Captain Buzz: An All-Smartphone Autonomous Delta-Wing Drone

Ramsey M. Faragher
rmf25@cl.cam.ac.uk

Oliver R. A. Chick
oc243@cl.cam.ac.uk

Daniel T. Wagner
dtw30@cl.cam.ac.uk

Timothy Goh
tg319@cl.cam.ac.uk

James Snee
jas250@cl.cam.ac.uk

Brian Jones
bdj23@cl.cam.ac.uk

Computer Laboratory, University of Cambridge, UK

ABSTRACT

Fully autonomous hobbyist drones are typically controlled using bespoke microcontrollers, or general purpose low-level controllers such as the Arduino [1]. However, these devices only have limited compute power and sensing capabilities, and do not easily provide cellular connectivity options. We present *Captain Buzz*, an Android smartphone app capable of piloting a delta-wing glider autonomously. Captain Buzz can control servos directly via pulse width modulation signals transmitted over the smartphone audio port. Compared with traditional approaches to building an autopilot, Captain Buzz allows users to leverage existing Android libraries for flight attitude determination, provides innovative use-cases, allows users to reprogram their autopilot mid-flight for rapid prototyping, and reduces the cost of building drones.

Categories and Subject Descriptors

I.2.9 [**Computing methodologies**]: Artificial intelligence—*Robotics*[Autonomous vehicles; sensors]

General Terms

Design; Experimentation

Keywords

Smartphone; Fixed-Wing UAV;Autonomy; PID control; Pulse-Width Modulation

1. INTRODUCTION

In recent years there has been an explosion in growth of drones for researchers, commercial uses [2], and hobbyists [3]. These low-cost devices allow users with a small budget to buy and build drones that can be controlled autonomously.

Part of the success of drones has been their "hackability"—drones can be adapted for new tasks, and enhanced with additional components. However, current consumer drones are often built on bespoke micro-controllers that cannot be reprogrammed and hobbyist drones are built on quite simple micro-controllers such as the Arduino [4]. The programming interfaces, programming workflow, lack of inbuilt sensors, and limited compute power of micro-controllers all reduce their ease of adaptation to create innovative use cases or perform computationally-intensive tasks such as on-line visual processing.

We present a delta-wing drone that is controlled entirely by an Android app, including low-level operations such as commanding the servos. By controlling a drone using an on-board smartphone, a user can make use of extensive existing library support, deploy new versions of the software in flight, access a plethora of inbuilt sensors, and exploit multiple wireless connectivity options.

We use a delta-wing airframe, standard servos, and a Nexus 5 smartphone. The audio port of the smartphone is connected to the servos on the drone via an amplifier to increase the voltage provided by the headphone connection to the operating range of the servos. This amplifier is only needed because we use standard servos; had we sourced low voltage servos we would not have needed this extra amplifier component.

Controlling an autonomous drone using just an onboard smartphone offers the following advantages:

High extensibility. Captain Buzz is implemented entirely in Android. This allows developers to use existing Android capabilities, such as voice recognition, OpenCV, integrated communications services, etc. Innovative and complex solutions can be developed that are infeasible with typical drone controller technology.

Live coding of drones. By using Android's WiFi hotspot and the remote Android Debugging Bridge, Captain Buzz can be entirely reprogrammed mid-flight. This allows for rapid development of flight logic, dynamic changes to flight characteristics, and could even provide a human-in-the-loop supervised-machine-learning-based flight control system in the future.

Low cost. With the prevalence of smartphones, users no longer have to purchase a series of electronic components such as Arduino controllers, GPS units, and sensor boards in order to piece together a complete sensor, computation and communications package. While top-end smartphones can be relatively expensive devices

DroNet'15, May 18, 2015, Florence, Italy.

ACM 978-1-4503-3501-0/15/05 ...$15.00.
http://dx.doi.org/10.1145/2750675.2750682 .

to purchase, turnover of devices is very high with only two years being the average time between device upgrades for smartphone users in the UK and the US [5]. Previous-generation smartphones are therefore readily available at no cost to many hobbyists. Even in cases where smartphone screens are damaged, or battery life is reduced, they can still be used as an autopilot since the screen is not needed and the phone could be connected to the main power source used by the drone if needed. In many cases therefore, a new smartphone may be expensive but an old one may be free. It should be noted in any case that there are many smartphones that contain everything needed for this project that can be purchased for less than $100USD [6], whereas the total cost of the Arduino-based solution recommended for autopilot purposes is $160USD [7].

Connectivity and networking. Smartphones are incredibly well-connected devices, typically offering multiple wireless interface options including cellular and WiFi datalinks. The scope for ease of both long range communication and short range ad-hoc co-operation between multiple smartphone-based drones is an attractive reason to investigate the potential for all smartphone solutions.

In this paper we describe the challenges in dealing with increased lag from using a smartphone, rather than a simpler system with faster throughput such as the Arduino (§2); the design considerations of Captain Buzz's current airframe (§3), and an approach for controlling servos using a smartphone's audio output (§4.1).

The contributions of this paper are:

- We present the first all-Android-controlled delta-wing drone, showing that despite increased latencies compared with existing technology, a modern smartphone is capable of providing an autonomous flight controller.
- We describe a system design that, to the best of our knowledge, provides the first drone autopilot that can be entirely reprogrammed during flight.
- We present a design for a dual-purpose amplifier board that can increase the voltage provided by a smartphone audio socket in order to provide servo control, and also de-multiplex multiple Pulse-Width-Modulation (PWM) signals superimposed on each audio channel to boost the number of available control channels from the smartphone.

2. A SMARTPHONE AUTOPILOT

A smartphone contains everything that is needed to provide a fixed wing autopilot [8]. Location, ground speed and altitude can be measured using the GNSS receiver and MEMS barometer. Aircraft attitude can be measured using the MEMS accelerometer, gyroscope and magnetometer [9]. Radio communication is provided via cellular and Wi-Fi connectivity. Control output can be provided by the audio port, or a micro-USB On-The-Go (OTG) port. A smartphone contains an excess of processing power and memory for this purpose, and is powered by its own battery, providing a telemetry fail safe should the main battery of the airframe fail. The drawback of a smartphone based autopilot is the increased lag from sensing input to command output due to the lack of a low-latency interface for generating PWM output.

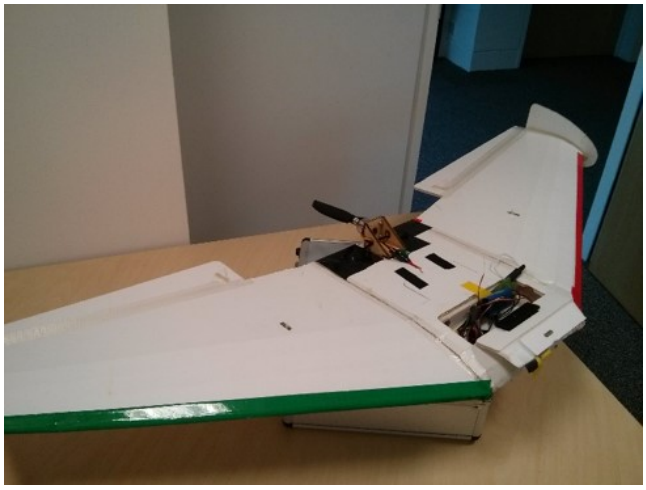

Figure 1: The delta-wing airframe with two-channel elevon control.

Modern smartphones that contain multicore CPUs and GPUs are more capable computing systems than the general purpose or dedicated microprocessors typically used for autopilot controllers. This means that the smartphone platform is excellent for developing further system capabilities involving real-time processing of visual data or processing data from sensors for other purposes than flight control.

3. AIRFRAME DESIGN

Captain Buzz uses a delta-wing airframe. This gives us three advantages over a conventional airframe: (i) Only two channels are required to control the elevons of a delta-wing glider. The airframe can therefore be controlled using the left and right audio channels from the smartphone audio output. (ii) The delta-wing provides the best lifting surface, and plenty of fuselage space, so is well suited to a project that needs to accommodate bulky payloads. (iii) A delta-wing design can also be rapidly constructed, and so new airframes can be constructed quickly to accommodate redesigns, damage, etc.

Figure 1 shows the airframe currently in use. It is constructed from foam board and folded into shape. The smartphone is mounted securely underneath the wing, as shown in Figure 2. Foam board has proven resilient to both poor weather, and abrupt landings. We use a custom-design for prototyping, but the Captain Buzz app can control any commercially-available delta-wing glider.

3.1 Driving Servos From a Smartphone Audio Port

A smartphone provides two possible electronic outputs to control servos: (i) The analog audio port (ii) A micro-USB On-The-Go (OTG) port. We use the audio port, as all smartphones provide such a feature, but not all smartphones provide OTG connectivity. Moreover, direct synthesis of PWM signals is not possible using the OTG USB interface, a separate synthesizer board would need to be developed.

Figure 2: The smartphone is mounted in a bay underneath the delta-wing.

The maximum output voltage across the terminals of a smartphone audio output connector is 1 volt. However, servo control messages need to span at least 3 volts to be registered. Rather than attempt to source or build low voltage servos we boost the volume of the output from the audio socket by adding an amplifier.

3.2 Multiplexing the Servo Channels

The amplifier board can be designed in such as way as to increase the number of control channels available to us. We present an audio amplifier capable of de-multiplexing four channels superimposed on the two channel audio output from a smartphone. This project does not use more than two control channels, but we recognize the great utility of this particular amplifier design to permit future expansion and further system improvements, such as autonomous motor control. Most fixed-wing aircraft require at least four channels for full control, and so this amplifier board provides this flexibility to move to a traditional aileron-elevator-rudder control system too. The availability of four channels also permits the development of an all-smartphone quadcopter. The design of our demultiplexing amplifier board is available at the Captain Buzz Google+ web page.[1]

Rather than simply outputting the PWM signals between 0 V and 1 V on each audio channel, we synthesise a channel between 0 V and −0.5 V, and an independent channel between 0 V and +0.5 V, and multiplex both onto the same audio channel (left or right). We then demultiplex these two sub-channels using op-amps on the amplifier board and amplify each of them to the desired 5 V before passing them on to the servos. This amplifier board therefore provides four independent channels for control from one smartphone audio port. In principle, this approach can be extended to provide more than four independent channels.

[1] `https://plus.google.com/118376347783210514294/posts`

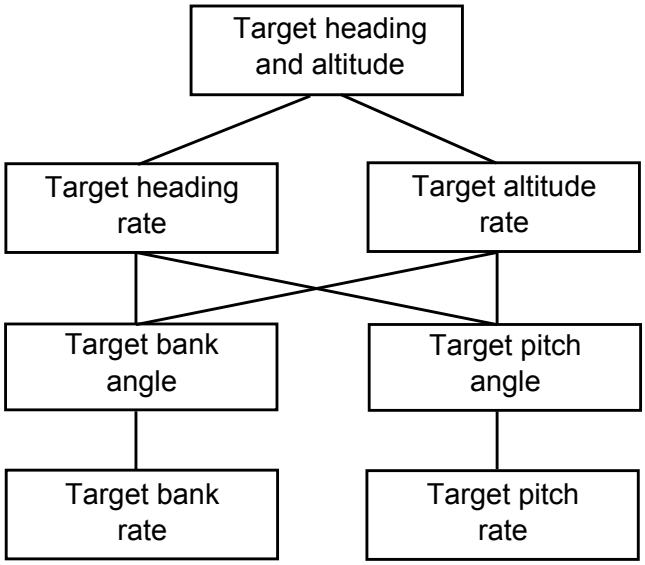

Figure 3: The PID control loops involved in this flight controller, and their interactions.

4. APP DESIGN

4.1 Generating Servo Control Signals

The Captain Buzz app exposes an interface that allows the autopilot to specify a desired deflection of each elevon. The app generates a corresponding PWM signal with a width between 1–2 ms to control the deflection of the servos.

Servo control channels are synthesized in this manner, and output on the smartphone left and right audio channels. If more than two control channels are needed then multiplexing can be employed as described in §3.2.

4.2 Smartphone Flight Controller

A benefit of using Android is the availability of both raw and processed sensor data. We compare our own sensor fusion algorithms to determine gyro-smoothed magnetic heading and to determine gyro-smoothed roll and pitch with the stock values returned directly from Android calls. Even though our code aims to take into account specific motion modelling for a fixed-wing glider, the Android-provided data is good enough to use in our flight controller, reducing the size of our codebase and lowering the software processing overhead of Captain Buzz.

We use standard Proportional-Integral-Differential (PID) feedback loops [10, 11] to provide flight control. Specifically, we have three nested PID loops, as shown in Figure 3: the inner-most loop controls the aircraft's bank rate; then a loop around this controls the target angle; and an outer loop controls the change in rate of heading. Furthermore, there is coupling between the target heading rate and the target pitch, since the delta-wing can make sharp turns (the current limit is 30°), during which there is a substantial reduction in lift provided by the wings. We compensate for this by increasing pitch.

Due to the number of nested PID loops involved in moving from simple roll and pitch control up to waypoint following, we need a way of rapidly searching the PID tuning space in flight. We achieve this by setting up the smartphone as a Wi-

Fi hotspot and pushing new parameters to the smartphone live in flight using the Android Debugging Bridge. We push a completely new version of the autopilot application to the smartphone in flight while the human operator is in control, in a few seconds. To the best of our knowledge this is the first time that a fully autonomous drone has been debugged and fully reprogrammed while in flight, and is a testament to the flexibility and rapid-prototyping possibilities of an all-smartphone autonomous platform for hobbyists.

5. CHARACTERISING LAG

Due to the use of the Android audio framework as an actuator control system, control lag terms are larger than those commonly found in bespoke flight control systems with embedded bare-metal application design, such as the Arduino-based systems. In this section we discuss how we minimize the latency exhibited with Captain Buzz, and how it is still capable of controlled flight, although with some small heading overshoot and damped oscillation remaining in our final system when waypoint following.

5.1 Experimental Setup

Finding a smartphone with the minimal processing lag from sensor input to servo control is an important step in the system design; the system lag from sensor input to control output dominates the flight characteristics of this system. We could choose between a Samsung S3, LG Nexus 4 and LG Nexus 5 smartphone as our autopilot controller. In order to test these three devices for the minimum lag, we built an Android app that measures the lag, using a method based on the *Larsen Test*. The app outputs a PWM test tone on the left and right audio channels when the accelerometers detect movement of the phone. The microphone input of an external recording device is connected to both a probe microphone, and to the audio port of the phone. A user applies a short impulse to the phone using a rod, such that the sound of this strike is picked up by the probe microphone. When the smartphone detects this impulse—through its accelerometers—it outputs a test tone, which is recorded by the external recording device, and flashes the notification LED. The notification LED has negligible latency (<1 ms), so allows us to decompose the end-to-end latency into input, and output latency. In this manner we perform a differential measurement to find the time between the impulse triggering the accelerometers and the resulting output command from the audio port. This experiment is analogous to a perturbation of the smartphone in flight triggering a response command to a servo controlling an elevon.

5.2 Reducing Latency

We employ multiple techniques to minimize the latency from an impulse being applied to the airframe, to a response being exhibited. We determine that use of the standard Android `AudioTrack` to synthesize pulse width modulation results in a round-trip latency of 180 ms. This lag manifests itself in flight with noticeable, but tolerable sluggishness. The principle component in this latency is Android's buffering of audio tracks. Each Android model specifies a minimum buffer size that applications must use to output audio. On the Nexus 5, and Nexus 4 this is 7680 bytes, and on the S3 it is 17640 bytes. 2-byte samples are read from the buffer at 48 kHz. Therefore this buffering accounts for 80 ms of the lag on the Nexus 5.

We therefore do not use the Android AudioTrack, but rather write a synthesizer—in C—that generates pulse-width modulated signals, and writes the generated signal directly to the soundcard. The synthesizer is linked into the Captain Buzz app using the *Java Native Interface*. The custom synthesizer allows us to create arbitrary-sized audio buffers, and enqueue them to play when the previous buffer finishes playing. We can therefore create buffers that are a single pulse-width period in size. To achieve minimal latency we reduce the pulse-width period from the standard 20 ms to 19.5 ms, to align our data with the audio card's internal frames, thereby reducing latency from realigning frames. Moreover, we make further software optimizations to minimize latency: high thread priorities, zero-copy data path, and minimal allocations.

5.3 Results

The latency of our most laggy device (Samsung S3) is 480 ms ± 65 ms before applying any of the optimizations discussed in §5.2. Following the optimizations, our fastest devices, the LG Nexus 4 and Nexus 5 exhibit latencies of only 115 ms ± 7 ms and 116 ms ± 10 ms respectively.

We use Android Traceview to determine that the processing time from reading the accelerometer, executing the PID loops, and outputting a signal, accounts for 1.1 ms.

When the autopilot attempts to maintain straight and level flight, the lag is insignificant, with Captain Buzz able to exhibit a meaningful response to small gusts of wind. However, when performing waypoint-following the lag is exacerbated by the multiple control loops; slight overshoot and damped oscillation on turning onto new headings are observed.

As efforts continue to reduce Android audio latency[2], the effects on our flight controller caused by lag will be reduced.

We could have built a custom version of Android that exposes a low-latency audio channel, but we feel this defeats the ease of adoption that we design into Captain Buzz. However, if future Android versions allow apps to output audio with low latency, the sluggish effects we have observed in our flight trials caused by lag would be reduced further.

6. FLIGHT TESTING

Flight tests took place throughout 2014. Figure 4 shows a screenshot of a video stream of one flight test. Further videos of Captain Buzz controlling the delta-wing airframe in static and in-flight demonstrations are available on YouTube.[3][4]

6.1 Experimental Setup

The Nexus 5 smartphone runs a Wi-Fi hotspot and is securely mounted in the payload bay of the delta-wing glider. A laptop connects to the the smartphone Wi-Fi hotspot and the Captain Buzz application is launched remotely via Android Debug Bridge (ADB). We conduct ground pre-flight tests for both the smartphone control system and for direct human control, exercising the control surfaces and verifying that the airframe is fit for flight. The aircraft is then manually flown to a safe height, and manoeuvred under direct manual control to further verify control surface responsive-

[2] `https://code.google.com/p/android/issues/detail?id=3434`

[3] `https://www.youtube.com/watch?v=990tmLXrxfk`

[4] `https://www.youtube.com/watch?v=-D5B6CrHizY`

Figure 4: Captain Buzz in flight with a stream of video (main), and telemetry. All flights follow CAA regulations; visual contact is maintained at all times, and the view from the human controller is recorded live with Google Glass (top-left).

ness. It is then levelled out, and configured for a gentle gliding descent. ADB initializes the the Captain Buzz application. When the control loop state is synchronized to the operating state, the human pilot hands elevon control over to the smartphone, using a *buddy box*.[5] The buddy box, mounted in the airframe, lets a human pass elevon control between themselves and the autopilot as desired.

Multiple tests have been carried out from straight and level flight (heading and pitch hold), through orbiting single waypoints and on to waypoint following.

The ADB link to the airframe while it is in flight permits quick, interactive tuning of major control loops by a ground observer without the traditional "land, recover, and reconfigure" cycle that is required with standard designs.

6.2 Safety Considerations

Take off and landing are performed by a human pilot and all flights are performed on private land far from roads or populated areas. The Civil Aviation Authority's *Information and guidance associated with the operation of Unmanned Aircraft Systems (UASs) and Unmanned Aerial Vehicles (UAVs)* [12] are followed at all times.

The master controller (a human using a standard radio transmitter) maintains throttle control at all times but can select when the smartphone app is in control of the elevon servos via the buddy box mounted in the airframe. The human operator can take back control of the elevons in an instant, and at any time. There is also a backup control link to the smartphone via a laptop, as discussed below.

Our airframe carries an audible low-battery indicator alarm that sounds as the main battery approaches low charge. Moreover, the ADB link provides us with an indication of the battery level of the smartphone. Throughout flight, Captain Buzz streams telemetry over the ADB link, which can be used to tune the autopilot, or identify failings of the airframe.

6.3 PID Tuning

Observation of the behaviour of the airframe allows the PID loops to be tuned rapidly through successive in flight tests by adjusting the parameters in flight using the Android Debug Bridge over a Wi-Fi link. Each of the desired autopilot capabilities require a new bank of PID control loops to be tuned before moving on to the next.

Our initial tests were of rate controllers on the roll and pitch axes. These controllers have a target law of zero roll/pitch rate, effectively maintaining airframe attitude. The controller implementation is a classic Proportional-Integral-Derivative (PID) loop system, with fused roll and pitch obtained from the smartphone inertial sensors, and actuator outputs on the audio socket driving the elevon servos. Gain tuning approximately follows Ziegler–Nichols [13].

The result of this experimentation is a flight controller capable of maintaining straight and level flight through wind buffeting or a poorly-trimmed airframe, maintaining safe bank angles in turns, and following waypoints.

The rate controller is extended to provide angle-of-roll and angle-of-pitch laws, along with a heading-hold controller. This is a higher-order controller specifying the desired roll and pitch angles, as well as desired magnetic heading. Although higher-order roll and pitch angles could be commanded as a simple extension of roll-rate and pitch-rate laws, the yaw controller has no direct equivalent: there is no direct yaw actuator. Taking a cue from manual flight operations, we add static "bank-and-yank" yaw excitation, where target roll/pitch control authority is sacrificed to induce a pitch-up and bank manoeuvre, initiating yaw.

For safety purposes, throttle control is maintained by the human operator at all times. To achieve waypoint following, the autopilot compares the current GPS position to the desired waypoint position, then intercept heading and pitch are calculated. The autopilot then sets its target heading to this intercept bearing. Once the current GPS position is within some threshold separation of the current waypoint position, such as ten metres, the current waypoint is replaced with the next in the list.

7. FUTURE DEVELOPMENTS

At this stage the smartphone autopilot only controls the elevons of a delta-wing platform, although we have two spare control channels through the development of our multiplexing audio amplifier. Motor control could therefore be also provided by the autopilot, or a traditional airframe comprising ailerons, rudder and elevator could be tested. The multiplexing audio amplifier also allows for the development of a quadcopter system based around an all-smartphone concept.

We have not yet incorporated the use of the smartphone camera to provide information for command and control such as horizon detection, optical flow, or visual SLAM [14]. This would be a useful direction for future work, in order to aid the attitude determination of the platform, especially during extended periods of manoeuvring. The ability to extend the Captain Buzz app to perform some of this processing without adding further components further demonstrates the benefits of an all smartphone drone.

Tuning the PID parameters through test flights is a time consuming aspect of this project. A future development could be to manually control the aircraft in flight via the wireless ADB (perhaps using an Android-based app running on a tablet or smartphone on the ground). Since the control commands would pass through the Captain Buzz app running on the smartphone mounted in the aircraft, the Captain Buzz app could use the human inputs as part of a

[5] `http://www.hobbyking.co.uk/hobbyking/store/%__20002__Wireless_Buddy_Box_System_4CH_Dual_RX_Controller_.html`

supervised training scheme to learn optimal values for various PID parameters. Such an in-flight supervised-machine-learning training scheme can only feasibly be performed by an all-smartphone drone, and would not be possible with traditional systems based on Arduinos or similar simple controllers.

8. RELATED WORK

In 2014 Intel used Android to control a quadcopter [15]. Intel use the smartphone bluetooth radio to communicate with a bespoke bluetooth servo controller board, and benefited from an board stabilization board integrated into the quadcopter. Similarly, Flone is a quadcopter carrying a smartphone that is controlled by a separate microcontroller board [16]. The andro-copter project uses an Android smartphone but flight control is again provided by an Arduino [17]. Romo is a small robotic tank that originally used the audio output from a smartphone to control its tracks but depends on the lightning connector of a modern iPhone [18].

9. CONCLUSIONS

We have developed and demonstrated the first all smartphone delta-wing autonomous drone controlled via servo commands provided by the audio output of the smartphone. We have determined that the lag associated with this method were not severe enough to prevent the development of a useable fixed wing autopilot. There were great benefits to this approach, such as live debugging and reprogramming in flight and ease of redundant communication mechanisms. The ability to run other apps on the smartphone controller in parallel with the autopilot open up the opportunity for a great range of science applications and other hobby projects. The multiplexing amplifier that we developed provides a means to increase the number of effective control channels available from the two-channel audio output, this paves the way for an all-android quadcopter as a future project.

Hybridization with an Arduino or similar microcontroller could provide reduced lag in the control loops and improve overall flight performance while in increasing the weight and complexity of the system, but this project has demonstrated that this hybridization is not necessary unless an agile airframe is required.

10. ACKNOWLEDGEMENTS

This work was supported by Google Inc., the Engineering and Physical Sciences Research Council; and CSR, Cambridge.

11. REFERENCES

[1] "Arduino." `https://www.arduino.cc`.

[2] "Hexacopter changes the way tv reporters work." `http://www.bbc.co.uk/news/business-24712136`. Accessed: 2015-03-28.

[3] "Build a drone." `http://www.buildadrone.co.uk`. Accessed: 2015-03-28.

[4] J. Amahah, "The design of an unmanned aerial vehicle based on the ardupilot," *Georgian Electronic Scientific Journal: Computer Science and Telecommunications*, no. 5(22), pp. 144–153, 2009.

[5] R. Entner, "International comparisons: The handset replacement cycle," *Recon Analytics*, 2011.

[6] "Low cost smartphones." `http://www.1mtb.com/top-9-best-unlocked-android-smartphone-mobile-under-100-dollar-usd`. Accessed: 2015-03-28.

[7] "Apm 2.5." `http://store.3drobotics.com/products/apm-2-dot-5-plus-assembled-set-top-entry`. Accessed: 2015-03-28.

[8] J. Tabarracci and P. Currier, "A blueprint for a fixed-wing autopilot on an android smartphone," in *Southeastcon, 2013 Proceedings of IEEE*, pp. 1–6, April 2013.

[9] H. Martin, P. Groves, M. Newman, and R. Faragher, "A new approach to better low-cost mems imu performance using sensor arrays," 2013.

[10] N. Minorsky., "Directional stability of automatically steered bodies," *Journal of the American Society for Naval Engineers*, vol. 34, no. 2, pp. 280–309, 1922.

[11] Sufendi, B. Trilaksono, S. Nasution, and E. Purwanto, "Design and implementation of hardware-in-the-loop-simulation for uav using pid control method," in *Instrumentation, Communications, Information Technology, and Biomedical Engineering (ICICI-BME), 2013 3rd International Conference on*, pp. 124–130, Nov 2013.

[12] "Information and guidance associated with the operation of unmanned aircaft systems (uass) and unmanned aerial vehicles (uavs)." `https://www.caa.co.uk/default.aspx?catid=1995`. Accessed: 2015-03-28.

[13] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," *trans. ASME*, vol. 64, no. 11, 1942.

[14] P. Williams and M. Crump, "All-source navigation for enhancing uav operations in gps-denied environments," in *Proceedings of the 28th International Congress of the Aeronautical Sciences*, 2012.

[15] "How to develop an intelligent autonomous drone using an android smartphone." `https://software.intel.com/en-us/articles/how-to-develop-an-intelligent-autonomous-drone-using-an-android-smartphone`. Accessed: 2015-04-02.

[16] "Is it a phone? is it a drone? no, it's a flone!." `http://www.theguardian.com/technology/blog/2014/jul/24/phone-drone-flone`. Accessed: 2015-04-02.

[17] "Andro-copter." `https://code.google.com/p/andro-copter`. Accessed: 2015-04-02.

[18] "Romo." `http://www.romotive.com`. Accessed: 2015-04-02.