**LAB\_1** - EECE 5554 Robotics Sensing and Navigation

**Term:** Spring 2020

**Deadline**: 11:59 PM on Thursday, Jan 30, 2020

**Hardware/ Sensors**

1. USB based GNSS puck, issued one per team.

**Data collection Policy**

Everyone in the team needs to

1. Write their own device driver for data acquisition.
2. Collect data individually.

Plan data collection schedules in your team, so everyone gets a fair amount of sensor time to collect data and of course you can talk to your teammates and ask questions on Piazza.

**What to Submit for Lab 1 ?**

1. **Write Device Driver for GNSS puck to parse $GPGGA**

As we will see in class the GNSS puck provides a number of differently formatted messages. We will focus our attention on the messages that are formatted according to the $GPGGA format.

We need a driver to read in serial data from the puck, parse it for the latitude, longitude and altitude. *We have provided an example device driver for a depth sensor in the appendix section, so that you can use that as a template.*

You need to then convert the latitude and longitude to utm using the python package “utm” as discussed in class.

Define a custom ros message with header, latitude, Longitude, Altitude, utm\_easting, utm\_northing, Zone, letter as fields.

Your ros node should then publish all these values in your custom defined ros msg.

1. **Go outside and collect data**  
   **2.1.** **Stationary data outdoors**

Go outside and collect 10 minutes of data at one spot in a rosbag.

Read this data into matlab.

Examine the utm data (by plotting it or doing statistics on it)

What does this say about GPS navigation? What is a good error estimate?

What can you say about the distribution of noise in the signal?

**2.2 Walk in a straight line outdoors**

In a new rosbag recording, walk in a straight line for a few hundred meters.

Examine the utm data (by plotting it or doing statistics on it)

What does this say about GPS navigation when moving? How does the error estimate change as you move as opposed to stay in a spot? What can you say about the distribution of noise in this case?

1. **Analyse the data as asked above and write your observations in a brief report and submit a pdf copy of it on Blackboard.**
2. **Push your device driver and analysis code to gitlab.**

**Grading Rubric (10 Points)**

* 2 points for working device driver
* 3 points for Analysis of Stationary Data (sec 2.1)
* 3 points for Analysis of Moving Data (sec 2.2)
* 2 Points for overall Presentation of Report

Late submission policy is mentioned in the syllabus.

**How to Submit Lab1**

1. In your class repo ‘EECE5554\_RoboticsSensing’, create a directory called LAB1
2. Inside LAB1, create sub-directory structure ‘catkin\_ws/src’.
3. Copy the catkin\_package used for this assignment under src.

Your repo structure should look similar to ‘<Path\_to\_repo>/EECE5554\_RoboticsSensing/LAB1/catkin\_ws/src/<your\_catkin\_pkg\_dir>’

1. Push your local commits to (remote) gitlab server. You can verify this by visiting gitlab.com and making sure you can see the commit there.
2. Upload your pdf report to Blackboard

**Appendix**

1. Python based ros node for Depth sensor on an auv

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

import rospy

import serial

from math import sin, pi

from std\_msgs.msg import Float64

from nav\_msgs.msg import Odometry

def paro\_to\_depth(pressure, latitude):

'''

Given pressure (in m fresh) and latitude (in radians) returns ocean depth (in m.). Uses the formula discovered and presented by Leroy and Parthiot in: Claude C. Leroy and Francois Parthiot, 'Depth-pressure relationships in the oceans and seas', J. Acoustic Society of America, March 1998, p1346-.

'''

# Convert the input m/fw into MPa, as the equation expects MPa.

pressure = pressure \* 0.0098066493

# Gravity at Latitude.

g = 9.780318 \* (1 + 5.2788e-3\*sin(latitude)\*\*2 -

2.36e-5\*sin(latitude)\*\*4)

# Now calculate the 'standard ocean' depth.

Zs\_num = (9.72659e2\*pressure - 2.512e-1\*pressure\*\*2 +

2.279e-4\*pressure\*\*3 - 1.82e-7\*pressure\*\*4)

Zs\_den = g + 1.092e-4\*pressure

return Zs\_num / Zs\_den

if \_\_name\_\_ == '\_\_main\_\_':

SENSOR\_NAME = "paro"

rospy.init\_node('depth\_paro')

serial\_port = rospy.get\_param('~port','/dev/ttyS1')

serial\_baud = rospy.get\_param('~baudrate',9600)

sampling\_rate = rospy.get\_param('~sampling\_rate',5.0)

offset = rospy.get\_param('~atm\_offset',12.121) # in meter ??

latitude\_deg = rospy.get\_param('~latitude',41.526) # deg 41.526 N is Woods Hole

port = serial.Serial(serial\_port, serial\_baud, timeout=3.)

rospy.logdebug("Using depth sensor on port "+serial\_port+" at "+str(serial\_baud))

rospy.logdebug("Using latitude = "+str(latitude\_deg)+" & atmosphere offset = "+str(offset))

rospy.logdebug("Initializing sensor with \*0100P4\\r\\n ...")

sampling\_count = int(round(1/(sampling\_rate\*0.007913)))

port.write('\*0100EW\*0100PR='+str(sampling\_count)+'\r\n') # cmd from 01 to 00 to set sampling period

rospy.sleep(0.2)

line = port.readline()

port.write('\*0100P4\r\n') # cmd from 01 to 00 to sample continuously

latitude = latitude\_deg \* pi / 180.

depth\_pub = rospy.Publisher(SENSOR\_NAME+'/depth', Float64, queue\_size=5)

pressure\_pub = rospy.Publisher(SENSOR\_NAME+'/pressure', Float64, queue\_size=5)

odom\_pub = rospy.Publisher(SENSOR\_NAME+'/odom',Odometry, queue\_size=5)

rospy.logdebug("Initialization complete")

rospy.loginfo("Publishing pressure and depth.")

odom\_msg = Odometry()

odom\_msg.header.frame\_id = "odom"

odom\_msg.child\_frame\_id = SENSOR\_NAME

odom\_msg.header.seq=0

sleep\_time = 1/sampling\_rate - 0.025

try:

while not rospy.is\_shutdown():

line = port.readline()

#print line

if line == '':

rospy.logwarn("DEPTH: No data")

else:

if line.startswith('\*0001'):

odom\_msg.header.stamp=rospy.Time.now()

try: pressure = float(line[5:].strip())

except:

rospy.logwarn("Data exception: "+line)

continue

pressure\_pub.publish(pressure)

depth\_mes = paro\_to\_depth(pressure - offset, latitude\_deg)

depth\_pub.publish(depth\_mes)

odom\_msg.pose.pose.position.z = -depth\_mes

odom\_msg.header.seq+=0

odom\_pub.publish(odom\_msg)

rospy.sleep(sleep\_time)

except rospy.ROSInterruptException:

port.close()

except serial.serialutil.SerialException:

rospy.loginfo("Shutting down paro\_depth node...")