January 24, 2016
Meeting 2 of Spring 2016

Pat and Mason's Demo

We can now use the logging parameters to control the desired pitch, yaw, and roll values. We can plot this data and export it into applications such as Excel.

Possible reason why eulerYawDesired is negative in stabilizer.c

Reference: http://owenson.me/build-your-own-quadcopter-autopilot/#toc3

There is a part where he explains what happens when yaw > 181:

"Notice that yaw isn't behaving as expected, the yaw is locked to your yaw stick - so when your yaw stick goes left 45degrees the quad rotates 45 degrees, when you return your stick to center, the quad returns its yaw. This is how we've coded it at present, we could remove the yaw stabilize controller and just let the yaw stick control yaw rate - but whilst it will work the yaw may drift and won't return to normal if a gust of wind catches the quad. So, when the pilot uses the yaw stick we feed this directly into the rate controller, when he lets go, we use the stab controller to lock the yaw where he left it.

As the yaw value goes from -180 to +180, we need a macro that will perform a wrap around when the yaw reaches -181, or +181. So define this near the top of your code:

```
#define wrap_180(x) (x < -180 ? x+360 : (x > 180 ? x - 360: x))
```

If you examine it carefully, if x is < -180, it adds +360, if it's > 180 then we add -360, otherwise we leave it alone."

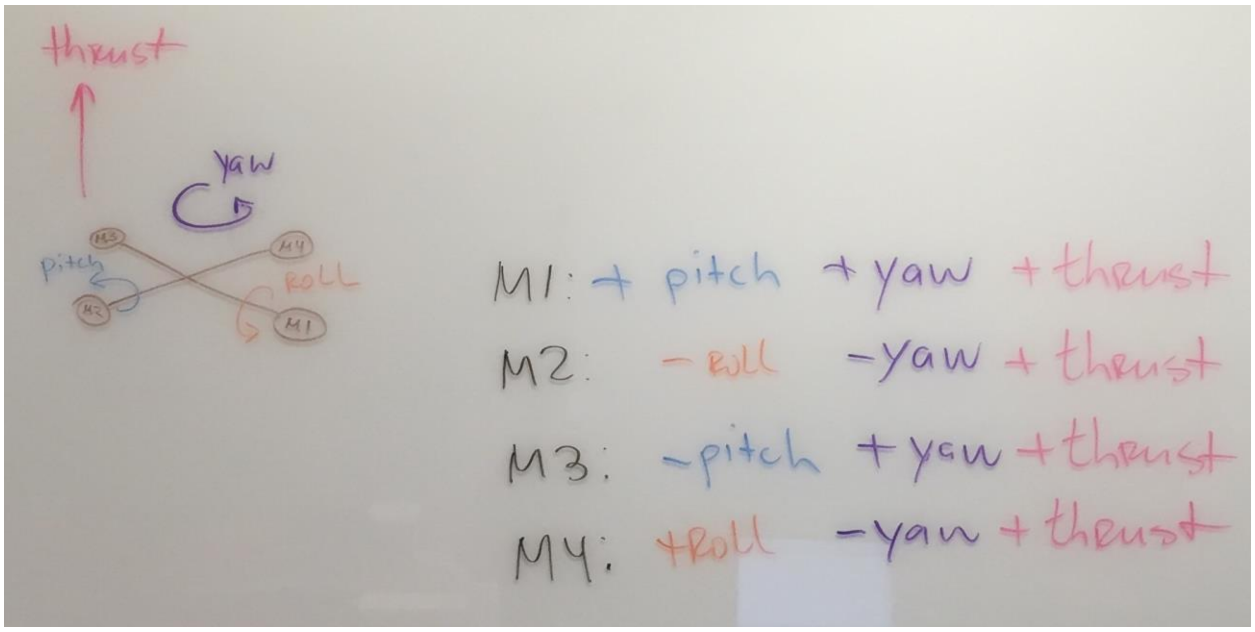
The fact that our yaw is negative in the code may be related to this, or it may be related to how the values are received from the controller.

Look at this code (in controller.c):

```
 // Update PID for yaw axis
 float yawError;
 yawError = eulerYawDesired - eulerYawActual;
 if (yawError > 180.0)
   yawError -= 360.0;
 else if (yawError < -180.0)
   yawError += 360.0;
 pidSetError(&pidYaw, yawError);
 *yawRateDesired = pidUpdate(&pidYaw, eulerYawActual, FALSE);
}
```

Here we adjust for yaw being bigger than 180. The negative sign is probably related to this.

Quadcopter Notes

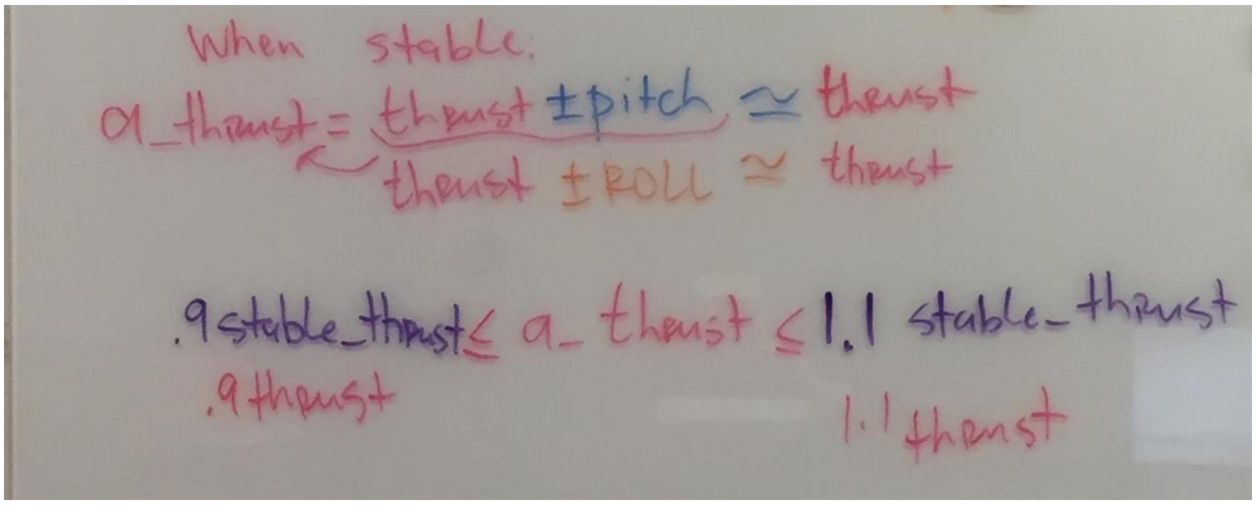


M1: + pitch + yaw + thrust

M2: − ROLL − yaw + thrust

M3: − pitch + yaw + thrust

M4: + ROLL − yaw + thrust



When stable:

$$\text{a\_thrust} = \underbrace{\text{thrust} \pm \text{pitch}}_{} \simeq \text{thrust}$$

$$\text{thrust} \pm \text{ROLL} \simeq \text{thrust}$$

$$.9\,\text{stable\_thrust} \leq \text{a\_thrust} \leq 1.1\ \text{stable\_thrust}$$

.9 thrust 1.1 thrust

Tasks

- → Find more ways to hover stably (set desired pitch/roll? minimize change? etc)
- → How to determine if the quad is stable?
- → Stable thrust
  - o 90 – 110% of “pink” thrust?
  - o 90 – 110% of some fixed value?
- → More Logging
  - o Get time -> thrust logs
  - o Printable format
  - o Examples/basiclog.py

Next Meeting

→ Talk about stable thrust
→ Decide on a stable hovering algorithm
→ Decide in which file we will put the code
→ Start hacking!