Lab 2: Drone Sensors

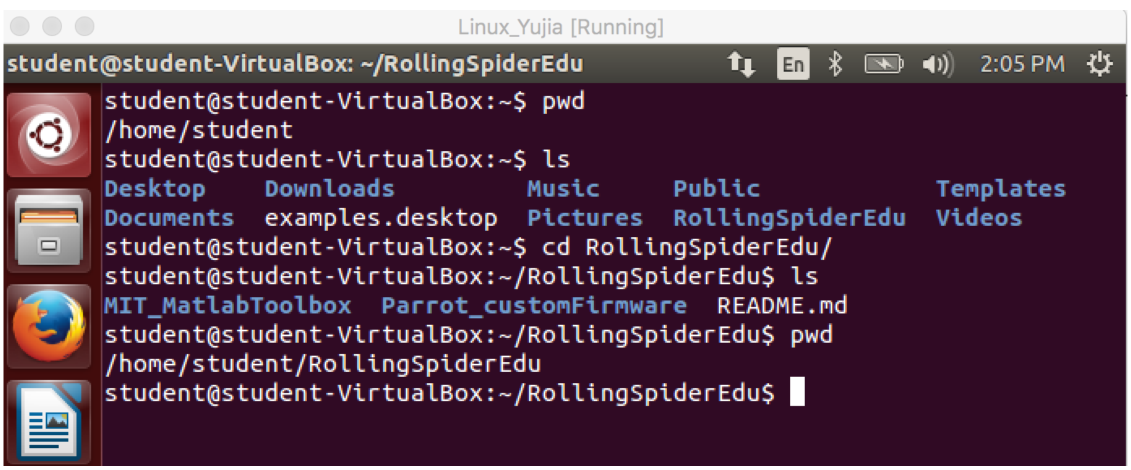
The purpose of this lab is familiarize you with the drone dynamics and sensors. Furthermore, you will characterize the bias and variance of the sensor measurements.

Basic Linux Terminal Commands and Text Editor Installation

In this section we will introduce basic Linux command lines.

- **pwd** print working directory. The “pwd” command allows you to know the directory in which you’re located.
- **ls** The “ls” command shows you the files in your current directory.
- **cd** change directory. The “cd” command allows you to change directories.

  “cd [directory name]” go to a certain directory
  “cd .. ” navigate up one level
  “cd ” navigate back to home



- **mkdir** The mkdir command allows you to create directories.
- open a text file. First install text editor with (here we choose Sublime. If you are familiar with Vim or other text editors, you are welcome to use them)

begin code
```
sudo add-apt-repository ppa:webupd8team/sublime-text-3
sudo apt-get update
sudo apt-get install sublime-text-installer
```
end code

Go to directory

`~/RollingSpiderEDU/MIT_MatlabToolbox/trunk/embcode`

Open rsedu_control.c with command line

begin code

```
subl rsedu_control.c
```

end code

Modify Drone Test Code, part 1

In this lab you will test the drone while holding it in your hands. In order to make this process safe, we need to set the power gain of testing process to 0.

- Go to directory "RollingSpiderEDU/MIT_MatlabToolbox/bin". Open `DroneTest.sh` with text editor. In the `DroneTest.sh` script file change the power gain from 10 to 0. Therefore, when you test the drone no power will be sent to the motors. In this same line you should also delete

  `\[1-9\]`

  so that the regular expression just contains

  `\[0-9\]*`

  This will allow the regular expression to detect the string when the power gain is only one digit (i.e. 0).

```
#!/bin/bash

# 1. change folder to where this script is
cd `dirname $0`

# 2. Telnet into drone, adjust PowerGain in paramsEDU.dat and start the SpiderFlight
/usr/bin/expect <<SCRIPT
set timeout -1;
spawn telnet 192.168.1.1;
expect "RS.edu] \$ ";
send "sed -i \"s/POWERGAIN : \[0-9\]*;/POWERGAIN : 0;/\" /data/edu/params/paramsEDU.c
expect "RS.edu] \$ ";send "killall dragon-prog\r"; expect "RS.edu] \$ ";
send "SpiderFlight.sh\r";
expect "RS.edu] \$ ";
send "exit\r";
expect eof
SCRIPT

```

Solving the Bug "ERROR: Please take off from level surface."

Connect the drone with "DroneConnect.sh" and "DroneKeyboardPilot.sh".

Look for "Sensorcal" result The first three figures denote acceleration on x, y, z axis.

If your drone is not on a level surface, the result is different:

```
rsedu_vis(): WARNING not enough distinct markers (colored balls) foun
d!
Batterylevel: 100.000000
Sensorcal: 0.121023 :: -0.118540 :: -9.532057 :: -0.037846 :: 0.01281
4 :: 0.004110 :: 100822.036556
*** Using a default buffer of size 4600 for logging variable tout
*** Using a default buffer of size 4600 for logging variable time
```

```
Batterylevel: 100.000000
Sensorcal: 2.184233 :: 4.249672 :: -8.204931 :: -0.067022 :: 0.017938
 :: -0.002945 :: 100820.533034
ERROR: Please take off from a level surface!
Saving logged data... DONE
```

Some groups are facing the error "ERROR: Please take off from level surface!" when run "DroneTest.sh", In this case, we see that the acceleration sensor is not accurate, the sensor output on a level surface

```
Got image processing connection, 22 !
rsedu_vis(): SUCCESS opening POSVIS-fifo!
rsedu_vis(): WARNING not enough distinct markers
rsedu_vis(): WARNING not enough distinct markers
rsedu_vis(): WARNING not enough distinct markers
Batterylevel: 100.000000
Sensorcal: 0.183767 :: -0.058746 :: -8.685301 ::
550 :: 101643.638214
ERROR: Please take off from a level surface!
Saving logged data... DONE
Good night!
```

Figure 1: Error from Group 17

should be "$0, :: 0, :: -9, 8$", when the sensor output is off by more than ± 0.7, error will be reported.

The solution here is to revise the code in rsedu_control.c. Open rsedu_control.c with text editor, and search for the error message:

```c
    //Stop if angled take-off
    //if((!FEAT_NOSAFETY) && fabs(9.81 + sensorCal[2]) > 0.7)
    if((!FEAT_NOSAFETY) && fabs(9.81 + sensorCal[2]) > 2.0)
    {
        run_flag = 0;
        printf("ERROR: Please take off from a level surface! \n");
        out->motors_speed[0] = 0;
        out->motors_speed[1] = 0;
        out->motors_speed[2] = 0;
        out->motors_speed[3] = 0;
        out->command = BLDC_CMD_STOP;
        return;
    }
```

Change the error threshold from ± 0.7 to ± 2.0, and save the file.

Run "DroneUploadEmbeddedCode.sh" again, this will compile the files and upload to the drone.

If you run the drone test again, no error will be reported.

Modify Drone Test Code, part 2

You will also need to change the flag `FEAT_NOSAFETY` to 1. This will relax the safety checks in the `rsedu_control.c` code. There are two ways to do this:

- (Recommend) With the drone connected via Bluetooth open a new terminal and run

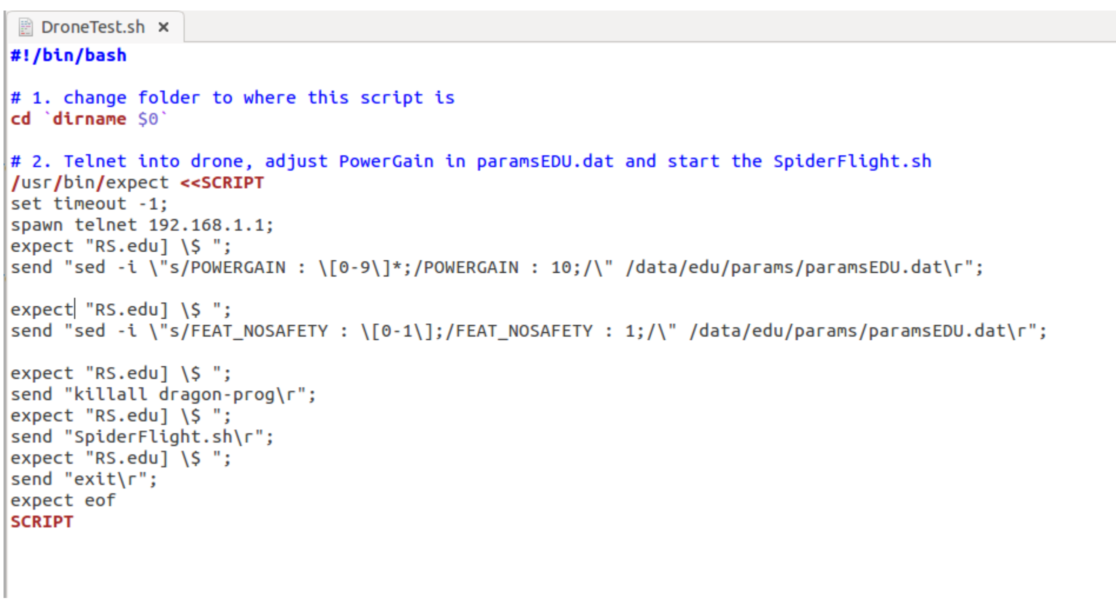
  ```
  $ telnet 192.168.1.1
  ```

  Then, run the command

  ```
  $ vi /data/edu/params/paramsEDU.dat
  ```

  This will open the `paramsEDU.dat` file with the `vi` text editor in the terminal. To replace the 0 by 1 move the cursor to the value to be changed using the keyboard and press `r` followed by the new value. To exit `vi` press type in `:x` followed by `Enter`.

  IMPORTANT!! Since this change disables the safety features make sure that you reverse it before flying the drone.



```
DroneTest.sh ×
#!/bin/bash

# 1. change folder to where this script is
cd `dirname $0`

# 2. Telnet into drone, adjust PowerGain in paramsEDU.dat and start the SpiderFlight.sh
/usr/bin/expect <<SCRIPT
set timeout -1;
spawn telnet 192.168.1.1;
expect "RS.edu] \$ ";
send "sed -i \"s/POWERGAIN : \[0-9\]*;/POWERGAIN : 10;/\" /data/edu/params/paramsEDU.dat\r";

expect "RS.edu] \$ ";
send "sed -i \"s/FEAT_NOSAFETY : \[0-1\];/FEAT_NOSAFETY : 1;/\" /data/edu/params/paramsEDU.dat\r";

expect "RS.edu] \$ ";
send "killall dragon-prog\r";
expect "RS.edu] \$ ";
send "SpiderFlight.sh\r";
expect "RS.edu] \$ ";
send "exit\r";
expect eof
SCRIPT
```

Figure 2: Modified `DroneTest.sh` file.

- Alternatively, you can modify the `DroneTest.sh` file by adding the following lines:

  ```
  send "sed -i \"s/FEAT_NOSAFETY : \[0-1\];

  /FEAT_NOSAFETY : 1;/\" /data/edu/params/paramsEDU.dat\r";

  expect "RS.edu] \$ ";
  ```

as shown in Figure 2. Note that the first two lines above should really be on the same line as in the figure.

Testing the Drone

Now you will perform various experiments with the drone in order to characterize the sensor performance and behavior. For some of these tests you will be holding the drone, so make sure that when running any experiments that you are using the `DroneTest.sh` script with the power gain set to 0.

Create a Word or Latex document with answers to the following questions.

1. **Acceleration Measurement**
   - Use abrupt translations in the three axes to determine the alignment and positive direction of the x, y, and z axes fixed to the body frame. Sketch the drone, highlighting the face (where the two blinking lights are). Plot on it the reference frame the drone uses. Add a VERY short description of your experiment with plots of the x, y, and z accelerations.
   - You will notice a big offset in the vertical acceleration measurement. What does it correspond to and why its sign is negative?

2. **Angular Velocity Measurements**
   - Do simple rotations tests and check that roll, pitch, and yaw measured by the gyros in the IMU are relative to the axes you found in Question 1. Also, check that the signs are correct according to the right hand rule. Add a VERY short description of your experiments and add plots of the angular velocities.
   - Plot the Euler angles. Find and open the `EstimatorOrientation.m` file in the `DroneCompensator/Estimator` Simulink block. Look in the `complimentaryfilter.m` function in this file and notice that the Euler angles are estimated by a simple integrator. Can you find an experiment that shows the estimation diverging (or just wrong)?
   - Which Euler angle set is the toolbox using? $X-Y-Z$? $Z-Y-X$? $X-Y-X$? Try to determine this with an experiment and confirm your results by looking at the `EstimatorOrientation.m` and `complimentaryfilter.m` files to see which matrix is used to transform between the body frame and the inertial frame. Add a VERY short description of your experiments and add plots of the Euler angles.

3. **Altitude Measurements from the sonar and pressure sensors**
   - Do simple altitude tests by moving the drone up and down both slowly and quickly. Add a VERY short description of your experiment and plots of the sonar and pressure sensor measurements and estimated altitude.
   - With the drone at a fixed altitude add and remove objects in the sonar range of the drone. Comment on your findings with short comments and plots. Specifically, try to estimate how close an object must be to effect the sonar reading of the drone.

4. **Sensor Bias and Variance** For this problem you will change the power gain of the drone to 10% so that the effect of the vibrations on the sensors can be studied. Attach the wheels to the drone and run `DroneTest.sh` with the drone sitting on a flat surface. Calculate the bias (mean of the signal) and the variance for the 3 acceleration and angular velocity measurements. Do the same for the sonar altitude and pressure measurements.

5. **Lateral and longitudinal velocity from the camera optical flow measurements**
   - Move the drone to record translational movement. Do the lateral and longitudinal velocity measurements v_x and v_y seem reasonable and are they aligned with the x and y axes measured before?
   - Repeat this test on different surfaces and at different altitudes and see how the estimation changes. Describe the effect of these changes and compute the bias and variance for the different tests. Report your findings with short comments and plots of the velocities.

6. **Turntable Test** For this part of the lab we will supply you with data from a turntable test as can be seen in this video:
   `https://drive.google.com/file/d/0B6lJEh13nYmnNnJ4SlVWSXZjYkE/view?usp=sharing`

   For this test the center of mass of the drone was 3.5 inches from the center of the turntable with the face of the drone (i.e. where the lights are) pointed to the center of the turntable as in Figure 3.

Data from this test can be downloaded from the following link:
`https://drive.google.com/file/d/0B6lJEh13nYmnNmVqRUdlSTRFclU/view?usp=sharing`

From the angular velocity data about the z-axis and the orientation of the drone calculate the expected resulting acceleration in the x and y directions. Do the calculated acceleration values agree with the measured values? Note that the sensor data has significant noise, especially the accelerometers, so you may find it beneficial to filter or average the data to get cleaner estimates of the measurements. Comment on your findings and show plots comparing the calculated and measured values.

Drone Dynamics

The dynamics of quadrotor drones have been extensively studied. Review the following references describing the quadrotor dynamics.

- "Robotics, Vision and Control" by Peter Corke can be downloaded for free from:
  `http://link.springer.com/book/10.1007%2F978-3-642-20144-8`
- "Quadrotor Modeling and Control" by Nathan Michael from Carnegie Mellon University which can be downloaded from bCourses
- "Multirotor Aerial Vehicles" by Robert Mahoney, Vijay Kumar, and Peter Corke can be downloaded from bCourses.

Figure 3: Orientation of drone on turntable

Compare the models and make sure they make sense. Note that each of these references define the axes and Euler angles of the drone differently. Part of the goal of this lab is to determine the axes that the accelerometers and gyros in the drones inertial measurement unit (IMU) measures about.