First Semester Report: Aerobatics Black Box

Maxwell Bakalos, Eli Carroll, Qi Luo, Yanbo Zhu
Team 11 - Aerobatics Black Box,
EC 463 Senior Design Project I, Boston University
maxbak@bu.edu, elivc@bu.edu, tomqiluo@bu.edu, zhuyanbo@bu.edu

***Abstract* – Aerobatics pilots have trouble visualizing their routines from a viewers perspective and figuring out which actions they perform result in a good flight. Our team's solution is an in-flight recording system. After a flight, the pilot can watch the recording back to learn what inputs caused which outputs and improve their abilities. This report outlines our progress on this project, proposed by our client, Dr. Kenneth Sebesta, in our first semester of Senior Design Project at Boston University.**

Keywords – aerobatics, black box, flight recorder

1 INTRODUCTION

Aerobatics is a type of stunt flying where pilots fly aircraft in unconventional maneuvers. This includes things such as loops, rotations, etc. usually done in a specific routine for competition or entertainment. As a pilot, it is difficult to visualize how your movements in the air will be seen from the audience's perspective on the ground. A perfect mid-air loop may look too elliptical on the ground and will fail to impress. Pilots do not know which exact steering angles, speed, and other inputs cause the best flights. Adjusting controls could yield better results, but they have no way of remembering exactly what they did when a flight turned out well. There is no high-tech equipment in the small aerobatic planes that could tell them what they did. The most complex device is often a digital clock!

Our client is an aerobatics pilot who wants to create a flight recording system that can play back the flight as a 3D animation. He is also a professor at BU who has experience with robotics [4]. This animation would show the position of the plane, its orientation, as well as the pilot's inputs and other information as needed. This flight recreation would give the pilot a better understanding of how their aerobatic routine appears to a grounded observer and allow them to improve it.

Fig. 1. Aerobatic planes performing aerial stunts

An existing software [1] will be used to display the flight path animation, so our team is primarily tasked with collecting the data and processing it into a neat format for the software. A BU Senior Design team from last school year started this project, and we are building off their results. However, their entire physical setup has been lost, so we have to rebuild it from scratch. Luckily, their code was mostly saved on GitHub, so we were able to access that for reference, though we will likely rewrite most of it to fit our new equipment.

Fig. 2. Examples of 3D flight visualization: (top) client provided, (bottom) X-Plane [10]

In order to record all of this data there will be multiple sensors installed on the plane including a GPS to record the location (latitude, longitude, altitude), an accelerometer for acceleration, a gyroscope for orientation (pitch, roll, yaw), and cameras to record dashboard meter values (engine RPM, airspeed, etc), as well as pilot steering and pedal motions. Thus there are two main sections of recording for the project: reading from the sensors we implement (GPS, accelerometer, gyroscope) and reading from the plane's information (cameras). Legally, our system cannot permanently modify the plane, so everything has to be easily detachable and non-invasive. This is why cameras must be used instead of directly physically accessing the dashboard.

2 CONCEPT DEVELOPMENT

The requirements from our client can be categorized into six areas, which are its case, performance, camera, GPS, IMU, power, and program. Each of these categories is described below.

A. Case

Since the testing aircraft has a small cockpit, the size of the entire product should be limited to a 15cm x 10cm x 6cm volume. The material should be durable and last more than 10 years without breaking. It should also have enough heat conductivity to release the heat produced by working components inside the box.

B. Camera

Our customer requires us to use cameras to capture the user input and dashboard dial readings, so there should be at least three cameras. The camera for the dashboard dial should be mounted behind the pilot and can be removed and reattached easily. It should not block the pilot's view and his movement and should be facing the dashboard directly. The cameras for the pilot's input contain two parts, the first one is for detecting the pilot's steering yoke or control stick input, so it should be mounted facing the pilot's hand. Another is used for detecting the pedal's movements, so it should be mounted facing the pilot's foot.

All the cameras must have a stabilization function to encounter the vibration of the cockpit that comes from the engine. The stabilization function is not limited to a built-in design of the cameras, it can also be realized by any mounted camera holders or cloud deck. Also, the influence of environmental lights should be considered since the cover of the dials is made of glass, so the reflection will influence the output video. We may use a polarizing lens to filter the lights and shadows.

C. Global Positioning System (GPS)

GPS module is used for detecting location data. It should form a stable connection with the satellites during the flight. However, the antenna must be facing toward the sky since the connection will get lost if the antenna is facing downwards. As a result, there must be at least two antennas facing different directions. The data gained by the GPS module should include the aircraft's longitude, latitude, and altitude information, and the refresh rate can be adjusted from 4 times per second to 10 times per second.

D. Inertial Measurement Unit (IMU)

It should detect the orientation of the plane. The IMU has an accelerometer module that will output the acceleration data in the x, y, and z directions in 3-D coordinates, and we can use some formula to calculate yaw, roll, and pitch of the aircraft based on those data. IMU also has a gyroscope which will directly give the pitch, yaw, and roll data.

E. Power

The main power supply of the black box comes from the 12V power source from the aircraft, which is similar to a cigarette lighter. We will use an adapter and a USB cable to connect the power source to the black box. We may also use 6V batteries as external power supplies for cameras.

F. Program

The programming of the black box is crucial. The requirements for each object are discussed as follows.

1) Camera: Since all the dials on the aircraft are mechanical, they are not able to output any digital data. The only way for us to obtain the data is by using image processing. The program should analyze the dashboard dials, control stick, and pedals and then convert them into a data sheet corresponding to an active time flow. The program should use either Opencv, or MATLAB, and use a manually assigned map to get readings from them.

2) GPS: The program for GPS should synchronize multiple data flows from different GPS modules and combine them as one single data flow. It should also limit the size of the data flow since there won't be enough memory for the storage unit if the data is too large.

3) IMU: After receiving the acceleration data in the x, y, and z directions, we can use the relationship between each of these values and use a program to calculate the yaw, roll, and pitch of the aircraft.

4) X-plane: After receiving, combing, processing, and analyzing the data, we should input the entire data file to X-plane, which is an open-source software, and generate a 3D animation that shows the recorded flight trajectory.

3 SYSTEM DESCRIPTION

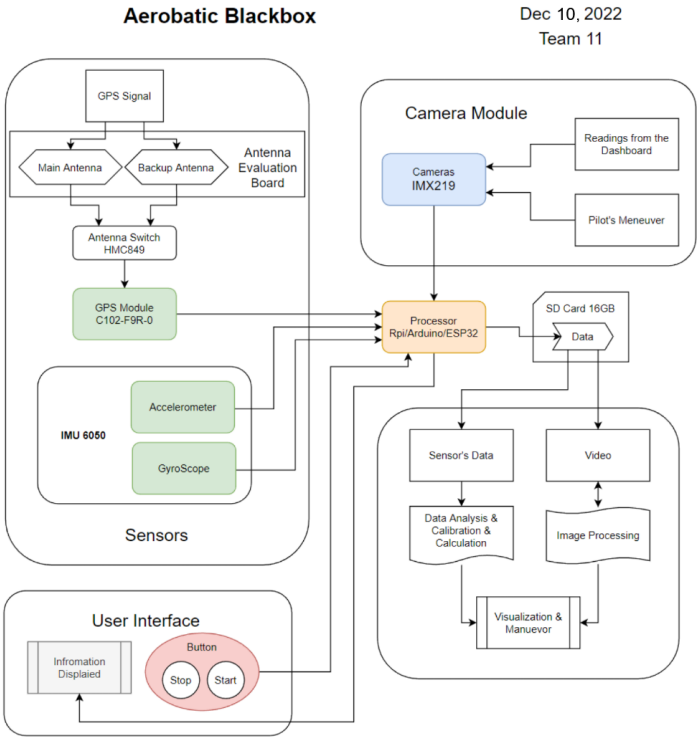


Fig. 3. System block diagram

A. System Functions

1) Data Collection and Visualization:

- Collect flight data which includes the position and orientation of the plane
- Model the plane with 6 degrees of freedom
- Input this data into X-plane which puts them into a built-in 3D visualizer

2) Pilot's Maneuver and Its relation to the Plane's Behavior:

- Collecting the maneuver and the readings from the instruments using image processing
- Display it alongside the visualization

B. *Technical Details*

1) Sensors:

a) GPS Module: We are using a GPS module to record the accurate position of the plane. We will use two antennas for the GPS and select the antenna with the strongest signal to feed to GPS. Thus, we ensure that the GPS module is always connected to the satellites even though the plane is upside down.

b) IMU: We are using the accelerometer and gyroscope in the IMU to record the accurate acceleration in 3 axes as well as pitch, roll, and yaw. We will use the acceleration to calculate the position and compare it to the result of GPS. Also, we will use the acceleration to calculate the pitch, yaw, and roll. Thus, we can calibrate both the position and orientation.

c) Cameras: We will use 8MP Raspberry Pi cameras to record everything in the cockpit. The Camera will both record the maneuver of the pilot, and the instruments in the cockpit. A stabilization system will prevent them from experiencing unnecessary shaking. We will use image processing to collect the data we need from these videos.

2) Dataflow:

The three sensors are now controlled by three main processors because of the constraints in computational capabilities and power. The GPS module and IMU draw a lot of power. Besides this, the Raspberry Pi is the only processor that we have access to and it can read and write to an SD card. However, the number of ports on Raspberry Pi is limited. Thus, we need another processor to extend the number of ports (Namely, the ESP32 has an I2C).

The ESP32 will control the IMU, Arduino will control the GPS, and the Raspberry Pi Zero will control the camera, along with the duty of writing the video and all the collected data to the SD card. There will be a UDP Server running on the Raspberry Pi, and the ESP32 and Arduino will have a UDP client running and pushing packets to the Raspberry Pi. Eventually, the Raspberry Pi will write all the data, along with the video to the SD Card.

As all the data is stored in the SD card, we will plug it into our laptop. We will input the position (GPS) and orientation (IMU) data to X-plane and visualize it. Then, we will input the video into our image processing function.

3) Image Processing

Reading the dashboard dials will enable us to record information such as the engine RPM, carb temp, amps, airspeed, altitude, vertical speed, acceleration, etc. directly from the plane's systems. On a separate computer, the video files from each camera will be read into a program that will process each frame and output a numerical result for each dial it looks at. This will correspond to each data point for the other sensors.

4) Casing

For the prototype and demo product, the case shall be made of acrylic plates to ensure visibility, because we want to present our product to an audience. For the final product, the case shall be made of black plastic with a ventilation hole design, or thin aluminum alloy plates, to ensure the heat dissipation ability. Multiple openings will exist for USB ports so cables can be connected with the chip and sensors without opening the box. There will be a shock-damping layer between the outer case and the electrical components to prevent any electrical or wiring damage caused by cockpit vibration. Sensors and chips shall be mounted on the case using screws to ensure stability.

For cameras, a case with some stabilization function to encounter the vibrations from the cockpit and engine will help. It will steady the battery pack as well.

5) Power supply

The major power supply of the black box shall come from the 12V DC cigarette lighter in the cockpit. An adaptor will be used to connect the chip to the power supply. For sensors that are not mounted inside the box, such as cameras, multiple 6V batteries can be used.

C. *GUI (Graphical User Interface)*

1) Integrated Onboard Display

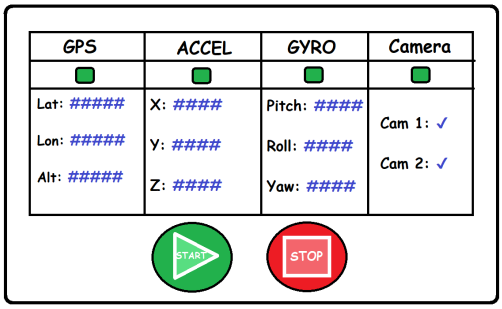


Fig 4. Example GUI on the integrated onboard display

The interface of our integrated onboard display will contain a start button and a stop button. It also displays flight information including the status of all the sensors, their readings, and an indicator showing if everything is working correctly. If some errors occur, the display should also show the errors.

2) Visualization on our Laptop

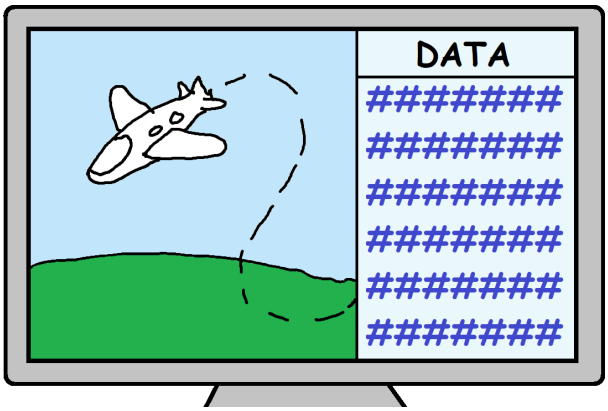


Fig 5. Example GUI on the laptop computer

The visualization on our laptop is a side-by-side display. On one side there should be a 3D visualization of our data from the X-plane program [1]. The other side will show the data values and pilot maneuvers.

4 FIRST SEMESTER PROGRESS

A. GPS

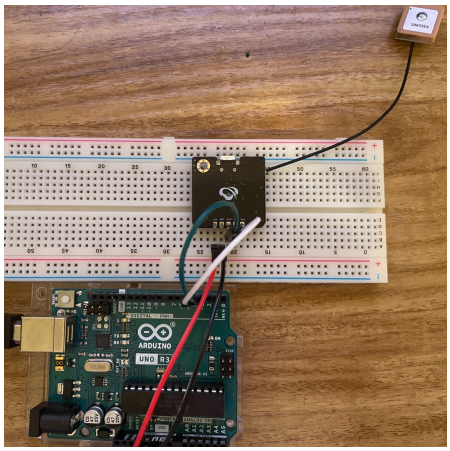


Fig 6. GPS setup (Arduino)

The GPS module is controlled by an Arduino. It is running on a C++ package called TinyGPS++. With TinyGPS++, we can use the GPS to read a position at a centimeter-level accuracy. This is our first step in modeling the plane in 6 degrees of freedom.

B. IMU

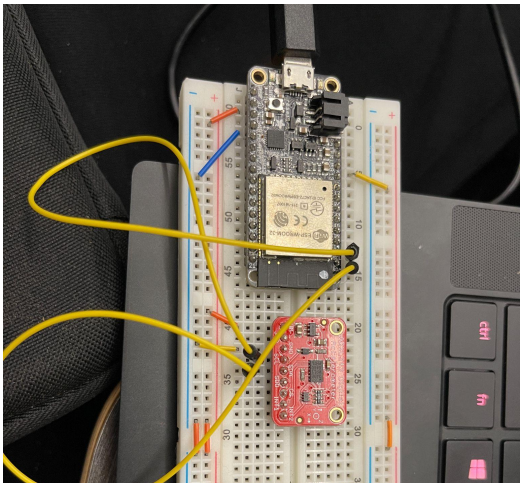


Fig 7. IMU6050 setup (ESP32)

The accelerometers and gyroscopes within the IMU 6050 operate based on registers in them. We use I2C to interface with the IMU and read or write to it.

C. Camera Setup

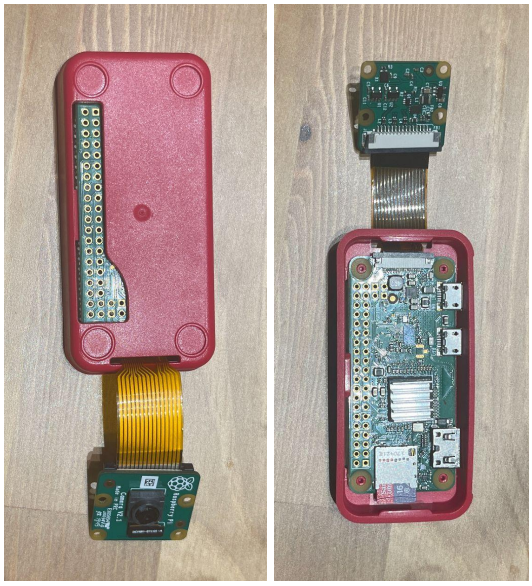


Fig 8. Camera setup (Raspberry Pi Zero)

The camera we are using is the basic 8MP R-Pi Camera, and we are going to use an npm package called *Motion* to control it. It will activate the camera as a webcam, and record everything, then write the video to the SD card. It

is an 8MP camera, and the set frame rate is 60 Hz, so it is about 100MB/s. The fastest rate that a Raspberry Pi can write to an SD card is about 50 MB/s, and the storage in an SD card is limited. Thus, we decided to turn down the frame rate to 24 Hz, so that no data is lost during the reading and writing process.

D. Image Processing

1) Downsampling: We do not yet have a video of the plane dashboard during a flight, so we used some photos taken from last year's team's GitHub along with some higher-quality photos taken in October. For now, we decided to downsample the photos first to speed up the calculation time. In order to do this the image must first be blurred to remove the high frequencies (large changes in pixel value between close pixels). This prevents aliasing when the subsequent image is then downsampled. We created the downsampling code and used MATLAB's *imgaussfilt()* function to add a Gaussian blur to the image.

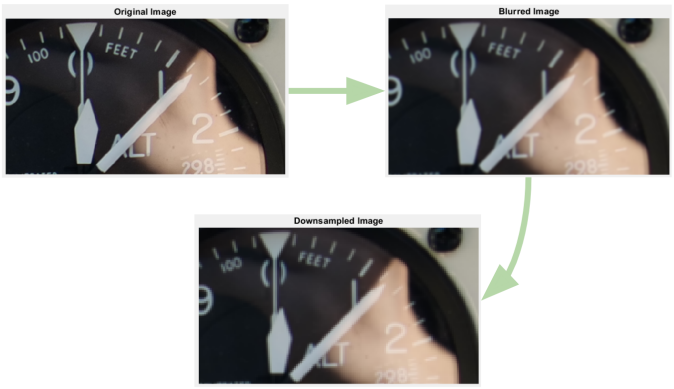


Fig. 9. Downsampling process (zoomed-in)

2) Bilateral Filtering: The paper [5] by Lauridsen et al. describes a method for using computer vision to read dials. They use bilateral filtering on the image as their first step. A bilateral filter, MATLAB's *imbilatfilt()*, smooths continuous regions of an image but preserves edges. Using this type of filter will reduce noise and make the image simpler but will keep objects' edges intact.

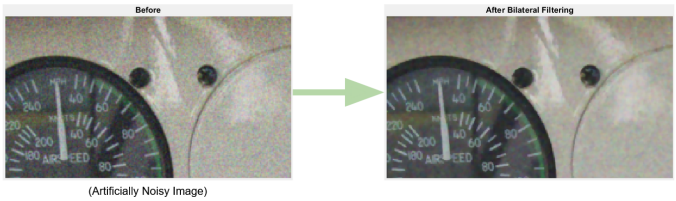


Fig. 10. Bilateral filtering process (zoomed-in)

3) Edge detection, adaptive thresholding, & circle detection: The next step is to accentuate the circles and dial pointer so that they can easily be found. There are multiple ways to do this. At first, we implemented edge detection and then later added adaptive thresholding. Edge detection works by computing a derivative approximation between neighboring pixels. If this derivative is above a certain threshold (if there is a large abrupt change in pixel values), then that area is marked as part of an edge. Different algorithms/filters do this, but we found the default method of MATLAB's edge() worked fine. The edges created by this function take the form of thin white lines which are too thin for the circle detection function to find.

To remedy this we wrote code to widen the edge lines in either a 1st or 2nd order neighborhood (user chooses) of pixels around a detected edge. For each pixel in the image: (1) detect white pixels in the neighborhood, (2) if detected, turn the currently selected pixel white as well.

Fig. 11. Pixel neighborhood visualization

We found inverting the edge-detected image so the background is white and the lines are black helps the circle detection function work better. After this, the Hough circular transform can be applied, *imfindcircles()*. A radius range is passed to this function as a parameter. We used 5% of the shortest dimension (width or height) of the image frame as the minimum radius and the full length of the shortest dimension as the maximum. This could be tweaked by reducing the minimum and maximum of the range if the dials are far away from the camera and thus small in the frame.

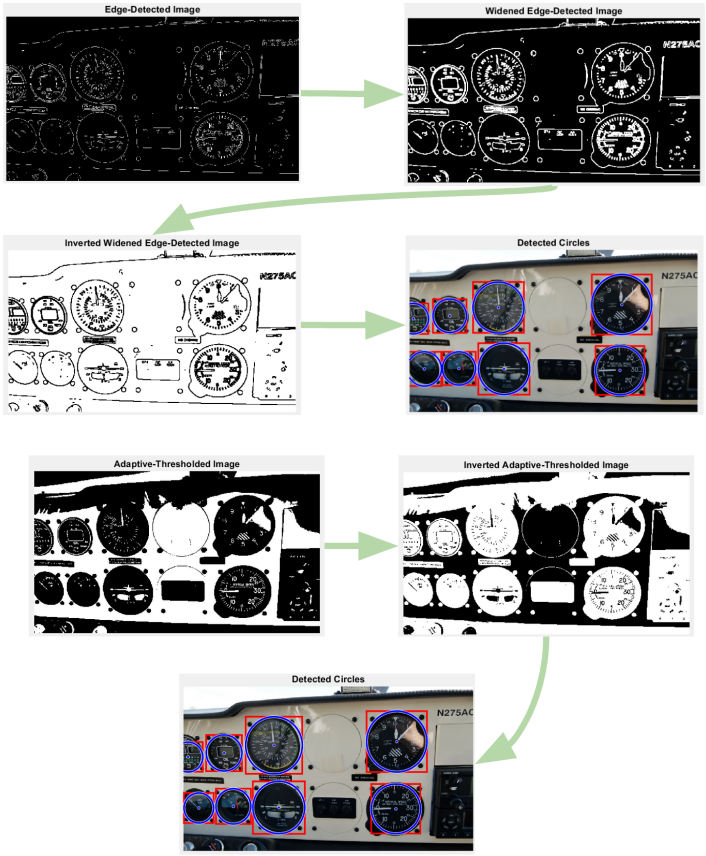


Fig. 12. (top) Edge-detection & circle detection process, (bottom) adaptive-thresholding & circle detection process

4) Pointer Detection: Lastly, we detect the dial pointers. First, each dial was cut out and put into its own separate image. The pixels outside the circle (greater than the radius length away from the circle center) were masked out and turned black.

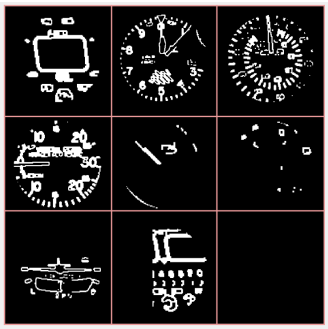


Figure 13: Cut out dials

After this, the Hough line transform [6] was used to find straight lines on each of these dial images. Only the longest lines within a length range from a fraction of the radius as a minimum to the radius length as a maximum were considered. This is because the pointers all start in the center, so they cannot be longer than the circle radius. The longest of the detected lines is chosen as the true pointer. We were able to detect dial pointers, but have not yet been able to get high-accuracy results or find the pointer angle. The current process is experimental and will probably be changed later.

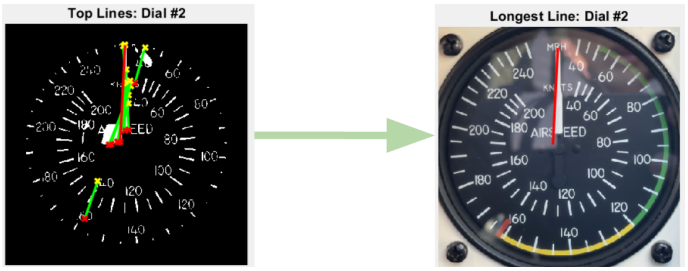


Fig. 14. Pointer detection

5 TECHNICAL PLAN

Task 1. IMU Calibration

The accelerometer and gyroscope in the IMU are very inaccurate, yet the data they are recording (acceleration and orientation) are very important to our project. The acceleration will be used to calculate the position using double integration. However, a minor error in acceleration can become a major one after double integration. Thus, we need the accelerometer to be very accurate. Also, IMU is the only sensor on our project that gives us access to the orientation of the plane. We can only read pitch, yaw, and roll from the gyroscope, and the only way to calibrate the gyroscope is by using the accelerometer.

Thus, the IMU needs to be constantly calibrated to ensure the best performance. However, calibration of the IMU in a traditional way is not very easy to perform. It includes dropping the IMU to put it into free fall and aligning the IMU precisely to the x, y, and z directions of the inner coordinates of the IMU. We will find an easier way to calibrate it. This way should be just like how we calibrate the accelerometer and gyroscope within our phone.

Task 2. GPS Signal Enhancement

As the plane goes upside down, it is very likely that the GPS to lose signal. Once the GPS gets disconnected from the satellites, the whole system will be shut down, and even if the GPS is connected to the satellites again, it will take at least 30-40 seconds for the GPS module to restart and

recalibrate. Given that a stunt flight usually only lasts a few minutes, losing 30 seconds of position data is unacceptable. Besides, as we mentioned earlier, the error in IMU can not be completely eliminated. Thus, it is unrealistic to use the data from IMU to substitute GPS for 30 seconds. The error accumulating over half a minute will be way off.

The only solution to solve the problem is to find a way to make sure that the GPS does not lose signal. After some research in the field and several phone calls to the company which built our GPS module (u-blox), we came up with a solution. We can use two antennas. However, all of the GPS modules on the market that are available to us are not compatible with 2 antennas. To fix this we decided to use a switch to connect the GPS module and the two antennas. If the chip on the switch detects that the signal on the antenna that the GPS is currently using is too weak, it will simply switch to the signals on another antenna.

The reason that this will work is simple. GPS losing signal in the air is due to the fact that during a stunt flight, the plane is very lightly to go upside down. In that case, the antenna on the top of the plane will be blocked by the body of the plane. Thus, the antenna will lose connection to the satellites and be turned off. In our solution, we put two antennas on opposite sides of the plane. So, if one antenna loses connectivity, the other one is guaranteed to be facing the satellites. This will ensure the continuity of the GPS signal. However, this solution is still all just a theory. We purchased the switch and a GPS module, and we will test out whether this solution will work or not.

Task 3. Image Processing Finalization

1) OpenCV: In the first semester, we mainly use MATLAB functions or apps to detect the dashboard dials, including circle detection functions and color thresholding. However, MATLAB is not powerful enough for the final product. We will use a more advanced language such as Python. There is an open-source package called OpenCV in the Python environment and it is able to detect selected objects in live videos.

Next semester we will explore OpenCV. We can use a mixture of our own code and existing packages to detect the dials and their pointers. Then we can manually create an angle-to-dial value map for each dial so that the program can calculate the readings based on the positions of the needles. Our program will be able to open the videos and then generate a data sheet containing the readings gained from the video corresponding to each frame.

2) CNN (Convolutional Neural Network): Darcy, a member of last year's group who works with our client, suggested that we should use CNN to train our image processing system. She said that it might help with the issue of light reflections on the dial. We could write a script to generate some ideal synthetic images of dials, maybe with some skew. Then perform edge detection on the ideal images. Theoretically, this should give perfect edges. Next, we would add some imperfections to the images, like making some random regions lighter to simulate reflections, or adding some noise. Finally, we would train a pretty basic CNN against the imperfect images and the perfect edges.

After edges are detected, we would then do circle detection / dial reading. From her experience last year, Darcy found that it was easiest to read a dial when the dial appeared circular in the image. Wherever the camera is placed in the cockpit, some of the dials will probably be viewed from an angle. To mitigate this, we could use the Hough ellipse transform, and use the equations of the ellipses to transform the image so the dials appear circular.

3) Dataset & Backup Plan: Thus far we have not been able to get a video during a flight. We found a dataset [3] that has multiple videos of dials which we could use as a replacement in the meantime.

Also, if the image processing for the pilot input movement analysis proves to be too difficult, we could simply display the unaltered video from the pilot maneuver camera to show what they were doing at that moment. This may be just as informative because the pilot will get a more comprehensible visual instead of just a “pedal depression index” or “control stick angle”.

Task 4. Data Synchronization

The data needs to be synchronized as we record it because eventually we will visualize and compare it with the pilot's maneuvers. There are two ways to do this, and both of them have their pros and cons, so we need to do some testing before we decide which one should be used in our project.

1) First Solution: Record the time when receiving the data. This will be the easiest to implement. However, due to the delay in UDP and general processing time differences, this solution will not be very accurate. Despite the possible

error, the result could still be acceptable, so we should test out this solution, and see if it meets our expectations.

2) Seconds Solution: If the first solution does not meet our expectations, we will try timestamps. A timestamp is a way to track the time of all the events happening in the system. Once data is recorded, the time stamp will be published to the system, and the data recorder will subscribe to this topic and get the timestamp of the data. We will have access to the precise time that the data reflects in the real world. This solution has a major disadvantage: it is very time-consuming. And, the processing power on Raspberry Pi is very limited. Implementing this solution is very difficult.

Task 5. Product Assembly

After all the sensors and programs have been tested and set up, we will build the actual product in two versions. The first version is for demo use, and we will make it transparent so that audiences will be able to see the internal design of the product. We will draw the design diagram and make a 3D model of the product using SolidWorks, and cut the blueprint on an acrylic plate using a laser cutter.

For the final product, we will create an aluminum alloy case from metal boards and screws. The shock-absorption layer will be made of an industrial sponge, but we will not use the entire sponge since it will prevent heat from escaping, so we shall cut multiple holes in it. If necessary, we could have a mini fan and wind channel to enhance heat dissipation. The main electrical components will be mounted on the bolts that are connected to the box. The cases for the external sensors will be made of the same material, but we will add a suspension design as a shock absorption solution to counter the vibrations in the cockpit.

Task 6. X-Plane

The first semester of our product development mainly focuses on the settings of each component, but the final goal is to generate a 3D animation. We can download any existing examples and projects that other people have made on X-Plane, and learn about the program. Once we understand it we will upload our own data to generate our own flight model. Once a new data set has been uploaded, we shall configure and calibrate the output animation and its settings. Finally, we shall select a test data set and generate a testing 3D animation.

Task 7. Testing and calibration

Once the prototype has been built, the most important task is to test the product during a real flight in the aerobatics airplane.

We shall attach the device in a position that is as close to the center of gravity of the aircraft as possible to ensure the accuracy of the GPS and IMU module without blocking the pilot's movement and view. The cameras should be mounted behind or above the pilot to catch the needed videos. All the equipment cannot be permanently installed and should be easily detachable.

6 BUDGET ESTIMATE

TABLE 1. Estimated Costs

Item	Description	Cost
1	GPS Module * 1	$311.25
2	Antenna * 2	$28.5
3	Antenna Switch * 2	$31.8
4	Antenna Adapter Cable * 15	$29.25
5	Accelerometer & Gyroscope * 1	$9.99
6	Camera * 2	$80.00
7	Flight Controller * 1	$84.99
8	Air Module	$99.00
9	Arduino Uno * 1	$28.00
Total Cost = **$702.79**		

The largest cost of our product is the cost of sensors. We plan to buy a better, more powerful GPS module that will not be affected by the direction of its antenna and is able to form a stable connection with the satellites easily. The cameras must capture the video of the dashboard dials clearly during day or night (but primarily day) and under any light condition. It also must have a built-in stabilization function to counteract the vibration from the engine and cockpit.

7 ATTACHMENTS

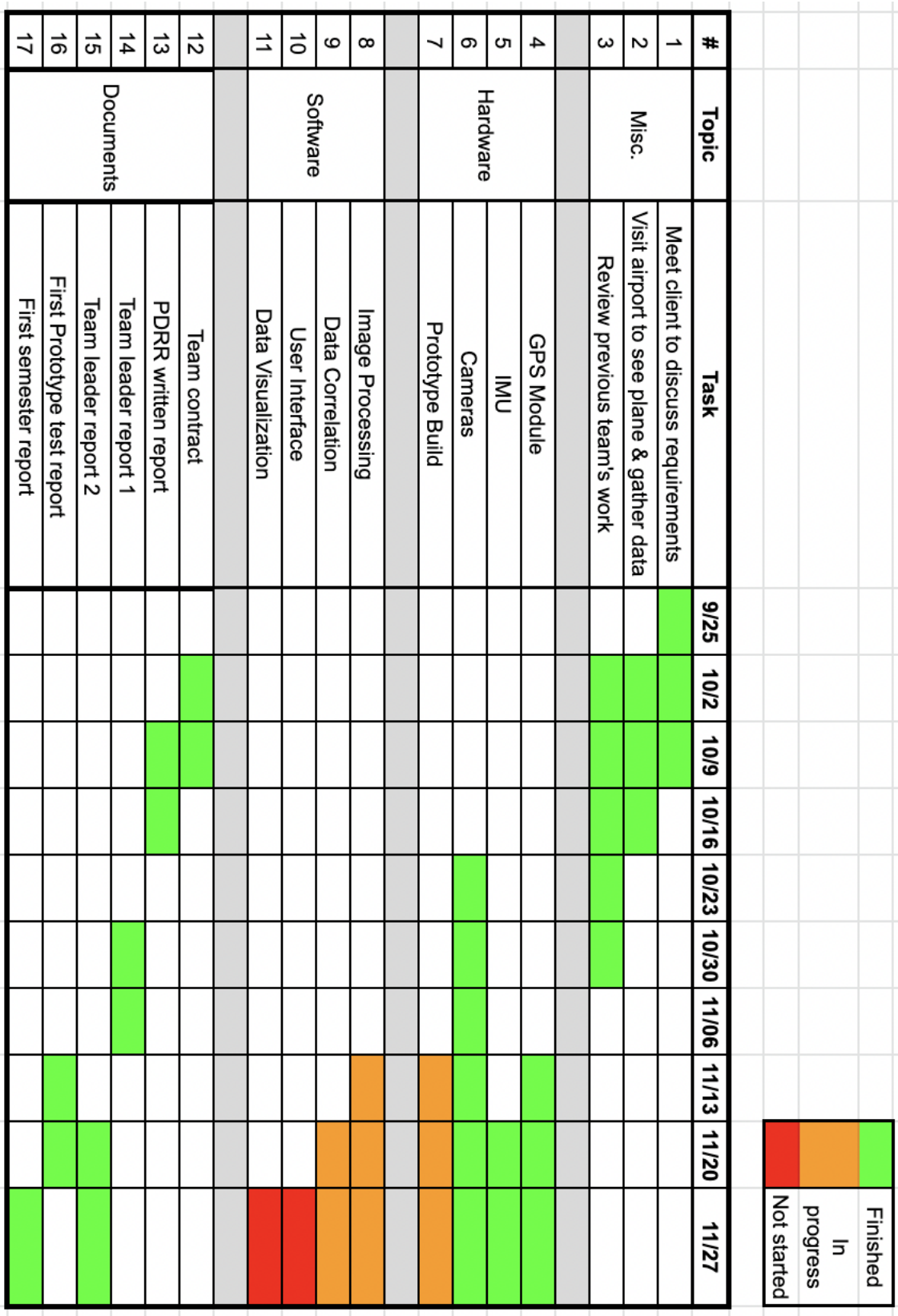
7.1 Engineering Requirements

Team #11 Project Name: Aerobatic Blackbox

TABLE 2. Engineering Requirements

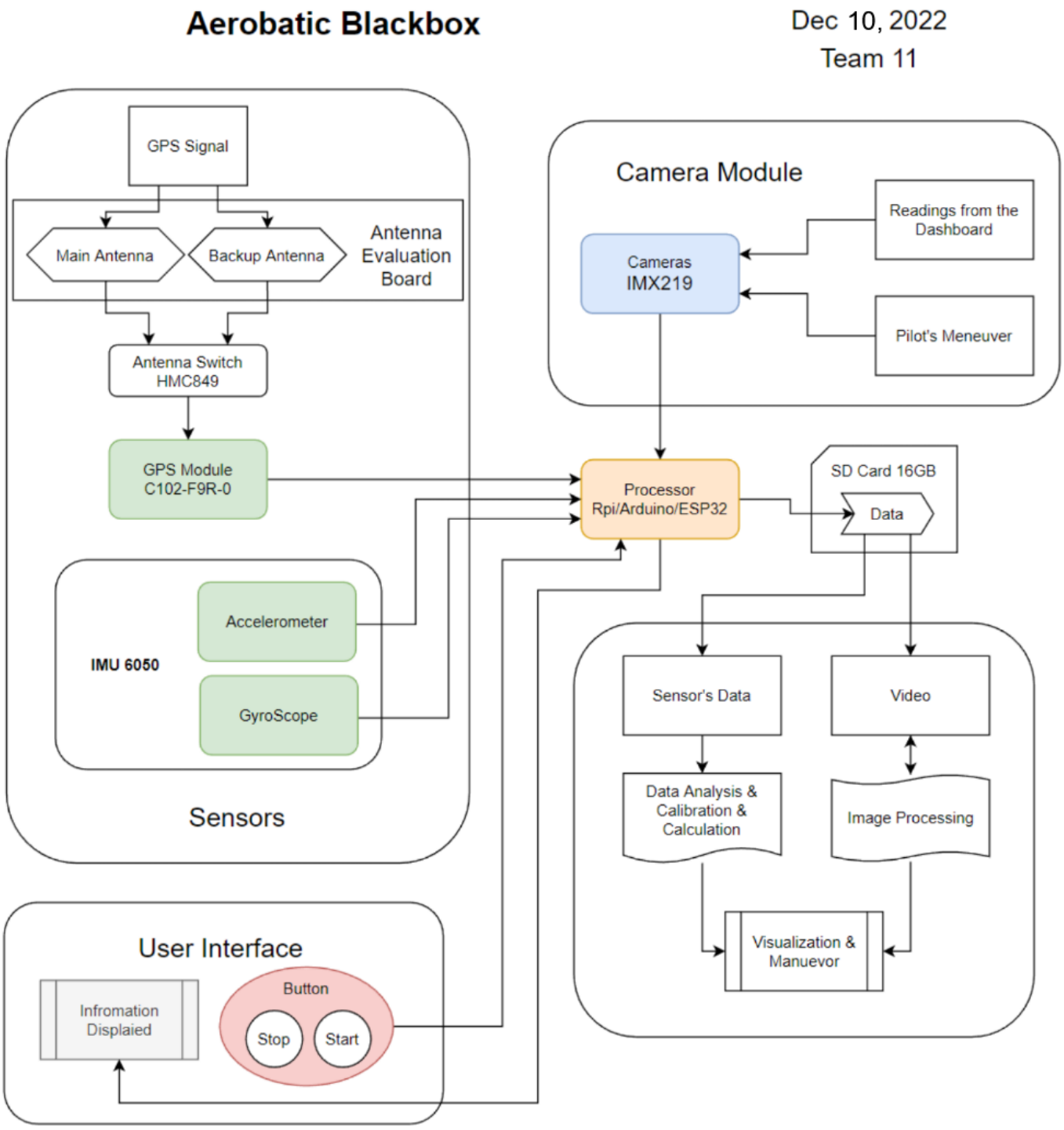
Requirement	**Value, range, tolerance, units**
Case Dimensions	15cm x 10cm x 6cm
Power	12V USB power supply, and 5V battery
Weight	~ 500g
Storage	128GB SD card
Reliability	Work in the range of -10℃ ~ 60℃
Endurance	1 year for a single 5V battery 10 years for the whole product
Ease of Use	5-minute video tutorial Start to record data once the start button is pressed Open source software – X-plane only requires users to input the data
Performance	Record variety of data, including GPS, aircraft gestures, pilot and dials readings Output a 3D animation after the data is inputted
Supportability	X-plane works on WINDOWS and MACOS The black box can be used on an aircraft without permanent modifications to it
Legal	No permanent modification to the aircraft, according to FIA regulations

7.2 Appendix 2 – Gantt Chart

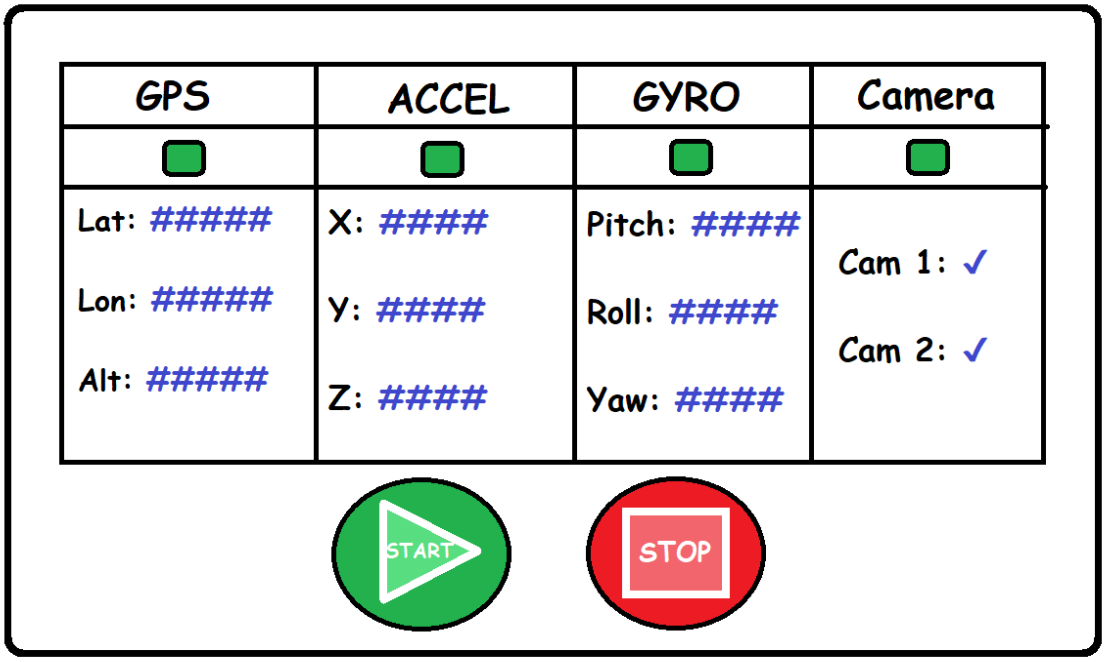


#	Topic	Task	9/25	10/2	10/9	10/16	10/23	10/30	11/06	11/13	11/20	11/27
1	Misc.	Meet client to discuss requirements										
2		Visit airport to see plane & gather data										
3		Review previous team's work										
4	Hardware	GPS Module										
5		IMU										
6		Cameras										
7		Prototype Build										
8	Software	Image Processing										
9		Data Correlation										
10		User Interface										
11		Data Visualization										
12	Documents	Team contract										
13		PDRR written report										
14		Team leader report 1										
15		Team leader report 2										
16		First Prototype test report										
17		First semester report										

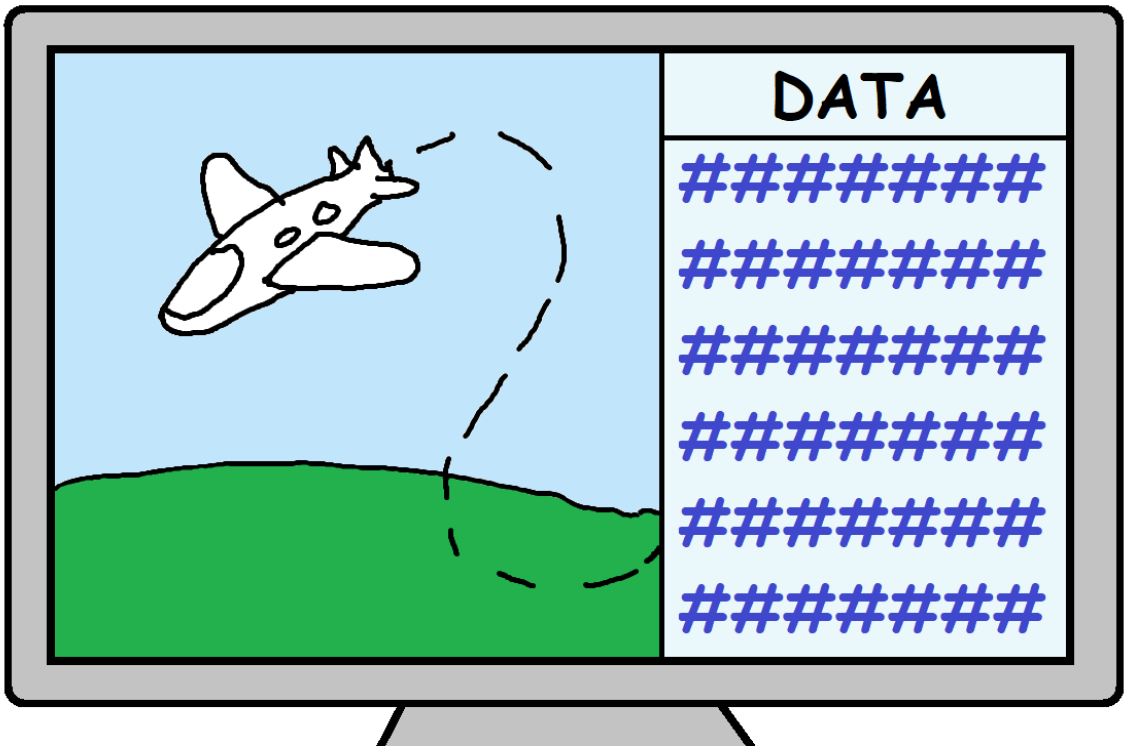
7.3 Drawings & Schematics



System Block Diagram



Example GUI on the integrated onboard display



Example GUI on the laptop computer

7.4 Team Information Sheet

Qi Luo: tomqiluo@bu.edu, 617-763-9599

Qi is a senior student majoring in computer engineering at Boston University. His interest concentrates on embedded systems and robotics. His specialty includes the two fields mentioned above, computer networking, and client and server interaction. The programming language that he is most familiar with is C, C++, JavaScript, and Typescript.

Yanbo Zhu: zhuyanbo@bu.edu,571-443-9927

Yanbo is a senior student majoring in electrical engineering at Boston University. He took mechanical engineering as his major before as well. His interest concentrates on aircraft designs, high-performance vehicle designs and driving, control systems, and robots. He is familiar with all kinds of electrical components and engineering tools.

Eli Carroll: elivc@bu.edu, 413-834-7503

Eli is a senior student majoring in electrical engineering at Boston University.

Maxwell Bakalos: maxbak@bu.edu, 978-793-3580

Max is a senior student majoring in electrical engineering at Boston University. His interests include digital image processing, signal processing, computer vision, electric vehicles, robotics, computer architecture, electromagnetics, renewable energy, etc.

References

[1] "Desktop - X-Plane." X-Plane. https://www.x-plane.com/ [Accessed: 7-Dec-2022]

[2] "Digital MEMS accelerometer data sheet ADXL343 - Adafruit Industries." [Online]. Available: https://cdn-learn.adafruit.com/assets/assets/000/070/556/original/adxl343.pdf?1549287964. [Accessed: 11-Dec-2022].

[3] J. Grassme, "Pressure Gauge Reader Data," Kaggle, 13-Dec-2018. [Online]. Available: https://www.kaggle.com/datasets/juliusgrassme/pressure-gauge-reader-data. [Accessed: 11-Dec-2022].

[4] K. Sebesta, "A-D converter the hard (but cheap!) way," Kenz Blog. [Online]. Available: http://www.eissq.com/DialADC.html. [Accessed: 11-Dec-2022].

[5] Lauridsen, J., Grassmé, J., Pedersen, M., Jensen, D., Andersen, S., & Moeslund, T. (2019). Reading Circular Analogue Gauges using Digital Image Processing. Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications. https://pdfs.semanticscholar.org/74a7/637e13cd69d8026aa15bf99ae96e914a4954.pdf https://doi.org/10.5220/0007386003730382

[6] MATLAB, "Line Detection | Student Competition: Computer Vision Training," YouTube, 27-Sep-2019. [Online]. Available: https://www.youtube.com/watch?v=zy0932i2yog. [Accessed: 11-Dec-2022].

[7] N. Singh, "Portable video streaming camera with Raspberry Pi Zero W," Hackster.io, 23-Oct-2019. [Online]. Available: https://www.hackster.io/narender-singh/portable-video-streaming-camera-with-raspberry-pi-zero-w-dc22fd. [Accessed: 11-Dec-2022].

[8] pauline95, "MPU6050 - sample rate vs output data rate," Arduino Forum, 05-Jun-2021. [Online]. Available: https://forum.arduino.cc/t/mpu6050-sample-rate-vs-output-data-rate/870912. [Accessed: 11-Dec-2022].

[9] "TinyGPS++ : arduiniana," TinyGPS++ | Arduiniana. [Online]. Available: http://arduiniana.org/libraries/tinygpsplus/. [Accessed: 11-Dec-2022].

[10] "X-Plane 12 Official Trailer." X-Plane. https://player.vimeo.com/video/771302771?autoplay=1 [Accessed: 7-Dec-2022]

Primary Section Authors

1 INTRODUCTION: **Maxwell Bakalos**

2 CONCEPT DEVELOPMENT: **Yanbo Zhu**, additional information from all group members

3 SYSTEM DESCRIPTION: **Qi Luo**, additional information from all group members

4 FIRST SEMESTER PROGRESS: (A,B,C) **Qi Luo**, (D) **Maxwell Bakalos**

5 TECHNICAL PLAN: **Yanbo Zhu**, **Qi Luo**, **Maxwell Bakalos**, **Eli Carroll**

6 BUDGET ESTIMATE: **Yanbo Zhu**, additional information from all group members

7 ATTACHMENTS: **Yanbo Zhu**, **Qi Luo**, **Maxwell Bakalos**, **Eli Carroll**