In4073
Embedded Real-Time Systems

Electrical Model Quad Rotor UAV
QR: Frames & Main Variables

- \( \mathbf{X} \), \( \mathbf{u} \), \( \mathbf{L} \), \( \mathbf{p} \), \( \mathbf{Z} \), \( \mathbf{w} \)
- Earth frame: \( \mathbf{x}, \mathbf{y}, \mathbf{z} \)
- Body frame: \( \mathbf{X}, \mathbf{u}, \mathbf{L}, \mathbf{p}, \mathbf{Z}, \mathbf{w} \)
- \( \mathbf{N}, \mathbf{r}, \mathbf{M}, \mathbf{q}, \mathbf{Y}, \mathbf{v} \)
QR Variables: Euler Angles
QR: Actuators

rotor 1 – rotor 4 through RPM, denoted by $\Omega$

driven by ES signals $ae_1 – ae_4$

$ae = 0x0000 \rightarrow \Omega = 0$
$ae = 0x03FF \rightarrow \Omega = \text{max}$
QR Dynamics (in hover)

\[ \frac{dq}{dt} = \frac{M}{I_Y} \]
\[ \frac{du}{dt} = \frac{X}{m} \]

\[ T_i = \text{rotor thrust} = f(\Omega_i) \]
\[ mg = \text{gravity} \]
\[ h = \text{rotor distance ref. center of gravity} \]
\[ I_Y = \text{heli rotation inertia in Y-axis} \]
QR: Rotor Actuators

In general

\[ Z = -b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \]
\[ L = b(\Omega_4^2 - \Omega_2^2) \]
\[ M = b(\Omega_1^2 - \Omega_3^2) \]
\[ N = d(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 - \Omega_3^2) \]

So compute \( \Omega_i \) (i.e., \( ae_i \)) from desired lift (Z), roll rate (L), pitch rate (M), and yaw rate (N) (see qrsim for example!)
QR: Gyro Sensor

Invensense IDG500

This is a dual-axis rotational-rate sensing device. It produces a positive output voltage for rotation about the X- or Y-axis, as shown in the figure below.

Orientation of Axes of Sensitivity and Polarity of Rotation

IDG-500

28-pin, 4mm x 5mm x 1.2mm QFN Package
QR: Accelerometer Sensor

STMicroelectronics LIS344AL

\[ a_x = \sin \Theta \ g \sim \Theta \ g \]
System SW view

joystick

PC

PC link

QR

actuators

sensors

ES

lift
roll
pitch
yaw

phi
theta

ae1
ae2
ae3
ae4

sax
say
saz
sp
sq
sr

In4073 Emb RT Sys (2011-2012)
control loop example (yaw rate):
\[ \text{eps} = \text{yaw} - \text{sr}; \]
\[ \text{L\_needed} = P \times \text{eps}; \]
\[ \text{ae1} \ldots \text{ae4} = f(\text{L\_needed}); \]

// measure deviation
// compute ctl action
// actuate, see slide 7
Calibration

- real \( p, q, r, \ldots \) are sensed in terms of \( sp, sq, sr, \ldots \)
- \( sp, sq, \ldots \) have a (voltage) bias (are not zero at rest)
- so need to calibrate all 6 sensors at rest:
  - let \( sr_0 \) be sensor output at rest
  - real estimate of \( r \) are given by \((z\) for zeroed\): \( zr = sr - sr_0 \)

\[ sr \quad \text{cal} \quad + \quad sr_0 \quad \text{mode} \quad + \quad zr \]
Filtering

- signals also need to be *filtered* to remove noise
- filtered signal input to embedded controller
Controller Modes

- controller mode: manual
- controller model: calibrate
- controller mode: control (yaw, pitch, roll)