Fixed-Wing Survey Drone

by
Danielle Johnson & Ben Gorgan

Dr. Joseph A. Driscoll (Faculty Advisor)

Bradley University
Electrical Engineering Department

Senior Project
Fall 2013 - Spring 2014

Abstract

This project will develop an unmanned aerial system, commonly referred to as a “drone”. The drone will start as a commercially-available radio-controlled airplane. The addition of microcontrollers, cameras, and other sensors will allow the aircraft to operate autonomously. The purpose of the drone is to perform aerial imaging surveys of user-specified regions, such as crop fields. Such systems are commercially available, but are expensive and lack some features our drone will provide. The market for unmanned aerial systems is projected to grow by $82 billion from 2015 to 2025, and create over 100,000 new jobs. The majority of this growth is expected to come from “precision farming.” This approach to agriculture uses large amounts of data to determine the health and quality of crops. Such data can include soil moisture from sensors spread throughout a field, and various types of aerial imaging. The goal is to use this data to determine where resources such as fertilizer and water are needed most, therefore avoiding waste and improving crop yield. This project will develop a drone to provide imagery for use in precision farming applications. By producing a high-resolution image of a crop field using specific optical filters, health of the crops can be determined. An important feature of our drone is its autonomous nature. The operator first defines the boundaries of the survey area via GPS coordinates in Google Earth. Next, the drone is hand-launched somewhere near the field. The drone determines its location (via GPS) and follows the navigational route allowing it to image the entire area. The drone lands in the location from which it was launched, or some other user-specified location, with RC transmitter control. The images are assembled into a single large image, which is filtered to generate the NDVI image. Here we report on our preliminary results for this ongoing project.

Table of Contents

Introduction

This project will develop an unmanned aerial system, commonly referred to as a “drone”. The drone will start as a commercially-available radio-controlled airplane. The addition of microcontrollers, cameras, and other sensors will allow the aircraft to operate autonomously. The purpose of the drone is to perform aerial imaging surveys of user-specified regions, such as crop fields. Such systems are commercially available, but are expensive ($10000+) and lack some features our drone will provide.

The market for unmanned aerial systems is projected to grow by $82 billion from 2015 to 2025, and create over 100,000 new jobs [1]. The majority of this growth is expected to come from “precision farming.” This approach to agriculture uses large amounts of data to determine the health and quality of crops. Such data can include soil moisture from sensors spread throughout a field, and various types of aerial imaging. The goal is to use this data to determine where resources such as fertilizer and water are needed most, therefore avoiding waste and improving crop yield.

This project will develop a drone to provide imagery for use in precision farming applications. By producing a high-resolution image of a crop field using specific optical filters, health of the crops can be determined. An important feature of our drone is its autonomous nature. The operator first defines the boundaries of the survey area via GPS coordinates. Next, the drone is hand-launched somewhere near the field. The drone determines its location (via GPS) and follows a navigational route imported from Google Earth allowing it to image the entire area. The drone lands in the location from which it was launched, or some other user-specified location, optional control from the RC transmitter.. The images are assembled into a single large image using Hugin image stitching software, and filtered through software to generate the NDVI image.

System Description

The drone will accept GPS (global positioning system) coordinates of a user-selected area of land. In this way, a particular region can be specified for the drone to survey. Using that data, it will follow the specified route to navigate the field and capture GPS-registered images of the entire area. The drone will be entirely autonomous, piloting itself on the programmed flight path, with the option to take manual control at any time. When the survey is complete, software will assemble all of the images it has captured into a single, high-resolution image of the entire area. That image will be filtered to obtain the NDVI image. After surveying the entire area, the plane will land where it was launched, or at another user-specified (via GPS) location.

The survey drone will have the following components:

- Ready-to-fly, commercial, radio control airplane (hand launched, electrically powered, servos and radio receiver included)
- Near-Infrared Digital Camera: for high-resolution aerial imagery
- Microcontrollers: for high-level autonomous behavior and image processing
- GPS Receiver
- An inertial measurement unit (IMU) containing three accelerometer axes, three gyroscope axes, three magnetometer axes, and a barometric sensor (for altitude).
- PWM servo driver: to move ailerons, elevator and rudder
- LiPo Battery
- BEC power converter: to convert battery power to correct voltage levels
- RC Servo Multiplexer: to switch between autonomous and RC mode

Block Diagrams

The fixed-wing survey drone has two primary subsystems. The first is the autopilot system, which includes the GPS receiver, the IMU, and the BeagleBone Black microcontroller. The second subsystem provides image processing. This system utilizes the BeagleBone Black (or equivalent) microcontroller, GPS receiver, and image processing software to capture all of the images and to create the final complete image.

Autopilot System

The autopilot subsystem (Figure 1) runs on the BBB, a Linux-based single-board computer. The developed autopilot software will be loaded onto the BBB where a route for the aircraft will be automatically generated based on the user input. Google Earth path input data will be sent to the BBB to process and establish navigational waypoints. The aircraft will follow its set path autonomously, driven by the autopilot system. If emergency remote control is necessary, manual operation can be enabled at any point during flight. A standard R/C transmitter will initiate a signal to the BBB to shut down, and the aircraft will respond only to manual control. The transmitter's signals will manipulate the servos and electronic speed control on the aircraft to direct flight.

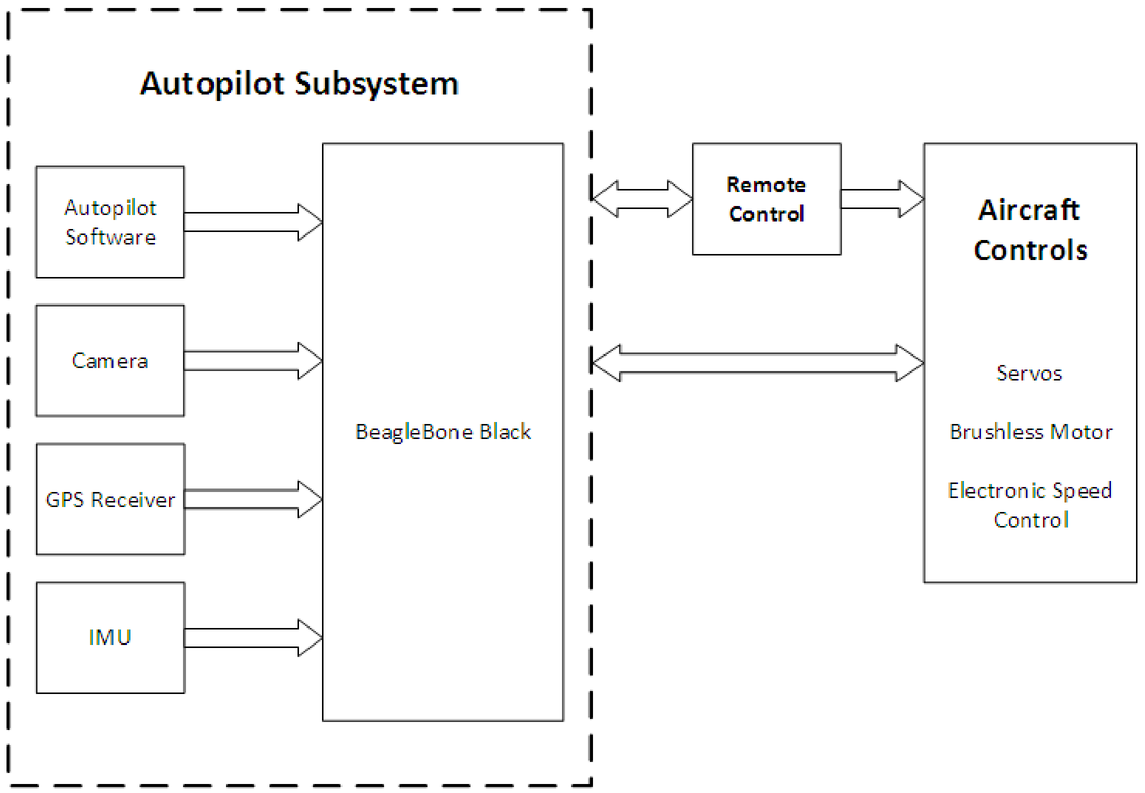


Figure 1. Autopilot Block Diagram

During flight, The IMU will communicate information from the gyroscope, compass, accelerometer, and barometer to the BeagleBone Black. The GPS will determine current location at the rate of 1Hz. The autopilot PID control software will automatically adjust the ailerons, elevator and rudder to maintain flight stability and GPS route navigation. The switch between manual control and autonomous control may be made at anytime using the RC transmitter.

Image Processing System

Figure 2 shows the data collection and delivery process. As the drone passes over the selected area, the down-facing camera will collect overlapping images that cover the entire area. As each image is taken, the BBB will store location data from the GPS receiver. After flight, these images will be assembled into one image of the entire user-selected area based on overlap and GPS location if necessary. This camera will also have optical filter capabilities to allow specialized imaging applications.

The camera will use an “infrablue” filter. This filter allows a near infrared image to be captured in the red channel of the camera, while still capturing the visible spectrum in the blue channel. After all the GPS data and images have been stored in the BBB, the image processing software will create a single large image of the surveyed area. After the images have been stitched together, further filtering will be done to provide an image that may be used to analyze crop health according the the Normalized Difference Vegetation Index.

The Normalized Difference Vegetation Index, or NDVI, is a simple indicator used by NASA to determine if there is live plant life. The index ranges from -1 to 1, with 1 appearing as a bright green. The scale is calculated using the equation, NDVI = (NIR - VIS)/(NIR + VIS), where NIR is the near infrared channel, and the VIS, is the visible channel. This works on the fact that plants absorb red and blue light, while reflecting the infrared.

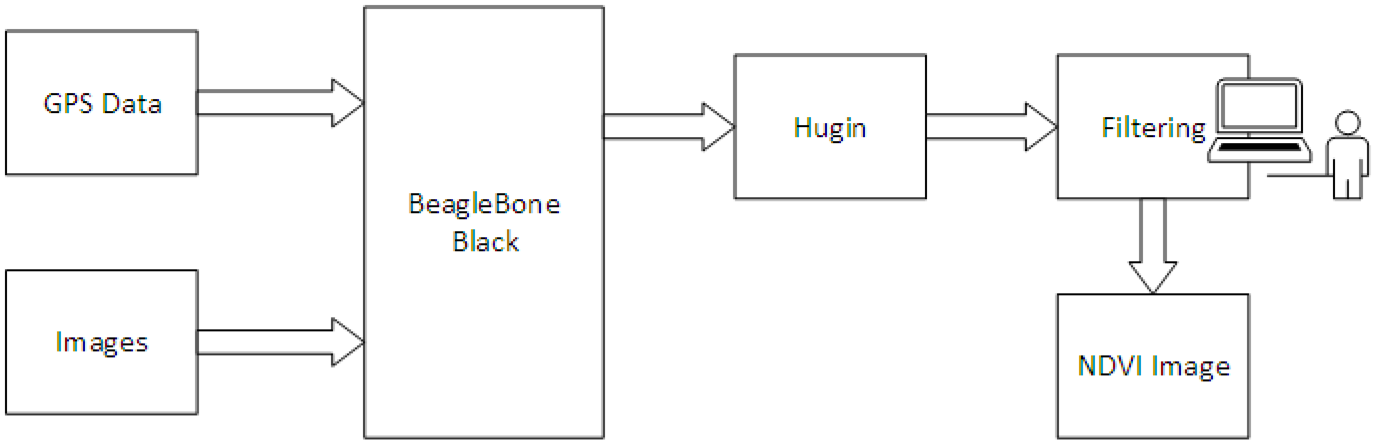


Figure 2. Image Processing Software and Final Filtering

Design Equations and Calculations

The only calculations needed for the project were bearing, distance to waypoint, and current heading. The bearing calculations use trigonometry to compare current latitude (φ) and longitude (λ) to the next waypoint coordinates. The function atan2 is simply arctangent with two inputs. The bearing (θ) calculation in degrees is:

$$\theta = \mathrm{atan2}(\sin(\Delta\lambda)*\cos(\varphi_2), \cos(\varphi_1)*\sin(\varphi_2) - \sin(\varphi_1)*\cos(\varphi_2)*\cos(\Delta\lambda))$$

In python code, the following lines calculate bearing:

```
y = sin(lon2-lon1) * cos(lat2)
x = cos(lat1) * sin(lat2)  - sin(lat1) * cos(lat2) * cos(dLon)
atan2(y, x)
```

The distance calculations are computed using the Haversine method shown below:

$$a = \sin^2(\Delta\varphi/2) + \cos(\varphi_1)*\cos(\varphi_2)*\sin^2(\Delta\lambda/2)$$
$$c = 2*\mathrm{atan2}(\sqrt{a}, \sqrt{1-a})$$
$$d = R*c/1000$$

The Haversine method uses trigonometric calculations to compare two points on a sphere, then scale to the sphere's radius. The python code compares current latitude (φ) and longitude (λ) in radians to the next waypoint coordinates and scales the distance to the Earth's radius (R = 6,371km). The final distance is in meters.

The heading is calculated by computing the arctangent of the most current magnetometer readings on the x and y axes and converting that value to degrees. In code, this calculation is:

```
heading=math.degrees(math.atan2(-m[1], m[0]))
```

Circuit Diagrams

Figure 3 shows the setup diagram of all of the components in the plane. The BeagleBone Black connects to the camera over a USB connection. The GPS connects over a UART serial connection, and the imu connects over an I2C interface. The RC mux switch connects to both the RC receiver and the PWM servo driver. When the RC transmitter channel is switched to RC control, the RC receiver drives the servos on the plane for moving the ailerons, elevator, rudder, and throttle. When the RC transmitter channel is switched to autonomous control, the BeagleBone Black drives the servos over the PWM servo driver connection. The BEC converters are necessary to convert the 11.1V battery to the 5V raw the BeagleBone Black and components require. Two BEC converters were used to avoid exceeding current limits. One BEC provides power to the servo signals, while the other provides power to the BeagleBone Black.

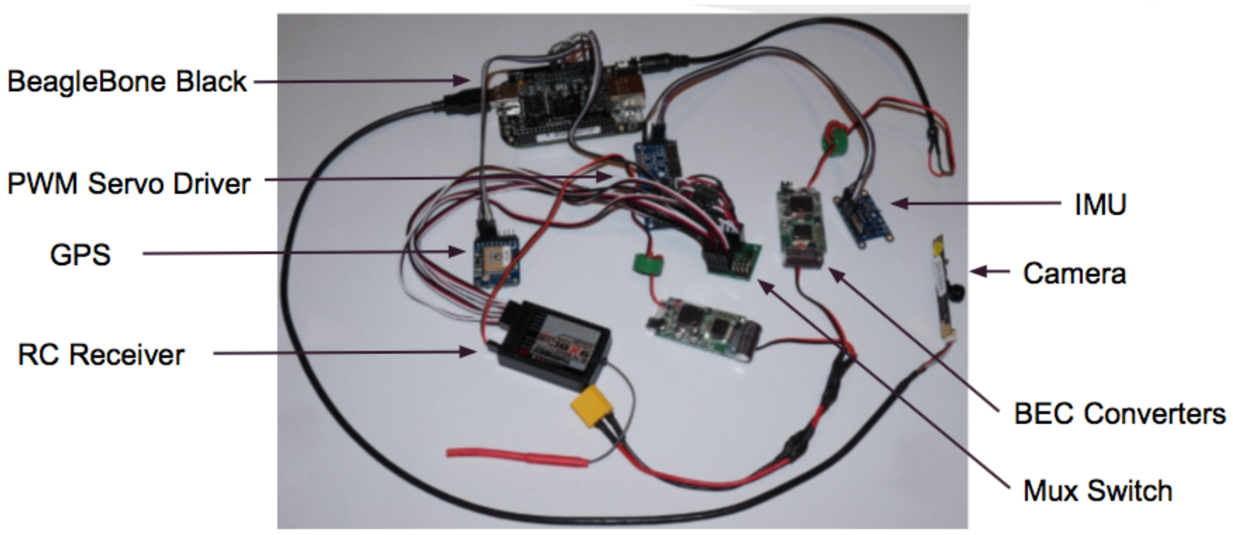


Figure 3 Plane components diagram

Program Flow Charts

The GPS Navigation system software flowchart is shown in Figure 4. First the user inputs a waypoint path in the Google Earth “GUI.” The steps for using Google Earth to save .kmz path files is listed below:

1. Open Google Earth
2. Locate survey area
3. Draw path of waypoints
4. Save path as a .kmz file
5. Transfer file to BeagleBone Black
6. Input file name to navigation program

Once the path is input to the program, user interface is complete. The code parses the file, computes the waypoints, calculates distance and bearing, and adjusts flight course if necessary by position the rudder with pwm signals. Once the last waypoint is reached manual control will be resumed and the user will land the plane.

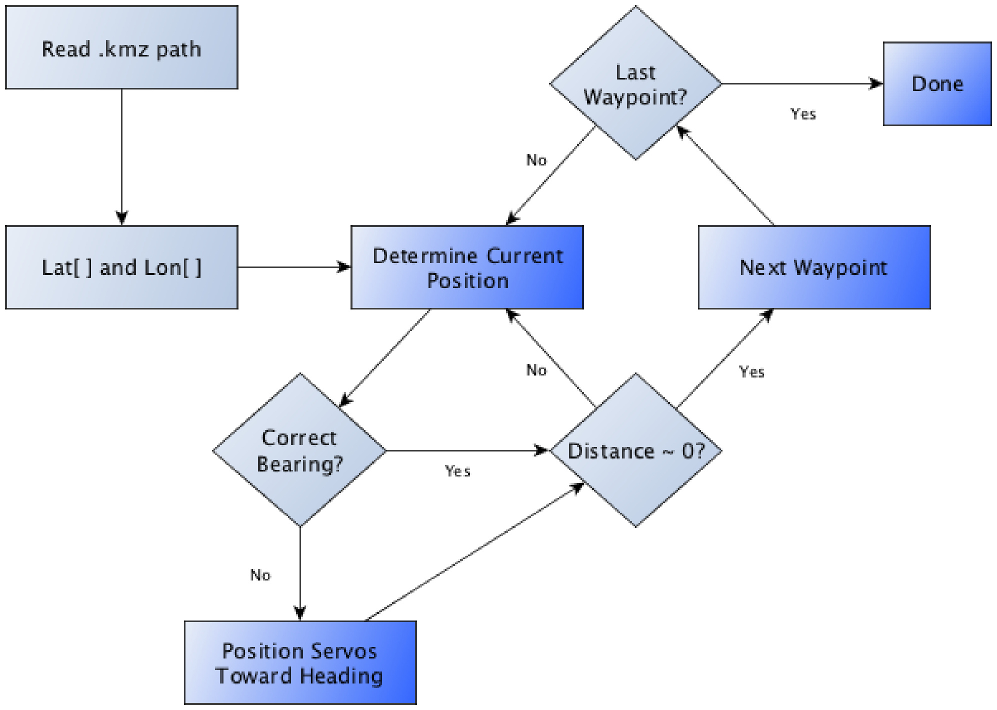


Figure 4. GPS Navigation System Flowchart

The PID servo control software flowchart is shown in Figure 5. All servos are initialized to center positions, the sensor buses are set up, and PID control sensor setpoints are initialized to level flight values. During flight, the sensors will provide readings about once per 80ms period. If adjustment is needed for stable flight, the current accelerometer values will be offset from the accelerometer setpoints. This error will be calculated and weighed with previous error to calculate the amount of servo position adjustment (to the corresponding ailerons, elevator, and rudder) needed to maintain stable flight. The process will be repeated in the loop for the entire autonomous portion of the flight.

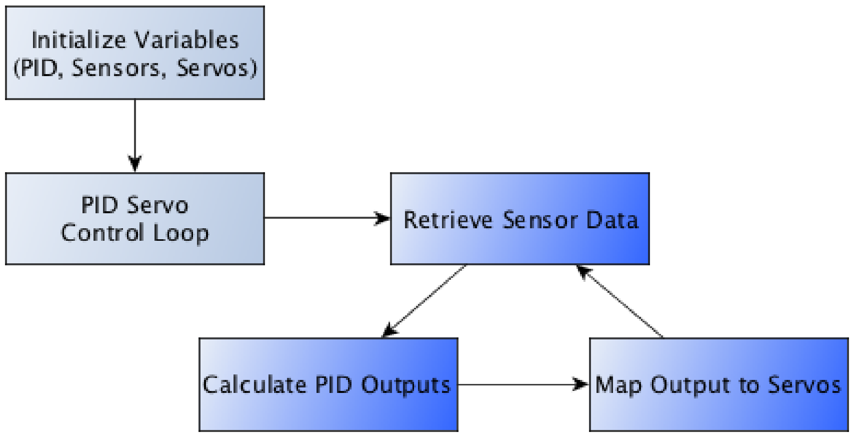


Figure 5. GPS Navigation System Flowchart

When the complete code is executed, the GPS navigation system and PID servo control loop are run in separate threads so timing is not an issue. The “take_picture” script is also run in a separate thread, which prompts the camera to take a picture every few seconds.

Software

Hugin is an open-source panorama-stitching software for Linux. With the addition of autopano-sift-C, an automatic control point generator, the software is capable of automated panorama generation. [7] Hugin is used to stitch the retrieved field images together into one image. The procedure is simple and includes three steps.

In Hugin Assistant:

1. Load all images
2. Select “Align”
3. Select “Generate Panorama”

Using *Infragram* we are able to analyze how healthy crops are using our near infrared images. Infragram works by compositing the infrared and visible channels using the infrablue camera. The result of the Infragram filtering shows that areas that have high photosynthetic activity are very bright. Comparing the image with Normalized Difference Vegetation Index, one can determine which areas are healthy or not. With the combination of our infrablue camera and the Infragram software we are able to develop images that assess crop health.

Programs

The complete code for the project is located in the zip file FWDrone.zip. A list of file descriptions is provided. *PID.py* is the main python script that must be run to start autonomous drone control.

Libraries to Install (not included in the zip file, the *pip install* command will be needed)
Bitstring
Pynmea2
Adafruit Python
Adafruit BBIO

Device Drivers
l3gd20py.py
lsm303.py
Adafruit_PWM_Servo_Driver.py
map_ultimateGPSv3.py

Calculation and Parsing Libraries
calc_dist_bear.py
kmz_parser.py

Test Files
all_test.py
i2c_test.c
PWM_test_I2C.py
PWM_test.py
test_ultimateGPSv3.py
thread_test.py

Google Earth .kmz Files
Alumni.kmz
soccer_field.kmz
Alum.kmz

Data Files
data.csv

data.xlsx
log.txt

Scripts
go.sh
take_picture.sh
take_pictures_loop.sh

Links to the libraries and python code we referenced are included below.

Infrapix	https://github.com/p-v-o-s/infrapix
Pynmea2	https://github.com/Knio/pynmea2

Kmz_parser
http://programmingadvent.blogspot.com/2013/06/kmzkml-file-parsing-with-python.html
Distance and Bearing
http://code.activestate.com/recipes/577594-gps-distance-and-bearing-between-two-gps-points/
Adafruit pwm servo driver library
https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code/tree/master/Adafruit_PWM_Servo_Driver

Analysis of Results

Most of the drone testing was done on the ground in simulation, connected to a laptop. Current magnetometer and accelerometer data readings were printed to the screen in Putty. PID positional output to the servos was also printed. When tested, IMU movement resulted in servo adjustment to compensate for the change in flight stability. Actual flight tests resulted in a short ascent followed by acceleration toward the ground. Weight distribution would need to be more carefully considered to be able to observe a successful flight.

Figure 6 shows accelerometer data values from conducting a simple test with the IMU. First, the IMU was tilted 90 degrees left, then 90 degrees right, resulting in spikes in the y-component of the accelerometer (red line). Then, the IMU was tilted 90 degrees forward, and 90 degrees backward, resulting in spikes in the x-component of the accelerometer (blue line). The green line represents the z-axis component of the accelerometer, which is equal to -1G when flight is stable (due to gravity). The x and y components are zero when flight is stable. Some variation is seen on the z component, since movement is not limited to x and y components.

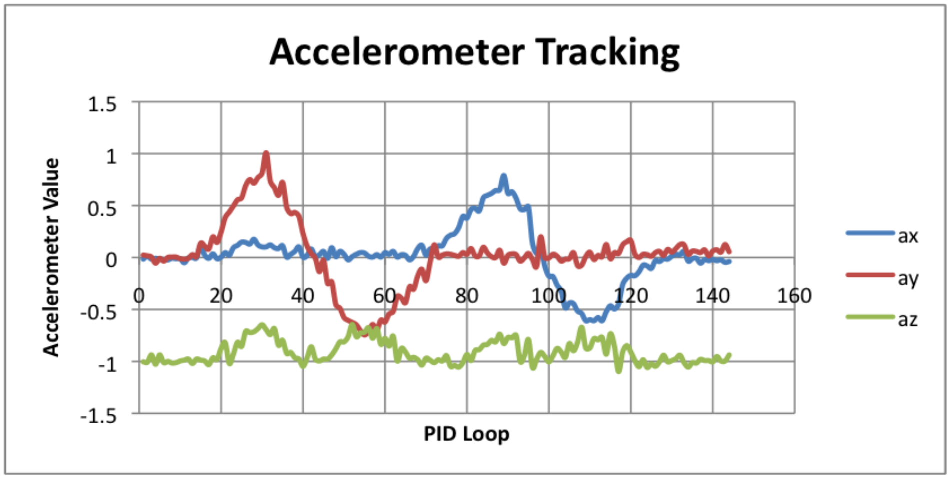


Figure 6. Accelerometer Reading after each PID Loop

The changes in aileron position are proportional to the error and integral error between accelerometer setpoint values and actual accelerometer values. In Figure 7, the blue line shows actual y-axis accelerometer data, while the red line shows aileron servo position response. In response to the increasing error between the accelerometer setpoint (zero) and the actual accelerometer value, the servo position changes based on how much stabilization compensation is needed. The servo reaches it's maximum (400) and minimum (200) positions during testing, since the IMU is tilted nearly 90 degrees from its stable flight position. The plot shows aileron position is adjusting in time with the changes in accelerometer readings, so the PID loop response time is appropriate for this application.

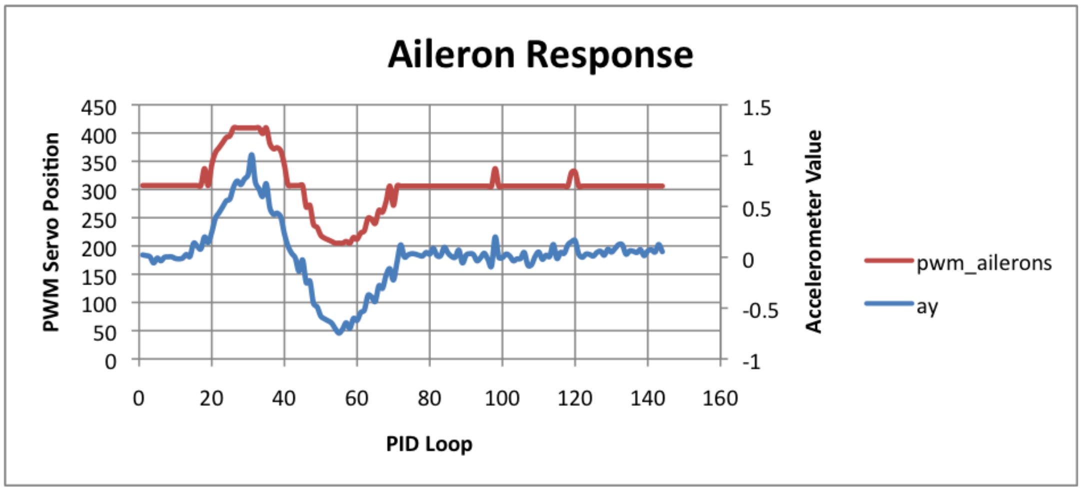


Figure 7. Aileron Response to Changes in Accelerometer values

To test the functionality of the GPS navigation system without the requirement of complete flight stabilization, the GPS navigation system was ported to an RC car. To test the code, a destination waypoint was set and the car moved forward until it reached within 1 meter of that destination point, then stopped. This result demonstrated our code's ability to analyze .kmz file types, correctly determine current GPS location, compare it to its waypoint destination, and recognize when the car has reached that destination.

An example of the Google Earth .kmz file used is shown below in Figure 8. The distance of the last navigation leg is shown in the screenshot. Figure 9 shows our program's correct calculation of that distance, based on corresponding latitude and longitude coordinates.

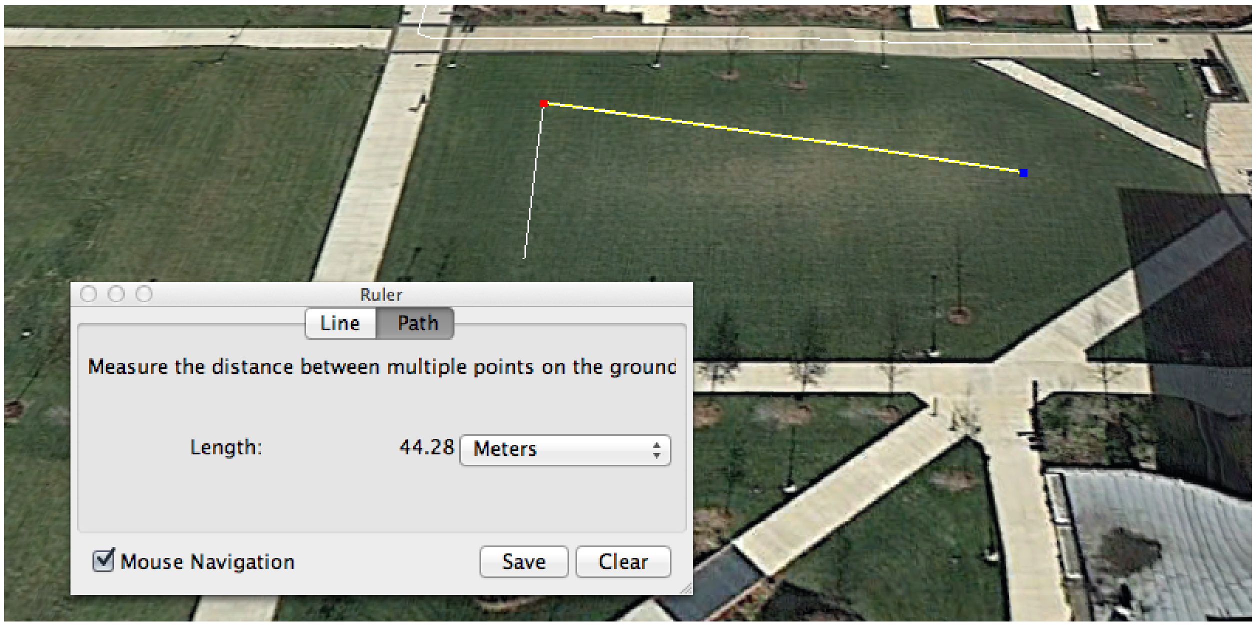
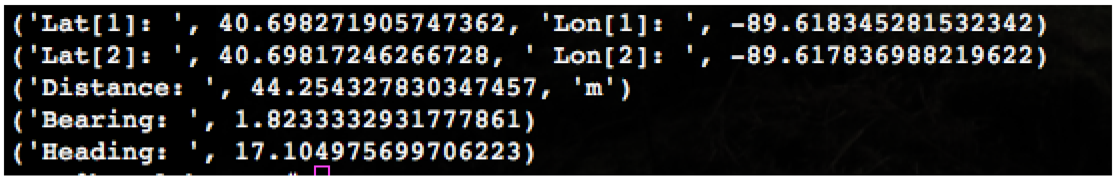


Figure 8 Google Earth .kmz file in the Bradley University Alumni Quad



```
('Lat[1]: ', 40.698271905747362, 'Lon[1]: ', -89.618345281532342)
('Lat[2]: ', 40.69817246266728, ' Lon[2]: ', -89.617836988219622)
('Distance: ', 44.254327830347457, 'm')
('Bearing: ', 1.8233332931777861)
('Heading: ', 17.104975699706223)
```

Figure 9 Program Coordinates, Distance, Bearing and Heading Calculations

The automatic image stitching procedure in Hugin was successful. Although we were not able to stitch aerial photos, we were still able to stitch panoramas with the procedure described in this document, as long as the photos chosen had some overlap.

NDVI image filtering was successfully implemented using photos obtained from the Infragram camera. The filtering may be done using the python library *Infrapix* or by uploading the image to infragram.org and selecting NDVI. An example is shown in Figure 10.

Data Sheets

Links to the product pages are provided here. The datasheet may be obtained from these sites.

Bixler Aircraft
http://www.hobbyking.com/hobbyking/store/__18214__hobbyking_bixler_epo_1400mm_kit_de_warehouse_.html

Turnigy RC Controller and Receiver
http://www.hobbyking.com/hobbyking/store/__8992__turnigy_9x_9ch_transmitter_w_module_8ch_receiver_mode_2_v2_firmware_.html

BeagleBone Black: http://beagleboard.org/Products/BeagleBone+Black

Adafruit MTK3339 GPS: http://www.adafruit.com/products/746

Adafruit IMU Breakout (L3GD20, LSM303, BMP180): http://www.adafruit.com/products/1604

Adafruit 16-Channel 12-bit PWM/Servo Driver: http://www.adafruit.com/product/815

Turnigy SBEC 5V 5A Power Converter:
https://www.hobbyking.com/hobbyking/store/__10313__Turnigy_5A_8_40v_SBEC_for_Lipo.html

Turnigy 2.2 LiPo Battery:
http://www.hobbyking.com/hobbyking/store/__9394__turnigy_2200mah_3s_30c_lipo_pack.html

Pololu 4-Channel RC Servo Multiplexer: http://www.pololu.com/product/2806

Infragram DIY Plant Analysis Webcam: http://www.adafruit.com/products/1722

Conclusions

In conclusion, each component of our project was successful, but complete integration would require more time. We were able to design a responsive PID servo control system, a functional GPS navigation system, a friendly Google Earth waypoint entry interface, and a method for near-infrared image retrieval, filtering, and stitching. Once the drone is operational, future work could include multi-drone collaboration, near-infrared imagery search and rescue, GPS package delivery, and obstacle avoidance capabilities. At this stage, the project is a solid prototype autopilot system that could serve as a foundation for other low-cost drone projects. Our final price list totals $343. With more time, careful documentation, and additional contribution and feedback from the drone community, this project could become an open-source autopilot system for the BeagleBone Black and a low-cost alternative to $7,000 commercial survey drones.

References

[1] Association for Unmanned Vehicle Systems International (AUVSI). *The Economic Impact of Unmanned Aircraft Systems Integration in the United States*. March 2013. Available at http://www.auvsi.org/econreport

[2] Chao, HaiYang, YongCan Cao, and YangQuan Chen. "Autopilots for Small Unmanned Aerial Vehicles: A Survey." *International Journal of Control, Automation, and Systems*. 8.1 (2010): 36-44. Web. 25 Sep. 2013. <DOI 10.1007/s12555-010-0105-z>.

[3] Felderhof, Leasie. *Linking UAV (unmanned aerial vehicle) technology with precision agriculture*. Diss. James Cook University, privately published, 2008. Web. <http://eprints.jcu.edu.au/8029/>

[4] Grenzdörffer, G. J. "THE PHOTOGRAMMETRIC POTENTIAL OF LOW-COST UAVs IN FORESTRY AND AGRICULTURE ." Diss. Rostock University, 2008. Web. <http://www.isprs.org/proceedings/XXXVII/congress/1_pdf/206.pdf>.>.

[5] Haitao Xiang, Lei Tian, "Development of a low-cost agricultural remote sensing system based on an autonomous unmanned aerial vehicle (UAV)", Biosystems Engineering, Volume 108, Issue 2, February 2011, Pages 174-190, ISSN 1537-5110, http://dx.doi.org/10.1016/j.biosystemseng.2010.11.010. <http://www.sciencedirect.com/science/article/pii/S1537511010002436>

[6] Jensen, A., M. Baumann, and Y-Q Chen. "Low-Cost Multispectral Aerial Imaging using Autonomous Runway-Free Small Flying Wing Vehicles." *Geoscience and Remote Sensing Symp*. (2008): 506-509. Web. 25 Sep. 2013. <http://mechatronics.ece.usu.edu/yqchen/paper/08/08C30_igarss_4051FinalPaper.pdf>.

[7] Duell, Terry. “Hugin Tutorial - Stitching Multi-row Photos Together.” Hugin Sourceforge. 2008. Web. <http://hugin.sourceforge.net/tutorials/multi-row/en.shtml>