J-NAV Project for Electronics Design Lab

Kieran Gupta, *Student Member, IEEE*
Blaze Sanders, *Student Member, IEEE*

***Abstract* — Current navigational technologies are unable to provide on-site navigation from building to building at large campuses and very few alternative solutions exist. Herein we describe J-NAV, a comprehensive GPS-based hardware solution that provides building to building directions at the Johns Hopkins University Homewood Campus. Since J-NAV utilizes a standardized database for storage of GPS coordinates and path information, given the proper data about a location J-NAV can be programmed to provide navigation solutions at any campus location.**

***Index Terms* — Campus, Global Positioning System, Navigation.**

I. Introduction

PRESENTED is a Progress Journal for our Electronics Design Laboratory Project entitled "J-NAV", a handheld portable GPS device that provides visual walking directions to any building at the Johns Hopkins University Homewood Campus. The J-NAV device has an optional enhancement of prerecorded descriptions and photos of relevant buildings in order to provide an enhanced audio-guided tour experience. Potential uses for this device include on-campus tours (administered by the Admissions Office), self-guided tours for visitors and VIPs, and navigational aids for new students or staff.

II. Background Research

A. Existence of a Need

There exists a large need for handheld navigation devices for use at many different types of localized campuses where existing navigational technologies such as Google Maps and satellite imagery do not provide enough detail to navigate from building to building. These locations (generally referred to as "campuses" in this paper) include college campuses, army research bases, corporate headquarters, and many other locations.

A large number of individuals often get lost while attempting to navigate on these campuses. The popular self-help website eHow.com even provides a page entitled "How to Avoid Getting Lost on a College Campus."[1] This proves the usefulness of a device dedicated to navigating on campuses which could be issued to visitors or new employees and students.

B. Related Research

There is significant research occurring at universities and laboratories which involve mobile computing platforms and GPS technology. Such "mobile mapping" initiatives are being used primarily to augment in-class learning with in-field activities and to assist with environmental and geographical surveying. For example, Vassar College implements Tablet PCs and GPS technology as part of a course in the Department of Earth Science and Geography[2].

However, there is very little research being done on developing a mobile device for providing building to building navigational assistance at campuses.

C. Related Products

Universities such as MIT and Stanford have developed iPhone Applications that provide functionality similar to that of J-NAV. The iStanford App allow users to view an interactive map of campus and determine their current location using the iPhone integrated GPS [3]. The MIT Mobile iPhone App allows users to view real-time GPS tracking information for the university shuttle system and also provides a searchable interactive campus map that can inform the user of what services are located in each building [4]. MIT utilizes an online interactive map (based on the Google Maps API) which shows a detailed map of campus with visible walking paths and information about each building [5]. Carnegie Mellon University has an iPhone App with many features one of which includes the ability to find your location on campus and thus manually navigate around campus [6].

However, none of these products provide the user with step by step directions on the best way to walk from present location to a destination building. In addition, none of these products feature an audio component. Furthermore, the iPhone Apps require that the user have an iPhone (an expensive device with an even more expensive data plan) and adequate cell phone reception to download the data since no data is stored locally on the device. The MIT online map requires a computer and an Internet connection and also does not indicate the user's present location.

III. Defining Functions and Features

A. Core Features

The following is a high-level description of the core features of the J-NAV device:

1) A color OLED screen output with a touchscreen input that allows the user to select a destination and see a real-time 2D map of his current position and the best path from present position to destination.
2) Optional audio-tour component which allows the user to plug in headphones into the device and hear pre-recorded audio descriptions of the different buildings. These audio files are stored on a microSD card and can be easily updated via a computer equipped with a standard SD card slot.
3) A high-performance rechargeable battery that will allow the device to operate for a minimum of 1hour. The theoretical operation period is 1.45 hours, since the hardware will draw approximately 1.8A and the rechargeable battery is rated at 2600mAh.

B. Supplemental Features

The following is a high-level description of the secondary features of the J-NAV device, which may be included in future versions of the device or if time permits:

1) A digital compass and accelerometer to collect extra data to help better guide the user in the correct direction and prevent mis-directions.
2) Standardization of the built-in database and an algorithm that computes paths between locations, such that a user can enter a set of new locations into the database in a standard form (such as latitude, longitude, path length) and the device can automatically compute the best paths between all possible locations. This will allow the device to be used on different campuses (such as JHMI, JHU APL, JHU East Campus, etc).
3) Different types of paths for different levels of accessibility ("high-mobility" path which may include stairs or steep slopes, "handicap-accessible" paths which involve only ramps and elevators)
4) Code and hardware optimization that will regulate power usage and processor usage in order to provide extended battery life on a single charge.

IV. Hardware Components

The following is an overview of the primary hardware components and the associated inputs and outputs where relevant.

A. Parallax Propeller CPU Chip

A 32-bit processor with 80MHz of processing power, 8 processor cogs, and 32 I/O pins for peripherals.

B. Color OLED Display

A 128x128 pixel 16-bit color OLED with an onboard controller, built-in microSD card slot and significant open source software available.

C. Nintendo DS Touchscreen

A 4-wire resistive analog touch-screen which can be interfaced with using an ADC.

D. GPS Receiver

A compact 1Hz GPS receiver with 5-10m position accuracy that outputs GPS data in adherence with the NMEA protocol.

E. Battery Charging Cradle

The device has an integrated 7.2V 2600mAh lithium-ion rechargeable battery which should be sufficient for a guaranteed minimum of 1.00 hour of continuous high-current operation (significantly longer time under typical operating conditions). We have also designed a "charging cradle." When the user places the J-NAV device into the charging cradle, a switch will automatically disconnect the battery unit from the J-NAV circuitry and connect it to the charging terminals on the charging cradle. This prevents the user from using the device while charging, resulting in problematic simultaneous charging and discharging of battery.

The J-NAV prototype consists of three 0.1-inch spacing 3.2" x 1.8" prototype boards inserted into a 5.7" x 3.3" x 2.0" black plastic project box. Injected molding plastic from Proto Labs, based on our 3D CAD models will be used for internal structure. Accurate CNC milling of project box will be performed on campus, to ensure fast precise fitting of touch-screen and control buttons on exterior of project box. An air circulation vent has been included on the back panel of the casing to allow air flow for the CPU chip and to provide manual access to the processor chip reset button.

A 2 amp quick fuse and a 4 amp NPN power transistor have been utilized to design recharging circuitry and control total device current draw.

V. Designing the PCB and hardware layout

Design of the PCB was completed using expressPCB[7] and the hardware layout design was completed using Autodesk AutoCAD[8]. Both tools allowed visualization and design of the physical layout and also permitted consideration of design constraints such as clean GPS signal reception, heat transfer, compactness, and easy and comfortable touchscreen manipulation. The figures below show a high-level circuit schematic of all hardware components in the J-NAV device.

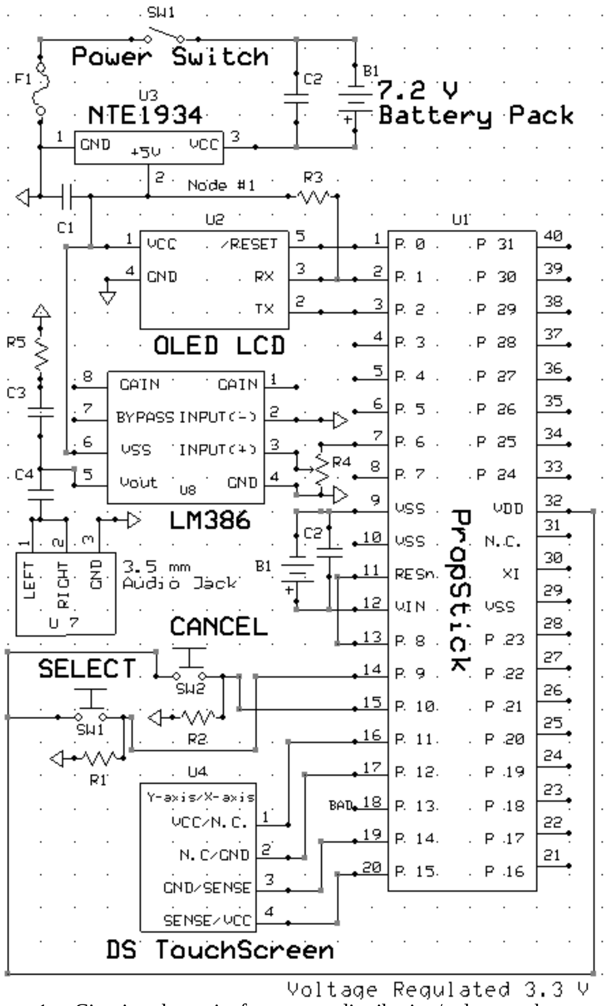

Fig. 1. Circuit schematic for power distribution/voltage subsystem, user interface subsystem (LCD, pushbuttons, touchscreen) and audio subsystem.

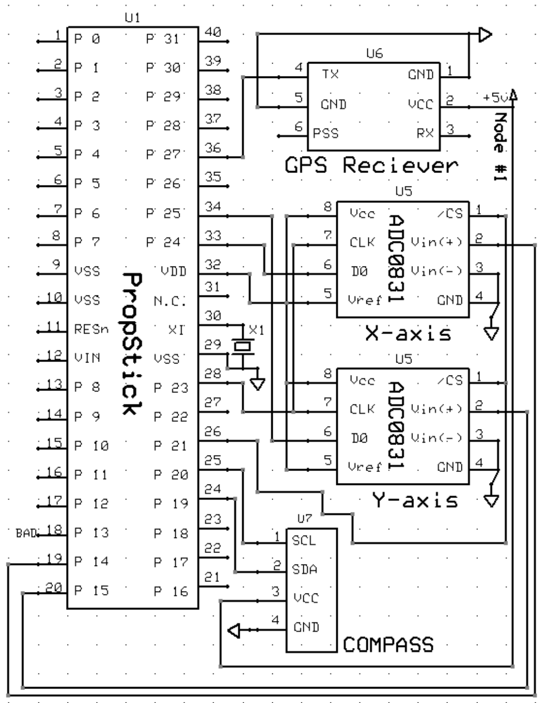

Fig. 2. Circuit schematic for navigation subsystem (GPS and digital compass) and touchscreen ADC.

See the Appendix for specific values of discrete components such as resistors and capacitors in the circuit schematics.

VI. Designing the User Interface (UI)

The preliminary design of the user-facing tactile interface will allow the user to interact with the J-NAV device. Four touchscreen directional arrows will be used to navigate the destination menu and to provide alternate functionality. Two physical pushbuttons will be used as "OK" and "Cancel" buttons to allow for user interaction with on-screen menus.

VII. Software Development

We utilized Google Earth desktop software to create a database of GPS coordinates (latitude, longitude) of all relevant buildings on campus. For each building, we recorded the GPS coordinates of a single entrance point to that building. While many buildings have multiple entrances, we chose the single most accessible entrance (based on location of the entrance in reference to other buildings and based on when the entrance is regularly unlocked).

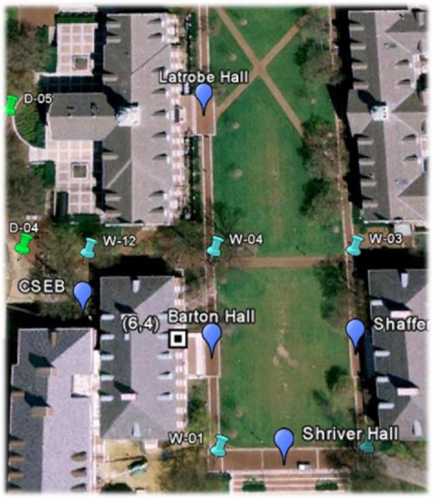

Fig. 3. A screenshot showing how we used Google Earth software to strategically place 60 nodes throughout campus in such a way that all buildings could be reached by traversing straight paths between nodes.

After collecting the coordinates of all the buildings, we proceeded to define supplemental waypoints or "Nodes." A Node is an intermediary GPS coordinate (between locations). A Node is typically an intersection on a pathway, and two Nodes are considered "linked" if there exists a walk-able path between the two Nodes. We optimized paths so as to create the least number of supplemental Nodes. Note that a building entrance GPS coordinate is technically a special type of Node (with a potential end destination at that Node). Using this approach, a path from one building to another is simply a sequence of linked Nodes. This is the approach used by most standard GPS navigation software.

In order to determine the path between Node A and Node B, we created an adjacency matrix which indicates which Nodes are directly linked via a path. This allows a near constant-time lookup of any path between two locations.

In the future, a desktop software tool (based on a Java graphical user interface) will be created as an add-on to this project. This software tool will allow for quick creation of the Node adjacency matrix by using a click-and-drag graphical approach.

Approximately 20 kilobytes of 32 kilobytes of available RAM have been utilized, with approximately 4,500 lines of code. Currently 3 of out the 8 parallel cogs/processors are being utilized. One cog for collecting GPS data, a second cog for collecting touch-screen and button input from user, and a third cog for calculating path solution between locations. Debug code is being written directly into application code to allow future work on the project. Java-style documentation is being used to comment code, again allowing for future work on project. Additionally, J-Share is being used as code repository.

A high-level explanation of our main driver program is as follows:

1. User turns on the J-NAV device and is prompted with the splash screen logo.
2. User waits for GPS receiver to acquire satellites and lock on to current position.
3. User is allowed to select an end destination from the list of all possible buildings on campus. Store this user input as `bldg_selection`, a number from 0-26 which corresponds to that building's index in the `LOC` and `PATH` arrays.
4. Find the nearest node by traversing the entire `LOC` array. This array stores the GPS coordinates of all the Nodes. For each index, `idx`, determine the distance between `LOC[idx]` and current location. The shortest distance is the nearest node. Store this index as `nearest_node`.
5. Determine the sequence of nodes from the nearest node to the end destination node by accessing the `PATH` array. This array is a 2D adjacency matrix which stores all the links (walkable paths) between nodes. Access `PATH[nearest_node, bldg_selection]`, store result as `next_idx`. Then repeatedly access `PATH[next_idx, bldg_selection]` until `next_idx = bldg_selection`. This is the complete path from current location to end destination building. Store this in a global array.
6. Display the path on-screen, node by node. Only display the next part of the path when the user has entered within range of the next sequential node. Before displaying each part of the path, extract any hazard data from the path information (stairs, roadway hazard, etc) and display this hazard information on screen.
7. When the user reaches the end destination, display the appropriate destination image and information on screen. Wait for user interaction before returning to Map mode.

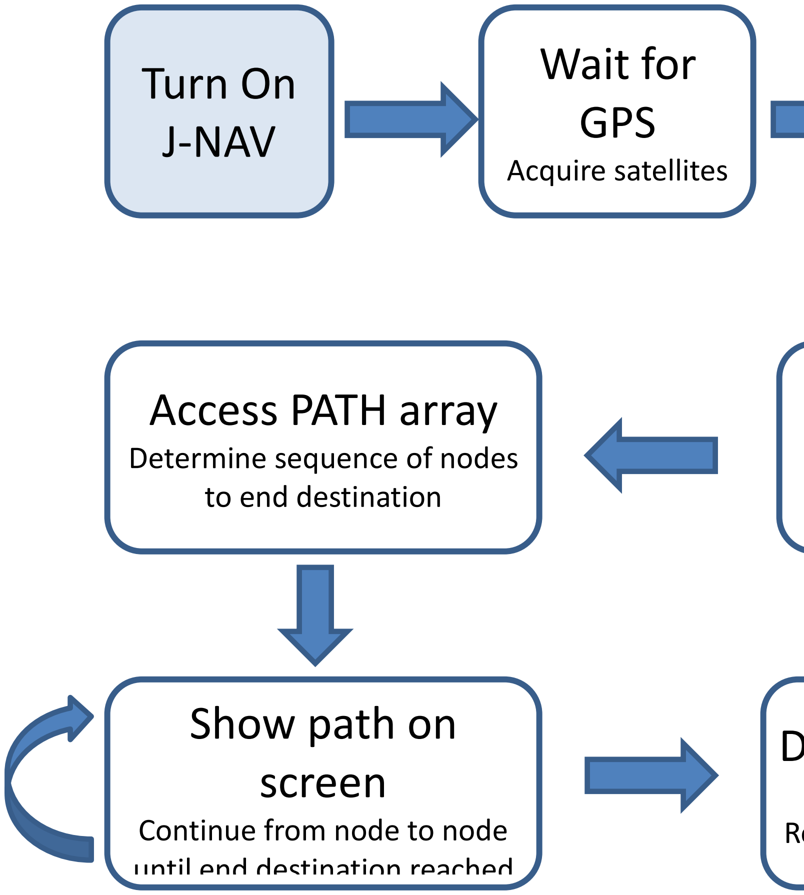
Turn On J-NAV
Wait for GPS
Acquire satellites
Access PATH array
Determine sequence of nodes to end destination
Show path on screen
Continue from node to node until end destination reached

Fig. 4. A flow chart showing the high-level steps (including user interaction) during normal operation of the J-NAV unit beginning with device start-up and ending at the user's selected final destination.

VIII. Algorithms

A. *Processing GPS Input Data*

The GlobalSat GPS receiver outputs data in simple ASCII format, in adherence with the NMEA 0183 protocol standard. There is vast documentation on interpreting data in this standard. We are utilizing open-source code to parse the GPS receiver output into latitude and longitude coordinates (as floating point numbers) which can in turn by used by our main driver program.

B. *Outputting Data to OLED Display*

The OLED screen uses a standard serial communication protocol. There is ample documentation of sample code for interfacing a Propeller Chip with the OLED. We used a freely available software called Graphics Composer[9] to segment a large map of campus into 128x128 mini-images which will then be uploaded to the microSD card in sequential memory locations. Our driver program will update the screen display with the relevant 128x128 mini-image which shows the user's current position. As the user moves out of the area visible in that 128x128 mini-image, the driver program will display the next image.

There will be two main modes of operation for the OLED. In "Map" Mode, the OLED will display the user's current position on a real-time updated 2D map of campus, with buildings and paths clearly visible. The map will update as the user moves. Optionally, the user can manually adjust the view of the map by using the directional arrows.

In “Location” Mode, the OLED will display a list of all available buildings on campus and will allow the user to select a destination building using the directional arrows. Once the user has selected a building, the OLED will switch to “Map” Mode, displaying the user’s present position and outlining the appropriate path to traverse to reach the destination.

C. Processing Touchscreen Input

The touchscreen uses a simple 4-wire resistance model to determine which part of the screen the user is touching. After applying analog-to-digital conversion, the values will be passed through a mathematical function which will compute and X- and Y-coordinates of the user’s touch which will be mapped to specific operations.

D. Reading Data from SD Card

Data can be processed from the microSD card using Serial Peripheral Interface Bus (SPI mode), a synchronous serial communication protocol that operates in full duplex mode. Audio files stored on the microSD card (in .wav format) will be read and output to the audio headphone jack. Since data only needs to be read from the microSD (and no data needs to be written), coding is simplified.

E. Outputting the Audio Signal

Audio files will be processed and output to the audio headphone jack using a power amplifier circuit.

F. Data Storage of Map

An image of a map of Homewood campus will be stored on the microSD card. The map will have buildings and paths clearly identified. Square sections of this map will be displayed on the OLED screen, overlaid with the user’s current position (represented as a dot) and the computed path to the destination (represented as a line).

A set of arrays (storing floating point values) will contain the latitude and longitude positions of all the buildings. A large two-dimensional adjacency matrix will store all the different conventional paths that can be traversed between buildings. When a path from one building to another is computed, it will actually be computed by connecting adjacent markers until the destination marker is reached. This computation can be implemented using Dijkstra's algorithm, which solves the single-source shortest path problem [10].

IX. Conclusion

The J-NAV device works as expected, successfully navigating the user between any buildings on campus. The primary issue with the device is accuracy. GPS technology itself has 5-10 meter accuracy, and due to further inaccuracies in the calculations within the software algorithms, the accuracy of J-NAV is approximately 10m. However, the accuracy of J-NAV will improve as the existing infrastructure technology is upgraded. A new fleet of GPS satellites is schedule to be launched. These satellites will provide enhanced civilian GPS signals, allowing for 1 meter accuracy and GPS navigation within buildings.[11]

There is a significant amount of future development that can add enhanced functionality and compatibility to the J-NAV device. In terms of hardware, audio processing hardware is required to provide audio feedback for the user. In addition, multiple sensors can be implemented to augment the accuracy of GPS positioning. A digital compass and a digital accelerometer can be used to provide improved positional awareness.

In terms of software, photos and descriptive text for all the buildings can be included so that J-NAV can also act as an on-campus informational reference. The necessary software infrastructure for this functionality has already been programmed. In addition, a more attractive graphical user interface can be designed. Lastly, different accessibility option can be programmed into the J-NAV device. Information about the difficulty of different paths (stairs, roadway hazard) is already included in the software. Using this information, an intelligent path-creation algorithm could generate multiple paths to the same destination, depending on the accessibility level of the user.

Appendix

The table below provides a summary of the Bill of Materials for construction of the J-NAV device.

Component	**Manufacturer**	**Cost**
Color Screen	4D Systems	$ 65.00
GPS Receiver	US GlobalSat	$ 59.95
Touchscreen	Nintendo	$ 9.95
Touchscreen connector	--	$ 3.95
Propeller Chip	Parallax	$ 79.99
Li-ion Battery	Tenergy	$ 59.99
Miscellaneous Parts	RadioShack vendor	$ 10.00
Total Cost		**$278.83**

Table 1. Bill of Materials for J-NAV hardware

The following shows an outline of the source code files used in our J-NAV software as well as file dependencies.

- **JNAV_Main_Driver**
  - PathMatrix
  - PushButtonController
    - ADC0831_Driver
  - GPS_Float_Lite
    - GPS_Str_NMEA_Lite
  - Clock
  - FullDuplexSerialPlus
  - FullDuplexSerial *[Library File]*
  - FloatString *[Library File]*
  - FloatMath *[Library File]*

The table below provides specific component values for the circuit schematics in Fig. 1 and Fig. 2.

Capacitor Values	**Resistor Values**	***VDD = 3.3V***
C1 = 100 μ F	R1 = 1000 Ω	**Fuse**
C2 = 1000 μ F	R2 = 1000 Ω	F1 = 2 A

C3 = 0.05 μ F	R3 = 1000 Ω	
C4 = 250 μ F	R4 = 100 kΩ	**Crystal**
	R5 = 10 Ω	X1 = 5 MHz

Table 2. Discrete component values for circuit schematics

ACKNOWLEDGMENT

The authors would like to thank Dr. Ralph Etienne-Cummings, Electronics Design Laboratory Professor, for his assistance and recommendations throughout the design process. In addition, the authors would like to acknowledge the following individuals for permission to adapt their source code for use with the J-NAV project: I. Kovesdi for use of his GPS interface code, Steve McManus for use of his OLED interface code, and Austin Bowen for use of his ADC conversion code.

REFERENCES

[1] "How to Avoid Getting Lost on a College Campus." http://www.ehow.com/how_2205035_not-lost-college-campus.html .
[2] "Mobile Mapping Using Tablet PCs and Geographic Information Systems (GIS) in Field-based College Courses." 7 October 2008. Vassar College. http://www.cfkeep.org/html/snapshot.php?id=71791110627159 .
[3] "iStanford" iPhone Application by Terriblyclever Design, LLC. http://itunes.apple.com/us/app/istanford/id292922029?#
[4] "MIT Mobile" iPhone Application. http://m.mit.edu/about/iphoneapp.html
[5] MIT Campus Map. http://whereis.mit.edu/
[6] CMU App iPhone Application by Carnegie Mellon Univ http://www.cmu.edu/cmuapp/
[7] ExpressPCB Software. http://expresspcb.com/
[8] Autodesk AutoCAD Produce. http://autodesk.com/autocad
[9] Graphics Composer. 4D Systems Pty Ltd. 2009.
[10] Mark Allen Weiss. *Data Structures and Algorithms Analysis in Java*. Pearson Education, 2nd edition. 2007.
[11] "GPS Power-Up: Get Ready for New Sense of Place." April 19, 2010. Wired Magazine. http://www.wired.com/magazine/2010/04/st_gpsreboot

Kieran K. Gupta (B.S. '11) is an electrical engineering and computer engineering undergraduate student at Johns Hopkins University. He has experience commercializing new technologies as Enterprise Manager of Hopkins Consulting Agency, a business and technology consulting company. In addition, he has software engineering experience working with Northrop Grumman Corporation and research experience working at the Johns Hopkins University Applied Physics Laboratory. He is also Co-Leader of the Johns Hopkins University Space Elevator Team, a multi-year research project with the goal of designing and building a prototype space elevator.

Blaze Sanders (A.S. '08, B.S. '10) is a student at Johns Hopkins University pursuing a concurrent Bachelor's / Master's degree in electrical engineering. He has industry experience working at NASA Johnson Space Center and NASA Marshall Space Flight Center in astronaut human factors and robotics respectively. He founded and co-led the Johns Hopkins University Space Elevator Team. He is SCUBA certified and has plans to obtain a private pilot's license, Class A skydiving license, and Electron Beam welding certification.