Universidade de Aveiro
2016
Departamento de Eletrónica, Telecomunicações e Informática

Sérgio Manuel Rama Cascão

Sistema de Telemetria da Temperatura de Pneus na Competição Automóvel

Universidade de Aveiro
2015
Departamento de Eletrónica, Telecomunicações e Informática

Sérgio Manuel Rama Cascão

Telemetria na competição automóvel de temperatura pneus

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica de Prof. Rui Martins, Professor do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

O júri

Presidente	Prof. Doutor Paulo Bacelar Reis Pedreiras Professor Auxiliar da Universidade de Aveiro
Vogais	Doutor Paulo Jorge de Campos Bartolomeu Diretor da Globaltronic - Eletrónica e Telecomunicações, S.A.
	Prof. Doutor Rui Manuel Escadas Ramos Martins Professor Auxiliar da Universidade de Aveiro (orientador)

Agradecimentos

Ao meu orientador, Professor Rui Manuel Escadas Ramos Martins pela disponibilidade, confiança e motivação durante esta etapa final do meu Mestrado.

À minha família que me ajudou durante estes anos todos da universidade, que me apoiaram neste meu percurso universitário.

Por último, mas não menos importante, agradeço aos meus colegas e amigos que dentro e fora do ambiente académico me ajudaram e tornaram mais fácil a conclusão desta etapa.

palavras-chave

Sensores temperatura, Veículo, Competição automóvel, Telemetria; Análise de dados, Microcontroladores.

Resumo

A presente dissertação tem como objetivo o desenvolvimento de um sistema protótipo de monitorização da temperatura dos pneus de um automóvel de competição. O veículo que implementará o sistema trata-se de um Norma M20 FC, pertencente à equipa Torres Rally Team.

O projeto comtempla toda a seleção, montagem e programação dos dispositivos físicos necessários à monitorização e comunicação, bem como o desenvolvimento do *Software* que armazena, processa e representa toda a informação gerada pelos sensores.

A grande motivação deste trabalho de investigação surge da necessidade crescente de otimização no *setup* do veículo em causa.

É portanto, de extrema importância a criação de um sistema de informação que processe e represente os dados da forma mais clara, simples e precisa, possibilitando a rápida e eficaz intervenção na parametrização do veículo e a consequente melhoria dos tempos na competição.

Numa abordagem mais técnica, este projeto visa o auxílio na afinação da direção do carro, fornecendo à equipa de apoio valores de temperatura dos quatro pneus em 3 zonas distintas (meio e extremidades), permitindo a identificação das áreas de maior desgaste. Os dados adquiridos, levarão a ações de parametrização do veículo que favoreçam a uniformização do desgaste do pneu.

De seguida, serão descritos os procedimentos e os componentes utilizados na comunicação entre dispositivos, sensores de medição de temperatura, assim como as ferramentas utilizadas no desenvolvimento de todo o Software, quer na parte do microcontrolador, quer na aquisição, tratamento e representação de toda a informação recolhida.

Keywords Sensors temperature, vehicle, automobile Competition, Telemetry; Data analysis, Microcontrollers

Abstract

The present Thesis has as its main purpose the development of a prototype system for the monitoring of tires' temperature of a competition car.
The car to be equipped with the device is a Norma M20 FC owned by Torres Rally Team.

The project includes the selection, assembly and programming of the physical devices necessary to both monitoring and communication as well as the development of the Software responsible for data storage, processes and representation all the data produced by the temperature sensors.

The main motivation of this research work comes up from the arising need for car setup´s enhancement. Thereby, it is extremely important the creation of an IT system which processes and shows data in the most clear, simple and precise way, allowing a quick end effective intervention and consequent race times' improvement.

On a technical approach, this project aims to assist on the setting of the car's steering system and aerodynamics, providing to the support team temperature values in three different areas of each tire (middle and extremities), allowing the identificationby measuring and delivering the tyre of the areas with major wear. The acquired data will lead to parametrization actions helpful for tyres wear standardization.

This work also describes all the involved procedures and the components used on the communication between devices, measuring sensors, as well as the tools used on Software's development, either on microcontroller or on data representation.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Códigos

Lista de Acrónimos

Acrónimo	Significado
AJAX	Asynchronous Javascript and XML
API	Application Programming Interface
ATLAS	Advanced Telemetry Linked Analysis System
BD	Base de Dados
CSS	Cascading Style Sheets
CPU	Central Processing Unit
ECU	Electronic Control Unit
FIA	Federação internacional do Automóvel
FOV	Field of View
GPIO	General Purpose Input/Output
GPS	Global Positioning System
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
I²C	Inter-Integrated Circuit
IDE	Integrated Development Environment
IR	Infra-red
JSON	JavaScript Object Notation
MDX	Multidimensional Expressions
MB	Mega Bytes
PCB	Printed Circuit Board
PHP	Personal Home Page
PWM	Pulse-Width Modulation
RF	Radio Frequency
RAM	Random Access Memory
RPM	Rotações por minuto
SQL	Structured Query Language
SRAM	Static Random Access Memory
SSH	Secure Shell
UART	Universal asynchronous receiver/transmitter
UI	User Interface
URL	Uniform Resource Locator

1. Introdução

1.1 Enquadramento

A presente tese nasce da parceria criada entre a *Torres Rally Team* e a Universidade de Aveiro que, entre outros objetivos, pretendia estudar do ponto de vista técnico como era possível melhorar o desempenho competitivo da equipa na competição em que estava inserida no Campeonato Nacional de Montanha. Na verdade, este já era bastante bom, registando inclusivamente vitórias em algumas etapas da referida competição, o que demonstrava que o carro *Norma M20 FC* (fabricado pela *Aurora Motosports*) utilizado pela equipa (aliado às qualidades reconhecidas de pilotagem do piloto e diretor da mesma) era um carro competitivo, pelo menos no panorama nacional. Na Figura 1 apresenta-se o *Norma M20 FC* utilizando já o esquema de cores que reflete a parceria com a Universidade de Aveiro.

Figura 1. Norma M20 FC da equipa Torres Rally Team.[1]

Foram realizadas diversas visitas de trabalho onde houve a oportunidade de reunir com pessoal técnico com experiência na preparação deste tipo de carros, tendo sido referido que, de entre outras possibilidades, a medição da temperatura dos pneus seria uma mais-valia, dada a importância primordial deste parâmetro no ajuste de um carro de corridas. Esta estratégia possibilitaria determinar as áreas de maior desgaste do pneu proporcionando a otimização da afinação dos diversos sistemas do veículo, nomeadamente da direção, suspensão e apêndices aerodinâmicos. Por outro lado, a obtenção de um desgaste uniforme dos pneus e, portanto, uma melhor utilização

destes, poderia ainda contribuir adicionalmente para uma melhoria dos tempos de prova e de qualificação.

A decisão de seguir por esta via foi confirmada quando se verificou que nas “6 Horas do Estoril 2014” a equipa francesa “*Team TFT*” (que aí se sagrou campeã da competição “*Challenge d’Endurance Protos*”) com um *NORMA M20 FC*, tinha um sistema semelhante ao que será o objetivo desta dissertação. De facto, tal sistema, que segundo os técnicos da equipa funcionava ainda “com muitos problemas” apenas existia num dos dois carros em competição da equipa, precisamente no carro que se tinha sagrado campeão. Obviamente que não querendo inferir que a vitória se deveu à utilização do sistema de medição da temperatura dos pneus, este facto confirmou que desenvolver um sistema destes e dotar a equipa *Torres Rally Team* com esta tecnologia, traria uma clara vantagem competitiva.

É na solução descrita para o problema acima enunciado que se enquadra a presente dissertação.

1.2 Objetivos

Como referido, a motivação para este trabalho foi o desenvolvimento de tecnologias capazes de melhorar o desempenho competitivo de uma equipa de competição automóvel, nomeadamente dotando-a da capacidade de conhecer a temperatura dos pneus. No entanto, cedo se percebeu que realizar tal tarefa por si só não seria verdadeiramente uma mais-valia, sem estar associada a um conjunto integrado de *Software* que, em conjunto, constituísse uma plataforma de suporte à equipa técnica da *Torres Rally Team*.

Assim, para além de um conjunto de *Hardware*, foi dada bastante importância ao desenvolvimento de uma ferramenta de *Software* que permitisse mostrar, armazenar e organizar os dados adquiridos, no caso concreto, a leitura da temperatura de cada pneu em três zonas distintas (meio e extremidades tal como na Figura 2).

Figura 2. Modo de captura da temperatura no pneu.

Obviamente os sensores não podem estar dentro do pneu (note-se que os pneus na competição automóvel são componentes altamente descartáveis). Assim como não podem estar nas jantes, procedeu-se à utilização de sensores de temperatura do tipo *contactless* baseados em Infra Red (IR). Todos os sistemas de comunicação recorrem à tecnologia Radio Frequency (RF) possibilitando uma fácil instalação na estrutura do veículo, sem prejuízo que numa versão final as comunicações no carro fossem cabladas por questões de fiabilidade.

Simultaneamente à medição da temperatura dos pneus, é necessário enviar os dados recolhidos para a equipa técnica com tempos de transmissão baixos, para que possam ser analisados o mais rápido possível, descartando-se as soluções em que os dados adquiridos são guardados no carro em memória não volátil (por exemplo um SD-CARD) e recolhidos no fim dos testes ou das provas. Deve-se dizer que esta solução é muito utilizada no âmbito da competição automóvel, pois tem vantagens de fiabilidade e custo, não sendo tão penalizadora tendo em consideração que os regulamentos da maioria das competições automóveis proíbem a utilização de "ajudas" das boxes em tempo de corrida ou qualificações. No entanto, na realização de testes, o conhecimento imediato dos dados sem que os carros tenham que regressar às boxes foi considerado suficientemente importante.

Os objetivos principais para o *Software*, para além do desenvolvimento da estrutura de comunicação e de captura das temperaturas, centraram-se numa cuidada interface gráfica para apresentação dos dados clara e intuitiva, de modo a que o tempo de resposta por parte da equipa possa ser o mais curto, possibilitando uma intervenção mais rápida e eficaz.

1.3 Estrutura da dissertação

Este documento está organizado em cinco capítulos. No capítulo 2 é apresentado o estado da arte referente à história da telemetria assim como o seu aparecimento e evolução na fórmula 1. Também são abordados dois sistemas utilizados para monitorização nesta competição e os sensores de temperatura mais comuns nesta modalidade.

No capítulo 3 é apresentado o *Hardware* assim como *Software* de todo o sistema desenvolvido.

No capítulo 4 apresentam-se os testes realizados e respetivos resultados.

No capítulo 5 são apresentadas as conclusões e propostas para melhoramentos futuros.

2. Estado da Arte

Neste capítulo vão ser apresentados os conceitos mais relevantes para este trabalho, como sejam a telemetria e os sensores para medição da temperatura dos pneus em carros de competição.

Será ainda contextualizado o estado da arte e algumas aproximações para a resolução do problema apresentado neste trabalho. Serão também focadas algumas das plataformas mais maduras que surgiram como forma de colmatar as várias necessidades existentes no âmbito das ferramentas de apoio às equipas da Fórmula 1, a telemetria e os sensores de medição de temperatura em pneus de carros de competição.

2.1 História da Telemetria

O termo telemetria pode ser aplicado a todos os sistemas que são medidos remotamente. Logo, para ter um sistema de telemetria é necessário ter um ou mais emissores e um recetor. Nos dias de hoje, a comunicação entre o transmissor e o recetor é feita, normalmente, sem fios, podendo embora ser criada através de ligação física (cabos).

Contextualizando o tema e dando a conhecer um pouco a sua origem, a telemetria terá sido inicialmente usada para monitorizar a distribuição de energia elétrica. O primeiro sistema surge no ano de 1912, em Chicago, tendo como base o uso de linhas telefónicas para transmissão de dados de várias centrais elétricas para respetiva supervisão [2]. A telemetria rapidamente se generalizou e passou a ser utilizada nas mais diversas situações, tal como ao contexto deste trabalho – os desportos motorizados.

No decorrer das décadas, esta tecnologia terá vindo a desempenhar um papel cada vez mais importante nos desportos motorizados, tornando possível um auxílio sustentado da equipa ao piloto. No final dos anos 80, as equipas conseguiam já enviar

dados do carro quando este passava perto das boxes na reta da meta, sendo a troca de informação efetuada em modo *Burst*[1][3].

No início dos anos 90, com o avanço da tecnologia, a comunicação passou a ser estabelecida de forma contínua na maioria das pistas. Porém, em circuitos como Monza, a interferência das árvores causava falhas na comunicação levando à sua interrupção e consequente perda de dados em determinadas zonas do traçado. Tal problema terá sido posteriormente solucionado (anos 2000), sendo a informação novamente transmitida assim que a comunicação fosse reestabelecida quando o carro voltasse a uma zona de cobertura [4].

Neste tipo de competição os sistemas de telemetria monitorizam o veículo a cada segundo e conseguindo assim disponibilizar previsões futuras, por forma a evitar acidentes ou até melhorar tempos durante as provas. São bons exemplos, o Moto GP e a Fórmula 1, desportos nos quais a telemetria aparece na sua forma mais densa e complexa.

Em 2001 foi estabelecida a comunicação nos dois sentidos, o que veio a permitir à equipa técnica a alteração das configurações dos veículos em tempo real. Passaria a ser possível, pelos engenheiros, realizar ajustes aos mapas do motor ou mesmo ao diferencial, garantindo o foco do piloto única e simplesmente na condução e otimização dos tempos de prova. Pouco mais tarde, em 2003, devido à forte evolução e ao controlo massivo do veículo proporcionado pela telemetria, a única comunicação bidirecional permitida passaria a ser via rádio entre piloto e equipa. As restantes comunicações viriam a ser banidas das competições pois defendia-se que atribuíam exagerado protagonismo à máquina, descredibilizando o talento do piloto [5][3][6].

2.2 O funcionamento da telemetria na Fórmula 1

Ao abordar os sistemas telemétricos, é imperativo realçar o mais sofisticado na atualidade, o utilizado na Fórmula 1. De facto, a Fórmula 1 é o desporto motorizado com maior índice de investimento, permanecendo na vanguarda da tecnologia, o que leva a que cada componente desenvolvido seja constantemente analisado e otimizado proporcionando uma melhoria contínua na segurança e eficiência do veículo [7].

[1] Burst mode ou modo rajada é um método de transferência de informação no qual os dados são reunidos e enviados todos de uma só vez, proporcionando assim uma transmissão de alta velocidade[62].

Atualmente, os veículos de Fórmula 1 estão equipados com um elevado número de sensores capazes de informar, de forma precisa e com um baixo tempo de atraso, as equipas técnicas acerca do comportamento do carro, permitindo, num determinado momento, averiguar a existência de quaisquer anomalias ou mesmo informar o piloto visando a melhoria da sua condução. Todos os dados são obtidos através de um *Electronic Control Unit* (ECU), encriptados e enviados, via antena telemétrica, para os postos de análise dos dados. A informação mais recente é armazenada em *buffer*, caso falhe a transmissão, e serão reenviados assim que a comunicação esteja disponível novamente [8].

Na fase seguinte, toda a informação é transferida para recetores RF colocados ao longo do circuito, garantindo a sua total cobertura, evitando falhas de comunicação ao longo da prova. A comunicação requer entre 1000 a 2000 canais, os quais funcionam a uma frequência de 1.5 GHz, ou similar. As frequências utilizadas são condicionadas, em cada prova, pelas autoridades locais, sendo permitidas apenas gamas de valores livres, evitando interferência com processos de comunicação alheios à prova. A quantidade de informação recolhida numa sessão de 90 minutos pode variar entre os 5 e os 6 Gigabytes, já em formato comprimido [8]. Após a receção da informação, é necessário proceder ao processamento e respetiva análise de modo a que esta seja representada de uma forma precisa e de fácil interpretação para a equipa de engenheiros.

Em 2006, a Federação Internacional do Automóvel (FIA) solicita o desenvolvimento de uma ECU, preconizando-a como única e comum a todos os carros da Fórmula 1 para os campeonatos de 2008, 2009 e 2010. A solução final dá pelo nome de Advanced Telemetry Linked Analysis System (ATLAS), sistema desenvolvido conjuntamente pela *McLaren Electronic Systems* e pela Microsoft. O ATLAS recolhe todos os dados provenientes do carro, armazenando-os numa Base de Dados (BD) Structured Query Language (SQL) a partir da qual são de imediato acedidos por computadores devidamente configurados. Uma das principais características deste sistema é a sua fácil integração com o Microsoft Excel, *Software* vastamente difundido, permitindo a rápida análise dos dados por parte do utilizador. A figura 3 representa o fluxo de informação, desde que é recolhida na ECU, até à sua representação nos terminais de análise [8]–[10].

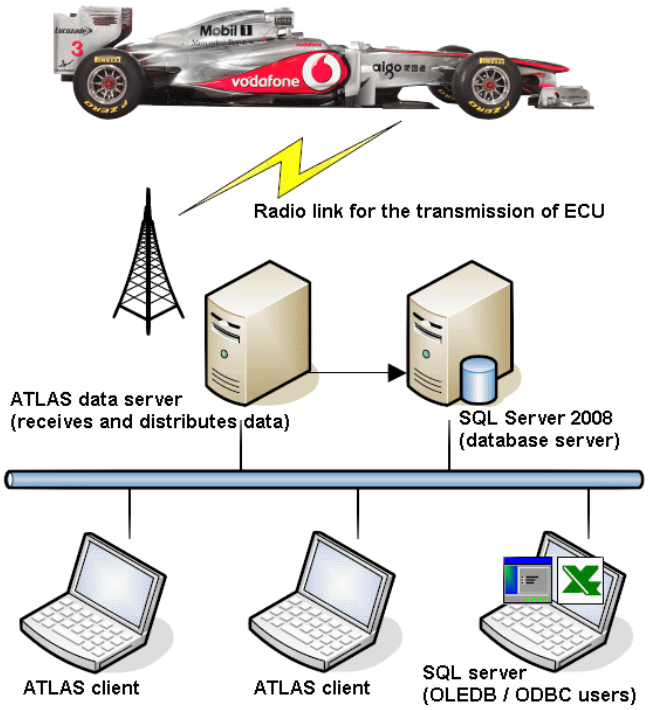

Figura 3. Esquema de comunicação do sistema ATLAS[8].

A figura 4 apresenta as principais variáveis telemétricas registadas pela equipa *Caterham F1 Team*. Note-se a simplicidade na apresentação dos dados, o que possibilita a rápida análise do comportamento do veículo ao longo do tempo (eixo 'xx). A linha vermelha representa Rotações Por Minuto (RPM), a azul corresponde à velocidade, a verde representa a aceleração lateral, a roxa corresponde à mudança em que se encontra a caixa, a laranja informa a posição do pedal acelerador e, por fim, a castanha representa a pressão do sistema de travagem [9].

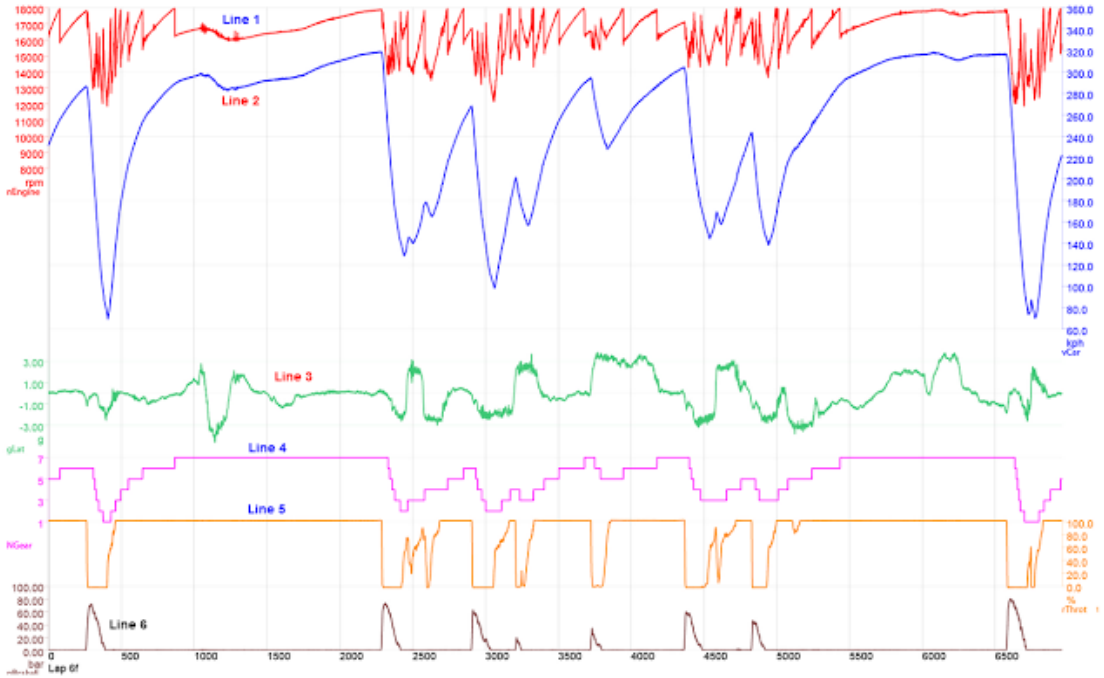

Figura 4. Representação da informação no Software ATLAS [9]**.**

Recentemente, a SAP, líder mundial em Software empresarial, tem dado suporte à *McLaren F1 Race* na análise de informação telemétrica. O HANA, plataforma utilizada, consiste numa combinação de *Hardware* preparado para processar grandes fluxos de dados em tempo real. Uma das grandes vantagens desta tecnologia reside no facto de os dados serem processados a partir da *Random Access Memory* (RAM) e não do disco rígido, o que permite à *Central Processing Unit* (CPU) o rápido acesso aos dados para respetivo processamento. No entanto, a RAM é uma memória volátil, propícia a perda de dados em caso de falha de alimentação no sistema. A fim de evitar este fenómeno, quando são detetadas alteração em qualquer das suas páginas, esta(s) são marcadas para o armazenamento em disco, garantindo a integridade dos dados [11][12].

A arquitetura desta plataforma é constituida pelo *index server,* onde é armazenada e processada toda a informação, atravéz de SQL ou Multidimensional Expressions (MDX). A camada *Persistence Layer* da BD é responsável pela durabilidade e atomicidade das transações. Este garante que a BD pode ser restaurada para o estado mais recente após o reinicio inesperado do sistema e que as transações são completamente executadas ou completamente desfeitas. O *index server* usa o servidor de *Preprocessor Server* para a análise de dados de texto e extrair as informações em que as capacidades de pesquisa de texto são baseados. O *name server* possui as informações sobre a topologia do sistema *SAP HANA*. Num

sistema distribuído, o *server name* sabe onde os componentes estão a funcionar e a informação presente em cada servidor. O bloco any app server consiste na componente que dá acesso aos dados presentes na DB. Por fim, a componente user interface (UI) onde são representados os dados pretendidos, a figura 5 representa toda a descrição feita [13].

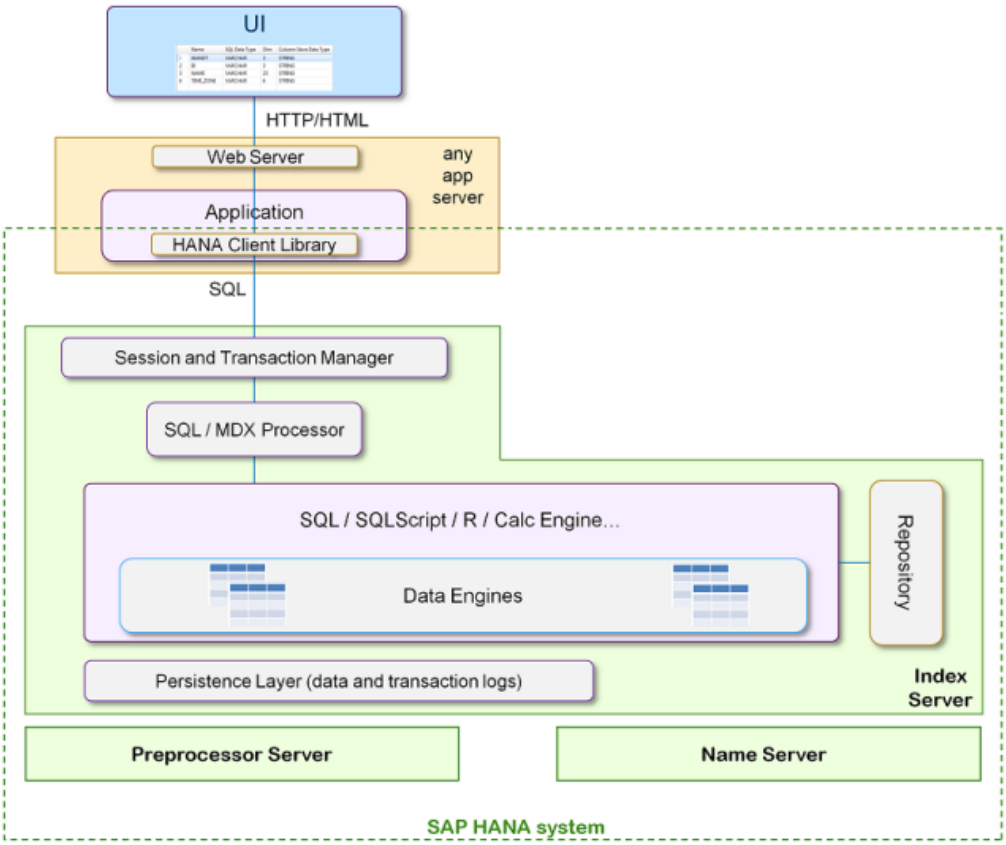

Figura 5. Arquitetura SAP [13].

A interface gráfica desenvolvida no âmbito do HANA, consegue disponibilizar uma elevada densidade de informação aliada a uma visualização simples e eficaz, disponibilizando previsões futuras sobre a troca de pneus ou o reabastecimento de combustível. Esta informação disponibilizada pela plataforma permite à equipa técnica tomar decisões mais oportunas e precisas. A figura 6 representa um *dashboard* exemplar com a representação da informação obtida da Vodafone *Mclaren Mercedes* (campeonato de 2002), é possível ver representada a velocidade a que se encontra o carro, as RPM, a mudança atual em que se encontra o veículo, o posicionamento das rodas dianteiras entre outros dados estatísticos relevantes à performance do carro e do piloto [10], [14].

Figura 6. Dashboard SAP referente ao Vodafone McLaren Mercedes [10] **.**

2.3 Sensores de temperatura para medição remota da temperatura de pneus em competição automóvel.

Os termómetros de infravermelhos foram originalmente desenvolvidos para medir a temperatura de objetos em situações em que um termómetro de contacto normal não pode ser utilizado, devido, por exemplo, ao acesso limitado ou à existência de movimento do objeto a ser medido. Os sensores infravermelhos são mais rápidos e precisos na medição de temperatura comparativamente com os sensores de contacto comuns. O princípio de funcionamento baseia-se na captação da radiação emitida pelo objeto, em que a temperatura registada varia diretamente com a quantidade de radiação emitida. Na figura 7 apresenta-se a constituição típica deste tipo de sensores.

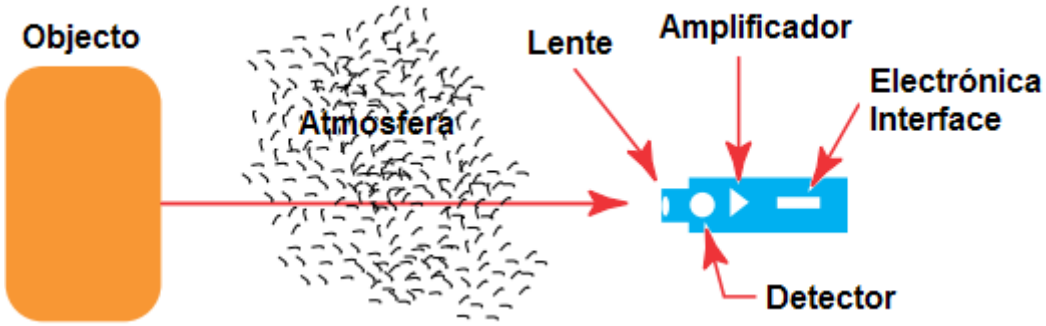

Figura 7. Esquema da constituição típica dos sensores de temperatura infravermelhos.

Perante a exigência das condições de medição de temperatura no pneu de um carro de competição, tem sido incessante o esforço para obtenção de dispositivos mais precisos e robustos. É neste meio de competição que surgem os sensores mais avançados utilizados na atualidade [15].

Uma das maiores referências é a *Texys International* (França), empresa que tem vindo a desenvolver, ao longo dos últimos 20 anos, sensores destinados especificamente a aplicações automóvel, competição e aeronáutica através da sua linha de dispositivos Texense. A empresa tem vindo a trabalhar com os profissionais do desporto motorizado desde 1999 e está envolvido em todas as principais corridas de automóveis campeonatos internacionais: Fórmula1*, MotoGP, WRC, Nascar IRL, Le Mans Series*. Uma vez que os sensores referidos estão inseridos na maior parte das equipas de desporto motorizado, vão assim de seguida ser apresentados os mais utilizados na captura de temperatura em pneus [16][17][18].

2.3.1 Sensores Temperatura Infravermelhos Texense IRN8C e Texense IRN4C

Os sensores *Texense IRN8C* (figura 8) e *IRN4C* (figura 9) foram desenvolvidos especificamente para aplicação na Fórmula 1. Uma vez instalados à distância correta, conseguem medir a temperatura em linha de oito pontos ao longo da largura do pneu, no caso do *IRN8C*, ou quatro pontos, no caso do *IRN4C* (aplicado em pneus de menor largura ou onde menor resolução espacial é permitida). A distância ideal a que devem ser colocados depende da largura do pneu e pode variar entre os 20 e os 70 centímetros.

A sua gama de aquisição de temperatura é -20ºC a 200ºC enquanto ao erro de leitura é inferior a ± 1%FS (a uma temperatura de 200ºC). A informação do fabricante refere ainda uma boa compensação de temperatura, mas não indica nenhuma especificação para o erro de *offset* que será relevante a temperaturas no início da sua gama de amostragem. Estes sensores são também extremamente compactos, o que torna as possibilidades da sua instalação abrangentes, tendo como dimensões 31 x 11 x 17mm, pesando ambos 15g.

A transferência de dados destes sensores tem como base o protocolo *CAN Bus* 2.0 B, comunicação em série que permite um controlo distribuído em tempo real com elevado nível de segurança. O sistema consegue uma aquisição máxima de 10 amostras por segundo.

Como já referido, as características apresentadas tornam os *Texense* extremamente robustos para o tipo de aplicações em questão. No entanto, o seu custo é o fator menos favorável, atingindo os valores de 860€ (*IRN4C*) e 1440€ (*IRN4C*) [19]–[24]

Figura 8. Sensor de temperatura por infravermelhos IRN8C [24]**.**

Figura 9. Sensor de infravermelhos Texsense IRN4C [25]**.**

2.3.2 Sensor Temperatura Infravermelho Texense IRN8W

O sensor *IRN8WS* (figura 10) tem por base a mesma estrutura tecnológica que os anteriormente apresentados, com a grande particularidade de integrar um sistema de comunicação sem fios. Foi desenvolvido e projetado para simplificar ligações e reduzir a fonte de potenciais problemas elétricos. Mais uma vez esta versão permite a leitura de oito pontos na largura do pneu, mas aqui a comunicação é feita por RF.

O sistema de comunicação é composto por um recetor IRN8W-M (*Master*) capaz de gerir, no máximo, 8 sensores IRN8W-S (*Slave*). Usando o protocolo CAN 2.0 (A ou B) para comunicação, o IRN8W-M permite a aquisição máxima de 10 amostras por segundo. Os dispositivos *Master* e *Slave* podem comunicar a frequências de radio de 868, 902 ou 915 MHz e conseguem uma autonomia máxima de vinte e quatro horas. As suas especificações técnicas são em tudo idênticas às dos seus antecessores, no entanto, as dimensões aumentam para os 44,5x21x14mm. Quanto aos valores do *Master* é de 639€ enquanto cada *Slave* custa 1930€ [16], [26], [27],[28],[29].

Figura 10. Sensores infravermelho wireless 8ch IR [16].

3 Análise de Requisitos e Arquitetura do Sistema Implementado

No decorrer do presente capítulo será apresentada a estrutura criada para implementação do sistema de telemetria automóvel (*Hardware* e *Software*). É importante salientar que todos os componentes foram selecionados conciliando uma forte preocupação na redução de custos com a garantia de um sistema suficientemente robusto para validação da solução desenvolvida.

Para uma melhor compreensão do sistema, este pode ser dividido em duas componentes principais: a primeira, representada na figura 11 à esquerda, que será designada por *Sensor Aggregator Gateway,* é responsável pela medição, recolha e transmissão dos valores de temperatura dos pneus; por sua vez a segunda componente, representada à direita da figura 11, designada por *Processing Unit*, recebe, processa, armazena e representa os dados adquiridos sob forma de gráfico de linhas e cada linha representa uma das três zonas do pneu. De seguida serão apresentados estes componentes detalhadamente.

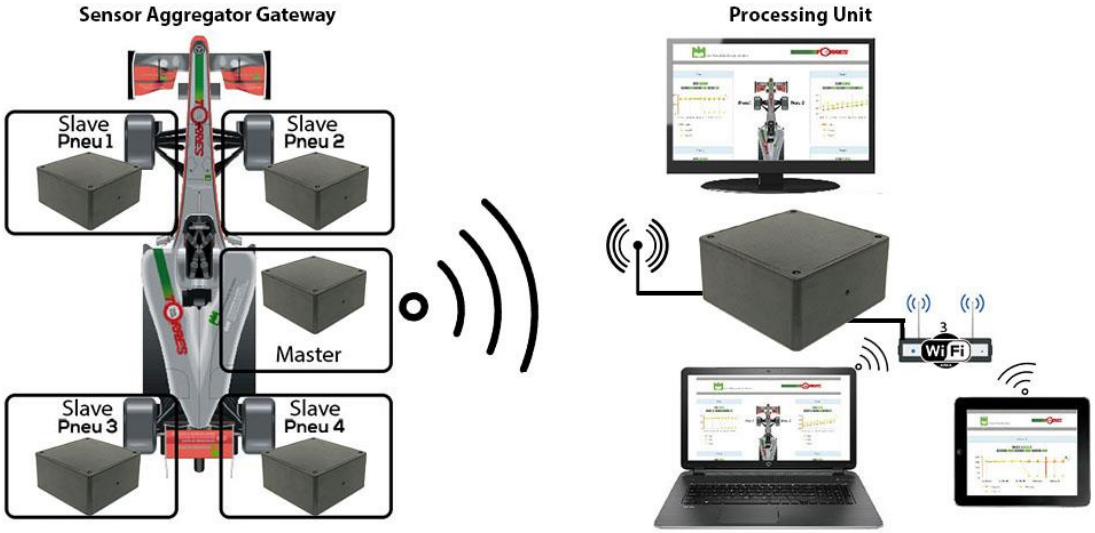

Figura 11. Representação do Processing Unit e Sensor Aggregator Gateway.

3.1 Sensor Aggregator Gateway Hardware.

Esta componente, a ser instalada no carro, consiste em quatro sistemas (um por cada roda) *Slave*, que enviam os dados da temperatura dos pneus, para um

módulo *Master* que controla todo o fluxo de informação. Na sua implementação, tanto os módulos *Slave* como *Master,* são baseados numa placa com microcontrolador, sendo que os primeiros são ainda compostos pelos três sensores de temperatura, distribuídos uniformemente ao longo da largura do pneu (meio e extremidades), e um módulo de comunicação RF (433 MHz), permitindo a comunicação com o sistema *Master* sem a necessidade de estabelecer a passagem de cabos pelo carro. A figura 12 representa a disposição dos sensores sobre a roda, representados com o círculo verde, e a vermelho a área onde é feita captura da respetiva temperatura. O *Master* possui o mesmo sistema de comunicação RF, que permite a comunicação com os módulos *Slaves* e com a *Processing Unit* que se encontra fora do carro (tipicamente, mas não obrigatoriamente) nas boxes.

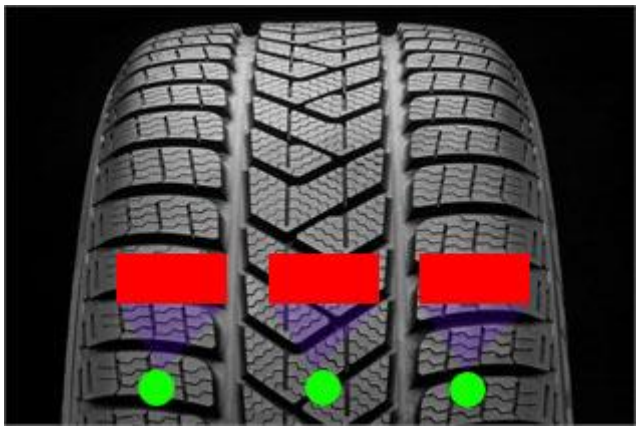

Figura 12. Esquema de leitura e posicionamento dos sensores sobre o pneu.

A solução *Master-Slave* foi projetada com o objetivo de simplificar o sistema, dado que todas as comunicações são controladas pelo *Master,* evitando assim colisões na ligação entres os diferentes módulos, assim como facilitar a inserção de mais *Slaves* no sistema, possibilitando assim a escalabilidade de todo o sistema. De referir que, se fosse considerada a solução de cada unidade de medição poder comunicar diretamente com a base, seriam necessários quatro transmissores/antenas de maior alcance e consequentemente mais caras, com acrescido consumo de energia, o que implicaria um componente com maior volume e peso total, causando como consequência maior dificuldade na sua implementação no veículo. Por outro lado, mais canais de comunicação, significa potencialmente maior perigo de interferência ou congestionamento com outras que existam no veículo (por exemplo comunicação áudio piloto-boxes) e com a dos outros veículos. Numa outra abordagem ao sistema desenvolvido, existe a possibilidade de agregar mais Slaves, com os

mesmos ou diferentes sensores, ao *Master*, e enviar tudo pelo mesmo *Master* (sistema modular).

Na solução proposta apenas dois módulos de comunicação de longo alcance são necessários (um no carro e outro nas boxes) evitando assim todas as adversidades descritas anteriormente. Uma vez que as frequências de rádio livres variam entre países, esta arquitetura tem a vantagem de facilitar a troca dos módulos de comunicação de longo alcance sem ter que alterar o modo como é feita a comunicação entre os diversos dispositivos.

Como já referido, a solução de comunicação por cabo, seria igualmente válida e até mais simples de implementar. No entanto, nas reuniões com a equipa técnica da *Torres Rally Team* considerou-se que a solução implementada seria a que levaria às menores alterações no carro. Uma vez que o sistema implementado é de baixo consumo de energia e, para funcionar, apenas necessita de uma alimentação de 5V, foi decidido pelo menos para o protótipo que a sua alimentação proviria de uma bateria dedicada para o efeito e não da bateria do veículo, cumpre assim o requisito solicitado pela equipa.

A figura 13 representa a composição dos quatro *Slaves* assim como do *Master* ilustrando também o modo de implementação de cada um dos módulos. Sendo de notar que cada *Slave* ficaria o mais próximo possível de cada pneu e o *Master* numa posição central do carro para que a força do sinal de comunicação fosse o mais semelhante possível entre os *Slaves*.

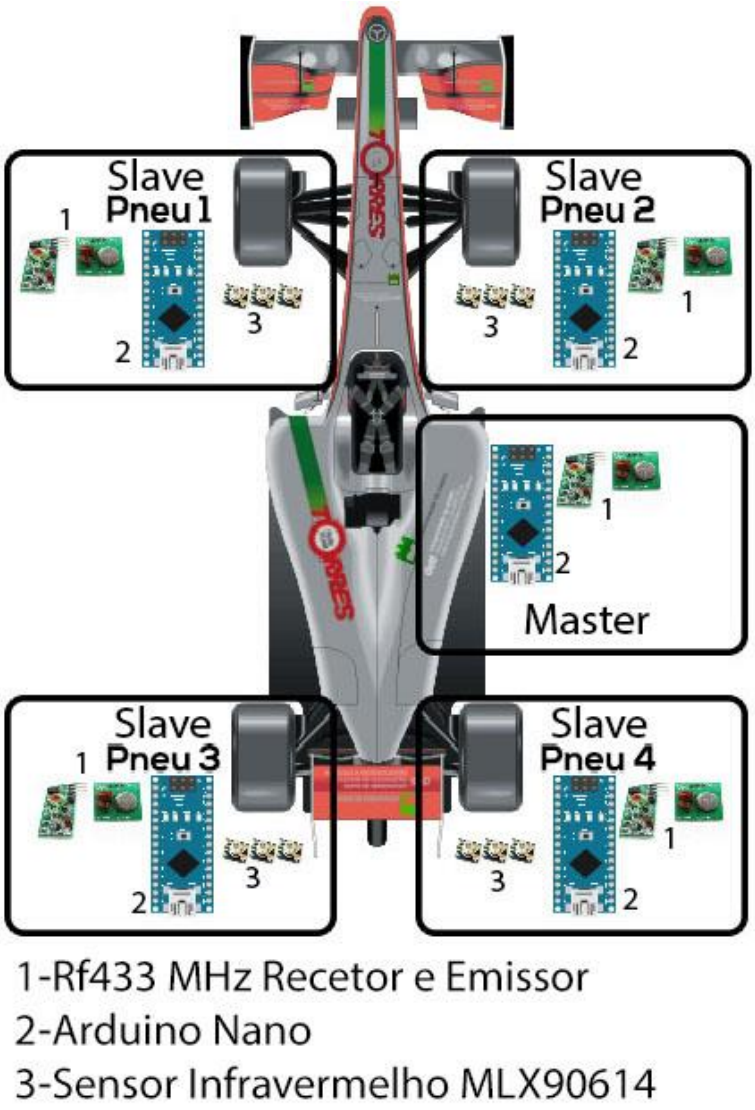

Figura 13. Hardware Sensor Aggregator Gateway.

As comunicações são comandadas pelo *Master* que de modo ordenado e sequencial envia em *broadcast* o identificador único de cada *Slave*. Por sua vez, todos os *Slaves* escutam as mensagens trocadas, mas só respondem com os valores das temperaturas capturadas quando o identificador corresponder ao seu. Quando o *Master* recebe a informação dos *Slaves* formata-a numa mensagem com o identificador "5-" e envia a mensagem para a *Processing Unit*. Este, ao receber a mensagem, trata a informação e armazena-a para posterior análise (em rigor, este "posterior" pode ser também quase simultaneamente). Importante referir que, todo este processo de comunicação é feito sem haver qualquer tipo de confirmação sobre a receção da informação, todas as comunicações têm o tempo máximo de um segundo para iniciar. Caso a comunicação não se inicie nesse período é descartada essa transferência de informação. Como a comunicação entre dispositivos funciona de

modo cíclico, alguns ciclos depois voltam a ser invocada para uma nova transferência de informação. Esta implementação garante assim uma menor latência entre a captura da informação e a sua representação. Para melhor interpretação desta descrição, a figura 14 representa de modo esquemático todo o processo descrito anteriormente. Posteriormente, utilizando uma abordagem *TOP-DOWN* serão descritos os componentes utilizados na implementação dos módulos *Master* e *Slave*.

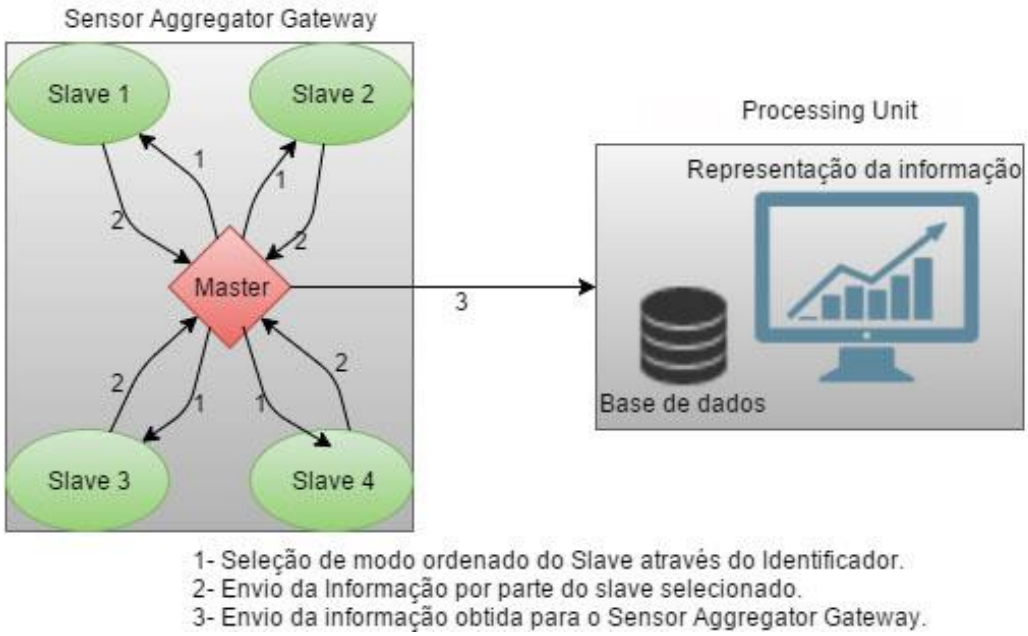

Figura 14. Esquema de comunicação entre os diversos dispositivos.

3.1.1 Módulo com microcontrolador

O módulo de microcontrolador utilizado para processamento da informação do sistema implementado é garantido por um Arduino Nano V3.0 (figura 15). O sistema embutido contém um processador *ATMega* 328 que funciona a uma frequência de 16MHz, contém oito portas de entrada analógicas, catorze portas de entrada/saída digitais e seis de *Pulse-Width Modulation* (PWM), tem uma memória *Static Random Access Memory* (SRAM) de 2 KB e uma memória *Flash* de 32KB. Este módulo de prototipagem eletrónica é amplamente reconhecido pela sua versatilidade na aplicação em tarefas robóticas.

O Arduino proporciona, não só uma plataforma de *Hardware Open-Source* de fácil utilização, como também um ambiente de desenvolvimento para a criação do *Software* de controlo, que pode ser descarregado gratuitamente no *site* oficial. A linguagem de programação utilizada, também esta *Open-Source*, dá pelo nome de

Processing e é baseada nas linguagens *C* e *C++*. A existência de um *bootloader*, traz consigo facilidades acrescidas, dispensando qualquer compilador ou *Hardware* adicional e, consequentemente, o uso de programadores. A Integrated Development Environment (IDE) disponibiliza ainda bibliotecas que simplificam a configuração e comunicação do *Hardware* conectado, permitindo assim o desenvolvimento rápido de aplicações em qualquer área [30], [31].

Figura 15. Arduino Nano ATMega 328 versão 3.0 [32]**.**

3.1.2 Sensores de Temperatura

O sensor de temperatura é um elemento crucial no desempenho, precisão e eficácia do sistema. O estudo e seleção deste dispositivo exigiu especial cuidado, tendo sido considerado três sensores de temperatura do fabricante *Melexis*: MLX90620, MLX90614ESF-DCI e o MLX90614xAA). Sendo baseados no mesmo princípio de medição, cada um deles tem uma particularidade que os torna distintos. No entanto, antes de se prosseguir com a sua apresentação, este documento segue com a apresentação do conceito de *Field Of View* (FOV) dos sensores de temperatura por infravermelho.

O *FOV* ou campo de visão no caso de instrumentos óticos ou sensores, é o ângulo através do qual o sensor é sensível à radiação eletromagnética. Na figura 16 está representado o conceito do FOV. Este depende da lente e/ou do tamanho do detetor utilizado pelo sensor.

Assim, no caso dos sensores de temperatura, se o seu FOV for muito elevado, há o risco de a área de temperatura a medir ser muito grande (a menos que sejam colocados praticamente juntos ao objeto de medição), o que é obviamente problemático no caso de um pneu a girar a grande velocidade.

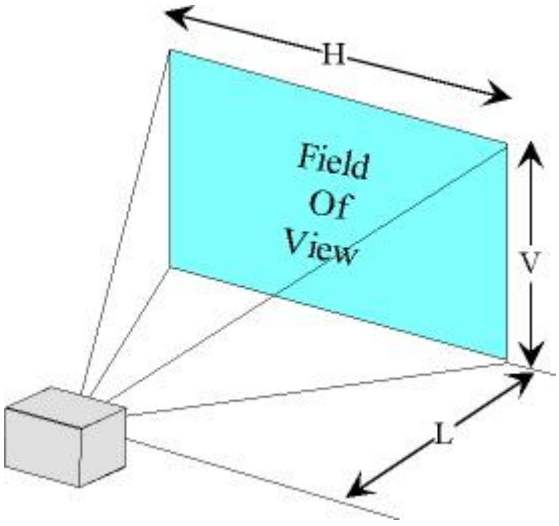

Figura 16. Representação do Field of View [33].

Tendo em consideração o FOV do sensor, a sua montagem deve ser realizada de modo a que variações do curso da suspensão não provoquem variações da distância do sensor ao pneu. Assim, a sua instalação deve ser colocada na parte superior do pneu, tal como representado na figura 17, de modo a que nas diferentes mudanças de direção, os sensores não percam o contacto visual com o pneu.

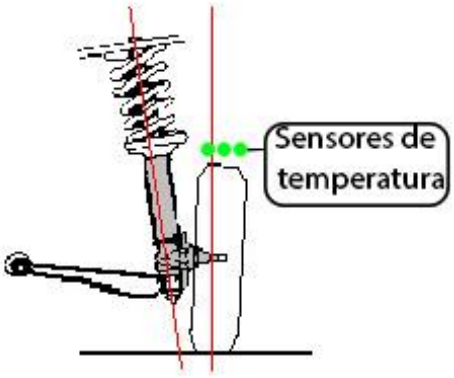

Figura 17. Diagrama de representação da instalação dos sensores de temperatura.

De seguida são então apresentados os sensores referidos anteriormente. A especificidade do MLX90620 é o facto de disponibilizar os valores das temperaturas do pneu sob forma de matriz. O MLX90614ESF-DCI concede um FOV muito reduzido, conseguindo assim uma leitura de uma área mais reduzida, mesmo se colocado a maior distância do objeto cuja temperatura se procura medir. Por fim, foi analisado o MLX90614xAA que, apesar do seu FOV elevado, apresenta um preço bastante

reduzido. Dado que do ponto de vista da interface é igual aos primeiros, permite a implementação de um sistema para prova de conceito a custo reduzido.

Sensor Temperatura Infravermelho Melexis MLX90620

Apresentado na figura 19, este sensor tem como característica particular a captura de temperatura sob forma de matriz, sendo deste modo possível o acesso à temperatura da superfície do objeto através das coordenadas da matriz via protocolo *Inter Integrated Circuit* (I^2C) é um protocolo de comunicação série suportado por todos os microcontroladores atuais). O tamanho da matriz corresponde a 4 linhas por 16 colunas assim como representado na figura 18.

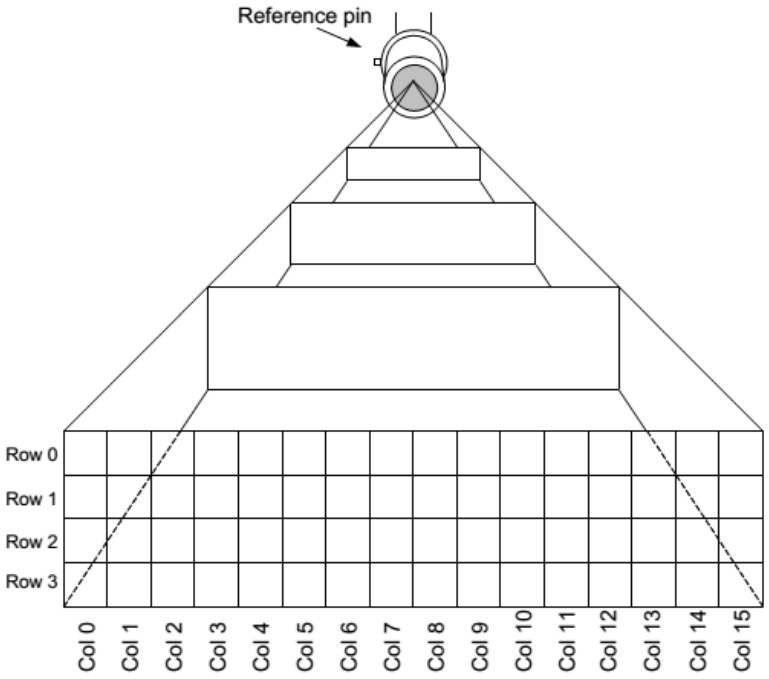

Figura 18. Posição dos pixéis em todo o FOV do sensor [34]**.**

O sensor está normalmente presente em áreas como é o caso da indústria no controlo de peças em movimento, ou no controlo de temperatura de dispositivos domésticos. Apresenta uma grande gama de temperatura que consegue capturar de -50ºC a 300ºC, suportando temperaturas ambiente entre -40ºC a 85ºC. Relativamente à frequência de captura de valores de temperatura possui capacidade entre 0.5Hz a 64Hz. Este sensor está disponível por cerca de 48€ em 100 unidades [35], [36].

Figura 19. Melexis MLX90620 [34]**.**

O sensor possui dois modos de programação do FOV: o modo largo em que o angulo de X é de 60º e o de Y é de 16,4 e o modo médio em que X e Y são representados por 40º e 10.4º respetivamente. A tabela 1 e 2 representam as medidas do tamanho da zona de captura dos valores de temperatura para diferentes distâncias.

Largo

Distância (cm)	**Medida Y(cm)**	**Medida X (cm)**
5	5,77	1,44
10	11,55	2,88
15	17,32	8,83
20	23,09	5,76

Tabela 1. Cálculos do FOV Temperatura para modo largo

Médio

Distância (cm)	Medida Y(cm)	Medida X (cm)
5	3,64	1,44
10	7,28	2,88
15	10,92	8,83
20	14,56	5,76

Tabela 2. Cálculos do FOV Temperatura para modo médio.

Sensor Temperatura Infravermelho Melexis MLX90614ESF-DCI

Estes sensores são capazes de medir uma gama de temperatura maior de temperaturas, entre os -70ºC e 380ºC, enquanto são certificados para eles próprios suportarem temperaturas de -40ºC e 125ºC. Relativamente à frequência de captura de valores de temperatura possui capacidade entre 0.5Hz a 64Hz. Assim, é ampla a diversidade aplicacional do *Melexis* MLX90614ESF-DCI (ver figura 20), como por exemplo, na área da saúde, na qual é utilizado para medição de temperatura corporal, na indústria, onde é aplicado para controlo de temperatura de componentes em movimento, ou na tecnologia automóvel, sendo usado como sensor de conforto térmico para controlo do ar condicionado. A comunicação deste sensor é assegurada também pelo protocolo I2C.

No entanto, a característica deste sensor mais relevante para a aplicação em discussão neste trabalho é o seu reduzido FOV, o que permite uma leitura mais focada da área do objeto, aumentando o rigor das medições efetuadas em muitas aplicações [34], [37].

Figura 20 .Melexis MLX90614ESF-DCI [38].

A tabela 3 apresenta o diâmetro da área de leitura em função da distância sensor-objeto. Observa-se que a variação do diâmetro em função desta distância é menor, quando comparada com outros dispositivos deste género, traduzindo-se assim

num sensor apropriado para aplicações com elevadas exigências de precisão do local a medir. O custo deste, rondando os 50€, torna-o uma opção mais dispendiosa.

MLX90614ESF-DCI FOV 5º	
Distância (mm)	Diâmetro (mm)
50	8,7
100	17,5
150	26,2
200	35,0

Tabela 3. Cálculo FOV para Melexis MLX90614ESF-DCI

Sensor Temperatura Infravermelho Melexis MLX90614xAA

Este sensor (representado na figura 21) também desenvolvido pelo fabricante *Melexis*, é uma solução viável para aplicações correntes, com relativo nível de exigência, aliando uma solução bastante completa a um baixo custo (15€, aproximadamente).

Figura 21. Melexis MLX90614xAA [39].

É o que apresenta dimensões mais reduzidas, logo é mais fácil a aplicação deste dispositivo nos mais variados ambientes. Possui capacidade de ler temperaturas entre os -70℃ e os 380℃ com moderada precisão e suporta ambientes com temperaturas dos -4℃ aos 125℃. Relativamente à frequência de captura de valores de temperatura possui capacidade entre 0.5Hz a 64Hz. O protocolo de comunicação é também o I2C.

A exatidão de leitura varia maioritariamente entre os ±0.5ºC e os ±2ºC de acordo com o gráfico apresentado na Figura 22. A grande limitação deste sensor é mesmo o seu elevado FOV. A tabela 4 demonstra que devido ao FOV de 90º o diâmetro da área de leitura é exatamente o dobro da distância sensor-objeto.

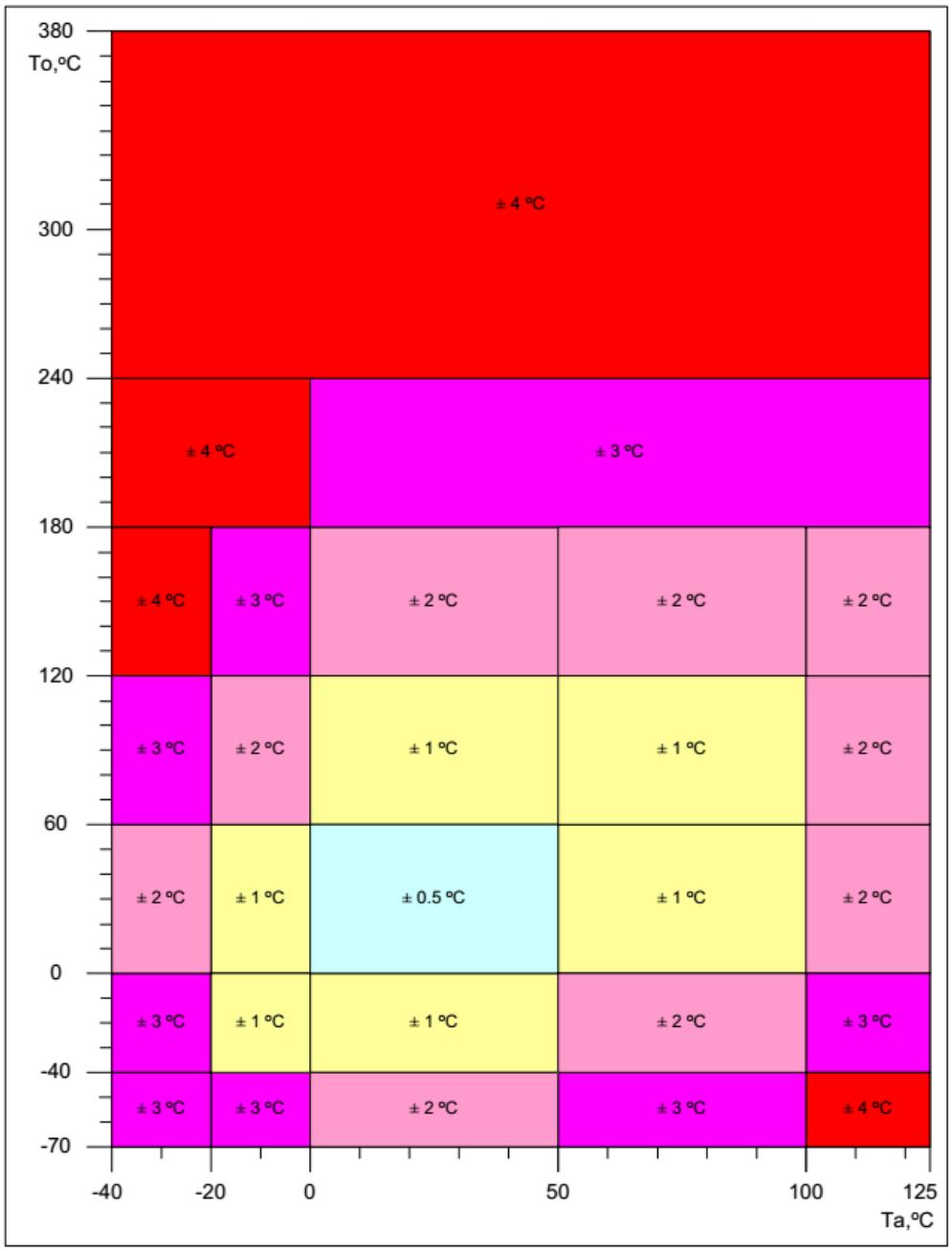

Figura 22. Gráfico precisão do Melexis MLX90614xAA [37].

Distância (mm)	Diâmetro (mm)
50	100,0
100	200,0
150	300,0
200	400,0

Tabela 4. Cálculo FOV para o Melexis MLX90614xAA.

Dentro dos sensores de infravermelhos, seria ainda possível acrescentar um grupo para aplicações de muito elevada temperatura, superiores a 1000 ºC que podem chegar aos 3700 ºC, mas que não apresenta interesse para este trabalho.

Dado o FOV do último, este dificilmente poderá ser utilizado na aplicação em questão, uma vez que implementado a 10 cm do pneu cobre praticamente toda a sua superfície. Assim, de entre os apresentados, a melhor escolha será o *Melexis* MLX90620 devido a sua capacidade de disponibilizar a temperatura do pneu na forma de matriz, e assim, possibilitar mais que uma amostra da mesma zona do pneu, sendo possível obter média de temperaturas das zonas pretendidas e consequentemente obter resultados mais precisos.

No entanto, tendo em conta o baixo preço e o facto de estar facilmente disponível para compra o *MLX90614xAA* acabou por ser a escolha para implementar o protótipo/demonstrador de conceito do projeto. De facto, em todos os outros aspetos este sensor revela ser uma boa opção para o presente projeto: a gama leitura ultrapassa largamente a temperatura máxima atingida por um pneu, tem uma exatidão suficiente para esta aplicação [37], [40] e ainda tem o tamanho mais pequeno.

Acresce que após alguma investigação sobre a parte ótica, foi possível constatar que pela aplicação de uma pequena lente de *Silicon* (disponível a preço reduzido) seria possível resolver o problema do seu FOV elevado. O *Silicone* é completamente opaco à luz visível, mas é transparente a comprimentos de onda onde a maioria da emissão espectral ocorre para temperaturas inferiores a 200º. Com isto é possível aumentar a sensibilidade do sensor ou a distância que o sensor pode medir um objeto de tamanho fixo. A figura 23 representa o conceito apresentado.[41]

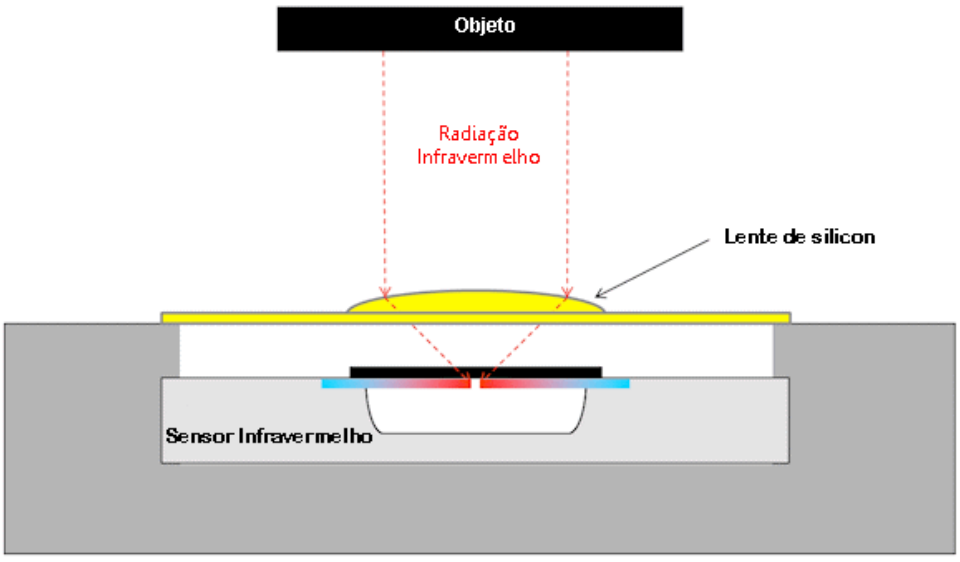

Figura 23. Implementação da lente Silicone sobre o sensor de temperatura infravermelho.

3.1.3 Módulo de comunicação RF

Este sistema de comunicação, que geralmente é utilizado em carros telecomandados ou controlo de aplicações simples, representa uma solução muito económica para estabelecer uma comunicação sem fios. Este componente é constituído por duas partes, transmissor e recetor, ambos representados nas figuras 24 e 25 respetivamente, permitindo operar a frequências de 433MHz ou 315MHz. [42]

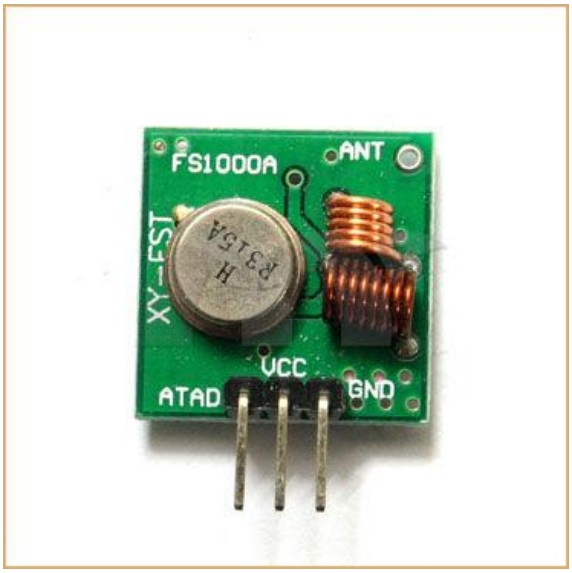

Figura 24. Rf 433MHz emissor [42].

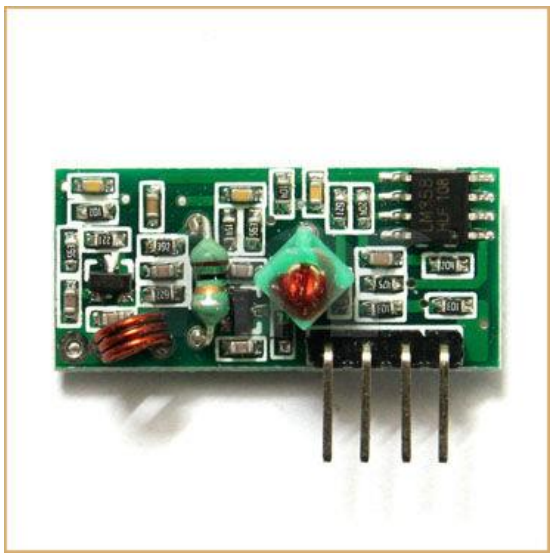

Figura 25. RF 433 MHz recetor [42].

Relativamente ao emissor RF, este funciona com uma tensão entre 3 a 12 V, consumindo uma corrente entre os 9mA e no máximo 40mA. Este módulo funciona até uma velocidade de cerca de 10 Kbps, conseguindo comunicar a uma distância máxima de 90 metros em campo aberto [42].

O recetor funciona a uma tensão de +5.0VDC e consome no máximo 5.5mA. A largura de banda é de 2MHZ e a velocidade de comunicação está adaptada à do emissor. Foi necessário o uso de uma antena a fim de aumentar o alcance da transmissão e assim representar um ambiente mais real [42].

De modo a conseguir aumentar a distância de comunicação entre dispositivos foi criada uma antena como a representada na figura 26 com 16 espiras de 2.5mm de diâmetro em que a extremidade que faz a ligação ao RF tem 17mm e a outra tem 53mm [43].

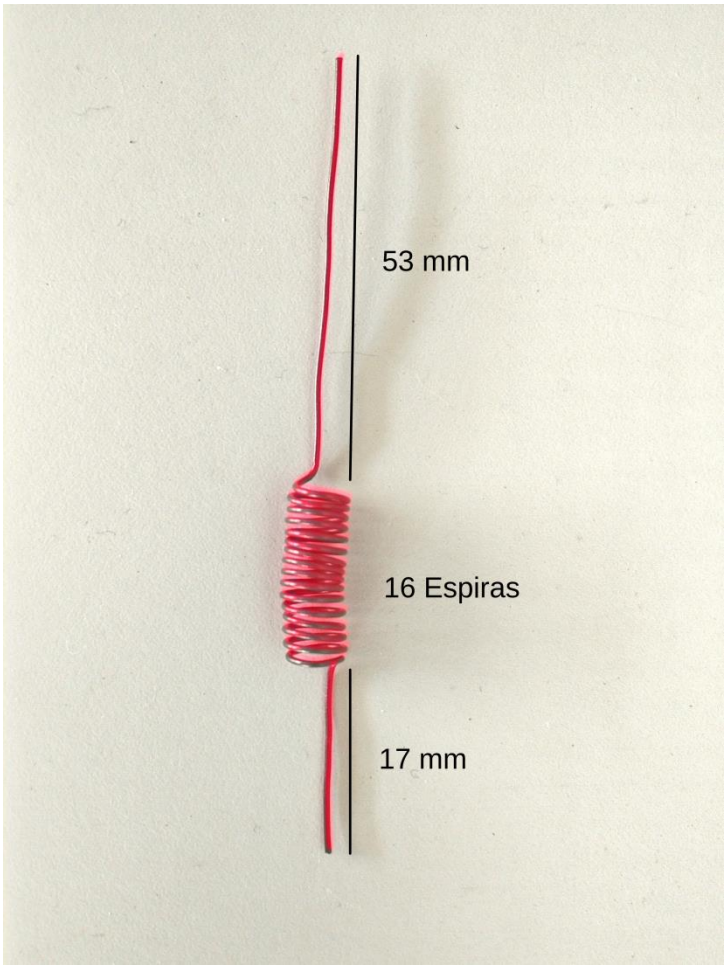

Figura 26. Antena desenvolvida para aumentar o alcance dos RF.

Para iniciar a comunicação entre o microcontrolador e o Transmissor RF a informação é convertida em série para que esta possa ser recebida pelo Transmissor

RF, o qual em seguida aplica a técnica de modulação digital *Amplitude Shift Keying* e transmite a informação pela antena. Importante referir que estes módulos também permitem a modelação *Phase Shift Keying* [44].

Como representado na figura 27, a técnica *Amplitude Shift Keying* representa o sinal logico 1 com uma amplitude e frequência fixa enquanto o sinal logico zero é representado com a onda a 0 [45].

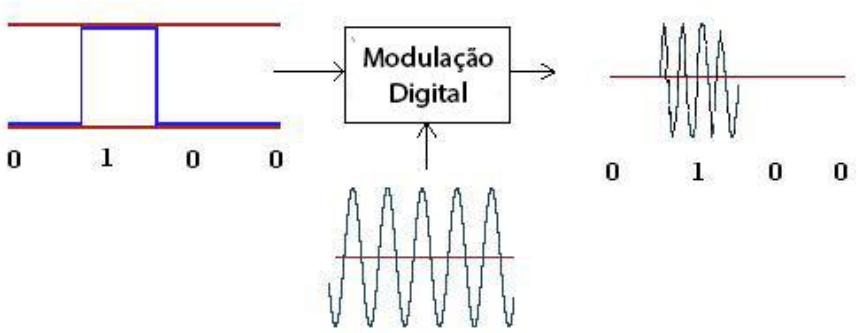

Figura 27. Esquema da técnica de comunicação Amplitude Shift Keying [45].

A técnica de modulação *Frequency Shift Keying* como está representada na figura 28 reproduz o sinal logico 1 com uma frequência muito mais alta que o bit-rate do sinal digital [45].

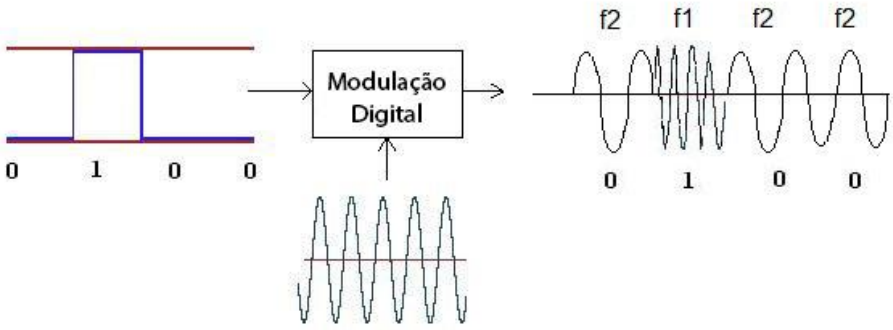

Figura 28. Esquema da técnica de comunicação Frequency Shift Keying [45].

Do lado do recetor a informação é recebida através da antena para posterior desmodulação. Depois desta temos na saída do recetor uma cópia do sinal de entrada (a menos de um atraso).

Com o auxílio do Arduino a informação é convertida de n-bits em serie, na parte do emissor, e de serie em informação de n-bits, do lado do recetor. A figura 29 representa todo este processo de comunicação entre as duas entidades [44], [45].

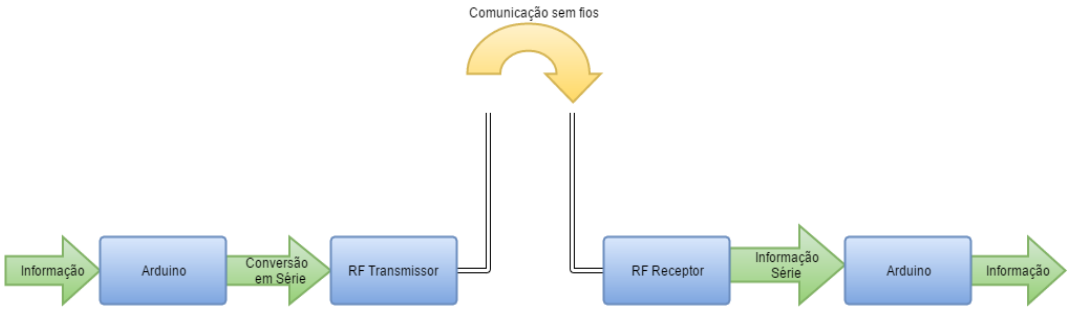

Figura 29. Modo como a informação é enviada e recebida pelos RF.

3.1.4 Circuitos e ligações

O circuito do dispositivo *Master*, o responsável pelo controlo de todas as comunicações do sistema quer na parte do *Sensor Aggregator Gateway* quer no *Processing Unit*, está representado na figura 30 onde é possível verificar a utilização de um Arduino, um recetor e um emissor. As respetivas ligações entre emissor e recetor são feitas utilizado os portos digitais 11 e 12 do Arduino, respetivamente. Esta ligação poderia ser implementada em qualquer um outro porto digital de modo a que permita a implementação do protocolo de comunicação *Universal Asynchronous Receiver/Transmitter* (UART). Este protocolo consiste numa comunicação série assíncrona em que cada dispositivo deve conter o seu *clock*, neste caso o Arduino e cada módulo RF. Neste protocolo, tipicamente, por cada byte transmitido são utilizados mais dois bits (designados por *start* bit e stop bits) o que lhe acrescentam um *overhead* de 20%.

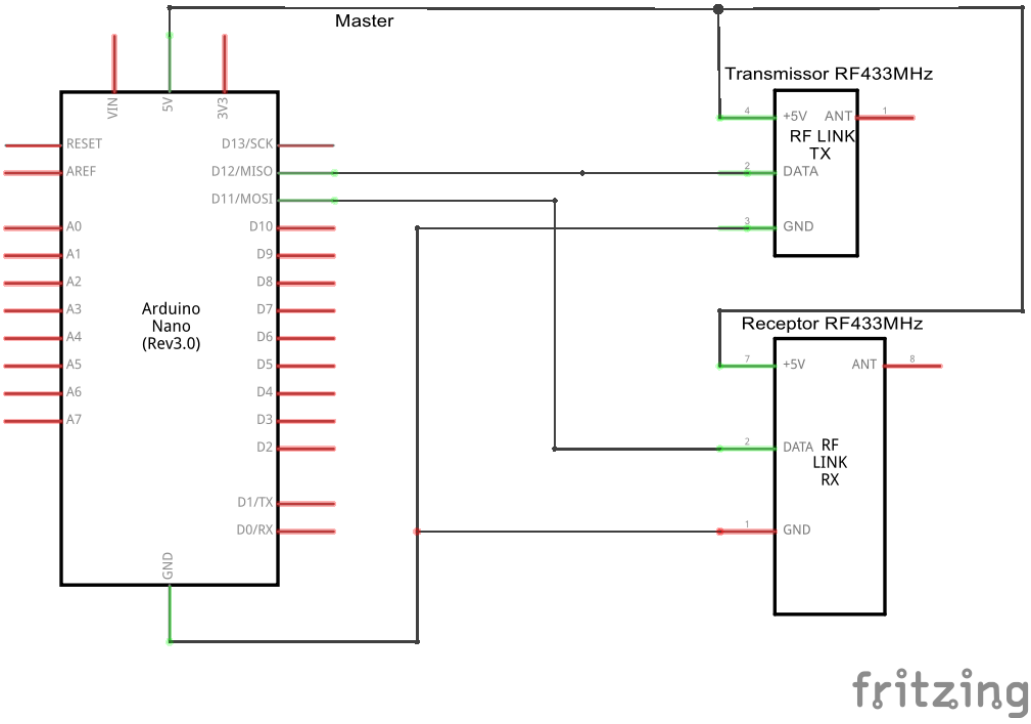

Figura 30. Esquema de ligações entre Arduino e os RF (Master).

Como já referido apesar de ter sido projetada uma solução completa, o protótipo construído apenas considerou um sensor de temperatura numa roda. A implementação correspondente é apresentada na figura 31. Obviamente que a solução em placa branca de prototipagem, é apenas para demonstração funcional em laboratório, e nunca poderá servir para instalação num veículo automóvel.

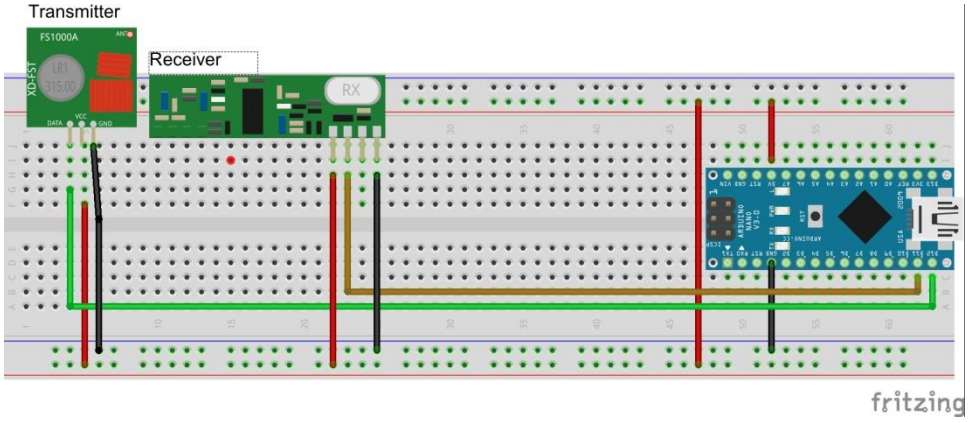

Figura 31. Disposição do Master na placa branca.

A implementação do circuito *Slave* (figura 32) responsável pela captura da temperatura nos pneus do veículo, para além dos RF presentes no dispositivo *Master*, é também utilizado o sensor temperatura infravermelho MLX90614.

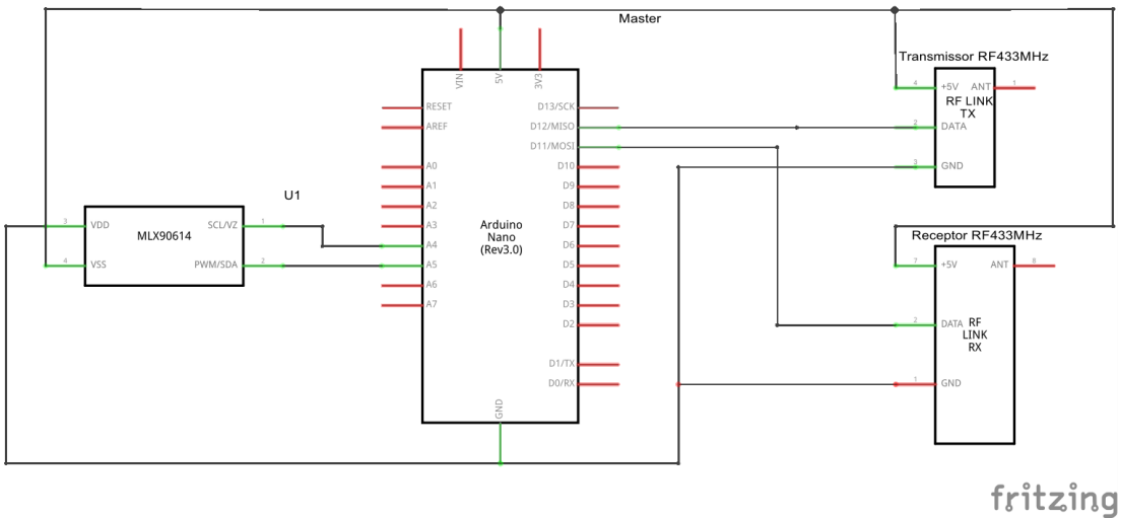

Figura 32. Esquema de ligações entre Arduino os RF e sensor de infravermelho (Slave).

A comunicação com este sensor é feita por meio do protocolo I2C, o que levou à utilização das portas A4 e A5 do Arduino. A porta A4 tem ligação ao porto SDA (serial data) do sensor, sendo por ela que são transmitidos todos os dados. Enquanto a porta A5 faz ligação à porta SCL (*serial clock*), responsável por transmitir o sinal *clock* entre as 2 entidades. A figura 33 representa as ligações descritas.

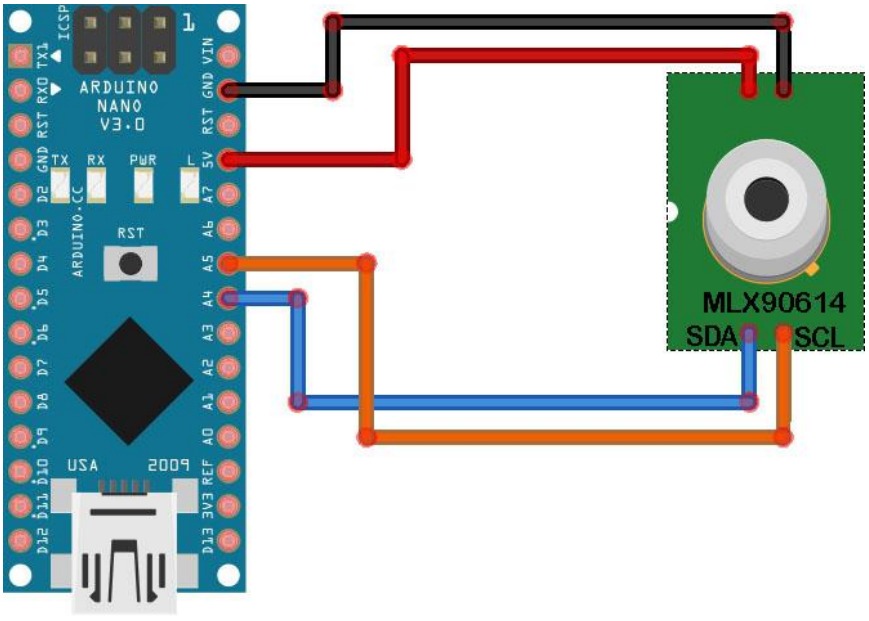

Figura 33. Ligação entre Sensor de temperatura MLX90614 e Arduino Nano.

A figura 34 representa a componente *Slave* e a implementação do circuito na placa branca. Como é possível verificar, ao contrário da componente *Master*, apresenta o sensor de temperatura na sua composição.

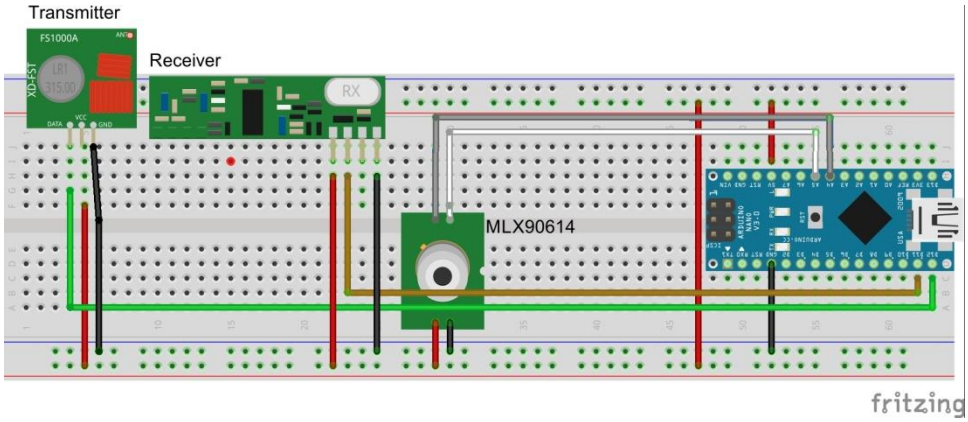

Figura 34. Disposição do Slave na placa branca.

3.2 Sensor Aggregator Gateway Software

Tal como referido anteriormente, todo o projeto do *Sensor Aggregator Gateway* foi desenvolvido sobre a plataforma Arduino, o que possibilitou recorrer à utilização de uma vasta seleção de bibliotecas de funções disponíveis e assim simplificar o desenvolvimento do *firmaware* do projeto, nomeadamente da comunicação entre os diferentes *Slaves* com o *Master* e na leitura da temperatura.

Na comunicação entre os módulos RF foi utilizada a biblioteca *Virtual Wire*, que foi desenvolvida para os RF 433MHz e que por sua vez estão a ser usados neste trabalho. Esta biblioteca foi escolhida porque oferece recursos de troca de mensagens curtas onde serão enviadas as trocas de informação entre dispositivos e continua a ser amplamente utilizada, apresentando uma grande comunidade de apoio [46], [47],[48].

Na comunicação com o MLX90614, a biblioteca utilizada foi a disponibilizada pela Adafruit, que consiste numa empresa de desenvolvimento de *Hardware Open Source fundada* em 2005. Esta biblioteca fornece todas as configurações de comunicação assim como de acesso às capturas de temperatura [49], [50].

Na implementação do código 1 da parte do *Slave* foi necessário importar as duas bibliotecas para se poder funcionar com os RF 433 MHz e com o sensor de infravermelho. Como é possivel verificar no código apresentado, são importadas as bibliotecas Wire.h e a Adafruit_MLX90614.h referentas à utilização dos sensors de infravermelho e a VirtualWire.h é alusiva ao uso dos RF433 MHz.

```
#include <VirtualWire.h>
#include <Adafruit_MLX90614.h>
```

Código 1. Importação das bibliotecas utilizadas pelo Arduino Master do Sensor Aggregator.

Ainda na parte do sensor de infravermelhos, após a importação das bibliotecas é necessário criar uma instância de modo a poder utilizar as funcionalidades da biblioteca assim como a inicialização da configuração do sensor. Depois de implementados os passos anteriores, basta chamar mlx.readObjectTempC() para que seja recebida a temperatura em graus Celsius. O código 2 representa o modo como é obtida a temperatura em que a função Serial.print() é usada para escrever o valor da temperatura na consola.

```
Adafruit_MLX90614 mlx = Adafruit_MLX90614(); // Criação da
instância

void setup()
{
mlx.begin();// inicia a configuração do sensor
}
void loop() {
Serial.print(mlx.readObjectTempC());// escreve a temperatura do
objeto para o qual está direcionado
}}
```

Código 2. Modo como é feita a captura da temperatura.

Relativamente à utilização dos módulos RF433 MHz é necessário inicializar a velocidade de comunicação quer para o recetor como para o emissor. No caso do recetor, depois da inicialização, é necessário chamar a função vw_get_message(message, &messageLength) para que seja carregada a mensagem recebida pelo recetor. Já relativamente ao emissor basta em qualquer altura chamar a função send() com a mensagem que se pretende enviar, que executa o processo de envio da mensagem. O código 3 representa o processo de envio e receção de mensagens.

```
byte message[VW_MAX_MESSAGE_LEN];//buffer para as mensagens recebidas
byte messageLength = VW_MAX_MESSAGE_LEN;//tamanho da mensagem
void setup()
{
vw_rx_start(); // Inicializa o receptor
vw_setup(2000); // Bits por segundo
}
Void loop(){
//Emissor
Send(”mensagem a enviar”);//envia a mensagem
//Receptor
if (vw_get_message(message, &messageLength)) // Verifica se chegou
alguma mensagem sem bloquear o programa
{
Serial.print("Recebido: ");
for (int i = 0; i < messageLength; i++)
{
Serial.write(message[i]); //carrega a mensagem recebidapara o array
mensagem
}
}
}
Void send(char * message)
{
vw_send((uint8_t *)message, strlen(message));//envia a mensagem
vw_wait_tx();//espera que a mensagem seja enviada
}
```

Código 3. Modo de configuração e comunicação dos RF.

Na implementação utilizada no trabalho por parte dos *Slaves*, todos têm um identificador referente ao número da roda a que pertencem, inicialmente todos estão à escuta das mensagens trocadas entre *Master* e *Slaves* só quando recebem o seu identificador por parte do *Master* é que procedem à recolha de informação por parte dos sensores de infravermelhos e a transmitem para o *Master*. A mensagem enviada é formatada com R seguido do numero do pneu a que se refere seguido de “#T1= #T2= #T3” em que seguido do sinal igual é inserido a temperatura de cada sensor. A figura 35 representa como é formatada a mensagem.

Figura 35. Formato da mensagem transmitida do Slave para o Master.

O código 4 ilustra o modo como é recebido o identificador por parte do *Slave* do pneu 1 e feito o envio da informação recolhida.

```
if (vw_get_message(message, &messageLength)) // Verifica se há
mensagem sem bloquear
  {
    for (int i = 0; i < messageLength; i++) //Carrega a mensagem
recebida.
    {
      Serial.write(message[i]);
    }
if (message[0] == '1') //O Slave verifica se o valor que está na
posição 0 da mensagem recebida corresponde ao seu identificador.
    {
double T1=mlx.readObjectTempC(); //carrega a temperatura em graus
Celsius do objecto para a variável T1
String T1ToString= doubleToString(R1,2); // converte em string o valor
da temperatura
String messageToSend = "R1#T1="+T1ToString+"#T2=163#T3=178\n";
//formata a mensagem que se pretende enviar, neste caso apenas 1 o
valor T1 foi lido do sensor Infravermelho
char charBuf[VW_MAX_MESSAGE_LEN];
messageToSend.toCharArray(charBuf,messageToSend.length()+1);//converte
a mensagem em Char
send(charBuf);//envia a mensagem
```

Código 4. Comunicação entre Master e Slave.

O código é muito semelhante em todos os *Slaves* instalados no carro, apenas diferindo no seu identificador estando este associado ao número do pneu como está representado na figura 11. No caso do *Master*, o qual tem a função de controlar toda a comunicação do sistema, o código implementado apenas usa a biblioteca *Virtual Wire* logo todas as configurações são iguais às apresentadas nos *Slaves*.

Como neste trabalho apenas foi implementado um *Slave,* iremos tratar a comunicação entre o *Slave 1 /Master* e *Master/Processing Unit*. Esta comunicação é iniciada com o envio do identificador da roda, como já foi descrito anteriormente, e é aguardado no máximo um segundo pela resposta do *Slave*. Caso não haja qualquer resposta do *Slave* é enviado o identificador do próximo e assim sucessivamente. A comunicação entre *Master* e *Processing Unit* apenas é efetuada quando existe resposta de um *Slave*, na mensagem a enviar apenas é inserido no início da mensagem o identificador “5-#” que representa o identificador do *Processing Unit*. O código 5 representa o algoritmo descrito. Foi também implementado no *Master* um verificador de tamanho de mensagem, que sempre que uma mensagem é recebida proveniente de um *Slave,* a mensagem é descartada caso não contenha o tamanho mínimo de 17 caracteres. Esta implementação garante assim a troca de mensagens inválidas ou incompletas entre o *Sensor Aggregator Gateway* e o *Processing Unit*.

```
send("1");// Envia o identificador ao Slave
  if (vw_wait_rx_max(1000)) //aguarda no máximo um segundo pela
resposta do Slave 1
  {
    if (vw_get_message(message, &messageLength)) // Verifica a
mensagem recebida e o tamanho dela
    {
      for (i = 0; i < messageLength; i++) //carrega a mensagem
recebida
      {
        message[i];
      }
      strMessage= String((char *)message);//converte em string a
mensagem recebida
    }
  }
if(strMessage!="") // se a mensagem recebida não for vazia
    {
strMessage ="5-#"+ strMessage +'\0';
    sendRaspberry(strMessage);//envia para o Processing Unit a
informação recebida pelo Slave
    }
```

Código 5. Processamento da informação por parte do Master.

3.3 Processing Unit Hardware

A *Processing Unit* é a estação-base que permanecerá nas boxes, com a finalidade de receber os dados enviados pelo *Master* do *Sensor Aggregator Gateway*, representar a informação de modo claro e intuitivo e armazenar esses mesmos dados para análise posterior. Nesta fase, as mensagens recebidas via RF 433MHz, com o identificador da *Unit Processing*, são filtradas de modo a capturar apenas o conteúdo relevante, são posteriormente inseridas numa base de dados *Mysql*, de forma a poder ser acedida pelos diversos serviços responsáveis pela representação desta. A figura 36 representa a implementação do *Processing Unit*.

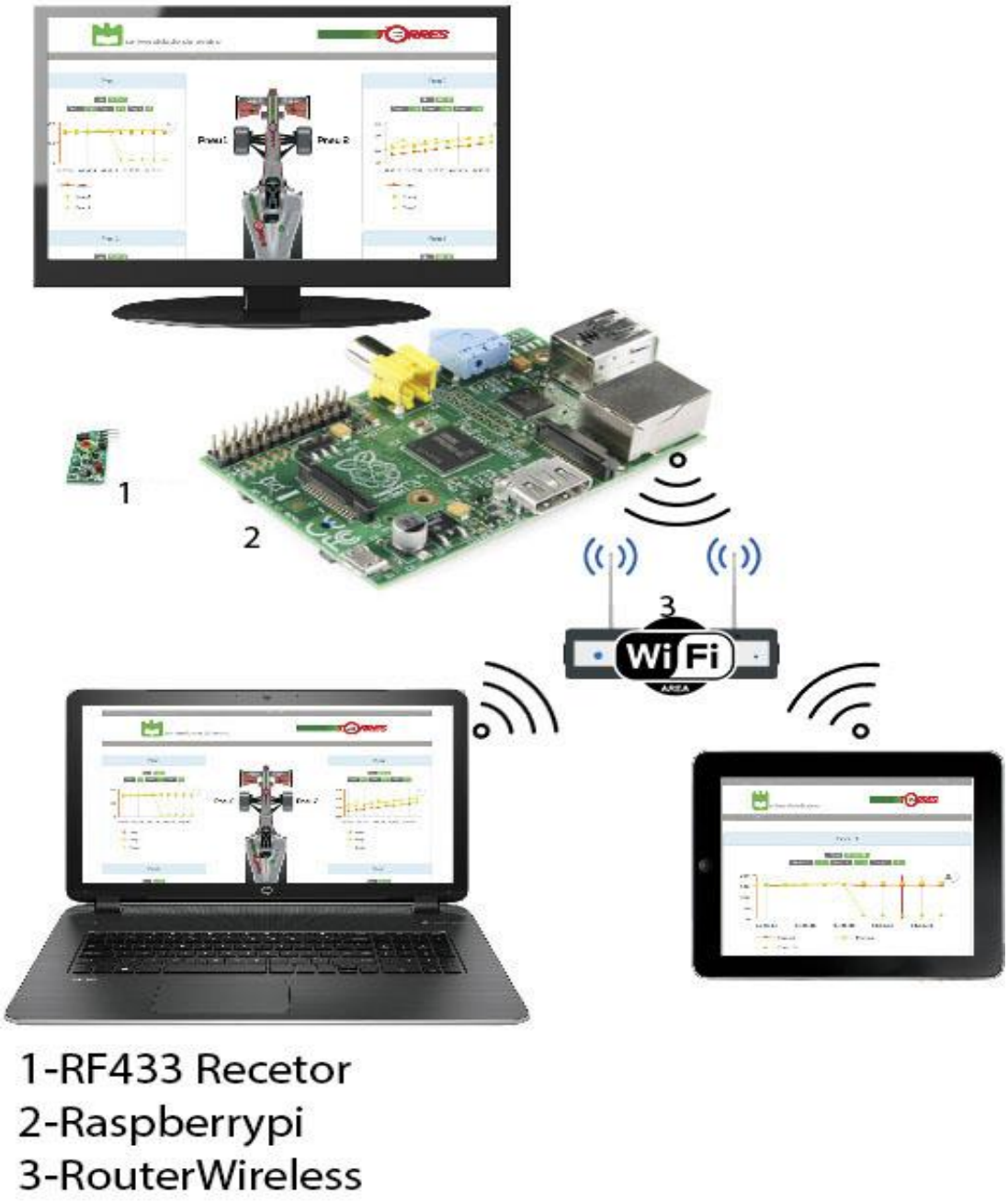

Figura 36. Hardware Processing Unit.

Analogamente ao realizado para o *Sensor Aggregator*, vamos de seguida apresentar em detalhe cada um dos blocos ilustrados na figura34.

3.3.1 Raspberry Pi

Esta plataforma computacional foi desenvolvida pela *Raspberry Pi Foundation*, organização de caridade britânica, com o intuito de motivar a investigação da ciência

da computação. De facto, todos os componentes foram escolhidos de modo a rentabilizar o seu custo. Os sistemas operativos são baseados em *Linux* e a maior parte deles são gratuitos. Na verdade, é bastante popular devido ao seu baixo custo e baixos consumos energéticos o que a torna bastante versátil na implementação de projetos das mais diversas áreas, tais como a monitorização, automação e o *Cloud Computing*.

Relativamente às suas características (ver figura 37), esta plataforma conta com uma saida de video *HDMI*, com resolução de 1920 x 1200 pixels e de outra saída de video *RCA*. Contém duas fichas *USB* para ligação de periféricos, uma saída de audio e uma ligação ethernet *RJ45*. O facto deste dispositivo disponibilizar oito pinos programáveis de General Purpose Input/Output (*GPIO)*, possibilita a conecção dos mais diversos sensores, motores ou relés, para controlo de outros equipamentos [51]–[53].

Foi conectado um RF433 MHz capaz de receber a informação proveniente do Master presente no carro.

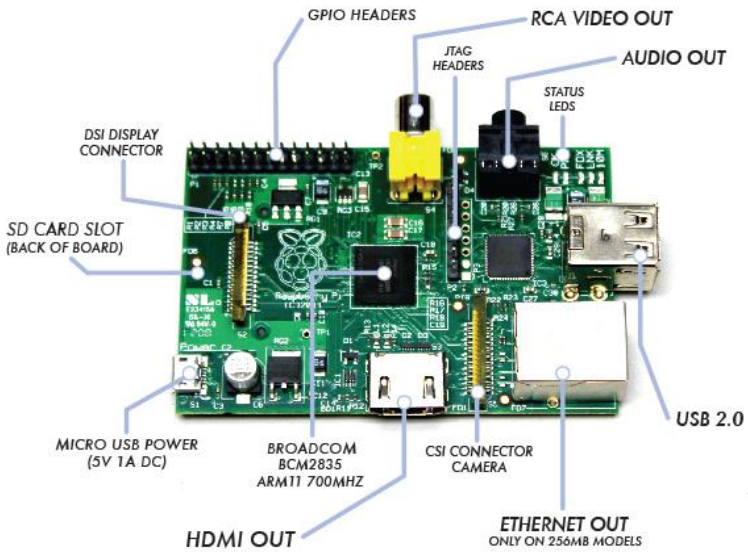

Figura 37.Raspberry PI [53].

3.3.2 Router Wifi

Foi utilizado o OEM3g Router *Wifi* da figura 38 a fim de possibilitar a comunicação entre o *Processing Unit* via ligação *ethernet* e *Wifi* com diferentes terminais para acesso à plataforma. Uma vez que este *router* é 4G possibilita o acesso à plataforma via *Internet*.

Figura 38. Router 3g/4g OEM3g wi-fi [54]**.**

Na figura 39 é possível verificar como foram feitas as ligações do *Rasberry Pi* ao módulo de comunicação RF 433 MHz que apenas recebe informação proveniente do *Master* e dai apenas estar presente o recetor, enquanto que na figura 40 são ilustradas tais ligações.

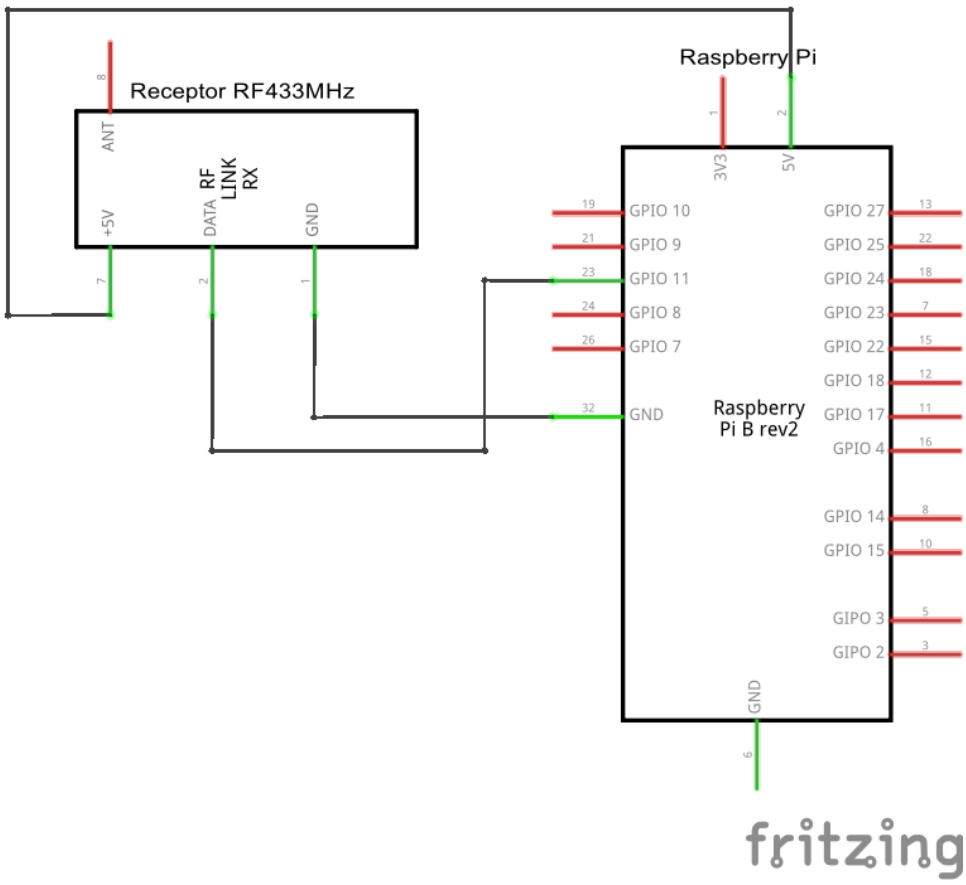

Figura 39. Ligação do Raspberry pi ao RF recetor.

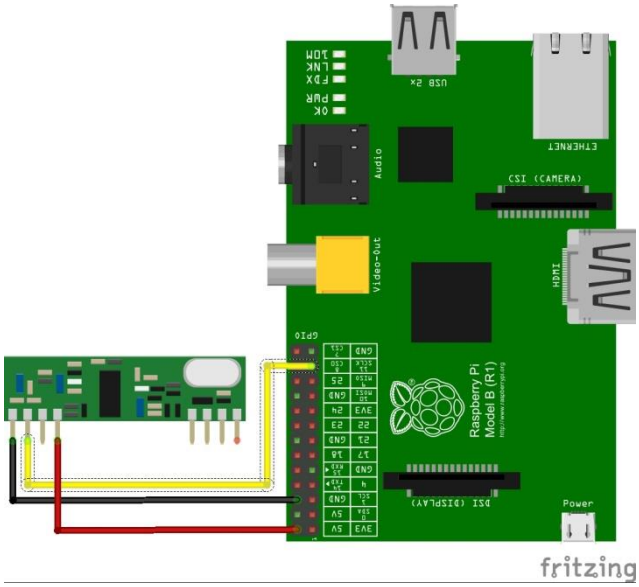

Figura 40. Esquemático da ligação do Raspberry Pi.

3.4 Processing Unit Software

Nesta secção irão ser analisadas detalhadamente todas as tecnologias utilizadas nesta parte do *Software*. De salientar que, toda a informação será armazenada no *Raspberry Pi* assim como todo o acesso aos dados será feito através de uma página *web* ajustável a todos os tamanhos de ecrãs, possibilitando assim o acesso em multiplataforma.

3.4.1 Receção e análise de dados

Esta parte do projeto foi desenvolvida na linguagem de programação *Python*, uma linguagem de programação orientada a objetos de alto nível, não sendo necessário compilar o seu código. É uma linguagem multiplataforma que a permite funcionar em vários sistemas operativos sem a necessidade de alteração do código. É uma linguagem livre, que faz com que seja amplamente utilizada, oferecendo assim uma grande comunidade de apoio [55].

A comunicação do *Raspberry Pi* e o recetor RF433 MHz é feita utilizando a *Pigpio daemon*, que consiste num processo que corre em *backgroud* e permite a comunicação com o *GPIO*. Para ter acesso ao *GPIO*, por parte do *Python* foi utilizada

uma biblioteca *Virtual Wire* compatível com a utilizada no carro, por parte do Arduino, que facilita a configuração e comunicação com os RF 433MHz. De salientar que esta biblioteca é dependente do *Pigpio* daemon e não funciona sem que este esteja em execução [56].

Ainda nesta parte do projeto, foi implementada a comunicação com uma base de dados *MySQL*, sendo esta parte analizadada mais à frente do trabalho. Nesta base de dados estão guardadas todos os dados recebidos do Arduino Master para de seguida ser representada.

Efectivamente, está disponível, também, um sistema de registo em que é possivel verificar os dados que estão a ser recebidos por parte do RF 433MHZ, assim como os dados que estão a ser inseridos na base de dados, para o caso se ocorrer algum problema, seja mais fácil a sua identificação. A figura 41 representa as camadas da aplicação e as diferentes tecnologias utilizadas na sua implementação.

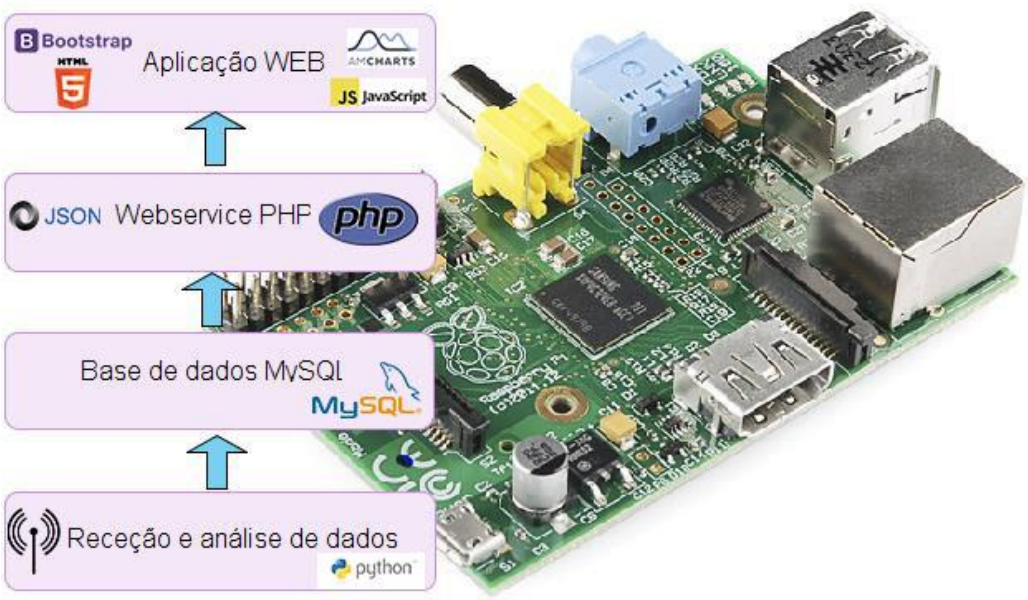

Figura 41. Diferentes camadas de Software presentes no Raspberry Pi.

Na programação desta parte do trabalho foram utilizadas as seguintes bibliotecas :

- *MySQLdb* – Utilizada na comunicação com a Base de dados *MySql*.
- *datetime* – Para obter a data e hora e inserir o registo na base de dados.
- *pigpio* – Biblioteca utilizada para a comunicação com o GPIO.
- Vw – Biblioteca utilizada para a comunicação com os RF 433 MHz.

À semelhança dos *Slaves* instalados no carro, aqui o *Raspberry PI* está sempre à escuta das mensagens trocadas pelo *Master* e apenas actua quando a mensagem

recebida é iniciada com o seu identificador "5-". Assim como no *Master* foi implementado o sistema de controlo de erros de transmissão, caso a mensagem recebida seja inferiror a 17 caracteres a mensagem é descartada. O código 6 representa como é carregada a informação proveniente do RF e respetiva insersão na base de dados.

```
while rx.ready(): //Aguarda Informação

 data ="".join(chr (c) for c in rx.get())// carrega a informação
recebida para a variável data

receiver=data.split("-#") //Criado um array com a informação recebida
dividida por -# em que na posição 0 está o identificador do
destinatário

if a[0] == "5":// caso o identificador recebido for o 5 a informação é
para ser tratada caso contrario é ignorada.

info= receiver [1].split("#") //é criado um array com a informação
separada por #

//O restante código consiste em colocar os valores em variáveis para
posterior inserção na base de dados.
query = "INSERT INTO tblTemperaturaPneus (NrPneu, T1, T2,
T3,DataRegisto,IDCarroTblCarro) VALUES ('%d', '%d', '%d', '%d', '%s'
,'%d')" %
(int(roda),int(temperatura1),int(temperatura2),int(temperatura3),now,1
) //é criada a query para inserir na base de dados
    # db.query(query)//Função que recebe a query e insere na base de
dados
```

Código 6. Carregamento da informação por parte do Processing Unit

3.4.2 Base de dados

Ao projetar a arquitetura da base de dados foi tido em conta a escalabilidade do trabalho, facto que irá possibilitar a implementação, num outro carro, sem que seja perdida a consistência da base de dados.

Na verdade, o Sistema de Gestão e Base de Dados utilizado neste trabalho foi, como já referido, o *MySQL*, um sistema *Open-Source,* que se tornou líder para aplicações *web* [57].

Relativamente à estrutura da base de dados, esta é composta por 2 tabelas, a *tblCarro* e a *tblTemperaturaPneus*. Na primeira, é onde poderão ser inseridos os diversos automóveis que implementam o sistema desenvolvido. Na segunda, é onde serão registados os valores das temperaturas e dos respetivos pneus. A tabela *tblCarro* tem como chave primária o campo *IDCarro* usado como chave estrangeira pela tabela *tblTemperaturaPneus*, fazendo assim a relação de um para muitos entre as

duas tabelas. A figura 42 representa as duas tabelas com os diversos atributos e relações entre elas.

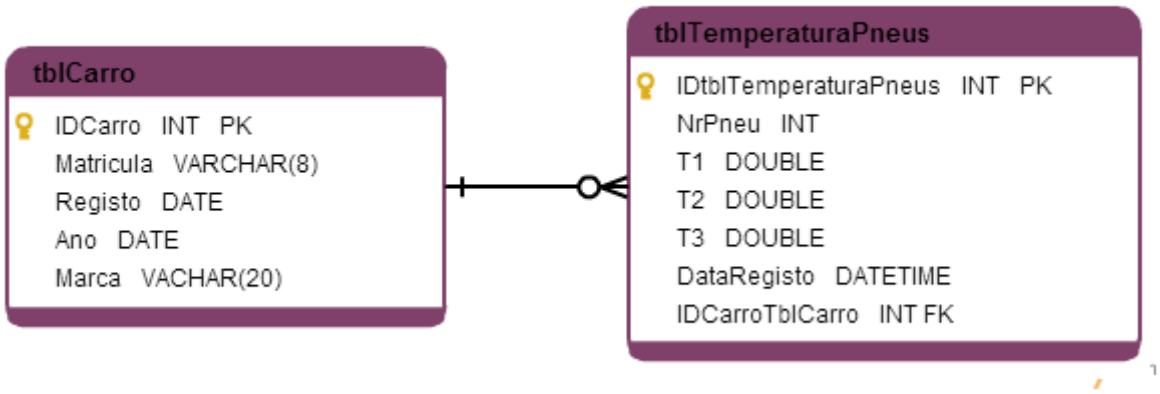

Figura 42.Diagrama base de dados MySQL [58].

3.4.3 Componente aplicacional

Esta componente foi desenvolvida como aplicação *web* com recurso a HyperText Markup Language (*HTML)*, *Java Script* e Cascading Style Sheets (CSS). Foram também utilizadas as *frameworks Bootstrap* e *amCharts* as quais irão ser descritas de seguida.

Bootstrap é uma framework gratuita de HTLM CSS e *JavaScript* mais popular para desenvolvimento *responsive* adaptável a plataformas móveis. Esta plataforma fornece HTML e CSS baseado em modelos de *design* para tipografia, formulários, botões de navegação, entre outros, o que oferece ao programador a capacidade de criar facilmente designs *responsive.* Esta ferramenta foi desenvolvida por Mark Otto e Jacob Thornton no *Twitter*, tendo sido lançada como um produto *Open-Source* em Agosto de 2014 no *GitHub* [59].

A utilização desta ferramenta no projeto permite que qualquer plataforma consiga carregar a plataforma *web* da forma mais versátil, para posterior interação com o utilizador. É importante referir que, dependendo do tamanho do ecrã, a página *web* é carregada de modo diferente, como é possível verificar na figura 43, em que é possível visualizar a página a ser carregada num *Tablet* e num computador.

Figura 43. Modo como é carregada a informação no computador ou em dispositivos móveis.

O *amCharts* é uma *framework* que foi lançada em 2004 inicialmente só com a componente *amMap* desenvolvida por Antanas Marcelionis, é livre sob uma licença *linkware* que consiste em manter os *links* representados sobre qualquer componente desta biblioteca. A figura 35 representa um dos quatro gráficos com a representação da temperatura dos três sensores, o tempo é representado pelo eixo dos XX enquanto o valor da temperatura de cada sensor é representado no eixo dos YY, a linha laranja, amarela e verde representam o sensor 1, 2 e 3 respetivamente [60].

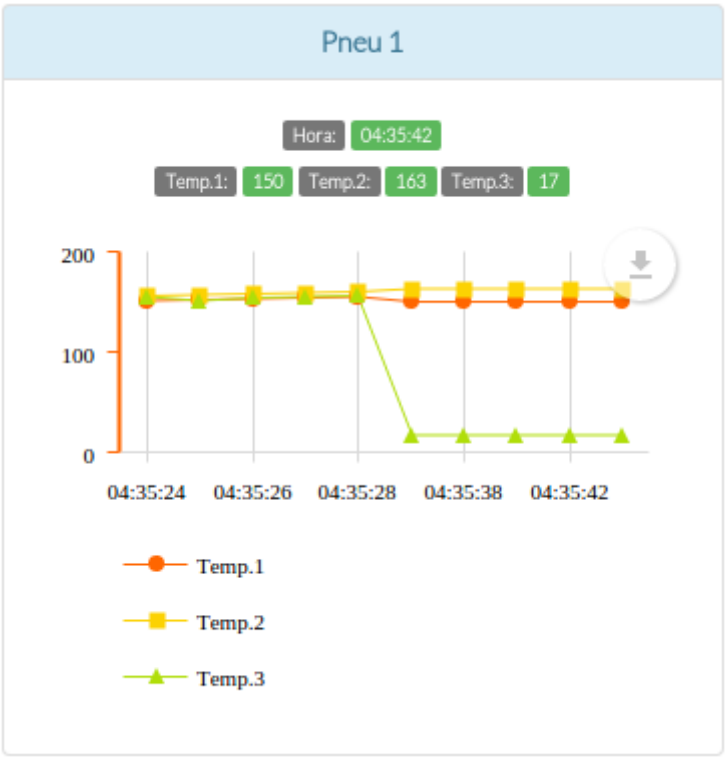

Figura 44. Modo como é representada a temperatura de cada pneu.

Com a aplicação desta *framework* no projeto foi possível representar os gráficos animados consoante a chegada dos valores referentes, possibilitando assim uma melhor compreensão da informação recebida. Esta ferramenta possui também outras funcionalidades como é o caso de possibilitar anotações sobre o próprio gráfico assim como a sua exportação em diversos formatos, como é possível verificar na figura 45. É também dada a possibilidade da seleção das linhas do gráfico que se pretendem visualizar, ocultando assim todas as restantes.

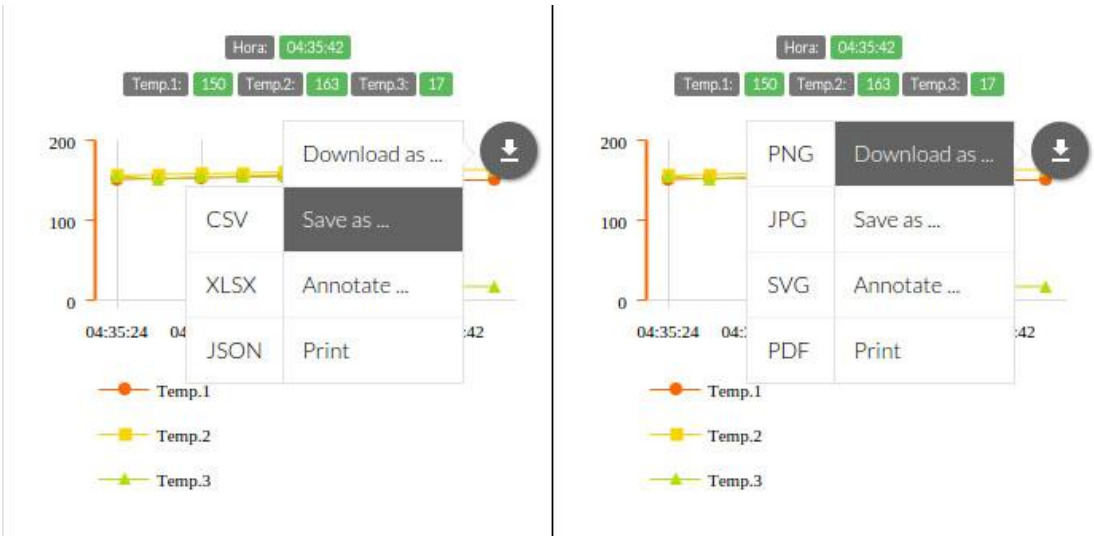

Figura 45. Formatos de exportação dos graficos.

A atualização da informação é feita por uma função em *JavaScript,* a qual atualiza os valores de segundo a segundo; a informação vem sob forma de *JavaScript Object Notation* (JSON) (a qual iremos falar mais a frente), quando o pedido de informação é invocado por *Hypertext Transfer Protocol* (HTTP) em que o endereço é o do próprio *Raspberry* onde está toda a plataforma instalada. O código 7 representa o modo como é carregada a informação referente a cada pneu para as diferentes variáveis.

```
var dataP1  = AmCharts.loadJSON('http:// localhost
/graficoPneu1.php');
var dataP2  = AmCharts.loadJSON('http:// localhost
/graficoPneu2.php');
var dataP3  = AmCharts.loadJSON('http:// localhost
/graficoPneu3.php');
var dataP4  = AmCharts.loadJSON('http:// localhost
/graficoPneu4.php');
```

Código 7.Modo de carregamento da informação referente a cada pneu no JavaScript da página web

Depois de carregada a informação dos diferentes pneus, a informação é processada e representada pelas funções da biblioteca *amChart* assim como é representado o último registo referente a cada pneu. Como auxílio destas ferramentas foram implementados quatro *webservices* que enviam a informação relativa aos últimos dez registos das temperaturas das 4 rodas do carro. Esta informação vem sob forma de *JSON* que é um formato de troca de dados leve, fácil de ler e de escrever. É baseado num subconjunto da linguagem de programação *JavaScript* mas completamente independente de qualquer linguagem [61].

Esta comunicação sob a forma de *JSON* é desenvolvida em Personal Home Page (*PHP)*, são estes os responsáveis por selecionar a informação na base de dados *MySQL*, e a formatar sob a forma de *JSON* para que esta seja carregada e representada pela *framework amChart,* como foi descrito anteriormente. As operações efetuadas sobre a base de dados são as representadas no código 8:

```
<?php
$link = mysql_connect( 'localhost','username', 'password' ); //cria a
ligação a base de dados usando o endereço, username e password de
acesso à mesma
$db = mysql_select_db( 'RaceCarDataBase', $link );//seleciona a base
de dados que pretende aceder com a respetiva ligação criada
anteriormente
$query = "SELECT * from (SELECT T1,T2,T3,DataRegisto FROM
tblTemperaturaPneus where NrPneu=1 ORDER BY IDtblTemperaturaPneus DESC
LIMIT 10 ) as t order by DataRegisto asc";//é criada a query a
executar de modo a obter os últimos dez registos do pneu 1

$result = mysql_query( $query );//é executada a query criada sendo os
dados recebidos pela variável result
?>
```

Código 8. Modo como é carregada a informação da base de dados pelo serviço do PHP.

Depois de recebidos os dados na variável "*result"*, são escritos sob o formato *JSON* que consiste num formato leve para troca de dados entre sistemas computacionais, para serem recebidos pelo solicitador da informação. O código 9 representa a forma como é formatada a informação recebida da base de dados.

```php
<?php
header('Content-Type: application/JSON');//cabeçalho JSON
$prefix = '';
echo "[\n";
while ( $row = mysql_fetch_assoc( $result ) ) {//percorre linha a
linha o resultado obtido na query e formata os resultados em JSON
  echo $prefix . " {\n";
  echo '  "T1": ' . $row['T1'] . ',' . "\n";
  echo '  "T2": ' . $row['T2'] . ',' . "\n";
  echo '  "T3": ' . $row['T3'] . ','. "\n";
  echo '  "DataRegisto": ' .'"'. date("H:i:s",
strtotime($row['DataRegisto'])) . '"' . "\n";
  echo " }";
  $prefix = ",\n";
}
echo "\n]";
?>
```

Código 9. Formatação da informação em JSON.

4. Resultados

Neste capítulo serão apresentados os testes e experiências realizadas com o sistema. Um dos testes consiste na captura do tempo de cinco mensagens desde a solicitação do Master até à sua inserção na base de dados, que é quando estes ficam disponíveis para posterior acesso. Foram também realizados testes de trocas de 1000 mensagens a fim de se conseguir calcular o tempo médio de envio e receção de cada mensagem. Por fim foram feitos testes complementares ao sensor de temperatura utilizado no trabalho de modo a conseguir perceber a sua fiabilidade.

Na realização dos testes feitos foram utilizadas ferramentas como o monitor serie da IDE do Arduino, onde é representado o registo das operações efetuadas pelo código presente em cada Arduino. Também foi utilizado o Secure Shell (SSH), que nos permitiu o acesso remoto ao *Porcessing Unit* e, assim, também permitiu o acesso ao registo das operações executadas pelo código aqui presente. Por fim, foi utilizado o *Php MyAdmin*, que é uma aplicação *web* para gestão de base de dados, que nos permite aceder a todos os registos presentes nesta base de dados.

4.1 Captura tempos de troca de mensagens.

Ao realizar este teste foram realizadas varias análises relativamente aos resultados de certas operações no que diz respeito aos tempos de execução das seguintes operações:

- Tempo de envio e receção da seleção do *Slave* pelo Master
- Tempo de leitura do sensor de temperatura (*MLX90614*)
- Tempo de envio da informação do *Slave* para o *Master*
- Tempo de envio da informação do *Master* para o Processing Unit
- Tempo de inserção da informação na base de dados

Todas estas operações foram analisadas recorrendo à captura de cinco mensagens numeradas correspondendo a uma amostra de temperatura respetivamente. Estas mensagens descrevem todo o processo de comunicação desde a recolha de informação até à inserção na base de dados da mesma. Os testes foram executados apenas para o caso de um pneu e de um sensor de temperatura, apesar de toda a mensagem ser formatada de modo a conter a informação associada aos 3 sensores e assim apresentar o tamanho real. A captura dos tempos foram feitos da

parte do Sensor Aggregator Gateway com o uso da função millis() que retorna o número de milissegundos desde que o Arduino começou a correr o programa. Foram capturados os valores dessa função antes da execução da operação que se pretende monitorizar e, logo após essa operação é feita a diferença entre o valor atual da referida função e o valor capturado antes da execução da mesma - assim é possível obter em milissegundos o tempo que cada operação demorou a executar. Sempre que o *Master* executa todas as operações desde a seleção do *Slave*, receber as diferentes temperaturas e envia para o Processing Unit, neste caso em que apenas temos o cenário de um pneu para monitorizar é de imediato repetido o ciclo referido. Na aplicação real cada *Slave* só irá voltar a comunicar depois de todos os outros terem sido pelo menos selecionados.

A figura 46 foi capturada do monitor série da IDE do Arduino Master em que são apresentados os tempos de execução das diferentes instruções, onde é possível verificar que a seleção do *Slave* por parte do *Master* varia entre 48 e 50 milissegundos, o tempo de receção da informação por parte do *Slave* varia entre 252 e 253 milissegundos e o tempo de envio da mensagem para o Processing Unit varia entre 263 a 265 milissegundos. Foi também capturado o tempo de execução de todas as operações do *Master* que também varia entre 612 a 614 milissegundos.

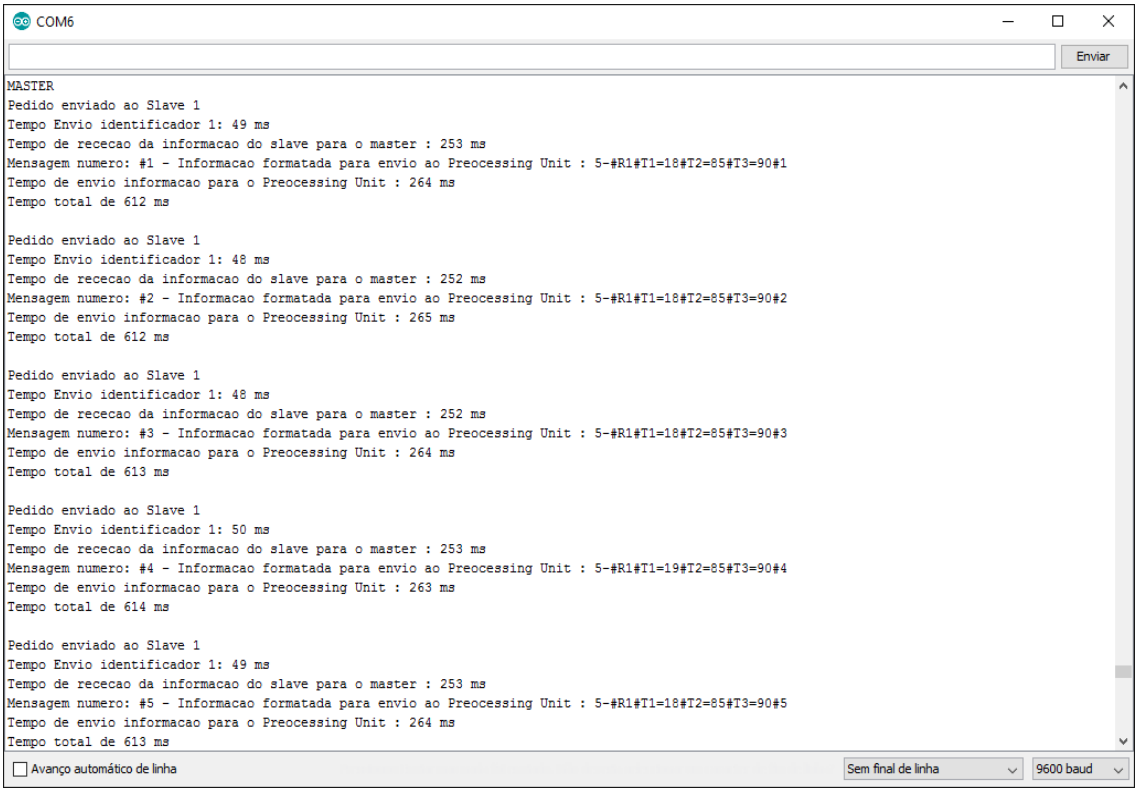

Figura 46. Troca de mensagens por parte do Master.

Na parte do *Slave,* é possível ver na figura 47 que os tempos de leitura da temperatura apresenta um valor constante nas cinco mensagens de 9 milissegundos e que o tempo de execução de todas as operações por parte do *Slave* varia entre 256 e 257 milissegundos. É importante notar que o tempo de execução que recebe a informação é sempre menor que o tempo da função que envia e assim haver uma diferença de alguns milissegundos na receção da informação no *Master* em relação ao tempo de execução de todas as operações por parte do *Slave*. Como é possível também verificar na mesma imagem, o *Slave* recebe todos os identificadores que o *Master* envia e apenas responde quando o identificador corresponde ao seu.

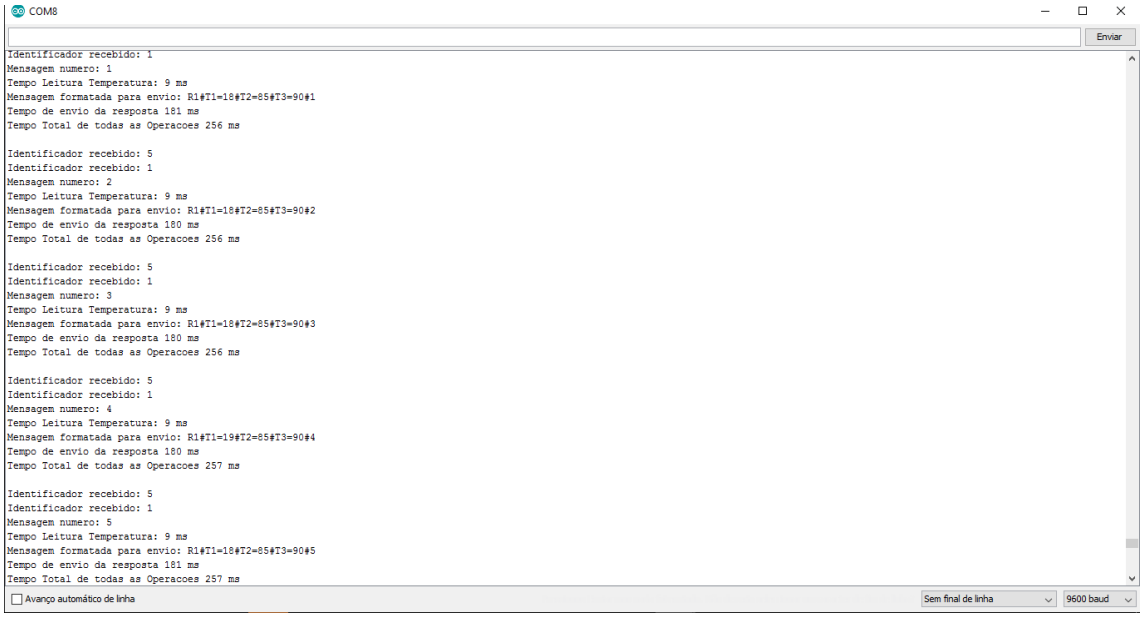

Figura 47. Troca de mensagens por parte do Slave.

Na receção das mensagens por parte do Processing Unit, é feito o tratamento da informação para que seja inserida na base de dados e assim possibilitar o seu acesso. É possível verificar que após a receção da informação, todas as operações executadas têm um tempo a variar entre 14.47 e os 22.41 microssegundos como prova a figura 48.

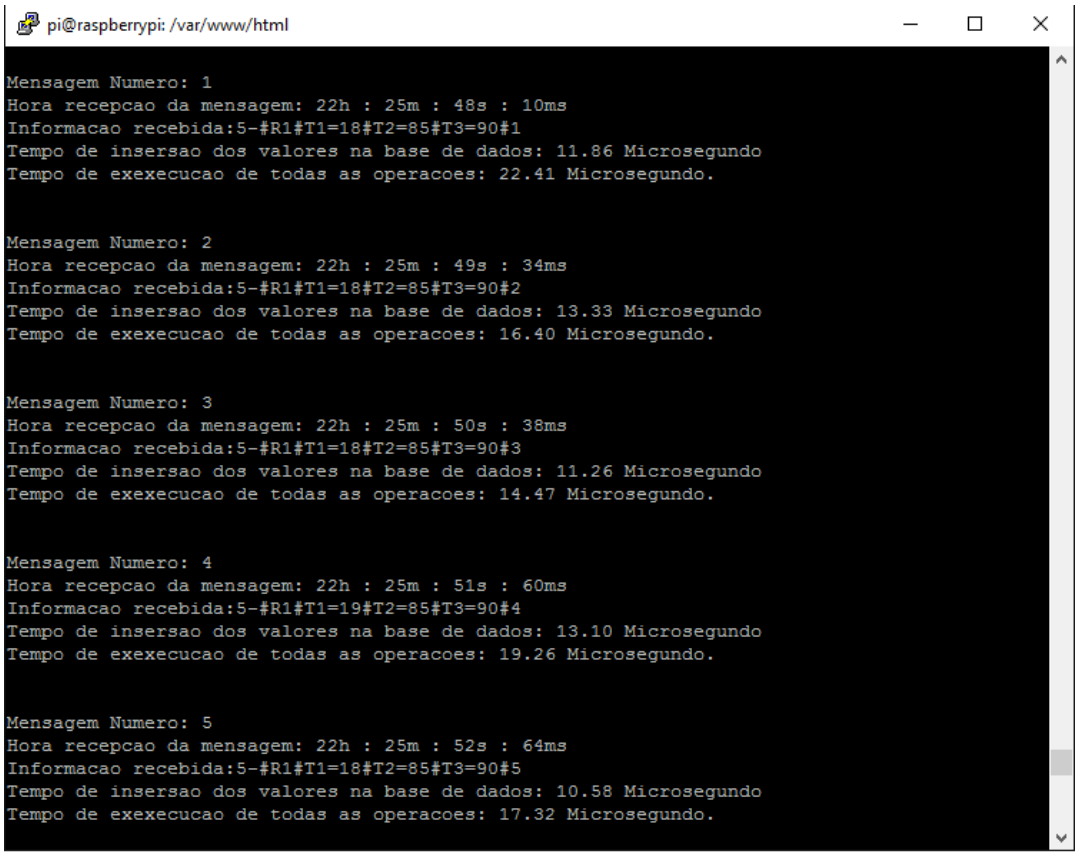

Figura 48. Receção das mensagens por parte do Processing Unit.

A Figura 49 destaca as cinco mensagens inseridas na base de dados através do Php MyAdmin em que estas estão destacadas no retângulo vermelho. Depois de esta informação ser inserida na base de dados é desde logo possível o seu acesso para posterior análise por parte da equipa técnica da Torres Rally Team.

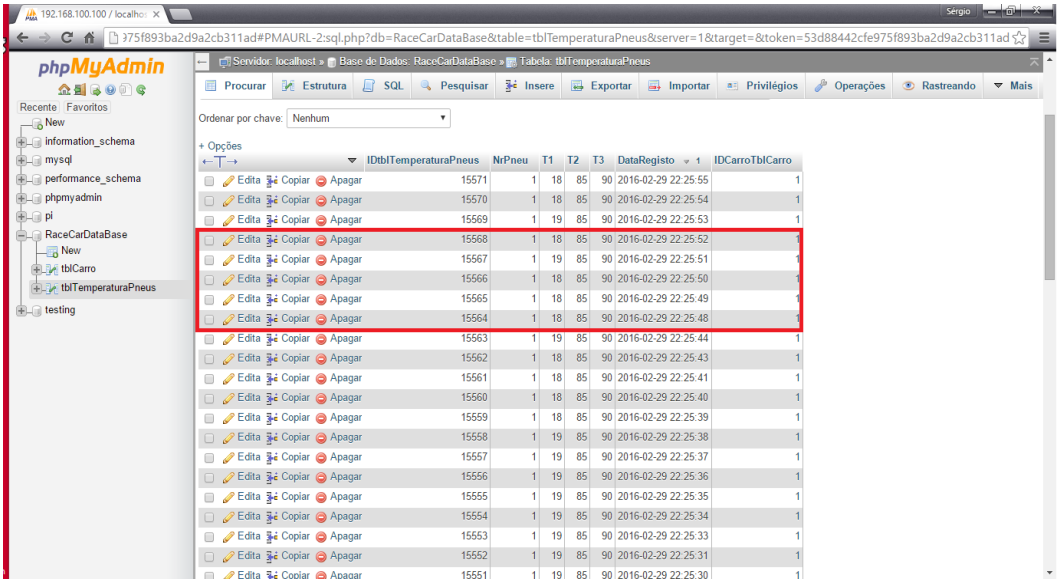

Figura 49. Listagem da informação inserida na base de dados por parte do PHPMyAdmin.

4.2 Média do tempo de troca de mensagens.

Nesta parte também foram realizados testes para calcular a média de tempo para 1000 mensagens, podendo assim ter uma aproximação realista dos tempos obtidos. As seguintes imagens irão descrever a mesma comunicação realizada anteriormente, mas com os respetivos tempos médios de cada um dos componentes.

Como é possível verificar na figura 50 o tempo médio de envio de um identificador é de 48 milissegundos enquanto o tempo que o *Slave* demora a recolher e processar toda a informação é de 183 milissegundos, o envio desta informação para o Processing Unit é de 192 milissegundos.

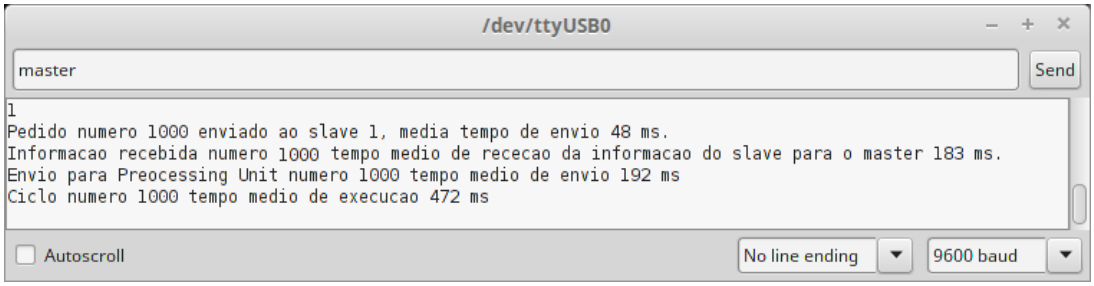

Figura 50. Captura dos tempos médios do Master.

O *Slave* demora em média 1 milissegundo a capturar a temperatura e 185 milissegundos a executar a função de envio de informação para o *Master.* Todas as operações do ciclo demoram em média 188 milissegundos como demonstra a figura 51.

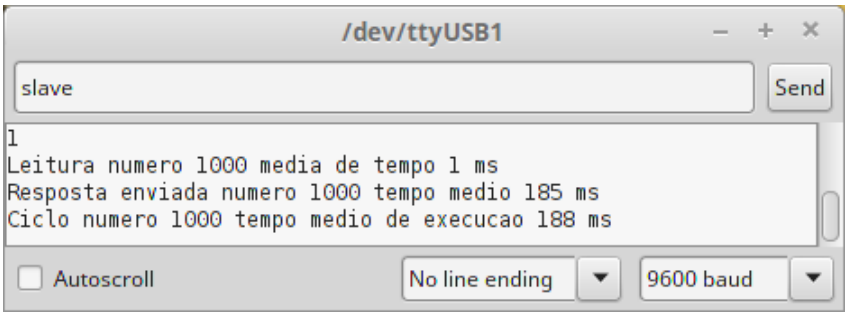

Figura 51. Captura dos tempos médios do Slave.

A Processing Unit para as 1000 mensagens demora em média 190.63 microssegundos a executar todas as operações, sendo 183.21 microssegundos o

tempo de inserção na base de dados da informação recebida, como é possível verificar na figura 52. É importante mencionar que depois da informação estar inserida na base de dados a mesma pode ser acedida a qualquer momento.

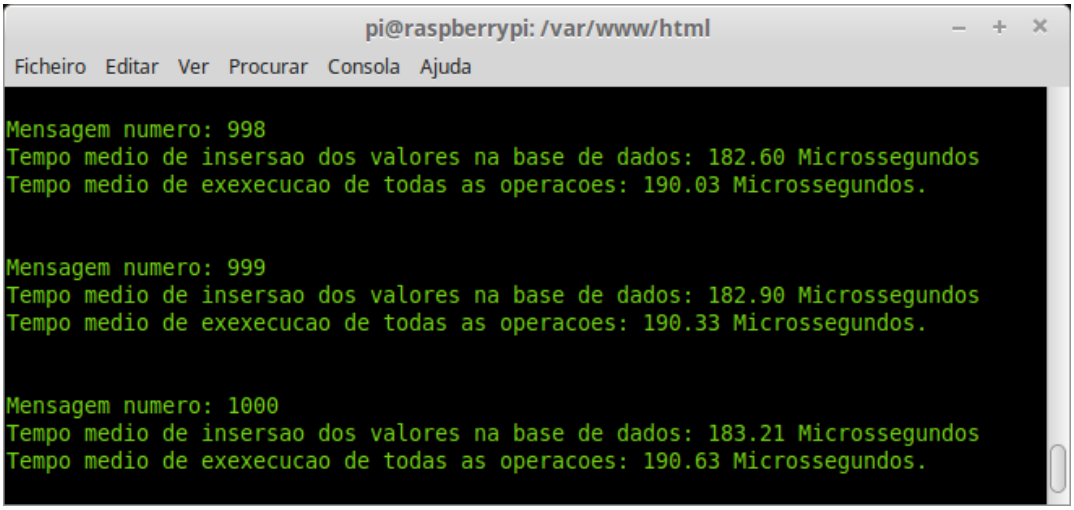

Figura 52. Captura dos tempos médios do Processing Unit

4.3 Testes complementares.

Foram também realizados testes sobre o sensor de temperatura utilizado neste projeto, um deles em que foi possível comprovar que ao passar uma camada de ar quente entre o objeto e o sensor não é afetada a leitura da temperatura. A figura 53 representa o teste feito em que para projetar o ar quente foi utilizado um termo ventilador, foi feita a captura da temperatura do objeto antes de ser submetido a este teste e o resultado final foi que em ambos os casos a temperatura era exatamente igual 20.3º.

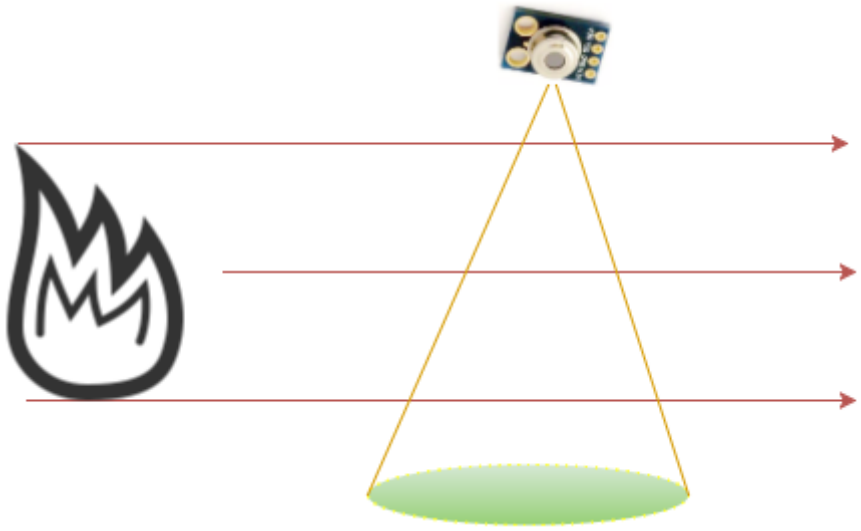

Figura 53. Captura de temperatura com a projeção de ar quente entre o sensor e o objeto.

Um outro teste realizado foi o aquecimento de uma placa metálica com tamanho de 30cm x 28 cm, em que foi utilizada uma pistola de ar quente para direcionar o calor para o centro da placa e assim realizar as capturas. A figura 54 representa a imagem capturada pela câmara térmica sobre a placa em que o centro apresenta a temperatura mais elevada da placa de 37.4º. Foi de imediato feita a medição da temperatura com o sensor utilizado no projeto a uma distância de 5 cm em que o mesmo mediu 36.7º, importante referir que esta temperatura capturada corresponde a uma média feita numa circunferência com diâmetro de 10cm.

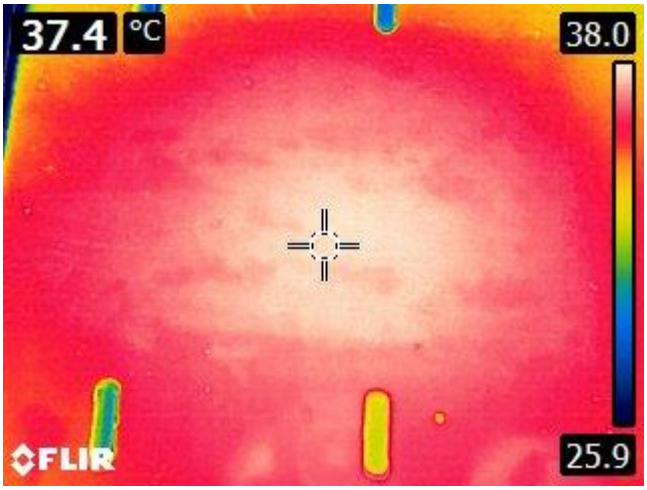

Figura 54. Imagem capturada por uma câmara térmica FLIR TG165.

Os dispositivos RF433 MHz utilizados no desenvolvimento do projeto manifestaram inúmeros problemas de comunicação, o que dificultou esta parte do trabalho. Na realização dos testes os RF teriam que estar imóveis e rigorosamente sintonizados a uma distância máxima de dois metros para que fosse possível a comunicação entre os diversos dispositivos.

5. Conclusão

No meu percurso formativo tive oportunidade de contactar e aprender várias linguagens de programação, algumas de baixo nível em que a linguagem se aproxima da linguagem máquina, outras muito próximas da linguagem humana, denominadas por linguagens de alto nível. Estive envolvido em projetos de diferentes áreas, desde programação à gestão, permitindo-me desenvolver capacidades que vão além da lógica habitual de um programador.

O começo deste projeto, que envolveu áreas totalmente novas para mim e com tecnologias que nunca tinha utilizado anteriormente, constituiu um desafio que colocou à prova as bases adquiridas ao longo do meu percurso académico e que se mostraram uma mais-valia na rápida adaptação a um contexto tecnológico diferente. Neste projeto passei por várias etapas desde o início de construção de simples tabelas de base de dados, programação dos microcontroladores até ao desenvolvimento da plataforma web. Concluída a parte funcional do projeto, foi necessária a revisão de todas as especificações tendo em vista a otimização da performance do sistema. Em determinadas situações foi necessário reescrever o código por completo, garantindo dessa forma a rapidez na execução das tarefas desenhadas e programadas. Concluí que por vezes é necessário rever e refazer o trabalho sempre com sentido crítico por forma a encontrar as melhores soluções. O trabalho de desenvolvido mostrou-se útil, também, para perceber a grande importância do planeamento no desenvolvimento de uma aplicação desta dimensão. As tecnologias inicialmente escolhidas permitiram provar a estabilidade e os métodos de implementação. Estas tecnologias deram a conhecer as melhores ferramentas para a construção de uma página *web* suportada por uma programação simples e flexível. A documentação disponibilizada teve um papel essencial na construção de um portal relativamente complexo. A maioria das bibliotecas era ainda acompanhada por uma boa documentação e de muitas fontes de informação alternativas.

Foram várias as adversidades paralelas que condicionaram o desenvolvimento do projeto. A mais crítica está relacionada com a comunicação entre os diversos microcontroladores. Os dispositivos RF433 MHz manifestam inúmeros problemas de comunicação. Na realização dos testes os RF teriam que estar imóveis e rigorosamente sintonizados para que fosse possível a comunicação entre os diferentes dispositivos.

O estudo da comunicação entre o *Sensor Aggregator Gateway* e a *Processing Unit,* permitiu concluir que, apesar dos problemas descritos, o acesso à informação apresenta tempos bastante satisfatórios e uma rápida compreensão do utilizador sobre a informação apresentada. Tal facto prova que o portal é uma ferramenta eficiente, prática e avançada que permite efetuar análises em tempo reduzido possibilitando uma atuação imediata por parte da equipa da *Torres Rally Team*.

A motivação principal no desenvolvimento desta tese prendeu-se com o desafio de construir uma ferramenta simples e fácil de utilizar que assistisse à afinação de um automóvel de competição através da medição da temperatura dos pneus. Foi igualmente motivador construir uma plataforma que permite poupar recursos no que diz respeito aos equipamentos utilizados, uma vez que qualquer terminal, seja este um Tablet, computador pessoal ou telemóvel consegue aceder à informação. Esta funcionalidade aumenta consideravelmente a versatilidade e portabilidade da ferramenta, permitindo a sua fácil instalação e acompanhamento da equipa nas diversas deslocações.

A respeito dos tempos de comunicação entre os diversos componentes e a partir dos testes realizados é possível que uma captura de temperatura desde que é solicitada pelo *Master* até que é inserida na base de dados demora em média ≈423 milissegundos sendo 48 milissegundos de seleção do *Slave* mais 183 milissegundos até receber a informação por parte do *Slave* mais 192 milissegundos de envio da informação para *Processing Unit* mais 190.63 microssegundos que é o tempo de execução de todas as instruções desde a receção da informação até à sua inserção na base de dados, a figura 55 representa todas estas operações com os respetivos tempos associados. Uma vez implementado este projeto com os 4 pneus e os 3 sensores é possível prever que estes tempos aumentem em média mais 2 milissegundos até o *Master* obter resposta por parte do *Slave,* pois cada sensor de temperatura demora em média 1 milissegundo a retornar o valor da temperatura.

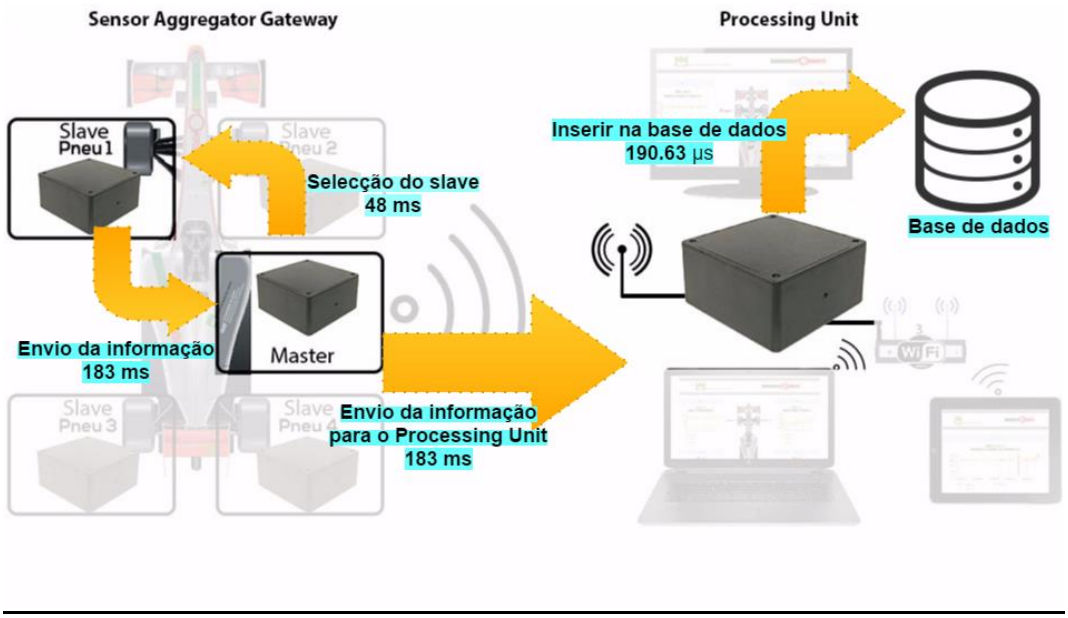

Figura 55. Diagrama dos tempos de cada operação.

5.1 Trabalho futuro

O trabalho futuro deverá passar por novas abordagens para facilitar a implementação física do *Sensor Aggregator Gateway* no automóvel, o que entre outras, implica o desenvolvimento de uma Printed Circuit Board (PCB) que contenha os vários componentes de modo a ser possível a sua integração numa caixa capaz de os proteger do ambiente hostil de um carro de competições.

Uma outra implementação possível seria a possibilidade de adicionar um módulo Global Positioning System (GPS) ao *Sensor Aggregator Gateway*. Este *upgrade* tornaria todo o projeto um pouco mais interessante na medida em que possibilitaria um cruzamento de dados entre o posicionamento GPS e as temperaturas dos respetivos pneus, e assim disponibilizaria para a equipa técnica as temperaturas dos pneus e modificaria a aplicação web de modo a conseguir fazer uma monitorização do posicionamento do veículo na pista.

Uma vez que o sensor de temperatura utilizado neste trabalho apenas foi usado como forma de prova de conceito, um outro trabalho futuro passaria pela aquisição de um sensor de temperatura MLX90620 descrito anteriormente neste documento, que visa a captura de temperatura sob a forma de matriz. Com esta implementação, o uso de apenas um sensor por roda, possibilitaria o acesso a qualquer zona do pneu. A Figura 56 representa um esquemático de como seria feita a captura da informação. Com esta implementação seria possível obter a média de

quatro temperaturas da mesma zona do pneu e assim oferecer uma temperatura mais precisa de cada zona.

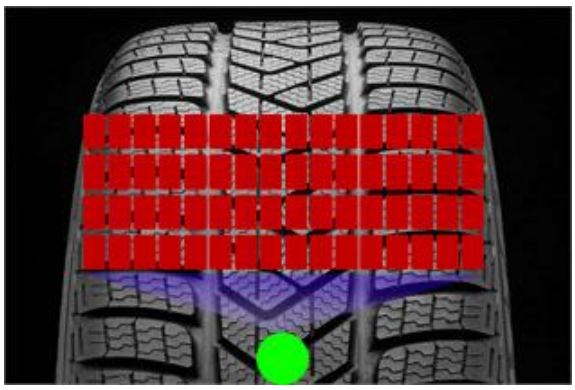

Figura 56. Representação do modo de leitura das temperaturas do pneu através de um sensor matriz.

Ainda sobre esta questão da escolha do sensor, uma alternativa seria o uso de três sensores Melexis MLX90614ESF-DCI que possuem um FOV muito reduzido o que permite um foco mais preciso sobre a zona de leitura da temperatura. A figura 57 (esta figura é a mesma que já apresentada na figura 12, mas que se repete por conveniência) representa a disposição dos sensores e a zona de captura dos valores de temperatura no pneu.

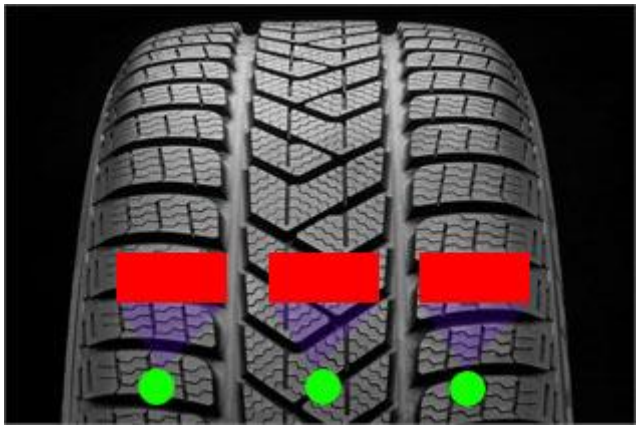

Figura 57. Representação do modo de leitura das temperaturas do pneu através do sensor Melexis MLX90614ESF-DCI.

De modo a garantir uma comunicação mais estável entre todas as partes, uma outra abordagem possível seria a implementação de um sistema de comunicação RF capaz de fazer a cobertura de toda a área da pista onde decorre a prova.

5.2 Aprendizagem

Ao longo deste trabalho foi necessário aprender várias tecnologias e utiliza-las de forma correta, tendo a maioria sido totalmente novas para mim. Neste seguimento,

consigo afirmar que a minha adaptação a esta nova realidade tecnológica foi rápida e eficaz, não tendo demonstrado grandes dificuldades a este nível. Durante o projeto foi ainda necessário aprender o funcionamento dos diferentes dispositivos assim como a sua comunicação de forma a melhorar a performance.

Foram também melhorados os conhecimentos já adquiridos na universidade, como por exemplo a programação de sistemas embutidos sempre com o objetivo de otimizar a performance. As linguagens de programação utilizadas, algumas delas já conhecidas por mim, foram as seguintes:

- HTML
- JavaScript
- Python
- MySql
- PHP
- JSON
- Diversas Application Programming Interface (API)
- C/C++

Bibliografia

[1] Seman aberta, “Semana Aberta da Ciência e da Tecnologia da Universidade de Aveiro,” *semanaberta.ua.pt*, 2014. [Online]. Available: http://semanaberta.ua.pt/prog_detalhe.asp?id=1021&a=2014&tp=ent&ent=8. [Accessed: 02-May-2016].

[2] The Editors of Encyclopædia Britannica, “Telemetry,” *Margaret Rouse*, 2005. [Online]. Available: http://www.britannica.com/technology/telemetry. [Accessed: 24-Jan-2016].

[3] MIE, “Then and Now – Telemetry in F1,” *formula1blog*, 2014. [Online]. Available: http://www.formula1blog.com/f1-news/then-and-now-telemetry-in-f1/. [Accessed: 22-Jan-2014].

[4] J. ALLEN, “HOW MONZA IS HELPING TO SHAPE A REVOLUTION IN AUTOMOTIVE WORLD,” *jamesallenonf1*, 2012. [Online]. Available: http://www.jamesallenonf1.com/2012/09/how-monza-is-helping-to-shape-a-revolution-in-automotive-world. [Accessed: 24-Jan-2016].

[5] E. Colombari, “Anos de dados e 100 sensores: saiba como funciona a telemetria na Fórmula 1,” *terra.com.br*, 2013. [Online]. Available: http://esportes.terra.com.br/automobilismo/formula1/anos-de-dados-e-100-sensores-saiba-como-funciona-a-telemetria-na-formula-1,4d52c056b2c72410VgnVCM5000009ccceb0aRCRD.html.

[6] R. Lopes, “F-1 ou videogame? Pilotos reclamam do excesso de botões no volante,” *globoesporte.globo.com*, 2011. [Online]. Available: http://globoesporte.globo.com/motor/formula-1/noticia/2011/03/f-1-ou-videogame-pilotos-reclamam-do-excesso-de-botoes-no-volante.html. [Accessed: 24-Jan-2016].

[7] The Economic Times, “How money flows in world’s most expensive sport | The Economic Times,” *The Economic Times*, 2015. [Online]. Available: http://economictimes.indiatimes.com/formula-1-india-how-money-flows-in-worlds-most-expensive-sport/race-promoters-source-of-revenue-ticket-sales/slideshow/10536529.cms. [Accessed: 27-Jan-2016].

[8] Metasphere, “Telemetry Data Journey F1 - Metasphere,” *metasphere*. [Online]. Available: https://www.metasphere.co.uk/telemetry-data-journey-f1/. [Accessed: 24-Jan-2016].

[9] K. Varbanov, “Short guide to F1 Telemetry - Spa circuit,” *F1 Framework*, 2013. [Online]. Available: http://f1framework.blogspot.pt/2013/08/short-guide-to-f1-telemetry-spa-circuit.html. [Accessed: 25-Jan-2016].

[10] Stillbrandworks, “The Still – McLaren F1 Race Dashboard,” *stillbrandworks.com*. [Online]. Available: http://stillbrandworks.com/sap/f1-dashboard/. [Accessed: 25-Jan-2016].

[11] Saphanatutorial, “Introduction to SAP HANA (In Memory Database),” *saphanatutorial.com*. [Online]. Available: http://saphanatutorial.com/sap-hana-database-introduction/. [Accessed: 26-Jan-2016].

[12] Saphanatutorial, “Backup and Recovery,” *saphanatutorial.com*. [Online]. Available: http://saphanatutorial.com/sap-hana-backup-and-recovery/. [Accessed: 26-Jan-2016].

[13] S. HANA, “SAP HANA Modeling Guide For SAP HANA Studio.”

[14] Crash, “F1 News - Chinese F1 Grand Prix: SAP gets McLaren livery treatment in China,” *crash.net*, 2014. [Online]. Available: http://www.crash.net/f1/news/203341/1/sap-gets-mclaren-livery-treatment-in-china.html. [Accessed: 25-Jan-2016].

[15] A. Smith, “How to Measure Temperature Without Contact,” *calex.co.uk*, 2015. [Online]. Available: https://www.calex.co.uk/how-to-measure-temperature-without-contact/. [Accessed: 26-Jan-2016].

[16] Sens2b, “New texense® IRN8W Wireless digital infrared temperature sensor - Sens2B | Sensors Portal,” *sens2b-sensors.com*, 2014. [Online]. Available: http://www.sens2b-sensors.com/directory/companies/texys-international/item/new-texense-irn8w-wireless-digital-infrared-temperature-sensor?category_id=546. [Accessed: 27-Jan-2016].

[17] kvaser, “TEXYS International.” [Online]. Available: https://www.kvaser.com/associates/texys-international/. [Accessed: 26-Apr-2016].

[18] K. Zurvalec, “New Sensors Provide Dead-On Diagnostics,” *PRI Magazine*, 2013.

[19] Texys, “IRN8c - Lens 0º or 90º 8- Channel Infrared Tyre Temperature sensor for CAN Bus.”

[20] Texys, “IRN4 C - Texense sensors - By Texys,” *texense.com*. [Online]. Available: http://www.texense.com/en/produits/controls_3/irn4-c-test-departments-equipment-tire-temp-measurement-testing-sensors_80.html. [Accessed: 16-Jan-2016].

[21] Can-cia, “CAN in Automation (CiA): History of the CAN technology,” *can-cia.org*. [Online]. Available: http://www.can-cia.org/can-knowledge/can/can-history/. [Accessed: 16-Jan-2016].

[22] Texys, “IRN4C - Texense sensors - By Texys,” *texense.com*. [Online]. Available: http://www.texense.com/en/produits/world-pro_1/irn4c-tire-tyre-temperature-sensor-multi-channel-can-motorsport-tire-temp_72.html. [Accessed: 27-Jan-2016].

[23] Milspecwiring, “TEXYS 4 channel IR tire temp sensor- lens 0° - CAN output,” *milspecwiring.com*. [Online]. Available: http://milspecwiring.com/store/index.php?route=product/product&path=79_101&product_id=687. [Accessed: 27-Jan-2016].

[24] Milspecwiring, “TEXYS 8 channel IR tire temp sensor- lens 0° - CAN output,” *milspecwiring.com*. [Online]. Available: http://milspecwiring.com/store/index.php?route=product/product&path=79_101&product_id=688. [Accessed: 27-Jan-2016].

[25] A. Labs, “Texense 4 Channel Tire Temperature Sensor | Autosport Labs,” *Autosport Labs*. [Online]. Available: https://www.autosportlabs.com/product/texense-4-channel-tire-temperature-sensor/. [Accessed: 27-Jan-2016].

[26] Texys, “IRN8WS - Texense sensors - By Texys,” *texense.com*. [Online]. Available: http://www.texense.com/en/produits/world-pro_1/irn8ws-wireless-tire-temperature-sensor-wireless-array-sensor-tire-temp-sensor-motorsport-electronic_180.html. [Accessed: 27-Jan-2016].

[27] Texys, “wireless 8ch IR tire temperature sensor.”

[28] Renvale, “Wireless Infrared Tyre Temperature Sensor - 8 Channel,” *renvale.com*. [Online]. Available: http://www.renvale.com/shop/index.php?route=product/product&path=59_152&product_id=74. [Accessed: 26-Apr-2016].

[29] Renvale, “Wireless Master Receiver,” *renvale.com*. [Online]. Available: http://www.renvale.com/shop/index.php?route=product/product&path=59_152&product_id=73. [Accessed: 26-Apr-2016].

[30] arduino, “Arduino - FAQ,” *arduino.cc*. [Online]. Available: https://www.arduino.cc/en/Main/FAQ. [Accessed: 29-Jan-2016].

[31] Esacademy, “I2C FAQ - Embedded Systems Academy,” *esacademy.com*. [Online]. Available: http://www.esacademy.com/en/library/technical-articles-and-documents/miscellaneous/i2c-bus/frequently-asked-questions/i2c-faq.html. [Accessed: 29-Jan-2016].

[32] DealExtreme, “V3.0 para Arduino Nano,” *dx.com*, 2011. [Online]. Available: http://www.dx.com/pt/p/arduino-nano-v3-0-81877. [Accessed: 25-Jan-2016].

[33] M. Chen, “Field Of View Dreams … Why it does not matter or we are hosed,” *tlvexp.ca*, 2011. [Online]. Available: http://www.tlvexp.ca/2013/09/field-view-dreams-matter-hosed/. [Accessed: 05-May-2016].

[34] Melexis, “MLX90620 16x4 IR array,” 2012.

[35] boxtec, “Infrared Thermometer Sensor (medical) MLX90614ESF-DCI - Sensors Temperature - Boxtec Onlineshop,” *boxtec.ch*, 2012. [Online]. Available: http://shop.boxtec.ch/infrared-thermometer-sensor-medical-mlx90614esf-dci-p-40658.html?language=en. [Accessed: 11-Mar-2016].

[36] ecnmag, “Active pixel infrared thermometer array overcomes cost hurdles associated with thermal imaging,” *ecnmag.com*, 2012. [Online]. Available: http://www.ecnmag.com/product-release/2012/03/active-pixel-infrared-thermometer-array-overcomes-cost-hurdles-associated-thermal-imaging. [Accessed: 11-Mar-2016].

[37] Melexis, “MLX90614 family.”

[38] Future Electronics, “MLX90614 Series 5.5 V 13.5 mA Gradient Compensated Infra Red Thermometer - TO-39 - See more at: http://www.futureelectronics.com/en/Technologies/Product.aspx?ProductID=MLX90614ESFDCIMELEXIS3003055#sthash.mIBMqecu.dpuf,” *futureelectronics.com*. [Online]. Available: http://www.futureelectronics.com/en/Technologies/Product.aspx?ProductID=MLX90614ESFDCIMELEXIS3003055. [Accessed: 16-Mar-2016].

[39] Mercado Libre, “Sensor Ir Temperatura Arduino Pic Mlx90614,” *mercadolibre.com.ar*. [Online]. Available: http://articulo.mercadolibre.com.ar/MLA-615316112-sensor-ir-temperatura-arduino-pic-mlx90614-_JM. [Accessed: 28-Jan-2016].

[40] Pirelli, “A Pirelli regressa a casa com pneus médios e duros,” *Pirelli*, 2013. [Online]. Available: http://www.pirelli.com/tyre/pt/pt/news/2013/09/02/a-pirelli-regressa-a-casa-com-pneus-medios-e-duros/. [Accessed: 29-Jan-2016].

[41] H. K. Vasco Polyzoev, “Demystifying Thermopile IR Temp SensorsDemystifying Thermopile IR Temp Sensors | Sensors,” *Texas Instruments*, 2014. [Online]. Available: http://www.sensorsmag.com/sensors-mag/demystifying-thermopile-ir-temp-sensors-13157. [Accessed: 11-May-2016].

[42] M. Rawashdeh, “RF 315/433 MHz Transmitter-receiver Module and Arduino,” *instructables.com*, 2013. [Online]. Available: http://www.instructables.com/id/RF-315433-MHz-Transmitter-receiver-Module-and-Ardu/. [Accessed: 30-Jan-2016].

[43] diy_bloke, “433 MHz Coil loaded antenna,” *instructables*, 2015. [Online]. Available: http://www.instructables.com/id/433-MHz-Coil-loaded-antenna/. [Accessed: 30-May-2016].

[44] Mayank Prasad, “RF Module Interfacing without Microcontrollers » maxEmbedded,” *maxembedded.com*, 2011. [Online]. Available: http://maxembedded.com/2011/09/rf-module-interfacing-without-microcontrollers/. [Accessed: 08-Mar-2016].

[45] rfwireless-world, “ASK vs FSK vs PSK-Difference between ASK,FSK,PSK modulation,” *rfwireless-world.com*. [Online]. Available: http://www.rfwireless-world.com/Terminology/ASK-vs-FSK-vs-PSK.html. [Accessed: 08-Mar-2016].

[46] pjrc, “VirtualWire Library, for very cheap wireless communication,” *pjrc.com*. [Online]. Available: https://www.pjrc.com/teensy/td_libs_VirtualWire.html. [Accessed: 30-Jan-2016].

[47] M. McCauley, “Documentation for the VirtualWirecommunications library for Arduino.1.0 IntroductionArduino is a low cost microcontroller with Open Source hardware, see http://www.ardu-ino.cc. VirtualWire is a communications library for Arduino that allows multiple Ardu-,” 2013.

[48] airspayce, “VirtualWire: VirtualWire library for Arduino and other boards,” *airspayce.com*. [Online]. Available: http://www.airspayce.com/mikem/arduino/VirtualWire/. [Accessed: 30-Jan-2016].

[49] Adafruit, “About Adafruit - Press, Limor Fried / Ladyada & more...,” *Adafruit*. [Online]. Available: https://www.adafruit.com/about. [Accessed: 31-Jan-2016].

[50] Lady Ada, “Wiring and Test | Using Melexis MLX90614 Non-Contact Sensors | Adafruit Learning System,” *adafruit.com*, 2015. [Online]. Available: https://learn.adafruit.com/using-melexis-mlx90614-non-contact-sensors/wiring-and-test. [Accessed: 31-Jan-2016].

[51] F. Kaup, P. Gottschling, and D. Hausheer, “PowerPi: Measuring and modeling the power consumption of the Raspberry Pi,” in *39th Annual IEEE Conference on Local Computer Networks*, 2014, pp. 236–243.

[52] O. A. Paiva and R. de O. Moreira, “Raspberry Pi: a 35-dollar device for viewing DICOM images,” *Radiol. Bras.*, vol. 47, no. 2, pp. 99–100, Apr. 2014.

[53] Matthew Murray, “Raspberry Pi Review & Rating | PCMag.com,” *pcmag.com*, 2012. [Online]. Available: http://www.pcmag.com/article2/0,2817,2407058,00.asp. [Accessed: 31-Jan-2016].

[54] ksrplayer Limited, “Mini 3G/4G Wifi Router USB Smart Wireless Router Wireless AP Portable 3G/4G MiFi Wireless-N USB WiFi Hotspot Router AP 150Mbps Wlan 2 in 1 Many Colors,” *ksrplayer.net*. [Online]. Available: http://www.ksrplayer.net/?Product/Wholesale-Computer-Accessories-From-China/Mini-3G-4G-Wifi-Router-150Mbps-USB-WiFi-Hotspot-AP.html. [Accessed: 15-Feb-2016].

[55] python, “What is Python? Executive Summary | Python.org,” *python.org*. [Online]. Available: https://www.python.org/doc/essays/blurb/. [Accessed: 02-Feb-2016].

[56] abyz, “pigpio library,” *abyz.co.uk*, 2016. [Online]. Available: http://abyz.co.uk/rpi/pigpio/pigpiod.html. [Accessed: 02-Feb-2016].

[57] mysql, “MySQL :: About MySQL,” *mysql.com*. [Online]. Available: http://www.mysql.com/about/. [Accessed: 02-Feb-2016].

[58] JGraph Ltd, “Draw.” [Online]. Available: https://www.draw.io/. [Accessed: 10-Apr-2016].

[59] w3schools, “Bootstrap Get Started,” *w3schools.com*. [Online]. Available: http://www.w3schools.com/bootstrap/bootstrap_get_started.asp. [Accessed: 04-Feb-2016].

[60] amCharts, “About | amCharts,” *amCharts.com*. [Online]. Available: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf. [Accessed: 04-Feb-2016].

[61] Ecma International, *The JSON Data Interchange Format*. Geneva: json.org, 2013.

[62] J. W. Valvano, *Embedded Microcomputer Systems : real time interfacing.*, Second Edi. Austin: Thomson-Engineering Publishers, 2011.