Experiment C8
BiCopter (2D Quadcopter)
Procedure

Deliverables: Checked lab notebook, In-Lab Demonstrations

Overview

Quadcopter drones have become a common toy for hobbyists and children, and an important tool with many industrial and military applications. Unlike conventional aircraft that have inherent mechanical stability, quadcopters rely on constant feedback from an inertial measurement unit (IMU), specifically a gyroscope, to adjust the thrust of the four motors propellers and keep the craft level. Inexpensive MEMS IMUs, lightweight batteries, and efficient brushless motors make this possible.

In this lab, you will create a 2-dimensional version of a quadcopter—a “BiCopter”. Mounted to a bearing located at its center of gravity, the BiCopter will only be free to rotate to some pitch angle θ, as shown in Fig. 1 below. You will develop a feedback controller that will take the measured value of $\boldsymbol{\theta}$ (measured by an IMU) and use it to adjust the two thrusts F_1 and F_2, thus keeping the BiCopter level at some desired angle $\boldsymbol{\theta_s}$.

Neglecting the mass of the airframe, we may treat the BiCopter as two point masses, each a distance R from the pivot in the center, which yield the rotational equation of motion

$$2mR^2\ddot{\theta} = RF_1 - RF_2, \tag{1}$$

where m is the mass of each motor. Both of the motor thrusts will be adjust using proportional-derivative feedback, such that $F_1 = F_0 - k_p(\theta - \theta_s) - k_d\dot{\theta}$ and $F_2 = F_0 + k_p(\theta - \theta_s) + k_d\dot{\theta}$, where $2F_0$ is the quiescent thrust needed to lift the weight of the BiCopter.

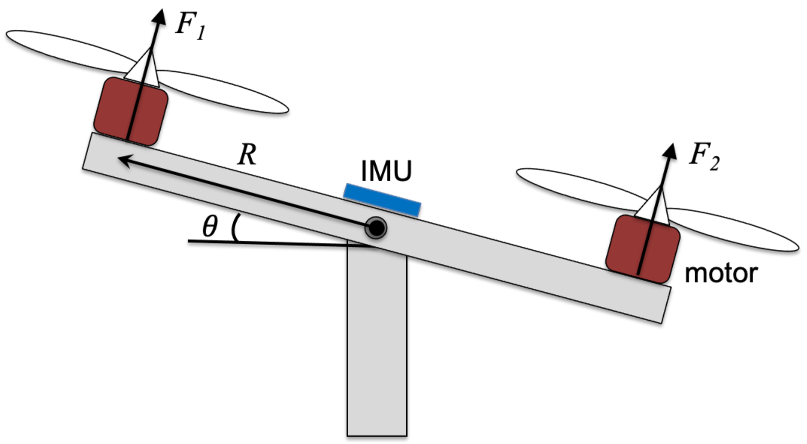

Figure 1 – A 2-dimensional quadcopter or “BiCopter” is constrained to rotate about a bearing located at its center of gravity. Feedback from an IMU is used to stabilize the system at a desired pitch angle.

Subsystem A: Inertial Measurement Units (IMU) – MEMS Gyroscope

You will begin by mounting, wiring, and testing the BNO-055 9-axis IMU. It will be used to measure the pitch angle θ and pitch rate (angular speed) q.

Procedure

1. Use the 6 mm shaft to mount the BiCopter to the L-brackets on the slotted rail. Secure the ends of the shaft using shaft collars.
2. Plug the BNO-055 IMU into the small breadboard in the center of the BiCopter.
3. Dedicate two separate unused rows as +5V and GND bus lines. Use short jumper wires to connect the corresponding pins on the IMU to these bus lines.
4. Use command strips to mount the Arduino UNO to the slotted rail (not to the BiCopter). Connect the SDA and SCL I^2C pins on the IMU to the corresponding pins on the Arduino. Use jumpers wires that are long enough to allow the BiCopter to rotate.
5. Download the BNO-055 9-axis IMU code from the C8 website, and save it to your C8 folder.
6. Connect the Arduino to the lab computer, select the appropriate COM port, and load the IMU code to the Arduino.
7. Open the "Serial Monitor" to see the output displayed as text. Make sure the Baud rate is set to 115200.
8. Slowly rotate the BiCopter to its limits. You should see the angle and angular speed change for one of the three axes of rotation. Which axis is the BiCopter rotating about?
9. In the main loop of the sketch, delete the IMU measurements and subsequent "serial.print()" functions for the two unused axes of rotation. Also, delete any "delay()" functions in the main loop.
10. Open the "Serial Plotter" to see the output displayed as a graph. Slowly rotate the BiCopter back and forth between its limits. You should see the line on the graph change.
11. Modify the main loop of the code to print the time in ms, angle in degrees, and angular speed in deg/s. Save a copy of the code to your code library, and give it an intelligent file name.
    a. It should print in the format "time, angle, angular speed". This will make it easier to import your data into Matlab.
    b. Use the millis() function to get the time and store it as a float point variable. Convert the units from milliseconds to seconds.
12. Test the sub-system to make sure it works. **Demonstrate it to the lab instructor or TA to receive credit on your score sheet.**

Subsystem B: Brushless DC Motors and Electronics Speed Controllers

You will now interface the Electronic Speed Controllers (ESC) for both of the Brushless DC (BLDC) motors with the Arduino, so that the thrust F_1 and F_2 can be controlled.

Safety First – You must wear safety glasses for the remainder of this lab.

Procedure

1. Make sure the Arduino is turned off and disconnected from any power source including the USB cable.
2. Connect the three banana plugs on each ESC to the three wires on each motor.
3. Plug two 12V DC power supplies into a surge protector power strip. You will use the switch on the power strip as your main kill switch. Make sure it is switched OFF.
4. Each ESC will have its own separate 12V DC power supply. Connect the red and black wires on each of the ESCs to the + and - screw terminal adapters for the 12V DC power supply.
5. Use jumper wires to make the following connections to the ESCs:
   a. Both ESC brown wires → GND bus line on small breadboard
   b. Both ESC red wires → +5V bus line on small breadboard
   c. First ESC yellow wire → Pin 9 on Arduino
   d. Second ESC yellow wire → Pin 8 on Arduino
6. Download the Brushless Motor ESC code from the C8 website, and save it to your C8 folder.
7. Connect the Arduino to the lab computer, select the appropriate COM port, and load the ESC code to the Arduino.
8. After the code uploads successfully, press the reset button on the Arduino and quickly turn on the power strip. Things might get crazy, so **keep your finger on the kill switch.**
9. You should a series of beeps emanating from the motors, then they will begin spinning.
10. Carefully verify that the air is blowing in the correct direction (downward). If a motor is spinning backwards, turn OFF the power strip, and switch and two of the three banana plug motor-to-ESC connections. This should change the direction of rotation.
11. The ESC reads a 50 Hz digital pulse train, where the pulse width corresponds to the motor speed.
    a. 1000 μs corresponds to minimum speed (minimum thrust)
    b. 2000 μs corresponds to maximum speed (maximum thrust)
12. Try changing the pulse width in the main loop from 1100 to 1400 μs. You should see the motors spin faster after you re-upload the code.
13. Test the sub-system to make sure it works. **Demonstrate it to the lab instructor or TA to receive credit on your score sheet.**

Subsystem C: Cable Management

After you have verified that the IMU and Motors work correctly, use zip ties to fasted the ESCs and their wires to the airframe. The red and black 12V lines should hang down from the center, near the bearings.

Show your tidy cable management to the TA to receive credit on the score sheet.

Design Challenge 1 – Proportional Feedback

Safety First: You must power OFF the system before you adjust anything. Do not put your hands near the motors when the system is powered up.

You will now use the measured pitch angle θ to compute the power or thrust of each motor with just proportional feedback $\boldsymbol{F_1 = F_0 - k_p(\theta - \theta_s)}$ and $\boldsymbol{F_2 = F_0 + k_p(\theta - \theta_s)}$. There is not much natural damping in this system, so it will oscillate with only proportional feedback.

- Use a set-point $\theta_s = 0$ and an ESC pulse width of 1100 for the quiescent thrust F_0.
- Combine your codes from the previous sub-systems into a code that will measure the angle θ and compute the ESC pulse width value using proportional feedback.
- Estimate a value for the proportional gain k_p to be the maximum ESC pulse range value of 1000 divided by the maximum angular displacement $\Delta\theta_{\max}$.
- It may be tricky getting the sign convention correct, as the measured angle and angular speed depend on the orientation of the IMU sensor, and the definition of F_1 and F_2 depends on which motor ESC was plugged into which Arduino pin. Think it through. Trial-and-Error it. Make it work.
- Try different values of k_p to see how they affect the dynamic response of the BiCopter.
- Poke it with a stick. Observe the dynamic response.
- **IMPORTANT:** When you are satisfied with the damper's performance, save the angle vs. time data from the Arduino serial monitor or PuTTY. **Demonstrate the system to the TA to receive credit on the score sheet.**

Design Challenge 2 – Proportional-derivative Feedback

You will now use the measured angle θ and angular speed $\dot{\theta}$ to compute the thrust of each motor using the proportional-derivative feedback formula $F_1 = F_0 - k_p(\theta - \theta_s) - k_d\dot{\theta}$ and $F_2 = F_0 + k_p(\theta - \theta_s) + k_d\dot{\theta}$.

- Use a set-point $\theta_s = 0$ and an ESC pulse width of 1100 for the quiescent thrust F_0.
- Estimate a value for the derivative gain k_d to be the maximum ESC pulse range value of 1000 divided by the measured maximum angular speed.
- Again, it may be tricky getting the sign convention correct for the feedback gains. Think about it. Trial-and-Error it. Make it work.
- Try different values of k_p and k_d until you find values that quickly dampen the oscillations.
- Poke it with a stick. Observe the dynamic response.
- When you are satisfied with the BiCopter's performance, save the angle vs. time data from the Arduino serial monitor or PuTTY. **Demonstrate the system to the TA to receive credit on the score sheet.**

Design Challenge 3 – Pilot Control

You will now implement a joystick for a pilot to adjust the setpoint θ_s.

- The joystick is essentially just an analog potentiometer. Connect the GND and 5V joystick wires to the appropriate bus lines using long jumper wires. Use the DMM to verify that the analog output voltage changes and you move the joystick.
- Connect the analog output of the joystick to one of the analog inputs on the Arduino. Use the analogRead() function to read the value.
- Use the map() function to map the joystick signal to a setpoint angle θ_s.
- Test the system. You should be able to safely control the pitch of the BiCopter with the joystick.
- **Demonstrate the system to the TA to receive credit on the score sheet.**

Clean-up

To receive full credit, you must return the lab bench to its initial state:

- Unplug the DC power supplies.
- Cut the zip ties, and remove the wiring from the airframe.
- Put anything that belongs in your tool kit back into the tote bin.

Data Analysis and Deliverables

Using LaTeX or MS Word, make the following items and give them concise, intelligent captions. Make sure the axes are clearly labeled with units. Plots with multiple data sets on them should have a legend. **Additionally, write one or two paragraphs describing the plots/tables. Any relevant equations should go in these paragraphs.**

IMPORTANT NOTE: Add a horizontal line denoting the set-point whenever it is applicable.

1. A plot with the following angle vs. time traces (both traces on the same plot):
   a. The oscillation of the BiCopter with only proportional feedback. (Show 4 full periods of oscillation.)
   b. The oscillation of the BiCopter with well-tuned proportional-derivative feedback. (Show 4 full periods of oscillation.)

Appendix A

Equipment

- Standard Mechatronics lab kit in small tote (above lab bench)
- 40-20 Rails
- C-clamp
- 6mm Shaft Rotary Encoder 1024 P/R w/ mounting bracketSKU: RB-Spa-1042
- RoboGaia 3 Axis Encoder Counter Arduino Shield (SKU 00021)
- Cytron MD10C motor driver board
- 5202 Series Yellow Jacket Planetary Gear Motor, 19.2:1 Ratio, 24mm Length 6mm D-Shaft, 312 RPM
- GoBilda 1121 Series Low-Side U-Channel (5 Hole, 144mm Length)
- 1309 Series Sonic Hub (6mm Bore) – SKU 1309-0016-0006
- 1140 Series Aluminum Baseplate (6mm Thickness, 144mm Diameter)
- 1400 Series 1-Side, 2-Post Clamping Mount (12mm Bore)
- 4100 Series Aluminum Tube (10mm ID x 12mm OD, 300mm Length)
- Sky-Watcher S20540 Star Adventurer Counter Weight Kit