BLE Controlled Quadcopter using nRF52 dev board

Saahil Parikh | | Jason Li

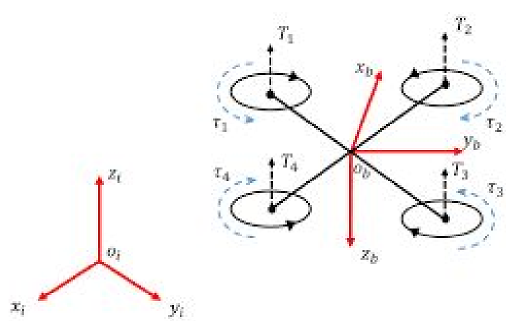
https://github.com/SaahilParikh/Quadcopter

1. Introduction

The main idea behind this project is to create a ble enabled quadcopter. In this paper, I will try to take you from the raw physical analysis, to the implementational techniques used, and will end with remarks on the strengths and weaknesses of this implementation. In the end, the goal is to show how a controlled system like this could be easily implemented. Improving on top of this project by using positional sensors (lidar, GPS) for localization would yield a plethora of amazing products: transportation, robotic waiters, faster emergency response.

2. Overview

So, what is a quadcopter, and how can we model it? To answer that, consider the simple diagram below.



There are four upward forces created by motors, and one downward gravitational force. Using this diagram we can actually start creating some really insightful conclusions! Firstly, movement along the roll and pitch axis are controlled by very simple torque equations:

$$\tau roll = \tau right - \tau left$$
$$\tau pitch = \tau front - \tau back$$

Furthermore, angular displacement around pitch is a function of the moments of the propeller motor system.

$$Mi = moment\ from\ i$$
$$Myaw = (M1 + M3 - M2 - M4)$$

Since moments are just a type of torque, yaw can be written equivalently expressed as:

$$\tau yaw = (\tau 1 + \tau 3 - \tau 2 - \tau 4)$$

Note that the torques for yaw are physically different from the torques along pitch and roll. The torque along yaw is a byproduct of the drag forces acting on the propeller, while the forces along pitch and roll are a function of the lift from the propellers.

Now we have a set of equations defining attitude torques on the quadcopter. However, unless you have an actuator that takes systemic desired torque as an input, we can't really do anything meaningful yet. We have to manipulate these equations to get variables that we can work with. This derivation will be tedious but necessary.

2.1. Physical Derivation

Our end goal for this derivation will be to end up with equations that tell us how fast to run our motors. Let's start again by looking at roll and pitch. Notice how using $\tau = f * r$ we can describe the torque around these axises in terms of thrust (T) and the distance from the motor to the center of rotation (L).

$$\tau roll = (T1 + T4) * L - (T2 + T3) * L$$
$$\tau pitch = (T1 + T2) * L - (T3 + T4) * L$$

We now need to find an equation for the thrust outputted by a motor in terms of something that we can manipulate. To do this, let's try to understand what thrust really is and what it does. Thrust is a force. From high school physics, we learned that if a force moves an object a distance, then it does work on that object. In that context, the propellers on the quadcopter displace air particles with a thrust T over a distance d. Taking the derivative, we now know the power expended by a motor in terms T, d and time.

$$P = T * \frac{d}{\Delta t}$$

Using momentum theory, and the prerequisite of uniform inflow (Schwartzberg, 1975), we find that:

$$P = \frac{T^{3/2}}{\sqrt{2\rho A}}$$

Where the denominator signifies the fluid pushed away with a propeller of radius A.

Now let's try to relate that to the motor. A motor is an element in an active circuit, and therefore has a current passing through it. Additionally, a motor has a resistance and therefore a voltage drop across it. That's actually pretty amazing! Using P=IV we now see a path to relate thrust with properties of the motor.

Okay, so what's the voltage across a rotor? Well that's just the I*R where R is the resistance of the motor windings. However, since a motor is a reactive element, it will also produce a back EMF. The voltage across a motor is the voltage across the windings due to resistance coupled with the back EMF. The back EMF is a product of some constant Kv and the angular velocity of the motor's rotor.

$$V = I * Rmotor + \mathrm{Kv} * \mathrm{w}$$

While a larger voltage across a motor usually increases the angular velocity of the motor, a higher current increases the torque of the motor proportional to some constant.

$$\tau = Kt * (I - Io)$$

Io is the no load current.

Manipulating that equation to get current as the dependent variable we get

$$I = \frac{\tau - Kt*Io}{Kt}$$

Now we can solve for power in terms of the physical variables contributing to I and V. However, before we do that, let's eliminate some of the negligible terms. (Gamazo-Real, 2010) Firstly, the resistance of a good motor's windings is usually very small. Therefore, we will consider it as zero.

$$\mathrm{P} = \frac{(\tau - Kt*Io)*Kv*w}{Kt}$$

To simplify this model even more, we are going to eliminate the term Kt-Io because it is assumed that it is on a much smaller order of magnitude than τ. (Luukkonen, 2011)

$$\mathrm{P} = \frac{Kv}{Kt} w\tau$$

Now we can relate our equation for power given by motor properties and our equation for power given by momentum theory.

$$\frac{Kv}{Kt} w\tau = \frac{T^{3/2}}{\sqrt{2\rho A}}$$

$$\frac{Kv}{Kt} w\sqrt{2\rho A} = \frac{T^{3/2}}{\tau}$$

Notice that thrust is parallel to the motor's axis of rotation. Additionally, recall from high school physics that $\tau = F x\, r$. That means that there is some r determined by the propeller -- blade angle, area, and such. Therefore there is a constant which is often denoted as Kτ relating torque to thrust.

$$T = K\tau * \tau$$

Finally, we have a way to express the upward thrust given by the propeller in terms of things that we can actually measure!

$$\sqrt{T} = \frac{K\tau * Kv}{Kt} w\sqrt{2\rho A}$$

$$T = \left(\frac{(K\tau * Kv)}{Kt} * \sqrt{2\rho A}\right)^2 * w^2$$

Everything but the angular velocity is a constant.

$$T = k * w^2$$

The upward thrust output by a motor is equal to a proportionality constant, whose value is given by all the derivations above, multiplied by the square of the angular velocity.

Now we can answer what we set out to report: pitch and roll in terms of things that we can manipulate.

$$\tau roll = Lk((w1^2 + w4^2) - (w2^2 + w3^2))$$

$$\tau pitch = Lk((w1^2 + w2^2) - (w3^2 + w4^2))$$

Okay, so we still have to derive yaw. Recall that yaw is the torque due to a drag force. We will only be calculating drag on the very tips of the propeller but the approximation is close enough (Luukkonen, 2011). We will use the equation for drag force (Fd) given by fluid dynamics (Serway, 2004) and multiply it by the radius of the propeller to get the torque from each motor over yaw.

$$\tau = Fd(wR) * R$$

$$\tau = \tfrac{1}{2} CA\rho(wR)^2 * R$$

Consolidating the constants.

$$\tau = b * w^2$$

$$\tau pitch = b((w1^2 + w3^2) - (w2^2 + w4^2))$$

2.2. Controller Math

From highschool physics we know that $I = \frac{L}{w}$. We use this to find the kinematics for movement (Bouabdallah, 2004). I am going to omit this, and just show how once we find the kinematics we can use it to create a controller.

$$\tau = I * \left(kd * \frac{d\alpha}{dt} + kp * \alpha + ki \int_0^T \alpha dt\right)$$

Building a system of linear equations to model this movement relates the 3 equations for attitude derived earlier and a new equation for z axis stabilization.

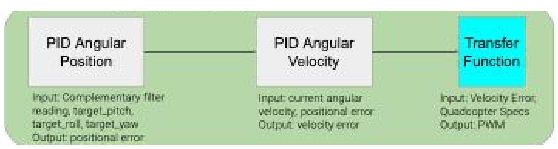
$$k \sum_{i=1}^{4} wi^2 = M_{copter} * g * sec(pitch)sec(roll)$$

In the end, the resulting linear equation -- the reason we had to go through all that derivation -- has the following output.

$$\delta_1 = \frac{1}{4}\left(\frac{I_{yaw}\varepsilon_{yaw}}{b} + \frac{I_{roll}\varepsilon_{roll}}{KL} + \frac{I_{pitch}\varepsilon_{pitch}}{KL} + \frac{mg}{cos(pitch)*cos(roll)}\right)$$
$$\delta_2 = \frac{1}{4}\left(-\frac{I_{yaw}\varepsilon_{yaw}}{b} + \frac{I_{roll}\varepsilon_{roll}}{KL} + \frac{I_{pitch}\varepsilon_{pitch}}{KL} - \frac{mg}{cos(pitch)*cos(roll)}\right)$$
$$\delta_3 = \frac{1}{4}\left(\frac{I_{yaw}\varepsilon_{yaw}}{b} - \frac{I_{roll}\varepsilon_{roll}}{KL} + \frac{I_{pitch}\varepsilon_{pitch}}{KL} - \frac{mg}{cos(pitch)*cos(roll)}\right)$$
$$\delta_4 = \frac{1}{4}\left(-\frac{I_{yaw}\varepsilon_{yaw}}{b} - \frac{I_{roll}\varepsilon_{roll}}{KL} + \frac{I_{pitch}\varepsilon_{pitch}}{KL} + \frac{mg}{cos(pitch)*cos(roll)}\right)$$

where δ_i is the square of the angular velocity of motor i and error ε

The angular velocity controller on this plant is connected to an angular displacement controller for a total of 6 PID controllers. I will note that the velocity controllers have a pretty high Kp to push for faster alignment. The integration constants are non-existent along the roll and yaw because those are axially symmetric. Though, much is omitted for brevity.

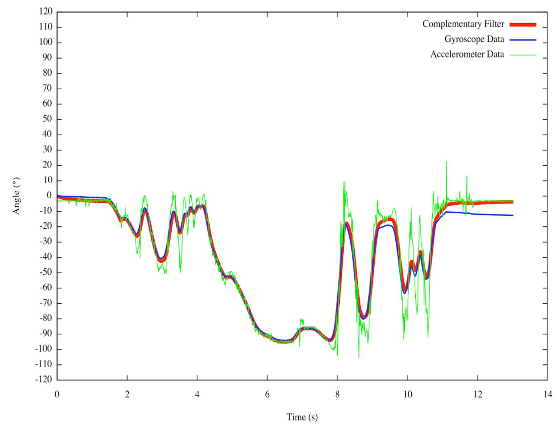


3. Software

3.1. Flight Controller

3.1.1. Sensors. Now that we have equations, it's time to talk about the workhorses of the copter. Since the copter needs to know its current angular displacement and velocity, a gyroscope is essential for this robot. One big problem with a MEMS gyroscope integration is drift. In order for the quadcopter to stay airborne for long periods of time it needed reliable accelerometer angular data (Fisher, C. J., 2010). However, during movement, accelerometer readings are very noisy. A complementary sensor fusion algorithm is used with the gyroscope, accelerometer, and magnetometer to address these differing strengths and weaknesses.

In an attempt to eliminate sensor biases even more, calibration functions for the gyroscope and accelerometer are made to take out all background noise before flight. Calibration works by polling the sensors around a hundred thousand times while the quadcopter is stationary (with an assumed 1 g gravitational field), and subtracting this averaged value from sensor polls during flight.



[1]

Additionally, a median filter is used on the accelerometer readings to achieve even more clarity. Once the controllers and sensors work together to figure out the angular velocity to spin the motor, it's time to actually tell the motor to spin. Since we have brushless motors, we can use an ESC.

3.1.2. PWM. ESCs take in a PWM signal, and in turn energize motor windings in sequence to produce the desired motor rotation speed. ESCs nowadays support a 400 Hz cycle. In order to communicate with the motor, PWM signals between 1 ms to 2 ms are sent every 2.5 ms using the nRF PWM driver library.

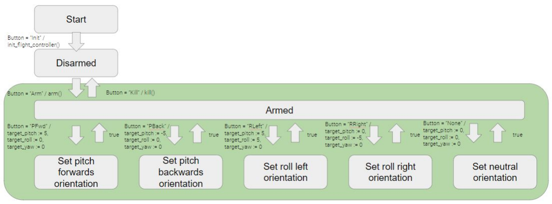
Now that we have set up all the necessary infrastructure, creating a flight controller is quite trivial. The pseudocode is as follows:

```
poll_sensors()
calculate_errors()
calculate_pwm() //via derived equation
update_pwm()
```

3.2. FSM / BLE

The copter uses a basic finite state machine (FSM) to update attitude targets, initialize sensors and controllers, arm motors, and toggle flight. Inputs to the FSM are given from the client computer to the copter via Bluetooth Low Energy (BLE). The PC regularly writes a BLE characteristic containing the current state of the buttons and the FSM updates accordingly. When the sensor initialization state and the flight state are enabled (green zone in diagram), every 50 ms cycle consists of a sensor value check, controller update, and motor actuation.

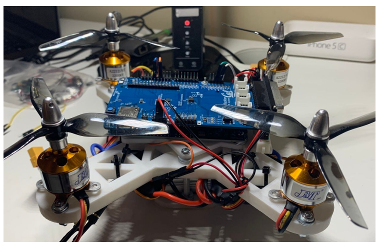
[1] We did not make this figure, but it shows data very similar to that of our copters output log



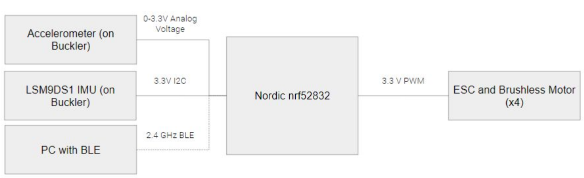
Above: Diagram of the BLE update FSM

4. Hardware

The quadcopter, mechanically, has a custom 3D printed frame and 4 propellers.



Electronically, the copter consists of an nRF52832 microcontroller, sensors via the Berkeley Buckler, electronic speed controllers (ESCs), brushless motors, a power converter, and a battery.



The above wiring diagram details how the sensors and BLE interface with the nRF board. Additionally, notice the GPIO connections were made to ESCs, and 5V connections were made to the 5V input on the board. Battery connections were not included on the diagram, but the battery is connected to all four ESCs and also a power converter to regulate its voltage from nominally 11.1 V to the 5V required by the input on the nRF development board.

5. Course Concepts

As mentioned prior, this project uses a BLE controlled FSM, sensors, I/O (for pwm), and discrete time systems.

6. Results

In the end, the quadcopter communicated via BLE and achieved pretty robust attitude control and we are very proud of the results. However, its altitude was pretty variable. There are a couple of reasons behind this. The main one is due to the square of angular velocity being mapped to PWM experimentally[2].

Unfortunately, because the voltage of the batteries drains as they discharge, selecting a certain PWM value to balance out the weight of the quadcopter did not work, since the thrust produced by that PWM value decreased as the batteries discharged. The best way to prevent this is to run a velocity controller on each individual motor in addition to the other controllers. Another idea is to use a pressure sensor to keep track of z's position a bit better. A project of this nature -- and with this budget -- is intended to take a while to tune, in the end we are very happy with our results and are pleased with how powerful our controllers turned out.

7. References

Bouabdallah, Samir & Noth, A. & Siegwart, Roland. "PID vs LQ Control Techniques Applied to an Indoor Micro Quadrotor. Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems.", 10.1109/IROS.2004.1389776, vol.3, 2004, 2451 - 2456.

Fisher, C. J., 2010. "Using an Accelerometer for Inclination Sensing" Analog Devices application note AN-1057 v0.

Gamazo-Real, José Carlos et al. "Position and speed control of brushless DC motors using sensorless techniques and application trends." Sensors (Basel, Switzerland) vol. 10,7 (2010): 6901-47. doi:10.3390/s100706901

Luukkonen, T. "Modelling and Control of Quadcopter", School of Science, Espoo, 2011

Schwartzberg, Milton A. "Rotor Induced Power", U.S. Army Aviation Systems Command, 1975, 3

Serway, Raymond A.; Jewett, John W. (2004). Physics For Scientists and Engineers (6thed.).Brooks/Cole. ISBN978-0-534-40842-8.

[2] links to videos and experimental data in github