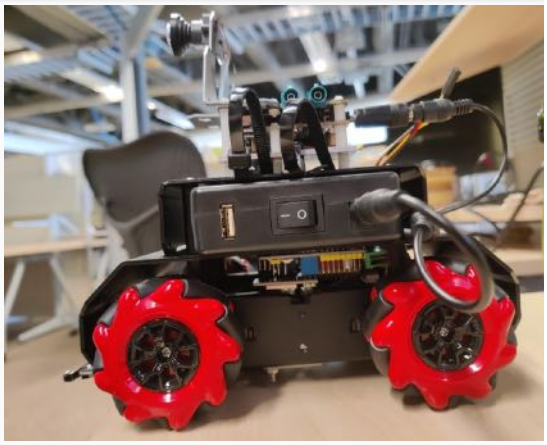
CSE276A: Mbot Mega with RB5 Demo (ROS)

Henry (Hengyuan Zhang)
Credit to David Paz for Initial version

Overview

- Mbot Mega, RB5 and ROS
- ROS packages
- Running ROS Master
- Node graphs
- Control for Mbot

Mbot System Overview



- Robot Body: Mbot Mega
  - Omnidirectional Drive
  - battery
  - Joystick controller
- Robot Head: Qualcomm RB5
  - Kryo 585 CPU
  - 8GB memory
  - Qualcomm Adreno 650 GPU
  - 2 Cameras (Wide Angle and tracking) and IMU
- Robot Brain: Software
  - System: Ubuntu 18.04
  - ROS Dashing/Melodic (need to install)
  - custom ROS node: joystick control, apriltag detection

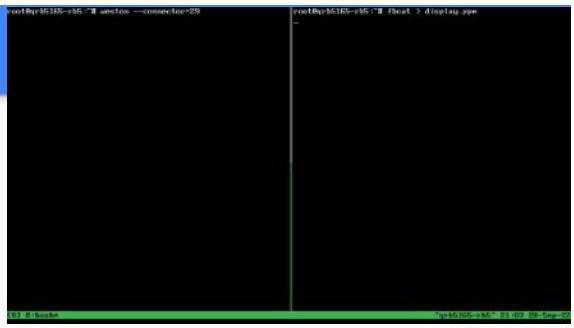
RB5 USB-C connection

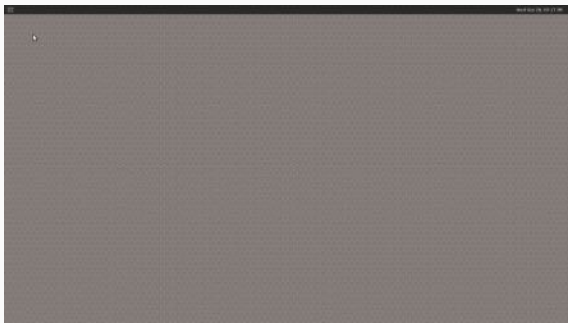
- Once the device is flashed, you can connect to the RB5 through USB-C and use adb
  - adb stands for Android Debug Bridge, it is a versatile command-line tool that lets you communicate with a device.
  - adb devices will show the devices being connected

    ```
    adb devices
    List of devices attached
    e3edf963 device
    ```
  - use adb shell to start a terminal within RB5
    - then you can connect RB5 to the internet through nmcli
      - nmcli –ask dev wifi connect “WiFi-SSID”
    - after which you can obtain the ip address of your RB5
      - ifconfig

RB5 CLI and GUI

- CLI
  - Command Line Interface
  - tmux: creating multiple terminal sections
- weston display
  - a minimal graphical interface
  - `weston -connector=29`
  - weston-terminal: click the icon on the top left

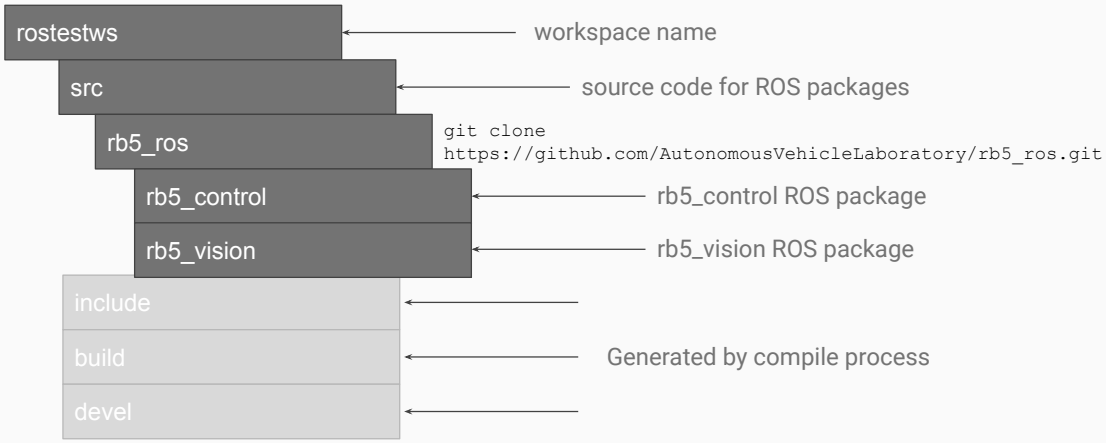




RB5 SSH

- Connecting over a WiFi /LAN
  - Set-up WiFi connection
  - Ensure client is within the same network (IP address assigned by dynamically)
    - connect rb5 to the same wifi as your host machine (non use
      - using wpa_supplicant.conf (see assembly instructions)
      - or you can use nmcli
  - Access over SSH
    - ssh root@192.168.x.x
    - default password: oelinux123

ROS packages



ROS packages (rb5_control / rb5_vision)

- Add ROS Melodic to your path

```
$ source /opt/ros/melodic/setup.bash
```

- Build ROS packages

```
$ cd /path/to/rostestws/
$ catkin_make
$ catkin_make --only-pkg-with-deps package_name
```

- Add workspace to your path

```
$ source /path/to/rostestws/devel/setup.bash
```

Running ROS Master and jetbot_ros

- Initialize an instance of ROS Master

```
$ roscore
```

- Run jetbot_ros nodes

```
$ rosrun rb5_control mpi_control_node.py
$ rosrun joy joy_node
$ roslaunch rb5_vision rb_camera_main_ocv.launch
```

jetbot motors node

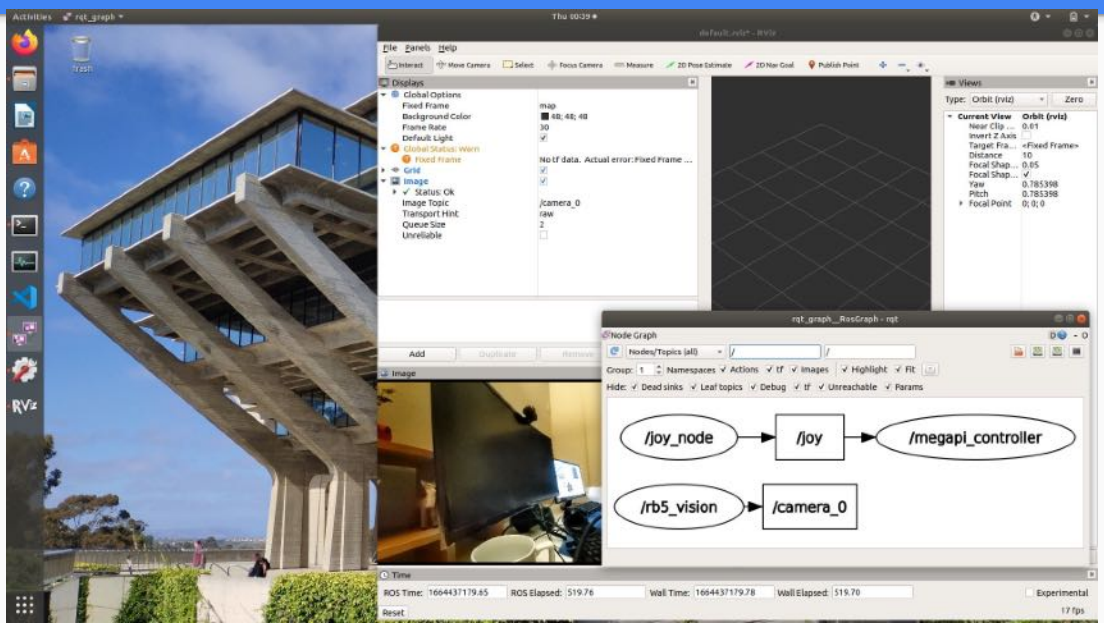
joystick node

camera node

- Visualizing camera data with RViz (requires gnome desktop)

```
$ sudo apt-get install ros-melodic-rviz
$ rviz
```

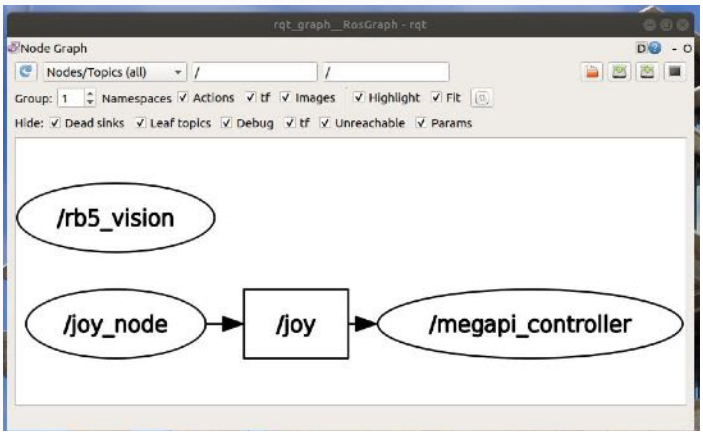
Running ROS Master and jetbot_ros



Node Graphs

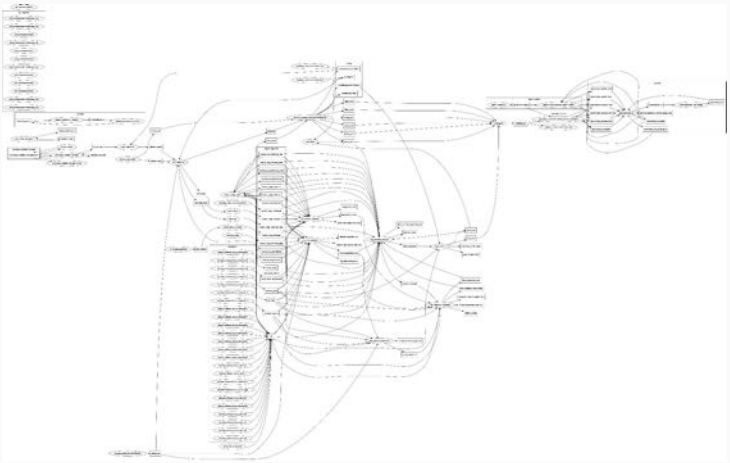
- Visualize node connectivity using rqt
  - `$ sudo apt-get install ros-melodic-rqt-graph`
- Useful for debug purposes

`$ rqt_graph`



Node Graphs

- However, not always ideal for large projects
- Other tools:
  - `rosnode`
    - `list`
    - `info`
  - `rostopic`
    - `list`
    - `info`
    - `hz`
    - `echo`
  - `rosbag`
    - `record`
    - `play`
    - `play -r 0.2`



Control for Mbot Mega

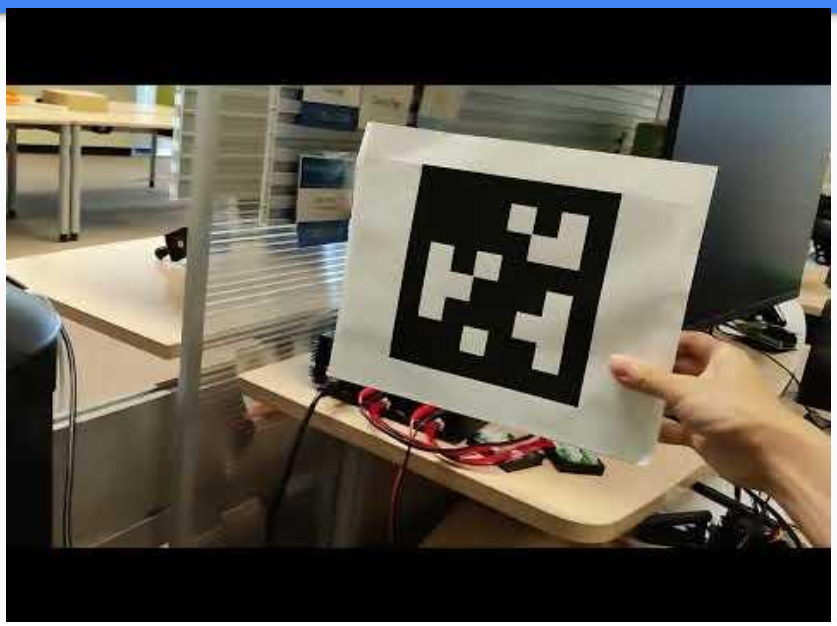
- The mpi_control node subscribes to /joy topic published by joy_node
  - /joy
    - axes[0]: forward, backward
    - axes[1]: slide left, right
    - axes[2]: rotate clockwise, counterclockwise

Control for Mbot Mega (Code segments)

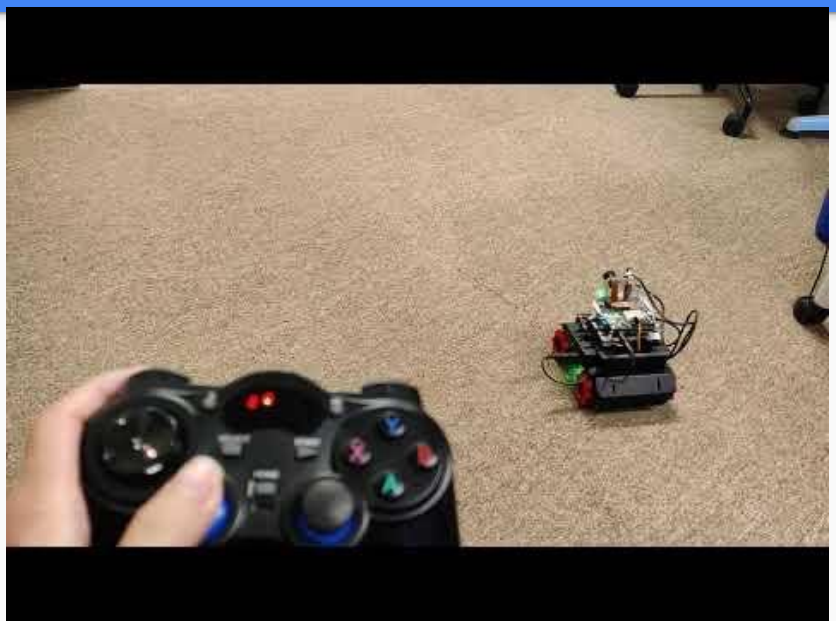
```
def joy_callback(self, joy_cmd):
    v_slide = self.v_max_slide * joy_cmd.axes[0]
    v_straight = self.v_max_straight * joy_cmd.axes[1]
    v_rotate = self.v_max_rotate * joy_cmd.axes[2]
    if abs(joy_cmd.axes[0]) <= 0.1 and abs(joy_cmd.axes[1]) <= 0.1:
        self.mpi_ctrl.carStop()
    elif abs(joy_cmd.axes[0]) <= 0.1:
        self.mpi_ctrl.carStraight(v_straight)
    elif abs(joy_cmd.axes[1]) <= 0.1:
        self.mpi_ctrl.carSlide(v_slide)
    else:
        self.mpi_ctrl.carMixed(v_straight, 0, v_slide)

mpi_ctrl_node = MegaPiControllerNode()
ospy.init_node('megapi_controller')
rospy.Subscriber('/joy', Joy, mpi_ctrl_node.joy_callback,
queue_size=1)
rospy.spin()
```

```
MFR = 2     # port for motor front right
MBL = 3     # port for motor back left
MBR = 10    # port for motor back right
MFL = 11    # port for motor front left

def setFourMotors(self, vfl=0, vfr=0, vbl=0, vbr=0):
      self.bot.motorRun(self.mfl,vfl)
      self.bot.motorRun(self.mfr,vfr)
      self.bot.motorRun(self.mbl,vbl)
      self.bot.motorRun(self.mbr,vbr)

   def carStop(self):
      self.setFourMotors()

   def carStraight(self, speed):
      self.setFourMotors(-speed, speed, -speed, speed)

   def carRotate(self, speed):
      self.setFourMotors(speed, speed, speed, speed)

   def carSlide(self, speed):
      self.setFourMotors(speed, speed, -speed, -speed)

   def carMixed(self, v_straight, v_rotate, v_slide):
      self.setFourMotors(
          v_rotate-v_straight+v_slide,
          v_rotate+v_straight+v_slide,
          v_rotate-v_straight-v_slide,
          v_rotate+v_straight-v_slide
      )
```

Apriltag detection and joystick control



joystick control



Additional Documents

- RB5 Tutorial
  - https://autonomousvehiclelaboratory.github.io/RB5_Robotics_Tutorials/
- Qualcomm developer Network RB5 guide
  - https://developer.qualcomm.com/qualcomm-robotics-rb5-kit
- Qualcomm developer Network RB5 Forum
  - https://developer.qualcomm.com/forums/hardware/robotics/qualcomm-robotics-rb5-kit
- Mbot Mega Tutorial Page
  - https://support.makeblock.com/hc/en-us/sections/1500001152162-mBot-Mega
- Mbot Mega interface:
  - https://www.yuque.com/makeblock-help-center-en/mblock-5/megapi-pro
- ROS Tutorials
  - http://wiki.ros.org/ROS/Tutorials
- Questions?