Politécnico
de Viseu

Escola Superior
de Tecnologia
e Gestão de Viseu

SISTEMA DE ALERTA PARA PREVENÇÃO DE ACIDENTES COM MÁQUINAS AGRÍCOLAS (Tratores)

Pedro Correa Persson

Dissertação

Mestrado em Engenharia Mecânica e Gestão Industrial

Trabalho efetuado sob a orientação de
Professora Doutora Olga Maria Sousa Contente
Professor Doutor Daniel Filipe Albuquerque

Março de 2023

Politécnico
de Viseu

Escola Superior
de Tecnologia
e Gestão de Viseu

SISTEMA DE ALERTA PARA PREVENÇÃO DE ACIDENTES COM MÁQUINAS AGRÍCOLAS (MOTOS 4)

Pedro Correa Persson

Dissertação

Mestrado em Engenharia Mecânica e Gestão Industrial

Trabalho efetuado sob a orientação de

Professora Doutora Olga Maria Sousa Contente
Professor Doutor Daniel Filipe Albuquerque

Março de 2023

Agradecimentos

Agradeço minha família por todo o suporte e compreensão, que sem esse não seria capaz de ter concluído esse longo e árduo caminho. Agradeço o meu amigo Nathaniel Fitch e a toda a comunidade Sturgeon's House por não só me fazer ter uma melhor compreensão do que é ser um bom engenheiro, mas também ter uma melhor visão sobre o nosso mundo. Agradeço aos meus orientadores e à equipa AgroSafeBox pelo suporte e pela oportunidade de participar esse projeto que abriu novos horizontes para mim. Agradeço aos meus orientadores Olga Contente e Daniel Albuquerque pelo seu papel de bons professores. Por fim e não menos importante agradeço aos meus gatos, sempre ao meu lado fiéis e compreensivos mesmo nos piores momentos.

Resumo

Os acidentes envolvendo tombamento de veículos agrícolas, em especial os tratores e moto 4, são muito comuns e apresentam alto índice de fatalidade devido às características destes veículos e do seu ambiente de utilização. O objetivo principal desta dissertação consiste na criação de um sistema de alerta ao condutor, com base num modelo de estabilidade dinâmica do veículo (a escolher e validar). Para tal, foram estudados diversos modelos matemáticos de estabilidade e métodos para a obtenção das variáveis necessárias (centro de gravidade, momentos de inércia e ângulos críticos) para criação do modelo do veículo. Para a determinação do índice de estabilidade, que é um valor adimensional para caracterizar se o veículo está estável ou em não, é necessário conhecer em tempo real as acelerações lineares e as velocidades angulares do veículo. Nesse sentido, foi criado um sistema de monitorização e aquisição de dados utilizando uma unidade de medição inercial (IMU) e um programa em linguagem Python com uma interface gráfica do utilizador (GUI). De modo a obter um ambiente controlado, os ensaios foram inicialmente realizados com um veículo à escala. Os algoritmos de estabilidade e o sistema de monitorização e aquisição de dados foram validados, utilizando o veículo à escala, num plano inclinado ajustável onde um obstáculo simula um evento de instabilidade dinâmica. Os resultados obtidos demonstram que o sistema implementado consegue prever a iminência do tombamento.

Palavras-Chave

Tratores, Veículos agrícolas, Estabilidade de veículos, Estabilidade dinâmica, Prevenção de acidentes, IMU

Abstract

Accidents involving the overturning of agriculture vehicles, especially tractors e ATVs, are quite common and have a high fatality rate due to those vehicle's characteristics and their work environment. The aim of this thesis consists in the creation of an alert system to the operator, based on a dynamic stability model of the vehicle. To this end, several mathematical models of stability and methods to obtain the necessary variables (center of mass, moment f inertia and critical angles) were reviewed to create the vehicle's model. For the determination of the stability index, it's necessary to know in real time the linear accelerations and angular velocities of the vehicle. To this end, a system for data monitoring and acquisition was developed, using an inertial measuring unit (IMU) and a program developed in the Python language with a graphical user interface (GUI). To obtain a controlled environment, the tests were initially conducted with a reduced scale model of the vehicle. The stability algorithm and the data monitoring and acquisition system were validated, using the vehicle in an adjustable inclination plane where an obstacle was placed to simulate an event of dynamic instability. The results obtained demonstrate that the implemented system can predict the imminence of the overturning.

Keywords

Tractors, Agricultural Vehicles, Vehicle Stability, Dynamic Stability, Accident Prevention, IMU

Índice Geral

Índice de figuras

Índice de Tabelas

Siglas

ATV	*All Terrain Vehicle* (Veículo Todo Terreno/Moto-Quatro)
CG	Centro de Gravidade
CSV	*Comma-separated values* (Valores Separados por Virgula)
ESTGV	Escola Superior de Tecnologia e Gestão de Viseu
GPS	*Global Positioning System* (Sistema de Posicionamento Global)
GUI	*Graphical User Interface* (Interface Gráfica do Utilizador)
IMU	*Inertial Measurement Unit* (Unidade de medição inercial)
IPV	Instituto Politécnico de Viseu
LED	*Light-Emitting Diode* (Diodo Emissor de Luz)
RP4	*Raspberry Pi 4*
TSI	*Índice de Estabilidade do Trator*
USB	*Universal Serial Bus* (Porta Serial Universal)

Notação

Maiúsculas Latinas

$\overline{I_{xx}}$	Momento de inércia no eixo de tombamento
$\overline{I_{zz}}$	Momento de inércia no eixo empinamento
I_{xx}	Momento de inércia no eixo X
I_{zz}	Momento de inércia no eixo Z

U_{CG}	Velocidade absoluta do Centro de Gravidade
U_{XX}	Velocidade de deslocamento do veículo
X_L	Bitola do veículo
Y_{CG}	Altura do CG

Minúsculas Latinas

X_{abs}	Distância absoluta do CG ao eixo de empinamento no plano XY
Y_{CGcrit}	Altura crítica do CG
Y_{CGf}	Altura final do CG
Z_{abs}	Distância absoluta do CG ao eixo de tombamento no plano ZY
r_2	Raio da roda traseira
m	Massa total do veículo

Minúsculas Gregas

α_x	Ângulo no eixo X
α_z	Ângulo no eixo Z
ω_x	Velocidade angular no eixo de tombamento
ω_z	Velocidade angular no eixo de empinamento
Υ	Ângulo de tombamento
β	Declive frontal
θ	Ângulo do terreno
λ	Ângulo de empinamento
ϕ	Ângulo critico

Símbolos

$\times$	Operador de produto vetorial

1 Introdução e Motivação

A segurança do trabalho em tratores e outros veículos agrícolas é de extrema importância para sociedade, pois, apesar do trator ser um equipamento indispensável na agricultura moderna é também muito perigoso.

Este Trabalho é cofinanciado pelo Centro 2020 – Portugal 2020 e FEDER, projeto Centro 01-0247-FEDER-047199 em copromoção de I&D "AgroSafeBox: Intelligent Alert System for AgroVehicles Rollover and Driver Safety", executado em consórcio pela ProjectBox, empresa promotora e os copromotores são IPV, UA e TIS.

1.1 Objetivos

O principal objetivo deste trabalho é o desenvolvimento de um sistema de alerta do tombamento para de máquinas agrícola (tratores e motos-quatros), capaz de informar o seu operador sobre o risco de tombamento em situações dinâmicas, como as geradas pela colisão com um obstáculo, onde apenas a medição da inclinação do veículo não é suficiente. O objetivo secundário desse trabalho são o desenvolvimento de um algoritmo de estabilidade capaz de discernir essas situações com base em sensores de fácil obtenção e que não requeira modificação no veículo em estudo; condição necessária para atender o principal objetivo. Consequentemente também são objetivos terciários o desenvolvimento de equipamentos e softwares necessários para a aplicação e validação do algoritmo.

1.2 Metodologia

A metodologia adotada neste trabalho envolve em uma revisão da literatura, seguida por uma exposição teórica dos métodos e ferramentas que foram utilizados. Depois tendo sido criados os circuitos da box e os programas necessários para o seu funcionamento. Posteriormente a realização de ensaios em ambiente controlado (em laboratório), seguidos por ensaios em campo com veículos reais, tendo o intuito de validar a ideia do sistema proposto.

O trabalho decorre em três fases distintas. A primeira foi o desenvolvimento de um algoritmo que, tendo em conta as grandezas físicas adquiridas, permita avaliar a estabilidade do veículo, por análise dinâmica. A segunda é a criação de um protótipo que permita adquirir todas as grandezas envolvidas em tempo real para a validação dos algoritmos desenvolvidos. A terceira fase consiste no estudo e implementação do sistema e da fase de testes em condições reais num trator à escala. Esta última fase, permitirá verificar e validar o modelo e trabalho

realizado nas duas primeiras fases, além de avaliar e corrigir atempadamente quaisquer imprevistos que possam ocorrer na instalação dos equipamentos no trator real.

1.3 Estrutura do Documento

Este documento encontra-se dividido em seis capítulos. No presente capítulo são expostas as motivações e objetivos do trabalho. No capítulo 2 é feita uma exposição do estado da arte caracterizando o problema e os veículos envolvidos (tratores e moto quatros). No capítulo 3 é apresentado o modelo de estabilidade, assim como os métodos necessários para a obtenção de variáveis relevantes como o centro de gravidade (CG) e os momentos de inércia. O capítulo 4 aborda todo o sistema de obtenção e monitoração de dados. Identifica e justifica a escolha dos sensores, microcontroladores, softwares e abordagens utilizados. O capítulo 5 contém as descrições dos métodos, do equipamento e dos valores de referência estipulados (setups) durante os ensaios experimentais, tanto com modelos em escala como com um trator real. E também contém os resultados e o tratamento de dados. Por fim o capítulo 6 apresenta as conclusões do trabalho e considerações para trabalho os futuros.

2 Estado da Arte

A segurança do trabalho em tratores e outros veículos agrícolas é de extrema importância para sociedade, pois, apesar do trator ser um equipamento indispensável na agricultura moderna é também muito perigoso. O trator representa 32% das fatalidades na agricultura, mais de 50% dos acidentes envolvem o tombamento (M S Abubakar, et al, 2010). A maioria dos acidentes de tombamento ocorre em terrenos acidentados, principalmente no caso de inclinação lateral, com o trator realizando manobras, mas também pode ocorrer em terrenos planos ao atingir algum obstáculo devido à geometria do veículo (nenhuma suspensão, entrei eixos e bitola curta, elevado centro de gravidade, etc.) (M.S. Abubakar, et al, 2010). Medidas de proteção ao tombamento como o arco de proteção (Rollover Protection Structure) e as cabines fechadas (que também protegem o condutor das intempéries) reduzem as chances de ferimento e morte durante um tombamento e já são implementadas na agricultura com alguma frequência, porém são medidas remediais que não previnem o tombamento (M.S. Abubakar, et al, 2010).

Devido a esta importância o estudo do tombamento e a sua predição é de extrema relevância. Os testes em ambiente real não são práticos de realizar devido à enorme variedade de tratores e configurações existentes (bitola pode ser ajustada, como as dimensões dos pneus, tipos de alfais, entre outros parâmetros do trator), (Zhen Li, et al, 2016), o que acarreta demora, custo e atualização constante; mesmo utilizando modelos em escala. Para tal é desejável criar um modelo matemático da estabilidade do trator que possa realizar simulação por via de computadores. Esses modelos podem ser de vários tipos, nomeadamente: estáticos, quase estáticos, ou dinâmicos (Iman Ahmad, 2011). Os modelos estáticos são os mais simples e assumem uma velocidade, inclinação e taxa de curva constantes. Ao passo que no modelo dinâmico essas variáveis não são fixas, o que melhora a simulação da realidade, porém torna o método muito mais complexo.

Em geral existem três grandes categorias de modelos de estabilidade, os baseados na geometria do veículo, na energia do sistema e na relação entre as forças de contato. O primeiro é adequado a situações estáticas por não levar em conta efeitos dinâmicos. Os métodos que levam em conta a energia do sistema fazem uma comparação entre a energia potencial do

veículo com a energia cinética do seu movimento relativo aos eixos de rotação, é um método quase-estático que já foi utilizado em veículos fora de estrada a baixa velocidade. O terceiro método monitoriza a força de contato das rodas, a sua carga, e tenta descobrir o momento em que existe descolamento entre a roda e o solo, condição necessária, mas não suficiente para o tombamento, porém tem o inconveniente do tipo de sensor necessário (Steven C. Peters & Karl Iagnemma, 2008).

Cada tipo de modelo matemático também possui inúmeras variações e estratégias de ataque ao problema. Por exemplo, utilizado tanto a conservação de energia do sistema como as condições de atrito entre as rodas do trator e o solo, se observa que existe uma relação muito forte na estabilidade geral do trator com o limite de derrapagem, mais relevante que a predição direta do angulo de tombamento (Iman Ahmad, 2011). Outro método, o momento de estabilidades, que se baseia na distribuição das forças de contato entro o solo e as rodas do veículo, porém de forma indireta, o que minimiza os custos e inconvenientes relativos aos sensores utilizados (Steven C. Peters & Karl Iagnemma, 2008).

A simulação da estabilidade de tratores responde apenas a parte do problema, é necessário método de medição e predição em tempo real do veículo para que se tenha utilidade real; sistemas proativos de prevenção ao tombamento possam ser criados. É necessária uma grande quantidade de sensores para se ter medições das variáveis relevantes, porém devem ser robustos, discretos e de fácil instalação (Pedro Renato Pereira Barros, 2012). Em especial os sensores que medem força (células de carga) apresentam um maior desafio, na instalação, que inclinômetros e acelerómetros já que a maioria dos tratores não possuem suspensão. No caso de existir elementos de suspensão a aplicação é relativamente direta e fácil, como no caso de atuadores hidráulicos (D. Denis, et al, 2016). A instalação de células de carga em tratores mais comuns é possível, porém os métodos utilizados não são práticos fora do ambiente experimental, como células de carga em uma estrutura entre a roda e a parede interna dos pneus (Carlos Rodrigues dos Santos Neto, 2012). Algo que dificulta ainda mais a utilização desse último método é o fato de uma parte dos tratores utilizarem lastros líquidos dentro dos pneus. A comunicação dos sensores dentro das rodas também pode ser problemática devido a impossibilidade de se utilizarem cabos.

Outra parte importante do problema se dá na sua aplicação, o fator humano. Como a interação dos sistemas de proteção com os utilizadores e autoridades se dará? Qual é a percepção e atitude de condutores e proprietários a essa e outras tecnologias, quais são os obstáculos legais e ergométricos que poderão ocorrer? Um dos grandes problemas do fator humano é a natureza da agricultura europeia, em especial a Portuguesa, Italiana e outras, que

são extremamente fragmentadas com inúmeras pequenas propriedades, o que não só dificulta o estudo preciso do problema como também deve ser levado em conta a menor capacidade financeira dessas propriedades e natureza que visa subsistência, situação em que novas tecnologias podem ser consideradas como “desnecessárias”. Esse grupo que pode ser chamado de “relutantes” representam em torno de 21% de uma amostra realizada na Itália. Por outro lado, o grupo mais aberto a adoção de novas tecnologias os “donos inovadores” são normalmente proprietários de fazendas maiores, enquanto existe um terceiro grupo os “culturalmente dispostos” que são uma mistura dos dois, estão abertos a inovações, mas por alguma razão ainda não as possui, representam 25% da amostra (Eugenio Cavallo, et al, 2014).

Além do fator comportamental, também deve ser levado em conta, para além da prevenção é o que fazer após um acidente. Enquanto a maioria da literatura se limita a parte teórica dos modelos de estabilidade, ou a parte experimental, alguns trabalhos focam na parte pós acidente. Uma característica comum a vários trabalhos é a utilização de um sistema automático que informa a localização, data e hora do acidente a autoridades e serviços médicos, o que garante que um pedido de socorro seja enviado mesmo que o condutor do veículo esteja incapacitado. Existem vários métodos para tal, como SMS, chadas telefónicas e e-mails (B. Liu & A.B. Koc, 2015).

Com essa breve explicação dos trabalhos e técnicas utilizadas para explorar o tema, o nosso projeto pode agora ter um direcionamento mais claro. É obvio que o modelo a ser utilizado é o dinâmico, pois oferece uma simulação mais completa e adequada para utilização fora de estrada, além de ser mais flexível podendo ser utilizada para quadriciclos (obviamente com diferentes parâmetros e levando em conta o fato desses veículos possuírem suspensões.) Também fica claro que o tipo de sensor a ser utilizado é o IMU, nomeadamente o MPU-9250 pois é um equipamento barato, confiável e versátil podendo ser utilizado por diversos tipos de controladores e computadores (Raspberry Pi, Arduino, etc.) com a linguagem Python. Mais especificamente deverá ser feito uma medição indireta da força de contato de cada roda com o solo por via de IMU.

Na parte de software é planejado a utilização de um ou mais controladores do tipo Raspberry Pi, se comunicando entre si e com um controlador central por via de websocket. Cada Raspberry Pi, cliente, faz a medição do seu sensor, converte os dados do acelerómetro e giroscópio para a medição de angulo e envia os dados para o controlador central, host, que por sua vez utiliza os dados dos clientes para realizar a simulação em tempo real da estabilidade do veículo, além de realizar o log da estabilidade do veículo, e de acordo com a situação alerta o

condutor ou enviar mensagens por SMS e Email para as autoridades e serviços de socorro no caso de tombamento.

O trabalho pode ser separado em três grandes áreas. A primeira é a criação de um algoritmo para a averiguação da estabilidade do veículo, pela técnica dinâmica. O segundo é a criação de um programa que utiliza esse algoritmo, assim como a comunicação entre os computadores e o envio de mensagens externas. A terceira parte é averiguação e implementação do sistema, testes reais em escala e no trator real. Não será possível utilizar programas de simulação (ex: ADAMS e Car Sim) para testar o sistema devido a restrições de técnicas e financeira. Essa última parte tem a importante finalidade de verificar a validade do trabalho realizado nas duas primeiras partes, além de descobrir e corrigir qualquer nuance ou imprevistos que possam ocorrer na instalação dos equipamentos no trator real.

Até o momento, o trabalho realizado foi o de atualização bibliográfica, para se determinar soluções e técnicas já exploradas por outros grupos. Também foi realizado a instalação e teste de sensores IMU e de GPS no Raspberry Pi, o estudo da comunicação por websocket e um programa em linguagem Python para a medição do angulo com base no IMU MPU-9250. Por fim foi feito um outro programa para a simulação da estabilidade de um trator, porém não em tempo real e com base no modelo estático exposto por um trabalho anterior do grupo (H. Silva, et al, 2017).

Apesar de não estar contemplado no escopo do nosso trabalho, ao ter um sistema de controlo de estabilidade em tempo real do veículo, abre-se a possibilidade se incorporar outras tecnologias de prevenção de acidentes e mitigação dos seus danos, como por exemplo sistemas de ROPS automáticos (Tomás Ballesteros, et al, 2013).

2.1 Tratores

Os tratores agrícolas são máquinas autopropelidas projetadas para tracionar, transportar e fornecer potência para máquinas e implementos agrícolas. São veículos complexos, empregados para impelir ou fornecer força estacionária para uma larga variedade de implementos agrícolas. (Varella, 2011).

Os tratores agrícolas são constituídos de motor, sistema de transmissão, sistema hidráulico e rodados. Todos esses componentes estão montados em uma estrutura denominada chassi. O chassi é a estrutura geral do trator, formada pela união de todos os seus órgãos constituintes e deve oferecer resistência aos esforços de torção provenientes da tração. Os tratores agrícolas podem ser montados em quatro tipos de estruturas de chassis: Monobloco, Chassi propriamente

dito, Semichassi e Chassi articulado (Varella, 2011). Para efeitos desse trabalho, não foi considerado o chassi articulado, pois é um tipo de chassi menos comum e ao contrário dos outros tipos de chassi, deve ser tratado como dois corpos rígidos interconectados, mas com grande liberdade de movimento entre si; é um caso especial que deve ser tratado a parte.

Mais importante para esse trabalho é o sistema de suspensão e a bitola, normalmente a suspenção é rígida e a bitola dianteira pode ter a sua dimensão ajustada (normalmente sempre é mais estreita que a bitola traseira).

Ou seja, um trator "normal", independentemente do seu tipo de chassis, motor e tração pode ser tratado como um único corpo rígido.

2.2 Moto Quatro

Os quadriciclos ou moto quatros, são pequenos veículos fora de estrada, normalmente para um ocupante, mas podem transportar até quatro pessoas, ou seja, variam de tamanho entre uma moto e um carro subcompacto. Normalmente possuem um sistema de tração nas quatro rodas e um sistema de suspenção independente de grande curso com rodas e pneus adequados para a tração em terrenos não pavimentados.

São normalmente utilizados para atividades recreativas, para deslocações dentro de propriedades, mas também tem utilidade como veículo de trabalho já que podem transportar cargas, equipamentos e até alfais e outros implementos agrícolas. Podem ser considerados na sua utilização como pequenos tratores.

O ajuste da suspenção para a utilização fora de estrada (grande curso e molas "macias") paradoxalmente tornam o veículo sujeito ao tombamento e capotamento.

As motos-quatro são também veículos muito mais rápidos que tratores. Os fatores anteriormente mencionados tornam a análise do movimento destas duas tipologias de veículos agrícolas diferente. As motos-quatro devem ser tratadas como um conjunto de corpos independentes, com movimentação entre si diferentes.

3 Análise do Modelo

Para compreender os modelos de estabilidade estáticos e dinâmico utilizados neste trabalho é importante conhecer algumas métricas do veículo em estudo. Nomeadamente certas características físicas e geométricas como sejam a: bitola, a distância entre eixos, a massa total

do veículo, os raios das rodas dianteiras e traseiras. Esses são dados de fácil obtenção, tanto por medições em campo quanto como por consulta a catálogos de fabricantes ou mesmo a especificações publicadas em revistas técnicas. No entanto também são necessárias algumas outras características do veículo que não tão universalmente disponíveis como: a localização do CG e a distribuição de massa em cada eixo. Alguns fabricantes disponibilizam essas últimas informações, porém não é prática comum; portanto é necessário determiná-las.

3.1 Algoritmo de Estabilidade

O modelo de estabilidade dinâmico que é utilizado no trabalho é baseado no modelo de estabilidade desenvolvido por Ahmadi 2003. É um modelo matemático que utiliza a conservação de energia total do CG com o fim de se obter o TSI. Tem especial utilidade no caso mais comum e perigoso de tombamento, que é quando um trator está se deslocando ao longo de uma encosta e atinge um obstáculo, o que faz com que o veículo que estava estaticamente estável se tornar dinamicamente instável e tombar.

O modelo de Ahmadi é aplicável apenas para tratores e outros veículos sem suspensão, pois trata o veículo como um único corpo rígido, mas para essa aplicação é de longe o modelo mais simples de se utilizar. Foram feitas algumas modificações do modelo para ser utilizado nesse trabalho devido ao fato do artigo de Ahmadi ter um caráter puramente teórico, com todas as variáveis utilizadas sendo conhecidas de antemão. Por exemplo no artigo, Ahmadi, já conhece as dimensões e geometria do obstáculo que o trator irá encontrar, a inclinação do terreno etc. No nosso caso o obstáculo será sempre uma incógnita que terá de ser deduzido indiretamente através dos dados obtidos pelo IMU. O modelo de Ahmadi também é puramente o algoritmo de estabilidade, sem levar em conta o processamento de dados, ou mesmo como eles foram obtidos pelos sensores ou quais tipos de sensores.

O nosso modelo, utiliza uma série de variáveis que podem ser classificadas em dois grupos, os dados do trator (a sua geometria, massa, distribuição de peso e momento de inércia), e os dados obtidos pelo IMU que são as velocidades angulares e as acelerações lineares. A partir dessas variáveis se pode obter outras como a localização do CG e os ângulos nos eixos X e Z.

A distância direta do CG até o eixo de empinamento (veio traseiro do trator) ao longo do plano XY é X_{abs}, e é calculada trigonometricamente pela diferença da altura do CG e do raio da roda traseira e pela distância de CG no eixo X. A distância direta do CG até o eixo de tombamento (extremo da bitola média do trator) ao longo do plano ZY é Z_{abs} que também pode ser calculada por via trigonométrica, e é a altura crítica Y_{CGcrit} do CG (altura do CG em que

este está sobre a linha vertical sobre o ponto de apoio, ou seja a altura máxima do CG para se ter equilíbrio estático). Os eixos de coordenadas podem ser observados nas figuras 1 e 2.

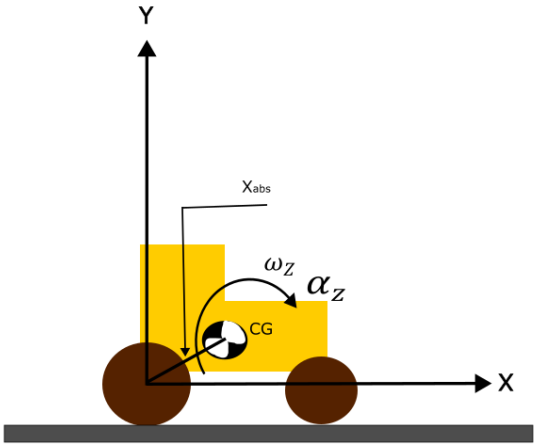

Figura 1 - Plano XY

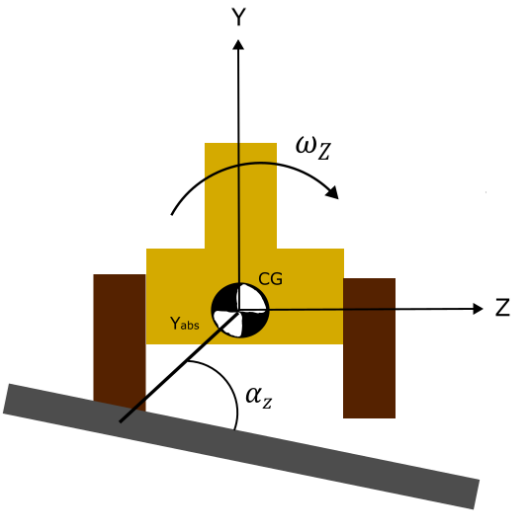

Figura 2 - Plano YZ

Os ângulos de tombamento e empinamento são dados por Υ e λ nas equações (3.1) e (3.2) respetivamente:

$$\Upsilon = \hat{a}ngulo\ de\ tombamento = \alpha_x - \theta \qquad (3.1)$$

$$\lambda = \hat{a}ngulo\ de\ empinamento = \alpha_y - \beta \qquad (3.2)$$

O angulo do terreno θ é obtido por base nas medições do IMU, através da equação (3.3), no nosso caso o programa em Python para o registo de dados tem a funcionalidade de inclinómetro. Contudo devido a utilização do IMU para esse fim, o programa está constantemente a medir o angulo, do trator, que no caso de tombamento ou de atingir um obstáculo será diferente, sendo necessário guardar o último ângulo medido antes de um evento de rápida mudança de velocidade e aceleração como sendo o valor da inclinação do terreno.

$$\theta = \hat{a}ngulo\ do\ terreno \qquad (3.3)$$

O ângulo crítico ϕ pode ser obtido experimentalmente ou calculado com base na geometria do veículo.

$$\phi = \hat{a}ngulo\ crítico = arctg\left(\frac{2Y_{CG}}{X_L}\right) \qquad (3.4)$$

A velocidade do CG é U_{CG} é calculada pelo produto vetorial da velocidade de deslocamento linear do trator U_{XX}, as velocidades angulares e os seus raios.

$$U_{CG} = \overrightarrow{U_{XX}} + \overrightarrow{\omega_X} \times \overrightarrow{Z_{abs}} + \overrightarrow{\omega_Z} \times \overrightarrow{X_{abs}}, com\ \overrightarrow{U_{XX}} = U_{XX} \cdot \vec{\imath} \qquad (3.5)$$

$$\overrightarrow{\omega_X} \times \overrightarrow{X_{abs}} = d_r \cdot \omega_X \cdot (\cos(\lambda \cdot \vec{\jmath}) - \sin(\lambda \cdot \vec{\imath})) = h_{crit} \cdot \omega_Z \cdot (\cos(\lambda) - sen(\lambda)) \qquad (3.6)$$

$$\begin{aligned} \overrightarrow{\omega_Z} \times \overrightarrow{Z_{abs}} &= d_r \cdot \omega_X \cdot \left(\cos(\Upsilon \cdot \vec{\jmath}) - \sin\left(\Upsilon * \vec{k}\right)\right) \\ &= h_{crit} \cdot \omega_X \cdot (\cos(\Upsilon) - sen(\Upsilon)) \end{aligned} \qquad (3.7)$$

Os momentos de inércia utilizados são em relação aos eixos de tombamento e de empinamento, contudo normalmente os momentos de inércia são obtidos em relação ao CG. Sendo necessário utilizar o teorema de Steiner ou teorema dos eixos paralelos.

$$\overline{I_{XX}} = I_{xx} + m \cdot {Z_{abs}}^2 \qquad (3.8)$$

$$\overline{I_{ZZ}} = I_{yy} + m \cdot {X_{abs}}^2 \quad (3.9)$$

A altura final do CG é Y_{CGf} que é obtida através da seguinte equação:

$$Y_{CGf} = \frac{m \cdot ({U_{CG}}^2 - {U_{XX}}^2)}{2} + \frac{\overline{I_{XX}} \cdot \omega_r^2 + \overline{I_{ZZ}} \cdot \omega_X^2}{2} + (Y_{CGcrit} \cdot \sin(\theta + \phi) \quad (3.10)$$

O TSI é o valor final a ser obtido, um número adimensional que pode variar de 0 a 1 e indica o grau de instabilidade do trator. Com base no TSI se pode definir graus de risco e consequentemente alertar o condutor.

$$TSI = \frac{Y_{CGcrit} - Y_{CGf}}{Y_{CGcrit} - Y_{CG},} \quad (3.11)$$

Como fica evidente pela equação (3.11) e pela geometria do trator, a altura crítica nunca pode ser igual ou menor que a altura do CG, e quanto maior a altura crítica menor será o TSI. Por sua vez também pode se observar que quando a altura final é igual a altura critica o TSI é igual a zero. No caso de o trator estar em um plano nivelado a altura final é igual a altura estática do CG e o TSI é igual a 1. Portanto O TSI varia entre 0 e 1, de menos estável a mais estável.

3.2 Estudo dos Ângulos Críticos

Para determinação das coordenadas X e Z do centro de gravidade do trator recorreu-se ao equilíbrio de momentos em torno dos eixos longitudinal e transversal. Foi escolhido um método simplificado, o método da dupla pesagem, baseado no método utilizado por (Silva, et al, 2017), para medir a coordenada Y do centro de gravidade. O método da dupla pesagem consiste na utilização de uma balança sob cada uma das rodas de um modelo de trator em escala reduzida, com o intuito de medir a força de reação em cada roda.

Inicialmente a medição foi realizada num plano horizontal (nivelado a 0°), como ilustrado pela figura 1, em seguida o conjunto foi inclinado em relação a um eixo paralelo ao eixo longitudinal do trator em incrementos de 5° até atingir os 30°. Outro ensaio foi realizado

com a mesma metodologia, porém inclinando o trator em relação ao seu eixo transversal (coincidente com uma paralela ao eixo traseiro). Esses ensaios estão representados pelas figuras 4 e 5 respetivamente.

A partir dos dados obtidos pelos ensaios dá-se características geométricas do trator, utilizando as equações deduzidas por (Silva, et al, 2017), é possível definir as coordenadas do centro de massa do trator e consequentemente os seus ângulos de inclinação críticos (ângulo que representa o limiar do tombamento num senário estático).

Figura 3 - Medição do centro de massa e ângulos críticos

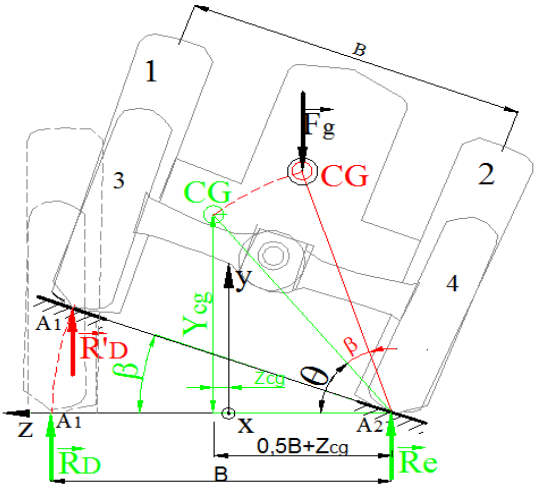

Figura 4 – Esquema para ensaio de tombamento

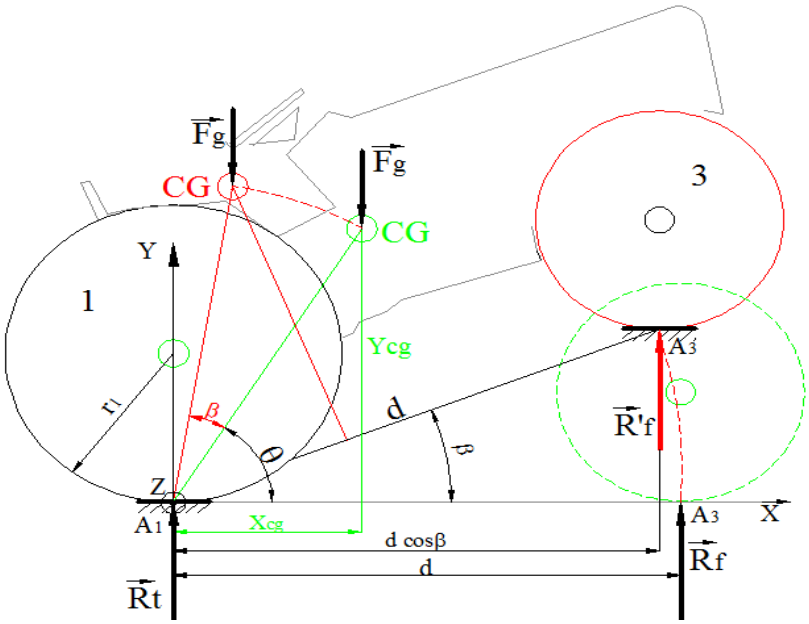

Figura 5 – Esquema para ensaio de empinamento

Os resultados obtidos pelos ensaios, juntamente com outras características do trator, foram compilados na tabela 1

Tabela 1 - Características do modelo à escala do Trator FENDT 1050

Dados Geométricos	Valor
Distância entre eixos - X_L [mm]	194
Comprimento da bitola dianteira - b [mm]	136

Comprimento da bitola traseira – B [mm]	119
Raio roda dianteira - r_1 [mm]	50
Raio roda traseira - r_2 [mm]	62
Massa do trator - m (kg)	1,366
Coordenada X do CG - X_{CG}[mm]	96,1
Coordenada Y CG - Y_{CG} [mm]	92
Coordenada Z CG - Z_{CG} [mm]	1,2
Ângulo critico de tombamento - β (°)	33,3
Ângulo critico de empinamento - β (°)	46,17

3.3 Determinação do Momento de Inércia

Os modelos de estabilidade dinâmicos contemplados necessitam de um parâmetro intrínseco do veículo a mais em relação aos modelos estáticos: O momento de inercia. A obtenção desse parâmetro é necessária tanto em modelos em escala reduzida de tratores e outros veículos, assim como os veículos reais ou qualquer outra maquete ou corpo utilizado para testes. Porém é uma característica notoriamente difícil de se obter diretamente, sendo necessário acesso ao projeto do veículo ou a sua recriação fidedigna, o que é algo absolutamente irreal considerando o elevado número de peças que um trator possui e as consequentes horas trabalho necessárias para a sua recriação. Também não é conveniente depender da consulta a fabricantes de veículos para a obtenção desse dado, já pode existir um desfasamento entre os dados de projeto e um certo veículo na realidade, também existe a questão da cooperação com os fabricantes.

Portanto um método indireto de determinação do momento de inércia é desejado. Porém esse tipo de método também tem os seus inconvenientes como por exemplo limitações no tamanho e tipo de veículo (métodos de pêndulo e suspensão) ou possuem equipamentos de grandes dimensões e elevado custo.

Por essas razões, para o caso do trator real, foi sugerida utilização do método descrito no artigo “Experimental determination of moments of inertia for an off-road vehicle in a regular engineering laboratory” (P.E. Uys et al, 2006). Oferecendo uma solução pratica e simplificada, criada contemplando as condições e equipamentos encontrados em um típico laboratório de engenharia mecânica de uma instituição de ensino superior.

O método consiste na obtenção do momento de inercia através da sua relação matemática com o fator de amortecimento de um sistema em oscilação. Tal sistema sendo o veículo em estudo, suportado em um lado por um pivô e no outro por uma mola, sendo assim possível criar uma oscilação periódica.

Os equipamentos necessários são simples e de fácil acesso: Células de carga, molas com características conhecidas, simples suportes metálicos, sistema de monitoração de dados etc. Podendo esses testes ser realizados tanto em escala como em veículos reais com os equipamentos disponíveis nos laboratórios da ESTGV (Escola Superior de Tecnologia e Gestão de Viseu).

Outra característica que torna esse método bastante atrativo para ser aplicado neste trabalho é o fato de tratores poderem ser simplificados como um único corpo rígido, não sendo necessário lidar com os problemas e inconvenientes de um sistema de suspensão, que iria requerer o seu isolamento do corpo principal do veículo e o seu próprio estudo. Infelizmente tal simplificação não é possível no caso das motos-quatros, porém o método contempla veículos com suspensão.

3.4 Projeto e Construção de Máquina para Medir Experimentalmente os Momentos de Inércia

Foi decidido pela utilização de dois métodos, um para o modelo de trator em escala reduzida e outro para o trator real. Para o primeiro caso, as dimensões e massas relativamente pequenas do modelo permitem uma manipulação mais fácil, o que permite a utilização de um dispositivo do tipo mesa giratória. No caso do trator real, como já referido na seção anterior, o método adequado é o de Uys.

Para esse fim, foi projetado uma máquina, figura 6 que consiste em um veio colocado na vertical sobre rolamentos, ao qual são fixados um tambor e uma barra de fixação do objeto em estudo. Por sua vez o tambor está conectado a um cabo que passa por uma polia e orienta-o na vertical, no final desse cabo está um peso de massa conhecida. Com isso o peso está livre para cair sobre o efeito da força gravítica, por sua vez fazendo o veio e o objeto em estudo girar. Um diagrama de corpo livre do sistema está ilustrado na figura 7.

Ao comparar o tempo que leva para o peso cair com a máquina vazia, com o tempo da máquina com o objeto é possível determinar o momento de inercia deste.

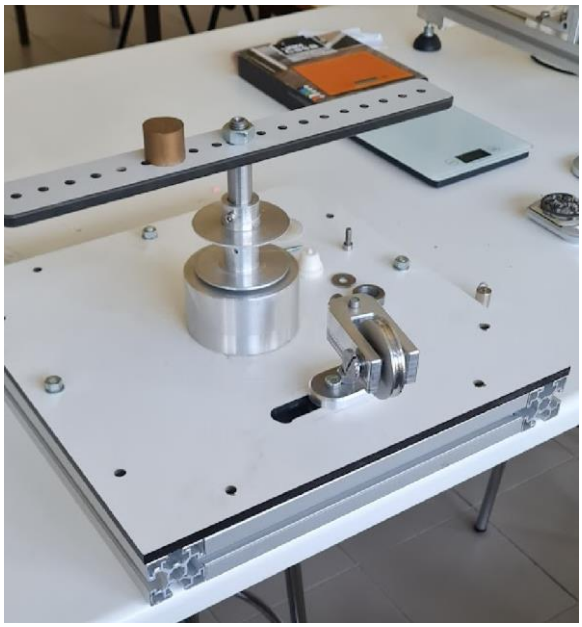

Figura 6 - Dispositivo para medição do momento de inércia

Antes de realizar os ensaios com o trator apoiado na mesa giratória, foram também realizados uma série de ensaios com a máquina vazia e com um disco de ferro de massa e geometria conhecidas, com o intuito de calibrar a máquina

O momento de inércia do trator foi obtido pelas seguintes equações utilizando o método energético utilizando as seguintes equações:

$$m \cdot g \cdot h = \frac{m \cdot v^2}{2} + \frac{I \cdot \omega^2}{2} \qquad (3.12)$$

A primeira equação (3.12) é a do balanço energético do sistema, no lado esquerdo está a energia potencial do bloco de massa m suspenso a uma altura h e com a aceleração gravítica g. No lado direito da equação está a energia cinética do bloco com velocidade linear v, e também a energia cinética rotacional do dispositivo com I sendo o momento de inércia e ω a velocidade angular.

$$m \cdot g \cdot h = \frac{m \cdot v^2}{2} + \frac{I \cdot \omega^2}{2} + f \cdot t \qquad (3.13)$$

A segunda equação (3.13) é obtida ao adicionar um novo termo no balanço energético, com o objetivo de contabilizar as perdas do sistema (devido ao atrito) que é o produto da taxa de consumo energético f pelo tempo t de queda do bloco.

$$\frac{I \cdot \omega^2}{2} = f \cdot t' \qquad (3.14)$$

A terceira equação (3.14) é aplicada separadamente, com o objetivo de se obter a taxa de consumo energético em que t' representa o tempo que a máquina leva para parar a sua rotação após a queda do bloco, sendo obtido o momento de inercia para um dado eixo resolvendo essa equação pondo I em evidencia. Os momentos de inércia obtidos pelos ensaios estão na tabela 2.

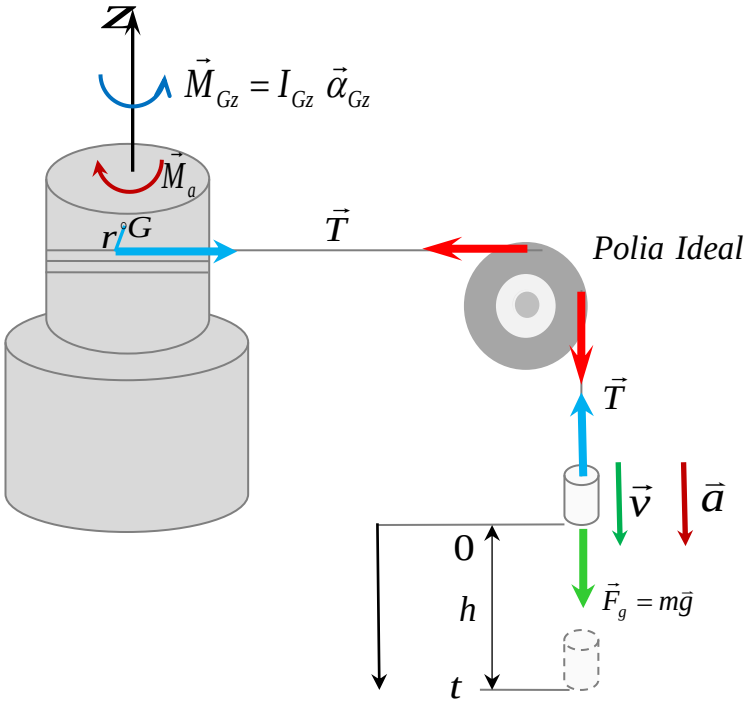

Figura 7 - Diagrama de corpo livre do sistema

Tabela 2 - Momentos de inércia do trator FENDT 1050

Momento de inércia no eixo	Valor
I_{XX} - [kg · m^2]	0,03099
I_{YY} - [kg · m^2]	0,01205
I_{ZZ} - [kg · m^2]	0,01739

4 Sistema de Aquisição e Monitorização de Dados

Para a aquisição e processamento de dados optou-se por usar um microcomputador *Raspberry PI 4B*, uma vez que se trata de um PC de pequenas dimensões que permite a conexão de sensores externos. Para além da aquisição de dados, a utilização de microcomputador permite a implementação de algoritmos complexos em tempo real devido ao seu grande poder

de cálculo. Adicionalmente o *Raspberry PI 4B* foi dotado de bateria o que o permitirá a sua utilização em ambientes remotos operando de forma autónoma.

Para a aquisição das variáveis relevantes para o algoritmo de estabilidade foi usado um sensor de medição inercial, do inglês *Inertial Measurement Unit* (IMU). Este tipo de sensor é, normalmente, constituído por acelerómetros e giroscópios, podendo conter em certos casos outro tipo de sensores tais como bússolas. Um IMU permite determinar o movimento e orientação de um dado objeto, veículo ou pessoa de maneira autossuficiente (sem recorrer a um referencial externo) através das acelerações lineares e velocidades angular (acelerómetro e giroscópio, respetivamente) (Zhao, J., 2018).

A arquitetura do sistema está representada na figura 8. O sistema pode ser dividido em dois blocos funcionais destintos, o primeiro, a instalar no veículo, é responsável pela aquisição e registos de todos os dados (num ficheiro CSV) a cada 5 ms durante a realização da experiência. Este bloco também é responsável pelo envio dos dados, via *WiFi* (a cada 100 ms), para o segundo bloco do sistema, o qual é responsável pela visualização e monitorização dos dados em tempo real.

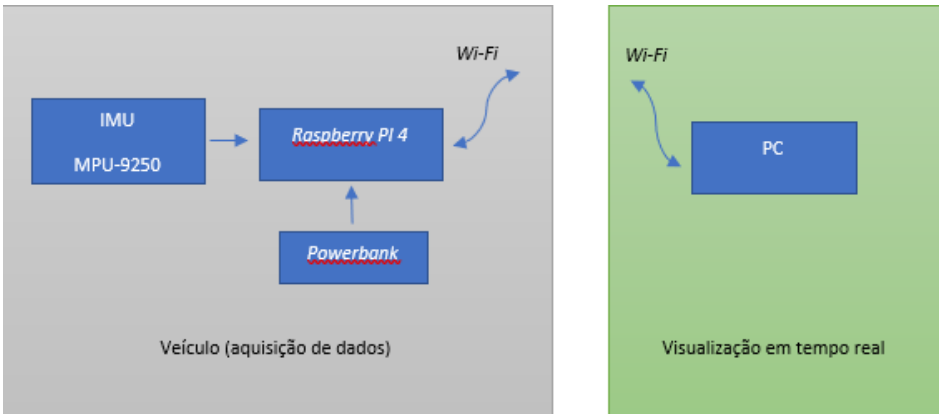

Figura 8 - Fluxograma da comunicação entre os componentes do sistema de aquisição de dados

Tendo em consideração os requisitos do sistema a desenvolver (precisão, custo, facilidade de utilização, etc.) optou-se por utilizar o MPU-9250 da empresa *InvenSense* (TDK) estando as suas principais características presentes na Tabela 3. O MPU-9250 é um IMU de 9 eixos que contém um acelerómetro de 3 eixos, um giroscópio de 3 eixos e uma bússola também de 3 eixos.

Tabela 3 - Características dos sensores do MPU-9250

Acelerómetro	
Amplitude	$\pm 2, \pm 4, \pm 8 \; e \; \pm 16 \; g$
Resolução	0,0060 até 0,00479 m/s^2
Tendência do Zero Inicial	X,Y: 60 mg; Z: 80 mg
Variação Inicial do Fator Escala Inicial	3%
Não-Linearidade	0,5%

Giroscópio	
Amplitude	$\pm 250, \pm 500, \pm 1000\ e\ \pm 2000°/s$
Resolução	0,00763 até 0,06098°/s
Taxa de Zero Inicial	± 5°/s
Variação Inicial do Fator Escala Inicial	3%
Não-Linearidade	0,1%
Magnetómetro	
Amplitude	± 4800 µT
Resolução	0,6 µT/LSB (14 bit) ou 15 µT/LSB (16 bit)
Variação Inicial do Fator Escala Inicial	5%

Neste trabalho optou-se por configurar o acelerómetro para operar a $\pm 8\ g$, enquanto o giroscópio está configurado a $\pm 1000°/s$. Resultando na seguinte resolução média $0{,}03049°/s$ para o giroscópio e $0{,}0002441\ g$ para o acelerómetro.

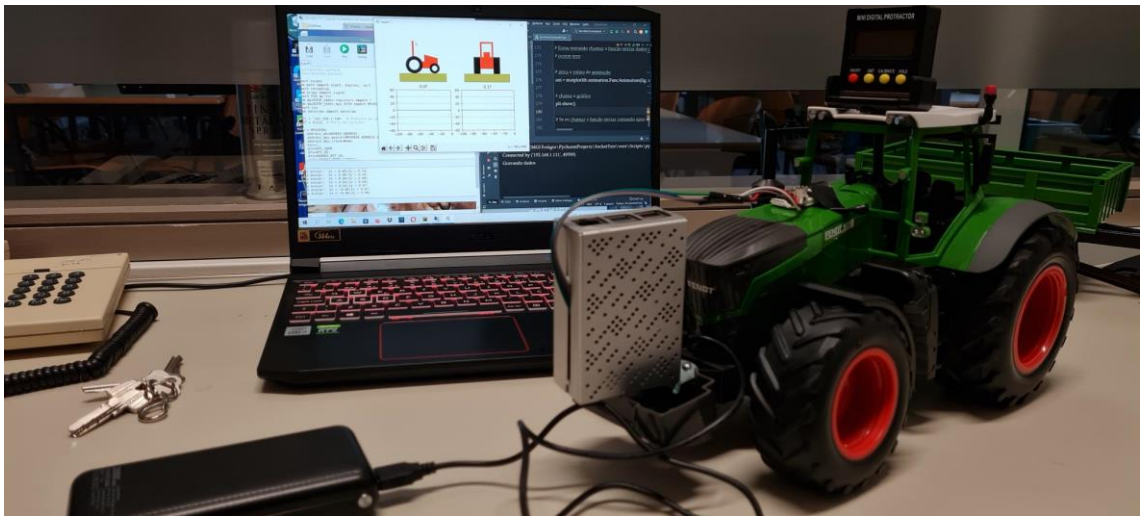

Figura 9 - Instalação do sistema de aquisição de dados

Para a determinação da inclinação do veículo recorreu-se à estimativa da direção da força gravítica obtida através das acelerações lineares fornecidas pelo IMU. Na verdade, a inclinação do veículo pode ser representada pela sua inclinação no eixo longitudinal, θ_L, e no eixo transversal, θ_T, as quais foram obtidas através das seguintes fórmulas:

$$\theta_L = \tan^{-1}\left(\frac{g_y}{g_z}\right) \tag{4.1}$$

$$\theta_T = \tan^{-1}\left(\frac{-g_x}{\sqrt{g_y^2+g_z^2}}\right) \tag{4.2}$$

Em que (g_x, g_y, g_z) é a estimativa do vetor da aceleração da gravidade em coordenadas cartesianas. Para a obtenção da estimativa do vetor da aceleração da gravidade as acelerações obtidas através do IMU (a_x, a_y, a_z) foram filtradas passo-baixo com uma frequência de corte de 0.1 Hz. Deste modo, será possível obter a componente constante presente no vetor aceleração a qual será uma boa estimativa do vetor da aceleração da gravidade.

4.1 Hardware

O Raspberry Pi 4 precisa de uma fonte de alimentação que deva cumprir uma série de requisitos para ser utilizada nos testes. Deve ser uma fonte que minimize a quantidade de cabos e conexões, dispensando a necessidade de uma tomada externa Tudo isto, para além de tornar o sistema de aquisição móvel, também elimina um cabo de alimentação externo que poderia interferir nos testes de campo. A fonte de alimentação também deve ser suficientemente compacta para poder ser instalada no modelo em escala reduzida do trator, preferencialmente não sendo maior que o próprio Raspberry Pi 4. Finalmente, será também ter em atenção a corrente elétrica máxima consumida e a tensão elétrica de operação, sendo neste caso, 3 A e 5 V respetivamente. Tendo tudo isto em consideração, optou-se por usar.

Um *powerbank* comercial. Tendo este a corrente e tensão necessárias, assim como uma capacidade de 6000 mAh o que garante uma boa autonomia do sistema.

Outra razão secundaria para escolha de um *powerbank* foi o facto de ser um equipamento externo conectado via cabo USB, não sendo necessário para tal qualquer modificação do hardware. Permitindo também a sua instalação remota em relação ao RP4, sendo mais versátil a sua instalação num modelo em escala reduzida ou num veículo real.

Para se obter uma aquisição de dados adequada, é necessário reduzir ao máximo o ruido do sinal, pequenas vibrações que geram resultados que não correspondem à realidade do comportamento relevante do sistema a ser medido. Portanto são necessários filtros que removem essas pequenas variações dos dados adquiridos, tornando-os mais consistente. Porém se o filtra for demasiadamente restritivo os dados adquiridos podem também não representar o comportamento real do sistema já que somente movimentos muito bruscos seriam adquiridos.

Para resolver esse problema de achar o meio termo ideal, que pode ser muito estreito, uma solução é a aplicação de dois filtros. Um digital e um físico. O filtro digital implementado é um do tipo passo abaixo, como será visto na secção 3.2.2, usando a biblioteca Python FIR, tem a função de remover o ruído do sinal após esse ter sido adquirido pelo sensor IMU e antes

de gravá-lo. Para o filtro físico, que remove o ruído do sinal antes desse ser adquirido pelo sensor, foi decidiu construir uma base antivibrações.

A base antivibrações consistem em três partes, uma base retangular fixa que é colocada diretamente no veículo, uma segunda base retangular, de menores dimensões a qual o sensor IMU é fixado, e um sistema de suspensão constituído por quatro pedaços de espuma ou material amortecedor que suspendem a segunda base. Dessa forma é criada uma ligação flexível entre a primeira e segunda base, capaz de absorver vibrações, mas permitindo que deslocamentos maiores e mais longos sejam transmitidos ao sensor.

O desenho da base antivibrações foi baseado em exemplos existentes no mercado, que consistem no mesmo arranjo, com as quatro peças absorventes nos cantos exteriores da segunda base, inclinados a 45 graus e conectados a pequenos braços na primeira base. O seu projeto foi realizado com recurso a software CAD (Autodesk Inventor) e a sua construção foi feita através de impressão 3D do tipo SLA (estereolitografia) utilizando uma resina fotossensível, dessa forma garantindo uma prototipagem rápida, eficiente e precisa.

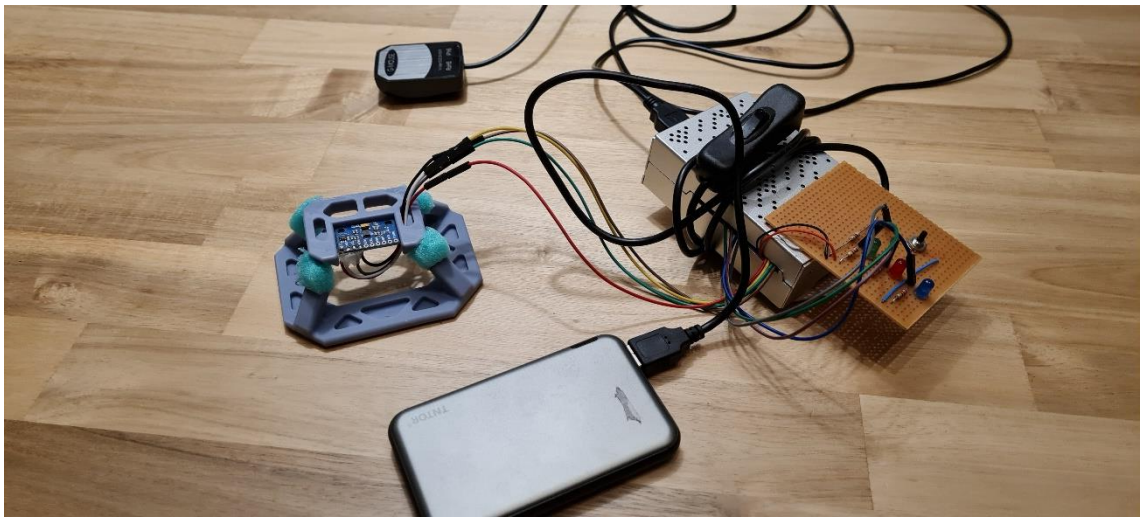

Figura 10 - Segunda versão do sistema de aquisição de dados para ensaios em campo, com a nova base antivibrações

Por fim, de salientar que tanto o filtro físico como o filtro digital necessitam de um ajuste para situações diferentes devido à natureza diferente do ruido para cada sistema e para cada situação de utilização.

A realização dos ensaios em campo, com um trator real a circular em caminhos rurais em condições normais de trabalho, traz alguns desafios únicos. O principal desafio é não contar com um ambiente totalmente controlado como nos ensaios realizados em laboratório com o modelo de trator. Em especial a conexão na mesma rede entre o servidor e o cliente, a uma

longa distância por um longo período é incerta. Por esse motivo foi escolhido a criação de um segundo programa em Python para os ensaios em campo.

Esse sistema, no seu cerne, realiza a mesma função de aquisição e registo de dados, porém é comandado de forma diferente, através de um botão e três LEDs para indicar o estado do registo dos dados em CSV. Dessa forma o sistema consiste apenas no IMU, uma *breadboard*, os sensores e a fonte de alimentação, figura 11 e a sua versão final na figura 13, e o circuito na figura 12.

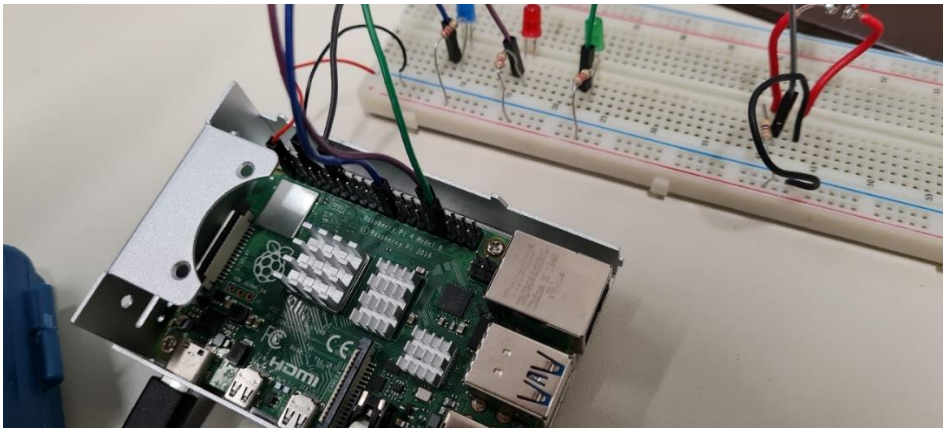

Figura 11 - Circuito e conexões do botão e LEDs multifunções

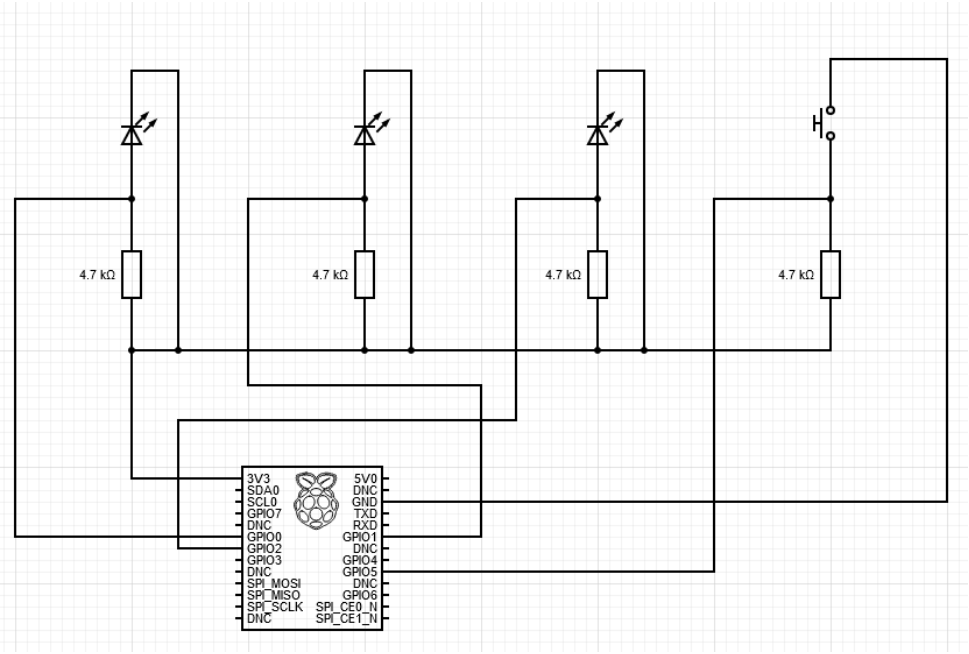

Figura 12 - Circuito do sistema LED e botão

Outro problema com o ensaio em campo, é a impossibilidade da utilização de um sistema de laboratório (por exemplo sensor ultrassónico) para a obtenção do deslocamento e consequentemente velocidade de deslocamento do trator. Obter esses dados diretamente do trator obrigaria à necessidade de modificar o veículo, o que não é prático nem desejável. Portanto foi decidido utilizar um sistema GPS para se obter esses dados. O modelo escolhido foi o VK-162 da empresa *G-Mouse* que é conectado ao RPB4 via USB e fornece dados de latitude, longitude, velocidade altitude. Dessa forma é possível, em um momento posterior a descrição aproximada (devido ao erro inerente do sistema GPS) do trajeto que o trator percorreu durante o ensaio. Esse módulo de GPS também possui uma base magnética, o que facilita a sua colocação no trator.

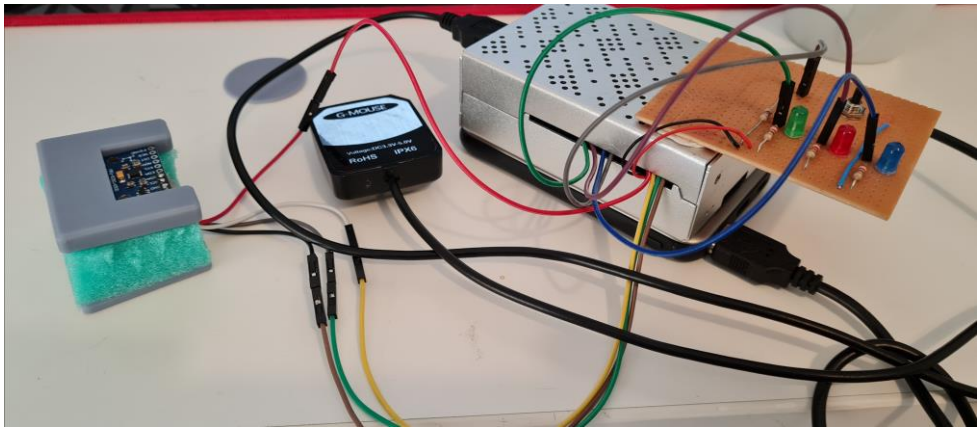

Figura 13 - Versão preliminar do sistema de aquisição de dados

4.2 Software

Apesar do RP4 ser um computador completo, com a sua própria GUI, é preciso ter acesso a esta. Obviamente, não é prático a utilização de uma ligação direta a um conjunto de periféricos (rato, teclado e monitor). Sendo necessário o comando do RP4 através de um acesso remoto a partir de outro computador, o que também é útil na sua utilização no dia a dia.

Porem o acesso remoto não é a melhor solução, devido a significativo *lag* o que torna o comando e controlo durante os testes pouco prático. Para tal, optou-se por recorrer a uma outra solução durante os testes para minimizar o *lag*.

A solução escolhida foi a programação em *socket*, que divide as funções do programa em dois programas distintos, o servidor e o cliente, cada um sendo executado num computador diferente. Foi decidido que o cliente seria o programa no RP4, que tem as funções de aquisição de dados do sensor e seu processamento, o registo de dados (ou *log*) em CSV no RP4 e por fim o envio dos ângulos para o servidor. O servidor tem a função de receber os dados de inclinação

gerados pelo cliente no RP4 e apresentá-los para o utilizador sob a forma de um gráfico com animação em tempo real, podendo assim ser verificado em tempo real a inclinação do trator nesses dois eixos.

O servidor também tem outra função, não critica para os ensaios, mas que traz grande comodidade, que é o envio de comandos para iniciar e terminar o registo de dados em CSV. Assim sendo, será possível começar e parar a gravação, assim como gravar múltiplos ficheiros de CSV (em sequência) sem ter de reiniciar a conexão entre o cliente e o servidor.

Por fim o programa cliente também possui uma funcionalidade de calibração do "zero" do sensor MPU9250, fornecida pela biblioteca utilizada pelo sensor (JMDEV).

Uma interface gráfica do utilizador (GUI) foi criada com o fim de se ter uma melhor visualização do comportamento do modelo de trator durante os ensaios. A GUI tem duas partes: A primeira consiste em uma animação da inclinação do modelo de trator nos eixos horizontais e transversais, assim como um gráfico desses dados ao longo do tempo, como se pode observar no exemplo da Figura 9. A segunda parte consiste num painel de controlo com botões para o comando da função de registo de dados em CSV, e uma luz de alerta que pode mudar de cor de verde, amarelo e vermelho de acordo com o grau de risco de tombamento ditado pelo TSI.

Em termos de programação foi escolhida a biblioteca *tkinter* para *Python* devido a sua simplicidade e fácil integração com versões anteriores do código.

5 Validação

No início dos trabalhos de dissertação, foram realizados uma série de testes, com o objetivo de averiguar a estabilidade de um modelo em escale reduzida de um trator. Tendo também o objetivo de validar a instrumentação e métodos de aquisição de dados criados até aquele momento. Em especial a validação do sensor IMU instalado no modelo (para adquirir as acelerações lineares e velocidades angulares) e um transdutor de deslocamento ultrassónico (para se obter a velocidade linear de deslocamento do modelo).

Após a realização com sucesso dos ensaios em laboratório, foram realizados ensaios de campo para averiguar como o sistema se comporta em situações reais, com velocidades mais elevadas, vibrações do pavimento e do motor de combustão, assim como qualquer imprevisto que possa surgir fora de um ambiente controlado de laboratório. Dessa forma foram realizados dois grupos de ensaios, o primeiro com um trator real e o segundo com um carro.

5.1 Ensaios em Laboratório

Para simular uma situação de instabilidade dinâmica foi construída uma maquete, que consiste numa prancha de madeira, afixada a um par de braços pantográficos atuados por parafusos sem fim. Sobre a superfície superior da prancha de madeira foi colocada uma manta de espuma para melhorar a tração da superfície. Também sobre a prancha foi afixado um obstáculo de madeira, na forma de uma rampa.

Com essa configuração é possível ajustar a inclinação da prancha, também é possível mudar o tamanho e formato do obstáculo. Dessa forma é possível simular situações que uma máquina agrícola pode encontrar durante a sua utilização, como o seu deslocamento ao longo de um declive e o impacto com um obstáculo, (como por exemplo uma rocha ou um tronco de árvore) que pode levar ao tombamento do trator, mesmo que este esteja estaticamente estável.

Para simular uma máquina agrícola foi adquirido um modelo de trator em escala reduzida. Esse modelo é controlado via comando rádio e é capaz de deslocamento linear para frente e para trás assim como mudança de direção. O modelo apesar de ser uma representação com fidelidade meramente estética, possui características em comum com um trator real, em

especial proporções geométricas similares e a ausência de suspensão (como na grande maioria dos tratores), podendo assim, para todos os efeitos, ser considerado como um corpo rígido.

Foram necessárias algumas modificações no modelo de trator, com o fim de permitir a instalação do sensor IMU, de um inclinómetro, do *powerbank* e do RP4. O IMO foi afixado a uma base de resina à frente da cabine, para garantir uma localização próxima do CG e uma colocação paralela ao chão. O inclinómetro foi colocado num suporte sobre a cabine, com o fim de permitir uma rápida verificação da inclinação do modelo e compara com a inclinação determinada pelo programa em *Python*. O RP4, tem uma pequena massa, porem tem um volume considerável, para além de ser necessário usar vários cabos para o conectar ao IMU, por esse motivo este foi colocado externamente na dianteira do modelo, na localização dos lastros dianteiros de um trator real. A monitorização dos dados em tempo real foi feita num computador externo conectado via *websocket* ao RP4, enquanto o registo foi feito no próprio RP4, no qual foram guardadas as variáveis relevantes num ficheiro do formato CSV.

Por fim o *powerbank* foi colocado na parte inferior do chassis do modelo, sob o eixo traseiro, afixado via fita dupla-face. Ao contrário do RP4 o powerbank possui dimensões reduzidas, mas uma massa considerável, tendo sido a sua localização escolhida com dois propósitos, uma menor interferência na altura do CG do modelo e uma distribuição de massa mais para o eixo traseiro, garantindo melhor tração do modelo.

Para a obtenção da velocidade de deslocamento linear do trator, foi um utilizado um transdutor de deslocamento ultrassónico. A monitorização e aquisição de dados foi feita por um computar com software específico instalado, sendo capaz de fornecer o deslocamento do modelo, a sua velocidade linear e a sua aceleração linear. O sensor foi instalado em uma das extremidades do plano inclinado, garantindo assim que se obtêm os dados mencionados independentemente da inclinação utilizada para um dado teste.

Para averiguar a estabilidade do modelo, este foi colocado sobre o plano inclinado, com um dado declive, a uma certa velocidade. Após atingir a sua velocidade máxima, o modelo então colide, com as rodas do seu lado direito, um obstáculo colocado na superfície do plano inclinado, criando assim uma situação de instabilidade dinâmica. Ao mesmo tempo os dois sensores adquirem as variáveis relevantes. A inclinação do plano é aumentada até que se observe o tombamento do modelo. Para garantir o funcionamento adequado do transdutor ultrassónico, é necessário colocar o modelo em escala reduzida do trator a uma distância inicial não inferior a 40 centímetros do sensor.

Os ângulos escolhidos do plano inclinados foram 0°, 10°, 20°, 25° e 30°, aumentando o angulo até que se obtenha uma situação de estabilidade critica (limiar do tombamento). Para

cada ângulo de inclinação, foram realizados uma série de 3 testes na mesma situação, com o objetivo de criar um valor médio e eliminar erros de medição.

Por fim foram realizados uma série de ensaios com os mesmos equipamentos do primeiro ensaio com o modelo em escala reduzida, mas com uma metodologia diferente. Com o intuito de se verificar se o TSI obtido através do algoritmo de estabilidade consegue prever o tombamento. Com os dados adquiridos nos ensaios sendo posteriormente alimentados no algoritmo de estabilidade numa folha de cálculo Excel.

A principal diferença de metodologia foi a colocação do trator, no momento inicial da aquisição de dados, num plano de referência para a calibração do IMU por um intervale de tempo curto para que os dados adquiridos pelo sensor se estabilizassem.

O primeiro ensaio foi de natureza estática, com o trator orientado no plano inclinado de forma a simular um declive lateral. A posição inicial do plano foi a horizontal (zero graus de inclinação), depois de um intervalo de tempo o plano foi inclinado gradualmente em passos de dez graus tendo um intervalo de tempo de espera a cada passo até que se observar o tombamento lateral do trator.

O segundo ensaio também foi de natureza estática e possuiu uma metodologia similar ao primeiro, porém com o trator orientado para simular a subida de um declive. O procedimento foi repetido até que ocorra o empinamento.

O terceiro ensaio foi de natureza dinâmica, com o trator a se deslocar ao longo do plano inclinado com uma inclinação de zero graus, sem atingir o obstáculo. O quarto ensaio dinâmico foi realizado com uma inclinação de vinte graus, atingindo o obstáculo. O quinto ensaio foi similar ao quarto, mas a dez graus de inclinação. O Sexto ensaio foi similar aos anteriores, mas com uma inclinação de vinte e cinco graus. Por fim o sétimo ensaio foi uma repetição do quarto ensaio, com o mesmo ângulo de 20 graus para verificar a ocorrência de tombamento.

5.1.1 Resultados

O sensor IMU registou aproximadamente a mesma inclinação do plano medida por um inclinómetro digital. Existindo em alguns casos uma discrepância de alguns graus. Tal discrepância pode ser explicada por dois motivos: a primeira é a instalação do sensor IMU que não está perfeitamente alinhado com os eixos do modelo o que pode acarretar erros de medição, mesmo tendo o seu zero calibrado. A segunda razão é que o modelo de trator foi colocado de forma manual, sem um controle exato da sua posição, então o angulo medido pode variar com a orientação do modelo no plano. Apesar de tais discrepâncias o sensor foi capaz de medir e

acompanhar a inclinação, aceleração e velocidade angular do modelo, registando o evento de instabilidade dinâmica seguido pelo retorno a estabilidade ou o tombamento.

Para os testes em que o modelo de trator transpõe o obstáculo com as duas rodas de um lado, não foi observado o tombamento para os ângulos de 0, 10 e 20 graus, sendo observado a ocorrência de tombamento em parte dos testes a partir dos 25 graus e o tombamento ocorreu em todos os testes realizados com uma inclinação de 30 graus, como pode ser visto na figura 14, onde se observa aos 7 segundos a colisão com o obstáculo, seguido pelo aumento de angulo até o ponto critico e depois o tombamento, sendo posteriormente retomado a posição normal manualmente..

Em contraste na figura 15 observa-se o mesmo comportamento inicial, mas com um pico máximo de inclinação muito menor e mais breve, retornando logo à a posição original.

Para os testes em que somente a roda traseira direita transpõe o obstáculo, realizados para os ângulos de 25 e 30 graus, obtiveram-se os mesmos resultados. Tombamento em alguns dos casos para 25 graus e em todos os testes em 30 graus.

Quanto aos resultados obtidos pelo transdutor ultrassónico, foi verificado uma velocidade

máxima de deslocamento do modelo de entre 0,4 e 0,5 m/s, um tanto similar, porém não idêntica à velocidade de 2,2 km/h fornecida pelo fabricante.

Seguindo a realização dos ensaios, os dados adquiridos pelo IMU e pelo transdutor foram compilados e tratados por meio de folhas de cálculo Excel. Também calculando a aceleração angular por meio da derivação da velocidade angular.

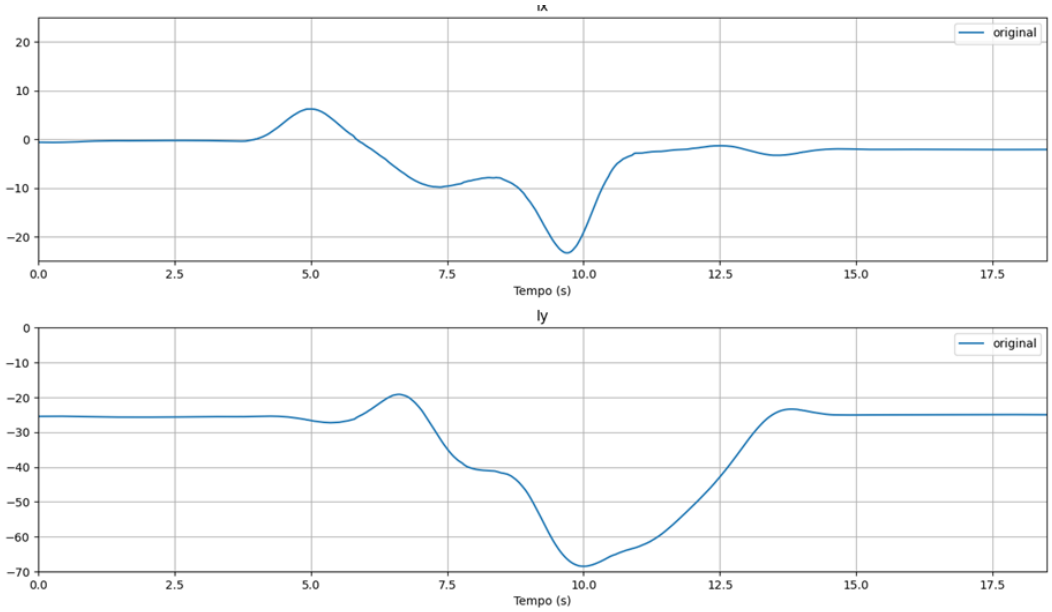

Figura 14 - Inclinação, teste 30 graus

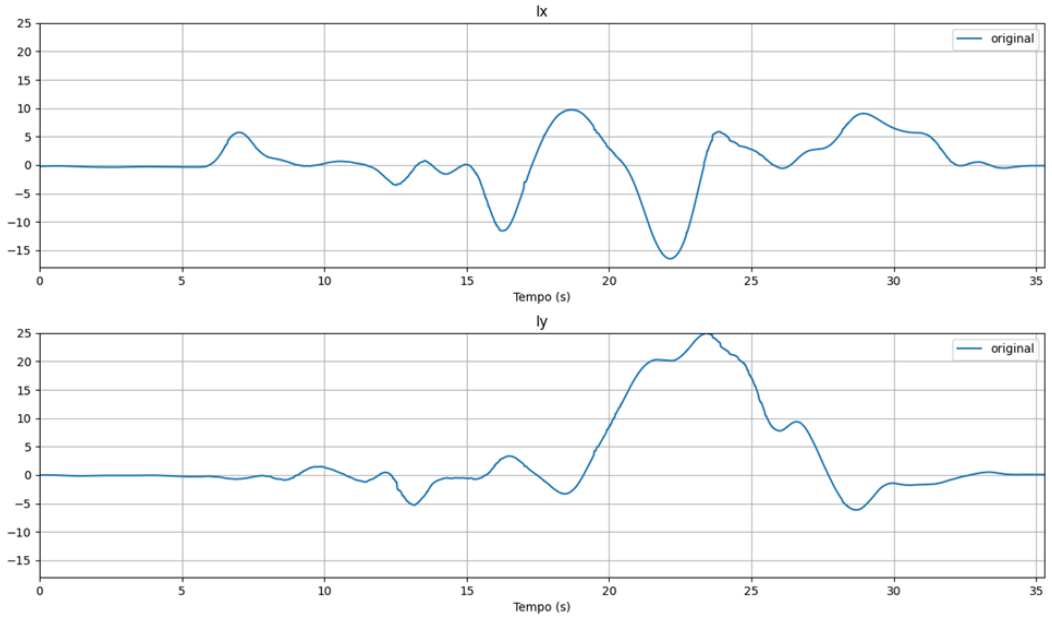

Figura 15 - Inclinação, teste 0 graus

No primeiro ensaio é possível observar que o TSI se manteve constante quando o trator está parado, tendo o valor inicial de aproximadamente 1 como seria esperado. Quando o plano é inclinado ocorrem duas situações, o TSI diminui e existe uma flutuação devido a operação manual do sistema de inclinação. Quando o plano atinge um patamar e para de ser inclinado, o valor do TSI estabiliza. Esse comportamento repete-se até que a inclinação atinge os 37,5 graus, o ângulo crítico do trator, o que acarreta o seu tombamento. Simultaneamente o TSI atinge o valor zero. Logo que o tombamento este foi interrompido manualmente e o trator reposto na posição horizontal.

Esses resultados, como podem ser vistos na figura 16, do primeiro ensaio mostram que o algoritmo de estabilidade funciona adequadamente para situações estáticas e o TSI também pode ser usado para essas situações. Os grandes picos do TSI e da inclinação no início e final do gráfico representam a manipulação do modelo de trator em escala, assim como a estabilização dos valores iniciais do IMU quando o programa de aquisição de dados é iniciado.

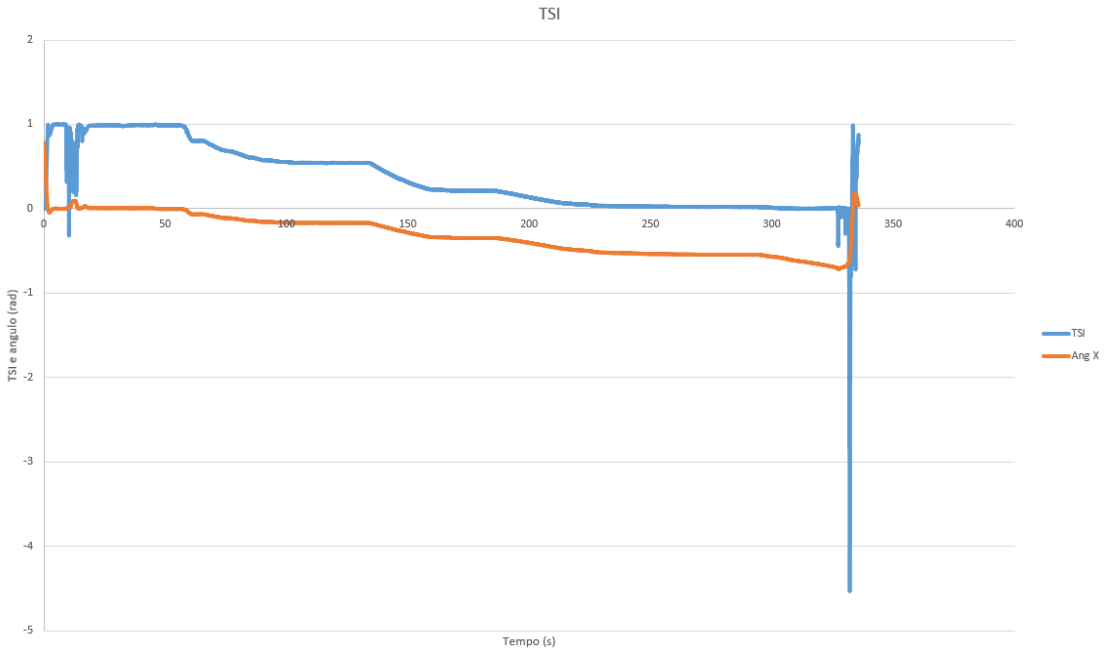

Figura 16 - Ensaio estático, com variação da inclinação no eixo longitudinal

O segundo, figura 17, ensaio demostra os mesmos resultados do primeiro, porém em relação ao empinamento, inclinação do eixo Y. Aproximadamente na metade do ensaio e no final foi observado a derrapagem do trato, com pouca alteração nas inclinações e nas velocidades angulares. Isso é refletido pelo ruido mais elevado do TSI, mas que não chega a acusar o empinamento. Nos momentos em que o TSI tem valor igual ou inferior a zero coincidem com intervenções manuais para parar a derrapagem do trator (por meio da colocação de calços).

Esse ensaio demostra que o algoritmo de estabilidade não é adequado para situações de derrapagem, apenas de tombamento e empinamento, mas este não é o objetivo do mesmo. Sendo necessário outro método para essas situações, já que a derrapagem por si só não acarreta riscos tão grandes como o tombamento e o empinamento, mas pode levar ao descontrole o veículo e consequentemente a essas situações de maior perigo.

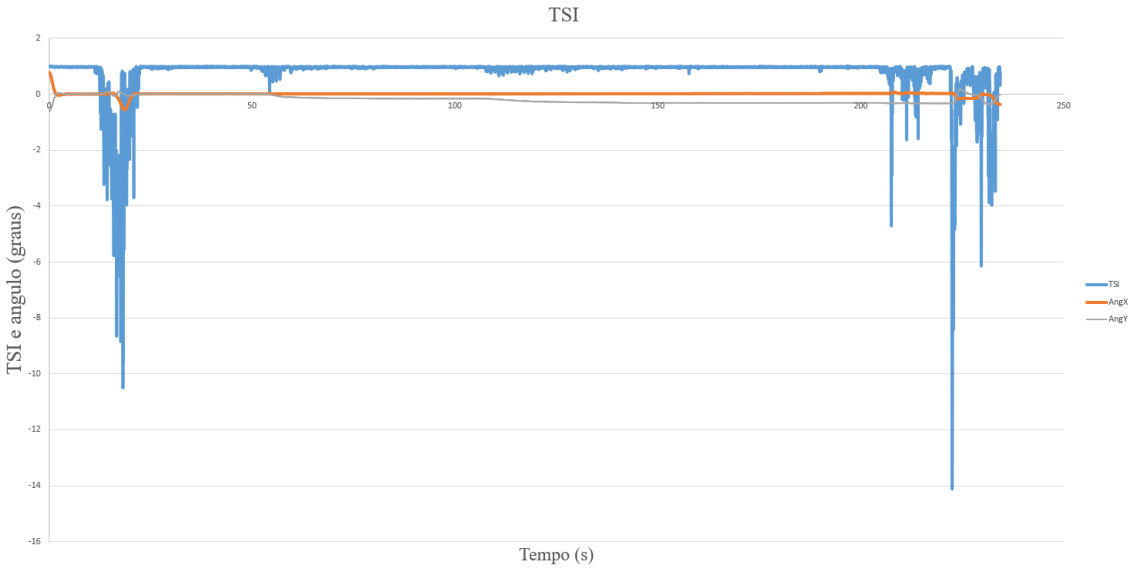

Figura 17 - Segundo ensaio. TSI em azul

No terceiro ensaio, figura 18, observa-se que durante o deslocamento o TSI se manteve perto do valor 1, de estabilidade total, com flutuações devido não somente ao ruido, mas também à rugosidade da superfície e dura (que causa pequenos pulos no modela de trator) do plano inclinado combinado com as rodas do tipo fora de estrada. Os únicos eventos notáveis no TSI representam a manipulação do trator para se realizar outro percurso.

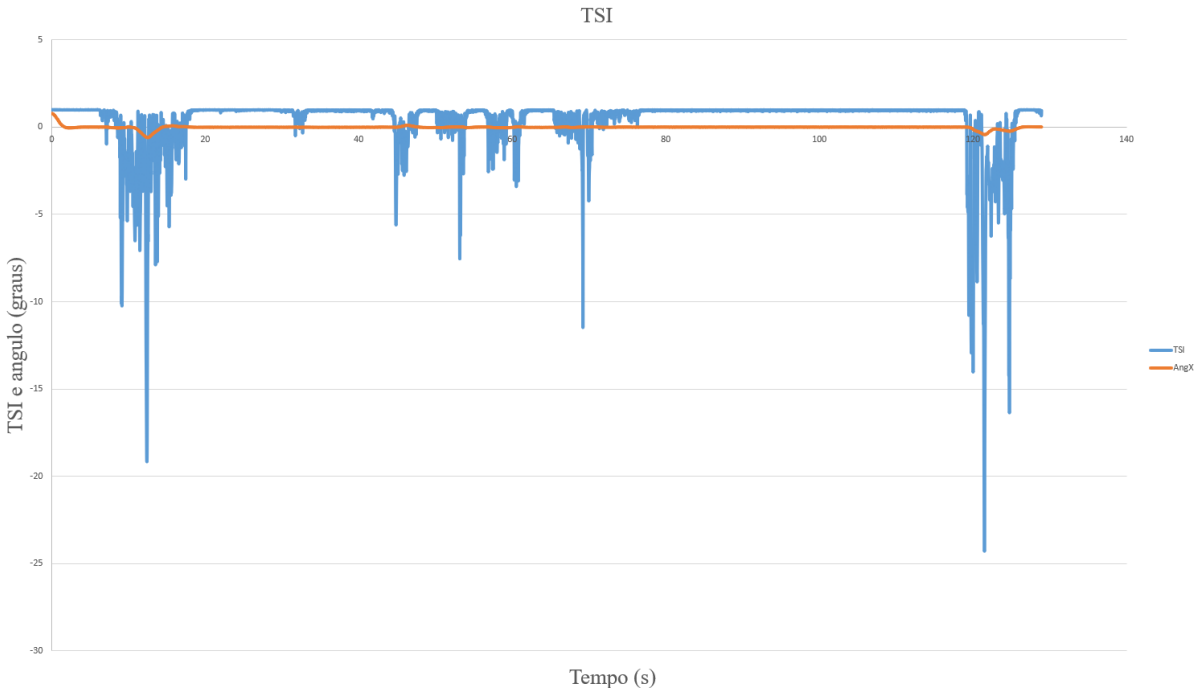

Figura 18 - Terceiro ensaio, dinâmico sem atingir o obstáculo

No quarto ensaio, com o trator estando significativamente inclinado se percebe uma redução do valor de "base" do TSI, ao atingir o obstáculo o trator sofre um evento de instabilidade, mas não chega a tombar, isso representa um caso em que o TSI deve ser reduzido, mas nunca chegar a valores nulos ou negativos. Essa sequência foi repetida diversas vezes, sempre repondo o manualmente o trator até o início do percurso.

Nos dados é evidente que o TSI atinge valores iguais e inferiores a zero, o que significa que o algoritmo produziu um "falso alarme", acusou o tombamento, mas esse não ocorreu. Apesar dessa sensibilidade em demasia do algoritmo esse foi capaz de identificar com exatidão o início dos eventos de instabilidade, como pode ser visto aos 55 segundos na figura 19.

Os resultados obtidos nessa série de ensaios demonstram que o algoritmo de estabilidade tem a capacidade de indicar o início do tombamento, o TSI de um valor positivo para zero (início do tombamento) e por fim um valor negativo (durante o tombamento), tanto a inclinação do plano como a velocidade do trator e a presença ou não de obstáculo tem influência no TSI como seria esperado

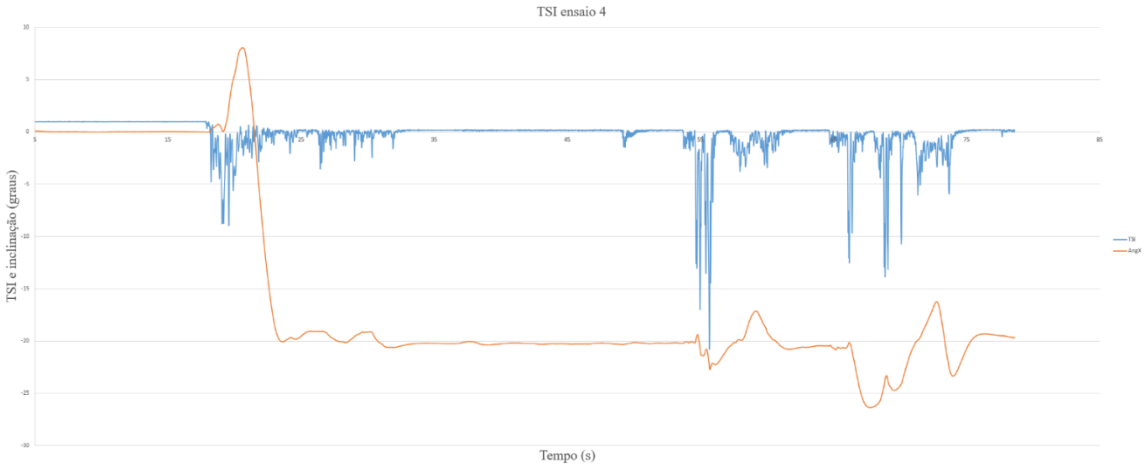

Figura 19 - Quarto ensaio, dinâmico com colisão com o obstáculo

No quinto ensaio, o trator não sofreu tombamento, mesmo estando inclinado, porém o TSI acusou tombamento como no ensaio anterior. E como visto anteriormente apesar de "falso alarme" o TSI teve picos negativos em simultâneo com o início do evento de instabilidade dinâmica, como pode ser visto a aproximadamente 30, 37 e 50 segundos na figura 20.

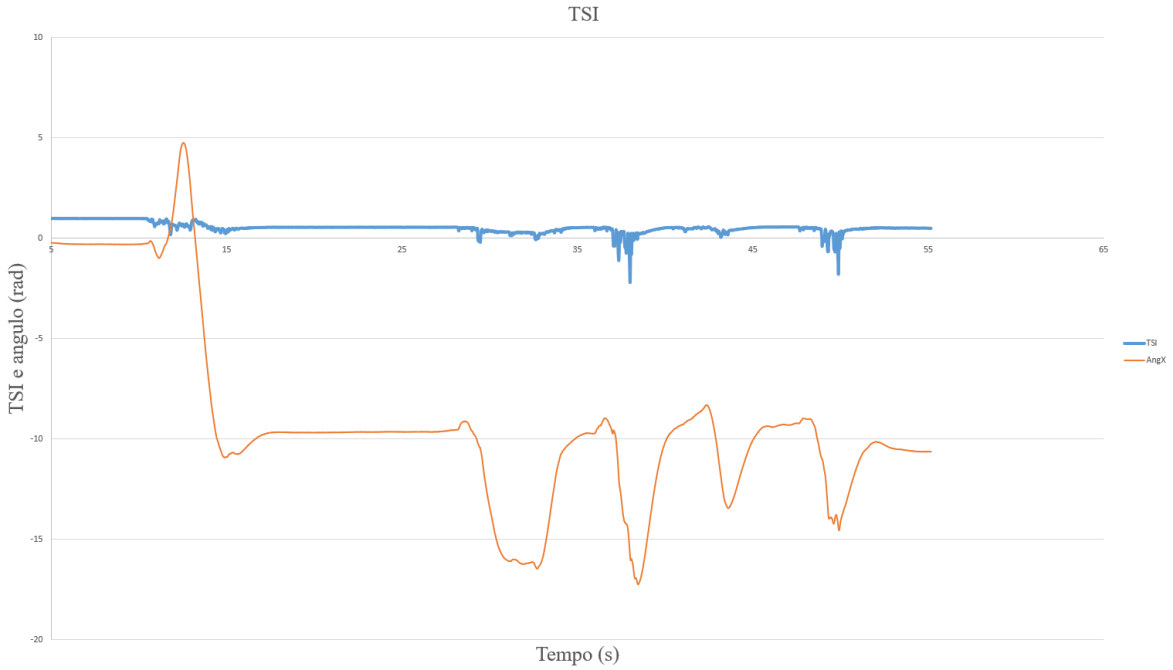

Figura 20 - Quinto ensaio, dinâmico, com colisão com o obstáculo

O sexto ensaio teve resultados similares ao quarto, devido a inclinação ainda maior o TSI acusa o tombamento quando o trator está deslocando-se pelo plano inclinado, mesmo estando estável, e como nos ensaios anteriores existem fortes picos negativos do TSI assim quando o trator atinge o obstáculo. Mas ao contrário dos ensaios anteriores houve o tombamento, o que gerou grandes picos negativos no TSI durante o tombamento que acabam dando lugar a valores positivos maiores que 1 devido a elevada inclinação, bastante evidente nos dois últimos tombamentos na figura 21. Também devido a elevada inclinação o controlo do trator se mostrou difícil, requerendo constantes ajustes na sua trajetória para atingir o obstáculo, isso se deve ao fato de possuir um diferencial aberto o que faz o trator naturalmente "puxar" para a direção do declive, esse fenómeno pode ser observado no gráfico quando existem breves picos de redução da inclinação no eixo X e consequente aumento momentâneo do TSI.

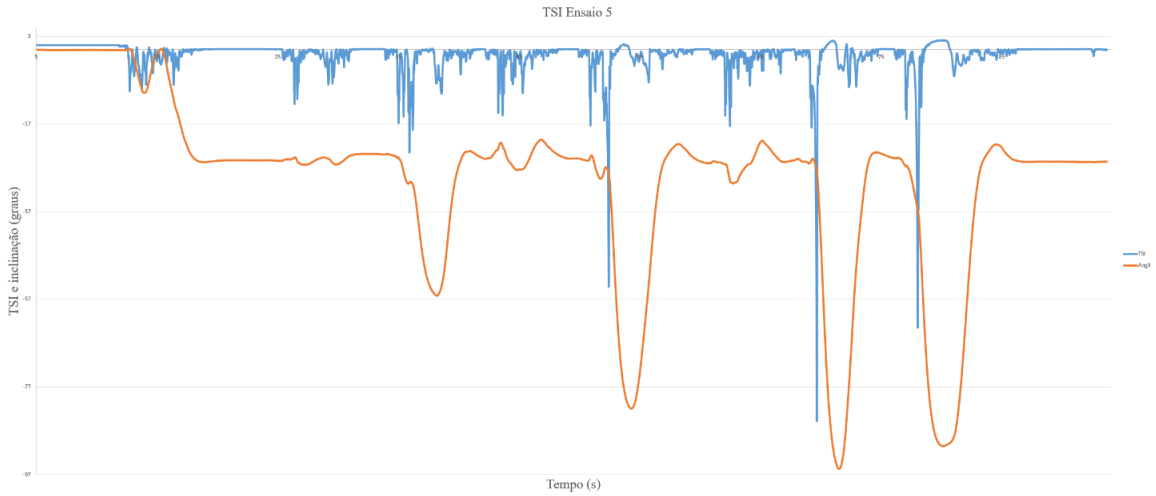

Figura 21 - Sexto ensaio, dinâmico

No sétimo ensaio tevês o mesmo comportamento do quinto ensaio, porém com o trator tombando. Neste caso existiram falsos alarmes quando o trator estava estável, mas os maiores picos negativos do TSI ocorreram justamente no início do tombamento.

Porém existiram diversas situações em que o algoritmo de estabilidade forneceu “falsos alarmes”, isto é, casos em que o trator estava dinamicamente estável, mas o TSI indicou o tombamento, evidente aos 55 segundos na figura 22 por exemplo. Esse fenômeno não é universal a todos os ensaios, estando presente nos dinâmicos onde existe uma inclinação significativa do plano em que o trator estava. Uma possibilidade para explicar esse comportamento é a presença de ruído nos dados que foram alimentados no algoritmo de estabilidade.

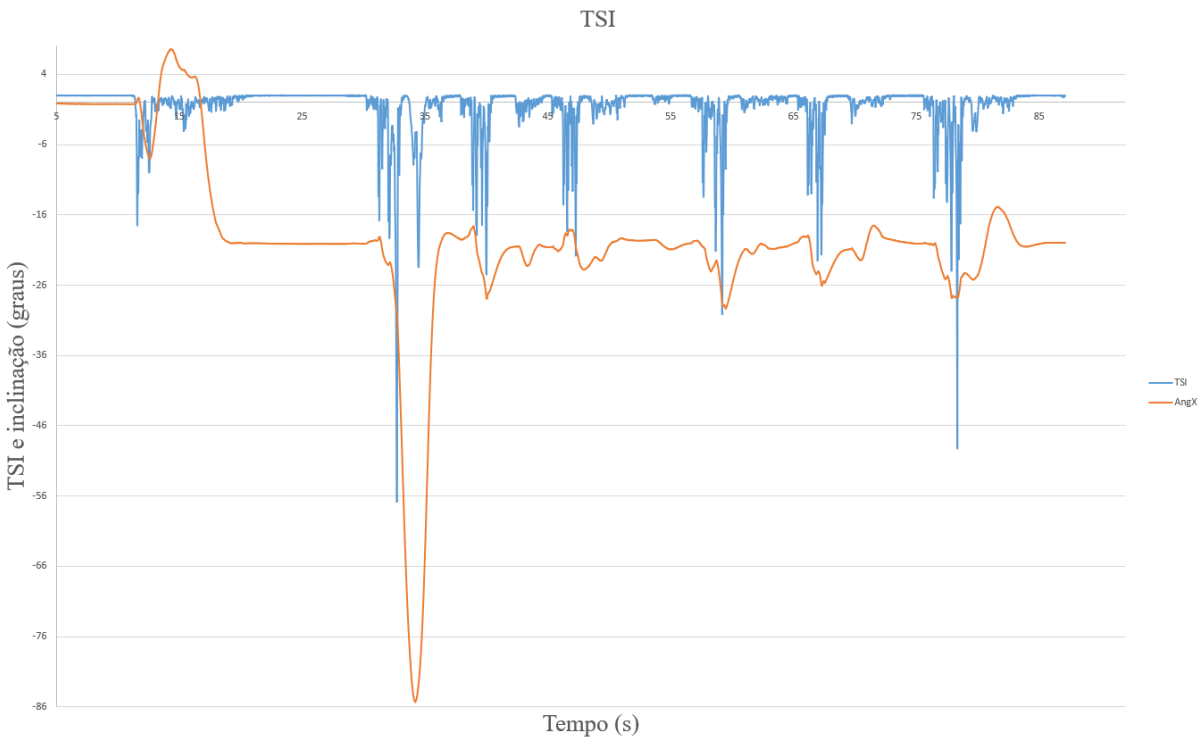

Figura 22 - Sétimo ensaio, dinâmico

Em ensaios anteriores o ruido foi observado, mas para a simples análise de inclinações e comportamento geral do veículo ele não teve impacto apreciável, porém devido a sensibilidade do algoritmo os picos de aceleração, breves, porém elevados, causam esses "falsos alarmes". Esse fenómeno torna-se evidente nos ensaios em que o trator se encontra significativamente inclinado, e consequentemente o TSI está mais próximo de zero. Em contrapartida quando a inclinação é nula, o ruído não causa falsos alarmes durante o deslocamento do trator, somente quando atinge o obstáculo. O obstáculo representa um caso extremo, tendo a mesma altura do vão livre do trator.

Também pode ser observado que no caso do ensaio 7, o TSI apresentou um valor superior a 1, o que à primeira vista pode ser visto como um "falso positivo", mas este ocorre durante o tombamento, quanto o trator está inclinado de tal forma que a altura final do CG acaba por ser inferior a altura estática do CG, o que acarreta um número positivo do TSI. É um fenómeno que não causa problemas, mas deve ser compreendido e considerado na programação de um sistema de alerta.

5.2 Ensaios em Campo

Os materiais utilizados para estes ensaios foram o sistema de aquisição de dados constituído pelo RBP4, o IMU, powerbank e o painel de controlo com LEDs e botões. O sistema de aquisição de dados foi instalado sobre o capô de um trator da Escola Superior Agraria de Viseu, numa posição próxima do centro de massa do trator e que não interferisse com o operador do veículo, como pode ser observado na figura 25.

Figura 23 - Sistema de aquisição de dados instalado no trator

Foi utilizada a primeira versão do programa de aquisição de dados, sem GPS e sem base anti vibrações.

Com o sistema de aquisição de dados instalado no trator, foram realizados cinco ensaios em diferentes condições similares a operações rotineiras do trator

O primeiro ensaio constitui no deslocamento do trator, inicialmente dentro da garagem até uma estrada de terra e cascalho, percurso em preto na figura 23. O trator percorreu uma linha reta, inicialmente em um plano nivelado e liso, após sair da garagem percorreu um pequeno declive e parou na estrada aproximadamente nivelada. O teste ocorreu a baixa velocidade e o terreno em geral não era muito acidentado. Inicialmente o sistema de aquisição

de dados foi ligado por alguns segundos antes do deslocamento e desligado alguns segundos após a paragem do trator. Sempre com o motor ligado.

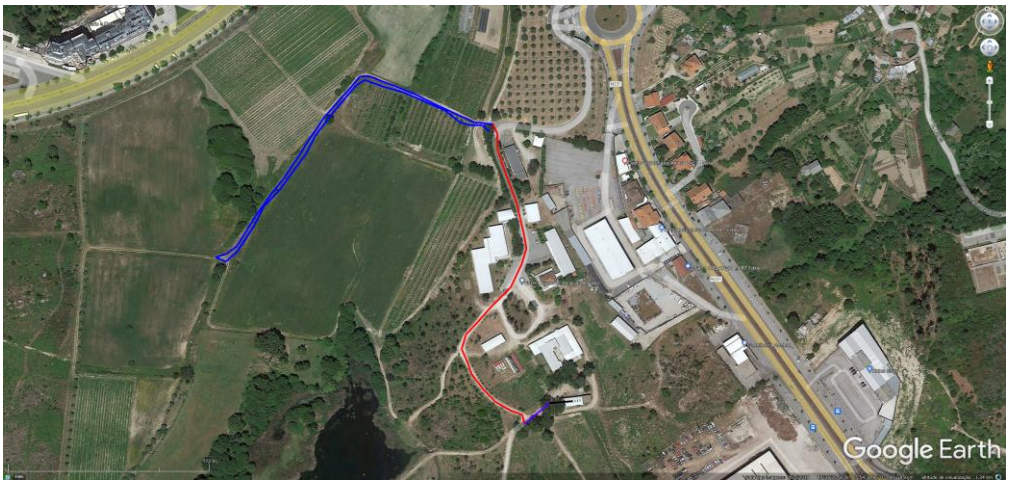

Figura 24 - Trajeto dos quatro primeiros ensaios

O segundo ensaio consistiu no deslocamento do trator numa estrada plana de terra e cascalho, em linha reta até o início de uma subida, A velocidade foi moderada e o terreno ligeiramente acidentado. O percurso desse teste está representado a roxo nas figuras 23 e 24.

Figura 25 - Trajeto do quinto teste

O terceiro ensaio consistiu no deslocamento do trator em uma estrada de terra e cascalho, a partir do ponto de parada do segundo ensaio até o estacionamento das Agrárias, em vermelho nas figuras 23 e 24. O trajeto era moderadamente acidentado e possui na sua primeira parte uma subida longa com algumas curvas. Na segunda parte o trajeto foi inicialmente composto por um asfalto liso, nivelado e em linha reta. As duas secções foram realizadas a uma velocidade moderada. Novamente o sistema de aquisição de dados foi ligado antes do início do deslocamento e desligado um tempo após a paragem. Sempre com o motor em funcionamento.

O quarto ensaio foi o teste mais longo dos realizados nesse dia. Consistiu no percurso de ida e volta de um trajeto muito acidentado, com curvas, desníveis e obstáculos significativos. O trajeto realizado foi em uma estrada de terra e cascalho muito acidentada. A velocidade de deslocamento foi mais elevada.

O quinto ensaio consistiu no deslocamento do trator com as rodas direitas sobre a calçada e as rodas do lado esquerdo na estrada. Dessa forma o trator se deslocou em linha reta, porém com uma inclinação lateral. A estrada era de asfalto e a calçada de pedra, sendo pouco acidentadas. Deslocamento ocorreu a uma baixa velocidade e sem a utilização de alfaia agrícola.

5.2.1 Resultados

Os resultados obtidos foram em grande parte satisfatórios, pois os dados adquiridos pelo IMU representaram o comportamento do trator durante os cinco ensaios. Mostrando na média as inclinações e movimentos do veículo.

O programa também se mostrou estável e a operação do botão e LEDs indicadores foi satisfatória e de fácil uso. O powerbank apresentou autonomia mais do que o suficiente para 3 horas de testes.

Porém foi observado um excesso de ruido em todos os testes e alguns picos de aceleração muito rápidos e violentos, que não condiziam com a situação real, e por consequência a inclinação (que é obtida com base nas acelerações.)

Como consequência desses resultados foi decido melhorar o programa de aquisição de dados e implementar a base anti vibrações para reduzir o ruido dos dados adquiridos.

5.3 Ensaios com Carro

Foram também realizados ensaios com o sistema de aquisição de dados num carro, com o fim de afinamento do código do programa, do filtro e do GPS.

Para esses ensaios o material utilizado foi o sistema de aquisição de dados que consiste do RBP4, do sensor IMU, instalado em uma base anti vibrações, um powerbank e a antena GPS. O sistema foi instalado no interior do veículo, um carro de passageiros Smart Four Two, sobre o painel central por questões de ergonomia e de não necessidade de modificar o veículo.

Os ensaios realizados nessa configuração foram três: Um estático de longa duração para verificar a consistência e estabilidade do sistema; um em circuito para simular um trajeto com

um trator e a consistência do GPS em relação aos dados do IMU; por último um teste estático, para obter um perfil do ruído de fundo.

O primeiro ensaio estático de longa duração consistiu na aquisição de dados do IMU e do GPS por quatro horas numa posição fixa, sem movimentos.

O segundo ensaio consistiu em percorrer um circuito de 6 voltas ao redor do Hotel Monte Belo em Viseu, como mostra o mapa da figura 17, tendo sido realizado em vias públicas dentro dos limites de velocidade e normas de trânsito. Tentando manter a velocidade, acelerações e condução relativamente constante.

O terceiro ensaio foi realizado no interior do veículo, mas com o mesmo parado inicialmente com o motor e sistema de ventilação e áudio desligados e posteriormente com os mesmos ligado.

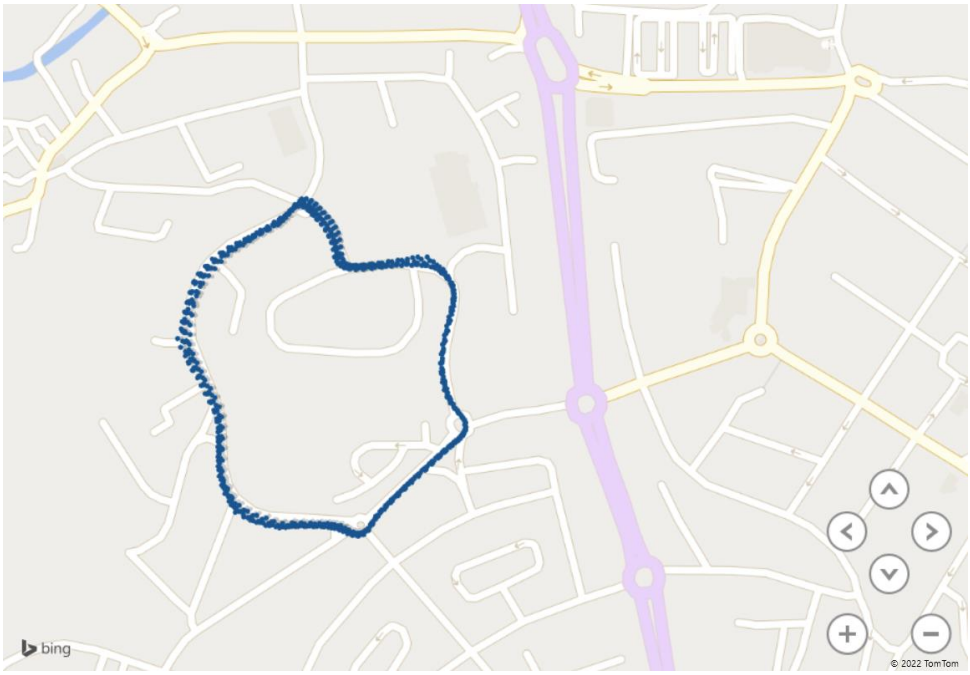

Figura 26 - Mapa feito com os dados do GPS

5.3.1 Resultados

O primeiro teste apresentou os resultados esperados de desvio mínimos na leitura do GPS, dentro da margem de erro do dispositivo. A taxa de amostragem do novo código mostrou-se boa, muito mais consistente que a do sistema inicial de monitorização. Porém os dados do IMU apresentaram deriva aparente após um certo tempo, figura 27. Mas essa deriva pode ser explicada pelo fato que o sistema esteve exposto a luz solar direta durante o verão, o que acarretou em uma elevada variação de temperatura e consequente flexão da base anti vibração e interface entre a mesma e o sensor IMU. Essa hipótese é provável devido ao fato da deriva ter sido consistente com o período em que o sistema esteve exposto ao sol e após esse período os dados voltaram a valores semelhantes aos iniciais.

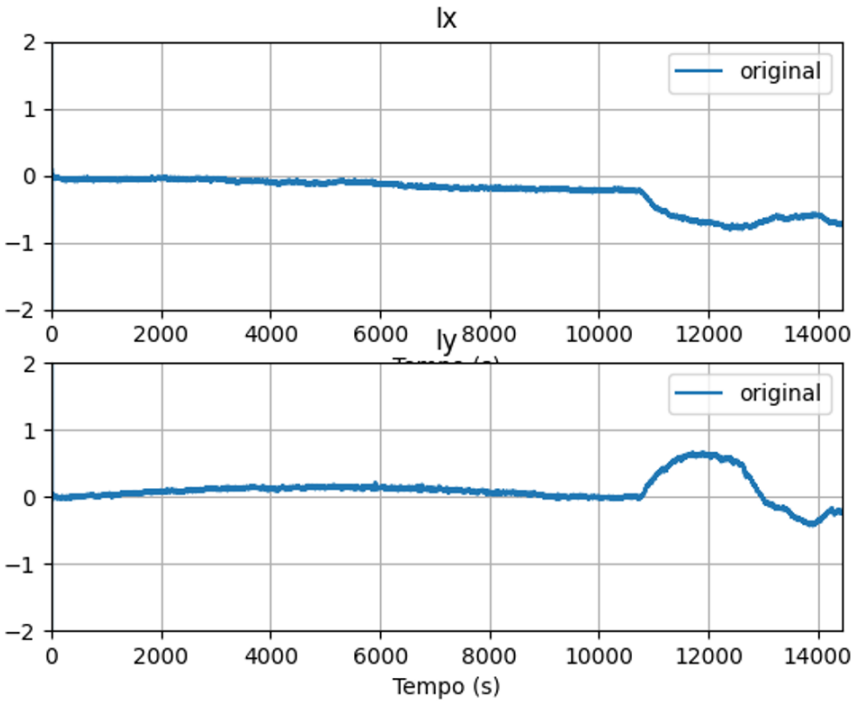

Figura 27 - Inclinações do ensaio estático de longa duração

O segundo ensaio teve bastante êxito, tanto o GPS como o IMU adquiriam dados que são consistentes com o circuito percorrido (figura 26) e o comportamento do carro durante o ensaio (figura 28). Para além disse observou-se um valor elevado de ruído de fundo e para além de picos esporádicos de aceleração devido a movimentos do painel do carro, o que dificultou a análise dos dados.

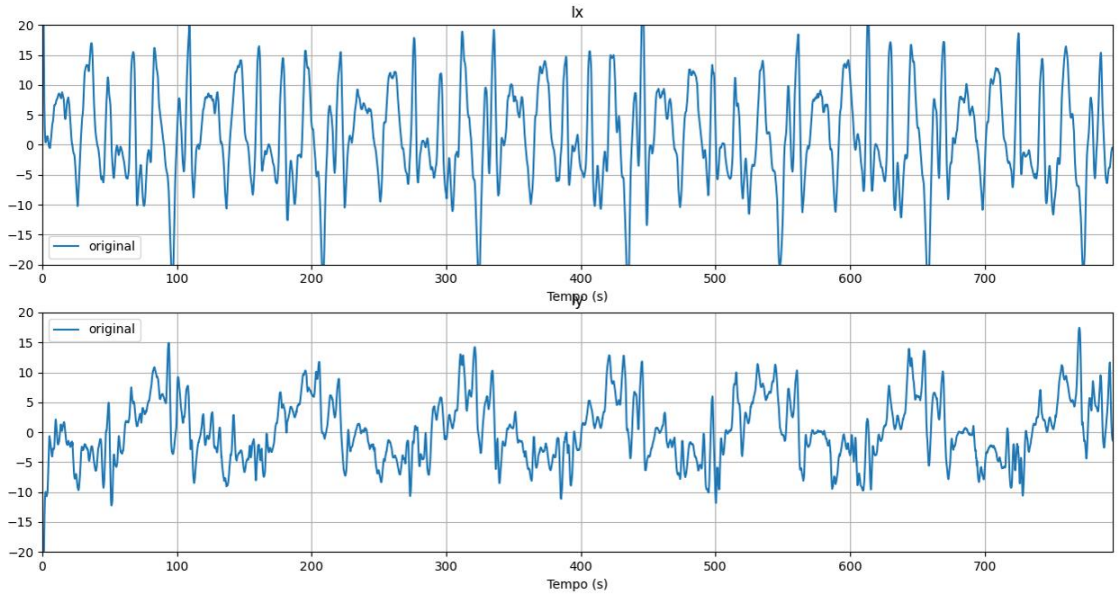

Figura 28 - Inclinações do teste de circuito

O terceiro teste confirmou a presença do ruido de fundo proveniente do carro e do seu motor, não se observou os fortes picos do segundo teste. Nesse sentido pode-se concluir que filtro também pode ser melhor ajustado para ser menos permissivo, e que o ruído causa leituras falsa de inclinação como na figura 29.

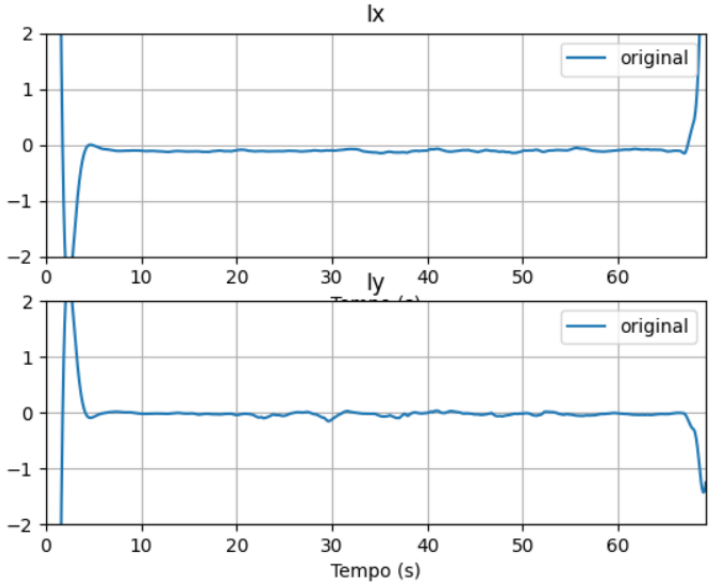

Figura 29 - Inclinações do

6 Conclusão e Discussão dos Resultados

A principal conclusão que pode ser obtida a partir dos resultados dos ensaios é a de que o sensor IMU consegue, através das variáveis adquiridas, descrever o comportamento do

modelo de trator em escala reduzida, ficando evidente quando as rodas deste atingem o obstáculo. O transdutor de deslocamento também consegue descrever a velocidade de deslocamento linear do modelo de forma satisfatória.

Também foi possível verificar através dos ensaios com o modelo em escala que as rodas traseiras são as mais críticas em termos de estabilidade, sendo que por várias vezes o trator não tombou quando uma das rodas dianteiras atingiu o obstáculo, mas somente quanto a roda traseira o fez.

Quanto ao TSI e ao algoritmo de estabilidade, este apresentou resultados positivos, mas com algumas ressalvas na sua aplicação. Primeiro o algoritmo consegue de fato prever o início de tombamento numa situação dinâmica, porém a janela de tempo entre essa previsão e o tombamento em si é demasiada curta para uma intervenção humana direta, então o foco para a sua aplicação deve ser dado a situações em que o operador possa ter algum controle, já que o valor do TSI mesmo que na zona estável pode ser dividido em zonas de grau de risco. Em paralelo a deteção imediata do evento de tombamento também é útil no caso do envio de mensagem de socorro ou para a utilização em algum outro sistema de segurança do trator (como um ROPS ou airbag).

Em segundo lugar o algoritmo de estabilidade se mostrou demasiadamente suscetível ao ruido nos dados adquiridos pelo IMU, o que causa falsos alarmes. Mas como esse comportamento é sempre de natureza a apresentar falsos sinais de perigo e não falsos sinais de segurança, pode se deduzir que o algoritmo em si é viável, mesmo que seja sensível e demasiadamente “cauteloso”. Uma redução dessa sensibilidade tornaria o algoritmo útil para ser utilizado na prática.

6.1 Trabalhos Futuros

A principal melhoria que pode ser feita é um melhor controle dos ruídos, tanto na sua origem por meio de uma base antivibrações, como pela sua eliminação por meio de um filtro propriamente calibrado.

A segunda grande melhoria que pode ser considerada seria a construção de um veículo para testes, com uma boa qualidade de construção (para a eliminação de folgas) e com uma estrutura modular que permita a variação das suas características automotivas (entre eixos, bitola, etc.) assim como um controlo mais fino do seu CG e locais adequados para a instalação dos equipamentos de medição.

O software também requer considerável trabalho para poder atender todos os requisitos do mundo real, quando este possui limitadas fontes de entrada de dados (utilizando

preferencialmente apenas um IMU), atualmente é apenas útil em situações com o terreno previamente conhecido. Sendo, portanto, necessário encontrar um método em que o algoritmo consiga distinguir da inclinação do terreno e do veículo por meio de algum artificio de programação ou por outros sensores.

Algumas outras melhorias que podem ser feitas são a instalação do powerbank diretamente no modelo de trator em escala reduzida. Isso tem o objetivo de melhorar a estabilidade do funcionamento do RP4, que sofreu vários resets devido a manipulação manual do cabo de alimentação. Outra sugestão de melhoria seria criar um sistema que guiasse o modelo de trator em escala reduzida durante o seu percurso no plano inclinado. Pois devido ao facto que o modelo de trator possui um diferencial aberto, este tem a tendência natural de alterar a sua trajetória quando está se deslocando em uma superfície inclinada, o que necessita de ajustes manuais constantes. Tal sistema de guia também iria facilitar a localização inicial do modelo, fundamental para o bom funcionamento do transdutor de deslocamento.

Outra melhoria que pode ser feita, podendo gerar um trabalho inteiro sobre o tema, é a eliminação dos ruídos pela via mecânica. Estudando qual a melhor forma de isolar o sensor de vibrações e ruídos de fundo sem que esse se torne demasiadamente insensível ou lento para se ter utilidade. Podendo estudar variáveis da base antivibrações como a sua dimensão, elementos de suspensão, frequência natural e etc. para vários cenários de utilização.

Por questões de tempo não foi possível desenvolver o sistema de alerta propriamente dito, sendo necessário um estudo posterior para caracterizar o grau de risco do TSI e classificá-lo em três níveis, seguro (luz verde), atenção (luz amarela) e perigo (luz vermelha), assim como o hardware e software necessários. Tendo sido feito a base para esse trabalho. O que se pode dizer desse futuro estudo é que deve ser feita uma relação entre um gradiente contínuo da probabilidade de tombamento (o alerta) e o TSI que pela sua natureza é um valor discreto que diz o quão próximo o veículo está de sofrer o tombamento, não a sua probabilidade.

Referências Bibliográficas

Abubakar et al, (2010). A Review of Farm Tractor Overturning Accidents and Safety. Pertanika J. Sci. & Technol. 18 (2), 377–385.

Ahmadi, I., (2013). Turkish Journal of Agriculture and Forestry. Development of a tractor dynamic stability index calculator utilizing some tractor specifications. Turk J Agric For, 37, 203-211. Retrieved from http://doi.org/10.3906/tar-1103-19.

Barros P.R., (2012). simulação da capacidade de tração de um trator 4x2 com tração dianteira auxiliar em diferentes condições de superfície. (Tese de Doutoramento não editada, Programa de Pós-Graduação em Engenharia Agrícola). Universidade Federal de Viçosa, Viçosa, Brasil.

Cavallo, E., Ferrari, E., Bollani, L., Cocci, M., (2014). Attitudes and behaviour of adopters of technological innovations in agricultural tractors: A case study in Italian agricultural system. Agricultural Systems. 130, 44-54. Retrieved from https://doi.org/10.1016/j.agsy.2014.05.012.

Denis, D., Thuilot, B., Lenain, R. (2016). Online adaptive observer for rollover avoidance of reconfigurable agricultural vehicles. Computers and Electronics in Agriculture. 126, 32-43. Retrieved from https://doi.org/10.1016/j.compag.2016.04.030.

Li, Z., Mitsuoka, M., Inoue, E., Okayasu, T., Hirai, Y., Zhu, Z., (2016). Study on Roll Instability Mechanism and Stability Index of Articulated Steering Vehicles. Hindawi Publishing Corporation, Mathematical Problems in Engineering. Retrieved from http://dx.doi.org/10.1155/2016/7816503.

Liu & Koc, (2015). Field Tests of a Tractor Rollover Detection and Emergency Notification System. J Agric Saf Health. 21(2), 113-127.DOI: 10:13021

Lower, T., Monaghan, N., Rolfe, M., (2016). Quads, Farmers 50+ Years of Age, and Safety in Australia. Safety. 2(2), 12. Retrieved from https://doi.org/10.3390/safety2020012.

Neto C.R., (2012). Análise Experimental da Estabilidade Direcional de Veículos Agrícolas de Rodas em Terrenos Declivosos. (Tese de Doutoramento não editada, Programa de Pós-Graduação em Engenharia Agrícola). Universidade Estadual de Campinas, Faculdade de Engenharia Agrícola, Campinas, Brasil.

Peters, S. C. and Lagnemma, K., (2009). Stability measurement of high-speed vehicles. Vehicle System Dynamics. 47(6),701-720. Retrieved from http://doi.org/10.1080/00423110802344636.

P.E. Uys, P. S. Els, M. J. Thoresson, K. G. Voigt, W. C. Combrinck, (2006). Experimental Determination of Moments of Inertia for an Off-Road Vehicle in a Regular Engineering Laboratory. *International Journal of Mechanical Engineering Education*. 2006;34(4):291-314. doi:10.7227/IJMEE.34.4.2

Silva, H., Trindade, A., Gaspar, D., Marques, F., (2017). 13º Congresso Ibero-americano de Engenharia Mecânica: Avaliação da Estabilidade para a Segurança de Tratores Agrícolas. Lisboa

Varella, C.A.A., (2011). Introdução ao Estudo dos Tratores Agrícolas.

1 Apêndice

1.1 Código do Sistema de Aquisição e Monitorização de Dados – Cliente

```
#!/usr/bin/env python3
# !/usr/bin/env python3

import socket
from math import atan2, degrees, sqrt
import threading
from scipy import signal
import FIR as fir
from mpu9250_jmdev.registers import *
from mpu9250_jmdev.mpu_9250 import MPU9250
import csv
from datetime import datetime

HOST = '192.168.1.100'  # Endereço do servidor
PORT = 65432  # Porta do servidor

mpu = MPU9250(
    address_ak=AK8963_ADDRESS,
    address_mpu_master=MPU9050_ADDRESS_68,
    address_mpu_slave=None,
    bus=1,
    gfs=GFS_1000,
    afs=AFS_8G,
    mfs=AK8963_BIT_16,
    mode=AK8963_MODE_C100HZ)

mpu.reset()
mpu.calibrateMPU6500()  # Calibrate sensors
mpu.configure()

abias = mpu.abias  # Get the master accelerometer biases
abias_slave = mpu.abias_slave  # Get the slave accelerometer biases
gbias = mpu.gbias  # Get the master gyroscope biases
gbias_slave = mpu.gbias_slave  # Get the slave gyroscope biases

Nfir = 100
h = signal.firwin(Nfir, 0.001)

fir_x = fir.FIR()  # filtro FIR para aplicações de tempo real - Componente ax
fir_y = fir.FIR()  # filtro FIR para aplicações de tempo real - Componente ay
fir_z = fir.FIR()  # filtro FIR para aplicações de tempo real - Componente az
fir_wx = fir.FIR()  # filtro FIR para aplicações de tempo real - Componente wx
fir_wy = fir.FIR()  # filtro FIR para aplicações de tempo real - Componente wy
fir_wz = fir.FIR()  # filtro FIR para aplicações de tempo real - Componente wz

fir_x.setTaps(Nfir)  # dimensão do filtro
fir_y.setTaps(Nfir)
fir_z.setTaps(Nfir)
fir_wx.setTaps(Nfir)  # dimensão do filtro
fir_wy.setTaps(Nfir)
fir_wz.setTaps(Nfir)
```

```
for i in range(Nfir - 1):
    fir_x.setCoeff(h[i], i)
    fir_y.setCoeff(h[i], i)
    fir_z.setCoeff(h[i], i)
    fir_wx.setCoeff(h[i], i)
    fir_wy.setCoeff(h[i], i)
    fir_wz.setCoeff(h[i], i)

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  # configura a socket

s.connect((HOST, PORT))  # liga ao servidor

Ix = 0
Iy = 0
ay = 0
ax = 0
az = 0
wx = 0
wy = 0
wz = 0

current_datetime = datetime.now()
str_current_datetime = str(current_datetime)
file_name = str_current_datetime + ".csv"

escrita_ativa = False


def ler_sensor():
    global Ix
    global Iy
    global ax
    global ay
    global az
    global wx
    global wy
    global wz
    global fir_x
    global fir_y
    global fir_z
    global fir_wx
    global fir_wy
    global fir_wz
    global escrita
    global escrita_ativa

    threading.Timer(0.005, ler_sensor).start()  # inicia o contador da função de leitura do sensor

    A = mpu.readAccelerometerMaster()  # leitura dos dados do acelerometro
    ax = fir_x.filter(A[0])
    ay = fir_y.filter(A[1])
    az = fir_z.filter(A[2])
    Ix = degrees(atan2(ay, az))  # inclinação em relação ao eixo x "tombamento"
    Iy = degrees(atan2(-ax, sqrt(ay * ay + az * az)))  # inclinação em relação ao eixo y "empinamento"
```

```
    B = mpu.readGyroscopeMaster()  # leitura dos dados do giroscopio
    wx = fir_wx.filter(B[0])
    wy = fir_wy.filter(B[1])
    wz = fir_wz.filter(B[2])

    if escrita_ativa:  # grava em csv os dados relevantes do IMU
        escrita.writerow({'Data': datetime.now().strftime('%Y-%m-%d'), 'Timestamp':
datetime.now().strftime('%H:%M:%S'
                                                                '.%f'),
                          'Ix (Graus)': Ix, 'Iy (Graus)': Iy, 'ax (m/s^2)': ax * 9.80665, 'a0x (m/s^2)': A[0] * 9.80665,
                          'ay (m/s^2)': ay * 9.80665, 'a0y (m/s^2)': A[1] * 9.80665, 'az (m/s^2)': az * 9.80665,
                          'a0z (m/s^2)': A[2] * 9.80665, 'wx (rad/s)': wx * 0.0174533, 'w0x (rad/s)': B[0] * 0.0174533,
                          'wy (rad/s)': wy * 0.0174533, 'w0y (rad/s)': B[1] * 0.0174533, 'wz (rad/s)': wz * 0.0174533,
                          'w0z (rad/s)': B[2] * 0.0174533})

    return Ix, Iy


ler_sensor()  # chama a função ler sensor, reiniciando o loop


# rotina para o envio de dados
def envia_valores():
    texto = "Ix = {0:.2f};Iy = {1:.2f};".format(Ix, Iy)

    threading.Timer(0.1, envia_valores).start()  # inicia o contador da função de envio de dados

    print("Dados a enviar: ", texto)

    s.send(bytes(texto, 'utf8'))


envia_valores()  # chama a função envia valores, reiniciando o loop


def recebe_ordens():
    global escrita
    global ficheiro
    global escrita_ativa

    data = s.recv(1024)  # recebe comandos do servidor
    chave = data.decode('utf8')  # variavel para inicio e fim de escrita dos dados coletados pelo imu em ficheiro
csv

    threading.Timer(0.1, recebe_ordens).start()  # ativa o envio de novos dados para daqui a 0.1s

    if chave == "gravar":
        print("\n!!!!!!!!! Ordem para início de registo recebida !!!!!!!!!\n")
        if not escrita_ativa:  # se a escrita já estava ativa ignora
            print("Início de registo")
            escrita_ativa = True
            current_datetime = datetime.now()
            str_current_datetime = current_datetime.strftime("%d-%m-%Y_%H-%M-%S")
            file_name = "LOG_" + str_current_datetime + ".csv"
            ficheiro = open(file_name, "w", newline="")
            campos = ['Data', 'Timestamp', 'Ix (Graus)', 'Iy (Graus)', 'ax (m/s^2)', 'a0x (m/s^2)', 'ay (m/s^2)',
```

```
                'a0y (m/s^2)',
                'az (m/s^2)', 'a0z (m/s^2)', 'wx (rad/s)', 'w0x (rad/s)', 'wy (rad/s)', 'w0y (rad/s)',
                'wz (rad/s)', 'w0z (rad/s)']
            escrita = csv.DictWriter(ficheiro, fieldnames=campos)
            escrita.writeheader()

    if chave == "parar":
        print("\n!!!!!!!!! Ordem para terminar de registo recebida !!!!!!!!!\n")
        if escrita_ativa:  # se a escrita já estava inativa ignora
            escrita_ativa = False
            ficheiro.close()  # é preciso fechar o ficheiro quando termina a escrita
            print("Fim de registo")


# chama a rotina de recebe ordens (deve ser a última a ser chamada para não bloquear o programa)


recebe_ordens()
```

1.2 Código do Sistema de Monitorização e Aquisição de Dados - Servidor

```
#!/usr/bin/env python3
import socket
from math import radians, sin, cos
import matplotlib.animation
import numpy as np
from matplotlib import pyplot as plt
from matplotlib.patches import Polygon, Circle
import keyboard

y = 3.5

######### Polígonos para a animação ##############################
C_corpo = np.array(
    [[-3, y - 2], [4.5, y - 2], [4.5, y + 1], [0.5, y + 2], [0.5, y], [-2.5, y], [-2.5, y + 6], [-3, y + 6]])
C_chao = np.array([[-15, y - 3.5], [15, y - 3.5], [15, y - 12], [-10, y - 12]])
C_roda_f = np.array([3.5, y - 2])
C_roda_t = np.array([-2.5, y - 1])
C_rt_L = np.array([[-2, y - 3.5], [-2, y + 1.5], [-4, y + 1.5], [-4, y - 3.5]])
C_rt_R = np.array([[2, y - 3.5], [2, y + 1.5], [4, y + 1.5], [4, y - 3.5]])
C_ct = np.array(
    [[-2, y - 2], [2, y - 2], [2, y + 1.5], [1, y + 1.5], [1, y + 3], [-1, y + 3], [-1, y + 1.5], [-2, y + 1.5]])
C_arco = np.array(
    [[-2, y + 1], [-2, y + 6], [2, y + 6], [2, y + 1], [1.5, y + 1], [1.5, y + 5.5], [-1.5, y + 5.5], [-1.5, y + 1]])
corpo = Polygon(C_corpo, facecolor='r')
chao = Polygon(C_chao, facecolor='y')
roda_t1 = Circle(C_roda_t, radius=2.5, facecolor='k')
roda_t2 = Circle(C_roda_t, radius=1.5, facecolor='w')
roda_f1 = Circle(C_roda_f, radius=1.5, facecolor='k')
roda_f2 = Circle(C_roda_f, radius=0.8, facecolor='w')
Rt2L = Polygon(C_rt_L, facecolor='k')
Rt2R = Polygon(C_rt_R, facecolor='k')
Ct = Polygon(C_ct, facecolor='r')
arco = Polygon(C_arco, facecolor='r')
chaoT = Polygon(C_chao, facecolor='y')
fig, axs = plt.subplots(2, 2)

axs[0, 0].add_patch(corpo)
axs[0, 0].add_patch(chao)
axs[0, 0].add_patch(roda_t1)
axs[0, 0].add_patch(roda_t2)
axs[0, 0].add_patch(roda_f1)
axs[0, 0].add_patch(roda_f2)
axs[0, 0].axis('square')
axs[0, 0].axis('off')
axs[0, 0].axis([-7, 7, y - 6, y + 6])

axs[0, 1].add_patch(Rt2L)
axs[0, 1].add_patch(Rt2R)
axs[0, 1].add_patch(Ct)
axs[0, 1].add_patch(chaoT)
axs[0, 1].add_patch(arco)
axs[0, 1].axis('square')
axs[0, 1].axis('off')
```

```
axs[0, 1].axis([-7, 7, y - 6, y + 6])
##############################################################

#### Gráficos Animados ####
t = range(-100, 1)
xlist = list(np.zeros(101))
linha_x, = axs[1, 0].plot(t, xlist)
axs[1, 0].axis([-100, 0, -60, 60])
axs[1, 0].grid(b=None, which='major', axis='y')
ylist = list(np.zeros(101))
linha_y, = axs[1, 1].plot(t, ylist)
axs[1, 1].axis([-100, 0, -60, 60])
axs[1, 1].grid(b=None, which='major', axis='y')

HOST = ''  # Aceitar ligação de qualquer local
PORT = 65432  # Porta de ligação

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

s.bind((HOST, PORT))

s.listen()

conn, addr = s.accept()

print('Connected by', addr)

Ix = 0
Iy = 0


def onkeypress(event):
    if event.name == "f2":
        cmd = "gravar"
        conn.send(str.encode(cmd))
        resposta_cliente = str(conn.recv(1024), "utf-8")
        print("Gravando dados")
    if event.name == "f7":
        cmd = "criar"
        conn.send(str.encode(cmd))
        print("CSV criado")
    if event.name == "f4":
        cmd = "parar"
        conn.send(str.encode(cmd))
        print("Parou de gravar")


keyboard.on_press(onkeypress)


def update(i):
    global xlist
    global ylist
    global Ix
    global Iy
```

```
data = conn.recv(1024)

texto = data.decode('utf8')

texto_separado = texto.split(";")

# print("Dados Recebidos:", len(texto_separado), " => ", texto)
# preciso remover esse print para poder fazer o input dos comandos

if len(texto_separado) > 2:  # recebido pelo menos um Ix e um Iy
    for n in range(int(len(texto_separado) / 2)):
        exec(texto_separado[0], globals())  # executa o primeiro Ix ou Iy
        exec(texto_separado[1], globals())  # executa o segundo Ix ou Iy

        # atualiza a informação dos gráficos
        xlist.append(Ix)  # acrescenta a nova inclinação em x
        xlist = xlist[-101:]  # remove o primeiro elemento
        ylist.append(Iy)  # acrescenta a nova inclinação em y
        ylist = ylist[-101:]  # remove o primeiro elemento

linha_x.set_data(t, xlist)
linha_y.set_data(t, ylist)

# coloca o ângulo atual no título do gráfico
axs[1, 0].set_title("{0:.1f}º".format(Ix))
axs[1, 1].set_title("{0:.1f}º".format(Iy))
# animação do trator
R = np.array([[cos(radians(Iy)), sin(radians(Iy))],
              [-sin(radians(Iy)), cos(radians(Iy))]])  # matriz de rotação eixo x

# aplica a rotação aos polígonos (eixo x)
R_chao = C_chao.dot(R)
R_corpo = C_corpo.dot(R)
R_roda_f = C_roda_f.dot(R)
R_roda_t = C_roda_t.dot(R)

roda_t1.set_center(R_roda_t)
roda_t2.set_center(R_roda_t)
roda_f1.set_center(R_roda_f)
roda_f2.set_center(R_roda_f)
corpo.set_xy(R_corpo)
chao.set_xy(R_chao)

R = np.array([[cos(radians(Ix)), sin(radians(Ix))],
              [-sin(radians(Ix)), cos(radians(Ix))]])  # matriz de rotação eixo y

# aplica a rotação aos polígonos (eixo y)
R_rt_L = C_rt_L.dot(R)
R_rt_R = C_rt_R.dot(R)
R_ct = C_ct.dot(R)
R_arco = C_arco.dot(R)
R_chao = C_chao.dot(R)

Rt2L.set_xy(R_rt_L)
Rt2R.set_xy(R_rt_R)
Ct.set_xy(R_ct)
```

```
    arco.set_xy(R_arco)
    chaoT.set_xy(R_chao)

    # enviar_comando()



# ativa a rotina de animação
ani = matplotlib.animation.FuncAnimation(fig, update)

# chama o gráfico
plt.show()
```

1.3 Código do Sistema de Aquisição de Dados para Ensaios em Campo – Primeira Versão

```
from math import atan2, degrees, sqrt
from scipy import signal
import FIR as fir
from mpu9250_jmdev.registers import *
from mpu9250_jmdev.mpu_9250 import MPU9250
import csv
from datetime import datetime
from gpiozero import LED, Button
import time
import serial
from multiprocessing import Process, Queue, Value

ser = serial.Serial('/dev/ttyACM0', 9600, timeout=1)  # abre a porta série

# processo para leitura do sensor
def ler_sensor(fila, estado):
    Nfir = 100
    h = signal.firwin(Nfir, 0.001)
    tempo_anterior = 0
    contador = 0

    mpu = MPU9250(
        address_ak=AK8963_ADDRESS,
        address_mpu_master=MPU9050_ADDRESS_68,
        address_mpu_slave=None,
        bus=1,
        gfs=GFS_1000,
        afs=AFS_8G,
        mfs=AK8963_BIT_16,
        mode=AK8963_MODE_C100HZ)

    mpu.reset()
```

```
mpu.calibrateMPU6500()  # Calibrate sensors
mpu.configure()

abias = mpu.abias  # Get the master accelerometer biases
abias_slave = mpu.abias_slave  # Get the slave accelerometer biases
gbias = mpu.gbias  # Get the master gyroscope biases
gbias_slave = mpu.gbias_slave  # Get the slave gyroscope biases

fir_x = fir.FIR()  # filtro FIR para aplicações de tempo real - Componente ax
fir_y = fir.FIR()  # filtro FIR para aplicações de tempo real - Componente ay
fir_z = fir.FIR()  # filtro FIR para aplicações de tempo real - Componente az
fir_wx = fir.FIR()  # filtro FIR para aplicações de tempo real - Componente wx
fir_wy = fir.FIR()  # filtro FIR para aplicações de tempo real - Componente wy
fir_wz = fir.FIR()  # filtro FIR para aplicações de tempo real - Componente wz

fir_x.setTaps(Nfir)  # dimensão do filtro
fir_y.setTaps(Nfir)
fir_z.setTaps(Nfir)
fir_wx.setTaps(Nfir)  # dimensão do filtro
fir_wy.setTaps(Nfir)
fir_wz.setTaps(Nfir)

for i in range(Nfir - 1):
    fir_x.setCoeff(h[i], i)
    fir_y.setCoeff(h[i], i)
    fir_z.setCoeff(h[i], i)
    fir_wx.setCoeff(h[i], i)
    fir_wy.setCoeff(h[i], i)
    fir_wz.setCoeff(h[i], i)

Ix = 0
Iy = 0
ay = 0
ax = 0
```

```
az = 0
wx = 0
wy = 0
wz = 0

while estado.value > 0:  # enquanto estiver a escrever dados

    tempo = datetime.now()
    dif_t = float(datetime.timestamp(tempo)) - tempo_anterior
    while dif_t < 0.01:  # fica à espera pelo tempo exato para iniciar uma nova aquisição
        tempo = datetime.now()
        dif_t = float(datetime.timestamp(tempo)) - tempo_anterior

    tempo_anterior = float(datetime.timestamp(tempo))

    data = datetime.now().strftime('%Y-%m-%d')
    timestamp = datetime.now().strftime('%H:%M:%S.%f')

    A = mpu.readAccelerometerMaster()  # leitura dos dados do acelerometro
    ax = fir_x.filter(A[0])
    ay = fir_y.filter(A[1])
    az = fir_z.filter(A[2])
    Ix = degrees(atan2(ay, az))  # inclinação em relação ao eixo x "tombamento"
    Iy = degrees(atan2(-ax, sqrt(ay * ay + az * az)))  # inclinação em relação ao eixo y "empinamento"

    B = mpu.readGyroscopeMaster()  # leitura dos dados do giroscopio
    wx = fir_wx.filter(B[0])
    wy = fir_wy.filter(B[1])
    wz = fir_wz.filter(B[2])

    contador = contador + 1

    if contador == 10:  # envia dados a cada 100 ms (10 ms * 10)
        contador = 0
```

```
            fila.put([data, timestamp, Ix, Iy, ax, ay, ax, wx, wy, wz])

        tempo = datetime.now()
        dormir = float(datetime.timestamp(tempo)) - tempo_anterior - 0.002
        time.sleep(dormir)  # dorme o tempo restante com uma folga de 2 ms

def ler_GPS():
    while True:
        while ser.read().decode("utf-8") != '$':  # espera pelo inicio da string
            pass  # Do nothing

        linha = ser.readline().decode("utf-8")  # Lê uma string completa

        valores = linha.split(",")  # separa os dados por linha

        print(valores[0])

        while valores[0] != "GPGGA":
            while ser.read().decode("utf-8") != '$':
                pass
            linha = ser.readline().decode("utf-8")
            valores = linha.split(",")
            print(valores[0])
        t = time.strptime(valores[1], "%H%M%S.%f")
        print("Hora do GPS (UTC): {0:02d}:{1:02d}:{2:02d}".format(t.tm_hour, t.tm_min, t.tm_sec))

        print("Número de Satélites:", valores[7])

        if float(valores[7]) > 3:
            lat1 = float(valores[2][:2])  # começa no início até o index 2 (não incluido)
            lat2 = float(valores[2][2:])  # do index 2 até ao fim...
            lat = lat1 + lat2 / 60.0;
            lon = float(valores[4][:3]) + float(valores[4][3:]) / 60.0
```

```
            print("Coordenadas: {0:s}{1:.6f} {2:s}{3:.6f}".format(valores[3], lat, valores[5], lon))

            print("Precisão (HDOP):", valores[8])

            print("Altitude ({0:s}): {1:s}".format(valores[10], valores[9]))


def escreve_dados(fila, estado):
    print("\n!!!!!!!!! Ordem para início de registo recebida !!!!!!!!!\n")

    current_datetime = datetime.now()
    str_current_datetime = current_datetime.strftime("%d-%m-%Y_%H-%M-%S")
    file_name = "LOG_" + str_current_datetime + ".csv"
    ficheiro = open(file_name, "w", newline="")
    campos = ['Data', 'Timestamp', 'Ix (Graus)', 'Iy (Graus)', 'ax (m/s^2)', 'ay (m/s^2)',
              'az (m/s^2)', 'wx (rad/s)', 'wy (rad/s)', 'wz (rad/s)']
    escrita = csv.DictWriter(ficheiro, fieldnames=campos)
    escrita.writeheader()

    while estado.value == 1:
        dados = fila.get()
        escrita.writerow({'Data': dados[0], 'Timestamp': dados[1],
                          'Ix (Graus)': dados[2], 'Iy (Graus)': dados[3],
                          'ax (m/s^2)': dados[4] * 9.80665, 'ay (m/s^2)': dados[5] * 9.80665,
                          'az (m/s^2)': dados[6] * 9.80665,
                          'wx (rad/s)': dados[7] * 0.0174533, 'wy (rad/s)': dados[8] * 0.0174533,
                          'wz (rad/s)': dados[9] * 0.0174533})

        time.sleep(0.08)  # dorme 80 ms

    print("\n!!!!!!!!! Ordem para terminar de registo recebida !!!!!!!!!\n")
    ficheiro.close()
```

```
if __name__ == '__main__':

    blue = LED(11)
    red = LED(12)
    green = LED(13)
    button = Button(17)
    count = 0

    while True:

        button.wait_for_press()
        blue.on()  # Liga

        button.wait_for_release()
        blue.off()  # Desliga

        count = count + 1

        if count == 1:
            red.off()  # Liga
            green.on()  # Desliga
        elif count == 2:
            red.on()  # Desliga
            green.off()  # Liga
            fila = Queue()  # cria uma fila para o envio dos dados do processo de leitura do sensor para o processo de escrita
            estado = Value('i',
                           1)  # i de inteiro e começa com o valor inicial um // permite os processos conhecerem o estado global para saberem quando devem terminar
            p_ler_sensor = Process(target=ler_sensor, args=(fila, estado))  # processo de leitura do sensor
            p_escreve_dados = Process(target=escreve_dados,
                                      args=(fila, estado))  # processo para escrita dos dados num ficheiro CSV
            p_ler_sensor.start()  # inicia o processo de leitura do sensor
            p_escreve_dados.start()  # inicia o processo de escrita dos dados
        elif count == 3:
```

```
red.on()  # Desliga
green.on()  # Desliga
estado.value = 2  # informa o processo de escrita do ficheiro CSV que deve terminar
p_escreve_dados.join()  # aguarda que o processo termine
estado.value = 0  # informa o processo de leitura dos dados do sensor que deve terminar
p_ler_sensor.join()  # aguarda que o processo termine
fila.close()  # fecha a fila de dados
count = 0
```

1.4 Condigo do Sistema de Aquisição de Dados para Ensaios em Campo – Segunda Versão

```
import threading
from scipy import signal
import FIR as fir
import numpy as np
from mpu9250_jmdev.registers import *
from mpu9250_jmdev.mpu_9250 import MPU9250
import csv
from datetime import datetime
from gpiozero import LED, Button
import serial
import time
from multiprocessing import Process, Queue, Value, Array
from gps3 import agps3



# processo para leitura do sensor
def ler_sensor(fila, estado):
    b_i, a_i = signal.butter(2, 0.003)  # filtro para a inclinação
    b_a, a_a = signal.butter(2, 0.200)  # filtro para a acelaração linear
    b_w, a_w = signal.butter(2, 0.200)  # filtro para a velocidade angular
    tempo_anterior = 0
    contador = 0

    mpu = MPU9250(
        address_ak=AK8963_ADDRESS,
        address_mpu_master=MPU9050_ADDRESS_68,
        address_mpu_slave=None,
        bus=1,
        gfs=GFS_1000,
        afs=AFS_8G,
        mfs=AK8963_BIT_16,
        mode=AK8963_MODE_C100HZ)
```

```
mpu.reset()
mpu.calibrateMPU6500()  # Calibrate sensors
mpu.configure()

abias = mpu.abias  # Get the master accelerometer biases
abias_slave = mpu.abias_slave  # Get the slave accelerometer biases
gbias = mpu.gbias  # Get the master gyroscope biases
gbias_slave = mpu.gbias_slave  # Get the slave gyroscope biases

z_Iax = signal.lfilter_zi(b_i, a_i)  # estado do filtro para aplicações de tempo real - Componente ax
z_Iay = signal.lfilter_zi(b_i, a_i)
z_Iaz = signal.lfilter_zi(b_i, a_i)
z_ax = signal.lfilter_zi(b_a, a_a)  # estado do filtro para aplicações de tempo real - Componente ax
z_ay = signal.lfilter_zi(b_a, a_a)
z_az = signal.lfilter_zi(b_a, a_a)
z_wx = signal.lfilter_zi(b_w, a_w)
z_wy = signal.lfilter_zi(b_w, a_w)
z_wz = signal.lfilter_zi(b_w, a_w)

tempo_atual = float(datetime.timestamp(datetime.now()))

tempo_seguinte = tempo_atual + 0.005  # próxima aquisição será daqui a 5ms

while estado.value > 0:  # enquanto estiver a escrever dados

    while (tempo_atual < tempo_seguinte):  # fica à espera pelo tempo exato para iniciar uma nova aquisição
        tempo_atual = float(datetime.timestamp(datetime.now()))

    tempo_seguinte += 0.005  # adiciona 5 ms ao tempo seguinte

    timestamp = datetime.now().strftime('%H:%M:%S.%f')

    data = datetime.now().strftime('%Y-%m-%d')
```

```
        A = mpu.readAccelerometerMaster()  # leitura dos dados do acelerometro

        Iax, z_Iax = signal.lfilter(b_i, a_i, [A[0]], zi=z_Iax)
        Iay, z_Iay = signal.lfilter(b_i, a_i, [A[1]], zi=z_Iay)
        Iaz, z_Iaz = signal.lfilter(b_i, a_i, [A[2]], zi=z_Iaz)

        Ix = np.degrees(np.arctan2(Iay, Iaz))  # inclinação em relação ao eixo x "tombamento"
        Iy = np.degrees(
            np.arctan2(-Iax, np.sqrt(Iay * Iay + Iaz * Iaz)))  # inclinação em relação ao eixo y "empinamento"

        ax, z_ax = signal.lfilter(b_a, a_a, [A[0]], zi=z_ax)
        ay, z_ay = signal.lfilter(b_a, a_a, [A[1]], zi=z_ay)
        az, z_az = signal.lfilter(b_a, a_a, [A[2]], zi=z_az)

        B = mpu.readGyroscopeMaster()  # leitura dos dados do giroscopio
        wx, z_wx = signal.lfilter(b_w, a_w, [B[0]], zi=z_wx)
        wy, z_wy = signal.lfilter(b_w, a_w, [B[1]], zi=z_wy)
        wz, z_wz = signal.lfilter(b_w, a_w, [B[2]], zi=z_wz)

        contador += 1

        if contador == 4:  # envia dados a cada 20 ms (5 ms * 4)
            contador = 0
            fila.put([data, timestamp, Ix[0], Iy[0], ax[0], ay[0], ax[0], wx[0], wy[0], wz[0]])

        tempo_atual = float(datetime.timestamp(datetime.now()))
        dormir = tempo_seguinte - tempo_atual - 0.002
        if (dormir > 0):
            time.sleep(dormir)  # dorme o tempo restante com uma folga de 2 ms


def escreve_dados(fila, estado, dados_GPS):
    print("\n!!!!!!!!! Ordem para início de registo recebida !!!!!!!!!\n")
```

```
    current_datetime = datetime.now()
    str_current_datetime = current_datetime.strftime("%d-%m-%Y_%H-%M-%S")
    file_name = "LOG_" + str_current_datetime + ".csv"
    ficheiro = open(file_name, "w", newline="")
    campos = ['Data', 'Timestamp', 'Ix (Graus)', 'Iy (Graus)', 'ax (m/s^2)', 'ay (m/s^2)',
            'az (m/s^2)', 'wx (rad/s)', 'wy (rad/s)', 'wz (rad/s)', 'lat', 'lon', 'alt', 'vel']
    escrita = csv.DictWriter(ficheiro, fieldnames=campos)
    escrita.writeheader()

    while estado.value == 1:
        dados = fila.get()

        # tempo = datetime.now()
        # tempo_anterior = float(datetime.timestamp(tempo))
        escrita.writerow({'Data': dados[0], 'Timestamp': dados[1],
                          'Ix (Graus)': dados[2], 'Iy (Graus)': dados[3],
                          'ax (m/s^2)': dados[4] * 9.80665, 'ay (m/s^2)': dados[5] * 9.80665,
                          'az (m/s^2)': dados[6] * 9.80665,
                          'wx (rad/s)': dados[7] * 0.0174533, 'wy (rad/s)': dados[8] * 0.0174533,
                          'wz (rad/s)': dados[9] * 0.0174533,
                          'lat': dados_GPS[0], 'lon': dados_GPS[1], 'alt': dados_GPS[2], 'vel': dados_GPS[3]})

        # tempo = datetime.now()
        # dif_t = float(datetime.timestamp(tempo)) - tempo_anterior
        # print("Tempo: ", dif_t) # tempo médio de escrita 0.3 ms mas por vezes chegou aos 1.1 ms
        time.sleep(0.010)  # dorme 10 ms que mesmo assim fica com uma folga muito grande...

    print("\n!!!!!!!!! Ordem para terminar de registo recebida !!!!!!!!!\n")
    ficheiro.close()


def ler_GPS(fila, estado, dados_GPS):
    gps_socket = agps3.GPSDSocket()
```

```
    data_stream = agps3.DataStream()
    gps_socket.connect()
    gps_socket.watch()

    for new_data in gps_socket:
        if new_data:
            data_stream.unpack(new_data)
            print('Altitude = ', data_stream.alt)
            print('Latitude = ', data_stream.lat)
            print('Longitude = ', data_stream.lon)
            print('Speed = ', data_stream.speed)

            if data_stream.lat != "n/a" and data_stream.lon != "n/a" and data_stream.alt != "n/a" and
data_stream.speed != "n/a":
                dados_GPS[0] = float(data_stream.lat)  # Latitude
                dados_GPS[1] = float(data_stream.lon)  # Longitude
                dados_GPS[2] = float(data_stream.alt)  # Altitude
                dados_GPS[3] = float(data_stream.speed)  # Velocidade

        if estado.value != 1:
            gps_socket.close()
            break
        time.sleep(0.0010)  # dorme 1 ms ---> Mudar para um valor adequado


if __name__ == '__main__':

    blue = LED(11)
    red = LED(12)
    green = LED(13)
    button = Button(17)
    count = 0

    while True:
```

```
        button.wait_for_press()
        blue.on()  # Liga
        button.wait_for_release()
        blue.off()  # Desliga


        count = count + 1
        if count == 1:
            red.off()  # Liga
            green.on()  # Desliga
        elif count == 2:
            red.on()  # Desliga
            green.off()  # Liga
            fila = Queue()  # cria uma fila para o envio dos dados do processo de leitura do sensor para o processo de escrita
            estado = Value('i',
                           1)  # i de inteiro e começa com o valor inicial um // permite os processos conhecerem o estado global para saberem quando devem terminar
            dados_GPS = Array('d', range(4))  # (d -> double) array partilhado para os dados do GPS (4 valores)
            p_ler_sensor = Process(target=ler_sensor, args=(fila, estado))  # processo de leitura do sensor
            p_ler_GPS = Process(target=ler_GPS, args=(fila, estado, dados_GPS))  # processo para leitura do GPS
            p_escreve_dados = Process(target=escreve_dados, args=(
                fila, estado, dados_GPS))  # processo para escrita dos dados num ficheiro CSV
            p_ler_sensor.start()  # inicia o processo de leitura do sensor
            p_ler_GPS.start()  # inicia o processo de leitura do GPS
            p_escreve_dados.start()  # inicia o processo de escrita dos dados
        elif count == 3:
            red.on()  # Desliga
            green.on()  # Desliga
            estado.value = 2  # informa o processo de escrita do ficheiro CSV que deve terminar
            p_escreve_dados.join()  # aguarda que o processo termine
            p_ler_GPS.join()  # aguarda que o processo termine
            estado.value = 0  # informa o processo de leitura dos dados do sensor que deve terminar
            p_ler_sensor.join()  # aguarda que o processo termine
            fila.close()  # fecha a fila de dados
            count = 0
```