Autonomous Airborne Electronic Warfare System

Charlie Shoup , Matt Blessing, Ben Broadhead and Enes Oguz
Computer Engineering, University of Utah

***Abstract*—Currently implemented battlefield electronic warfare systems (EWS) are expensive, heavy, and complex, primarily because they are designed to operate on a target from a safe distance. Our project addresses these concerns by miniaturizing the EWS and delivering it directly to the target area via a low cost commercial unmanned aerial vehicle (UAV) platform. This project delivers a prototype with the ability to fly autonomously and have an electronic warfare package attached to it. The electronic warfare package will am radio frequency communication across many different wavelengths including those used by cellphones.**

***Index Terms*—EWS, UAV, RF, GPS, FC, PWM, DARPA, MALD, Rpi3, BEC, GCS**

I. Introduction

Autonomous air launched drones with electronic warfare capabilities are currently in development for the United States Air Force and Navy. Currently these drones are expensive and require specialized deployment methods (large aircraft, large vehicles). This research proves that mobile robotic electronic warfare systems can be inexpensive, robust and powerful and can operate more efficiently by exploiting vulnerabilities in communications protocol management broadcasts. Experiments were performed with fixed wing and multi-rotor drone airframes, various software defined radios and an autopilot system based on a Raspberry Pi 3 with a Navio2 guidance shield running various versions of Ardupilot. Radio frequency denial of service focused on the most common communication protocols used in the developed world, namely the 802.11 family and Universal Mobile Telecommunications System (UMTS) based cellular broadcasts.

The current existing technology is only useful in a limited set of situations. Our prototype broadens the range of applications for this technology allowing for rapid deployment from a wide range of platforms (infantry, armored vehicles, boats and close air support aircraft). Our design provides all military units access to a cost-effective autonomous electronic warfare system.

II. Background

Advances in motor and energy storage technology coupled with inexpensive micro controllers has lead to explosive growth in the drone market. Although these devices are now nearly ubiquitous, most advanced autonomous drone technology is currently utilized for military and police work. However, many of these organization rely on systems which are overly expensive and robust given the current state of the base technology.

Despite the existing gap between available technology and currently deployed systems, various organizations have been exploring possible low cost, expendable drone systems. Using this existing body of research, primarily from Raytheon, National Oceanic and Atmospheric Administration (NOAA), BAE Systems and Defense Advanced Research Projects Agency (DARPA), we are convinced that lightweight consumer drone airframes and motors can provide enough lift force for enough time to serve as a reliable platform for an EWS.

A. Past UAV Projects

There are many uses for aerial UAV's, some of which are surveillance, security, and entertainment. There have been numerous UAV projects that have been done in the past that have influenced our group to develop this autonomous flying drone.

1) Landmine Detection Project: Although conceptual, a flying UAV that can detect landmines can prove to be quite useful [1]. This project was done to reduce the amount of human losses due to removing active landmines from the field. It was found to be quite effective at locating and marking active landmines buried in the ground. It did this by using a micro controller, metal detector, and a GPS chip as stated in [1]. The UAV was not designed to be autonomous, thus needing a human to be behind the remote control for the UAV. The body of the UAV was a quadcopter which used four motors to spin the rotors for movement and control.

2) Confetti Drone Project: This project was based off the entertainment factor of drones. The article details different forms of entertainment that can be provided using drones. Through commercials, TV shows, movies, concerts, to even personal use such as walking a dog. [2] The main study done in this project was to make a drone that could shoot confetti effectively for entertainment purposes. To achieve this the Confetti Drone engineers used a quad-copter with a mechanism to hold the confetti canister on top of the drone and dispense it accordingly. The quadcopter and confetti canister were both controlled by the user of the drone at time of flight.

The main difference between these projects stated above is that they all have different purposes. Some similarities between them is that they are all controlled by the user at time of operation. They both use some type of micro controller to perform certain tasks. Plus, both projects are using a quad-copter drone body.

The purpose of the drone is to carry a EWS module to a specified destination. Using a micro controller to perform tasks as well, while planning on configuring it in such a way to achieve autonomous flight. Whereas the projects mentioned

above all aimed to have a user control the drone throughout its flight time.

B. DARPA and Raytheon Current Tech

DARPA has some of its research going into UAV's. They have chosen various private companies to lead in this type of research. A company that is one of the leads is called Raytheon. Raytheon is currently developing Miniature Airborne Launched Decoys (MALD)'s. Currently announced to the public, Raytheon has told us that there are numerous versions of the MALD's.

For this project, the group really kept an eye on two of the models as shown in Fig. 1. They are called Type 1 and Type 2. Type 1 is just a decoy that projects a fake aircraft targeted to antiaircraft systems, Type 2 is a MALD-J and is very close to the Type 1 MALD but it is equipped with a jamming module. Most of our motivation has come from this model of the MALD. For more information on the MALD's various models and operation dates check out [3].

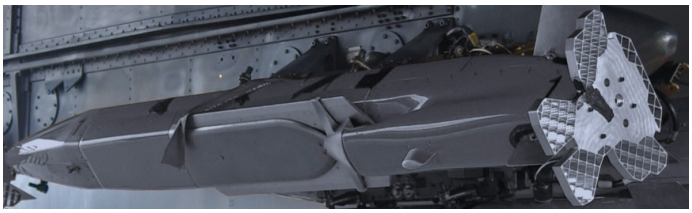

Figure 1. A picture of the MALD from Raytheon [4]

The referenced article mentions an "experimental variant" of the MALD which is called the MALD-V. It was from this variant of the MALD research that the group stumbled across the MALD-J variant which is represented as Type 2 in our case. The most useful information the group have been able to gather about the Type 2 is from a short little video Raytheon has posted on their website along with a really short description of the MALD-J variant. The posted video demonstrates what Raytheon wants to achieve with this system which can be seen from [4].

From the short video by Raytheon the idea is to jam or "distract" radar systems so friendly aircraft can safely complete their mission. The project aims to try and achieve relative stats close to the MALD-J system.

III. Project Overview

In order to facilitate development of our airborne electronic warfare system, the group has broken the project down into the three major subsystems detailed below. This is of course a brief overview of what we want to achieve throughout this project. There will be more details in the later sections of this paper.

A. Airframe

The most basic component of any airborne system is the airframe. This component serves to encapsulate the vehicle's systems and provide sufficient lifting force to keep the vehicle and payload airborne while being robust and cost effective in relation to the vehicle's planned role. For this particular project, the group needed an airframe which was inexpensive with a relatively high payload capacity for its size.

To facilitate testing of our autopilot, the group initially chose the FT Spear flying wing for our prototype. The Spear is unique in that it is constructed entirely of foam core board and balsa wood, making it extremely cost effective and easily field repairable. For more information on this airframe, see [5]. The group quickly learned that elevon planes are very difficult to fly, ultimately deciding to use the F40-kit because a quad-copter is easier to control than a fixed wing plane.

Figure 2. The FT Spear is the prototype airframe. [5]

B. RF Emitter (before)

Radio frequency jamming is the process of interfering with an opposing RF signal by overwhelming it with a similar signal of far greater amplitude. In order to successfully jam an emission, it is necessary to have a RF emitter which can generate a waveform with the same frequency and modulation. As the opposing signal and an antenna which can emit the generated signal and a power source robust enough to sustain the signal for a practical amount of time.

Typical RF jamming systems require large amounts of power in order to operate due to exponential attenuation of electromagnetic field strength as distance from an emitter increases linearly.

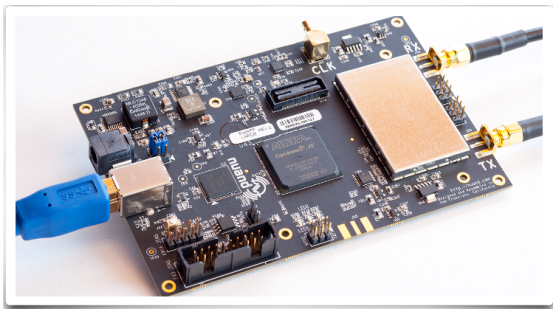

Figure 3. The bladeRF is the RF emitter for the EWS. [6]

Considering that our system needs to function as a drone payload, saving weight is a primary concern. In order to address this, we decided to use the Blade RF software defined radio transceiver shown in Fig. 3 to generate an offensive

signal, a 6400 mAh Venom LiPo battery with an optional directional antenna to focus the signal.

C. Blocking 802.11 Communication

In order to effectively inhibit the wireless communication of a target with minimal equipment, it makes sense to attack the most widely used protocol first. The 802.11 radio protocol, known commonly as 'WiFi', is ubiquitous and is used for high speed data transfer by a wide range of devices.

The 802.11 protocol comes in many forms. Initially the protocol was designed to support frequency and time division multiplexing, methods that allow many devices to communicate with a single access point without undue interference. As software defined radios supporting the protocol have decreased in price, the standard has become the target of numerous attacks. To compensate, variants have been developed in order to provide increased security.

Despite developments, wireless data transmission is far from perfect. In fact, some of the very traits that made multiplexed data transmission viable have ended causing it's longest standing and most frustrating vulnerabilities. The first issue with 802.11 wireless transmission is the issue of client authentication. First, it is necessary to state that current 802.11 implementations support robust encryption. Even in an adversary can capture frames transmitted between a source and destination, the data segment of the frame holding TCP/IP packet or other transmission will be unreadable. The problem with this is that a client device needs to communicate with a host access point in order to set up keys and create the encrypted connection. To do this, the host and client need to exchange unencrypted management frames to establish one another's identities.

Management frames have a number of functions, however the ones with with this paper are concerned are called authentication frames. Authentication frames tell a device that a connection has been successfully made with the host or conversely, that the client's allocated connection period is up and the client needs to reconnect. As can be inferred, abuse of these unprotected authentication frames is the most tried and true method of removing a target client from a network. The deauthentication attack can be implemented in various ways, assuming that the adversary has access to a 802.11 capable radio. This attack, while powerful, has some drawbacks. The largest drawback is that while the deauthentication packets can be sent to a network's multicast address, an address on a wireless network which forwards to all clients, some access points block these transmissions after a certain number of repeats. In order to get around this, the adversary needs a radio capable of capturing 802.11 management frames so that client specific details can be extracted and deauthorization packets can be sent to specific targets, ignoring the multicast and possible filtering. The second issue with the deauthorization attack is that the packets need to be sent frequently in order to truly deny service. While this is fine if the adversary is attempting to remove a single device from a network, this strategy does not scale for networks with large numbers of devices and multicast filtering. Finally, some highly secure networks, usually at large institutions, implement a variant of the 802.11 protocol called 802.11w. This variant uses special Cisco access points to protect management frames. Simply, any client has a window of time to refute management frames received by the host. In an 802.11w protected environment, deauthorization attacks are not an option.

Considering that deauthorization is not a scalable solution, a more robust 802.11 denial method is needed. While it is possible to use a traditional RF jammer and saturate the necessary frequency ranges with noise, hardware filtering can make this attack power intensive and heavy. Luckily, there is another solution. Earlier in this document, it was mentioned that 802.11 gained much of it's popularity due to robust multiplexing. It turns out that this multiplexing support is also it's greatest weakness.

In order to deal with multiple clients communicating simultaneously, 802.11 uses a standard known as 'listen before speaking'. This standard, a type of time division multiplexing borrowed from Ethernet communication and possibly even older wired systems, requires that the client check a line, or channel in the wireless case, for use. If the channel is unused, client A begins transmitting data. If another client B needs to use the line and A is currently transmitting, B sets a backoff timer which is a sufficiently random value for B to wait before attempting to transmit again. This system works extremely well, since multiple clients collisions are resolved by sufficient randomization of backoff timers. Herein lies the problem. If a client decides to go rogue and transmit legitimate data, not noise, without any breaks, all of the other clients will be stuck continually setting backoff timers and never being able to transmit. This effectively jams the network.

Implementing this sort of 'listen before speaking' attack is easier said than done. It requires 802.11 radios with support for a promiscuous capture mode and modifiable firmware. Luckily, Belgian security researcher Mathy Vanhoef [7] has researched this topic extensively and provided us with access to modified firmware. However, some minor modifications to the Vanhoef implementation allowed the system to be quickly switched on and off to facilitate operation on multiple radio units and channels.

D. Blocking Cellphone Communication

DISCLAIMER: All the radio frequency testing and study were operated in controlled environment, a Faraday cage, where no RF signal gets out. The project utilized a low power software defined radio (SDR) to prevent any sort of legal issues, while being aware of the laws and FCC regulations. This study was done by University of Utah senior project students solely for academic purposes and must not be repeated by others.

There are many ways to hinder an individual from communicating. First, the kind of communication type that is going to be jammed must be determined. This could be a communication through cellular, internet, or other means. We have shown two methods of jamming the internet connection

either via deauthentication or RF-jamming. The group has also discovered the ways of interfering cellular connections. The two different ways of jamming are mentioned below.

Way 1: One of the ways the group studied to hinder someone's cellular communication is through plain RF jamming. This method used to be very popular a decade ago when phones couldn't switch to a new band. In order to achieve this method, an attacker would have to broadcast an RF signal which is on the same radio frequency as the cellphone's operating frequency. This method would disrupt the communication between the cellphone and the cellphone tower. However, since there are so many radio frequencies all around us, phones are now made to switch to a different band (which uses a different frequency) if a band is being jammed or simply interfered with. Therefore RF jamming is not as popular as it used to be back when phones could not switch to a new band easily. One of the ways of making this method more efficient is to implement a RF-sniffer which will find out the opponent's current band, transmit that frequency to the RF-emitter and RF-jam that particular frequency. That way whichever band the opponent's switches to, it will be caught by the RF-sniffer and be blocked through the RF-emitter. An open source software called GNU radio could be used to emit a certain frequency from an SDR.

Way 2: Another method we studied to hinder someone's communication is through creating a rogue cellphone tower which acts like the provider. Even though the idea sounds complicated, this can be achieved using variety of SDRs (software defined radios). There are a couple of open source applications than can turn an SDR into a cellphone tower. For GSM, YateBTS and OpenBTS could serve this purpose whereas for LTE you need OpenLTE or SrsLTE. There are differences between setting up GSM versus LTE base stations. For instance, LTE requires a lot more hardware than GSM does due to the nature of the technology.

LTE requires a eNodeB implementation, thus creating a fake cellphone tower is harder and requires extra hardware to be accomplished. GSM on the other hand is more prone to attacks such as these and the complexity of the base station is a lot less compare to the LTE. When a UE (phone) is connecting to a LTE tower, the protocols below have to be authenticated. An attack can still occur without needing all these steps provided in figure 4. All the attacker cares about what is happening between UE and eNodeB which is shown by figure 5.

One might wonder how a phone can determine an authentic eNodeB from an imposter. An attacker can create an eNodeB to advertise itself as the cellphone tower of the victim's provider. All the attacker would have to change is MCC and MNC numbers which indicates the provider's information. If that is the case, the mobile device will connect to any eNodeB that advertises itself with the right information. Figure 5 shows how the LTE protocol is not completely secure since our UE will connect to an eNodeB with the correct provider information. UE will try to connect to the strongest signal with the correct credentials.

There are also various of ways a user can protect themselves

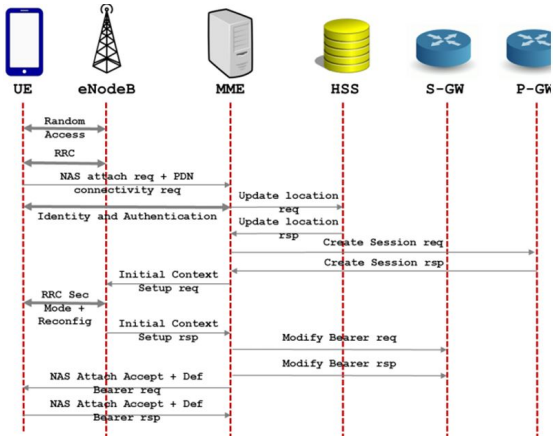

Figure 4. The stages a UE has to handshake for a LTE communication. [8]

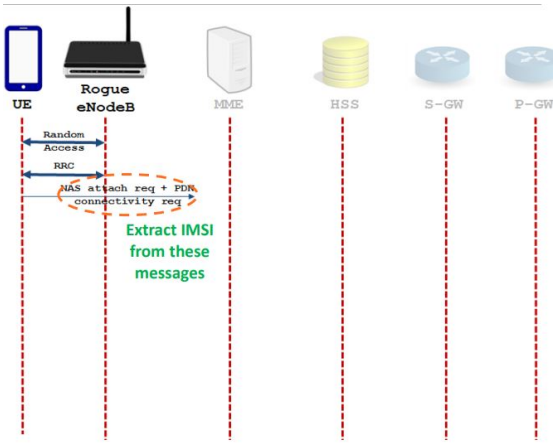

Figure 5. The rogue eNodeB does not need all the stages. [8]

from connecting a fake cellphone tower. Various mobile applications can keep track of the cellphone tower a user's phone connects to. Some of these apps can check the validity of the cellphone tower from the database on their system. If the database does not contain this cellphone tower based on its location, an alert can be sent to the user as well as the provider.

E. Flight Controller

A flight controller is a necessary component in any autonomous airborne vehicle. This system is sometimes referred to as an autopilot which is responsible for controlling the four motors in our quad-copter and other system signals. Inputs from various peripherals are used in the airborne vehicle.

Using the Navio2 as our autopilot system. The Navio2 is an external shield that can be placed on top of a Raspberry Pi. The shield itself comes with all of the necessary components needed for autonomous flight including a gyroscope, accelerometer, Global Positioning System (GPS) chip, barometer chip(checks the altitude of the aircraft), and rails to connect other peripherals as shown in Fig. 6.

The main peripherals used were the 3-axis gyroscope, accelerometer, Global Positioning System (GPS) chip, and

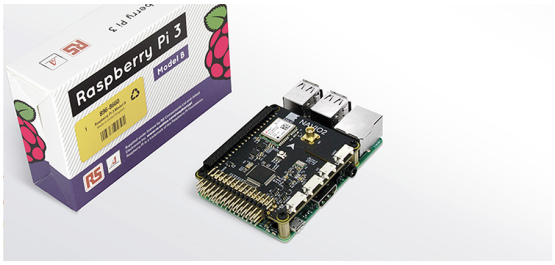

Figure 6. The Navio2 is the heart of the flight controller.

the altitude chip. These provided the necessary feedback to the Raspberry Pi in order to achieve autonomous flight. Autonomous operations include auto leveling in the air, hovering about target area, and communicating between the RF module and EWS module (WiFi jamming).

IV. Project Details

A. Airframe Choice

An appropriate airframe was necessary to carry our payload. Initially, fixed wing airframes were our preferred choice due to increased airtime and more efficient aerodynamics. The previously mentioned FT Spear was chosen due to its inexpensive cost. Our first few prototypes of the drone were built using this frame. This design was implemented using servos to control the elevons and one brushless DC motor to propel the plane. Getting this airframe to fly correctly proved to be an insurmountable challenge for us. The group crashed our prototypes during every flight test, usually requiring the need to order a new frame. This proved to be time consuming and frustrating, thus motivating us to switch designs.

Eventually a quad-copter design was deemed more appropriate for our desired functionality. Many quad-copter frames fell outside of our budget or were overkill for the desired design. The Hobbypower F40 was the perfect combination of affordability and performance. A quad-copter also seemed better form a design standpoint and it is possible to hover the offensive payload over a target. The quad-copter was also much easier to control and was more predictable. Initial flights with this design also resulted in crashes, but were not catastrophic. Crashing the quad-copter resulted in and broken frame arm or leg at worst. Purchasing a second frame was never necessary. Due to the quad-copter's lower speed, crashing was never a serious issue.

B. Autonomous Embedded System Module

The Navio2 is the main component of our projects flight control system. The previously mention peripherals on the Navio2 help monitor flight conditions. The gyroscope gives the flight controller (FC) rolling readings such as, nose up, nose down, slight right, slight left, hard right, hard left, etc. The FC also checks these readings against a value given by the accelerometer to see how far off the vehicle is from its center of balance. Monitoring these levels allows the drone to maintain steady flight en route to its target. GPS coordinates are read in from the GPS chip and antenna. This data is checked every so often by the FC to see if the vehicle has reached its destination. The barometer chip provides altitude readings to ensure the drone stays above ground. The flight controller then sends the appropriate pulse width modulation (PWM) to the motors. Fig. 7 shows how these individual modules contribute to autonomous flight.

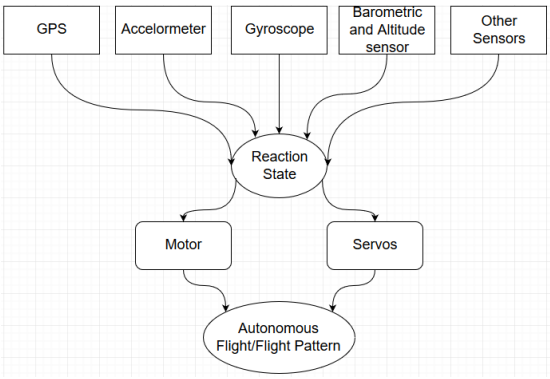

Figure 7. Simple flow chart explaining how the Navio2 peripherals contribute to the flight controller.

The reaction state in the center of the flow chart is broken down into several chunks running in an endless loop. These 'chunks' are ordered and flagged by priority, meaning that we will be checking the gyroscope data and accelerometer data more often than the GPS data. The reason behind this choice is because we need to ensure that the vehicle is always stable during flight time. When this is achieved, then check the position of the vehicle and compare it to the preloaded destination coordinates designated by the user before the vehicle was launched.

When the vehicle has reached a certain radius and altitude near the destination point chosen by the user, the FC will then trigger an event which will cause the vehicle to hold a flight pattern by the destination. The flight pattern is going to be a circle formation allowing the vehicle to stay relatively close to the target destination and allowing time for the FC to send a signal to the EWS module.

C. Potential Flight Controller Alternatives

There are many other autopilot systems out there such as Pixhawk, Erie-Brains2, PXFmini, and APM2. These are all great autopilot systems. The Pixhawk offers the most add-ons an autopilot can come with, such as five UART ports, two CAN ports, I2C and SPI ports, multiple micro USB ports and multiple voltage inputs. This allows the Pixhawk to be the one of the most customizable autopilot systems out there as shown in figure 8. There are a lot of connections that can be used for other peripherals wanted to be used by potential buyers. This autopilot was a little much for what the project needed, but it could potentially be used for another project in the near future.

The Erle-Brain2 offers quite a bit less functionality but still holds true to being a really good autopilot system, as

Figure 8. PixHawk A overkill autopilot

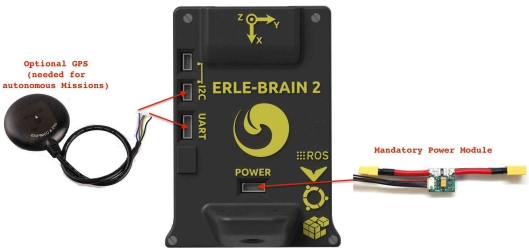

Figure 9. Erle-Brain2 Minimal autopilot

it offers a UART connection, and two I2C connections along with a power connection. This flight system is very good if you want a very basic aircraft with minimal peripheral, as show in figure 9. The Erle-Brain2 did not however offer us the functionality we wanted for our system.

D. Selected Flight Controller

One of the main reasons the group picked up the Navio2 was because it was the only autopilot system that could actually attach to our Rpi3 and could communicate directly to and from it. This allowed us to have only one power module along with one BEC (Battery Elimination Circuit) which connects directly to our battery to supply power for the Navio2, Rpi3, and the BladeRF. This configuration of the payload played out well for our quad-copter. The basic power module steps down the voltage so battery can power our payload without the need to worry about sending to much power into the system. The basic power module can be seen in figure 10.

Figure 10. Simple Power-Module

The Raspberry Pi 3 provided us with four USB ports which could used to attach wireless adapters for Wifi jamming and the BladeRF for GSM interception. With the Navio2 being directly attached to the Rpi3, this allowed us to have one less communication puzzle to solve, as shown in Fig 11.

Figure 11. Finished quad-copter design ready for flight.

E. Ground Station and Mission Planning

Mission Planner is a open source Ground Control Station (GCS) capable of loading and configuring parameters into an autopilot system through telemetry. The group wanted to use an open source solution to have a better understanding of how the ground station functioned. Mission Planner can be run on any device capable of running a modern Windows operating system. Mission Planner provides us with data coming from the auto pilot system such as GPS coordinates, altitude, and directional data. This in turn can be used to override parameters during flight or before take off. Mission Planner also provides valuable feedback to the user through a graphical interface. A screenshot of the Mission Planner software in action can be seen in figure 12.

Mission Planner provides a number of useful parameters to customize flight paths. There are many different supported vehicle types such as quad-copters, fixed wing aircraft and rovers. The group eventually used a quad-copter setup due to difficulties with the fixed wing aircraft as mentioned above. Mission planner requires a number of test flights to tune and calibrate the flight controller. The fixed wing design would not fly long enough to collect useful data. Our efforts at trying to use a fixed wing design made transitioning to a quad-copter much easier.

Figure 12. Demo of Mission Planner.

There are many different kinds of GCS's to pick from such as QGroundControl, APM Planner, and MAVProxy. Each of these choices provided their own benefits and challenges. MAVProxy is a completely command line based GCS, whereas

the other choices are very similar to Mission Planner but are Linux based. Mission Planner is the most popular choice and has the most readily available documentation. Mission Planner is also the most stable GCS available and offered the functionality that was wanted for the project.

1) Communication Between Modules: In order to achieve our goals, the group wrote custom scripts to get all of the individual modules to communicate and operate correctly. One such script collected GPS coordinates from the flight controller and provided this information to our electronic warfare package. This script calculated a jamming radius from a preset coordinate to ensure jamming only occurred at specific locations. As jamming throughout the entire duration of the flight was not a good solution. Jamming in selective areas also improved battery life. Primarily using BASH scripts to achieve most of the communication between modules. Python scripts were written to perform more complex mathematical calculations and other higher level tasks.

The radius calculation script is done in Python as Python is better suited for mathematical computations than BASH. In the python script named "pymavGPS.py" you can see the calculation which is basically using the formula for a circular hit box, $X^2 + Y^2 = R^2$ where X is the latitude and Y is the longitude. Taking these coordinates and multiplying them with a constant that converts them from degrees to feet measurements making it easier for us to get the radius around the target. The code is also using MAVLink to translate the data coming in from the peripherals of the drone.

MAVLink manages all of the messages that get sent from the GCS and the drone. Using some of the flags that get sent from the drone to the GCS, the group was able to get live GPS data from the drone during flight. More reading on MAVLink can be found in [9] also checking out the common message documentation can show you how much information we can get from the drone, but for our case the project used the GPS data.

F. Difficulties with Autonomous Flight

Throughout the development of this project our group encountered several difficulties that the group had to get resolved. The first one was wiring all of the ESC wires above the Navio2. This caused many problems during flight, as the wires going over the flight system were causing a lot of electromagnetic interference leading to very poor flight. Once realizing that the wiring configuration was causing interference, disassembling and re-wiring the drone was a necessary step. Doing so eliminated a tremendous amount of interference and allowed us to get closer to realizing autonomous flight.

The second difficulty was not realizing that a cover was needed to protect the barometer chip on the Navio2. This of course was a very obvious problem when the group would see the drone flying correctly but whenever there was a slight breeze or the drone was trying to raise to the correct altitude, the drone would suddenly lose altitude. We discovered that the barometer chip is sensitive to airflow and sunlight. When exposed to these situations, the chip will provide unreliable readings causing erratic flight.

The last difficulty was calibrating the compasses correctly. For calibrating the compass the user needs to cover multiple axises to get the most accurate reading from the compasses. There were several calibration setups done for the compasses. In mission planner there was a live calibration for the compasses, this calibration shows all of the axises that need to be covered. During the calibration it is necessary to move the drone around in the correct orientation to get the correct readings. The live calibration can be seen in Fig 13.

In the figure the red dot is where the compasses are orientated and the idea is to move it around the X, Y, and Z axis to get a nice circle around the origin of the graph. The more data points you have the more accurate compass readings become. The colors represent how strong the reading is during the calibration, so seeing a little bit of red in between the Z and Y axis is a little bad, but has not lead to any problems with autonomous flight. Noting that $compass_1$ is the left most graph and $compass_2$ is the middle graph. The live calibration was okay but there was also another calibration that is preferred by the drone community, this is a on-board calibration. This calibration has no visual aid besides a little bar. During the calibrations it was found that using a rotating chair was the easiest way to get accurate readings.

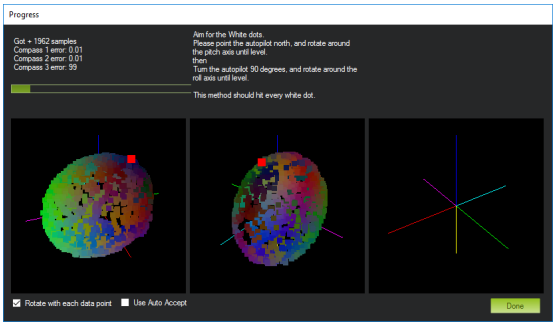

Figure 13. Live Calibration Of Compasses On Navio2

V. Testing

Due to the nature of this project, there were many different types of tests that needed to be done. There were two main parts of the project that needed testing: Software systems and 2.4GHz WiFi jamming.

Software Systems

At the heart of the project is the Linux kernel. Our project needed something that could handle a flight controller and manage the radios necessary for jamming wireless signals. The group tried many different types of Raspberry Pi supported Linux distributions such as Ubuntu MATE, Kali Linux and Raspbian. Our initial plan was to create our own flight controller software and interface it with the Rpi3. It was quickly learned that this is difficult and fell outside of our time constraints. Instead, the project uses open-source implementations of flight controllers created for the Raspberry Pi.

A pre-configured version of Raspbian that supported the use of the Navio2 hardware was eventually used for the project [10].

A. 2.4 GHz Wi-Fi Jamming

One of the main parts of the electronic warfare package is the 2.4GHz jamming. Two TP-Link WN722N v1 wireless adapters with modified firmware are used to create continuous interference on multiple channels, thus blocking clients from connecting to access points. The code for the firmware was provided by Belgian security expert Mathy Vanhoef, with details of the jamming found in his paper *Advanced Wi-Fi Attacks Using Commodity Hardware* [7]. In order to test the range and effectiveness of the jamming, a python script was created that would send 100 pings/second to a website. A live graph would update and show how many pings were acknowledged. The range of the jamming is highly dependent on the antennas used on the wireless adapters. Our final implementation used the stock antennas the came with the wireless adapters. Testing was done on an American football field as it was easy to test how far our jamming signals could reach. The drone would be placed in one end zone of the field and a laptop with an internet connection in the other end zone. The laptop was moved 1 yard/second closer to the drone. After numerous tests, it was found that the jamming radius was approximately 30 yards. An example test is shown in figure 14.

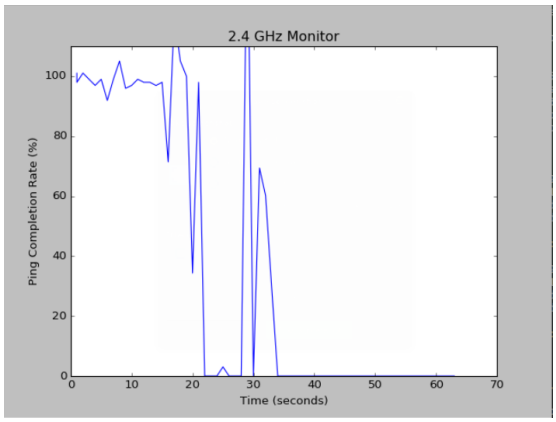

Figure 14. Graph from jamming scripts. Tests began from 50 yards out and moved closer to jamming source at 1 yard/second. Jamming becomes effective at about 30 yards.

VI. Assigned Tasks to Members and Scope of Project

Table I
Team Member Responsibilities

Team Member	Position
Charlie Shoup	Primary RF Engineer
Matt Blessing	Primary Embedded Systems Engineer
Ben Broadhead	Primary Software Engineer
Enes Oguz	Primary Hardware Engineer

The design approach were broken into several stages. The first stage was working on the FC and drone body to get it flying autonomously and determine the technical specifications of the EWS module. The second stage was working on the weight and wiring of the drone body to make the drone suitable for adding the RF module along with configuring the RF module to do the task at hand.

Table II
Parts List

Item	Qty	Price/Unit ($)	Total Cost ($)
Hobbypower F40 Kit	1	130.00	130.00
Venom 20C LiPo	1	65.00	65.00
bladeRf Radio	1	400.00	400.00
TP-LinkWN722N	2	15.00	30.00
Raspberry Pi 3	1	35.00	35.00
ODROID XU4Q	1	85.00	85.00
Navio2	1	165.00	165.00
GPS Antennae	1	30.00	30.00
BEC	1	25.00	25.00
Power Module	1	15.00	15.00
Wireless Transmitter	1	140.00	140.00
Wireless Receiver	1	30.00	30.00
3DR Telemetry Radio	1	25.00	25.00
Total Cost			**1175.00**

A. Risk Assessment

There were a substantial number of risks inherent to this project. The first and most basic risk was the goal of autonomous airborne drone operation. Within the last handful of years, a number of University of Utah ECE senior projects have concerned themselves with autonomous flight. However, based on professor feedback, few if any have been as successful as initially intended. Unlike many of those who have come before, the drone aimed to use a fixed wing airframe or quad-copter frame.

The second major risk to the project was the entirety of the EWS payload. Tackling this challenge was more problematic than autonomous flight, however worthy effort was made to grow the team's knowledge base.

During the beginning of this project a fixed-wing drone was used to get familiar with how the flight systems worked. As time went on it was apparent that fixed-wing drones were a little too complicated to fly and quickly switched to a quad-copter drone. This switch was a very easy transition because the group had already got very acquainted with airframe motors, ESC's and propellers. This knowledge allows us to build a quad-copter very quickly and were able to get it up in the air in less then a week. This was considered as a risk because the group used a lot of time to get familiar with what type of motors were needed along with what type of propellers were necessary to achieve flight.

VII. Conclusion

Building an autonomous airborne drone with an EWS payload was no easy task. The resources collectively referenced by this paper do not provide an immutable blueprint for the final implementation of this project. They represent two semesters

of research and testing by which the team has convinced itself and others that the stated project goals were met.

In conclusion, this project demonstrated that it is possible to build a low cost autonomous flying electronic warfare system. This project shows that is it possible to apply this design on fixed wing drones or quad-copters with a relatively low cost allowing others to further develop similar, more advanced designs.

A. Acknowledgements

Special thanks goes out to Neil Cotter, Omar Shami, and Jon Davies. These people were very helpful during the course of this project. Thanks to Neil Cotter for his mentoring and support. Thanks to Omar Shami for dealing with our amazing group and always weighing the drone frame. Thanks to Jon Davies for the lab access lending us a battery charger.

Also thanks Ken Stevens for the amazing senior project course.

References

[1] Y. Ganesh, R. Raju, and R. Hegde, "Surveillance drone for landmine detection," pp. 33–38, Sept 2015.

[2] G. Quiroz and S. J. Kim, "A confetti drone: Exploring drone entertainment," in *2017 IEEE International Conference on Consumer Electronics (ICCE)*, Jan 2017, pp. 378–381.

[3] Raytheon, "Raytheons mald models of interest," 2017, [Online; accessed 10-April-2017]. [Online]. Available: https://en.wikipedia.org/wiki/ADM-160_MALD

[4] ——, "Raytheons mald-j model," 2017, [Online; accessed 10-April-2017]. [Online]. Available: http://www.raytheon.com/capabilities/products/mald/

[5] F. Control, "Ft spear," 2017, [Online; accessed 23-March-2017]. [Online]. Available: http://store.flitetest.com/ft-spear/

[6] Nuand, "bladerf," 2017, [Online; accessed 23-March-2017]. [Online]. Available: https://www.nuand.com/

[7] M. Vanhoef and F. Piessens, "Advanced wi-fi attacks using commodity hardware," in *Proceedings of the 30th Annual Computer Security Applications Conference*. ACM, 2014, pp. 256–265.

[8] R. P. Jover, "Exploring lte security and protocol exploits," 2016, [Online; accessed 1-December-2017]. [Online]. Available: http://www.ee.columbia.edu/~roger/LTE_open_source_HackerHalted.pdf

[9] MAVLink, "Mavlink cite," 2017, [Online; accessed 4-December-2017]. [Online]. Available: http://qgroundcontrol.org/mavlink/start

[10] EMLID, "Emlid rasbian image," 2017, [Online; accessed 04-December-2017]. [Online]. Available: https://docs.emlid.com/navio2/common/ardupilot/configuring-raspberry-pi/