

Pilot Enhanced Guidance and Augmented Sight Utility System (P.E.G.A.S.U.S.)

Luke Anderson, Frank Murphy, Ethan Dill

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

Abstract — The Pilot Enhanced Guidance and Augmented Sight Utility System (or “P.E.G.A.S.U.S.”) is a compact heads-up display system intended to enhance situational awareness for pilots in general aviation aircraft, such as a Cessna 172 or other comparable private civilian aircraft. By utilizing external flight parameter sensor input data, the P.E.G.A.S.U.S. system is able to project its graphical user interface through a series of optics onto an optical combiner to overlay the HUD onto the forward view of the pilot’s environment. This increased visibility of flight parameters allows for a pilot to be able to gain flight context and insights without having to direct their attention away from their in-flight line of sight.

I. INTRODUCTION

The student-led P.E.G.A.S.U.S. project is a portable, stand-alone heads-up display (HUD) for small aircrafts. This system is no larger than a briefcase and is designed to rest atop the dash of an airplane in line of sight of the pilot and provides essential flight data to improve situational awareness. Currently, no affordable and portable HUD option exists for small aircrafts. Typically, a new aircraft either comes with a built-in HUD or does not. Alongside the lack of product in this space, the cost of purchasing an aircraft with a HUD raises the price of the aircraft astronomically. Hence, the P.E.G.A.S.U.S. project, which is designed to be an affordable and portable HUD that pilots can utilize to improve situational awareness in new ways. This system uses a 9-axis inertial measurement unit (IMU) to provide attitude and heading data as well as an altimeter to provide pilot with pressure and altitude data. These sensors are read and processed by an ESP-32 microcontroller unit (MCU). From there, a Raspberry Pi4 receives data from the ESP-32 of current flight data then displays the received data in a graphical user interface (GUI) that is projected on to a screen attached to the system. The image from the GUI is then circularly polarized and projected through a Petzval Lens system which creates and

magnifies the image as well as corrects for distortions / aberrations. Then, the image gets folded via a series of mirrors to ensure a compact package and is then projected onto dielectric glass and then a self-dimming element. This ensures that the image is projected at infinity, not doubled, and visible against the backdrop of a pilot’s forward-facing environment. From there, the pilot reads the data and can know critical flight data in real-time without having to move their eyes from the main field of view to examine instruments on the dash. This system aims to fill a gap in the aviation industry by bringing portable and reliable HUDs to more aircraft.

II. SYSTEM CONCEPT

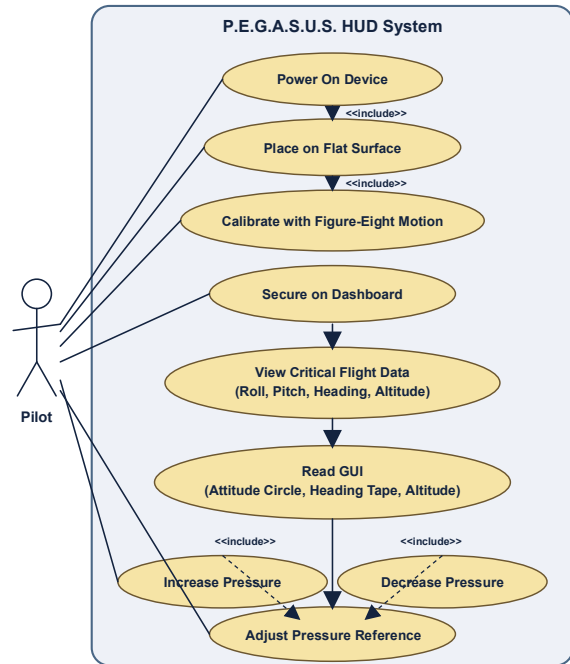


Fig 1. P.E.G.A.S.U.S. Use Case Diagram [1]

A. Overview

This system’s use is straightforward in design. The system is designed around the Cesna 172 aircraft, the target user is pilots who pilot these planes. As such, the display must present critical flight data in an easy to digest format. Hence the process to use this system is simple. First, the pilot turns the device on and places it on a flat surface for a few seconds to calibrate. Once it has sat on a flat surface, the pilot must calibrate it by rotating it in a figure eight motion. This calibration is crucial as it ensures minimal

drift and high accuracy of sensors. From there, the pilot places and secures the device on their dashboard and aligns it with their vision.

Now that the device has been powered on, calibrated, and secured, it may be used by the pilot. The pilot will see a drawn graphical user interface with a rotating attitude circle in the middle that corresponds to their roll and pitch. On the top of the GUI, there will be a heading tape that displays compass heading. At the bottom of the GUI, the measured numbers for roll and pitch are displayed to the pilot in case the center attitude circle is not precise enough. Then finally, on the right is a rectangle that contains the altitude in feet.

This altitude is related to the local pressure and as such, a system has been designed to allow the pilot to adjust the pressure of the device for altitude calculations. This adjustment has been made simple by using two buttons. One button increases the pressure, and one decreases the pressure. The pilot can reference the pressure gauge on the exterior of the system housing to accurately assign current pressure to the device.

B. Hardware

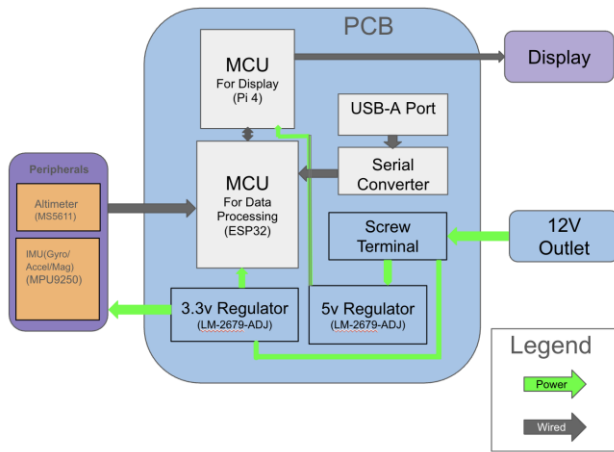


Fig 2. Hardware Block Diagram

The block diagram above shows how the system will interact electrically. All the green lines indicate power and grey means there is a communication line. Starting with, the most important thing, the power source. It will flow into the system using a screw terminal. It will then split off between the 3.3V and 5V regulators. From here the 5V regulator powers, the Raspberry Pi 4 and the 3.3V regulator is responsible for the two sensors and the ESP-32.

Next, looking at the wiring for the data, it starts at the USB port where the code can be downloaded to the ESP-32 and the Raspberry Pi 4. It cannot go direct to the ESP-32, or the data would not work, so it first passes through a serial converter to be translated to UART. After this it is then passed onto the MCUs. The data comes in from the two sensors to the ESP-32 where it is processed and then sent to the Raspberry Pi 4 to create the display. This is where it is lastly passed along to the screen to make the final picture.

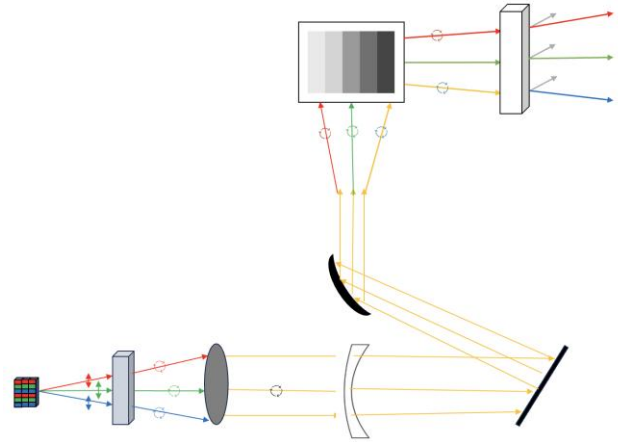


Fig 3. Optical Hardware Diagram

The above diagram illustrates the optical path flow as well as the relevant system components. Linearly polarized light is indicated with vertical arrows and circularly polarized light is indicated with circles. From left-to-right, starting at the bottom left, the components are as follows: LCD Display Screen → $\lambda/4$ (quarter wave) polarizing film → Achromatic Doublet (positive Petzval) element → Negative Meniscus (Negative Petzval) corrector element → Flat Fold mirror → Curved Fold Mirror (to the top left of the fold mirror) → photochromic film (above the curved mirror) → Optical combiner (to the right of the photochromic film on the top of the diagram).

It should also be noted that the diagram illustrates the difference at the optical combiner interface of one of the images being transmitted through the dielectric glass vs. one image being reflected. This illustrates the elimination of a “ghost” image.

C. Software

The software overview is best thought of as a single chain of events that go from the sensors to the display. First, the sensors read the attitude, altitude, and heading data. This

data is then processed by the ESP32. On the ESP32, data fusion and filtering is done to ensure accurate data is being recorded and outside interference is minimal. From here, the ESP32 packages up the data and sends it over UART to the Raspberry Pi4. The Raspberry Pi4 acts as our graphics engine due to limited functionality of the ESP32 to render graphics. Upon receiving the data from the ESP32, the Raspberry Pi 4 decodes the CBOR packet and assigns the values to their corresponding variables. These variables are then displayed to the pilot in a GUI. The pilot may then read the attitude, altitude, and heading data and make necessary flight adjustments as needed. Alongside the display, data is periodically logged to ensure debugging is easy.

III. SYSTEM COMPONENTS

The best way to breakdown the system is to look at all the different individual components. This will include all of the designed and purchased parts that were used to be able to build the device we need.

A. Microcontroller

The most important part of this project is the ESP-32 S3 WROOM. This chip was chosen for its availability for coding libraries and its simplistic board layout. It has a dual core 32-bit processor that is more than capable of computing needs and has sufficient memory. We are not utilizing it, but for future improvements it has Wi-Fi and Bluetooth capabilities. It operates at 3.3V and requires at its peak 350mA.

While the ESP-32 is capable of creating an image for the display it is not its strong suit. With this in mind a Raspberry Pi 4 was used in tandem to create a better image. The Pi 4 has a quad-core 64-bit processor and has both the USB-A and HDMI built onto the board to connect directly to the display. Unlike the ESP-32 the Pi 4 is a real application processor making it more capable for connecting to the screen and creating a quickly updating high clarity image. The Pi 4 runs at 5.1V and uses 3A of current.

B. Altimeter

One of the two sensors for this project is the MS5611 Barometric Pressure Sensor. It is capable of measuring precise altitudes from -500m to 10000m. It is accurate within 10 cm. It is powered by 3.3V rail and operates a low current using less than 2mA. It also has a secondary feature with a built-in temperature sensor. It has a small package but has side pins to make it solderable, and I2C capable of

being able to attach via a data and clock signal to the ESP-32 for computing.

C. Inertial Measurement Unit

The second sensor is the MPU-9250. It has 9 axes of measurement. There is a 3 axes accelerometer, a 3 axes gyroscope, and a 3 axes magnetometer. These all work together to get the full orientation of the roll, pitch, and yaw. All the resolutions are 16-bit, and it is another low power system. All the axes combined use up to 4mA and operate at 3.3V. It is another small package, but it has the same side pins to be able to properly mount and solder. It is also I2C capable of making it simple to hook up to the ESP-32 with the data and clock lines.

D. Serial Converter

For the ESP-32 to be able to be coded, it requires the USB data to be converted into UART serial data. It does this by taking in the D+ and D- from the USB and taking it into the device. It then translates the data to UART and sends it to the USB using a transmit and receive lines.

E. Regulator

The power supply for the system is a 12V 10amp wall outlet. This is more than enough power than we need. There are two different voltages needed and cause of this the LM-2679ADJ was selected. This can be used for both the 3.3V and the 5.1V. While we don't need it the regulators are capable of taking in anywhere between 8V to 40V and outputting 1.2V to 38V. It is also capable of outputting 5A, and neither the 3.3V or 5V rails require more than 3.5A.

F. Display

The micro-display used for this system is a TFT Liquid Crystal Display that is 640 x 480 pixels. This thin-film transistor display has transistors integrated within each pixel to ensure a high contrast and bright image. This allows for good visibility and readability of the system in various ambient lighting environments. In addition, the resolution of the display, especially at the small form factor of the physical display itself, is still high enough to ensure a crisp HUD image and graphical user interface for the pilot. In addition, this display allows for a competitive bill of materials due to the fact that it sits at a low cost.

G. Quarter Wave Polarizing Film

In order to make our system visible within a series of different conditions, a quarter wave polarizing film is incorporated after the display. Because of the fact that the output from the TFT screen is linearly polarized, the light emitted from this device is not able to be viewed in a polarized setting, such as with polarized sunglasses,

common in the aviation industry. To solve this problem, a quarter wave polarizer was introduced to circularly polarize the light emitted from the display in order to make it visible in various polarization conditions.

H. Petzval Lens System

The lens system chosen to project the image from the display into the rest of the optical components was a Petzval Lens system. Lens architecture allows for the separation of a positive and negative lens element as well as the stop in the system. This is advantageous for a couple of reasons. First of all, a single positive component, such as a large diameter achromatic doublet, excels at the image formation and magnification of the image from the TFT display. The magnification of achromatic doublets is governed by the thin-lens equation:

$$M = \frac{h_i}{h_o} = \frac{d_i}{d_o} \dots (1)$$

Where M is the magnification, h_i is the image height, h_o is the object height, d_i is the image distance, d_o is the object distance.

Once this image is formed, however, various aberrations are introduced, the most prominent of which is pin-cushion distortion, leading to distorted edges with a more in-focus center. This is to be expected with the image output of an achromatic doublet (as it is a positive lens element).

To combat this pin-cushion distortion, the second element in the Petzval architecture is introduced. This is a weak negative corrector, in this case a negative meniscus lens. Because the two elements are separate, this allows you to shift the stop of the system (which is between the positive and negative element). With the positive achromatic doublet having a (relatively) small focal length of 75mm and the negative meniscus lens having an effective focal length of -500mm, this system allows for a compact lens architecture that pairs nicely with our compact housing.

I. Fold Mirror System

The fold mirror architecture allows for fitting the optical path into a compact form, allowing for the project to meet its housing constraint and be able to be properly integrated into the tight spaces in an aircraft. The fold mirror will fold the optical axis so that the image is able to be projected upward onto our optical combiner, allowing it to be increasingly visible to our pilot.

J. Optical Combiner

The dielectric glass optical combiner allows for the overlay of the final magnified and projected HUD image onto the forward-facing environment. Essentially, this combiner offers two distinct contributions to the system. First of all, the dielectric glass eliminates what is called a double image or a “ghost” image. If the HUD were to be projected onto regular glass, such as the actual cockpit glass of a Cessna 172, a double image would appear and the heads-up display would have a distorted output. The optical combiner “combines” lets only one of the images pass through (via its transmissive property) while displaying the proper image (via its reflective property). The second contribution that this dielectric glass has to the system is the fact that in it allowing for some light to reflect (our HUD image) and some light to transmit through (the pilot’s forward-facing environment), it allows for our image that is projected at infinity to appear as if it is part of the pilot’s external environment or overlaid on that environment. This is the chief advantage of a heads-up display as the pilot does not have to divert visual attention away from their surroundings to scan their instruments when their instrument data is right in front of them.

IV. HARDWARE DETAILS

A. Electrical Hardware Details

All of the system components used on the electrical side that were mentioned previously in section III besides the Raspberry Pi 4 due to its complexity and the device being purchased not designed, will be further explained here.

With the MCU being the most important part of the board, the selection of the ESP-32 was what the rest of the board was revolved around. While it is not the fastest and is not designed for peripherals like the STM-32 is, its capability is more than enough for the scope of this project. The only data that needs to be processed is coming from the two different sensors.

All this data will be transferred across the SCL and SDA data lines. SCL stands for Serial Clock Line, and the SDA stands for Serial Data line. These two create the two-line communication bus for I2C. The SCL is responsible for when to read and write data and is ultimately a timer. The SDA is what passes along all of the sensor readings, commands, and addresses. The best part of I2C is that all of the lines for different sensors can be connected. Both the MS5611 and the MPU9250 will connect their Data and clock lines together and are still able to communicate separately. This makes it much easier to wire considering there are only two pins on the ESP32 required to get the required data from the sensors.

To collect this data the MS5622 and the MPU9250 use precise tools and set reference numbers to make precise measurements. The MS5611 measures pressure by using a tiny MEMS diaphragm. It will push air onto it and measure the difference in the internal electrical signal. It can then calculate the altitude based on this pressure. The pressure will have an exact correlation with an accurate as long as the baseline pressure is set before flight.

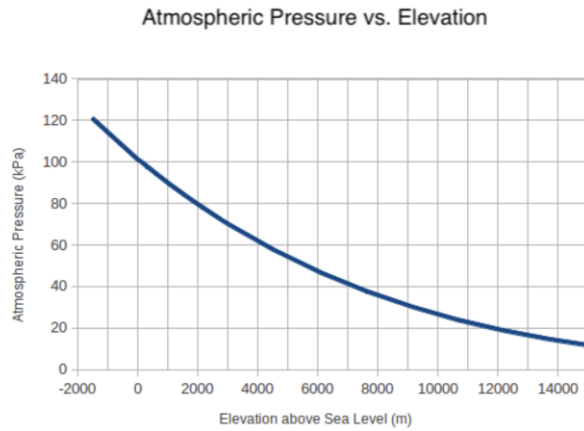


Fig 4. Pressure vs Elevation Chart

The IMU works a little differently and instead uses three different three axes systems to create a full picture. The accelerometer will measure the tilt and the gravity direction. This combined measure linear acceleration. The gyroscope measures the rotation rate and how fast it is turning. Lastly, the magnetometer measures the earth's magnetic field and acts like a compass.

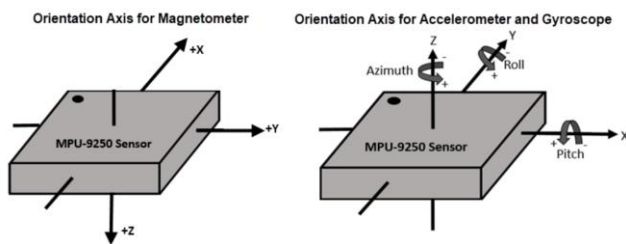


Fig 5. MPU-9250 Sensor Axis Orientation

The serial converter while it does not affect the system while in flight it is crucial to making sure the ESP-32 know what it is supposed to be doing. The Serial Converter only connects between the ESP-32 and the USB port. It takes in the USB data which comes in as packets. Inside the serial converter, a USB controller decodes the packets into simple bytes. The decoder is then able to send out the packets as

TX and bit by bit serial data. This is the format the ESP-32 is able to understand and carry out.

B. Optical Hardware Details

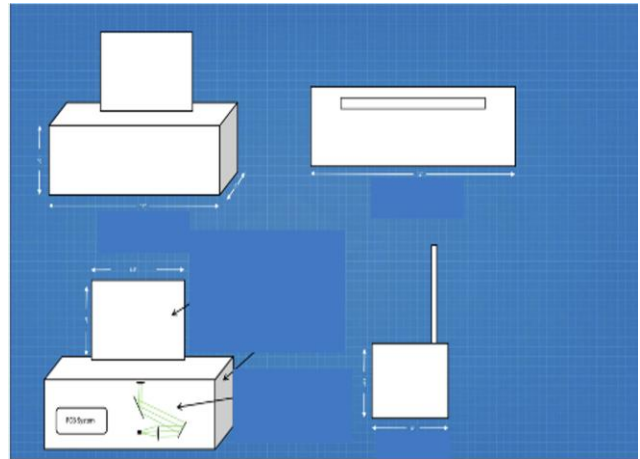


Fig 6. Housing Hardware Diagram

In regard to the optical hardware used for this project, the following hardware is used. The LCD used is a TFT LCD with a 640 x 480 LCD screen with a diagonal of 3.5 inches. The polarizing film is a $\lambda/4$ wave film that converts linearly polarized light into circularly polarized light. The achromatic doublet used is a 2 inch diameter ThorLabs Achromatic Doublet with a focal length of 75mm. The negative meniscus lens also has a 2 inch diameter with an EFL of -500mm. The mirrors are standard 3 inch x 3 inch flat fold mirrors. The photochromic film spans the dimension of the optical combiner. The dielectric glass optical combiner is a 60/40 Transmissive/Reflective (T/R) glass.

The mechanical hardware used to integrate the various optical components of the system is an optical cage assembly with rails. An optical rail assembly was considered to perform this function but ultimately was not chosen as the rail system that was designed used optical posts and benchtop components that were not suitable for an aircraft system. This cage configuration allows for ease-of-use in terms of greater system alignment and also ensures that our compact form is met. In addition, portions of this system can be 3d printed with PLA filament. To create rigidity and packaging / housing for the mirrors and the optical combiner for the system, various mounts and holders were designed in SOLIDWORKS as well as

OnShape to be able to be printed as well. This allows for fixed system components with respect to the system optics.

V. SOFTWARE DETAILS

To ensure the data is useable to the pilot, it must be both accurate and current. If either of these conditions are not met, the result may be catastrophic for the pilot. Old data can be just as problematic as incorrect data. This mentality is crucial to the software of this system.

A. Overall System Flow

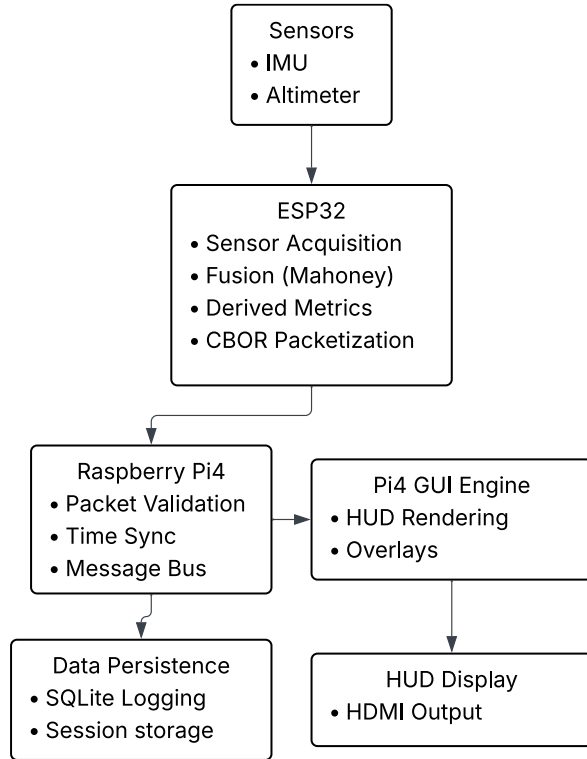


Fig 7. Overall System Flow

The overall system flow is referenced in Fig 7 where we go from sensors all the way to the HUD display. To summarize this flowchart, we start from our 9-axis IMU and altimeter. These are used to measure attitude, heading, and altitude. The altitude is computed using formula (1). Attitude encompasses roll and pitch which are computed using formulas (2) and (3) respectively. Heading has been more complicated due to magnetic interference with various solution approaches having been taken.

$$h = 44330 \left(1 - \left(\frac{P}{P_0} \right)^{0.1903} \right) \dots (2)$$

$$\phi = \text{atan} \left(\frac{2(q_0 q_1 + q_2 q_3)}{1 - 2(q_1^2 + q_2^2)} \right) \dots (3)$$

$$\theta = \text{asin}(2(q_0 q_2 - q_3 q_1)) \dots (4)$$

$$X_h = m_x \cos(\theta) + m_z \sin(\theta) \dots (5)$$

$$Y_h = m_x \sin(\phi) \sin(\theta) + m_y \cos(\phi) - m_z \sin(\phi) \cos(\theta) \dots (6)$$

$$\psi = \text{atan} \left(\frac{Y_h}{X_h} \right) \dots (7)$$

The current solution involves a hybrid approach between a Mahony filter and direct yaw correction. The base of this correction starts with tilt compensation shown in formula (7) which is used to correct heading for roll and pitch. Roll and pitch affect heading due to the use of the accelerometer in the IMU. In the standard Mahony filter, magnetometer data is computed using a cross-product correction similar to that done in accelerometer calculations. However, this correction has errors around 180° of error which can cause system failures. To solve this, a yaw correction is performed that compares current yaw estimate with magnetometer-based heading as shown in (insert). This correction is then applied to the heading calculation as a small rotational adjustment to produce a more stable heading. All calculations are performed on the ESP32 and then packeted in a CBOR packet which is sent to the Raspberry Pi4. This way, there is no polling delay and packets are sent as soon as they are ready.

$$e_\psi = \psi_{mag} - \psi \dots (8)$$

$$\Delta\psi = e_{\psi K_p dt} \dots (9)$$

Once the Raspberry Pi4 receives the packet, it is validated for any issues caused during transmission. If any issues are found, they are corrected and bad data is thrown out. If a time sync error occurs, a resync is attempted which then leads to the ESP32 and Raspberry Pi4 syncing up again. If they cannot sync, this will produce a system error which is logged in a SQLite database. Assuming the data is correctly received, it is processed and rendered by the Qt graphics library which is then displayed over HDMI to the LCD.

B. ESP32 Flow

The ESP32 begins by booting up FreeRTOS, recalling that FreeRTOS was the chosen OS for the ESP32 because it allows the fastest development with the lowest overhead. From here, peripherals such as I2C, SPI, UART, and GPIO are initialized to allow for the sensors to be read by the system. After, the sensor read tasks are initialized to the corresponding frequencies for those sensors. When the sensors are being read/booted up, the data must be “cleaned” and outliers in sensor readings must be purged hence the calibration and preprocessing steps. To best calibrate this system, it is best to place the system on the desired surface and let it calibrate for 5-10s. After, the ESP32 performs data filtering and fusion using Mahony filters and direct yaw correction to allow for minimal overhead and fast data fusion (citation 61).

After fusion, the data is processed as outline in the system flow and then packeted in a CBOR packet. This packet is a lightweight serial packet designed to transmit easily over UART. It contains the essential fields pertaining to altitude, attitude, and heading. After the packet is sent, the process starts over again and again as long as the device is powered on.

C. Raspberry Pi 4 Flow

The Raspberry Pi4 begins by automatically loading the GUI upon startup. This is used with process directives that run immediately upon startup that load the GUI to the full size of the Pi4 display. A listener is opened upon startup as well to receive the CBOR packets over UART. These packets are read, parsed, and assigned to their corresponding variables. After parsing, the data is then formatted in the GUI and displayed to the pilot.

Alongside the GUI display, there are various logging procedures to ensure errors can be caught and debugged easily. These procedures are rather straightforward, the main one being related to error logging. The error logger takes any and all errors encountered such as timing out of sync, incorrect packet parsing, or any other errors and inserts them into a SQLite table. Another crucial logging process is the system logs which say when the system was booted up and when certain tasks execute. The final data logging piece is related to the CBOR packet data. Roughly every second, one full CBOR packet is logged, this is done to allow the developer access to the raw CBOR data in case there is issues with data accuracy that must be resolved. Alongside all data entries are time stamps to ensure the time related to data, logs, and errors is known.

VI. CONCLUSION

The P.E.G.A.S.U.S. Heads-Up-Display system was created to provide a solution for increasing situational awareness in civil and general aviator's operation flow without having a cost-prohibitive design and implementation model. With the system that we have built, the HUD allows for a pilot to have a general overlay of flight parameter data incorporated with their view of their front-facing environment. This allows for better situational awareness as the pilot does not have to divert their attention from their environmental view to scan instruments. In implementing the design for this system, there were a series of hurdles. Namely, integration of various sensor architectures with the overall system as well as optical image formation and distortion correction. As with any successful engineering endeavor, these hurdles were able to be overcome. Through the implementation of manual sensor calibrations for sensor implementation, the sensor integration issue was able to be solved. Through the use of a Petzval lens architecture, the problem of image formation and distortion correction with optics was able to be solved. Housings and packaging posed a design challenge as well, but we were able to incorporate an optical cage system as well as a broader box-like housing design to fit our heads-up display into a compact, form-fit package that allows for ease-of-installation into a standard aircraft. Overall, we were able to design a compact and affordable solution for the lack of affordable heads-up displays in general aviation aircraft.

BIOGRAPHIES

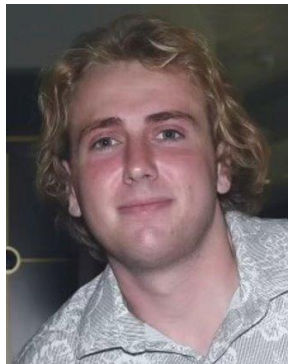


Luke Anderson is a Computer Engineering student with a minor in Physics that will graduate with his bachelors in May 2026. Currently, Luke works as a Software Engineer Intern at Airmeez Inc, working on microservices related to telecommunications services. After graduation, Luke has

plans to work as a Software Engineer full-time and develop SaaS platforms or other software programs.



Ethan Dill is an Optics & Photonics Engineering student and will graduate with his bachelor's degree in Optics & Photonics Engineering in May of 2026. Ethan currently works as a Photonics Engineer and Associate Program Manager at Micross semiconductor components in Apopka, FL. Throughout his undergraduate tenure, Ethan has worked as technician at a laser scanning galvanometer company, worked as a research engineer characterizing photovoltaic solar cells with infrared imaging techniques for the Florida Solar Energy Center/ UCF, has started a projects club at UCF CREOL, been the director of UCF's Optical Society of America Chapter, and much more. After graduating with his bachelor's degree, Ethan plans on working in industry, going for his master's degree, starting his own venture, and/or doing something else with his life.



Frank Murphy is an Electrical Engineering student that will graduate with his Bachelors in May of 2026. After graduation Frank plans to work as an Electrical Engineer at Naval Air Warfare Center Aircraft Division doing Safety and Procedure.

ACKNOWLEDGEMENT

The authors wish to acknowledge their committee members and reviewers for their insight and assistance on this project. Namely Dr. Shin Tson Wu, Dr. Enxia Zhang, Dr. Di Wu. In addition, we wish to thank and acknowledge Dr. Aravinda Kar, Dr. Chung Yong Chan, Dr. Arthur Weeks, and the rest of the ECE and CREOL faculty that assisted on this project.

REFERENCES

- [1] OpenAI. *ChatGPT conversation*. 2026, <https://chatgpt.com/share/69c6fb72-68e8-8326-ac3d-a9ba2a209bc2>
- [2] Espressif Systems, "ESP32 Datasheet," 2016. [Online]. Available: https://cdn.sparkfun.com/datasheets/IoT/esp32_datasheet_en.pdf
- [3] Raspberry Pi Foundation, "Raspberry Pi 4 Model B Datasheet," 2019. [Online]. Available: <https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf>
- [4] TE Connectivity, "MS5611 Barometric Pressure Sensor Datasheet," 2011. [Online]. Available: <https://www.te.com/commerce/DocumentDelivery/DDEController?Action=showdoc&DocId=Data+Sheet%7FMS5611-01BA03%7FB3%7Fpdf>
- [5] TDK InvenSense, "MPU-9250 9-Axis MEMS Motion Tracking Device Datasheet," 2015. [Online]. Available: <https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-9250-Datasheet.pdf>