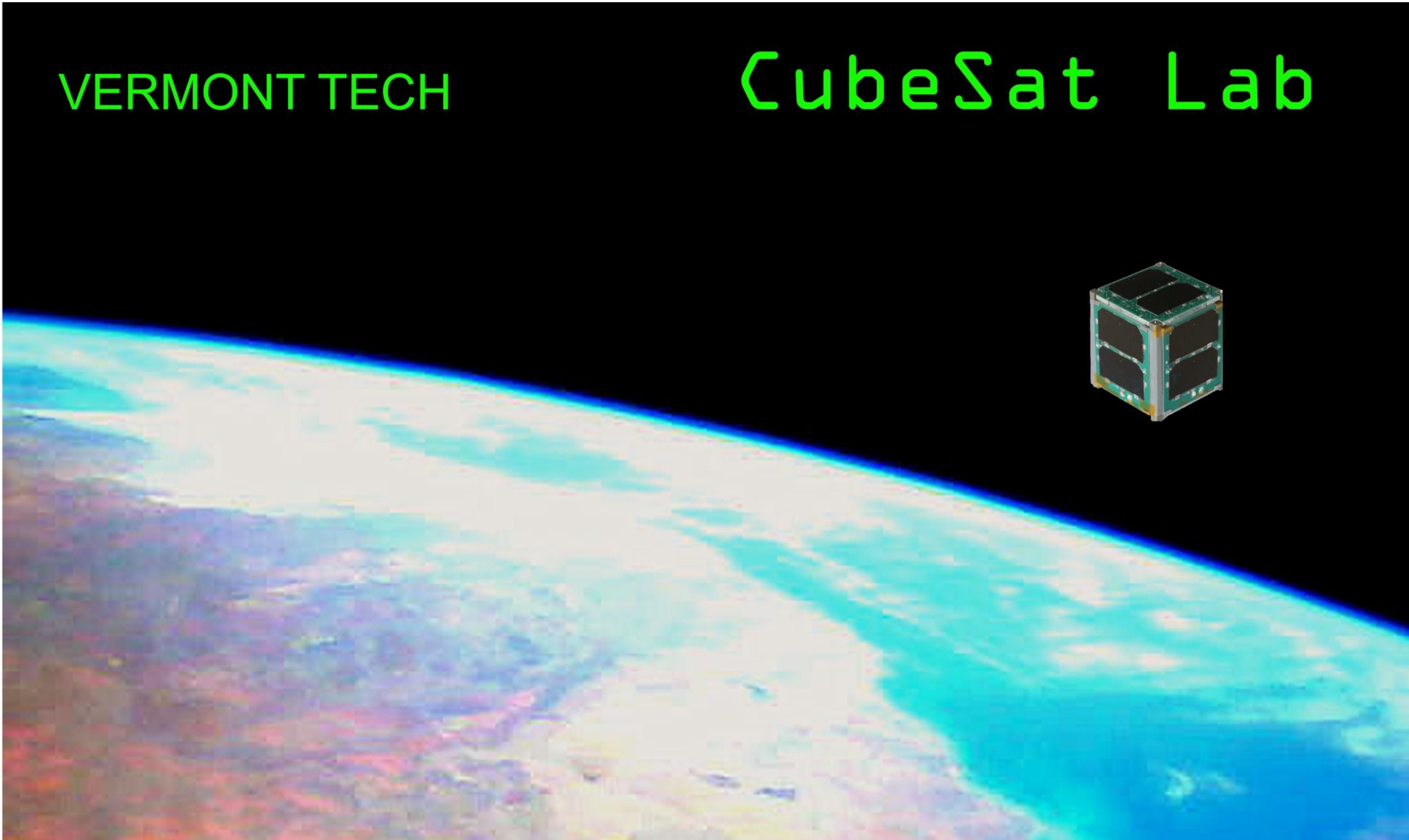
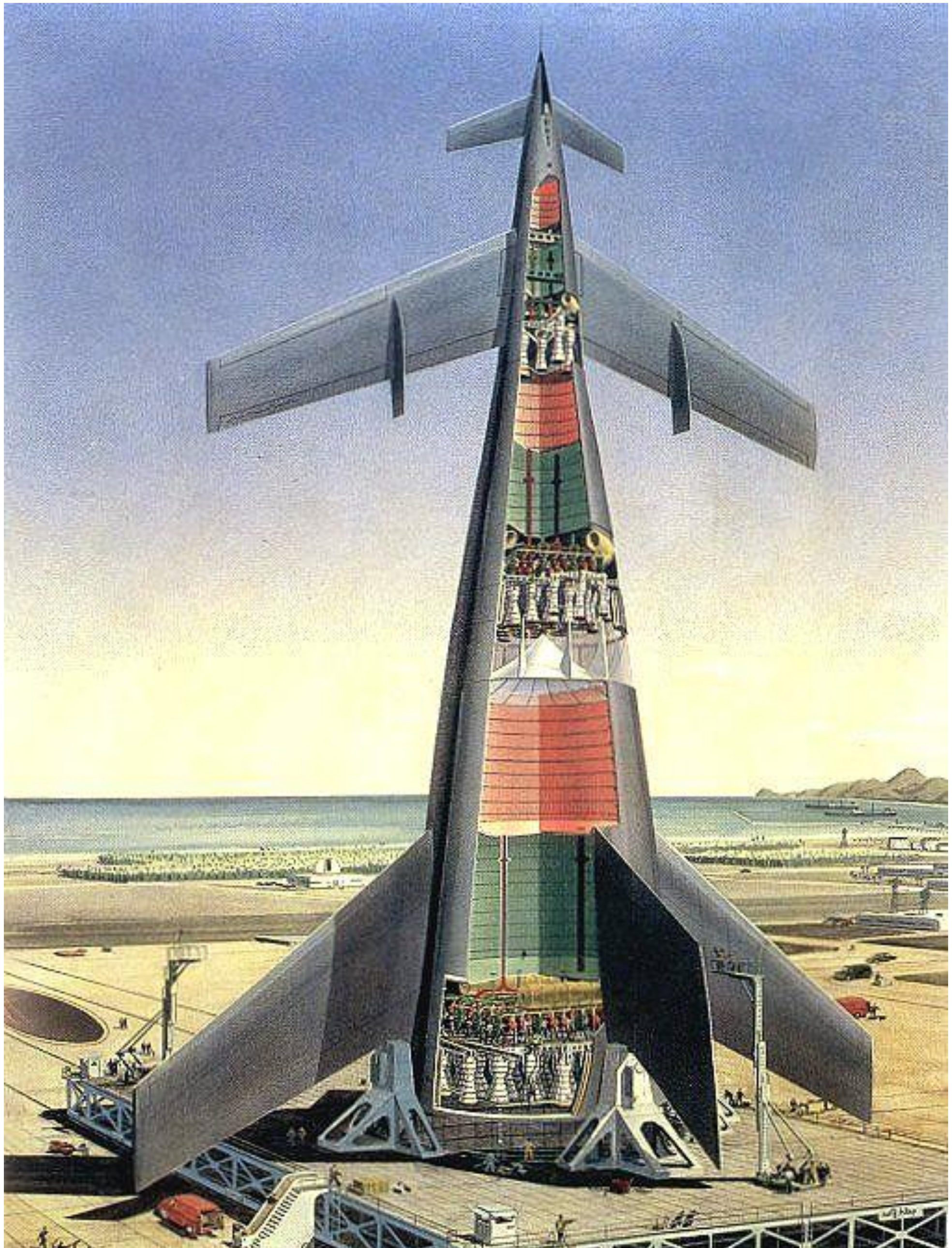
From Physicist to Rocket Scientist and How to make a CubeSat that Works!

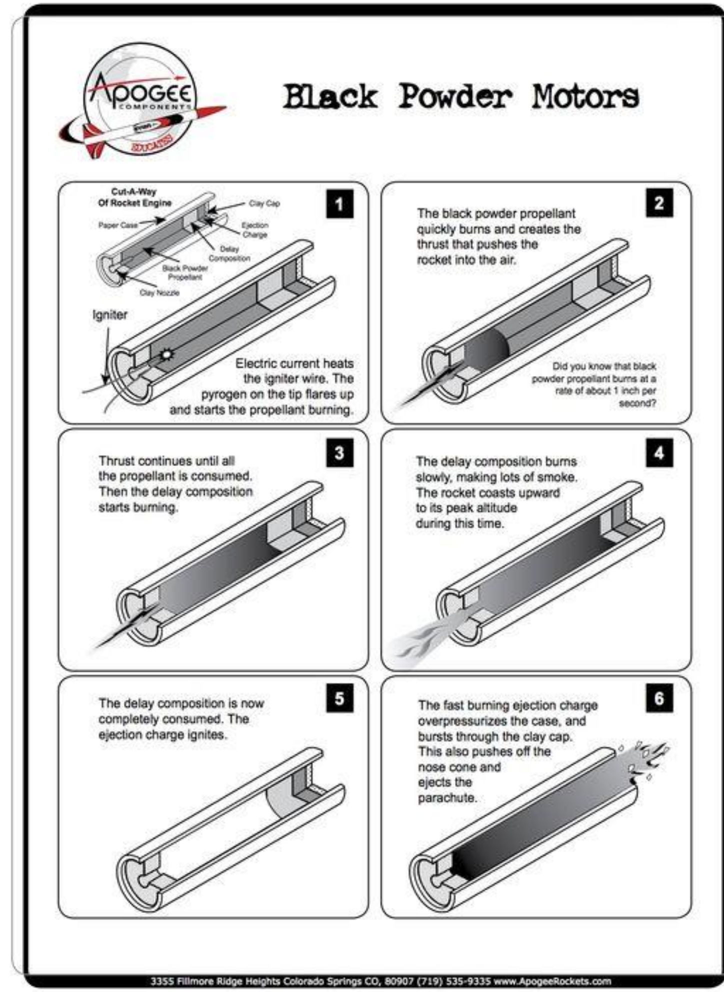
Dr. Carl Brandon
carl.brandon@vtc.edu
Vermont Technical College
+1-802-356-2822
Randolph Center, VT 05061 USA
http://www.cubesatlab.org



I became interested in science early, following the Collier's Magazine space series in the 1950s



My First Rocket, in First Grade
Roselle Avenue School



For my eighth grade science fair, I created a high altitude/space instrument system:

Geiger counter, UV, temperature & humidity sensors, multiplexor, transmitter, home made from scratch oscilloscope.

At Michigan State, in the summer after my Freshman year, 1963, I started using their vacuum tube computer, MISTIC

Facts about MISTIC

- MISTIC contained 2,610 vacuum tubes for processing and memory.
- Arithmetic Unit and Storage was in a cabinet that was 10 feet (3.0 m) high and 11 feet (3.4 m) long.
- Electrostatic memory of 1,024 by 40 bit words.
- Computations were output on a Teletype printer at the rate of 10 characters per second.
- 12,500 word magnetic drum storage.
- Input by Friden Flexowriter punched paper tape.

I was working on the design of the extractor coil for our new 60MeV, 64” Cyclotron.

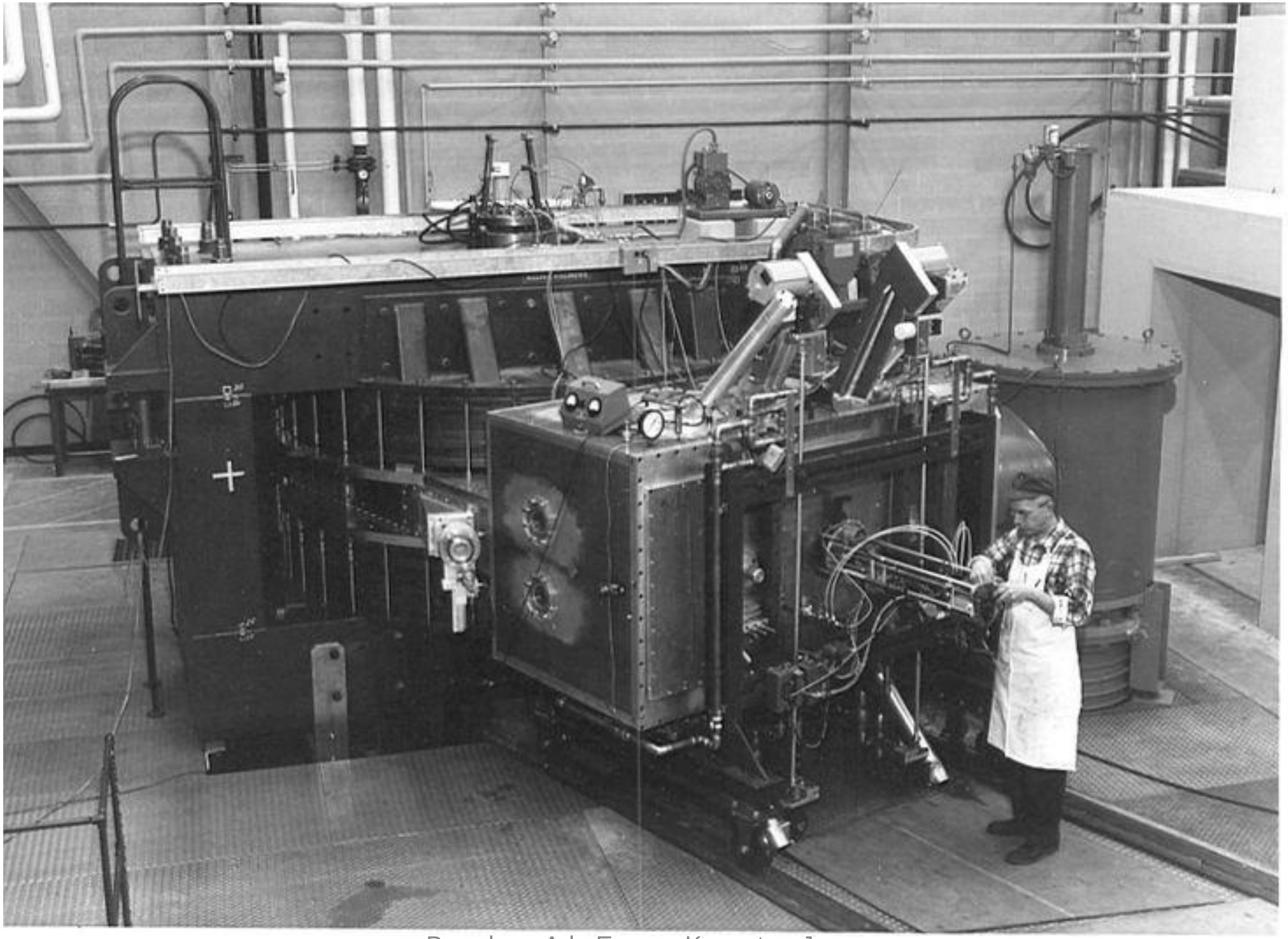
We got a new Control Data 160A for data input

For the new Control Data 3600

Michigan State K-50 Cyclotron Scale Model Magnet Henry Blosser, my advisor.

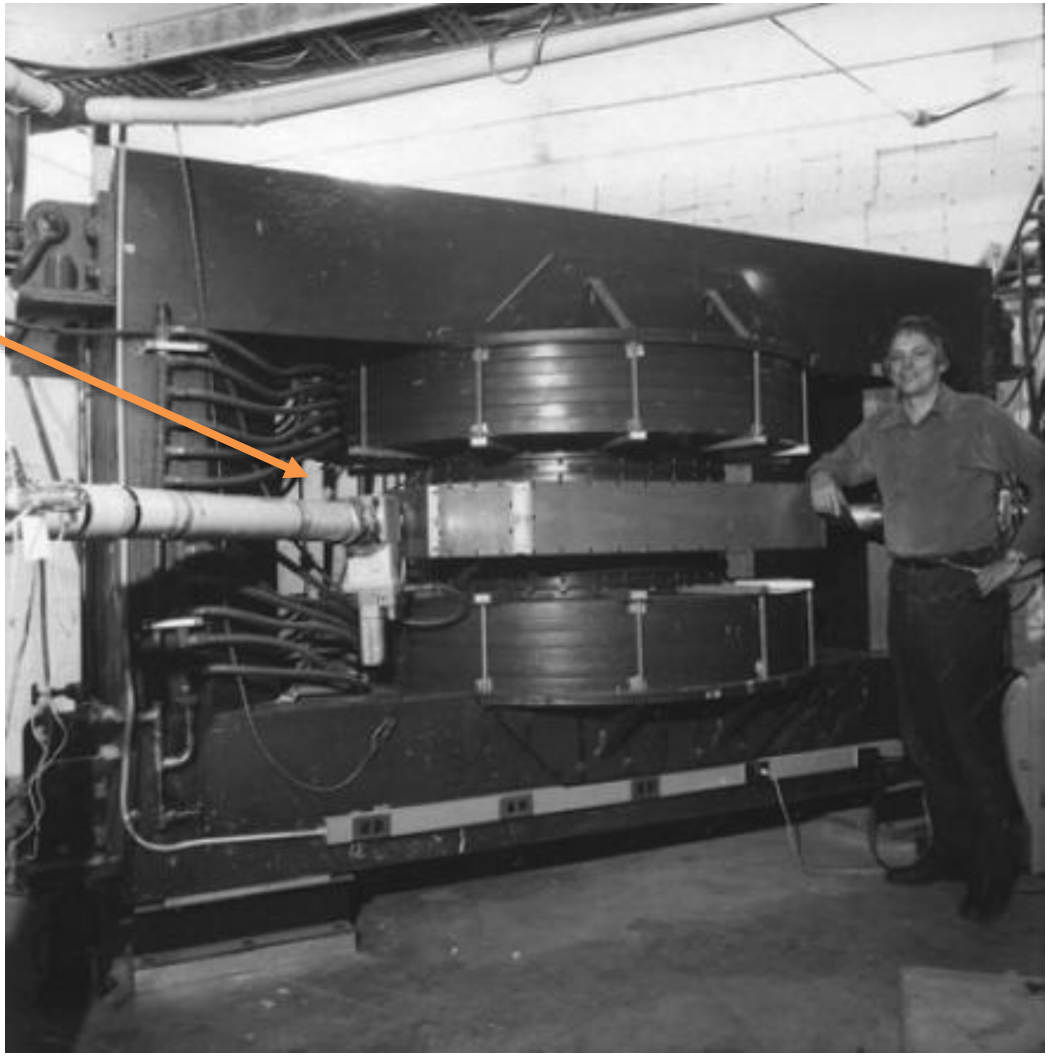
Michigan State K-50 Cyclotron magnet

Michigan State K-50 Cyclotron



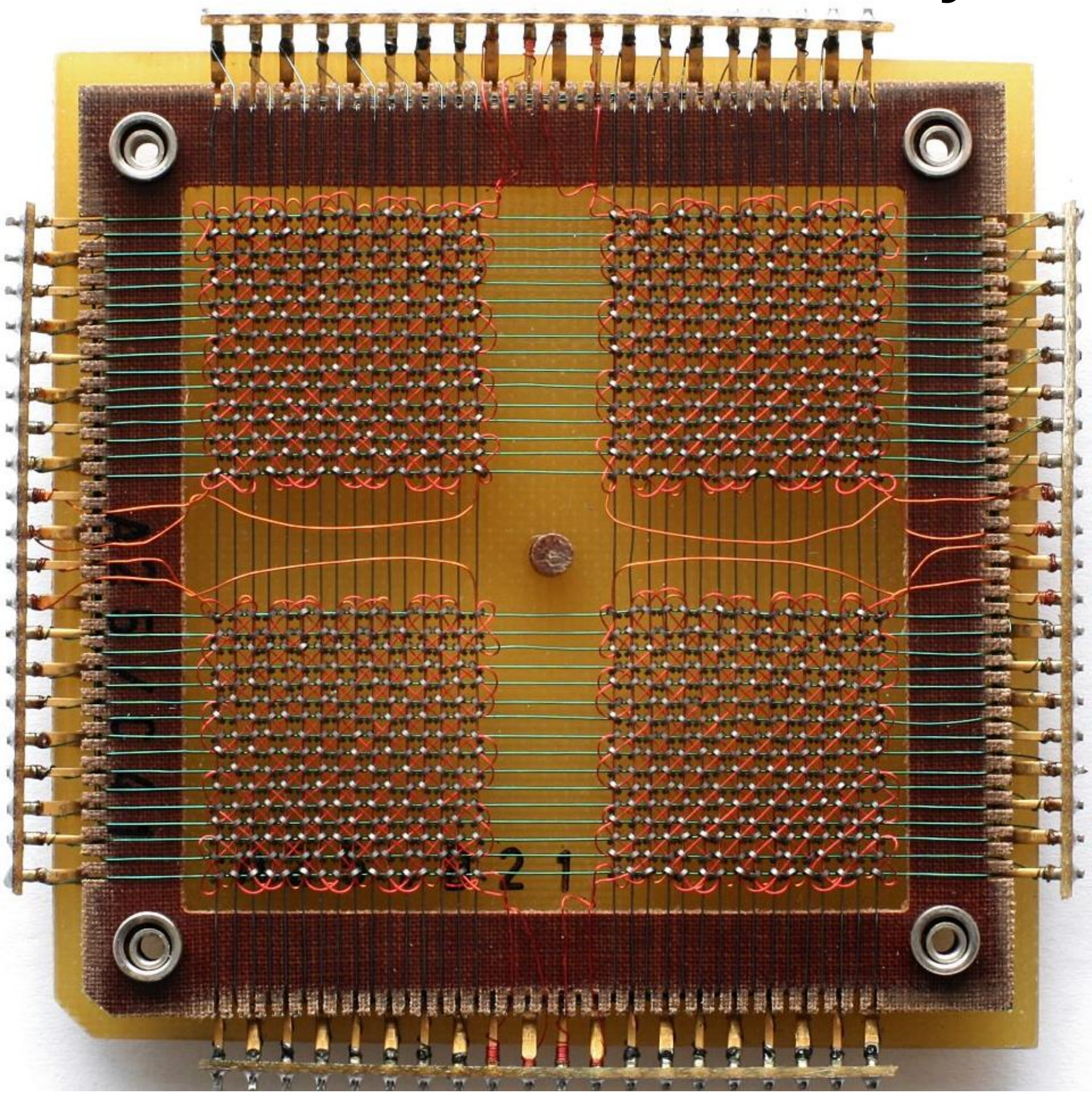
Michigan State K-50 Cyclotron

Extractor
Coil

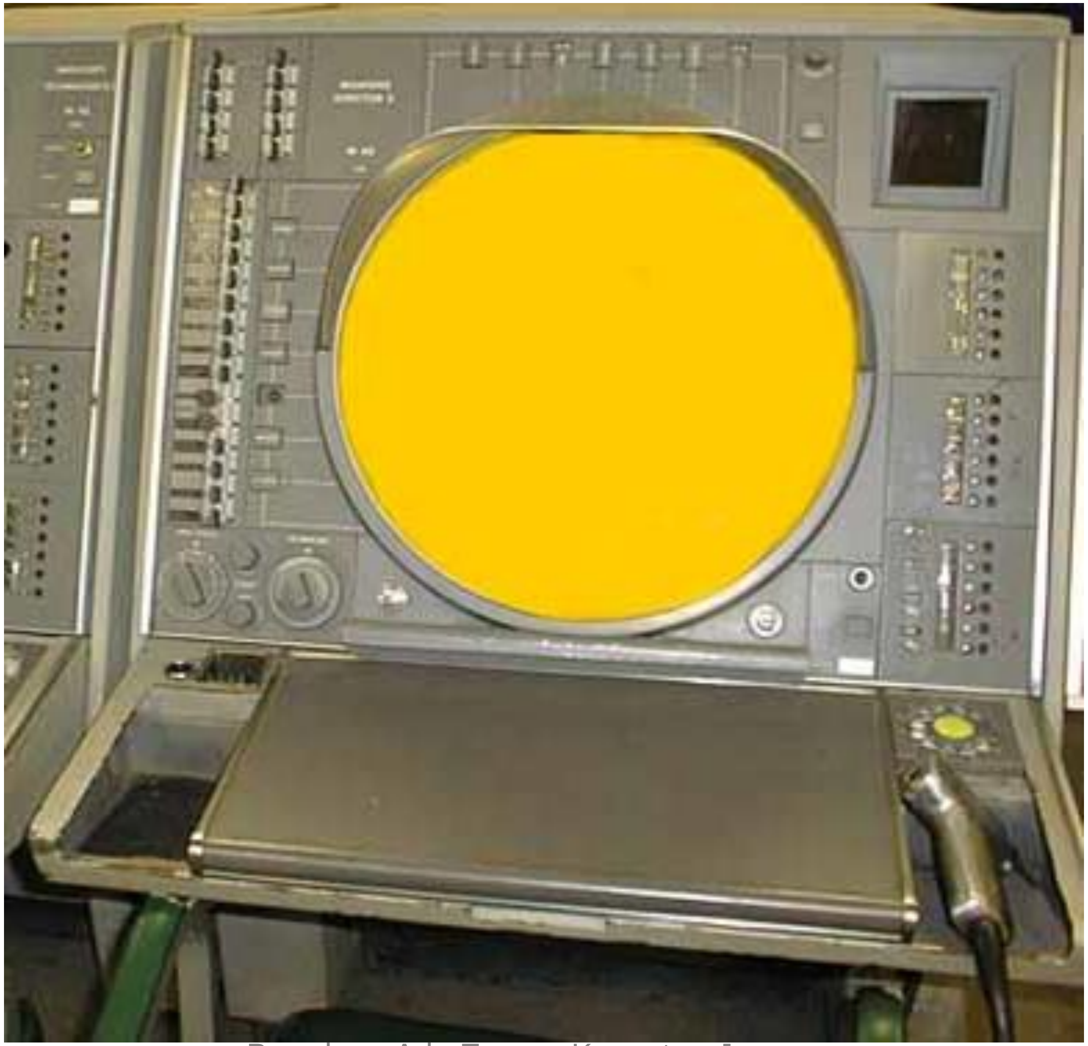


IBM Yorktown Heights Basic Research Lab Magnetic Properties of Europium Oxide At the Curie Point, Summer of my B.S.

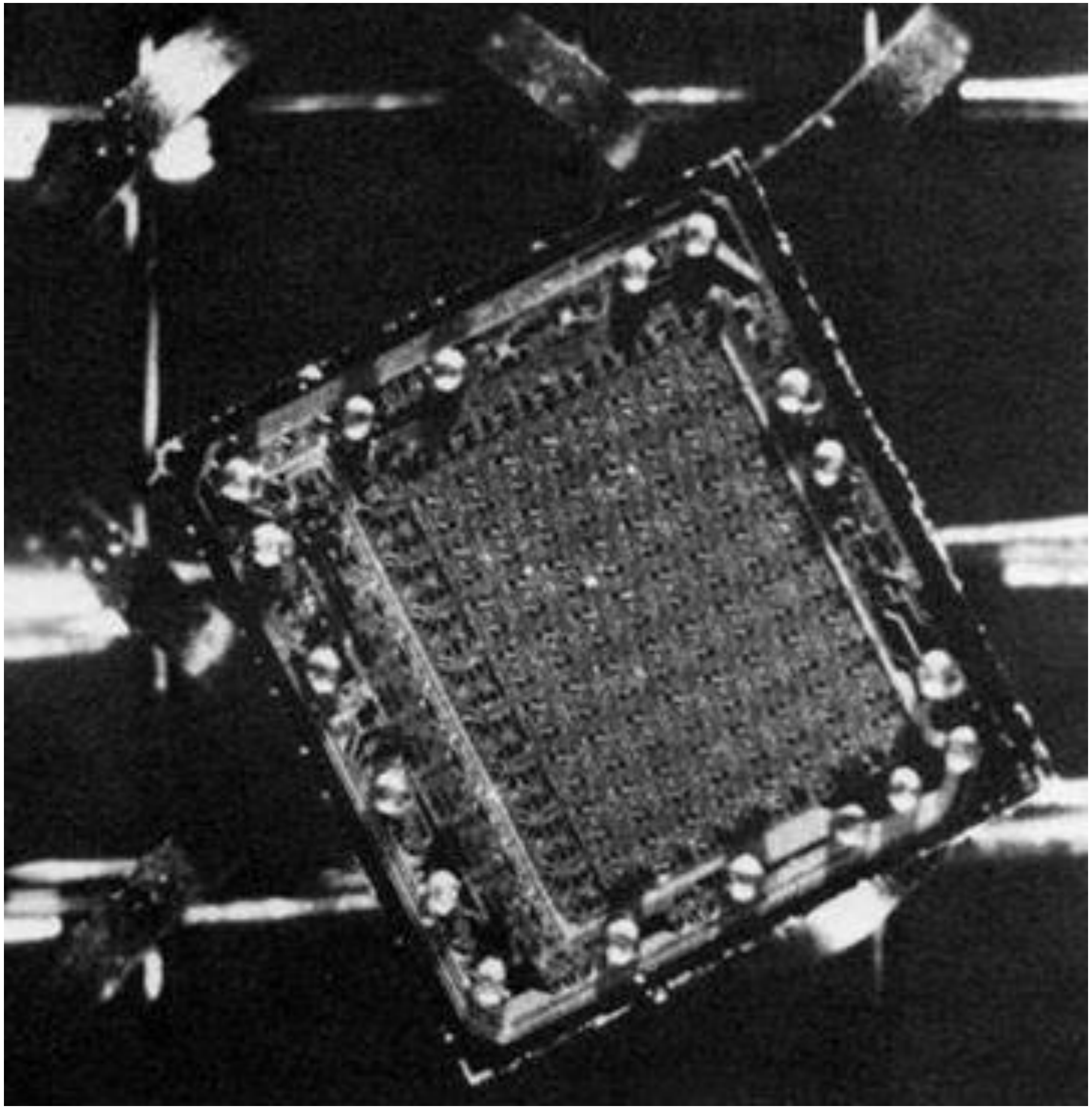
1,024 Bit Core Memory



SAGE Vector Graphic Terminal For IBM Memory Chip Design



128 Bit Memory Chip on Cores



VERMONT TECH

IBM 128 Bit Memory Chip Layout

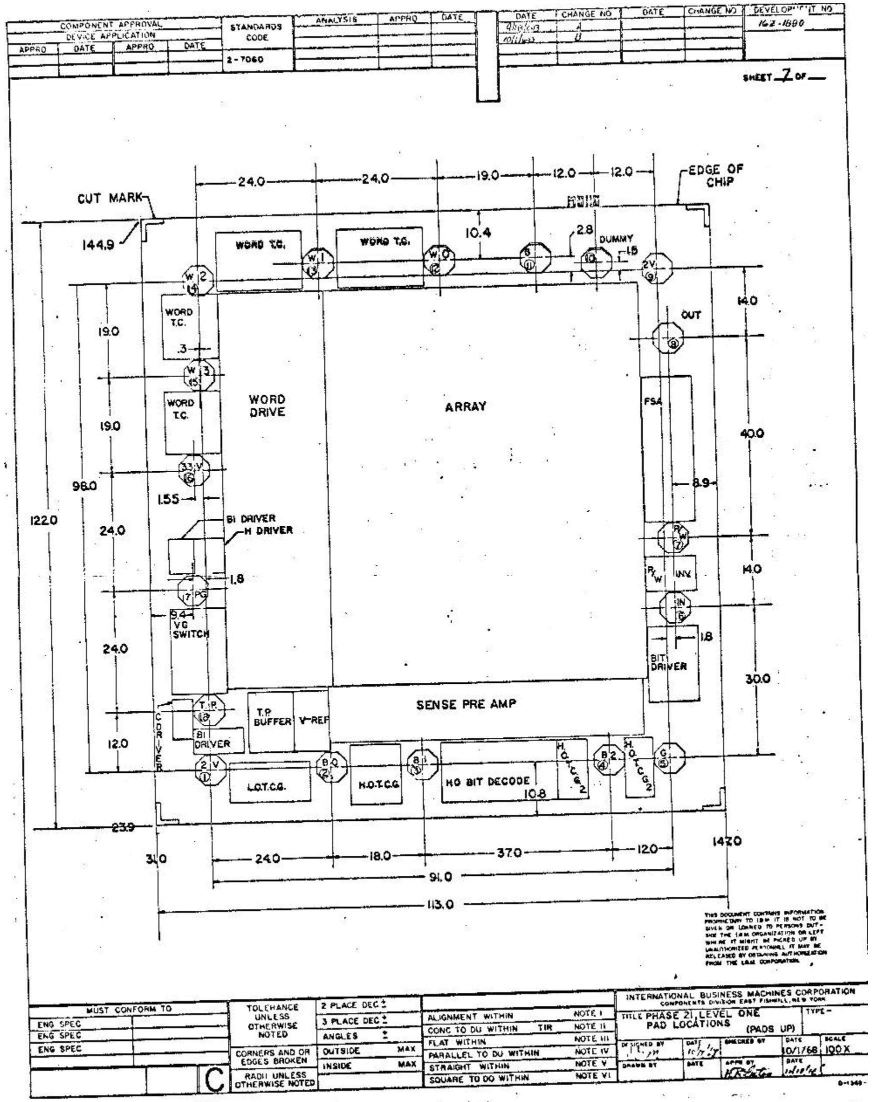
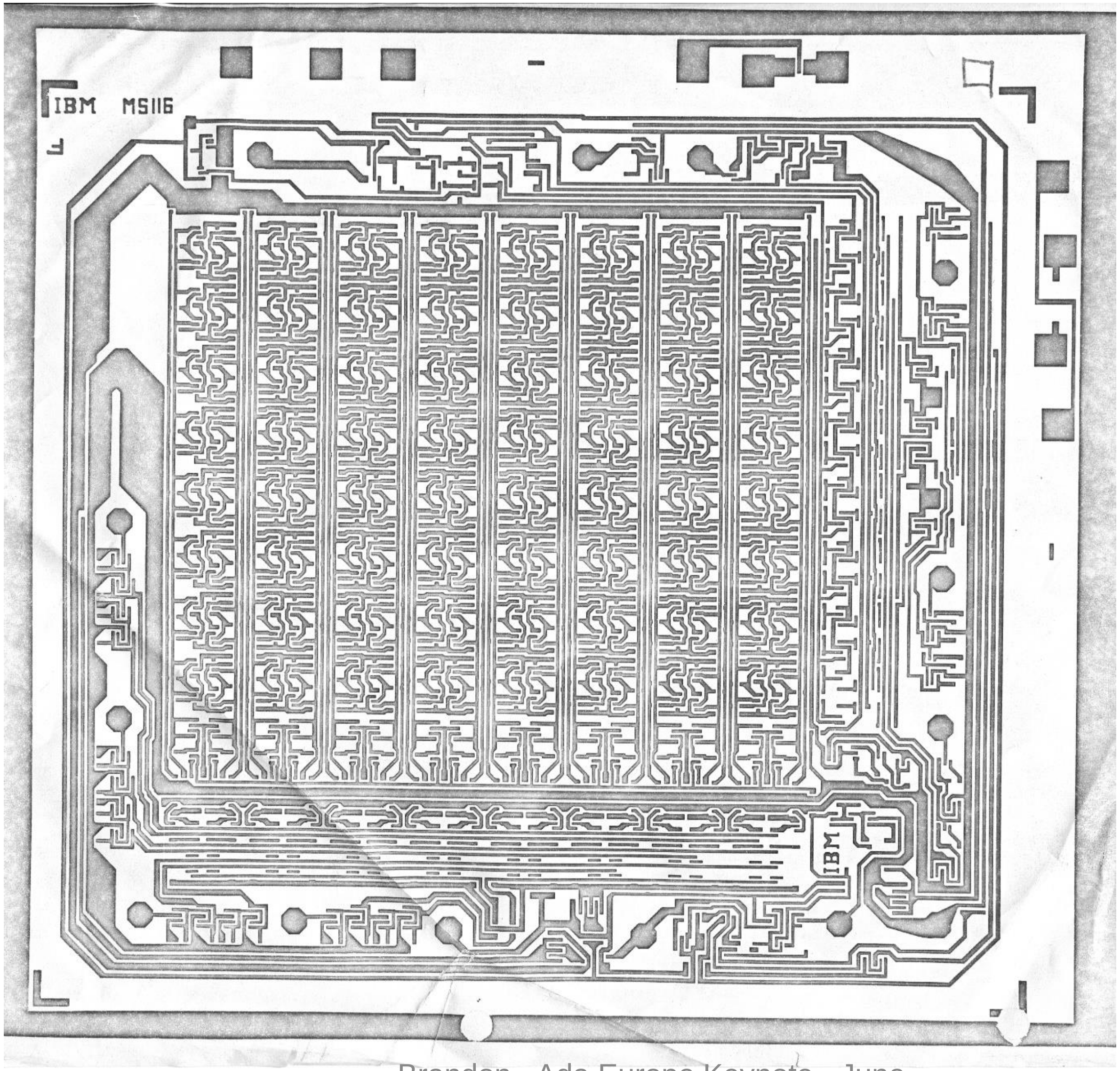


Fig. 44

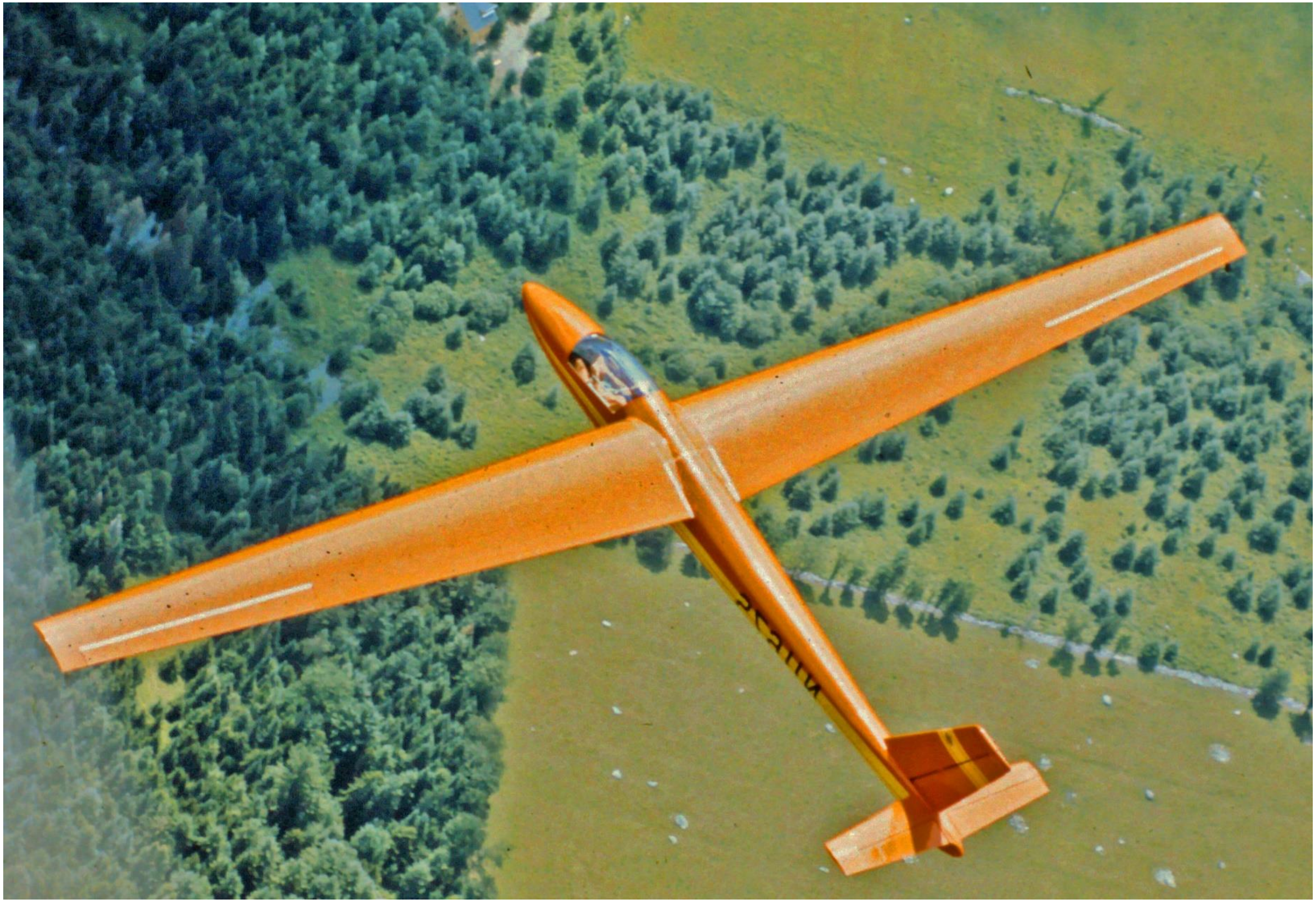
66

IBM 128 Bit Memory Chip Metallization



1/8 Inch
Square

My club Schweitzer 1-34 Sailplane



My old Cessna 182, 3 trips to the West Coast

Personal Aerodynamics Research

Teaching my 10 year old son, Jack to fly the PW-6 Sailplane

Intrepid, Twice America's Cup Winner
I did a little horizontal aerodynamics

Seagull Soaring Flight for my M.S.

VERMONT TECH

Bat Flight for my Ph.D.

VERMONT TECH

Inelastic moose collision experiment, 300 kg moose, 1,500 kg SAAB, 36 m/s

Gravitational potential energy experiment. Initiated with explosives.


White River Valley Herald

HOME & GARDEN See "B" Section

Randolph, Vermont April 9, 1981 26 Pages 35 cents per copy

Garden Center Underway In R.C.

Child Seat Law Supported

No Action Taken On One-Manager

Photos are by Chuck Fulkerson

And Down It Goes

A landmark fell Saturday in South Randolph, as demolition expert Carl Brandon tipped over an old silo at David Jarmy's home, formerly the McKinney farm. The barn had burned down 25 years ago, cracking some of the silo tiles, and the remaining silo was an eyesore, Jarmy decided.

Silo heads down—in the right direction ...

Until finally ...

There's not much left.

Demolition expert Carl Brandon surveys his handiwork.

Two Charged with Break-in At Gaysville Post Office

Should D.C. Have Right To Vote?

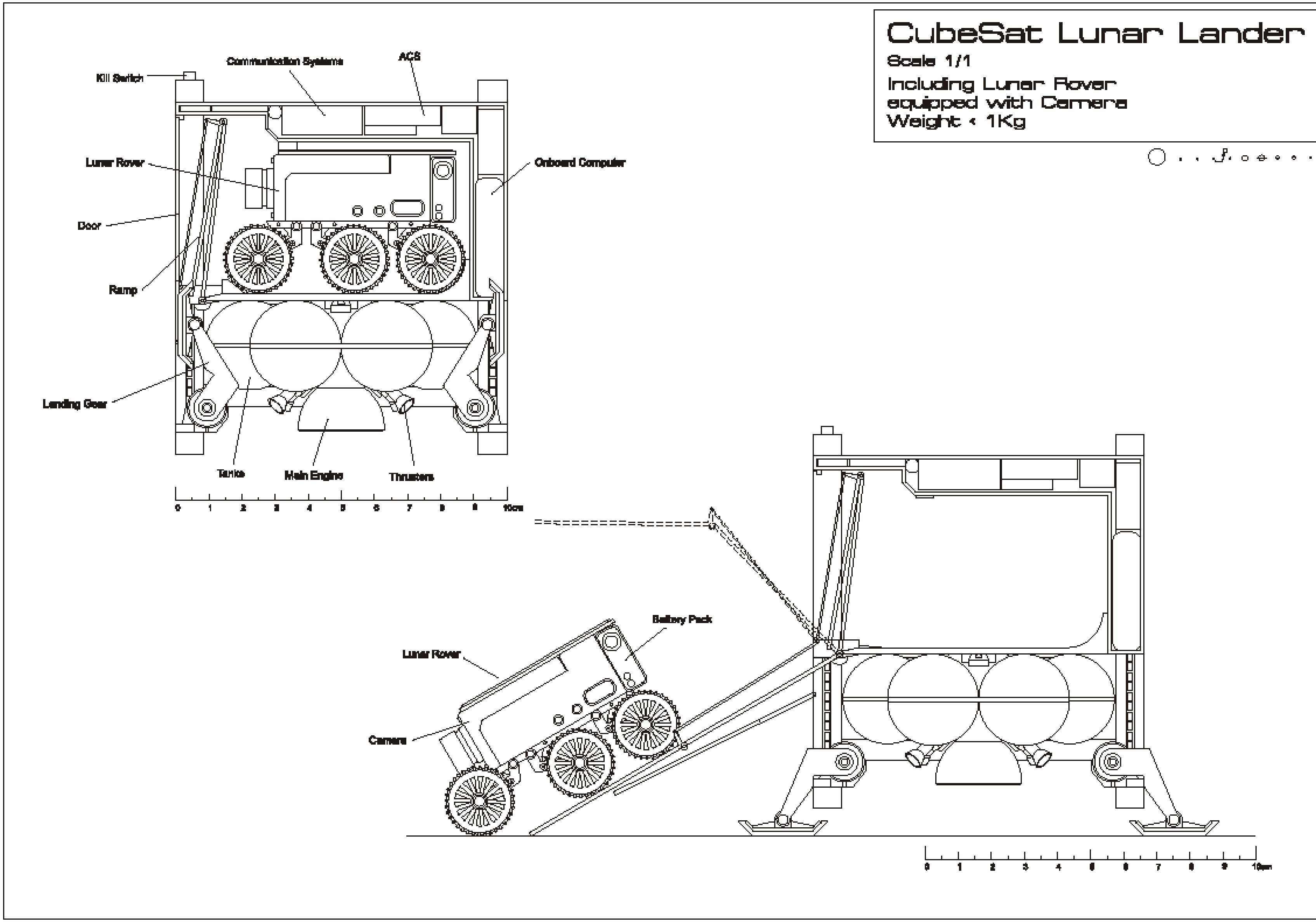
Bomb Scare Empties RUHS

Inside The Herald

My 8 year old son, Jack, starting Up RPI's nuclear reactor

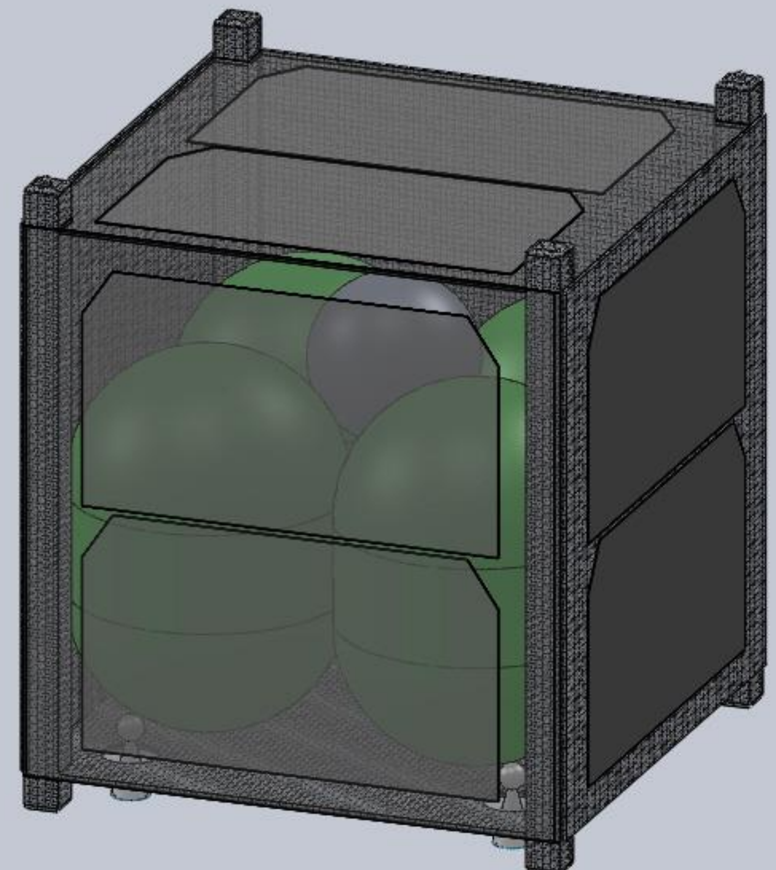
Aalborg Universitet

VERMONT TECH



VERMONT TECH

Monopropellant 1U Lunar Lander CubeSat

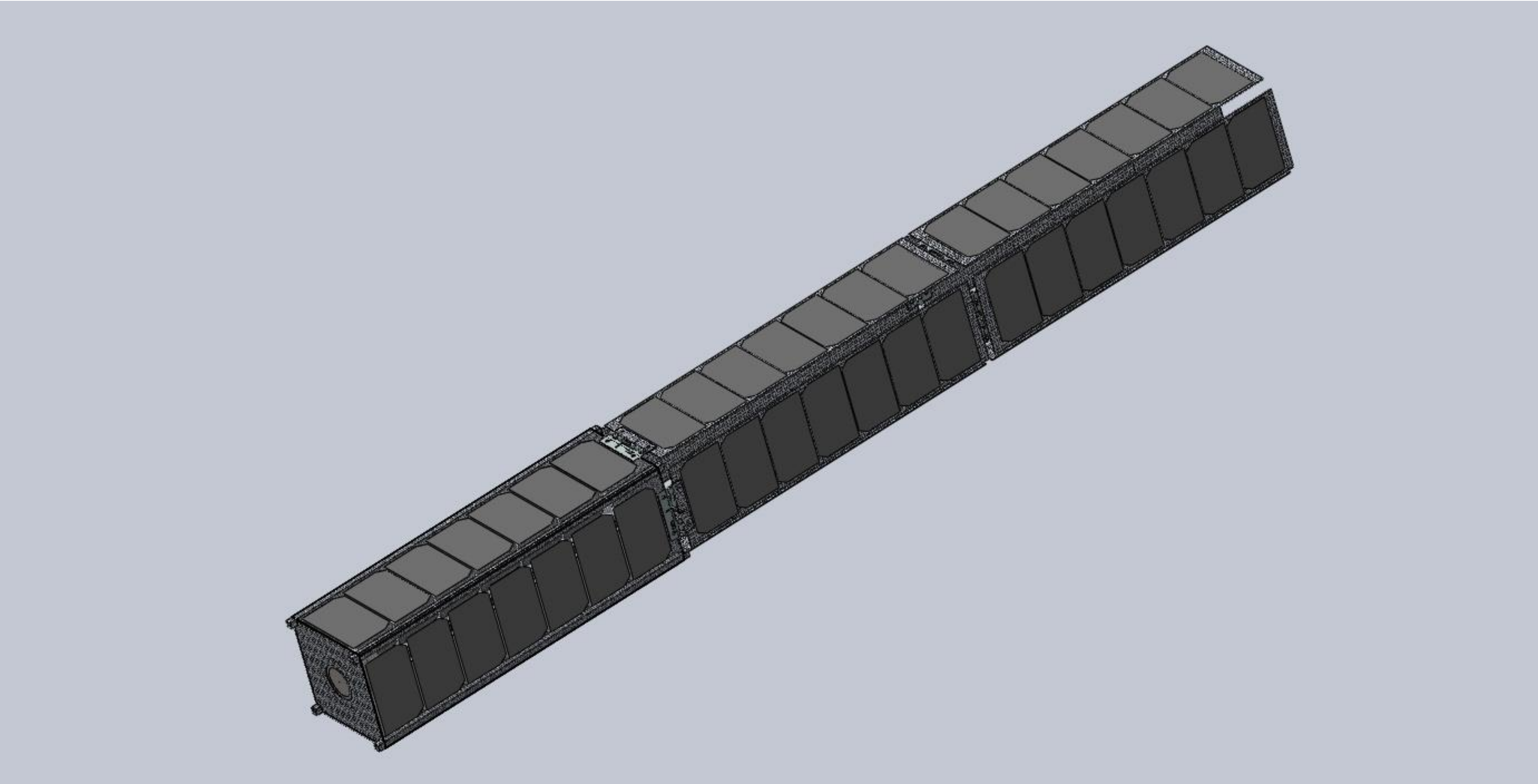


I was an invited speaker (of the Moon Society) to the Space Development Conference, along with Scott Carpenter, John Glenn and Buzz Aldrin. I spoke about sending CubeSats to the Moon.

Monopropellant hydroxyl-ammonium nitrate Thruster, Busek BGT-X5, 0.5N, 225s ISP

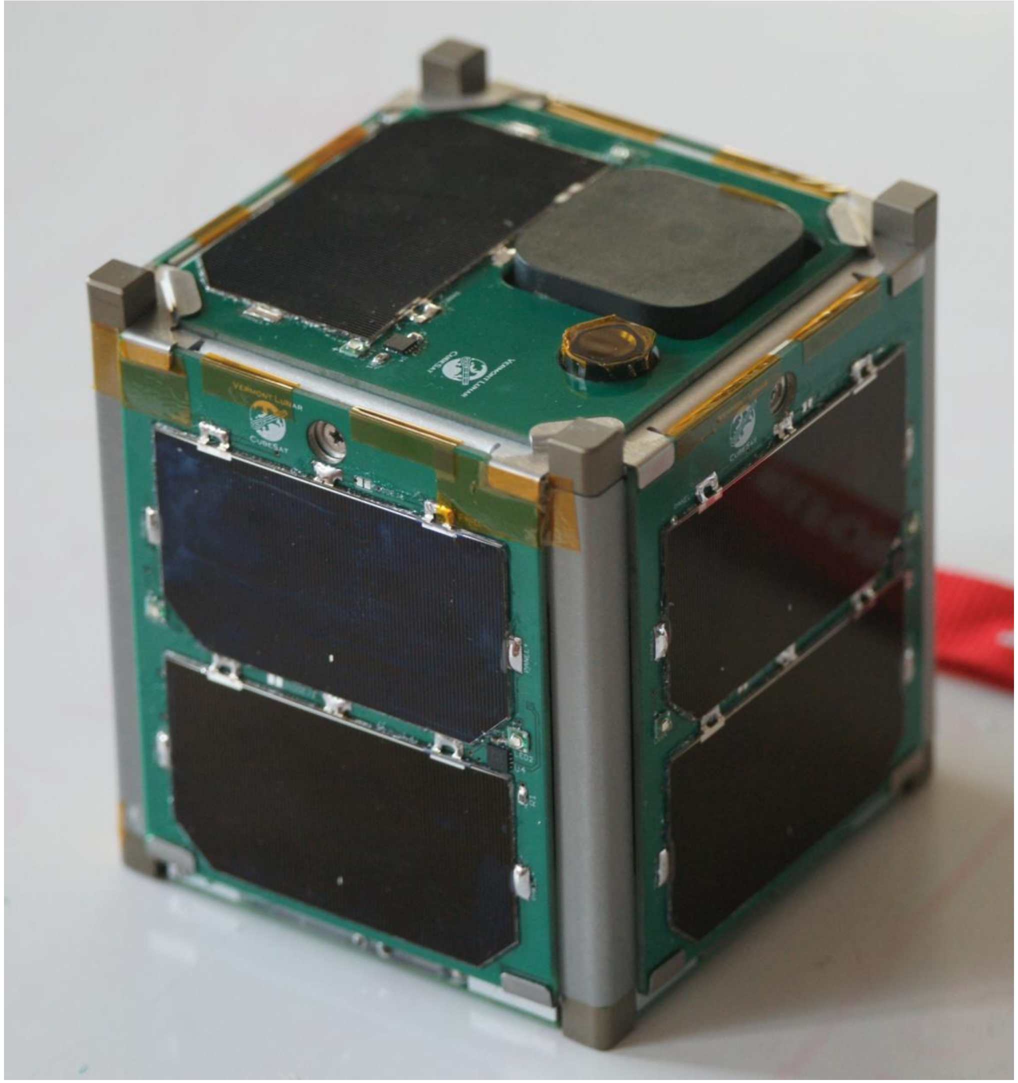
Monopropellant 2U Booster CubeSat

3U Ion Drive CubeSat with PV panels



Vermont Lunar CubeSat

VERMONT TECH



Vermont Lunar CubeSat (10 cm cube, 1 kg)

Vermont Lunar CubeSat

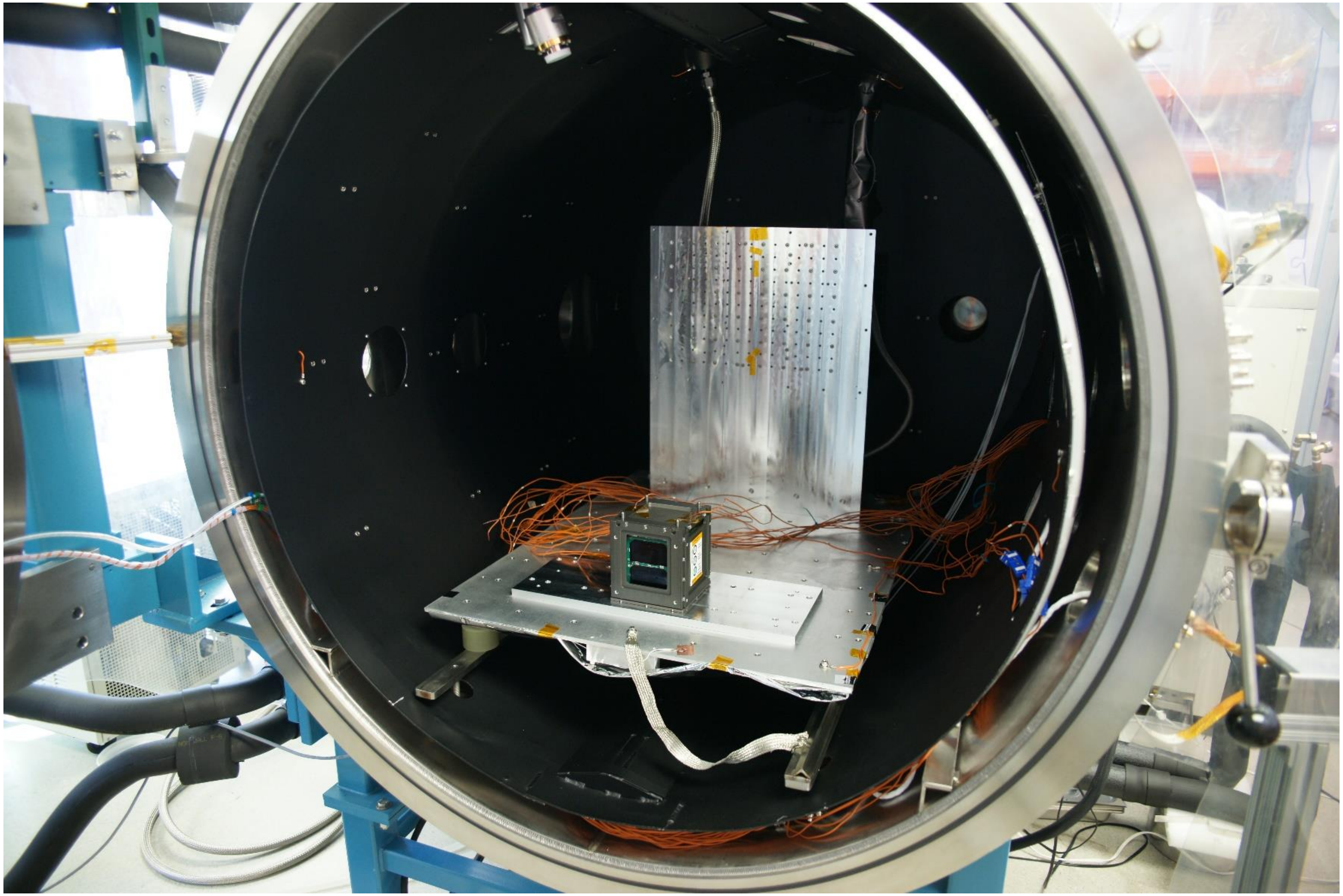
VERMONT TECH

It worked until our reentry on November 21, 2015:

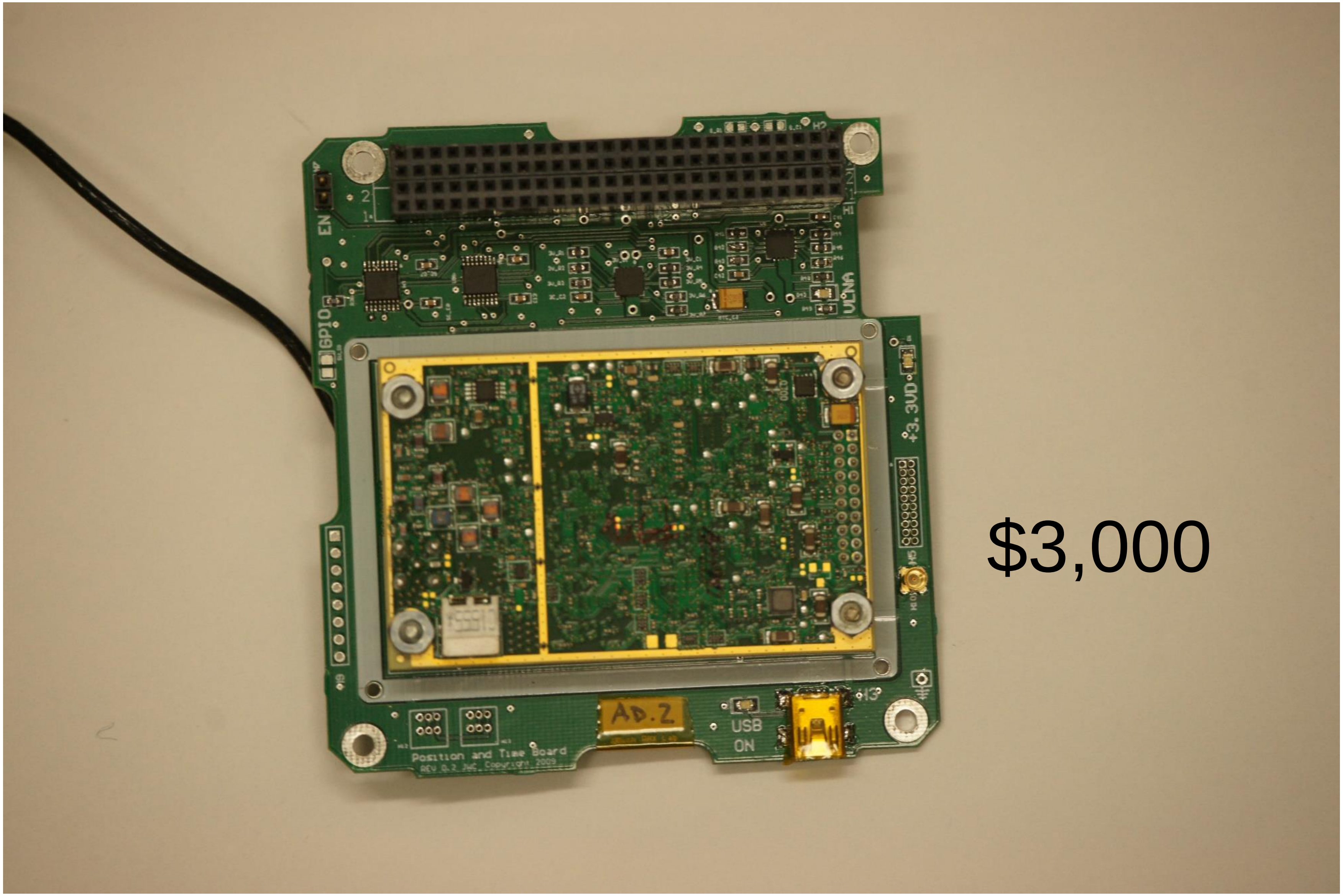
- We completed 11,071 orbits.
- We travelled about 472,000,000 km (293,000,000 miles), equivalent to over 3/4 the distance to Jupiter.
- Our single-unit CubeSat was launched as part of NASA's ELaNa IV on an Air Force ORS-3 Minotaur 1 flight November 19, 2013 to a 500 km altitude, 40.5^{o} inclination orbit and remained in orbit until November 21, 2016. **It is the only one of the 12 ELaNa IV university CubeSats that operated until reentry, the last one quit 19 months earlier.**
- We communicated with it the day before reentry over the Pacific
- We were the first university satellite from New England or NY
- We were the only successful university satellite on the east coast until this year
- **Follow our project at cubesatlab.org**

650 kW Vibration Tester, 100 g's at 4 cm amplitude, BAE Systems

Vacuum thermal bakeout
6 hours at 60^{o}C, UNH



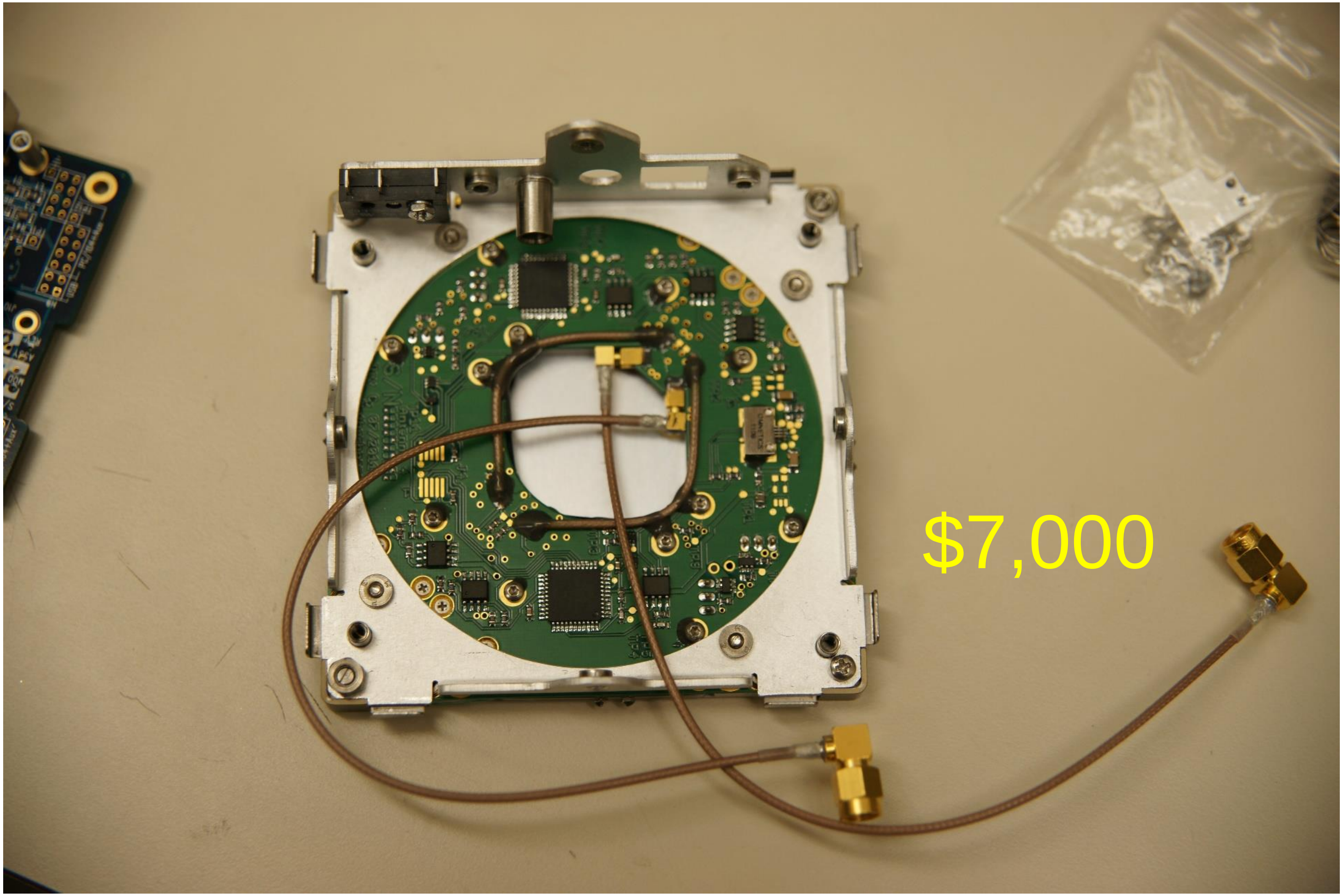
GPS Board



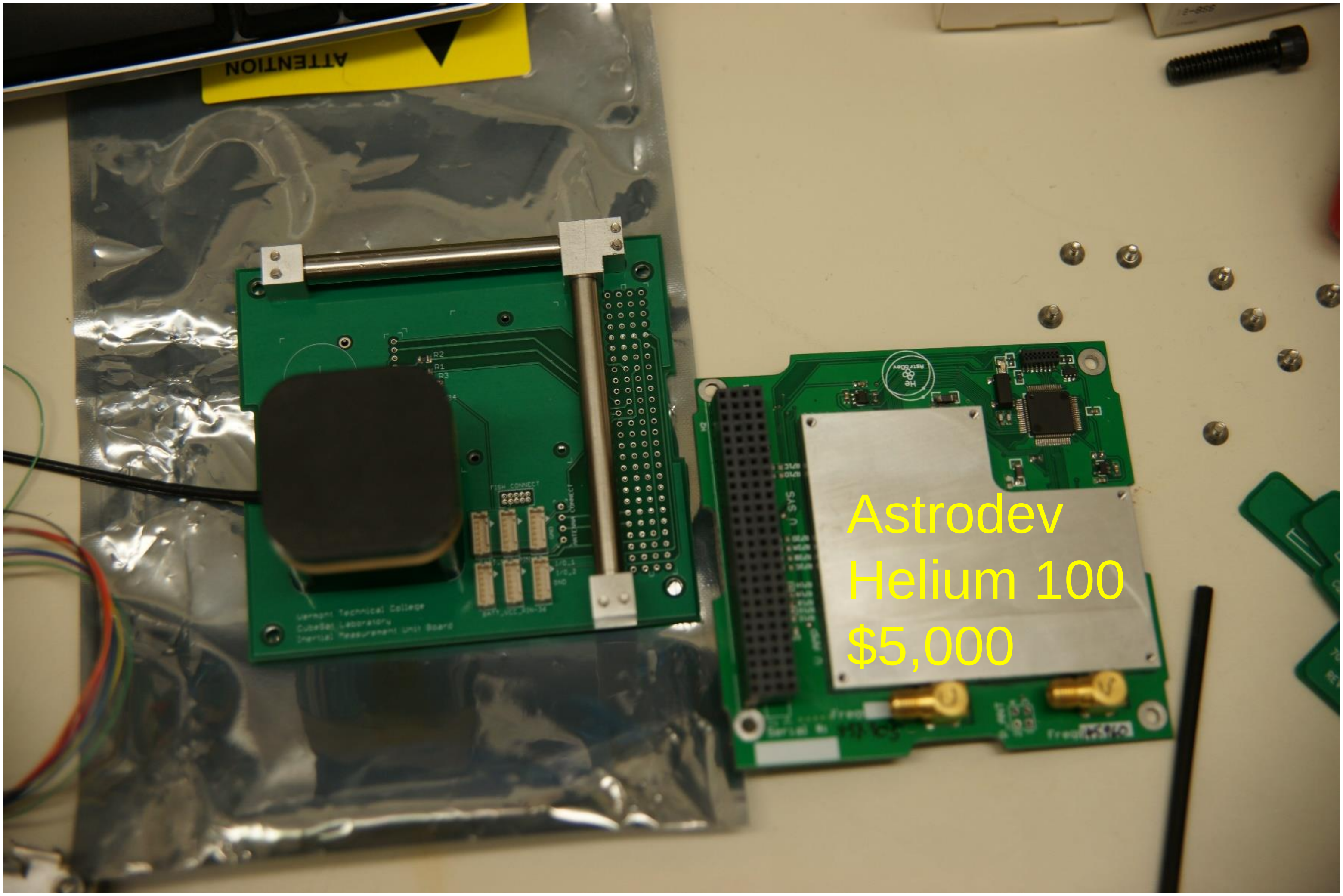
$3,000

Deployable 2m & 70cm crossed dipoles.

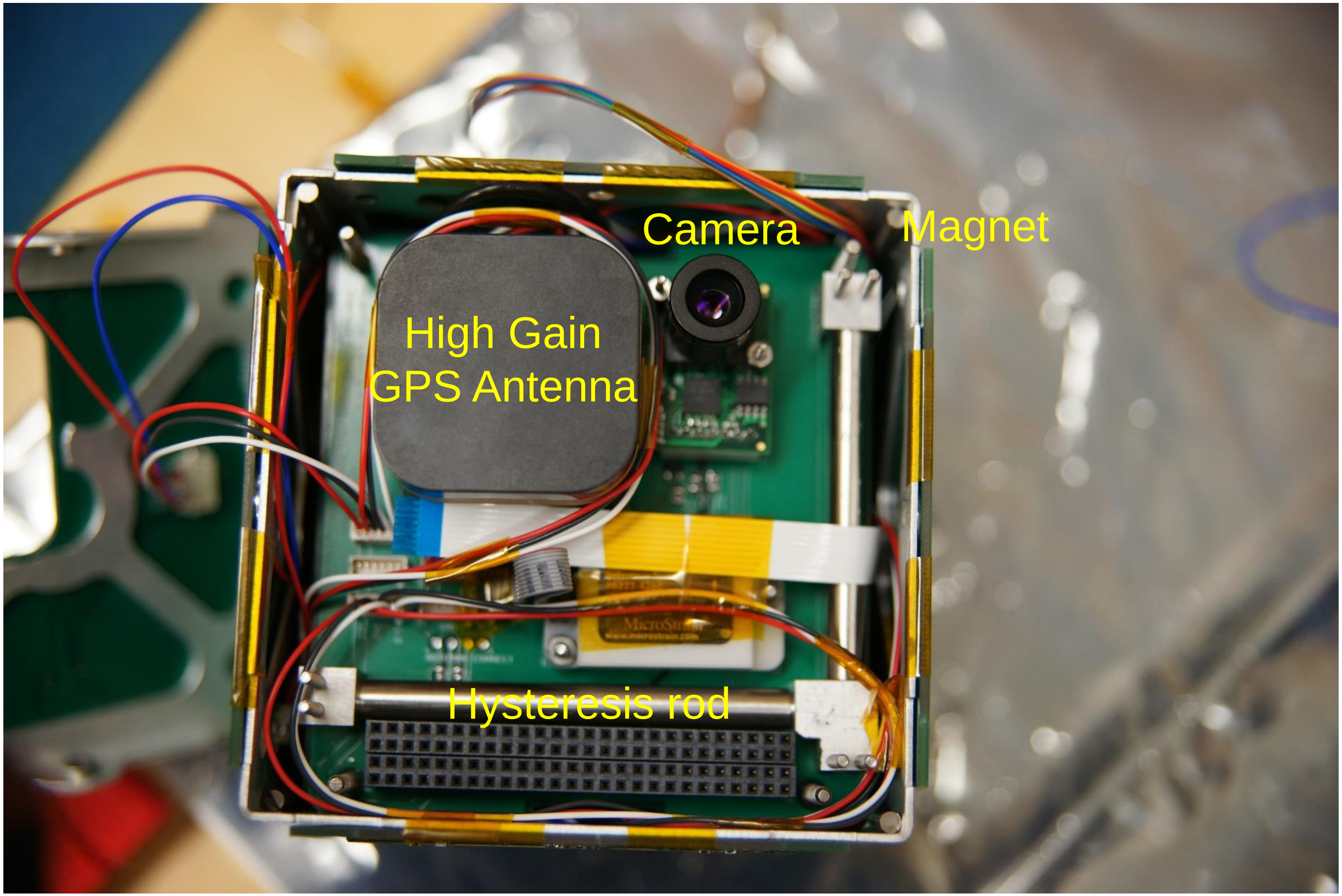
ISIS (Netherlands, not Syria)



GPS antenna, hysteresis rods (left) 2m receiver/70cm transmitter (right)



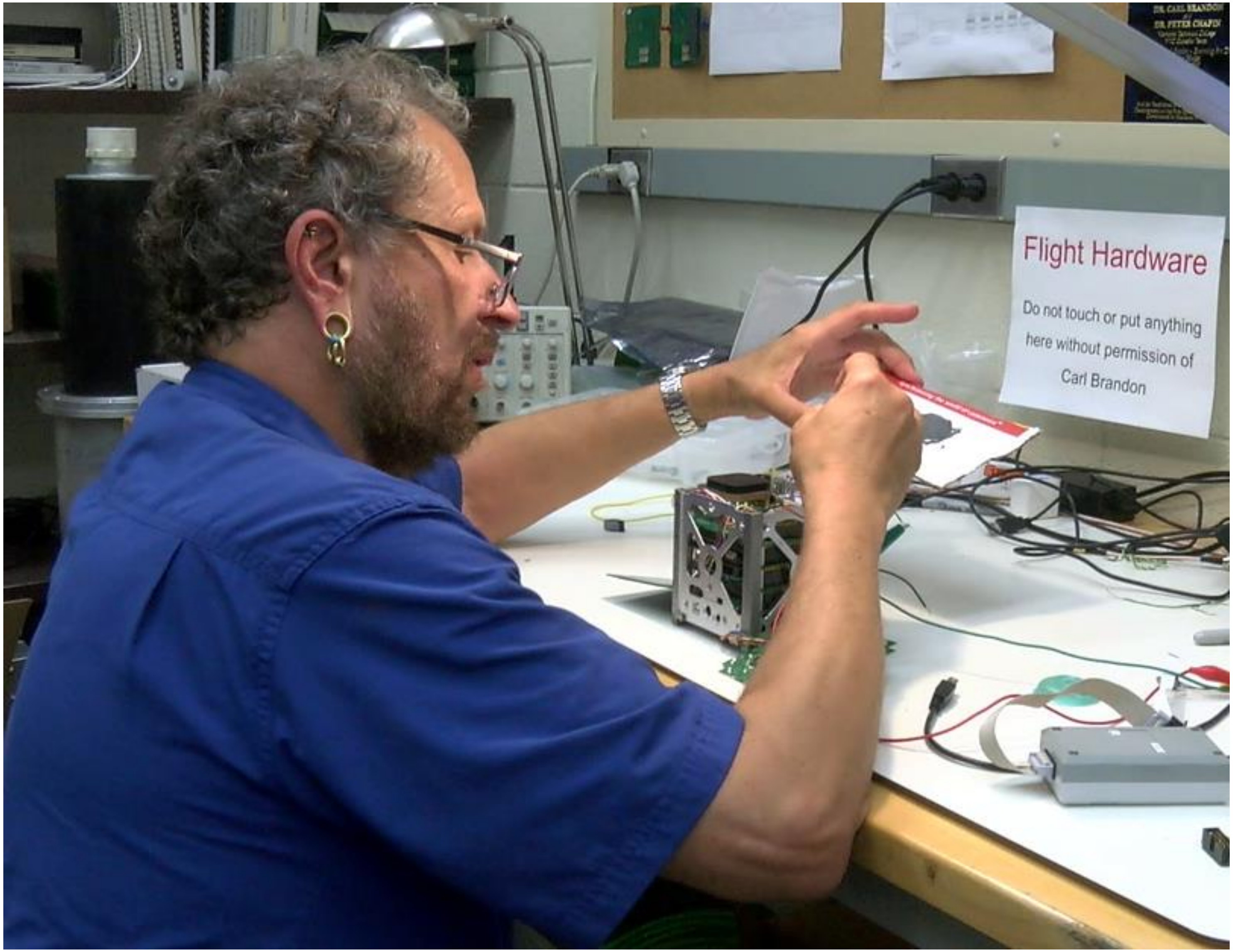
GPS antenna, hysteresis rods, inertial measurement unit & camera.



Camera, inertial measurement unit.

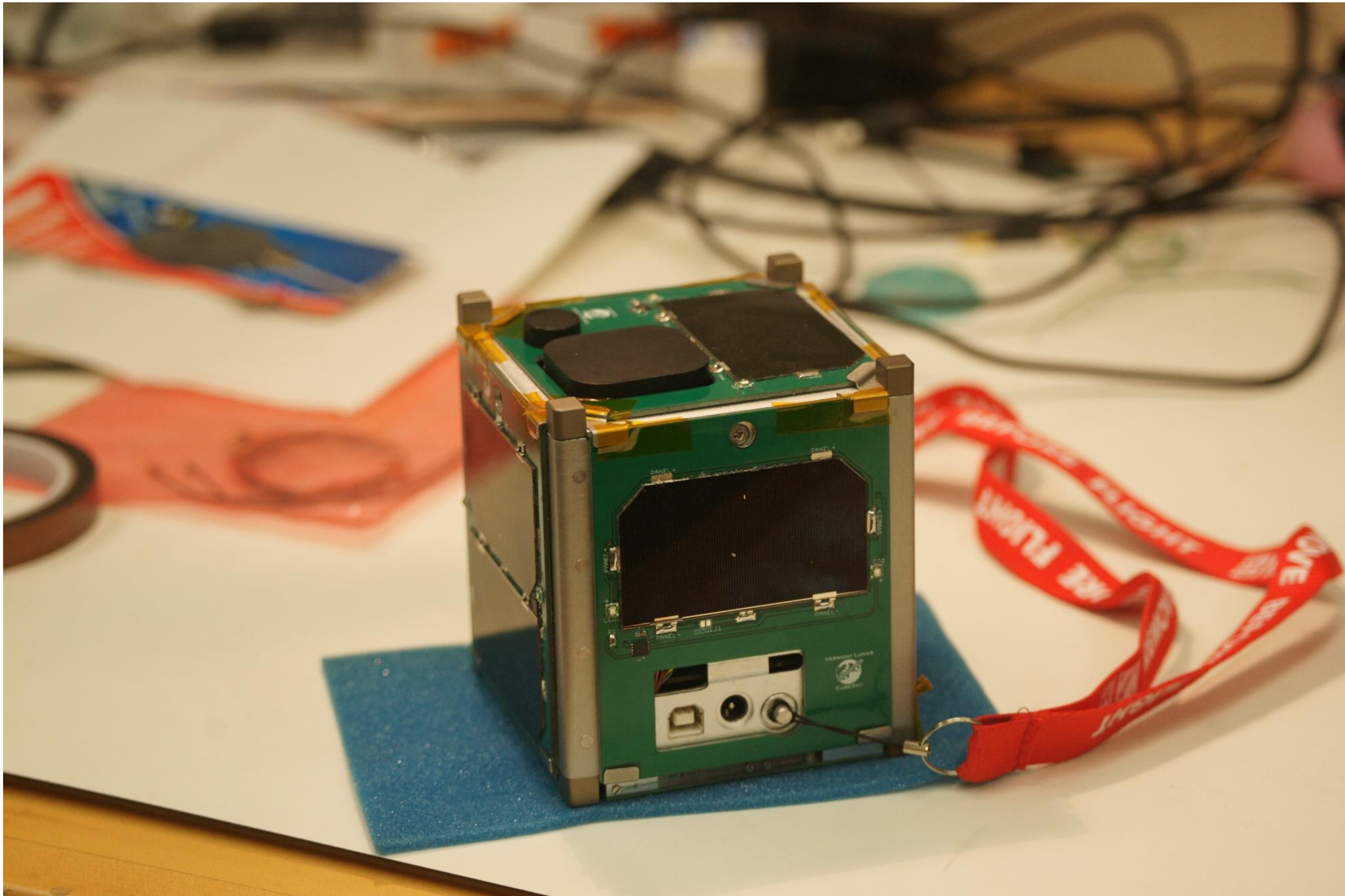
MAD RIVER GLEN
SKI IT IF YOU CAN

Assembling the CubeSat

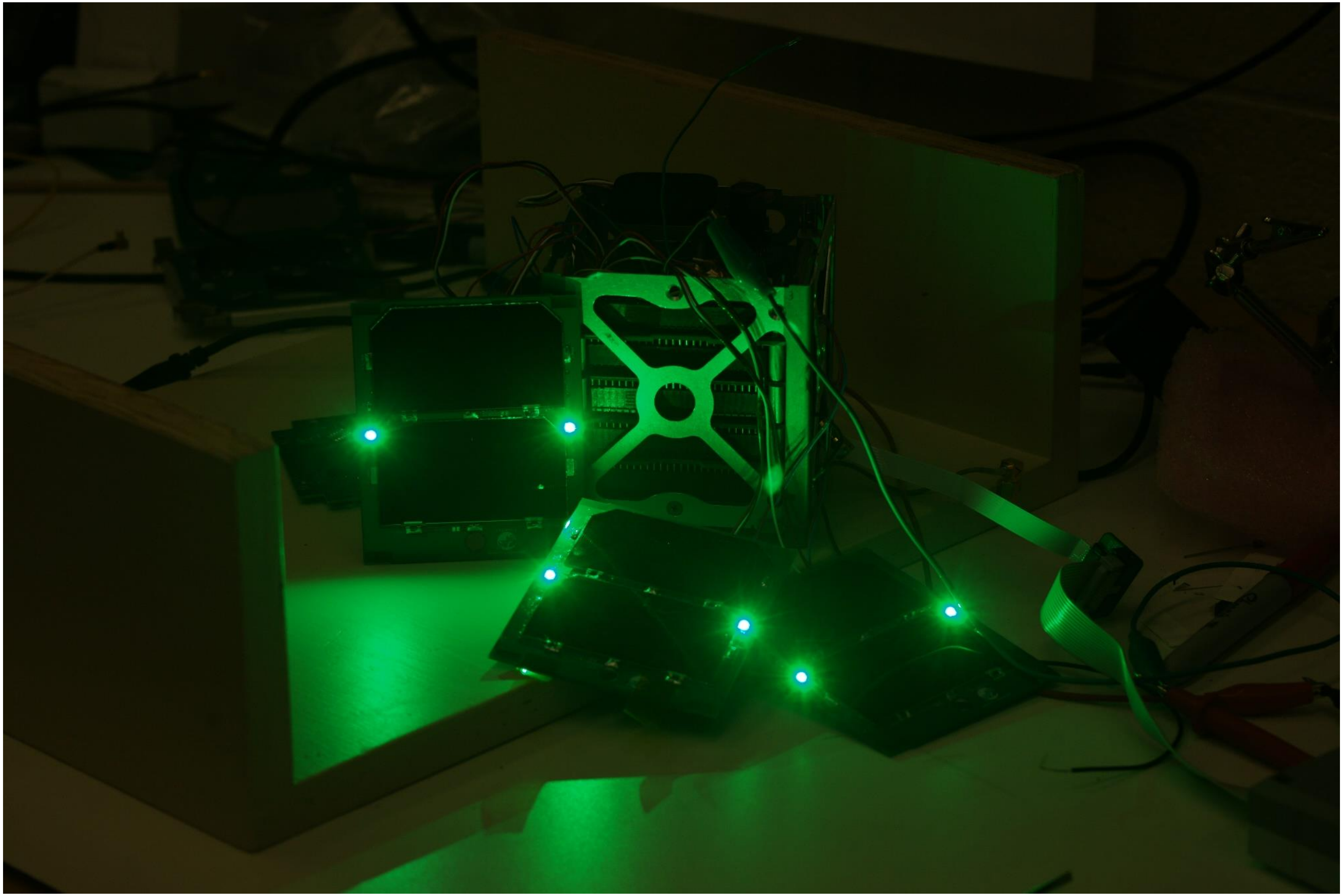


VERMONT TECH

Assembled Vermont Lunar CubeSat



Testing the LEDs



VERMONT TECH

ELaNa IV Launch Minotaur 1 – Wallops Island November 19, 2013, 8:15 PM

I am with my two software students, Dan and India, and my son, Jack

ELaNa IV Launch Minotaur 1 – Wallops Island November 19, 2013, 8:15 PM

First two stages are Minuteman II, third and fourth stages are Pegasus second and third stages, 19.2 m high, 36,200 kg

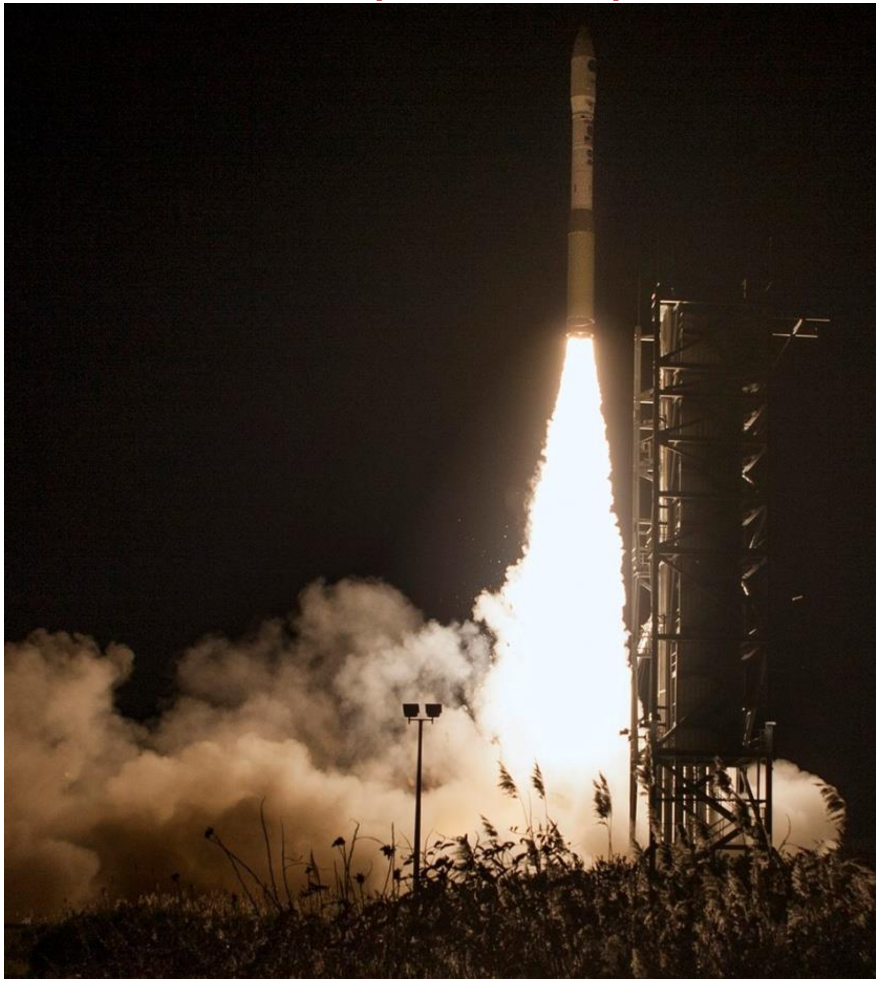
Minotaur 1

Minuteman II

Pegasus air dropped launch vehicle

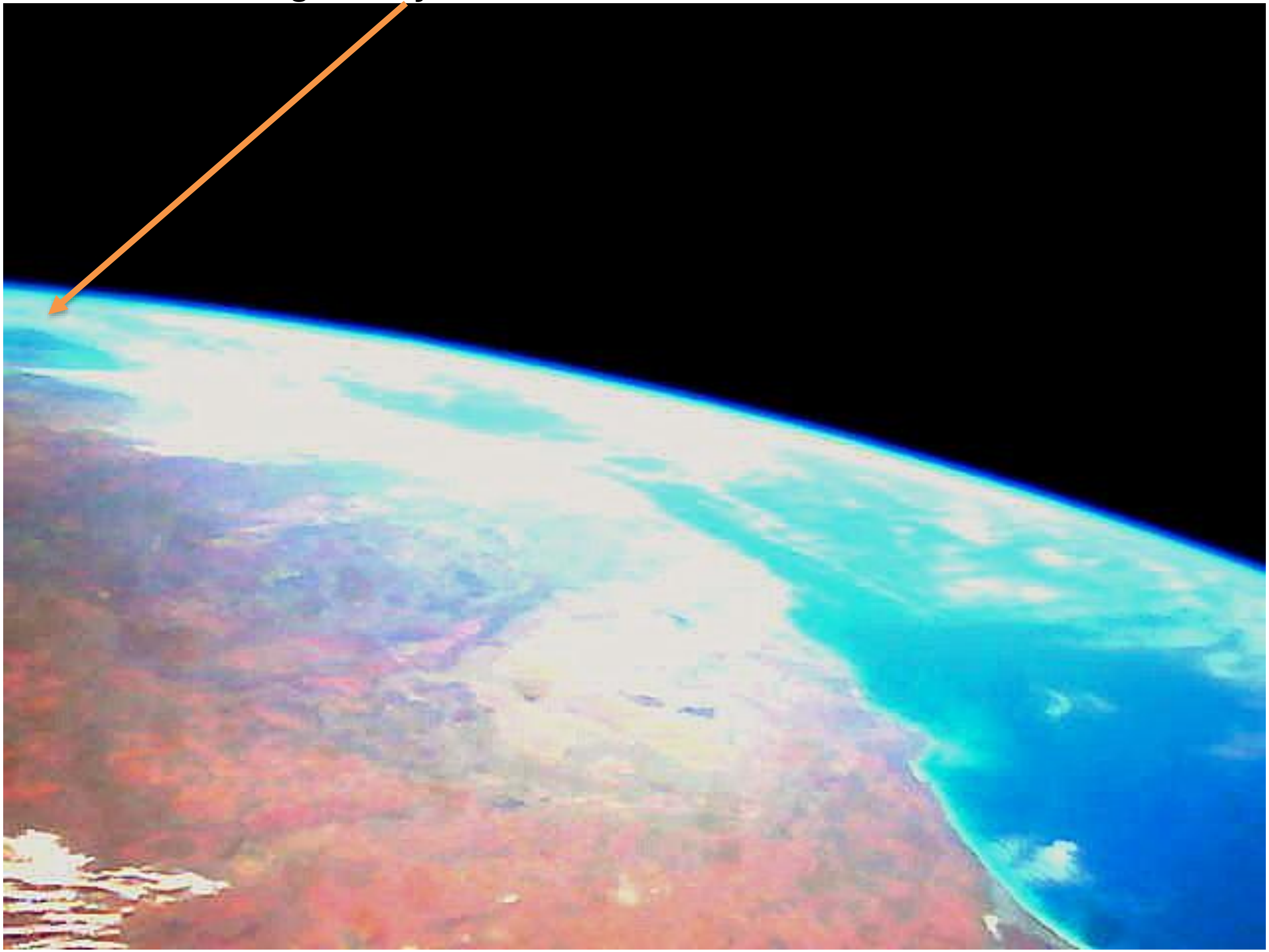
First two stages are Minuteman II, third and fourth stages are Pegasus second and third stages

ELaNa IV Launch Minotaur 1 – Wallops Island November 19, 2013, 8:15 PM



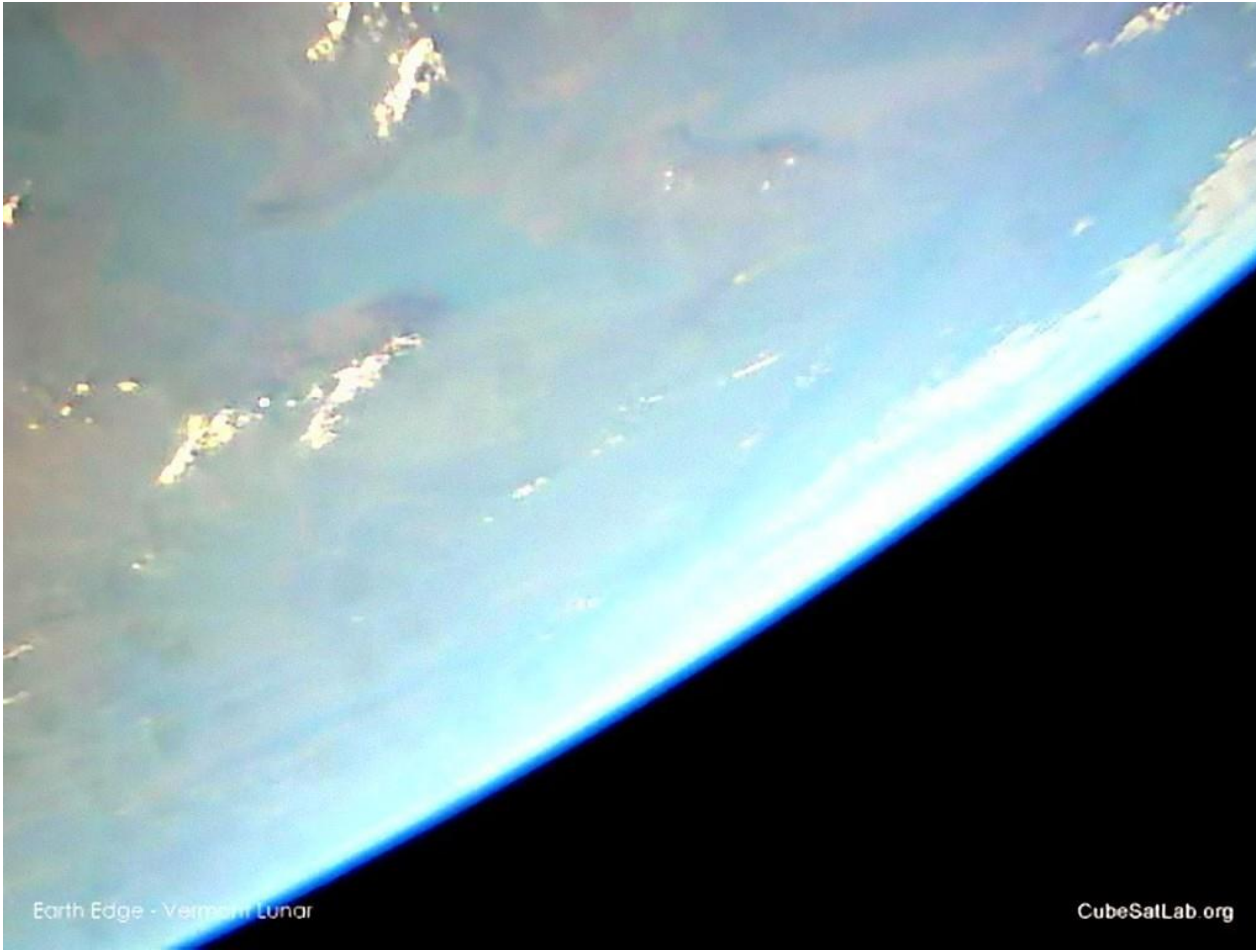
First two stages are Minuteman II, third and fourth stages are Pegasus second and third stages

Missing Malaysian airliner



Our first picture of Earth, The North coast of Western Australia

Clouds over the ocean, June 2015, 19 months after launch.

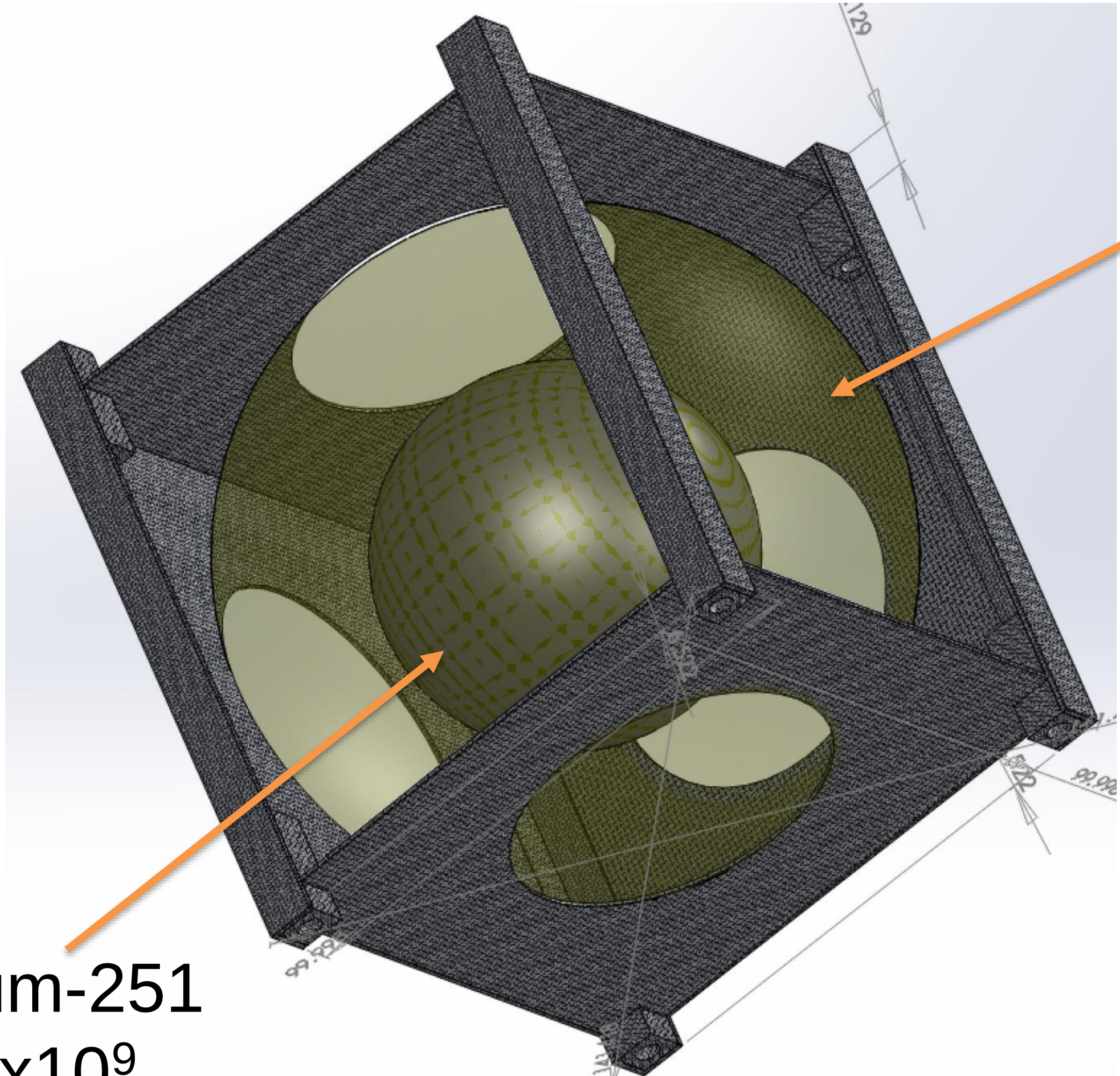

Earth Edge -
Lunar
CubeSatLab.org

Africa

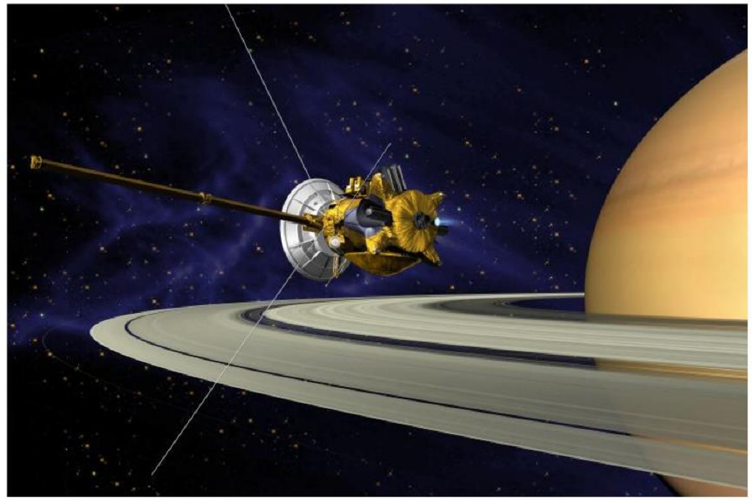

See-through Earth - Vermont Lunar
CubeSatLab.org


CubeSatLab.org

Large Area Orbital Debris Mitigation



Just Another Critical Project powered by Ada



National Aeronautics and Space Administration via Wikipedia

Cassini-Huygens
NASA-ESA Mission to Saturn

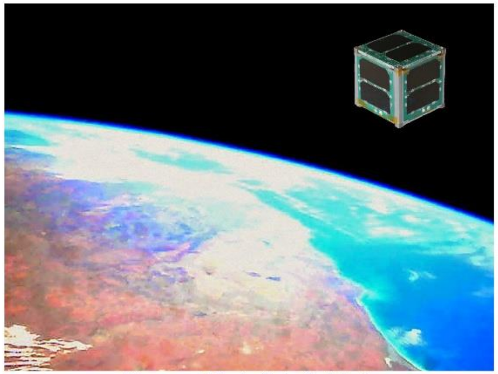
WHERE THE SOFTWARE REALLY HAS TO WORK

ACM Special Interest Group
on the Ada Programming Language
www.sigada.org

M.B. Feldman

Just Another Critical Project powered by Ada

WHERE THE SOFTWARE REALLY HAS TO WORK



Vermont Tech CubeSat Lab
CubeSat Image Taken over Australia
(CubeSat Itself Photoshopped In)

Vermont Technical College CubeSat

Launched from Wallops Island, VA, Nov. 19, 2013

This is the only fully successful university CubeSat of 12 launched together, and the only one in Ada.

ACM Special Interest Group
on the Ada Programming Language
www.sigada.org

M.B. Feldman

Why We Use SPARK/Ada

ELaNa IV lessons for CubeSat software:

- NASA's 2010 CubeSat Launch Initiative (ELaNa)
- Our project was in the first group selected for launch
- Our single-unit CubeSat was launched as part of NASA's ELaNa IV on an Air Force ORS-3 Minotaur 1 flight November 19, 2013 to a 500 km altitude, 40.5º inclination orbit and remained in orbit until reentry over the central Pacific Ocean, November 21, 2016, **after two years and two days. Eight others were never heard from, two had partial contact for a few days, and one worked for 4 months.**
- The Vermont Lunar CubeSat tested components of a Lunar navigation system in Low Earth Orbit

Vermont Lunar CubeSat *SPARK 2005* software

- 5991 lines of code
- 4095 lines of comments (2843 are SPARK annotations)
- A total of 10,086 lines (not including blank lines)
- The Examiner generated 4542 verification conditions
- All but 102 were proved automatically (98%)
- We attempted to prove the program free of runtime errors
- Which allowed us to suppress all checks
- The C portion consisted of 2239 lines (including blank lines), mostly SD card driver we purchased
- Additional provers in SPARK 2014 would improve this

Software Development Comments for our first CubeSat

- SPARK caught errors as we refactored the software as we developed greater understanding of the hardware
- SPARK helped the discipline of the software during turnover as some students graduated and were replaced
- Although we did not have a formal development process, without SPARK we probably would not have completed the project with the limited personnel resources and tight time constraint
- Although the CubeSat is limited to 1.3kg, the paperwork is unlimited 

Four aerospace software failures that would have been prevented with SPARK/Ada:

- Mars Science Laboratory Sol-200 Memory Anomaly
- Ariane 5 initial flight failure
- Boeing 787 generator control computer shutdown
- Boeing 787 avionics reset

Language Comparison

UK Ministry of Defense C-130J software study:
The anomalies per 1,000 lines of code (average):

- for C was 97
- for Ada 95 was 25
- for SPARK/Ada 95 was 4

Newer Tokeneer project (for NSA)

- For SPARK/Ada 2005 was 0.4

Productivity of 38 lines of code per programmer day (about what our student achieved, also), compared with 10 to 12 lines of code when using C.

We are now using the even newer SPARK/Ada 2014

Language Comparison
Real world data

- **If your student programmers do not know SPARK/Ada, it takes about two weeks to become productive**
- **SPARK/Ada productivity of 38 lines of code per programmer day, compared with 10 to 12 lines of code when using C**
- **After three weeks, the new SPARK/Ada programmer has caught up with the C programmer**
- **For a 10,000 line program, the SPARK/Ada programmer would finish in 1.09 years (4 errors)**
- **For a 10,000 line program, the C programmer would finish in 3.33 years (970 errors)**

Mars Science Laboratory

Sol-200 Memory Anomaly

- Six months after landing on Mars, uncorrectable errors in the NAND flash memory led to an inability of the Mars Science Laboratory (MSL) prime computer to turn off for its normal recharge session.

- This potentially fatal error was apparently due to two pieces of its C software having pointers which pointed to the same memory. Curiosity has about 3.5 MLOC written in C. (One would expect about 35,000 errors, they have corrected about 1,500 so far)

- SPARK/Ada would have prevented this almost fatal error in a 2.5 billion dollar spacecraft.

Ariane 5 initial flight failure:

Good

Bad, 37 seconds later

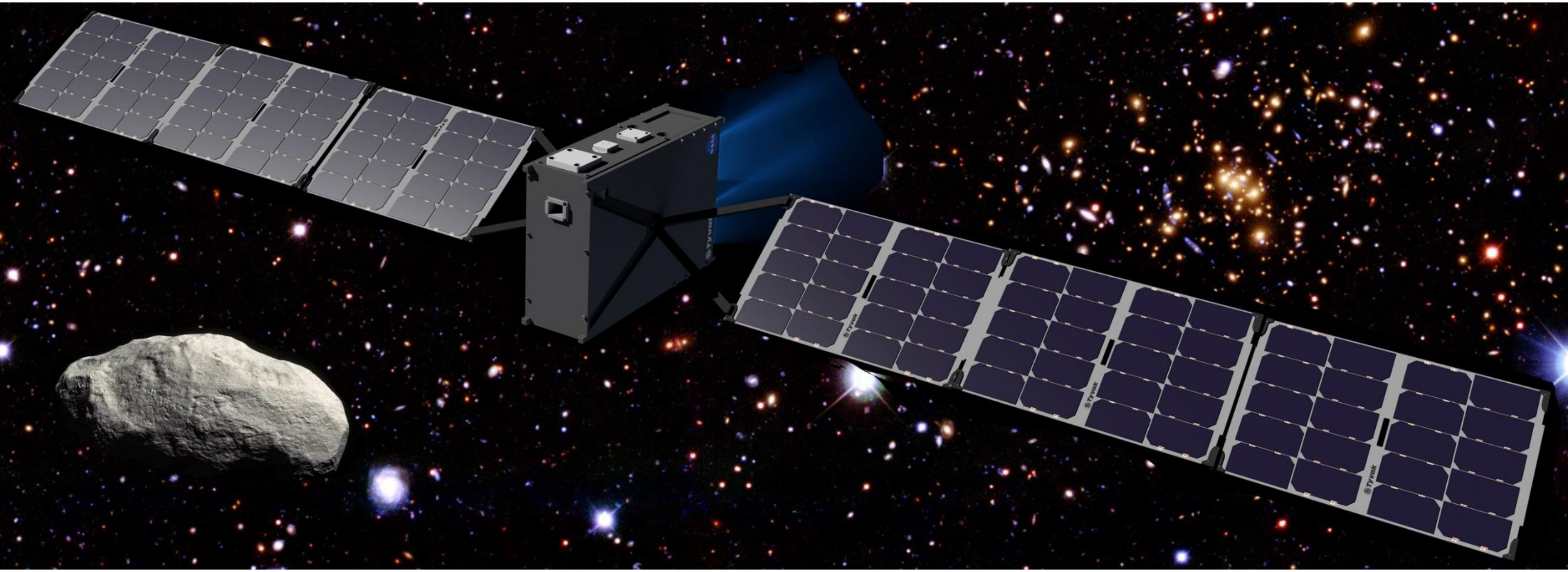
Ariane 5 initial flight failure:

- Software reused from Ariane 4, written in Ada
- The greater horizontal acceleration caused a data conversion from a 64-bit floating point number to a 16-bit signed integer value to overflow and cause a hardware exception.
- “Efficiency” considerations had omitted range checks for this particular variable, though conversions of other variables in the code were protected.
- The exception halted the reference platforms, resulting in the destruction of the flight.
- Financial loss over $500,000,000.
- SPARK/Ada would have prevented this failure

Boeing 787 generator control computer:

- There are two generators for each of two engines, each with its own control computer programmed in Ada (Airbus Rolls Royce controllers are in SPARK)
- The computer keeps count of power on time in **centiseconds** (used by stopwatches) in a 32 bit register
- Just after 8 months elapses, the register overflows
- Each computer goes into “**safe**” mode shutting down its generator resulting in a complete power failure, causing loss of control of the aircraft
- The FAA Airworthiness Directive says to shut off the power before 8 months as the solution
- There is now a second 787 reset problem
- SPARK/Ada would have prevented both

Deep Space Application



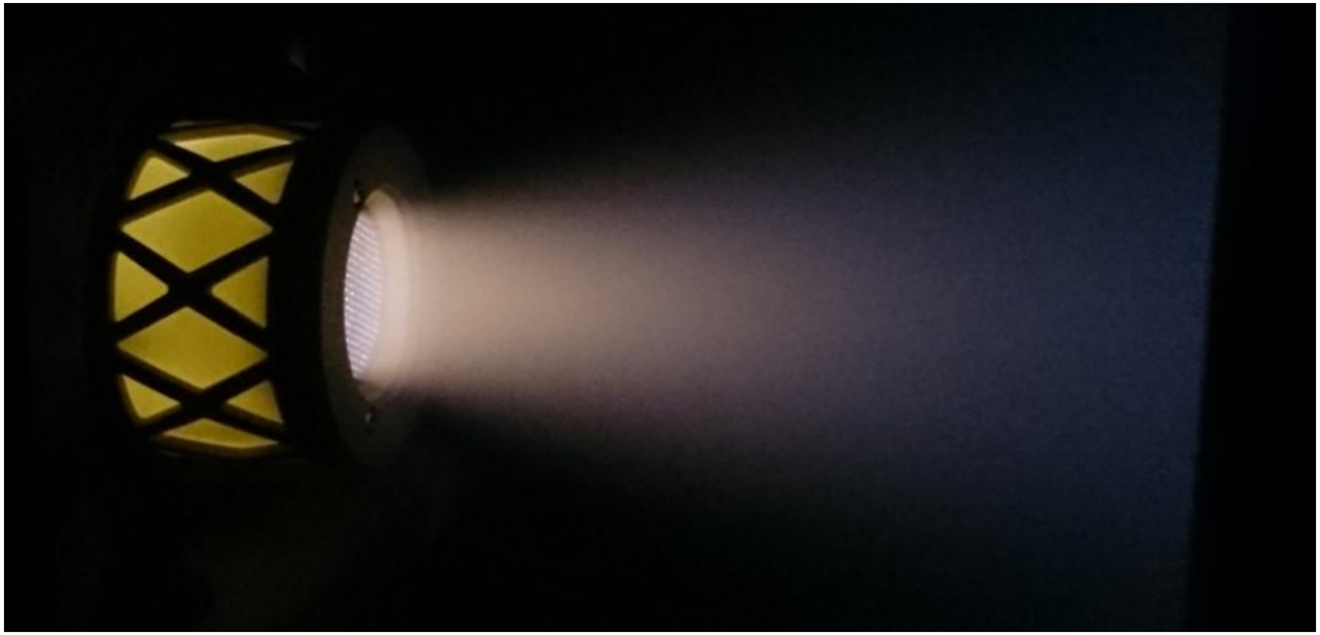
6U CubeSat with ion thruster

Deep space mission

Deep Space Application

I was hoping for a ride, but at 10cm x 20cm x 30cm, found I wouldn't fit

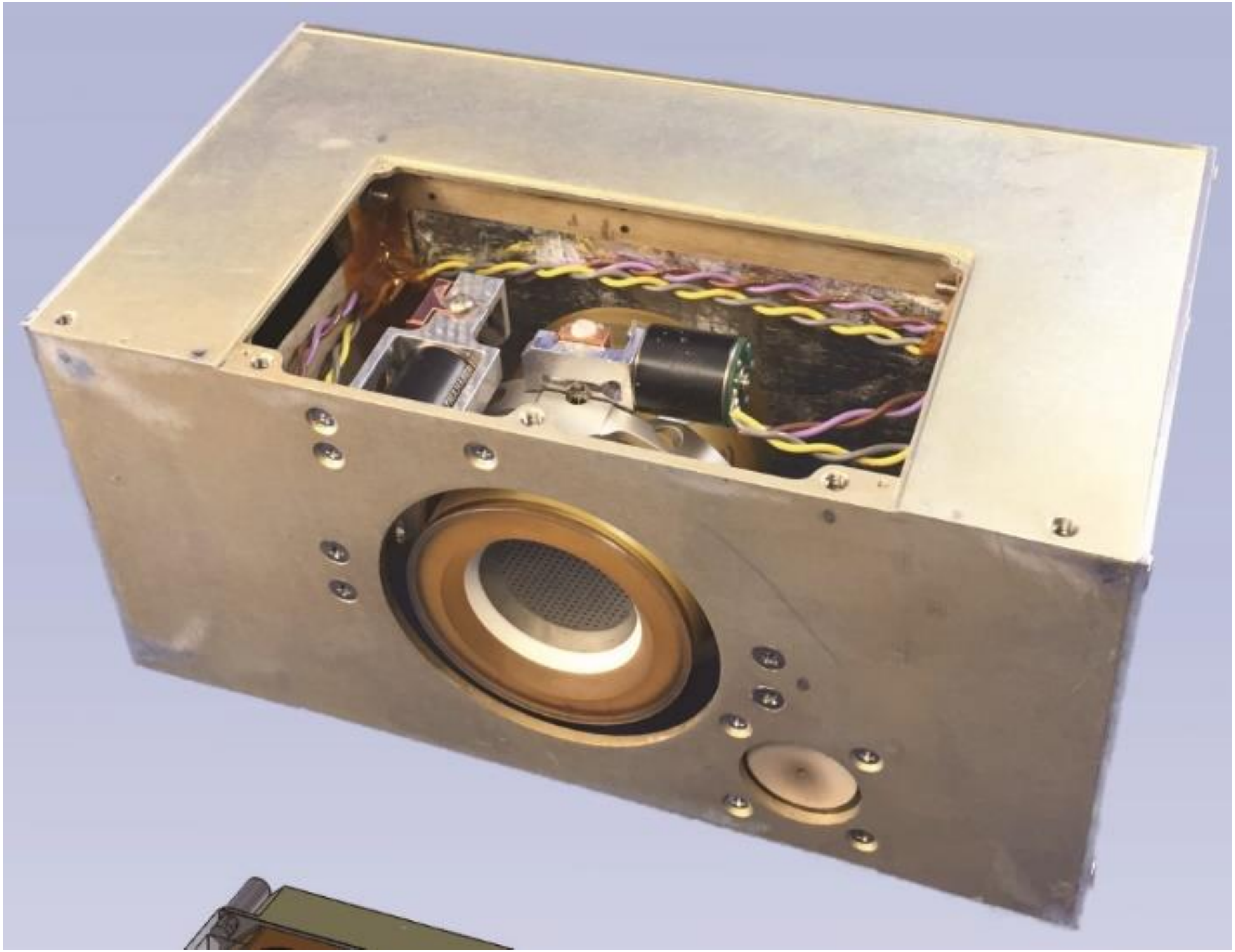
Busek Ion Thruster



BIT-3 Iodine Propellant

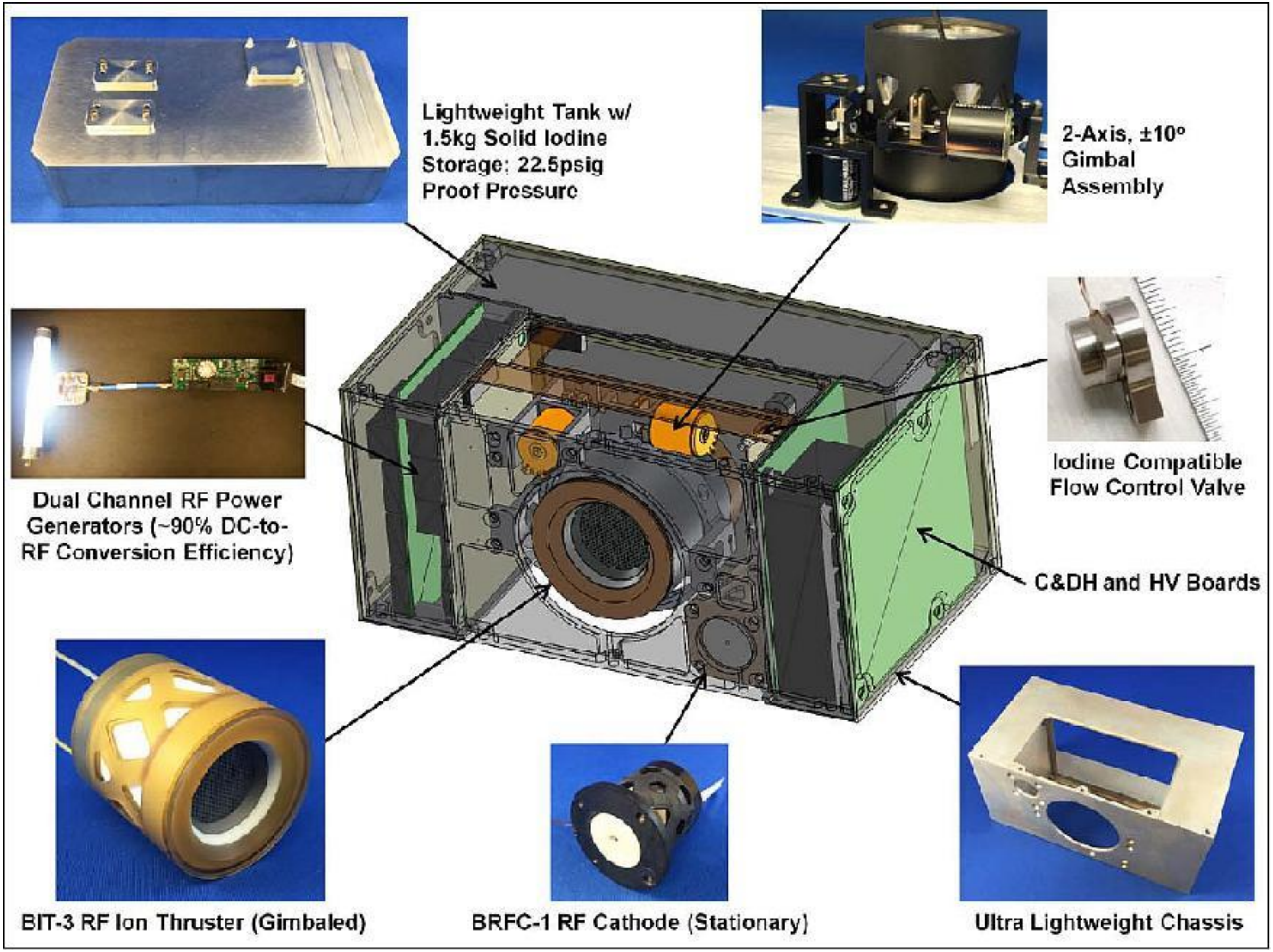
75W, 1.24 mN, 2.5 cm beam width, I_{SP} = 2,640
For a 6U, 14 kg spacecraft with 1.5 kg iodine:
Delta-V = 2,900 m/s

Busek Bit-3 Ion Thruster



Isp = 2,640 s, Iodine mass = 1.5kg, Δv = 2,900 m/s, 8,600 hours of thrust

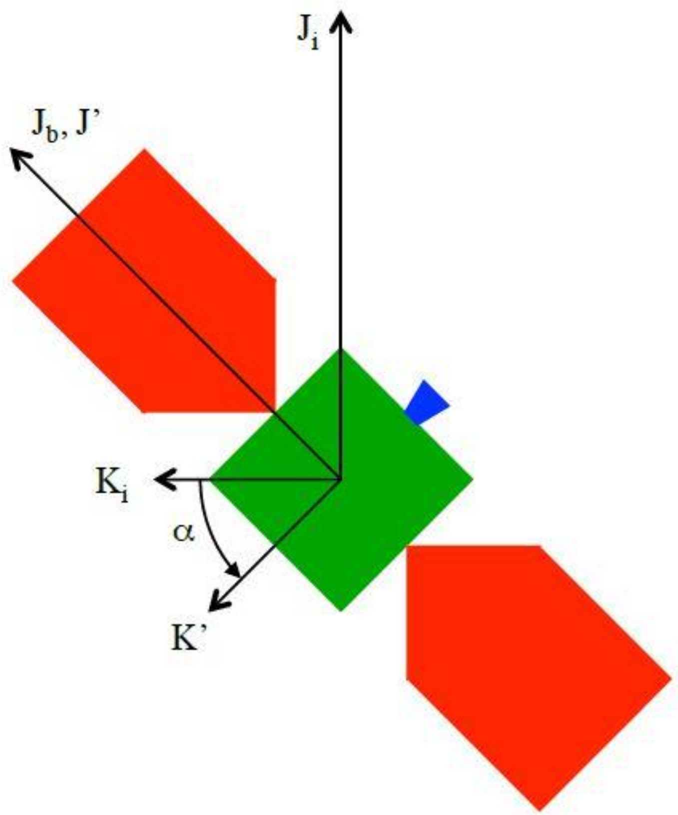
Busek BIT-3 Ion Thruster



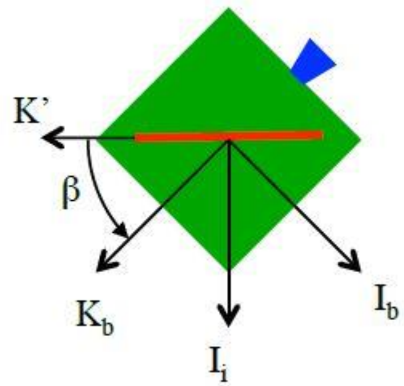
VERMONT TECH

Spiral Thrusting for 3 axis control with a 2 axis thruster

Software by Chris Farnsworth, M.S.S.E. student at Vermont Technical College



First Rotation (about I_i)



Second Rotation (about J')

Algorithm by Thomas M. Randolph, Timothy P McElrath, Steven M. Collins, David Y. Oh: NASA Jet Propulsion Lab

Spiral Thrusting for 3 axis control with a 2 axis thruster

$$\begin{vmatrix} X' \\ Y' \\ Z' \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{vmatrix} \begin{vmatrix} X_i \\ Y_i \\ Z_i \end{vmatrix}$$

Rotation around I

$$\begin{vmatrix} X_b \\ Y_b \\ Z_b \end{vmatrix} = \begin{vmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{vmatrix} \begin{vmatrix} X' \\ Y' \\ Z' \end{vmatrix}$$

Rotation around J

$$\begin{vmatrix} X_b \\ Y_b \\ Z_b \end{vmatrix} = \begin{vmatrix} \cos\beta & \sin\alpha\cos\beta & -\cos\alpha\sin\beta \\ 0 & \cos\alpha & \sin\alpha \\ \sin\beta & -\sin\alpha\cos\beta & \cos\alpha\sin\beta \end{vmatrix} \begin{vmatrix} X_i \\ Y_i \\ Z_i \end{vmatrix}$$

Matrix product gives the result of both rotations

JT65 Weak Signal Protocol

Joe Taylor (my physics prof, 1993 Nobel Prize)

Each message contains 72 (378 with FEC) bits over 48 seconds

With a 3m dish, @ 9 GHz, you can reach Jupiter (4.45 AU)

Calculated Performance

SNR (dB)	Channel symbols	Bits
−18	46.9	281 10.1
−20	39.6	237 8.4
−22	31.9	191 6.9
−24	23.1	139 4.9
−26	15.5	93 3.3
−28	9.6	58 2.1

Actual Performance

Frequency (MHz)	432
Lossless antenna gain (dBi)	22.40
Solar Flux at 432 MHz (SFU)	44.0
Tx power at antenna (W)	100
EME path loss (dB)	261.6
G/Ta (dB/K)	5.5
G/Ts (dB/K)	1.6
Y Sun (dB)	9.9
EME S/N in B=2500 Hz (dB)	-23.0
EME S/N in B=50 Hz (dB)	-6.0

JT65 Weak Signal Protocol

MarCO (6U, 10cm x 20cm x 30cm, 14kg) with 4 W Iris-2 X-Band (9 GHz) Radio, relay for InSight, 60 cm × 34 cm antenna, >28 dB gain (1m dish is 37 dB)

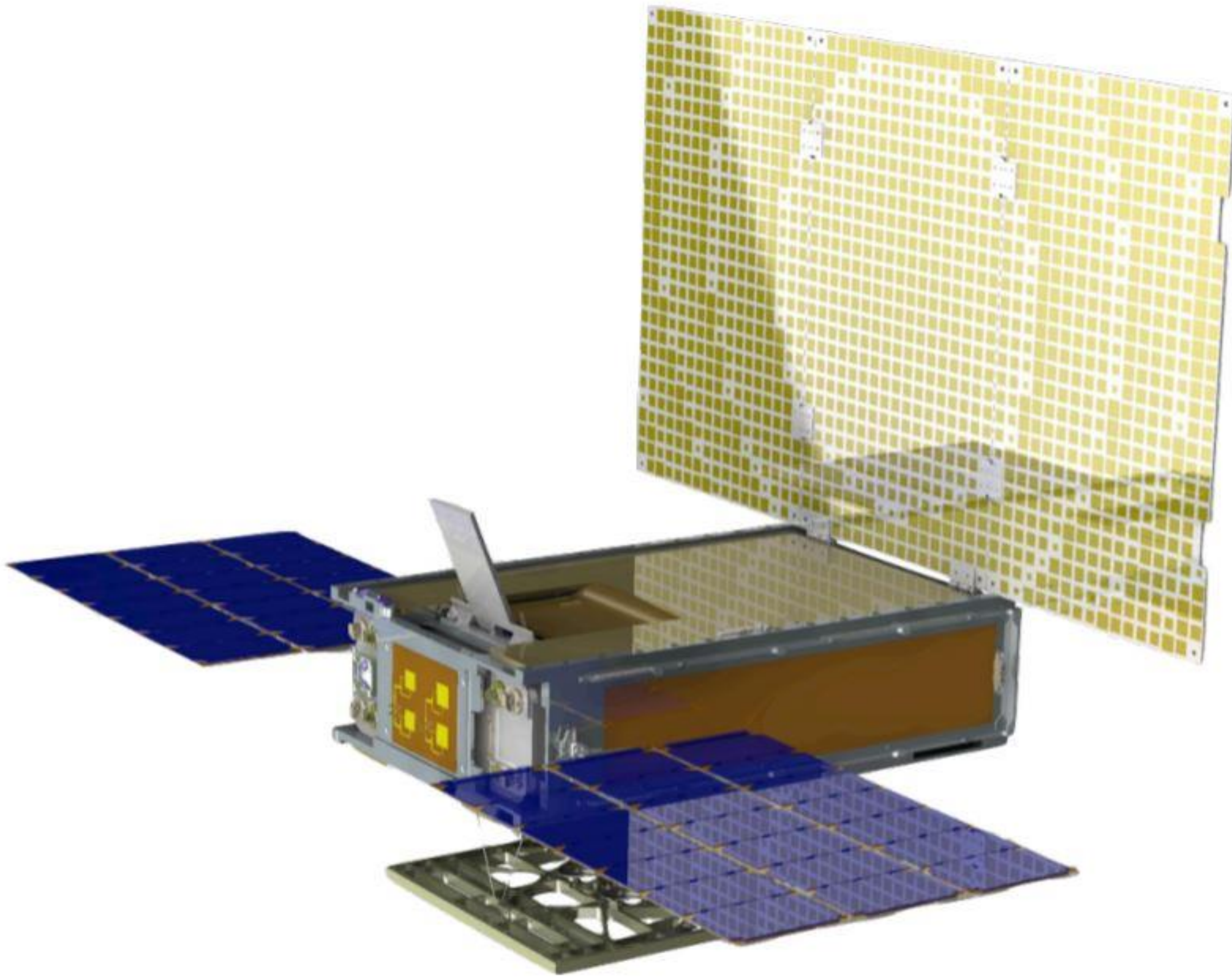


Figure 1 – CAD model rendering of a MarCO CubeSat. The large vertical panel is the high-gain reflectarray, capable of transmitting 8 kbps from Mars to the Deep Space Network's 70m dish in Madrid, Spain.

Flight Software based on *CubedOS*

- *Intended to be a general purpose framework for CubeSat flight software*
- Written **in SPARK**; proven free from runtime errors
- Provides inter-module message passing framework
- Provides services of interest to flight software
- Can integrate existing Ada or C runtime libraries
- *Conceptually similar to NASA's cFE/CFS except written in SPARK (not C).*
- Non ITAR parts on GitHub, ITAR parts from us

Current Software Environment

- Linux with various cross compilers
- SPARK 2014 with Ravenscar runtime
- We have VxWorks 6.8 capability

Current Development Team

- VTC: 2 faculty, 2 students (1 MS, 1 BS)
- Students trained and supervised by Peter Chapin

CubedOS Verification Goals

- No flow errors
- Show freedom from runtime error
- Other correctness properties as time allows

CubedOS Testing

- Unit tests
- Some additional test programs (x86)
- Hardware development system (PowerPC)
- Hardware "FlatSat" to be fabricated

Continuous Integration

- We use Jenkins-CI (https://jenkins.io/)
- Every night…
  - … builds & executes unit test programs
  - … does SPARK flow analysis
  - … does SPARK proofs
- Build considered to have failed if unit tests fail
  - Requiring successful proofs for "successful" build too high a bar

Software Architecture

- Collection of “modules” that pass messages
  - Each module reads messages from exactly one mailbox
  - Each module contains a message processing task
  - Modules all execute concurrently
- Collection of libraries
  - Passively called from multiple modules

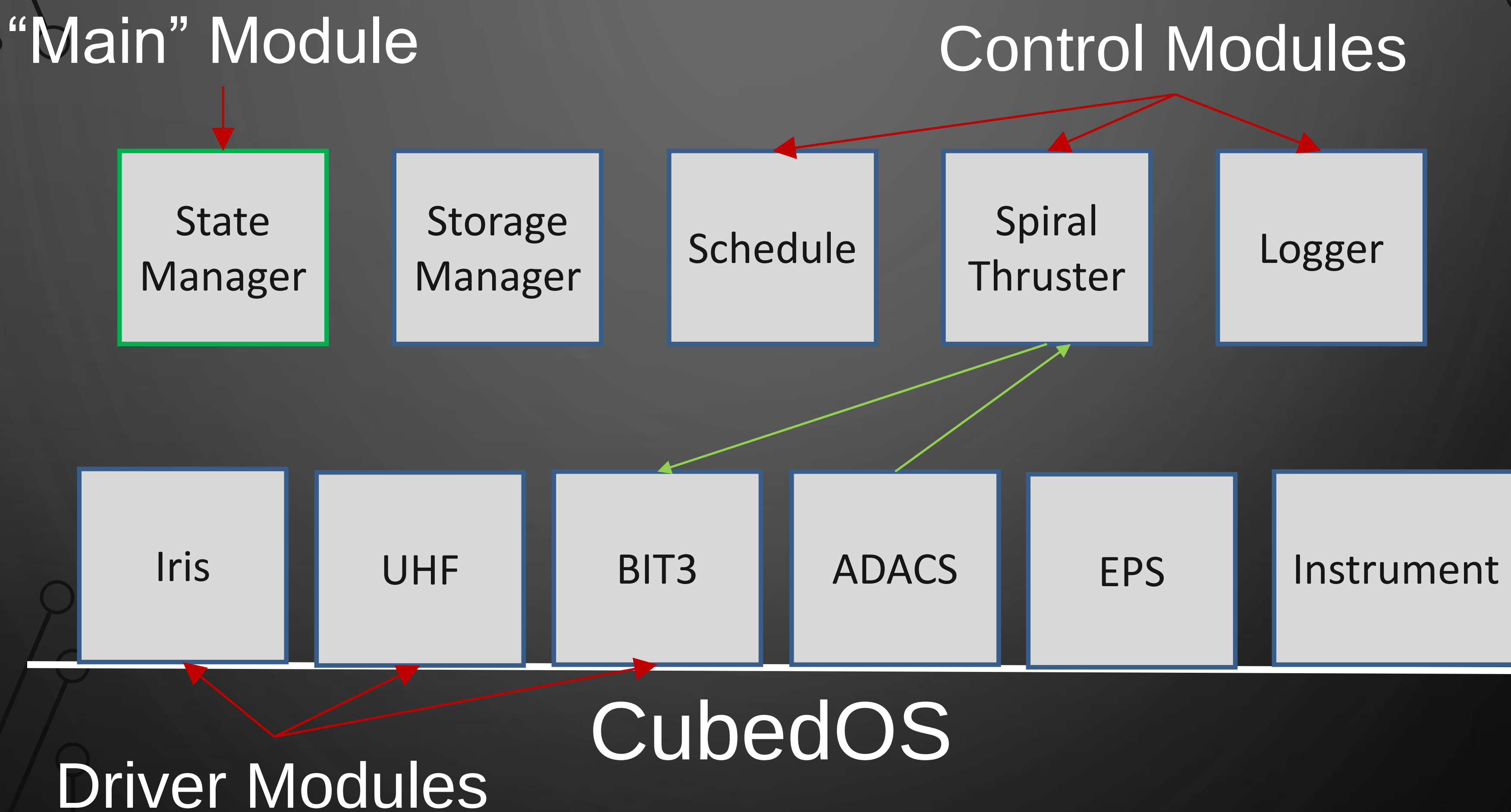
Software Architecture

- CubedOS comes out-of-the-box with:
  - A set of standard server modules
    - Timing services
    - Publish/Subscribe services
    - File system interface
    - Communication protocols (e. g., CFDP)
    - … etc
  - A set of library facilities
    - CRC, Packet encoding/decoding, data compression

Small Spacecraft Flight Software

- A CubedOS application
  - Application modules for:
    - Device drivers for subsystem hardware
    - Spacecraft state manager (“main” module that initiates and coordinates other activity)
    - Command scheduler
    - Implementation of CubedOS standard file system interface

Software Stack (Spacecraft Modules)



CubedOS Mailboxes

```
generic
   Module_Count : Positive;
   Mailbox_Size : Positive;
   Maximum_Message_Size : Positive;
package CubedOS.Generic_Message_Manager is
 type Message_Record is
      record
         Sender     : Module_ID_Type;
         Receiver   : Module_ID_Type;
         Message_ID : Message_ID_Type;
         Priority   : System.Priority;
         Size       : XDR_Size_Type;
         Payload    : XDR_Array;
      end record;
 type Message_Array is array(Message_Index_Type) of Message_Record;
 protected type Mailbox is … end Mailbox;
 Mailboxes : array(Module_ID_Type) of Mailbox;
end CubedOS.Generic_Message_Manager;
```

Mostly for future expansion

XDR encoded message parameters

CubedOS Mailboxes

- Each instantiation of the message manager creates a “communication domain”
- Multiple communication domains possible
- Each module has unique ID within its domain
- Each module has a single task that reads its mailbox and handles/dispatches messages
- Message parameters are encoded/decoded *at runtime* into octet streams and installed into the receiver’s mailbox

CubedOS Modules

- Each module is a hierarchy of packages
  - Complex modules might have multiple private child packages to support implementation
- Some_Module.API
  - Contains subprograms for encoding/decoding messages
  - ***Generated automatically by the XDR2OS3 tool*** from a high level message specification
- Some_Module.Messages
  - Contains the message loop and message handling

CubedOS Modules

- Module communication is point-to-point
  - Sender names receiver explicitly
  - Receiver learns sender ID from message header
  - Replies returned via (dynamically specified) ID
- Server modules
- Can be written without knowledge of clients
- Provided by third party libraries
- Future work
  - supporting CubeSat swarms using distributed message passing between CubedOS domains on different spacecraft

Advantages

– Lots of behavior deferred to runtime
  - Flexible and dynamic communication patterns
  - Easily extensible via module libraries
  - OOP-like behavior
    – Many different implementations of the same module API are possible; clients need not know which implementation they are using

Disadvantages

- *Lots of behavior deferred to runtime!*
  - Message encoding/decoding overhead (space and time)
  - Loss of type safety (compare with well-typed protected object entry calls)
- *Not the SPARK way!*
  - But… type safety issue mitigated somewhat by XDR2OS3

Problem with Mailboxes

- SPARK won’t track information flow through arrays
  - *“high: multiple tasks might queue on protected entry "message_manager.mailboxes.receive”*
- We suppress this message!
- Can’t track flow between modules
  - We must take responsibility for initialization, etc.
  - But… this allows flexible communication
- Full strength of SPARK within modules
- NOTE: *Must ensure modules have unique IDs!*

XDR2OS3

File server mxdr file

Sean Klink, M.S.S.E. student at Vermont Technical College

```
message struct -> Read_Request{
    File_Handle_Type       Handle;
    Read_Size_Type          Amount;
};

message struct <- Read_Reply {
    Valid_File_Handle_Type      Handle;
    Read_Result_Size_Type       Amount;
     opaque Message_Data[1024];
} with message_invariant =>
Amount <= Message_Data'Length;
```

Generated spec file

```
 function Read_Request_Encode
    (Sender_Domain : Domain_ID_Type;
     Sender    : Module_ID_Type;
     Handle    : Valid_File_Handle_Type;
     Amount    : Read_Size_Type;
     Priority : System.Priority := System.Default_Priority)
return Message_Record
    with Global => null;

function Read_Reply_Encode
    (Receiver_Domain : Domain_ID_Type;
     Receiver : Module_ID_Type;
     Handle    : Valid_File_Handle_Type;
     Amount    : Read_Result_Size_Type;
     Message_Data     : CubedOS.Lib.Octet_Array;
     Priority : System.Priority := System.Default_Priority)
return Message_Record
    with
      Global => null,
      Pre => Amount <= Message_Data'Length;
```

Generated body file

```
function Open_Request_Encode
   (Sender_Domain : Domain_ID_Type;
    Sender      : Module_ID_Type;
    Mode       : Mode_Type;
    Name       : String;
    Request_ID : Request_ID_Type;
    Priority   : System.Priority := System.Default_Priority) return Message_Record
  is
    Message : Message_Record := Make_Empty_Message
      (Sender_Domain => Sender_Domain,
       Receiver_Domain => Domain_ID,
       Sender     => Sender,
       Receiver   => ID,
       Message_ID => Message_Type'Pos(Open_Request),
       Priority   => Priority);

    Position : XDR_Index_Type;
    Last : XDR_Index_Type;
  begin
    Position := 0;
    XDR.Encode(XDR.XDR_Unsigned(Mode_Type'Pos(Mode)), Message.Payload, Position,
Last);
    Position := Last + 1;
    XDR.Encode(XDR.XDR_Unsigned(Name'Length), Message.Payload, Position, Last);
    Position := Last + 1;
    XDR.Encode(Name, Message.Payload, Position, Last);
    Position := Last + 1;
    XDR.Encode(XDR.XDR_Unsigned(Request_ID), Message.Payload, Position, Last);
    Message.Size := Last + 1;
    return Message;
  end Open_Request_Encode;
```

```
procedure Open_Request_Decode
    (Message : in  Message_Record;
    Mode : out Mode_Type;
    Name : out String;
    Name_Size : out Natural;
    Request_ID : out Request_ID_Type;
    Decode_Status : out Message_Status_Type)
is
   Position : XDR_Index_Type;
   Raw_Mode : XDR.XDR_Unsigned;
   Raw_Name_Size : XDR.XDR_Unsigned;
   Raw_Request_ID   : XDR.XDR_Unsigned;
   Last : XDR_Index_Type;
begin
   Decode_Status := Success;
   Name := (others => ' ');
   Request_ID := Request_ID_Type'First;
   Position := 0;
   if Decode_Status = Success then
      XDR.Decode(Message.Payload, Position, Raw_Mode, Last);
      Position := Last + 1;
      if Raw_Mode in Mode_Type'Pos(Mode_Type'First) ..
Mode_Type'Pos(Mode_Type'Last) then
         Mode := Mode_Type'Val(Raw_Mode);
      else
         Decode_Status := Malformed;
         Mode := Mode_Type'First;
      end if;
   end if;
```

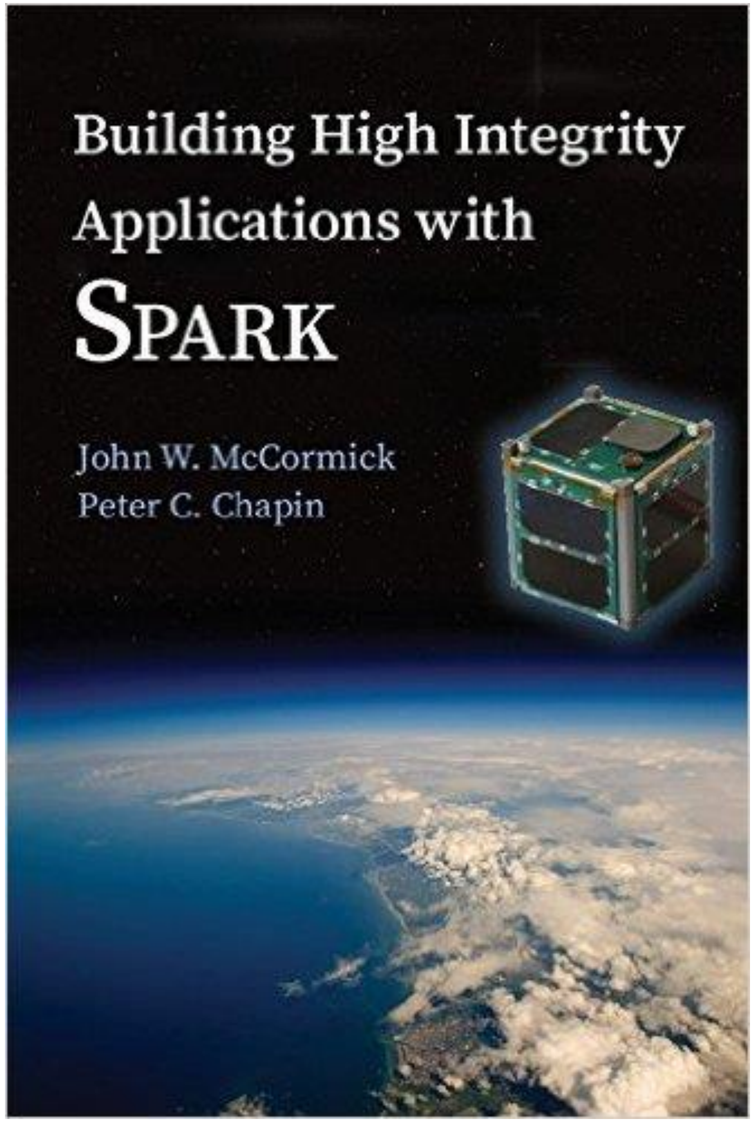
```
 if Decode_Status = Success then
     XDR.Decode(Message.Payload, Position, Raw_Name_Size, Last);
     Position := Last + 1;
     if Raw_Name_Size in XDR.XDR_Unsigned(Natural'First) ..
XDR.XDR_Unsigned(Natural'Last) then
        Name_Size := Natural(Raw_Name_Size);
    else
       Name_Size := 0;
    end if;
    if Name_Size < 1 then
        XDR.Decode(Message.Payload, Position, Name(Name'First .. Name'First +
(Name_Size - 1)), Last);
    end if;
  end if;
  if Decode_Status = Success then
      XDR.Decode(Message.Payload, Position, Raw_Request_ID, Last);
      Position := Last + 1;
      if Raw_Request_ID in XDR.XDR_Unsigned(Request_ID_Type'First) ..
XDR.XDR_Unsigned(Request_ID_Type'Last) then
          Request_ID := Request_ID_Type(Raw_Request_ID);
          Decode_Status := Success;
      else
           Decode_Status := Malformed;
      end if;
   end if;
end Open_Request_Decode;
```

Why not NASA’s cFE/CFS?

VERMONT TECH

- “cFE/CFS” = “Core Flight Executive / Core Flight System”
- Similar architecture
  - Uses publish/subscribe (not point-to-point)
  - Uses CCSDS space packets for messages
- cFE written in C. Not verified
- We hope to eventually offer CubedOS as a competing SPARK platform for spacecraft software
- possible CubedOS/CFS bridge that will translate messages between the systems

A SPARK 2014 Book is Available



Deep Space Network Ground Stations

The 70m Dish at Goldstone, California
X-band, 74 dB gain, 12 TW radar

183 kW array on my field, 5.25 kw on my garage, heat pumps (heat and hot water), Tesla Powerwall & Car

Mileage for My Solar Powered Tesla

VERMONT TECH

My Tesla is charged from my photovoltaic array which converts the fusion of hydrogen via the proton-proton chain:

The net result is four protons become one helium nucleus with the release of about 25 MeV.

Since the pressures and temperatures are so high in the sun, we'll look at liquid hydrogen. Its density is 70.8 kgm^{-3} (1,000 l). One gallon = 3.785 l, (1,000 l m^{-3} / 3.785 l $gallon^{-1}$ = 264.2 gallons m^{-3}), 70.8 kgm^{-3} / 264.2 gallons m^{-3} = 0.2680 kg $gallon^{-1}$ / 0.00108 kg mol^{-1} (H) = 265.85 mol $gallon^{-1}$ (H) /4 = 66.46 mol (4 x H) $gallon^{-1}$ x 6.02 x 10^{23} mol^{-1} = 4.00 x 10^{25} (4 x H) $gallon^{-1}$ x 25 MeV $(4 \times H)^{-1}$ = 1.00 x 10^{27} MeV gallon-1 / 1.602 x 10^{-13} J $(MeV)^{-1}$ = 1.602 x 10^{14} J $gallon^{-1}$ / 3,600 J Wh^{-1} = 4.450 x 10^{10} Wh $gallon^{-1}$ / 314 Wh $mile^{-1}$ (my Tesla average) = 1.42 x 10^{8} miles per gallon = 142 million mpg (60 million km/l). If we use the actual H density in the center of the sun, where fusion takes place, of 150,000 kgm^{-3}, we get 3.00 x 10^{11} miles per gallon = 300 billion mpg (127 billion km/l). If we had 20 gallons (76 l), we could drive over one light year!

Vermont’s First Astronaut

It **is** rocket science!

$$\Delta V = I_{sp}\, g \ln\left(\frac{m_i}{m_i - m_p}\right)$$

Rocket equation

$$\tau = \dot{m}\, u_e + (P_e - P_a) \cdot A_e$$

Thrust Equation

$$r = \frac{a(1 - e^2)}{\mathbf{1 + e(\cos \nu)}}$$

Orbital Equation

$$m_p = m_i \left[1 - e^{-\left(\Delta V / I_{sp} g\right)}\right]$$

Mass of needed propellent

$$\xi = \frac{1}{2} v^2 - \frac{\mu}{r}$$

Energy

$$\dot{\Phi} = \left[\frac{I_3 \dot{\psi}}{(I_1 - I_3)\cos\theta}\right]$$

Precession Rate

$$I\dot{\omega} + \omega \times I\omega = M$$

Rigid body rotational dynamics

$$\left\langle \frac{d\Omega}{dt} \right\rangle_{1rev} = \frac{-\frac{3}{2}(n) J_2 \cos i}{(1 - e^2)^2} \left[\frac{R_\oplus}{a}\right]^2$$

RAAN change due to J_2

Acknowledgements

- NASA Vermont Space Grant Consortium
- Vermont Technical College
- AdaCore, Inc. (GNAT Pro, SPARK Pro)
- Applied Graphics, Inc. (STK)
- Busek (BIT-3 Iodine ion drive)
- NASA Jet Propulsion Lab (Iris-2 Radio)

From Physicist to Rocket Scientist with a Few Detours

How to make a CubeSat that Works!

Dr. Carl Brandon
carl.brandon@vtc.edu
Vermont Technical College
+1-802-356-2822
Randolph Center, VT 05061 USA
http://www.cubesatlab.org