


December 2017

2Fly with RPi - Evaluating Suitability of the Raspberry Pi3 B Single Board Computer for Experimental Glass Cockpit Embedded Applications

Donald R. Morris

Southern Illinois University Carbondale, dmorris@siu.edu

Follow this and additional works at: <http://opensiuc.lib.siu.edu/jasa>

 Part of the [Aviation and Space Education Commons](#), [Aviation Safety and Security Commons](#), and the [Maintenance Technology Commons](#)

Recommended Citation

Morris, Donald R. (2017) "2Fly with RPi - Evaluating Suitability of the Raspberry Pi3 B Single Board Computer for Experimental Glass Cockpit Embedded Applications," *Journal of Applied Sciences and Arts*: Vol. 1 : Iss. 3 , Article 5.
Available at: <http://opensiuc.lib.siu.edu/jasa/vol1/iss3/5>

This Article is brought to you for free and open access by OpenSIUC. It has been accepted for inclusion in Journal of Applied Sciences and Arts by an authorized administrator of OpenSIUC. For more information, please contact opensiuc@lib.siu.edu.

2Fly with RPi - Evaluating Suitability of the Raspberry Pi3 B Single Board Computer for Experimental Glass Cockpit Embedded Applications

1. Introduction

The Raspberry Pi3 B is a small computer board about a half inch thick and a little smaller than the dimensions of a standard playing card. It is the latest installment in the Raspberry Pi line of computers that was created by the Raspberry Pi foundation. This line of computers was originally intended to teach children how to interact with computers (RPi General History, n.d.). It can be purchased for just \$35 to \$40. The combination of power, price, and simplicity have combined to make this board a favorite of hackers and developers (Brown, 2017). Around 375,000 of these little computers were sold in the first year that they were available (Miller, 2017). They have shown up in an incredible number of applications, including aircraft data acquisition. The purpose of this project is to evaluate the suitability of this board for powering an experimental low cost glass cockpit primary flight display which includes synthetic vision.



Figure 1: Standard Six Pack Instruments and Locations (FAA, 2012). Altered by Author.

2. Historic Aircraft Instrumentation

Early aviators had no cockpit instruments and had to rely entirely on the senses that were common to all humans. Since the human body was not designed for flight, these senses proved to be rather poorly suited for flight control. Cockpit instrumentation was therefore developed to aid the aviator in situational awareness – particularly when visibility was poor. The basic flight instruments – later standardized into the “6 pack,” – include the instruments shown in figure 1.

- 1) The attitude gyro (Artificial Horizon): This instrument informs the pilot of the pitch and roll angles of the aircraft. Sometimes this is paired with navigational inputs to produce a horizontal situation indicator (HSI).
- 2) The airspeed indicator (ASI): This instrument informs the pilot of the forward speed of the aircraft relative to the air – but not of speed the aircraft relative to the ground.
- 3) The altimeter: This instrument informs the pilot of the altitude above sea level – but not the height above terrain.
- 4) The vertical speed indicator (VSI): This informs a pilot of trends in altitude – but not of the actual altitude of the aircraft.
- 5) The direction gyro (DG): Paired with a compass, this instrument informs the pilot of the direction the aircraft is facing – but not the direction in which it is traveling.
- 6) The turn and slip indicator: This instrument informs the pilot of trends in direction – but not of the direction it is facing. It also informs the pilot of the coordination between the aileron and rudder input.

For a number of years, these standard so-called “flight” instruments were the basis for pilot situational awareness and formed the basis for a “well equipped” cockpit (Williamson, 1937). In many small general aviation (GA) aircraft, these instruments still form the basis for flight today.

Back in the late 50’s and early 60’s, military aircraft began implementing computer based cockpit systems. These systems integrated well with weapons technology, and the outputs were displayed on glass cathode ray tubes in the cockpit – hence the term “glass” cockpits. Migration of military technology to commercial aviation was well under way in the late 1970’s, with rapid expansion of system capabilities and implementation. In today’s commercial aircraft, the more than 100 instruments and gauges from a similar airliner of the past are integrated into only a few space saving displays (NTSB, 2010).

In the 1980’s and 1990’s, GA aircraft owners expressed interest in similar technology. Price was an obvious barrier to implementation. The author vividly recalls his shock upon learning that the primary flight display (PFD) he was installing on a business jet had a sticker price of \$320,000. While such systems

could – and were – realistically implemented on multi-million dollar jets, migration into single engine piston powered aircraft (so-called piston single) required lower cost options. Fortunately, costs have continued to fall.

Significant commercial installation in new piston singles began in 2003 with Cirrus. Cessna and Piper soon followed. By 2006, 92% of new piston singles were delivered with some form of glass cockpit (NTSB, 2010). The retrofit market was also busy. Garmin (G series), Avidyne (Entegra), and Aspen Avionics (Evolution) are the relatively large players in the piston single field. Modern certified glass cockpit retrofit packages typically cost between \$15,000 to \$30,000 depending on brand and options. Unfortunately, this is still too expensive for many small aircraft owners. Many of these small piston singles are worth under \$20,000 (AOPA, n.d.), and installing a flight display that exceeds the value of the aircraft it is to be installed in does not make a lot of economic sense.

Up until this point, we have only considered cockpit suites. FAA has recently (2016) issued a supplemental type certificate (STC) which allows the Dynon EFIS D10A or Dynon EFIS D100 to be installed in several certified piston single aircraft. This currently defines the extreme low cost floor of “glass cockpit” technology as installed in factory built aircraft (EAA, 2016). These systems cost as little as \$2600 (Dynon, n.d.), but do not include moving map GPS or synthetic vision.



Figure 2: Parts of a PFD. Garmin shown. Others Similar. (FAA, 2012). Altered by Author.

3. Components of a Glass Cockpit

As has been previously mentioned, the term “glass cockpit” evolved along with the evolution of computer based instrumentation. As such, there is not a standard definition of what does and what does not constitute a glass cockpit. The National Transportation Safety Board defined a glass cockpit aircraft as having at least a primary flight display (PFD) (NTSB, 2010). This seems to be a reasonable definition, and will be adopted in this paper. The General Aviation Manufacturers Association (GAMA) defines a PFD as “a single physical unit that displays all of the following: altitude, airspeed, airplane direction and flight attitude. Other information may be displayed on the PFD, but this additional information cannot obstruct any items required on the PFD.” (GAMA, 2005). Figure 2 shows the parts of a typical PFD, and the numbers correspond to those shown in figure 1.

4. Synthetic Vision

While the information communicated through a classic six pack or modern PFD is sufficient to allow a pilot to remain in control of his or her aircraft, it does not allow the pilot to know the aircraft’s location with respect to other objects on the surface of the earth or in the air. This is obviously useful information, and a display that can show this is said to have *synthetic vision*.

There are many forms of synthetic vision. These can include real time sensors such as Radar and Sonar that can see through the clouds. However, the simplest form of synthetic vision is a computer constructed representation of the terrain. This is generated from a database using the known location of the aircraft, and is typically displayed on the PFD as shown in figure 3. Often the color of the terrain is coded by relative altitude (Parrish, et. al., 2008). Terrain may be shaded in a green color if it is significantly below the aircraft and therefore not an impact hazard. Terrain that is near the height of the aircraft may be shaded yellow, and is an impact risk. Terrain that is above the aircraft may be shaded red, and should be avoided in flight.

While terrain representation is an important part of synthetic vision, another critical aspect is obstacle avoidance. Obstacles such as bridges and towers represent significant hazards to blind flight. The pilot is far less likely to collide with these hazards if they are mapped and shown on the PFD. Pilots also need to be aware of airport locations and airspace boundaries, so these are appropriate to add as well. Like the terrain, these must be maintained in a digital database. Unlike the terrain which is relatively stationary, it is critical that these databases be updated regularly for safe flight. This helps to ensure that the database in the aircraft matches the real world. Without this correspondence, obstacle databases are useless.

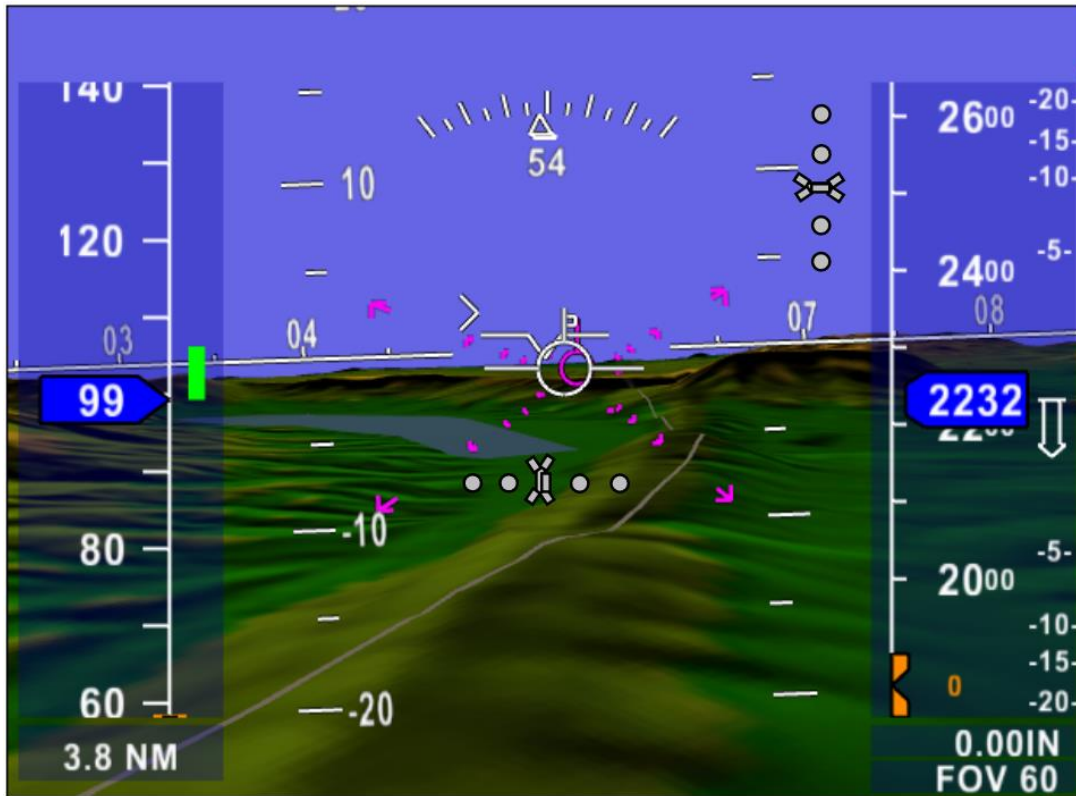


Figure 3. Synthetic Vision on a PFD display. Source: Parrish et. al., 2008

Finally, synthetic vision may also be used to prevent collision with other air traffic. While a few aircraft have Radar capable of sensing other traffic, this is no longer the only way of knowing where other aircraft are located. With onboard global positioning system (GPS) receivers, modern aircraft know their own position and flight path. If this information is shared, it can be used by any other aircraft in the vicinity to alert its flight crew to any impending conflicts. A relatively new communication standard known as automatic dependent surveillance – Broadcast (ADS-B) has been developed precisely to share this information. It is currently in its implementation phase. At a minimum, ADS-B requires an aircraft to broadcast its position and flight path (ADS-B out). For full use of the ADS-B system, the aircraft should also be able to receive data (ADS-B in), including the position and flight paths of other aircraft in the area. Additional data functionality is also included with ADS-B in, including satellite based weather (AOPA, 2015). ADS-B as a system is still in its early implementation phase, but is expected to provide significant safety advantages in flight.

5. Requirements for the Experimental PFD

Compared to most standalone computers of today, the Raspberry Pi 3B is clearly lacking in power. Before its suitability to power a PFD with synthetic vision can be evaluated, the minimum acceptable feature set of the PFD must be determined. These features presented in this section have been chosen from practical experience as well as available data and best practices.

5.1 Sensors

In addition to accuracy and sensitivity, the sensors used for early conceptual implementation must be affordable, commonly available, have low power requirements, operate at a 3.3 volt logic level, and be capable of communicating with the central embedded computer via serial communication. The preferred serial protocol is I2C. This is a two-wire serial interface for communications between electronic components, and is a common communications standard known for its robustness and error tolerance.

The sensors selected for this phase of the project consist of the following:

1. Air data (Pitot and Static Pressure). The two separate air pressures measurements must be provided by two separate chips. The Bosch BMP280 chip meets all of the requirements above. It has a basic pressure sensitivity of 0.16 Pa, with a root mean square noise of 1.3 Pa. As an added bonus, this chip can report temperature to the nearest 0.01 °C (Bosch Sensortec, 2015). The noise floor of this chip allows for a practical altitude resolution of about 3 feet.
2. Attitude Sensation (3 Axis Accelerometer and 3 Axis Piezo Gyroscope). These six sensors are contained within a single Invensense MPU6050 chip. The gyroscope portion of the chip features 16 bit readouts with user selectable full scale deflection ranges of ± 250 , ± 500 , ± 1000 , or ± 2000 degrees per second. Root mean square noise levels are scaled by the full scale deflection rates, and are as low as 0.05 degrees in the most sensitive settings. The accelerometer portion of the chip also features 16 bit output, but full scale deflection is user selectable to ± 2 , ± 4 , ± 8 , or ± 16 g. Noise performance of the accelerometer is not reported in the data sheet (Invensense, 2013).
3. Magnetic Direction Sensation (3 Axis Magnetometer). The Honeywell HMC5883L chip allows sensation of the earth's magnetic field. It can determine magnetic heading with an accuracy of 1 to 2 degrees (Honeywell, 2013). This is the only sensor selected that does not exceed the capabilities of classic analog gauges.
4. GPS input. The NEO-7G GNSS sensor is the only chip selected that cannot interface with I2C. Luckily, it can interface with USB, and so can

still be used. This chip features a horizontal positioning accuracy of 2.5 meters, a velocity accuracy of 0.1 m/s, and an instantaneous heading accuracy of a half of a degree (U-blox, 2014).

5.2 Functionality

For the initial phase of the project, the PFD must have the six functions previously discussed and shown in figure 2. In addition, the PFD must do the following:

1. Be capable of terrain generation. Terrain generation will be based on US geological survey data sets (Terrain Data, 2017). The initial project does not need to contain a terrain sample of the entire world or even the entire US, but it must contain at least a 200 by 200 mile area of terrain. The terrain generation must be able to show at least the familiar height based green/yellow/red altitude zones for terrain avoidance.
2. Be capable of displaying tower obstacles. These will be based on FAA/FCC databases (DOF, 2017), and will be displayed by generic symbols rendered on the PFD. Only the obstacles within the sample area need to be included in the PFD database.
3. Be capable of displaying airport information. The airport information will be based on FAA databases (Airport Diagrams, 2017). Only the airports within the sample area need to be included in the PFD database.
4. Be capable of displaying moving map information. The image source for the maps will be the VFR sectional prepared by the FAA (VFR Raster Charts, 2017). The map must be scalable and rotatable, so that it can be displayed in a “track up” configuration. Only the map information that corresponds to the terrain sample area needs to be included in the PFD database.

While there are many other features that are needed for a practical, flyable result, the above features will be considered sufficient to demonstrate proof of concept.

5.3 Terrain Rendering

Terrain rendering places very high demands on any processor, and efficient routines to render terrain are the subject of a great deal of research. There are several strategies used to create terrain, but almost all of them involve loading the data from storage memory into the processor’s working memory in manageable sized chunks. When the user strays out of the area covered in memory, new data must be loaded for a new area. This is not unlike the way that aviation charts have always functioned. The FAA has divided the United States into a number of sections. Sectional charts are available, as shown in figure 4.

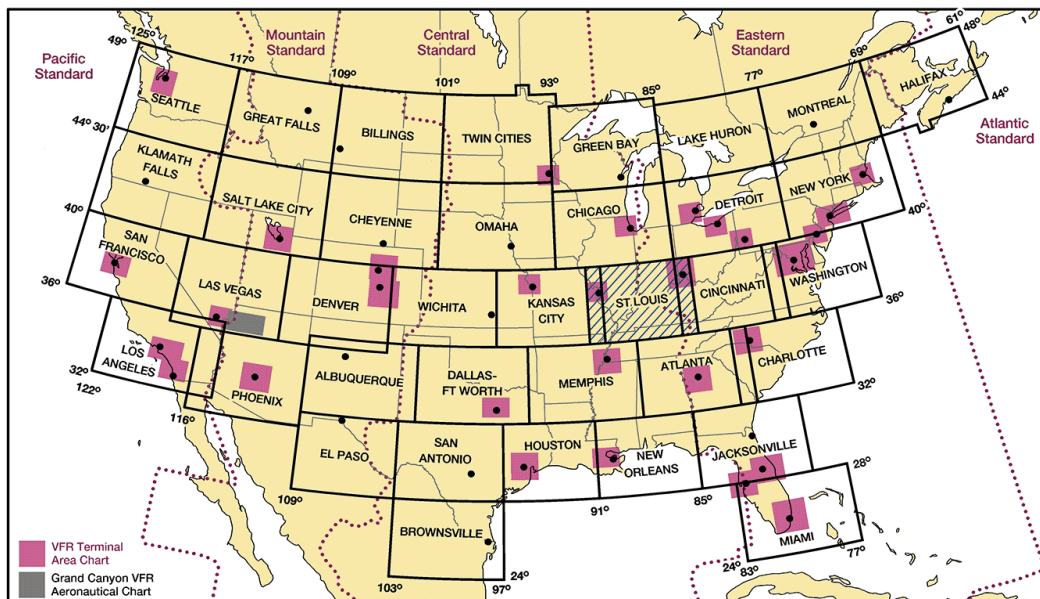


Figure 4: FAA Sectional Index. (VFR Raster Charts, 2017)

These sectionals cover areas of approximately 300 by 240 miles. It would be extremely convenient if the sectional divisions already determined and mapped by the FAA could correspond to the chunks of data entered into the processor's memory.

5.4 Moving Map Considerations:

A GPS moving map display is a two dimensional (2D) task. In its simplest form, a 2D map must be loaded into memory. The GPS sensor is queried to get location and track. The map is then scaled, rotated, and panned in order to correspond to the necessary display. It is then displayed on the screen, and a symbol of an aircraft to represent the current position relative to the map is drawn in the appropriate location.

The preparation of a map is certainly the most difficult part of this process. Luckily, this has already been done. The FAA offers digital sectional raster downloads that are prepared at 300 DPI on a 1:500,000 scale (VFR Raster Charts, 2017). This means that 300 pixels corresponds to 500,000 inches, or 41,667 feet. Reducing this, we find a single pixel on the map corresponds to approximately 139 feet – about the width of a runway. A typical sectional image occupies around 225 MB when uncompressed. Using modern single board computers, it is possible to work with this quantity of memory while executing a program – but it is not practical. Luckily, it is possible to use a technique known as “blitting” to copy only a portion of the dataset into working memory (Christophe, 2012). This makes it possible to free up resources for other tasks.

5.5 Terrain Rendering Considerations:

Successful terrain rendering is based on a series of compromises. If too much terrain data is used, the processor will bog down during rendering and will not be able to complete the job in a timely fashion. If too little terrain data is included, the resulting image will be misleading and dangerous. It is therefore important to set reasonable goals for terrain rendering.

The most important determination involves the maximum number of triangles that the system will be required to render. Each triangle is an approximation for one section of terrain as defined by three points. If the data points for the terrain map are kept close together, a great many triangles will be needed. If, on the other hand, the triangles are too far apart, important and potentially life-saving details of the terrain may not be visible on the PFD. In this demonstration phase of the project, a terrain map resolution of 6 data points per mile has been selected.

The next important compromise to be considered is the distance away from the aircraft that the terrain will be rendered. Too small of a distance is impractical. There is not enough warning of impending collision for the pilot to take action. Too large of a distance is also impractical, as the number of polygons to render increases rapidly with distance. In this demonstration phase of the project, 30 miles of maximum display has been selected.

The next compromise involves the resolution of altitude recorded. Private pilots must demonstrate the ability to control an airplane to ± 100 feet. The altitude chips that have been identified for this project can report altitude to approximately ± 2 feet. There is no reason to keep track of partial feet (inches) of elevation. This allows the use of more rapid integer math for the processor. In this demonstration phase of the project, a vertical resolution of 1 foot has been selected.

5.6 Processing Requirements

Now that there are some target numbers, initial calculations are possible. The data type of integer requires two bytes of memory, and can represent numbers between -32,768 to 32,767. If the numbers are taken to be feet, that is lower than the lowest spot and higher than the highest spot on earth. Each data point in memory will require a total of two bytes. A nautical mile is equal to 6076 feet. If we divide the nautical mile into six segments, we find that our data points will be 1012 feet and 8 inches apart. A six by six grid of points requires 36 data points per nm^2 , which requires 72 bytes per nm^2 . At this rate of data usage, a 300 by 240 nm stretch of terrain would require about 5,184,000 bytes, or about 5.2 Mb of data. This is easily within the capabilities of most single board computers.

Our defined six by six data points per nm^2 also allows us to begin approximating the number of polygons that must potentially be rendered. There

are twice as many triangles in a grid as there are rectangles, so a six by six grid will require 72 separate polygons be rendered. If a distance of 30 miles in front of the aircraft and 15 miles to each side of the aircraft must be rendered, this translates to a grid of 30 by 30, or 900 nm². This means that the computer must potentially render 64,800 triangles per update.

Rendering 64,800 triangles is a big job. This is unfortunate, because the rendering must occur quickly. Video gaming experience has dictated that a frame rate of around 60 frames per second is ideal for human perception. Faster refresh rates are not perceived as advantageous to the typical user. Frame rates between 60 and 30 frames per second result in a small but noticeable drop in quality. Frame rates from 30 to about 15 frames per second drop rapidly in quality. Depending upon the user, frame rates of below 15 range from annoying to unusable. For the demonstration phase of this project, a frame rate of 15 has been selected as the minimum acceptable frame rate. Multiplying this minimum acceptable refresh rate by the potential 64,800 polygons per frame yields a staggering 972,000 triangles per second that must potentially be rendered and shaded.

Processing all of those triangles is a very heavy processing task (Strugar, 2010), but it is one that is very commonly asked of computers. In order to accomplish this, most computers employ a separate processor known as a graphics processing unit (GPU). The GPU specializes in rapidly drawing shapes such as triangles. A computer with a GPU has the ability to use hardware acceleration on graphics as long as the software to support the GPU is available.

5.7 Geospatial Considerations:

Up until now, any consideration of terrain data has been based on a two dimensional Cartesian coordinate grid. The surface of the earth does not lend itself well to being represented as a two dimensional grid. Various systems of projection are used to project the curved surface of the earth in a flat two dimensional space, but this always produces some distortion. The larger the size of the surface, the greater the error associated with this form of projection.

The terrain datasets that have been selected for this project are not based on an X Y Cartesian coordinate system. Instead, they are based on Latitude and Longitude coordinates (Terrain Data, 2017). These coordinates are based on angles, and not on fixed distances. Latitude is expressed in degrees away from the equator. These degrees are measured relative to the center of the earth. Equal latitude lines cross the earth parallel to each other. This means that over the surface of the earth, degrees of latitude correspond to relatively constant distances. Longitude is an angle east or west of a reference meridian. Equal longitude lines meet at the poles of the earth. This means that any given longitude angle has a maximum distance equivalent at the equator and correspond to

progressively less and less distance as latitude increases. In other words, latitude angles do not correspond to a fixed distance across the surface of the earth.

The fact that latitude angles cannot be approximated by a constant value means that the processor is going to have to do even more calculations. Fortunately, the change in scale occurs at a slow enough rate that for all regions not near the poles, the graphics representing the terrain that is generated will appear to be very normal with only minor adjustments to the spacing of the direction representing latitude on the grid. This will have the effect of creating a flat projection of the earth's surface centered on the aircraft. These ratio adjustments can be adjusted at the time of terrain loading, and will not prove a significant factor in the speed of the program. While the earth does not technically conform to a flat projection, the amount of error is so small that it will be imperceptible to the human eye. In addition, the projection error will fade from existence as the terrain approaches the aircraft.

6. Evaluating the Raspberry Pi 3B's Potential to Power the Experimental PFD

As already mentioned, the Raspberry Pi 3B can be purchased for \$35 to \$40. Basic specs of the single-board computer include a Broadcom Quad Core 64 bit CPU running at 1.2 GHz, a VideoCore IV GPU running at 400 Hz, and 1 GB of system shared RAM.



Figure 5: Raspberry Pi 3B with Playing Cards. Image source: Author

Beginning with the 64 GB of maximum data storage available, it is possible to determine if the required databases can fit on the Pi 3B. Deliberately overestimating the average sectional map file from the FAA to be 250 MB, the 40 or of these files that would be needed to store the entire US surface into memory

will come to only 10 GB. This is the largest use of space on the card, and its only use is to support a moving map GPS system. The next largest chunk of data is needed for the terrain database. Each sectional was able to fit into a terrain file of under 6 MB. This means that 40 such chunks of data would use less than a quarter of a GB. The Raspbian operating system requires about 5 GB on the card. Airport and obstacle data is of unknown size, but small in comparison to the other databases. The maximum sized 64 GB card is not needed. The 32 GB card is sufficient to handle the Raspbian operating system, the program required to run as a PFD, and the data files necessary to cover the entire US or other similarly sized operational area. The Raspberry Pi 3B is therefore capable of storing the data required for its potential role.

In terms of working memory (RAM), the Raspberry Pi 3B has 1 GB. This is shared RAM between the CPU and the GPU. By default, the GPU is allocated 64 MB. This is user selectable, and largely corresponds with the number of frame buffers the GPU can render. Higher intensity graphics may require more GPU memory. High screen resolutions also demand higher amounts of GPU memory. Higher resolutions take a longer time to render triangles. The target resolution for a proof of concept display is low – around 800 by 600 pixels. Past user experiences seem to indicate that allocating 128 MB to the GPU is appropriate. Around 375 MB of ram should be ample to run the OS and program (Memory Split, 2013). This leaves a half of a GB for storage of maps and terrain files – again within the specs of the Raspberry Pi 3B.

The Raspberry Pi 3B supports serial communications, including UART, SPI, I2C, and USB. Connecting the Pi to any of the peripheral chips previously selected for this phase of the project should not prove to be difficult.

The Raspberry Pi 3B is most likely to have trouble driving an experimental PFD in the area of processing power. Whether the CPU and GPU will be able to handle their tasks in a timely fashion has much to do with the skill with which it is programmed. Many users have reported acceptable performance while completing processor hungry tasks. Others report less performance. The primary difference appears to be whether or not the program was optimized to run on a low power CPU, and whether or not the proper drivers for OpenGL Hardware Acceleration are utilized. Properly configured, even slower models of the Raspberry Pi have been documented while rendering 1,200,000 triangles per second (Nadalutti, 2016). Based on this benchmark, the Raspberry Pi 3B is able to handle its task *if* it configured and programmed properly.

The final question of suitability involves robustness. Is the Raspberry Pi 3B robust enough to handle an aircraft environment? Since their introduction in 2012, Raspberry Pi's have been used in robotics, drones, automotive dashboards, automotive engine control, CNC machines, breweries, and even a “cyber rhino” (Hackaday.io, n.d.). Creators and users of these and many other projects report

varying levels of success. One known issue is cooling. The Raspberry Pi 3B stops functioning when its CPU reaches 93 °C. It will reach this temperature if asked to perform near maximum capacity for more than a few minutes at a time (JeGX, 2016). Most users choose to add heat sinks to the processors to allow them to run cooler. In a critical role such as in an aircraft, a dedicated cooling air stream would probably be wise. Since it is normal to provide cooling air to aircraft radios, this is not likely to be problematic in a PFD application. In the end, the robustness of a Pi based system will have to be determined by long periods of experimentation. However, there is nothing to suggest that the Pi cannot handle the stress, and a large amount of anecdotal evidence to suggest that it can.

7. Evaluating Software Strategies

7.1 Choice of Programming Language

A number of different software packages are available to program the Raspberry Pi. Most Pi programming is done using Python scripting language. Since the performance of Python based solutions is typically not as fast as possible (Wilkinson, 2012), this language was not seen as practicable in this application. Two other languages were analyzed for suitability. These included Pascal as implemented in Lazarus (Lazarus, n.d.), and C++ as implemented by Microsoft Visual Studio (Sonnino, 2017). Both of these solutions are capable of performing at or near the maximum capabilities of the Raspberry Pi. After a great deal of experimentation, C++ (Visual Studio) was chosen as the programming environment – primarily due to the increased amount of code samples and tutorials available to help with programming.

7.2 Generating the Terrain

As has already been noted, the highest processor load for the demonstration PFD will come from rendering terrain. The previous calculations in this paper were based on simple “brute force” terrain rendering (Strugar, 2010). Fortunately, there are several well researched methods that optimize terrain rendering in order to decrease potential load with little to no visual loss. Most of these routines involve varying techniques of decreasing polygon count as the terrain moves away from the viewer.

The so-called stateless, one-pass adaptive refinement (SOAR) algorithm was developed in the early 2000’s (Lindstrom and Pascucci, 2001). This algorithm was designed to optimize both memory and computational efficiency. Several sample implementations of this algorithm are available online. One of the important aspects of this algorithm is easy culling of any underlying polygons,

which allows minimizing impact on the CPU. This algorithm is the preferred method for terrain generation with the Raspberry Pi drivers in their current state. The continuous distance-dependent level of detail (CDLOD) is another interesting algorithm for terrain generation (Struger, 2010). Where SOAR emphasizes placing large chunks of terrain data into memory, CDLOD emphasizes paralleling the GPU and CPU pipes in order to minimize bottlenecks. In the world of computer graphics, the nine years of time between Lindstrom and Pascucci's SOAR algorithm and Struger's CDLOD algorithm are very significant. However, current implementation of the GPU drivers on the Raspberry Pi do not appear to handle any OpenGL calls greater than 2.1. No evidence of successful CDLOD implementation using this restriction has been found. However, Marek Mauder has demonstrated successful implementation of SOAR on low powered Android based devices (2013). However, should more advanced drivers be developed for the Raspberry Pi (supporting at least OpenGL 3.0), this method would become the preferred method.

8. Conclusion

After investigating the performance required to implement a functional PFD and investigating the specifications and performance of the Raspberry Pi 3B, it is apparent that this single board computer could reasonably form the computing basis for an experimental PFD based solution as long as the following conditions are met.

1. The program executed on the Raspberry Pi must be written in such a way as to optimize the power of the Raspberry Pi including both the CPU and the GPU.
2. The Raspberry Pi must be proven to possess enough ruggedness to stand up to the conditions encountered in a typical single engine piston aircraft.

Of these two conditions, the second is the most terrifying. This is because any experimental solution that is implemented needs to be employed in such a way as to ensure that any problems that are discovered do not result in harm to the person who discovers them. This leads us to a very important topic – safety.

9. Safety

Simply put, there are never any guarantees that mechanical devices will not fail. If these mechanical devices are critical safety components such as cockpit flight instrumentation, a failure can reasonably be expected to cause death and destruction. Should any experimental glass cockpit solution ever be built, one of the most important aspects of its safe deployment is testing. The other aspect is redundancy.

9.1 Testing

Testing of any experimental PFD solution should begin first on the ground in a laboratory environment where there is little to no possibility of catastrophic consequences to failure. When laboratory testing indicates a reasonable level of reliability, the next phase of testing should be in vehicles. The PFD could be built into the passenger dashboard of a car. This would allow large numbers of running hours to accumulate without high risks or high costs. Another option is for the PFD to be set up as an independent device that could be carried in existing aircraft. A passenger on the aircraft could monitor the PFD results and crosscheck against the existing safe instruments of the aircraft. Any errors could then be identified and tracked down. In the long run, however, the time is going to come when the experimental system would need to be installed in an experimental aircraft, and a flight testing program would need to be initiated.

Maintaining safety in a flight testing program would continue to be very important. Safety at this phase would be largely dependent on the backup instrumentation of the aircraft and the type of flights it was operated on. System redundancy would also be a big factor. Even highly tested professional solutions such as the Avidyne Entegra and the Garmin G-1000 require back up instrumentation. Typically, an airspeed indicator, altimeter, and artificial horizon are provided, along with a compass. These are considered sufficient to operate an aircraft in instrument meteorological conditions (IMC) in the event of a PFD partial or complete failure. For an experimental PFD, these instruments would be ideal. However, there is no need for the artificial horizon if the aircraft is not operated in IMC.

The danger of operating any aircraft with limited cockpit instrumentation is directly related to the type of flight conditions in which the aircraft is flying. As such, one of the most important factors to safe test flying is limiting the conditions under which test flights can be made. Avoiding IMC has already been discussed. Avoiding night flight is another significant factor in risk reduction.

9.2 Automatic Battery Backup

Another area of important redundancy lies in the power supply to the system. Since modern electronics use so little power, it is not difficult to incorporate battery backups into modern avionics. Experimental systems need not be any different. These backup batteries can be actively charged while the aircraft is run under normal conditions. This would ensure that they are fresh and capable in the event of an electrical system failure. Avionics can be made to automatically switch over to backup power for operation if the units are switched on not by the conventional positive switching, but by grounding the negative line. Since a grounded negative line will remain grounded regardless of whether or not the power supply has been interrupted, this grounding of the negative line is a

desirable mechanism for switching on a modern electronic component that has a self-contained battery backup. Once the component is on, determination of whether or not power is available on the main power line can be the driving factor between operating in battery backup or in regular power modes.

9.3 Sunlight Viewability

The best computer system in the world is useless if there is no output. For a glass cockpit, the primary output is the screen. Unfortunately, most computer screens are not readable in sunlight. This type of display is not suitable for aircraft usage. This is not a place to cut costs.

10. Future Work

Up until now, research has concentrated on evaluating the required features of a synthetic vision capable experimental PFD, and whether the Raspberry Pi 3B single board computer could reasonably be expected to power such a PFD. Having concluded that it can, work must now shift towards implementing the system described in this paper. This work is already in progress, and it is hoped that a basic PFD (without synthetic vision or GPS mapping) will be operational by the end of 2017.

References

Aircraft Owners and Pilots Association. (2015). *Air Traffic Services Brief -- Automatic Dependent Surveillance-Broadcast (ADS-B)*. Retrieved from <https://www.aopa.org/advocacy/advocacy-briefs/air-traffic-services-brief-automatic-dependent-surveillance-broadcast-ads-b>

Aircraft Owners and Pilots Association. (n.d.). Data from *VREF Aircraft Valuation*. Retrieved from <https://www.aopa.org/go-fly/aircraft-and-ownership/buying-an-aircraft/vref-aircraft-valuation>

Airport Diagrams. (2017). In *Federal Aviation Administration Aeronautical Information Services*. Retrieved from https://www.faa.gov/air_traffic/flight_info/aeronav/digital_products/dtpp/

Bosch Sensortec. (2015). *BMP280 Digital Pressure Sensor Data Sheet*. Retrieved from <https://cdn-shop.adafruit.com/datasheets/BST-BMP280-DS001-11.pdf>

- Brown, Eric. (2017). *Ringin in 2017 with 90 Hacker-Friendly Single Board Computers*. Retrieved from <http://linuxgizmos.com/ringin-in-2017-with-90-hacker-friendly-single-board-computers/>
- Christophe, Bessis. (2012). *Gamedev Glossary: What Is "Blitting"?* Retrieved from <https://gamedevelopment.tutsplus.com/articles/gamedev-glossary-what-is-blitting--gamedev-2247>
- Digital Obstacle File (DOF). (2017). In *Federal Aviation Administration Aeronautical Information Services*. Retrieved from https://www.faa.gov/air_traffic/flight_info/aeronav/digital_products/dof/
- Dynon. (n.d.). *EAA EFIS Kits w/ Required Items*. Retrieved from <http://dynonstore.com/#!/EAA-EFIS-Kits-w-Required-Items/c/20465353/offset=0&sort=normal>
- Experimental Aviation Association. (2016). *EAA STC Dynon EFIS D10A/D100*. Retrieved from <https://www.eaa.org/en/ea/aviation-communities-and-interests/pilot-resources/eaas-low-cost-stc-program/ea-stc-dynon-efis-d10a-d100>
- Federal Aviation Administration. (2012). *FAA-H-8083-15B Instrument Flying Handbook*. Washington, DC. Retrieved from <http://www.faa.gov>
- General Aviation Manufacturers Association. (2005). *GAMA Publication #12 - Recommended Practices and Guidelines for an Integrated Cockpit / Flightdeck in a 14 CFR Part 23 (Or Equivalent) Airplane. Version 2.0*. Washington DC. Retrieved from <http://www.gama.aero>
- Garmin (2014) *GARMIN Celebrates the Tenth Anniversary of G1000*. Retrieved from <http://garmin.blogs.com/pr/2014/06/garmin-celebrates-the-tenth-anniversary-of-g1000.html#.WaXHSemQxPY>
- Hackaday.io (n.d.). *995 Projects tagged with "raspberry pi."* Retrieved from <https://hackaday.io/projects?tag=raspberry%20pi&page=1>
- Honeywell. (2013). *3-Axis Digital Compass ICHMC5883L*. Retrieved from https://cdn-shop.adafruit.com/datasheets/HMC5883L_3-Axis_Digital_Compass_IC.pdf

- Invensense. (2013). *MPU-6000 and MPU-6050 Product Specification Revision 3.4*. Retrieved from <https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf>
- JeGX. (2016). *(Tested) Raspberry Pi 3 vs Raspberry Pi 2: CPU and GPU Benchmarks (+ Burn-in Test)*. Retrieved from <http://www.geeks3d.com/20160322/tested-raspberry-pi-3-vs-raspberry-pi-2-cpu-and-gpu-benchmarks-burn-in-test/>
- Lazarus. (n.d.). *Lazarus*. Retrieved from <https://www.lazarus-ide.org/>
- Lindstrom, P. and Pascucci, V. (2001). *Visualization of Large Terrains Made Easy*. Retrieved from <http://www.pascucci.org/pdf-papers/vis2001.pdf>
- Mauder, Mareck. (2013). *Android Terrain Rendering: Vertex Texture Fetch, Part 1*. Galfar's Lair. Retrieved from <http://galfar.vevb.net/wp/tag/terrain-rendering/>
- Memory Split? (2013) Raspberry Pi Forums. Retrieved from <https://www.raspberrypi.org/forums/viewtopic.php?f=91&t=58245>
- Miller, Paul. (2017). *Raspberry Pi sold over 12.5 million boards in five years*. The Verge. Retrieved from <https://www.theverge.com/circuitbreaker/2017/3/17/14962170/raspberry-pi-sales-12-5-million-five-years-beats-commodore-64>
- Nadalutti, Johann. (2016). *Raspberry 2 Running Monogame 3.5*. Retrieved from <http://community.monogame.net/t/monogame-on-raspberrypi/8143/16>
- National Transportation Safety Board. (2010). *Introduction of Glass Cockpit Avionics into Light Aircraft* (NTSB/SS-01/10 PB2010-917001). Washington, DC. Retrieved from <http://www.nts.gov>
- Parrish, Russell V., Bailey, Randall E., Kramer, Lynda J., Jones, Denise R., Young, Steven D., Arthur, Jarvis J. III, Prinzel, Lawrence J. III, Glaab, Louis J., and Harrah, Steven D. (2008). *Aspects of Synthetic Vision Display Systems and the Best Practices of the NASA's SVS Project*. NASA publication TP-2008-215130.

- Sonnino, Bruno. (2017). *Internet of Things - Working with Raspberry Pi and Windows 10*. Microsoft Developers Network Magazine. Retrieved from <https://msdn.microsoft.com/en-us/magazine/mt808503.aspx>
- Struger, Filip. (2010). *Continuous Distance-Dependent Level of Detail for Rendering Heightmaps (CDLOD)*. Retrieved from http://www.vertexasylum.com/downloads/cdlod/cdlod_latest.pdf
- Terrain Data. (2017). In WebGIS.com. Retrieved from <https://www.webgis.com/terraindata.html>
- U-blox. (2014). *U-blox 7 GNSS Modules Datasheet*. Retrieved from https://www.u-blox.com/sites/default/files/products/documents/NEO-7_DataSheet_%28UBX-13003830%29.pdf
- VFR Raster Charts. (2017). In *Federal Aviation Administration Aeronautical Information Services*. Retrieved from https://www.faa.gov/air_traffic/flight_info/aeronav/digital_products/vfr/
- Wilkinson, Darren. (2012). *Benchmarking python speed on the Raspberry Pi*. Retrieved from <https://darrenjw2.wordpress.com/2012/07/02/benchmarking-python-speed-on-the-raspberry-pi/>
- Williamson, G. W. (1937). *Instrument Planning: The New Service Blind-Flying Panel Described*. Flight. Retrieved from <https://www.flightglobal.com/pdfarchive/view/1937/1937%20-%202317.html>