HASP Payload Specification and Integration Plan

Payload Title: University of Bridgeport High-Altitude Testbed

Payload Class: Small [✓ Large] (circle one)

Payload ID: # 11

Institution: **University of Bridgeport**

Contact Name: Bashar Alhafni

Contact Phone: (203) 916-3867

Contact E-mail: balhafni@my.bridgeport.edu

Submit Date: June 24, 2016

I. Mechanical Specifications:

A. Measured weight of the payload (not including payload plate)

Final assembly of the Chassis is complete, and the electrical Layout and wiring is underway. The Servo Trays have been drawn and are being 3D-printed this week. All of the Electronics was placed in the chassis, and weights approximating the Servo Trays and wiring were added.

This measured weight is 5.050 Kilograms, which compares favorably with the piece-part estimate below. The unknown elements that remain estimates comprise much less than 10% of the total, so we believe this measurement to be quite close to the final weight. The piece-part estimate is included in Table 1 below for reference. The estimates below for Wiring, Aluminum Angle, the ARM, and nut and bolts were all on the high side of what the actuals will be.

Table I. HASP Weight Estimate

Sheet Al 0.020 thick 6061-T6 dimensions (1415g/m^2)	sq. meter	Weight, g	
Front	0.098	139	
Starboard (1)	0.0690	98	
Starboard (2)	0.0329	47	
Stern	0.0978	138	
Port	0.0900	127	
Total		**549g**	+/- 23
Chassis Top PVC, white, UHMW PE,		**429g**	+/- 23

FR4 Sheet .063, one side 1 oz. copper (2765 g/m^2)	sq. meter	Weight, g	
Front	0.0733	203	
Starboard (1)	0.0541	150	
Stern	0.0733	203	
Port	0.0733	203	
E-Bay Floor	0.0614	170	
Total		**927g**	+/- 45

FR4 Sheet .125, no copper (4880 g/m^2)	sq. meter	Weight, g	
T-BAY Floor	0.0614	299	
Servo Tray	0.0440	215	
Total		**514g**	+/- 23

FR4 Sheet .032, no copper (1220 g/m^2)	sq. meter	Weight, g	
(2 sheets) E-Bay Floor	0.0614	**75g**	+/- 5

Aluminum Angle Bracket	**680g**	+/-136
Nuts and Bolts, standoffs, all fastener hardware	**454g**	+/- 181
Foam Insulation DOW SUPER TUFF-R 3, ½” R-Value 3.3 (.5218 W/m°K)	**417g**	+/- 23
Servo Tray 3D printed parts	**454g**	+/- 181
Wiring and Connectors	**907g**	+/- 454
Arm	**227g**	+/- 91

Electronic Parts and Circuits

Part	Weight, g
Arduino Mega 2650 (x2)	70
Prototype Shield	50
Potentiometers (x5)	30
GPS and Cable	84
ALL Servos (QTY 8, actual wt.)	278
Buck Voltage Regulator(x2)	180
Current Sensors	30
Servo Drive Board	20
Sensor Input Board	20
Equipment Bay Interface Bd.	90
N-Channel MOSFET Board	85
TOTAL	**937g** +/- 91g

The RSS (root of the sum of the squares) of all the weight uncertainties is 635 g.

TOTAL PROJECT WEIGHT (based on estimates) **6570 +/- 635 g.**
Plus weight of HASP Base plate

Current measurement/estimate after Chassis construction 5050 g (plus base plate)

B. Provide a mechanical drawing detailing the major components of your payload and specifically how your payload is attached to the payload mounting plate.

Figure 1. Assembly isometric view

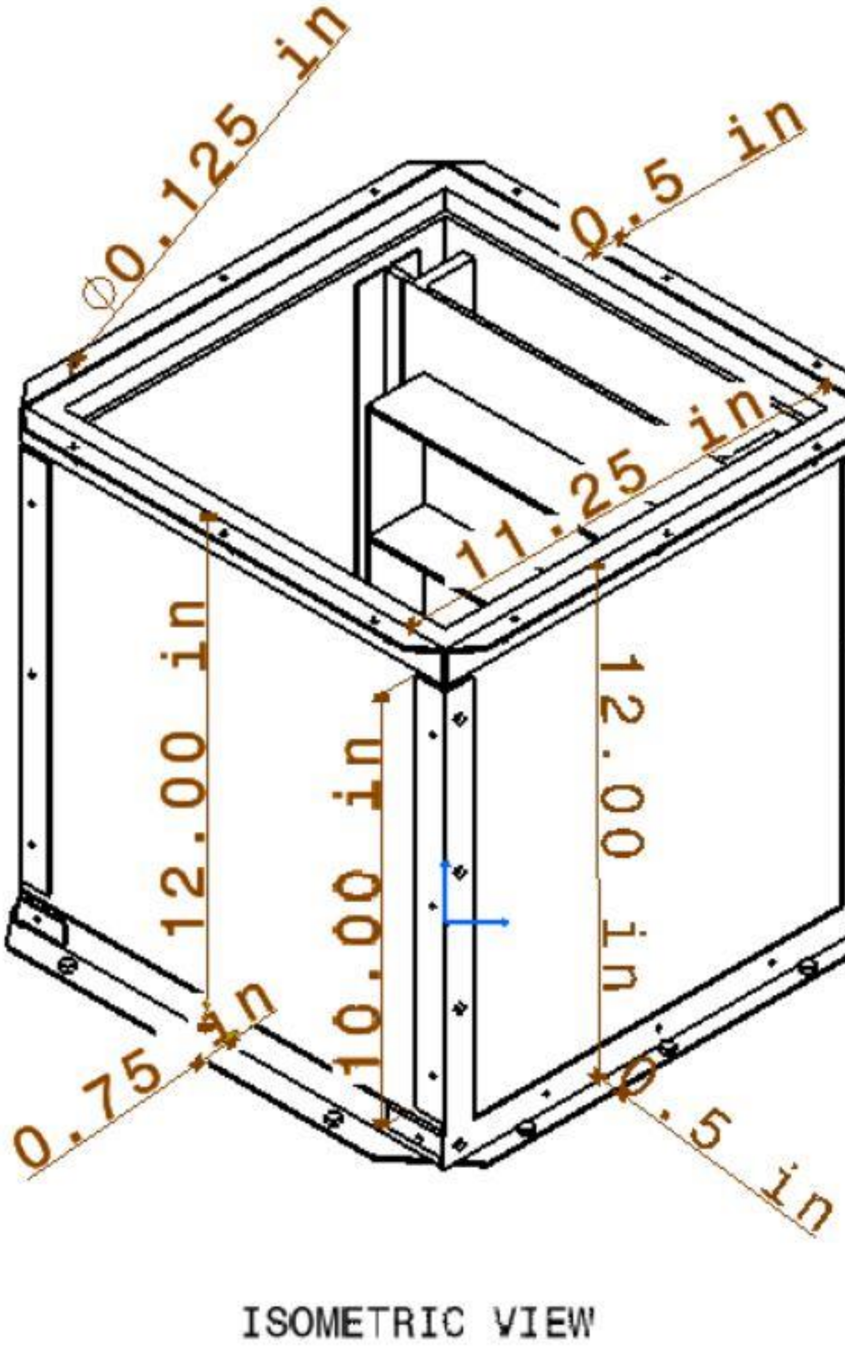


Figure 2. Container mounted on base plate

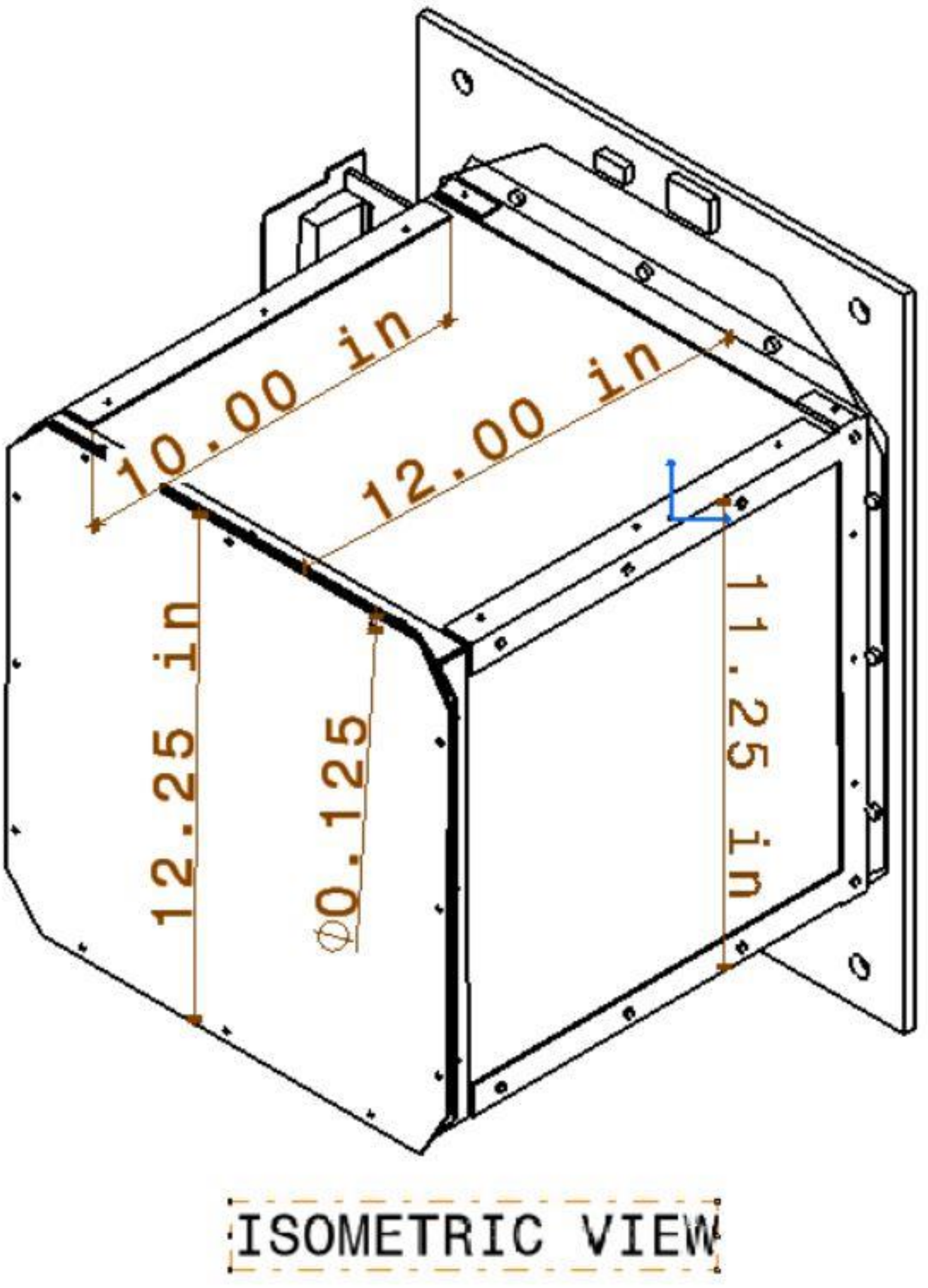
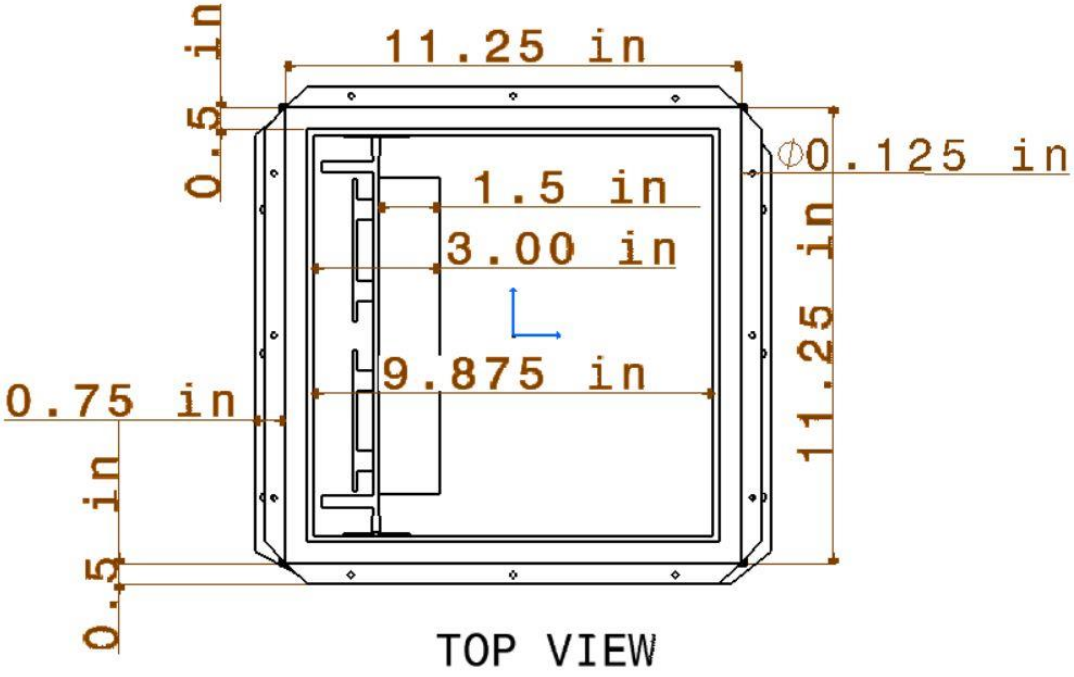


Figure 3. Container top view



In the figure above, the Starboard side (with the drawer) is at the bottom. All the flanges on the top and bottom edges are 0.5 inches wide, except two: the Bow and Stern bottom flanges are 0.75 inches wide. The Port and Starboard flanges attach to the base plate with a total of six 6-32 bolts and locknuts. The Bow and Stern flanges attach to the base plate with a total of six #8 bolts and locknuts.

Note: the drawings reflect as-built hardware, and all bolts are presently #6 hardware. Where we have room for larger nuts on the Bow and Stern sides, we can increase the bolt size to #8 or possibly #10, and we intend to.

Figure 4. Container top plate

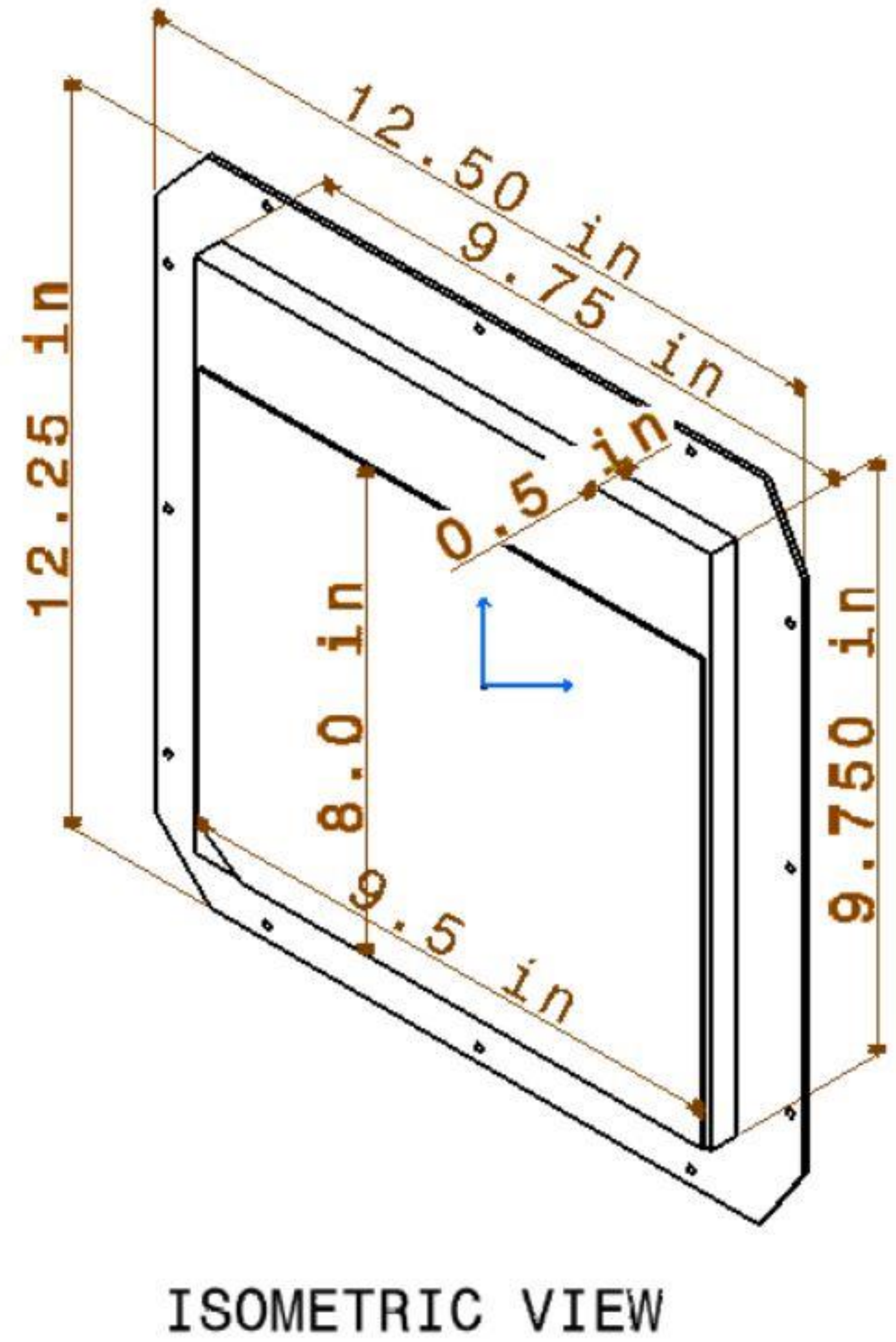
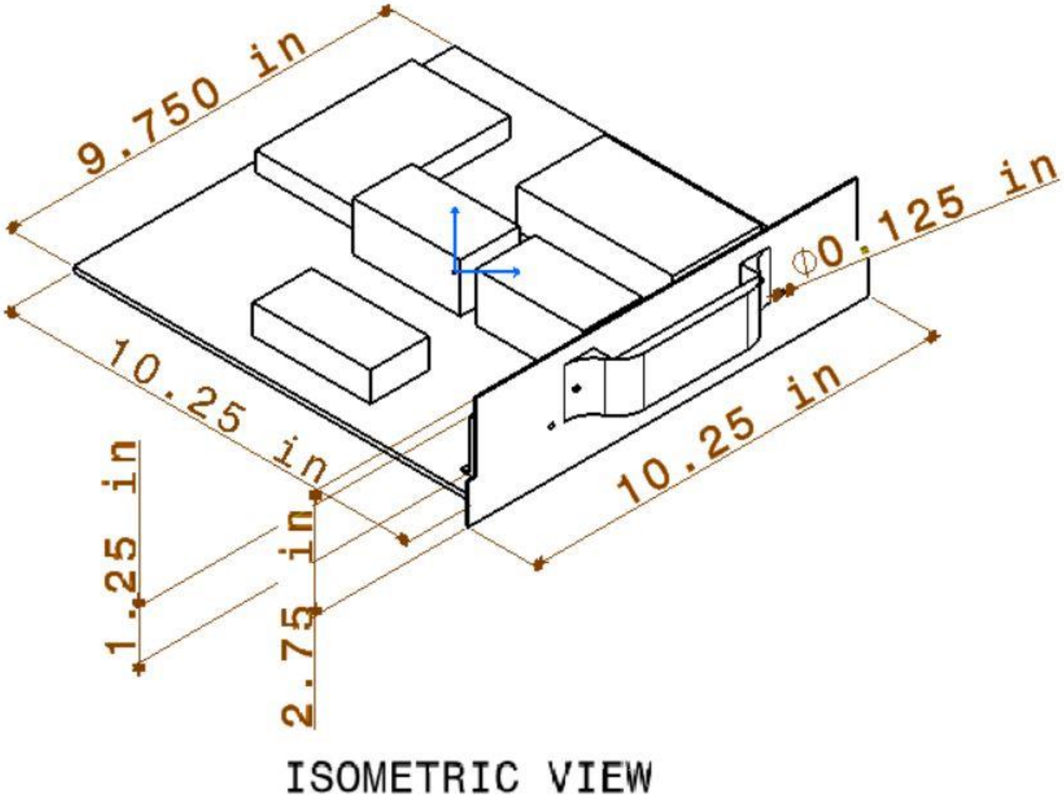
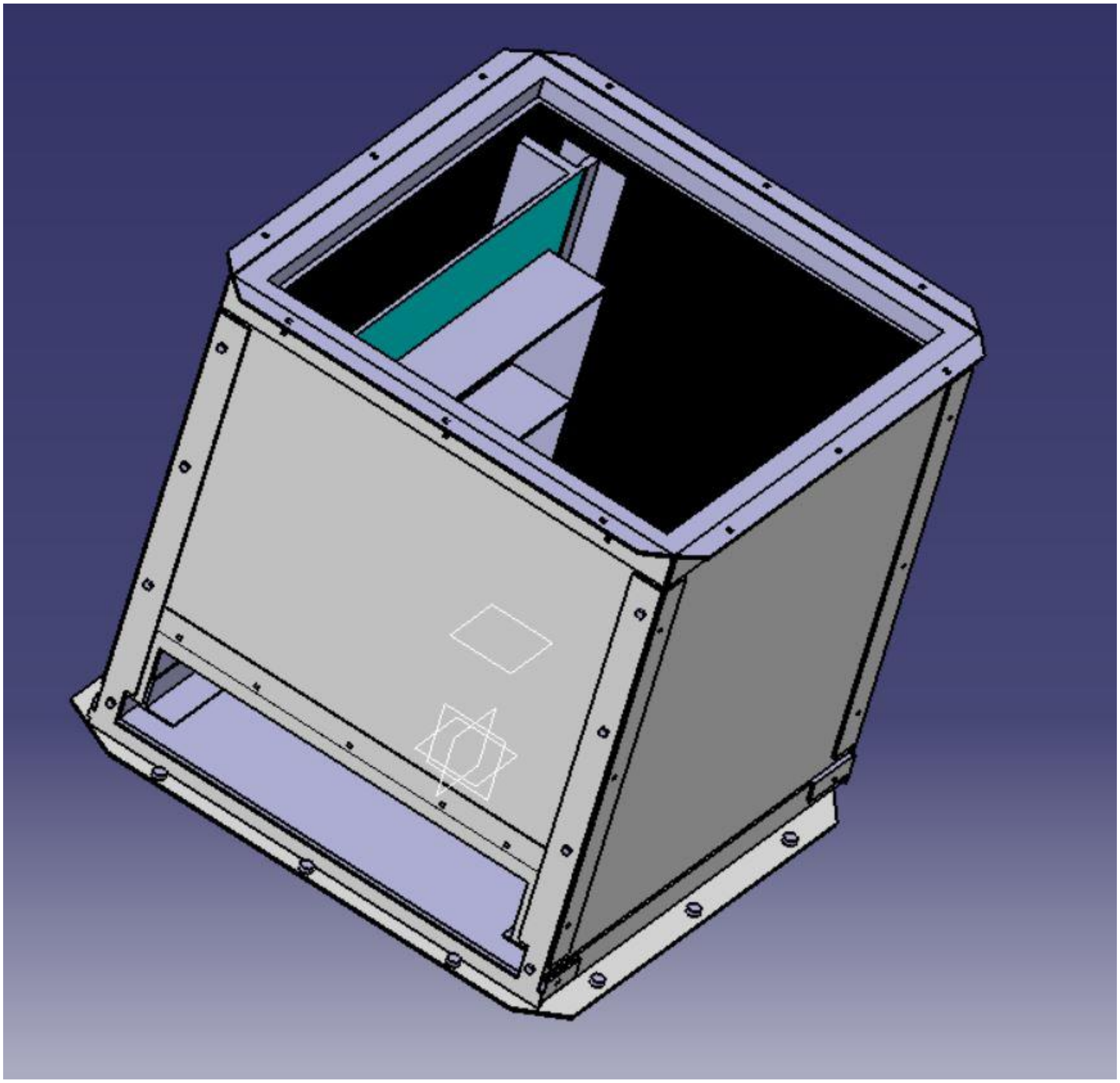


Figure 5. Electronics drawer



The drawer Slides into the chassis on rails. The front is lifted slightly to clear the flange, then drops down to lock in place. The rear of the drawer is held down firmly against the rails by foam insulation. In flight the front of the drawer will be secured to the chassis with 2 #6 sheet metal screws. The Starboard bottom flange keeps the drawer in place, the drawer must be lifted to be removed.

Figure 6. Container



Looking into the Test Bay. The “Servo Tray” slides in on rails from the top, and contains 4 servo/potentiometer pairs on the front of the board. The rear of the tray has two circuit boards. One is a servo driver that makes PWM signals and distributes power to seven servos, and the other distributes power to the potentiometers and routes the wiper signals to the analog inputs of the Arduino down in the Equipment Bay. There is a small robot arm on the right side of the Test Bay above that has three servos on it. These have analog feedback signals to measure their position. Also in the Test Bay: Heater Control Board, an Arduino for Test, and a high-altitude GPS head.

Figure 7. Assembly image, showing lower Electronics Bay on the Starboard side

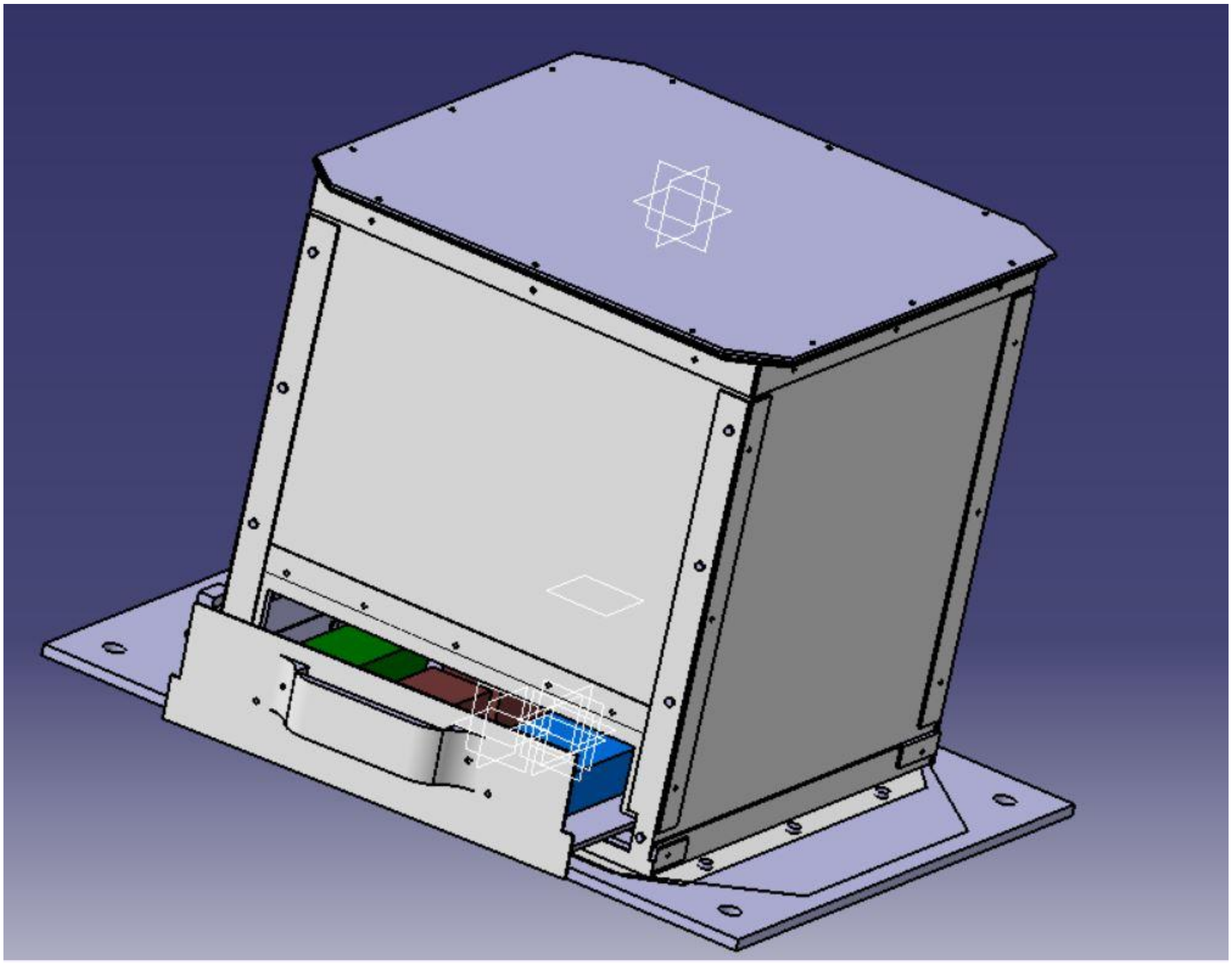
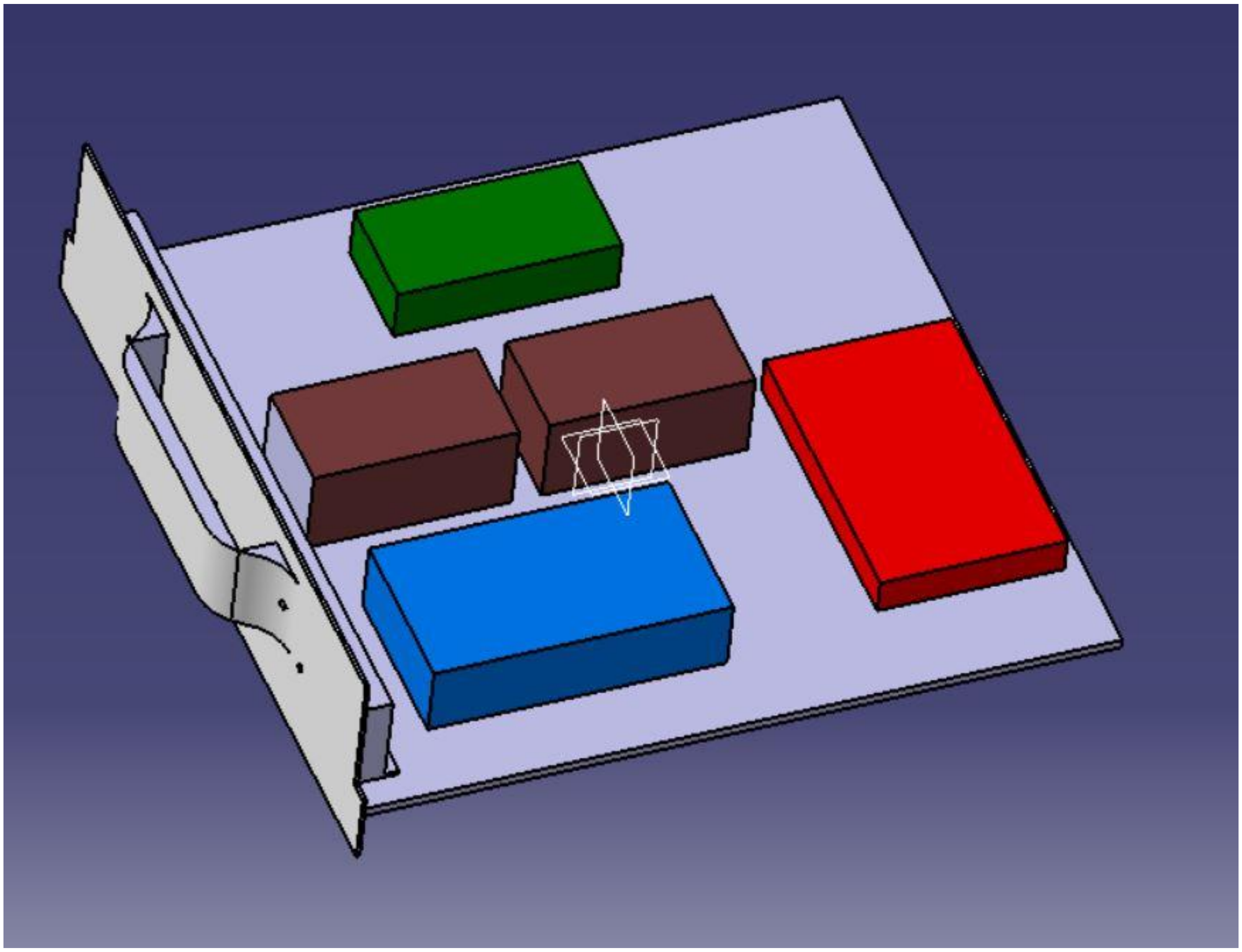


Figure 8. Electronics bay layout



Electronics Bay Layout Key:

- Blue: Control Processor Arduino MEGA 2560
- Green: Power Distribution, terminal blocks, and 30V Current Monitor
- Red: Sensor Input conditioning, Digital 1,2, and 3 Control circuits, Analog 1 and 2 drivers, Pressure Sensor, 6V Current Monitor, 6V Linear Regulator
- Brown: Two DC-DC converters and terminal block for power distribution.

Note: The Heater Control board, with 6 N-channel MOSFETS, and the 30V Fuseblocks for the Heaters, has been moved to the Test Bay.

C. If you are flying anything that is potentially hazardous to HASP or the ground crew before or after launch, please supply all documentation provided with the hazardous components (i.e. pressurized containers, radioactive material, projectiles, rockets…)

Nothing in the payload is potentially hazardous.

D. Other relevant mechanical information

No other information.

II. Power Specifications:

A. Measured current draw at 30 VDC

Several changes have been made which affect the power draw, but the total power requirement remains at less than 51 watts.

1. We deleted the solid-state relays in favor of N-channel enhancement MOSFETS to turn on the heaters.

2. We put each wall on its own heater control circuit. One wall may be in the sun, and would need less heat. With this approach we can guarantee each wall and the ceiling of the Test Bay will be at the same temperature.

3. We put all the heaters on the 30-volt supply. (Each heater has its own 1A fuse, and each has its own MOSFET for uP control.) By using different gauges of nichrome wire we are able to put an equal amount of power/area on each surface. Since we are designing the heaters (for 33V) they will each draw the designed power +/- a couple percent at 33V, and of course 82% of that at 30V. It is our intention to turn on the heaters until each wall reaches the target temperature. The target temperature will be 10°C initially, and we will try to hold that until we reach altitude. We will then drop the target temperature in the Test Bay by 10 degrees C per hour, down to ambient. The Equipment Bay will remain at 10°C, but may be dropped to -10°C.

4. We discovered the servos draw power in 10-millisecond pulses and the switching power supply cannot respond fast enough. The Servo voltage was dropping about a volt in tiny pulses. We decided to increase PS2 to 8 volts, and then use a Low-Dropout Linear Regulator behind it to drop the voltage to six volts for the servos. This linear regulator handles the current pulses well, and the 6-volt supply remains steady. This will make it much easier to calculate the power draw and energy consumed by each of the servos, at the expense of a little power.

5. We also gave up the idea of diode-heading the two supplies to provide a redundant source of power for the Arduino. So, PS1 is now 5.0 volts, and PS2 is now 8 volts, with the Linear Regulator taking that down to six volts. See Table II and Figure 9.

6. Note also on Figure 9 that the two "High-Side Current monitors" are Hall-effect devices, so the resistor labeled "shunt" is actually 0 ohms.

Table II. Maximum Power Draw, Electronic Parts and Circuits

Part	**PS1 Current 5.0 Volts (mA)**	**PS2 Current (8V switcher/6VLinear at 8 Volts (mA)**
Arduino Mega 2650 (x2)	160	
2 RS-232 Converters	20	
Potentiometers (x5)	3	
GPS	80	
ALL Servos (*active one at a time*)		2000
Current Sensors	10	
Servo Drive Board	10	
Sensor Input Board	10	
Equipment Bay Interface Bd.	50	
TOTAL	**343 mA**	**2000 mA**
Watts used, electronics	1.72 W	16.0 W
Buck Voltage Regulator loss PS1 (85%)	0.26 W	
Buck Voltage Regulator loss PS2 (85%)		2.4 W

TOTAL ELECTRONICS POWER (1.72 + 0.26 + 16.0 + 2.4) **= 20.38 Watts, but 18 watts of this is ONLY if a servo is straining to its limit.**

The servos we are testing have stall current in the range of .an Amp. While we are allocating 2A to this, it is doubtful we will ever see over one amp. As the servos are only activated once per minute, for most of the time ALL the electronics will use less than 3 watts.

Table III. Maximum Power Draw, Heaters

Note: these are actual, measured values:

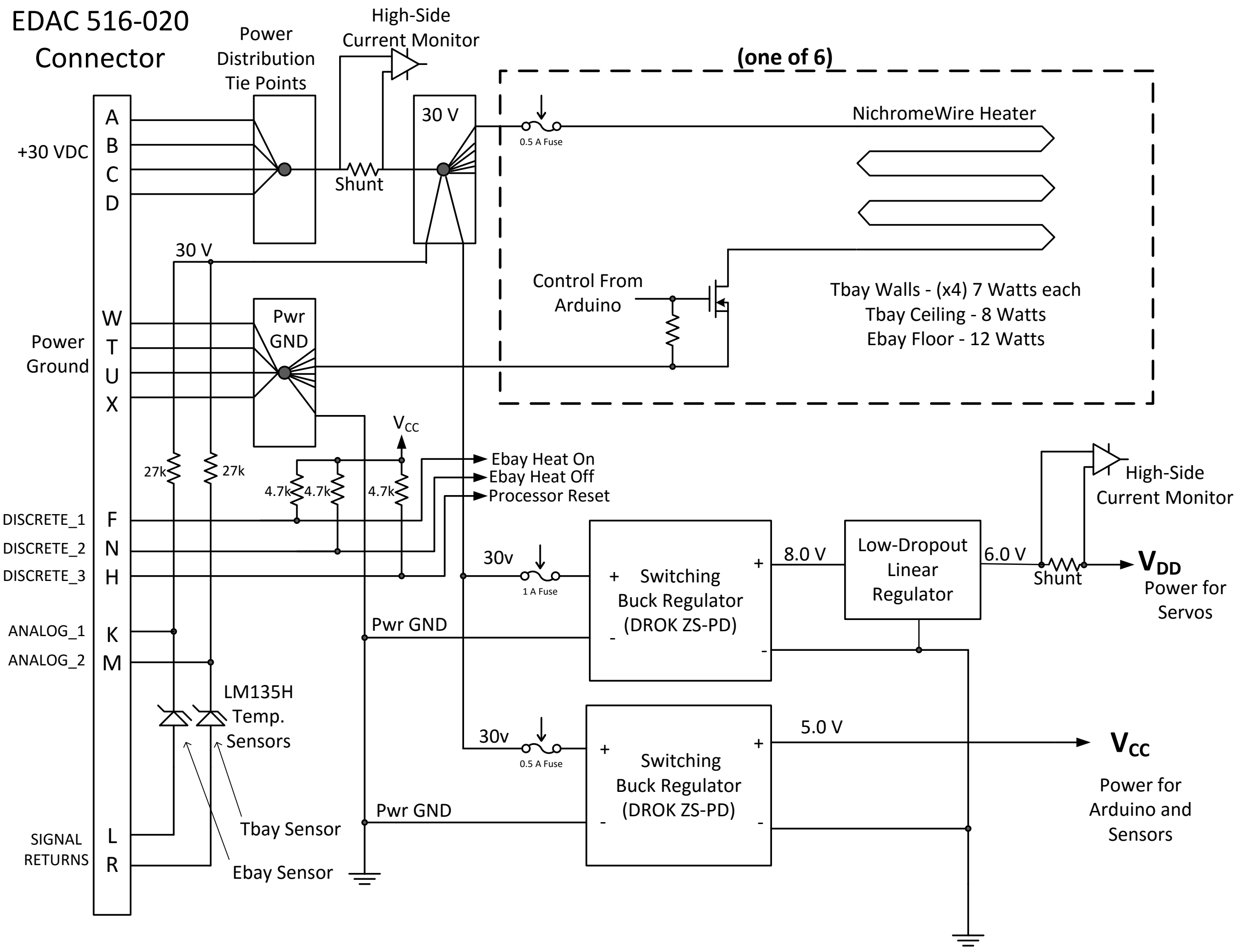
Test Bay Heater BOW	7	Watts
Test Bay Heater STERN	7	Watts
Test Bay Heater PORT	7	Watts
Test Bay Heater STARBOARD	7	Watts
Test Bay Heater CEILING	8	Watts
Electronics Bay Heater	13	Watts
TOTAL HEATER POWER	**49**	**Watts**

NOTE: The Heaters will be designed to deliver the power shown *at a Voltage of 33 Volts*. A high battery voltage will not cause us to draw excessive current. At 30V battery, the total Heater power cannot exceed 40 watts. Also, software will control the heaters, and *the ONLY time all 6 heaters will be ON simultaneously is when the servo power is off.*

Thus, the maximum power draw (with one 7W heater turned off) is (49 - 7 + 20.38) = **62.4 watts**, giving us plenty of margin. The only time the servos draw 18 watts is when one is under maximum strain, which is worst-case and is not expected to occur.

B. If HASP is providing power to your payload, provide a power system wiring diagram starting from pins on the student payload interface plate EDAC 516 connector through your power conversion to the voltages required by your subsystems.

Figure 9. Power Distribution



C. Other relevant power information

None.

III. Downlink Telemetry Specifications:

A. Serial data downlink format: [✓ Stream]   Packetized   (circle one)

B. Approximate serial downlink rate (in bits per second)

Data will be sent to the HASP from the Payload at 4800 bps, with 8 bits, no parity and one stop bit. Data will be ASCII Text and not 8-bit binary. The format of all downlink data will be per Table IV.

Table IV. Downlink data format

Byte #	Hex	Description
1	01	**SOH** Start of Header
2	xx	**Record Type** (ASCII 0-7, see below)
3	2C	Comma
4-12		**UTC Time** in hhmmss.ss format, with hh > 30 if not known
13	2C	Comma
14-16		**Data Record length** in bytes (001-999) n bytes
17	0D	**CR** (Carriage Return)
18	0A	**LF** (Line Feed)
19	02	**STX** Start of Text
20 to (n+19)		**Data Record** (n bytes long, can contain Carriage Rtn/Line Feed/Tab)
n+20	03	**ETX** End of Text
n+21 to n+23		8-bit **Checksum**, in ASCII from 000 to 255
n+24	0D	**CR** (Carriage Return)
n+25	0A	**LF** (Line Feed)
n+26	04	**EOT** (End of Transmission

C. Specify your serial data record including record length and information contained in each record byte.

There will be several types of Records:

0: Startup Message. Data field is "SYSTEM READY"

1: Time Sync Message. Issued when our own GPS first gets a valid fix, or when we receive a fix from HASP. Data Record will be "TIME SYNC ACHIEVED"

2: Command Response. The Data section for this record includes the last command received, and the status of the command. It will be sent after the command is processed.

3: Package Status. Issued whenever there is a change of status, or every ten minutes. Data Record approximately 80 bytes.

4: OWN GPS Message. Data Record 100 bytes (one GPS $GPGGA sentence). Sent once per minute.

5: HASP GPS Message. A copy of the HASP GPS Message. Data Record 125 bytes. Sent once per minute.

6: Test Data Message. This is the report of all the heaters, all the temperature sensors, three monitored voltages, the pressure sensor, and the Arduino board test status. Also the position, energy, and time info for each of the 7 servos. The Data Record is 999 bytes long, and the report will be sent once per minute.

7. ALARM Message. Issued only when a failure occurs. The Data Record is 80 bytes.

In Summary, Messages 4, 5, and 6 will be sent every minute, for a total of 1399 bytes. Every 10 minutes, a status message will add another 105 bytes. It is important to note that the software is not complete, and changes may have to be made.

The **DATA RATE** will be 4800 bps while transmitting data, which will take 2.92 seconds every minute, and 3.14 seconds every tenth minute. We will begin this transmission at the 30-second mark each minute, when we are synchronized to GPS Time.

…so if HASP closes the file and opens another on the even minute, we should be between transmissions, and not in the middle of one.

The **AVERAGE DATA RATE** is thus (13990 + 105 = 14095) bytes in 10 minutes, and since each byte is 10 bits (8+ start and stop bits), we have 140950 bits transmitted in 600 seconds, or **235 bits/second.**

D. Number of analog channels being used:

Both Analog channels will be used.

E. If analog channels are being used, what are they being used for?

ANALOG 1 – An analog voltage between 0-5v proportional to the temperature in the Electronics Bay. Specifically, Voltage = 10 mv per °K

ANALOG 2 – An analog voltage between 0-5v proportional to the temperature in the Test Bay. Specifically, Voltage = 10 mv per °K

F. Number of discrete lines being used:

We will use three discrete lines. They will be momentary low-active signal inputs to our payload.

G. If discrete lines are being used what are they being used for?

1. DISCRETE 1 – Turn ON Electronics Bay Heater. This will set a latching relay. The heater will automatically turn off if the temperature rises above 30°C.
2. DISCRETE 2 – Turn OFF Electronics Bay Heater
3. DISCRETE 3 – Reset the control microprocessor.

H. Are there any on-board transmitters? If so, list the frequencies being used and the transmitted power.

There are no radio transmitters within the payload.

I. Other relevant downlink telemetry information.

None.

IV. Uplink Commanding Specifications:

A. Command uplink capability required: [✓ Yes] No (circle one)

B. If so, will commands be uplinked in regular intervals: Yes [✓ No] (circle one)

C. How many commands do you expect to uplink during the flight (can be an absolute number or a rate, i.e. *n commands per hour*)

Before Launch we intend to issue the power up command, the Flight-Line Test Command, and the Start Test Command. During flight, no commands should be necessary unless something goes wrong that requires human intervention. If that happens, there may be a burst of 6 commands per hour for two hours: we may issue a standby command, several test commands, and then a Start Test command again.

D. Provide a table of all of the commands that you will be uplinking to your payload

The payload will expect to receive Commands in the format defined in the HASP Interface Manual Table 4. The following characters will be received:

SOH, STX, Byte1, Byte2, ETX, CR, LF

where Byte1 and Byte2 define the command. It is a good idea to incorporate some data redundancy or checksum to verify the data. We have chosen simply to make Byte1 = Byte2. If a data error occurs and those two bytes are not the same, a "Command Response" will be sent indicating a data error.

We will keep our choice of command bytes within the printable ASCII set, as in Table V.

The software to interpret these commands is being written now, and we know additional commands will be necessary. This document will be updated when the list is complete.

Table V. Command Data Format

Byte1-Byte2	Command
00	Turn DC Power OFF
11	Turn DC Power ON
22	STOP Tests and Assume Standby State
33	Give Hardware Reset to Arduino in Test Bay
44	Start Testing (Exits Diagnostic Mode)
55	Resume Temperature Profile
66	Execute Bench Diagnostics (interactive, with terminal connected to serial port)
77	Execute Flight Line test
88	Launch (to activate processing if Launch was not automatically detected)
99	Request Status
AA	Set TBay Temperature to 35°C

BB	Set TBay Temperature to 10°C
CC	Set TBay Temperature to 0°C
DD	Set TBay Temperature to -10°C
EE	Set TBay Temperature to -20°C
FF	Set TBay Temperature to -30°C
GG	Set TBay Temperature to -40°C
HH	Set TBay Temperature to -50°C
II	Turn EBay Heaters off completely
JJ	Set EBay Temperature to 35°C
KK	Set EBay Temperature to 10°C
LL	Set EBay Temperature to -10°C
MM	Set Fail Code for Servo 1
NN	Set Fail Code for Servo 2
PP	Set Fail Code for Servo 3
QQ	Set Fail Code for Servo 4
RR	Set Fail Code for Servo 5
SS	Set Fail Code for Servo 6
TT	Set Fail Code for Servo 7
UU	Clear all Fail Codes
VV	Execute Servo Test once and report
WW	Establish Baseline for all Servos (saves Servo Data to be compared against tests)
XX	Discover Servo Limits and use but do not save
YY	Start Logging data to SD card (default)
ZZ	Stop Logging data onto SD card and close logfile
aa	Clear and reinitialize logfile
bb	Print Baseline Servo data to Serial Port (Ground Bench command only)
cc	Turn off EBay Heater Completely
dd	Unassigned

E. Are there any on-board receivers? If so, list the frequencies being used.

   There are no radio receivers within the payload.

F. Other relevant uplink commanding information.

   None.

V. Integration and Logistics

A. Date and Time of your arrival for integration:

July 31, 2016

B. Approximate amount of time required for integration:

We expect integration will take 2.5 hours, then about a day for temp/vacuum tests, and perhaps a day to repair and retest.

C. Name of the integration team leader:

Phil Carroll

D. Email address of the integration team leader:

pcarroll@my.bridgeport.edu

E. List **ALL** integration participants (first and last names) who will be present for integration with their email addresses:

Phillip Carroll: pcarroll@my.bridgeport.edu

Dr. Jani Macari Pallis: jpallis@bridgeport.edu

Lawrence Reed: reedla@comcast.net

F. Define a successful integration of your payload:

- Powering on and observing SYSTEM READY with no faults.
- Observation of Time Sync Message, if either GPS can get a fix (if indoors, this is bypassed)
- Successful bidirectional communications with HASP, including GPS data
- Heater Tests
- Successful operation of Servo test with good Baseline data
- Successful operation during the Temperature-Altitude tests, and replacement of any failed components.

G. List all expected integration steps:

1. Connect the Payload plate to the HASP, connect power and Discrete signals with the EDAC connector, and the serial DB9.
2. Issue the HASP command to power on the payload
3. Observe the SYSTEM READY message and the TIME SYNC message
4. Execute several commands as described in the Integration Test Procedure, including servo heater tests
5. Power Down and report to the Temperature-Altitude Chamber
6. Repeat tests without running the heater tests
7. Start the Chamber and Give the START TESTING command to the payload. Observe the data from the payload.
8. Replace Servos that failed during temp-altitude testing. Recalibrate unit with new servos and re-baseline their operation.
9. Re-run steps 1-4 above.

H. List all checks that will determine a successful integration:

1. The SYSTEM READY message is sent without any faults displayed. This indicates the Arduino can control all the heaters and all the servos, it can read the temperature and pressure sensors and servo feedback, and it has observed that the Arduino in the Test Bay is running its BLINK program. It has also checked voltages and found them to be within limits.
2. The Integration Test Procedure will verify that the Pressure Sensor, GPS, and all temperature sensors are working, that the Arduino in the Test Bay can be controlled, and that each of the servos can be moved, and its position read with analog feedback.
3. The Test Procedure will also verify proper communications with the HASP, and verify the HASP can reset the Control Processor and successfully send commands to it.
4. The Heater Tests will exercise all the heaters, measure the power they draw and watch the (correct) temperature sensor climb when the heater is turned on.
5. Once The Temperature-Altitude Chamber has started its profile and the START TESTING command has been sent, watch the reported data for anomalies. Verify the Temperature Profile within our payload can be maintained, regardless of chamber temperature.
6. Be ready to replace servos that failed during the Temperature-altitude cycle. This is expected.

I. List any additional LSU personnel support needed for a successful integration other than directly related to the HASP integration (i.e. lifting, moving equipment, hotel information/arrangements, any special delivery needs…):

   We do not expect we will require special assistance outside of the HASP integration.

J. List any LSU supplied equipment that may be needed for a successful integration:

   We will bring the equipment we require, but a general set of tools and soldering iron/solder would be appreciated.