# Responsive Flight Software Development & Verification Techniques for Small Satellites

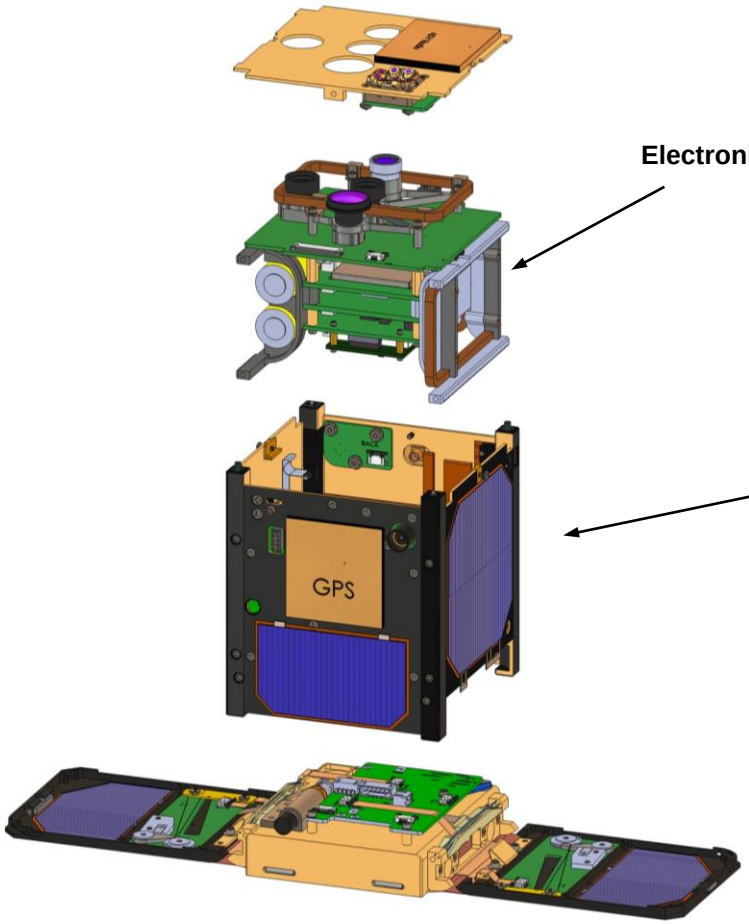
Darren Rowen
The Aerospace Corporation

Vehicle Systems Division
9 November 2012

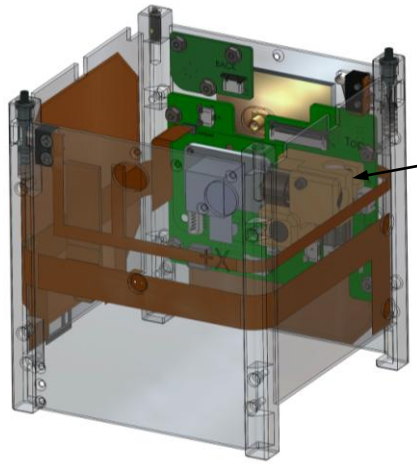
# Overview

- Project Background
- System Architecture
- Software Process
- Single-Axis System Test

# AeroCube Exploded View



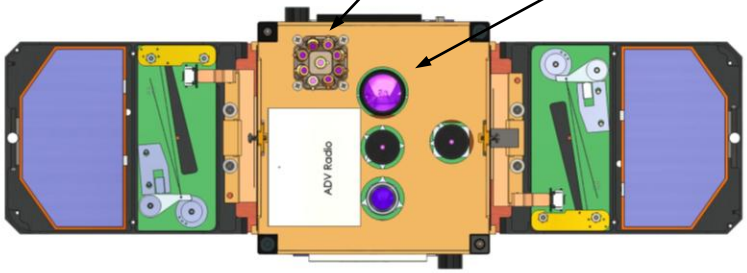

Electronics stack



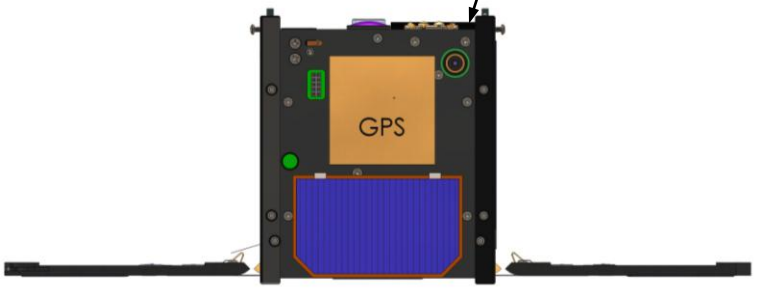
Reaction wheels hard-mounted to satellite body

Earth sensor

Visible cameras

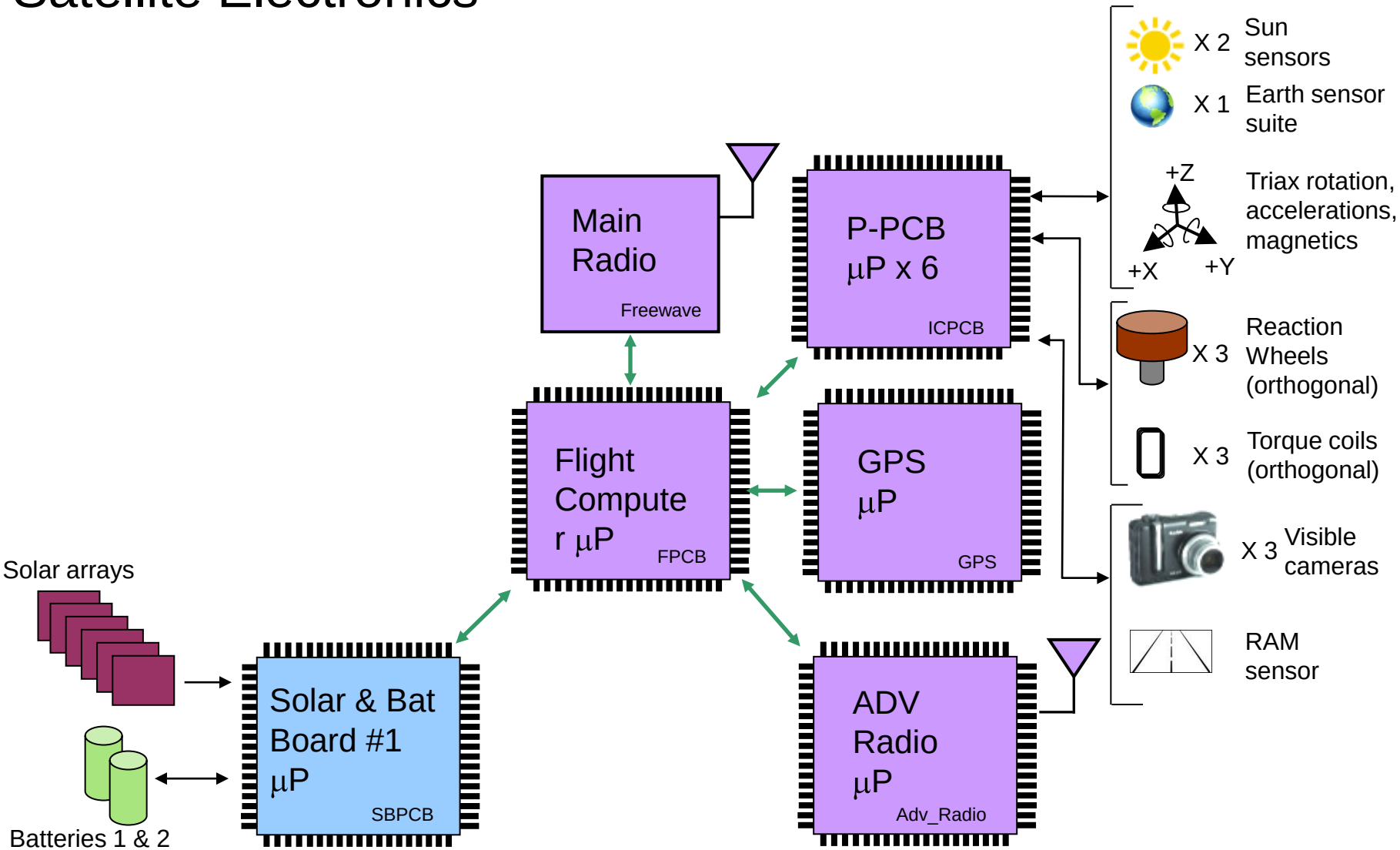



Sun sensor



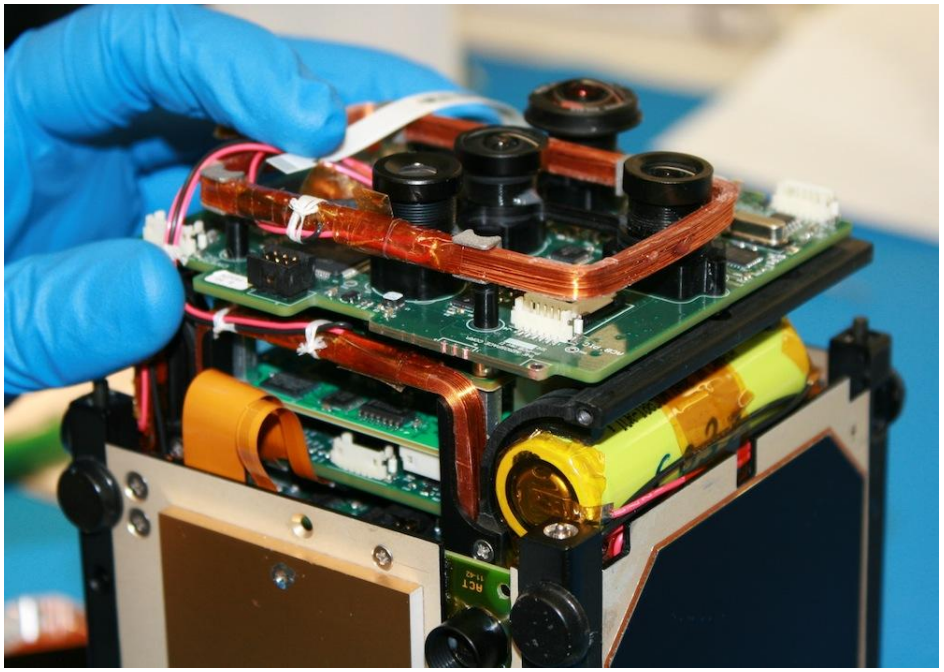

***Satellite is 10x10x10 cm and Weighs <1.5 kg***

# Satellite Electronics



# Satellite Electronics

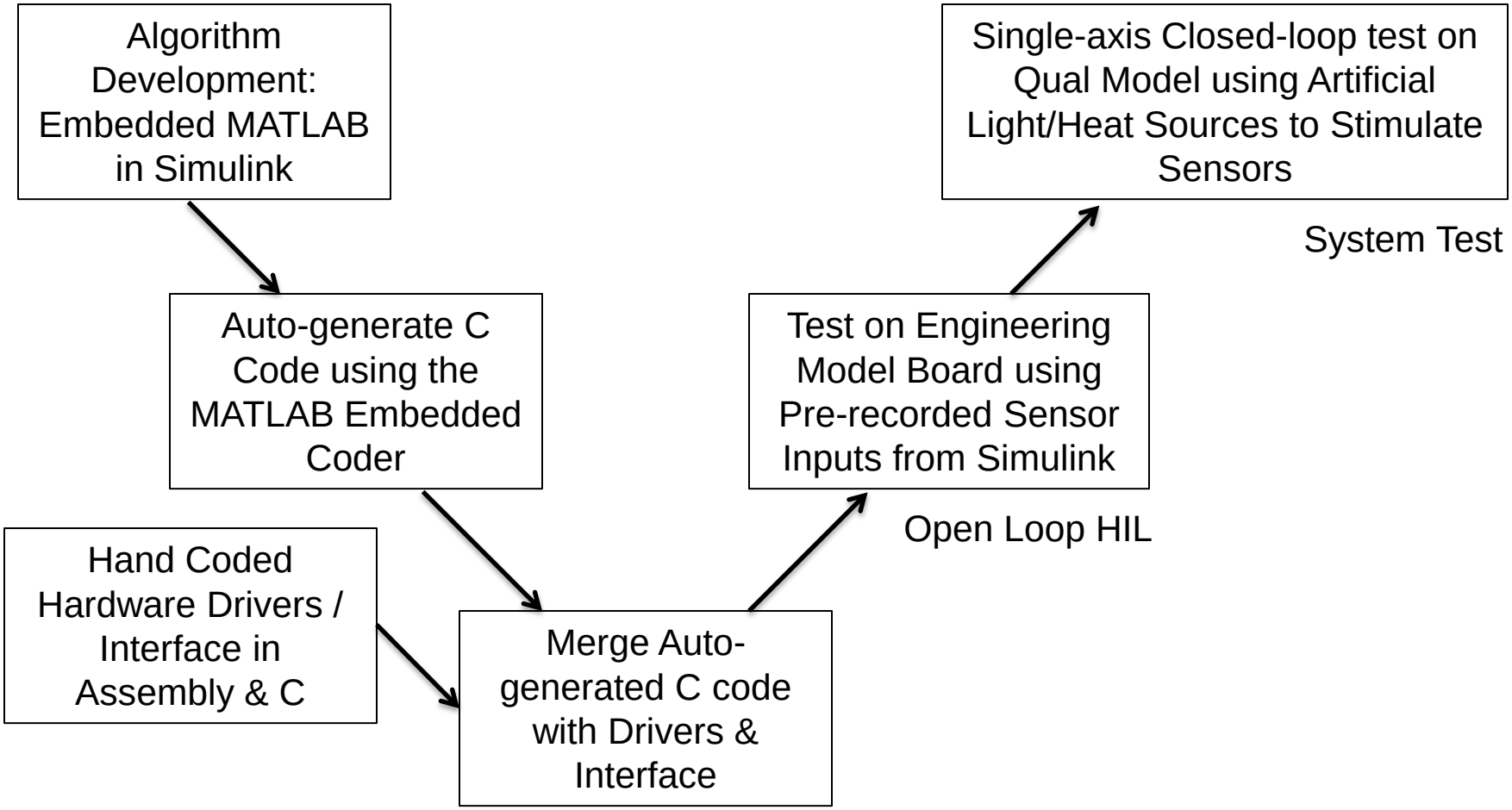
- Attitude Control on a PIC Processor
  - *8-bit architecture*
  - *Unfolded 32-bit floating point math*
  - *No operating system*
  - *Timing through on-chip timers*



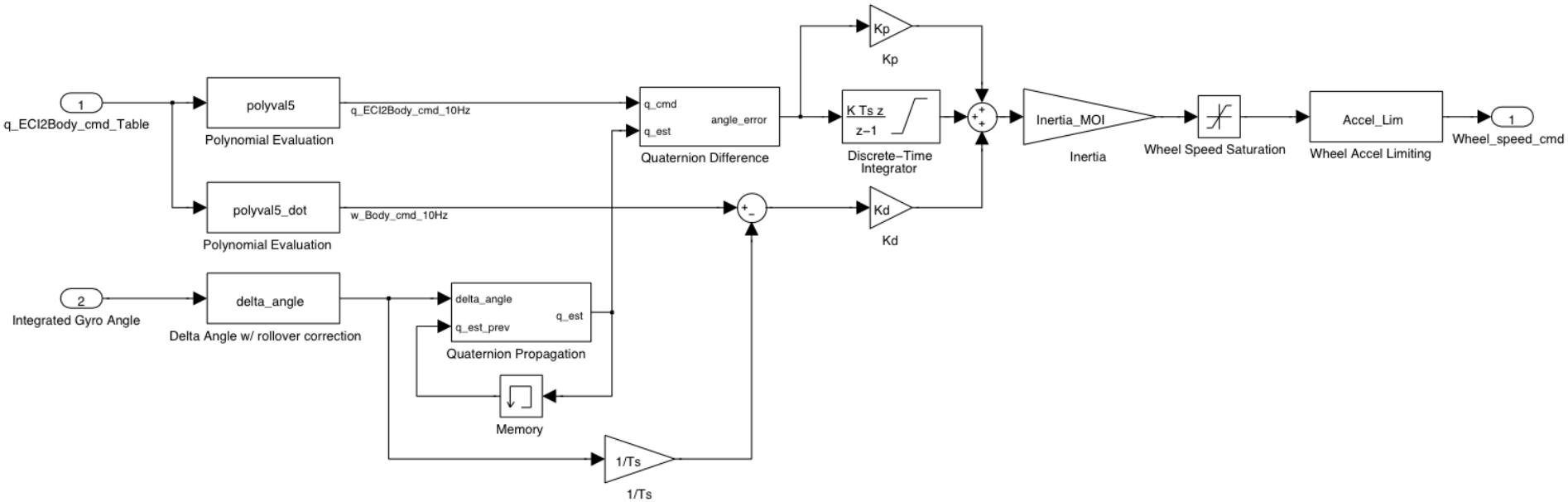
# Objectives

- Develop an enhanced attitude determination & control algorithm for an existing hardware architecture
  - *Support both nadir-pointing and inertial tracking of ground uploaded profiles*
  - *Estimate gyro bias onboard to reduce attitude error when Earth or Sun data is not available*
- Implement algorithm on an 8-bit PIC
- Deliver shipment code in 2-3 months, with an opportunity to patch on-orbit 6 months later
- Verify algorithm & code using both engineering & qualification model hardware via ground test

# Software Development Process Flow



# Control System Block Diagram



# Flight Software Auto-Code Generation

- ACS Algorithms were developed in MATLAB
- The MATLAB Coder was used to auto-generate C code that was later merged with hand coded low level drivers and supporting command and data handling functions

/Users/dwr27059/Documents/AeroCube/Attitude_Control.m

```
persistent Integrated_Gyro_Angle_prev Quat_ECI2Body_est Integral_State

if isempty(Integrated_Gyro_Angle_prev)
    Integrated_Gyro_Angle_prev=zeros(3,1);
    Quat_ECI2Body_est=[0;0;0;1];
    Integral_State=zeros(3,1);
end

% Quaternion Interpolation
q_ECI2Body_cmd_10Hz=QuatProp_rate(q_ECI2Body_cmd_1Hz,Ts,w_Body_cmd_1Hz);

% Delta Angle w/ rollover correction
deltatheta=Integrated_Gyro_Angle-Integrated_Gyro_Angle_prev;
for n=1:3
    if deltatheta(n)>deltatheta_Max
        deltatheta(n)=deltatheta(n)-Integrated_Gyro_Angle_RolloverNeg;
    elseif deltatheta(n)<deltatheta_Min
        deltatheta(n)=deltatheta(n)+Integrated_Gyro_Angle_RolloverPos;
    end
end
Integrated_Gyro_Angle_prev=Integrated_Gyro_Angle;

% Quaternion Propagation
Quat_ECI2Body_est=QuatProp_deltatheta(Quat_ECI2Body_est,deltatheta);

% Quaternion Difference
QuatError=QuatDiff(Quat_ECI2Body_est,q_ECI2Body_cmd_10Hz);
QuatError=MakeQuatCanonical(QuatError);

Att_Control_Error=2*QuatError(1:3);

% Rate Error
w_Body_est=deltatheta/Ts;
Att_Rate_Error=w_Body_cmd_1Hz-w_Body_est;

% PID
% Backward Euler Discrete-Time Integrator
Integral_State = Integral_State + Ki*Ts*Att_Control_Error;
for n=1:3
    if Integral_State(n) > Integral_State_Max
        Integral_State(n)=Integral_State_Max;
    elseif Integral_State(n) < Integral_State_Min
```

MATLAB Code

/Users/dwr27059/Documents/AeroCube/Attitude Control/Attitude Control.c

```
  }

  c2_QuatError[3] = (1.0 - c2_mag2 / 8.0) + c2_mag2mag2 / 384.0;
  c2_QuatAdd(c2_QuatError, *c2_q_ECI2Body_cmd_1Hz, c2_q_ECI2Body_cmd_10Hz);
  for (c2_k = 0; c2_k < 3; c2_k++) {
    c2_mag2 = (*c2_Integrated_Gyro_Angle)[c2_k] -
      c2_Integrated_Gyro_Angle_prev[c2_k];
    if (c2_mag2 > c2_b_deltatheta_Max) {
      c2_mag2 -= c2_b_Integrated_Gyro_Angle_RolloverNeg;
    } else {
      if (c2_mag2 < c2_b_deltatheta_Min) {
        c2_mag2 += c2_b_Integrated_Gyro_Angle_RolloverPos;
      }
    }

    c2_Integrated_Gyro_Angle_prev[c2_k] =
      (*c2_Integrated_Gyro_Angle)[c2_k];
    c2_y[c2_k] = c2_mag2*c2_mag2;
    c2_theta[c2_k] = c2_mag2;
  }

  c2_mag2 = c2_y[0];
  for (c2_k = 0; c2_k < 2; c2_k++) {
    c2_mag2 += c2_y[c2_k + 1];
  }

  c2_mag2mag2 = c2_mag2 * c2_mag2;
  c2_sin_term = (0.5 - c2_mag2 / 48.0) + c2_mag2mag2 / 3840.0;
  for (c2_k = 0; c2_k < 3; c2_k++) {
    c2_QuatError[c2_k] = c2_theta[c2_k] * c2_sin_term;
  }

  c2_QuatError[3] = (1.0 - c2_mag2 / 8.0) + c2_mag2mag2 / 384.0;
  for (c2_k = 0; c2_k < 4; c2_k++) {
    c2_dv2[c2_k] = c2_Quat_ECI2Body_est[c2_k];
  }

  c2_QuatAdd(c2_QuatError, c2_dv2, c2_Quat_ECI2Body_est);
  for (c2_k = 0; c2_k < 4; c2_k++) {
    c2_QuatError[c2_k] = 0.0;
  }
```

Auto-generated C Code

***Streamlined Flight Software Development***



Darren.Rowen@aero.org
Control Analysis Department

AEROSPACE

# Flight Software Integration

- Leveraged legacy AeroCube software as a starting point
  - *Low-level hardware drivers already in place*
  - *Mixed language architecture: C and Assembly*

- Implemented auto-code generated attitude control algorithm
  - *Memory map optimization (90% program memory utilization)*
  - *Added support for on-orbit reprogramming*

- Developed optimized telemetry storage system
  - *Multi-rate storage to reduce data size*

- Partitioned flash memory for algorithm inputs & telemetry storage

# Open Loop HIL Testing

- Verified timing margin
- Identified & corrected underflow error in auto-generated code
  - *Embedded code generator used a hard coded double precision value for an epsilon in a library function instead of single precision*
  - *Feedback provided to The Mathworks*
- Identified & corrected interface bugs
- Identified hardware error with the real-time clock register roll-over
  - *Sampling issue in the real-time clock integrated circuit*
  - *Corrected with a software detection algorithm*
- Compared stored telemetry output to simulation predicts

# System Testing

- Verified timing margin
- Identified & corrected senor read no-response issues
  - *Added timeouts based on expected sensor read times*
- Identified hardware design flaw – reaction wheel vibrations at high rotation rates corrupting gyro senor data
  - *Demonstrated that reduced wheel speeds alleviated the problem*
  - *Changed reaction wheel mounting for follow-on mission*
- Identified & corrected Sun sensor processing database constants
  - *Improved gyro bias estimation performance*
- Compared stored telemetry output to simulation predicts

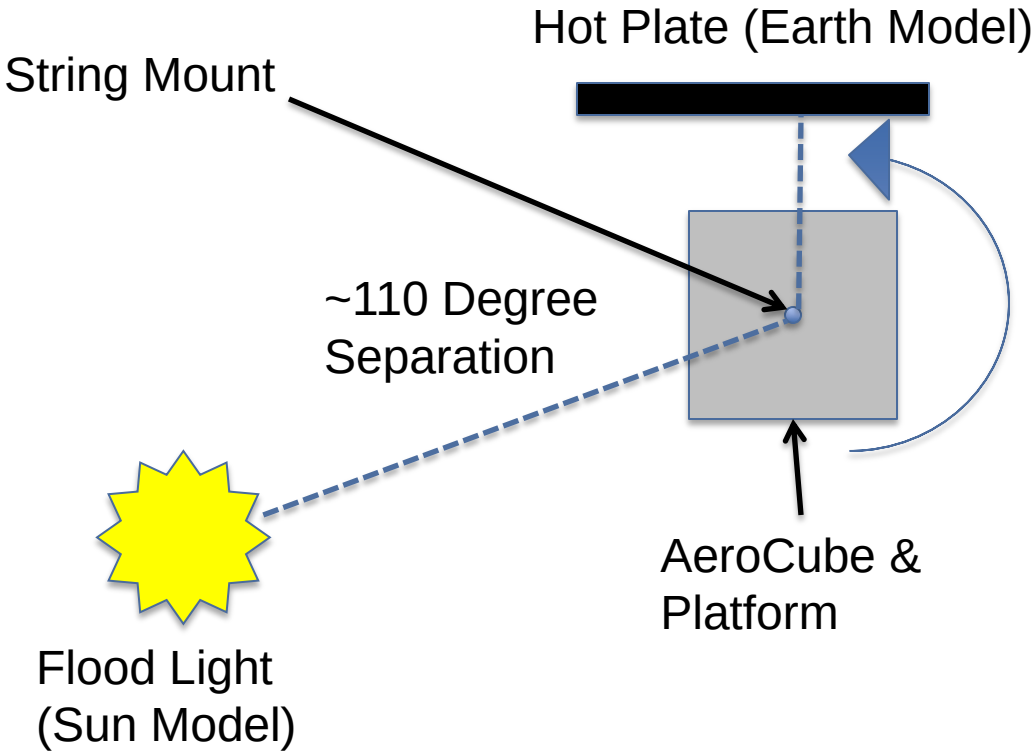
# Single-Axis System Test

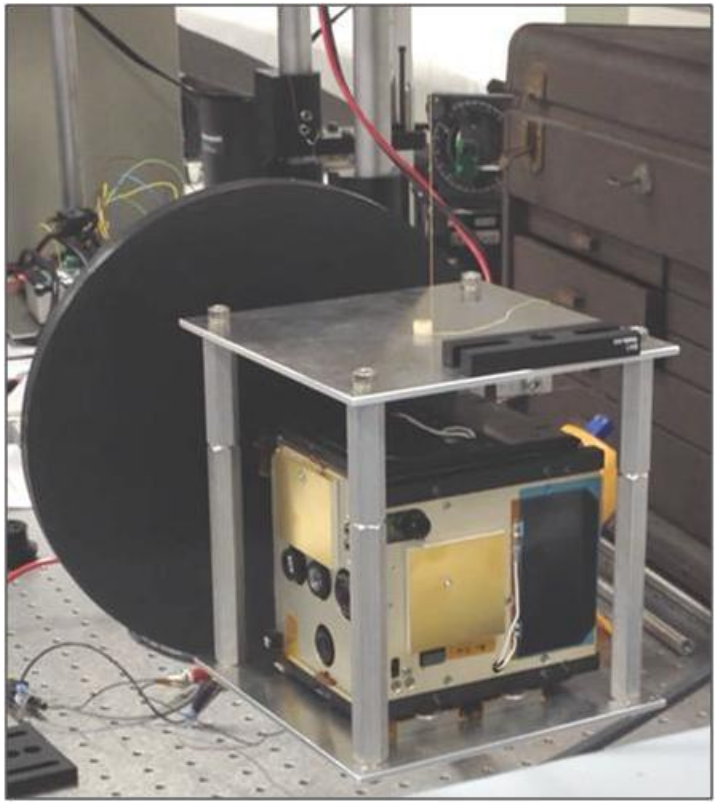
# Overview

- Test Setup
- Scenario Con-ops
- Simulation Comparison
- Gyro Bias Estimation Results

# Test Setup

*Overhead view*



Spacecraft on hanging string fixture with Earth target visible

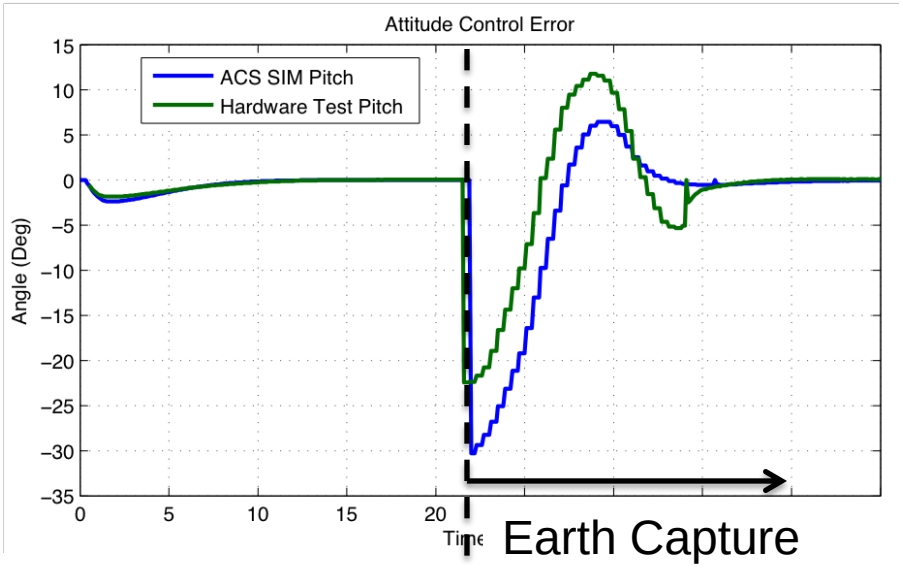
***Low Cost System Test Identified Several Software and Hardware Issues***

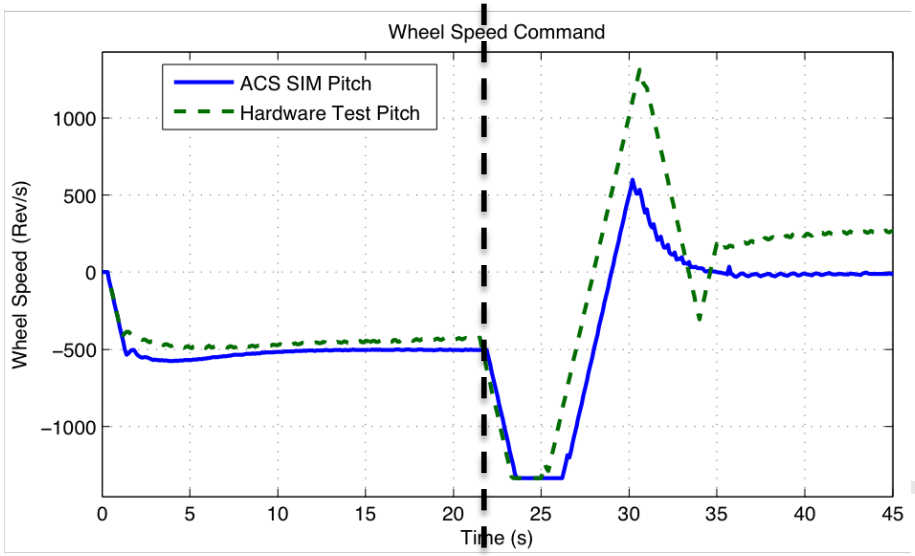
# Scenario Con-ops

| Simulation Event | Time (sec) |
|---|---|
| Start Earth acquisition<br>- Y-axis Earth search (about string axis) | 0 |
| Found Earth, pull-in | 22 |
| Start Z-axis Sun search (perpendicular to string) | 32 |
| Found Sun (already in field of view) | 32 |
| Attitude determination initialization<br>- Switch to ephemeris based Nadir-Sun commanding<br>- Start gyro bias estimation | 34 |
| Transition to inertial tracking mode<br>- Maintain Nadir-Sun attitude | 316 |
| Begin inertial profile | 388 |
| 180 Deg Y-axis slew maneuver (about string axis) | 388 |
| Gyro bias estimation halted – no sensor data | 397 |
| Return to Nadir-Sun attitude | 588 |
| End inertial profile & test | 788 |

# Closed Loop Performance

*Earth Acquisition*

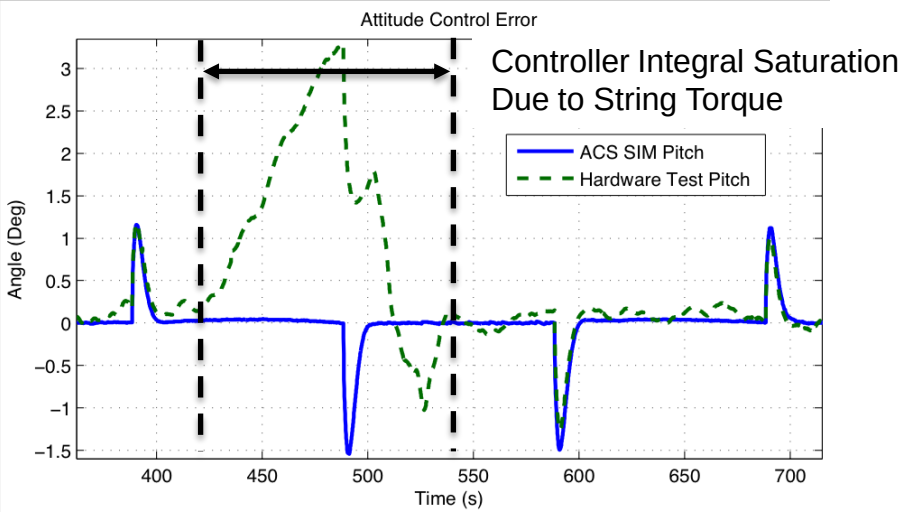


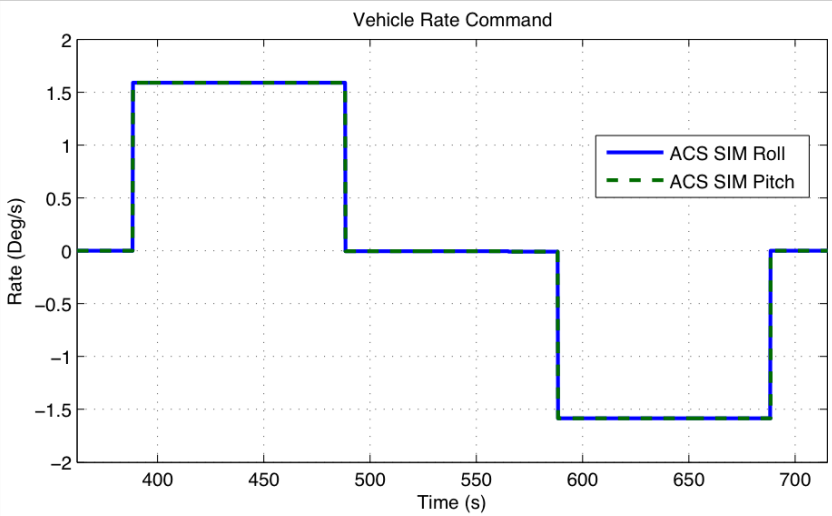




- Differences attributed to:
  - *Larger inertia for hardware test due to under estimate of platform inertia*
  - *String disturbance torque in hardware test but not simulation*
  - *Earth sensor model mismatch at large off center angles*

# Closed Loop Performance

*Inertial Tracking*



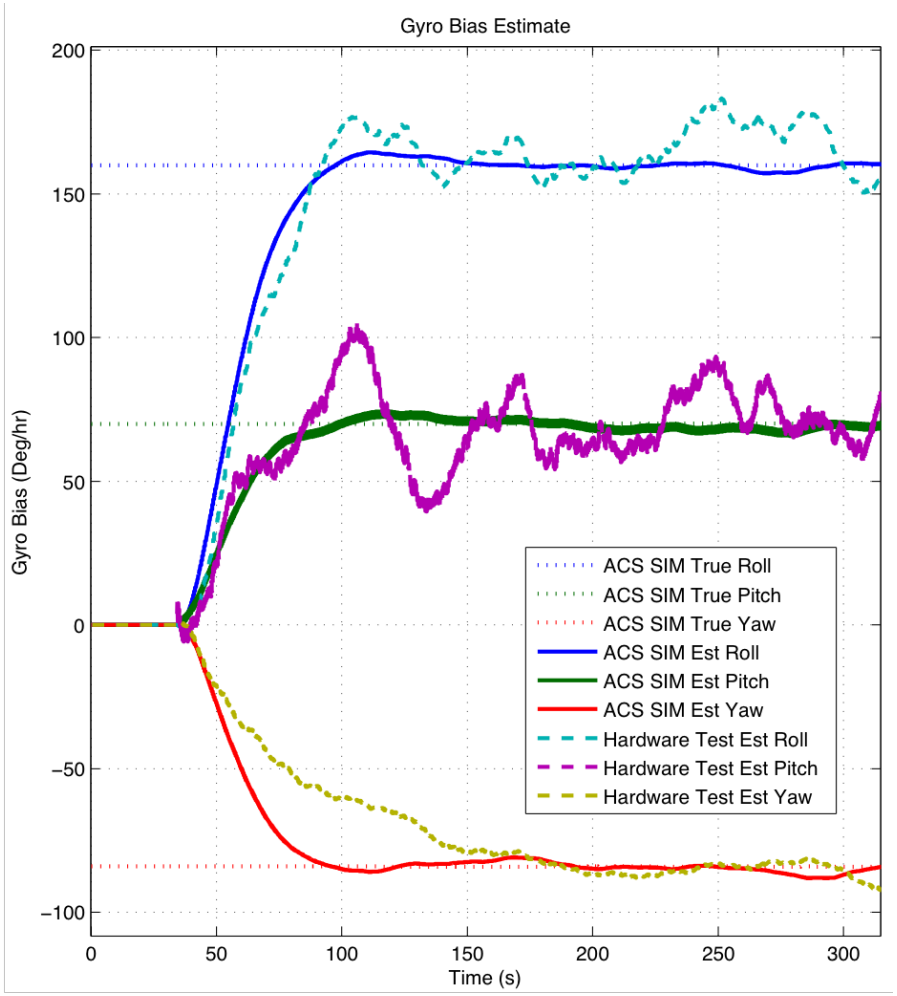

- During Slew about +Y the controller's integral term saturated due to string restoring torque
- Slew about -Y did not exhibit this problem because it was "unwinding" the string so the restoring torque wasn't a big factor
- Attitude control error is in agreement with simulation predicts despite larger disturbance environment


Darren.Rowen@aero.org
Control Analysis Department




AEROSPACE

# Gyro Bias Estimation Results




- Simulation "True" bias selected based on mean value of test estimate during steady State for comparison purposes
- Note that test estimate is likely noisier in roll and pitch due to swinging of the test platform on the string (and resulting variations in the earth sensor readings)
- Results shown are for when both Earth and Sun data are being used for the estimate
- Behavior matches simulation predicts fairly well
- Verifies estimation algorithm, sensor alignments, database selections

# Thank you