Flying Android Project

Andrew Bardagjy Brian Ouellette Paul Varnell
Jean-Pierre de la Croix

ECE 4007 L01

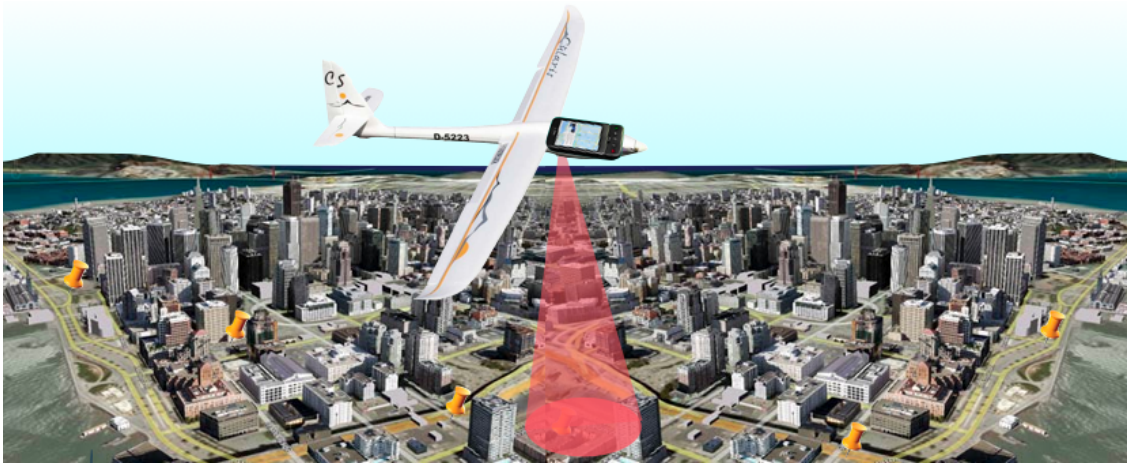
December 12, 2009

Contents

Executive Summary

The Flying Android Project aims to design and fabricate software and hardware systems needed for a smartphone to fly a small, inexpensive, unmanned aerial vehicle. The project is sponsored by the Department of Defense's (DOD) Rapid Response Technology Office (RRTO) in conjunction with the Georgia Tech Research Institute (GTRI). The sponsors are interested in a platform that provides a rapidly deployable reconnaissance solution and can be widely distributed to warfighters to gather intelligence information to support their mission. The software running on the smartphone will control the flight of the aircraft without human intervention. The software loads the GPS waypoints from a file and direct the aerial vehicle to travel to each waypoint. When the plane reaches a waypoint, it takes a photograph of the location and stores it locally. The plane can operate for at least 15 minutes before it needs to land and recharge. The project aims to fly the aircraft with only the built in functionality of the smartphone. The cost of the vehicle is approximately $900. The vehicle is based on the Android powered G1 developer smartphone and the Multiplex EasyStar glider. During the development, there were three principle milestones. These milestones are interfacing the G1 hardware to the aircraft, design of the safety pilot failover, and implementation of flight controls. The first two milestones have been met and the flight control is underway. Once flight control is demonstrated, a variety of future work is possible, including new interfaces to the plane from the user.

1 Introduction



The Flying Android Team spent $2,000 of funding to develop the smartphone powered aerial vehicle. The Flying Android Project is a two semester project and is currently underway.

Unmanned Aerial Vehicles (UAVs) are increasingly sought for applications as diverse as target illumination and crop dusting [1]. Presently, UAVs are expensive, custom machines which often suffer from limited availability while they are under repair [2]. There is a growing need for an inexpensive UAV that can be assembled from readily available off-the-shelf components to serve hobbyist, research, government, and commercial customers [3].

In particular, government agencies such as the project's sponsor, the Department of Defense's Rapid Response Technology Office (RRTO) in conjunction with the Georgia Tech Research Institute (GTRI) are interested in such a platform to provide rapidly deployable reconnaissance solutions.

By using a smartphone to pilot a commercially available airframe, a low cost UAV can be rapidly constructed from commonly available components. In particular, the Android powered G1 developer smartphone and the Multiplex EasyStar glider are well suited to such an application as they both retail for around $200.

1.1 Objectives

Software for the phone was written to estimate the vehicle's pose, compute traversable paths, and keep the vehicle steady in flight. A hardware interface was constructed to allow the phone to communicate with the vehicle. The camera on the smartphone was used to provide aerial reconnaissance data to the customer.

When given Global Positioning System (GPS) coordinates of a set of waypoints, the system will autonomously plot a flyable route between the points, execute the route and capture aerial photographs at specified coordinates. The vehicle is hand launched and lands with the help of a human pilot. For safety reasons, a safety pilot is always able to take control of the vehicle in the event of an emergency.

1.2 Motivation

Accurate intelligence data is essential to increase the effectiveness of the modern warfighter and reduce unnecessary collateral damage. Presently, the number of intelligence assets (vehicles that produce intelligence data) are fewer than the number of customers. Thus, assets are scarce and current up-to-date intelligence imagery is difficult to obtain [4].

This project's aim is to provide a small, inexpensive, unmanned aerial vehicle that can be widely distributed to the warfighter so he can gather intelligence information to support his mission.

1.3 Background

Many products are currently available that can provide up-to-date intelligence imagery to the warfighter. Unfortunately, a typical unmanned aerial vehicle costs up to $50,000,000 to design and $500,000 to acquire. Furthermore, the development cycle of such a vehicle is 5-10 years.

In recent years, smaller, more affordable UAVs have become available; however, they

require custom electronics to navigate. This project leverages technology advances in modern cellphones to further reduce development time and acquisition cost.

2 Project Description and Goals

The goal of the Flying Android Project was to design and build the software and hardware needed for a smartphone to fly an aircraft. The project is being sponsored by the Department of Defense's Rapid Response Technology Office (RRTO) in conjunction with the Georgia Tech Research Institute (GTRI). The goal of the project is for the smartphone to fly the aircraft with its built-in functionality alone. The final product is intended primarily for military use.

3 Technical Specifications

3.1 Functional Specifications

Table 1 details the features required for the Flying Android Project.

The plane must be able to operate for at least 15 minutes to complete an entire mission. Complicated missions with increased range may require extended flight times, but this specification provides a baseline for mission times. During testing the plane was able to operate with a full payload for 15 minutes without any loss of power. The plane was brought down after 15 minutes but in extreme scenarios could operate beyond that limit.

The plane must be able to carry a load of up to 500 grams, including the weight of the smartphone and the interface board, while maintaining the desired minimum sustained flight time. If the payload is lighter than expected, the flight time may be extended or additional hardware may be added. The plane was able to carry the full hardware of approximately 500 grams during initial tests.

The aircraft must navigate without human intervention using data from the phone's

Global Positioning System (GPS) receiver, compass, and accelerometers. The vehicle must be able to accurately photograph targets on the ground from the air. The aircraft must take photographs which are centered no more then five meters from the desired reconnaissance target. The typical resolution of commercial GPS is approximately one to two meters, so this specification is tolerant to some inaccuracies in navigation and control. The onboard GPS sensor is used to characterize the aircraft's navigation accuracy. Observed GPS resolution is within two meters and updates occur around once a second. The update frequency is a problem when the aircraft is traveling near its top speed. The aircraft currently does not navigate without human control.

To effectively function as an aerial reconnaissance platform, the camera on board must be of sufficient resolution to discern objects on the ground from flight altitude. The 3.2 megapixel camera standard on the G1 developer smartphone is sufficient for this purpose. This is a common camera resolution for smartphones, so if the target smartphone is changed, it will likely meet this specification.

In the event that the navigation system fails, an RC (Remote Control) safety pilot must always be available. The safety pilot is activated and controlled via an RC controller on the ground. The safety pilot was tested and works correctly.

Table 1: Functional specifications.

Feature	**Specification**
Minimum sustained flight time	15 minutes
Payload capacity	500 grams
Navigation accuracy	5 meters
Image quality	3.2 megapixel
Safety	RC safety pilot available at all times

3.2 Interface Board Specifications

The aircraft carries the G1 developer smartphone and a board that interfaces the phone and the plane's controls as well as the safety pilot. The specifications for the interface board are shown in Table 2. In order to fall below the carrying capacity specification from Table 1 the board weighs less than 350 grams.

The G1 communicates to the interface board via a serial TTL level signal from the USB connector. The G1's ability to output information over a serial TTL signal is confirmed. The commands are then relayed to the servos connected to control surfaces and the motor controller driving the thrust motor. These commands are communicated via a pulse-width-modulated (PWM) signal.

Additionally, the board must be generally rugged as the vehicle may encounter rough air, turbulence, and harsh landings. Furthermore, the phone and the interface board must be mounted in such a way to resist vibration caused by sudden gusts of wind. The board and plane are both mounted inside the foam fuselage of the plane to minimize all these factors.

Finally, the interface board is powered from the battery which drives the airframe. The power consumed by the interface board will affect mission endurance and must be considered during component selection. The interface board consumes minimal power and does not reduce mission endurance below the specification.

Table 2: Interface board specifications.

Feature	**Specification**
Weight	350 grams
Interface to phone	Serial TTL
Interface to motor/servos	PWM signal

3.3 Software Specifications

The required specifications of the software running on the G1 developer smartphone are shown in Table 3. GPS and compass integration are integral to the autonomous control of the plane and are essential to successfully navigate GPS waypoints [2]. GPS and compass data were successfully captured for use and logged.

Additionally, the receipt of a phone call or text message while in flight will not effect the operation of the aircraft. Due to the Android operating system, the software is robust enough to handle this situation. A currently unimplemented feature which will be added is a warning message in the event of a low battery condition on the phone or aircraft. Also unimplemented is a feature that if the aircraft is lost during a sortie and the phone is still active, it will broadcast its GPS location via text message to aid recovery.

Table 3: Software specifications.

Feature	**Purpose**
GPS integration	Retrieve location
Compass integration	Retrieve heading
Robustness	Handle texts and low battery in flight
Lost and found recovery	GPS recovery via text

4 Design Approach and Details

4.1 Design Approach

4.1.1 Flight Hardware

The heart of the system is the G1 developer smartphone, a smartphone manufactured by HTC which runs the Android operating system. The developer version of the smartphone is the same as a retail T-Moble G1, but it is software unlocked (rooted) and will accept any SIM card (unlocked) [5]. The phone has a 3-axis accelerometer, a 3-axis magnetic compass, GPS, and a camera. It can communicate via GSM, WiFi, and Bluetooth [6].

The smartphone was integrated with a Multiplex EasyStar ARF Electric RC aircraft. The aircraft has two control surfaces: the tail elevator and the tail rudder. The elevator is controlled by a single servo and determine the pitch of the aircraft. Pitch determines whether the aircraft will gain or lose altitude. The tail rudder is used to control the yaw of the aircraft. A second servo moves the tail rudder left or right, which causes the aircraft to turn in the same direction [7].

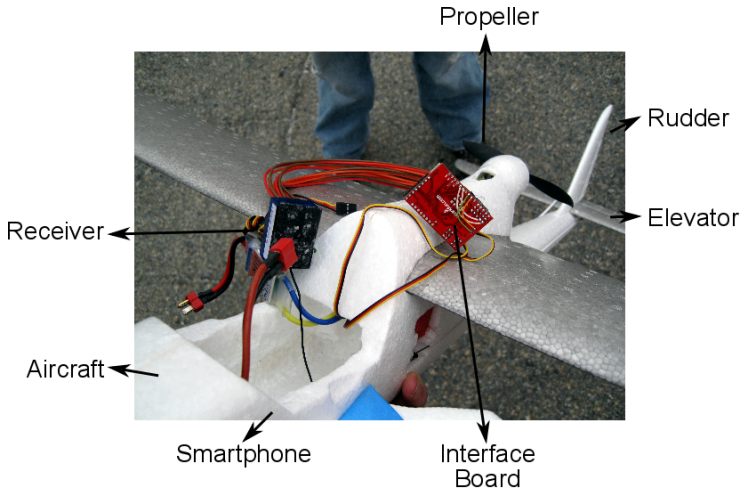


Figure 1: Installed hardware.

The aircraft was propelled through the air by a DC brushless motor powered push propeller. Push propellers are mounted facing backwards and physically push the aircraft for-

ward. Since DC brushless motors require three phased AC input, an electronic speed control (ESC) unit was used to convert a DC signal into the necessary AC signal.

The servos and motor (through the ESC) are connected to an interface board with a microcontroller based on the ArduPilot as seen in Figure 2. Software on the microcontroller was designed to convert the serial motor commands from the smartphone into PWM signals for the motor and servos. Refer to Figure 1 for an overview of the actual hardware.

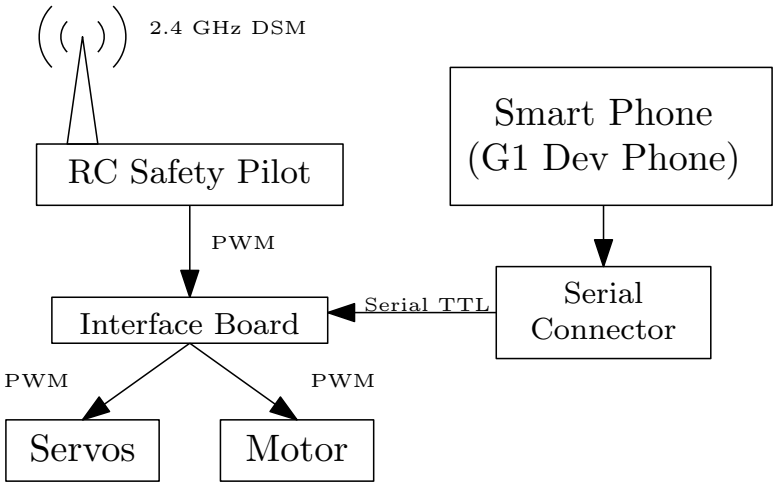


Figure 2: Hardware block diagram.

The smartphone was intended to electronically control the servos and motor by sending commands to the interface board. As shown in Figure 3, the G1 developer smartphone has a proprietary 11-pin HTC ExtUSB (Ext[ended] USB) Mini B port customarily used to connect the smartphone to a computer. Pins 2 and 3 in Figure 3 are used as the switch and ground pins; however, these pins were repurposed for serial TTL transmit (TX) and receive (RX), respectively [8].

An HTC ExtUSB 11 Pin USB Connector with a SparkFun breakout board was used to connect both pins to the interface board as seen in Figure 2. The smartphone sends commands directly through the serial TTL line to the microcontroller on the interface board.

A RC safety pilot receiver was also connected to the interface board to provide a fail safe in the event of control failure as seen in Figure 2. The RC safety pilot unit received signals from a 2.4GHz DSM (Digital Spectrum Modulation) transmitter operated by a pilot

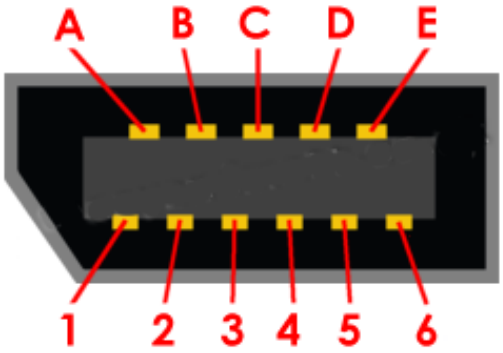


Figure 3: Extended USB Mini Port B the G1 Dev Phone.

on the ground. A 5-channel transmitter uses four channels to control the aircraft, leaving the fifth channel open for custom use. When the fifth channel was activated, the interface board ignored commands from the smartphone and used the PWM signals from the safety pilot to fly the aircraft instead. The safety pilot not only provides a fail safe for when the smartphone fails, but was also used for assisted takeoffs and landings.

4.1.2 Flight Software

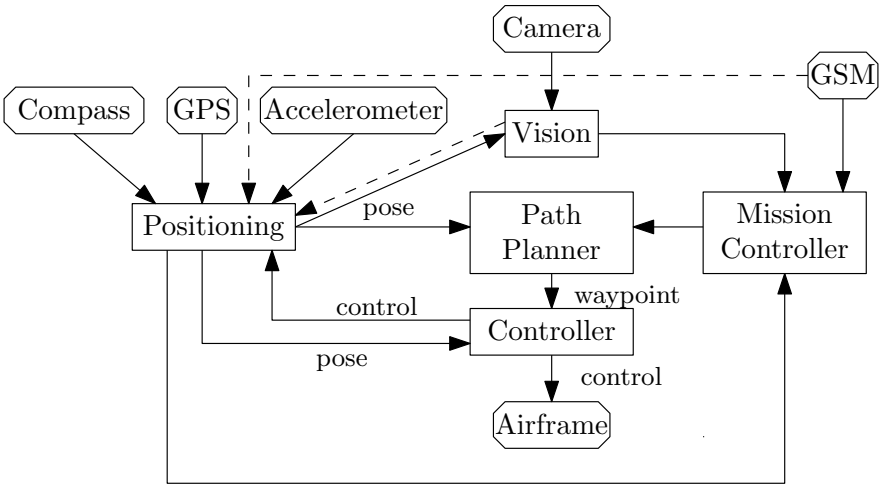


Figure 4: Software block diagram.

The overall software architecture is shown in Figure 4. Each square box is a software component that runs in its own thread. Octagonal boxes represent inputs and outputs to and from the device.

The flight controller was designed to send signals to the airframe control surfaces and propeller to keep the plane on the path generated by the path planning system and stabilize the aircraft. It used the same flight dynamics models as the positioning system along with the position data for accurate control. Proportional-integral-derivate (PID) controllers were used to stabilize the aircraft in flight. The controllers were based on controller code from the ArduPilot project; however, the code was ported to the Android framework. The controllers were intended to also allow the aircraft to navigate along a given path.

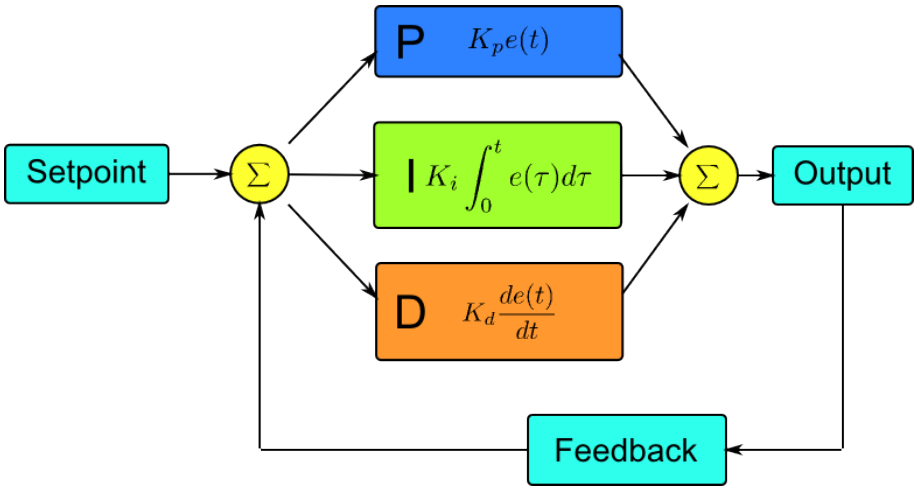


Figure 5: Block diagram of PID controller.

Figure 5 shows a block diagram of the PID controller. Three controller were implemented to control the roll, pitch, and throttle of the aircraft. The setpoint was calculated from the difference in bearing and distance between the aircraft and the desired waypoint. The controller minimized the error between the two positions and returned a motor control value.

The ArduPilot was configured for a different set of hardware: XY tilt sensor, Z sensor, airspeed sensor, and NMEA GPS; therefore, the output of the sensors on the smartphone were converted into values that were appropriate for the ported controllers. The accelerometers were used to substitute for the XY and Z sensors, while the GPS data points were integrated to estimate airspeed from ground speed.

The mission controller handled all of the high level details for a particular mission. In the case of the primary mission of GPS waypoint navigation and photography, the mission

controller read the predetermined GPS waypoints from a file and sent them to the path planning system. It also sent requests to the vision system to take pictures when the plane reaches each waypoint. In the future, the mission controller may communicate with the ground station.

The positioning system calculated the most likely position of the plane using the combined reading from all of sensors which measure position, including the GPS, accelerometer, compass, and feedback from the flight controller. It was intended to use models of the sensors and flight dynamics which will be created offline or found from similar existing applications. Additionally, an extended Kalman Filter was intended to combine the sensor measurements and generate a probabilistic representation of the state estimate.

The path planning system was intended to generate followable paths between the GPS waypoints using the flight dynamics model of the plane and send the paths to the flight controller.

The vision system retrieved images from the camera and processed them. In the future, the vision system may do more extensive image processing, such as computing the optical flow between images, which provides the relative motion of the camera to the ground [9].

In the future, the plane may also be controllable using a web applet. The web applet may allow the user to enter GPS waypoints dynamically and see the flight of the plane in real time. The web applet will run on a web server on the ground that will communicate with the smartphone on the plane using the phone's data plan.

Android is a open source operating system for smartphones. It provides a portable framework for writing mobile software. The G1 developer smartphone is built on top of the Linux kernel which provides a Java API for using GSM data connections, accessing sensors, and doing many other common phone tasks [5]. All of the software written for the plane was written using the Android Java API, because it significantly reduces the difficultly of accessing the hardware [10].

4.2 Codes and Standards

4.2.1 Flight Hardware

As previously mentioned, the G1 developer smartphone has a proprietary HTC ExtUSB (Ext[ended] USB) Mini B port. ExtUSB combines standard mini-USB with audio and video pins in an 11-pin connector. miniUSB connectors are commonly used on smaller devices and are approximately 3x7mm. USB is designed to be used for communicating between a computer and peripheral devices. The USB 2.0 specifications, standardized by the USB Implementers Forum (USB-IF), rate data transfer speeds at up to 12Mbit/s (or 1.5MByte/s) [11].

Pins 2 and 3 on the ExtUSB connector was used for serial TTL (transistor-transistor logic) communication. Commands sent over the transmit and receive lines are sent at TTL-compatible logic levels. An input voltage between 0 and 0.8 V with respect to ground is considered as low, while a voltage between 2.2 and 5.0 V is considered high. Similarly, output voltages between 0 and 0.4 V are considered low, while voltages between 2.6 and 5.0 V are considered high [12].

The safety pilot receiver and transmitter use 2.4GHz DSM technology to communicate. DSM systems operate on the 2.4 to 2.4835 GHz radio frequency band. Channels are spaced 1MHz apart, allowing for a total of 79 channels. The operational range of DSM systems is 3000 ft. The DSM systems are subject to all regulations in Part 15 (Radio Frequency Devices) set by the FCC (Federal Communications Commission) [13].

4.2.2 Flight Software

The Android operating system has several guidelines for application design that promote code re-usability. The Android Java API has methods for creating standard foreground and background threads and an interface to an efficient, hardware-based IPC (Inter-Process Communication) [14]. Each software component in the Flying Android Project was imple-

mented using these interfaces, so they could be replaced or used in a different application with minimal effort.

4.3 Constraints, Alternatives, and Tradeoffs

4.3.1 Flight Hardware

The G1 was chosen because it has all of the hardware features the project requires and has an easy to use development environment. There are other smartphones that meet most the project specifications, but were not as attractive as the G1. The Apple iPhone 3GS has comparable hardware to the G1, but the development environment is limited mostly to Macs. The Palm Pre and the Nokia N900 both have capable hardware and software, but they lack a digital compass, which makes navigation much less accurate.

The serial connector could be replaced by a DTMF (Dual-tone multi-frequency) decoder on the interface board. Sending commands in the form of DTMF tones over the audio port of the smartphone removes the need to build the connector and repurpose the pins on the ExtUSB port. It would also enable the use of the application on other Android phones as the only G1 specific software is used for serial communication. However, this alternative is difficult to implement and more prone to error due to the nature of audio signal processing [15]. Using DTMF tones would also reduce the available bandwidth to send commands to the interface board. It is crucial that the commands from the smartphone are correctly transmitted to the interface board.

A multi-rotor aircraft could be used instead of the fixed-wing aircraft. Rotary aircraft are capable of hovering, which is useful for taking photos from a static position. The limited capabilities of the sensors on the smartphone would make controlling a rotary aircraft during hovering nearly impossible without external hardware [16].

A more common radio carrier frequency band used for RC aircraft is the 72 MHz band. These radio systems use frequency modulated (FM) carrier waves to communicate. 72 MHz radio systems are low cost; however, they are susceptible to radio interference, especially

ID	Task Name	Duration	Start	Finish
1	**Interface G1 with Airframe**	**32 days**	**Wed 9/16/09**	**Thu 10/29/09**
2	Pick Microcontroller, Airframe, and Other Parts	1 day	Wed 9/16/09	Wed 9/16/09
3	Order Parts	5 days	Thu 9/17/09	Wed 9/23/09
4	Build Serial Connector and Cable	14 days	Thu 9/24/09	Tue 10/13/09
5	Write Program to Send Serial TTL Commands from G1	2 days	Wed 9/16/09	Thu 9/17/09
6	Build Interface Board	14 days	Thu 9/24/09	Tue 10/13/09
7	Read Serial TTL Command from G1 on Interface Board	2 days	Wed 10/14/09	Thu 10/15/09
8	Convert Serial TTL Commands to PWM Signals	2 days	Wed 10/14/09	Thu 10/15/09
9	Send PWM Signals to Servos	2 days	Fri 10/16/09	Mon 10/19/09
10	Interface ESC with Interface Board	3 days	Wed 10/14/09	Fri 10/16/09
11	Control DC Brushless Motor	2 days	Mon 10/19/09	Tue 10/20/09
12	Test G1 Control of Airframe	7 days	Wed 10/21/09	Thu 10/29/09
13				
14	**Safety Pilot Unit**	**17 days**	**Wed 9/16/09**	**Thu 10/8/09**
15	Pick Receiver and Transmitter	3 days	Wed 9/16/09	Fri 9/18/09
16	Order Receiver and Transmitter	5 days	Mon 9/21/09	Fri 9/25/09
17	Interface Receiver with Interface Board	2 days	Mon 9/28/09	Tue 9/29/09
18	Program Receiver and Transmitter for Fail-Over	2 days	Mon 9/28/09	Tue 9/29/09
19	Program Interface Board for Fail-Over	2 days	Mon 9/28/09	Tue 9/29/09
20	Test Fail-Over	7 days	Wed 9/30/09	Thu 10/8/09
21				
22	**Flight Control and Phone Software**	**21 days**	**Wed 9/16/09**	**Wed 10/14/09**
23	Receive and Store GPS Coordinates	3 days	Wed 9/16/09	Fri 9/18/09
24	Handle Incoming Calls and Texts	7 days	Wed 9/16/09	Thu 9/24/09
25	Program Flight Controls	21 days	Wed 9/16/09	Wed 10/14/09
26	Store Images from Camera	3 days	Wed 9/16/09	Fri 9/18/09
27	Tag Images with GPS Coordinates	3 days	Mon 9/21/09	Wed 9/23/09
28	Generate Paths from Waypoints (GPS)	14 days	Wed 9/16/09	Mon 10/5/09
29	Take Photos at Waypoints	3 days	Tue 10/6/09	Thu 10/8/09
30				
31	**Testing**	**54 days**	**Thu 9/24/09**	**Tue 12/8/09**
32	Test Hardware without Airframe	7 days	Fri 10/30/09	Mon 11/9/09
33	Test Airframe without Hardware	7 days	Thu 9/24/09	Fri 10/2/09
34	Test Flight Controls	7 days	Tue 11/10/09	Wed 11/18/09
35	Test (unpowered) Flight with Hardware	7 days	Thu 11/19/09	Fri 11/27/09
36	Final Demonstration	7 days	Mon 11/30/09	Tue 12/8/09

Figure 6: Project schedule, tasks and milestones.

crosstalk. Crosstalk occurs when two transmitters are operating at the same frequency (or channel). When crosstalk occurs, the aircraft may exhibit erratic behavior. 2.4 GHz radio systems are more expensive; however, they experience very low interference from other radio signals[17]. Since radios will be used for the safety pilot system, the radio communications need to be shielded from interference as much as possible.

4.3.2 Flight Software

The complexity of the software is constrained by the processing power of the smartphone. The extent to which the software is limited by the processing power available on the smartphone has not been a problem so far, but it is likely that some algorithms which are commonly used in similar applications will be infeasible on the smartphone, particularly in the vision

and positioning systems.

The software for the plane is written using the Android Java API, but it is also possible to write low level software for Android in C using a similar API. Software written in C might execute faster than Java software; however, it will take longer to write the software in the Android C API, since it provides much less functionality than the Java API.

5 Schedule, Tasks and Milestones

The project was divided into the following four major milestones:

- G1 hardware interface to the aircraft
- Design of the safety pilot failover
- Implementation of the flight controls and smartphone software
- Hardware and software testing

The first three milestones address the various functionalities that were needed to implement a working product. These three milestone were worked on in parallel by the team. The last milestone consists of multiple different tests of the hardware and software. Figure 6 shows the milestones denoted by a bold font face. Below each milestone is a list of scheduled tasks for that milestone, including projected duration, start dates, and end dates.

The project started on September 16, 2009, and is estimated to be complete in May 2010. Refer to Appendix A for a Gantt Chart for the first half of this project.

6 Project Demonstration

During the week of November 30, 2009, we visited the Battle Lab at Ft. Benning (as seen in Fig. 7) to develop and evaluate our design. While testing at Ft. Benning, protection of the smartphone and flight hardware was given highest priority. First, the hardware was assembled and tested on the ground. Flight was then simulated by moving the aircraft via

(a) Andy waves to the camera.

(b) Preparing the aircraft.

(c) Programming in the cold.

Figure 7: Pictures at Ft. Benning

car and foot. Next, the plane was tested without flight control hardware by an experienced pilot.

Once the control hardware was integrated with the plane, short unpowered flights began. Next, the aircraft was flown under the watchful eye of a safety pilot to test the sensors on the smartphone. Now that all of these tests have been successfully completed, short autonomous flights will begin.

The final demonstration of the completed project will be a test flight of the airplane flying autonomously. The airplane will fly to a series of preprogrammed GPS waypoints as seen in Figure 8. Prior to the test, distinct visible targets will be placed at each waypoint so that the plane's location can be confirmed through the photographs. At each GPS waypoint it will take an aerial photograph of a target on the ground before proceeding to the next

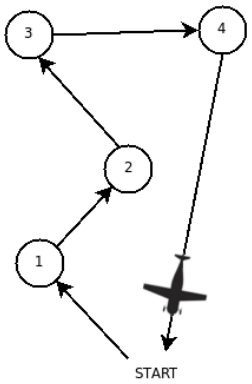


Figure 8: Waypoint navigation.

waypoint.

For this demonstration, the vehicle was controlled by the safety pilot until it was at altitude. In the future, the smartphone will be placed in control of the aircraft and begin navigating to waypoints. After capturing images at each location the plane will return to the launch site and land by gliding under control of the safety pilot. The images will then be examined to determine if the mission was a success.

7 Results and Acceptance Testing

The Flying Android Project demonstrated the capabilities of the G1 and the aircraft at Ft. Benning, GA on November 30 through December 2, 2009. We had planned to spend the first two days testing and do the final demonstration on the final day; however, rain prevented any flights on the last day. The UAV operated for the specified 15 minutes before it needed to land and recharge. During the test flights, the UAV recorded images and tagged them with geolocation data, which were retrieved from the SD card upon landing. We were also able to demonstrate the operation of the safety pilot mode. We were not able to demonstrate autonomous waypoint following and some other functionality, but most planned features were implemented.

(a) Demo of the launch platform.

(b) Shortly after takeoff.

(c) Photo from flight.

Figure 9: Pictures at Ft. Benning

On the first day of testing, we began by flying the aircraft with a phone surrogate, with

equivalent weight and position of the real phone. We did this to assure that the aircraft could fly without risk of damaging the phone. Our second flight we logged pictures to the SD card. These two flights were done without using the interface board; we used the stock flight hardware. On the second day, we used our final flight hardware, including the interface board. We demonstrated proper functioning of the safety pilot, as well as logging all of the sensor readings to the SD card during flight. Our final day of testing and demonstration was canceled due to rain. We did not have a chance to demostrate autonomous flight, but a preliminary design is implemented. An overview of the demonstration procedure is shown in Fig. 9.

8 Marketing and Cost Analysis

8.1 Marketing Analysis

Researchers, hobbyists, and government agencies alike share a desire for low cost, rapidly deployable unmanned aerial vehicles. Researchers desire such a platform to develop autonomy, clustering, and navigation algorithms. Hobbyists seek such a vehicle primarily to generate artistic aerial photographs.

Government agencies, by far the most lucrative customer of such a system, are interested in unmanned aerial vehicles for surveillance, targeting and reconnaissance applications. In the employ of a government agency, the aircraft could be programmed to patrol a border, track and follow a suspect or illuminate a target for munitions delivery. These applications are very desirable to many government agencies. Indeed, the sponsor of this project is a government agency.

8.2 Cost Analysis

Table 4 indicates costs of the project. As indicated in the table, the total cost of the project is $1,458. The cost to travel to Ft. Benning for a demonstration is included in the overall

budget.

The electrical system, which includes phones, batteries and PCBs, requires the most expensive hardware.

Table 4: Estimated system costs.

System	**Cost**
Mechanical	$285.54
Electrical	$547.67
Software	$25
Travel	$600
Total	$1458.21

The costs presented in Table 4 represent actual expenses to the sponsor. A more complete breakdown of system and equipment costs can be found in Table 5 in Appendix B.

9 Conclusions and Future Work

The Flying Android Group has begun preliminary prototype testing and started fielding our aircraft. Several piloted test flights have been completed with all flight hardware in place. Test flights have demonstrated camera functionality, GPS and sensor data logging, and mission endurance. Autonomous flight is the next major milestone for the project. Major aspects of autonomous flight, such as the communication between the G1 and the flight hardware and the PID controller, are complete and awaiting further testing and integration into a complete system. The design is still in the prototype phase and is actively being developed.

References

[1] K. Wong, "Survey of regional developments: civil applications," *Australian UAV Special Interest Group Coordinator, School of Aerospace, Mechanical and Mechatronic Engineering, University of Sydney*, 2001.

[2] R. Beard, D. Kingston, M. Quigley, D. Snyder, R. Christiansen, W. Johnson, T. McLain, and M. Goodrich, "Autonomous vehicle technologies for small fixed wing UAVs," *AIAA Journal of Aerospace Computing, Information, and Communication*, vol. 2, no. 1, pp. 92–108, 2005.

[3] D. Gormley, "Uavs and cruise missiles as possible terrorist weapons," *New Challenges in Missile Proliferation, Missile Defense, and Space Security, Monterey, Calif.: Monterey Institute of International Studies, Center for Nonproliferation Studies, Occasional Paper*, vol. 12, pp. 3–9, 2003.

[4] R. Kumar, "Tactical Reconnaissance: UAVS Versus Manned Aircraft," 1997.

[5] O. Alliance, "Google and the Open Handset Alliance announce Android open source availability," *Retrieved from http://www. openhandsetalliance. com/press_102108. html*, 2008.

[6] J. Eddy and P. Eddy, "Google on the Go: Using an Android-Powered Mobile Phone," *Que*, 2009.

[7] D. McLean, "Automatic Flight Control Systems." *PRENTICE HALL PRESS, 200 OLD TAPPAN ROAD, OLD TAPPAN, NJ 07675,(USA), 550*, 1975.

[8] P. Qualcomm, R. Memory, and E. Slot, "HTC TyTN II," *Stylus*, vol. 10, p. 2, 2008.

[9] B. Horn and B. Schunck, "Determining optical flow," *Computer vision*, vol. 17, pp. 185–203, 1981.

[10] R. Paul, "Why Google chose the Apache Software License over GPLv2 for Android," *Ars Technica,(November 06, 2007)*, 2007.

[11] U. Specification, "Revision 2.0," in *The USB Implementers Forum (USB-IF)*, 2000.

[12] D. Lancaster, *TTL cookbook*. Sams Indianapolis, IN, USA, 1974.

[13] D. Floam, G. Sugar, and N. Diener, "Monitoring for radio frequency activity violations in a licensed frequency band," Feb. 2 2006, US Patent App. 11/345,391.

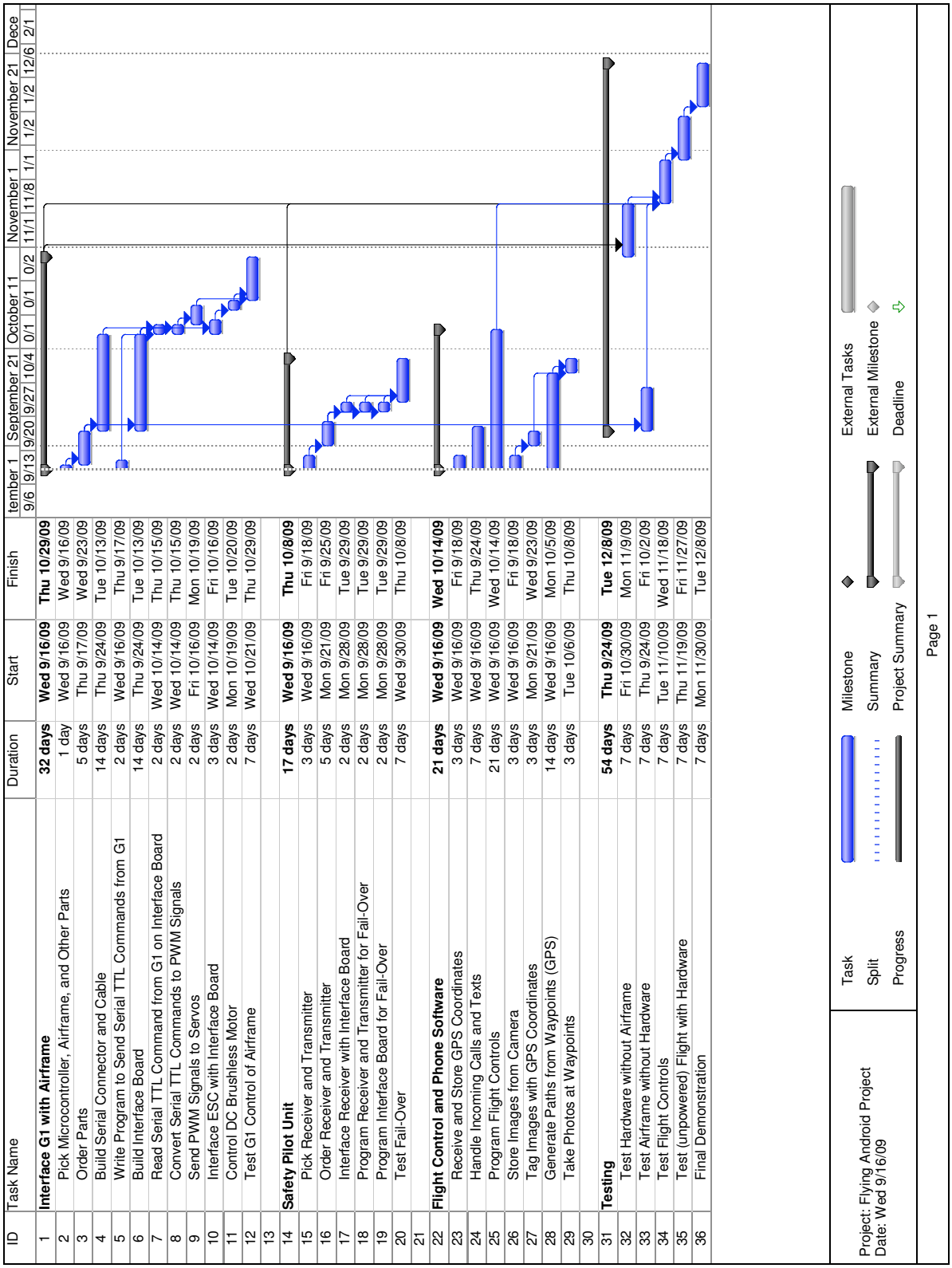
[14] R. Simmons and D. James, "Inter-Process Communication: A Reference Manual," *IPC Version*, vol. 6, 1997.

[15] J. Hayes and B. Tunzi, "DTMF Communication system," May 2 1978, US Patent 4,087,638.

[16] J. Morris, M. Van Nieuwstadt, and P. Bendotti, "Identification and control of a model helicopter in hover," in *American Control Conference, 1994*, vol. 2, 1994.

[17] J. Park, S. Park, D. Kim, P. Cho, and K. Cho, "Experiments on radio interference between wireless lan and other radio devices on a 2.4 ghz ism band," in *Vehicular Technology Conference, 2003. VTC 2003-Spring. The 57th IEEE Semiannual*, vol. 3, 2003.

A Gantt Chart



ID	Task Name	Duration	Start	Finish
1	**Interface G1 with Airframe**	**32 days**	**Wed 9/16/09**	**Thu 10/29/09**
2	Pick Microcontroller, Airframe, and Other Parts	1 day	Wed 9/16/09	Wed 9/16/09
3	Order Parts	5 days	Thu 9/17/09	Wed 9/23/09
4	Build Serial Connector and Cable	14 days	Thu 9/24/09	Tue 10/13/09
5	Write Program to Send Serial TTL Commands from G1	2 days	Wed 9/16/09	Thu 9/17/09
6	Build Interface Board	14 days	Thu 9/24/09	Tue 10/13/09
7	Read Serial TTL Command from G1 on Interface Board	2 days	Wed 10/14/09	Thu 10/15/09
8	Convert Serial TTL Commands to PWM Signals	2 days	Wed 10/14/09	Thu 10/15/09
9	Send PWM Signals to Servos	2 days	Fri 10/16/09	Mon 10/19/09
10	Interface ESC with Interface Board	3 days	Wed 10/14/09	Fri 10/16/09
11	Control DC Brushless Motor	2 days	Mon 10/19/09	Tue 10/20/09
12	Test G1 Control of Airframe	7 days	Wed 10/21/09	Thu 10/29/09
13				
14	**Safety Pilot Unit**	**17 days**	**Wed 9/16/09**	**Thu 10/8/09**
15	Pick Receiver and Transmitter	3 days	Wed 9/16/09	Fri 9/18/09
16	Order Receiver and Transmitter	5 days	Mon 9/21/09	Fri 9/25/09
17	Interface Receiver with Interface Board	2 days	Mon 9/28/09	Tue 9/29/09
18	Program Receiver and Transmitter for Fail-Over	2 days	Mon 9/28/09	Tue 9/29/09
19	Program Interface Board for Fail-Over	2 days	Mon 9/28/09	Tue 9/29/09
20	Test Fail-Over	7 days	Wed 9/30/09	Thu 10/8/09
21				
22	**Flight Control and Phone Software**	**21 days**	**Wed 9/16/09**	**Wed 10/14/09**
23	Receive and Store GPS Coordinates	3 days	Wed 9/16/09	Fri 9/18/09
24	Handle Incoming Calls and Texts	7 days	Wed 9/16/09	Thu 9/24/09
25	Program Flight Controls	21 days	Wed 9/16/09	Wed 10/14/09
26	Store Images from Camera	3 days	Wed 9/16/09	Fri 9/18/09
27	Tag Images with GPS Coordinates	3 days	Mon 9/21/09	Wed 9/23/09
28	Generate Paths from Waypoints (GPS)	14 days	Wed 9/16/09	Mon 10/5/09
29	Take Photos at Waypoints	3 days	Tue 10/6/09	Thu 10/8/09
30				
31	**Testing**	**54 days**	**Thu 9/24/09**	**Tue 12/8/09**
32	Test Hardware without Airframe	7 days	Fri 10/30/09	Mon 11/9/09
33	Test Airframe without Hardware	7 days	Thu 9/24/09	Fri 10/2/09
34	Test Flight Controls	7 days	Tue 11/10/09	Wed 11/18/09
35	Test (unpowered) Flight with Hardware	7 days	Thu 11/19/09	Fri 11/27/09
36	Final Demonstration	7 days	Mon 11/30/09	Tue 12/8/09

B Cost Development

The following costs represent actual expenses to the sponsor less labor costs.

Table 5: Estimated system costs.

Name	**Vendor PN**	**Cost Ea.**	**Qty**	**Total Cost**
Android Dev Phone	Android Dev Phone 1	$400.00	1	$400.00
Dev Membership	Dev Membership	$25.00	1	$25.00
Medium CA	LXPT39	$5.99	1	$5.99
CA Accelerator	LXPT42	$3.99	1	$3.99
Battery Connector	LXKX39	$3.09	8	$24.72
Multiplex Easy Star RTF	LXFXV0	$189.99	1	$189.99
Arduiplot	GPS-08785	$24.95	2	$49.90
G1 Connector	DEV-09078	$3.95	3	$11.85
Headers	PRT-00743	$3.95	2	$7.90
Pins	PRT-00117	$2.95	2	$5.90
Programmer	BOB-00718	$14.95	1	$14.95
F-F Servo Cable	ROB-09187	$2.95	10	$29.50
Micro serial cable	GPS-09123	$2.95	1	$2.95
Multiplex Power Pack	LXPVF5	$79.99	1	$79.99
Propellor	LXPVF3	$2.79	2	$5.58
Travel	na	$150.00	4	$600.00
			Total	$1,458.21