Notes for Team 3

Testing for moving arm:

1. ros2 launch xarmrob xarm_keyboard.launch.py
2. ros2 run xarmrob final_move_keyboard

Block location: [0.25, 0, 0.03]
Camera location: [0.1, 0, 0.03]

'bucket1': [0.02, -0.165, 0.11]
'bucket2': [0.02, 0.165, 0.11]

Bucket1:

Final Project: See Green Ball -> Dunk in Green Basket, do same for Red and Blue

What we need:

Materials we need:

- Color sensor
- Red Ball, Green Ball, Blue Ball (can be box or any other graspable object)
- Red Bucket, Green Bucket, Blue Bucket (would be nice to be colored for visual)

What we need to do:

1. Begin planning for two buckets and two colors. Add the third color in at the end.
2. Map out robot workspace and pre-determine x,y coordinates for red bucket, blue bucket, place in front of color sensor and ball location.
3. Find an optimal way to grip the block.
4. Using these coordinates, create nodes which move from one location to another
5. We will have if-statements on the input received from color sensor
6. Choose to either loop once hand has finished dumping the object, or stop and we need to re-run the command (maybe stop is easier).

Work that can be done asynchronously/parallel;

Physical work on Robot	Code	Project Proposal (slideshow and demo probably done last)
• Gathering materials • Wiring the color sensor • Mounting the color sensor • Mapping out where to put buckets/block/color sensor in the workspace	• ROS2 node for putting the picking up the box, putting it close to color sensor, moving from color sensor to each of box, return to neutral/box pickup • ROS2 node for integrating with sensor	• Project proposal writeup which is 3 pages • Slideshow which is presented in 3.5 minutes • Demo video

Tuesday: 12/3 - Begin code for moving arm and identifying x,y coordinates. (put a piece of paper under the robot and draw it out?). Make a plan for how to get materials (drive and find a toy store?)
Thursday: 12/5 - Have materials to test grasping and manually moving to locations
Weekend: make a ROS2 node that picks up blocks and moves them throughout the cycle.
Tuesday: 12/10 - last day of classes. Integrate node with readings from color sensor
Thursday: 12/12 - meet to finish slides, demo video, writeup
Friday: 12/13 - Submit Project Proposal with code, slides, and writeup
Saturday: 12/14 - Demo Day

Helpful forward kinematics video: https://www.youtube.com/watch?v=mO7JJxaVtkE

Limit values:
600

If colcon build fails due to being unable to find a file, rm -rf build/

VNC viewer:
User: pi
password: raspberry03

WinSCP (File Transfer Tool):
Hostname: get from realvnc -> connection details -> copy and paste the IP address
User: pi
password: raspberry03

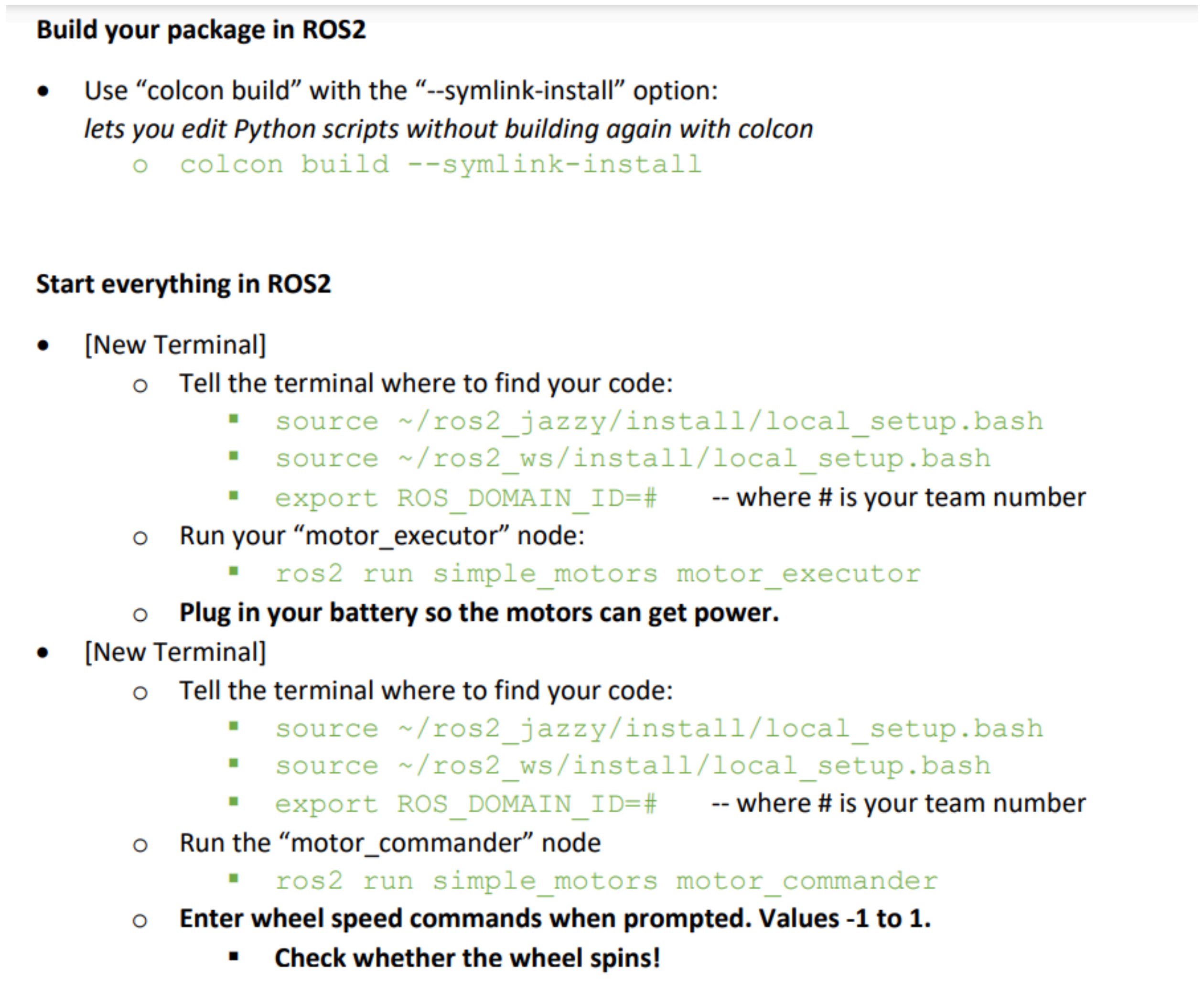
Motors run py (to test 2 motors)

Sensor Debug:
Python3 raw_sensors_debug.py | grep A0
Or grep E1 (right) or E0 (left) for wheel encoders

Motor Instructions

If no executable found for ROS, change setup.py to add the file names

Make new terminal after build, since it sources again



Build your package in ROS2

- Use “colcon build” with the “--symlink-install” option:
  lets you edit Python scripts without building again with colcon
  - `colcon build --symlink-install`

Start everything in ROS2

- [New Terminal]
  - Tell the terminal where to find your code:
    - `source ~/ros2_jazzy/install/local_setup.bash`
    - `source ~/ros2_ws/install/local_setup.bash`
    - `export ROS_DOMAIN_ID=#` -- where # is your team number
  - Run your “motor_executor” node:
    - `ros2 run simple_motors motor_executor`
  - **Plug in your battery so the motors can get power.**
- [New Terminal]
  - Tell the terminal where to find your code:
    - `source ~/ros2_jazzy/install/local_setup.bash`
    - `source ~/ros2_ws/install/local_setup.bash`
    - `export ROS_DOMAIN_ID=#` -- where # is your team number
  - Run the “motor_commander” node
    - `ros2 run simple_motors motor_commander`
  - **Enter wheel speed commands when prompted. Values -1 to 1.**
    - **Check whether the wheel spins!**

To convert from analog to distance on ultrasonic sensor:

```
analog_volts = analog_level * (3.3/1023)
        distance_meters = analog_volts / (1.0/1.0) * 1.0
    #####

        # Create a Message that will hold the out-going data
        A0_proc = Float32()

        # Pack the message with the processed data
        A0_proc.data = distance_meters

        # Publish the newly packed Message
        self.publisher_A0_proc.publish(A0_proc)
```

To run sensor to motor:

```
Python sensors_node.py  //gets sensor data
Python sensors_processor.py //converts sensor data to meters
Python sensor_to_motor_commands.py //executes motor based on meter data
```

9/26/2024

How to change the file the runs at the start

```
source ~/ros2_jazzy/install/local_setup.bash
source ~/ros2_ws/install/local_setup.bash
export ROS_DOMAIN_ID_3
```

Go to raspberry pi then use the nano command below

```
nano ~/.bashrc
```

Paste the following lines at the bottom of it

```
source ~/ros2_jazzy/install/local_setup.bash
source ~/ros2_w2/install/local_setup.bash
export ROS_DOMAIN_ID_3
```

Ros2 run turtlesim to test if it worked

How to read info from sensors

(2 ways)

Raw data

```
Ros2 run basic_motors_and_sensors sensors_node
Ros2 topic echo /sensors_A0
```

Processed data

```
Ros2 run basic_motors_and_sensors sensors_processor
Ros2 topic echo /sensors_A0_proc
```

```
Ros2 run rqt_graph rqt_graph
```

```
Ros2 run basic_motors_and_sensors servos_executor
```

```
Ros2 run basic_motors_and_sensors
```

Green yrllloe pink

500-2400 is the range of the sensor

- 50 times a second, so 20000 micorseconds

```
Ros2 run basic_motors_and_sensors motor_executor
```

```
Ros2 run basic_motors_and_sensors motor_commander
```