Instituto Universitário de Lisboa

Departamento de Ciências e Tecnologias da Informação

Captura de UAVS Através de *Spoofing* de Sinal GPS

João Filipe de Quadros Gaspar

Dissertação submetida como requisito parcial para obtenção do grau de

Mestre em Engenharia de Telecomunicações e Informática

Orientador:
Doutor Pedro Joaquim Amaro Sebastião, Professor Auxiliar
ISCTE-IUL

Coorientador:
Doutor Nuno Manuel Branco Souto, Professor Auxiliar
ISCTE-IUL

Dezembro, 2018

Agradecimentos

Em primeiro lugar gostaria de agradecer ao meu orientador, Professor Doutor Pedro Sebastião, por toda a cooperação e apoio durante o desenvolvimento da dissertação, tanto como ao meu coorientador, Professor Doutor Nuno Souto, pela sua disponibilidade e apoio nas dificuldades que foram surgindo.

Ao Professor Doutor Francisco Cercas pela disponibilidade de um espaço ideal para concentração e desenvolvimento da dissertação.

À minha família, o meu muito obrigado por todo o apoio e encorajamento durante todo o período de estudos académicos, em especial à minha irmã Sancha Gaspar pela ajuda na verificação ortográfica da dissertação.

Aos meus colegas que me acompanharam durante toda essa caminhada, pela vossa entreajuda e amizade.

Por fim agradecer ao Instituto Superior de Ciências do Trabalho e da Empresa e ao Instituto de Telecomunicações por terem contribuido imensamente para a minha formação; por terem dispertado o espírito de sacrificio tão necessário nesta fase e por testarem esta minha vontade imensa de saber e fazer mais. São de facto instituições de excelência.

A todos os que enumerei o meu sincero “Obrigado”.

Resumo

O aumento de utilização de drones por parte de civis, tem crescido exponencialmente. Com o aparecimento de diferentes marcas no mercado o seu preço de aquisição tem diminuído e os seus sistemas de controle e programação de voos autónomos têm melhorado significativamente, o que tem implicado a ocorrência de maior número de acidentes e situações de perigo com drones em voos sobre áreas protegidas como aeroportos, eventos públicos e zonas militares. Sendo escassos os sistemas capazes de extinguir as ameaças provocadas por voos indevidos de drones lança uma oportunidade de se avançar no desenvolvimento de produtos e serviços para eliminar essas advertências aos drones, principalmente naqueles que possuem sistema de voo autónomo por sinais de *Global Navigation Satellite Systems* (GNSS).

Utilizando plataformas programáveis de *Software Defined Radio* (SDR) e simulando sinais de *Global Positioning System* (GPS), é possível transmitir sinais falsos, desviar, ou até mesmo tomar o controlo sobre drones, cuja rota de voo dependa da informação obtida pelo sistema GPS. Neste sentido pretende-se induzir no drone, erro quanto à sua localização e, assim, desviar o destino final. Como situação particular, o trabalho a apresentar nesta dissertação descreve o conjunto de *hardware* e *software* utilizados, bem como os algoritmos eficientes implementados, necessários para executar o *spoofing* de drones de forma inteligente e eficaz.

Recorrendo também a testes realizados, em laboratório com diferentes tipos de recetores, que comprovam ser possível, estática e dinamicamente, realizar o *spoofing* de um recetor de GPS e ganhar o controlo, indiretamente, de um drone em voo autónomo.

Palavras-Chave: Drone, *Global Navigation Satellite System, Global Positioning System, Software Defined Radio, Spoofing*.

Abstract

The increased use of drones by civilians has grown exponentially. With the appearance of different brands in the market the acquisition price has decreased and its autonomous flight control systems have improved significantly, which has resulted in a greater number of accidents and dangerous situations with drones flying over protected areas such as airports, public events and military zones. Being scarce systems capable of extinguishing the threats caused by improper drone flights is an opportunity to move forward with the development of products and services to eliminate these threats with drones, especially those with autonomous flight systems by Global Navigation Satellite Systems.

Using programmable Software Defined Radio platforms and simulating Global Positioning System signals, it is possible to transmit false signals, divert, or even take control of drones whose flight path depends on the information obtained by the GPS system. In this sense it is intended to induce in the drone an error of its location and, thus, to divert its final destination. As a particular situation, the work presented in this dissertation describes the set of hardware and software used, as well as the efficient algorithms implemented, necessary to execute the spoofing of drones in an intelligent and effective way.

Using tests carried out in the laboratory with different types of receivers, proving that it is possible to statically and dynamically perform the spoofing of a GPS receiver and indirectly gain control of a drone in autonomous flight.

Keywords: Drone, Global Navigation Satellite System, Global Positioning System, Software Defined Radio, Spoofing.

Índice

Agradecimentos ..... **i**

Resumo ..... **iii**

Abstract ..... **v**

Índice ..... **vii**

Índice de Tabelas ..... **xi**

Índice de Figuras ..... **xiii**

Lista de Acrónimos ..... **xv**

Capítulo 1 - Introdução **1**

1.1. Enquadramento do Tema 1

1.2. Motivação e Objetivos 2

1.3. Contribuições 2

1.4. Estrutura e organização da dissertação 3

Capítulo 2 – Estado da Arte de UAVS, Sistema GPS e *Spoofing* **5**

2.1. *Unmanned Aerial Vehicles* 5

2.1.1. Tipos de Veículos 5

2.1.2. Sistemas de Comunicação 6

2.2. Sistema GNSS 7

2.2.1. Descrição do Sistema GPS 8

2.2.2. Frequências, Códigos e Modulações GPS 9

2.2.3. Trilateração e Localização com GPS 10

2.3. *Spoofing* 11

2.3.1. Tipos de *Spoofing* 12

2.3.2. GPS *Spoofing* 12

2.3.3. Trabalhos Relacionados GPS *Spoofing* 13

Capítulo 3 - Implementação **19**

3.1. Descrição do Sistema 19

3.2. Arquitectura 20
3.3. *Hardware* e *Software* 21
3.3.1 Hardware 21
3.3.2 Software 25
3.4. Desenvolvimento Analítico 27
3.4.1. Determinação da Localização do Drone 27
3.4.2. Desvio do *Unmanned Aerial Vehicle* 30
3.5. Comunicação 35
3.6. Circuitos Eletrónicos e Alimentação 38
3.6.1. Circuitos Eletrónicos 38
3.6.2. Alimentação 39
Capítulo 4 - Testes e Discussão de Resultados 43
4.1. *Software* bladeGPS 43
4.1.1. Testes *Indoor* 43
4.1.2. Testes *Outdoor* 47
4.2. Sensores 50
4.2.1. Sensor Lidar 51
4.2.2. Sensor *Pitch* 52
4.2.3. Sensor Bússola 54
Capítulo 5 - Conclusões e Trabalhos Futuros 57
5.1. Principais Conclusões 57
5.2. Propostas de Investigação Futura 58
Bibliografia 59
Apêndices e Anexos 63
Apêndice A. Prototipagem 64
A.1. Protótipo Laboratorial 64
A.2. Protótipo Real 65

Apêndice B. Manual de Utilizador 68

B.1. Procedimentos pré-utilização 68

B.2. Procedimentos de utilização 68

Anexo C 71

C.1. *Datasheet* Lidar 71

C.2. *Datasheet* MPU6050 73

C.3. *Datasheet* lsm303d 74

Índice de Tabelas

Tabela 2-1 Categorias dos UAVS e as suas características [2], [4], [5] 6
Tabela 2-2 GPS L1 carateristicas técnicas [10] 10
Tabela 3-1 NMEA 2.0 [40] 31
Tabela 3-2 Consumos energéticos bladeRF com amplificador XB-300 [42] 40
Tabela 4-1 Valores de erro medidos pelo sensor Lidar 51
Tabela 4-2 Valores de erro medidos pelo sensor MPU6050 53
Tabela 4-3 Valores de erro medidos pelo sensor lsm303d 55

Índice de Figuras

Figura 2-1 Trilateração e localização por satélite [11] 11
Figura 2-2 Demonstração dos componentes, geometria e atrasos no spoofing gps [16] 13
Figura 2-3 GPS spoofer civil [17] 14
Figura 2-4 Diagrama de blocos sistema de spoofing [17] 14
Figura 2-5 Hornet Mini UAV [17] 15
Figura 2-6 Esquema utilizado para testes de spoofing [17] 15
Figura 2-7 USRP B210 [19] 16
Figura 2-8 bladeRF x40 [19] 16
Figura 2-9 Recetor GPS smartphone Nexus 5 [19] 16
Figura 2-10 Samsung Note 3 e Iphone 6 [19] 17
Figura 3-1 Esquema total do sistema de spoofing 20
Figura 3-2 Sensor Lidar-Lite v3 [21] 21
Figura 3-3 MPU6050 sensor Roll,Pitch e Yaw [23] 22
Figura 3-4 Roll, Pitch e Yaw de smartphone [24] 22
Figura 3-5 Pololu lsm303d ecompass [25] 23
Figura 3-6 Recetor GPS MAX-7Q-0-000 [27] 23
Figura 3-7 Arduino Uno Rev3 [28] 23
Figura 3-8 Raspberry pi 3 model B [29] 24
Figura 3-9 bladeRF x40 [30] 24
Figura 3-10 Software utilizado para desenvolvimento e execução do sistema 25
Figura 3-11 Visual Studio Code e Arduino IDE [28], [32] 25
Figura 3-12 Framework Electron [33] 26
Figura 3-13 Ubuntu MATE 16.04 [34] 26
Figura 3-14 Vista de plano lateral cálculos analíticos 28
Figura 3-15 Vista de plano superior cálculos analíticos 28
Figura 3-16 Interface do software bladeGPS 30
Figura 3-17 Estrutura de uma sentence GPGGA 32
Figura 3-18 Ilustração de exemplo de spoofing estático 32
Figura 3-19 Ilustração de exemplo de spoofing dinâmico 33
Figura 3-20 Ângulos e calculos analíticos de spoofing dinâmico 34
Figura 3-21 Troca de mensagens ao premir gatilho em modo spoofing estático 35
Figura 3-22 Troca de mensagens ao premir gatilho em modo spoofing dinâmico 37

Figura 3-23 Troca de mensagens ao soltar gatilho 38
Figura 3-24 Circuito eletrónico do sistema 39
Figura 3-25 Bateria Li-PO 2200 e regulador de tensão output 5.5V 41
Figura 4-1 Aplicação GPSTest [43] 44
Figura 4-2 Spoofing de smartphone com aplicação GPSTest indoor 44
Figura 4-3 u-blox M8 GNSS Evaluation Kit 45
Figura 4-4 Spoofing recetor u-blox M8 GNSS Evaluation Kit indoor 45
Figura 4-5 Recetor u-blox MAX-7Q 46
Figura 4-6 Output do programa de teste recetor u-blox MAX-7Q spoofing indoor 46
Figura 4-7 Spoofing de smartphone com aplicação GPSTest e GoogleMaps outdoor .. 48
Figura 4-8 Spoofing recetor u-blox M8 GNSS Evaluation Kit outdoor 49
Figura 4-9 Output do programa de teste recetor u-blox MAX-7Q spoofing outdoor 50
Figura 4-10 Introdução de valores no software bladeGPS 50
Figura 4-11 Sensor Lidar implementado na arma 51
Figura 4-12 Gráfico de precisão de medição do sensor Lidar 51
Figura 4-13 Sensor Pitch implementado na arma 52
Figura 4-14 Teste laboratorial ao sensor MPU6050 (pitch) 52
Figura 4-15 Teste laboratorial ao sensor MPU6050 com aplicação de smartphone 53
Figura 4-16 Gráfico de precisão de medição do sensor MPU6050 53
Figura 4-17 Sensor de bússola implementado na arma 54
Figura 4-18 Teste laboratorial ao sensor lsm303d (bússola) 54
Figura 4-19 Teste laboratorial ao sensor lsm303d com aplicação de smartphone 55
Figura 4-20 Gráfico de precisão de medição do sensor lsm303d 56
Figura A - 1 Protótipo laboratorial 64
Figura A - 2 Resultados do protótipo laboratorial através de ecrã 64
Figura A - 3 Projeto Autodesk Inventor Profissional de protótipo real 65
Figura A - 4 Construção das caixas para as plataformas que integram o sistema 66
Figura A - 5 Estrutura principal do protótipo 66
Figura A - 6 Integração de gatilhos, seletores de modo e interruptor de alimentação ... 67
Figura A - 7 Ligações electrónicas e de alimentação no interior do protótipo 67
Figura B - 1 Descrição e localização de cada componente do protótipo desenvolvido . 69

Lista de Acrónimos

ANAC	Autoridade Nacional de Aviação Civil
API	Application Programming Interface
ARM	Advanced RISC Machine
ASCII	American Standard Code for Information Interchange
BOC	Binary offset carrier
BPSK	Binary Phase-Shift Keying
C/A	Coarse/Acquisition code
CDMA	Division Multiple Access
CSS	Cascading Style Sheets
DNS	Domain Name System
FPGA	Field-Programmable Gate Array
GNSS	Global Navigation Satellite System
GPGGA	Global Positioning System Fix Data
GPS	Global Positioning System
HALE	High Altitude Long Endurance
HTML	HyperText Markup Language
I2C	Inter-Integrated Circuit
IBW	Instantaneous Bandwidth
ICSP	In Circuit Serial Programming
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IP	Identification Protocol
ISCTE	Instituto Superior de Ciências do Trabalho e da Empresa
ISM	Industrial, Scientific and Medical
IUL	Instituto Universitário de Lisboa

LED	Light Emitting Diode
Li-PO	Lithium-Ion Polymer
LNA	Low-Noise Amplifier
MAC	Media Access Control
MALE	Medium Altitude Long Endurance
MAV	Micro Aerial Vehicle
MEMS	Micro-Electro-Mechanical Systems
NAVSTAR	Navigation Satellite with Time And Ranging
NMEA	National Marine Electronics Association
NUAV	Nano Unmanned Aerial Vehicle
OSQZSS	Open Source Quasi-Zenith Satellite System
PPS	Precise Positioning Service
PWM	Pulse Width Modulation
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
SCL	Signal Clock
SDA	Signal Data
SDR	Software Defined Radio
SNR	Signal Noise Ratio
SoC	System on a Chip
SPI	Serial Port Interface
SPS	Standard Positioning Service
SR	Sample Rate
STDIN	Standard Input Stream
STDOUT	Standard Output Stream
TCP	Transport Comunication Protocol

TUAV	Tactical Unmanned Aerial Vehicle
UAV	Unmanned Aerial Vehicle
USB	Universal Serial Bus
VANT	Veículos Aéreos não Tripulados
VGA	Variable Gain Amplifier
WiFi	Wireless Fidelity

Capítulo 1 - Introdução

Neste primeiro capítulo será explicado o enquadramento e quais as motivações que levaram ao interesse no desenvolvimento da presente dissertação, tal como os objectivos de investigação a serem atingidos, algumas contribuições produzidas pelo autor da dissertação e por fim como a dissertação se apresenta estruturada.

1.1. Enquadramento do Tema

A utilização de *Unmanned Aerial Vehicles* (UAVS), usualmente conhecidos como drones, tem aumentado significativamente. O aumento da sua popularidade, acessibilidade e aumento de marcas/modelos disponíveis no mercado, levou a uma facilidade de compra e diminuição do preço de aquisição. Estes veículos são aparelhos cuja sua manipulação passou a ser cada vez mais acessível, tendo em conta que os mesmos possuem sistemas de controlo automático para controlo do voo e da estabilidade.

Todos os drones com capacidade de voar/planar são considerados aeronaves. E, como todas as aeronaves estão sujeitas a regras de segurança e normas legais, os drones não são exceção, tal como se confirma pelo Regulamento nº 1093/2016 de 24 de novembro, emanado pela Autoridade Nacional de Aviação Civil (ANAC).

Devido à difícil monitorização dos drones e, à constante ocorrência de acidentes, nomeadamente, com voos em zonas próximas de aeroportos, áreas militares, áreas restritas, proibidas, perigosas ou reservadas, foram emergindo algumas soluções para o combate destes acidentes com o desenvolvimento e utilização de *jammers*, armas de fogo e falcões treinados com o propósito de «caçar» os drones, as quais podem levar à danificação do aparelho em si, ferir o animal responsável pelo abate, bem como à diminuição da segurança pessoal ou de cidadãos presentes no local onde o aparelho intercetado poderá vir a aterrar de forma descontrolada.

Outra das soluções possíveis, para o controlo de drones em zonas ou situações descritas no parágrafo anterior, consiste no *spoofing* do comando do drone, independentemente de este ser controlado por navegação por satélite, WiFi ou outro tipo de ondas rádio. Conseguindo assim ganhar o controlo do veículo e desviar da área protegida, ou até mesmo aterrar o aparelho.

1.2. Motivação e Objetivos

O fator cativante que levou à escolha deste tema de dissertação consiste na atual necessidade de criação de dispositivos de proteção contra os voos de drones em zonas não autorizadas, uma vez que se trata de um tema de segurança pública e de um projeto com uma enorme potencialidade comercial. Pois trata-se de uma alternativa já testada, com alguns casos de sucesso, mas, até à presente data em Portugal, não se efetivou num produto final que consiga solucionar o problema.

Por esta ser uma forma menos evasiva ou violenta de controlar ou desviar a rota do drone, pois não abate o mesmo de forma descontrolada, torna-se também na solução mais desafiante e de maior dificuldade de concepção.

O desenvolvimento do trabalho de investigação que é apresentado nesta dissertação, consiste na conceção e implementação de um sistema capaz de transmitir sinais falsos de navegação por satélite, que poderão conduzir a uma nova rota ou um desvio do percurso do drone. Este sistema fundamenta-se na geração de sinais de navegação por satélite e, atrair o drone a mudar a sua fonte de navegação por satélite passivamente ou forçadamente utilizando um sistema de *jamming* desenvolvido na dissertação de outro aluno. Todos esses progressos e contribuições têm como fim ser integrados numa «arma» protótipo, tornando o sistema móvel e adaptável a qualquer cenário ou terreno.

1.3. Contribuições

Participação na Noite Europeia dos Investigadores – Ciência na Cidade, no dia 28 de setembro de 2018, no Museu Nacional de História Natural e da Ciência da Universidade de Lisboa.

Artigo aceite e apresentado na conferência Global Wireless Summit (GWS- 2018) 25 a 28 de novembro de 2018, Chiang Rai, Thailand.

1.4. Estrutura e organização da dissertação

O presente estudo está organizado em cinco capítulos que pretendem refletir as diferentes fases até à sua conclusão.

O primeiro capítulo introduz o tema da investigação, a motivação, objetivos, principais contribuições e uma breve descrição da estrutura da dissertação.

O segundo capítulo reflete uma visão geral dos *Unmanned Aerial Vehicles* (UAVS), enquadramento teórico do sistema GPS, tipos de *spoofing* e a sua aplicação no GPS.

O terceiro capítulo é dedicado à implementação do sistema desenvolvido, com descrição detalhada de todos os processos e progressos produzidos durante o período experimental.

O quarto capítulo apresenta a execução de testes laboratoriais, práticos e a análise dos resultados obtidos.

No quinto e último capítulo apresentam-se as conclusões deste estudo bem como as recomendações e trabalhos futuros.

Capítulo 2 – Estado da Arte de UAVS, Sistema GPS e *Spoofing*

Neste capítulo é feita uma breve descrição dos UAVS, explicação do funcionamento dos sistemas GNSS, mais especificamente do sistema GPS explicando como se encontra implementado e o seu funcionamento. Uma breve introdução à técnica de *spoofing*, mais aprofundadamente a sua utilização no GPS. Por fim, uma descrição de vários testes e experiências, já alcançadas por outros autores, de *spoofing* de sinais GPS.

2.1. *Unmanned Aerial Vehicles*

Um UAV, ou em português Veículo Aéreo não Tripulado (VANT), é constituído por um veículo e os mecânismos, logística e equipamentos necessários para o seu funcionamento [1]. Quando se fala em drone é muito fácil imaginar um veículo robusto capaz de executar qualquer tarefa autonomamente, mas na verdade o drone é apenas um componente de um UAV.

Os UAVS atualmente são muito utilizados para aplicações civis como lazer, monitorização de agricultura e filmagens, mas também para fins militares, ajudando as forças militares em situações ou cenários de alto risco para vidas humanas e para vigilância de locais remotos.

Um UAV é estruturado e desenvolvido para respeitar as seguintes componentes: o veículo, sistemas de lançamento e recuperação, estação de controle e sistemas de comunicação; e mecanismos de transporte e apoio [2] [3]. Para o desenvolvimento desta dissertação apenas será necessário focar mais as componentes de estação de controle e sistemas de comunicações dos UAV, uma vez que será onde o sistema de *spoofing* irá atuar.

2.1.1. Tipos de Veículos

Os UAVS são classificados de acordo com as seguintes categorias: Nano (NUAV), Micro (MAV), Mini (MUAV), *Tactical* (TUAV), *Medium Altitude Long Endurance* (MALE), e *High Altitude Long Endurance* (HALE) [2] [3]. As aeronaves para grandes

altitudes, como HALE ou MALE, têm necessariamente uma configuração de asa fixa, enquanto que os veículos para baixa altitude, como os TUAV e abaixo, permitem o uso de configurações com vários motores, podendo ser um helicóptero ou um multi-rotor.

Na Tabela 2-1 estão representadas as várias categorias dos UAV e as suas principais características.

Tabela 2-1 Categorias dos UAVS e as suas características [2], [4], [5]

Categoria	Peso [kg]	Altitude [m]	Autonomia [h]	Distancia [km]	Configuração	Operações	Aplicação
HALE	450-13500	>15000	> 24	transglobal	Asa fixa	Militar	Reconhecimento e vigilância
MALE	450-13500	5000-15000	< 24	500	Asa fixa	Militar	Reconhecimento e vigilância
TUAV	15-450	500-5500	< 24	100-300	Asa fixa Multi-rotor	Militar	Suporte aéreo e vigilância
Mini UAV	< 20	300-3000	< 1	< 30	Asa fixa Multi-rotor	Militar/Civil	Aplicações civis e vigilância
Micro UAV	< 2	< 1500	< 1	< 20	Asa fixa Multi-rotor	Militar/Civil	Aplicações civis e vigilância
Nano UAV	< 0.025	< 100	< 1/2	< 0.5	Asa fixa Multi-rotor	Militar/Civil	Vigilância

Os MAVs são os mais comercializados para uso civil, porque detêm uma estrutura simples e um custo reduzido de obtenção e manutenção. São mais simples de transportar e descolar, capazes de transportar quantidades consideráveis de carga e são muito versáteis.

2.1.2. Sistemas de Comunicação

Normalmente quando se fala das estações de controlo dos UAV, fala-se em estações de controlo terrestres, no entanto também existem as estações de controlo aéreas e estações de controlo aquáticas mas estas apenas são utilizadas para aplicações militares [2].

As estações de controlo apresentam uma combinação de *hardware* e *software*, dos quais o operador é capaz de controlar e monitorizar o UAV. O operador tem à sua disposição uma interface que recebe a telemetria do veículo e envia os comandos para o mesmo. A informação de telemetria é enviada do veículo para o operador e consiste em informação de altitude, velocidade, posição de GPS, níveis de bateria e outras informações dependendo da quantidade de sensores que o veículo contém. Então uma estação de controlo apresenta as seguintes capacidades:

- Telemetria e visualização de vídeo, com medidores e indicadores visuais do estado atual do UAV;
- Transmissão de comandos, com possibilidade de selecionar diferentes modos de voo, iniciar e/ou interromper missões;
- Navegação por geolocalização, também conhecido como modo autónomo, que consiste em conduzir o veículo escolhendo uma localização geográfica num mapa;
- Salvar registos e toda a telemetria para posterior análise;
- Configuração dos parâmetros do controlador de voo.
- Fornecer um mecanismo de transmissão dos sinais para comando fiável, devido a possíveis erros de transmissão.

O GPS é uma unidade muito importante num UAV porque fornece um instrumento de navegação para automatizar missões. É utilizado para monitorar a posição do veículo, mas também pode ser usado para o piloto automático do UAV e navegar automaticamente, usando coordenadas fornecidas pelos sistemas GNSS.

2.2. Sistema GNSS

Os sistemas GNSS são de um nível de complexidade elevada, pois trata-se de uma variedade de subsistemas diferentes a funcionar em conjunto. Por mais evidente que sejam os satélites a parte mais «visível» do sistema, para o funcionamento correto são necessárias infraestruturas terrestres, para que os sinais cheguem corretamente aos terminais dos utilizadores. Os utilizadores apenas têm acesso a um *link* no sistema, o *downlink* vindo dos satélites da constelação, sendo assim impossível que o sistema tenha um limite de utilizadores, pois o sinal de *downlink* é transmitido em *broadcast* [6].

2.2.1. Descrição do Sistema GPS

O sistema GPS é o único explorado nesta dissertação, visto ter sido o primeiro sistema a entrar em funcionamento, sendo, nos dias de hoje, o sistema com mais utilização.

A arquitectura do mesmo é dividida em vários segmentos: segmento do utilizador, ou *user segment*, que é constituído por todos os tipos de recetores de sinais GNSS GPS, um segmento do espaço, ou *space segment*, que reúne todos os satélites constituintes da constelação, e um segmento terrestre, ou *ground segment*, que é responsável por monitorizar, controlar e atualizar as estações [7], [8].

Segmento terrestre

Principais funções são:

- Monitorizar os satélites;
- Definir as órbitas para cada satélite para prever os dados das efemérides e do almanac;
- Determinar a altitude e localização de cada satélite e enviar para os satélites as correções para se manterem na órbita correta [7], [8].

Segmento do espaço

Constelação base composta por 24 satélites, constituída por seis órbitas quase circulares com uma inclinação de 55° referenciadas com o plano equatorial a uma altitude de 20183 km. Cada satélite consegue fazer uma circunferência ao torno do planeta em exatamente 11 horas 57 minutos e 58 segundos. Tornando assim possível em qualquer posição no planeta obter quatro satélites em linha de vista para condições atmosféricas normais. A constelação foi oficialmente declarada operacional em 1995.

Principais funções são:

- Receber do segmento terrestre as correções de rota e corrigir a mesma;
- Distribuir os sinais de GNSS [7], [8].

Segmento do utilizador

É constituído por uma grande variedade de recetores, desde militares, recetores produzidos em massa para uso civil e até para fins científicos. As suas principais funções são:

- Receber os sinais correspondentes a sistemas GNSS e avaliar o seu estado;
- Realizar as medidas do tempo de propagação;
- Realizar as medidas devido ao efeito de Doppler;
- Calcular a localização do recetor;
- Calcular a velocidade do terminal e providenciar medida temporal [7], [8].

2.2.2. Frequências, Códigos e Modulações GPS

A escolha de uma frequência a ser utilizada é um processo complexo com muitas questões técnicas. A principal preocupação é utilizar uma gama de frequências com a capacidade de propagação pela atmosfera terrestre mantendo uma boa qualidade. Após as mesmas serem escolhidas já existe a possibilidade de começar a desenvolver os transmissores e recetores para os sistemas que as vão utilizar. O sistema GPS possui 3 gamas de frequências, L1, L2 e L5 sendo L2 e principalmente L5 frequências ainda com algum desenvolvimento [9]. Nesta dissertação foi apenas abordada a gama de frequências L1.

L1 é gama mais utilizada mundialmente, funciona na frequência 1.57542GHz e o seu acesso é por CDMA Tabela 2-2. É constituída por três sinais diferentes: código C/A, código P e código M.

Código C/A: tornou-se no código mais utilizado e importante, código destinado a uso civil, pois foram desenvolvidas muitas soluções no mercado para utilização do sistema GPS. Tem como característica um milissegundo de comprimento a um *chipping rate* de 1.023 Mbps [10].

Código P: é um código de precisão e com usos militares, muitas vezes é utilizado o código P(Y) no lugar do código P quando se utilizam os sistemas de anti-s*poofing*. Tem como características sete dias de comprimento a um *chipping rate* de 10.23Mbps e garantir confidencialidade e autenticação [10].

Código M: foi concebido exclusivamente para usos militares e para eventualmente vir a substituir o código P e P(Y). Tem melhores características para resistir ao *jamming* e garante melhor desempenho e mais flexibilidade que o código P(Y) [10].

Tabela 2-2 GPS L1 carateristicas técnicas [11]

Nome do Serviço	**C/A**	**Código P(Y)**	**Código M**
Modulação	BPSK(1)	BPSK(10)	$\mathrm{BOC}_{\sin}(10,5)$
Comprimento do Código	1023	6.19×10^{12}	_____

Pode-se então concluir que o GPS, na gama de frequências L1, divide-se em dois tipos de sinais GNSS transmitidos:

- Primeiro, sinal aberto para uso civil;
- Segundo, mais robusto e com maior precisão, para uso militar.

Para esse feito são utilizados os códigos acima descritos e foram criados dois serviços: *Standard Positioning Service* (SPS) e o *Precise Positioning Service* (PPS), que se enquadram, respetivamente, com os dois tipos de sinais diferentes.

SPS trata-se de um serviço que pode ser acedido por qualquer utilizador normal (civil). É baseado no código C/A *Coarse/Acquisition sequence*. Enquanto que o serviço PPS está apenas acessível a utilizadores autorizados (militar). Utiliza código P, que permite uma maior precisão da localização no globo, mas impossível para os terminais civis conseguirem obter uma localização em tempo real. Por fim o mesmo código P está encriptado, ou seja, código P(Y), asegurando assim que apenas os terminais autorizados conseguem desencriptar e utilizar esses sinais.

2.2.3. Trilateração e Localização com GPS

Para obter a localização de um terminal recetor utilizando os sistemas GPS de forma simplificada, é relativamente simples. Sabendo pelo menos as distâncias a que o recetor se apresenta de três satélites distintos, Figura 2-1, e conjuntamente a localização dos próprios satélites, é possível determinar a localização do utilizador, utilizando técnicas de trilateração [12].

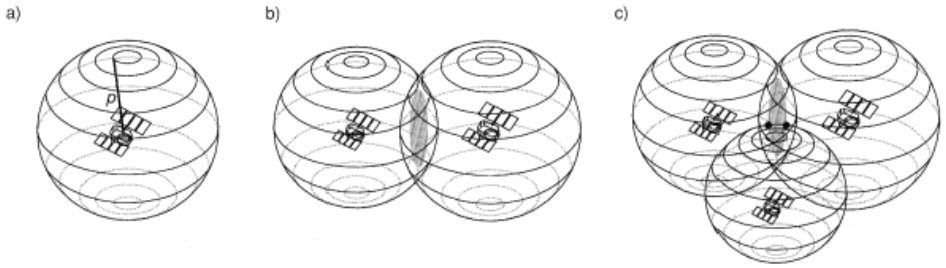

Figura 2-1 Trilateração e localização por satélite [12]

Assumindo que o relógio do recetor está sincronizado com os relógios dos satélites, que a ionosfera e a troposfera não implicavam um ligeiro atraso no sinal e a inexistência do ruído de transmissão, com apenas um satélite seria possível determinar a distância do recetor ao satélite sobre uma esfera centrada no satélite em que o raio da esfera será a distância do recetor ao satélite, como se pode ver na ilustração a) da Figura 2-1. Com a adição de mais um satélite na equação, ilustração b) da Figura 2-1, o recetor ao receber sinal de um segundo satélite, também se encontra a uma distância do valor do raio da esfera centrada no segundo satélite. Estas duas esferas vão se intercetar e formar uma circunferência, e nos seus limites serão todos os pontos possíveis em que o recetor se poderá encontrar. Mesmo assim trata-se de uma localização com erro muito elevado, levando a que pelo menos sejam necessários três satélites para uma localização com precisão. Ao adicionar um terceiro satélite os pontos em que o recetor poderá estar serão reduzidos a dois, como se vê na ilustração c) da Figura 2-1, em que normalmente um desses dois pontos se encontra no espaço, levando a ser excluído e assim apenas resultar um ponto da localização do recetor. Quantos mais satélites o recetor tem em linha de vista maior será a precisão da localização do mesmo [12].

2.3. *Spoofing*

O *spoofing*, de modo geral, é uma prática fraudulenta ou maliciosa na qual a comunicação é enviada a partir de uma fonte desconhecida, disfarçando-se de fonte conhecida pelo destinatário. A utilização do *spoofing* é mais comum em mecanismos e redes de comunicação que não possuem alto nível de segurança, como acontece no sinal GPS civil, em que o mesmo não possui qualquer tipo de encriptação ou autenticação, de forma a proteger ou a comprovar que o sinal provém de uma fonte fidedigna ou a não ocorrência de repúdio do sinal.

2.3.1. Tipos de *Spoofing*

Existem variados sistemas ou serviços em que se podem aplicar técnicas de *spoofing*, nomeadamente:

- TCP/IP *spoofing*;
- *Spoofing* email;
- *Spoofing* GPS;
- MAC *spoofing*;
- DNS *spoofing*.

2.3.2. GPS *Spoofing*

Todos os aparelhos que utilizam rádio frequência para comunicação, têm a vulnerabilidade de que a informação transmitida se encontra disponível para todos no ar, meio onde se propagam as ondas eletromagnéticas. Mas os sistemas que têm menores níveis de proteção da informação estão mais expostos aos ataques abaixo descritos:

***Spoofing* simples:** técnica capaz de gerar sinais GNSS falsos. Pode ser posto em prática utilizando:

- *Hardware* de baixo custo para receber e reproduzir os sinais GNSS. Os simuladores de sinais personalizados podem ser inseridos na configuração para controlar e modificar alguns dos parâmetros da transmissão.
- *Hardware* comercial, são normalmente caros e mais complexos de manipular, mas tratam-se de plataformas com maiores capacidades de processamento e transmissão de sinais eletromagnéticos [13].

***Spoofing* intermediário:** neste caso, o atacante gera de forma sincronizada sinais falsos tentando, simultaneamente, atacar cada canal do recetor alvo, executando o alinhamento de fase de código entre sinais recebidos falsos e genuínos [14].

***Spoofing* com múltiplas antenas transmissoras:** técnica avançada, utilizada principalmente contra um recetor de múltiplas antenas, em que cada antena transmissora do atacante combina com uma antena recetora correspondente [15].

***Spoofing* com antenas de alto ganho:** ataque aprimorado, com base no uso de antenas com ganho suficiente para separar os sinais GNSS do ruído, incluindo, por exemplo, *chips* de código desconhecidos ou encriptados [16].

***Spoofing* sofisticado:** pode ser realizado por um conjunto de atacantes coordenados e sincronizados, capazes de atacar o recetor da vítima de forma organizada. Além disso, têm informações de posição tridimensional sobre os centros de fase das suas antenas e o centro de fase da antena da vítima, podendo assim ultrapassar contramedidas complexas, como as que se baseiam na estimativa do ângulo de chegada [14].

2.3.3. Trabalhos Relacionados GPS *Spoofing*

Daniel P. Shepard, Jahshan A. Bhatti, e Todd E. Humphreys do departamento de Engenharia Aeroespacial da Universidade de Austin Texas [17] estudaram e exploraram as vulnerabilidades dos sistemas de GPS em drones, com o intuito de desviar ou ganhar o controle sobre as aeronaves.

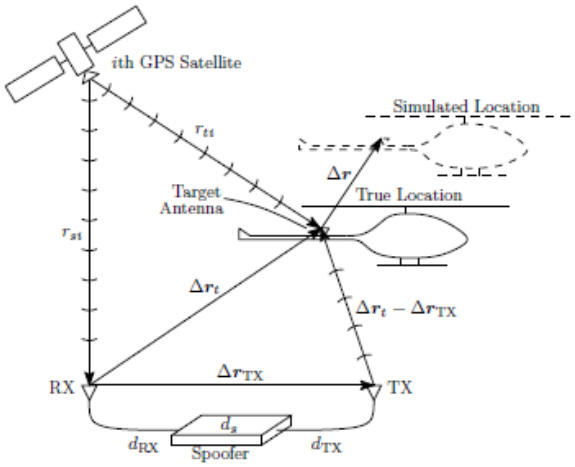

Figura 2-2 Demonstração dos componentes, geometria e atrasos no spoofing gps [17]

O atacante, ou *spoofer*, gera sinais falsos de GPS para todos os sinais autênticos que consegue receber. Os sinais falsos recebidos pelo recetor do drone chegam alinhados com os verdadeiros sinais de GPS, tendo em conta os tempos de atraso e fases, como se ilustra na Figura 2-2 [17].

Os mesmos autores do estudo anteriormente descrito demonstraram [18] que é possível realizar o *spoofing* de GPS civil a veículos aéreos não tripulados, drones.

O sistema utilizado nos testes foi uma versão melhorada do sistema original referenciado em [19], desenvolvido pelo Laboratório de Rádionavegação da Universidade de Austin Texas Figura 2-3. Tratando-se de realizar o *spoofing* ao sistema de GPS civil apenas interessa ao sistema receber e analisar as gamas de frequência L1, L2 e sinais que utilizam o código C/A.

Inicialmente o sistema recebe os sinais autênticos de GPS e utiliza-os para adaptar os seus sinais falsos com as informações provenientes dos sinais verdadeiros.

Figura 2-3 GPS spoofer civil [18]

Como se pode ver pelo diagrama de blocos na Figura 2-4 o *control module* lê as componentes dos sinais GPS (fase do código, fase da portadora e efeito de Doppler) através de um recetor. Essas componentes são modificadas utilizando modelos de medição lineares e utilizadas para criar sinais GPS falsos.

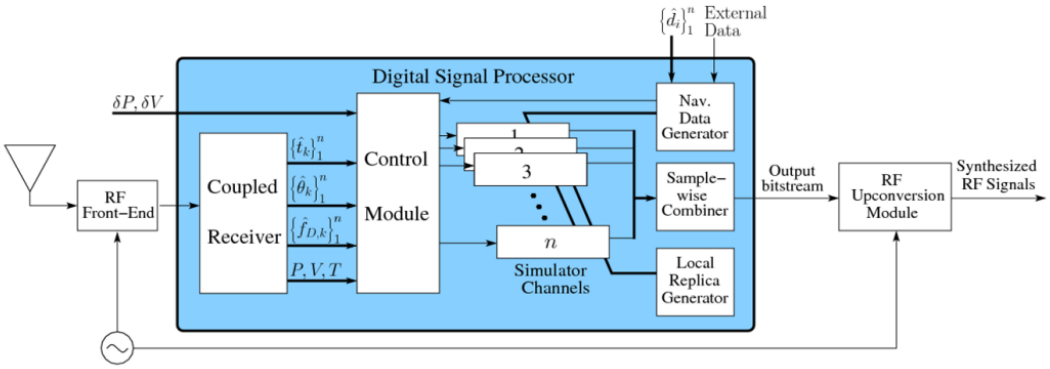

Figura 2-4 Diagrama de blocos sistema de spoofing [18]

Para a demonstração do funcionamento do sistema de *spoofing* a vítima foi um veículo aéreo não tripulado Hornet Mini UAV (Figura 2-5), propriedade da Universidade. Veículo com capacidade de navegação por satélite civil. O esquema utilizado para realizar os testes de *spoofing* foi o mostrado na Figura 2-6 com uma distância entre emissor (*spoofer*) e recetor (Hornet Mini) de aproximadamente 650 metros.

Figura 2-5 Hornet Mini UAV [18]

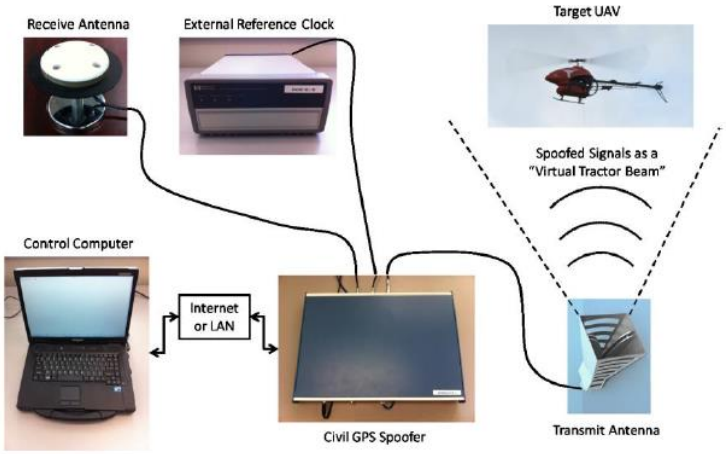

Figura 2-6 Esquema utilizado para testes de spoofing [18]

O Hornet Mini foi comandado manualmente para a posição de 650 metros de distância e elevado para uma altitude de aproximadamente 12 metros. Após encontrar-se no local foi alterado o comando para navegação por satélite e para se manter na mesma posição. De seguida iniciaram a emissão de sinais GPS falsos certificando-se que as fases de código estavam alinhadas com os sinais originais e rapidamente obtiveram o controle sobre a aeronave [18]. A única desvantagem é que o *spoofer* necessita de contato visual com a aeronave para não a aterrar ou desviar de forma descontrolada.

Outro exemplo de sistema desenvolvido para realizar o *spoofing* de GPS é o apresentado em [20], criado pela UnicornTeam, uma equipa de investigadores de segurança que tem como principal foco a segurança de sistemas que utilizam tecnologias rádio. Esta é uma equipa que integrou o grupo da DEF COM 23 *vendors*, ou seja, fez parte das equipas que participaram nas conferências da DEF COM 23 [21].

O sistema foi criado utilizando *Hardware* SDR e dois *Softwares*, simulador de sinais GPS chamado gps-sim-hackrf e recetor de sinais GPS com o nome de GNSS-SDR. Inicialmente para os primeiros testes foram gravados sinais GPS utilizando a placa USRP

B210 Figura 2-7. Após a gravação dos sinais autênticos de GPS os mesmos foram retransmitidos utilizando a placa bladeRF Figura 2-8.

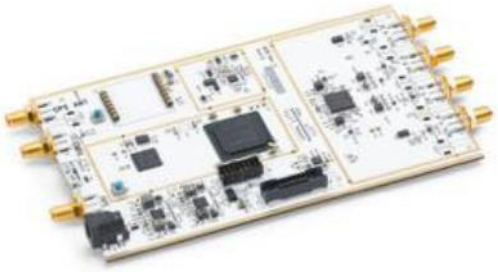

Figura 2-7 USRP B210 [20]

Figura 2-8 bladeRF x40 [20]

Essa experiência de gravar e retransmitir sinais GPS foi atingida com sucesso pela confirmação de um terminal móvel, no caso um telemóvel Nexus 5, que recebeu a localização e tempo exatos de onde e quando a gravação teria sido realizada Figura 2-9.

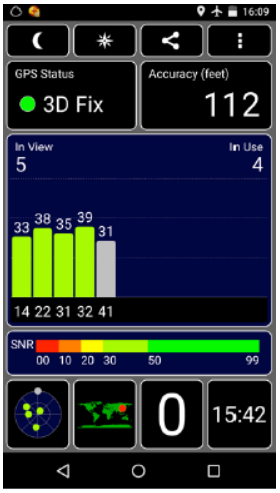

Figura 2-9 Recetor GPS smartphone Nexus 5 [20]

Mas o facto de terem conseguido gravar e retransmitir não era o ideal, então seguiram com a investigação de criar ou utilizar efemérides disponíveis e transmitir sinais GPS utilizando o *software* gps-sim-hackrf. O primeiro objetivo seria de descarregar a

informação atualizada das efemérides dos satélites, através do website: ftp://cddis.gsfc.nasa.gov/gnss/data/daily/. Tendo em conta todas as variáveis que influenciam a boa construção e receção do sinal (efeito de Doppler, forma de onda e tempo de transmissão), utilizando o *software* gps-sim-hackrf e ficheiro atualizado das efemérides dos satélites, conseguiram novamente realizar o *spoofing* do GPS ao mesmo telemóvel Nexus 5, mas também a um Samsung Note 3 e Iphone 6 Figura 2-10.

Figura 2-10 Samsung Note 3 e Iphone 6 [20]

Capítulo 3 - Implementação

Neste capítulo será apresentada a explicação do funcionamento do sistema de *spoofing* desde a sua projecção até à concepção final. Enumeração e descrição de todo o *hardware* e *software*. Descrição dos desenvolvimentos analíticos e algoritmos eficientes utilizados para determinação de localização do drone e percurso atual, e resolução de localização ou trajetória a simular. Também se faz um esclarecimento sobre os aspetos de comunicação utilizados em todos os subsistemas utilizados. E por fim apresentação detalhada dos circuitos electrónicos desenhados e implementados.

3.1. Descrição do Sistema

O sistema de *spoofing* desenvolvido nesta dissertação opera recorrendo a plataformas de prototipagem eletrónica de *hardware* aberto, plataformas de SDR e de *System on a Chip* (SoC) como processador central do sistema.

O objetivo principal, com a motivação associada à oportunidade identificada, baseia-se na implementação de um sistema móvel capaz de transmitir sinais de GPS falsos, e, indirectamente, ganhar o controle sobre o drone e proteger uma área ou local da intrusão do mesmo. Por outro lado, também seria fundamental considerar o sistema independente de sistemas terceiros, isto é, que fosse capaz de chegar ao seu objetivo principal sem necessitar de informação ou interação com outros sistemas.

Após investigação sobre sinais de GPS, verificou-se que seria necessário determinar a localização atual do drone para conseguir criar sinais GPS falsos o mais próximo dos sinais verdadeiros, e desviar ou tomar o controle do aparelho gradualmente sem que este detecte. Para tal seria necessário integrar sensores ao sistema, para que consegui-se determinar a localização do drone. Assim o sistema de *spoofing* móvel teria de ser capaz de determinar a localização do drone e a sua trajetória atual, através dos sensores e cálculos analíticos, criar sinais de GPS falsos de forma a desviar o aparelho da sua posição ou rota atual e transmitir esses sinais de forma diretiva sem interferir com terminais de sinais GNSS próximos.

3.2. Arquitectura

Elaborou-se um esquema geral do todo o sistema, representado pela Figura 3-1, para uma fácil perceção do seu funcionamento.

Como se verifica na Figura 3-1, como sistema central de comando e processamento tem-se um Raspberry Pi, (a) da Figura 3-1, onde se encontrará a decorrer o processo central que vai coordenar e distribuir as tarefas pelos outros componentes auxiliares.

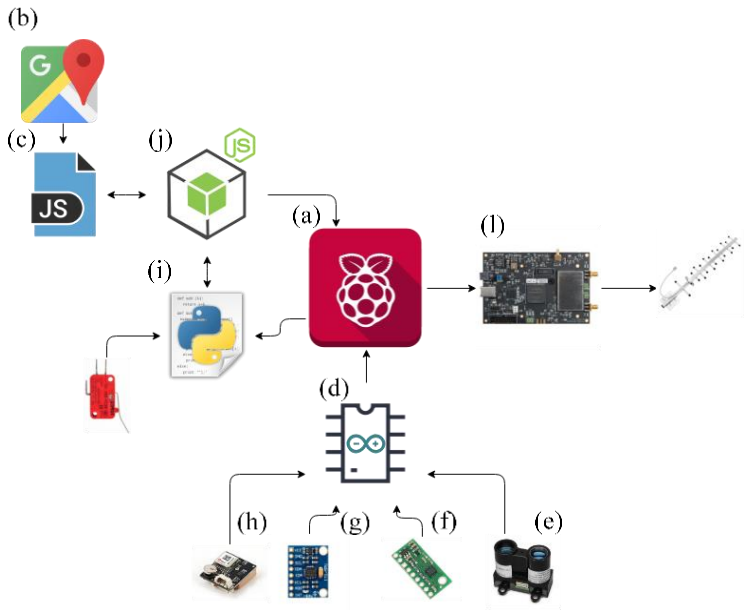

Figura 3-1 Esquema total do sistema de spoofing

Começando pelo canto superior esquerdo, (b) da Figura 3-1, definiu-se como fonte de mapas atualizados a Maps JavaScript API. API essa disponibilizada livremente pela Google, que permite utilizar funções já existentes e desenvolver uma interface gráfica que apresente os resultados do sistema e o seu estado atual. De seguida tem-se como recetor e processador da Maps JavaScript API um código JavaScript, (c) da Figura 3-1, uma vez que a mesma é disponibilizada para programação JavaScript. Código responsável por descarregar os mapas, calcular as localizações e/ou trajetórias falsas e criar as mensagens NMEA completas para mais tarde serem utilizadas na simulação dos sinais GPS.

Também elaborou-se código C++ no Arduino, (d) da Figura 3-1, para realizar a leitura dos sensores e recetor, respetivamente (e), (f), (g) e (h) da Figura 3-1, para a sensorização da arma. Esse código, em C++, fica responsável pela leitura individual dos valores dos sensores e recetor GPS, e enviar os valores dos mesmos para o Raspberry Pi, através da porta serie ligada diretamente entre o Raspberry e o Arduino.

Necessariamente programou-se código Python, (i) da Figura 3-1, que permite a leitura dos seletores de modo (*jamming* e *spoofing* estático ou dinâmico) e dos gatilhos da arma. Nesse código Python serão recebidos os valores dos sensores e recetor medidos no Arduino, através da leitura da porta serie. Valores esses que serão, posteriormente, encaminhados para o Nodejs, (j) da Figura 3-1, por forma a serem entregues ao código em JavaScript, para que possam ser utilizados nos cálculos que prosteriormente são descritos 3.4.

Relativamente à transmissão dos sinais GPS falsos, através da placa SDR bladeRF, (l) da Figura 3-1, onde através do sistema central, (a) da Figura 3-1, e do *software* simulador de sinais GPS, são criados e enviados para a bladeRF e transmitidos através de uma antena modelo Yagi diretiva e com ganho de transmissão.

3.3. *Hardware* e *Software*

3.3.1 Hardware

Para o funcionamento total do sistema de *spoofing* são utilizados e configurados variados *hardware's*.

Considerando que todo o sistema será integrado numa arma móvel, começamos pela sensorização da mesma. A arma terá na sua constituição 3 sensores e um recetor que complementam o funcionamento do sistema.

Sensor Lidar

Figura 3-2 Sensor Lidar-Lite v3 [22]

Sensor Lidar -Lite v3, Figura 3-2, possui um funcionamento similar ao de um radar, mas de forma mais seletiva, concentrando a leitura na largura de um feixe laser. É um

sensor capaz de emitir um feixe laser, a funcionar em 905 nanometros, Anexo C.1. *Datasheet* Lidar, e obter a distância de um objeto ou obstáculo através da reflexão do sinal no mesmo. É utilizado em muitas aplicações para deteção de obstáculos durante a navegação de veículos não tripulados, como no caso dos drones [23].

Sensor Pitch

Figura 3-3 MPU6050 sensor Roll,Pitch e Yaw [24]

A placa utilizada é a MPU6050, Figura 3-3, que contém um acelerómetro e giroscópio em circuitos MEMS que permite medir uma força ou mudança de posição do aparelho, Anexo C.2. *Datasheet* MPU6050. Esta aplicabilidade está presente em todos os *smartphones* da atualidade permitindo assim saber a orientação do smartphone e criar casos de uso para essa funcionalidade, como por exemplo a rotação automática do ecrã. É possível determinar os ângulos de 3 eixos Roll, Pitch e Yaw, representados na Figura 3-4 com as equações explicadas em [25]. Mas para o objetivo procurado é apenas necessária a medição do Pitch, possibilitando assim medir o ângulo que a arma faz face ao plano horizontal na terra, como será explicado posteriormente.

Figura 3-4 Roll, Pitch e Yaw de smartphone [25]

Sensor Bússola

Figura 3-5 Pololu lsm303d ecompass [26]

Trata-se da placa pololu lsm303d, Figura 3-5, que possui um magnetómetro, sensor capaz de sentir o campo magnético, que é utilizado em conjunto com um acelerómetro por forma a auto corrigir-se, permitindo calcular o direcionamento da bússola com compensação de tilt, como explicito nas equações em [27] e Anexo C.3. *Datasheet* lsm303d, gerado pela inclinação da arma. Com a utilização desta placa é possível determinar a orientação da arma, ou seja, é possível verificar a direcção para a qual a mesma se encontra apontada.

Recetor de GPS

Figura 3-6 Recetor GPS MAX-7Q-0-000 [28]

Foi utilizado o recetor ublox MAX-7Q-0-000, Figura 3-6, para localização da própria arma. Através da receção de sinais GNSS, este recetor determina qual a posição do recetor no planeta.

Arduino Uno

Figura 3-7 Arduino Uno Rev3 [29]

Os 4 componentes anteriormente descritos estão todos ligados a um Arduino Uno Rev3, Figura 3-7, com as seguintes características: 14 pinos de entrada / saída digital (dos

quais 6 podem ser usados como saídas PWM), 6 entradas analógicas, um cristal de quartzo de 16 MHz, uma conexão USB, um conector de energia, um conector ICSP e um botão de reset, [29].O mesmo procede à leitura e obtenção dos dados dos sensores e do recetor GPS e de seguida emite os resultados já calibrados e tratados para o Raspberry pi 3 model B através de comunicação Serie.

Raspberry Pi

Figura 3-8 Raspberry pi 3 model B [30]

Tem-se como núcleo principal de processamento e gestão do sistema central, um Raspberry pi 3 model B, Figura 3-8. Semelhante a um computador de pequenas dimensões, visto que o mesmo possui todas as capacidades de um computador normal. O Raspberry indicado possui as seguintes características: processador 1.2GHz 64-bit quad-core ARM CortexA53, 1GB de memória RAM, wireless integrado WiFi 802.11n e Bluetooth 4.1, 40 pinos de entradas e saídas e retro compatibilidade com Raspberry Pi 1 e 2, [30]. Este aparelho permite uma rápida performance durante a execução do sistema de *spoofing*, mantendo níveis acessíveis de consumo energético.

BladeRF x40

Figura 3-9 bladeRF x40 [31]

Para transmissão dos sinais de GPS falsos é utilizada uma plataforma de *Software Defined Radio* (SDR), mais concretamente é utilizada a placa bladeRF x40, Figura 3-9. Apresenta uma arquitetura de rádio básica, mas capaz de abranger técnicas de modulação e esquemas básicos de codificação de telecomunicações. Possui capacidade de

comunicação USB 3.0 e um chip FPGA completamente programável para desenvolvimento de sistemas mais rápidos [32]. Foi efetuada a escolha da placa bladeRF tendo em conta os seus baixos consumos energéticos e a sua versatilidade, visto ser para integrar num sistema móvel.

3.3.2 Software

Durante o desenvolvimento do trabalho apresentado na dissertação foram utilizados e originados dois tipos de *software*, respectivamente *software* utilizado para o desenvolvimento e *software* criado e utilizado durante a execução do sistema de *spoofing* de GPS, como é demonstrado na Figura 3-10.

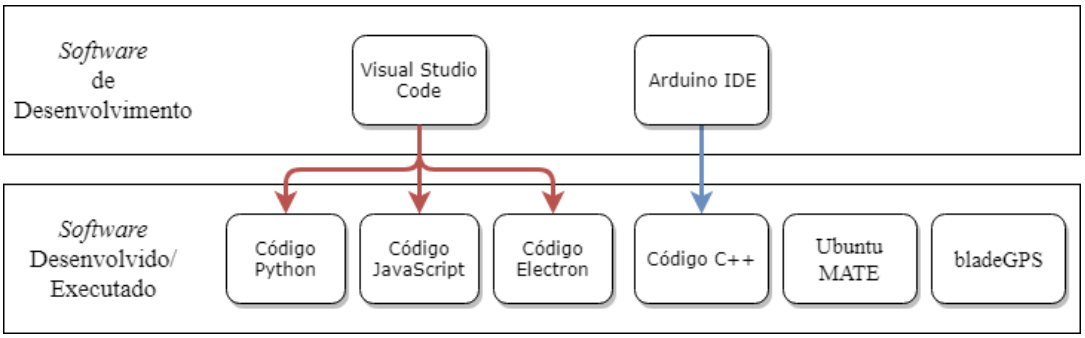

Figura 3-10 Software utilizado para desenvolvimento e execução do sistema

Visual Studio Code e Arduino IDE

Figura 3-11 Visual Studio Code e Arduino IDE [29], [33]

Para o desenvolvimento do sistema foi utilizado o editor de código fonte Visual Studio Code, desenvolvido pela Microsoft, e o Integrated Development Environment (IDE) *Open-Source* Arduino *Software*, Figura 3-11. Desta forma foi possível desenvolver os variados códigos utilizados no sistema final, nomeadamente códigos Python, JavaScript, HTML e Nodejs no Visual Studio Code e o código C++ no IDE da Arduino, [29], [33].

Electron

Figura 3-12 Framework Electron [34]

Após a integração de todos os subsistemas e finalizada a programação de todos os códigos que integram o produto final, foi criada uma aplicação utilizando a *framework* Electron, Figura 3-12, na programação Nodejs. Essa *framework* permite criar aplicações *desktop*, com ou sem interface gráfica, para múltiplas plataformas, ou seja, Windows, Linux e Mac, utilizando tecnologia *web* como JavaScript, HTML e CSS. A mesma foi criada por Cheng Zhao mas agora é desenvolvida pelo GitHub e está em constante progresso, [34]. Assim, poder-se-á afirmar que a aplicação desenvolvida com a *framework* Electron trata-se de um dos *softwares* utilizados durante a execução do sistema.

Sistema Operativo Ubuntu MATE

Figura 3-13 Ubuntu MATE 16.04 [35]

Outro *software* utilizado durante a execução do sistema é o sistema operativo instalado no Raspberry pi 3 model B. Sistema operativo Ubuntu MATE 16.04, Figura 3-13, construído a partir de um núcleo Linux e baseado em Debian, a imagem do mesmo foi criada por Martin Wimpress e Rohith Madhavan e otimizada para uma limpa e rápida utilização em sistemas Raspberry pi 2 e 3, [35]. O mesmo foi escolhido com base na performance do Raspberry pi utilizado e da possível integração da aplicação em Electron ser compatível com o mesmo.

bladeGPS

É utilizado também o *software open source* bladeGPS, desenvolvido pela OSQZSS (Open Source Quasi-Zenith Satellite System), [36], que permite simular sinais GPS em tempo real em conjunto com a placa bladeRF. *Software* esse que teve de ser adaptado com umas ligeiras alterações no *baudrate* do *output*, pois excedia os limites do USB do Raspberry pi 3 model B.

Por fim, e não menos importantes, foram utilizados outros *softwares open source* complementares que permitiram, nomeadamente a instalação do sistema operativo no cartão microSD no Raspberry pi e a instalação de todas as componentes necessárias para o correto funcionamento da aplicação Electron no Raspberry pi 3 e, por sua vez, no sistema operativo Ubuntu Mate.

3.4. Desenvolvimento Analítico

Inicialmente foram estudados todos os processos pelos quais o sistema de *spoofing* iria passar, para que no fim se atingisse os objetivos principais e um bom funcionamento do mesmo.

3.4.1. Determinação da Localização do Drone

Inicialmente, foi analisado como seria possível determinar a localização do drone, visto que o sistema de *spoofing* necessita desse fator para uma construção de sinais GPS falsos mais próximos dos reais, mantendo assim o controle da situação, desviando o aparelho de forma controlada e com previsão do seu destino final.

Ficou delineado que a determinação da localização do drone seria feita pela própria arma. A mesma terá, na sua constituição, 3 sensores e um recetor que em conjunto levarão a esse efeito, nomeadamente: sensor Lidar ou de telemetria (medição da distância do drone), sensor de *tilt* ou *pitch* (medição da inclinação da arma), sensor de bússola (medição da orientação da arma) e recetor de sinais GPS (deteção do posicionamento da arma).

Observando a Figura 3-14 e a Figura 3-15 abaixo apresentadas, e com uma breve explicação analítica, fica simples de compreender como a arma é capaz de determinar a localização do drone com alguma robustez.

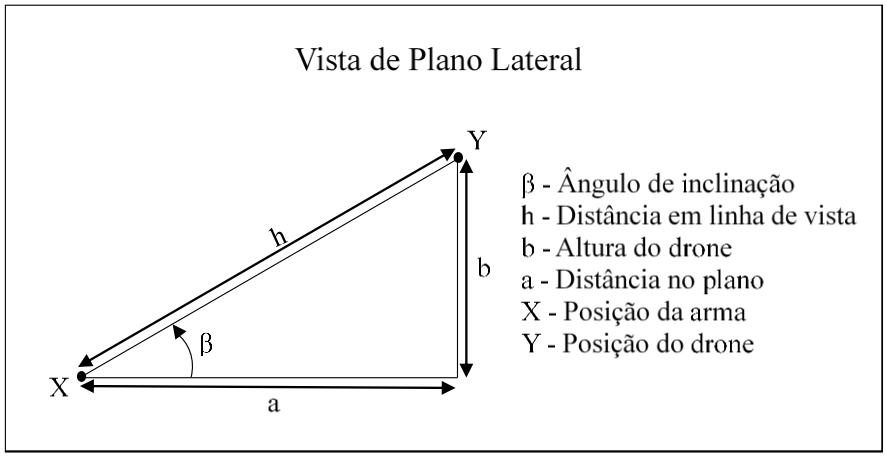

Figura 3-14 Vista de plano lateral cálculos analíticos

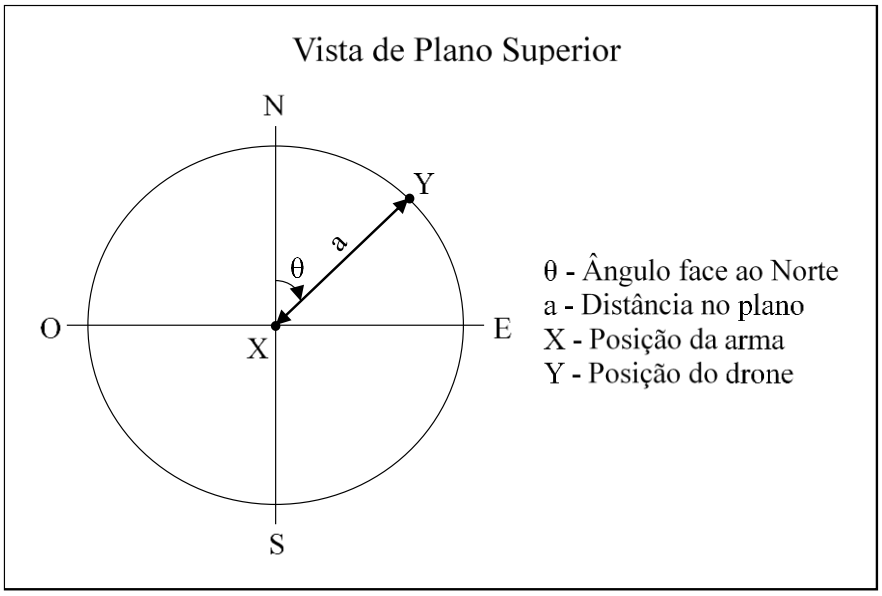

Figura 3-15 Vista de plano superior cálculos analíticos

Associando as medições dos sensores aos ângulos e medidas presentes nas Figura 3-14 e Figura 3-15 a localização determina-se da seguinte forma:

- Distância do drone em linha de vista: Sensor Lidar = *h*;
- Ângulo de inclinação da arma: Sensor de *Pitch* = β;
- Ângulo de orientação da arma: Sensor de Bússola = θ;
- Localização da arma: recetor GPS = X.

Após obter os quatro valores acima, dos sensores e recetor, determina-se o ponto Y (localização do drone no planeta). Sabendo os valores de 'h' e de 'β' calcula-se 'a' através da Equação 1 e 'b' através da Equação 2 abaixo apresentadas:

$$a = \cos(\beta) \times h \quad (1)$$

$$b = \sin(\beta) \times h \quad (2)$$

De seguida, obtendo-se o resultado da distância do drone no plano horizontal (variável 'a') associado ao valor exportado pelo sensor de bússola (variável 'θ' Figura 3-15), o valor médio do raio do planeta Terra [37] e, sabendo a localização da própria arma (variável 'X' Figura 3-14 e Figura 3-15) determinou-se a localização do drone (variável 'Y' Figura 3-14 e Figura 3-15). Sabendo que a variável 'X' se divide em duas componentes latitude (LatX) e longitude (LongX) calculou-se as duas componentes da variável 'Y', latitude (LatY) e longitude (LongY) através das Equações 3 e 4 referenciadas em [38], respetivamente.

- LatX - Valor de latitude da localização da arma;
- LongX - Valor de longitude da localização da arma;
- LatY - Valor de latitude da localização do drone: LatY;
- LongY - Valor de longitude da localização do drone: LongY;
- R_{earth} - Valor médio do raio do planeta Terra (6378.137) km.

$$\text{LatY} = \text{asin}\left(\sin(\text{LatX}) \times \cos\left(\frac{\frac{a}{1000}}{R_{earth}}\right) + \cos(\text{LatX}) \times \sin\left(\frac{\frac{a}{1000}}{R_{earth}}\right) \times \cos(\theta)\right) \quad (3)$$

$$\text{LongY} = \text{LongX} + \text{atan}\left(\frac{\cos\left(\frac{\frac{a}{1000}}{R_{earth}}\right) - \sin(\text{LatX}) \times \sin(\text{LatY})}{\sin(\theta) \times \sin\left(\frac{\frac{a}{1000}}{R_{earth}}\right) \times \cos(\text{LatX})}\right) \quad (4)$$

De referir que o valor de 'a' e 'R_{earth}' necessitam de estar na mesma escala (quilómetros) e todos os ângulos devem ser introduzidos nas fórmulas em radianos e não em graus. Finalmente, utilizando todos os sensores e recetor de GPS, obteve-se a

localização do drone no ponto 'Y' (latitude e longitude) e a sua altura acima da arma pelo valor de '*b*', Figura 3-14.

3.4.2. Desvio do *Unmanned Aerial Vehicle*

Com o posterior conhecimento da localização do drone, através do processo anteriormente descrito, atingiu-se o próximo processo que atua no sistema de *spoofing* de GPS. Estudando os tipos de funções disponíveis no simulador bladeGPS, encontrou-se um ponto que é explorável para o fim que o sistema de *spoofing* necessita, ou seja, encontrou-se forma de construir as mensagens GPS iguais às mensagens reais, conhecendo, à partida, a localização atual do drone.

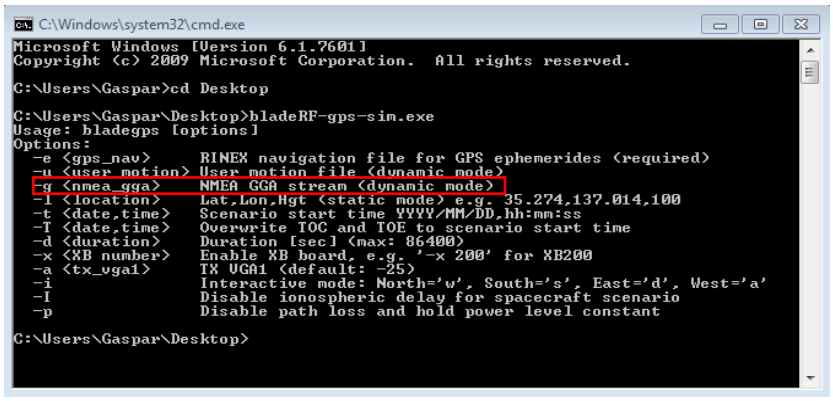

Figura 3-16 Interface do software bladeGPS

Uma das funções disponibilizadas pelo simulador bladeGPS é de utilizar mensagens NMEA, marcação a vermelho na Figura 3-16, para a simulação de sinais GPS dinamicamente, possibilitando uma maior liberdade na construção das mesmas, com o intuito de conseguir desviar o drone da zona proibida ao seu voo.

As mensagens NMEA foram desenvolvidas pela National Marine Electronics Association, que desenvolveu especificações que definem a interface entre vários sistemas e equipamentos eletrónicos marítimos. A comunicação para recetores de sinais GPS está definida nessas especificações [39].

A maioria dos programas de computador que providenciam informação de posicionamento em tempo real entendem e esperam receber informação em formato NMEA. A ideia das mensagens em formato NMEA é de enviar uma linha de dados chamada de *sentence* que é totalmente independente das linhas posteriores e anteriores,

contendo as informações na *sentence* formatadas conforme a categoria de dispositivo que as vai receber, indicado por um prefixo de duas letras. No caso dos recetores GPS o prefixo é *Global Positioning* (GP) Tabela 3-1, [40].

Tabela 3-1 NMEA 2.0 [41]

GPAPB	Piloto Automático B
GPBOD	Origem para destino – antes de G-12's não transmitir
GPGGA	Dados correção
GPGLL	Dados Lat/Lon – antes de G-12's não transmitir
GPGSA	Dados gerais de receção via satélite
GPGSV	Informação detalhada dos Satélites
GPRMB	Dados mínimos recomendados ao seguir um percurso
GPRMC	Dados mínimos recomendados
GPRTE	Direcionar dados, somente quando há uma rota ativa
GPWPL	Dados do ponto de referência, somente quando há rota ativa

Cada *sentence* começa pelo caracter '$' e acaba com '*' e o valor do *checksum* (representado por dois números hexadecimais). O *checksum* é calculado com uma operação XOR de todos os caracteres entre '$' e '*'. Toda a informação é contida numa única linha com os vários dados separados por virgulas, e representada em texto ASCII. Nunca poderá ultrapassar os 80 carateres por *sentence*. O primeiro dado consiste numa palavra que define o tipo de dados encontrados na *sentence*. Cada tipo de dados tem a sua própria interpretação e é definido no padrão NMEA. A *sentence* GGA fornece dados de correção GPS [41]. Na Figura 3-17 apresenta-se um exemplo de uma *sentence* GPGGA com definição de cada dado presente na mesma.

$GPGGA,090000.00,3845.19975115,N,00910.36577529,W,1,05,2.87,160.00,M,-21.3213,M,,*6F

Em que:

GGA *Global Positioning System Fix Data*

090000 *Fix taken at* 09:00:00 UTC

3845.19975115,N Latitude 38° 45.1997' N

00910.36577529,W Longitude 9° 10.3657' W

1 *Fix quality*: 0 = *invalid*

1 = GPS *fix* (SPS)

2 = DGPS *fix*

3 = PPS *fix*

4 = *Real Time Kinematic*

5 = *Float* RTK

6 = *estimated (dead reckoning)*

7 = *Manual input mode*

8 = *Simulation mode*

05 *Number of satellites being tracked*

2.87 *Horizontal dilution of position*

160 *Altitude, Meters, above mean sea level*

00 *Height of geoid (mean sea level) above WGS84 ellipsoid*

(empty field) time in seconds since last DGPS update

(empty field) DGPS station ID number

*6F *the checksum data, always begins with **

Figura 3-17 Estrutura de uma sentence GPGGA

Desta forma, sabendo como são formadas as *sentences* nas mensagens NMEA, o processo de construção de sinais para o desvio da posição do drone fica sintetizado na elaboração de uma mensagem NMEA com uma sequência de *sentences* que indique uma falsa localização atual ao veículo, que leva a que o mesmo tende para corrigir a sua posição atual e, assim, altere a sua posição real para uma posição fora da área restrita, Figura 3-18.

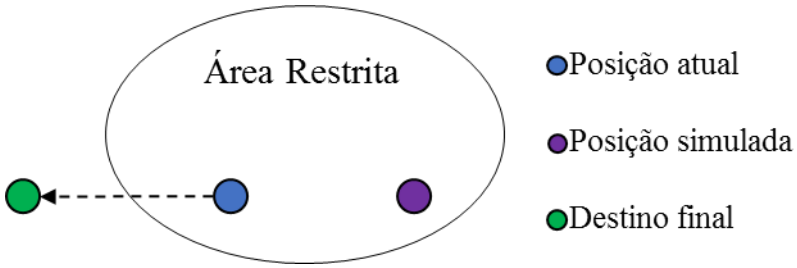

Figura 3-18 Ilustração de exemplo de spoofing estático

A formação das *sentences* também foi construída de forma dinâmica, ou seja, simulando uma localização em movimento, fazendo com que o drone corrija a sua rota em continua deslocação, mas com uma rota que o encaminhe para a área de aterragem Figura 3-19. Para isso, é necessário retirar duas localizações do drone através das medições dos sensores e recetor da arma, como explicado anteriormente com o auxílio da Figura 3-14 e Figura 3-15, pois com essas duas localizações destintas (marcações a vermelho na Figura 3-20) e o intervalo de tempo entre as mesmas, o sistema é capaz de determinar não só o sentido da rota do drone como a sua velocidade de deslocação Figura 3-19.

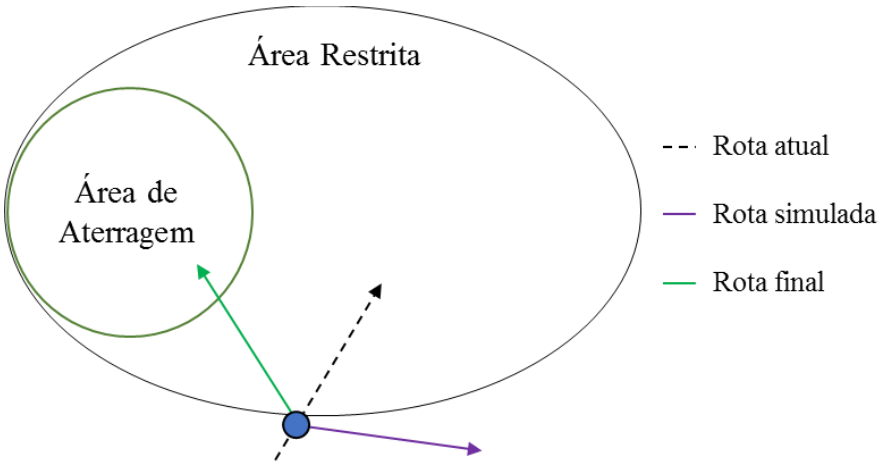

Figura 3-19 Ilustração de exemplo de spoofing dinâmico

Funções utilizadas da Maps JavaScript API e a sua aplicabilidade no sistema de *spoofing*:

- *computeDistanceBetween()*: calcular a distância entre a segunda localização do drone, obtida pelos sensores da arma, e a área de aterragem, "Rota final" da Figura 3-19;
- com*puteHeading()*: determinar os ângulos α_1 e α_2, Figura 3-20, respectivamente, ângulo da reta do precurso original do drone ("Rota atual" na Figura 3-20) e ângulo da reta entre drone e área de aterragem ("Rota final" na Figura 3-20) face ao norte;
- *computeOffset()*: dado uma determinada direção α_3, Figura 3-20, determinada através da Equação 5, um local de origem (última localização do drone através dos sensores da arma) e a distância a percorrer (determinada através da função

computeDistanceBetween), calcula-se as coordenadas de destino, ou seja, a "Rota simulada" Figura 3-20.

Utiliza-se a Equação 6 para calcular a velocidade instantânea do drone. Com todos essas funções e equações utilizadas e descritas anteriormente foi possível criar uma rota contrária à original do drone, que fará com que o mesmo se desvie e se desloque para uma zona reservada ao "abate" do aparelho.

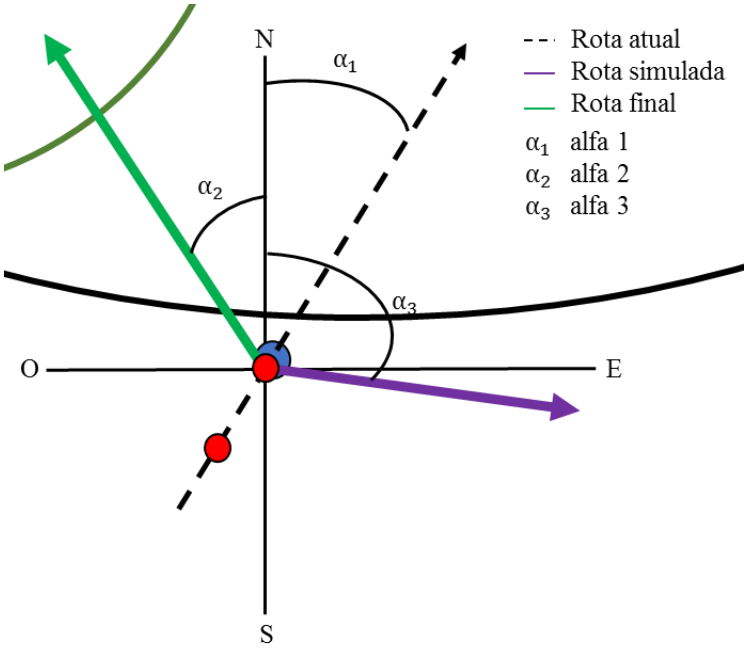

Figura 3-20 Ângulos e calculos analíticos de spoofing dinâmico

$$\alpha_3 = (\alpha_1 - \alpha_2) + \alpha_1 \quad (5)$$

- *D* - Distância percorrida entre localizações (metros);
- *T* - Intervalo de tempo entre localizações (segundos);
- $Velicidade_{média}$ - Velocidade média do drone (metros/segundo).

$$Velocidade_{média} = \frac{D}{T} \quad (6)$$

3.5. Comunicação

Para um correcto funcionamento de todos os procedimentos sucessivos que levam a realizar o *spoofing* de GPS e evitar *buffer overflow* por parte dos vários subsistemas integrantes, foi desenvolvido algo similar a um protocolo de comunicação entre os mesmos, desenrolando-se sequencialmente, mas com capacidade de funcionar com outros processos em paralelo.

Para tal, na aplicação central (código de Nodejs) foi utilizado o módulo *childprocess* que fornece a capacidade de iniciar processos "filhos", que por padrão, após a criação desse processo são estabelecidos canais de comunicação STDIN e STDOUT entre o processo principal e o "filho", permitindo a comunicação e comando entre ambos.

Abaixo, na Figura 3-21, é apresentado um diagrama de mensagens da comunição entre os vários códigos quando o sistema é inicializado, e quando é pressionado o gatilho da arma com a mesma em modo de *spoofing* estático.

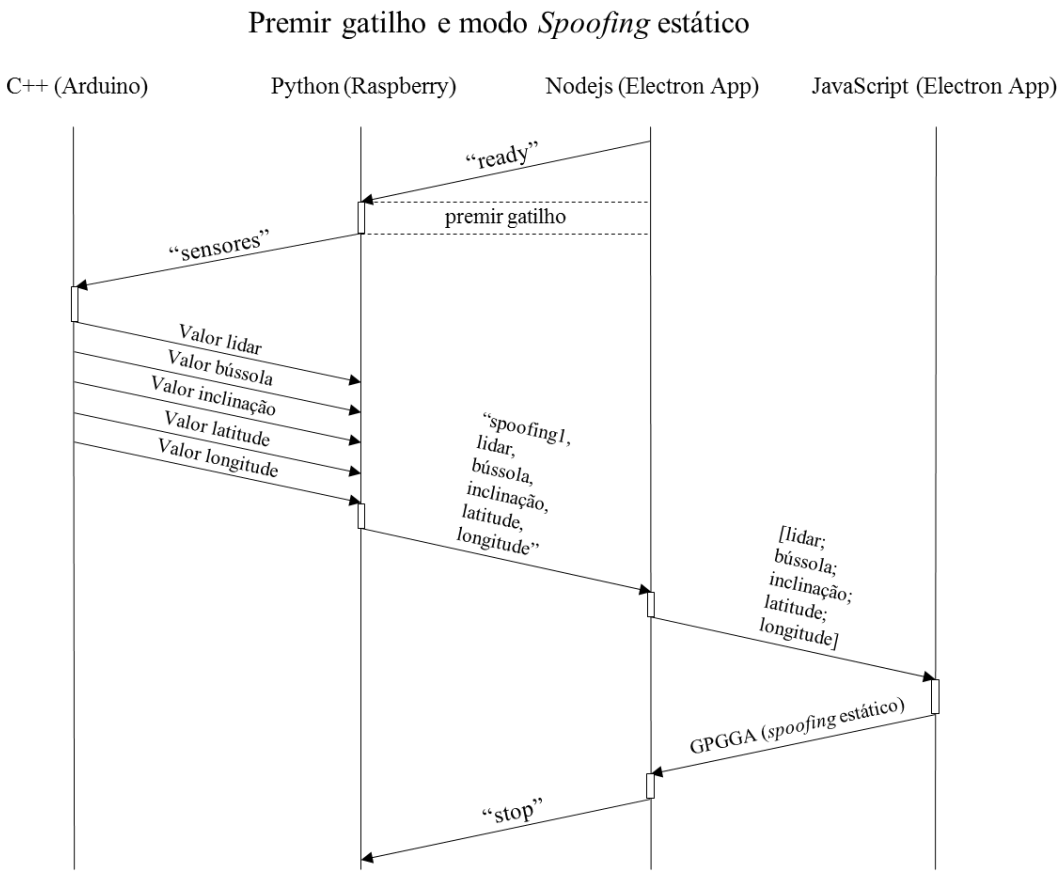

Figura 3-21 Troca de mensagens ao premir gatilho em modo spoofing estático

Ao ligar o sistema de *spoofing* de GPS não só é inicializado o mapa predefinido na interface gráfica, como também um processo paralelo Python, em que o código nesse processo é responsável por verificar em que modo a arma se encontra selecionada (*spoofing* estático ou dinâmico) e esperar pela mudança do estado do gatilho para pressionado. Logo após iniciar esse processo Python é enviada uma mensagem ("ready") do Nodejs para o mesmo através do canal STDIN, informando que a aplicação central se encontra preparada para receber resposta e atuar sobre os valores recebidos. Depois dessa mensagem o sistema encontra-se numa fase de espera, em que o gatilho da arma seja pressionado. Ao se confirmar esse evento, o próprio código Python envia uma mensagem ("sensores") para o Arduino, através da porta serie ligada entre Arduino e Raspberry, que ao receber o alerta inicia a leitura de todos os sensores e responde os seus valores através da comunicação serie. O processo Python ao receber os valores de todos os sensores e verificar que os mesmos se tratam de valores realistas encaminha-os para o Nodejs através do canal STDOUT e termina-se o processo Python. O Nodejs organiza a informação recebida conforme o modo de *spoofing* selecionado, e envia os valores para o JavaScript através do emissor de eventos webContents.send da *framework* Electron. Este último, ao receber esses valores no JavaScript, efectua todos os cálculos analíticos, anteriormente explicados no ponto 3.4, e retorna a mensagem NMEA com todas as *sentences* GPGGA corretamente criadas. A partir desse ponto o sistema inicia a transmissão dos sinais de GPS falsos através da placa bladeRF até que o gatilho deixe de ser pressionado.

Semelhante ao processo de comunição entre os vários códigos quando a arma se encontra em modo de *spoofing* estático, abaixo na Figura 3-22 é apresentado um diagrama de mensagens da comunição entre os vários códigos quando o sistema é inicializado e pressionado o gatilho da arma com a mesma em modo de *spoofing* dinâmico.

Todos os processos e trocas de mensagens e informação funcionam de forma igual, excepto à primeira leitura dos valores dos sensores ligados ao Arduino, pois é necessário soltar e voltar a pressionar o gatilho de forma a obter-se duas leituras distintas, ou seja, duas localizações do drone, para os cálculos de determinação de rota e velocidade do drone.

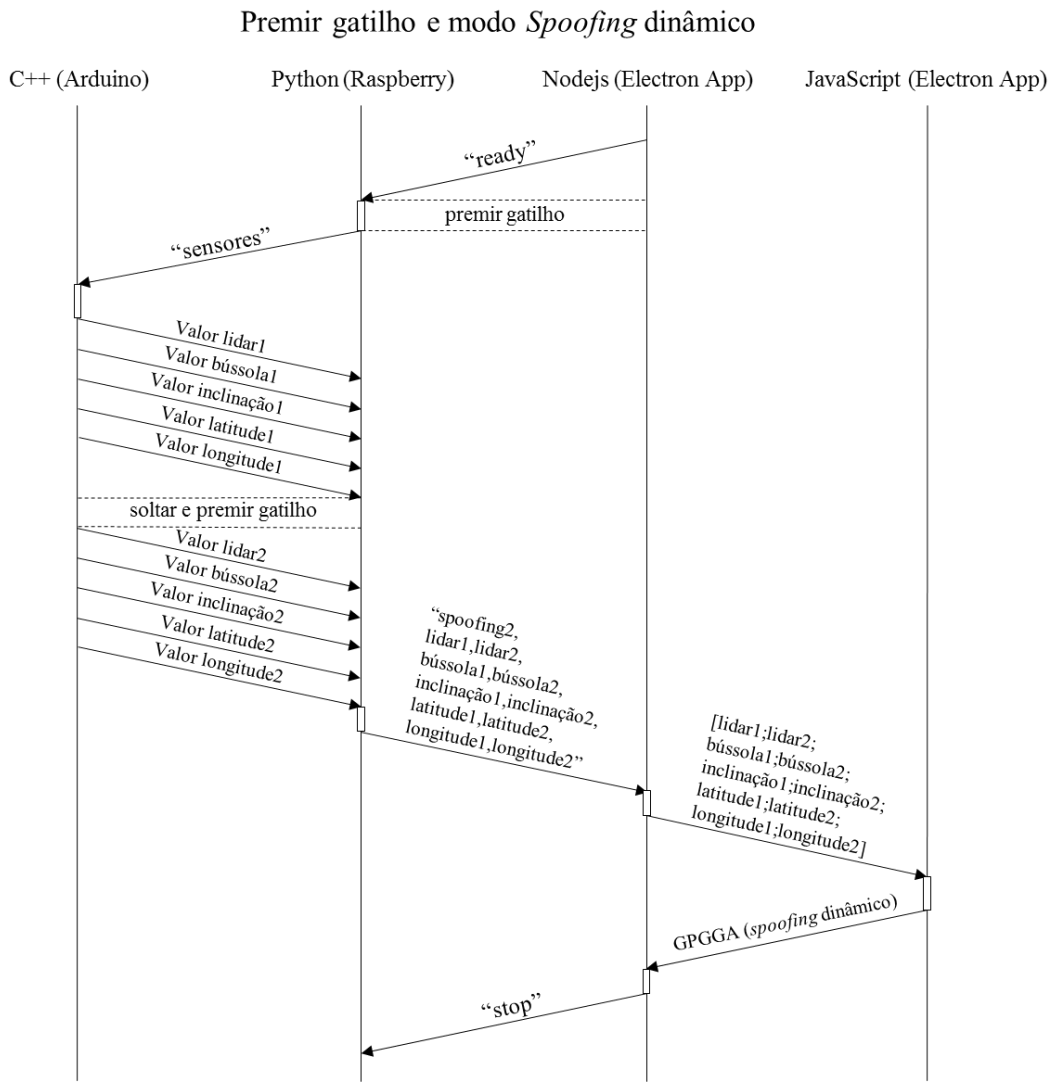

Figura 3-22 Troca de mensagens ao premir gatilho em modo spoofing dinâmico

Para ambos os modos de *spoofing* após o gatilho ser premido (finalizar toda a troca de informação e iniciar a transmissão dos sinais de GPS falsos), é criado um processo paralelo Python. Neste processo, o código é responsável por esperar pela mudança de estado do gatilho para solto. Logo após inciar esse processo Python é enviada uma mensagem ("stop") do Nodejs para o mesmo através do canal STDIN, informando que o sistema se encontra a transmitir e está preparada para receber resposta de comando para desligar transmissão. Abaixo na Figura 3-23, é apresentado um diagrama de mensagens da comunição entre os vários códigos quando é solto gatilho da arma com a mesma a transmitir em modo de *spoofing* estático ou dinâmico. Após soltar o gatilho o processo Python envia uma mensagem ("triggerOff") de confirmação de gatilho solto para o Nodejs, através do canal STDOUT e termina-se o processo Python. O Nodejs ao receber

essa mensagem de gatilho solto, envia uma mensagem (“clear”) para o JavaScript a informar que o mesmo deve limpar todos os pontos e retas desenhados na interface gráfica, elimina o processo do *software* bladeGPS para terminar a transmissão de sinais GPS e arranca com o processo paralelo Python inicialmente explicado, com código responsável por verificar em que modo a arma se encontra selecionada (*spoofing* estático ou dinâmico) e novamente esperar pela mudança de estado do gatilho para pressionado.

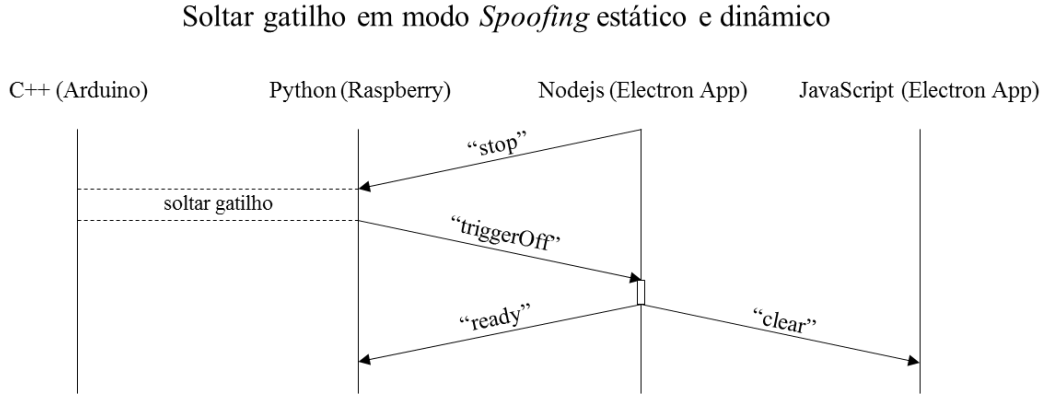

Figura 3-23 Troca de mensagens ao soltar gatilho

3.6. Circuitos Eletrónicos e Alimentação

3.6.1. Circuitos Eletrónicos

Após determinação do *hardware* e *software* que o sistema utiliza, funcionalidades que o mesmo apresenta e procedimentos que necessita de respeitar, foi necessário desenhar os circuitos eletrónicos, o mais otimizados possível, de modo que se consegui-se um sistema fiável, porém flexível. Para tal foi utilizado o *software open-source* Fritzing [42], que permitiu desenhar o circuito, Figura 3-24, com a integração de todos os sensores e interação entre as plataformas Arduino e Raspberry Pi.

As ligações dos sensores MPU6050, lsm303d e Lidar, respetivamente os componentes a), b) e c) da Figura 3-24, são feitas totalmente com o Arduino, são todos alimentados apartir da fonte de 5V e GND da plataforma e a comunicação é feita com recurso ao protocolo i2c, ou seja, os terminais *Signal Data* (SDA) de todos os sensores ligados entre si e ao pino A4 do Arduino e os terminais *Signal Clock* (SCL) também ligados entre si e ao pino A5 do Arduino. Funcionando o Arduino como *master* e todos os três sensores como *slave*, comunicando de forma sincronizada pelo *clock* do *master* (Arduino).

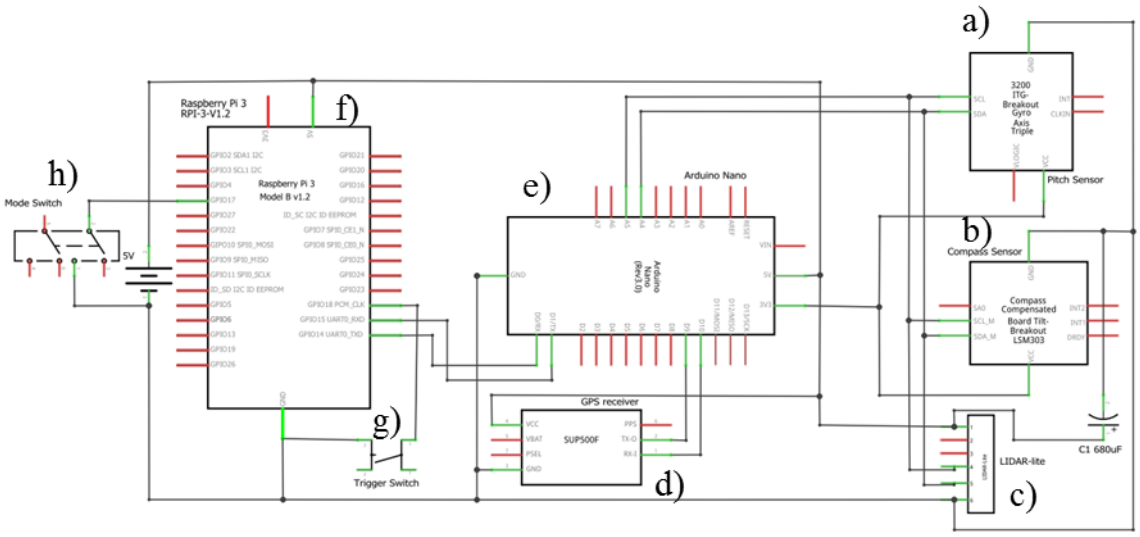

Figura 3-24 Circuito eletrónico do sistema

O recetor de GPS ublox MAX-7Q-0-000, componente d) da Figura 3-24, também é alimentado apartir do Arduino, mas pela fonte de alimentação de 3.3V e GND. A comunicação é feita através de *Serial Port Interface* (SPI) ligando o Rx do recetor ao pino digital 9 do Arduino e o Tx do recetor ao pino digital 10. Conseguindo assim uma comunicação assíncrona, apenas necessitando que os *clocks* do Arduino e recetor sejam de um valor o mais aproximado possível.

A parte correspondente à plataforma Raspberry Pi, componente f) da Figura 3-24, são as ligações com o interruptor do gatilho da arma, interruptor do seletor de modo de *spoofing*, respetivamente componentes g) e h) da Figura 3-24, e ligação/comunicação com o Arduino. Os interruptores são ligados aos pinos de entrada e saída do Raspberry Pi, nomeadamente, gatilho ao pino 12 e seletor de modo pino 15. Enquanto que a ligação entre Raspberry Pi e Arduino é feita através de *Universal Serial Bus* (USB).

3.6.2. Alimentação

Idealizando o sistema como sendo móvel, é de extrema importância estudar todos os pormenores de consumos elétricos de todas as componentes do sistema. Para que se consiga determinar a solução mais adequada de bateria como fonte de energia a utilizar, possibilitando que o protótipo final tenha a capacidade de ser utilizado com continuidade

sem necessitar de recarregar ou trocar a bateria, por exemplo, durante a vigilância de um soldado numa zona militar (8 horas de vigilância).

Foram retiradas informações de consumos de todos os aparelhos e calculada a dimensão mínima da bateria a utilizar.

Na Tabela 3-2, abaixo demonstrada, estão apresentados os valores dos consumos energéticos da plataforma bladeRF em conjunto com o amplificardor XB-300. Amplificador esse que não é utilizado neste sistema devido à sua limitação de banda de frequência, *industrial, scientific and medical* (ISM) *radio bands*, ou seja, apenas nos 2.4GHz, mas foram utilizados esses valores, testados em [43], pois indicam com maior aproximação ao conjunto (bladeRF e amplificador) a ser utilizado no sistema. Como o sistema apenas irá utilizar a transmissão da plataforma bladeRF e com *Variable Gain Amplifier* 1 e 2 nos valores máximos foram anotados como consumo mínimo do conjunto bladeRF e amplificador os 8.584W apresentados na Tabela 3-2.

Tabela 3-2 Consumos energéticos bladeRF com amplificador XB-300 [43]

RX	TX	RX / TX Freq [MHz]	IBW [MHz] / SR [MSPS]	RX Gains [dB] LNA / VGA1 / VGA2	TX Gains [dB] VGA1 / VGA2	Power [W]
OFF	OFF	2402 / 2400	28 / 40	6 / 30 / 30 (max)	-4 / 25 (max)	2.314
ON	OFF	2402 / 2400	28 / 40	6 / 30 / 30 (max)	-4 / 25 (max)	3.133
OFF	ON	2402 / 2400	28 / 40	6 / 30 / 30 (max)	-4 / 25 (max)	8.584
ON	ON	2402 / 2400	28 / 40	6 / 30 / 30 (max)	-4 / 25 (max)	9.013

Os valores dos consumos máximos das plataformas Arduino e Raspberry Pi foram obtidos através de medições reais com o sistema em funcionamento. Para tal, foi utilizado um amperímetro em série com alimentação das plataformas, tendo-se obtido um valor de corrente instantânea, na plataforma Arduino, de aproximadamente 0.65mA, e na plataforma Raspberry Pi de 520mA. Determinando os seus consumos elétricos em Watts com as Equações 7 e 8:

$$\text{Energia}_{\text{Arduino}} = 5 \times 0.065 = 0.325 \text{ W} \quad (7)$$

$$\text{Energia}_{\text{Raspberry}} = 5 \times 0.520 = 2.6 \text{ W} \quad (8)$$

Como as medições dos consumos da plataforma Arduino foram feitas com todos os sensores acoplados, os seus consumos já se encontram contabilizados nos valores de consumo totais do Arduino.

Com os valores anteriormente referidos e calculados, sabendo que a bateria será de 12V com o regulador para 5V, torna-se possível dimensionar os valores de bateria necessários para que o sistema consiga estar ligado durante as 8 horas. Para tal utiliza-se a Equação 9:

$$Bateria_{SemTransmissão} = \frac{2.6 + 2.3 + 0.325}{12} \times 8 \cong 3.48 \text{ Ah} \quad (9)$$

Calculando também o valor necessário de bateria com o sistema sempre ligado e em transmissão durante as 8 horas, através da Equação 10:

$$Bateria_{Transmissão} = \frac{2.6 + 8.584 + 0.325}{12} \times 8 \cong 7.67\text{Ah} \quad (10)$$

Como os valores calculados para a bateria, nas Equações 9 e 10, não são valores de *standard* de mercado e o sistema não necessita de permanecer ligado e a transmitir 100% do tempo, mas sim quando existir alguma ameaça, optou-se pelo valor comercial de 2.2Ah. Conseguindo assim uma bateria Li-PO de menores dimensõems, peso e mais facilmente integrável no protótipo móvel, Figura 3-25.

Figura 3-25 Bateria Li-PO 2200 e regulador de tensão output 5.5V

Capítulo 4 - Testes e Discussão de Resultados

Neste capítulo são descritos os testes realizados aos vários subsistemas, com o intuito de se verificar a existência de possíveis erros de programação ou concepção dos mesmos. Os testes encontram-se enumerados como testes *indoor* e *outdoor* ao *software* bladeGPS com vários recetores de GPS diferentes e aos sensores, de forma individual, para determinação das suas limitações e possíveis influências nos resultados finais.

4.1. *Software* bladeGPS

Para os testes realizados com o *software* bladeGPS foram utilizados três tipos de recetores, *smartphone*, recetor u-blox M8 GNSS Evaluation Kit e u-blox MAX-7Q. Em conjunto com a plataforma da Nuand BladeRFx40 e um computador com sistema operativo windows, de forma a testar o *spoofing* em execução.

4.1.1. Testes *Indoor*

A realização dos seguintes testes, designados de *indoor*, foram efetuados dentro de um edifício, ou seja, sem a influência de sinais de GPS reais, visto que *indoor* não é possível obter localização por GPS dado o baixo nível de potência do sinal. Testando assim o comportamento dos vários recetores sem que os mesmos possuam localização.

SmartPhone

Foram concreticados testes com um *smartphone* para verificação de que muitos dos recetores de GPS instalados nesses aparelhos são vulneráveis aos ataques de *spoofing*. Nos testes realizados foi utilizado como vítima de *spoofing* o *smartphone* LG L90. Para tal foi instalada a aplicação GPSTest [44], Figura 4-1, no aparelho, aplicação esta que permite uma visualização em tempo real de quais os satélites visíveis, nível de potência de sinal GPS e localização no mapa mundo.

Figura 4-1 Aplicação GPSTest [44]

Neste teste foi utilizado uma distribuição Linux baseada em Debian, em conjunto com o *software* bladeGPS, de forma a simular a constelação NAVSTAR-GPS [20]. Com este sistema foi possível simular, com coordenadas definidas pelo utilizador, a localização do *smartphone* em relação à sua posição real. Como se vê na Figura 4-2 à esquerda, são detectados vários sinais de satélites com bom SNR, no centro são mostrados alguns dos satélites simulados em linha de vista na constelação e à direita é apresentada a localização falsa através das coordenadas de latitude, longitude e com o marcador vermelho no mapa mundo. Sendo que a sua verdadeira localização não é a apresentada pela aplicação, Caracas Venezuela, mas sim Lisboa Portugal. Com este teste verificou-se que é possível enganar a localização de um recetor GPS instalado em *smartphone* em ambiente *indoor*.

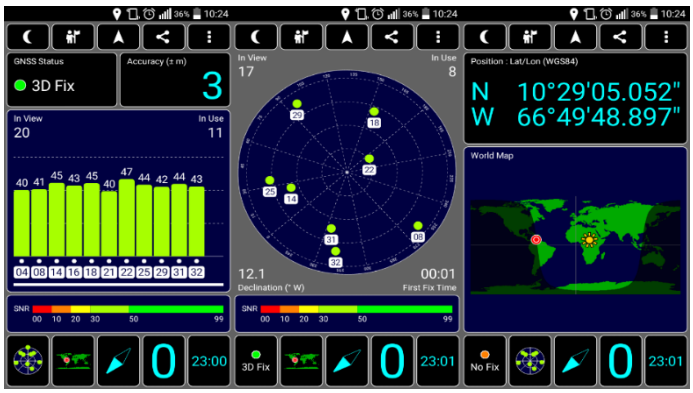

Figura 4-2 Spoofing de smartphone com aplicação GPSTest indoor

u-blox M8 GNSS Evaluation Kit

Para uma maior diversidade de testes foram também produzidos testes de *spoofing* de GPS ao recetor u-blox M8 GNSS Evaluation Kit, Figura 4-3. O kit de avaliação u-blox M8 permite uma avaliação simples das tecnologias de posicionamento. Possui uma interface USB integrada que fornece alimentação, o que elimina a necessidade de uma fonte de alimentação externa, e transferência de dados em alta velocidade. O recetor foi utilizado com um computador via interface USB em conjunto com *software* u-center, que

se trata de uma ferramenta poderosa para avaliação, análise de desempenho e configuração de recetores GNSS da u-blox.

Figura 4-3 u-blox M8 GNSS Evaluation Kit

Neste teste foi utilizado novamente uma distribuição Linux baseada em Debian, em conjunto com o *software* bladeGPS. Foi possível simular, com coordenadas definidas pelo utilizador, a localização do recetor u-blox M8 GNSS Evaluation Kit em relação à sua posição real. Como se vê na parte inferior da Figura 4-4, são detetados vários sinais de satélites GPS com bom SNR em linha de vista na constelação, na parte superior do monitor é apresentado o mapa mundo com a localização simulada através de um marcador verde. Sendo que a sua verdadeira localização não é a apresentada pelo u-center, Washington, DC Estados Unidos da América, mas sim Lisboa Portugal. Com essa experiência verificou-se que é possível enganar a localização de um recetor GPS u-blox preparado para avaliar sistemas de localização num ambiente *indoor*.

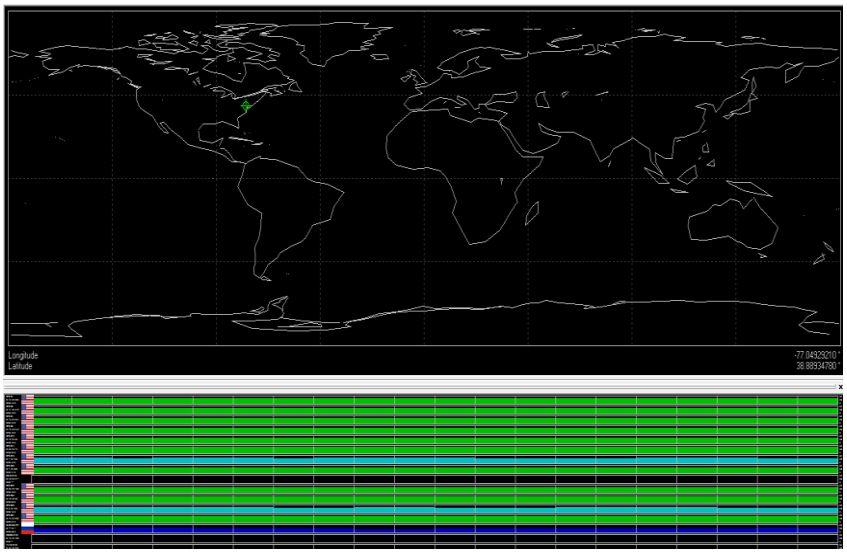

Figura 4-4 Spoofing recetor u-blox M8 GNSS Evaluation Kit indoor

u-blox MAX-7Q

Tendo em conta que o recetor u-blox MAX-7Q, Figura 4-5, é muito utilizado para implementar em drones dos mais variados tipos, terrestres, aéreos e aquáticos para condução autónoma dos mesmos, também foi testada a sua resposta ao *spoofing* de GPS. O recetor tem a capacidade de receber sinais GNSS e conseguir obter a sua localização através dos mesmos.

Foi utilizado em conjunto com um Arduino, de forma a comunicar a sua localização e, assim, através da interface serie do Arduino visualizarmos a informação relativa à sua localização atual.

Figura 4-5 Recetor u-blox MAX-7Q

Nesta experiência esse tipo de recetor não apresentou qualquer tipo de resistência ao receber os sinais de GPS *indoor* criados e transmitidos pelo *software* bladeGPS e plataforma bladeRF, aceitou a localização integrada nos sinais falsos. Como se pode verificar na Figura 4-6, o recetor identifica a sua localização conforme os valores introduzidos no *software* bladeGPS, ou seja, a sua verdadeira localização não é a exibida pelos valores de latitude e longitude na serie do Arduino, que corresponde a Pyongyang Coreia do Norte, mas sim Lisboa Portugal. Com esse ensaio verificou-se que em ambiente *indoor* é possível enganar a localização de um recetor GPS u-blox muito utilizado em drones.

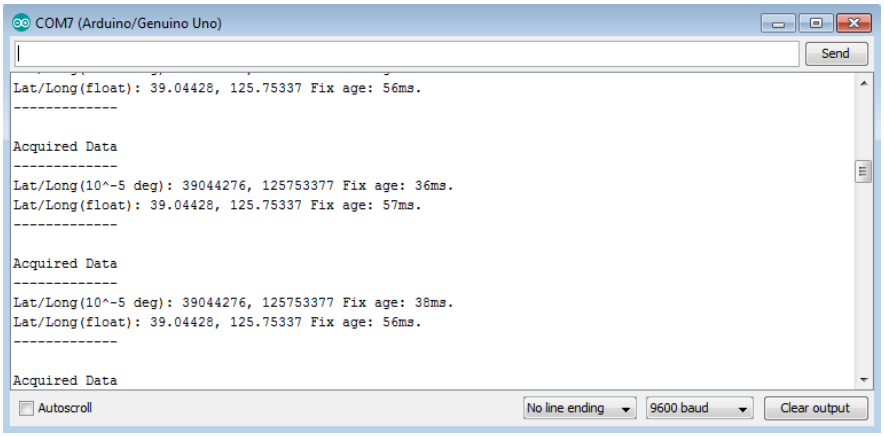

Figura 4-6 Output do programa de teste recetor u-blox MAX-7Q spoofing indoor

4.1.2. Testes *Outdoor*

A realização dos seguintes testes *outdoor* foi efetuada no exterior, ou seja, com a influência de sinais de GPS reais, visto que em *outdoor* é possível obter localização por GPS e assim, testar o facto do recetor já ter localização através do GPS real e depois ser vítima de *spoofing* de GPS.

SmartPhone

Foram concretizados testes *outdoor* em *smartphone* para verificação de que muitos dos recetores de GPS instalados nesses aparelhos são vulneráveis aos ataques de *spoofing*, mesmo depois de já terem obtido localização através de GPS real. Nos testes realizados foi utilizado como vítima de *spoofing* o *smartphone* LG L90. Com os mesmos procedimentos anteriormente descritos no teste *indoor*.

Foi utilizado novamente o sistema operativo Linux baseada em Debian, em conjunto com o *software* bladeGPS. Como indicado anteriormente, o teste desenrolou-se no exterior sem qualquer obstrução do sinal GPS real.

Após o *smartphone* obter localização real foi iniciada a transmissão de sinais de GPS falsos, e passados cerca de 3 minutos o sistema perdeu localização, tendo em conta que detetou outro sinal GPS (sinal GPS falso transmitido pela plataforma bladeRF), e de seguida aceitou o sinal de GPS falso possivelmente porque o mesmo apresenta melhor SNR. Nestas condições, e com os resultados anteriormente descritos, foi possível simular, com coordenadas definidas pelo utilizador, a localização do *smartphone* em relação à sua posição real. Como se observa na Figura 4-7, em baixo à esquerda são detetados vários sinais de satélites com bom SNR, no centro são mostrados alguns dos satélites simulados e satélites reais em linha de vista da constelação e à direita é apresentada a localização falsa do *smartphone* na aplicação GoogleMaps, em que a posição real do *smartphone* seria no ponto vermelho exibido na mesma, e não no ponto a azul representado pela aplicação.

Com este teste verificou-se que é possível enganar a localização de um recetor GPS instalado em *smartphone* em ambiente *outdoor*, ou seja, o aparelho já tinha localização real.

Figura 4-7 Spoofing de smartphone com aplicação GPSTest e GoogleMaps outdoor

u-blox M8 GNSS Evaluation Kit

Inicialmente foi ligado o kit de avaliação u-blox M8 ao computador em conjunto com o *software* u-center, esperando-se conseguir localização do GPS real para o mesmo. Após o recetor conseguir a sua localização deu-se início à transmissão de sinais de GPS falsos. Passados cerca de 2 minutos o recetor já havia detetado os novos sinais de GPS, mas só passados cerca de 30 minutos é que aceitou os sinais de GPS criados pelo *software* bladeGPS e transmitidos pela plataforma bladeRF.

Trantando-se de um recetor utilizado para avaliação e análise de desempenho de sistemas GNSS, possui características que o tornam menos suscetível a ataques de *spoofing* e *jamming*. Daí o seu comportamento em demorar tanto tempo a aceitar os sinais de GPS falsos transmitidos.

Como se vê na Figura 4-8, são detetados vários sinais de satélites GPS reais com baixo nível de SNR e outros sinais de GPS com nível de SNR mais elevado, na parte inferior do monitor são mostrados alguns dos satélites simulados e reais em linha de vista da constelação, no centro do monitor é apresentado o mapa mundo com a localização simulada através de um marcador verde, enquanto que a localização real do recetor seria no ponto vermelho do mapa.

Com esta experiência verificou-se que é possível enganar a localização de um recetor GPS u-blox M8 em ambiente *outdoor* já com localização definida, mas para o intuito do sistema desenvolvido o tempo que demora a aceitar os sinais seria um ponto critico do sistema de *spoofing*. Possivelmente utilizando técnicas de *jamming* antes de iniciar a

transmissão do *spoofing* seria uma boa opção para acelerar o processo de enganar o recetor com sinais de GPS falsos.

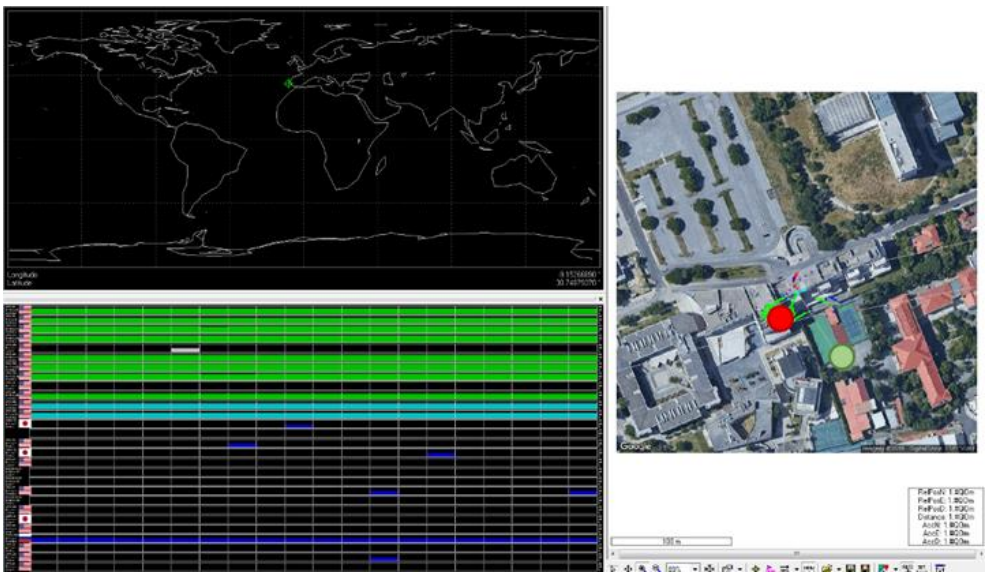

Figura 4-8 Spoofing recetor u-blox M8 GNSS Evaluation Kit outdoor

u-blox MAX-7Q

Por fim, nos testes ao *software* bladeGPS, foi novamente testado o recetor u-blox MAX-7Q, Figura 4-5. Nesta experiência o recetor já com localização através de sinais GPS reais, não apresentou novamente qualquer tipo de resistência ao receber os sinais de GPS falsos, criados e transmitidos pelo *software* bladeGPS e plataforma bladeRF, aceitou a localização integrada nos sinais falsos. Como se pode verificar na Figura 4-9, o recetor muda a sua localização (assinalado a vermelho latitude e longitude) conforme os valores introduzidos no *software* bladeGPS, Figura 4-10, ou seja, a sua verdadeira localização não é a exibida pelos segundos valores de latitude e longitude na *serie* do Arduino, mas sim pelos primeiros, Figura 4-9.

Com esse ensaio verificou-se que é possível enganar a localização, em ambiente *outdoor*, de um recetor GPS u-blox muito utilizado em drones.

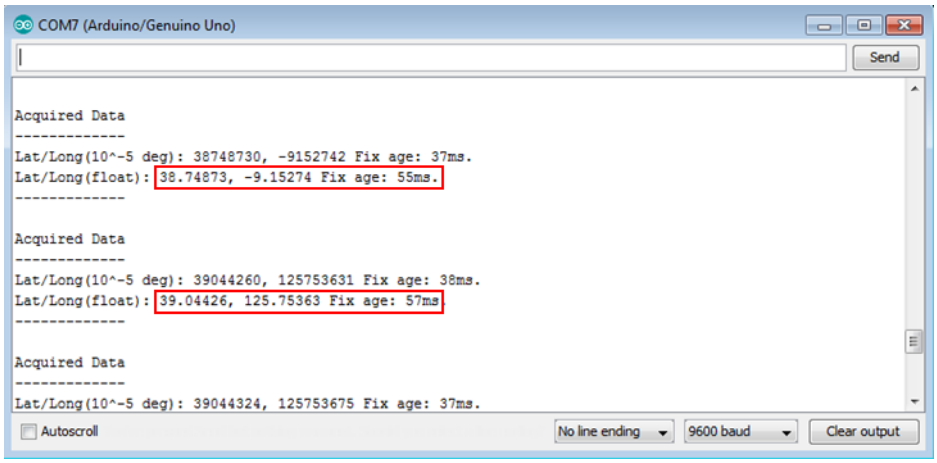

Figura 4-9 Output do programa de teste recetor u-blox MAX-7Q spoofing outdoor

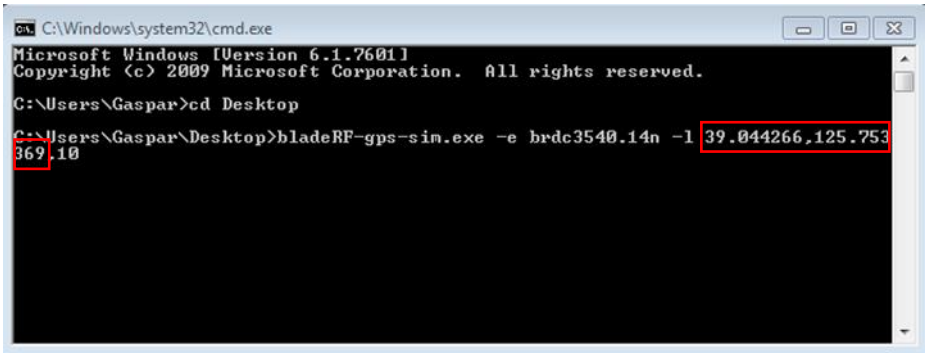

Figura 4-10 Introdução de valores no software bladeGPS

4.2. Sensores

Para os testes realizados aos sensores utilizou-se um computador com o IDE *Open-Source* Arduino *Software* em conjunto com a plataforma Arduino Uno. Foram elaborados testes laboratoriais com o intuito de testar, individualmente, cada um dos sensores em questão, determinando os seus possíveis erros de leitura e programação mais eficiente para os mesmos.

4.2.1. Sensor Lidar

Figura 4-11 Sensor Lidar implementado na arma

Os testes laboratoriais efetuados ao sensor trataram-se de testes de precisão de medição. Para o teste foi utilizada uma fita métrica de 10 metros e feitas medições de metro a metro. E com o aumentar da distância, também se apresenta um aumento do erro de medição, como se verifica nos resultados do teste na Tabela 4-1 e pelo gráfico da Figura 4-12, que apresenta uma linha de tendência linear ascendente, com algum declive. Os valores de erro foram de encontro com o indicado no datasheet do sensor apresentado no Anexo C.1. *Datasheet* Lidar. Mas como os valores de erro não são muito elevados, não conseguem influênciar significativamente as medições finais para a determinação da localização do drone.

Tabela 4-1 Valores de erro medidos pelo sensor Lidar

Distância [m]	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Erro de medição [m]	0.026	0.031	0.017	0.02	0.08	0.12	0.1	0.14	0.11	0.13

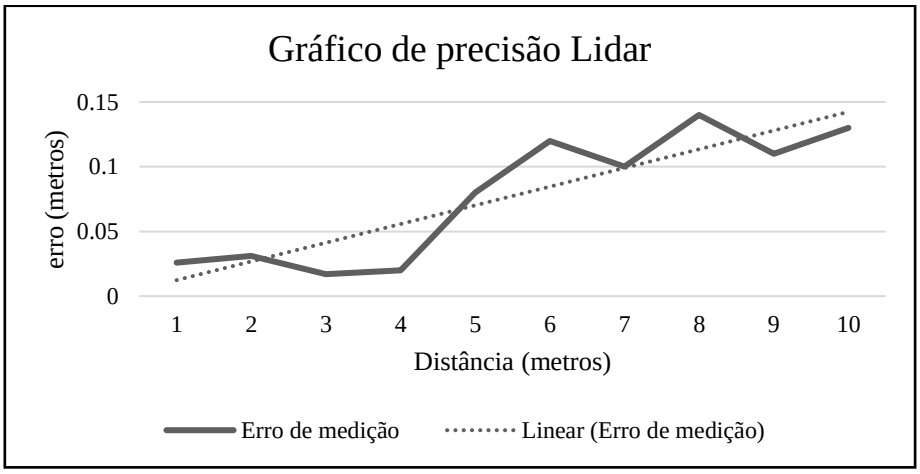

Figura 4-12 Gráfico de precisão de medição do sensor Lidar

4.2.2. Sensor *Pitch*

Figura 4-13 Sensor Pitch implementado na arma

O dispositivo utilizado foi o MPU6050, como mencionado em 3.3.1 *Hardware*. Na Figura 4-13 o mesmo já se encontra acopolado à estrutura da arma.

Foi elaborado um teste de precisão recorrendo à utilização de um *smartphone* e o seu giroscópio. Através de uma aplicação gratuita disponível na google play, Angle Meter [45], e acopolando o sensor MPU6050 ao *smartphone*, como demonstrado na Figura 4-14, foram retiradas medições com um intervalo cinco graus, numa gama dos 0º aos 90º.

Figura 4-14 Teste laboratorial ao sensor MPU6050 (pitch)

Registados os valores indicados pela aplicação, lado direito da Figura 4-15, e os valores medidos pelo sensor MPU6050, lado esquerdo da Figura 4-15, foi calculada a média dos 15 valores apresentados pelo sensor e a diferença entre ambos os valores das plataformas. Com esses resultados foi construído um gráfico representativo do erro e o seu comportamento com a variação do ângulo, Figura 4-16.

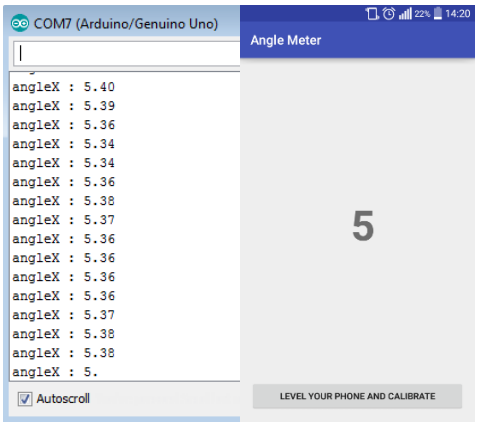

Figura 4-15 Teste laboratorial ao sensor MPU6050 com aplicação de smartphone

Verificou-se que com o aumento do ângulo, também se apresenta um ligeiro aumento do erro de medição, como se pode averiguar nos valores da Tabela 4-2 e no gráfico da Figura 4-16, que apresenta uma linha de tendência linear ascendente, de muito baixo declive, levando a que esses erros não consigam de forma alguma influênciar as medições finais para a determinação da localização do drone.

Tabela 4-2 Valores de erro medidos pelo sensor MPU6050

Ângulo [º]	**0**	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**	**55**
Erro de medição [º]	0.02	0.37	0.49	0.1	0.12	0.28	0.41	0.25	0.52	0.32	0.48	0.42

Ângulo [º]	**60**	**70**	**75**	**80**	**85**	**90**
Erro de medição [º]	0.74	0.22	0.18	0.24	0.36	0.08

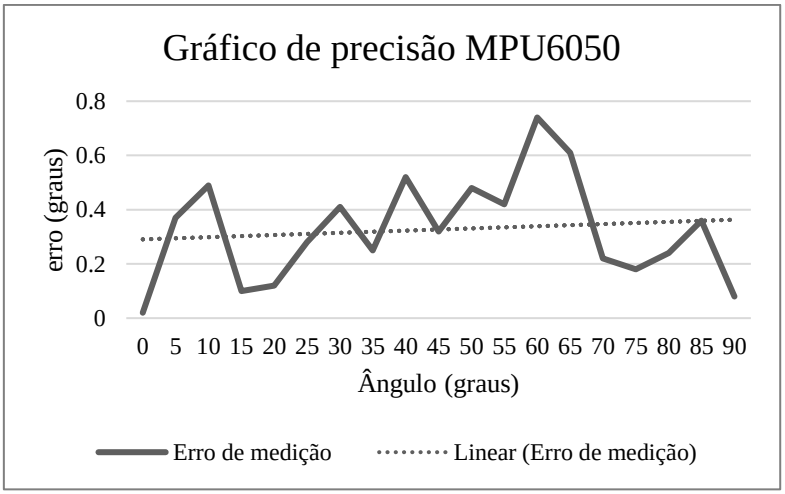

Figura 4-16 Gráfico de precisão de medição do sensor MPU6050

4.2.3. Sensor Bússola

Figura 4-17 Sensor de bússola implementado na arma

Trata-se do sensor pololu lsm303d, como mencionado em 3.3.1 *Hardware*. Na Figura 4-17 a mesma já se encontra acopolada à estrutura da arma.

Foi elaborado um teste de precisão recorrendo à utilização de um *smartphone* e o seu magnetómetro. Através de uma aplicação gratuita, disponível na google play, Bússola [46], e acopolando o sensor lsm303d ao *smartphone*, como demonstrado na Figura 4-18, foram retiradas medições com um intervalo de 10º, numa gama dos 0º aos 360º.

Figura 4-18 Teste laboratorial ao sensor lsm303d (bússola)

Registados os valores proporcionados pela aplicação, lado direito da Figura 4-19, e os valores medidos pelo sensor lsm303d, lado esquerdo da Figura 4-19, foi calculada a média dos 15 valores apresentados pelo sensor e a diferença entre ambos os valores das plataformas. Com esses resultados foi construído um gráfico representativo do erro e o seu comportamento com a variação do ângulo, Figura 4-20.

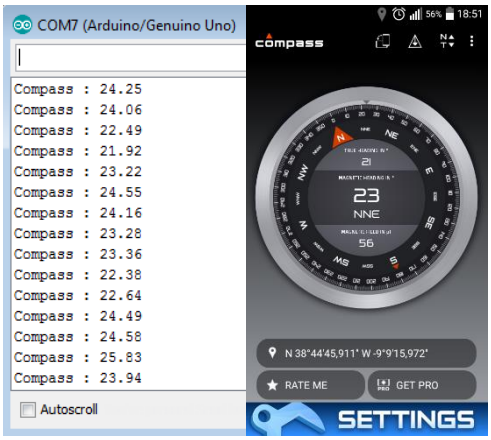

Figura 4-19 Teste laboratorial ao sensor lsm303d com aplicação de smartphone

Com o variar do ângulo, não se apresenta um aumento do erro de medição, como se verifica nos valores da Tabela 4-3 e no gráfico da Figura 4-20. Apresenta-se sim uma linha de tendência linear horizontal, demonstrando que os erros medidos variam, mas com uma certa coerência nas diferenças de valores entre plataformas. Os valores registados de erro já atingem valores que poderam, eventualmente, influênciar minimamente a determinação da localização do drone.

Tabela 4-3 Valores de erro medidos pelo sensor lsm303d

Ângulo [º]	**0**	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**	**110**	**120**
Erro de medição [º]	1.03	1.86	1.12	2.01	2.32	1.65	2.78	2.65	1.98	1.21	1.54	1.3	1.78

Ângulo [º]	**130**	**140**	**150**	**160**	**170**	**180**	**190**	**200**	**210**	**220**	**230**	**240**	**250**
Erro de medição [º]	1.59	2.06	0.83	1.45	1.81	1.08	1.09	2.04	1.64	1.12	1.95	1.11	2.16

Ângulo [º]	**260**	**270**	**280**	**290**	**300**	**310**	**320**	**330**	**340**	**350**	**360**
Erro de medição [º]	1.27	1.38	1.34	2.42	3.48	1.69	1.84	2.09	1.54	1.62	1.65

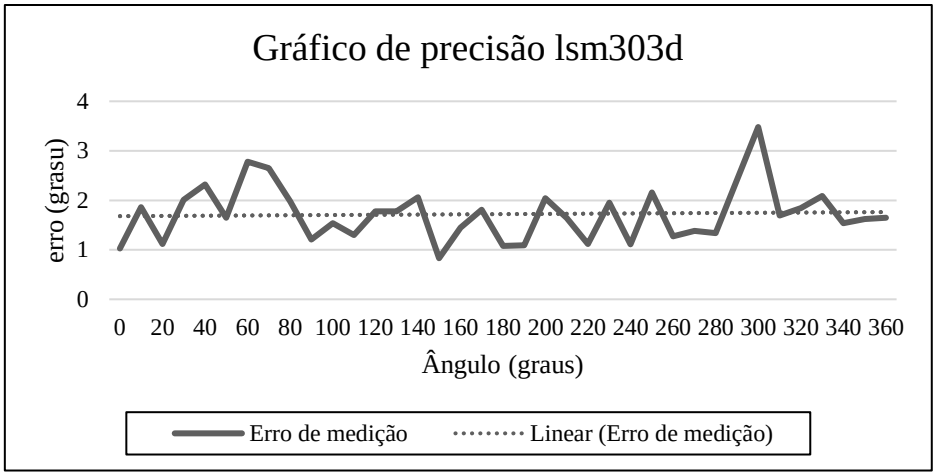

Figura 4-20 Gráfico de precisão de medição do sensor lsm303d

Capítulo 5 - Conclusões e Trabalhos Futuros

5.1. Principais Conclusões

Esta investigação revelou que é possível utilizar equipamento SDR para transmitir, receber, gravar e reproduzir quaisquer sistemas de comunicação de rádio. Com este tipo de tecnologia é possível recriar sinais idênticos aos sinais reais de sistemas já existentes, o que torna possível elaborar técnicas avançadas de ataques, como gravar e reproduzir o sinal de um comando de portão ou porta de garagem, até mesmo transmitir sinais capazes de bloquear as comunicações de um sistema. No passado, elaborar um sistema de rádio era algo extremamente complexo e requeria habilidades de *design* e construção de *hardware*, com a utilização das plataformas SDR é necessário atuar ao nível do processamento digital de sinais e *frameworks* de *software* para SDR.

Conseguir a implementação do sistema desenvolvido e apresentado na presente dissertação sem a aplicação de plataformas SDR, seria uma tarefa com demasiadas questões técnicas. Com a utilização dessas plataformas e com o desenvolvimento correcto de *software* e processamento digital de sinais e algoritmos eficientes é possível, com desafio adicional, desenvolvimento de sistemas de *spoofing* e de sistemas e telecomunicações de elevada aplicabilidade na navegação de veículos e testes reais a nível experimental.

Com o desenvolvimento do sistema de *spoofing* por GPS, comprovou-se que se consegue utilizar uma vulnerabilidade de um determinado sistema e aproveitar a mesma para criar algo com aplicabilidade prática. Mesmo sendo um sistema falível e com algumas limitações, é facilmente aperfeiçoado com a possibilidade de integrar novas técnicas ou tecnologias que apresentem novas soluções. Pois considerando que na atualidade já existem soluções de proteção contra técnicas de *spoofing*, e uma constante investigação e desenvolvimento das mesmas, é inevitavel o desenvolvimento de novas técnicas para encontrar formas de ultrapassar as proteções futuras, sendo necessário estar em constante investigação e desenvolvimento nestes assuntos.

5.2. Propostas de Investigação Futura

Para futuros trabalhos apresentam-se as seguintes propostas:

- Evolução para que consiga transmitir sinais de outros sistemas GNSS, como Galileu, Beidou e Compass;
- O sistema de *spoofing* ser capaz de interagir com sistemas externos, isto é, sistemas de radar caso o mesmo não seja capaz de determinar a localização do drone;
- Aplicar um algoritmo de controlo de potência capaz de adaptar a potência transmitida à distância que o drone se encontra, para tornar o sistema mais eficiente e sustentável.

Bibliografia

[1] Federal Aviation Administration, "Unmanned Aircraft Systems (UAS) Frequently Asked Questions," 2018. [Online]. Available: https://www.faa.gov/uas/faqs/. [Accessed: 24-Sep-2018].

[2] A. Reg, *Unmanned aircraft systems : UAVs design, development and deployment*. Reston, Va American Institute of Aeronautics and Astronautics 2010, 2010.

[3] J. Gundlach and A. I. of A. and Astronautics, *Designing unmanned aircraft systems : a comprehensive approach / Jay Gundlach, Aurora Flight Sciences, Manassas, Virginia*, 1st ed. Virginia: American Institute of Aeronautics and Astronautics, 2012.

[4] L. Petricca, P. Ohlckers, and C. Grinde, "Micro- and nano-air vehicles: State of the art," *Int. J. Aerosp. Eng.*, vol. 2011, 2011.

[5] R. Weibel and R. J. Hansman, "Safety Considerations for Operation of Different Classes of UAVs in the NAS," *AIAA 3rd "Unmanned Unlimited" Tech. Conf. Work. Exhib.*, no. September, 2004.

[6] A. Oxley, *Introduction to GPS*. 2017.

[7] N. Samama, "GNSS System Descriptions," in *Global Positioning*, John Wiley & Sons, Inc., 2007, pp. 131–161.

[8] GPS.gov, "Global Positioning Systems Directorate Systems Engineering & Integration Interface Specification IS-GPS-200." p. 224, 2018.

[9] N. Samama, "Development, Deployment, and Current Status of Satellite-Based Navigation Systems," in *Global Positioning*, John Wiley & Sons, Inc., 2007, pp. 57–93.

[10] E. D. Kaplan, *Understanding GPS: Principles and Applications-2nd Edition*, 2nd ed. 2006.

[11] J. . Á. Rodríguez and U. FAF Munich Germany, "GPS Signal Plan - Navipedia." [Online]. Available: http://www.navipedia.net/index.php/GPS_Signal_Plan. [Accessed: 11-Jan-2018].

[12] R. B. Langley, "The mathematics of GPS," *GPS World*, vol. 2, no. 7, pp. 45--50,

1991.

[13] F. Dovis, *GNSS Interference Threats & Countermeasures*. Artech House, 2015.

[14] B. M. Ledvina, W. J. Bencze, B. Galusha, and I. Miller, “An In-Line Anti-Spoofing Device for Legacy Civil GPS Receivers,” *Proceedings of the 2010 International Technical Meeting of The Institute of Navigation*, San Diego, CA, pp. 689–712, Jan-2010.

[15] M. L. Psiaki and T. E. Humphreys, “GNSS Spoofing and Detection,” *Proc. IEEE*, vol. 104, no. 6, pp. 1258–1270, 2016.

[16] M. G. Kuhn, “An asymmetric security mechanism for navigation signals,” *Int. Work. Inf. Hiding, IH*, vol. 3200, pp. 239–252, 2005.

[17] A. J. Kerns, D. P. Shepard, J. A. Bhatti, and T. E. Humphreys, “Unmanned aircraft capture and control via GPS spoofing,” *J. F. Robot.*, vol. 31, no. 4, pp. 617–636, 2014.

[18] D. P. Shepard, J. a Bhatti, T. E. Humphreys, and A. a Fansler, “Evaluation of Smart Grid and Civilian UAV Vulnerability to GPS Spoofing Attacks,” *Ion Gnss*, pp. 3591–3605, 2012.

[19] T. E. Humphreys, B. M. Ledvina, V. Tech, M. L. Psiaki, B. W. O. Hanlon, and P. M. Kintner, “Assessing the Spoofing Threat: Development of a Portable GPS Civilian Spoofer,” *Proc. 21st Int. Tech. Meet. Satell. Div. Inst. Navig.*, no. September 2008, pp. 2314–2325, 2008.

[20] H. Lin and Y. Qing, “GPS SPOOFING Low-cost GPS simulator,” 2015.

[21] “DEF CON 23 Hacking Conference,” 2015. .

[22] Sparkfun, “LIDAR-Lite v3 Hookup Guide - learn.sparkfun.com.” [Online]. Available: https://learn.sparkfun.com/tutorials/lidar-lite-v3-hookup-guide. [Accessed: 30-Jul-2018].

[23] J. D. Petty, J. N. Huckins, and A. David, “(12) Patent Application Publication (10) Pub . No .: US 2002/0187020 A1,” vol. 1, no. 19, 2002.

[24] ElectronicWings, “Sensors Modules - Mpu6050 Gyroscope Accelerometer Temperature Sensor Module | ElectronicWings.” [Online]. Available: http://www.electronicwings.com/sensors-modules/mpu6050-gyroscope-

accelerometer-temperature-sensor-module. [Accessed: 30-Jul-2018].

[25] M. Pedley, "Tilt Sensing Using a Three-Axis Accelerometer," *Free. Semicond. Appl. notes*, pp. 1–22, 2013.

[26] Pololu, "Pololu - LSM303D 3D Compass and Accelerometer Carrier with Voltage Regulator." [Online]. Available: https://www.pololu.com/product/2127. [Accessed: 30-Jul-2018].

[27] F. Semiconductor, "Implementing a Tilt-Compensated eCompass using Accelerometer and Magnetometer Sensors," pp. 1–21, 2013.

[28] u-blox, "MAX-7 series | u-blox." [Online]. Available: https://www.u-blox.com/en/product/max-7-series. [Accessed: 30-Jul-2018].

[29] Arduino Uno, "Arduino Uno Rev3." [Online]. Available: https://store.arduino.cc/arduino-uno-rev3. [Accessed: 30-Jul-2018].

[30] raspberry pi, "Raspberry Pi 3 Model B - Raspberry Pi." [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. [Accessed: 30-Jul-2018].

[31] bladeRF Naund, "bladeRF x40 | Nuand -." [Online]. Available: https://www.nuand.com/blog/product/bladerf-x40/. [Accessed: 31-Jul-2018].

[32] Nuand, "Nuand | bladeRF Software Defined Radio." [Online]. Available: https://www.nuand.com/. [Accessed: 25-Jan-2018].

[33] Microsoft, "Visual Studio Code - Code Editing. Redefined." [Online]. Available: https://code.visualstudio.com/. [Accessed: 30-Jul-2018].

[34] C. Zhao and GitHub, "Electron | Build cross platform desktop apps with JavaScript, HTML, and CSS." [Online]. Available: https://electronjs.org/. [Accessed: 30-Jul-2018].

[35] R. Madhavan and M. Wimpress, "Ubuntu MATE for the Raspberry Pi 2 and Raspberry Pi 3 | Ubuntu MATE." [Online]. Available: https://ubuntu-mate.org/raspberry-pi/. [Accessed: 30-Jul-2018].

[36] "OSQZSS." [Online]. Available: https://blog.goo.ne.jp/osqzss. [Accessed: 30-Jul-2018].

[37] Dr. David and R. Williams, "Earth Fact Sheet." [Online]. Available:

https://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html. [Accessed: 01-Aug-2018].

[38] C. Veness, “Calculate distance and bearing between two Latitude/Longitude points using haversine formula in JavaScript,” *MIT License*, 2017. [Online]. Available: https://www.movable-type.co.uk/scripts/latlong.html. [Accessed: 21-Sep-2018].

[39] Dale DePriest, “NMEA data.” [Online]. Available: https://www.gpsinformation.org/dale/nmea.htm. [Accessed: 17-Sep-2018].

[40] B. Park, J. Lee, Y. Kim, H. Yun, and C. Kee, “DGPS enhancement to GPS NMEA output data: DGPS by correction projection to position-domain,” *J. Navig.*, 2013.

[41] G. P. S. C. Page, “GPS - NMEA sentence information,” 2011. [Online]. Available: http://home.mira.net/~gnb/gps/nmea.html. [Accessed: 18-May-2018].

[42] Fritzing, “Fritzing.” [Online]. Available: http://fritzing.org/home/. [Accessed: 14-Sep-2018].

[43] Nuand, “bladeRF Power Consumption,” 2017. [Online]. Available: https://github.com/Nuand/bladeRF/wiki/bladeRF-Power-Consumption. [Accessed: 18-Sep-2018].

[44] “GPS Test – Apps no Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.chartcross.gpstest&hl=pt_BR. [Accessed: 30-Aug-2018].

[45] hahoteknik, “Angle Meter – Aplicações no Google Play,” 2016. [Online]. Available: https://play.google.com/store/apps/details?id=com.tiltmeter. [Accessed: 12-Sep-2018].

[46] gabenative, “Bússola – Aplicações no Google Play,” 2016. [Online]. Available: https://play.google.com/store/apps/details?id=com.gn.android.compass. [Accessed: 12-Sep-2018].

Apêndices e Anexos

Apêndice A. Prototipagem

Neste apêndice encontram-se demonstrações e registo fotográfico de como foram criados os protótipos de laboratório e em ambiente real do sistema de *spoofing*.

A.1. Protótipo Laboratorial

Antes do design e construção de qualquer estrutura para acomodar o sistema de *spoofing*, foram agregados todos os subsistemas conjuntamente com o intuito de testar e verificar o correto funcionamento do mesmo, como na Figura A - 1. Para isso, foram utilizadas placas de laboratório para as ligações e um monitor para *feedback* dos *outputs* do sistema, como se pode ver na Figura A - 2.

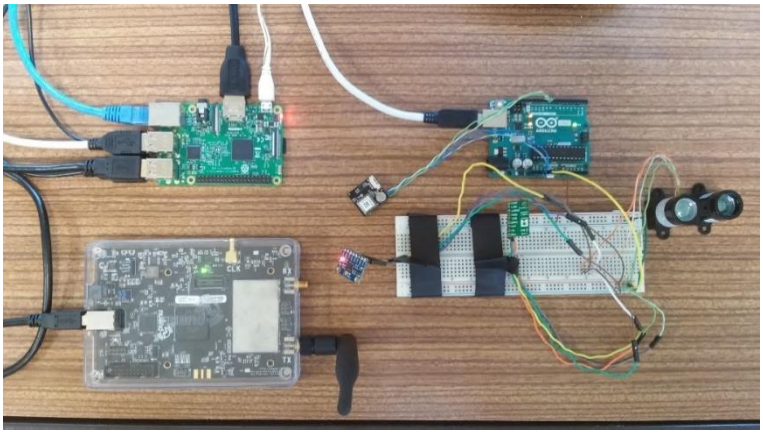

Figura A - 1 Protótipo laboratorial

Foi elaborado um protótipo laboratorial por forma a identificar possíveis erros de funcionamento e assim mitigar os mesmos antes de se instalar o sistema numa estrutura física (arma).

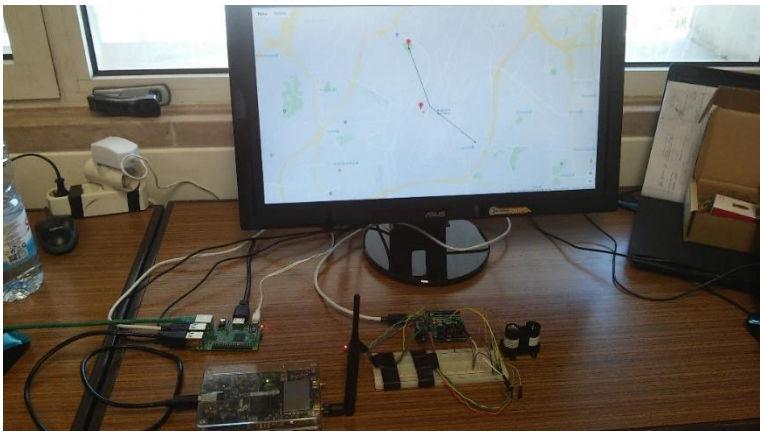

Figura A - 2 Resultados do protótipo laboratorial através de ecrâ

A.2. Protótipo Real

Inicialmente foi editado um projeto disponível gratuitamente em *Autodesk Inventor Professional 2018 Student version* de uma arma real que serviu de inspiração (Colt m4a1 Carbine), modificando o projeto, nomeadamente, adicionando as alocações necessárias para integrar a tecnologia utilizada no sistema (Arduino, Raspberry, sensores e BaldeRF), e retirando partes desnecessárias na estrutura da arma como o cano e o carregador Figura A - 3.

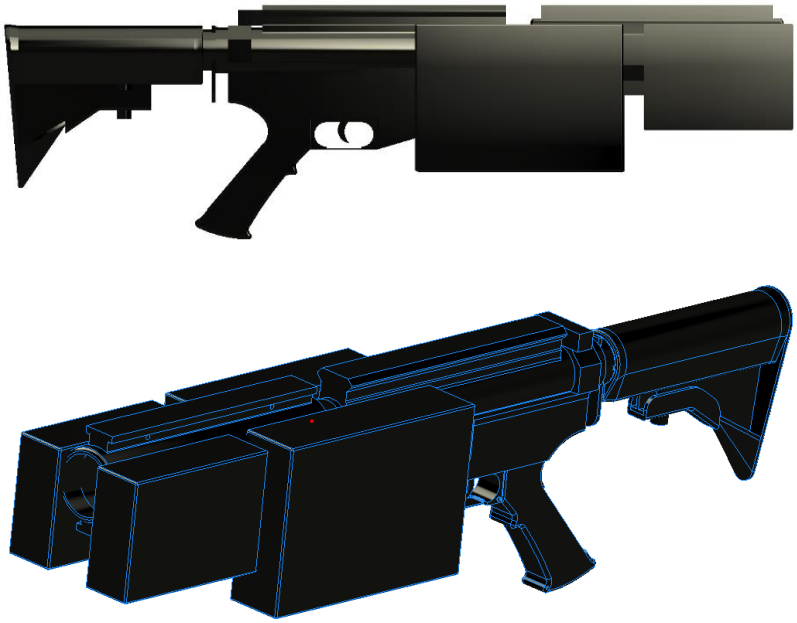

Figura A - 3 Projeto Autodesk Inventor Profissional de protótipo real

Depois de retiradas todas as medidas das dimensões das plataformas integrantes do sistema, iniciou-se a maquinação das caixas onde seriam alocadas essas mesmas tecnologias. O processo de construção passou desde a utilização de uma máquina fresadora para conseguir a profundidade das caixas (processo de perfuração para encaixe das tampas), até à elaboração dos orifícios para os cabos dos periféricos ligados às plataformas, Figura A - 4, foram desenvolvidas no âmbito do trabalho apresentado.

Figura A - 4 Construção das caixas para as plataformas que integram o sistema

Após a construção das quatro caixas para as plataformas, projetou-se a estrutura e *design* da arma, nomeadamente, comprimentos, espessura e largura da mesma. Todas as medidas foram designadas, contando com toda a eletrónica que seria inserida na arma, representado pela Figura A - 5 abaixo.

Figura A - 5 Estrutura principal do protótipo

Pode-se verificar, através da Figura A - 6 como foram acomodados, na estrutura do protótipo, os vários interruptores necessários para o funcionamento do sistema, como os gatilhos, interruptor de alimentação e interruptores de seleção de modo de funcionamento.

Mas também o espaço criado, na coronha da arma, para a bateria responsável pela alimentação de todo o sistema.

Figura A - 6 Integração de gatilhos, seletores de modo e interruptor de alimentação

Na Figura A - 7 é demonstrado como se encontram ligados, no interior do protótipo, todos os componentes eletrónicos do protótipo, nomeadamente os sensores, recetor de GPS e alimentação para todas as plataformas que integram o sistema de *spoofing* desenvolvido.

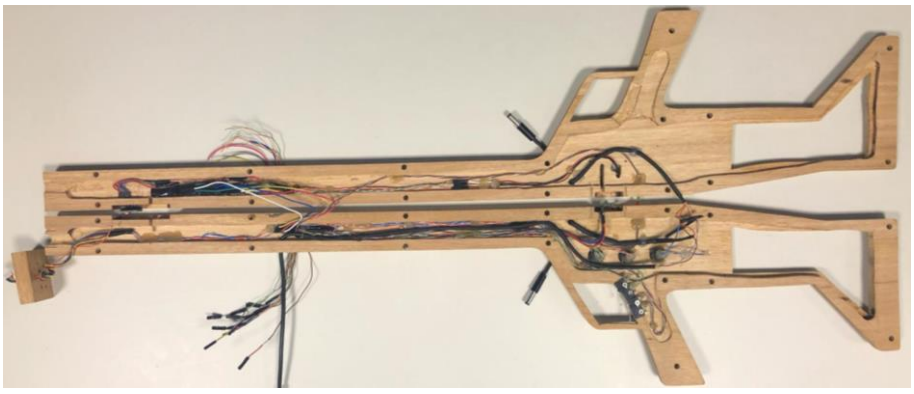

Figura A - 7 Ligações electrónicas e de alimentação no interior do protótipo

Apêndice B. Manual de Utilizador

Neste apêndice são apresentados os procedimentos iniciais antes de utilizar o sistema de *spoofing* e como utilizar o mesmo.

B.1. Procedimentos pré-utilização

Procedimentos iniciais antes de utilizar o sistema de *spoofing*:

- Verificar ligações de todos os sensores;
- Verificar ligação do recetor GPS;
- Verificar o nível de carga da bateria;
- Confirmar ligação entre Arduino e Raspberry Pi por USB;
- Confirmar ligação entre plataformas SDR e Raspberry Pi;
- Confirmar ligações entre plataformas SDR e antena Yagi.

B.2. Procedimentos de utilização

Após respeitar e confirmar todos os procedimentos de pré-utilização descritos anteriormente, encontram-se reunidas todas as condições para utilizar o sistema de *spoofing*. Mas antes apresenta-se abaixo, com auxílio da Figura B - 1, uma descrição e localização de todos os componentes que integram o protótipo:

1. Sensor Lidar posicionado na frente da arma;
2. Plataforma Arduino com todos os sensores e recetor GPS ligados;
3. Plataforma BladeRF 1;
4. Gatilho um;
5. Gatilho dois;
6. Recetor de GPS;
7. Bateria Li-PO;
8. Sensor MPU6050;
9. Sensor lsm303d;
10. LED verde;
11. LED vermelho;
12. Plataforma Raspberry Pi;
13. Platafroma BladeRF 2;

14. Interruptor de seletor de modo de *spoofing*;
15. Interruptor de seletor entre *jamming* e *spoofing*;
16. Interruptor geral de alimentação do sistema.

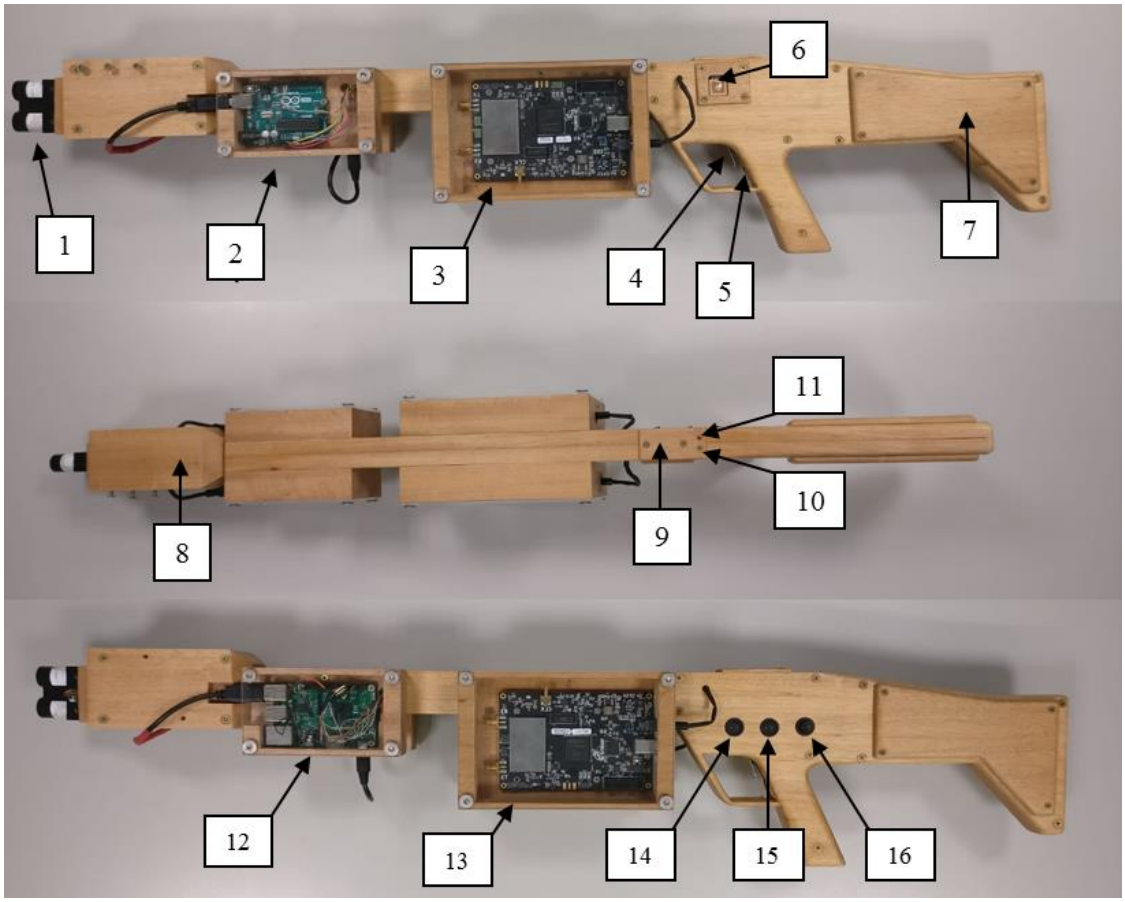

Figura B - 1 Descrição e localização de cada componente do protótipo desenvolvido

Em primeiro lugar o utilizador necessita de ligar o sistema utilizando o interruptor 16 da Figura B - 1. Após uns breves segundos o sistema iniciou-se totalmente e encontra-se preparado atuar.

Partindo do pressuposto que o desejado pelo utilizador é utilizar o *spoofing* integrado no protótipo, então o interruptor 15 da Figura B - 1 deve encontrar-se na posição 1.

De seguida, conforme a situação a atuar, poderá utilizar um dos dois modos de *spoofing* disponíveis no sistema:

Cenário 1: drone encontra-se num voo estático, ou seja, a planar na mesma posição, então o interruptor 14 da Figura B - 1 deve estar na posição 0;

Cenário 2: drone encontra-se num voo dinâmico, ou seja, em deslocamento, então o interruptor 14 da Figura B - 1 deve estar na posição 1.

Cenário 1

O utilizador apenas necessita de apontar a arma diretamente para o drone e premir o interruptor 4 da Figura B - 1 (Gatilho um). De seguida, o utilizador deve manter sempre o interruptor pressionado e a arma direcionada para o drone enquanto são medidos os valores dos sensores. Se a medição dos sensores não for bem-sucedida o LED vermelho, componente 11 da Figura B - 1, acende e assim o utilizador recebe resposta visual do sistema que necessita de retirar novas medições. Para tal, o utilizador apenas necessita de soltar e premir novamente o gatilho, e ao conseguir bem as medições dos sensores, o LED verde, componente 10 da Figura B - 1, acende informando o utilizador das corretas medições. O sistema efetua, em simultâneo, os cálculos necessários para determinar a localização do drone e construção da localização falsa a transmitir. De seguida, o sistema inicia a transmissão dos sinais de GPS falsos e ambos os LEDs verde e vermelho ligam e desligam de forma intermitente, indicando ao utilizador que o sistema encontra-se a transmitir. Para terminar a transmissão, o utilizador apenas necessita de soltar o gatilho e o sistema termina a transmissão e está preparado para novas medições.

Cenário 2

O utilizador necessita de apontar a arma diretamente para o drone e retirar duas localizações. O processo de retirar as medições é efectuado da mesma forma que no Cenário 1. Retirando as primeiras medições o LED verde, componente 10 da Figura B - 1, liga e desliga, de forma intermitente, sinalizando o utilizador que a primeira medição já se encontra efetuada com sucesso. A segunda medição requer que o utilizador largue o gatilho e volte a premir o mesmo. De seguida, com a segunda medição bem-sucedida, o sistema efetua os cálculos para determinar as duas localizações e constrói a rota contrária para levar o drone em direção à área de aterragem. É iniciada a transmissão dos sinais de GPS falsos e os LEDs verde e vermelho ligam e desligam de forma intermitente. Para terminar a transmissão, o utilizador apenas necessita de soltar o gatilho e o sistema termina a transmissão e está preparado para novas medições.

Anexo C

C.1. *Datasheet* Lidar

Physical

Specification	Measurement
Size (LxWxH)	20 × 48 × 40 mm (0.8 × 1.9 × 1.6 in.)
Weight	22 g (0.78 oz.)
Operating temperature	-20 to 60°C (-4 to 140°F)

Electrical

Specification	Measurement
Power	5 Vdc nominal 4.5 Vdc min., 5.5 Vdc max.
Current consumption	105 mA idle 135 mA continuous operation

Performance

Specification	Measurement
Range (70% reflective target)	40 m (131 ft)
Resolution	+/- 1 cm (0.4 in.)
Accuracy < 5 m	±2.5 cm (1 in.) typical*
Accuracy ≥ 5 m	±10 cm (3.9 in.) typical Mean ±1% of distance maximum Ripple ±1% of distance maximum
Update rate (70% Reflective Target)	270 Hz typical 650 Hz fast mode** >1000 Hz short range only
Repetition rate	~50 Hz default 500 Hz max

*Nonlinearity present below 1 m (39.4 in.)

**Reduced sensitivity

Interface

Specification	Measurement
User interface	I2C PWM External trigger
I2C interface	Fast-mode (400 kbit/s) Default 7-bit address 0x62 Internal register access & control
PWM interface	External trigger input PWM output proportional to distance at 10 µs/cm

Laser

Specification	Measurement
Wavelength	905 nm (nominal)
Total laser power (peak)	1.3 W
Mode of operation	Pulsed (256 pulse max. pulse train)
Pulse width	0.5 µs (50% duty Cycle)
Pulse train repetition frequency	10-20 KHz nominal
Energy per pulse	<280 nJ
Beam diameter at laser aperture	12 × 2 mm (0.47 × 0.08 in.)
Divergence	8 mRadian

C.2. *Datasheet* MPU6050

Parameter	Rating
Supply Voltage, VDD	-0.5V to +6V
VLOGIC Input Voltage Level (MPU-6050)	-0.5V to VDD + 0.5V
REGOUT	-0.5V to 2V
Input Voltage Level (CLKIN, AUX_DA, AD0, FSYNC, INT, SCL, SDA)	-0.5V to VDD + 0.5V
CPOUT (2.5V ≤ VDD ≤ 3.6V)	-0.5V to 30V
Acceleration (Any Axis, unpowered)	10,000g for 0.2ms
Operating Temperature Range	-40°C to +105°C
Storage Temperature Range	-40°C to +125°C
Electrostatic Discharge (ESD) Protection	2kV (HBM); 200V (MM)
Latch-up	JEDEC Class II (2),125°C ±100mA

Cross-Axis Sensitivity vs. Orientation Error

Orientation Error (θ or Φ)	Cross-Axis Sensitivity (sinθ or sinΦ)
0º	0%
0.5º	0.87%
1º	1.75%

C.3. *Datasheet* lsm303d

Features

- 3 magnetic field channels and 3 acceleration channels
- From ±1.3 to ±8.1 gauss magnetic field full-scale
- ±2g/±4g/±8g/±16g selectable full-scale
- 16 bit data output
- I^2C serial interface
- Analog supply voltage 2.16 V to 3.6 V
- Power-down mode/ low-power mode
- 2 independent programmable interrupt generators for free-fall and motion detection
- Embedded temperature sensor
- Embedded FIFO
- 6D/4D orientation detection
- ECOPACK® RoHS and "Green" compliant

Applications

- Compensated compass
- Map rotation
- Position detection

Symbol	Parameter	Test conditions	Min.	Typ.(1)	Max.	Unit
Vdd	Supply voltage	-	2.16		3.6	V
Vdd_IO	Module power supply for I/O		1.71	1.8	Vdd+0.1	
Idd	Current consumption in normal mode(2)			110		µA
IddSL	Current consumption in sleep-mode(3)			1		µA
Top	Operating temperature range		-40		+85	°C

1. Typical specifications are not guaranteed.
2. Magnetic sensor setting ODR = 7.5 Hz, Accelerometer sensor ODR = 50 Hz.
3. Linear accelerometer in sleep-mode and magnetic sensor in power-down mode.