Project Report for Raspberry Pi Companion Computer and Sensor Package

By Brandon Hickey, Bachelor of Science

A Project Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in the field of Mechanical Engineering

Advisory Committee:

Jeffrey Sabby, Chair

Mingshao Zhang

Keqin Gu

Graduate School

Southern Illinois University Edwardsville

May 2022

Abstract

By Brandon Hickey

Chairperson: Professor Jeffrey Sabby

• Structure the title ABSTRACT and the title of your manuscript in the same

format as your major/chapter headings.

• Double space to the first line of text.

• Double space the text

• The abstract must start on its own page.

Table of Contents

**No table of contents entries found.**

• Major headings (ABSTRACT), chapter headings (INTRODUCTION) and second

level headings (Problem Statement) if included are required to be shown in the Table

of Contents; however, third, fourth, and fifth level headings are optional.

• Do not include the Table of Contents in your Table of Contents

• REFERENCES and APPENDICES are major parts of your thesis. They should be

formatted like the major parts in your front matter and should be placed flush left.

These are not counted as chapters since they are major sections like ABSTRACT.

• Page numbers must be justified to the right and fan out to the left. Your page

numbers must line up under each other to the right.

• Connecting dots must connect your heading and the page number.

• Do not allow your wording to stick out over your page numbers. If this happens,

move the wording to a run over line.

• Run over lines are indented at least to the first letter of the line above.

List of Figures and List of Tables

• Each list should be positioned on its own page in the front matter

• Run over line indented at least 3 spaces.

• Right justify of page numbers.

• LIST OF FIGURES or LIST OF TABLES heading at the top of the page should be

formatted like the rest of the major parts of your thesis.

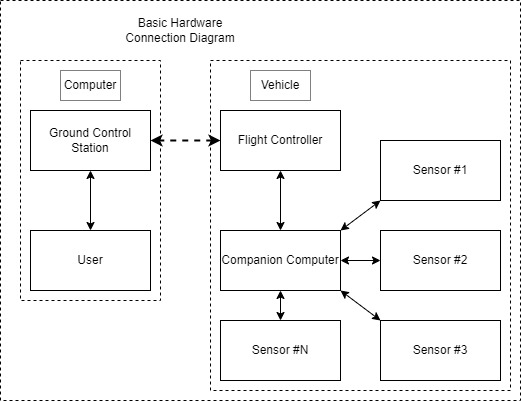
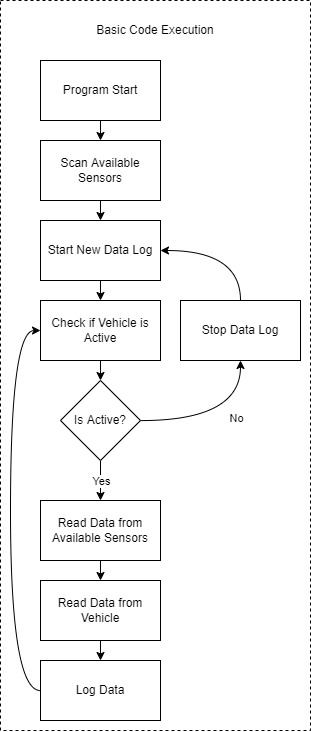
• Headings/captions MUST match the entry in the List of Figures or List of Tables.

• Single spacing within entries and double space in-between entries.

• Connecting dots to page numbers

Body

<https://web.mit.edu/course/21/21.guide/th-form.htm>

1. Introduction
   1. The goal of this project is to develop a raspberry pi based sensor package that would perform sensor measurement from additional onboard sensors and combining these measurements with sensor data from the flight controller in a single data log and provide data feedback to the user through a single telemetry system.  
        
      This project is design to work with STEM kits developed through the Department of Defense National Defense Education Program and to provide potential application to additional areas in environmental data sensors.  
        
      One fundamental use of remotely operated vehicles is to gather information. Unmanned vehicles often employ several different sensors like GPS or IMU to provide information regarding state and orientation of the vehicle. This information can then be used by the vehicle for movement and be sent back to the user. However, a vehicle could be developed with the goal of gathering information like temperature or humidity levels in different locations. This is not very useful to the operation of an unmanned vehicle, and with each additional sensor added, the vehicle must take more time to perform these data measurements along with maintain its desired movement.   
        
      Instead we can use what is known as a companion computer. “A companion computer is a device that travels on-board the vehicle and controls/communicates with the autopilot over a low-latency link. Apps running on a companion computer can perform computationally intensive or time-sensitive tasks, and add much greater intelligence than is provided by the autopilot alone.” What this means is that we can assign tasks that do not aid in the operation of our vehicle to the companion computer instead. In our case, we will use the companion computer to perform sensor measurement to measure a variety of environmental states and log this information along with the vehicles current location and orientation.   
        
      To do this, we will program our companion computer with the capability of interacting with an array of possible sensors that could be equipped depending on the kit and environment. The attached sensors should be easily changed with minimal to no change in our companion computer’s software.   
        
      The result of this project is that we were able to design a companion computer that is able to read data from \*X TYPES\* of sensors for environments including underwater. The companion is also able to send and receive data from both the flight controller and ground control station, allowing us to log sensor data with vehicle location and orientation. We have also included the capability to record video data using a raspberry pi based camera.
2. Background
   1. DoD NDEP Project
      1. This project is conducted as part of a research grant by the Department of Defense (DoD) through the National Defense Education Program (NDEP) with the objective of developing STEM education and outreach initiatives for Highschool and Undergraduate students with a focus on the fields of rocketry and space. Our focus is on the development of educational kits that will provide students with the tools and instruction necessary to create vehicles associated with space and exploration. The expected kits include a variety of vehicles that include rockets, drones, rovers, and submarines. As part of this program, we will use a controller that can used by any kit with minor changes required to existing controller configuration. To satisfy this requirement, we have chosen to use the Navio2 Autopilot Hat and a Raspberry Pi 4 microcomputer as our flight controller.
      2. \*Image\*?
   2. Navio2
      1. The Navio2 is a preconfigured HAT, which is an add-on board that can be mounted to a Raspberry Pi to expand its functionality. The Navio2 is designed to be a flight controller and contains additional sensors not found on the Raspberry Pi. These include a GNSS Receiver for satellite tracking, a high-resolution barometer for altitude, dual inertia measurement units (IMU) that provide accelerometer, gyroscope, and magnetometer readings, and finally additional input and output connections. To operate the Navio2, the Raspberry Pi is required to run an autopilot software called Ardupilot.
      2. Image
   3. Ardupilot
      1. Ardupilot is an open-source autopilot software that is used throughout the drone and remote-control community. It is configured to work with six different vehicles including copter, plane, rover, and submarine. Ardupilot is run on the Raspberry Pi along with a Robot Operating System (ROS) program known as MAVROS which provides communication for autopilot systems using a communication protocol known as MAVLink. MAVLink provides the ability to send and receiving data from the flight controller as well as sending commands from our ground control station to our vehicle.   
           
         A ground control station, or ground control software, (GCS) is the platform such as a phone or computer application that is connected to the flight controller through wireless connection such as Wi-Fi or radio telemetry. There are several different software available of which we have chosen to use a program known as Mission Planner. A ground control station provides numerous benefits. We can observe real-time data of our flight controller to include features such as altitude, speed, heading, GPS location, and battery life. Additionally, we can configure missions and waypoints for autonomous movement and limit the speed and distance of our vehicle with respect to the location of our ground control station.
   4. Desire to expand the usage of our kits
      1. Using just the Navio2 and Raspberry Pi, we are able to fully control our various kits as well as look at real-time data being transmitted back to our ground control station. However, we lack the ability to add additional sensors to our flight controller. The Navio2 uses the almost all available GPIO (General Purpose Input Output) pins of the Raspberry Pi, leaving only three available. Of these three, two of them are UART pins, leaving only a single GPIO pin. This means we are unable to connect with sensors that require I2C and SPI communication protocol. While the Navio2 does have the capability of reading two analog pins, they are coded for sensors that would be beneficial to the flight controller such as an analog air speed sensor rather than a sensor that provides little to the operation of the vehicle.
   5. Technology that exists
   6. The Aim
   7. Our Purpose
      1. The overall focus of our research (DoD NDEP) is the development of 10 STEM kits for high school and undergraduate level students that will cover technical areas related to space and rocketry. To do this we looked to using a single flight controller that could be used interchangeably throughout all ten kits. Our choice is the Navio2 autopilot HAT, which is an attached module for a Raspberry Pi system to act as our flight controller for all kits. With the Navio2 + Raspberry Pi, we are able to operate all of our kits, however we lack the ability to gather a large array of data during each kits operation. The Navio2 requires the use of most of the Raspberry Pi’s GPIO (General Purpose Input Output) pins , leaving only three pins free from limited use. Including limited pins, there is also the risk of slowing down the flight control system by trying to add sensors that would be not used by the flight controller and only for data logging. In order to allow for data collection with limited impact on the flight controller performance, we choose to design a second raspberry pi as a companion computer that would receive data from both the flight controller and ground control station and include that in its own data logging to provide location and time specific sensory readings that could be used interchangeably with each individual kit without requiring different programs.
   8. Is it previously done?
      1. While there is a relatively large amount of documentation regarding the development of companion computers, the are generally based off of a similar hardware system (Raspberry Pi, Nvidia, Arduino) and then are customized by the user for their specific desire. Our companion computer therefore, will be an excellent template that could be used throughout different Navio2 systems given its interchangeability and limited impact on the Navio2.  
           
         \*Maybe more on different existing companions, include how they are similar, support my version, what mine does differently, and what deficiencies it would address\*  
           
         There has been work done with attaching companion computers to other types of flight controllers, however I have found limited information regarding using a Navio2 flight controller, which we are choosing to use because of the STEM kits we are also developing for the DoD Project.
   9. Why Are we Doing This
      1. While the stem kits do not require these extra sensors, the development of the capability will allow the kits to be used for more than just learning and understanding in space and rocketry. Including sensors like temperature, pressure, humidity, and other environmental features would expand the potential of the kits allowing those with more limited experience and knowledge in the development of kits such as these to perform data collection for their own use.  
           
         The issue with using only the Navio2 is the lack of sensors. The sensors onboard the Navio2 are limited to just those that are used in the process of flight: GPS, Barometer, and IMUs. This provides little to collecting extraneous data about the environment. Depending on how the kit is used, it can become a sensor collection platform
   10. Constraints
       1. Constraints for this system is that the companion computer will have a low strain on the flight controller, be useable between different kits, and store data from both the flight controller and the companion together for viewing.
   11. Methods and Tools
   12. Mavlink
       1. Raspberry Pi, python, coding,? Topics include cameras/optics, remote sensing, communications, data acquisition, telemetry, mobility, rocketry, robotics, and programming. Individual kits will typically involve integrating microprocessors (Arduino, Raspberry Pi, Jetson Nano…), programming (C, C++, Python), sensing, actuation, and mechanical hardware.
   13. The Aim
       1. The aim of this project is to develop a companion computer system to interface with a Navio2 flight controller and use additional arrays of sensors to collect environmental data throughout the vehicles operation and to record this information along with vehicle information either in real-time or after mission completion. Research Question(s): In order to show viability, this project must be able to operate on more than one vehicle type without needing to modify the companion computer software.
   14. Extra
       1. Again, long descriptions of details are to be avoided and references to suitable sources of detailed information should be given instead. Other background information could consist of the sequence of events leading up to the present situation or the results of earlier investigations. You could also discuss such things as any cost or time constraints imposed on the project. Your background section should end with a clear statement of the research questions problem your project is trying to answer. These will reflect the aim of your project, but will be different in that they explain the problem you are attempting to solve, e.g.,
3. Methodologies
   1. System Requirements
      1. Given the desired capability of our Companion Computer, there are several features we aimed to adhere to for our system to operate as desired. The first and foremost is that our companion computer must be able to interact with the Navio2 flight controller. The Navio2 is the one unchanged feature between all of our kits, and therefore our Companion Computer must be compatible with it.  
           
         Because we are looking for a low-impact companion computer, we also limit the number of physical connections between the flight controller and companion which are power and communication. Since all sensor measurements are done through the companion, we only require a power supply and an ability to communicate to the Navio2 and ground control. With a single communication line we are able to not only interact with the Navio2, but also able to take advantage of the features of the Mavlink communication protocol to also send and receive messages from the ground control station through the Navio2 telemetry link.  
           
         Along with the capability of performing all sensor measurements on just the companion computer, we also want to be able to keep track of and compare the system time of both companion and flight controller. This will allow us to ensure that environmental data we collect will correspond to the correct location data provided by the flight controller.  
           
         Given that the STEM kits we are developing are made to cover several different environments, it will require a companion computer that is able to be used in each location without complex changes in between missions. For example, if we wanted to measure an environments pressure, we would require different type of pressure sensor depending on if the kit was a drone flying through the air, or a submarine traveling beneath the water. To this end, the companion system should be able to change sensors without requiring any change in its programming. To go one step further, it would be even more beneficial if we are able to change our equipped sensors without shutting down the power (Hot-swappable).   
           
         Since we want to log both the sensor data collected from the equipped sensors and data such as vehicle location and orientation, our system must be capable of sending and receiving data from both the physically connected flight controller and the wirelessly connected ground control software. Being connected to the ground control station means that our companion can send information back to the user in real-time. While this data is not being logged, it can provide the user with information regarding current data measurements and could even be used to develop alarms or other messages sent based on the data collected.  
           
         Finally, all data from the companion computer and desired information from the flight controller should be logged in a single location for later review and analysis. The sensor and vehicle data should be reported in a format that is simple to view and extrapolate from.
   2. Hardware Connections
      1. The basic setup of our system will be similar to that of \*FIGURE X\* shown below. We have two main components of our system: computer and vehicle. For our computer, we have a relatively limited number of sections to focus on, the main one being the ground control station software.   
         
   3. Software Execution
      1. Include Schematic  
         
4. Design
   1. The purpose of the Specification and Design sections is to give the reader a clear picture of the system you plan to create, in terms of the capability required The design then gives the top-level details of how the software system meets the requirement. It will also identify constraints on the software solution, that are important in guiding decision making throughout the development process.   
        
      how it does so (design)
   2. Hardware Connection
      1. Given these specifications, we have developed a design of our companion computer that will serve as the basis for our system and future changes.
      2. Single power Connection Either from the BEC or from Servo rail of the Navio2
      3. UART Connection between Navio2 and Raspberry Pi for communications
      4. Include Schematic
   3. Software Connection
      1. I2C and Analog Sensor Communication
      2. Data Storage on SD Card
      3. Communication to Ground Control and Navio2 Through Mavlink Protocol
      4. Include Schematic
   4. Software Execution
      1. Include Schematic
   5. Extra
      1. A common approach is to first define the user or business requirements, then describe the static architecture, identify modules and groups of closely connected modules, and then to apply other views to each of these groups. Fine details, specifically details of code, should be left out.
      2. We strongly recommend that you make extensive use of diagrams, such as entity-relationship diagrams, UML diagrams, state charts, or other pictorial techniques (see Section 5.4 for more detailed advice on this).
      3. As well as describing the system, it is important that you justify its design, for example, by discussing the implications of constraints on your solution and different design choices, and then giving reasons for making the choices you did. Typically these implications will relate to the aims of the project and to aspects of it discussed in the Background section.
      4. The design of the system will almost certainly have evolved while you were developing it. Obviously you should describe its final state but often there are good reasons for describing intermediate states, too; for example, if you want to discuss the details of the design method used or to highlight learning that you later refer to in the Reflection section.
      5. If you do this, take special care to make sure the reader does not get confused between different stages of the design. If you are not designing a system, but testing a hypothesis for a more scientifically oriented project, specification and design sections may not be required in quite the same form (see Figure 3.3 and Figure 3.4). The specification instead becomes a description of the problem and what is required of a solution. The design becomes a description of your approach to solving the problem and your suggested soltuion(s). For instance, if you are designing an algorithm to solve a particular problem you would have a problem statement section and then a section describing one or more suggested algorithms to solve the problem. Later in the - 12 - Results and Evaluation section you then describe how to design experiments to test how well the algortihm(s) solve the problem and present your experimental results with an evaluation of your suggested solutions.
5. Implementation
   1. The Implementation section is similar to the Specification and Design section in that it describes the system, but it does so at a finer level of detail, down to the code level. This section is about the realisation of the concepts and ideas developed earlier. It can also describe any problems that may have arisen during implementation and how you dealt with them. Do not attempt to describe all the code in the system, and do not include large pieces of code in this section. Complete source code should be provided separately (see Appendix B and submission guidelines). Instead pick out and describe just the pieces of code which, for example:   
      • are especially critical to the operation of the system;   
      • you feel might be of particular interest to the reader for some reason;   
      • illustrate a non-standard or innovative way of implementing an algorithm, data structure, etc..   
        
      You should also mention any unforeseen problems you encountered when implementing the system and how and to what extent you overcame them. Common problems are:   
      • difficulties involving existing software, because of, e.g., o its complexity, o lack of documentation;   
      • lack of suitable supporting software;   
      • over-ambitious project aims.   
        
      A seemingly disproportionate amount of project time can be taken up in dealing with such problems. The Implementation section gives you the opportunity to show where that time has gone.
6. Results and Evaluation
   1. In this section you should describe to what extent you achieved your goals. You should describe how you demonstrated that the system works as intended (or not, as the case may be). Include comprehensible summaries of the results of all critical tests that were carried out. You might not have had the time to carry out any full rigorous tests – you may not even got as far as producing a testable system. However, you should try to indicate how confident you are about whatever you have produced, and also suggest what tests would be required to gain further confidence. This is also the place to describe the reasoning behind the tests to evaluate your results, what tests to execute, what the results show and why to execute these tests. It may also contain a discussion of how you are designing your experiments to verify the hypothesis of a more scientifically oreinted project. E.g., describe how you compare the performance of your algorithm to other algorithms to indicate better performance and why this is a sound approach. Then summarise the results of the tests or experiments. - 13 - You must also critically evaluate your results in the light of these tests, describing its strengths and weaknesses. Ideas for improving it can be carried over into the Future Work section. Remember: no project is perfect, and even a project that has failed to deliver what was intended can achieve a good pass mark, if it is clear that you have learned from the mistakes and difficulties. This section also gives you an opportunity to present a critical appraisal of the project as a whole. This could include, for example, whether the methodology you have chosen and the programming language used were appropriate.
7. Future Work
   1. It is quite likely that by the end of your project you will not have achieved all that you planned at the start; and in any case, your ideas will have grown during the course of the project beyond what you could hope to do within the available time. The Future Work section is for expressing your unrealised ideas. It is a way of recording that „I have thought about this‟, and it is also a way of stating what you would like to have done if only you had not run out of time1 . A good Future Work section should provide a starting point for someone else to continue the work which you have begun.
8. Conclusion
   1. The Conclusions section should be a summary of the aims of project and a restatement of its main results, i.e. what has been learnt and what it has achieved. An effective set of conclusions should not introduce new material. Instead it should briefly draw out, summarise, combine and reiterate the main points that have been made in the body of the project report and present opinions based on them. The Conclusions section marks the end of the project report proper. Be honest and objective in your conclusions.
9. Reflection
   1. We believe in the concept of “lifelong learning”. One of the principles applied throughout the assessment during your studies is that of the value of reflection. We believe that it is important that we reflect upon our performance in order to identify “transferable learning”, that can be carried over into future activities. Reflection should focus on what Argyris calls “double loop learning”; this is where we identify, not relatively “simple skills”, such as the mastery of a new programming language, but the impact of what we have done on the assumptions, concepts and ideas we used to make decisions about our work. For example, a “reflective practitioner” would try to identify the characteristics of the problem that has been addressed, and consider whether assumptions or decisions about the relevant approach to solving that problem had been appropriate, in order to make a better decision in relation to problems that might be encountered in the future.
10. References
    1. In Section 2 we said that you should relate your work to that of other people. Other work explicitly cited should be listed in the Reference section and referred to in the text using some kind of key. It is important that you give proper credit to all work that is not strictly your own, and that you do not violate copyright restrictions. It may be desirable to provide a Bibliography section separately from the reference section. In general, references are those documents/sources cited within the text. The bibliography lists documents which have informed the text or are otherwise relevant but have not been explicitly cited. References should be listed in alphabetical order of author‟s surname(s), and should give sufficient and accurate publication details. For example, Chikofsky, EJ, Cross, JH. 1990. Reverse Engineering and Design Recovery: A Taxonomy. IEEE Software, 7(1):13-17. Date, CJ. 2000. An Introduction to Database Systems, 7th Edition. Addison-Wesley. are acceptable references. There are various conventions for quoting references. For example, you can quote the name of the author and the year of publication, e.g. For more information see [Chikofsky et al, 1990]. A more detailed description is given by Date [2000]. There are several other variations. For example, some authors prefer to use only the first three or four letters of the name, e.g. [Chi1990] or just to number the references sequentially, e.g. [3]. It can be helpful to the reader if, for books and other long publications, you specify the page number too, e.g. [Date 2000, p. 23]. Whatever convention you choose, be consistent. Information Services provide a number of leaflets which describe in detail accepted ways of presenting references. For example, guidance on the Harvard Style of citing and referencing may be viewed at http://www.cardiff.ac.uk/insrv/resources/guides/inf057.pdf. Whatever style of referencing you adopt, it is critical that you are assiduous in acknowledging the sources you have used; failure to do so may lead to suspicions of unfair practice and an investigation into whether or not your work reflects the standards expected of academic research. Guidance on plagiarism and how to avoid it is available at http://learningcentral.cf.ac.uk/bbcswebdav/institution/INSRV/S tudy%20Skills/plagiarism2/new/index.html. Note that it is seldom sufficient to simply “cut and paste” material from other sources. When you take material from someone else‟s work, you are doing so because it helps support your argument, or justify decisions you are making. It is therefore essential to make it clear why - 15 - you have included material from other sources; in other words, you need to critically assess the work of others, whether it is supporting your position or not: • If the material you are citing from another source supports your position, you must explain why it should be trusted. For example, material from a published journal will, normally, have been peer-reviewed and can therefore be considered to have some validity, according to subject matter experts. Much of what is published on the Internet cannot be regarded in the same way, however. • You will often find that there are conflicting views in the published material; in such cases you must explain which view you favour and why, before relying on the material to support your position. • If other writers have taken a different position to the one you support, you must explain why the reader should accept your ideas rather than those proposed elsewhere. In summary, you need to ensure that you have clearly assessed the relevance of referenced material to the development of your position, or your argument, and demonstrated that you are justified in taking this material to be authoritative.

* Introduction and objectives
* Proposed research
* Results
* Conclusions
* Future work

* Introduction to the project NDEP
* Dicussion of the desire for additiona sensor package, with potential reference to the environmental department and the potential of further use by adding an external sensor package that would be able to connect to multiple kits with only a need for a powe connection and single usb line in order to take different readings while getting flight controller data at teh same time to compare data to location and system run time.
* Bibliography

Bibligraphy

Appendix

<https://www.researchgate.net/publication/328354840_A_Study_of_automated_image_capturing_HDI_environment_using_NAVIO_2>

1. <https://ieeexplore.ieee.org/ielaam/43/8496924/8412592-aam.pdf>

* Literature review
  + Sources
    - <https://dl.acm.org/doi/pdf/10.1145/3444950.3444958>
    - <https://ieeexplore.ieee.org/abstract/document/8869659?casa_token=0VUJdv3XQGQAAAAA:BncpcmkcF2gH3NStYU2cxmLOeGxrFhqcxR8KzNLHeD1GH_dJ4eFMJKDLhKDG7tBM9uZ6-3DGEQ>
    - <https://www.researchgate.net/publication/334707975_A_Full_Distributed_Multipurpose_Autonomous_Flight_System_Using_3D_Position_Tracking_and_ROS#pf2>
  + Navio2 is limited in expanding sensors through the available GPIO Pins
    - <https://community.emlid.com/t/how-to-use-navio2-gpio-free-pins/22123>
    - <https://docs.emlid.com/navio2/dev/pinout/>
    - “IO17 and IO18 pins on UART are available. GP26 (pin 37) on Raspberry Pi is also free. It may be used with lengthened extension header.”
    - There are 3 free pins available after connecting the Navio2 system. GPIO 17,18, and 26. This is a limit in terms of being able to attach additional sensors to use these pins. In our sensor package we plan to use a variety of communication protocols such as SPI, UART, and I2C. There are not enough physical pins available
  + Navio2 Is meant to be as real-time as possible for regulating and controlling our vehicle.
    - As we increase the demands on the system, trying to add additional sensors, cameras, and data logging: we will bog down the flight controller, leading to potential issues in performance
  + There is a “potential” lack of external sensor packages in use with the navio2
    - This will be something I need to look at more since I am not certain. There is not a lot of information in terms of using the Navio2 for more scientific areas. I would like to look more into some of the items on sources I have given I have not seen much in the way of a Navio2 working with a companion. I do see one source that looks like using a Navio2 with a raspberry pi (whether this is just as a HAT or not we will see) to do imagery.
    - <https://dl.acm.org/doi/pdf/10.1145/3444950.3444958>
    - <https://ieeexplore.ieee.org/abstract/document/8869659?casa_token=0VUJdv3XQGQAAAAA:BncpcmkcF2gH3NStYU2cxmLOeGxrFhqcxR8KzNLHeD1GH_dJ4eFMJKDLhKDG7tBM9uZ6-3DGEQ>
    - <https://www.researchgate.net/publication/334707975_A_Full_Distributed_Multipurpose_Autonomous_Flight_System_Using_3D_Position_Tracking_and_ROS#pf2>
    - <https://ieeexplore.ieee.org/document/8574657>
    - In terms of searching the Emlid forums and other places on the internet for people using the Navio2 in sensor packages similar to ours, it does lack a definitive presence. In terms of the instructions available on the Emlid documentation, the code examples are only runnable when the Ardupilot Flight Control system is inactive, meaning you could not try to perform the sensor examples they provide while also performing the required actions of the flight controller, showing a demand for a companion system to perform the additional actions.

<https://pats.cs.cf.ac.uk/wiki/lib/exe/fetch.php?media=project-report.pdf>