

路径规划没做变动只优化了一些参数，减小局部地图范围、减小规划频率、增加运动恢复等，使得运行丝滑一点。还有手柄操作等一些其他小功能。

节点的启动为,代价地图参数与movebase参数为costmap_common_params.yaml, move_base_params.yaml

```
roslaunch hf_navigation hf_navigation.launch
```

```
#costmap_common_params.yaml

footprint_padding: 0.01

plugins:
- {name: obstacles_layer, type: "costmap_2d::ObstacleLayer"}
- {name: inflater_layer, type: "costmap_2d::InflationLayer"}

obstacles_layer:
  observation_sources: scan
  scan: {
    sensor_frame: scan_merge,
    data_type: LaserScan,
    topic: /scan_full_filtered,
    marking: true,
    clearing: true,
    min_obstacle_height: -0.4,
```

```
     max_obstacle_height: 2.0,
     obstacle_range: 2.5,
     raytrace_range: 3,
     inf_is_valid: true}

# 膨胀半径, 根据机器人大小设置,之前有碰撞应该是这个的问题
inflater_layer:
 inflation_radius: 1 #0.45
```

```
#move_base_params.yaml

shutdown_costmaps: false

controller_frequency: 15.0
controller_patience: 15.0

planner_frequency: 0 #路径规划的频率
planner_patience: 1.0 #等待多久进行地图清理恢复操作

oscillation_timeout: 0.0
oscillation_distance: 0.5

recovery_behavior_enabled: true #通过清理地图进行恢复
clearing_rotation_allowed: false #通过旋转进行恢复, 比较激进

conservative_reset_dist: 0.5 #
max_planning_retries : 2

recovery_behaviors:
  -name: 'conservative_reset'
  type: 'clear_costmap_recovery/ClearCostmapRecovery'


recovery_behaviors:  # 自恢复行为
  - name: 'conservative_reset'
    type: 'clear_costmap_recovery/ClearCostmapRecovery'


conservative_reset:
  reset_distance: 0.5
  #layer_names: [static_layer, obstacle_layer, inflation_layer]
  layer_names: [obstacles_layer]  #名字要与costmap_common_params.yaml中的一致
```