Post Launch Assessment Review

2017-2018 High Altitude Student Payload Flight

Robotic Arm Manipulation and Materials Matching: RA(M^3) or RAM

Julie Hoover

James Acevedo, Seth Close, James Cowell, Daniel Daugherty, Soham Pai Kane, Daniel Koris, Meredith Murray

NORTH CAROLINA SPACE GRANT

Abstract

The need for satellite maintenance is growing as established satellites run out of fuel and degrade in the harsh environments of space. NASA's Satellite Servicing Projects Division (SSPD) is developing unmanned craft to provide a lower-cost, lower-risk method of extending legacy satellites' lifetimes. While only two active satellites are designed for maintenance, the next generation plans to leverage robotic refueling and maintenance.

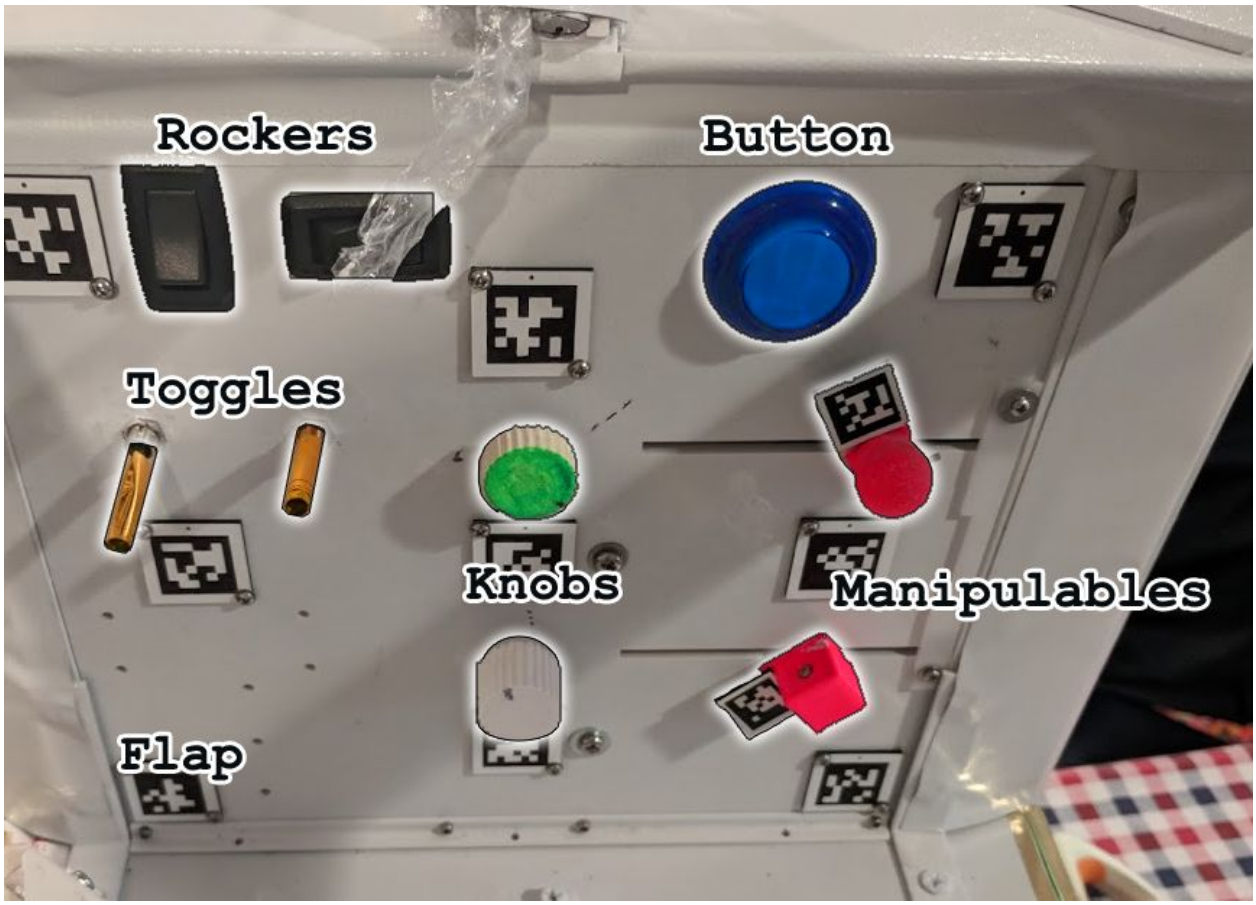
The HASP program is a balloon flight sponsored by LA Space Grant which allows student payloads to fly on a CSBF-managed balloon for 12-18 hours of sub-orbital flight at ~120 kft. HASP is traditionally launched from Ft. Sumner, NM during the fall flight campaign. Our experiment for HASP tested the limits of relatively low-powered, semi-autonomous robotic dexterity, running repeated iterations of tests that require precision actuation, computer vision, and force-torque sensors. A robotic arm performed simple tasks: toggling switches, twisting knobs, and opening/closing velcro flaps on a board studded with manipulable items (termed the "Busy Box.") Our experiment evaluated the impacts of extended use in extreme conditions. Orbital missions must contend with traditional lubricants evaporating in low-pressure environments, survive direct and uninterrupted sunlight, and control heating/cooling in the absence of convecting atmosphere. Over the duration of the flight we measured any degradation in performance, response time, and accuracy.

Scientific Goals

1. Control Systems: Kinematics & ROS

Our first goal was to develop and kinematically model an arm capable of the types of actions that our Busy Box required:

- pushing *(blue arcade button)*
- pushing against resistance *(black rocker toggle switches)*
- translating and rotating against resistance *(lengthened steel/brass toggles)*
- gripping a captive surface with jaws, twisting *(knobs)*
- gripping a hinged surface with jaws and translating *(flap [disconnected])*
- gripping a free-rotating target with jaws and translating against resistance *(manipulables)*



The flight-ready Busy Box. The flap, rockers, and manipulables were de-scoped from the 2018 flight.

The next step was to develop control software for manipulating the arm according to the developed kinematic models. The Robot Operating System (ROS) was an obvious framework choice to write our software on.

2. Fiducials & Computer Vision

Our second goal was to prove our fiducials, laser-etched from 2-ply black and white acrylic, through sustained IR and UV exposure in a near-space environment. We wanted to ensure that they would not fade, warp, or separate. We also wanted to ensure, by capturing images from the arm camera every 15 seconds, that the ROS computer vision libraries could consistently and reliably detect the fiducials through a number of lighting conditions. Any significant patterns in failed detections would be crucial in informing best-practices for future missions which rely more heavily on computer vision.

A set of vinyl-cut fiducials, from the AprilTag family 16h5 (left) and laser-etched 2-ply acrylic fiducials from AprilTag family 36h11 (right), both on laser-cut acrylic plates.

3. Servo Strength/Dexterity/Stamina

Our third goal was to test the performance of commercial off-the-shelf (CotS) components, i.e. servos, controllers, and cameras, and their ability to deliver consistent torque and precise orientation during sustained operations at float altitude. We were particularly interested in their thermal profiling and recorded temperature data inside the servos as well as at various other components and points on the hull. We were further interested in the performance of mechanical moving parts, with particular attention paid to bearings that traditionally need lubrication.

Design Constraints

To meet the stringent HASP design parameters within a tight timeline and with limited budget, we decided to emulate tried-and-true designs. Initially the team designed several arm configurations that theoretically would fit within the project footprint and would have enough articulation to manipulate components on a fixed vertical plane orthogonal to the arm base. We bought several commercial off-the-shelf (CotS) remote-controlled robotic arms to test operation and compare design strengths and weaknesses.. The team eventually narrowed the design that was a four degree-of-freedom (4DoF) system based on the simplicity of the kinematics and the overall footprint of the space allotted.

We based much of the design for the RAM arm base on the CotS arm due to its stability and ease of rotation. We constructed it out of milled 6061 aluminum plates that sandwiched a large-diameter, thin ball bearing (BC Precision VA025CP0) with mounting points above and below the base. The layered design and mounting configuration mitigated cantilever motion while the arm was in use which provided considerable stability and fatigue resistance.

We milled the upper and lower arm sections out of one-piece 6061 aluminum c-channel, with flat plate brackets at the elbow, and made the wrist from layered 6061 aluminum plating. We used the rigid, sturdy aluminum servo bodies as structural components, leveraging Robotis HK-101 brackets to mate Dynamixel hardware to our arm pieces. The gripper consisted of the servo body and a combination of 3D printed material and aluminum plating.

Final Payload Overview

Structural Design

We welded a structural frame out of 0.0625” aluminum C-channel uprights and diagonal-cut 0.125” one-inch aluminum angle. This provided a scaffolding that would protect the payload from all reasonable impacts. For drawings of this and more, please see our application, available on the HASP website under the 2018 flight payloads.

Demonstrating the structural integrity of our frame by putting our 100kg team lead atop it.

Into this we installed a middle shelf of 3/32” aluminum. This shelf acts as a mounting surface for the Busy Box and the arm, as well as a radiator panel for all our power-intensive components (RPi and voltage regulators.) The middle shelf was mounted to the frame with #4 bolts and brackets cut from the same 0.125” one-inch aluminum angle.

The Busy Box mounts to the middle shelf with 0.0625” two-inch-long galvanized steel L-brackets and #8 bolts, which in turn secure aluminum U-channel that encloses the 0.125” thick acrylic. We later ended up added a stiffener bar at the top, as the whole system proved too dynamic and was flexing backwards when buttons at the top were pressed.

Heat Regulation

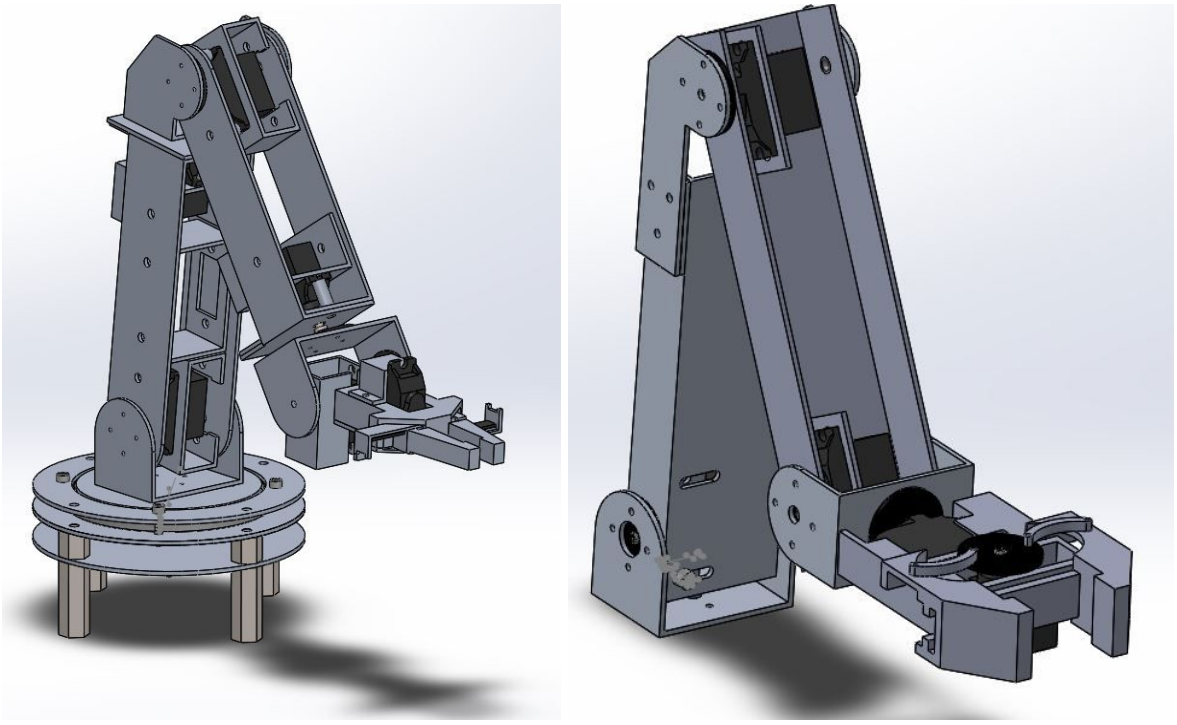
Overheating due to solar radiation load was a concern for flight. Each panel was painted white with high-gloss Rust-Oleum appliance epoxy, and surfaces that were more vulnerable to overpaint (like the anodized black aluminum servo bodies) were painted by hand with white Chip-Fix appliance touch-up. Where feasible, our wiring was bundled with gold mylar sheets and gold mylar tape. We also lined the interior of our electronics payload with a single layer of 5-mil silver mylar. We would have preferred to use multiple layers with tulle bridal netting spacers, but convenience and time prevented it.

Our voltage regulators generate a considerable amount of heat, so we designed aluminum mounting hardware which doubled as heatsinks, providing a thermal path to the middle shelf. We secured the voltage regulators to the heatsinks with epoxy and applied a layer of silicone thermal paste between the heatsinks and the plate. (Hasty additions of new voltage regulators at Integration did not allow us to manufacture heatsinks for the new components; only 3 of the 5 regulators flew with heatsinking.) Our RPi flew in an aluminum enclosure from Wicked Aluminum, tailor-made for the RPi 3 B+ and further customized to snake our CSI cable out. The RPi case was similarly mounted directly to the underside of the shelf with a layer of silicone thermal paste.

The only camera which survived multiple de-scopes, our Adafruit Raspberry Pi “Spy Camera”, was mounted to a small 3/16” aluminum heatsink block with thermal epoxy. The block was then friction-fitted into a 3D-printed white Kynar mounting bracket, which was in turn bolted to the outside of the Dynamixel servo housing using the extant threaded holes.

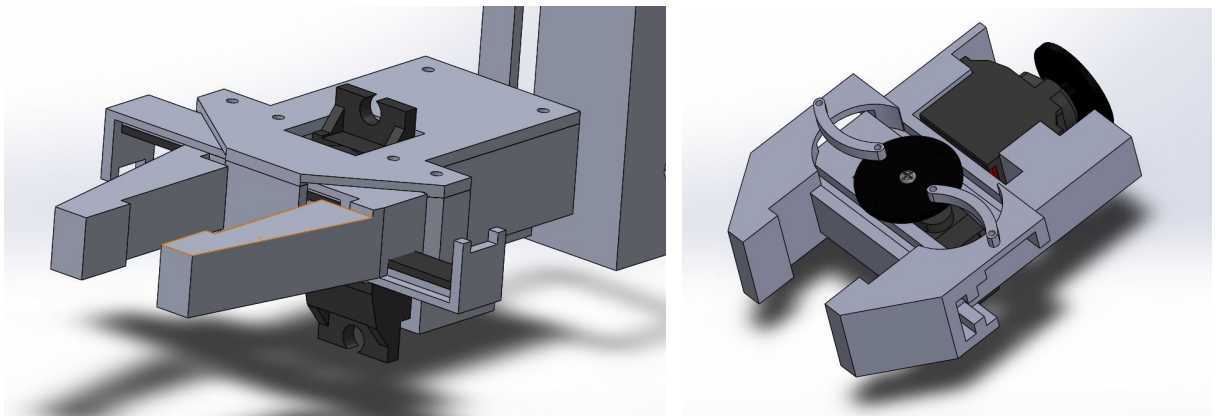
Hardware: Arm & Gripper

Our arm went through many revisions during the months of design. One of the largest engineering challenges was to design a gripper with few moving parts that was robust enough to withstand the harsh environments of near-space and strong enough to manipulate the various items on the busy box.

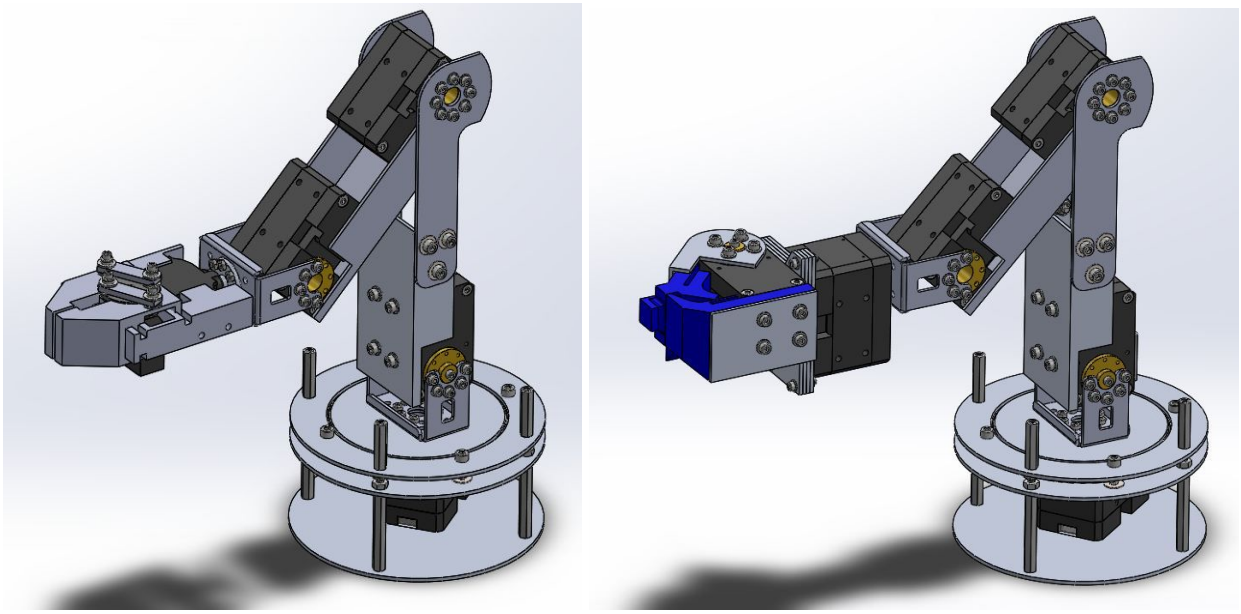


Early CAD versions of the arm; v1 (left) and v2 (right, shown without its turntable 'shoulder').

Our first design, v1, had a much more delicate end-effector and more complex internal structure. We quickly moved on to a second revision: v2 featured a broader gripping surface, which gave us more margin for error in the kinematics, and a simplified internal arm structure. The simplification allowed us to build the arm using mostly CotS C-channel with drilled holes rather than requiring a series of custom bent pieces of sheet metal which, for our manufacturing pipeline, would have introduced significant error-stacking.

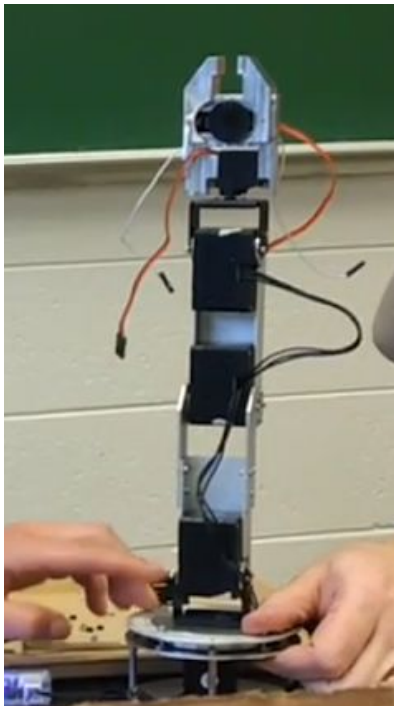


We eventually discarded the small Batan servos which we had positioned at the end of the arm, hoping to reduce the mass and thereby reduce the torque required at shoulder and elbow. Replacing them with the heavier, more expensive, and larger Dynamixel servos proved quite worthwhile. Not only was the interchangeability of flying 6 standardized servos very important, it vastly simplified the commanding challenges for our electrical and software teams.



CAD versions of the arm; v3 (left), and v4, our flight model (right).

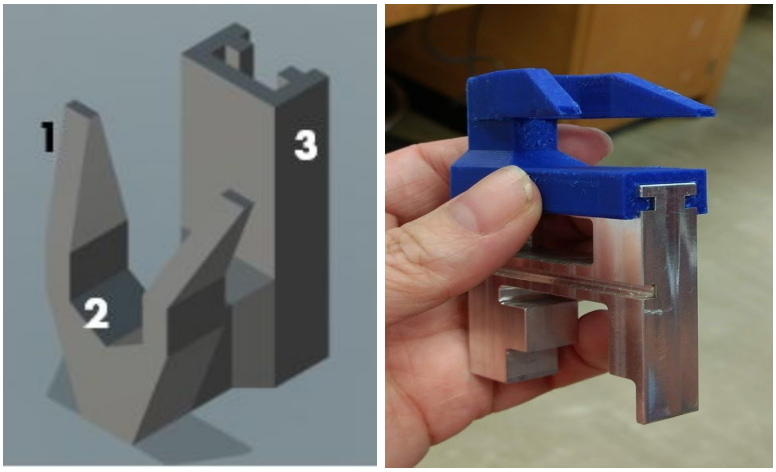
Versions 2 and 3 both featured small linkages: pieces of metal that connected diametrically-opposed bearings on the servo horn to the in-line track-mounted jaws. These linkages transformed rotary movement into linear motion, opening and closing the gripper jaws. In v2, the linkages were small, delicate, and curved; these were extremely difficult to manufacture. Even a simpler, larger linkage as shown in v3 (straight linkages can be seen in the picture above, contrasting the curved versions prior) proved tricky to make. Furthermore, the custom-milled aluminum blocks for the gripper base and the jaws were also very time-consuming and, even with fine machining and lubricants, would bind too often to be considered reliable.



The v3 prototype. Note the extra, separate wiring required for the different servos.

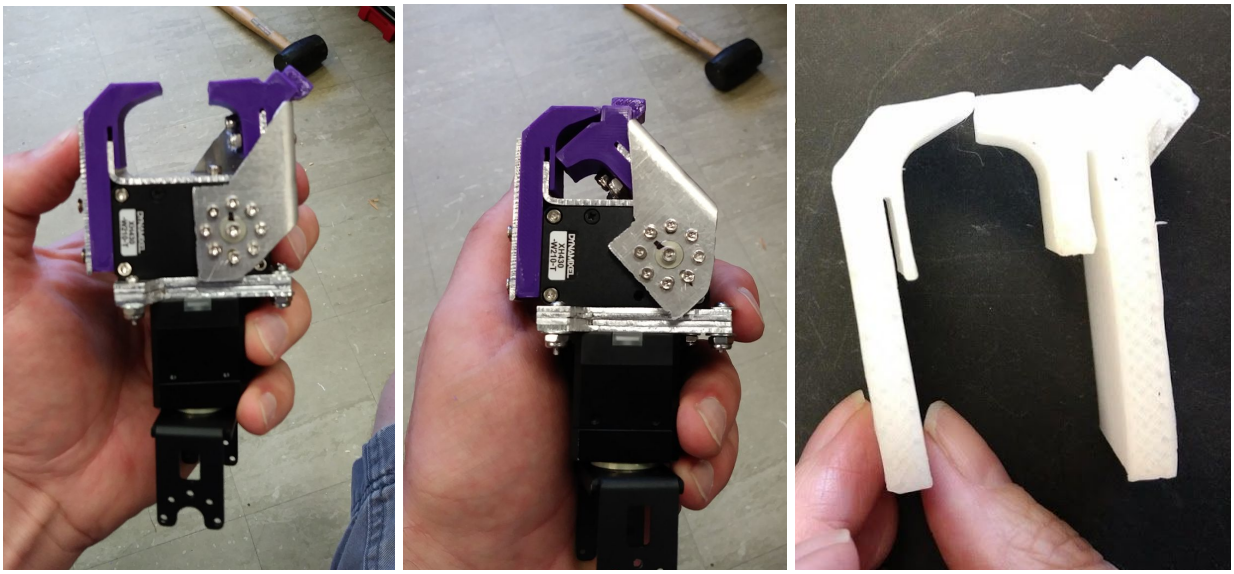
With these constraints in mind, and with mounting schedule pressure, we finally settled on the elegant, simple, and robust gripper design shown in v4.

After a particularly troubling series of failures when we had burned out our last remaining test backup, we designed a static gripper (or "fist") to replace the articulating designs. The fist, originally a worst-case de-scope, ended up being seriously considered for the final product due to the exigency and timing of the failures mentioned above. We designed the fist to address all the same tasks as the articulated gripper, without any moving parts. The canted, V-shaped horns (1, below) were designed as a self-centering feature to guide the open-ended hexagonal wrench (2, below) around the potentiometer knobs. The sliding rack (3, below) allowed it to mate easily with our current arm hardware. Finally, the fist was 3D-printable in a single run and required very little team time to modify or manufacture. We commend the mechanical team and PI with their insistence on spending the time to design it; having a fallback position available gave us some peace of mind during difficult times.



CAD of the static "fist" gripper (left) and the final product (right) mounted on hardware from an earlier iteration.

The final flight gripper incorporated aspects of a pincher-style gripper and the simplicity of the fist, using the Dynamixel servo as the main structure of the body. The gripper "jaws" were made with 3D-printed Kynar and reinforced with custom 6061 aluminum brackets. One side of the gripper remained static while the other was mounted to the servo wheel. The articulation of the wheel was such that the gripper could act as a pincher but could also fold into itself to create a fist-like static gripper that could manipulate most of the objects on the busy box without the need for articulation from the servo. A nub-like "knuckle" protrudes from the front of the end-effector when the gripper is in fist-mode, providing a precise pushing surface for targets like the button. The knuckle was designed to be in-line with the hand assembly such that it was centered around the revolute axis of the hand, eliminating the need for kinematic refactoring.



Final flight gripper design, shown in its 'closed' articulation-mode position (left) and its folded-up fist-mode position (middle). The final flight jaws, in white Kynar, are shown (right).

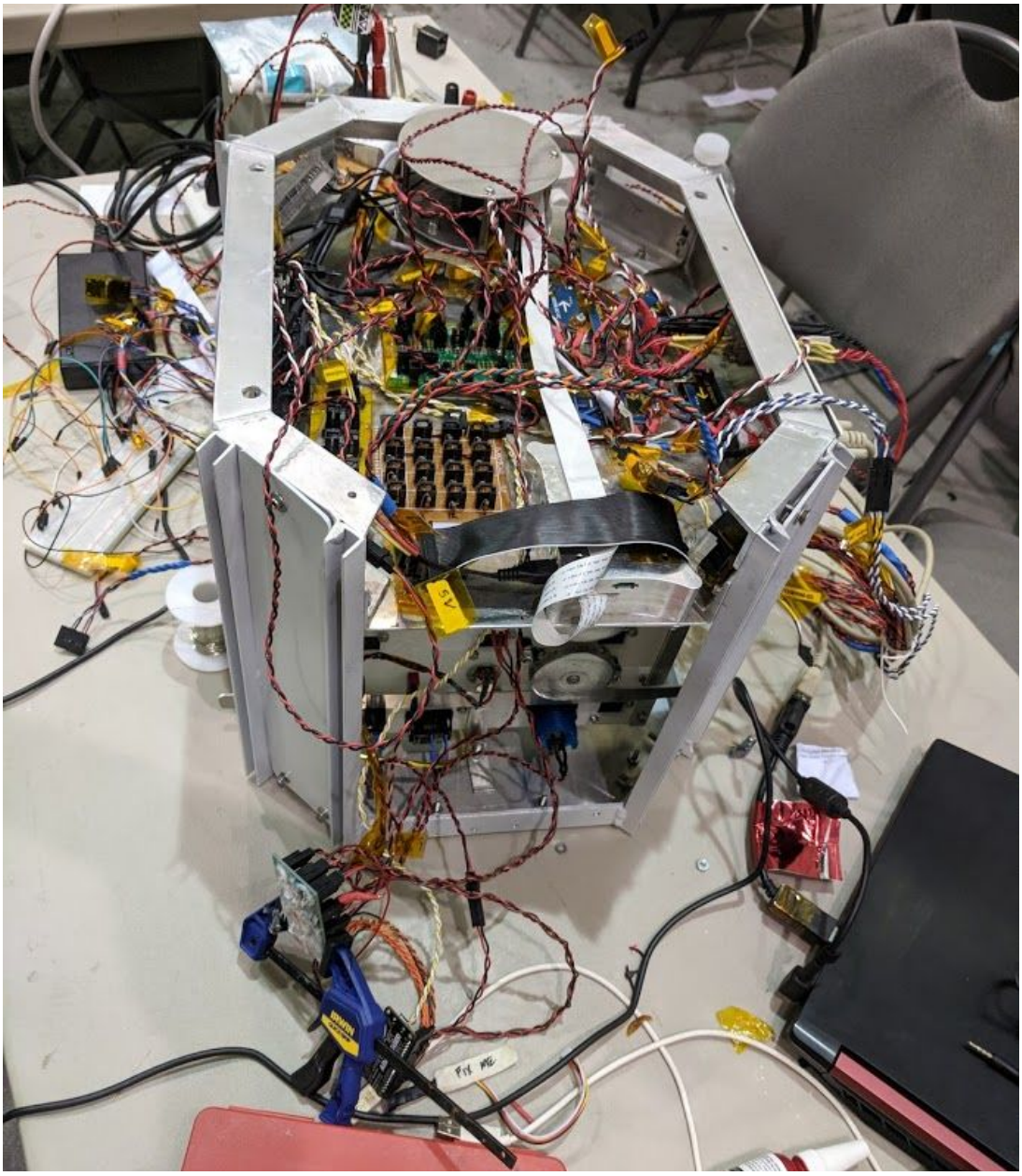
Hardware: 3D Printing

Our final flight plastic parts for the gripper were 3D-printed in Arkema PVDF (polyvinylidene fluoride), or Kynar. Kynar is a UV-resistant material with a higher plastic temperature. We had concerns about PLA, a common 3D-printing material, deforming under the intense heat that we sometimes measured in our servos, and we were concerned about ABS (another common printing plastic) going brittle under the UV exposure as described in reference materials. For an 18-hour flight, this was likely over-cautious, but we pursued other material options regardless. Due to the thermoelasticity of the plastic, it was very difficult to print with, shrinking slightly as it cooled and causing the plastic to frequently warp and pull off the build plate. We were only able to print the final parts after much pre-processing (tweaking the CAD and the printer settings) and post-processing (filing and careful detachment.) All other non-mission-critical 3D-printed parts were printed in ABS for ease.

3D-printing was a very helpful tool, providing much of the precision of custom machining without the cost, weight, and time. We used it to make a mount for LED lights, keeping the Busy Box illuminated after dark, for custom-fitted camera mounts cameras, solar shielding for the servos (which doubled as cable routing channels), and many of the knobs and other surfaces for the Busy Box. This was a excellent way to be able to test and prototype many different parts before we used them.

Hardware: Interior Payload Configuration

We shielded most components from EMI and solar load by enclosing them inside the "electronics basement" under the midplate or behind the Busy Box. As mentioned above, power-intensive components were mounted to the underside of the midplate. Though we had initially mounted other components to a Delrin plate which could be detached from the bottom of the frame (and would be sandwiched between the frame and PVC mounting plate) this proved to be more awkward than it was worth; we eventually mounted everything to the underside of the frame. Future iterations of RAM should follow this model, i.e. all electronics mounted to a single surface which is mechanically grounded. Future iterations of RAM should also consider electronics-bay access which does not require un-mounting the entire frame.

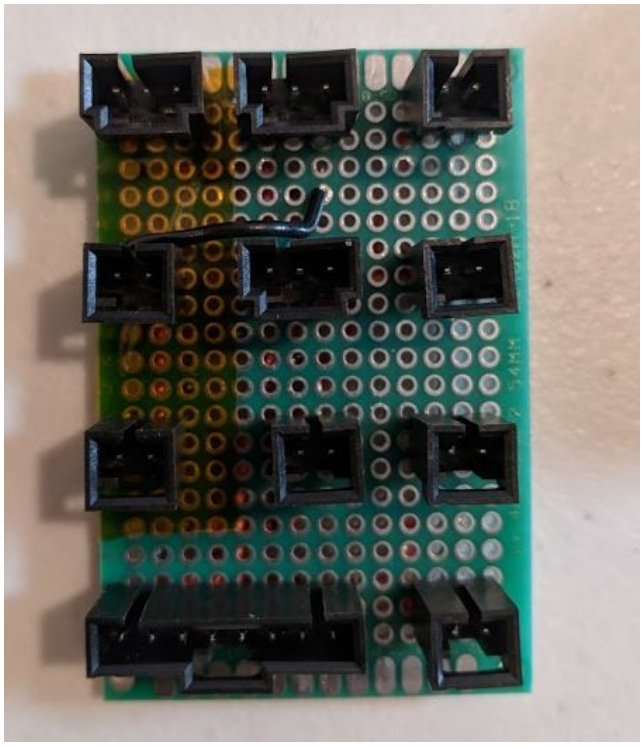


The RAM wiring configuration, mid-maintenance, during Palestine integration. This picture included as a warning parable about ease-of-access when choosing where and how to mount electronics.

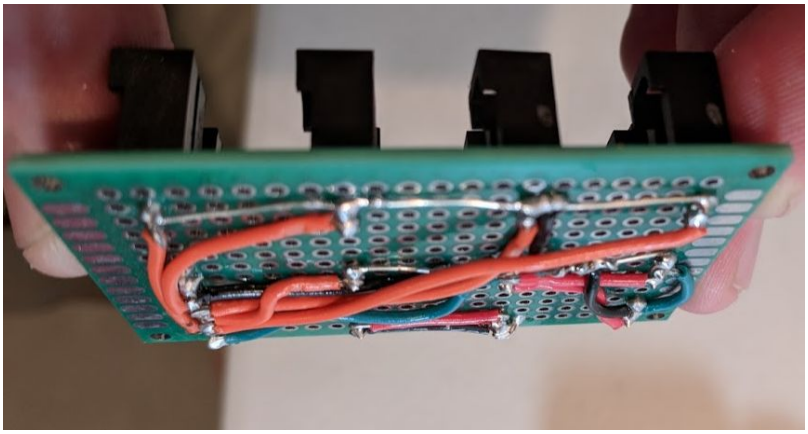
Where possible, similar electronics components were grouped together onto bus-boards. We built a board for up to 16 Dallas thermometers to plug into, unifying their data and power lines to eliminate clutter. We also made a bus-board for I^2C connections and for Busy Box components. We made our boards by hand with jumpers, Molex connectors, and wiring harnesses; future iterations of RAM should consider having custom circuit boards printed.

Based on our experiences with the HASP 2017 flight (where almost all connections were direct-soldered or used screw terminals) we standardized to a single connector family, the Molex

SL latching connectors (e.g. Molex P/Ns 50-57-9402, 16-02-0102, 70-10-70001, 16-02-0114, 70-54-30036.) Almost all boards were fitted with pinned headers, using socketed receptacles on both ends of the connecting cables. We occasionally extended connections with pin-to-socket cables. This gave us significant modularity and, though it required more initial work to install all the connectors, absolutely was worth the time investment, especially during time-sensitive repairs. We weighed the risk of extra points of failure which came from adding potential breaks in the circuits, though **over the course of the entire project we never found a single bad connection** (even though we used the cheaper "service-grade" crimp tool (63811-1000) rather than the more expensive manufacturer-recommended crimp tool.)



Molex SL-series headers assembled onto a "bus board" for routing power and data around the Busy Box with modular cables.



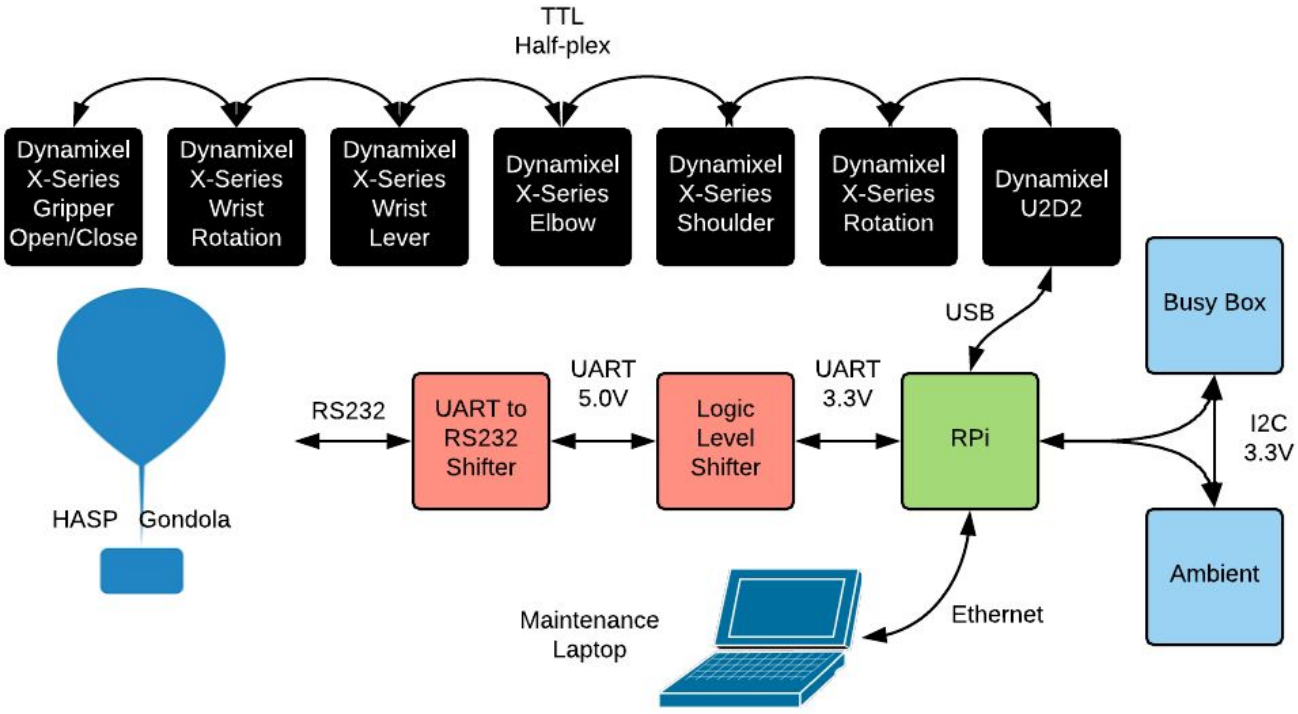
Hardware: Electronics

Item	Quantity	Purpose
Raspberry Pi	**1**	The RPi was the heart and soul of the entire project. It controlled data flow through the network and arm movement. The RPi was chosen because it could run ROS and had the processing power to manage all the devices of the sub-systems (cameras, servos, Arduinos, network management).
Arduino Feather	**2**	Arduino Feather #1: - monitored all Dallas Thermometers - monitored all BME280s - acted as a clock synchronizer for the whole payload. Arduino Feather #2: - controlled Busy Box - monitored the data from all the switches, button, and potentiometer - reported Busy Box values to the RPi on request. Arduino Feathers were chosen for their small size, low power requirements, available pin-out, and onboard SD storage which allowed them to backup all their own data.
Dynamixel X-Series Servos	**6**	The Dynamixel X-Series Servos comprised part of the arm itself, including the gripper. These servos were selected because of their overall high quality of design and capabilities. They fit within our power requirements and were capable of outputting the torque required to perform all kinematic tasks. In addition to that, they were able to report on their own internal temperature. This was a huge bonus since the team then did not have to run individual temperature sensors per servo. The ability to "daisy-chain" them in a single, serialized 3-wire power and data spine which could use the manufacturer's cables and ports also allowed us to cut down on wiring.
Sparkfun RS232 Shifter	**1**	The Sparkfun shifter was used to convert our 5V TTL UART signals into RS232 UART signals. It was chosen because it was a known, robust product with flight legacy on tUR.
Sparkfun Logic Level Shifter	**1**	The Sparkfun logic level shifter was used to shift the RPi's 3.3V logic level to a 5V logic level required for RS232.
BME280	**2**	The Adafruit BME280 utilizes a robust Bosch temperature chip that gives temperature, pressure, and humidity readings. These were chosen because they are known good sensors with flight legacy on tUR.
Dallas Thermometers	**15**	We selected Dallas DS18B20 temperature sensors for their small form factor, ease of integration, and overall simplicity (at the tradeoff of their coarse granularity.) Furthermore, using the OneWire library, we were able to compact their wiring down into 4 channels, cutting down further on the amount of soldering required.
Toggle Switch	**2**	These Busy Box switches provided a unique kinematic motion goal for the arm. We lengthened them by crimping fine-diameter brass tubing over top of the steel toggle arm. This

		gave the arm more leverage; though the arm was able to flip the short versions, the act required more torque than the servos were "comfortable" putting out over a period of time. Additionally, lengthening the switches gave us a more forgiving target for our first project; we suggest tightening the tolerances for future iterations.
Button	**1**	The Adafruit 5cm arcade button, poetically dubbed "Blue Button," provided a unique kinematic motion goal for the arm to reach. It functioned well through all thermal vacuum tests and was chosen for flight for its simplicity of installation and operation.
Potentiometers	**2**	We salvaged potentiometers from old decade boxes, then 3D-printed knobs of two different lengths and adhered them to the potentiometers with epoxy. The different lengths allowed us to test the precision of the gripping motion with two different levels of gripping difficulty.

Network: Overview

RAM contained four different networks and protocols in total. These networks served to allow for communication between the HASP gondola and the Payload, the Payload and its various subsystems, and a maintenance port that allowed the team to connect a laptop to easily access the Raspberry Pi (RPi). The total network consisted of one RPi, two Arduino feathers, and six Dynamixel X-Series servos.



Network: RS232

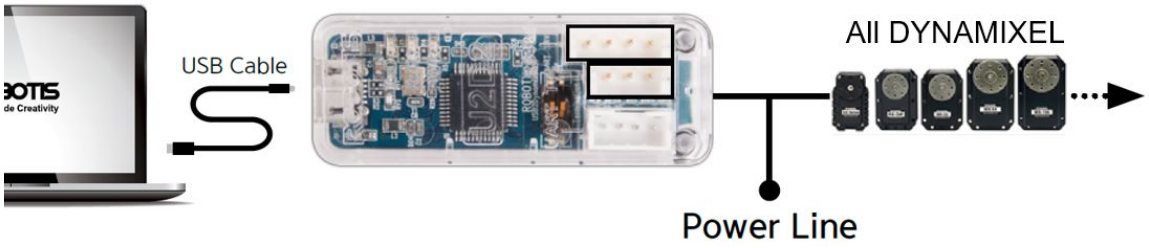
The network connected the RPi and the HASP gondola, relaying GPS telemetry and commands from the ground team and sending our data from the payload to the ground. The RPi cannot support RS232 natively, as it runs a standard UART at a 3.3V logic level for Serial communication. We shifted the serial lines coming out of the RPi to a 5.0V logic level and then routed them into a Sparkfun UART to RS232 converter. From there, the data went into the HASP gondola and communicated at a 4800 baud rate with standard protocol of 8N1.

Network: I^2C

The payload consisted of three total microcontrollers. The original design called for four, but RAM's power system was having trouble producing enough power for two RPis and we needed to remove one. Since the RS232 network was our primary means of sending commands to and receiving telemetry from the payload, one microcontroller was nominated as a network master. The RPi was selected for this task and acted as the guardian and manager of other microcontrollers. Any commands received over the RS232 line that were meant for other microcontrollers were routed by the RPi to the specific microcontroller via the I^2C network. The I^2C also allowed the RPi to query the subsequent microcontrollers for their telemetry: Busy Box state, temperature, pressure, and clock synchronization.

Network: USB & TTL Halfplex

In order to control the servos, the RPi communicated with a proprietary Dynamixel controller called the U2D2. The U2D2 is a network adaptor of sorts that converts signals sent via USB into a TTL Halfplex that the Dynamixel servos use to communicate. The RPi communicated with the laptop at a 50k baud rate; the U2D2 controller then converted those signals to a TTL halfplex and sent them down the chain of Dynamixel servos. This network is a single-link, which, on the plus side, greatly reduces the amount of wiring needed. On the other hand, if any element in the network fails, everything after it (assuming it starts from the RPi) will no longer be reachable.



Network: Ethernet

The final network protocol was a maintenance protocol and was not used during flight. However, it was a part of the original design. Initially, the two RPis would have communicated using the ROS framework over ethernet. Once the second RPi was removed, that ethernet port was co-opted to serve as a maintenance port. The team could connect a laptop easily right to the RPi and use SSH to log into the RPi to perform any code maintenance. This ended up being highly useful.

Software: Busy Box & Ambient

We made the software for the Busy Box and the Ambient systems using Arduino. The designs of the programs themselves were simple, robust, linear programs. Each held their own internal register of data from their various sensors. On each loop iteration, the program read sensors and updated the registers. Upon request, each Arduino would share their entire registers over I^2C with the RPi Master.

The Ambient Arduino had one special task that it was responsible for. The RPi shared, once a minute over I^2C, the GPS pulse received from the HASP gondola. It would then synchronize its internal millisecond-based timer, and share it back to the RPi master which then propagated the time out to all other subsystems. This allowed the whole system to be synchronized as best as possible.

Software: RPi Master & Arm Control

The RPi Master was responsible for managing the entire network, calculating the forward and inverse kinematics, and driving the control systems for the arm servos. To do this, the software was written on top of the ROS framework. There are many advantages to using ROS. ROS allows the designers to safely and easily take advantage of multi-core processors, it provides a way for multiple devices of any operating system to communicate, and offers many pre-built nodes that can seamlessly run alongside custom-built nodes.

Our software team built six different custom nodes: Arm Motion Driver, Arm Kinematics, Arm Trial, Path Planning, Ram Network, and the Cam Monitor:

- **Arm Motion Driver:** This node played a vital role for the payload. It employed the ROS Dynamixel Workbench library to control our Dynamixel X-Series servos. We used this library to save development time. The library itself did not utilize all the functionality capable of the servos themselves, but it allowed enough for our purposes and let us focus our energy elsewhere. Using the library, the node read servo positions and updated desired positions. The servo states were published globally on the ROS network for other nodes to use as needed.

- **Arm Kinematics:** This node was responsible for monitoring the servo positions that were published by the **Arm Motion Driver** node. As soon as a fresh batch of servo positions was published, the node performed forward kinematics on them and published the forward kinematics over the ROS network for all nodes interested. In addition, it offered a service for doing inverse kinematics for any node that needed them.

- **Arm Trial:** This node was responsible for performing trials that were commanded by the ground team. Each trial was broken into a series of motions. Each motion was planned through a ROS service call to the **Path Planning** node. The **Path Planning** node returned a series of Cartesian (*xyz*) coordinates. These coordinates were then fed as an action to the **Arm Motion Driver**. When the motion was completed, the **Arm Motion Driver** reported complete and the Trial then proceeded to the next motion until all motions in the trial were complete.

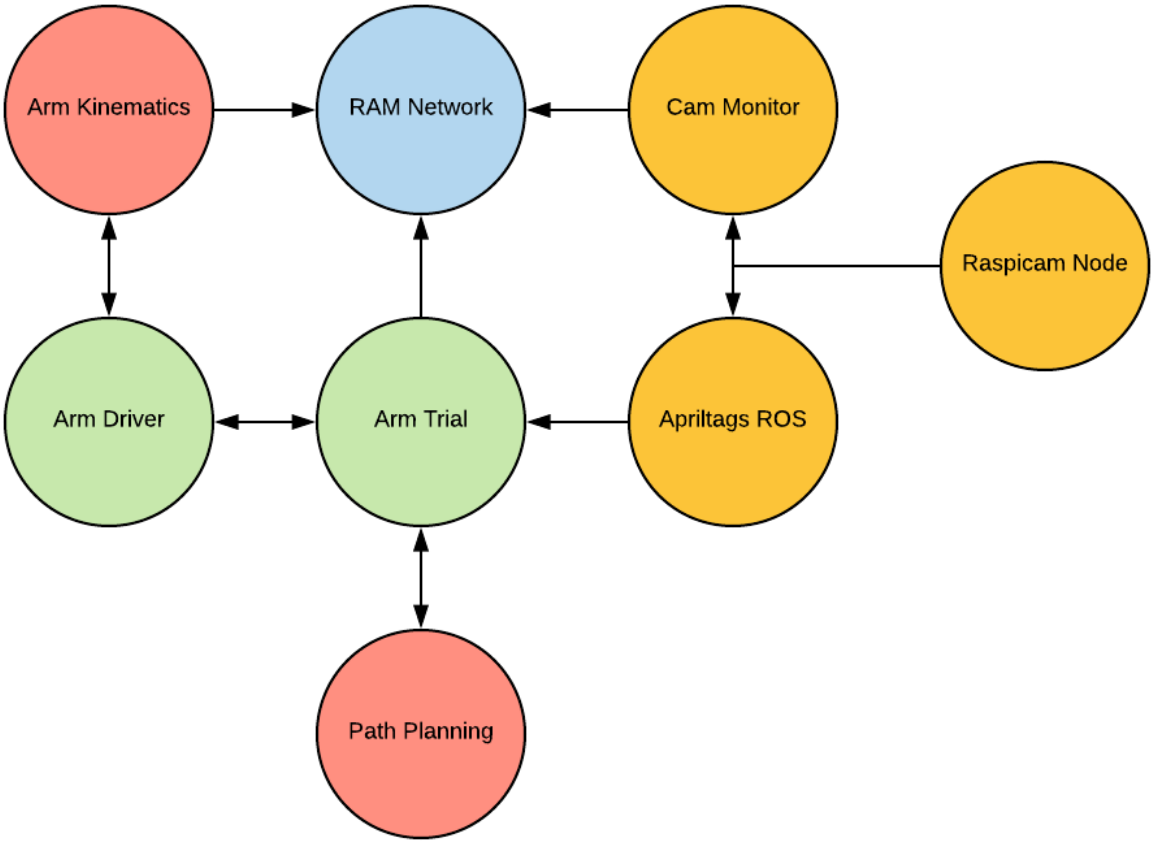
- **Path Planning:** This node has one simple function: offer a ROS service to generate a series of *xyz* coordinates for a single motion. The number of *xyz* coordinates created depended on the desired precision of the motion. The starting location, shape of motion, precision, and ending location were all configurable through the service.

- **Ram Network:** This node managed all communications over the network. It was responsible for communicating with the payload and its subsystems: the Arduinos over I^2C, the HASP Gondola over RS232. It was also responsible for collating the data gathered from the Arduinos, the Arm Trial node, images from the Cam Monitor system, and the servo states and downlinking them all to the ground. In addition, it distributed commands received from the ground to the proper subsystems.

- **Cam Monitor**: This node monitored the prebuilt nodes that were operating the camera systems. It sat above these nodes and subscribed to their output and saving the images recorded to the RPi's internal storage. It would also share its most recent capture when requested.

For prebuilt nodes, the software team relied on two nodes: AprilTags ROS and Raspicam Node:

- **AprilTags ROS:** This node subscribes to a given image stream and processes images as they are published, looking for AprilTags, and then posting all information gathered from the image.
- **Raspicam Node:** Raspicam node is a node specifically for use with RPi cameras. It is highly configurable and publishes an image stream that the **AprilTags ROS** node was able to subscribe to.



Kinematics

The original plan was to use ROS MoveIt! to handle the arm's kinematics. However, the barrier for entry proved harder than doing the kinematics by hand. While MoveIt! has many strengths for more complex robots and more kinematically-complex environments, it was ultimately not worth the effort for RAM.

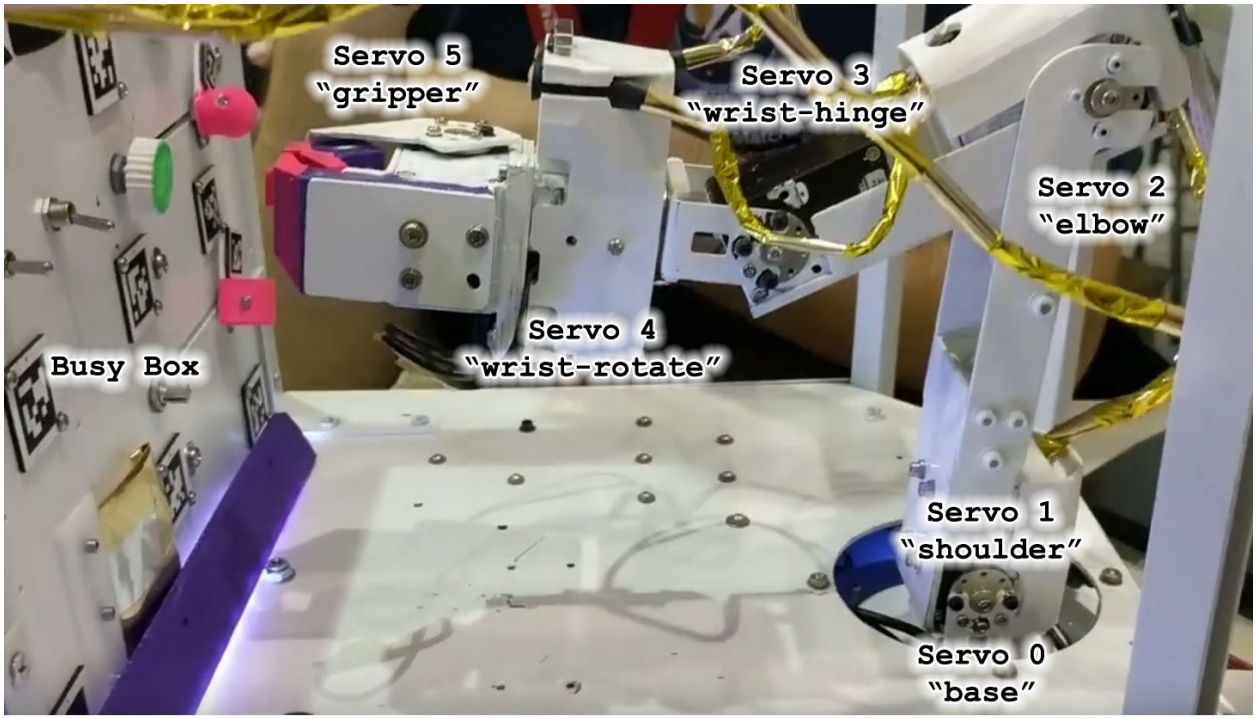
Forward kinematics answers the question "if we know the joint angles, what is the Cartesian (*xyz*) position of our end effector?" Forward kinematics is a simple task once the arm has been modeled with the Denavit-Hartenberg (D-H) Parameters. The D-H Parameters are a set of numbers that relate one joint position to the next joint position in the chain of servos. Once these are known, the user can populate homogeneous transformation matrices with the values at a given instant. Once the matrices are populated, they are multiplied together in order from base to end-effector tip and the result is a final transformation matrix. The end *xyz* position can be pulled as a column vector from the fourth column of the final matrix.

Joint (n)	Θ	α	R	D
0-1	Joint Position	90°	0	2.60 cm
1-2	Joint Position	0°	15.46 cm	0
2-3	Joint Position	0°	8.93 cm	0
3-4	Joint Position	0°	13.00 cm	0

The D-H Parameters for RAM.

$$\left[\begin{array}{ccc|c} \cos\theta_n & -\sin\theta_n\cos\alpha_n & \sin\theta_n\sin\alpha_n & r_n\cos\theta_n \\ \sin\theta_n & \cos\theta_n\cos\alpha_n & -\cos\theta_n\sin\alpha_n & r_n\sin\theta_n \\ 0 & \sin\alpha_n & \cos\alpha_n & d_n \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

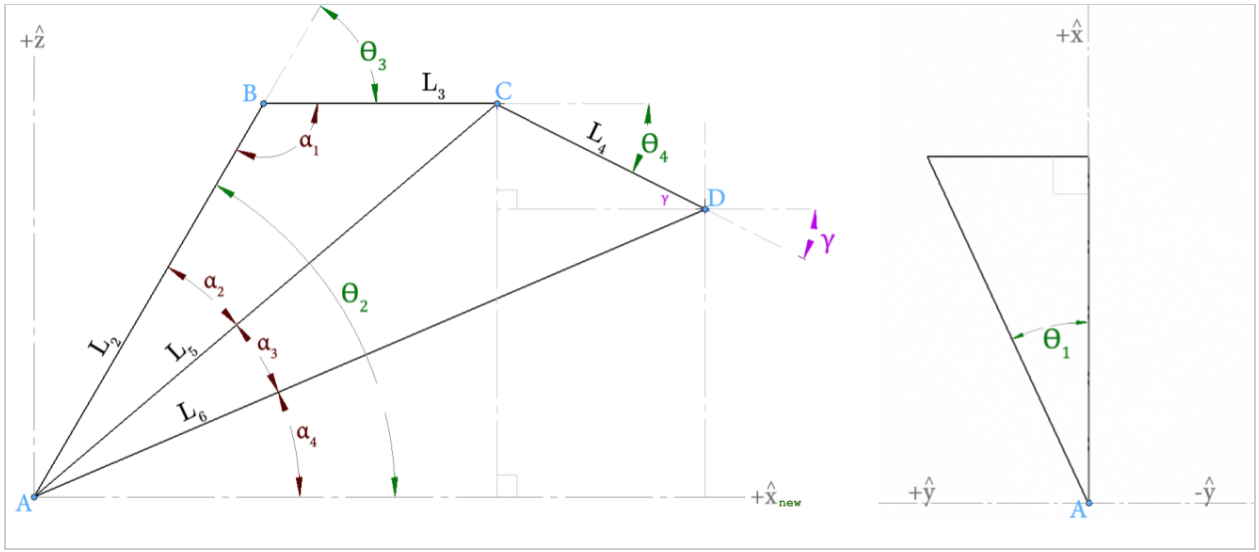
A standard D-H parameter transformation matrix.



The six servos in RAM. Kinematic calculations were only needed for the first four joints. For most kinematic intents and purposes, the entire arm assembly above the fourth servo ("servo 3") is a single, more-or-less static stump.

Calculating inverse kinematics is a far more complicated procedure: if one wants to know which joint angles will yield a particular *xyz* position for the end effector (EE), what joint positions are needed? The above-outlined **forward** kinematics method will work for any robotic

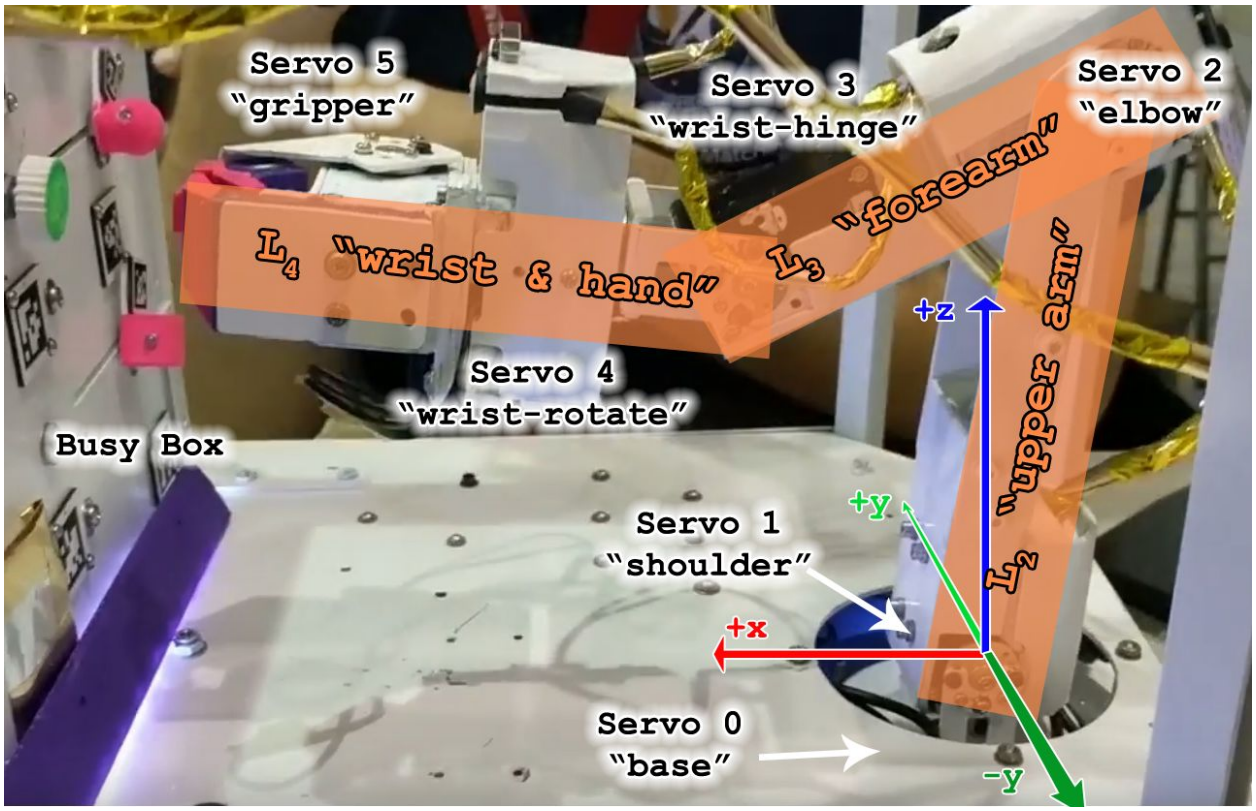
arm of any design, but there is no single technique that covers **inverse** kinematics for all arms. There are computational brute-force methods, but, for a 4DoF arm like RAM, it was easy to use a graphical method. This technique draws a representation of the arm in its xz and $x_{(new)}z$ plane, like a 2D graph. From there, one can fix one angle in the set of the equations.



Inverse Kinematics Graphs. In this graph, consider L_2 as the shoulder-elbow ('upper arm'), L_3 elbow-wrist ('forearm'), L_4 as wrist-to-fingertips ('hand') as illustrated below.

Given	Equations
Desired X Position: x	$L_5 = \sqrt{(\sqrt{x^2+y^2} - L_4 \times cos(\gamma))^2 + (z + L_4 \times sin((-1)\gamma) - L_1)^2}$
Desired Y Position: y	$L_6 = \sqrt{(z-L_1)^2 + (x^2+y^2)}$
Desired Z Position: z	$\alpha_1 = cos^{-1}((L_5{}^2 - L_2{}^2 - L^2{}_3) / ((-2) \times L_2 \times L_3))$
Desired EE Position: $\gamma = 0^\circ$	$\alpha_2 = cos^{-1}((L_3{}^2 - L_5{}^2 - L^2{}_2) / ((-2) \times L_5 \times L_2))$
Length 1: L_1=2.6cm	$\alpha_3 = cos^{-1}((L_4{}^2 - L_5{}^2 - L^2{}_6) / ((-2) \times L_5 \times L_6))$
Length 2: L_2=15.465cm	$\alpha_4 = tan^{-1}((z-L_1) / (\sqrt{x^2+y^2}))$
Length 3: L_3=8.927cm	$\theta_1 = tan^{-1}(y / x)$
Length 4: L_4= 13.00cm	$\theta_2 = \alpha_2 + \alpha_3 + \alpha_4$
	$\theta_3 = (-1) \times (\pi - \alpha_1)$
	$\theta_4 = \gamma - \theta_2 - \theta_3$

We decided to fix the orientation of our EE to be parallel to the xy-plane (parallel to the “floor"). We made this decision, an otherwise arbitrary constraint, as a common-sense way of maximizing our gripper’s chances of having the best grip interface with the Busy Box and the least likelihood of snagging on other elements of the Busy Box during any given movement. From there, plotting arm movement was a matter of detailed, thorough, and painstaking analytical trigonometry to derive the joint-angle of each servo for any desired *xyz* and EE position.



Electrical: Power

To power the whole system, the module was connected to 30V, 2.5A DC power supply provided by the HASP platform via an EDAC 560 plug. We installed a cascade of buck-converter regulators to step the voltage down. We built two power buses (12V and 5V) and connected all devices into the bus with standardized Molex SL-series 2-pin latching connectors.

Originally, we had a single-chain cascade of three regulators ([**1x**] 30->20, [**1x**] 20->12, [**1x**] 12->5) but we discovered a bottlenecking issue during testing. To ensure we had the capacity to deliver enough power, especially the high amounts required by RPis during boot-up, we added and re-distributed to the configuration shown above: ([**1x**] 30->20, [**3x**] 20->5, [**1x**] 30->12.)

Additionally, we installed Adafruit PowerBoost 1000C power supplies in line with our RPis to ensure they had enough power available during boot-up. We initially had two RPis, each with their own PowerBoost; however, even with the individual PSUs, we were seeing fatal brownouts on boot-up and needed to remove one RPi late in development. Future iterations of RAM should use a fully-provisioned PSU (other teams have used e.g. a WinSystems PPM-DC-ATX-P.)

During testing, we ran our power through a F5011-ND 2.5A fuse and opened our benchtop power supply "wide-open" to ensure we allowed for and could detect excessive current draw (in the past we have made the mistake of restricting our power draw to exactly the allowed ceiling, which hid over-draw errors from ourselves.) Additionally, we attached an oscilloscope across various points of the power spine during boot-up and arm operation to ensure we detected any excessive current draw in the servos. Our results from these over-draw tests were suspiciously quiet; we were concerned that we had configured something incorrectly for quite some time. Eventually, we confirmed that the even power draw reflected the situation accurately; our expensive servos do an excellent job of evening out current demands with their internal power supplies and we never were in any danger of blowing our fuse.

Electrical: Data

Our data connects to HASP via a DB9 connector wired to the RS232 convention. Inside HASP, our DB9 connects to a Sparkfun RS232 to serial shifter. The shifter then connects to a Sparkfun bi-directional 5V to 3.3V logic level converter, as our shifter expects 5V but the RPi and Arduino Feather M0 output at a logic level of 3.3V.

The logic level converter conveys serial communication from the shifter to the GPIO pins on the RPi which acts as the network 'master'. The RPi receives all the information from the BusyBox Arduino, the ambient Arduino ('Ada'), and the arm. The BusyBox Arduino only monitors the switches, buttons, and other components on the Busy Box. The ambient Arduino monitors our Dallas temperature sensors and our BME280 atmospheric sensors.

Testing

We expected to encounter a wide range of temperatures during flight, anywhere from the upper atmosphere's average temperature of -55 °C to the 55 °C that could be reached when HASP lands and rests in the Arizona desert. Furthermore, the lack of atmosphere at float altitudes prevents standard conduction- and convection-based cooling, causing dangerous buildup of heat in spaceflight hardware. We sought to test RAM and its components in similar environments using the tools available to us. We scheduled three steps of testing:

1. feasibility and preliminary endurance tests at DTCC,
2. "mini-integration" endurance tests at the NASA Langley, and

3. penultimate endurance tests at HASP integration at CSBF in Texas shortly before launch.

Last year we determined that 50 °C was a reasonable testing max temperature. At Durham Tech we use a Forma CO_2 Incubator for elevated temperature tests in order to prove that our components would still function properly after being exposed to high temperatures. Deep freeze tests were also conducted at Durham Tech using a Revco Lab Freezer on any new components. We tested them down to -80 °C. These tests showed that our servos would power on successfully once we reached float. Our standard thermal vacuum testing is still limited by our $300 vacuum chamber and Harbor Freight pump.

Testing: Preliminary Servo Evaluation

Early testing with inexpensive Batan servos and some of the lower-end Dynamixel offerings proved them unequal to the task. The Batan servos added complexity for wiring and command and offered less precision. The AX- and MX-series Dynamixel servos did not survive endurance-oriented vacuum testing, perhaps due to their diminished heatsinking capacity through their plastic casings. Ultimately, we determined that a few different versions of the Dynamixel X-series, with aluminum casings, met our needs best for both vacuum performance and the projected torque required of them.

Testing: Kinematics and Control Software

Testing the kinematics was simplest starting with forward kinematics. By using easy angles, 0° or 90°, it was possible to input joint position such that the human eye could easily verify the Cartesian-coordinate position. Through this technique, we could observe that our forward kinematics were correct in several situations which represented the most common orientations for the arm.

With tested, verified forward kinematics, we could then insert random joint angles and produce awkward, or edge-case, *xyz* and EE positions. With that done, the resulting generated *xyz* and EE positions could be fed back into the inverse kinematic equations. If the joint angles it generated matched the ones from forward kinematics, we could demonstrate our inverse kinematics were working properly. Repeating this process with several sets of joint angles, it became clear that both forward and inverse kinematics were correct and robust.

Even with proofed equations, not all theoretically-possible arm positions actually translated to positions which were actually-possible for our physical arm. For example, a given joint might not be able to rotate to 5° due to joint position limitations. To avoid commanding the arm to impossible positions, we hard-coded in positional limits. To determine the real-world, practical limits of each joint, we used the Dynamixel control software to temporarily place the arm in untorqued mode, manually rotating the joints to their various minima and maxima, and recording the positions.

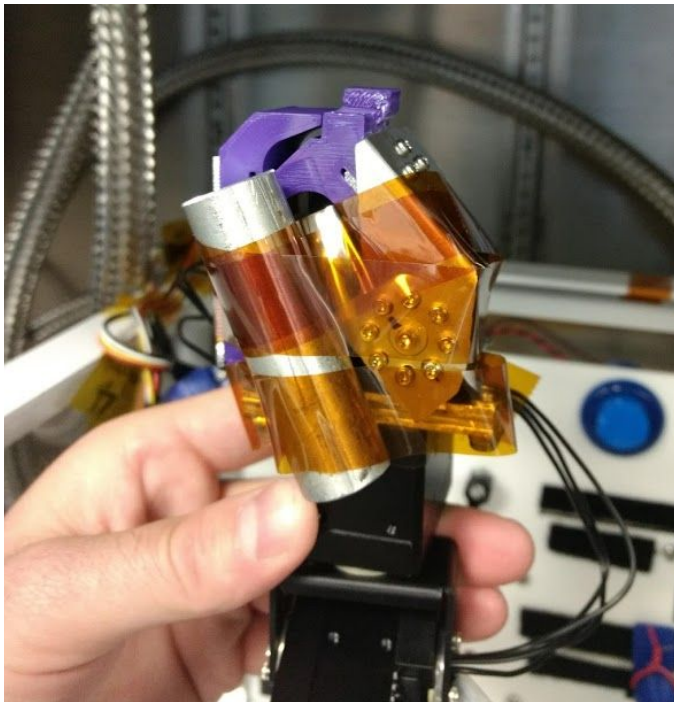
With these hard minima and maxima established, we could then adjust the code to reject kinematic positions that were impossible or risky for the arm. Should the arm ever enter these positions, we coded in a fallback condition to — if the invalid positions were part of a path — discard them, or — if the invalid positions were an endpoint — return to "home".

Testing: LaRC Vacuum

In July of 2018, we were invited to use one of the small vacuum chambers at NASA Langley Research to stress-test the payload at both extremes of cold and heat, in preparation for our HASP-required thermal/vac test. These tests lasted for a total of 12 hours over the span of two days. Before running the tests, the team spent a day installing and configuring an array of temperature sensors in the chamber and on the payload.

Our primary goal of the test was boolean: would the payload survive during a full-systems run? We were testing the thermal, electrical, and mechanical durability of our systems under load and at the extremes of anticipated flight temperatures. As a secondary test goal, we observed and videographed the arm during operation to identify any anomalous behavior. The arm passed all of the tests with no issues and no sudden power drops. However, we did observe a constant tremble in the arm. The team had to adjust the PID values; the differential constant was far too high, causing the arm to be sensitive to over- and under-shooting its goal position. Tuning this in the servo configuration files resolved the shaking.

The payload was configured differently for each individual test. For the vacuum chamber, the temperature was the only variable that changed. For the first day, the chamber was cooled to -55 °C, with the following day's test at +55 °C. We shaped custom steel hardware to clip our payload frame securely to the 'cold plate', the chilled aluminum plate on the bottom of the chamber.



State-of-the-art NASA mounting hardware holding a dummy 50g weight on our flight gripper for the Langley test.

For the cold test, we ran the payload with no thermal paste between the plate and the chamber. All were cameras mounted but not running, and we'd placed a 50g weight on the arm to simulate some of the forces the arm would feel during flight. We held the payload at 5 mbar and 55 °C for 4 hours and 5 mbar and -55 °C for 4 hours with sustained arm operations.

Testing: Fiducial Adhesives & Coatings

Our research led us to cut our fiducial tags out of Oracal 651, a fleet-grade vinyl that is rated for duration and performance in high temperatures. We used a spare sheet of aluminum (painted with the same white epoxy as RAM) as the base for our vinyl testing, adhering several fiducials to the aluminum sheet. We then let them sit for several hours to ensure a good bond. We applied some as though they were flight hardware — i.e. carefully — and slapped some on with deliberate air bubbles underneath.

We spent a day cycling an aluminum piece with the adhered fiducials between a freezer and incubator to test how the vinyl reacted to changes between extreme temperatures. The panel was placed in freezer that got down to roughly -50 °C and, while the metal frosted over, the vinyl seemed unaffected. It did not chip or flake off or otherwise seem damaged. The aluminum bowed slightly after the first freeze but the curve did not increase with more rounds of testing. After an hour in the freezer the panel was transferred immediately into an incubator that warmed it to roughly +50 °C for an hour. The vinyl seemed mostly unchanged, though it was a bit

stretchier when prodded with a hard object like a screwdriver. The vinyl did not slide or move, and the edges remained pristine, which meant they would be read properly.

After completing the temperature testing, we cycled the aluminum-and-vinyl panels through two vacuum cycles, drawing them down to ~20 mbar of pressure over the course of 10-15 minutes and then re-pressurizing. While we noticed some ballooning of the intentional air bubbles, those tags did not fail and remained detectable by computer vision. The tags without air bubbles showed no changes at all and also remained detectable by computer vision.

We also subjected the tags to significant infrared and ultraviolet load from a 500W halogen work lamp with its UV filters removed. We placed the aluminum test plate approximately 10 cm from the bare lamp bulb and measured temperature, elasticity, color durability, and adhesive tackiness for 90 minutes. Using an infrared laser thermometer, we measured a peak plate temperature of 114 °C at the 75-minute mark, well outside our expected parameters. At the hottest, the tags did not slide or move under the force of gravity or when prodded lightly with a screwdriver. We observed no negative impact to the vinyl (discoloration, deformation, cracking, or bubbling) during this test.

We also painted several fiducials in a variety of different enamels to see if they would hold up better than the vinyl in the extreme temperatures. These included:

1. Rust-Oleum black spray appliance epoxy, given its proven legacy of near-space flight and excellent adhesion and relative resistance to coefficient of thermal expansion (CTE) mismatches.
2. Oracal 651 “fleet-grade” matte vinyl, selected for its high UV resistance and proven durability on heated metals (stainless steel truck bodies are notoriously absorptive of solar heat) as well as for its low-albedo finish.
3. Sharpie permanent marker.
4. Sharpie eXtreme fade-resistant permanent marker.
5. Rust-Oleum 213174 acrylic lacquer, selected because our team lead was unable to read package labeling properly and thought it was a bonding epoxy.

These were subjected to similar UV-lamp stresses in addition to repeated extreme thermal cycling; we would place the plate, with fiducials, in our incubator oven, raise it to ~50 °C, then place it in our deep-freeze cell preservation freezer until ~ -60 °C. We repeated this cycle four times and observed no deleterious effects.

Most of this proved moot, however, when — for purely manufacturing reasons — we ended up flying fiducials made of Johnson Plastics 2-ply 0.0625” white/black acrylic. We did, however, apply an old testing vinyl tag to the side of the hull for flight to see how well it survived. No deleterious effects were noticed on the tag when it returned to us after landing.

Flight

RAM had a connectivity issue during the hang test; the RPi was losing the USB connection to our USB-driven U2D2 servo controller. Our software engineer resolved the issue by creating a symbolic link which pointed to the U2D2 as available, and we were able to successfully reintegrate. The first launch attempt was scrubbed due to rain in the area, but the second launch attempt went smoothly. Dan Koris sent commands from off-site in Colorado, and RAM performed nominally throughout the flight and satisfactorily completed the tasks on the busy box. The CSBF team successfully recovered the payload with no structural damage.

A still from the high-definition stream CSBF flight camera.

Statistics

The statistics for the flight, as provided by the HASP program:

Balloon Manufacturer	Winzen
Balloon Type	Zero pressure, 1 cap (W11.82-1E-37 CSBF #979)
Balloon Size	11.82 million cubic feet

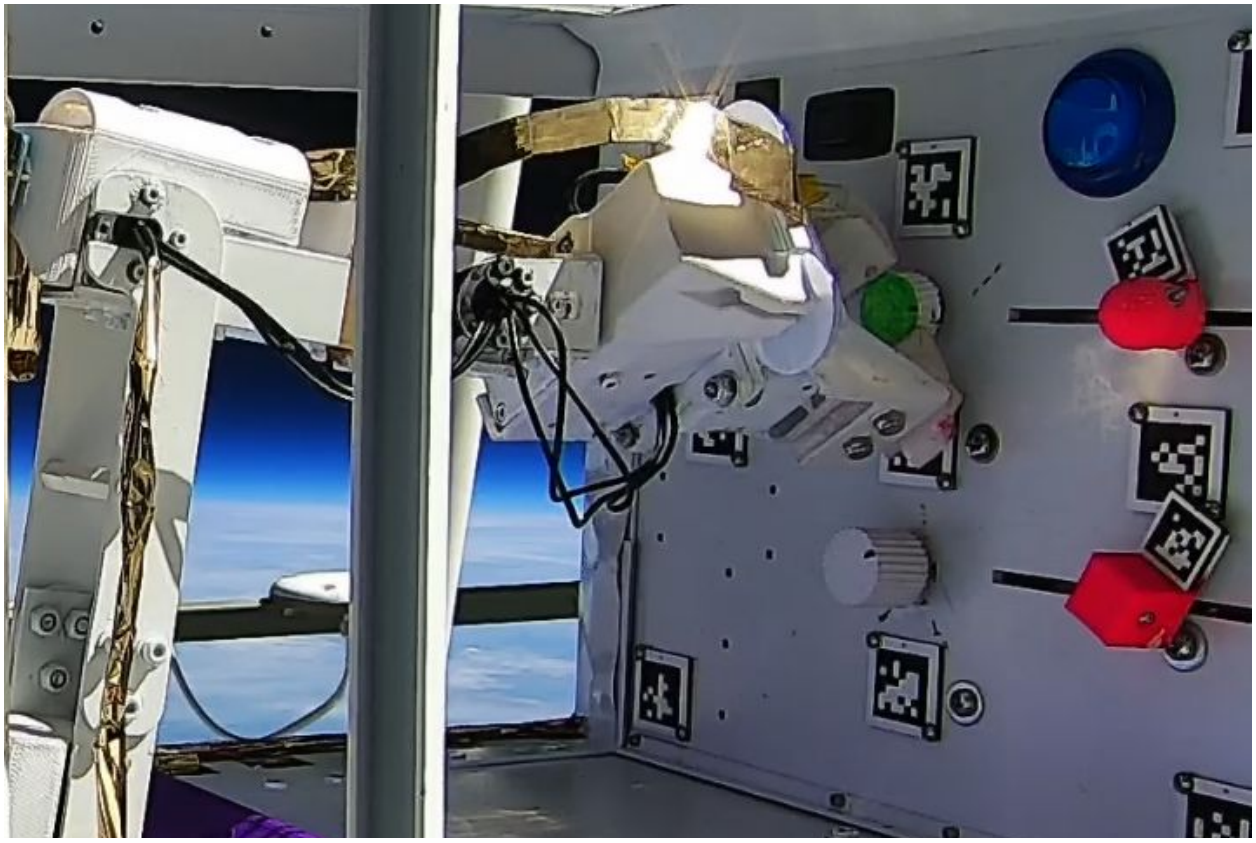
Parachute Diameter	79 feet
HASP Weight	411 pounds
SIP Weight	589 pounds
Balloon Systems	458 pounds
Ballast	542 pounds
Altitude with Ballast	122,500 feet
Altitude without Ballast	126,000 feet
Ballast for Drive-Up	140 pounds
Ballast for Sunset	259 pounds

Methods:

In order to test the designed kinematics and servo performance, we designed individual trials, triggered by commands from the ground team over serial uplink. Each trial consisted of a set of motions that would drive the arm through positions that would either turn knobs, push buttons, or toggle switches.

We wrote eighteen trials to cover any foreseeable flight action that was needed. These trials were written in Lua, which ran on top of the C++ code and ROS network. Lua was used because it integrates very well with C++, and because our roboticist has a lot of experience with it. Its use allowed the team to script trials on the fly without having to recompile the code every time a trial needed to be changed. This decreased trial development time drastically.

Each trial had a specific ID number attached to it. The system extracted this ID from serial uplink commands, allowing the arm to know which actions to perform. If a trial was already in progress, this trial ID would be queued for execution when the current action finished.



RAM halfway through twisting the slender upper potentiometer.

Trials 1-3 were reserved for basic "housekeeping" motions and had no scientific value. Trials 4 & 5 were distinct building-block and preliminary motions. Trials 6-14 were single-action trials with only one discrete "Busy Box" action in them, e.g. "flip left switch up," "rotate upper knob to the right," etc. We implemented these separately to reset the board in case a more complex trial failed a specific action. Trials 20-22 were our scientific objectives. These trials consisted of a collection of actions.

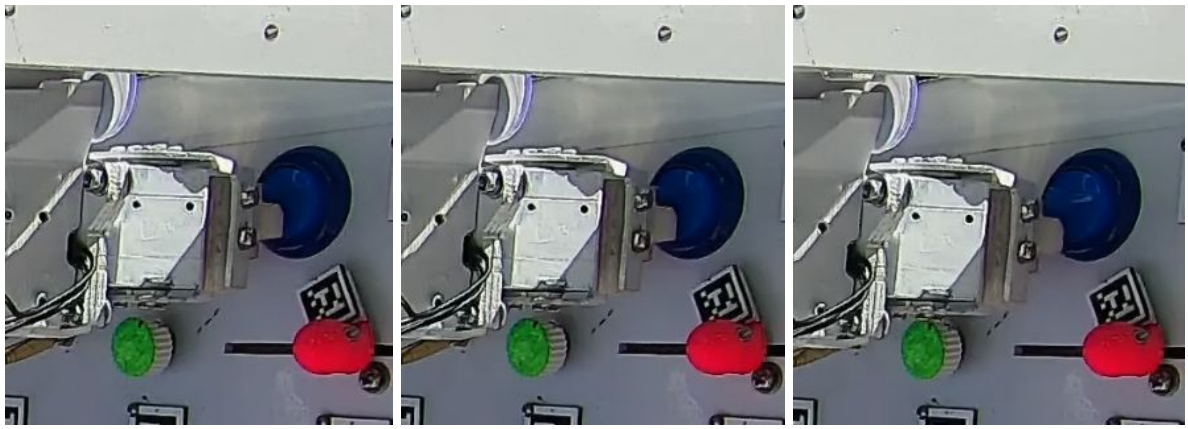
Trial #	Trial Description
1	Ungrip the "Operational Support Rod," and move to the Home Position (a position that faced the Busy Box head-on.)
2	Regrip the "Operational Support Rod" from any position.
3	Send the arm to the Home Position from any location.
4	Blueprint trial used to develop all subsequent trials. All developed trials began their development life as trial four's script.
5	"Warm-up" trial, a long series of no-contact actions. The arm would track around in front of the Busy Box but never actually touch it.

6	Push the blue button.
7	Toggle right switch from the ‘down’ position to ‘up’.
8	Toggle right switch from the ‘up’ position to ‘down’.
9	Toggle left switch from the ‘down’ position to ‘up’.
10	Toggle left switch from the ‘up’ position to ‘down’.
11	Turn upper potentiometer (knob) to the right and then back the same amount.
12	Turn lower potentiometer (knob) to the right and then back the same amount.
13	Turn upper potentiometer (knob) to the left and then back the same amount.
14	Turn lower potentiometer (knob) to the left and then back the same amount.
20	1. Press blue button. 2. Right switch down to up. 3. Left switch down to up. 4. Right switch up to down. 5. Left switch up to down. 6. Press blue button. 7. Home position. 8. Upper potentiometer right turn. 9. Upper potentiometer left turn. 10. Lower potentiometer right turn. 11. Lower potentiometer left turn.
21	1. Press blue button. 2. Upper potentiometer right turn. 3. Upper potentiometer left turn. 4. Lower potentiometer right turn. 5. Lower potentiometer left turn. 6. Press blue button. 7. Upper potentiometer right turn. 8. Upper potentiometer left turn. 9. Lower potentiometer right turn. 10. Lower potentiometer left turn. 11. Home position.
22	1. Right switch down to up. 2. Left switch down to up. 3. Right switch up to down. 4. Left switch up to down. 5. Home position. 6. Right switch down to up. 7. Left switch down to up. 8. Right switch up to down. 9. Left switch down to up. 10. Home position.

Results

Results: Kinematics

Over the duration of the flight, the data shows a overall 60% success rate on all objectives. However, these numbers may be a little deceptive. The upper potentiometer broke sometime after launch and stopped providing any reliable feedback, and the arm should not be judged for its failure.



Three stills from flight, taken one second apart, showing an unsuccessful button press.

The blue button failed to detect presses most of the time. This is surprising, given that we were clearly able to see the button being thoroughly depressed by the arm on our high-definition camera feed for the first attempts. On review of footage from later in the flight, however, the knuckle seems to only brush the button and barely apply any pressure. Our best explanation is that it got frozen out of position or the kinematics were not reliable enough to always depress it deeply enough to make an electrical connection. Mechanically, it still works in post-flight ground testing. Similarly, video feedback showed that while the upper potentiometer was not giving good digital feedback, it was, however, actually physically rotating. Thus, the only complete failure was a single toggle in third trial (trial type 22).

Trial Type	Button Presses *[completed/attempted/%]*			Switch Toggles *[completed/attempted/%]*			Knob Twists *[completed/attempted/%]*			Total Completion		
6	1	/ 1	100%	0	/ 0	—	0	/ 0	—	**1**	/ **1**	100%
6	1	/ 1	100%	0	/ 0	—	0	/ 0	—	**1**	/ **1**	100%
20	1	/ 2	50%	2	/ 2	100%	2	/ 4	50%	**5**	/ **8**	63%
21	0	/ 2	0%	0	/ 0	—	4	/ 8	50%	**4**	/ **10**	40%
22	0	/ 0	—	4	/ 4	100%	0	/ 0	—	**4**	/ **4**	100%
20	0	/ 2	0%	2	/ 2	100%	2	/ 4	50%	**4**	/ **8**	50%
22	0	/ 0	—	4	/ 4	100%	0	/ 0	—	**4**	/ **4**	100%
22	0	/ 0	—	3	/ 4	75%	0	/ 0	—	**3**	/ **4**	75%
21	0	/ 2	0%	0	/ 0	—	4	/ 8	50%	**4**	/ **10**	40%
20	0	/ 2	0%	2	/ 2	100%	2	/ 4	50%	**4**	/ **8**	50%
20	1	/ 2	50%	2	/ 2	100%	2	/ 4	50%	**5**	/ **8**	63%
21	0	/ 2	0%	0	/ 0	—	4	/ 8	50%	**4**	/ **10**	40%
22	0	/ 0	—	4	/ 4	100%	0	/ 0	—	**4**	/ **4**	100%
Total	**4**	**/ 16**	**25%**	**23**	**/ 24**	**96%**	**20**	**/ 38**	**53%**	**47**	**/ 78**	**60%**

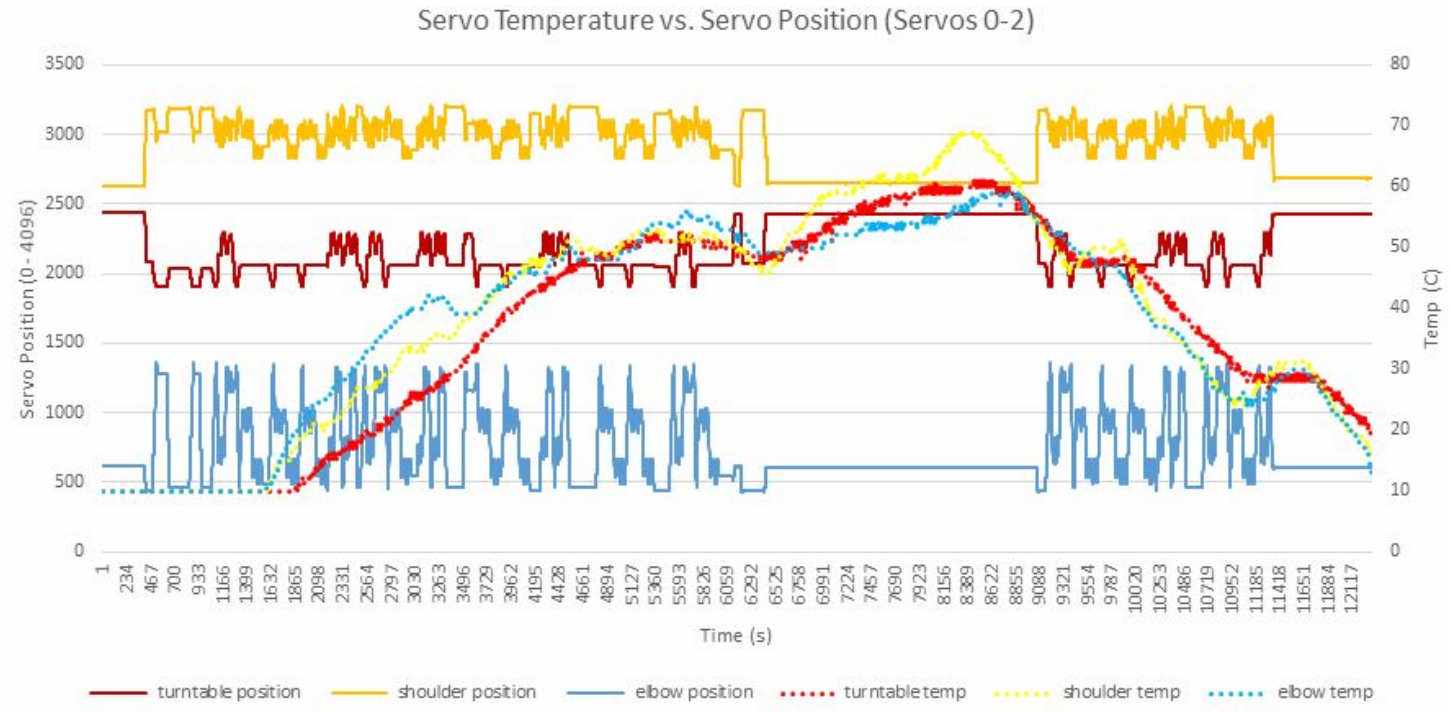
Results: Fiducials & Computer Vision

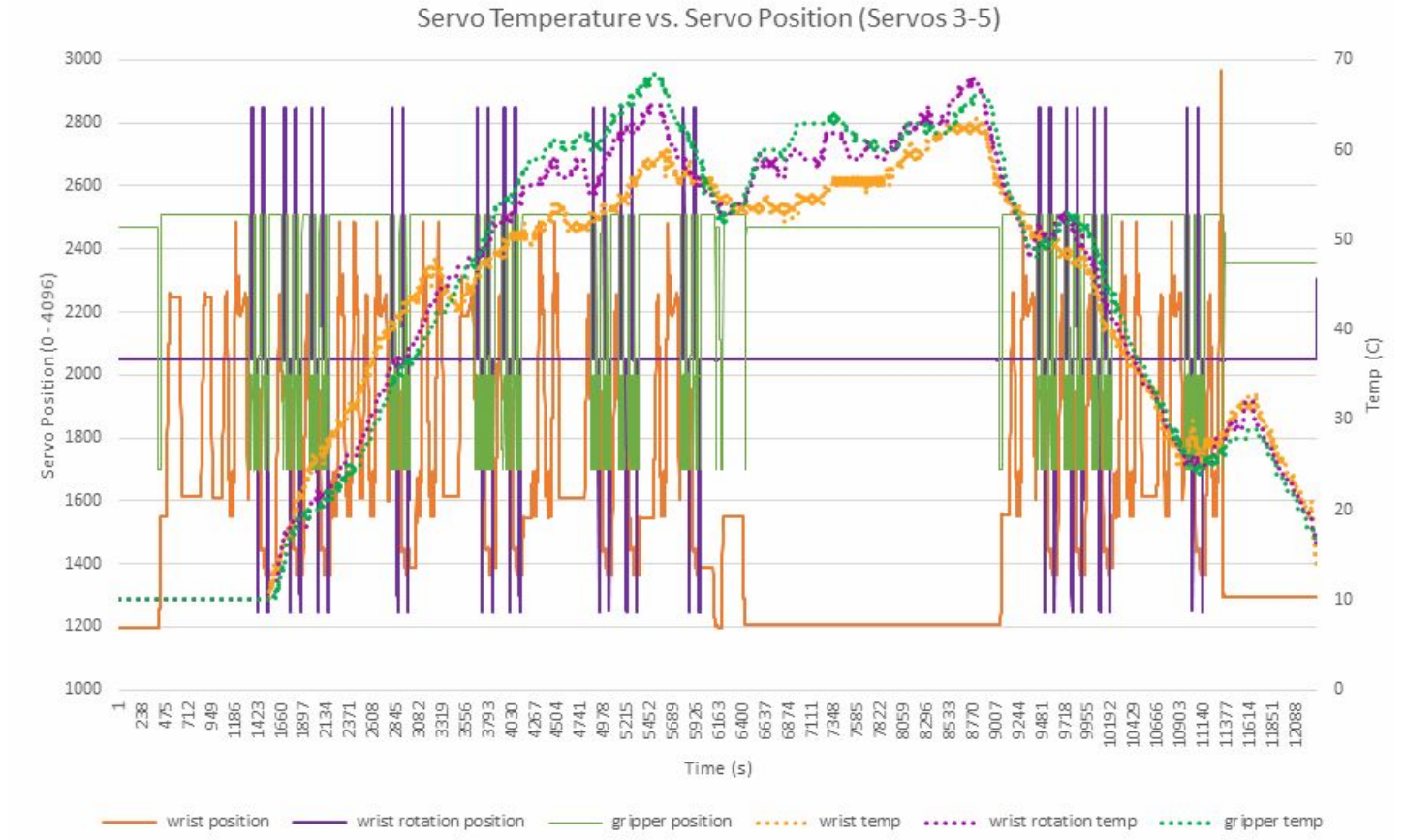
Due to time constraints and power issues we ended up cutting almost all computer-vision out of our core experiment. Our only remaining camera, an Adafruit RPi 'spy' camera, failed shortly after/during launch due to unknown circumstances; the hardware remains unresponsive after recovery. Consequently, there was no computer vision component active during flight. Originally, the software fell behind in production, and the idea was to simply capture images and post-process them. However, the camera was verified to be working just before its final attachment to HASP, and then after that did not produce a single valid image. We recommend a future experiment fully devoted to camera survivability and robust computer vision under the wide range of lighting conditions on HASP, studying:

- aggregated detection rate
- patterns of flaws in detection
- fiducial integrity/durability: compare vinyl appliques, painted acrylic, and laser-etched two-ply tags
- glare issues, especially with an eye towards post-processing as a proof of concept for live, in-flight glare correction using software

Results: Servo Performance

The Dynamixel line servos performed admirably during testing and flight. Their temperature sensors do not read temperatures below 10 °C; however, they functioned quite well in almost every environment we subjected them to.



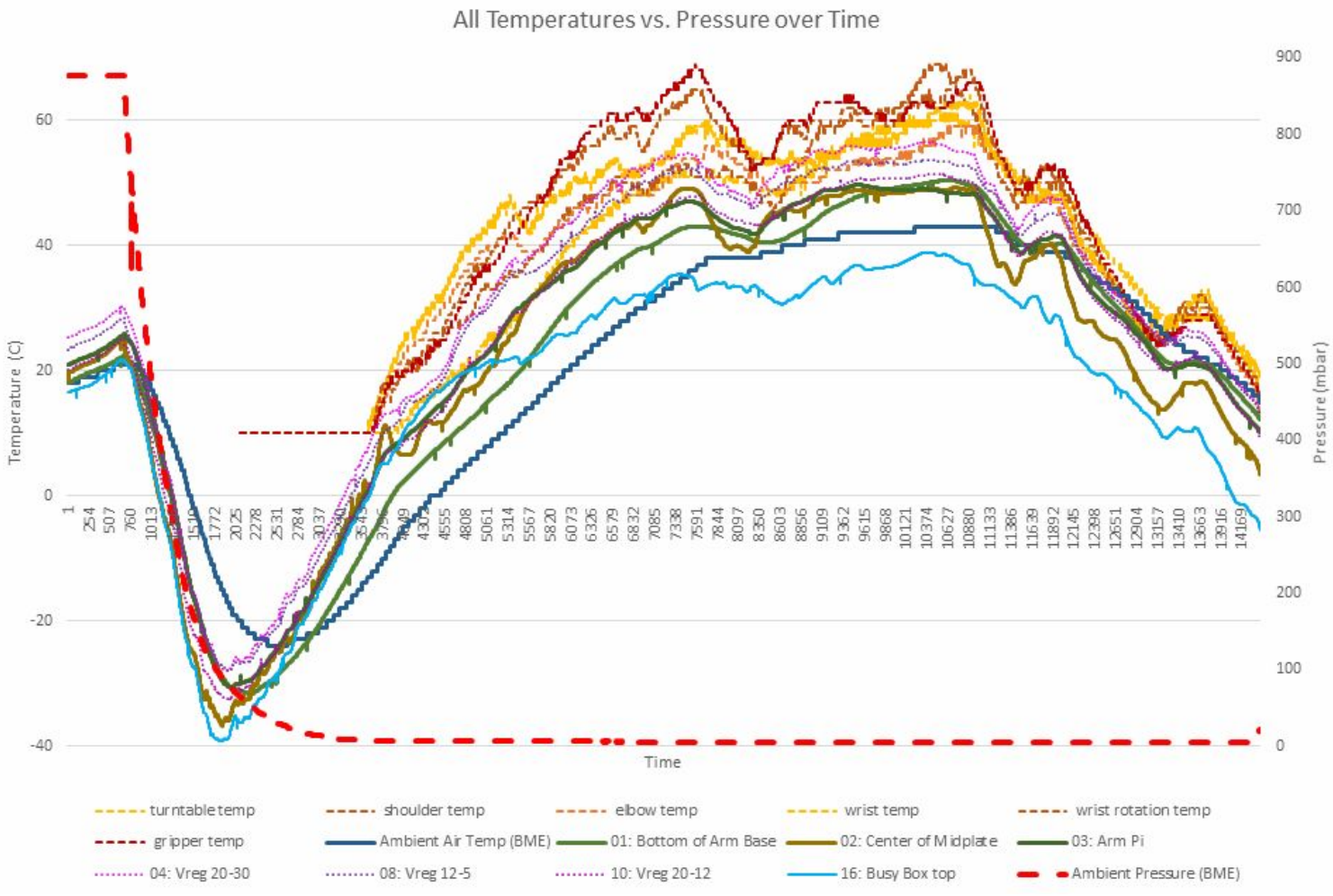


The highest internal temperature recorded by any servo during flight was at the shoulder (responsible for bearing the entire cantilevered weight of the arm) where it reached 69 °C, still well below the manufacturer specified upper limit of 80 °C. During flight operations, no measurable performance degradation was detected.

The servo temperature-versus-position graphs show, in two different groups, the temperature of the servos relative to their range of motion. The solid lines depict servo position, i.e. where they were commanded to rotate relative to their 0-position. Large derivatives of position show significant movements. Note that there is no strong correlation between servo position and temperature increase, and that temperature build-up or bleed-off seems largely connected to the length of exposure to high/low ambient temperatures. Recall also that our servos are unable to report temperature below 10 °C.

Results: Thermal Profile

Looking at all temperatures over the course of the flight, we see that the hottest temperatures were recorded by the gripper. The gripper, though it had the least work required of any of the servos, also had the least solar shielding. The second- and third-highest temperatures are the two wrists, which had the second- and third-least solar shielding and similar loads. The lowest temperatures are observed in the upper portion of the Busy Box, where our BME sensors and a Dallas thermometer are marked in blue.



Servo temperatures are marked with dashes. Internal voltage regulator temperatures are dotted. Hot colors like red indicate that the temperature is from a component on the exterior of the payload. Cooler colors like blue and violet signify that the component was enclosed more deeply within the payload. All fields have been washed clean of ~300 clearly-bad outliers, which are likely the result of bouncing grounds.

Conclusions

Scientific Conclusion: Kinematics & ROS

Our first goal was to kinematically model and control an arm capable of the ranges of motion, degrees of precision, and error-correction required for our mission. Despite significant setbacks along the way, RAM was successful in demonstrating a 4DoF arm which operated as-instructed for the entire flight.

Scientific Conclusion: Fiducials & Computer Vision

Our second goal was to prove our fiducials would survive in the near-space environment and to prove our ability to detect them at every point in the mission. While we were able to definitively show fiducial robustness, we were not able to test their visual acquisition and recognition due to our camera failure.

Scientific Conclusion: Servo Strength/Dexterity/Stamina

Our third goal was to test the performance of commercial off-the-shelf (CotS) servos. We are very pleased with our results; RAM has shown that Dynamixel servos are an excellent and performant solution for float operations. We were quite pleased with the mechanical mounting hardware, thermal resilience, ease of commanding, simplicity of wiring, and sustained strength of the motors.

The non-zero amount of ‘play’ or wiggle-room in the servos, even while activated and locked, is a cause for potential concern in a precision robotics environment. In our estimation, this did not impact our dramatically-simplified flight goals in any significant way, but it may be a concern with tasks requiring a finer degree of control and precision. We are unsure how this can be improved and recommend further conversations with the manufacturer. We are also curious how a true computer-vision-driven system might compensate for this with real-time feedback, and recommend future research on the matter.

Lessons Learned: Technical Design

The following bullets are call-out summaries of content already previously mentioned above. **They are re-listed here for the convenience of the reader:**

- Projects running RPi should strongly consider a standalone full power supply to regulate power draw and insulate against brownouts, especially during boot-up.
- Running Lua on top of C++ and ROS significantly reduced the amount of effort required for implementing trials in our environment.
- Building an open ethernet port (that is easily, physically accessible from outside the hull) was a huge time-saver, allowing SSHing directly into the RPi to maintain & update code.
- We strongly recommend mounting as many internal components to a single mechanical ground for ease of access. This can take several forms: a captive mounting surface which remains attached to the PVC mount plate that the hull slides over, or a central rack which slides out, drawer-style, or a door-accessible plate. Having components from different mount surfaces wired together is a significant challenge; one must contend with carefully holding multiple surfaces together, or disconnect and reconnect cables constantly, to avoid straining/damaging the connections. This method of wiring should be avoided. This is somewhat mitigated with quick-release lock-tab connectors but still a concern for future missions.
- Standardizing electronics — particularly servos — is excellent where possible. This provides interchangeability, a standard commanding protocol, a standard wiring protocol, and generally reduces complexity.
- If possible, consider designing custom circuit boards to print once a semifinal design has been settled on. The cost is low relative to the time savings, and it can further aid maintainability.
- Future experiments should consider incorporating photometers, oriented normal to the four major sides of the payload, to help determine solar direction for correlating with local temperature minima and maxima as well as glare angles.

Lessons Learned: Management, Workflow, & Assembly

Looking back, our student leadership has identified several key decisions which caused the team to be quite far behind schedule. This delay cascaded down through the months, burning valuable development and testing time which, in turn, prevented us from troubleshooting issues with our power system and computer vision in time to fix them for flight.

One critical failure was having a core engineering team member hyper-focus on thermal equilibrium; they spent so much time testing individual components and materials for thermal conductance and resilience when instead the team should have been building iterative ‘minimum viable products’. **We strongly recommend focusing on building series of generational, working prototypes as a fundamental development paradigm** in order to keep the whole team aligned around producing something which will satisfy the core science objectives.

Similarly, we hyper-focused on machining precision to a degree that significantly disrupted our production pipeline. It would have been far more valuable, at three particular

points in the project, to build something that was Good Enough in an afternoon rather than wait three weeks for an external machinist to produce. Tangentially, extra delays appeared when our machining resources proved over-burdened, wasting several more weeks of development time. This is another example of how keeping focused on making the next working model would have impelled us to move on rather than waiting around for unnecessarily high-precision components. We resolved this through the insistence of our PI to move ahead with lower-precision, in-house work, which was further enabled through the loan of a band saw.

Demographics & Alumnus Tracking

Jimmy Acevedo graduated from North Carolina Central University in May of 2018, and after three successful NASA internships he was hired as the Intern Coordinator and STEM Engagement Specialist at ASRC Federal / NASA Goddard Space Flight Center. Send him the resumes of any promising students you know.

Daniel Koris graduated with an AS from Durham Tech in May of 2018 and had an internship at NASA Goddard in the Satellite Servicing Projects Division over summer 2018. Now he has transferred to CU Boulder for Computer Science, and he expects to graduate in 2021.

Meredith Murray has had two back-to-back internships with Johnson Space Center, having been asked to stay on for another semester in her role and was asked to interview for the NASA Pathways program.

Ryan Theurer from GOAT (HASP 2017) graduated from UNC Chapel Hill in May 2018 and participated in the NASA DEVELOP program. Now he is a Technology Advisory Consultant at Ernst and Young.

Name	Start Date	End Date	Role	Student Status	Race	Ethnicity	Gender	Disabled
Jimmy Acevedo	10 Oct 2017	14 Dec 2018	Student Team Lead	Undergrad	White	Hispanic/Latinx	Male	No
Seth Close	08 Apr 2018	Present	Mechanical Engineering	Undergrad	White	Non-Hispanic/Latinx	Male	No
James Cowell	01 June 2018	Present	Engineering	Undergrad	White	Non-Hispanic/Latinx	Male	No
Daniel Daugherty	10 Oct 2017	Present	Mechanical Engineering	Undergrad	White	Non-Hispanic/Latinx	Male	No
Daniel R. Koris	10 Oct 2017	Present	Electrical & Software Lead	Undergrad	White	Non-Hispanic/Latinx	Male	No
Spencer Lee	01 June 2018	Present	Engineering & Fabrication	Undergrad	White	Non-Hispanic/Latinx	Male	No
Meredith Murray	10 Oct 2017	Present	Documentation & Social Media	Undergrad	White	Non-Hispanic/Latinx	Female	No
Spencer Boyd	10 Oct 2017	Present	Machinist Consultant	Undergrad	White	Non-Hispanic/Latinx	Male	No
Landon Fernandez	05 Jan 2018	01 Jun 2018	Aerospace Engineering	Undergrad	White	Hispanic/Latinx	Male	No
Soham Pai Kane	15 Dec 2017	01 May 2018	Software	Undergrad	Asian	Non-Hispanic/Latinx	Male	No

Presentations

[1]J. Acevedo and D. Daugherty, "Opportunities with NC Space Grant", North Carolina Space Grant High Altitude Balloon Launch Keynote, Hickory, NC, 2018.

[2]J. Hoover and D. Brush, "Orange County Career Readiness", Cedar Ridge High School, Hillsborough, NC, 2017.

[3]J. Hoover and J. Acevedo, "NASA Undergraduate Research", Durham Kiwanis Club, Durham, NC, 2017.

[4]J. Hoover, "NASA Undergraduate Research", NASA Langley Hampton, VA, 2017.

[5]J. Hoover, J. Acevedo, and D. Daugherty, "High Altitude Ballooning", State of North Carolina Undergraduate Research and Creativity Symposium, 2016.

[6] J. Hoover and T.G. Guzik, "Creating a NASA Workforce Pipeline through High Altitude Ballooning", American Geophysical Union Fall Meeting, 2018.

Press Releases/Articles

[1] F. Sergiou, "Durham Tech Team Flexes Robotic Arm at NASA Langley", NASA.gov, 2017
https://www.nasa.gov/feature/langley/durham-tech-team-flexes-robotic-arm-at-nasa-langley

[2] Durham Tech, "Durham Tech to partner with NASA for second consecutive year", 2018
https://www.durhamtech.edu/news/pressreleases/2018/jan30HASP.pdf

[3]Durham Tech, "Durham Tech partners with NASA, headed to the edge of space", 2017.

[4]B. Granath, "Students Turn NASA Robotic Experience to High Altitude Technology", Spaceport Magazine, 2018.

[5]M. Wynn, "Durham Tech Students Participate in NASA Initiative", Durham Magazine, p. 23, 2017.

Videos

Durham Tech NASA Projects 2016
Undergraduate Research Student Perspectives
End of Semester Science and Engineering Projects

Radio

WCHL