# Module 6 - IMU


**Lesson Objectives:**

1. Learn basic fundamentals of an inertial measurement unit (IMU).
2. Utilize Redshift Lab's Serial Interface tool to calibrate the IMU.

**Agenda:**

1. Purpose
2. IMU
3. ICE 6
4. Summary
5. Assignments
6. Next time

# 1 Purpose

In practice, an IMU provides the yaw velocity and orientation. These values are part of the robot state and allow the robot to navigate more accurately. Combined with data from the tachometers these values provide the odometry of the robot to estimate change in position over time. We will primarily use the IMU to perform 90 and 180 degree turns.

# 2 IMU

`https://www.pololu.com/product/2764`:

The UM7 orientation sensor from Redshift Labs is an Attitude and Heading Reference System (AHRS) that contains a three-axis accelerometer, rate gyro, and magnetometer. It combines this data using an Extended Kalman Filter to produce attitude and heading estimates. Unlike a typical inertial measurement unit (IMU), which only provides raw sensor readings, the UM7 features an onboard microcontroller that combines sensor data using a sophisticated Extended Kalman Filter (EKF) to generate orientation estimates 500 times a second.

**Pololu USB AVR Programmer v2.1** (`https://www.pololu.com/product/3172`)

The Pololu USB AVR Programmer v2.1 is a compact, low-cost in-system programmer (ISP). The programmer provides an interface for communicating with the UM7 orientation sensor.

# 3 ICE 6.

This ICE has two parts, first you will calibrate the IMU sensor and then you will create a subscriber that displays the orientation data provided by the IMU.

**Calibration** As described above, there are a number of different sensors that work together to provide the attitude and heading estimates for the UM7 Orientation Sensor. These sensors are sensitive to magnetic fields which are unique to locale and device. As you will learn in future ECE classes, all electronic devices create small magnetic fields. Even electrons traveling over a wire create magnetic fields. The UM7 Orientation Sensor is strategically placed in the center of the robot for best attitude and heading performance, however, this location is also in the center of a number of magnetic fields. We can use Redshift Lab's Serial Interface Software to calibrate the UM7 to provide a more accurate attitude and heading reading.

## 3.1 Calibrate the UM7 Orientation Sensor:

`https://redshiftlabs.com.au/support-services/um7-calibration-procedure/`

1. Download the Serial Interface Software Version 3.1.5 to your **student laptop**: `https://redshiftlabs.com.au/support-services/serial-interface-software/`
2. Plug the Pololu USB AVR Programmer v2.1 into your computer. You may want to move the USB cable so it is easier to turn rotate the robot when calibrating.
3. Start Interface Software.

   Launch Redshift Labs serial interface software by double clicking the icon on your desktop or finding it in your browser.
4. Set COM Ports

   You will need to configure which virtual COM port has been allocated to your USB serial converter PCB. Select **"Serial Interface > Serial Settings > Port > COMX** (where X is the allocated COM port number on your PC, in my case, it is the higher number).
5. Connect to your UM7

   Select **"Serial Interface > Serial Settings > Connect"**.
6. Read Data

Read the data from within your UM7 or UM7-LT, select **"Serial Interface > Configuration > Read"**. A progress bar should load and once it's finished the configuration data should be displayed. Press the + button next to **"Broadcast Rates – Raw Data"** now select **"All Raw Broadcast Rate"**, verify that it's set to 20 Hz. If not, type 20 into the corresponding text box and press **"Serial Interface > Configuration > FLASH"**.

7. Plot Data

   (a) From the tabs select "Data"

   (b) Click the + icon next to **"Euler Angles"**

   (c) Tick Roll, Pitch, and Yaw checkboxes

   (d) Press "Serial Interface > Data > Create Graph from Selected".

   (e) A real time graph should be displayed in a separate window of Roll, Pitch, and Yaw. e suggest using your mouse to expand the window a little.

8. Start Magnetometer Calibration

   To begin magnetometer calibration select **"Serial Interface > Mag Calibration > Start Data"** you should notice that the **"Collected Data Points:"** begin counting up. While the software is taking samples, rotate your UM7-LT around in a spherical position capturing as many different samples of the surrounding magnetic field as possible. Keep well away from anything that will produce magnetic distortions in your sample set, i.e. ferrous metal or powerful magnets.

9. Stop Collecting Data

   When you see that the **"Collected Data Points:"** number turns green, that's an indication that you have collected the minimum samples required to perform a calibration. I usually collect about 500 samples. Now select **"Serial Interface > Mag Calibration > Stop Data"**.

10. Compute Biases

    You are now ready to compute the biases for calibration, select **"Serial Interface > Mag Calibration > Compute"**. You should now see some computed data in the **"Calibration Matrix"**. Select **"Serial Interface > Mag Calibration > Write to RAM"**.

11. Write Calibration Data

    You will want your calibration data to persist, so now select **"Serial Interface > Configuration > FLASH"**. This will write the calibration data into your UM7-LT, the calibration matrix will be stored even if the power is turned off and back on.

12. Zero Rate Gyros

    This is needed in the later sets of firmware. It will initialize your new calibration correctly. Select **"Serial Interface > Commands -> Zero Rate Gyros"**.

13. Set Magnetic Reference Vector

    Point your UM7-LT exactly North on the UM7 (North on the UM7 is left of front on the robot), now press **"Serial Interface > Commands > Set Mag Reference Vector"**.

14. Reset EKF

    You will now need to reset the EKF for the process to work correctly. You may notice that the graphs on your data plot are displaying incorrectly. Select **"Serial Interface > Commands > Reset EKF"**. Now look at the data plot of Roll, Pitch and Yaw. If you point the X orientation of the sensor North and hold the unit as level as possible, you should see the Roll, Pitch and Yaw converge to zero.

### 3.2 Ensure um7 ROS package is installed:

15. The UM7 ROS package is preinstalled on your robot. But as always, trust, but verify. Open a new terminal on your **Master** and use SSH to create a secure shell into the robot.

16. In your SSH terminal, ensure the ROS package, `um7`, is installed on your **Robot**:

```
pi@robot:~$ rospack find um7
```

If installed, the command should return the absolute path to the pakcage, similar to:

```
/home/pi/robot_ws/src/um7
```

If the command instead returns an error, then you need to install the package. This package has not been compiled for ROS Noetic, so you must download the source code into the workspace source folder. When you install a ROS package from source, you have to manually download any dependencies (using rosdep) and then compile it.

```
pi@robot:~$ cd ~/robot_ws/src
pi@robot:~$ git clone https://github.com/ros-drivers/um7.git
pi@robot:~$ cd ~/robot_ws
pi@robot:~$ rosdep install --from-paths src --ignore-src -r -y
pi@robot:~$ catkin_make
```

****NOTE**** It turns out the developers for the um7 ROS package did not adequately set up the dependencies. So you should have seen an error when running `rosdep` and `catkin_make` about a `serial` package not installed. To fix this dependency issue you need to install a ROS serial package to your source:

```
pi@robot:~$ cd ~/robot_ws/src
pi@robot:~$ git clone https://github.com/wjwwood/serial.git
pi@robot:~$ cd ~/robot_ws
pi@robot:~$ rosdep install --from-paths src --ignore-src -r -y
pi@robot:~$ catkin_make
```

### 3.3 Write the subscriber:

17. In a new terminal on the **Master**, create an `ice6` package on the and make it.

```
pi@master:~$ cd ~/master_ws/src
pi@master:~$ catkin_create_pkg ice6 geometry_msgs rospy
pi@master:~$ cd ~/robot_ws
pi@master:~$ catkin_make
pi@master:~$ source ~/.bashrc
```

18. Create an IMU node.

```
pi@master:~$ roscd ice6/src
pi@master:~$ touch imu_sub.py
```

19. Copy the below code.

You can edit via the terminal using nano, but it is often easier to use a GUI editor since we can. Browse to the subscriber and double-click. This will open the file in thonny (if it is open in any other editor, stop, raise your hand, and get help from an instructor)

```python
#!/usr/bin/env python3
import rospy
import math
from geometry_msgs.msg import Vector3Stamped
from um7.srv import *

class IMU:
    """Class to read orientation data from UM7-LT"""
    def __init__(self):
        self.yaw = 0

        # reset the IMU
        self.init_imu()

        # subscribe to the imu topic that is published by the
        # pre-built ROS imu node from the um7 package
        rospy.Subscriber('imu/rpy', Vector3Stamped, self.callback_IMU)

        self.ctrl_c = False
        rospy.on_shutdown(self.shutdownhook)

    # function to reset the gyro, filter, and magnetic reference
    def init_imu(self):
        rospy.wait_for_service('imu/reset')
        try:
            reset = rospy.ServiceProxy('imu/reset', Reset)
            # (gyros, EKF, mag_ref)
            reset(True, True, True)
        except rospy.ServiceException as e:
            print("service call failed: %s"%e)

    # The IMU provides yaw from -180 to 180. This function
    # converts the yaw to 0 to 360
    def yawConvert (self, yaw):
        return 360 + yaw if yaw < 0 else yaw

    # Print the current Yaw
    def callback_IMU(self, data):
        if not self.ctrl_c:
            # convert yaw from radians to degrees and
            # from -180 to 180 to 0 to 360
            self.yaw = self.yawConvert(data.vector.z*180/math.pi)
            print(f"Current heading is {self.yaw!s} degrees.")

    # clean shutdown
    def shutdownhook(self):
        print("Shutting down the IMU subscriber")
        self.ctrl_c = True

if __name__ == '__main__':
    rospy.init_node('imu', anonymous=True))
    IMU()
    rospy.spin()
```

20. Save and exit
21. Make the node executable

```
pi@master:~$ chmod +x imu_sub.py
```

22. Select the terminal with the secure shell to the `Robot` and display the serial ports connected to the robot:

    ```
    pi@robot:~$ python3 -m serial.tools.list_ports -v
    ```

    ****HINT**** Write this command down as you will use it often to determine which devices are connected to your ports!

23. Run the IMU publisher connecting to the Pololu USB AVR Programmer with the lowest port number.

    ```
    pi@robot:~$ rosrun um7 um7_driver _port:=/dev/ttyACM0
    ```

24. Open a new terminal on the **Master** and run the `imu_sub.py` node:

    ```
    pi@master:~$ rosrun ice6 imu_sub.py
    ```

25. Rotate the robot and observe the output.

## 4 Summary.

In this lesson you learned how to calibrate the IMU and get orientation data using the pre-built `um7` ROS package. In the lab that corresponds to this lesson you will apply this knowledge to turn the robot in 90 and 180 degree turns.

## 5 Assignments.

☐ Lab 2. Due X X X at 2359 (Demo via Teams Lab 2 channel, Report via Gradescope)

## 6 Next time.

- Lab 2.