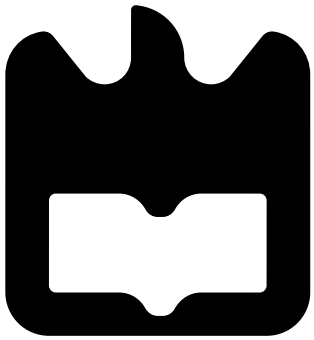
José Luís de Jesus Bettencourt Viana

E-Stop e consola de controlo e monitorização para o AtlasCar



José Luís de Jesus Bettencourt Viana

E-Stop e consola de controlo e monitorização para o AtlasCar

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestrado em Engenharia Mecânica, realizada sob orientação científica de Vítor Manuel Ferreira dos Santos, Professor Associado do Departamento de Engenharia Mecânica da Universidade de Aveiro.

O júri / The jury

Presidente / President

Prof. Doutor Jorge Augusto Fernandes Ferreira
Professor Auxiliar da Universidade de Aveiro

Vogais / Committee

Prof. Doutor Pedro Nicolau Faria da Fonseca
Professor Auxiliar da Universidade de Aveiro

Prof. Doutor Vítor Manuel Ferreira dos Santos
Professor Associado da Universidade de Aveiro (orientador)

Agradecimentos / Acknowledgements

Em primeiro lugar, deixo aqui o meu sincero agradecimento ao Prof. Doutor Vítor Santos, o meu orientador neste trabalho, que sempre me acompanhou e cativou, mostrando sempre o lado bom das coisas. Foi muito bom trabalhar consigo durante estes dois anos! Obrigado, Professor!

Ao grande Jorge muito obrigado por tudo o que me aturaste e me ajudaste, sem ti isto não estava aqui. OBRIGADO JORGE MANUEL!

A todos os aqueles que passaram por este laboratório e sobreviveram, e aos que cá ficam muito obrigado! A vossa companhia foi espetacular. Não vou estar a escrever o nome de todos porque são muitos e depois enchem isto tudo... =D
Um Agradecimento a todos aqueles com quem colaborei, convivi, conheci e me ajudaram durante estes 6 anos de Academia.

À minha família e amigos que sempre me apoiaram e estiveram do meu lado, muito obrigado por tudo.

A ti Inês, muito obrigado por me aturares e estares sempre do meu lado.

Palavras-chave Segurança, Arduino, Xbee, E-stop, monitorização, ROS, SQL, PHP.

Resumo

O veículo AtlasCar é um robô desenvolvido no Laboratório de Automação e Robótica do Departamento de Engenharia Mecânica da Universidade de Aveiro, para o estudo de sistemas de segurança ativos e passivos para auxílio à condução e soluções de condução autónoma.

Este trabalho tem como objetivo a criação de um sistema de emergência remota E-Stop, e uma consola de controlo e a criação de um sistema de monitorização para o veículo, complementando-se mutuamente com dois temas em torno da segurança.

O E-Stop e a consola de controlo consistem na criação de um sistema de atuação remota capaz de gerir 3 sistemas diferentes de atuação por ordem do utilizador: RUN, PAUSE e DISABLE, sendo um elemento necessário à participação no concurso internacional ELROB. Para isso, algum hardware e software foram desenvolvidos para o cumprimento de todas as diretivas impostas pelo mesmo.

O sistema de monitorização funciona sobre uma base de dados SQL, de onde são extraídos os dados para introdução numa página web. A atualização da base de dados é efetuada através do sistema ROS implementado no AtlasCar, tendo este de ter uma ligação à Internet, podendo ser através de rede wifi ou por 3G.

Todo o sistema foi testado e validado em operação real.

Keywords Security, Arduíno, Xbee, E-stop, monitorization, ROS, SQL, PHP.

Abstract

The AtlasCar vehicle is a robot developed at the Automation and Robotics laboratoru in the Department of Mechanical Engeneering at the University of Aveiro. The main objective of this project is to develop both active and passive advanced security systems, for implementation in driver assistance system and autonomous vehicles.

The main goal of this work is the development of a remote emergency E-stop, with a control console, and also the creation of a vehicle monitoring system. These twosystem complient each other in the question of safety.

The E-Stop and the control console are part of a remote atuation system that manages three diferent states of atuation by user order: The three states are RUN, PAUSE and DISABLE. This system is mandatory to be eligle to participate in the international ELROB competion. Both software and hardware were developed with this goal in mind, acording to the competition directives.

The monitoring system consists of a SQL database whose fields are published in a webpage. The refresh of the database is in charge of a ROS software module running on the atlascar using either a wifi or a 3g internet connection.

Both systems were tested and validated in real operation.

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Neste capítulo será feita uma pequena descrição do projeto em que esta dissertação se enquadra.

1.1 Projeto Atlas

O projeto ATLAS foi criado pelo grupo de Automação e Robótica do Departamento de Engenharia Mecânica da Universidade de Aveiro, com o principal objetivo de desenvolver novas soluções na área da condução autónoma. A principal motivação para o início deste projeto foi a participação no Festival Nacional de Robótica em 2003, no qual atingiu o quarto lugar, vencendo depois todas as edições de 2006 até à última participação em 2011. Na figura 1.1 são apresentados 2 protótipos de robôs Atlas para a participação na prova.

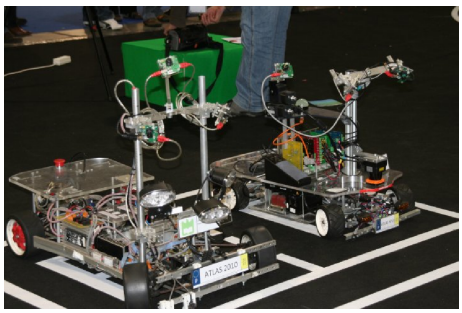


Figura 1.1: Atlas 2010 (esquerda) e Atlas MV (direita)

No seguimento dos vários triunfos obtidos pelos robôs Atlas em pista de ambiente interior, surgiu o desafio de desenvolver um protótipo à escala real, extrapolando muitas das técnicas já implementadas nos vencedores e fazendo com que essas fossem uma mais valia para aplicações reais, tendo como principal objetivo o auxílio à condução evitando situações de perigo, como colisões com veículos, obstáculos e o mais importante com peões, uma vez que são os elementos mais vulneráveis nas vias de circulação automóvel. O protótipo em desenvolvimento desde 2010 é o AtlasCar, figura 1.2. Trata-se de um automóvel adaptado com sensores e atuadores para que a sua condução seja possível quer por um condutor quer por um computador. Entre a lista de sensores conta com dois lasers

2D na frente do carro, com um laser 2D adaptado para fazer leituras 3D no tejadilho, com uma câmara stereo, duas câmaras, sensor de GPS, uma unidade inercial (*IMU*), um encoder acoplado a uma roda e 4 sensores de altura colocados na parte inferior do carro. Em relação aos atuadores, tem um sistema de direção adaptado, o acelerador é controlado eletronicamente, o travão e a embraiagem são atuados recorrendo a duas caixas com motores elétricos para que o seu controlo seja possível eletronicamente, [12].

Figura 1.2: AtlasCar, modelo Ford Escort SW de 1998, com os sensores

1.2 Objetivos

O trabalho apresentado tem como principais objetivos a criação de um sistema de paragem de emergência e um sistema de monitorização remota para o AtlasCar. O desenvolvimento de um sistema remoto de paragem de emergência para o Atlascar (E-STOP), de acordo com as regras de participação no ELROB 1.3.2, mas que também possa ser usado durante testes ou demonstrações. Utilizando para o efeito uma consola de controlo de radiofrequência, sendo esta o mais independente possível do software a correr nos computadores do veículo, recorrendo a unidades de micro controladores dedicados ou similires. Deve-se ainda procurar fazer a integração do sistema desenvolvido com a atual unidade de controlo do Atlascar(Gamepad XBOX). Pretende-se também o desenvolvimento de um módulo de software que emule um painel de monitorização remota do estado da máquina, usando como suporte de comunicação um protocolo TCP/IP ou similar.

1.3 Trabalhos relacionados

1.3.1 Veículos Autónomos

A condução autónoma é um assunto que tem entusiasmado e motivado vários grupos de investigação pelo mundo fora desde a segunda metade do século vinte desenvolvendo protótipos de veículos autónomos. Alguns dos que fizeram história serão apresentados na seguinte tabela:

Ano	Nome do projeto	Objetivos	Veículo
1970's	*Stanford Car* [7]	Apenas uma câmara. Controlo remoto c/ cabo.	
1980's	*VaMoRs* [16]	Identificar as linhas de uma auto estrada.	
1994	*VaMoRs-P* [17]	Identificação de estrada e objetos.	
1995	*Navlab 5* [9]	*"No Hands Across America."* 98% do percurso cumprido em modo totalmente autónomo	
1996	*ARGO* [3]	Primeiro protótipo desenvolvido com o objetivo de ter componentes *low-cost*.	
2004	*DARPA - the Grand Challenge* [5]	Primeira competição de veículos autónomos. Nenhum dos concorrentes completou o percurso de 150 milhas	
2005	*DARPA - the Grand Challenge* [5]	Primeiro veículo a completar o desafio. 150milhas em 6 horas e 54 min, levou o prémio de $ 2 milhões	
2006	*DARPA- Urban Challenge* [14]	Concurso DARPA mas em ambiente urbano. O vencedor que completou o percurso de 60 milhas em 4h e 10min.	
2006	*M-ELROB* [14]	Primeiro concurso europeu militar de condução autónoma.	
2007	*C-ELROB* [14]	Primeiro concurso civil europeu de condução autónoma.	

Tabela 1.1: Veículos autónomos mais relevantes da história. [18] [13] Adaptado [11]

Desde a criação do primeiro veículo autónomo que a sua circulação estava restrita a circuitos ou a ambientes controlados. Porém em junho de 2011 o estado do Nevada apresentou um projeto de lei que previa a circulação desse tipo de veículos na via publica, entrando em vigor a 1 de março de 2012, ou seja, a partir dessa data já era autorizada a circulação de veículos sem condutor nas vias desse estado. Em fevereiro de 2013 outros dois estados Americanos, Florida e Califórnia, também seguiram a lei implementada pelo estado do Nevada [10] [2].
Nas figuras 1.3 e 1.4 são apresentados dois modelos de veículos autónomos desenvolvidos nos E.U.A., que já circularam em vias públicas ao abrigo da lei implementada nos vários estados:

Figura 1.3: Carro autónomo desenvolvido pela Google

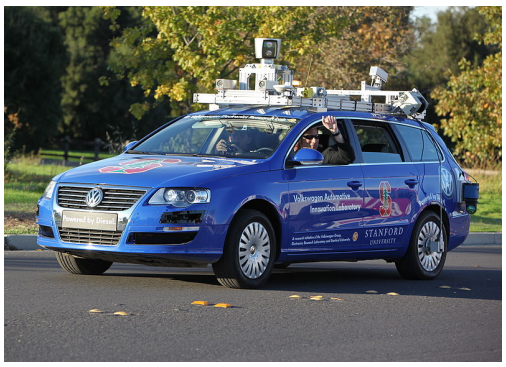


Figura 1.4: Carro autónomo da Universidade de Stanford

1.3.2 *European Land-Robot Trial* (ELROB)

O ELROB não é exatamente de uma competição, mas sim uma demonstração e comparação de funcionalidades e sua demonstração em ambientes reais.

Durante esse evento vários grupos de investigação, utilizadores e a industria automóvel demonstram as funcionalidades dos seus equipamentos, permitindo a troca de ideias entre os diferentes grupos, como se trata de um evento Europeu, apenas está autorizada a participação de equipas Europeias.
O principal propósito da criação deste evento é a utilização da tecnologia existente no momento para resolver problemas "em mãos", na área da robótica, usando qualquer estratégia para o conseguir. Esses avanços tecnológicos podem ser aplicados e adicionados aos sistemas já existentes em veículos para a população, salvando vidas e definindo novos caminhos de investigação a médio e a longo prazo. *"ELROB will enable Europe to re-engage in the benefits robotics can deliver now and the future."* [14]. A demonstração ocorre todos os anos, alternando entre a competição civil e militar.

Figura 1.5: ELROB LOGO

C-ELROB - ELROB civil

O evento acontece de dois em dois anos alternando com o evento militar. foi criado com o objetivo de permitir a equipas que não estejam interessadas no desenvolvimento de ferramentas para a industria militar, também pudessem demonstrar as suas potencialidades. No entanto, o número de participantes em relação ao evento militar é mais baixo. Sendo que os participantes não vêem este evento como uma competição ou demonstração, mas sim como uma maneira de provar que os seus sistema são funcionais, e podem ter aplicabilidade em sistemas reais. Sendo uma ótima oportunidade para aprenderem novas técnicas e aproximações para os seus robôs.

M-ELROB - ELROB militar

O ELROB militar tem mais incidência sobre sistemas para implementação na industria militar a curto prazo; os sistemas apresentados têm de ser capazes de cumprir tarefas militares como, por exemplo, desarmadilhar bombas remotamente, para em caso de falha de missão se evitem perdas humanas. Os cenários fornecidos representam o melhor possível cenários reias, pois os membros da organização fazem parte de equipas militares.*M-ELROB is explicitly designed to assess current technology to solve real world problems at hand.*

À semelhança do C-ELROB os cenários para demonstrações também são mantidos em segredo até dias antes da prova.

1.3.3 Robot Operating System

O sistema ROS é um ambiente de desenvolvimento especialmente criado para aplicação em robótica, e sobre o qual se encontra todo o sistema implementado no Atlascar. Este sistema é extremamente versátil e encontra-se organizado em *packages*. Isso significa que um sistema complexo pode ter vários *nodes*(módulos) para processamento, o que permite a uma fácil compreensão e organização do código, podendo este ser implementado em várias linguagens de programação diferentes. A comunicação entre módulos é efetuada através de publicação e subscrição de mensagens, os chamados *topics*(tópicos) (*strings* que identificam cada mensagem). Em projetos de grande dimensão é comum existirem diversos tópicos a serem publicados e subscritos pelos diferentes módulos existentes. A divisão em pacotes e a comunicação entre módulos por mensagens permite a fácil migração de pacotes entre projetos.

Todos os tópicos podem ser guardados para futura utilização, podendo ser feito um registo de todas as mensagens que são trocadas entre módulos durante a operação do robô, sendo chamado o comando *rosbag*, e o ficheiro criado do tipo *.bag*. Os conceitos referi-

dos e descritos são de extrema importância pois serão mencionados em várias ocasiões durante o texto.

1.3.4 Sistema de rádio controlo

Os sistemas de rádio modelismo contam com alguns sistemas que se prendende implementar no ambito desta dissertação; como os sistemas de telemetria e *fail-safe*, que serão explicados em seguida.

Sistema de Telemetria

A telemetria é uma tecnologia que permite a receção e envio de dados através de um meio de comunicação, sendo o envio efetuado entre o utilizador e o equipamento, normalmente associado a sistemas de rádio controlo. Estes sistemas permitem ao utilizador saber o estado da máquina enquanto esta se encontra em operação; isto leva a uma previsão de falha, o que pode evitar a destruição do equipamento.
Uma das aplicações destes sistemas é nos aviões de rádio modelismo, pois durante a operação do equipamento o utilizador não se encontra perto da máquina, tendo de alguma forma conseguir perceber o estado da mesma, como carga das baterias, posicionamento, em que direção se desloca, etc.. Representa-se na figura 1.6 um sistema de telemetria para implementação em equipamentos de rádio modelismo.

Figura 1.6: Conjunto RC de telemetria

Neste sistema o utilizador tem acesso a dados de sensores que são conectados à unidade de rádio que se encontra do modelo, enviando os dados para o comando, sendo estes depois representados no *display*.

Sistema de *failsafe*

O sistema de *failsafe* é também sistema de segurança para aplicação em modelos de rádio modelismo. Consiste num sistema capaz de detetar a falha de comunicação entre o comando e o recetor, que se encontra no modelo. O sistema é normalmente instalado entre o recetor e o servo a atuar; no caso dos auto modelismo são colocados entre o recetor e o servo responsável pelo acelerador que normalmente é o

mesmo que atua a travão, assim em caso de falha de comunicação ou bateria fraca o sistema de segurança é ativado, fazendo com que o servo ative os travões e o motor baixe as rotações. Na figura 1.7 encontra-se um dispositivo para instalação em carros de rádio modelismo.

Figura 1.7: Módulo *failsafe* de radio modelismo

1.3.5 *Robot Management System* (RMS) do ROS

O RMS é uma ferramenta de controlo remoto para robôs que operem sobre a plataforma ROS, podendo o controlo dos robôs ser efetuado via *web*. O sistema RMS está implementado sobre uma página *web* desenvolvida em PHP, que tem por trás uma base de dados MySQL. É desenvolvido de uma forma independente permitindo o controlo de alguns robôs; tem também a capacidade de gerir utilizadores, gestão da interface e gestão do conteúdo.
O servidor do RMS não necessita de ter instalado o sistema ROS; o acesso a mensagens e todo o conteúdo necessário para o controlo e monitorização do robô é efetuado através do sistema ros_bridge que permite a comunicação através de uma ligação *TCP/IP* entre um módulo ROS e uma interface web, o esquema de funcionamento pode ser consultado na figura 1.8 [15].

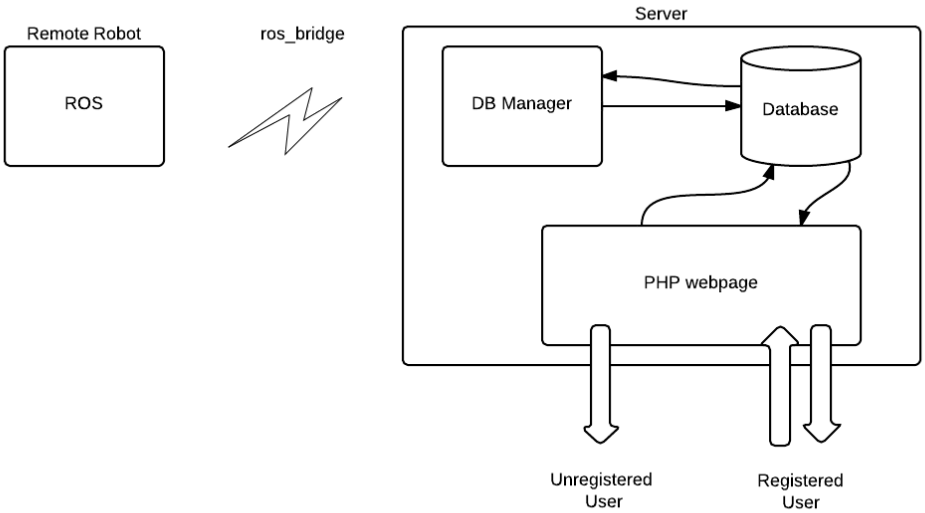


Figura 1.8: Esquema de funcionamento do RMS - Robot Management System

Capítulo 2

Sistema experimental

Neste capítulo apresenta-se todo o sistema experimental que teve de sofrer intervenção no âmbito desta dissertação, tudo relacionado com os sistemas de segurança do veículo.

2.1 Circuitos existentes

O Atlascar como todos os sistemas que envolvem automatismos necessita de uma unidade de emergência, algum tipo de dispositivo ou sistema que, em caso de avaria ou falha, possa ser ativado para imobilização da máquina.
O Atlascar possuí uma unidade responsável pelo controlo de todos os atuadores do veículo, essa unidade é um autómato industrial PLC, e a sua interação com os sistemas existentes no veículo podem ser consultados na figura 2.1.

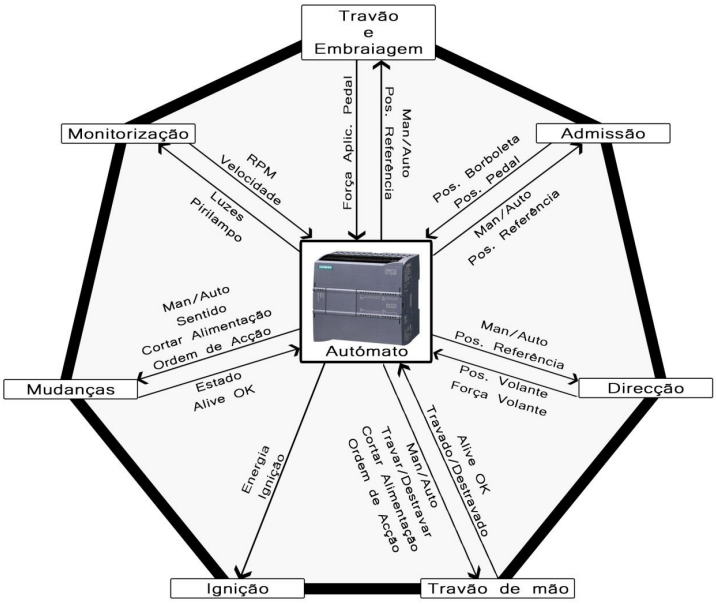


Figura 2.1: Interação entre o autómato e os diversos módulos, adaptado de [12]

Neste caso o veículo possui 2 linhas de emergência, como exemplificado na figura 2.2 uma capaz de ativar uma rotina de emergência no PLC, que se encontra esquematizada na figura 2.5, e outra que interrompe o circuito de alimentação dos atuadores do veículo.[12]

Existe ainda uma linha responsável pelo corte/ativação da ignição, que já anteriormente tinha sofrido alterações para que fosse possível ligar e desligar o veiculo através de instruções do PLC, estando esquematizado o sistema implementado na figura 2.3.

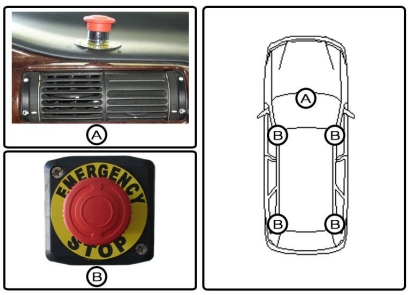


Figura 2.2: Ponto de ativação da linha de emergência presentes no Atlascar.

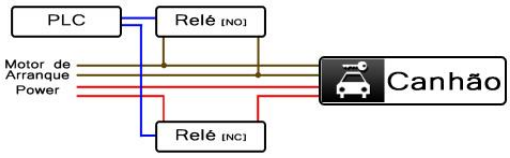


Figura 2.3: Esquema de funcionamento da ignição

Figura 2.4: Imagens das linhas existentes no AtlasCar. Adaptado[12]

A rotina de emergência é ativada quando um botão, da linha B na figura 2.2, é pressionado, interrompendo uma linha de sinal que está ligada ao PLC do veículo. Ao ser ativada a emergência dá-se início a uma série de ações como o acionamento das caixas do travão e da embraiagem, o fecho da borboleta do acelerador do veiculo e, por fim, colocando o volante no modo de atuação manual figura 2.5. Estes procedimentos são efetuados para retirar o controlo do veiculo aos computadores.

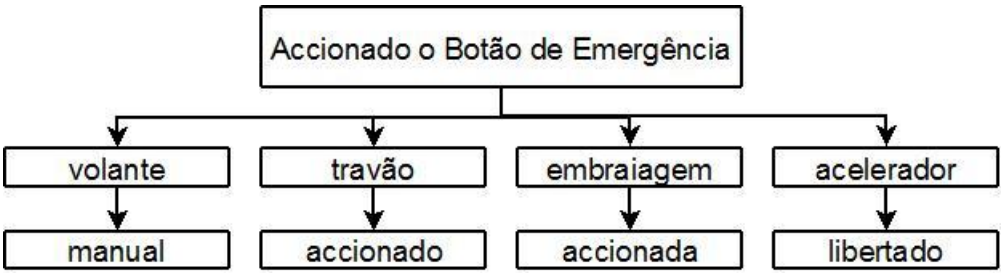


Figura 2.5: Rotina de Emergência implementada no PLC do AtlasCar. Adaptado [12]

2.2 Sistema de atuação das caixas

O AtlasCar, (figura 1.2), é um veículo com alguns anos, e por isso, os sistemas de travagem e de embraiagem são de atuação mecânica. Logo, se o objetivo seria ter um robô à escala real, ter-se-ia de alterar o sistema de atuação, fazendo com que fosse possível a atuação eletrónica do sistema de travagem e de embraiagem. O sistema pensado seria atuado eletronicamente, e quando atuado teria de ser capaz de movimentar os sistemas

de travão e de embraiagem do veículo, mas depois de implementado a condução manual do veículo teria de continuar a ser possível. Então foi implementado um sistema de atuação elétrica que através de um cabo de aço puxa os pedais do veiculo, figuras 2.6 e 2.7 .[12].

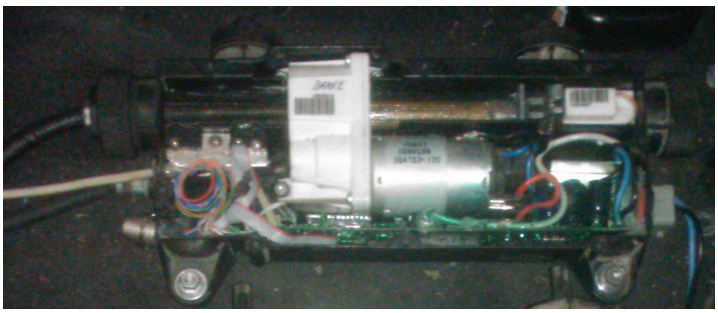


Figura 2.6: Caixa de atuação do pedal travão

Figura 2.7: Fixação do cabo ao pedal do travão

As caixas de atuação encontravam-se na parte inferior do veiculo; uma vez que possuíam cabos curtos aí seria o melhor ponto para a sua instalação; porém essa solução apresentou-se pouco interessante, dado que corriam o risco de embater com o chão e estavam muito expostas a poeiras e líquidos, o que poderia originar danos catastróficos. Assim pensou-se numa alternativa para a recolocação das caixas, um espaço onde estas ficassem protegidas. Foram propostas duas soluções de intervenção: uma recolocar as caixas no local destinado ao pneu suplente, figura 2.9, outra solução seria a recolocação por baixo do banco do condutor do veículo, figura 2.8.

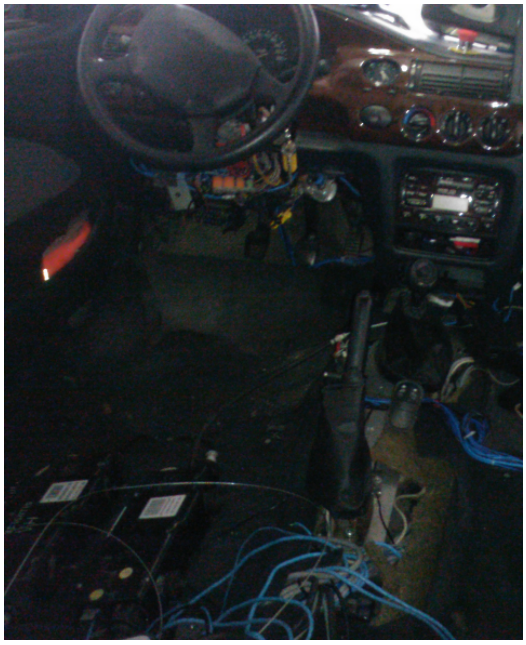


Figura 2.8: Dentro do habitáculo

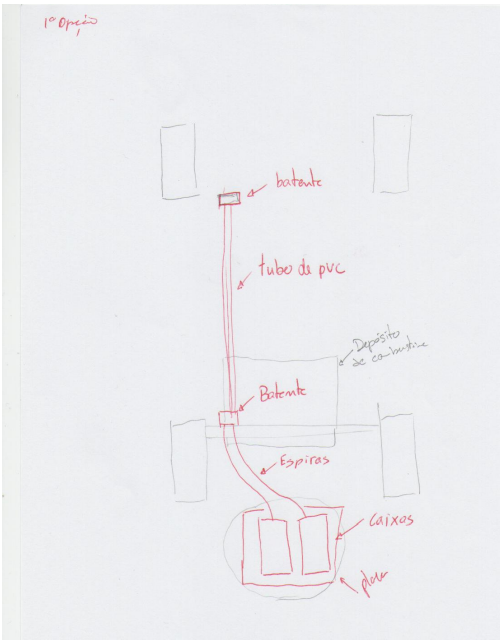


Figura 2.9: Local do pneu suplente

Figura 2.10: Possibilidades de recolocação das caixas de atuação

A melhor solução encontrada para instalação das caixas no carro, foi a sua colocação por baixo do banco do condutor, local em que estas ficariam relativamente próximas dos pontos de atuação. Assim, seria apenas necessário fazer passar os cabos para o exterior do habitáculo, sendo que a partir daí teriam de ser enxertado para os fazer chegar aos pontos de atuação. A extensão dos cabos é efetuada recorrendo a um sistema de 2 batentes instalados no fundo do veículo. O desenho dos batentes pode ser consultado no anexo A e a implementação do sistema pode ser consultada na figura 2.11.

Figura 2.11: Sistema desenvolvido para recolocação das caixas de atuação

2.3 Controlo do sistema de direção

A atuação do Atlascar é efetuada tendo em conta o estado da máquina. Para isso é necessário o envio de dados para o computador a bordo, dados esses com variáveis dos sensores instalados. O sistema de direção anteriormente instalado no veículo era

de baixa fiabilidade; era composto por duas polias e um *o-ring*, apresentando dois problemas, um era o erro de medição que ocorria pelo escorregamento na polia, o outro problema existente era o desgaste prematuro do sistema, sendo a substituição do *o-ring* uma tarefa muito complicada.
Assim, para um bom funcionamento do veículo foi pensada e implementada uma nova solução para o problema, que passa pela alteração do sistema de transmissão, do potenciómetro usado e o seu posicionamento.
Em primeiro lugar o problema foi analisado e discutido, pensado no melhor local para a recolocação do potenciómetro. Assim, surgiu a possibilidade de o potenciómetro ficar colocado por de trás do volante, para isso o relé dos "piscas"teria de ser recolocado; na figura 2.12 é identificado o local para instalação do potenciómetro.

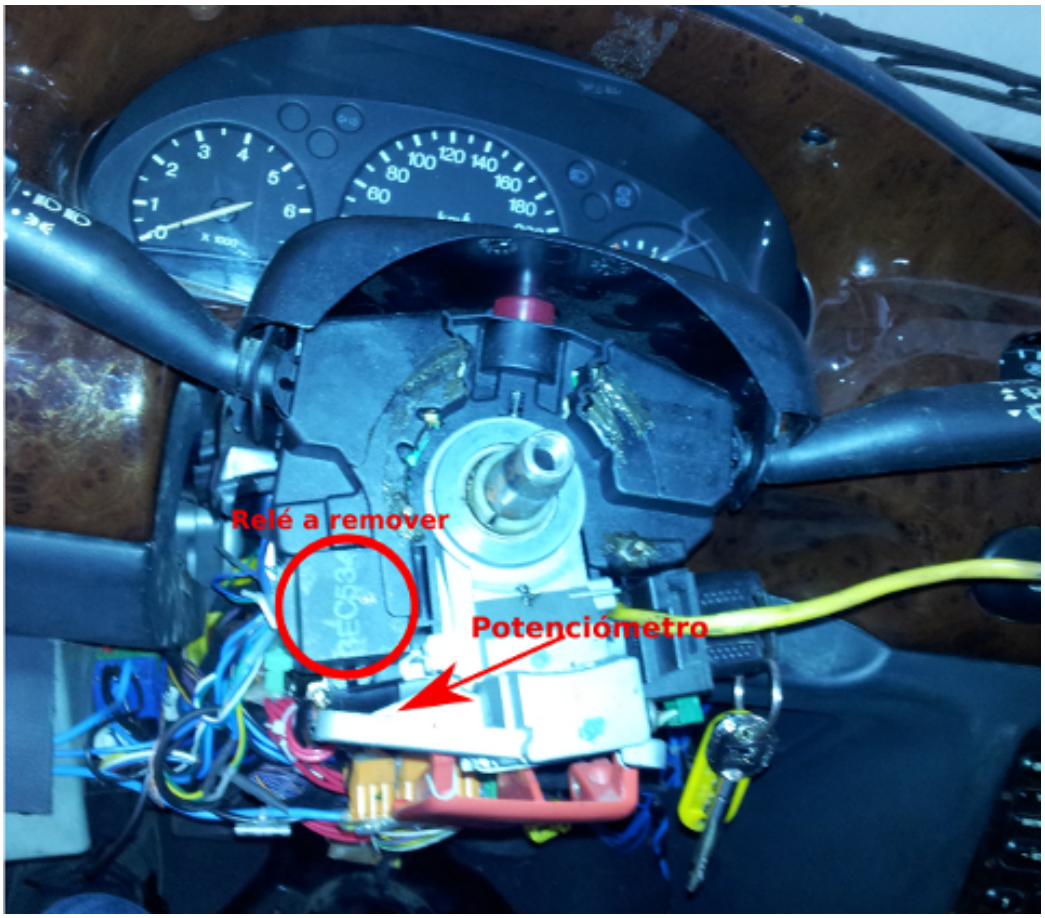


Figura 2.12: Esquema do novo posicionamento do potenciómetro de medição da posição do volante

Depois de aprovada a nova solução encontrada passou-se à instalação do mesmo. O material necessário para proceder a esta tarefa foi o seguinte:

- uma polia com sessenta dentes e um passo de 2.5mm
- uma polia com vinte dentes e um passo de 2.5mm
- uma correia com 235mm

- um potenciómetro de 10 voltas

- um suporte para instalação do potenciómetro

Com o material descrito anteriormente a relação de transmissão obtida é de 1:3, o que trouxe um beneficio ao sistema: a determinação do posicionamento do volante é mais preciso, pois o valor da tensão à saída do potenciómetro varia bastante com pequenos movimentos do volante, o que se traduz numa melhor determinação da sua posição do mesmo.

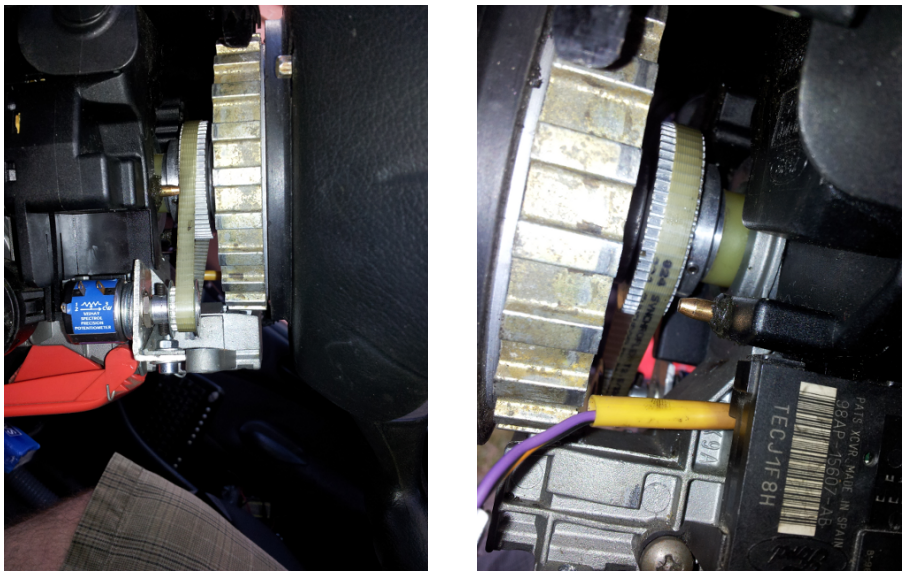


Figura 2.13: Sistema desenvolvido para recolocação do sistema de medição da direção

A interpretação do valor da variação da tensão do potenciómetro é feita através de um μC, um PIC 18F258. O seu código não sofreu qualquer alteração com a substituição do componente, pois em valor nominal o novo potenciómetro tem a mesma resistência que o anterior que se encontrava no carro, $10k\Omega$. A única intervenção a nível de software que se efetuou com esta alteração foi a redefinição dos extremos do potenciómetro, sendo que as variáveis alteradas são *steeringwheel_ min* =0.3 e *steeringwheel_ max* =4.4. Estes dois valores traduzem a leitura feita pelo PIC e é enviada para o computador quando o volante se encontra nos extremos do seu percurso.

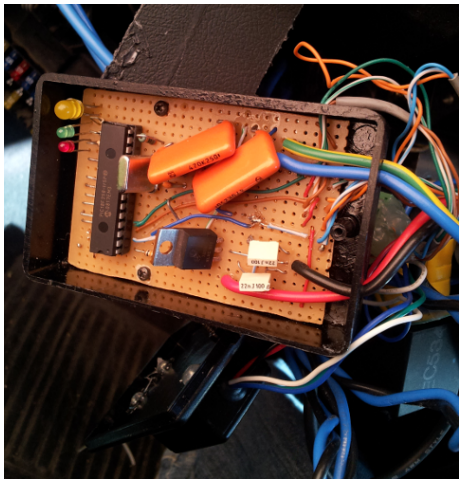


Figura 2.14: Imagem do μC responsável pela leitura do potenciometro da diteção

O reposicionamento do sistema de medição da direção levou a que se tivessem de proceder a uma série de alterações nos sistemas lá existentes; uma das alterações foi o reposicionamento do relé dos piscas, figura 2.15. A outra foi a instalação de um botão para a buzina do veículo, pois com o posicionamento da polia acoplada ao volante, o *slipring* responsável pela continuidade do circuito da buzina teve de ser removido; com isto, o acionamento da buzina do veículo deixará de ser no centro do volante passando a ser no botão do lado esquerdo, por baixo do manípulo dos piscas figura 2.16.

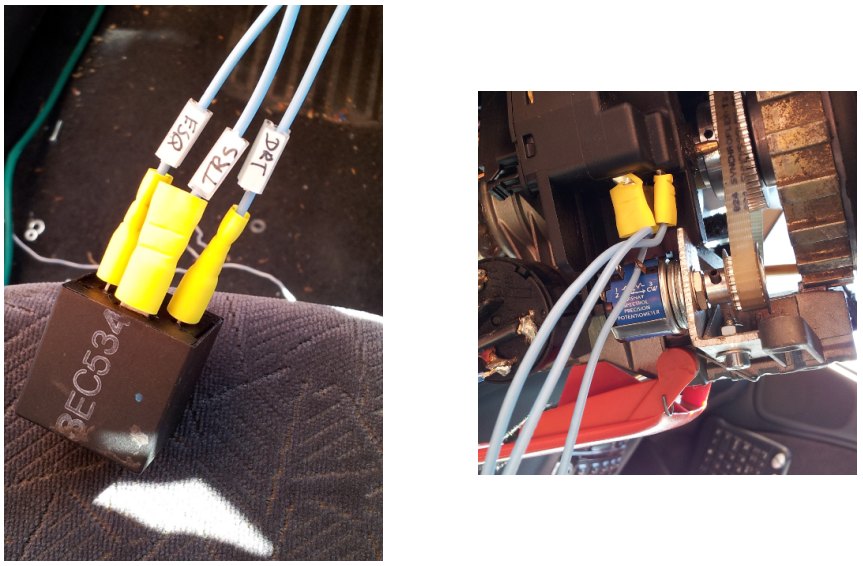


Figura 2.15: Reposicionamento do relé dos piscas do Atlascar

Figura 2.16: Nova posição do botão da buzina

Capítulo 3

E-Stop e consola de controlo

Neste capitulo será apresentada a metodologia para criação e implementação do sistema E-STOP e a consola de controlo no veículo Atlascar.

3.1 Estudo do problema

Uma das imposições do **ELROB** é a existência, nos veículos participantes, de um sistema de atuação remota que possibilite a sua imobilização se, durante uma demonstração, algo de errado aconteça, para evitar danos tanto nos veículos como nos espetadores. Assim, a existência de um comando para atuação remota dos modos do E-Stop seja obrigatório. Os modos de operação do sistema e as ações a tomar quando cada modo está ativo são imposições por parte da organização do evento. O sistema é constituído pelos seguintes modos [14]:

- **RUN**
- **PAUSE**
- **DISABLE**

O modo **RUN**, é o modo normal de operação do sistema, ou seja, o modo de normal funcionamento do carro. O modo **PAUSE**, quando ativo, tem de ser capaz de imobilizar totalmente o veículo, podendo este retomar novamente ao modo **RUN** remotamente. O modo **DISABLE**, sendo ativado o carro tem de ser imobilizado e a fonte de propulsão tem de ser desligada. Este modo não pode ser revertido remotamente, para o veiculo poder voltar ao modo automático o carro tem de ter intervenção humana. É um modo de emergência e a sua ativação deve ser o ultimo recurso.

3.1.1 Soluções encontradas

Depois de alguma pesquisa, foram encontradas duas soluções, que se adaptavam ao sistema a implementar. Uma delas um sistema de comando e recetor de rádio modelismo, pois modelos mais recentes incluem telemetria a outra solução passaria pela implementação de toda a rede sem fios, sobre Arduinos com módulos de Xbee. [1] [6]

O sistema de radio modelismo com telemetria, é extremamente fiável e com alcance aceitável para o problema proposto. A telemetria seria para monitorização de algumas

variáveis de sistema do carro, pois o utilizador, estando com o comando na mão e afastado um pouco do veículo, continuaria a ter acesso a alguma informação relevante do mesmo. Mas sendo um sistema muito fiável, também é muito fechado, ou seja, tem protocolos específicos de comunicação com os elementos e codificação de mensagens, logo a sua aplicação e adaptação para o problema proposto não seria trivial e de difícil expansão futura.
Em relação ao sistema sobre micro-controladores(μC) com módulos Xbee para comunicação sem fios, existe a vantagem de ser totalmente configurável e de comunicação bidirecional, o que permite a monitorização do sistema; na tabela 3.1 estão representados os sistemas tidos em conta para implementação do E-Stop.

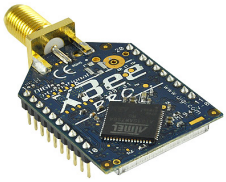
Modo de Comunicação	Rx/Tx	Imagens	Alcance
Modulo de Rádio Modelismo	Sistema de comunicação totalmente fechado Desconhecimento dos modos de comunicação entre o emissor e os sensores		500m (em campo aberto)
XBee-PRO® S2B	Totalmente configurável. Comunicação digital		90m (*Indoor*) 1500m (em campo aberto)
XBee-PRO® 868	Totalmente configurável. Comunicação digital.		550m (*Indoor*) 40km (em campo aberto c/ *dipole antena*)

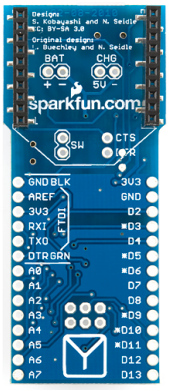
Tabela 3.1: Comparação dos vários sistemas encontrados para implementação do E-Stop

3.1.2 Escolha de um módulo

Depois de uma análise cuidada da tabela 3.1, e tendo em linha de conta a aplicação, a escolha recai sobre os módulos Xbee Pro S2B. São módulos de fácil futura expansão e em comparação com o Xbee Pro 868, a diferença mais significativa, além do preço, é o alcance que, para o problema em questão, não é muito significativo, pois o módulo sobre o qual recaí a escolha tem um raio de alcance suficiente para uma segura operação do veículo.

3.2 Sistema a implementar

Com a escolha efetuada no capítulo anterior apenas ficou definido o modo de transmissão de dados sem fios; a organização de mensagem, envio e processamento fica a cargo de μC. Selecionando para o efeito sistemas de μC Arduino, foi decidido usar para o comando remoto um Arduino FIO e para unidade do veículo um Arduino Mega.

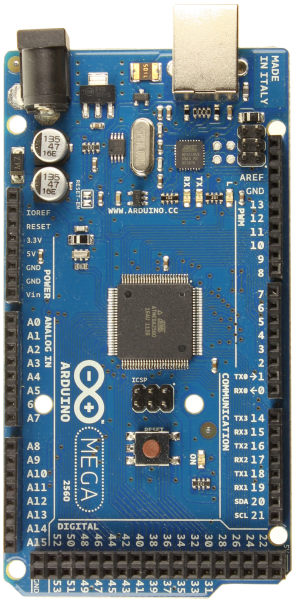


(a) Arduino Fio

(b) Xbee Pro S2B

Figura 3.1: Imagens do Sistema a ser implementado no comando E-Stop.

O sistema a implementar na unidade do veículo baseia-se no mesmo do comando remoto, apenas substituindo as placas de μC; esta escolha deve-se ao poder computacional do μC do comando remoto não ser suficiente para processamento das mensagens enviadas, mas essa questão será mais aprofundada adiante.



(a) Arduino Mega

(b) Xbee Pro S2B rpsma

Figura 3.2: Imagens do Sistema a ser implementado no AtlasCar para o E-Stop.

3.2.1 Funcionamento do E-Stop

O funcionamento do sistema a implementar com os componentes escolhidos anteriormente, é o seguinte:

- A ação do utilizador seleciona um dos seguintes modos do E-Stop (RUN, PAUSE e DISABLE), que é traduzida para o μC do comando através da seleção do botão apropriado.
- Depois de interpretada a entrada digital que se encontra ativa, é construída uma mensagem, para ser enviada para a unidade do veículo.
- A mensagem chega à unidade e aí é interpretada, ativando ou desativando as saídas correspondentes ao comando que é recebido.
- Em seguida a unidade do carro faz uma verificação das portas que se encontram ativas, e constrói uma mensagem para ser devolvida ao comando remoto.
- A mensagem que chega ao comando remoto é interpretada e é ativado um *led* correspondente ao estado E-Stop, isto para ser possível informar o utilizador do estado do E-Stop.

Sendo a figura 3.3, o esquema do funcionamento do E-Stop

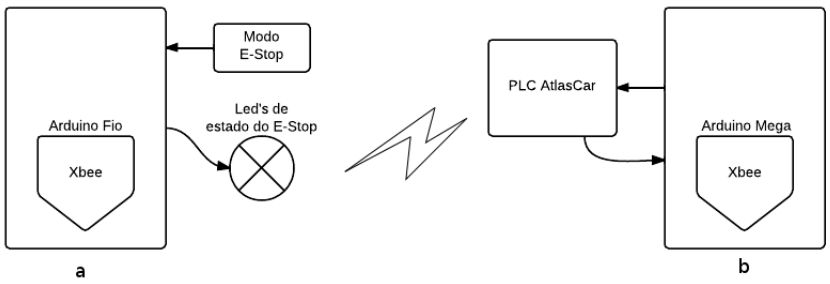


Figura 3.3: Esquema de funcionamento do E-Stop a)comando remoto b) sistema do carro

3.3 Configuração do sistema

Nesta seção serão explicados todos os procedimentos e rotinas implementadas para a configuração do sistema.

3.3.1 Rotinas de segurança

Um sistema de segurança tem de ser fiável, não podendo falhar ou fornecer instruções erradas; para isso não acontecer, neste caso em concreto, foram introduzidas no sistema algumas rotinas de segurança, procurando dar robustez ao sistema de E-Stop.

E-Stop independente de software
Uma das medidas tidas em linha de conta para implementação deste sistema, foi torna-lo o mais independente possível do controlo já existente no AtlasCar, tentando que não dependesse de rotinas de processamento internas do veículo. A única rotina de que se depende para a ativação de um modo do E-Stop é a de emergência do PLC,[12] que, quando ativada, imobiliza o veículo.

Envio de CRC *cyclic redundancy check*
Cada mensagem enviada tem um código de verificação para garantir que esta chega ao destino sem erros de informação, e que todos os bits são enviados.

Falha na comunicação
Em caso de perda de comunicação o modo de **PAUSE** é ativado na unidade do veículo, enquanto no comado remoto todos os leds são acesos, alertando o utilizador para o problema. Entende-se como perda de comunicação entre módulos, a não confirmação de um CRC durante dois segundos de atividade (sendo este tempo configurável).

Taxa de transmissão de mensagens
As mensagens são trocadas entre os elementos do sistema a uma taxa de 5 Hz, foi escolhido este valor pois é o ponto de equilibrio para o tempo de atuação no carro, que se quer o mais rápido possível, e o tempo de interpretação de mensagens por parte dos μC.

Mensagens Dinâmicas
As mensagens enviadas do comando para o carro e vice-versa são cadeias de caracteres, que contêm a informação necessária para serem tomadas ações. Mesmo que a informação a ser transmitida entre dispositivos seja repetida, a string enviada nunca é igual; tornando o *CRC* diferente em cada mensagem evitando falhas, a descrição das mensagens trocadas pode ser consultada na subsecção 3.3.2

Ações a tomar
As ações a tomar por parte do elemento recetor de uma mensagem, só são efetuadas depois da verificação *CRC*.

3.3.2 Estrutura das mensagens

Para ser trocada entre sistemas, a informação tem de ser organizada de forma a poder ser transmitida numa *string*, para que quando interpretada uma mensagem a ação certa seja tomada.
Assim a mensagem enviada do comando para o carro é uma *string* com o seguinte formato

0x02	STR1	COUNT	STR1	ESTOP-STAT	STR3	BAT-VAL	CRC16	0x03

Definição de cada campo da mensagem mostrada anteriormente:

0x02

Código que dá início à mensagem.

STR1

*String: "Hello "*Elemento presente para separação da mensagem a enviar.

COUNT

Esta é uma variável muito importante na *string* a construir; trata-se de um contador que a cada ciclo de código no comando remoto é incrementado. Implementando assim um sistema de segurança, pois a string a enviar é alterada a cada ciclo.

STR2

String: "e-stop " presente para separação da mensagem a enviar.

ESTOP-STAT

Variável em que é inserido o estado do E-stop para ser transmitido para o carro

- *15* - corresponde ao estado **RUN** do E-stop
- *25* - corresponde ao estado **PAUSE** do E-stop
- *35* - corresponde ao estado **DISABLE** do E-stop

Assim, quando a mensagem é tratada, a ação é tomada consoante o estado que é enviado para o carro.

STR3

String: "bat " presente para separação da mensagem a enviar.

BAT

Variável com o valor medido na entrada analógica referente ao divisor resistivo colocado no μC do comando remoto para saber o estado da bateria.

CRC16

Onde se encontra o *Cyclic redundancy check*(verificação cíclica redundante), que é um código de deteção de erros nas mensagens, é calculado com base nos carateres da *string* gerando uma string em *HEX* que quando a mensagem é recebida, é calculado novamente para verificar se algum caratere da mensagem foi perdido durante a transmissão da mesma.

0x03

Caratere com o qual se termina a *string* a enviar.

À semelhança das mensagens trocadas entre o comando remoto e unidade do veiculo, também a unidade do veículo envia mensagens de estado para o comando remoto tendo a mensagem trocada a seguinte forma:

0x02	COUNT	E-STOP	VAR1	VAR2	VAR3	CRC16	0x03

0x02

Caratere com o qual se dá início à mensagem.

COUNT

Esta é uma variável muito importante na *string* a construir; trata-se de um contador que a cada ciclo de código no comando remoto é incrementado. Implementando assim um sistema de segurança, pois o string a enviar é alterada a cada ciclo.

E-STOP

String com o estado do e-stop, para ser enviado para o comando

- *resume* - corresponde ao estado **RUN** do E-stop
- *pause* - corresponde ao estado **PAUSE** do E-stop
- *stop* - corresponde ao estado **DISABLE** do E-stop

VAR1	VAR2	VAR3

Espaço disponibilizado para futura monitorização de variáveis do Atlascar, não estando neste momento implementadas; seriam variáveis de monitorização para serem apresentadas na consola de controlo. Podem ser usadas para futura expansão do sistema.

CRC16

Onde se encontra o *Cyclic redundancy check*(verificação cíclica redundante), que é um código de deteção de erros nas mensagens, é calculado com base nos carateres da *string* gerando uma string em *HEX* que quando a mensagem é recebida, é calculado novamente para verificar se algum caráter da mensagem foi perdido durante a transmissão da mesma.

0x03

Caráter com o qual se termina a *string* a enviar.

3.3.3 Configuração dos Módulos Xbee

Os módulos Xbee são dispositivos de comunicação sem fios com diversas possibilidades de configuração.
Neste projeto os dois módulos foram configurados para comunicarem entre si, evitando assim tentativas de conexão a outros elementos o que poderia originar interferências. Foram configurados usando o software *X-CTU*, disponibilizado pela *DIGI* [6], o qual permite configurar todos os campos necessários para a comunicação entre os módulos. Todos os passos de configuração podem ser reproduzidos recorrendo ao tutorial que se encontra em anexo D.
As configurações possíveis de implementar com estes sistemas são inúmeras, e esse também foi um dos fatores que originou a escolha. As configurações usadas para este sistema estão descritas na tabela 3.2.

Campo	Comando Remoto	Unidade do Atlascar
Modo	Coordinator AT	End Device AT
ID	1234	1234
SC	FFFF	FFFF
SD	3	3
ZS	0	0
NI	COMANDO	ATLASCAR
DH	13A200	13A200
DL	408CBFFB	40918B44
BD	4(19200 bit/s)	4(19200 bit/s)
NB	0(sem paridade)	0(sem paridade)

Tabela 3.2: Tabela dos campos usados para configuração dos *Xbee's*

3.4 Implementação do E-Stop

Nesta secção apresenta-se todo o trabalho desenvolvido para a implementação dos sistemas e procedimentos descritos nas secções anteriores. tanto na instalação do sistema no comando do E-Stop bem como no módulo instalado no AtlasCar.

3.4.1 Comando do E-Stop

O AtlasCar pode ser controlado à distância por um comando da Xbox figura 3.4.

Figura 3.4: Comando usado no controlo remoto do Atlascar

Assim, depois de alguma pesquisa encontrou-se uma possível solução para o problema: o comando da Xbox tem um Chatpad (figura 3.5), que pode ser adaptado para implementação do sistema em causa.

Figura 3.5: Comando Xbox com o chatpad

Pensou-se numa possível integração de todos os componentes usando o *ChatPad Xbox 360*, e foi elaborado uma maqueta onde se integram todos os comandos e *led's*. A solução inicial previa a inclusão de um *LCD* no comando para disponibilizar algumas variáveis ao utilizador.

Figura 3.6: Protótipo inicial do comando E-Stop para integração no AtlasCar

Como o *Chatpad* é um elemento com espaço reduzido para albergar os componentes necessários para implementação do E-Stop, então o espaço teve de ser aumentado, recorrendo-se para isso a uma peça feita à medida, encontrando-se o seu desenho técnico em anexo C.
A organização dos elementos que compõem o comando E-Stop foi inicialmente pensada da seguinte forma figura 3.7.

Figura 3.7: Primeiro teste para organização dos elementos

Tendo em conta que o sistema a desenvolver seria para utilização humana, além dos aspetos funcionais, teve-se em linha de conta a ergonomia do sistema, porque sendo um sistema de emergência os comandos teriam de ser posicionados num local de fácil acesso. Assim depois de criação do prototipo do comando, no laboratório foi dado o sistema a alguns colegas para testarem a disposição dos botões e dos *leds*.
Depois de debatidos os pontos de vista de cada um e qual a melhor disposição dos

controlos, foi unânime que seria mais fácil a comutação entre o modo de *PAUSE* e *RUN* recorrendo a um movimento vertical, bem como seria mais intuitivo a disposição dos leds ser vertical, o que permitia uma melhor consulta do estado. Chegando-se assim ao resultado final figura 3.8.

Figura 3.8: Comando remoto final

Esquemas elétricos

O sistema implementado no comando é composto pelos seguintes elementos

- Botão de emergência - quando pressionado ativa o modo *DISABLE*
- Botão de 2 posições - para alternar entre o modo de *RUN* e *PAUSE*
- *LED's*; verde (*RUN*); amarelo (*PAUSE*); vermelho (*DISABLE*)
- 3 Resistências de 120 Ohm - para polarizar os *LED's*
- 2 Resistências de 430 Ohm - para fazer um divisor resistivo

A implementação de um divisor resistivo no comando remoto, prende-se com o facto da alimentação ser efetuada através de uma bateria, sendo necessária a monitorização do valor da tensão da bateria. Assim a única forma de o fazer é recorrendo a um divisor resistivo que se encontra ligado a uma entrada analógica do μC. Para o valor das resistências foram tidos vários fatores em consideração, sendo o mais importante o consumo energético que isso implicaria para o sistema, mas também era necessário passar corrente suficiente para poder ser feita a leitura na porta analógica do μC.

O esquema elétrico dos elementos que constituem o comando do E-Stop podem ser consultados na figura 3.9.

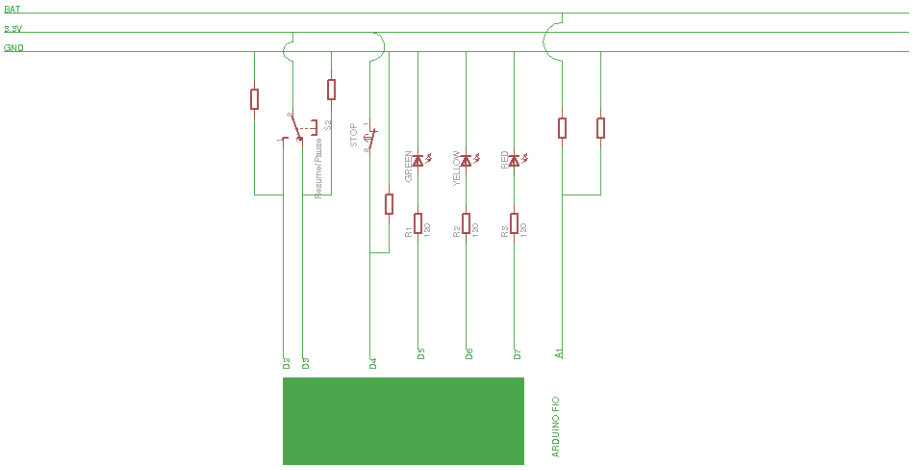


Figura 3.9: Esquema eléctrico do comando remoto

Depois da verificação do esquema elétrico e teste em bancada do sistema, passou-se para a fase de execução do projeto final, com o elemento de ampliação do *Chatpad*, modelado e maquinado com os furos corretos para introdução dos elementos do comando

Figura 3.10: Comando E-Stop implementado no AtlasCar com o cabo de alimentação

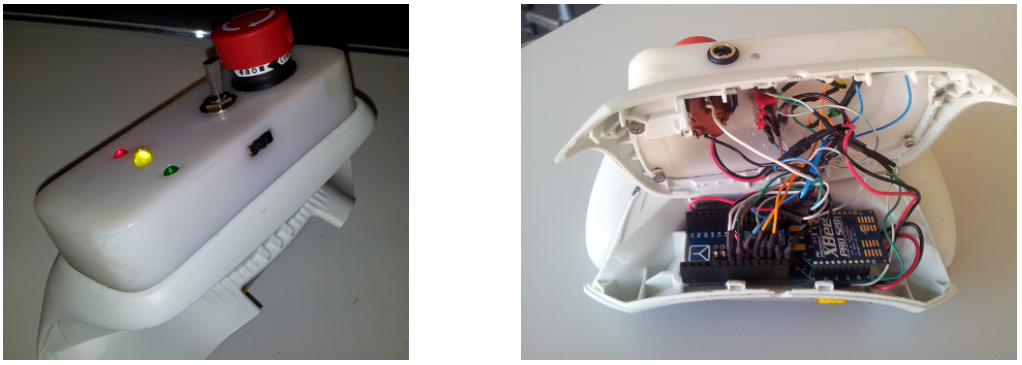


Figura 3.11: Imagens de diferentes perspetivas do comando remoto

3.4.2 E-Stop no Atlascar

Para o módulo instalado no AtlasCar e como descrito na secção 3.2, usou-se um Arduino Mega com o módulo Xbee. O sistema no carro terá de interromper algumas linhas já existentes, descritas na secção 2.1, como a linha responsável pela ativação da emergência no PLC e a linha de corte de energia ao motor.
A escolha destas duas linhas prende-se com o facto de serem as que melhor se adaptam às funcionalidades do E-Stop, pois como explicado na seção 3.1, o modo **PAUSE** terá apenas de imobilizar o veículo enquanto o **DISABLE** terá além de imobilizar desligar a fonte de propulsão do veículo.
A linha de emergência será cortada quando o modo de **PAUSE** for ativado, fazendo com que o carro seja imobilizado e podendo novamente voltar à ação quando esta deixar de estar interrompida. Enquanto o modo **DISABLE** além de interromper a linha de emergência do PLC, também interrompe a linha de alimentação do motor do veículo, desligando-o.

Circuitos de atuação

Para o corte das linhas usou-se relés; outras alternativas foram pensadas, como transístores, mas a que melhor se adequava ao problema em causa eram relés eletromecânicos. São equipamentos de custo reduzido e robustos. Uma das principais motivações para a escolha dos relés eletromecânicos com contacto normalmente aberto, prende-se com o facto de em caso de falha de energia e perda do controlo remoto o sistema ser capaz de imobilizar o veículo interrompendo as linhas. Também permite que o controlo de atuação sobre as linhas seja totalmente efetuado do lado do μC. No início de ciclo, por parte do μC, todos os contactos se encontram abertos, até à ordem de fecho dada pelo μC, evitando assim que o veículo entre em movimento sem ordem do utilizador.
Para a criação do sistema de atuação com relés foi necessário o seguinte material:

- acoplador ótico;
- Relés NO(normaly open);
- Resistências
- Díodos de proteção

Em seguida será feita uma descrição de todos os componentes, bem como o que motivou a sua escolha.

Opto acoplador

Este componente eletrónico foi usado como interruptor. É ativado pelo circuito digital de baixa tensão, 5V(Arduino Mega), e atua sobre a linha de tensão mais elevada 12V(linhas de energia do carro). Este componente é um isolador elétrico pois o acoplamento do circuito é efetuado por via ótica, existindo um led(emissor) do lado da linha de baixa tensão e um fototransistor que comuta o circuito de alta tensão.
Para o dimensionamento do circuito onde este componente se encontra, foram tidos em conta vários fatores, sempre orientados pelo *datasheet* que se encontra no anexo B.
Para ser usado como interruptor o transístor deve transitar entre o regime de saturação e o de corte. Para isso teve de ser efetuado um cálculo para verificação do regime de

atuação do transístor e da potencia dissipada no opto-acoplador.
Assumindo que se deseja uma tensão de $V_{CE(sat)} = 0.4V$ e usando o valor de resistência $R = 1.2k\Omega$, valor resistência da bobine do relé, então a corrente de coletor (I_C) é dada pela expressão:

$$V = R.I \Rightarrow I_C = \frac{12-0.4}{1.2} \Rightarrow I_C = 9.67mA$$

Com este valor de corrente no coletor do transístor e o $V_{CE(sat)}$ desejado, recorre-se à figura 12 do *datasheet*, que se encontra em anexo B. Verificando que a corrente do díodo I_F deve ser superior a 12 mA.

$$R_D \leq \frac{5-1.2}{12} \Rightarrow R_D \leq 310\Omega$$

Escolheu-se para isso uma resistência de 250 Ohm (R1 e R2)3.13, que garante o funcionamento do sistema para toda a gama de temperaturas desejadas e tendo em conta a tolerância dos componentes. Com estes valores garante-se também que o valor da potência dissipada é inferior ao limite presente na figura 6 do *datasheet*.
Os diodos de re-circulação(D1 e D2 figura3.13) são usados para impedir que quando a bobine é comutada, a sobretensão de interrupção de circuito danifique o transístor do opto acoplador.

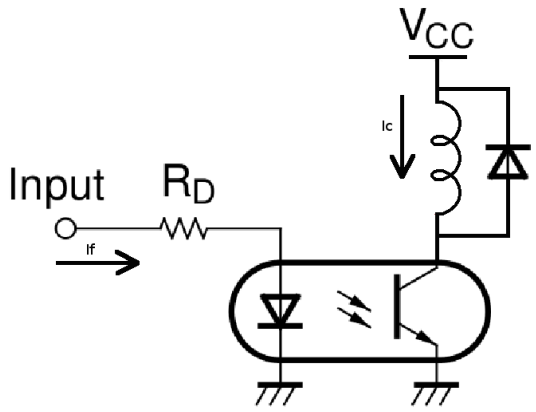


Figura 3.12: Circuito elétrico exemplificativo do opto acoplador

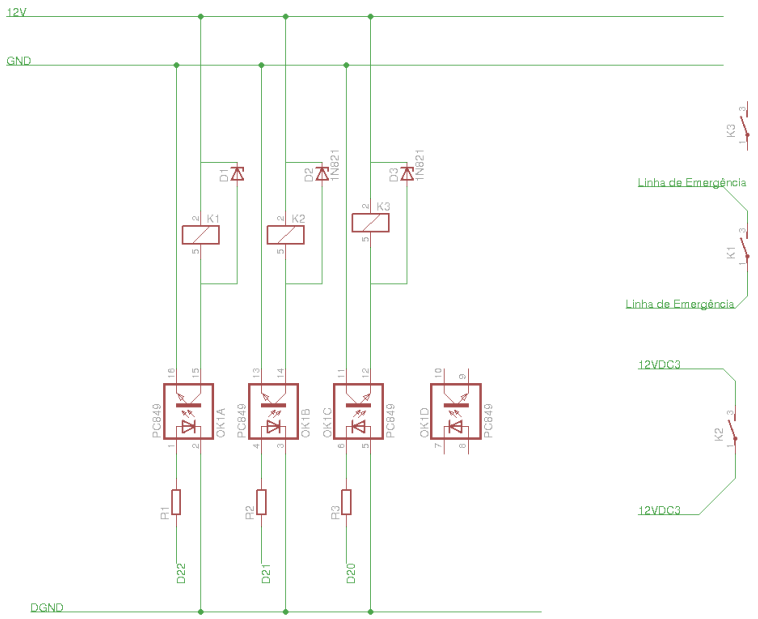


Figura 3.13: Esquema elétrico do sistema de atuação, sendo os *inputs* as saídas digitais do μC

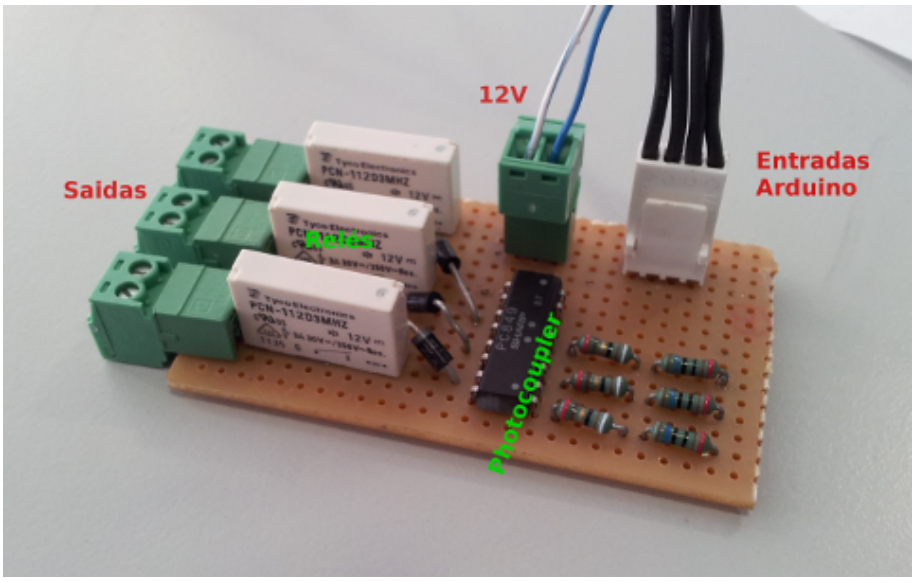


Figura 3.14: Placa desenvolvida para o sistema de atuação

Shield para o Xbee

Para os equipamentos Arduino são desenvolvidas placas especificas para comunicação com outros equipamentos, essas placas para integração dão pelo nome de *shield*, neste caso teria de ser usado um *Shield*, existindo vários tipos para comunicação com os diferentes equipamentos como ethernet, xbee, RFID e etc.. Na figura 3.15 está presente um *shield* para integração do xbee no Arduino UNO.

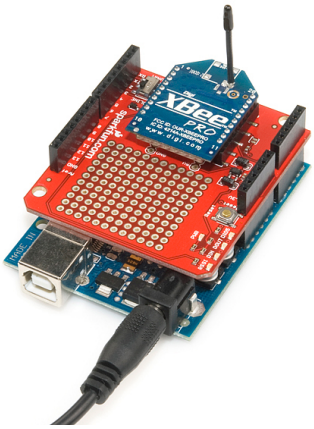


Figura 3.15: Exemplo de um *shield* xbee para o Arduino UNO

Não foi possível em tempo útil integrar um *Shield* para o Arduino Mega onde seria integrado o Xbee S2B, assim teve-se de desenvolver uma placa para garantir a comunicação do μC com o Xbee e vice-versa. Assim depois de perceber quais os pinos necessários para a comunicação criou-se um em placa furada, figura 3.16.

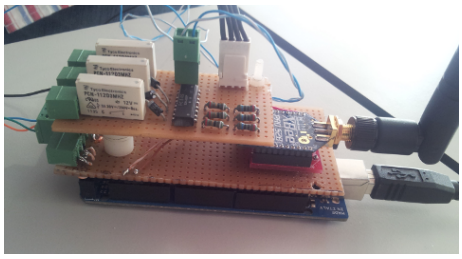


Figura 3.16: Shield do sistema E-Stop instalado no carro

Sistema manual em caso de falha do E-Stop

As linhas em que o E-Stop vai atuar foram interrompidas com relés do tipo NO (*Normaly Open*, normalmente aberto), ou seja, se não forem alimentados os relés interrompem o circuito. Isto implica que tanto a linha de emergência como a linha de alimentação do motor fiquem interrompidas e consequentemente o carro pode não funcionar corretamente. Para evitar que, em caso de falha do sistema E-Stop, o carro não funcione, foram colocados 2 interruptores, que em caso de falha do E-Stop podem ser comutados fechando as linhas que estavam interrompidas pelos relés, uma fotografia da caixa instalada no veículo encontra-se na figura 3.17.

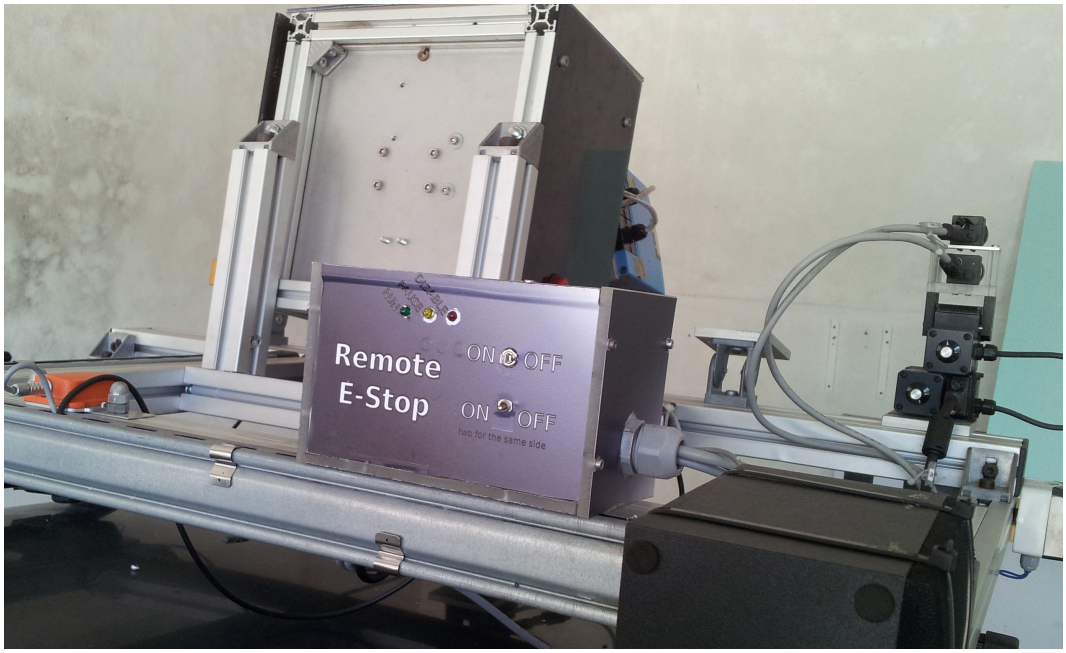


Figura 3.17: Sistema E-Stop instalado no carro

Capítulo 4

Sistema de monitorização

Neste capitulo apresenta-se todo procedimento efetuado para implementação do módulo de monitorização remota para o Atlascar.

4.1 O problema

O tema da monitorização aparece relacionado com o da segurança, daí a necessidade de monitorizar o estado da máquina remotamente. Interpreta-se como estado da máquina, o estado de algumas variáveis de medidas efetuadas a bordo do veículo como as rotações do motor, a velocidade, se este se encontra em modo de condução manual/automático entre muitos outros.

4.2 Estudo de soluções

A solução para implementação do sistema teria de passar pelos seguintes requisitos:

- o módulo teria interpretar as mensagens trocadas entre sistemas no veículo.
- possibilidade de transmissão de informação durante a operação do veículo e em "qualquer"local.
- a informação transmitida teria de ser acessível de qualquer local.
- teria de ser possível rever historiais de informação (registos).

Depois dos requisitos definidos iniciou-se uma pesquisa para encontrar uma solução capaz de satisfazer todos os requisitos. Foi encontrada uma solução, que pode ser consultada na secção 1.3.5, e que poderia ser adaptada para o problema em questão. Mas essa abordagem foi abandonada, pois tratava-se de um sistema muito complexo e a sua adaptação para o problema em causa seria uma tarefa árdua. Assim, optou-se por um outro tipo de abordagem, inspirado no sistema descrito anteriormente, mas adaptado aos objetivos propostos.
O sistema implementado tem seguinte funcionamento:

- Há um módulo **ROS** que subscreve as mensagens no carro, escrevendo os dados numa base de dados remota.

- Há uma pagina web que acede a essa base de dados, apresentando os valores da variáveis do sistema.

O esquema do sistema descrito está representado na figura4.1.

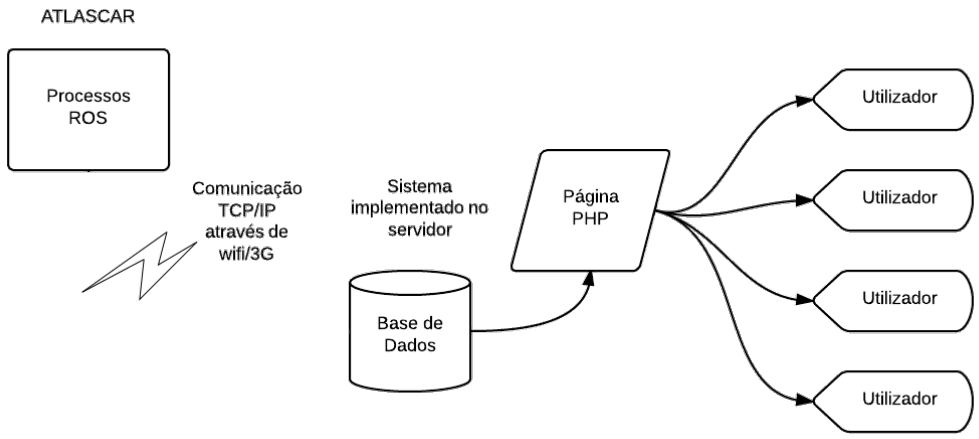


Figura 4.1: Esquema de implementação da monitorização do ATLASCAR

4.3 Módulo ROS

O módulo de **ROS** criado dá pelo nome de *remote_monitor*, dependendo da biblioteca *database_interface*. Essa biblioteca é responsável pela comunicação do módulo com a base de dados *Postgresql*.
O módulo criado consiste na subscrição de mensagens que são publicadas no sistema do Atlascar. As mensagens subscritas são as seguintes:

atlascar_base/ManagerSatus.h Nesta mensagem são retirados grande parte das variáveis relativas ao estado da maquina.

gps_common/GPSFix.h Esta mensagem contem variáveis relativas ao posicionamento do veículo, sendo essas obtidas a partir de um sensor *GPS* instalado na parte superior do veículo

As mensagens anteriormente descritas contêm variáveis que serão usadas para monitorização do veiculo, que se encontram na tabela 4.1.

Variavés subscritas		
Mensagem	**Variáveis**	**Descrição**
ManagerStatus	**speed**	Velocidade instantânea
	rpm	Rotações por minuto do motor
	db_time	data de inserção YYYY-MM-DD
	atlas_time	hora de inserção hh:mm:ss
	throttle_press	pressão exercida no pedal do acelerador
	brake_press	pressão exercida no pedal do travão
	clutch_press	Pressão exercida no pedal da embraigem
	throttle	atuação do acelerador
	brake	atuação do travão
	clutch	atuação da embraiagem
	gear	mudança do veículo
	steering_wheel	posição do volante
	auto_throttle	modo de atuação do acelerador
	auto_brake	modo de atuação do travão
	auto_clutch	modo de atuação da embraiagem
	auto_steering	modo de atuação da direção
	auto_iginition	modo de atuação da ignição
	hanbrake	atuação do travão de mão
	emergency	estado da emergência
	lights_minimum	estado das luzes de presença
	lights_medium	estado das luzes de cruzamento
	lights_high	estado das luzes de estrada
	lights_left	estado das luzes de sinalização de mudança de direção à esquerda
	lights_right	estado das luzes de sinalização de mudança de direção à direita
	lights_brake	estado das luzes de sinalização de travagem
	lights_reverse	estado das luzes de sinalização de marcha atrás
	lights_warning	estado da luz rotativa amarela de sinalização
	horn	estado da buzina
GPS_Fix	**lat_gps**	latitude da posição
	long_gps	longitude da posição
	atl_gps	altitude
	track_gps	direção do veículo tendo como referência o Norte
Vazios	**empty_field3**	campo em aberto para futuras variáveis
	empty_field4	campo em aberto para futuras variáveis

Tabela 4.1: Lista de variáveis subscritas e respetivas mensagens

Depois de todas as variáveis recebidas é iniciada a comunicação com a base de dados que se encontra alojada num servidor de IP fixo.
Neste caso em particular todo o sistema assenta sobre um servidor de base de dados *Postgresql* em *Linux*, com o qual a biblioteca implementada se encontra preparada para comunicar. A biblioteca usada encontra-se nos elementos de software desta dissertação, pois foi necessário fazer uma cópia local da mesma, visto não existir, até ao momento, nenhum repositório com para implementação desta biblioteca [4].

4.3.1 Funcionamento do Módulo ROS

A estrutura de funcionamento do módulo criado é a seguinte:

- Subscrição das mensagens do estado do carro;
- A ligação à base de dados é estabelecida;
- É efetuada a verificação de existência de elementos na base de dados remota, em caso afirmativo, é efetuada uma cópia, para um ficheiro no pc remoto, e depois a base de dados é apagada;
- Depois da verificação da ligação e do estado da base de dados, é dado inicio à escrita das variáveis na base de dados;
- Quando o programa é interrompido é criada uma cópia da base de dados para um ficheiro *.csv*, no computador remoto, o ficheiro é guardado na pasta */opt/atlas-car/*(no servidor da base de dados), o ficheiro criado tem a seguinte denominação *atlascar_YY_MM_DD_hh_mm_ss.csv*;
- Depois da cópia da base de dados, esta é limpa;
- O programa é terminado;

O esquema de funcionamento encontra-se representado na figura 4.2.

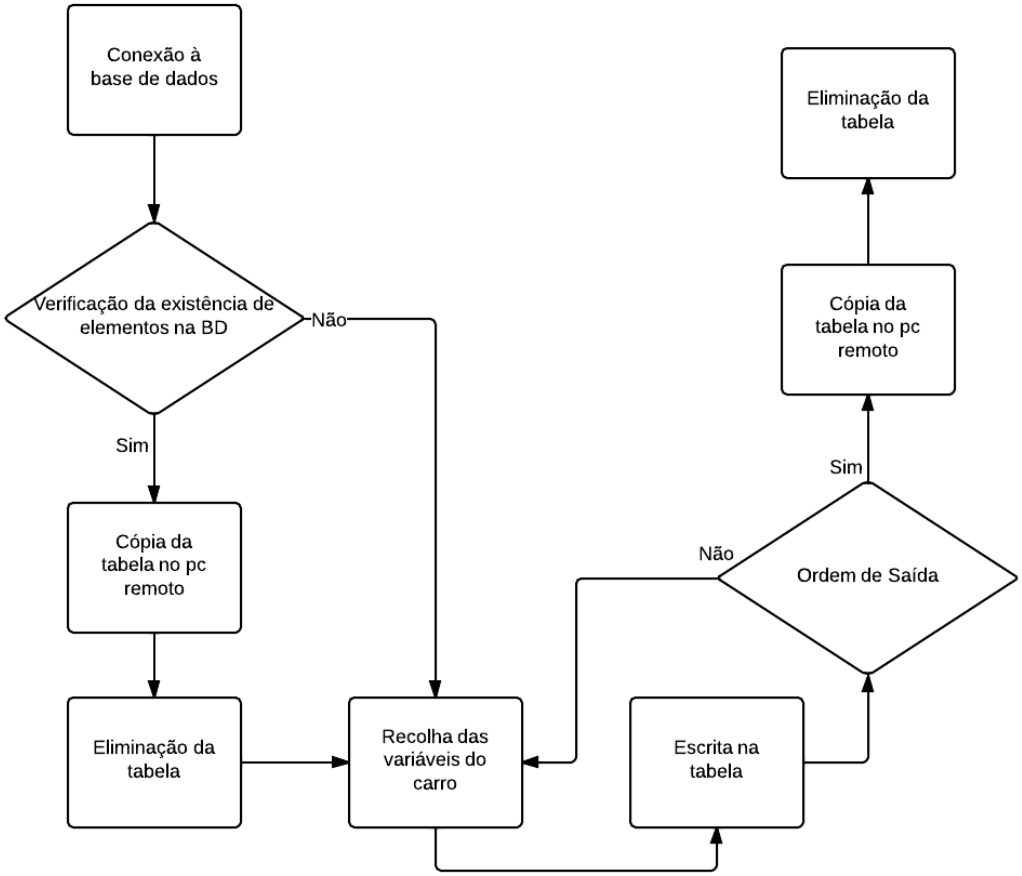


Figura 4.2: Esquema de simplificado do módulo ROS

O módulo encontra-se a escrever continuamente na base de dados, com um ciclo *while*, até que seja interrompido o programa, a condição de fecho é a interrupção forçada do módulo.

4.4 Funcionamento da base de dados

O sistema de armazenamento de dados está implementado numa tabela de uma base de dados em *Postgresql*. A tabela criada para este sistema tem o nome *data_table_atlas*, pode ser alterado, mas tem de se ter em atenção que o módulo ROS, com explicado na subsecção 4.3, e a página *web*, estão criados para comunicar com uma tabela com aquele nome. O modo de instalação e configuração do sistema *Postgresql* pode ser consultado no anexo E.

A tabela *data_table_atlas*4.2, contém os seguintes elementos:

Tabela data_table_atlas		
Variáveis	**Tipo**	**Descrição**
id	[int] PK	identificação de cada linha da tabela
rpm	[int]	Rotações por minuto do motor
speed	[double precision]	Velocidade instantanea
db_time	[date]	data de introdução da linha
atlas_time	[time without time zone]	hora de introdução da linha
throttle	[int]	atuação do acelerador
brake	[int]	atuação do travão
clutch	[int]	atuação da embraiagem
gear	[int]	mudança do veículo
steering	[double precision]	posição do volante
throttle_press	[double precision]	pressão exercida sobre o pedal do acelerador
brake_press	[double precision]	pressão exercida no pedal do travão
clutch_press	[double precision]	pressão exercida no pedal da embraiagem
auto_throttle	[int]	modo de atuação do acelerador
auto_brake	[int]	modo de atuação do travão
auto_clutch	[int]	modo de atuação da embraiagem
auto_steering	[int]	modo de atuação da direção
auto_iginition	[int]	modo de atuação ignição
hanbrake	[int]	atuação do travão de mão
emergency	[int]	estado da emergencia no veículo
lights_minimum	[int]	estado das luzes de presença
lights_medium	[int]	estado das luzes de cruzamento
lights_high	[int]	estado das luzes de estrada
lights_left	[int]	estado das luzes de mudança de direção à esquerda
çights_right	[int]	estado das luzes de mudança de direção à direita
lights_brake	[int]	estado das luzes de sinalização de travagem
lights_reverse	[int]	estado das luzes de sinalização de marcha atrás
lights_warning	[int]	estado da luz rotativa amarela de sinalização
horn	[int]	estado da buzina
lat_gps	[double precision]	latitude da posição
long_gps	[double precision]	longitude da posição
alt_gps	[double precision]	altitude a que se encontra o veículo
track_gps	[double precision]	direção do veículo tendo como referência o Norte
empty_field3	[int]	campo vazio para futuras variáveis
empty_field4	[int]	campo vazio para futuras variáveis

Tabela 4.2: Lista de variáveis subscritas e respetivas mensagens. *PK-Primary Key*

Para implementação da tabela 4.2 foi seguido o tutorial que se encontra em anexo ou no endereço:
`ros.org/wiki/sql_database/Tutorials/Installing%20a%20PostgreSQL%20Server`

4.5 Página web

A página *web* desenvolvida no âmbito do projeto apresentado tem de comunicar com a base de dados em *Postgresql*.
Esse procedimento só é possível com a adição ao código *html* de um script em *PHP* e de *JavaScript*, a organização das diferentes linguagens encontra-se exemplificado na figura 4.3. O código *PHP* é responsável por toda a parte de comunicação com a base de dados e obtenção dos campos da tabela. Os *scripts* em *JavaScript* são responsáveis pela parte dinâmica da página, ou seja, para fazer a apresentação das variáveis em intervalos de tempo regulares, bem como alguns elementos visuais presentes na página. Para uma página *web* estar acessível do exterior, esta tem de estar inserida num servidor http, para este caso em particular foi usado o *Apache2* para *LINUX*.

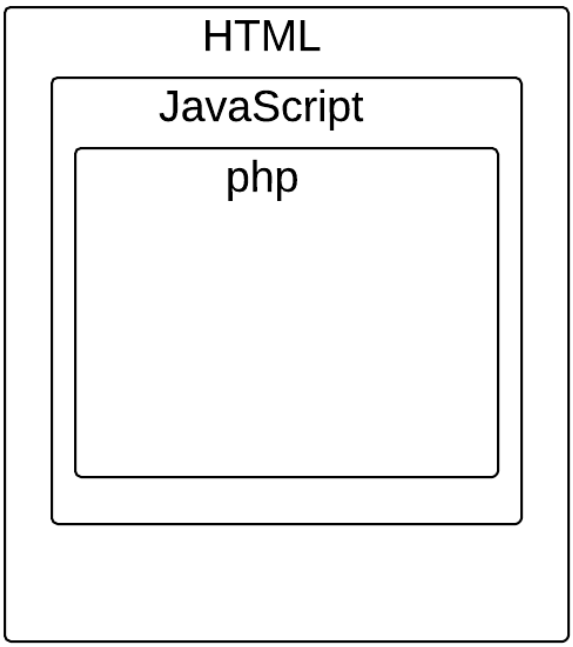


Figura 4.3: Esquema de simplificado da organização da página web

Em seguida serão apresentados os códigos e métodos usados para a criação da página de monitorização do Atlascar.

4.5.1 Código PHP

O código *PHP* implementado para esta página web, consiste num pequeno *script*, que tem o seguinte método de funcionamento:

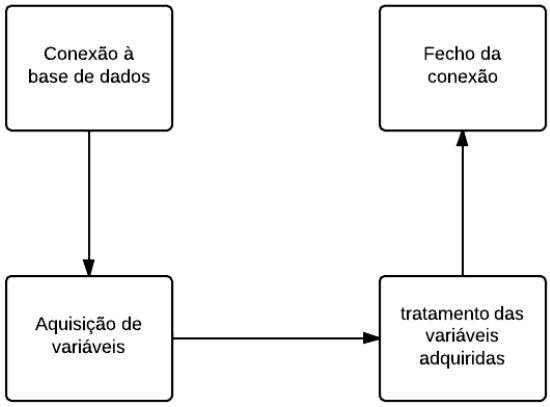


Figura 4.4: Esquema de simplificado do módulo PHP

De seguida apresenta-se o código usado na elaboração do comando PHP para obtenção dos dados da base de dados:

```
<?php
        <!--Login into database-->
        $host = "192.192.192.1";
        $user = "thisuser";
        $pass = "******";
        $db = "database_name";

        <!--database connect properties-->
        $con = pg_connect("host=$host dbname=$db user=$user password=$pass")
               or die ("Could not connect to server\n");

        <!--Query for database return the row that have the Max id -->
        $query = "SELECT * FROM data_table_atlas WHERE ID = ( SELECT MAX(ID)
                                                FROM data_table_atlas)";

        <!-- execute the query-->
        $rs = pg_query($con, $query) or die("Cannot execute query: $query\n");

        while ($row = pg_fetch_assoc($rs))
        {
               <!--
                this code is for getting the variables of the DB
               -->

               $speed=$row['speed'];
               $rpm=$row['rpm'];
               $lat_gps=$row['lat_gps'];
               $long_gps=$row['long_gps'];
               <!--
                The rest of the variables is like this
               $phpvariable=$row['name of row']
               -->
        }

        <!--Close DB connection-->
```

```
        pg_close($con);

        <!--Make a array with the values-->
        $val = array
        (
                "rpm" => $rpm,
                "speed" => $speed,
                "lat_gps" => $lat_gps,
                "long_gps" => $long_gps,
                <--"arrayvar"=> $phpvar-->
        );

        <!--export for Javascript-->
        print json_encode($val);
?>
```

Listing 4.1: Codigo PHP implementado

Para o reconhecimento das funções que se encontram no código *PHP* por parte do sistema operativo, uma biblioteca *PHP* tem de ser instalada, no caso em questão no sistema *UNIX*, usando o comando *sudo apt-get install php5-pgsql*.

4.5.2 Código *Javascript*

Depois da página PHP, subsecção 4.5.1, ser criada, a sua atualização não era possível sem que alguma ação por parte do utilizador fosse tomada, daí a introduçao do *Javascript* neste projeto. Além da adição de alguns *widgets* já implementados como o *Google maps* [8], o script criado em *Javascript* é responsável por invocar da função em PHP, para que os dados sejam os mais recentes que se encontram na base de dados; o código PHP é chamado a uma frequência de 5Hz, que é a taxa de atualização dos campos da página html.
Por outro lado, o *Javascript* também evita que a atualização da pagina tenha de ser total, podendo apenas alterar campos específicos da página web serem atualizados, os *div's*.

```
var map;//declare as global var
var marker;//declare as global var

//inicialization of Google maps
function initializeGoogleMap()
{
        // set latitude and longitude to center the map around in the begining
        var latlng = new google.maps.LatLng(37.77,-122.4);

        // set up the default options
        var myOptions =
        {
                zoom: 13,
                center: latlng,
                navigationControl: true,
                navigationControlOptions:
                {style: google.maps.NavigationControlStyle.DEFAULT,
                position: google.maps.ControlPosition.TOP_LEFT },
                mapTypeControl: true,
                mapTypeControlOptions:
```

```
                {style: google.maps.MapTypeControlStyle.DEFAULT,
                position: google.maps.ControlPosition.TOP_RIGHT },

                scaleControl: true,
                scaleControlOptions: {
                position: google.maps.ControlPosition.BOTTOM_LEFT
                },
                mapTypeId: google.maps.MapTypeId.ROADMAP,
                draggable: true,
                disableDoubleClickZoom: false,
                keyboardShortcuts: true
        };
        //define var map as a google maps object
        map = new google.maps.Map(document.getElementById("mapCanvas"), myOptions);

        if (false) {
                var trafficLayer = new google.maps.TrafficLayer();
                trafficLayer.setMap(map);
        }

        if (false) {
                var bikeLayer = new google.maps.BicyclingLayer();
                bikeLayer.setMap(map);
        }

        //set image as a new icon for marker in google maps
        var image = 'atlascar_icon.png';

        //add a new marker to the map
        marker = new google.maps.Marker(
        {
                 position: latlng,
                 map: map,
                 name: "atlascar",
                icon: image
        });
}


function timeout()
{

        // This event fires every 200ms
        $.ajax(
        {
                // ajax call starts
                url: 'serverside.php', // JQuery loads serverside.php
                dataType: 'json', // Choosing a JSON datatype
                success:function(data) // Variable data constains the data we get
                                        //from serverside
                {
                        //pick the var rpm sent from php
                        var x=document.getElementById("rpm");
                        x.innerHTML=data.rpm;
                        //pick the var speed sent from php
                        var x1=document.getElementById("speed");
                        x1.innerHTML=data.speed;
                        //the same way for all var
```

```
                        var lat=document.getElementById("lat");
                        lat.innerHTML=data.lat_gps;

                        var long=document.getElementById("long");
                        long.innerHTML=data.long_gps;
                        //keep the same structure for all the var


                        //this is to set a var with the coordinates sent by th GPS
                        //to send to google maps script
                        var atlas_pos = new google.maps.LatLng(
                               parseFloat(data.lat_gps), parseFloat(data.long_gps));
                        //update position of the marker
                        marker.setPosition(atlas_pos);
                        //update the position of the center of the map
                        map.setCenter(atlas_pos);
                }
        });

        return false;
}
//this sets the ajax code fire every 200ms
$(document).ready(function()
{

        setInterval(timeout, 200);
});
```

Listing 4.2: Codigo JavaScript implementado no sistema

4.5.3 Código *html*

A página html usada foi um template já existente, para manter a uniformidade de apresentação, ficando a pagina de monitorização com o mesmo aspeto que a pagina de apresentação do laboratório.
Dentro dessa pagina apenas foram criados os campos necessários para apresentação das variáveis a monitorizar. Para os campos poderem ser atualizados na página *html*, o *script* criado em *Javascript* teve de ser adicionado à página *html*, isso é efetuado pela adição de um *header*, que contém os *scripts* necessários para o correto funcionamento do *Javascript*. Os *headers* inseridos na página *html* são os seguintes:

```
<!-- header to include the made script called scrit_ze.js -->
<script src="//lars.mec.ua.pt/script_ze.js" type="text/javascript"></script>
<!-- This is the title of the page -->
<title>Atlascar LIVE</title>
<!-- script from google maps -->
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js"
    type="text/javascript"></script>
<script type='text/javascript'
    src='http://maps.google.com/maps/api/js?sensor=true'></script>
```

A página criada em html tem o seguinte aspeto:

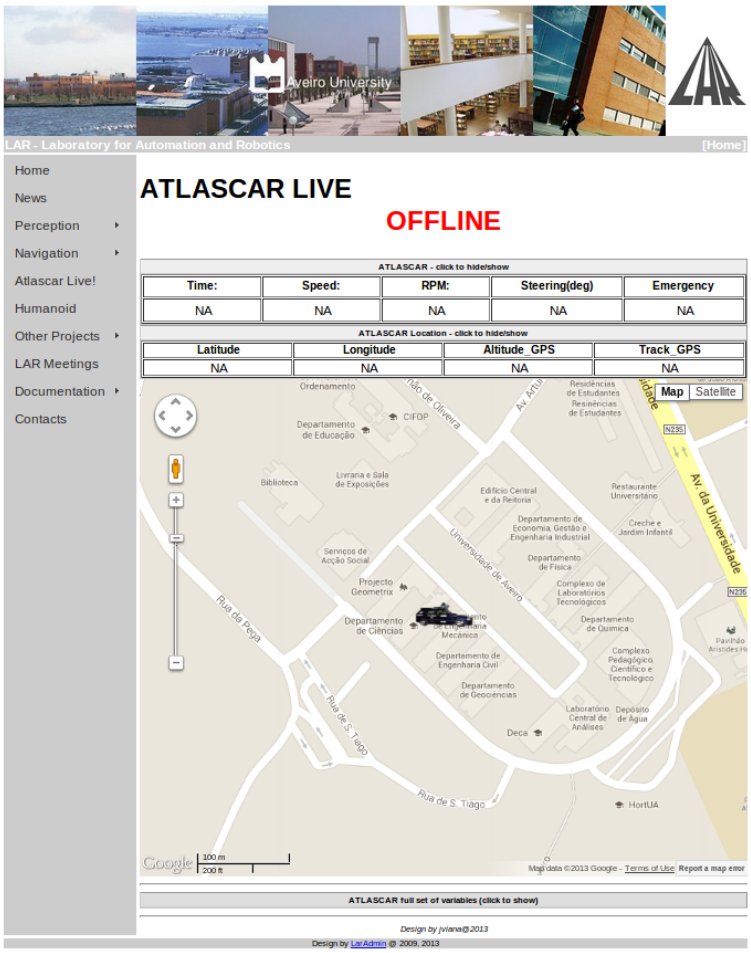


Figura 4.5: Página web criada para monitorização do Atlascar

4.6 Sistema de monitorização implementado

Depois de todos os procedimentos descritos anteriormente o sistema foi implementado com sucesso podendo a página em questão ser visualizada no endereço `lars.mec.ua.pt/atlascar_live` ou recorrendo ao QR code figura:4.6.

Figura 4.6: QR code para acesso ao site de monitorização do ATLASCAR

Capítulo 5

Experiências e Discussão

Neste capitulo serão apresentados resultados de testes experimentais, o trabalho futuro que pode ser desenvolvido sobre esta dissertação e as conclusões

5.1 Experiências realizadas

5.1.1 E-Stop e consola de controlo

Durante o seu desenvolvimento o sistema de E-Stop foi submetido a testes para verificação da sua fiabilidade. Os testes realizados serão descritos de seguida.

Teste de stress à comunicação
Durante a implementação do sistema, e enquanto este ainda se encontrava na bancada em placa branca, por várias vezes foi deixado a comunicar durante várias horas, verificando ao fim desse tempo se tudo tinha corrido na bem. Durante os testes, as ordens do comando remoto iam sendo alteradas, sendo durante alguns períodos comutadas repetidamente para introduzir erro ao sistema, comportando-se sempre de forma normal, comutando os leds como era suposto.

Teste de duração de bateria
O sistema foi deixado durante cerca de 3 horas a comunicar com a unidade do comando alimentada com a bateria, não baixando a tensão aos terminais da bateria dos 3.6V.

Teste de alcance da comunicação
Este tipo de teste foi o que sofreu mais repetições, pois trata-se do elemento mais importante do sistema até que distância nos podemos afastar do carro, continuando a ter controlo sobre o sistema.
Assim ainda em laboratório foram feitos alguns testes em interior, com muito ruído, pois todo o departamento tem cobertura de rede *wifi* o que poderia provocar interferências. Também os elementos estruturais do edifício provocam falhas, mas mesmo com estas interferências todas e no interior do departamento, foi atingido distâncias de comunicação de cerca de 30m.
Em campo aberto e depois de o sistema se encontrar instalado no veículo foram feitos vários testes de alcance. Em seguida são apresentadas imagens de locais

diferentes onde foram efetuados os testes aparecendo evidenciado o raio de ação que foi conseguido com o comando.

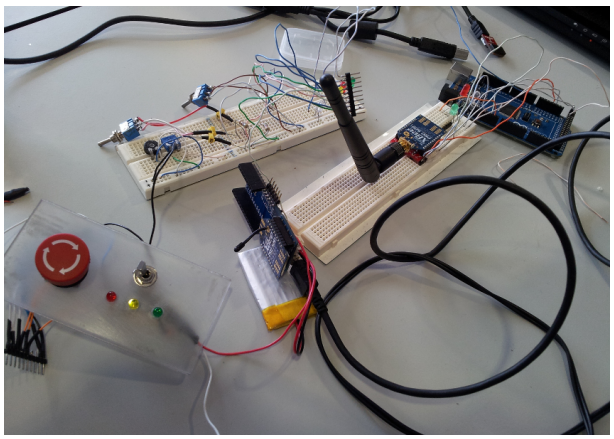


Figura 5.1: Montagem no laboratório para testes

Depois do sistema de E-Stop concluído e instalado no veículo foram efetuados alguns testes de alcance com o sistema em funcionamento. O teste efetuado foi o seguinte ativar e desativar o *PAUSE* repetidamente enquanto me ia deslocando em torno do carro e ao mesmo tempo afastando, verificando o ponto em que o sistema deixava de responder. Os testes foram efetuados em dois locais diferentes, um local na ESSUA que a área de alcance se encontra representada na figura 5.2 e o outro teste foi efetuado no novo Departamento da Universidade de Aveiro, ao lado da Casa do Estudante, nas figuras 5.3 e 5.4 pode ser visualizada a distância de atuação do E-Stop.

Figura 5.2: Imagem com a área de alcance o E-Stop

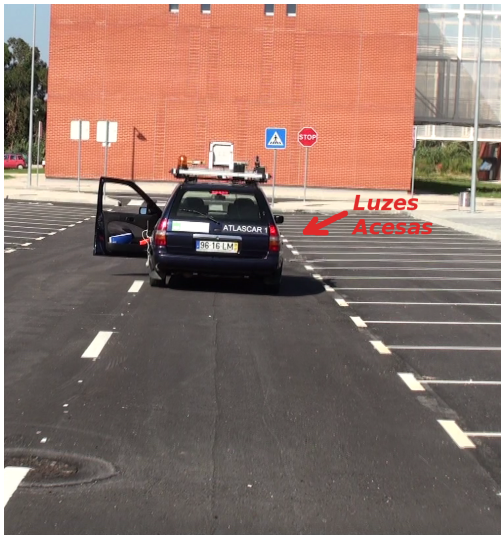


Figura 5.3: *PAUSE* ativo remotamente

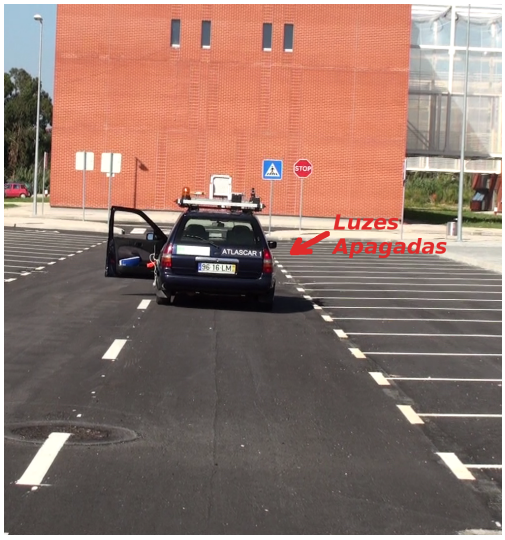


Figura 5.4: *RUN* ativo remotamente

Teste do modo DISABLE

No campo foram efetuados testes de comportamento do veículo quando ativo o modo *DISABLE*. Assim quando pressionado o cogumelo na consola de controlo o carro desliga-se instantaneamente e inicia a rotina de emergência(a mesma que é ativa com o modo *PAUSE*), o veículo só pode voltar a ser operado se o sistema for reiniciado através do corte de energia, os testes foram efetuados e validados no veículo.

Análise de falha

Em seguida serão apresentados alguns exemplos de falhas e de que forma o sistema de E-STOP responde a essas situações que poderiam comprometer o seu normal funcionamento.

Perda de comunicações

Se durante 2 segundos (este tempo pode ser configurado alterando o código no ARDUINO) o sistema do carro não receber uma mensagem aceitável, este imediatamente ativa o modo PAUSE, caso receba entretanto uma mensagem completa cumpre a ordem recebida. Por outro lado, caso a consola remota não receba mensagens aceitáveis durante o mesmo período de tempo todos os led's existentes na consola são acesos, alertando o utilizador que algo de errado se passa com a receção de mensagens na consola remota.

Avaria do sistema remote E-STOP

Em caso de avaria do sistema este pode ser eliminado do controlo do veículo, fazendo a comutação dos switchs que se encontram no sistema do carro para o modo OFF, caso esteja no modo ON a atuação do veículo apenas é efetuada pela consola remota.

Envio de mensagens incompletas

Se o sistema do carro ou a consola remota receber mensagens que não sejam da mesma estrutura como as descrita na subsecção 3.3.2, durante 2 segundos, então medidas de segurança são ativadas do lado do veículo o modo PAUSE é ativo e na consola remota são acesos todos os led's da consola.

Botão DISABLE

O cogumelo de emergência da consola remota encontra-se ligado como NC(normaly closed), fazendo com que em caso de avaria ou queda do dispositivo, e algum elemento do sistema de DISABLE seja afetado o modo é ativo no carro.

Remote E-STOP ON/OFF

O utilizador tem o máximo de controlo sobre o veiculo na comutação do sistema E-STOP entre ON e OFF, este aviso serve apenas para alertar que se por lapso se quiser usar o sistema e os switch não estejam no modo ON o sistema não funciona. Pois os switchs estão colocados em paralelo com os relés de atuação, se estes fecharem o circuito os relés de atuação não o conseguem abrir.

5.1.2 Sistema de monitorização

Para o sistema de monitorização não foram realizadas experiências propriamente ditas apenas à medida que se ia desenvolvendo fizeram-se testes de funcionamento, que foram todos bem sucedidos.
Na figura 5.5 encontram-se duas imagens do site desenvolvido, as imagens foram captadas durante um ensaio do veículo.

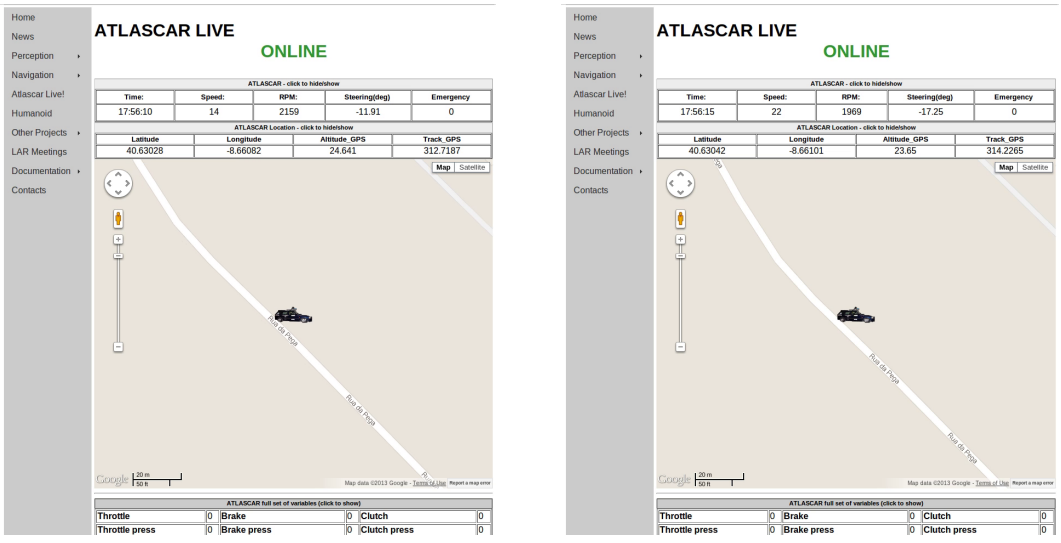


Figura 5.5: Imagens da página web desenvolvida durante ensaios do Atlascar

5.2 Discussão e conclusões

Os objetivos propostos para criação de um sistema de E-Stop e uma consola de controlo bem como a implementação de um sistema de monitorização foram cumpridos com exito, testados e validados em testes reais. Podendo toda a documentação e procedimentos efetuados ser consultada nos capítulos 3 e 4.
O sistema de E-Stop depois de pensada na solução para o problema, foi desenvolvido, criado e testado todo o sistema. Cumprindo com sucesso os objetivos de participação no ELROB, a imobilização do veículo e a possibilidade de desligar a fonte propulsão remotamente. O único elemento que não foi possível integrar na consola de controlo foi um LCD, no qual seriam apresentadas variáveis do estado da máquina enquanto a operação do veículo.
O sistema de monitorização inicialmente pensado assentaria sobre um sistema desenvolvido para ROS, RMS- Robot Management System, o seu modo de funcionamento pode ser consultado no Capítulo 1.3.5. Essa solução foi abandonada pela sua complexidade de funcionamento e de implementação, tratando-se de uma aplicação com funcionalidades excessivas para a finalidade pretendida, podendo num futuro próximo, já com o sistema RMS totalmente desenvolvido(pois ainda se encontra em fase de desenvolvimento), ser implementado no projeto, que para além da monitorização, o controlo do veículo seja possível. Assim o desenvolvimento de um sistema de monitorização foi iniciado usando, uma parte do sistema RMS, a biblioteca *database_interface*, que permite a comunicação de um sistema ROS com uma base de dados *PostgreSQL*. Desenvolvendo uma solução à medida para o problema em questão. Depois dos dados guardados na base de dados a sua apresentação aos utilizadores é efetuada através de uma página *web*, podendo todo o seu desenvolvimento ser consultado no capítulo 4.

5.3 Trabalho futuro

5.3.1 E-stop e consola de controlo

Adicionar o LCD à consola remota

O sistema encontra-se implementado com a possibilidade de um futura inclusão de um lcd no comando remoto. Como descrito anteriormente 3.3.2 a mensagem trocada do sistema do carro para o comando encontram-se com campos livres, onde podem ser adicionados campos para monitorização. Mas isso implica a comunicação com o carro.

5.3.2 Monitorização

Melhorar a interface de apresentação

A interface de apresentação das variáveis pode ser melhorada, deixando os valores de serem apresentados em tabelas passando a gráficos ou manómetros, o que pode tornar a interpretação de valores como a velocidade e RPM mais percetíveis.

Melhorar arquitetura da base de dados

Para que a cópia da base de dados não seja feita para um ficheiro local, e em vez disso serem criadas tabelas paralelas com os históricos de navegação do veículo, o

que permitiria ter acesso aos históricos de navegação através da página web, tendo o código da página ser adaptado.

Implementação do RMS

Com o software completamente desenvolvido e os sistemas preparados para o efeito penso que a implementação do RMS seria uma mais valia para o projeto, tanto a nível de monitorização e controlo do AtlasCar, mas também dos outros robôs do laboratório como os da figura 1.1. Assim seria possível numa só interface gerir e monitorizar os diferentes projetos existentes no laboratório.

Referências

[1] Arduino. Arduino web page. http://http://www.arduino.cc/. [Online; Acedido a 18 de fevereiro de 2013].

[2] AutonomousCar. wiki. http://en.wikipedia.org/wiki/Autonomous_car, 2012. [Online; Acedido a 21 de maio de 2013].

[3] Bertozzi. Vision-based Automated Vehicle Guidance:the experience of the ARGO vehicle. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.25.911, 1998. [Online; Acedido a 08 de maio de 2013].

[4] Matei Ciocarlie. Database interface. http://ros.org/wiki/database_interface?distro=groovy, 2011. [Online; Acedido a 15 de abril de 2013].

[5] DARPA. DARPA Grand Challenge. http://archive.darpa.mil/grandchallenge05/overview.html, 2005. [Online; Acedido a 07 de maio de 2013].

[6] Digi. Xbee RF Modules. http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/zigbee-mesh-module/. [Online; Acedido a 18 de fevereiro de 2013].

[7] Les Earnest. Stanford Cart. http://www.stanford.edu/~learnest/cart.htm, 2012. [Online; Acedido a 21 de maio de 2013].

[8] Google. Google Maps API v3. https://developers.google.com/maps/documentation/javascript/examples/, 2011. [Online; Acedido a 15 de maio de 2013].

[9] Navlab5. Navlab 5. http://www.cs.cmu.edu/image_archive/07robot_hof/navlab5/navlab5_2.jpg, 2012. [Online; Acedido a 07 de maio de 2013].

[10] Nevada. Driveless Nevada. http://www.greencarcongress.com/2011/06/ab511-20110625.html, 2011. [Online; Acedido a 23 de maio de 2013].

[11] Joel Pereira. *Estacionamento autónomo usando perceção 3D.* Tese de mestrado, Universidade de Aveiro, 2012.

[12] Tiago Rocha. *Piloto Automático para Controlo e Manobras de Navegação do Atlas-Car.* Tese de mestrado, Universidade de Aveiro, 2011.

[13] J. Schmidhuber. Prof. Schmidhuber's highlights of robot car history. http://www.idsia.ch/~juergen/robotcars.html, 2011. [Online; Acedido a 07 de maio de 2013].

[14] Frank E. Schneider. The European robot trial. http://http://www.elrob.org/, 2006. [Online; Acedido a 24 de março de 2013].

[15] Russell Toris. The Robot Management System. http://www.ros.org/wiki/rms, 2013. [Online; Acedido a 07 de maio de 2013].

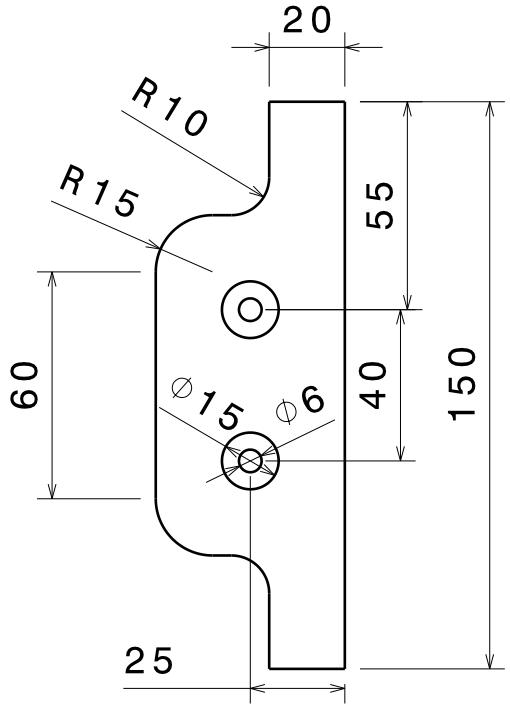
[16] VaMoRs. VaMoRs van. http://www.unibw.de/lrt8/institut/mitarbeiter/prof_wuensche/vamors/pic1_preview, 2011. [Online; Acedido a 07 de maio de 2013].

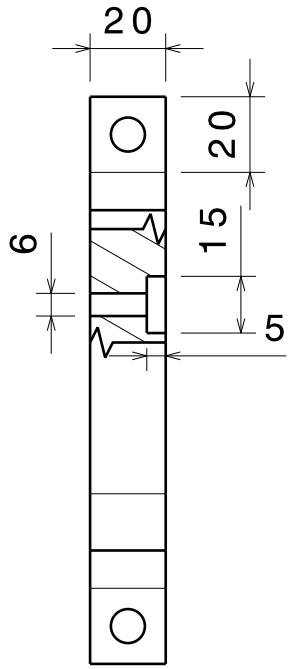
[17] VaMoRs-P. VaMoRs-P Mercedes S. http://www.flickr.com/photos/reinholdbehringer/1919686531/in/photostream/, 2011. [Online; Acedido a 07 de maio de 2013].

[18] Tom Vanderbilt. Autonomous Cars Through the Ages. http://www.wired.com/autopia/2012/02/autonomous-vehicle-history/?pid=1585, 2012. [Online; Acedido a 23 de maio de 2013].

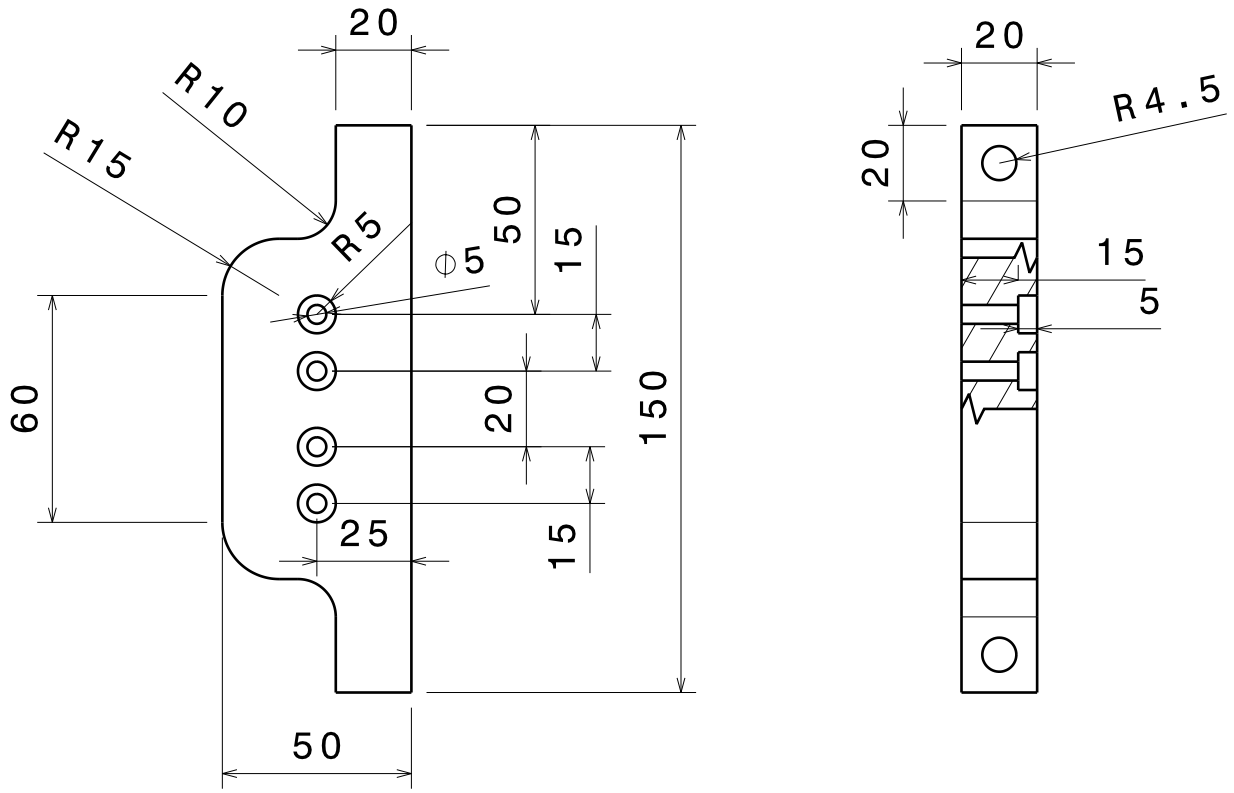
Apêndice A

Caixas de atuação Desenhos

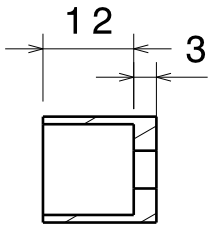


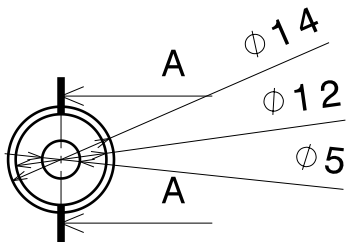


DESIGNED BY: José Viana DATE: 19-03-2012			I _
CHECKED BY: XXX DATE: XXX			H _
			G _
			F _
SIZE A4		DASSAULT SYSTEMES	E _
			D _
SCALE 1:2	WEIGHT (kg) 0,85	DRAWING NUMBER batente	SHEET 1/1 C _
			B _
This drawing is our property; it can't be reproduced or communicated without our written agreement.			A _

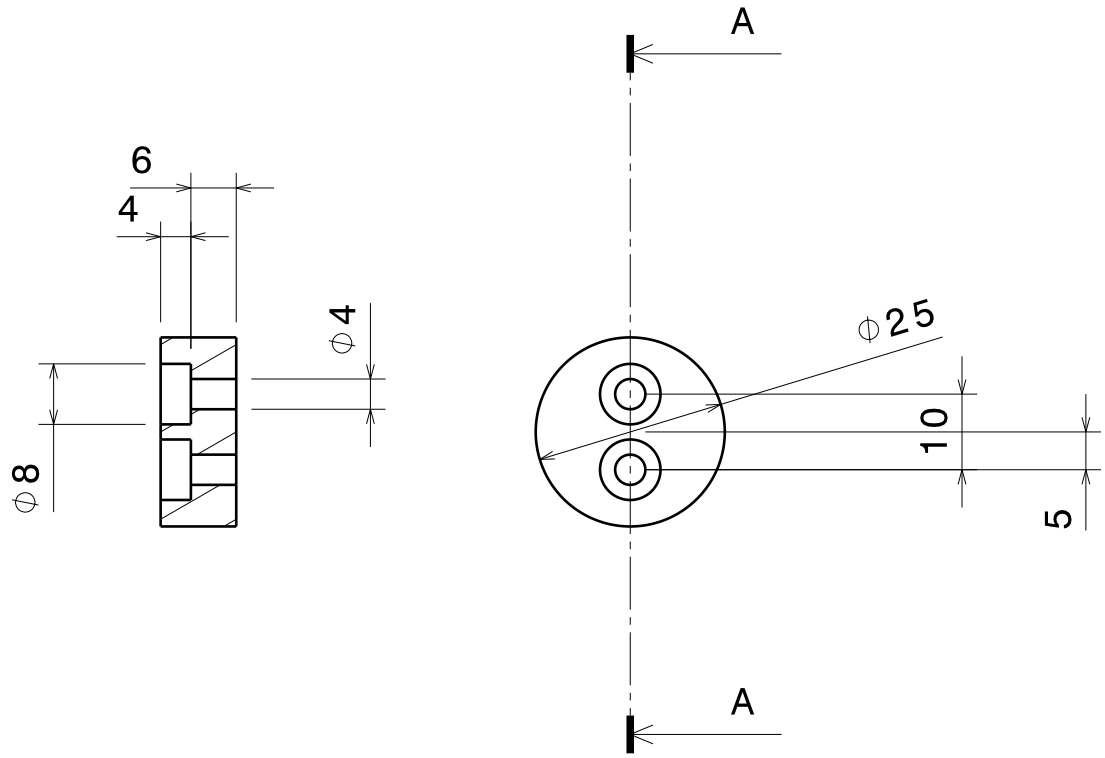


DESIGNED BY: José Viana DATE: 19-03-2012			I _
CHECKED BY: XXX DATE: XXX			H _
			G _
			F _
SIZE A4		DASSAULT SYSTEMES	E _
			D _
SCALE 1:2	WEIGHT (kg) 0,85	DRAWING NUMBER batente 4 furos	SHEET 1/1 C _
			B _
This drawing is our property; it can't be reproduced or communicated without our written agreement.			A _





DESIGNED BY: José Viana DATE: 26-03-2012			I _
CHECKED BY: XXX DATE: XXX			H _
			G _
			F _
SIZE A4		DASSAULT SYSTEMES	E _
			D _
SCALE 1:1	WEIGHT (kg) 0,00	DRAWING NUMBER CABOS	SHEET 1/1 C _
			B _
This drawing is our property; it can't be reproduced or communicated without our written agreement.			A _



DESIGNED BY: José Viana DATE: 26-03-2012				I _
CHECKED BY: XXX DATE: XXX				H _
				G _
				F _
SIZE A4		DASSAULT SYSTEMES		E _
				D _
SCALE 1:1	WEIGHT (kg) 0,01	DRAWING NUMBER FIM CABOS	SHEET 1/1	C _
				B _
This drawing is our property; it can't be reproduced or communicated without our written agreement.				A _

Apêndice B

Datasheets

PC829 Series

High Density Mounting Type Photocoupler

※ TÜV (VDE0884) approved type is also available as an option.

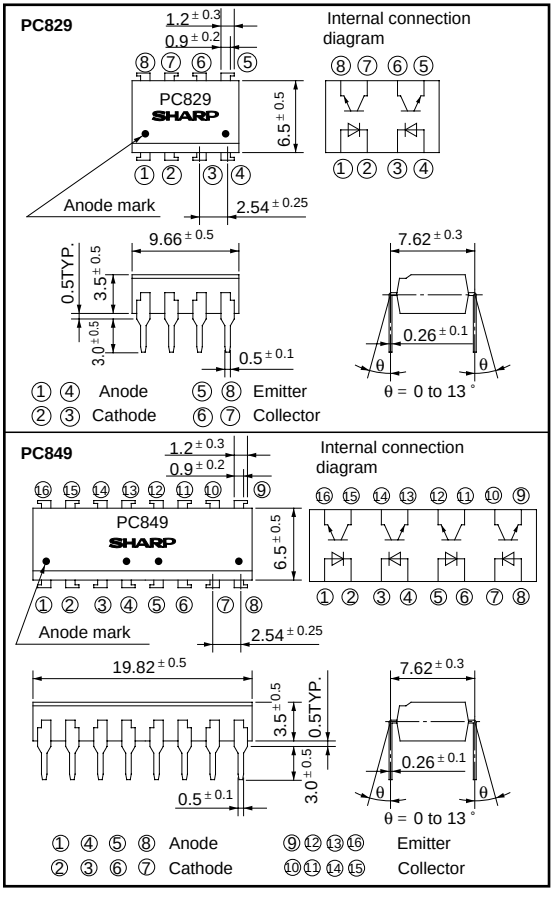
■ Features

1. Symmetrical terminal configuration
   PC829 : 2-channel type
   PC849 : 4-channel type
2. High current transfer ratio
   (CTR : MIN. 50% at I_F = 5mA, V_{CE} = 5V)
3. High isolation voltage between input and output (V_{iso} : 5 000V_{rms})
4. Recognized by UL, file No. E64380

■ Applications

1. Telephone exchangers
2. Computer terminals
3. System appliances, measuring instruments
4. Signal transmission between circuits of different potentials and impedances

■ Outline Dimensions (Unit : mm)



■ Absolute Maximum Ratings (Ta= 25°C)

Parameter		Symbol	Rating	Unit
Input	Forward current	I_F	50	mA
	*1Peak forward current	I_{FM}	1	A
	Reverse voltage	V_R	6	V
	Power dissipation	P	70	mW
Output	Collector-emitter voltage	V_{CEO}	35	V
	Emitter-collector voltage	V_{ECO}	6	V
	Collector current	I_C	50	mA
	Collector power dissipation	P_C	150	mW
Total power dissipation		P_{tot}	170	mW
*2Isolation voltage		V_{iso}	5 000	V_{rms}
Operating temperature		T_{opr}	- 25 to + 100	°C
Storage temperature		T_{stg}	- 40 to + 125	°C
*3Soldering temperature		T_{sol}	260	°C

*1 Pulse width<=100μs, Duty ratio : 0.001
*2 40 to 60% RH, AC for 1 minute
*3 For 10 seconds

■ Electro-optical Characteristics

(Ta= 25˚C)

Parameter			Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Input	Forward voltage		V_F	I_F = 20mA	-	1.2	1.4	V
	Peak forward voltage		V_{FM}	I_{FM} = 0.5A	-	-	3.0	V
	Reverse current		I_R	V_R = 4V	-	-	10	μA
	Terminal capacitance		C_t	V = 0, f = 1kHz	-	30	250	pF
Output	Collector dark current		I_{CEO}	V_{CE} = 20V, I_F = 0	-	-	10^{-7}	A
Transfer characteristics	Current transfer ratio		CTR	I_F = 5mA, V_{CE} = 5V	50	-	400	%
	Collector-emitter saturation voltage		$V_{CE(sat)}$	I_F = 20mA, I_C = 1mA	-	0.1	0.2	V
	Isolation resistance		R_{ISO}	DC500V, 40 to 60% RH	5 x 10^{10}	10^{11}	-	Ω
	Floating capacitance		C_f	V = 0, f = 1MHz	-	0.6	1.0	pF
	Cut-off frequency		f_c	V_{CE} = 5V, I_C = 2mA, R_L = 100Ω, - 3dB	-	80	-	kHz
	Response time	Rise time	t_r	V_{CE} = 2V, I_C = 2mA, R_L = 100Ω	-	4	-	μs
		Fall time	t_f		-	3	-	μs

Fig. 1 Forward Current vs. Ambient Temperature

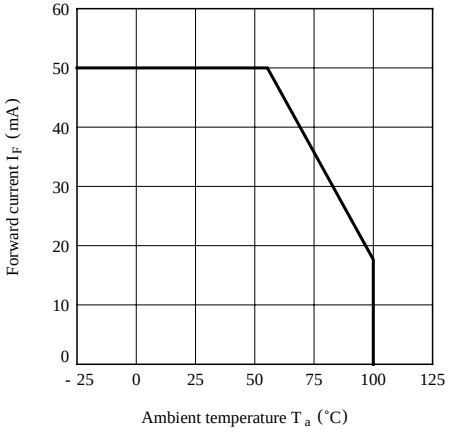


Fig. 2 Collector Power Dissipation vs. Ambient Temperature

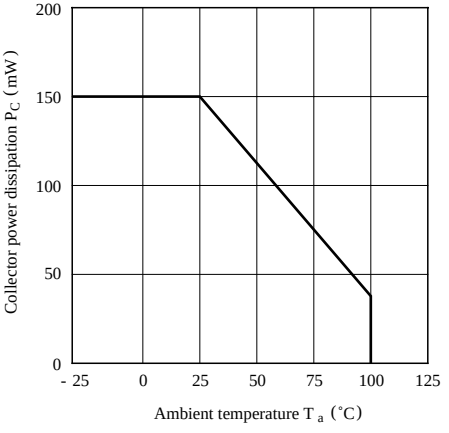


Fig. 3 Peak Forward Current vs. Duty Ratio

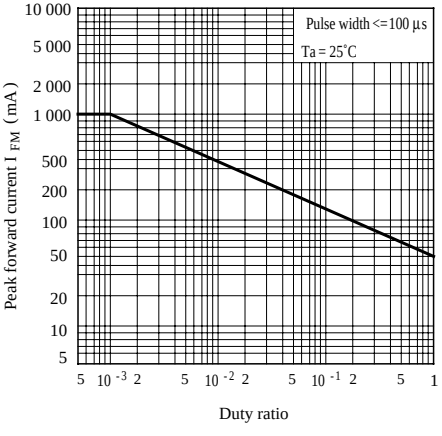


Fig. 4 Forward Current vs. Forward Voltage

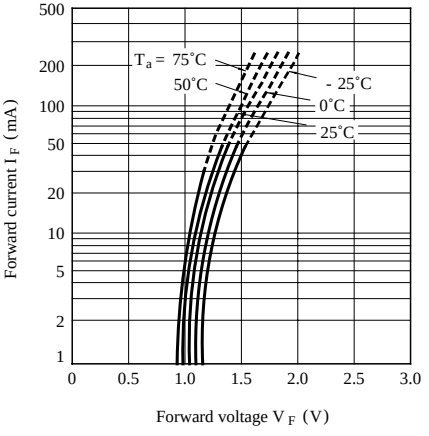


Fig. 5 Current Transfer Ratio vs. Forward Current

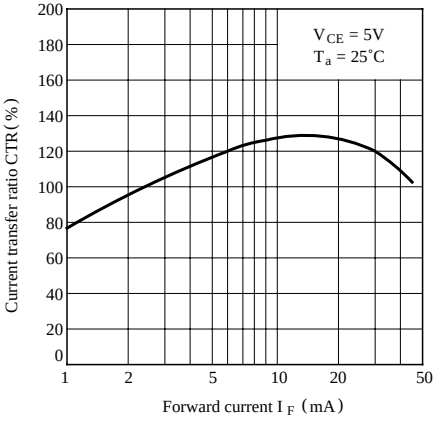


Fig. 6 Collector Current vs. Collector-emitter Voltage

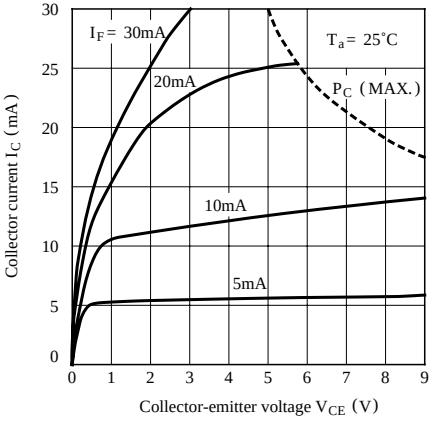


Fig. 7 Relative Current Transfer Ratio vs. Ambient Temperature

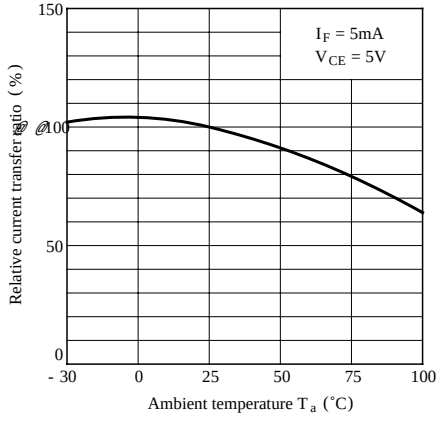


Fig. 8 Collector-emitter Saturation Voltage vs. Ambient Temperature

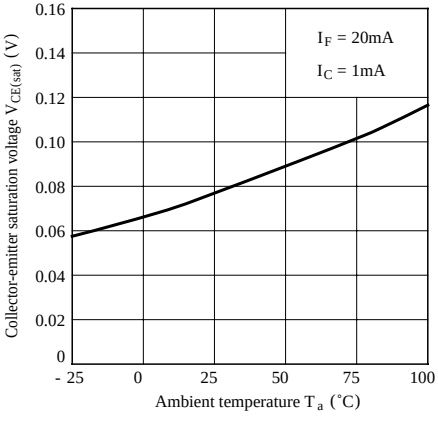


Fig. 9 Collector Dark Current vs. Ambient Temperature

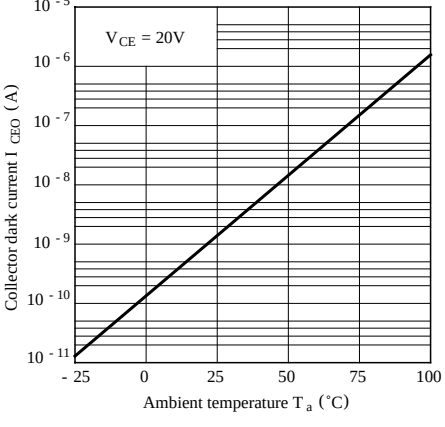


Fig.10 Response Time vs. Load Resistance

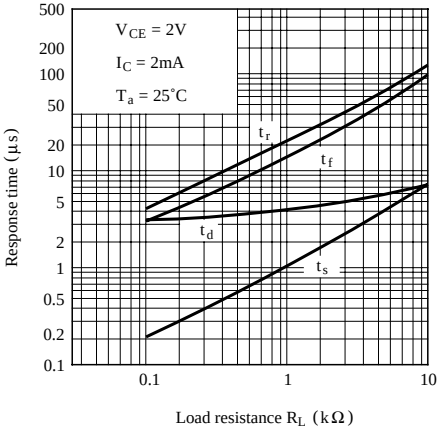
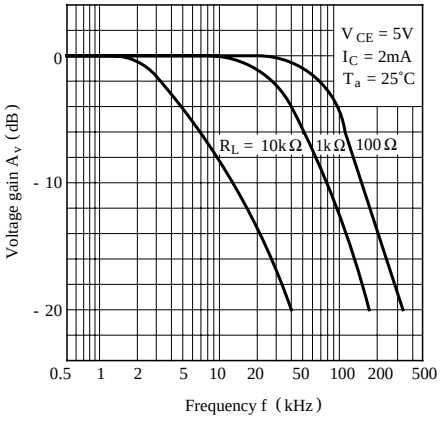


Fig.11 Frequency Response



Test Circuit for Response Time

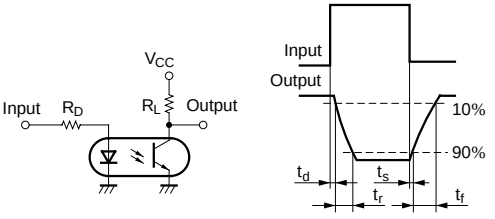
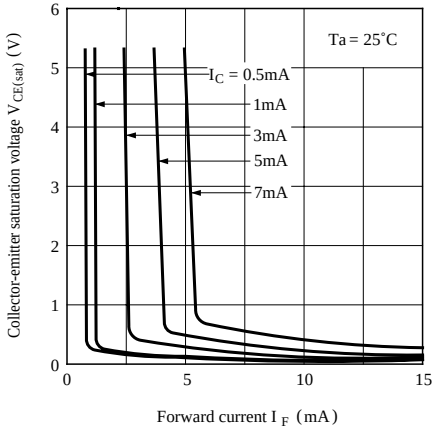
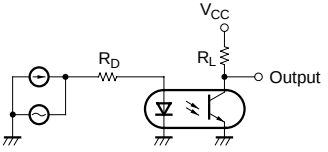


Fig.12 Collector-emitter Saturation Voltage vs. Forward Current



Test Circuit for Frepuency Response



● Please refer to the chapter "Precautions for Use"

Slim PCB Relay PCN

- **1 pole 3 A, 1 NO contact**
- **Only 5 mm wide**
- **3 A switching current**
- **Load range 1 mA up to 3 A**
- **Sensitive coil 120 mW**
- **Allows high function-/packing density**
- **Cadmium-free contacts**
- **Z type with reinforced insulation**
- **RoHS compliant (Directive 2002/95/EC) as per product date code 0424**

F0258-A

Applications

Centralized and decentralized heating control, extremely narrow interface elements, interface technology, timers, PLC's, I/O modules, I/O-ports

Approvals

VDE REG.-Nr. 6166; cULus E82292
Technical data of approved types on request

Contact data

Contact configuration	1NO
Contact set	bifurcated contact
Type of interruption	micro disconnection
Rated current	3 A
Rated voltage / max.switching voltage AC	240/277 VAC
Limiting continuous current	3 A
Maximum breaking capacity AC	750 VA
Limiting making capacity, max 4 s, duty factor 10%	5 A
Contact material	AgNi90/10
Minimum contact load	5V / 1mA
Mechanical endurance	$20x10^6$ cycles
Rated frequency of operation with / without load	10 / 1200 min^{-1}

Contact ratings

Type	Load	Cycles
PCN-1..D3M.(-,H).(-,Z)	3 A, 250 VAC resistive, 70°C, 20 cycles/min, EN61810-1	$1x10^5$
PCN-1..D3M.(-,H).(-,Z)	3 A, 30 VDC, resistive, 70°C, 20 cycles/min, EN61810-1	$1x10^5$
PCN-1..D3M.(-,H).(-,Z)	Pilot Duty B300 240 VAC, UL508	$6x10^3$
PCN-1..D3M.(-,H).(-,Z)	Pilot Duty B300 120 VAC, UL508	$6x10^3$

Coil data

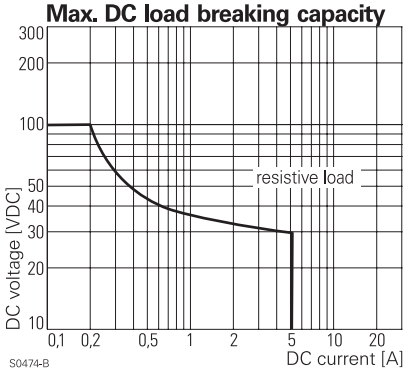
Rated coil voltage range DC coil	DC5...24 VDC
Coil power DC coil	typ 120 mW
Operative range	1

Coil versions, DC-coil

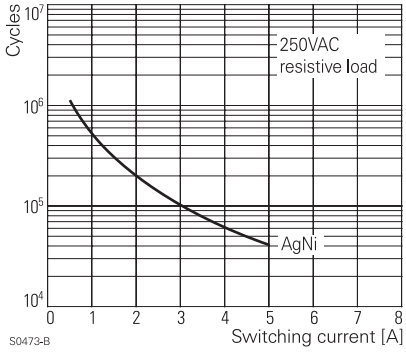
Coil code	Rated voltage VDC	Operate voltage VDC	Release voltage VDC	Coil resistance Ohm	Rated coil power mW
005	5	3.5	0.5	208±10%	120
006	6	4.2	0.6	300±10%	120
009	9	6.3	0.9	675±10%	120
012	12	8.4	1.2	1200±10%	120
024	24	16.8	2.4	4800±10%	120

All figures are given for coil without preenergization, at ambient temperature +23°C
Other coil voltages on request

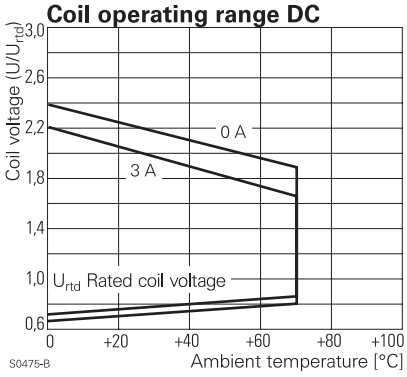
Max. DC load breaking capacity



Electrical endurance



Coil operating range DC



Datasheet Rev. EK1
Issued 2005/11
www.tycoelectronics.com
www.schrackrelays.com

Dimensions are in mm unless otherwise specified and are shown for reference purposes only.

Product specification according to IEC 61810-1. Product data, technical parameters, test conditions and processing information only to be used together with the 'Definitions' at schrackrelays.com in the 'Schrack' section.

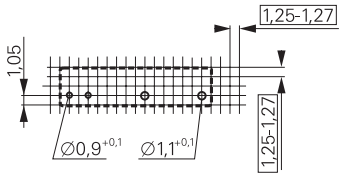
Specifications subject to change.

Slim PCB Relay PCN (Continued)

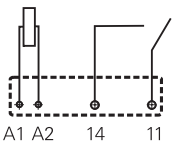
Insulation	
Dielectric strength coil-contact circuit	3000 V_{rms}
open contact circuit	750 V_{rms}
Clearance / creepage coil-contact circuit	≥ 3,5 / 3,5 mm
Material group of insulation parts	I
Tracking index of relay base	PTI 600
Insulation to IEC 60664-1	
Type of insulation coil-contact circuit	reinforced
open contact circuit	functional
Rated insulation voltage	277 V
Pollution degree	2
Rated voltage system	277 V
Overvoltage category as basic insulation	III
Overvoltage category as reinforced insulation	II

Other data	
RoHS - Directive 2002/95/EC	compliant as per product date code 0424
Flammability class according to UL94	V-0
Ambient temperature range	-30...+70 °C
Operate- / release time	typ. 5/2 ms
Bounce time NO / NC contact	< 1ms
Vibration resistance (function) NO / NC contact	10 g
Shock resistance (function) NO / NC contact	10 g
Shock resistance (destruction)	100 g
Category of protection	RTIII - wash tight
Mounting	pcb
Mounting position	any
Minimum mounting distance	0 mm
Resistance to soldering heat wash tight version	260°C / 5 s
Relay weight	3 g
Packaging unit	25 / 2000 pcs

PCB layout / terminal assignment
Bottom view on solder pins

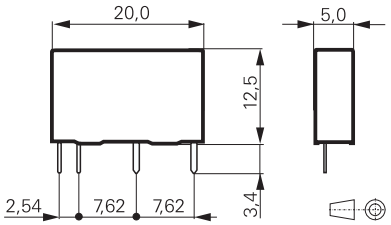


S0471-AA



S0471-AB

Dimensions



S0472-A

Product key

P	C	N	1			D	3	M	H	Z

Type
Number of contacts
1 1 pole
Coil
Coil code: please refer to coil versions table
Coil version
D standard 120 mW
Contact material
3 AgNi
Contact configuration
M 1 NO contact
Version
H wash tight
Insulation
Z tracking resistance of relay base PTI 600
Other types on request

Product key	Contacts	Coil	Coil	Contacts	Version	Part number
PCN-105D3MHZ	1-pole	5 VDC	standard	1 NO contact	wash tight	3-1461491-0
PCN-106D3MHZ		6 VDC	120 mW	AgNi	high insulation	3-1461491-1
PCN-112D3MHZ		12 VDC				3-1461491-3
PCN-124D3MHZ		24 VDC				3-1461491-6

Datasheet Rev. EK1
Issued 2005/11
www.tycoelectronics.com
www.schrackrelays.com

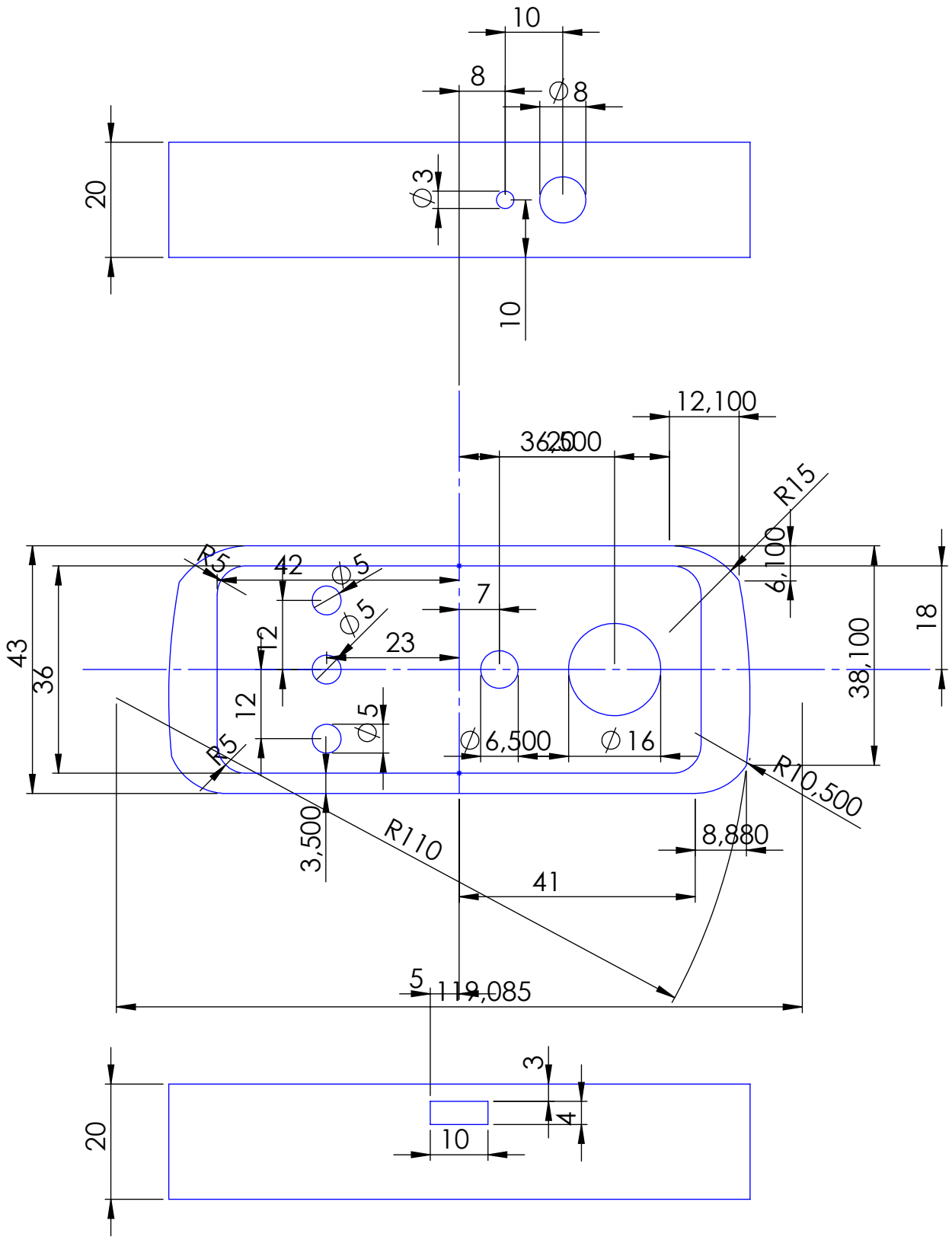
Dimensions are in mm unless otherwise specified and are shown for reference purposes only.

Product specification according to IEC 61810-1. Product data, technical parameters, test conditions and processing information only to be used together with the 'Definitions' at schrackrelays.com in the 'Schrack' section.

Specifications subject to change.

Apêndice C

Desenho técnico adaptador comando



UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	FINISH:		DEBUR AND BREAK SHARP EDGES		DO NOT SCALE DRAWING	REVISION
	NAME	SIGNATURE	DATE			TITLE:
DRAWN						
CHK'D						
APPV'D						
MFG						
Q.A				MATERIAL:	DWG NO. desenho_botoes	A4
				WEIGHT:	SCALE:1:1	SHEET 1 OF 1

Apêndice D

Configurar os módulos Xbee's

D.1 Funcionamento dos módulos XBee

Os módulos XBee são equipamentos que transmitem em RF(rádio frequência) e comunicam entre si; podendo comunicar apenas com um equimento e só com esse, ou em rede, existindo vários tipos de configurações para a rede. As redes podem ter configurações distintas:

- **MESH**(rede)- A *mesh network* é uma área local de trabalho que implementa um ou dois arranjos de comunicações, *full mesh topology* ou partial *mesh topology*.Na *full mesh topology*, cada elemento (*workstation* ou outro dispositivo) está conetado diretamente a cada elemento da rede (ou seja estão todos conetados uns aos outros). Na *partial mesh topology*, alguns dispositivos estão conetados a todos os outros, mas outros estão apenas conetados aos elementos com os quais trocam mais quantidade de informação. A *mesh network* é confiável e oferece redundância. Se por ventura algum dos elementos da rede deixar de funcionar, todos os outros continuam a comunicar entre si, diretamente ou através de outro ou outros intermediários da rede. As *Mesh networks* funcionam na perfeição quando os elementos se encontram dispersos e fora do alcance de uma linha comum.
- **STAR**(Broadcast) - apenas um elemento da rede comunica(coordinator) com todos os outros elementos, e estes apenas enviam informação para o *coordinator*.
- **Cluster tree**(*tree*) - um elemento principal da rede *coordinator*, comunica apenas com outro elemento, responsável por receber informação dos outros elementos e enviar a informação para o elemento principal.
- **Combinação de 3 tipos de montagem** - Combinação dos 3 tipos de configuração descritos anteriormente, todos os elementos comunicando entre si de forma diferente e integrados numa só rede de trabalho.

Em seguida uma breve explicação do que cada elemento representa na rede que integra.

Coordinator

Dispositivo responsável por toda a gestão da rede, elementos com quem vai comunicar e como estes comunicam ele, normalmente é o elemento que transfere informação para o exterior.

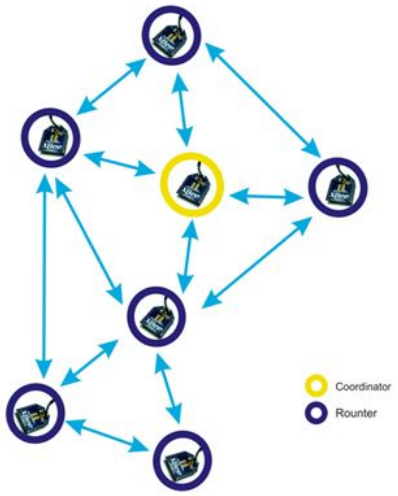


Figura D.1: Montagem *Mesh* de uma rede *Zigbee*

Figura D.2: Montagem *Star* de uma rede *Zigbee*

Router
é responsável por retransmitir a informação vinda do *coordinator* ou do *end device*. O router permite aumentar a distância entre entre *coordinator* e *end device*, permitindo assim a implementação de redes com dispositivos mais distanciados.

End device
é responsável por ler a informação de sensores ou outros dispositivos e permite também ligar/desligar actuadores.

D.1.1 Configuração dos módulos *Xbee's*

Nesta secção serão descritos os passos necessários para a correta configuração dos módulos *Xbee*, para aplicação no projeto desenvolvido no âmbito desta dissertação.
Em primeiro lugar o material necessário para a configuração dos dispositivos é o seguinte:

- dois módulos *Xbee's* - elementos a configurar para comunicarem um com o outro.

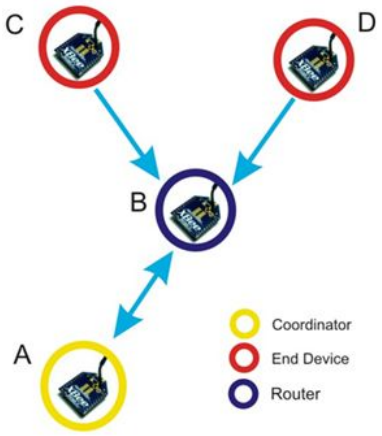


Figura D.3: Montagem *Cluster* de uma rede *Zigbee*

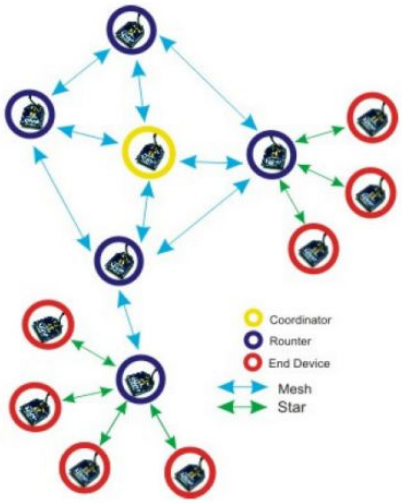
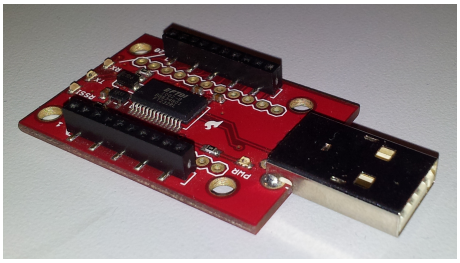


Figura D.4: Conjugação de todos os tipos de montagem de uma rede *Zigbee*

- adaptador para comunicação com os módulos D.1.1



- Software de configuração para os módulos *Xbee*, *X-CTU* Digi[6].

Já com todo o material disponível e com o software devidamente instalado, introduzir no computador o D.1.1, em seguida abrir o software *X-CTU*. Aparecendo a seguinte

janela D.5.

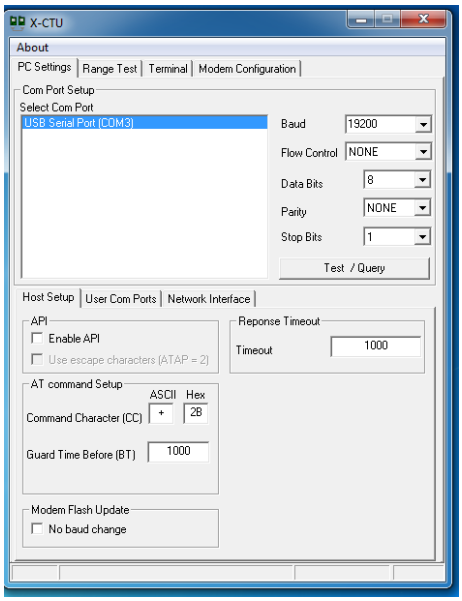


Figura D.5: Janela de início do programa *X-CTU*

Selecionar a porta onde se encontra conetado o sistema, neste exemplo como só existia um elemento conetado a uma porta capaz de suportar a comunicação com um módulo *Xbee*, é a única porta disponível para ser selecionada. Não esquecer também os parâmetros de comunicação, pois tradicionalmente os *Xbee's*, são configurados com uma *baud rate* de 9600bits/s, mas se esse parâmetro for alterado também o terá de ser nas configurações de comunicação.
Depois de todos os parâmetros configurados para iniciar a comunicação, pressionar o botão *Test/Query*. Se todos os parâmetros estiverem bem definidos deverá aparecer a janela. Em caso de erro voltar a pressionar o botão *Retry*, caso não apareçam os parametros relativos ao módulo que lá se encontra pode ter algum erro nos parâmetros de comunicação. Assim depois do modem ser encontrado anotar o valor apresentado no campo *Serial number*D.6, pois vai ser necessário.

Retirar o módulo e inserir o outro com o qual se vai fazer a montagem, e fazer novamente o procedimento descrito anteriormente, até obter o *Serial number* e anotá-lo.
Mudar para o separador *Modem Configuration*, selecionar o campo *Always Update Firmware*, em seguida pressionar o botão *Download new version*, que fará com que seja feito o Download da ultima versão desponivél de *Firmware*. Depois do *firmaware* atualizado pressionar o botão*Read*, sendo apresentados todos os campos de configuração como ilustrado na figura

Alterar os campos consoante a configuração desejada, neste caso como se trata de uma comunicação ponto a ponto, o *serial number* retirado do primeiro módulo deverá ser a primeira parte introduzida no campo **DH** 13A200 (isto para este caso em concreto) e a restante parte do código no campo **DL**D.8.

Poceder da mesma forma para o outro módulo *Xbee*, configurando-o corretamente, depois do segundo módulo configurado, abrir o separador *Terminal*, e verificar se o módulo se encontra conetado com o comando +++, ao qual deverá ter a resposta

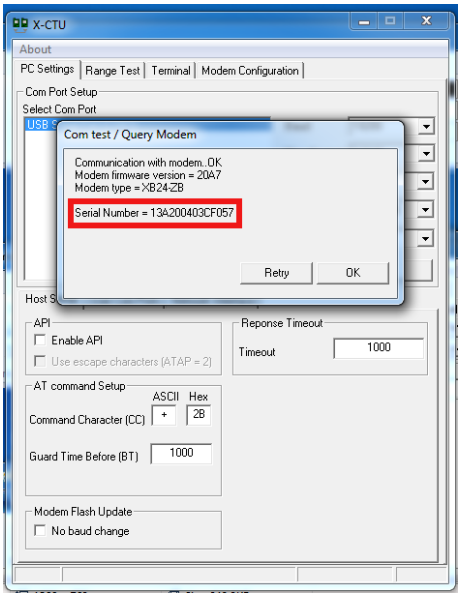


Figura D.6: Janela em que é apresentado o módulo encontrado no *X-CTU*

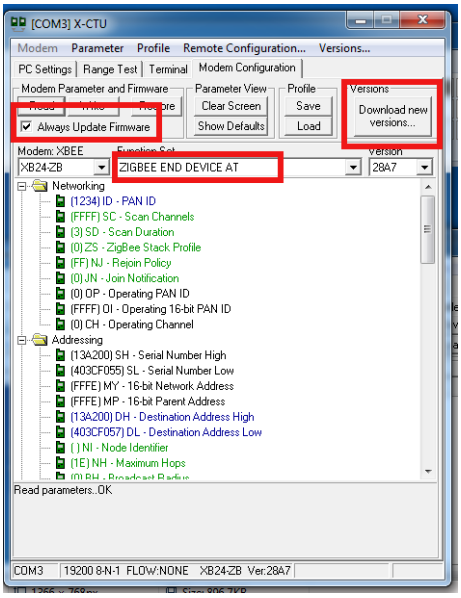


Figura D.7: Janela de configuração dos módulos no *X-CTU*

OKD.9, se de alguma forma conseguir ter os dois dispositivos conetados ao mesmo tempo, ao computador se tiver dois programas *X-CTU* abertos no separador *Terminal* poderá verificar a sua correta configuração, pois se enviar dados de um dos módulos estes serão apresentados no *Terminal* do outro. Se necessitar de fazer um teste de alcance com o *software*, pode faze-lo recorrendo ao separador *Range Test*

Em seguida será feita uma breve explicação dos campos que contemplam o separador *Modem Configuracion*, para ao alterar cada campo perceber o seu significado, assim só serão explicados os mais relevantes para este caso em concreto.

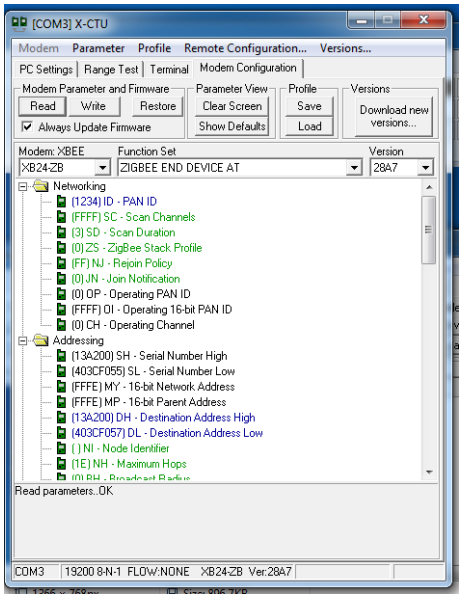


Figura D.8: Janela com os campos de configuração dos módulos no *X-CTU*

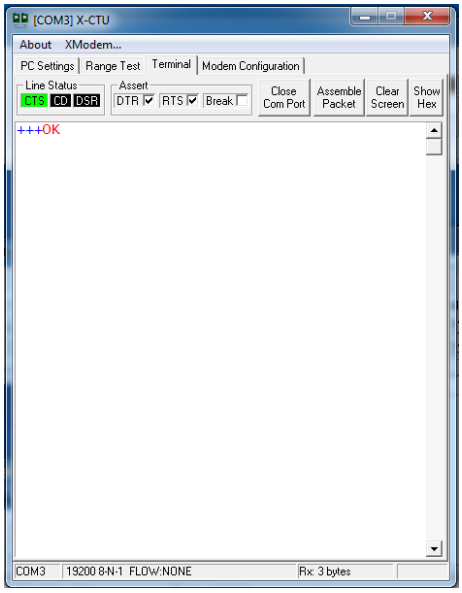


Figura D.9: Janela do terminal do módulos *Xbee* no *X-CTU*

- **CH** – O padrão ZigBee oferece a possibilidade de operar em vários canais. Para que dois ou mais módulos possam comunicar entre si estes devem operar no mesmo canal.
- **ID**– Este parâmetro define o PAN(Personal Area Network) ID. Para que dois ou mais módulos comuniquem entre si na mesma rede sem interferir com redes vizinhas, este devem ter o mesmo PAN ID e que é diferente das outras redes.
- **DL/DH** – Para que um módulo envie mensagens exclusivamente para outro módulo é necessário definir o endereço de destino da mensagem. O Destination Ad-

dress High (DH) e o Destination Address Low(DL) podem ser configurados de modo a que um módulo envie mensagens para um único endereço. Caso se pretenda que um módulo envie mensagens para todos os módulos que partilham o mesmo PAN ID deve-se definir DH=0 e DL=0xFFFF.

- **MY** – Para que outros módulos possam enviar mensagens exclusivamente para um módulo é necessário que este tenha um endereço. O parâmetro MY permite definir o endereço do módulo. Caso se pretenda que o módulo receba mensagens de todos os módulos que partilham o mesmo PAN ID deve definir-se MY=0xFFFF.
- **NI** – O parâmetro NI permite atribuir um nome ao módulo para que seja facilmente identificado na rede. O tamanho máximo está limitado a 20 caracteres.
- **ND** – Este parâmetro permite identificar todos o módulos que estão ligados em rede. Definições de comunicação RS232
- **BD** – Este parâmetro permite definir o Baud Rate da comunicação.
- **NB** – Este parâmetro permite definir o tipo de paridade ou desactivar.
- **D6** – Activar ou desactivar o controlo de fluxo RTS (Request To Send).
- **D7** – Activar ou desactivar o controlo de fluxo CTS (Clear To Send).

Apêndice E

Sistema de monitorização

It is appreciated that problems/questions regarding this tutorial are asked on answers.ros.org. Don't forget to include in your question the link to this page, versions of your OS & ROS, and also add appropriate tags.

1. Installation of a PostgreSQL server

Description: A step-by-step "cheat-sheet" for installing a PostgreSQL server on a Ubuntu machine. Does not cover any advanced installation procedures, just a very simple installation with no additional options. After installing the server, you can set up multiple databases, restore them from backup files, etc.

Tutorial Level: BEGINNER

Contents

2. Overview

This tutorial is a quick "cheat-sheet" for installing a PostgreSQL server on a Ubuntu machine. It is **not** meant to replace the detailed instructions that are provided along with PostgreSQL. However, we have found that some steps can be unclear in the instructions, and a step-by-step procedure can sometimes help. It also does **not** cover any advanced installation options.

This tutorial works for PostgreSQL 8.4. Here you can find the complete PostgreSQL 8.4 manual.

This tutorial only covers the installation of the database server itself. It does not cover the creation of individual databases once the server is up and running; an example of creating a database and populating it from a backup file can be found in the household objects database installation tutorial.

3. Install the PostgreSQL server

```
sudo apt-get install postgresql
```

- the installation process sets up data and configuration directories by itself, and tells you where they are. You can write them down, but there is another way of finding out where they are (below)
- Postgres server should start automatically after install process
- `ps auxw | grep postgresql` shows server is running as well as data and config directories

In the config directory of PostgreSQL you will find two very important files which we'll be using later:

- `pg_hba.conf`
- `postgresql.conf`

4. Create a PostgreSQL server account

General: PostgreSQL accounts are separate from OS accounts. During the installation process, both a PostgreSQL and an OS account will be created, both called **postgres**. The postgres account is a superuser, meaning it can add other PostgreSQL accounts.

What we need to do next is to login to PostgreSQL using the `postgres` account in order to create a new account for us to use.

4.1 Log in to the PostgreSQL server using root access

Become root, then su into the `postgres` OS account. That's it; if you are logged in to OS as the `postgres` account you can also log into PostgreSQL as the same account without a password.

Start the `psql` front end, then skip to the 'Add a new database user' section:

```
psql
```

4.2 Log in to the PostgreSQL server using sudo access

If you have sudo but not root access you should still be able to switch the the postgres user using sudo su and then open psql.

```
sudo su - postgres
psql
```

If that doesn't work see the next section, if it does then jump to the 'Add a new database user' section

4.3 Log in to the PostgreSQL server without root access

If you have sudo privileges, but not root access, and can not become the `postgres` OS account, you must edit `pg_hba.conf` to allow username `posgres` to be logged in by anybody without password

Find the lines:

```
# Database administrative login by UNIX sockets
local   all         postgres                          ident sameuser
```

Change `ident sameuser` to `trust`. **Remember to change back when done with this tutorial!!!**

After modifying `pg_hba.conf`, have the Postgresql server reload it by issuing a HUP signal:

```
ps auxw | grep postgresql
sudo kill -HUP xxxx
```

You should now be able to start the Postgresql front-end (`psql`) as the superuser postgres:

```
psql --username postgres
```

4.4 Add a new database user

Once you are logged into `psql` as `postgres`, you can add a user. It is recommended to start by adding a user WITHOUT superuser privileges, but with CREATEROLE and CREATEDB privilages. This is achieved by the following command:

```
CREATE ROLE willow LOGIN CREATEDB CREATEROLE PASSWORD 'willow';
```

You can then quit psql (\q).

If you have modified `pg_hba.conf` to let the `postgres` user connect without a password, be sure to revert the change.

5. Allow connections through TCP/IP

We will now enable TCP/IP connections to the PostgreSQL server.

Edit `pg_hba.conf`. At the end, where the list of authentication methods is, add the following lines. This allows anybody to connect through TCP/IP and access all databases, as long as they supply a right username and password. The password is md5 encrypted

```
# Anybody through TCP/IP with password
host    all         all         0.0.0.0   0.0.0.0     md5
```

Now you must let the server know it's supposed to accept TCP/IP connections from anybody. Edit the `postgresql.conf` file and set

```
listen_addresses = '*'
```

You will then need to restart the server altogether. The simplest way of doing this is to restart the machine.

5.1 Localhost connections

If you do not want to enable TCP/IP connections, one alternative is to allow your brand new user to connect from the local machine, WITH a password. Note that this means only code running on the same machine as the server will be able to connect.

```
# willow user can connect locally with password
local   all         willow                            md5
```

Remember that after modifying `pg_hba.conf` you must have the Postgresql server reload it by issuing a HUP signal.

```
ps auxw | grep postgresql
sudo kill -HUP xxxx
```

To check that it is working, try logging into `psql` on localhost as the new user:

```
psql --username willow --password --dbname postgres
```

6. PGAdmin3

PGAdmin is a graphical front-end to a database server. To install it use:

```
sudo apt-get install pgadmin3
```

The run it using:

```
pgadmin3
```

If you have set your server to accept TCP/IP connections, you can run pgadmin3 on any machine that can talk to your server machine over the network. If your server only accepts localhost connections, you'll have to run pgadmin3 on the same machine on which you just installed the server.

In `pgadmin3`, go through the following steps:

- `File` -> `Add Server...`
- type in the address of the machine that you installed the server on as well as the new postgres username you just created.
- connect to the server. If all goes well, you should see a list of databases running on the server (by default, there will be a single database available called `postgres`).

That's it, the server is up and running! You can now create your own databases on the server. Here is a tutorial for creating and restoring the household_objects database on your server, using a Willow Garage database backup file.

Except
where Wiki: sql_database/Tutorials/Installing a PostgreSQL Server (last edited 2012-11-20 16:17:51 by AnthonySoroka)