2014 International Conference on Unmanned Aircraft Systems (ICUAS)
May 27-30, 2014. Orlando, FL, USA

Towards the development of 1-to-n human machine interfaces for unmanned aerial vehicles

James George Jenner, Luis Mejias Alvarez

***Abstract*—Multi-touch interfaces across a wide range of hardware platforms are becoming pervasive. This is due to the adoption of smart phones and tablets in both the consumer and corporate market place. This paper proposes a human-machine interface to interact with unmanned aerial systems based on the philosophy of multi-touch hardware-independent high-level interaction with multiple systems simultaneously. Our approach incorporates emerging development methods for multi-touch interfaces on mobile platforms. A framework is defined for supporting multiple protocols. An open source solution is presented that demonstrates: architecture supporting different communications hardware; an extensible approach for supporting multiple protocols; and the ability to monitor and interact with multiple UAVs from multiple clients simultaneously. Validation tests were conducted to assess the performance, scalability and impact on packet latency under different client configurations.**

***Keywords*—Touch, thin client, cross-platform, HTML5, UAV, ground control station, open source.**

I. Introduction

As technology matures, the capabilities of autonomous systems will increase, opening new avenues for a range of tasks not realisable before. Tasks where the risk to humans is the primary concern will now be possible by a swarm of low cost, dependable and efficient vehicles shifting the human operator from the field or front line to a more sheltered location. The current technology trends indicate that in the foreseeable future the number of autonomous vehicles will outnumber the number of available operators. The paradigm of 1-2 operator(s) for one platform will soon not be sustainable anymore. Therefore, the investigation of new approaches and technologies to allow one operator to interact and command multiples vehicles (1-to-n) is due for investigation.

One major impediment to achieve this goal is the hardware-centric software model. If multiple heterogeneous hardware interfaces are due to exist for every autonomous vehicle, then the need for developing software packages for every target hardware becomes a burden, and ultimately a hurdle for the development of advanced human-machine interfaces. The development of software interfaces that are transparent to the hardware and are able to be executed in commodity hardware can potentially unleash a new market of applications for unmanned aerial vehicles in an industry that is predicted to achieve multibillion figures [1].

Since the release of the iPhone in 2007 [2] and the iPad in 2010 [3, p. 300], there has been rapid growth in the ownership of multi-touch devices, for example smart phones and tablets. For the fourth quarter of 2011 there were 158 million smart phone shipments [4], 15.4 million iPad shipments and 10.5 million Android based tablet shipments [5]. Within the enterprise, take-up of tablets has been unusually rapid for a new technology. Apple reported that by the fourth quarter of 2010 80% of Fortune 100 companies had adopted the iPad in some form [6].

IDC forecasts for 2015 include shipments of 450 million PCs, 1,000 million smartphones and 180 million tablets [7]. While the Computer Security Update published a forecast that the multi-touch market will be worth $5.5 Billion by 2016 [8]. This forecast is based on products for multi-touch including smartphones, tablets, laptops, televisions/LCD, tables and floors.

The use of multi-touch in mobile devices can be evidenced by product launches such as the recent mobile world congress in Barcelona, Spain [11]. Users are developing memory models from their smartphones and tablets. For example, images and maps can be pinched and zoomed. As such new users to touch screens initially attempt to pinch and zoom [12]. “Nearly everyone is beginning to interact frequently with multi-touch technology, including panel operators, engineers, managers, executives and other decision-makers” [13]. This is leading to new devices such as padphones [14], personal computers [15], kiosks, point-of-purchase (POP), point-of-information (POI) and industrial machinery [16]. POI and POP can be used for ATMs [17], vending machines, merchandising, medical facilities (doctors office, clinics and hospitals [18]), building and industrial automation.

However, the heterogeneous nature of the hardware and proprietary operative systems within each device are limiting the full potential for the use of this technology in society. This is driven, in part, by vendors releasing products designed to keep consumers in their ecosystem [19].

Within the mobile and tablet space there are three main platforms: iOS by Apple; Android by Google; and Windows 8 by Microsoft. Tabletops are still immature in comparison to the mobile platforms. The main implementations are either a version of the Windows operating system or a Linux distribution. This raises software development issues when attempting to target multiple platforms.

The immediate question facing the developer is what approach to use in developing across disparate platforms, whereby approach is defined in terms of the toolchain and architecture. The key toolchains and architecture available to the developer can be summarised as follows:

1) Develop a native application for each platform with no code reuse between platforms.

978-1-4799-2376-2/14/$31.00 ©2014 IEEE

2) Utilise MVC (Model-View-Controller) pattern to abstract the model into libraries common to all platforms, while the view and controller are implemented via a native application.
3) Develop in HTML5 and JavaScript running in an embedded native application utilising an embedded web renderer to run the application.
4) Develop a single application utilising Qt, a cross-platform application and UI framework.
5) Utilise third party commercial products that support generation of applications for the targeted platform.
6) Utilise or develop middleware for each targeted platform to run non-platform specific code that executes in a runtime.
7) Develop a model with transformations to generate the required artefacts for the targeted platforms.

There are several considerations that influence a developer's choice for the best environment for developing cross-platform software. Specific needs for the developer will determine which considerations are of highest importance. Factors such as required skill sets, learning curve and environmental requirements all contribute to the cost of development.

The work presented here is expected to contribute to the field of UAS HMIs by providing a cross-platform approach that enables the use of multiple heterogeneous hardware devices simultaneously to interact with multiple UAV platforms. The main contribution of this approach are 1) transparency in the support of touch based and mouse based user interaction; 2) Flexibility in the support for multiple vehicles and clients; 3) Scalability in the number of clients and UAV platforms and 4) Robustness to packets loss and unauthorised use client's side.

This paper is organized as follows. Section II provides an overview of the current technologies applicable to multi-touch devices. Section III introduces the proposed approach and its main modules. Section IV presents the architecture implemented to validate our approach. Section V describes the main software modules and Section VI presents the testing seup and main results. Finally, Section VII presents the conclusions and future directions for this paper.

II. Related work

Initial research into ubiquitous computing included a focus on new types of interaction, of which touch based interaction received considerable attention [20], [21], [22]. Multi-touch interaction has been researched progressively from the mid 1980's [23], [24]. However commercial availability of multi-touch based systems only started appearing in the mid-2000s [25], followed by consumer devices in the late 2000s. This resulted in an increase of research into multi-touch software architecture and frameworks [26], [27], [28], [29].

As the range of devices increased, the question of how to develop and support the different platforms became a concern. While management of multi-touch interaction is

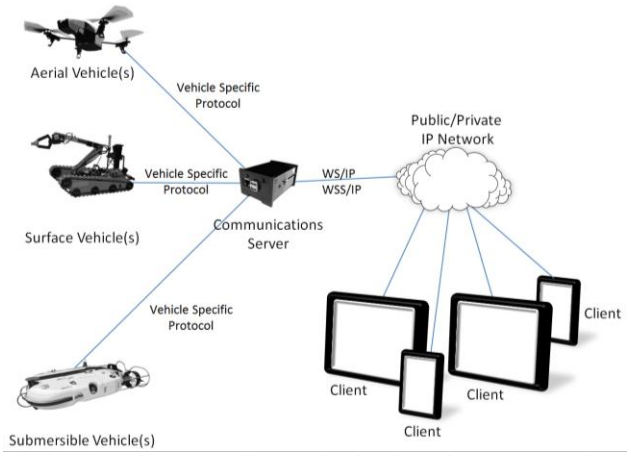

Fig. 1. Proposed system architecture for the physical system supporting multiple types of unmanned vehicles. Represented are UAVs (unmanned aerial vehicles, UGV (unmanned ground vehicles), and AUVs (unmanned underwater vehicles).

relatively new, supporting cross-platform development is not. Lessons learnt from cross-platform development can be applied to the issues of supporting multi-touch frameworks across multiple platforms. Thus an examination of current literature in the cross-platform development space is important.

Netscape utilised a cross-platform approach with their server-side software suite and client-side web browser in the 1990s [30]. Netscape's main goal was the minimisation of duplicate code. This was achieved via a common language that could be compiled on multiple platforms. However the major issue for Netscape was performance.

Other research into cross platform sensor networks found that stability, portability, flexibility and simplicity where obtained via software abstraction [31]. While there was an impact to performance with this approach, the impact was negligible.

An obvious option for abstraction is web client based development. Research into mobile software development and how web based development can be applied found that while the environment is immature, they provide versatility, economy, usefulness, less dependence on platforms, and the software development kits (SDKs) and are progressing towards full functionality in comparison to native development [32]. This further supports abstraction as a cross-platform approach, as the use of web based development allows a high level of abstraction with minimal coding for specific platforms.

HTML has matured enabling a cross-platform approach via HTML5, Cascading Style Sheets (CSS) and JavaScript. This is augmented by WebKit, an open source web browser engine and renderer. Apple's Safari and Google's Chrome web browser utilise WebKit [33]. It is available within Qt, iOS, and via APIs supporting multi-touch [33]. This positions HTML as an environment for multi-touch development across multiple platforms.

As HTML is not native it can be perceived that native code is better than HTML. As HTML abstracts the user

interface, issues with consistency for a given platform can be a concern [34]. Using a plastic interface may address consistency issues by allowing a native experience for the device and form factor [35], [36]. Charland et al. [37] addresses both performance and the interface when describing the trade-off between native software and web based software. While commenting that performance should be considered, they mention that tools like PhoneGap provide the ability to meet the divide in functionality.

HTML 5 garnered a lot of attention when LinkedIn and Facebook both decided that HTML 5 would be the solution for their multi-touch mobile solutions [38], [39]. The interest in HTML5 by the developer community lead to other companies testing the HTML 5 technology [40], [41]. This was slightly reversed by both Facebook and LinkedIn deciding that a native approach would be better for mobile apps [42], [43]. However various commentaries have noticed that the issues in these cases were not so much the HTML5 platform and technology, more an issue of how it was applied [44]. A general solution is to mix a native shell with a HTML5 core [45]. It should be noted that a key positive of HTML5 is the speed to develop and deploy, this is a large advantage for proof of concept software development [46].

III. Proposed Approach

A two-staged approach was decided upon for software development. The first stage was to focus on the multi-touch user interface while supporting multiple platforms. The second stage focused on communications from multiple clients with multiple vehicles simultaneously.

A. User Interface

The user interface is an enabler in widening the audience for interaction with unmanned vehicles. Intuitive user interfaces can be leveraged in supporting the casual user. This assists in reducing training and support requirements for deployed solutions. As such the user interface leveraged modern concepts for user interaction found in popular social software solutions (e.g. Google+, Facebook, etc). It should be noted that the focus was not purely on touch based interaction, but also mouse based interaction.

B. Multiple Vehicles

The ability to interact or monitor multiple vehicles was seen as a key enabler in extending functionality of unmanned vehicles. This would enable focusing on roles of users beyond the engineer and the pilot. The role of the 'consumer' would be a purveyor of the payload data in real-time scenarios. For example: fire crews monitoring hot spot information from multiple UAVs that are patrolling forestry areas.

A HTML5 solution was selected as the platform for the client. This was based on the ease of rapid development, portability between platforms and the skill sets available for software development. As HTML5 was selected for the client, WebSocket over IP was selected as the communication protocol between the client and the server.

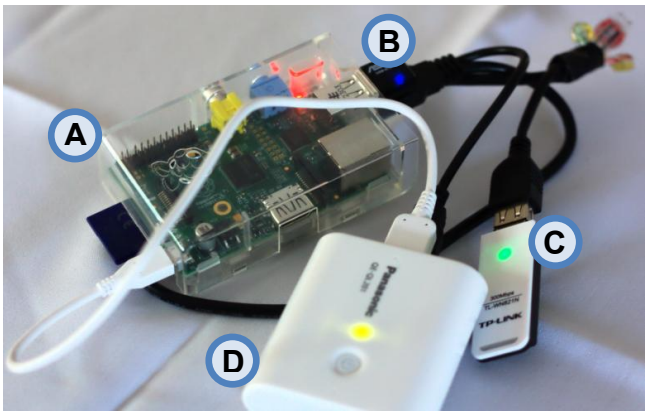

Fig. 2. Example portable server hardware based on a single board computer. The components are a Raspberry Pi (A), short range WiFi card (B) , long range WiFi card (C) and a battery pack (D).

Node.js was selected as the server platform. This was chosen due to the availability of a mature WebSocket library, the ability to share JavaScript between client and server, the low resource requirements to run Node.js and the portability of Node.js between platforms.

The intent was that the above elements would enable a thin client solution leveraging a lightweight server for dissemination of telemetry while allowing interaction.

IV. Architecture

The architecture is composed of several key components: clients, communications server, and vehicles (see Fig. 1). Clients are portable touch based devices to allow easy interaction by users with data from the vehicles. The communications server can be considered the orchestrator of data between vehicles and clients, enabling interaction between the two sides. The vehicles are autonomous vehicles with the ability to communicate telemetry and/or payload information.

Connecting the clients and communication server is an IP based network, which can be either public or private. The protocol between the clients and communications server is the WebSocket protocol. WebSocket has been developed with HTML 5 and has W3C candidate recommendation [47].

Communication between a vehicle and the communication server is via the physical layer(s) required by the vehicle. The condition being that a) the server can interact with hardware components required to facilitate communication and b) the protocol used by the vehicle is implemented by the communications server software.

The aim of the architecture is to facilitate interaction by multiple clients with multiple autonomous vehicles. The following requirements where defined to support the aim of the architecture:

1) All software components are to be cross platform capable;
2) Client software will support touch based and mouse based user interaction;

3) No dependency on specific hardware;
4) Software is to support portable hardware; and
5) Architecture is to support multiple vehicles and clients.

A. Vehicles

Integration of vehicle support within the solution was approached in two stages. Stage I was to test the software components of the architecture and demonstrate the viability of the various architectural decisions. Stage II was to apply the results of stage I, providing a working solution using multiple UAVs.

The Parrot AR Drone vehicle (version 1) was selected for Stage I. The Parrot doesn't include a full autopilot and has no GPS. The main feature of the Parrot is that it can hold station. This is achieved via ultrasonic altimeter and downward facing camera using optic-flow. Communications is achieved using TCP/IP over 802.11b/g. The parrot's WiFi must be operated in host mode but only allows a single connection.

Two Parrot AR Drone frames were selected for Stage II. They were fitted with PX4 autopilots (with IO module for the Parrot), 3DR GPS and 433/915 MHz radios. The PX4 autopilot includes a 3D gyroscope, 3D accelerometer, 3D magnetometer and barometer. The firmware for the PX4 autopilot utilises the MAVLink protocol. This provides a low bandwidth communication link via the 433/915 MHz radios for up to one kilometre. The radios come in pairs, one for the ground station and one for the vehicle.

B. Communications Server

The server provides IP based connectivity for clients via the WebSocket protocol. A custom JSON (Java Script Object Notation) messaging scheme is used for communications with clients. The software for the server was implemented in JavaScript running on Node.js. The advantage of JavaScript is that the same language can be used for the client and server. This allows creation of common libraries. Thus development effort can be reduced while increasing quality.

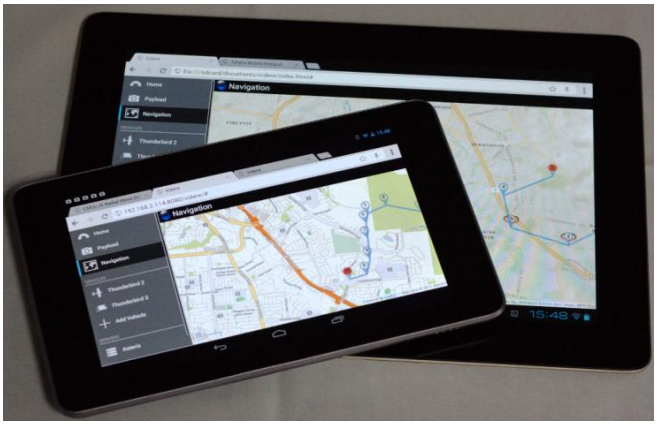

Fig. 3. A Nexus 7 and ASUS Transformer Prime running the client in the chrome browser while connected to the same server. The user interface displayed has the left side pane activated, which is overlaying the navigation area.

An advantage of Node.js is the low footprint (memory and CPU), allowing the use of low powered hardware. This was demonstrated via a single board computer (see Fig. 2). This uses a Raspberry Pi running an ARM 700 MHz processor with a Linux operating system. The advantages of such an approach are its size, power requirements, interfaces and price. For Stage I, the server was comprised of a Raspberry Pi (A), a USB WiFi card for client connectivity (B), a USB WiFi card for vehicle connectivity (C), and a battery pack (D). For Stage II the USB WiFi card (C) was replaced with a 3DR 915 MHz radio. The Stage II approach was also tested on a laptop where the server and client ran on the same device.

C. Clients

Clients facilitate interaction with the vehicles via the communication server. Any device that can run the client software can act as a client.

The software for the client was developed using HTML5, CSS3 and JavaScript. This allowed flexibility to support any platform that contains a browser that supports HTML5. Special focus was given to support WebKit. WebKit is the rendering engine used by Chrome, Safari, Opera and Midori web browsers (amongst others). Utilities like Phone Gap use WebKit to generate a native application for various mobile platforms. This includes iPhone/iPad, Android and Windows devices.

Stage I and Stage II were demonstrated on a Nexus 7 and ASUS Transformer Prime (see Fig. 3). Stage II was demonstrated on a Nexus 7, ASUS Transformer Prime, an iMac and a laptop running Ubuntu.

V. Software

The software can be divided into three components: client, server, and common (see Fig. 4). The client and server components run on their respective hardware platforms. The common components are available on both the client and server. The JavaScript language is utilised for all components. HTML5 and CSS3 are used for rendering on the client, while the server is run in a headless environment (i.e. no graphical user interface). All software components are freely available on GIT via the URL https://github.com/jamesjenner/videre.

A. Common

The common component's main focus is to remove duplicate code between the client and server components. This is implemented by defining sets of classes and associated methods. The main use is primarily communications.

All content for communication is formatted as JSON, thus leveraging JavaScript's ability for manipulating JSON data. Elements within the messaging purposely utilise English to enable simple debugging.

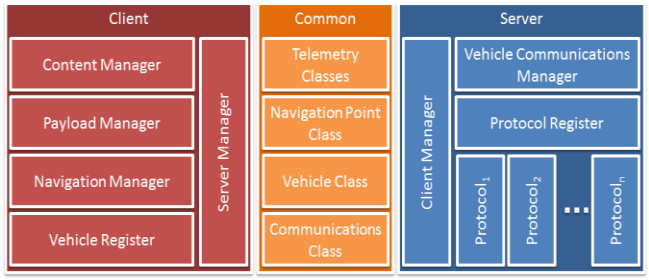

Fig. 4. Software architecture for client, server and common components.

B. Client

The client utilises HTML5, CSS3 and JavaScript. The client is comprised of several modules, while leveraging the common component for communication with the server.

The content manager orchestrates UI flow and the common components of the UI. The UI is grouped by areas and tabs. A left side pane manages navigation while a right side pane provides context aware options. The content manager keeps track of this information and orchestrates what is displayed based on interaction with the navigation pane and area tabs.

The payload manager tracks payload information. This includes map rendering and interaction with the payload UIs.

The navigation manager handles user interaction with a map for navigation purposes. This includes waypoint handling, waypoint state, planned path and actual path.

The vehicle register monitors vehicles and any associated telemetry. The vehicles register is utilised by the other managers to determine vehicle state. For example, the content manager uses the vehicle register to show telemetry.

The server manager controls all communication with the server. The server inspects messages, notifying other components as required.

C. Server

The server runs on Node.js. Node.js uses an event driven non-blocking I/O model which is programmable via the JavaScript language. This makes Node.js a natural fit as the communications coordinator between clients and vehicles.

The implementation of the server is divided into components with specific areas of responsibility. This is enforced via the inheritance and event model used by Node.js. This allows encapsulation of each component, increasing reliability by limiting impact of any changes.

A key aspect of the server is that it utilises a command line approach for configuration. Command line options cover all aspects of the server, which can be saved into a configuration file for ease of use. The options that are available are listed via the --help and -h command line options.

The client manager is responsible for all communication with clients. The client manager manages connection attempts, authentication, inbound and outbound messages. The client manager provides an authentication process which can support three modes.

The supported communication modes are unsecured, mixed mode and secured only. The use of unsecured is for situations where security isn't a concern, for example, when using a private network. Mixed mode communications transmits telemetry via an unsecured connection, while client based vehicle commands are transmitted via a secure connection. This is intended to reduce the load on the server and any connected clients, as encrypting packets create extra load on the server [48]. The secure mode requires all transmissions via a secure connection.

In addition to the communications channels, the client manager applies authentication based on user id and password. The password is confirmed via a persisted hash determined via bcrypt [49] with a multi round salt. This provides a secure method of authentication that is future proof for the foreseeable future [49].

The vehicle communications manager (VCM) is focused on coordinating communication with vehicles. The VCM was designed around the behaviour of the AR Parrot Drone V1 during Stage I. The key presumption was that a single communication device would connect to a single vehicle. During Stage II the VCM was refactored to support multiple vehicles per communication channel. This was required to support the MAVLink protocol and to test multiple vehicles simultaneously.

With Stage II, specific channels of communication can be configured at runtime. Each communication channel is exclusively associated with a protocol. A channel can be any physical connection that the server hardware supports. It should be noted that the network port is considered a single channel for network connections.

For Stage II, the vehicle communications manager connects to the defined channels when the server is started. The associated protocol is used to process messages that are received or sent via the communications channel. The protocol determines if a single or multiple vehicles are attached to the communication channel. This allows auto-discovery of vehicles via the communication channel, as determined via the associated protocol.

Internally the VCM allocates UUIDs (unique universal identifiers) to each vehicle registered via a communication channel. All information broadcast to clients regarding vehicles identifies the vehicle via the UUID. This is to avoid conflicts between different protocols, however can be used to group vehicle ids using the same protocol. For example: the MAVLink protocol uses a numeric identifier with a maximum value of 255. The use of UUID for vehicle identification avoids the potential for conflicts by clients [50].

D. User Interface

The client uses an interface designed for both touch and mouse based interaction. This allows for usage on touch devices as well as personal computer hardware.

The software consists of several components: action bar, main pane and side panes (left and right). The action bar is

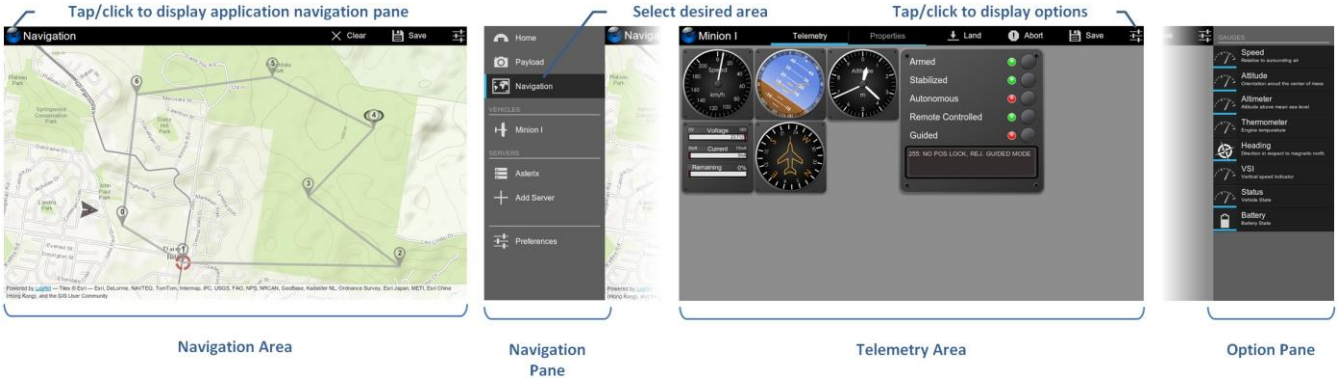

Fig. 5. The key areas of the client application are the navigation and telemetry areas. The user selects different areas via the navigation pane. The navigation pane will list all servers and all remote vehicles for connected servers. For each area there is an options pane on the right hand side, this is context sensitive. For the navigation area, a list of available map styles is presented, while the options for the telemetry area allow the selection of instrumentation panels. All instrumentation panels within the telemetry area can be repositioned via touch or mouse. The state panel allows a request to toggle a specific state, as well as displaying the state and any informational messages. A right side pane can be accessed via the icon on the right to hide/display available panels.

context sensitive, based on the active area. The action bar provides feedback, navigation and supports actions. Feedback is provided via the title (displaying the name of the current section) and icons that represent state. Navigation is performed via the tabs and controls to open side panes.

The left side pane provides multiple purposes. The primary purpose is the ability for the user to navigate between areas. However it is also used to determine the available vehicles, the defined servers and visual feedback as to which area is currently selected.

The right side pane is context sensitive controls for the active area. For example, when a map is displayed it provides a list of the map styles while indicating the active style. For telemetry it lists all the available gauges while indicating the active gauges. The main pane is the focus for the user and allows interaction for the area that is currently active.

The areas within the software consist of the following:

1) Home – the default area on startup;
2) Payload – payload related information;
3) Navigation – map based user interface for viewing and editing navigation details;
4) Vehicles – a separate area for each vehicle that has been registered with the connected servers, providing configuration, monitoring of telemetry and vehicle interaction;
5) Servers – a tool to define or edit servers; and
6) Preferences – definition of preferences that affect the various areas within the software.

Selection of an area is available via the left hand navigation pane, displayed via the activation button on the action bar. All areas are accessible via the navigation pane.

The navigation and vehicle areas are the primary focus of the software. The navigation area views all current vehicles while the vehicle area is specific to each vehicle. Stage I focused on the user interaction framework and a vehicle user interface. Stage I Parrot interaction requirements resulted in the vehicle area divided into the telemetry tab and remote control tab.

The remote control tab provided a user interface for controlling the Parrot AR Drone. As such the controls were designed around the commands required to fly a Parrot. The key commands were take-off, land, abort, vertical controls, and horizontal controls. The vertical and horizontal controls utilised a touch UI concept to emulate joysticks [51]. The remote controls were deemed to be temporary, as the goal of the software is to interact with autonomous vehicles.

The remote control tab was required for controlling the Parrot. This was because the standard Parrot is not an autonomous vehicle, thus direct control by a user was

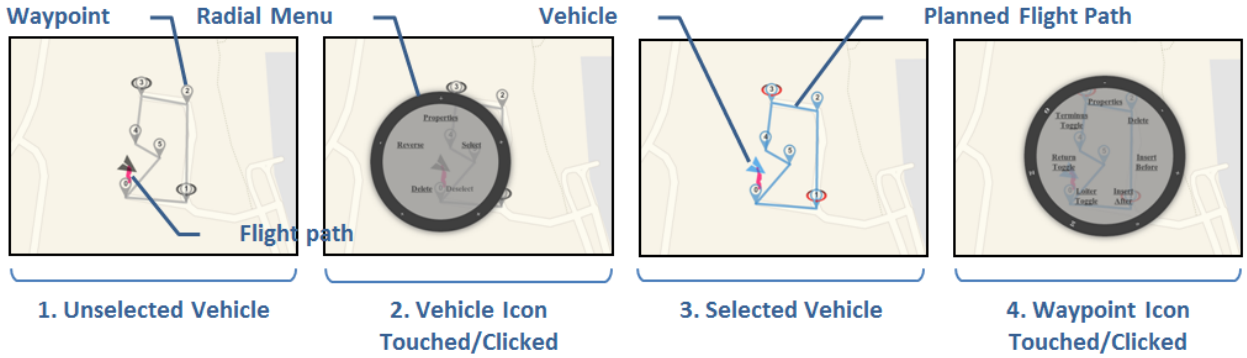

Fig. 6. Demonstration of how navigation interaction is performed. All navigation interaction is performed via radial menus. The user can perform various tasks such as editing the planned navigation path or tell the vehicle to target a specific waypoint. The waypoint reflects whether it is a terminating waypoint, includes a patrol command and if the vehicle is currently targeting the waypoint.

required.

The telemetry interface (see Fig. 5) was designed to support an unknown number of gauges and indicators. As such the telemetry user interface utilises the option pane to show and hide gauges. When a gauge is shown on the main window, the user can drag the gauges around. The default configuration is the traditional T layout for aircraft.

For Stage II, the vehicle area was refactored to support fully autonomous vehicles. The telemetry tab was retained, the remote control tab was removed and a properties tab was added. Two new gauges were introduced, the voltage gauge and a state control panel. The state control provides support for viewing the state of key attributes of the remote vehicle, while also allowing a request to change each state.

The navigation pane provides the following information for each vehicle available via the server:

1) Location;
2) Targeted waypoint;
3) Planned flight path;
4) Waypoints and associated information;
5) Actual flight path;

The navigation pane provides the ability to interact with the vehicle planned flight path via radial menus (see Fig. 6). By default a vehicle is not selected (as there may be more than one vehicle). To select a vehicle the user can touch/click on a waypoint or the vehicle. Then a radial menu will appear with the options, including an option to select the vehicle. Choosing the select option will change the path and vehicle from grey to blue. This indicates that the planned flight path is active and can be edited.

VI. Implementation and Testing

We implemented our multi-touch framework using the architecture depicted in Fig. 7. Four heterogeneous clients and two UAVs were used in our experiments. The main aim of this experiment was to validate our proposed approach by i) Implementing our framework in four different operating systems and ii) Evaluating the performance in terms of packet's traffic.

A. Hardware Architecture

Fig. 8. UAVs used during the testing procedure.

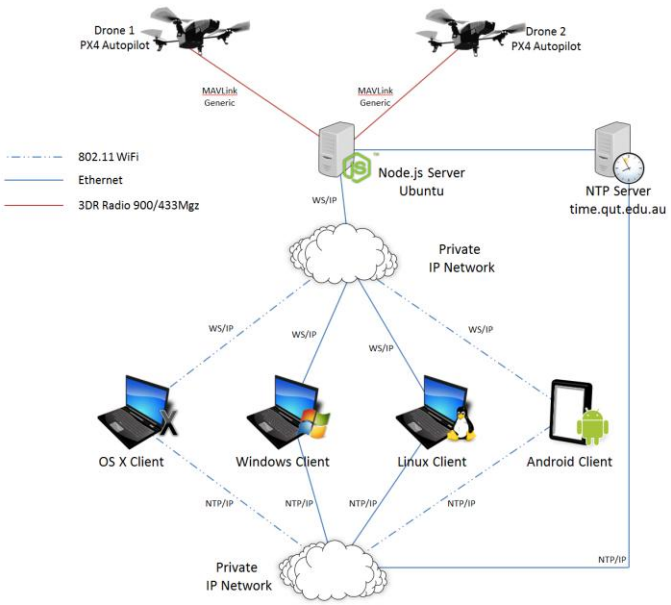

Fig. 7. Overview of the physical architecture used during testing.

The hardware architecture is a fair representation of what could be implemented in real systems. A mixture of different wireless/wired communication technologies, mobile devices and operative systems are used. A main server running Node.js is used as a gateway to communicate with each UAV. Communications between the server and each UAV is handled by two separate radio modems at 900Mhz and 433Mhz, respectively. Communication client-server is made via 802.11 (OSX and Android clients) and Ethernet (windows and Linux clients). Each drone is based on a Parrot v1 airframe in which control boards are removed and replaced with PX4 boards (a PX4IOAR Quad Carrier adapter board [52] and PX4FMU [53]). The PX4FMU is a flight management unit that provides an autopilot, utilising the MAVLink protocol for telemetry and interaction. The PX4IOAR is an adapter board which provides the interface between the PX4FMU and the Parrot AR drone motors.

Finally, time synchronisation was handled using the same

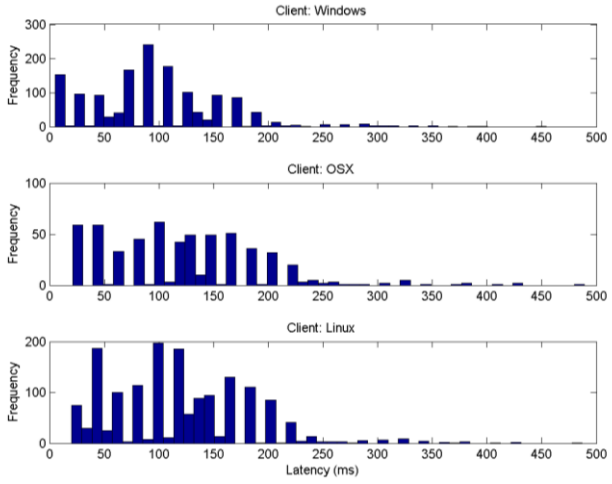

Fig. 9. Test 1. Time histogram of telemetry packets for each client.

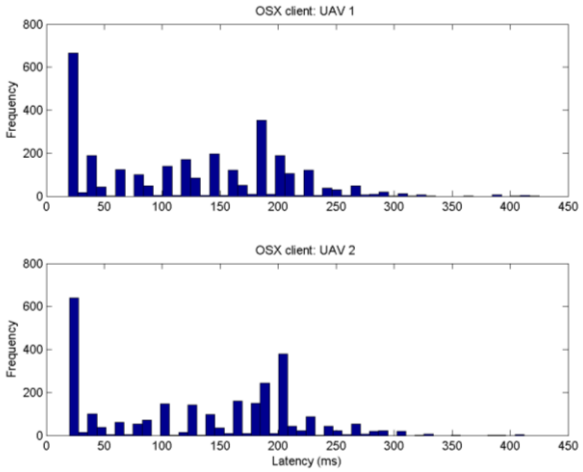

Fig. 10. Test 2. Packet latency times for UAVs 1 and 2 using client 2 (OSX).

network timeserver for all clients (except for nexus 7). In this way, differences in time were kept at a minimum.

B. Testing Procedure

Two experiments were conducted to evaluate the average latency and packet traffic load when clients and UAVs are increasingly added to the network. The first experiment was performed using one UAV and multiple clients (four). The second was conducted using two UAVs, and the same number of clients. Each client was added to the network at several time intervals, i.e. t = (0m, 2m, 4m, 8m) with corresponding clients (1, 2, 3, 4) respectively. Each packet is time-stamped at the server and client respectively. In addition to vehicle state each packet contains a packet number and an UAV ID. Using time, packet number and UAV ID we were able to match each packet sent from the server in each client. Therefore, the latency was computed using the time differences between same packets sent and received. Overall, the procedure for testing consisted in the following steps:

1) Synchronising all hardware using the same local time server.

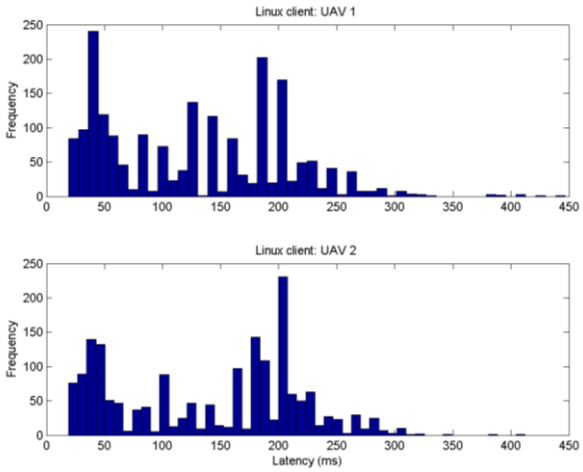

Fig. 12. Test 2. Packet latency times for UAVs 1 and 2 using client 3 (Linux).

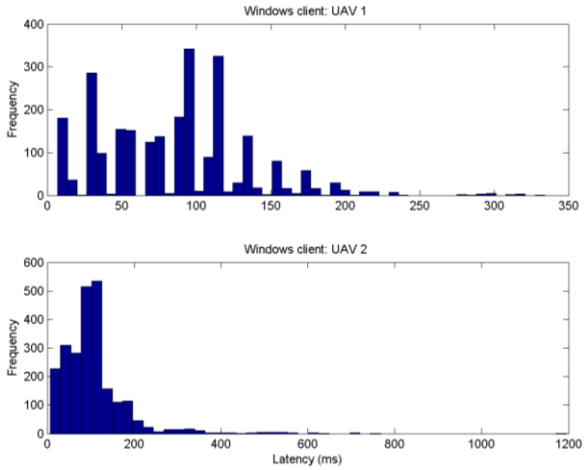

Fig. 11. Test 2. Packet latency times for UAVs 1 and 2 using client 1 (Windows).

2) Server was initialised and confirmed as running.
3) One (or both) UAV(s) was/were started.
4) The server automatically establishes a connection to the UAV(s).
5) Once server-UAV connection is established, each client would connect to the server at various time intervals.

Table 1 depicts the connection schedule for each client. Information was saved in log files at each end. The exception was the mobile device running Android due to an inability to synchronise with the time server and capture logs on the device.

Time (m)	Windows	OSX	Linux	Nexus 7
0	x			
2	x	x		
4	x	x	x	
6	x	x	x	x

Table 1 Client-server connection setup during the experiments. Clients were connected at time 0m, 2m, 4m and 6m, respectively.

C. Discussion

Timing results are depicted in Table 2. Examining the latency results we observe no major impact in time delay when the number of client connections increase. We note an increasing trend in latency times for clients 2 and 3 when compared with client 1. This conclusion is however within a limited scope given the IP network used in the experiments was shared with many other clients not directly involved in this test (not dedicated infrastructure).

An important property to highlight from our architecture is the ability to maintain reliable packet delivery rates (0% packet loss). This was achieved by a considered design choice of using a connection-oriented protocol. This is an important feature to ensure dependable UAV command and control using these types of frameworks.

For each client and the two experiments conducted, the latency times are shown in Figures 9-12. Figure 9 shows latency time when one UAV and three clients are using our

framework (Test 1). Whereas, Figures 10-12 show the latency times for each client receiving packets from both UAVs (Test 2).

		Test 1		
	UAV	**Windows**	**OSX**	**Linux**
Average latency (ms)	1	96.2081	123.1932	119.1819
Std (ms)	1	60.9680	72.3133	64.5322
		Test 2		
Average latency (ms)	1	85.6735	121.9851	127.8314
Std (ms)	1	49.5973	80.4291	77.1226
Average latency (ms)	2	106.4599	131.6521	140.1723
Std (ms)	2	80.0339	82.5465	78.6067

Table 2 Testing Results. Average latency and standard deviation in milliseconds. Client 1 (Windows), Client 2 (OSX), Client 3 (Linux) and Client 4 (Nexus 7).

The number of total packets received by each client was different due to the different duration in connection time. For instance, Figure 12 shows the packet histogram for client 1 for a connection time equal to 8m. This is evidence by large latencies spread in the x axis. In addition, an interesting behaviour is observed in Figures 9-12 where the distribution of latencies seems to obey a multimodal distribution. There appears to be a large number of packets arriving at ~0-50ms and at ~150-200ms.

Whilst these observed latencies might not seem suitable for real-time low-level command and control, the observed values are adequate for high-level interaction between vehicles and operators. For example, functions such as telemetry and vehicle state, waypoint and flight plan update, payload information, etc; are achievable within the current performance of the framework. During these experiments the focus was placed on the software side of the approach. However, by adopting a more thorough design of the network infrastructure the bandwidth and latency will greatly improve. The adoption hardware independent approaches such as the one proposed in this paper will be important for increasing performance of UAS operations, and also for reducing manpower needed for UAS missions.

VII. Conclusion and Future Work

In this paper we presented an architecture based on the multi-touch hardware-independent high-level interaction philosophy. This approach could be considered as critical enabler to the advancement of the civil UAS industry which is predicted to reach multi billion dollars figures. We investigated the impact and possible solutions for cross platform development to support multi touch hardware. Our architecture was tested demonstrating no noticeable increase in latency with the addition of multiple clients when using multiple UAVs.

However, scalability for many concurrent clients, many UAVs and feasibility of using multiple servers would benefit for further investigation. This would assist in further validating the architecture presented in this paper.

Another area for consideration is the support for payloads. Payloads by their nature may require specialist hardware, incorporate proprietary protocols and create problems with scalability. Further investigation should be considered into the potential for a standard protocol, while determining the feasibility of a plug 'n' play approach to payloads.

Credit

Windows is a registered trademark of Microsoft Corporation in the United States and other counties [54]. The Android robot is reproduced or modified from work created and shared by Google and used according to the terms described in the Creative Commons 3.0 Attribution License [55]. Fig. 1. Proposed system architecture incorporates derived works for an aerial vehicle[56], surface vehicle [57], submersible vehicle [58] and a communications server [59].

Fig. 7. Overview of the physical architecture used during testing incorporates derived works for the drone [56] and clock [60]. This diagram also utilises the following third party images: server [61], node.js logo [62], laptop [63], server [61], penguin tux [64], Microsoft Windows™ logo [54] and Android™ logo [55].

References

[1] "Teal Group Predicts Worldwide UAV Market Will Total $89 Billion in Its 2013 UAV Market Profile and Forecast," *Teal Group Corporation*, 17-Jun-2013. [Online]. Available: http://tealgroup.com/index.php/about-teal-group-corporation/press-releases/94-2013-uav-press-release. [Accessed: 03-Nov-2013].

[2] "Apple to release iPhone on June 29," *The Associated Press. Charleston Daily Mail*, 04-Jun-2007.

[3] Y. I. Kane, "Apple Sells 300,000 iPads on First Day - WSJ.com," *The Wall Street Journal*, 06-Apr-2010. [Online]. Available: http://online.wsj.com/article/SB10001424052702304017404575165621713345324.html. [Accessed: 24-Mar-2012].

[4] "Smart phones overtake client PCs in 2011 | Canalys," 03-Feb-2012. [Online]. Available: http://www.canalys.com/newsroom/smart-phones-overtake-client-pcs-2011. [Accessed: 25-Mar-2012].

[5] D. Sarno, "Android tablets gain market share versus iPad," *Buffalo News*, Buffalo, N.Y., 29-Jan-2012.

[6] "Worldwide Enterprise Tablet Market Forecast," *PR Newswire*, 30-Jan-2012.

[7] "It's an era of multiple devices per person," *Financial Express*, New Delhi, India, 27-Sep-2012.

[8] "Multi-Touch Market to be Worth $5.5 Billion by 2016," *Computer Security Update*, vol. 13, no. 1, Jan. 2012.

[9] I. Leung, "Touch-screens to be integrated into greater variety of electronics," *Electronics News*, May 2013.

[10] "Research and Markets: Global and Chinese Touch Screen (Panel) Industry Report, 2012-2013," *Business Wire*, New York, United States, 16-Apr-2013.

[11] C. Devi, "New line-up of Android devices," *New Straits Times*, Kuala Lumpur, Malaysia, 03-Mar-2013.

[12] "The Smarter Glass Group Brings Multi-Touch Solutions to OEMs," *Entertainment Close - Up*, Jun. 2012.

[13] “Leveraging AIS New Multi Touch Screen Technology Enhancements in Improving Building Automation Efficiency,” *Business Wire*, New York, United States, 13-Mar-2013.
[14] E. Zeman, “Mobile World Congress 2013: 9 Hot Gadgets,” *Informationweek - Online*, Feb. 2013.
[15] “Lenovo Unleashes New Windows 8 Touch Devices,” *Business Wire*, New York, United States, 06-Jan-2013.
[16] “AIS Introduces Multi-Touch Screen Monitors with ‘Gesture Experience’ for Slew of Next Generation Self-Service, Interactive Kiosks,” *Business Wire*, New York, United States, 03-Sep-2013.
[17] “Diebold Showcases an Expanded ATM Experience at Consumer Electronics Show,” *Professional Services Close - Up*, Jan. 2013.
[18] “American Industrial Systems Inc.; AIS New All-In-One Multi Touch Screen Panel PCs Give you the Capability to Easily Deploy Healthcare Self-Service Kiosks,” *Computer Weekly News*, May 2013.
[19] “Apple’s new iOS 7 makes bold statement,” *South Asian Media Net*, Lahore, Malaysia, 21-Sep-2013.
[20] M. Weiser, “The computer for the 21st century,” *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 3, no. 3, pp. 3–11, Jul. 1999.
[21] H. Ishii and B. Ullmer, “Tangible bits: towards seamless interfaces between people, bits and atoms,” in *Proceedings of the SIGCHI conference on Human factors in computing systems*, New York, NY, USA, 1997, pp. 234–241.
[22] M. Weiser, “Some computer science issues in ubiquitous computing,” *Commun. ACM*, vol. 36, no. 7, pp. 75–84, Jul. 1993.
[23] S. Lee, W. Buxton, and K. C. Smith, “A multi-touch three dimensional touch-sensitive tablet,” *SIGCHI Bull.*, vol. 16, no. 4, pp. 21–25, Apr. 1985.
[24] P. Wellner, “Interacting with paper on the DigitalDesk,” *Commun. ACM*, vol. 36, no. 7, pp. 87–96, Jul. 1993.
[25] K. Ryall, C. Forlines, C. Shen, M. R. Morris, and K. Everitt, “Experiences with and Observations of Direct-Touch Tabletops,” in *Proceedings of the First IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, Washington, DC, USA, 2006, pp. 89–96.
[26] F. Echtler and G. Klinker, “A multitouch software architecture,” in *Proceedings of the 5th Nordic conference on Human-computer interaction: building bridges*, New York, NY, USA, 2008, pp. 463–466.
[27] M. T. Gorg, M. Cebulla, and S. R. Garzon, “A Framework for Abstract Representation and Recognition of Gestures in Multi-touch Applications,” in *Advances in Computer-Human Interactions, 2010. ACHI ’10. Third International Conference on*, 2010, pp. 143 –147.
[28] T. E. Hansen, J. P. Hourcade, M. Virbel, S. Patali, and T. Serra, “PyMT: a post-WIMP multi-touch user interface toolkit,” in *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, New York, NY, USA, 2009, pp. 17–24.
[29] M. R. Morris, A. Huang, A. Paepcke, and T. Winograd, “Cooperative gestures: multi-user gestural interactions for co-located groupware,” in *Proceedings of the SIGCHI conference on Human Factors in computing systems*, New York, NY, USA, 2006, pp. 1201–1210.
[30] M. A. Cusumano and D. B. Yoffie, “What Netscape learned from cross-platform software development,” *Commun. ACM*, vol. 42, no. 10, pp. 72–78, Oct. 1999.
[31] M. Brzozowski, H. Salomon, K. Piotrowski, and P. Langendoerfer, “Cross-platform protocol development for sensor networks: lessons learned,” in *Proceedings of the 2nd Workshop on Software Engineering for Sensor Network Applications*, New York, NY, USA, 2011, pp. 7–12.
[32] L. Corral, A. Sillitti, G. Succi, A. Garibbo, and P. Ramella, “Evolution of Mobile Software Development from Platform-Specific to Web-Based Multiplatform Paradigm,” in *Proceedings of the 10th SIGPLAN symposium on New ideas, new paradigms, and reflections on programming and software*, New York, NY, USA, 2011, pp. 181–183.
[33] C. Björkskog, G. Jacucci, B. Lorentin, and L. Gamberini, “Mobile implementation of a web 3D carousel with touch input,” in *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, New York, NY, USA, 2009, pp. 48:1–48:4.
[34] K. Richter, J. Nichols, K. Gajos, and A. Seffah, “The many faces of consistency in cross-platform design,” in *CHI ’06 extended abstracts on Human factors in computing systems*, New York, NY, USA, 2006, pp. 1639–1642.
[35] J. E. Giron, S. Mendoza, and C. Torres-Huitzil, “Mechanism for dynamic deployment of plastic mobile cross-platform user interfaces,” in *Electrical Engineering Computing Science and Automatic Control (CCE), 2011 8th International Conference on*, 2011, pp. 1 –5.
[36] D. Raneburger, “Interactive model driven graphical user interface generation,” in *Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems*, New York, NY, USA, 2010, pp. 321–324.
[37] A. Charland and B. LeRoux, “Mobile Application Development: Web vs. Native,” *Queue*, vol. 9, no. 4, pp. 20:20–20:28, Apr. 2011.
[38] R. Shields, “Facebook readies iPad and HTML5 apps,” *New Media Age (Online)*, 17-Jun-2011.
[39] “LinkedIn Mobile for Android, iOS Gets HTML5 Treatment,” 17-Aug-2011. [Online]. Available: http://www.eweek.com/c/a/Mobile-and-Wireless/LinkedIn-Mobile-for-Android-iOS-Gets-HTML5-Treatment-708934. [Accessed: 19-Oct-2013].
[40] D. Clark, “HTML5: Technology Changing the Web,” *Wall Street Journal, Europe*, Brussels, United States, p. 22, 14-Nov-2011.
[41] C. Kanaracus, “Salesforce.com Bets Big on HTML5 Adoption: An upcoming HTML5-based application will deliver its software to any touch-enabled device,” *CIO*, vol. 25, no. 1, p. n/a, Oct. 2011.
[42] J. O’Dell, “Why LinkedIn dumped HTML5 & went native for its mobile apps,” *VentureBeat*, 17-Apr-2013. [Online]. Available: http://venturebeat.com/2013/04/17/linkedin-mobile-web-breakup/. [Accessed: 07-Oct-2013].

[43] C. Warren, “Zuckerberg’s Biggest Mistake? ‘Betting on HTML5’,” *Mashable*, 12-Sep-2012. [Online]. Available: http://mashable.com/2012/09/11/html5-biggest-mistake/. [Accessed: 07-Oct-2013].
[44] M. Asay, “HTML5 isn’t Facebook’s ‘biggest mistake’,” 14-Sep-2012. [Online]. Available: http://www.theregister.co.uk/2012/09/14/facebook_html_5_vs_native_apps/. [Accessed: 07-Oct-2013].
[45] A. Williams, “Debunking Some Myths About Native And HTML5 Hybrid Apps,” *TechCrunch*, 07-Jul-2013. [Online]. Available: http://techcrunch.com/2013/07/07/debunking-some-myths-about-native-and-html5-hybrid-apps/. [Accessed: 07-Oct-2013].
[46] E. Chickowski, “HTML5 Mobile Development: 7 Good Ideas (and 3 Bad Ones),” *Network Computing - Online*, Feb. 2013.
[47] “The WebSocket API.” [Online]. Available: http://www.w3.org/TR/websockets/. [Accessed: 07-Sep-2013].
[48] K. Kant, R. Iyer, and P. Mohapatra, “Architectural impact of secure socket layer on Internet servers,” in *2012 IEEE 30th International Conference on Computer Design (ICCD)*, 2012, pp. 27–34.
[49] N. Provos and D. Mazieres, “A Future-Adaptable Password Scheme,” in *USENIX Annual Technical Conference*, Monterey, California, USA, 1999.
[50] P. J. Leach, M. Mealling, and R. Salz, “A Universally Unique IDentifier (UUID) URN Namespace.” [Online]. Available: http://tools.ietf.org/html/rfc4122. [Accessed: 08-Sep-2013].
[51] S. Lee-Delisle, “JSTouchController,” *GitHub*. [Online]. Available: https://github.com/sebleedelisle/JSTouchController. [Accessed: 08-Sep-2013].
[52] “PX4IOAR Quad Carrier - PX4 Autopilot Platform.” [Online]. Available: https://pixhawk.ethz.ch/px4/modules/px4ioar. [Accessed: 19-Oct-2013].
[53] “PX4FMU Autopilot / Flight Management Unit - PX4 Autopilot Platform.” [Online]. Available: https://pixhawk.ethz.ch/px4/modules/px4fmu. [Accessed: 19-Oct-2013].
[54] “Windows Trademark Guidelines.” [Online]. Available: http://www.microsoft.com/en-us/legal/intellectualproperty/Trademarks/Usage/Windows.aspx. [Accessed: 13-Oct-2013].
[55] “Brand Guidelines | Android Developers.” [Online]. Available: http://developer.android.com/distribute/googleplay/promote/brand.html. [Accessed: 13-Oct-2013].
[56] N. Halftermeyer, *English: Parrot AR.Drone 2.0 flying*. 2012.
[57] U. S. N. photo by J. S. J. Ebalo, *English: Camp Lemonier, Djibouti (Oct. 26, 2004) - A robot used by Navy Explosive Ordnance Disposal Mobile Unit (EODMU) 8, Det. 18, carries an explosives deterrent device in its claw during an exercise. EODMU-8, Det. 18, based at Naval Air Station Sigonella, Sicily, deployed in October to Camp Lemonier, Djibouti. The primary goal of the unit is to ensure camp safety by responding to bomb threats and monitoring and securing the base areas including the main camp, flight line and supply area. U.S. Navy photo by Journalist Seaman Joe Ebalo (RELEASED)*. 2004.
[58] MKFI, *English: Bofors Double Eagle Mk II remotely operated underwater vehicle used for naval mine clearing by Finnish Navy. Photographed in Turku Forum Marinum during the Finnish Navy 2011 anniversary.* 2011.
[59] C. Berscheidt, *Built-to-Spec Raspberry Pi Case*. 2012.
[60] rihard, “Clipart - Clock + Calendar,” 31-Jan-2007. [Online]. Available: http://openclipart.org/detail/2998/clock-+-calendar-by-rihard-2998. [Accessed: 13-Oct-2013].
[61] mimooh, “Clipart - Server,” 12-Aug-2011. [Online]. Available: http://openclipart.org/detail/155101/server-by-saisyukusanagi. [Accessed: 13-Oct-2013].
[62] “node.js.” [Online]. Available: http://nodejs.org/logos/. [Accessed: 13-Oct-2013].
[63] metalmarious, “Clipart - Laptop,” *Open Clip Art*, 27-Aug-2007. [Online]. Available: http://openclipart.org/detail/5012/laptop-by-metalmarious. [Accessed: 13-Oct-2013].
[64] L. E., Simon Budig, Anja Gerwinski, *Penguin Tux, the Linux Mascot*. 2012.