ME136 Lab 1

Group 6

Trey Fortmuller
Joey Kroeger
David Dominguez Hooper

1. Running the motors:

(a) We adjusted lines 41-54 in UserCode.cpp to set the motor command for all motors to 50 if the blue button on the Xbox controller is pressed, and 0 otherwise.

UserCode.cpp

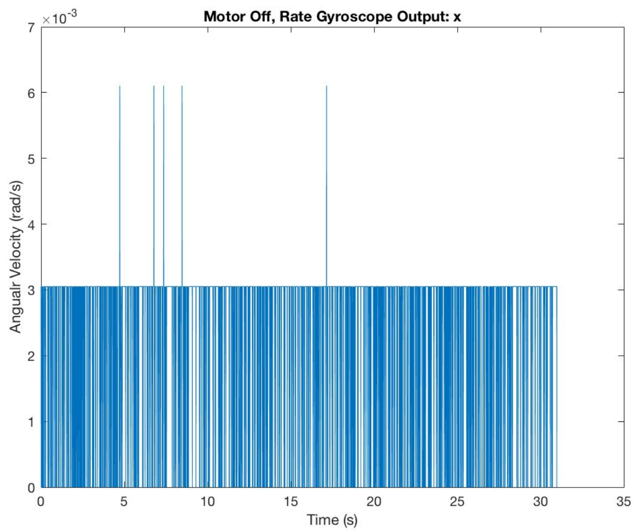
```
1 #include "UserCode.hpp"
2 #include "UtilityFunctions.hpp"
3 #include "Vec3f.hpp"
4
5 #include <stdio.h> //for printf
6
7 //An example of a variable that persists beyond the function call.
8 float exampleVariable_float = 0.0f;  //Note the trailing 'f' in the number. This is to force single precision floating point.
9 Vec3f exampleVariable_Vec3f = Vec3f(0, 0, 0);
10 int exampleVariable_int = 0;
11
12 //We keep the last inputs and outputs around for debugging:
13 MainLoopInput lastMainLoopInputs;
14 MainLoopOutput lastMainLoopOutputs;
15
16 //Some constants that we may use:
17 const float mass = 30e-3f;  // mass of the quadcopter [kg]
18 const float gravity = 9.81f;  // acceleration of gravity [m/s^2]
19 const float inertia_xx = 16e-6f;  //MMOI about x axis [kg.m^2]
20 const float inertia_yy = inertia_xx;  //MMOI about y axis [kg.m^2]
21 const float inertia_zz = 29e-6f;  //MMOI about z axis [kg.m^2]
22
23 const float dt = 1.0f / 500.0f; //[s] period between successive calls to MainLoop
24
25 MainLoopOutput MainLoop(MainLoopInput const &in) {
26   //Your code goes here!
27   // The function input (named "in") is a struct of type
28   // "MainLoopInput". You can understand what values it
29   // contains by going to its definition (click on "MainLoopInput",
30   // and then hit <F3> -- this should take you to the definition).
31   // For example, "in.joystickInput.buttonBlue" is true if the
32   // joystick's blue button is pushed, false otherwise.
33
34   //Define the output numbers (in the struct outVals):
35   MainLoopOutput outVals;
36 //  motorCommand1 -> located at body +x +y
37 //  motorCommand2 -> located at body +x -y
38 //  motorCommand3 -> located at body -x -y
39 //  motorCommand4 -> located at body -x +y
40
41   //  If the blue button is pressed, we set all motor outputs to 50
42   if (in.joystickInput.buttonBlue) {
43     outVals.motorCommand1 = 50;
44     outVals.motorCommand2 = 50;
45     outVals.motorCommand3 = 50;
46     outVals.motorCommand4 = 50;
```

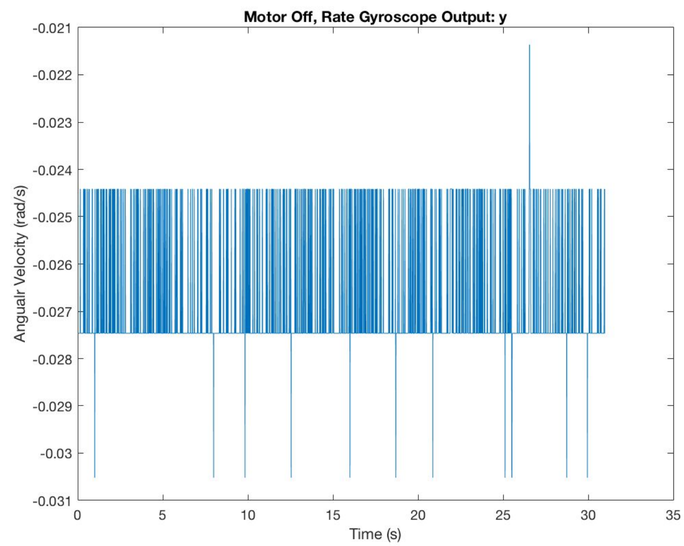
```
  }
  //  Otherwise, we want to set the all motor outputs to 0
  else {
    outVals.motorCommand1 = 0;
    outVals.motorCommand2 = 0;
    outVals.motorCommand3 = 0;
    outVals.motorCommand4 = 0;
  }



  //copy the inputs and outputs:
  lastMainLoopInputs = in;
  lastMainLoopOutputs = outVals;
  return outVals;
}

void PrintStatus() {
  //For a quick reference on the printf function, see: http://www.cplusplus.com/reference/cstdio/printf/
  // Note that \n is a "new line" character.
  // Also, note that to print a `float` variable, you have to explicitly cast it to
  //  `double` in the printf function, and explicitly specify precision using something
  //  like %6.3f (six significant digits, three after the period). Example:
  //   printf("  exampleVariable_float = %6.3f\n", double(exampleVariable_float));

  //Accelerometer measurement
  printf("Acc: ");
  printf("x=%6.3f, ",
         double(lastMainLoopInputs.imuMeasurement.accelerometer.x));
  printf("\n");  //new line
  printf("Gyro: ");
  printf("x=%6.3f, ", double(lastMainLoopInputs.imuMeasurement.rateGyro.x));
  printf("\n");  //new line

  printf("Example variable values:\n");
  printf("  exampleVariable_int = %d\n", exampleVariable_int);
  //Note that it is somewhat annoying to print float variables.
  //  We need to cast the variable as double, and we need to specify
  //  the number of digits we want (if you used simply "%f", it would
  //  truncate to an integer.
  //  Here, we print 6 digits, with three digits after the period.
  printf("  exampleVariable_float = %6.3f\n", double(exampleVariable_float));

  //We print the Vec3f by printing it's three components independently:
```

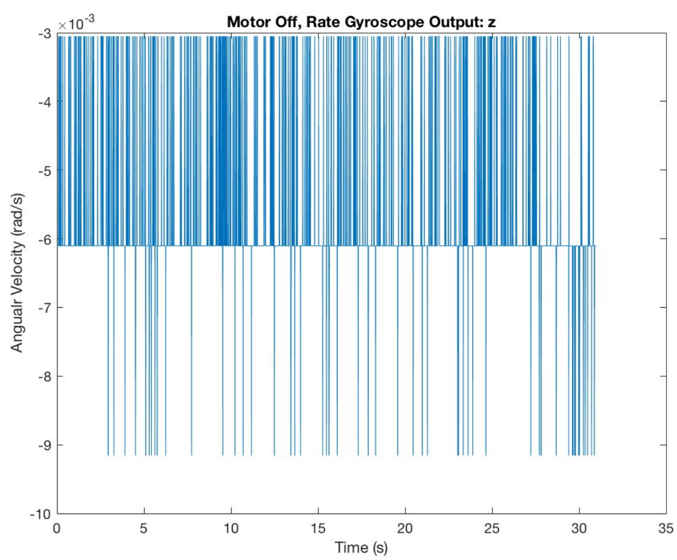
```
  printf("  exampleVariable_Vec3f = (%6.3f, %6.3f, %6.3f)\n",
         double(exampleVariable_Vec3f.x), double(exampleVariable_Vec3f.y),
         double(exampleVariable_Vec3f.z));

  //just an example of how we would inspect the last main loop inputs and outputs:
  printf("Last main loop inputs:\n");
  printf("  batt voltage = %6.3f\n",
         double(lastMainLoopInputs.batteryVoltage.value));
  printf("  JS buttons: ");
  if (lastMainLoopInputs.joystickInput.buttonRed)
    printf("buttonRed ");
  if (lastMainLoopInputs.joystickInput.buttonGreen)
    printf("buttonGreen ");
  if (lastMainLoopInputs.joystickInput.buttonBlue)
    printf("buttonBlue ");
  if (lastMainLoopInputs.joystickInput.buttonYellow)
    printf("buttonYellow ");
  if (lastMainLoopInputs.joystickInput.buttonStart)
    printf("buttonStart ");
  if (lastMainLoopInputs.joystickInput.buttonSelect)
    printf("buttonSelect ");
  printf("\n");
  printf("Last main loop outputs:\n");
  printf("  motor command 1 = %6.3f\n",
         double(lastMainLoopOutputs.motorCommand1));
}
```

2. Sensor Analysis:

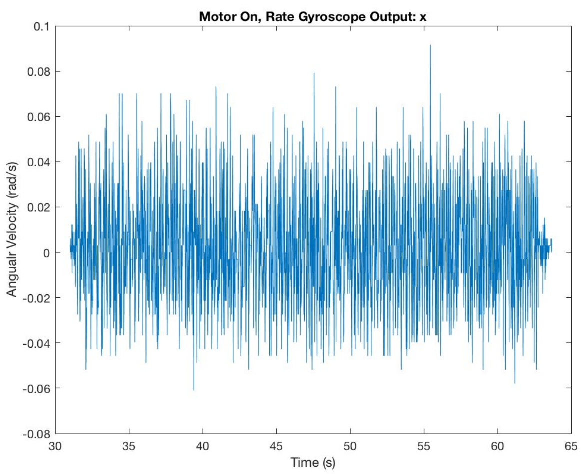
(a) X, Y, and Z rate gyro outputs with the motors turned off. We see very low angular velocities on all three axis (which is to be expected considering the quadcopter is sitting still on the table). In the graphs, as zoomed in as they are, we can see the resolution of the gyro is on the order of 10^{-3} rad/s.

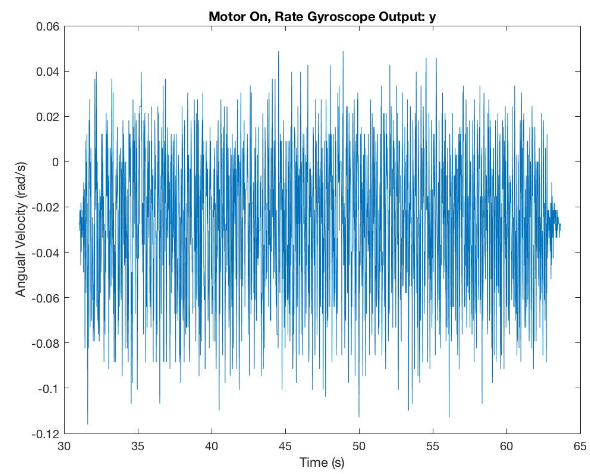


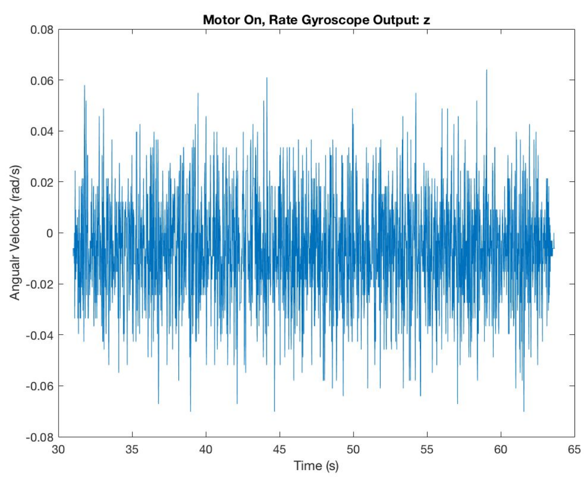




X, Y, and Z rate gyro outputs with the motors turned on. Significant motor vibrations are transmitted through the frame to the gyro onboard the flight controller. The vibrations on three axis are of similar magnitude and center around 0. This test confirmed that motor noise will likely have to be accounted for through software filtering but we are interested in how these noise profiles may change throughout the throttle range of the quadcopter, and whether these profiles would be significantly different if the quadcopter was hovering in air rather than resting on the table. Intuitively, we would imagine the table acts as a physical dampener to the oscillations caused by the motors.







(b) Mean and standard deviation of the rate gyros:

- OFF:

  Mean:

  X = 0.0024

  Y = -0.0269

  Z = -0.0058

  SD:

  X = 0.0013

  Y = 0.0012

  Z = 0.0011

- ON:

  Mean:

  X = 0.0021

  Y = -0.0269

  Z = -0.0064

  SD:

  X = 0.0225

  Y = 0.0292

  Z = 0.0196
- Comparison: (Comment on the effect of running the motors)

If we look at the mean values for when the motors are on vs off, there is little to no difference in the calculated value. This is because the values are centered very close to zero and any vibrations that the motor cause shouldn't change this value because it will oscillate through the mean value. However, if we compare the Standard Deviation Values we find the the standard deviation is an order of magnitude larger when the motors are on. This means that the displacement from the mean value is greater when the motors are on. Therefore, we can conclude that the motors are introducing a substantial amount of vibration to the system.

Below is the code we used to calculate the mean and standard deviation values

```
%takes struct_out as a data structure for values over the course of 30
%seconds when the motors were off and struct_out1 for when the motors were
%on

%calculate mean values when motors are off
x_off_mean=mean(struct_out.rateGyro.x);
y_off_mean=mean(struct_out.rateGyro.y);
z_off_mean=mean(struct_out.rateGyro.z);
```

```
%calculate mean values when motors are on
x_on_mean=mean(struct_out1.rateGyro.x);
y_on_mean=mean(struct_out1.rateGyro.y);
z_on_mean=mean(struct_out1.rateGyro.z);

%calculate standard deviation when motors are off
x_off_std=std(struct_out.rateGyro.x);
y_off_std=std(struct_out.rateGyro.y);
z_off_std=std(struct_out.rateGyro.z);

%calculate standard deviation when motor are on
x_off_std=std(struct_out1.rateGyro.x);
y_off_std=std(struct_out1.rateGyro.y);
z_off_std=std(struct_out1.rateGyro.z);
```

Trey Fortmuller
Joey Kroeger
David D. Hooper
Due 9/20/17

ME136 PS1

1) $F_L = F_w$ at the stall speed

$$F_L = \frac{1}{2}\rho V_\infty^2 A C_L = F_w$$

$$\sqrt{\frac{2F_w}{\rho A C_L}} = V_\infty = \sqrt{\frac{2(9.81\cdot 900)}{(1.225)(16)(1.6)}} = \boxed{23.73 \text{ m/s}}$$

2) a) propellor speeds

$$f_T = \frac{1}{4}f_w = \frac{1}{4}mg = C_T\Omega^2$$

$$\Omega = \sqrt{\frac{mg}{4C_T}}$$

$$= \sqrt{\frac{(.5)(9.8)}{4(6.41e\text{-}6)}} = \boxed{437.38 \text{ rad/s}}$$

$$= 4176 \text{ rpm}$$

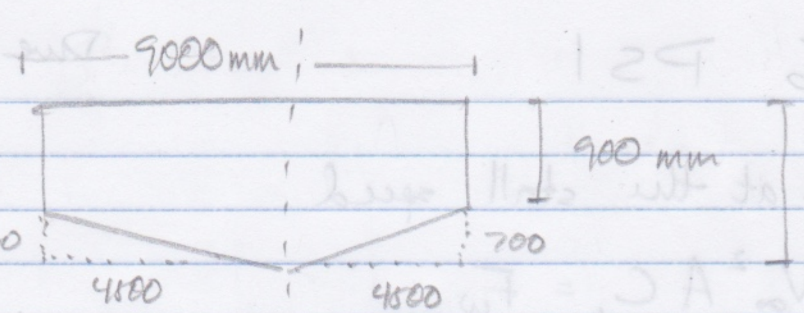
b) mechanical power at hover

$$P = \Omega\tau = \Omega\gamma f_T = \frac{\Omega\gamma mg}{4}$$

$$= \frac{(437.38)(.017)(.5)(9.81)}{4}$$

$$= \boxed{9.11 \text{ W}}$$

3)



$C_L = 2\pi\alpha \qquad m = 1000\ kg$

a) AR ?

$$S = [9000 \times 1600] - [4500 \times 700]$$

$$= 11.25 \times 10^6\ mm^2$$

$$AR = \frac{b^2}{S} = \frac{(9000)^2}{11.25 \times 10^6} = \boxed{7.2}$$

b) $$C_L = \left(\frac{AR}{AR+2}\right) C_\ell \alpha = 2\pi\left(\frac{AR}{AR+2}\right)\alpha^2$$

$$C_L = 2\pi\left(\frac{7.2}{9.2}\right)\alpha^2$$

$$C_L = \boxed{4.917\,\alpha^2}$$

c) $$f_L = f_w = \frac{1}{2}\rho V_\infty^2 A C_L = mg \quad \text{at } V_\infty = 55\ m/s$$

$$= \frac{1}{2}\rho V_\infty^2 A (4.917)\alpha^2 = mg$$

$$\alpha = \sqrt{\frac{2mg}{\rho V_\infty^2 A \cdot 4.917}} = \boxed{.309\ rad} = 17^\circ$$

4) $R_e = \frac{V_\infty l \rho}{\mu}$

$f_D = f_w$ at terminal velocity

$$f_D = \frac{1}{2}\rho V_\infty^2 A C_D(R_e) = mg$$

$$V_\infty = \frac{R_e \mu}{l \rho}$$

$$f_D = \frac{1}{2}\rho\left(\frac{R_e \mu}{l\rho}\right)^2 A\, C_D \qquad A = \frac{\pi l^2}{4}$$

$$f_D = \frac{1}{8}\cancel{\frac{1}{2}}\cancel{\rho}\left(\frac{R_e^2 \mu^2}{\cancel{l^2}\rho^{\cancel{2}}}\right)\left(\frac{\pi \cancel{l^2}}{\cancel{4}}\right) C_D$$

$$f_D = \frac{R_e^2 \mu^2 \pi C_D}{8\rho}$$

$$R_e^2 = \frac{f_D \rho}{\mu^2 C_D}\frac{8}{\pi} \qquad f_D = f_w \text{ at terminal velocity} = mg$$

$$R_e^2 = \frac{8}{\pi}\frac{mg\rho}{\mu^2}\frac{1}{C_D}$$

we plug in values for C_D and check that a corresponding R_e exists on the given curve.

$C_D = .4 \rightarrow R_e = 1.3 \times 10^6$ which is not on the curve
$C_D = .15 \rightarrow R_e = 2.1 \times 10^6$ which is on the curve, we use this value to calculate V_∞.

$$V_\infty = \sqrt{\frac{8mg}{\pi l^2 C_D}} = \boxed{157.34\ m/s} = 350\ mph$$