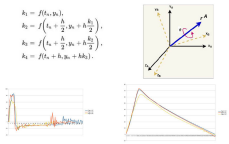


MIT Admissions

Maker Portfolio 2025–2026

Aidan Busby
3240 41st ave sw
Seattle, WA 98116, United States
2063130397
aidanbusby123@gmail.com

Submitted: January 5, 2026

Paid: $10.00 on January 5, 2026

Forms

Biographical information

***** indicates a required field

1. Date of Birth

03/10/2008

2. Are you submitting a first-year or a transfer application?

First-Year

2.1. Are you applying for Early Action or Regular Action?

Remember that all portfolios must be submitted by the corresponding application deadline for the cycle you are applying. Portfolios submitted after these dates will not be evaluated.

Regular Action

3. Application Reference Number

Your unique nine-digit application reference number can be found on the "Getting Started" page of your MIT Application at apply.mitadmissions.org/apply. Your reference number is also listed at the top of your applicant status page that appears after you submit your MIT Application.

765503910

Maker Portfolio

***** indicates a required field

Use the uploader tool later in the portfolio to share more media about your work.

1. What category should your primary reviewer should be an expert in?

This helps us match your portfolio to the reviewer best equipped to take a first look at it. If multiple categories apply, select the one that feels most relevant.

DO NOT select “Other” unless **NONE** of the listed categories apply. We will make sure that cross-disciplinary portfolios get seen by the right people.

Code

1.1. Code sub-category

Pick whichever option you think would best help us match your portfolio to the right specialists. (There are many other types of code not explicitly listed, and we do not specially value any one of these categories over others; these are just historically common categories used to help distribute portfolios.)

Machine Learning

2. What search keywords would you use to describe the contents of your portfolio?

This is used to help us match your portfolio to specific reviewers. Please provide a few comma-separated tags describing the project(s) you are featuring.

Examples of a helpful level of detail: foam cosplay armor, hand sewn stuffed animals, algae biofuel, PCB design, geospatial data analysis, linear regression classifier, lost-wax bronze casting, wooden puzzle box

Unhelpful tag examples: creative, sustainability (too vague, doesn't tell us what you made), the name of a programming language or chip you used (too niche, doesn't tell us what you made), art, app, AI, 3d print (also too vague)

> Active control "airbrake" system for rockety, brain-AI

3. Portfolio summary

Briefly summarize what you make.

> I built an active-control system "airbrake" for my high school's rocketry team Junior year. An active control system uses various sensors (an inertial measurement unit and barometer, in our case) to determine where the system is at, before using this information to modify the trajectory of the rocket to reach our target altitude.In our case, this meant using a physics simulator to determine the rocket's predicted trajectory, and manipulate servo-controlled flaps to alter it.
> Next, I discuss my recent adventures in brain-AI and Hierarchical Temporal Memory, and what I look forward to testing.

4. Project highlight

Share a highlight from one of your featured portfolio projects that feels most technically/ creatively interesting or important to you.

(Often what applicants assume is interesting to us is not what we actually find most compelling about a portfolio. We know there can be pressure to overthink what you "should" share, but by focusing on that, you may sell yourself short and fail to give us a clear impression of you. Trust your instincts about what feels most interesting to you.)

> I don't believe there is anything more elegant or beautiful than the concept of the continuous online learning paradigm behind HTM and other cortical learning algorithms.
> Active control systems constantly process inputs, in rocketry position and velocity data, in order to form predictions about the world and act on those predictions. Except, control theory is limited to narrow applications like reaching a specified altitude with a rocket.
> The core idea behind neocortex-inspired continuous online learning is to deploy an algorithm that constantly learns more about the world, forming refining its predictions about what is happening, generalizing and acquiring knowledge and intuition. But instead of being confined to narrow domains, HTM can learn and generalize anything, in theory.
>
> I find great joy in the idea of a system that constantly learns more about the world like this.

5. Maker context

Help us understand the environment you make in. For example, your workshop or makerspace, how you acquire your tools and materials, project constraints, and mentors or hobbyist communities that have supported your work. If you make physical projects, a photo of your workspace can be helpful to add in the media uploader section.

> For me, its not so much about where I'm making but who I'm making with. During rocketry season, this meant I might be in the middle of 60 acres park in Redmond, using Leo's 50lb battery back as a surface to solder on (I'm sure his mother would be pleased). I'd code in Nathan's car while we did tests on the inertial measurement unit by bombing down Charleston, the 45 degree hill near our school, or while eating one of many dinners at Chick-fil-a. That said, my mom's garage tends to be the sight of many high voltage and rocketry shenanigans, or messing around with my homemade gokart w/ friends

6. Collaborators

Honesty, teamwork, and collaboration are important to us at MIT. What was your personal role in each project? If you made changes to an existing project, explain what you started with and what you changed.

For any material you are submitting that is not 100% your own handcrafted work and original design, name your direct collaborators/mentors and briefly summarize their roles. Note which collaborators might also be submitting an MIT Maker Portfolio this year.

> My role in rocketry was designing the avionics software: the physics simulation, avionics,

control systems, and state-machine logic. The physical airbrake was designed and 3D-printed by Leo McClintock and Arthur Johnston (the latter of whom I believe is applying to MIT as well!), while Sam Kothe soldered and designer everything. I am in charge of the electronics this year, as Sam has graduated, on top of writing new code.
My brain-based AI work is inspired by/utilizes software pioneered by Numenta and the HTM community.

7. Sources & citations

Cite any work that was not handcrafted by you or your named collaborators. Examples: tutorials/guides, patterns (sewing, crochet, woodworking, etc.), existing nonstandard hardware (off the shelf parts, kit parts), existing firmware, existing models (3d printing, games, ML, etc.), assistive tools. If you relied on platforms or engines (game-making engines, pattern generators, etc.), summarize what work they did. If your work was directly inspired by other third-party work, cite it.

Coders: we recognize that you may be using assistive tools; if so, we require that you cite them and describe in detail how you used them.

Rocketry parts:
Teensy 4.1, Adafruit MPL3115A2, BNO055, LSM9DS1. Pretty standard parts, although in breakout board form.

HTM:

Numenta: https://www.numenta.com/
https://github.com/htm-community/htm.core

8. What motivates or excites you about what you make?

We are interested in the relationship between what you make and why you make it.

I love the idea of bringing machines to life, giving them the ability to interact and make sense of our world. I think it can help us understand ourselves, especially in the domains of AI, while obviously bringing a positive economic impact.

9. What have you made that holds the most personal meaning for you, and why?

This does not need to be something technically “impressive.” We are trying to better understand you as a person.

The day we had our first perfect flight, and I knew the airbrake worked, that all those months had paid off, held the most emotional significance to me. I learned that if I put my mind to it, if we all pushed through the pain, we could do it; especially after so much failure.

10.
If you could ask an expert in the field one question to help you grow or to improve your work, what would you ask?

I would ask them whether or not training a neural network to improve our active control system would be beneficial, or if sticking to traditional approaches like Runge-Kutta-4 simulations, PID.
I would also ask them if Model Predictive Control or neural nets might be an avenue for control.

11. [Optional] Personal website or Github, if applicable

We do not guarantee that we will review all supplementary material, but it may be helpful to us. (Again, what applicants assume will be most interesting to us may not be what we find most interesting.)

Just like the work you include in the Maker Portfolio, it should be clear that all linked material is your own work. If you submit a website, we must be able to clearly verify that the content on it was created by you. There is no bonus for having a website.

https://github.com/aidanbusby123/TARC-airbrakes (use the airbrakes directory. The RK4PID is what I am currently working on.) https://busbylabs.com, https://github.com/aidanbusby123/NeoContext

12. [Optional] Third-party context, if applicable

If you submitted work made under a specific set of rules or constraints, like for a competition or a grant, please give your reviewer some context. Provide the name of the organization, a link to an online copy of the constraints, and a brief summary in case we are not familiar. You can skip the summary for FRC and FTC projects, since we receive many of those.

The American Rocketry Challenge: https://rocketrychallenge.org/wp-content/
"
Rules of the 2026 American Rocketry Challenge:

1. Payload: Design a rocket that cradles one raw Grade A Large egg of 55 to 63 grams weight, carried in any orientation that must survive the flight uncracked
2. Altitude Goal: Reach an impressive 750 feet.
3. Flight Time: Safely return to Earth within 36 to 39 seconds.
"
(From their website)
The constraints of the 2025 season were:

two raw hens eggs

790 feet

41-44 seconds

Maker Portfolio acknowledgement

* indicates a required field

1. Checklist: Does my work meet the Maker Portfolio criteria?

All items must be true in order for a project to be appropriate for the Maker Portfolio.

I am submitting engineering or crafting projects that I made.|I have provided thorough citations for any work that I did not personally handcraft.|I am NOT submitting a musical composition, screenplay, or performance.|I am NOT submitting a traditional visual art or architecture portfolio.|I am NOT submitting poetry, essays, or creative writing.|I am NOT submitting a patent, patent application, business proposal, pitch deck, or legal document.|I am NOT submitting a project I made by following a tutorial or class assignment.

2. Are you submitting any code projects in your portfolio?

Yes

2.1.
I have included access to my code AND working demonstrations of my code project(s).

We find Github easiest to read, but other methods are acceptable.

Yes

2.2.
I understand that my project will not be reviewed if either of these is missing or inaccessible.

Yes, I understand

3. Please acknowledge that the work in this portfolio was created by you.

In the case of collaborative work, you have detailed your specific contributions and credited all collaborators.

I have provided collaborator credit and citations for any work that I did not personally handcraft, and all other work I am submitting is my own work.

Busby, Aidan 52218491438

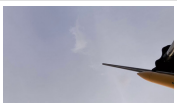
PORTFOLIO

MITMaker

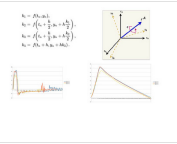
A live demo of the airbrake. Unfortunately, due to the high speeds of rocketry and our inability to procure a specialized rocket camera, this is the best I could come up with.

I demonstrate the airbrake's ability to deploy upon achieving a trajectory where it is predicted to overshoot the target apogee, and the various states that the airbrake is capable of occupying. These states are the PAD phase, when the rocket believes it isn't oriented within the appropriate angle interval for launch, the LAUNCH phase, the IGNITION and COAST phase (blue light), and the pseudo-phase designated by the yellow light signifying airbrake deployment.

(By the way, in the video when I said "inverted", I was referring to the inverted nature of the display).



rocketshot



Figuring out the maths

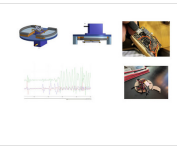
The first step to building the active control system was learning the mathematics behind it.

Our control system relies on three main systems: the navigation system, prediction system, and control system.

The navigation system was the first system I developed. It utilizes quaternions (visualized in the top right) to track the orientation of the rocket, which we need in order to calculate the velocity and acceleration components. Quaternions are 4D hyperspheres, and while they are very convenient for storing and applying rotations as they can be multiplied like matrices, the math and intuition behind them is a bit funky.

After the navigation was the simulation system. The simulator uses a 4th order Runge-Kutta method to approximate the future trajectory of the rocket given the current position and velocity, with far higher accuracy than a simple implementation of Newton's method.

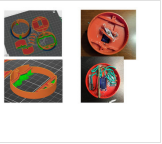
The two pictures below show some accelerometer and velocity data from early navigation tests.



Airbrake V0.1

Our first iteration of the airbrake was a simple contraption of laser-cut balsa wood, an SG90 servo, and an Adafruit Feather M0. It did not last long, although it provided indispensable for early tests of the navigation systems, such as the testing of our quaternion math visible in the bottom left.

Airbrake V0.1 suffered two major injuries. We were able to superglue it back together after the first, but after crash #2 we decided it was out of commission.



Airbrake V0.2

Our next airbrake version was V0. 4.1, V0.2 offered an environment to tra e a far more powerful microcontrolle calculations behind our work-in-pr ease.

Airbrake V1.1 (The pink airbrake)

Airbrake V1.1 was our first honest-to-god airbrake, and lasted nearly 30 flights until its untimely demise on the final launch day.

The pink airbrake was slimmer than previous versions, contained a powerful DS113MG servo with 3.5kg/cm of torque that could actuate the airbrake flaps under the high-airspeed conditions of launch.

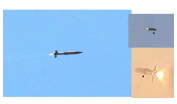
For navigation, it used an mpl3115A2 barometer and a LSM9ds1 Inertial Measurement Unit. It was our most accurate and calibrated, and once I corrected a fatal error in our simulation code that had evaded us for weeks, we were able to reach peak performance irrespective of motor errors, achieving multiple perfect flights.

Airbrake V1.2 (The red airbrake)

The red airbrake was the same design as the pink airbrake, except it used a BMP390 barometer. Despite this, poor mounting of the Inertial Measurement Unit, which contains the accelerometer, gyroscope, and magnetometer (which uses earths magnetic fields to find absolute orientation) on top of the servo induced an intolerable amount of electrical interference in our system, leading to both accelerometer drift and extreme barometric inaccuracy, which is included in a later slide.

The top left photo depicts the calibration script we used to calibrate the airbrakes, and which did poorly to calibrate the red airbrake.

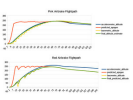
The bottom left image is from our final flight. No, the rocket should not look like that; the parachute fell out, so we were unable to qualify. The flight was so poor because of electrical interference it wouldn't have mattered anyways. By this time, the pink airbrake was dead.



Photos from flight

If you look closely at the top right photo, the yellow indicator light is visible, the same light as in my airbrake demo video.

The bottom right is from the included video clip, and depicts the airbrake flaps beginning to deploy after the rocket leaves the pad.



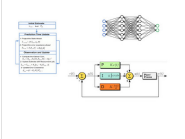
Flight data

The top photo depicts a successful flight from the pink airbrake.

The red line graph shows the predicted apogee (max altitude) of the rocket over time, the yellow shows the barometric altitude, the blue shows the altitude calculated solely from accelerometer data, and the green shows the final prediction from combining the two.
The accelerometer and barometer altitudes do differ by a

few meters, which is something that proper calibration and an Extended Kalman Filter should fix.

The bottom graph is from the red airbrake. The impact of the electrical noise from the servo is pretty apparent.



Next steps

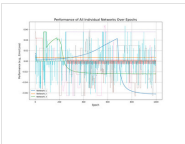
For this year's competition, we are working on a few approaches.

First, we have begun implementing a PID loop in place of fully deploying the airbrake at a specific time. This should generate less error.

We are also considering an EKF (Extended Kalman Filter) to improve the accuracy of our altitude estimates, instead of using a simple complementary filter.

If the PID and EKF approaches seem to be working, I am contemplating training a neural network using simulated flight paths for control.

Specifically, combining control strategies like Model Predictive Control with a neural network is one potential avenue. The amount of computational power required for full MPC is too high for a Teensy, while a neural network offers a promising way to potentially learn MPC optimization.

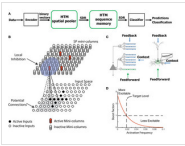


First brain-AI explorations

This section contains a description of my work with brain AI, specifically the Hierarchical Temporal Memory architecture.

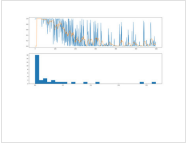
Before even stumbling on HTM, I attempted building what I later found out to be similar to HTM, although the network failed miserably. Above is a graph of the error of my particular network to memorize even a super simple simple dataset such as 12 Spanish to English translations.

The problem with my initial attempt at building brain-AI was that I had a problem with exploding weights; I introduced no inhibitory mechanism, or other way to ensure the network eventually converged on a stable state. This error might be considered similar to the issues of pure perceptrons or SNN's without a surrogate gradient.



HTM description

The solution to my problems was to take advantage of a preexisting brain-AI framework called Hierarchical Temporal Memory (HTM). HTM models the neocortex, the area of the brain responsible for conscious thought.
The neocortex contains several heavily interconnected layers. Higher neocortical layers combine information from broader spatial regions and operate on more abstract temporal timescales - think about the difference between processing each word in this paragraph and the overall meaning that you get from it.
HTM networks contain neurons in the form of groups of columns, where each column acts like one emergent neuron constructed from a number of actual neurons within each column. Having multiple neurons per column allows for different contexts even with the same pattern of column activations, permitting a single representation of neurons to represent information like a word alongside the context of that information, which is key.



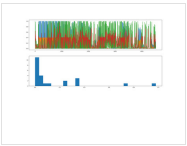
First test of HTM learning

One of my first tests of the HTM network. I fed a network of 1024 columns of neurons, 32 neurons per column, with training data consisting of 150 sentences and testing data consisting of another unique 150 sentences. Each of these sentences are of the following form:

A is in the X. B is in the Y. Where is (A or B)?

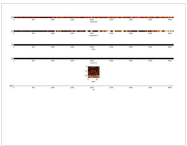
The goal is to train an HTM network that is capable of not only memorizing specific sequences, but is capable of logical induction and deduction as well.

The graph above shows the anomaly score for the network, which is effectively the network's error. Higher anomaly means the network was not expecting the correct token. After a few hundred sentences, the anomaly drops near zero. The histogram represents the distribution of anomaly scores for the testing data.
Despite the seemingly impressive results, it seems the encoding mechanism for the network was generating similar encodings for all inputs. This meant the network could just predict all outcomes for each sentence.



Test with second HTM layer

After the shortsightedness of the previous experiment, I was able to replicate similar results without less pronounced encoder problems, albeit over a larger number of timescales. In addition, I added a second layer to my HTM network utilizing a custom-made persistent firing mechanism to bridge temporal and spatial timescales: basically, I implemented an exponential decay function to maintain the activations of the first layer in a "working memory" layer outside of the main HTM network, and supplied these activations to the second layer as input. The anomaly for the second layer is seen in green, while its moving average is seen in red.
Even though the encoder problem is still present, I suspect that by refactoring my encoding logic to group semantically similar tokens together with similar encodings should alleviate this error for further testing. Regardless, the core problem of the anomaly logic is that just because the network predicts a token, doesn't mean that it's the best pred.



Visualization of HTM layers

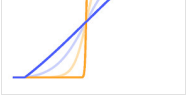
Above is a visualization of different HTM layers in action.

The top layer shows the predicted cells for the next timestep given the current activations. The second layer depicts the predicted cells for the second layer. The predictions for the second layer seem to cluster around certain areas, which is potentially problematic.

The third layer, with the word below it, illustrates the activations for the current input token at the input layer. In this photo, the network is still early in its learning, so the representation has no active neurons yet.

Below the activations for the input layer are the activations for the second layer, which receives an input consisting of the aforementioned working memory layer which can be seen below in the square-shaped figure.

Looking forward from HTM



Besides issues of properly encoding data, despite HTM's amazing ability to memorize sequences with far less training epochs than RNN's, HTM still falls short of the capabilities of RNNs, transformers, and other modern ML models.

I will continue my experiments with HTM and fix issues with the encoder before moving on.

I am planning my next steps around developing my next AI architecture with heavy inspiration from HTM. I will utilize matrix operations and NumPy, and opt for a more continous, less discrete approach than HTM. In effect, I will be using surrogate gradients in hopes of breaking free of HTM's inability to form logical generalizations, which I think is due to this discrete nature. Continuity works better in the ML world, in contrast to the binary, CPU optimized algorithmic nature of HTM.