Instituto Politécnico de Tomar

Escola Superior de Tecnologia de Tomar

Hugo Filipe Mendes Magalhães

Estágio: LINE.IPT

Dissertação de Relatório de Estágio

Mestrado em Engenharia Eletrotécnica

Especialidade em Controlo e Eletrónica Industrial

Orientado por:

Doutor Carlos Alberto Farinha Ferreira

Relatório de Estágio apresentado ao Instituto Politécnico de Tomar para cumprimento dos requisitos necessários à obtenção do grau de Mestre (Mestrado em Engenharia Eletrotécnica, Especialidade em Controlo e Eletrónica Industrial)

Tomar Julho de 2015

Dedico este trabalho ao meu avô

RESUMO

O presente relatório é desenvolvido no âmbito da unidade curricular de Projeto do 2º ano do Mestrado em Engenharia Eletrotécnica – Especialidade em Controlo e Eletrónica Industrial e expõe o trabalho desenvolvido durante o estágio de nove meses na empresa LINE.IPT – Laboratório de Inovação Industrial e Empresarial. Durante este estágio foram desenvolvidos projetos para empresas de renome, entre as quais se pode identificar a Mitsubishi e a Henriques & Henriques.

Os projetos em causa enquadram-se em diversas áreas, entre as quais se incluem a eletrónica e a automação industrial, normas e legislação, entre outras. O trabalho consistiu em analisar as funcionalidades pretendidas pelas empresas de modo a desenvolver o trabalho (muitas vezes sob a forma de um protótipo) a ir de encontro com esses objetivos.

Para a Mitsubishi foi desenvolvido um protótipo para aquisição de dados e controlo da qualidade de soldadura com aplicação SCADA para monitorização de dados adquiridos e importação para base de dados. Após adquiridos os dados e analisados é dada uma indicação ao operador da máquina através de um sinal luminoso com a indicação se a soldadura está "OK/NOT OK".

Para a Henriques & Henriques desenvolveu-se um caderno de encargos para o licenciamento e desenvolvimento de camaras hiperbáricas. Este trabalho contém informação relativa às normas, legislação e procedimentos necessários para o licenciamento e acreditação da camara.

Para a Pneusines iniciou-se o estudo e desenvolvimento de uma máquina automática para desempeno de jantes. Desenvolveu-se e simulou-se a solução mecânica e decidiu-se toda a parte de automação da máquina nomeadamente os equipamentos a utilizar.

Como maioritariamente a área de implementação é a industrial, existe a exigência de um elevado grau de robustez e fiabilidade dos sistemas desenvolvidos de modo a que estes estejam preparados para ambientes adversos. É importante referir que todos os projetos anteriormente identificados envolvem um grande leque de áreas dentro da engenharia eletrotécnica. Este tipo de trabalho pode ser considerado bastante exigente. Devido à transversalidade do trabalho efetuado, antes do início de todos os protótipos houve um largo

período de levantamento, estudo e aquisição de conhecimentos nas áreas em causa, que se seguiu do traçar de estratégias de investigação de possíveis soluções, seu teste sumário e posterior desenvolvimento de soluções à medida, que foram numa última fase postas à prova, num ciclo contínuo de teste e melhoria.

Em paralelo foram desenvolvidas tarefas de organização, orientação de estagiários, desenvolvimento de placas de circuito impresso (pcb's) e montagem de sistemas de energias renováveis, entre outros.

No sentido de atualizar e expandir os conhecimentos o estagiário frequentou ainda cursos e formações na área dos microcontroladores, novas tecnologias de eletrónica e da impressão 3D.

Palavras-chave: Projetos, Eletrónica, Automação Industrial, Normas, Microcontroladores, Impressão 3D.

ABSTRACT

This report is developed under the curricular unit Project on the 2nd year of the course Master in Electrical Engineering – Control and industrial electronics specialty. It exposes the developed work during the nine months of internship on the company LINE.IPT – Laboratory of industrial and company innovation. During the internship were developed projects for big name companies like Mitsubishi and Henriques & Henriques.

The developed projects were on the electronics and industrial automation areas, regulation and legislation, and includes also other tasks. The work consisted on analyzing the functionalities wanted by the companies and constructing the final system (most of the time the work consisted on developing a prototype) fulfilling those objectives.

For the Mitsubishi project a prototype was developed for data acquisition of a welding quality control with a "SCADA" application for monitoring acquired data and database import. After the data is acquired and analyzed a light signal is given to the operator of the welding machine indicating if the welding is "OK/NOT OK".

For the H&H project it was developed a specifications book for the licensing and development of hyperbaric chambers. This work contains information about the norms, regulation and the needed steps for the licensing of the chamber.

As the majority of the implementation areas are at industrial systems, there is high demand for robust and reliable systems so they can be ready for adverse environments. It's important to mention that the previous projects involve a great number of specialities in the Electrical engineering. This type of work can be considered very demanding. Due to the transversality of the work being done, prior to the beginning of every prototype there was a large period of studying and knowledge acquiring on areas of the projects, and, only after that, strategies for the investigation of possible solutions are planned, information and developing solutions adapted to the situation are summarized, and finally, it is followed by a continuous cycle of testing and improvement.

Simultaneously, several management tasks were developed, intern student orientation, "PCB" development and installation of sustainable energy systems, and others.

With the objective of updating and expanding knowledge's, the intern students participated on lectures and classes on the microcontrollers' areas, new electronic technologies and 3D printing.

Keywords: Projects, Electronics, Industrial Automation, Regulation, Microcontrollers, 3D Printing

AGRADECIMENTOS

Este espaço é dedicado a todos aqueles que deram a sua contribuição e apoio durante a duração deste estágio. A todos eles um enorme obrigado.

Em primeiro lugar agradeço a toda a minha família, namorada e amigos que sempre me apoiaram e ajudaram, não me deixando ficar mal em momento algum.

Ao meu orientador Professor Doutor Carlos Ferreira agradeço por todo o apoio, disponibilidade, e pelos conselhos que foram determinantes durante o decorrer do estágio e do desenvolvimento deste relatório.

Ao meu orientador da parte do LINE.IPT, o Professor Mestre Pedro Granchinho agradeço por ter acreditado em mim e me ter dado esta grande oportunidade de experiência de trabalho.

Ao professor Doutor Manuel Barros e ao Engenheiro Pedro Neves tenho a agradecer por todo o apoio dado durante o estágio e pela grande amizade demonstrada.

Por fim e em especial desejo expressar um grande agradecimento ao meu Avô Fernando Mendes por tudo o que significou e continua a significar na minha vida, um eterno obrigado.

Índice

Índice de figuras

Índice de tabelas

Lista de abreviaturas e siglas

.TXT - Identificação da extensão de um ficheiro de texto.

ABS - Material para impressão 3D (“Acrylonitrile Butadiene Styrene”).

AC - Corrente alternada (“Alternating Current”).

AMPOP - Amplificador operacional.

AP - “Access Point” ponto de acesso para comunicações “wireless”.

BAR - Grandeza de medição de pressão.

CC3000 - Circuito para comunicações “wireless”.

COOL - Zona de intervalo entre pré-soldadura e soldadura.

DC - Corrente continua (“Directional Current”).

FLAG - Variável para controlar estados num algoritmo.

IDE - Software de desenvolvimento integrado (“Integrated Development Environment”).

KEYPAD - Teclado numérico.

LCD - Módulo de visualização gráfica (“Liquid Crystal Display”).

LED RGB - Conjunto de três elementos Led num unico encapsulamento (“Red Green Blue”).

LM555 - Integrado oscilador.

MASTER - Módulo principal do protótipo que recebe e analisa a informação.

METRO - Estilo de interface gráfica da Microsoft.

MFTE - Mitsubishi Fuso Truck Europe.

MYSQL - Base de dados.

OK/NOT OK - Estado da qualidade da soldadura.

OPTOCOUPLER - Integrado para separar referências de sinal.

PLA - Material de suporte para impressão 3D (“Polylactic Acid”).

PYTHON - Linguagem de programação.

QT - Ambiente gráfico para aplicações.

REDFLY SHIELD - Circuito para implementar comunicações “wireless”.

RTC - “Realtime Clock” relógio implementado nos circuitos.

SCADA - Sistema de supervisão e aquisição de dados.

SCHMITT TRIGGER - Circuito comparador com histerese para gerar pulso de sinal.

SLAVE - Módulo secundário do protótipo que adquire os dados.

SLOPE - Zona de subida da tensão de soldadura.

TCP/IP - “Transmission Control Protocol/Internet Protocol” conjunto de protocolos de comunicações em rede.

TDAQ - Circuito retificador da tensão de soldadura.

VISUAL BASIC - Linguagem de programação da Microsoft.

WELD - Zona de soldadura estável da tensão de soldadura.

WIRELESS - Comunicações sem fios.

XBEE - Componente para comunicação “wireless” 802.15.4.

Lista de símbolos

$R(\Omega)$ - Valor da resistência de amostragem para valor de tensão do sensor de pressão.

$U(V)$ - Valor de tensão de alimentação para o sensor de pressão.

$V_{entrada}$ - Valor de tensão do sinal do botão de soldadura.

$R_{entrada}$ - Valor da resistência em série com sinal do botão de soldadura.

$I\_entrada$ - Valor de corrente do sinal de entrada no circuito "Schmitt Trigger".

$I_{repouso}$ - Valor da corrente consumida em repouso pela estação meteorológica.

$I_{datalogger}$ - Valor da corrente consumida pelo "datalogger".

I_{modem} - Valor da corrente consumida pelo "modem".

I_{com} - Valor da corrente consumida em comunicações pela estação meteorológica.

$C_{energetico}$ - Consumo energético diário da estação.

$t_{repouso}$ - Tempo total em que a estação se encontra em estado de repouso.

V_{painel} - Valor da tensão de funcionamento do painel solar

t_{com} - Tempo total em que a estação se encontra em estado de comunicações.

$E_{produzida}$ - Energia produzida diariamente pelo painel solar

Coeficiente - Coeficiente de irradiância solar.

P_{painel} - Valor da potência do painel solar.

1 Introdução

O presente relatório tem como objetivo apresentar o trabalho desenvolvido pelo aluno Hugo Filipe Mendes Magalhães durante o estágio curricular na empresa LINE.IPT.

O LINE.IPT surge da parceria entre o Instituto Politécnico de Tomar, a Camara Municipal de Abrantes, a Tagus Valley e a Nersant, como catalisador da inovação e desenvolvimento tecnológico da região, promovendo a competitividade no tecido empresarial. Sendo um centro de investigação inteiramente direcionado para as empresas, o objetivo do LINE.IPT é desenvolver novos produtos, tecnologias e processos e/ou melhoria/reconversão de produtos ou processos já existentes, diretamente aplicáveis na industria.

Durante este estágio desenvolveu-se trabalho na área de automação industrial, eletrónica, mecânica, legislação e licenciamento de produtos e gestão de projetos.

Para além de toda a investigação teórica que este trabalho exige, de modo a cumprir todos os objetivos pretendidos pelas empresas, desenvolveram-se, testaram-se e implementaram-se no terreno a esmagadora maioria dos protótipos e circuitos que foram criados em laboratório.

Os referidos projetos foram desenvolvidos maioritariamente para a indústria. Com o objetivo de se obterem bons resultados e durabilidade, o nível de fiabilidade e robustez foi sempre uma preocupação, de modo a que a necessidade de intervenção humana na manutenção e reparação dos equipamentos fosse assim diminuída.

Em termos de organização do presente relatório optou-se por apresentar o trabalho desenvolvido dividindo-o nos seus vários projetos constituintes, ou seja, cada capítulo diz respeito a uma empresa. Como identificador utilizou-se o nome da empresa, nomeadamente: Mitsubishi, Henriques & Henriques, Pneusines e Nieeport. Por fim será apresentado um último capítulo que diz respeito a outras tarefas que se desenvolveram nomeadamente: orçamentos, requisições de material, gestão de projetos, entre outros.

Os projetos e tarefas realizados envolvem um grande volume de informação, assim optou-se por apresentar a informação de forma sucinta para que se possam abordar todos os temas desenvolvidos sem que a informação se torne demasiada nem a leitura exageradamente "pesada".

De referir que, em virtude dos projetos serem bastante abrangentes em termos de áreas de especialidade estes não são unicamente fruto do trabalho do autor do relatório, sendo desenvolvidos em conjunto por equipas multidisciplinares de acordo com a constituição das equipas afetas a cada um.

2 Mitsubishi

O projeto "Sistema de Aquisição de Dados da Soldadura" tem como objetivo o desenvolvimento, instalação e replicação de um protótipo para controlo de qualidade das máquinas de soldadura por ponto da empresa Mitsubishi Fuso Truck Europe (MFTE), situada no Tramagal – Abrantes, de acordo com o contrato celebrado entre o LINE.IPT – Laboratório de Inovação Industrial e Empresarial (IPT - Instituto Politécnico de Tomar) e a MFTE (CI&DT n.º 01/14).

Atualmente a qualidade da soldadura é garantida por testes de verificação do equipamento, antes da operação, e por controlo físico dos pontos de soldadura realizado no produto final ou em amostras de teste através de testes semi-destrutivos ou mesmo destrutivos.

A possibilidade de garantir a monitorização contínua dos parâmetros de soldadura, nomeadamente a força de aperto, a corrente elétrica e o tempo de soldadura, ao suprimir muitos testes de controlo de qualidade destrutivos reduz significativamente o desperdício de material.

Com este projeto pretendeu-se desenvolver-se um sistema de monitorização e controlo da qualidade dos pontos de soldadura numa linha de montagem de carroçarias que permita medir/estimar os seguintes parâmetros de soldadura: força de aperto, corrente elétrica e tempo de soldadura. Adicionalmente será também medida a temperatura de refrigeração do equipamento.

Figura 1 - Máquina de soldadura por ponto.

A Figura 1 diz respeito a uma das máquinas de soldadura por ponto onde se vai implementar o protótipo.

Cada plataforma de soldadura por ponto é constituída pela máquina de soldadura que é manuseada pelo operador (Figura 1), e por um controlador (Figura 2) responsável por controlar todo o processo de soldadura, que tem a inteligência para controlar o sistema e ser parametrizável quanto ao n.º de ciclos, pré-soldadura, entre outros. Como não se tem acesso à sua lógica de funcionamento utilizaram-se métodos experimentais para tentar compreender o seu funcionamento. Este baseia-se numa alimentação de 400V (bifásica) e controla a corrente no primário de um transformador através de um transformador de corrente e dois tiristores em anti-paralelo, que são comandados por dois botões de operação e programa que fazem atuar a pinça comandada pneumáticamente.

Figura 2 - Controlador de uma máquina de soldadura.

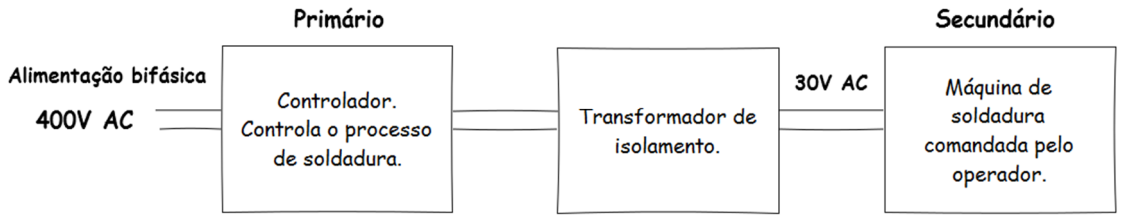


Figura 3 - Esquema geral de funcionamento do sistema de soldadura.

A Figura 3 apresenta a constituição e o esquema geral de funcionamento da plataforma de soldadura por ponto.

Os requisitos do sistema a desenvolver são os seguintes:

- Monitorizar os parâmetros de soldadura em tempo real;
- Garantir a estabilidade e normalização do processo, pela utilização de sequências de soldadura "standard";
- Validar o conjunto de pontos de soldadura efetuados (sequência de operação);
- Informar automaticamente a necessidade de limpeza dos elétrodos;
- Monitorizar a quantidade e qualidade dos pontos de soldadura efetuados.

O sistema a desenvolver deverá ter as seguintes funcionalidades (Tabela 1):

Tabela 1 - Funcionalidades pretendidas para o sistema

Item	Utilizador	Funções
1	Geral	Registo de cabine através de leitura ótica (com possibilidade manual) de forma a identificar parâmetros de processo vs produto. Registo do número de operador.
2	Operador	Validação ponto a ponto (100% avaliação e *feedback*), por meio de sinalizador ótico verde/vermelho.
3	Operador	Validação do conjunto de pontos (designado por JOB, n.º pontos), relativamente à sequência e condição de soldadura, por meio de sinalizador ótico verde/vermelho + total de pontos OK, associado ao Número de Cabine (leitura ótica ou introdução manual).
4	Operador	Passagem de equipamento para modo manual, para repetição de pontos, de forma a permitir a validação da sequência (apenas contabilização de pontos OK).
5	Operador	Emissão de aviso, por sinalizador ótico, para manutenção de elétrodos. Registo da ocorrência de execução e reinício dos contadores.
6	Operador Qualidade	Sistema de alarmes (parametrizável) para monitorização de: Pressão/Força de Aperto; Temperatura de Refrigeração;

		Intensidade e Tempo de Soldadura, com identificação do equipamento em questão, por meio de sinalizador ótico com 2 níveis de alerta (amarelo/vermelho), sendo o último (vermelho) também sinalizado com aviso sonoro.
7	Encarregado	Emissão de Relatório de falhas de manutenção de elétrodos e de falhas de sequência de operação.
8	Encarregado + Qualidade	Recolha dos vários resultados de processo em diferentes postos de linha e emissão de relatório final com indicação do número de cabine e OK (quando um posto tenha resultado não OK – NOK - para uma cabine, o resultado geral deve ser igualmente NOK, com indicação do posto).
9	Manutenção	Emissão de um relatório gráfico, através de indicadores parametrizáveis pelo utilizador (identificação do equipamento e intervalo de tempo) dos parâmetros de soldadura referidos em 6, com possibilidade de introdução de comentários para construção de histórico e de "download" dos dados para tratamentos adicionais.
10	Manutenção + Encarregado	Emissão de alerta (por "email") com base em valor parametrizável pelo utilizador das falhas por equipamento (falhas consecutivas e acumulado).
11	Manutenção	Atribuição de chave específica (acesso total) para realização de ensaios de forma a permitir intervencionar/testar o equipamento sem "poluir" o histórico de produção.
12	Qualidade	Cálculo da percentagem de falhas por pinça de soldadura, pela monitorização da percentagem de pontos repetidos no total de pontos dados (as pinças repetem pontos quando a intensidade real difere da programada em mais do que 5%).

O projeto designado por "Body in White – Process Monitoring" foi dividido em 3 fases de desenvolvimento:

Fase 1 - Medição de parâmetros de soldadura e operação: Aquisição dos sinais necessários para a classificação do estado das soldaduras e do estado das máquinas. A placa a desenvolver será montada numa máquina de soldar, a "Station SS3-Re-Spot, Weld Gun type C", N.º CM625201, e irá disponibilizar os sinais que serão processados através de um microcontrolador e transmitidos através de interface "wireless" para os vários "access points" a instalar na fábrica (Fase 2).

Fase 2 – Sistema de Comunicação, Interface e Teste piloto: Implementação numa máquina - Station SS3-Re-Spot, Weld Gun type C, Nº CM625201, de acordo os resultados obtidos na Fase 1 e consequente caderno de encargos definido, do sistema de aquisição de dados e desenho, desenvolvimento e implementação do sistema de comunicação (em tempo real) e interface de integração dos dados obtidos e monitorização do processo.

Fase 3 - Replicação para N postos.

2.1 Fase 1

2.1.1 Introdução à Fase 1

Para se proceder à análise em tempo real da qualidade de uma soldadura existem diversas variáveis físicas/parâmetros a que se pode recorrer, nomeadamente a tensão aos terminais de soldadura, a corrente, a pressão mecânica das garras, o som emitido durante o processo de soldadura, etc.

Numa primeira fase, com base em diversas publicações científicas [ref. 1 - 4], mediu-se o sinal sonoro da soldadura. Verificou-se que dado o ambiente industrial envolvente, este sinal apresentava uma elevada variabilidade, permitindo apenas retirar qualquer conclusão fidedigna acerca da qualidade soldadura com recurso a sensores de alta precisão e filtros bastante complexos, tendo-se optado por não incluir a medição deste sinal no protótipo desenvolvido.

O presente protótipo baseia-se na aquisição e tratamento de dados da tensão e corrente elétricas e da pressão mecânica.

Foram escolhidos os seguintes locais para a aquisição das diferentes variáveis:

- Tensão elétrica da soldadura: medida nos terminais da pinça de soldadura;
- Tensão do primário: medida no transformador de isolamento;
- Intensidade de corrente: medida no primário do transformador de isolamento;
- Pressão de ar: medida à entrada do êmbolo que aciona a pinça de soldadura.

Os dados recolhidos, ao serem cruzados com as caraterísticas próprias da máquina de soldar e com a sua parametrização, permitem obter uma informação detalhada do processo de soldadura. Esta informação é passível de ser armazenada numa base de dados e a sua utilização/objetivos podem ser diversos. Estes podem ser utilizados para a monitorização geral do processo de soldadura assim como para a verificação, em tempo real, da conformidade de cada ponto de soldadura. Pretende-se, assim, fornecer informação ao operador que o ajude a tomar decisões no sentido de obter/verificar uma maior qualidade do produto final.

Em resumo, o principal objetivo do protótipo, construído nesta primeira fase do projeto, consiste na monitorização e armazenamento dos dados das soldaduras com vista (numa segunda fase do projeto) à implementação de um sistema de controlo de qualidade da soldadura por ponto de cada uma das estações de soldadura existentes nas linhas de montagem do processo fabril.

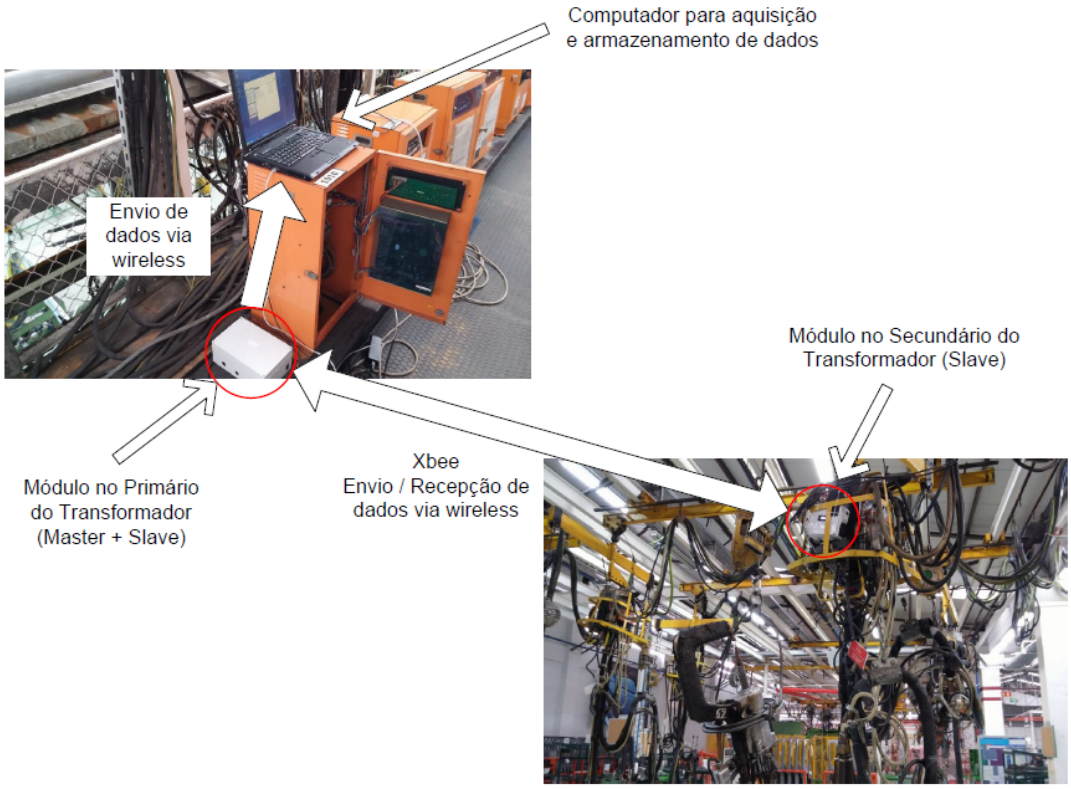


Figura 4 - Protótipo do sistema.

A Figura 4 pretende elucidar quanto ao esquema geral do funcionamento do protótipo. Em termos constitutivos e de arquitetura de funcionamento o protótipo desenvolvido possui dois módulos "SLAVE": um colocado no primário e o outro no secundário do transformador. O "SLAVE" colocado no secundário irá ler a tensão obtida na pinça de soldadura durante a operação e também a pressão de ar presente no circuito pneumático (de comando de fecho da pinça), enviando os dados via "wireless" para o módulo "MASTER". O "SLAVE" no primário irá ler a tensão e também a corrente de cada vez que ocorrer um ponto de soldadura, este "SLAVE" por estar fisicamente próximo do "MASTER" (na mesma placa de circuito impresso desenvolvida) enviará os dados via comunicação série. O "MASTER" uma vez possuindo os dados de ambos os "SLAVE" encaminha-os, via "wireless", para um computador, sendo necessário este possuir a aplicação desenvolvida em Visual Basic, a qual armazenará todos os dados em ficheiros de texto, um ficheiro por cada dia de laboração.

2.1.2 Análise teórica do processo de soldadura

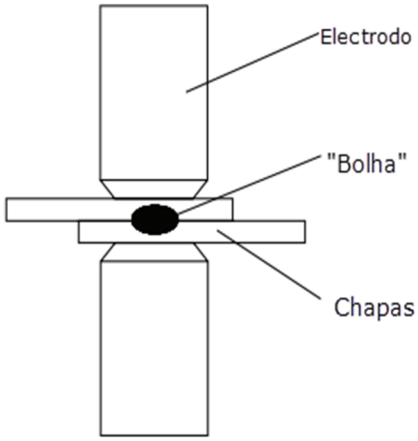


Figura 5 - Soldadura por ponto.

A Figura 5 diz respeito aos elementos intervenientes num processo de soldadura nomeadamente os elétrodos e as placas metálicas a soldar.

De modo a identificar quais os parâmetros fundamentais para identificar o "OK/NOT OK" da soldadura, foi necessário realizar uma investigação prévia ao sistema, sendo estes apresentados de seguida.

A máquina de soldar gera no secundário do transformador um sinal de tensão AC baseado no sinal da rede de 50Hz e com um factor de ciclo variável e controlado por dois tiristores. Esta apresenta um valor máximo de pico de 30 V e possui uma pressão de ar máxima de 6 bar na tubagem. No primário do transformador tem-se uma tensão AC de 400V e um valor de corrente máxima de 800 A. Estas foram as grandezas que se tiveram em conta durante o desenvolvimento dos circuitos de aquisição e acondicionamento de sinal.

Foram identificados os diversos parâmetros passíveis de serem utilizados para aquisição de dados e testes de conformidade, tendo sido obtidas as seguintes conclusões:

a) um dos parâmetros referenciado para monitorização foi a pressão entre as pinças e as placas, mas devido à necessidade da constante manutenção das pontas das pinças, esta opção foi rejeitada, optando-se apenas por colocar um sensor de pressão no tubo de alimentação a ar comprimido do pneumático;
b) por medição do valor de tensão no secundário, foram identificadas diferenças nas formas de onda quando a soldadura é "OK/NOT OK". Estas diferenças ocorrem não

só nos valores da amplitude de tensão como também no número de ciclos de fase da onda de cada soldadura.

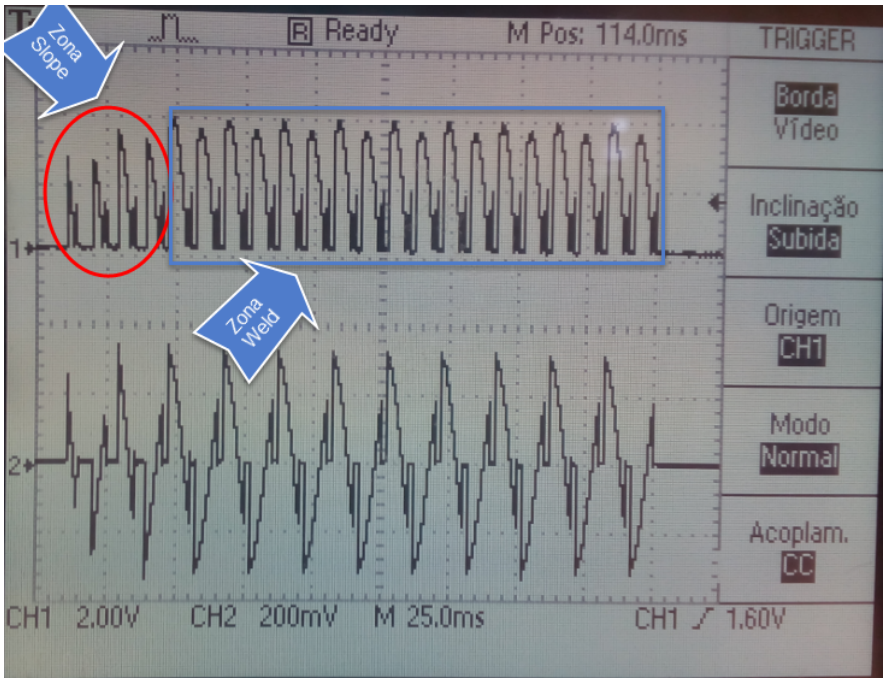


Figura 6 – Tensão num ciclo de soldadura, monitorizada com o osciloscópio.

As formas de onda apresentadas na Figura 6 têm assinaladas as diferentes fases de um processo de soldadura nomeadamente a zona "slope" que é a fase inicial da soldadura em que o valor da tensão sobe até atingir o valor de corrente pretendido para a soldura, a qual se encontra representada na zona "weld".

c) a forma de onda da corrente (canal 2 da Figura 6) também sofre alterações com soldaduras "OK/NOT OK". Por este motivo foi instalada uma sonda de corrente para monitorizar o seu valor no primário quando da realização da soldadura.

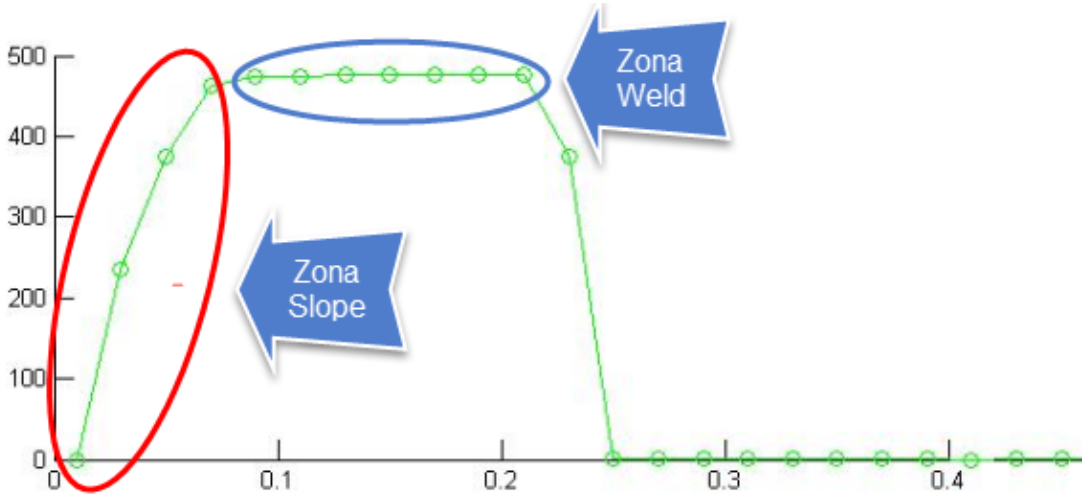


Figura 7 - Valor médio de tensão.

A Figura 7 representa o valor médio de tensão com as respetivas zonas identificadas.

Assim, os parâmetros elétricos monitorizados foram a tensão e a pressão no secundário do transformador e a tensão e a corrente no primário deste.

No que diz respeito às comunicações entre os diferentes módulos, considera-se que a melhor opção é utilizar comunicações "wireless", evitando a instalação de cablagem entre os diferentes módulos, o que, porque o primário do transformador e a máquina de soldadura estão em locais distintos e porque o número de máquinas a monitorizar é elevado (aproximadamente 70), implicaria um número de cabos e logística consideráveis. Devido ao ruído eletromagnético do local, terá de se optar por dispositivos capazes de funcionar nestas condições, e introduzir protocolos de funcionamento que permitam corrigir os erros de modo modo a que a informação enviada por "wireless" não seja corrompida.

Devido à necessidade da informação monitorizada poder ser importada para uma base de dados "MySql", a informação é armazenada num ficheiro de texto, com os dados separados pelo símbolo cardinal, facilmente exportável para base de dados.

De um modo geral, o protótipo terá o seguinte esquema de funcionamento (Figura 8):

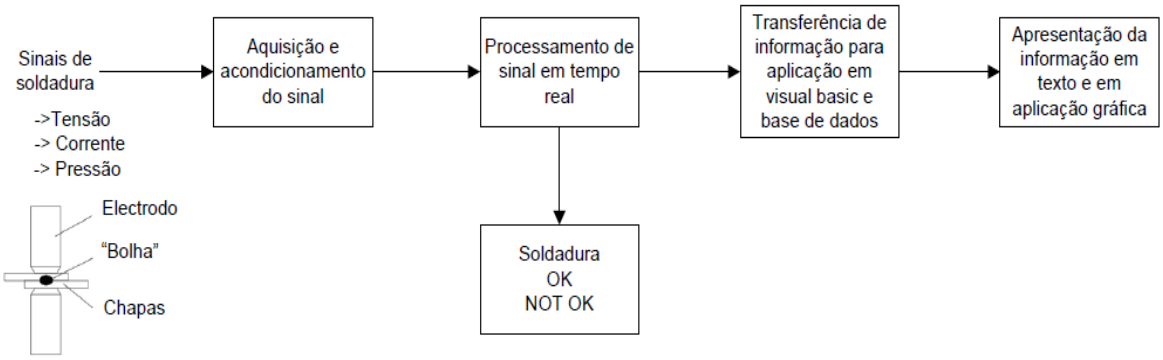


Figura 8 - Esquema geral de funcionamento.

2.1.3 Descrição do sistema

2.1.3.1 Constituição geral

O sistema é constituído por dois módulos de aquisição de dados e um "software" de interface com o computador. Um dos módulos de aquisição foi colocado na parte superior da máquina de soldar (secundário do transformador) e outro junto do controlador da máquina (primário do transformador).

Com o objetivo de tornar o protótipo o mais modular possível foram desenvolvidas dois tipos de placas eletrónicas, denominadas de "SLAVE" e "MASTER", com as seguintes caraterísticas:

- "SLAVE": Unidade responsável pela aquisição de sinais e respetivos cálculos de conversão para os valores reais de cada parâmetro medido (corrente, tensão e pressão).
- "MASTER": Unidade responsável por receber os pacotes de informação dos "SLAVE" e encaminhá-los para uma base de dados. É nesta placa que se poderá fazer a análise em tempo real da conformidade da soldadura.

2.1.3.2 Funcionamento do sistema

O funcionamento do sistema está ilustrado na Figura 9:

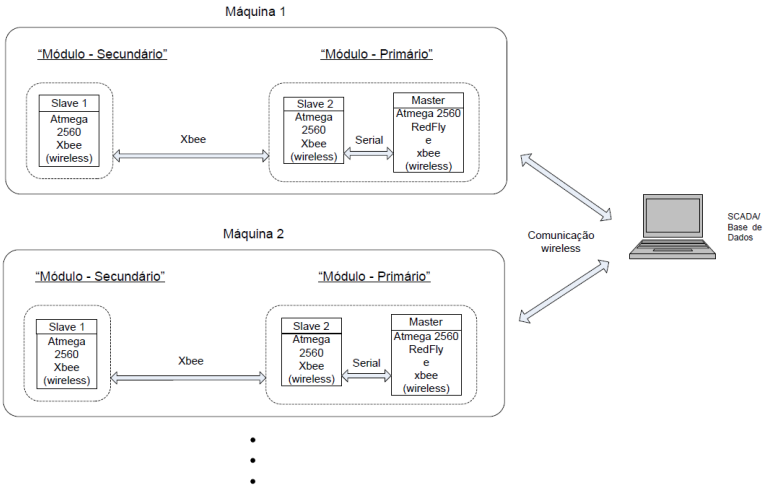


Figura 9 - Esquema de comunicações.

Quando ocorre uma soldadura, a placa "SLAVE" no secundário adquire os dados e envia-os para a placa principal "MASTER" via "wireless" ("Xbee"). Ao mesmo tempo, o "SLAVE" do primário adquire os dados encaminhando-os igualmente para o "MASTER" via comunicação série. O "MASTER" recebe os dados de ambos os "SLAVE", junta-os em pacotes de dados e envia-os via "wireless" para a aplicação "SCADA" no computador.

2.1.4 Módulo do secundário

2.1.4.1 Circuito "SLAVE" no secundário do transformador

O "SLAVE" do secundário contém um microcontrolador "Atmega2560", um circuito de leitura de tensão, um circuito de aquisição de valores de pressão e um circuito de alimentação para todo este módulo.

Nesta primeira fase utilizou-se as placas de desenvolvimento "Arduíno Mega" de modo a facilitar os testes e os circuitos desenvolvidos. Na fase seguinte o objetivo foi a substituição destas placas por outras desenvolvidas à medida do sistema pretendido para este projeto.

O circuito "SLAVE" do secundário adquire os valores de tensão e pressão da máquina enviando-os para o "MASTER" via "wireless". Para esta comunicação sem fios utilizou-se um dispositivo "Xbee" com o protocolo de comunicação 802.15.4 a uma frequência de 2.4GHZ. Estes dispositivos são bastante robustos e eficazes para redes sensoriais permitindo a configuração de diferentes parametros, tais como endereços de chegada e destino e os

modos de operação. A configuração de endereços é fundamental devido ao elevado número de máquinas que poderão ficar em funcionamento em simultâneo, em que cada uma delas terá o seu próprio endereço de modo a minimizar os possíveis erros na troca de informação. A aquisição de informação é sincronizada com o "SLAVE" do primário, ou seja, apenas quando há uma soldadura o protótipo deteta o evento, adquire e envia os dados para que o pacote de informação tenha ambos os valores do módulo secundário assim como os do primário referentes à mesma soldadura.

A Figura 10 ilustra uma primeira versão do módulo do Secundário.

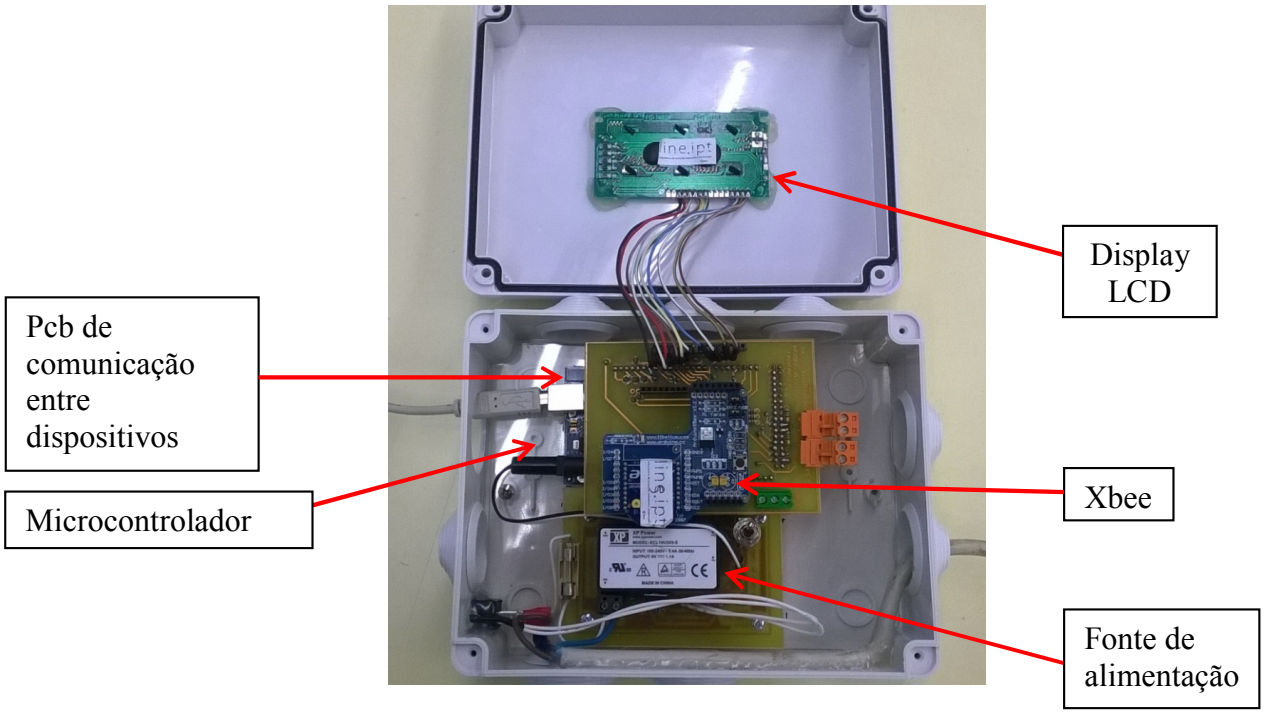


Figura 10 - Módulo SLAVE Secundário.

2.1.4.2 Circuito de leitura de tensão do secundário

O "SLAVE" do secundário é o responsável por adquirir valores de tensão e pressão da máquina de soldadura quando esta se encontra em funcionamento.

Para adquirir os valores de tensão desenvolveu-se um circuito denominado "TDAQ". Para o valor máximo de tensão de pico (30V) adquirido da máquina, o circuito converte-o para 5V de pico (valor máximo suportado pelas portas analógicas do microcontrolador). Como a forma de onda de tensão adquirida é "AC" e o microcontrolador não reconhece valores de tensão negativos, assim, esta é retificada através de um retificador de onda completa de

precisão, de modo a se obter uma onda de tensão entre 0 e 5V, correspondendo à conversão digital entre 0 e 1023 (“ADC” de 10 bits).

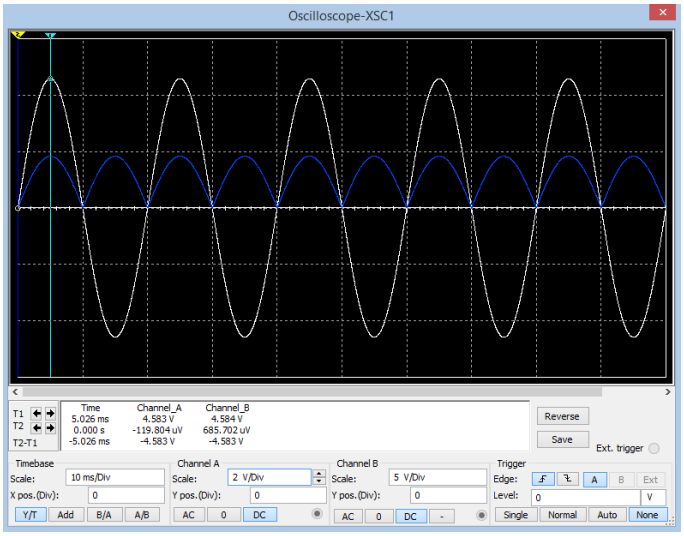


Figura 11 - Sinal de entrada AC e retificado (simulação).

A Figura 11 mostra o sinal “AC” original e o sinal retificado obtido durante a simulação.

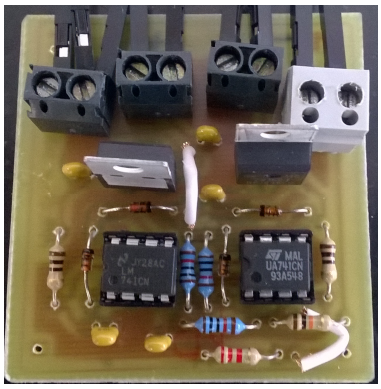


Figura 12 - Circuito de leitura de tensão.

A Figura 12 mostra o circuito retificador construído para testes. Esta é uma versão inicial que foi desenvolvida para testes em laboratório, portanto não se considerou relevante a apresentação do esquema desta versão. A versão final e os respetivos esquemas serão apresentados à frente.

2.1.4.3 Sensor de pressão

Para adquirir valores de pressão escolheu-se um sensor de pressão industrial da “Aplisens” (Figura 13), com capacidade para 10 bar de pressão máxima. Este é alimentado a 24V DC e

proporciona um sinal de saída em corrente de 4 a 20mA. O valor máximo de pressão de ar no sistema é de 6 bar, o que proporciona uma margem razoável em relação ao valor suportado pelo sensor.

Figura 13 - Sensor de pressão "Aplisens".

Como o sinal de saída do sensor é em corrente colocou-se uma resistência em série com a ligação à massa do sensor para dai retirar o valor da queda de tensão correspondente ao valor da corrente do sinal, o qual será diretamente adquirido pelo microcontrolador utilizado.

2.1.4.4 Algoritmo

O sinal de tensão gerado pela máquina durante uma soldadura tem uma frequência de 50Hz que é a mesma que a rede elétrica. Como se pretende uma aquisição dos valores em cada ciclo individual de soldadura torna-se necessário sincronizar a aquisição de dados no microcontrolador com os respetivos ciclos de soldadura. Para a aquisição de dados foi configurada uma interrupção de tempo (Timer3) com uma frequência de 2KHz de modo a ter-se um compromisso equilibrado entre resolução de aquisição e desempenho do microcontrolador, deste modo vão-se adquirir 40 amostras por ciclo (2000Hz/50Hz=40 amostras).

Esta amostragem permite a aquisição dos valores de tensão, corrente e pressão por ciclo de soldadura bem como a contabilização dos mesmos.

A interrupção está sempre a ser executada, e deste modo sempre a adquirir 40 amostras em cada ciclo da rede através de duas portas analógicas, assim que o microprocessador deteta

um valor analógico superior a um mínimo configurado inicia-se o tratamento dos dados adquiridos para assim guardar o numero de ciclos nas diferentes fases (“slope”, “weld” e “cool”) assim como a obtenção de um valor médio de tensão e corrente no final da soldadura.

Se o valor adquirido for inferior ao mínimo configurado (este valor mínimo é configurado de acordo com o valor adquirido em “vazio”) o microcontrolador rejeita os valores. Se forem executadas com sucessos todas as etapas do tratamento de sinal o microcontrolador envia os dados via “wireless” para o módulo Master no primário.

2.1.5 Módulo do Primário

2.1.5.1 Circuito “SLAVE” no primário do transformador

O circuito “SLAVE” no primário do transformador, representado na Figura 14 está colocado no primário do transformador, localizado no piso superior da planta industrial e é o responsável pela aquisição de valores de tensão e corrente do primário. Neste módulo, a diferença relativamente ao do “SLAVE” do secundário está no envio da informação, que é enviada para o “MASTER” através de uma comunicação série física e não “wireless” (pois neste caso o módulo “SLAVE” está colocado junto do módulo “MASTER” não necessitando de comunicações “wireless”).

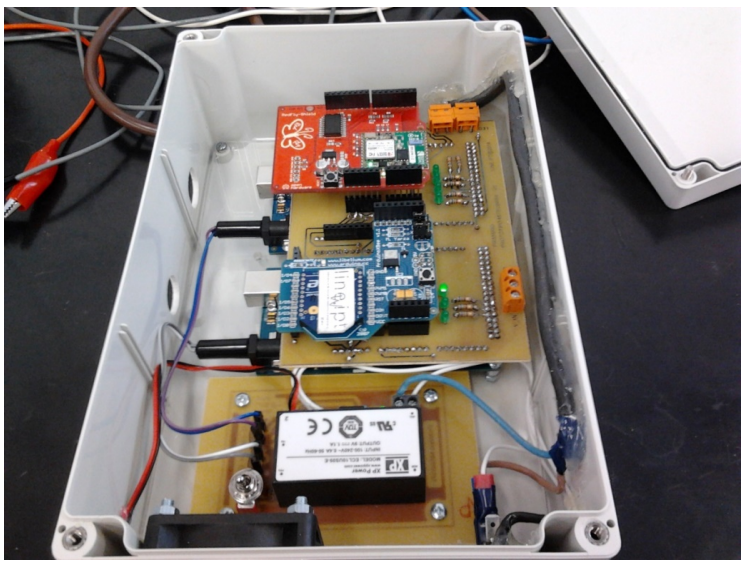


Figura 14 - Módulo no primário do transformador “SLAVE” + “MASTER”.

2.1.5.1.1 Circuito de leitura de corrente do primário

Para aquisição de valores de corrente recorreu-se a um transformador de intensidade "CTL-36-S56-20" que possui um intervalo de medição de 0.1A a 800A. Este transformador de intensidade é o mesmo utilizado pela "MFTE" em diversas máquinas. Colocou-se uma resistência de 1.5 Ohm, em paralelo com os terminais da sonda, para se obter uma tensão com um valor que permita ser adquirida diretamente pelo microcontrolador.

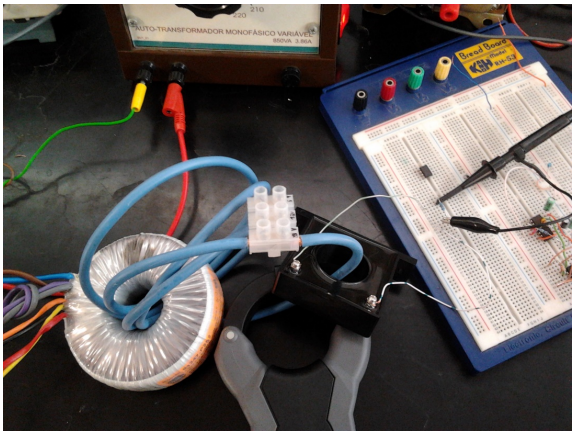


Figura 15 - Teste à sonda de corrente.

A Figura 15 mostra um dos testes efetuados à sonda de corrente utilizada.

2.1.5.1.2 Algoritmo

O algoritmo desenvolvido para o circuito "SLAVE" do primário é idêntico ao desenvolvido para o circuito do secundário com a diferença de que os dados em vez de serem enviados via "wireless" para o "MASTER" são enviados via ligação "Serial" física pois estes encontram-se no mesmo módulo.

2.1.5.2 Circuito "MASTER"

2.1.5.2.1 Receção de informação de ambos os SLAVE

O "MASTER" é o responsável pela sincronização das comunicações de cada protótipo, sendo, assim, o coordenador da informação. Ou seja, quando ocorre uma soldadura este recebe os dados dessa soldadura de cada "SLAVE" e junta toda a informação num pacote de dados para ser enviado via "wireless" para a aplicação no computador.

2.1.5.2.2 Envio de informação para aplicação Visual Basic no computador ("Wireless")

No "MASTER", para o envio dos dados via "Wireless", é utilizado o circuito "RedFly Shield" (Figura 16), que é um módulo com "Wifi/WLAN" para uma placa de desenvolvimento "Arduíno". Este circuito possui o protocolo de comunicação "IEEE 802.11b" a 2.4GHz sendo assim compatível com os protocolos mais vulgares de comunicação "Wireless".

Figura 16 - RedFly Shield.

Após o módulo "MASTER" receber a informação de ambos os "SLAVE", este comunica com o servidor da aplicação no computador, enviando a informação necessária (Figura 17).

Figura 17 - Módulo "MASTER" em comunicação com um PC.

2.1.5.2.3 Algoritmo

Nesta fase o módulo master é apenas coordenador de informação, se este receber dados tanto via "Wireless" ("Xbee") como pela ligação "Serial" irá verificar se o pacote de dados tem a estrutura correta. Caso esteja tudo em conformidade, o master enviará o pacote de dados do secundário e do primário para a aplicação "Visual Basic" no PC via "Wireless".

2.1.6 Aplicação SCADA

A aplicação desenvolvida em linguagem "Visual Basic" tem como objetivo guardar a informação recebida do protótipo num ficheiro de texto, para a sua posterior exportação para uma base de dados, e permitir uma monitorização gráfica dos valores adquiridos ou do número de soldaduras (a informação a ser visualizada nos gráficos será selecionada posteriormente).

Inicialmente, testou-se a exportação dos dados para um ficheiro "excel" em vez de ".txt" tendo-se verificado que a aplicação "Visual Vasic" era um pouco lenta a gravar os dados e essa operação requeria o sistema operativo "Microsoft Office". Optou-se, portanto, por gravar os dados num ficheiro de texto ".txt" (Figura 18 e Figura 19).

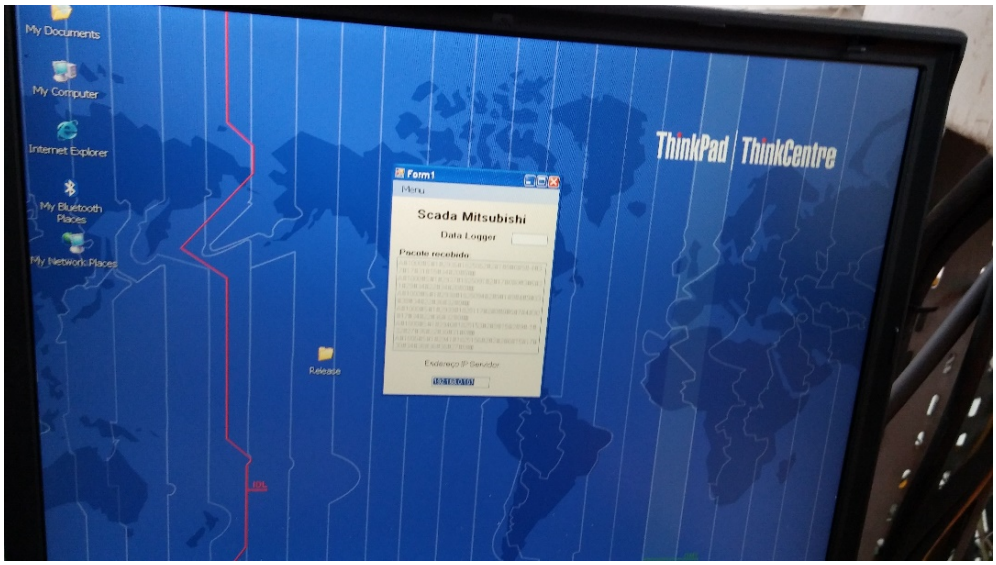


Figura 18 - Aplicação Visual Basic.

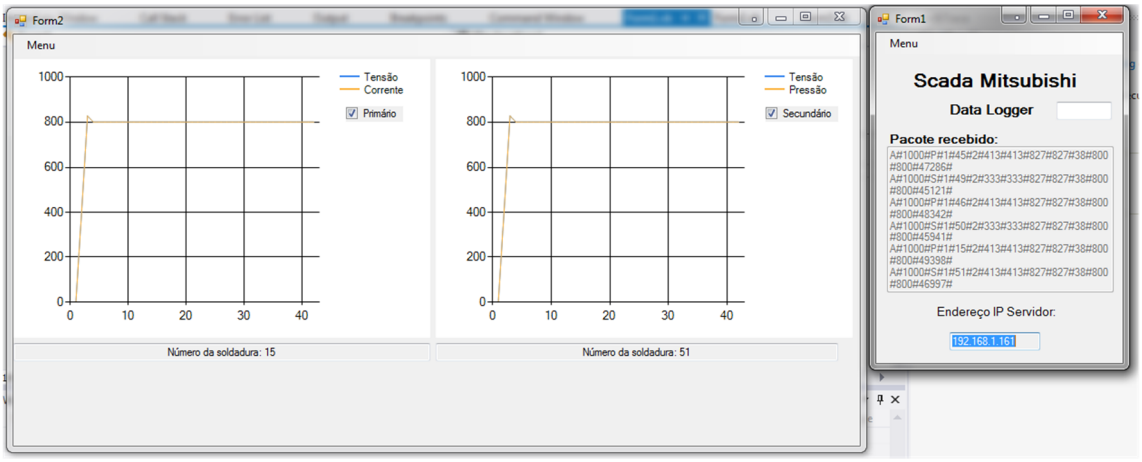


Figura 19 - Aplicação visual basic.

A aplicação desenvolvida tem uma estrutura e organização personalizáveis, permitindo acrescentar ou remover funções, se necessário.

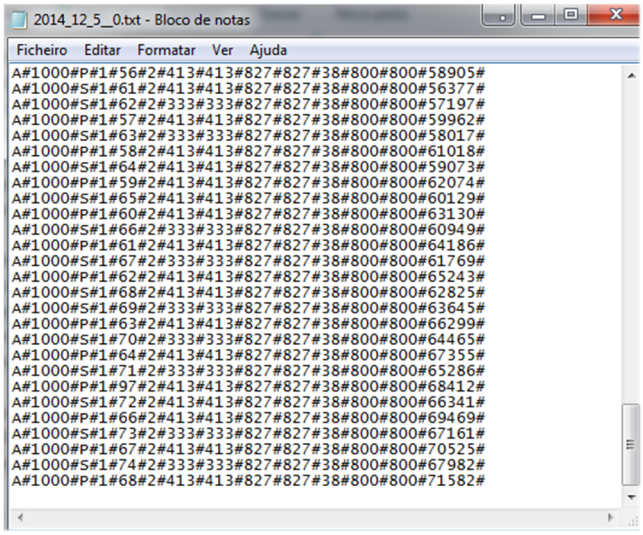


Figura 20 - Ficheiro de texto.

O pacote de dados tem os valores separados por um "#" de modo a facilitar a importação do ficheiro para uma base de dados (Figura 20).

2.1.7 Alimentação do protótipo

Para alimentar o sistema foi utilizada uma fonte de alimentação da "Schneider Electric" com a referência "ABL8FEQ24060" (Figura 21). Esta fonte permite alimentação de uma ou duas fases a 400V AC e tem como saída 24V de tensão e 6A de corrente.

Figura 21 - Fonte de alimentação.

Esta fonte permite a conversão de tensão da rede para tensão reduzida "DC", que pode ser utilizada para alimentar os sensores, os dispositivos de comunicação e também os microcontroladores. Deste modo não é necessário proceder a modificações nos sistemas de alimentação da fábrica, não perturbando assim o seu normal funcionamento.

2.1.8 Comunicações

Todas as máquinas têm um modo de comunicação "wireless" sendo por isso necessário a instalação de uma infraestrutura para suportar essas comunicações (Figura 22).

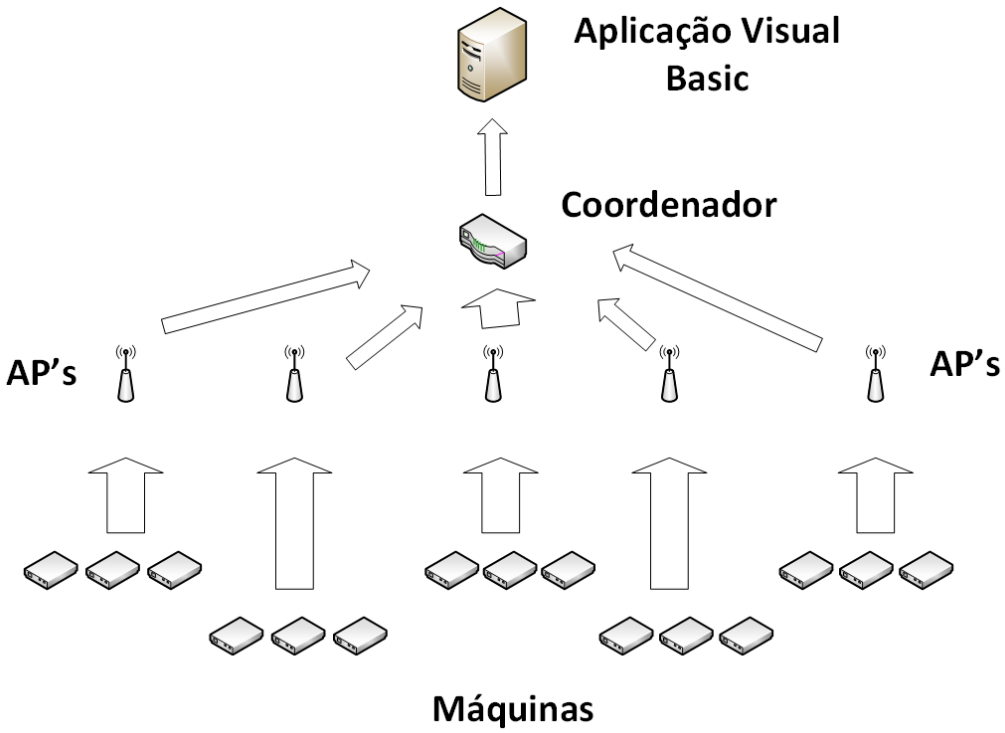


Figura 22 - Possível configuração de comunicações.

Para evitar congestionamentos de informação foi desenhada uma rede de comunicações onde cada 5 máquinas terão o mesmo ponto de acesso, "AP's" (Access Point). Para gerir esses "Ap's" vai ser necessário um coordenador para gerir o fluxo de informação da rede.

2.1.9 Análise de resultados obtidos

Com base nos testes efetuados à máquina durante uma soldadura conseguiram-se identificar as formas de onda da tensão, e com isto verificar o aspeto e resposta da tensão de diversas soldaduras e identificar pontos de soldadura "OK/NOT OK".

2.1.9.1 Testes de soldadura OK

Numa soldadura OK verifica-se que o valor de tensão sobe inicialmente (zona "slope") até atingir o fator de ciclo máximo (valor máximo de tensão) que corresponde à zona "weld" (soldadura) (Figura 23 e Figura 24).

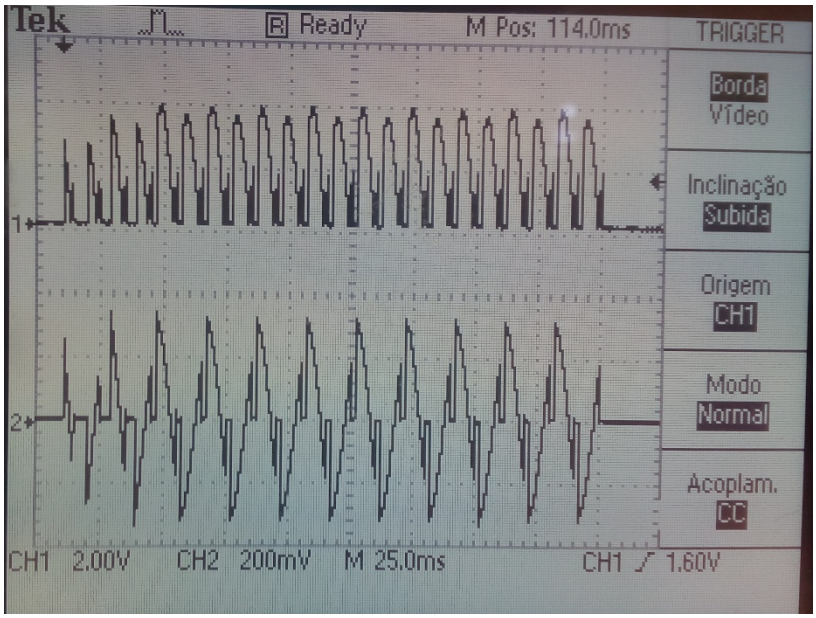


Figura 23 - Forma de onda de uma soldadura OK.

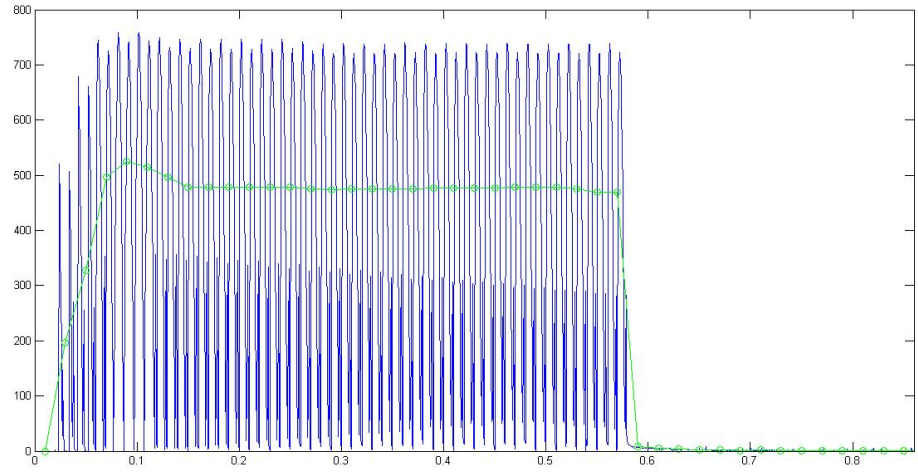


Figura 24 - Formas de onda retiradas com o protótipo.

Com o protótipo instalado adquiriram-se os dados referentes a um número relativamente elevado de soldaduras. Posteriormente, utilizando o "Matlab", analisou-se a informação e reconstruíram-se graficamente os perfis temporais da evolução da tensão ao longo do tempo (Figura 25).

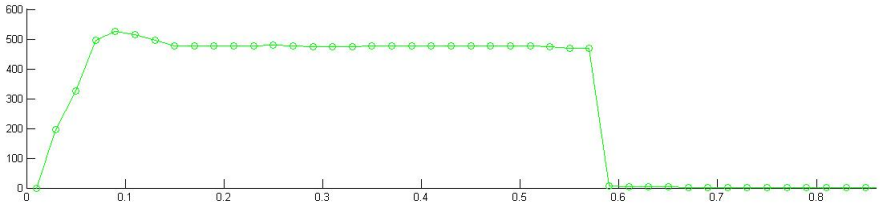


Figura 25 - Valor médio de tensão (valor adquirido da máquina, ver figura 18).

2.1.9.2 Testes com soldadura NOT OK

Na Figura 26 encontra-se representado um exemplo de uma soldadura "NOT OK".

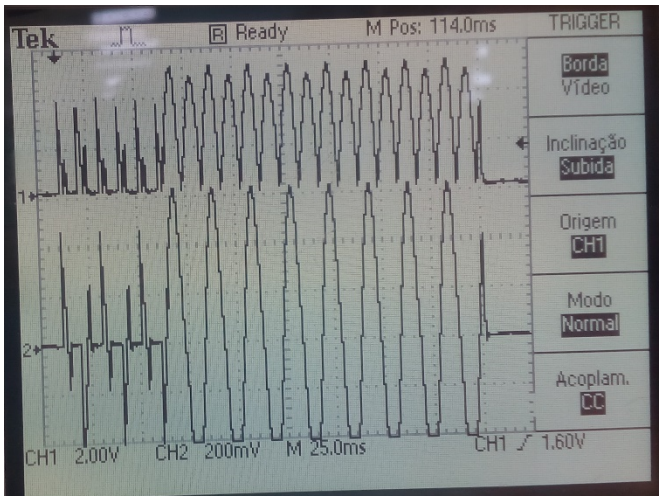


Figura 26 - Forma de onda da tensão de uma soldadura NOT OK.

Na Figura 27 identifica-se com maior detalhe a forma de onda do valor médio de tensão.

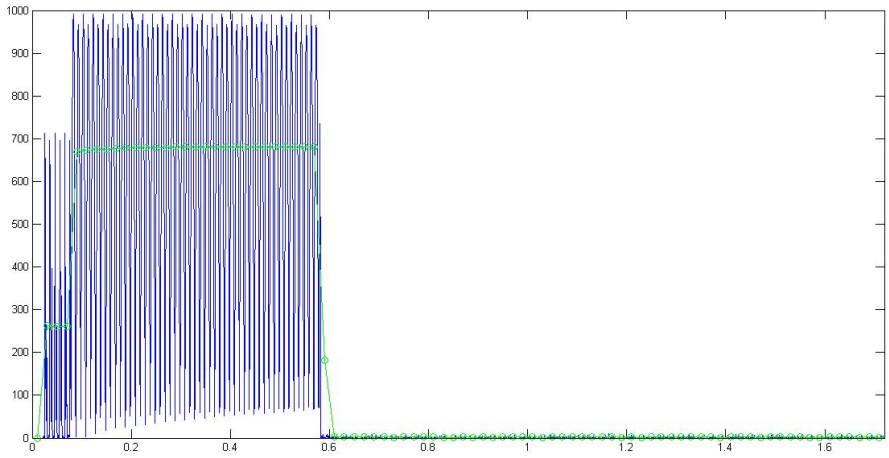


Figura 27 - Sinal no caso de ocorrer uma soldadura NOT OK (a azul identificam-se os ciclos do sinal de tensão e a verde o valor médio de tensão correspondente a esses ciclos).

Na figura acima está representada a forma de onda do valor médio de tensão de uma soldadura "NOT OK" (com as pinças em curto-circuito). Verifica-se que a fase "slope" tem um valor médio muito reduzido e na zona "weld" o valor de tensão atinge o valor máximo, fator de ciclo máximo, porque não há resistência entre as pontas da pinça de soldadura.

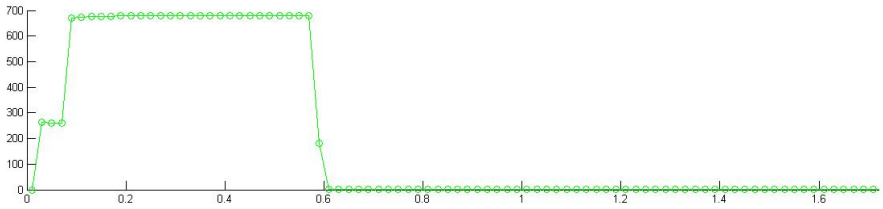


Figura 28 - Valor médio de tensão do secundário.

A Figura 28 mostra com mais detalhe o valor médio de uma soldadura "NOT OK".

2.1.9.3 Testes com pré soldadura - soldadura OK

É possível parametrizar as máquinas de modo a existir uma pré soldadura com o objetivo de melhorar a efetividade do processo de soldadura. Numa máquina com o programa de pré soldadura verifica-se uma primeira fase que serve de "aquecimento" às placas metálicas a soldar e uma segunda fase de soldadura (Figura 29 e Figura 30).

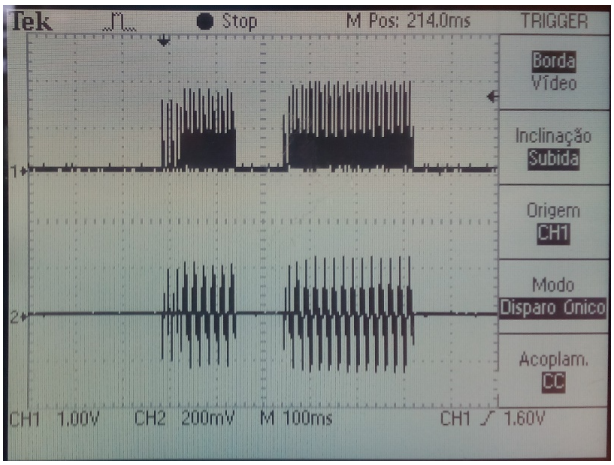


Figura 29 - Forma de onda de tensão da máquina com pré-soldadura (Canal 1 – Medição com protótipo / Canal 2 – Medição com pinça diferencial).

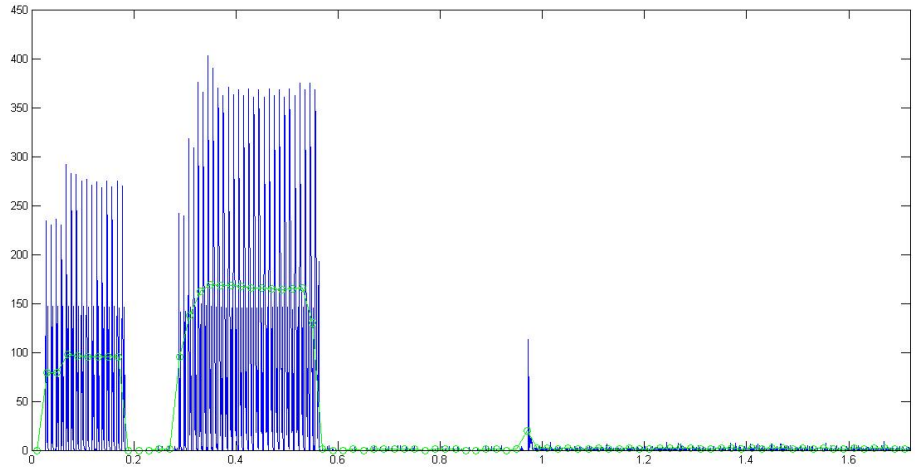


Figura 30 – Tensão (forma de onda azul) e valor da média da tensão (forma de onda verde) para um programa de pré-soldadura.

Analisando o sinal obtido pelo protótipo, verifica-se que este consegue identificar um programa de pré-soldadura corretamente, assim como calcular o seu valor médio (Figura 31). Verifica-se, também, que a fase da pré-soldadura é bastante mais reduzida que a fase principal da soldadura.

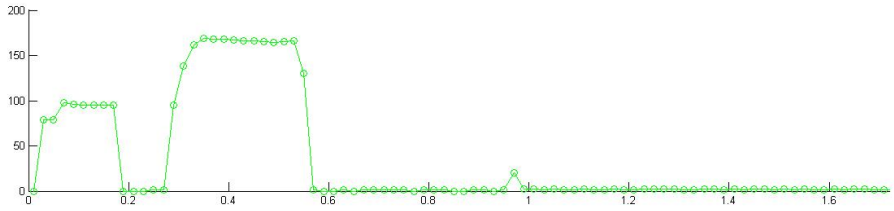


Figura 31 - Forma de onda do valor médio de tensão adquirido.

Verifica-se ainda que a fase da pré-soldadura possui uma zona "slope" com a duração de 2 ciclos atingindo de imediato o valor máximo de tensão na zona "weld". Na fase principal da soldadura identifica-se uma zona "slope" com a duração de três ciclos e uma zona "weld" com a duração de dez ciclos.

2.1.9.4 Testes com pré soldadura - soldadura NOT OK

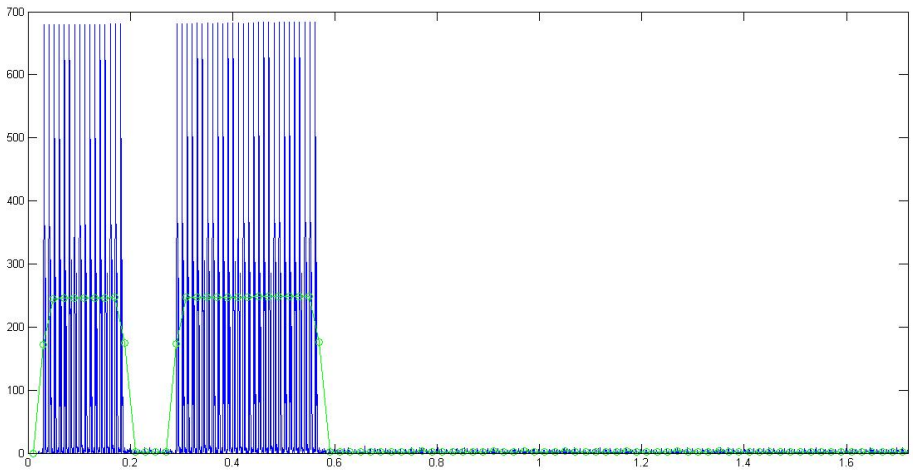


Figura 32 - Não soldadura com programa de pré-soldadura.

Para uma não soldadura com o programa de pré-soldadura, verifica-se que ambas as fases apresentam um "curto-circuito" pois a zona "slope" (Figura 32 e Figura 33) tem a duração de um ciclo, atingindo, assim, rapidamente o valor máximo de tensão.

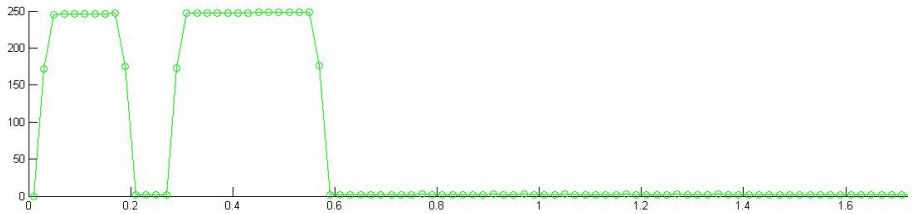


Figura 33 - Valor médio de tensão para uma soldadura NOT OK.

Assim, a duração da zona "slope", para ambas as fases da soldadura "NOT OK", tem apenas duração de um ciclo (valor muito baixo para uma soldadura correta).

2.1.9.5 Testes em ambiente real

A Figura 34 mostra a totalidade dos valores de tensão obtidos no secundário durante um dia de soldadura, concatenados num único gráfico. Como se pode verificar é possível identificar as curvas correspondentes aos diferentes programas disponíveis pela máquina, pois existem diversos níveis de tensão e números de ciclos diferentes.

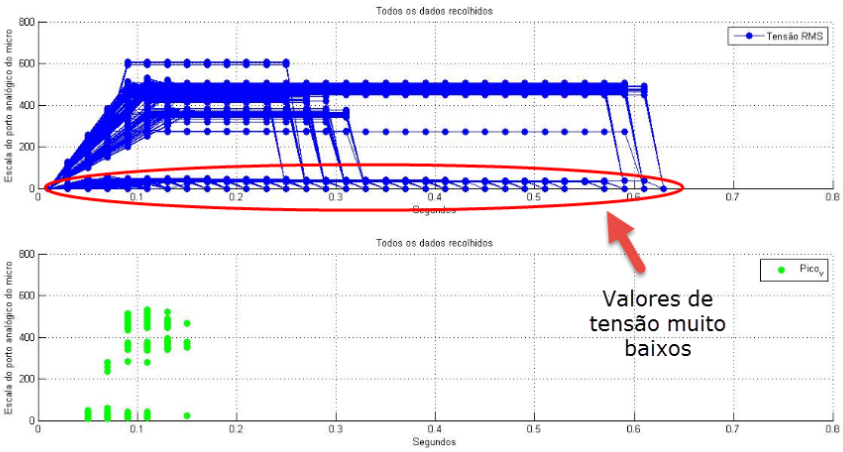


Figura 34 - Valores de tensão do secundário muito baixos (dados recolhidos do ficheiro txt durante um período de tempo de dois dias).

Durante a fase de testes verificaram-se três tipos de problemas diferentes a seguir identificados.

Problema 1 - Ruído eletromagnético

A existência de ruído eletromagnético (ondas com valores de tensão demasiado baixos para serem considerados uma soldadura. Chegou-se à conclusão de que este problema é provocado pela operação de outro transformador primário de outra máquina fisicamente próxima. O problema das "falsas" soldaduras resolve-se utilizando os dados do sensor de pressão e o acesso ao botão de soldadura, que identificará em tempo real a máquina em funcionamento.

Problema 2 – Dessincronização dos pacotes de dados

A dessincronização dos dados verificou-se durante o envio de informação dos "SLAVE" do primário e do secundário, devido à existência de ruído eletromagnético e ao facto de haver mais que uma máquina por transformador. Desta forma, para sincronizar os pacotes de dados do primário e do secundário monitorizou-se o sinal do botão de pressão/interruptor que comanda a soldadura da máquina, tanto no módulo secundário como no primário, garantindo assim que ambos os pacotes de dados dizem respeito à mesma soldadura.

Problema 3 – Identificação do programa de soldadura

Como cada máquina tem dois programas possíveis que serão selecionados pelo operador é fundamental distingui-los para avaliar se a soldadura é OK ou NOT OK, sendo por isso necessário monitorizar o sinal do comutador do programa.

2.1.10 Conclusões da fase 1

2.1.10.1 Aspetos técnicos

A análise dos resultados obtidos comprovou que o protótipo desenvolvido consegue efetuar a aquisição de dados durante uma soldadura, sendo possível identificar as principais zonas de soldadura, nomeadamente a zona "slope" e "weld" e o número de ciclos correspondente a cada zona.

Verificou-se ser necessário o acesso ao sinal do comutador de programa de soldadura na máquina, para ser possível comparação de parâmetros, que são diferentes para cada programa de soldadura bem como o número de ciclos de soldadura.

Devido às interferências nas cablagens verificadas no primário e à dessincronização dos dados adquiridos em cada soldadura (pacotes de dados do primário e secundário), é também necessário o acesso aos sinais de comando de soldadura e do sensor de pressão, para garantir que os valores adquiridos são os corretos e que dizem respeito à máquina em questão.

No que diz respeito às comunicações "Wireless" não se detetou qualquer tipo de problema.

Relativamente às tecnologias utilizadas, poderiam escolher-se outras alternativas (com base no preço e funcionalidade). Por exemplo, no que diz respeito aos sensores, os valores de mercado destes componentes variam entre os 90 e os 500 euros: esta variação de preço não se reflete em diferenças significativas nas suas caraterísticas básicas, mas sim nas marcas comerciais, o que poderá ter algum impacto na fiabilidade e precisão a longo termo.

No que se refere a opções relativamente aos microcontroladores utilizados, poderá ser necessária a introdução de microcontroladores de 32 bits, por estes possuírem maior capacidade de processamento, que será importante quando se introduzir o "módulo" de controlo de qualidade.

Para as comunicações "Wireless" existe a possibilidade de se substituir as "Redfly Shield" e optar pelos dispositivos da "Texas Instruments" "CC3000" com um custo ligeiramente mais reduzido e com as mesmas funcionalidades.

Na próxima fase de desenvolvimento introduziu-se um módulo "RTC" (relógio e data) para facilitar a exportação da informação para a base de dados. Poderá também ser colocado um sistema de introdução de parâmetros específicos de cada máquina e tipo de soldadura.

2.1.10.2 Inputs para a fase 2

A Fase 2 - Sistema de Comunicação, Interface e Teste piloto do projeto compreende a implementação do sistema desenvolvido numa máquina de soldar, para teste, aferição e validação do protótipo.

Os resultados obtidos na Fase 1 já incluem etapas previstas apenas para a Fase 2, mas que foram iniciadas porque deram consistência e permitiram a leitura dos dados obtidos permitindo o suporte à tomada de decisão, principalmente no que diz respeito ao caderno de encargos e escolha de componentes a incorporar no protótipo que se pretendia concluir na Fase 2.

Até este ponto de desenvolvimento e tendo em conta as funcionalidades que se introduziram na fase seguinte fez-se os respetivos cálculos com os componentes, sensores e materiais utilizados, estimou-se que o preço de venda para cada protótipo por máquina seja aproximadamente de € 3.000,00 já com mão-de-obra incluída.

2.2 Fase 2

Tendo sido a fase 2 do projeto aprovada iniciou-se por desenvolver otimizações aos circuitos implementados anteriormente e novas funcionalidades pedidas pela empresa.

De referir que algumas das funcionalidades propostas na Fase 2 foram desenvolvidas na Fase 1 nomeadamente o sistema de comunicações "wireless", pois para testar a funcionalidade do protótipo como um todo foi necessário implementar as comunicações "wireless" já na primeira fase pois só assim seria possível comprovar que o protótipo funcionava corretamente.

De referir que a instalação do sensor de pressão só foi possível nesta fase devido a problemas logísticos que a atrasaram.

2.2.1 Instalação do sensor de pressão

Após a receção do sensor de pressão anteriormente referido procedeu-se à sua instalação.

Electrical parameters	
Output signal	4 ÷ 20 mA, two wire transmission 0 ÷ 10 V, three wire transmission
Power supply	10.5 ÷ 36 V DC – two wire transmission 15 ÷ 30 V DC – three wire transmission 24 V AC
Load resistance (for current output)	$R[\Omega] \leq \frac{U_{sup}[V] - 10.5\,V}{0.02\,A}$
Load resistance (for supply output)	$R \geq 5\ k\Omega$

Figura 35 - Características elétricas do sensor.

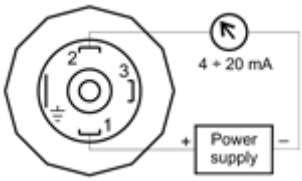


Figura 36 - Esquema de ligações do sensor de pressão.

Partindo do princípio que o protótipo instalado tem uma tensão disponível de 15V para a alimentação do sensor estimou-se o valor da resistência de amostragem com base na equação de cálculo de resistência de carga fornecida pelo "datasheet" (Figura 35 e Figura 36), obtiveram-se os seguintes valores:

$$R(\Omega) \leq \frac{U(V) - 10.5V}{0.02A} \quad \textbf{(1)}$$

$$R(\Omega) \leq \frac{15 - 10.5}{0.02} \quad \textbf{(2)}$$

$$R(\Omega) \leq 225\Omega \quad \textbf{(3)}$$

O valor da resistência escolhida foi de 220Ω que cumpre com os requisitos necessários.

O sinal de saída do sensor é em corrente por essa razão adquiriu-se a queda de tensão da resistência de carga, a qual reflete o valor de corrente.

Procedeu-se então à instalação do sensor de pressão na máquina (Figura 37).

Figura 37 - Sensor de pressão instalado na máquina de soldadura.

Verificou-se posteriormente com o osciloscópio o sinal de tensão de saída no evento de fecho da pinça de soldadura (Figura 38):

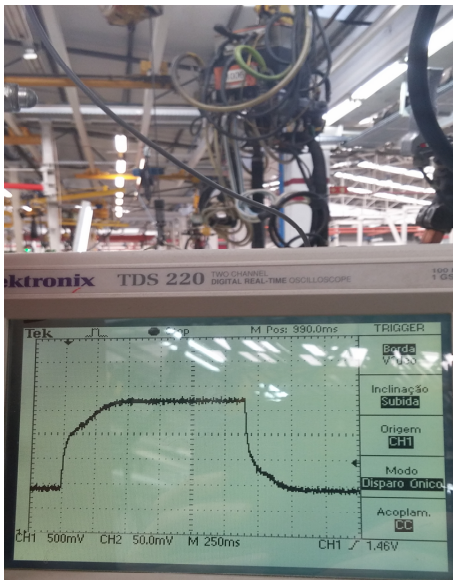


Figura 38 - Sinal de tensão de saída do sensor de pressão.

Ao observar a figura acima verificou-se que o sensor funciona corretamente e que para o valor máximo de pressão (10 bar) este apresenta um sinal de tensão de 4.4V que pode ser adquirido diretamente pelo microcontrolador.

2.2.2 Sistema de introdução de parâmetros

Devido à alteração dos parâmetros de soldadura das máquinas com base no tipo de material que está a soldar, o protótipo terá de ter esses parâmetros atualizados de modo a que possa comparar os valores adquiridos com os parametrizados e assim indicar se a soldadura está correta ou não. Para isso é necessário uma interface que permite o funcionário atualizar os parâmetros de soldadura sem que isto implique reprogramar o microcontrolador. Do ponto de vista do interface com o utilizador criaram-se três modos de operação selecionados por um interruptor, um modo para aquisição de dados em que não pode ocorrer introdução de parâmetros, um modo para introdução de parâmetros não havendo aquisição de dados e um terceiro modo "standby" para proceder à escolha de um dos outros dois modos.

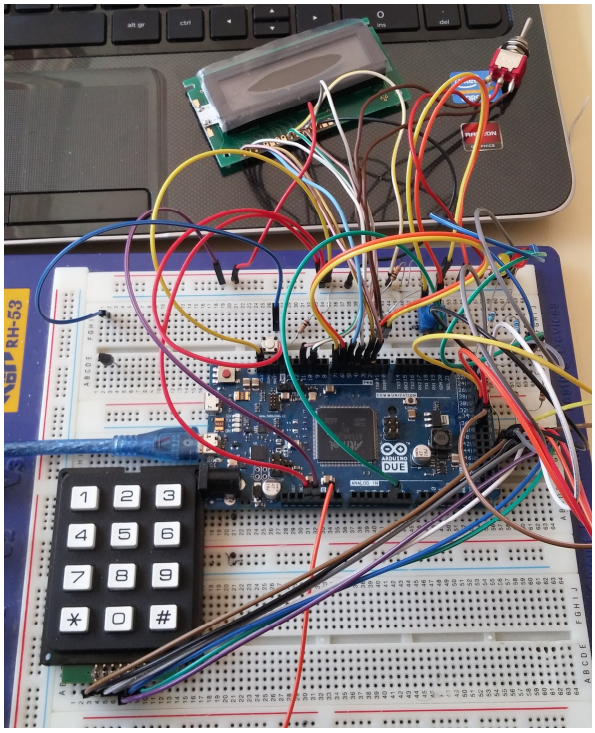


Figura 39 - Montagem de sistema de parâmetros.

A Figura 39 mostra o sistema de introdução de parâmetros desenvolvido.

A tecla cardinal serve para confirmar o parâmetro introduzido e o botão asterisco (*) serve para fazer o "reset" aos parâmetros introduzidos. O programa desenvolvido suporta a introdução dos parâmetros necessários. Se forem introduzidos mais dígitos que os permitidos surge uma mensagem de erro no LCD com indicação que o parâmetro introduzido está errado, é efetuado um "reset" apenas a esse parâmetro e volta-se a pedir a introdução do

parâmetro. Após a introdução de todos os parâmetros o programa coloca-os nas variáveis programadas para o controlo de qualidade da soldadura.

Possibilita-se assim, de uma forma simples, a introdução/modificação dos parâmetros de controlo de qualidade no protótipo desenvolvido.

2.2.3 Módulo RTC

Um dos requisitos pedidos pela Mitsubishi foi que o protótipo identificasse a data e a hora no pacote de dados da soldadura. Com esse intuito desenvolveu-se um módulo "RTC" para a contagem da hora e data. Recorreu-se ao IC "DS1307" que faz a contagem do tempo. Este permite o normal funcionamento do relógio mesmo com a falha da alimentação através de uma pilha externa de 3V e permite a comunicação com o microcontrolador via I2C (Figura 40).

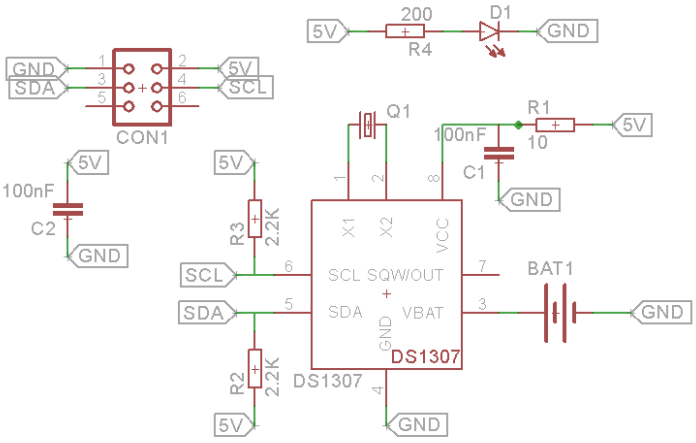


Figura 40 - Esquemático do circuito RTC.

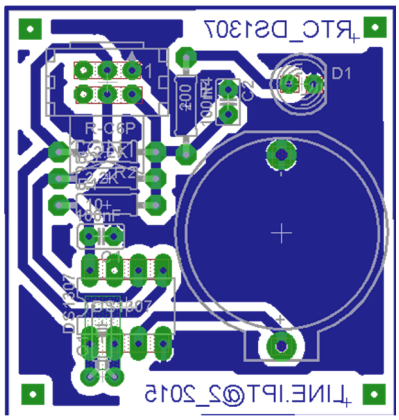


Figura 41 - PCB desenvolvida.

A Figura 41 mostra o desenho da “PCB” desenvolvida.

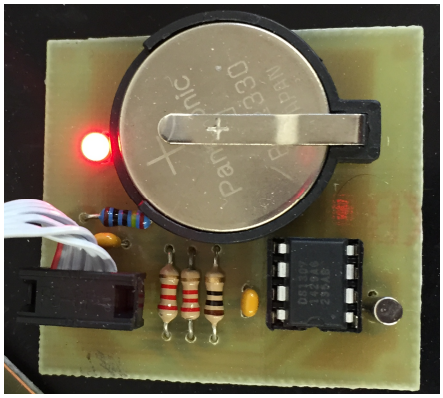


Figura 42 - Placa RTC desenvolvida.

A Figura 42 mostra o módulo final.

2.2.4 Módulo SD

Para que o sistema seja o mais seguro possível implementou-se um circuito para guardar os dados de soldadura quando as comunicações “wireless” falham recorrendo a um cartão de memória do tipo “SD”. Se o módulo não receber a confirmação que o pacote de dados foi recebido com sucesso este guarda automaticamente esse pacote de dados no cartão “SD”. O circuito desenvolvido comunica com o microcontrolador via “SPI” e contém um “socket” para introdução de cartões “SD”. Foi ainda necessária a utilização de um IC “4050” para fazer “level-shift” dos sinais “SPI” pois os sinais do microcontrolador tem um nível de tensão de 5V enquanto que os do cartão “SD” são de 3.3V.

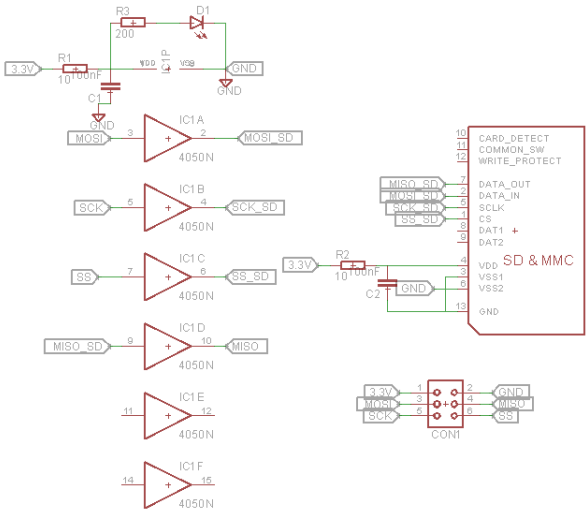


Figura 43 - Esquemático do circuito SD.

A Figura 43 mostra o esquema do circuito “SD” desenvolvido.

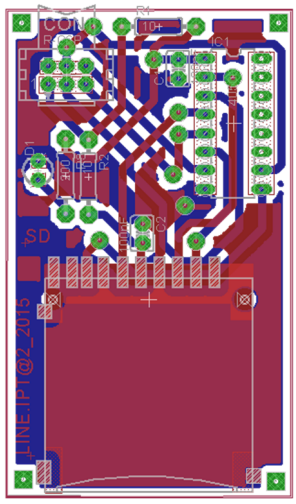


Figura 44 - PCB desenvolvida para o circuito SD.

A Figura 44 mostra a PCB desenvolvida do circuito SD.

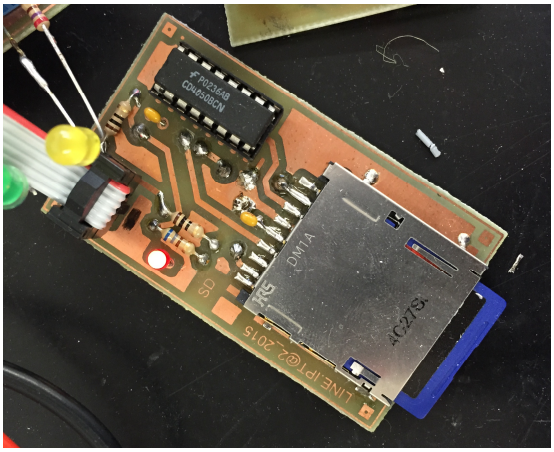


Figura 45 - Circuito SD desenvolvido.

A Figura 45 mostra o protótipo do módulo desenvolvido.

2.2.5 Módulo Xbee

Para se diminuir o custo do protótipo e também o tamanho do mesmo, retirou-se a a placa de interface para o “Xbee” utilizada anteriormente e desenvolveu-se um circuito simplificado para a integração deste com o microcontrolador. Este circuito comunica via ligação “Série” com o microcontrolador, contém um regulador de tensão de 3.3V, leds de estado e um “level-shift” implementado com recurso a um divisor de tensão para a porta de receção de dados do “Xbee” pois este funciona com níveis de tensão de 3.3V enquanto que

o microcontrolador funciona com 5V. No sentido inverso não é necessário elevar o nível de tensão do sinal de saída do xbee (sinal de entrada do microcontrolador) pois o microcontrolador interpreta corretamente o sinal de 3.3V (Figura 46).

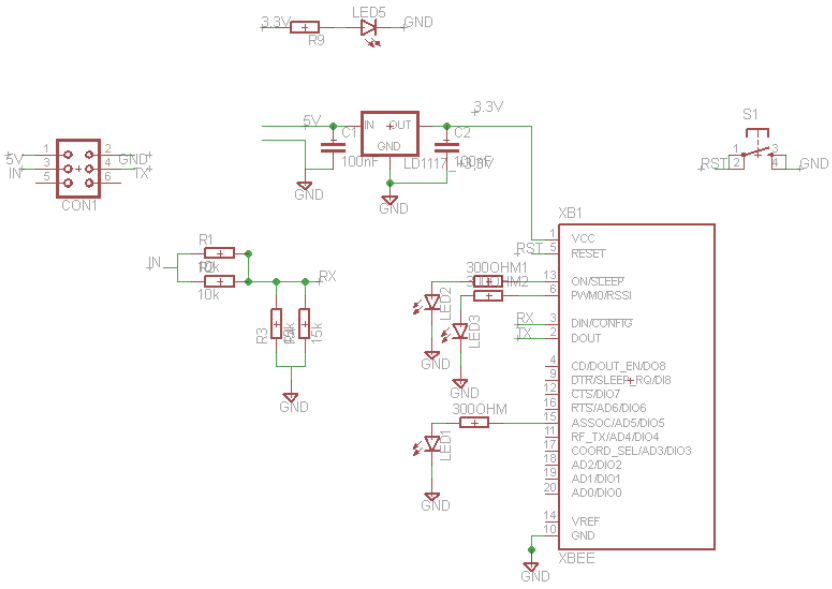


Figura 46 - Esquemático do circuito Xbee.

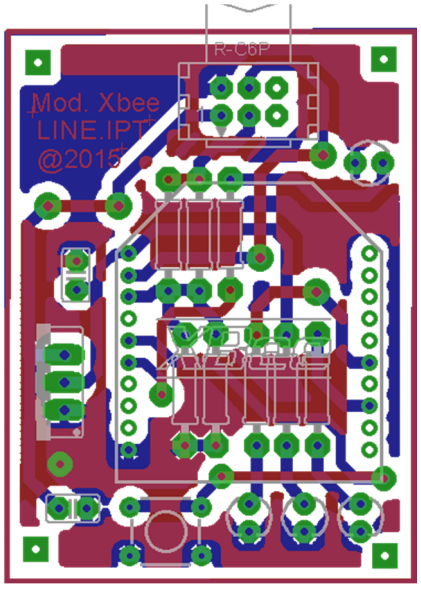


Figura 47 - PCB do circuito Xbee.

A Figura 47 mostra o ficheiro ".brd" do circuito.

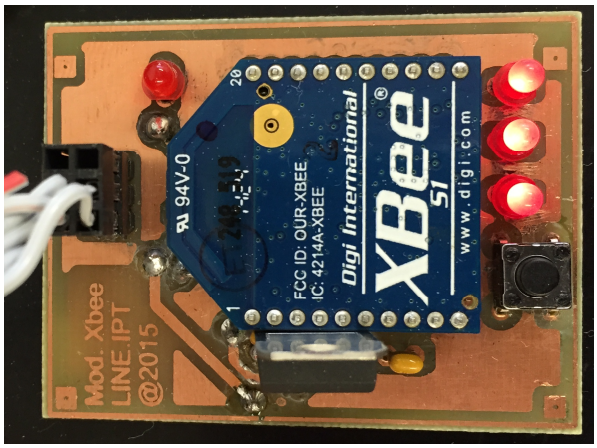


Figura 48 - Circuito desenvolvido.

A Figura 48 mostra o módulo "Xbee" final.

2.2.6 Módulo TDAQ otimizado.

O circuito de leitura de tensão desenvolvido anteriormente dependia de uma fonte de alimentação externa para fornecer a tensão negativa aos "ampops" responsáveis pela retificação da onda da tensão de soldadura. Para eliminar a fonte externa adicionou-se ao circuito uma implementação denominada "Charge Pump" com recurso a um IC "LM555" para gerar a tensão negativa de alimentação dos "ampops". Estes circuitos têm um inconveniente de a corrente de saída estar limitada a um dos condensadores do circuito, mas após testar verificou-se que a corrente consumida pelos dois "ampops" é na ordem dos 6mA. Visto que o "Charge pump" é capaz de fornecer cerca de 25mA, este valor é suficiente para o circuito. Este fornece no máximo 12V de tensão negativa, para manter os "ampops" em equilíbrio colocaram-se reguladores de tensão de 9V nas alimentações garantindo assim o sinal de saída com os dois valores de tensão (positiva e negativa) equilibrados (Figura 49).

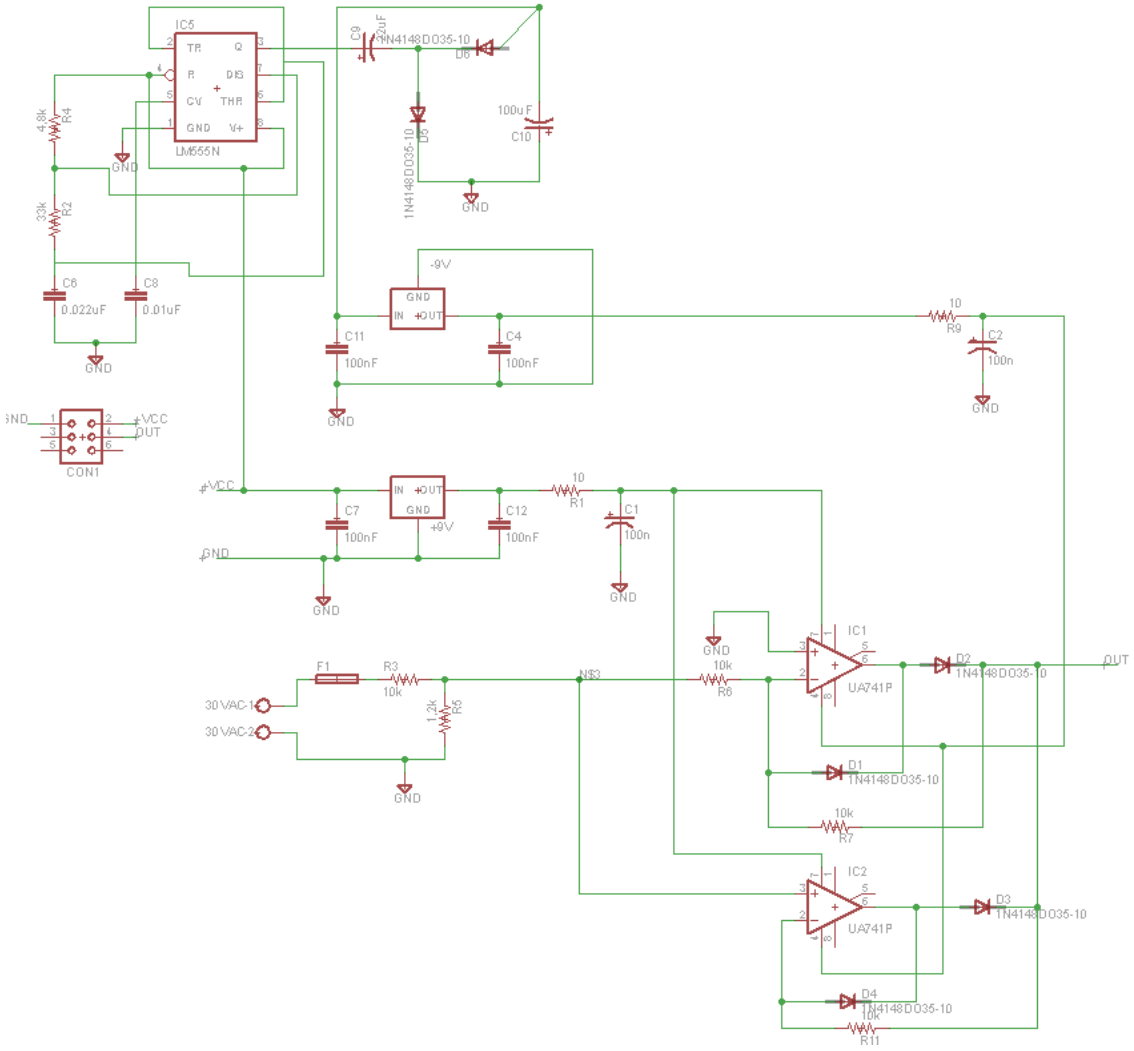


Figura 49 - Esquemático do circuito “TDAQ” otimizado.

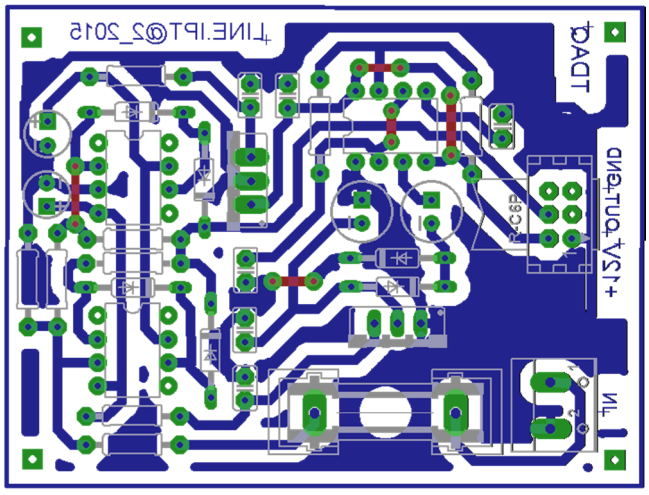


Figura 50 - PCB desenvolvida para o circuito “TDAQ” otimizado.

A Figura 50 mostra a PCB desenvolvida do circuito.

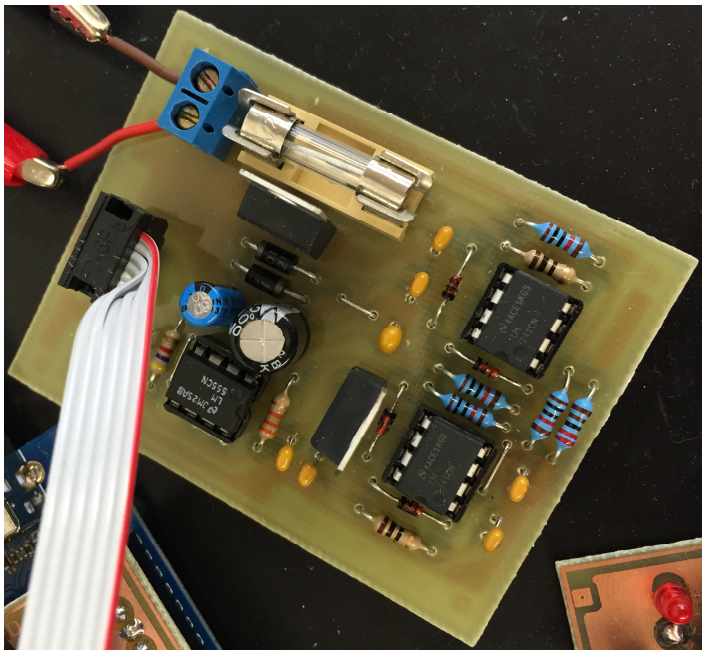


Figura 51 - Circuito "TDAQ" desenvolvido.

A Figura 51 mostra o módulo final desenvolvido.

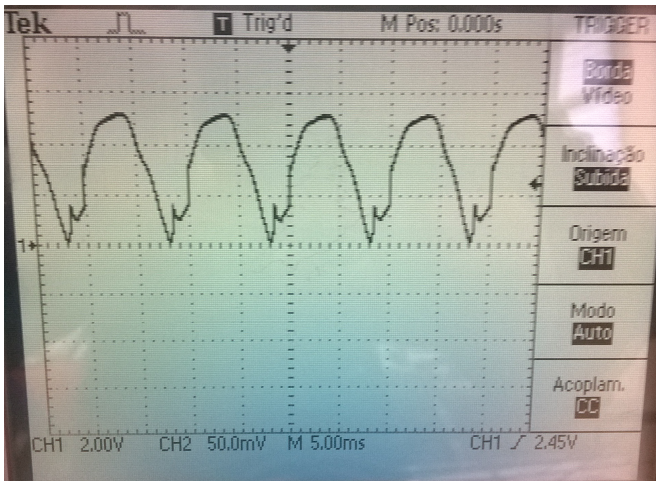


Figura 52 - Sinal de saída do circuito testado em laboratório.

A Figura 52 mostra um sinal de saída de teste do módulo "TDAQ".

2.2.7 Módulo CC3000

Para se testar uma alternativa à "Shield RedFly" para as comunicações "wireless" desenvolveu-se um circuito para integração do módulo "CC3000" da "Texas Instruments" ao protótipo. Utilizou-se uma "breakout-board" da "Adafruit" que já contém toda a

eletrónica essencial ao funcionamento do módulo. Este comunica com o microcontrolador via protocolo “SPI” (Figura 53).

Figura 53 - Breakout board CC3000.

Como o módulo “master” pode conter mais dispositivos que comunicam via “SPI”, colocou-se um IC “74HC15N” que é um “buffer” com o “Enable Pin” para colocar as pistas em baixa ou alta impedância. Deste modo é possível controlar com exatidão o acesso ao módulo por parte do microcontrolador (o protocolo “SPI” contém o canal “CS” (“Slave Select”) para controlar o acesso de vários dispositivos ao mesmo master, mas como as bibliotecas utilizadas poderão não utilizar essa funcionalidade, optou-se por introduzir o “buffer” para separar os sinais) (Figura 54).

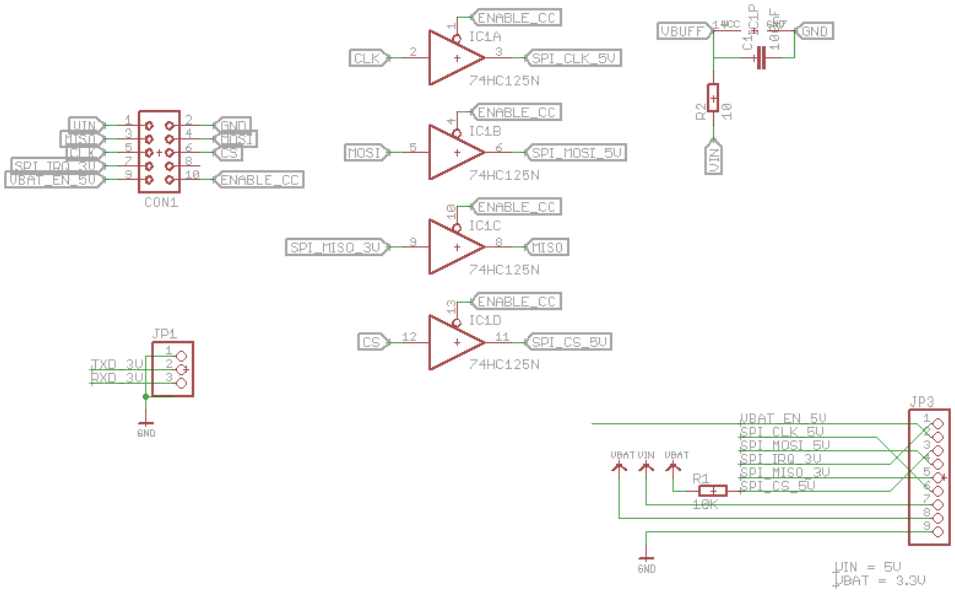


Figura 54 - Esquemático do circuito CC3000.

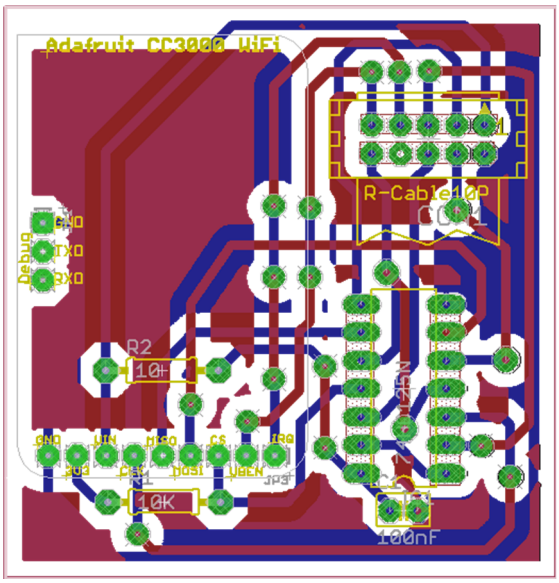


Figura 55 - PCB desenvolvida para o circuito CC3000.

A Figura 55 mostra o ficheiro ".brd" do circuito.

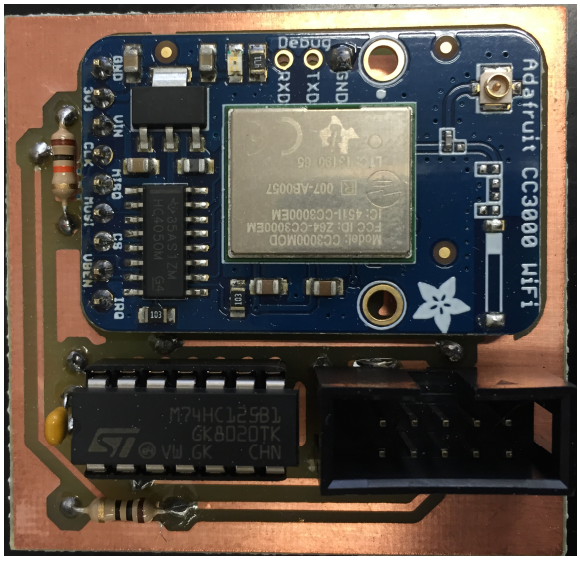


Figura 56 - Circuito CC3000 desenvolvido.

A Figura 56 mostra o módulo final desenvolvido.

2.2.8 Otimização da aplicação SCADA

A aplicação "Scada" necessitou de várias modificações para estar apta e apresentável a funcionar na "MFTE". Introduziram-se caixas de texto com o objetivo de adquirir todos os valores monitorizados e de realizar as suas respetivas conversões para valores reais. Introduziu-se uma visualização gráfica da forma de onda de tensão adquirida sendo esta

"construída" com base nos valores adquiridos (ciclos das diferentes fases, declives e valores médios).

Converteu-se também toda a aplicação para a nova interface "Metro" da "Microsoft" de modo a ter uma apresentação de acordo com os novos sistemas operativos (Figura 57).

Figura 57 - Nova aplicação Scada.

2.2.9 Otimização do funcionamento do sistema

Os circuitos referidos anteriormente foram implementados no protótipo adicionando assim novas funcionalidades essenciais ao cumprimento dos objetivos traçados para o projeto (circuito "RTC", "Xbee", "SD" entre outros).

Implementaram-se também interrupções externas para controlador quando se adquire os dados de uma soldadura, ou seja, apenas quando o botão é pressionado é que se adquire dados, reduzindo assim o número de valores "falsos" adquiridos. Desde que se pressiona o botão de soldadura até as pinças fecharem e soldadura demora aproximadamente 2 segundos, por isso adicionou-se uma interrupção para contar o tempo a partir do momento em que é pressionado o botão. Se o tempo for superior a 3 segundos e não se adquiriram dados não ocorreu soldadura e por isso faz-se o "reset" à "flag" responsável pelo comando da soldadura para voltar ao estado inicial.

Implementou-se o envio do "Ok/Not Ok" da soldadura do "MASTER" para o "SLAVE" de modo a dar a indicação ao operador da máquina se a soldadura está correta ou não. O "MASTER" recebe os dados do "SLAVE" do secundário e do primário compara os valores

recebidos com os parâmetros configurados e envia via “Xbee“ para o “SLAVE” do secundário a indicação se a soldadura está correta ou não.

2.2.10 Sinal do botão de soldadura

De modo a inserir o controlo de aquisição de soldadura pelo botão da máquina, e a identificação do programa de soldadura através do botão comutador investigaram-se as suas ligações. Verificou-se que o sinal proveniente do botão de soldadura está em série com o botão de comutação de programa sendo por isso um canal independente para cada programa (Figura 58).

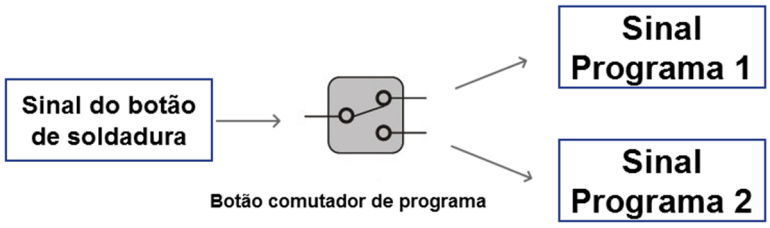


Figura 58 - Funcionamento geral do interruptor (botão de soldadura).

No terreno observou-se que ambos os sinais têm aproximadamente 11V de tensão e que quando o botão não é pressionado encontram-se com este valor. Quando o botão é pressionado o sinal do canal selecionado vem a zero e é este valor que dá a indicação ao controlador da máquina para iniciar uma soldadura.

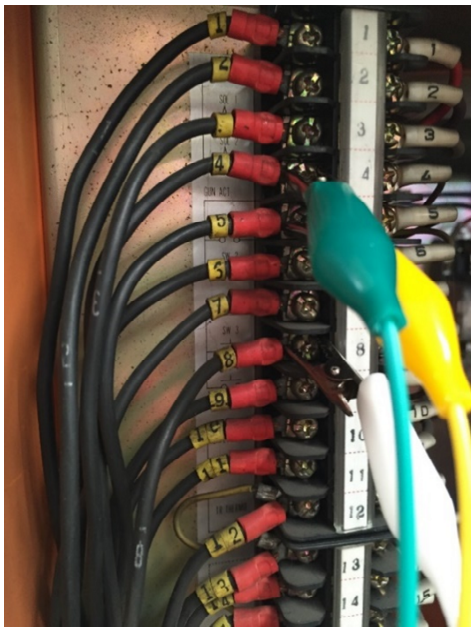


Figura 59 - Ligações de ambos os sinais do botão.

A Figura 59 mostra as ligações no controlador.

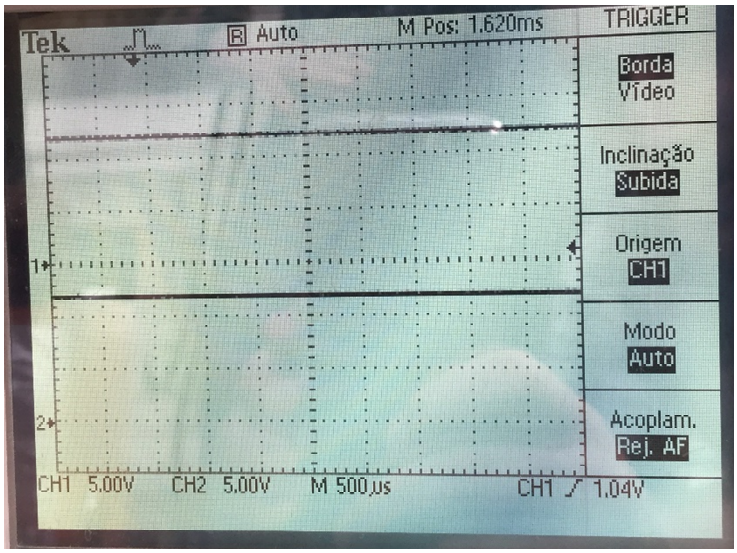


Figura 60 - Visualização de ambos os sinais sem o botão pressionado.

A Figura 60 mostra ambos os sinais do botão de soldadura na situação de não pressionado.

No microcontrolador o processo de soldadura é monitorizado com recurso à utilização de interrupções externas, ou seja, cada vez que o botão for pressionado dispara uma interrupção que dá a indicação para iniciar a aquisição dos dados. Como existem dois sinais possíveis, configuraram-se duas interrupções externas para assim ser possível identificar qual o programa de soldadura selecionado.

Para garantir que o pulso que ativa a interrupção seja “limpo” de interferências desenvolveu-se um circuito com recurso a um IC “schmitt trigger” para que quando o sinal de um dos dois canais baixar o seu valor de tensão (quando o botão é pressionado) enviar um pulso “limpo” com um valor de 5V para ativar a interrupção.

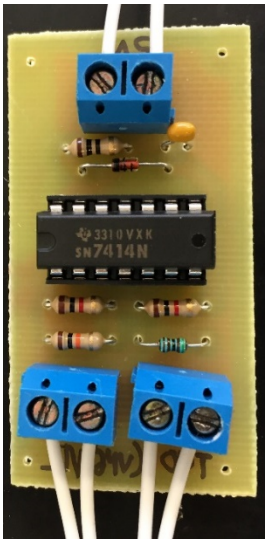


Figura 61 - Placa Schmitt trigger.

A Figura 61 mostra o módulo inicial “Schmitt Trigger” desenvolvido.

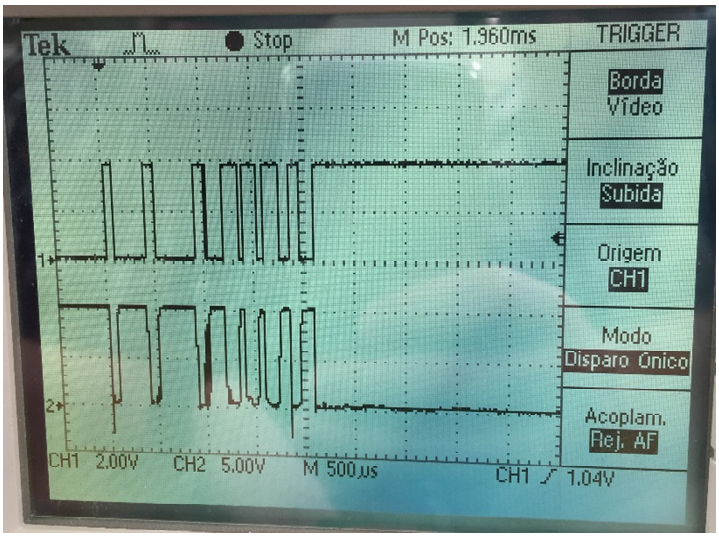


Figura 62 - Canal1: Saída para o circuito / Canal 2 - Sinal do botão de soldadura.

A Figura 62 mostra os sinais de entrada e saída do módulo.

Ao testar verificou-se que quando o sinal do botão de soldadura vem a zero o circuito gera um pulso “perfeito” com um valor de aproximadamente 4.5V. Relativamente ao “boucing” do sinal introduziram-se no algoritmo proteções para não prejudicar a aquisição dos dados,

ou seja, assim que é detetado o primeiro pulso o microcontrolador não considera nenhuma nova soldadura enquanto a atual não for processada independentemente dos pulsos gerados.

Utilizando os circuitos desenvolvidos até este ponto e com recurso ao "Matlab" adquiriram-se os sinais dos sensores instalados (pressão, tensão e corrente) à saída dos circuitos desenvolvidos.

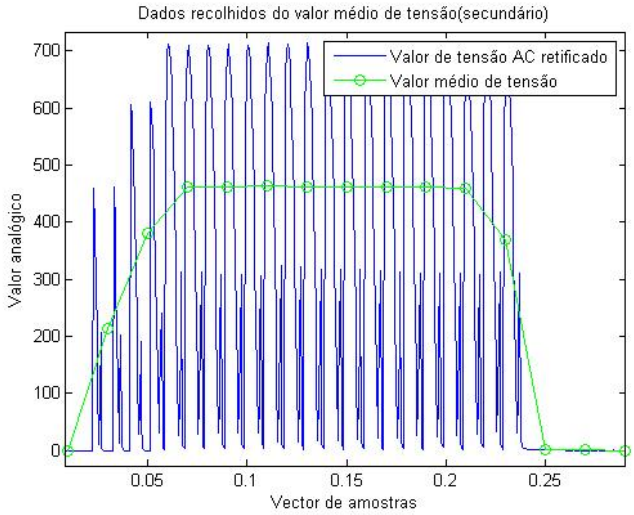


Figura 63 - Valor de tensão AC retificado.

Pode ver-se que o valor de tensão à saída do circuito "TDAQ" encontra-se corretamente retificado (Figura 63).

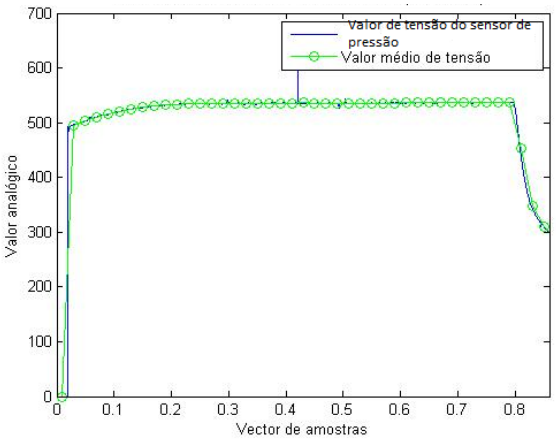


Figura 64 - Sinal do sensor de pressão.

A (Figura 64) mostra a fase inicial de subida do valor de pressão até que estabiliza num valor médio durante a soldadura.

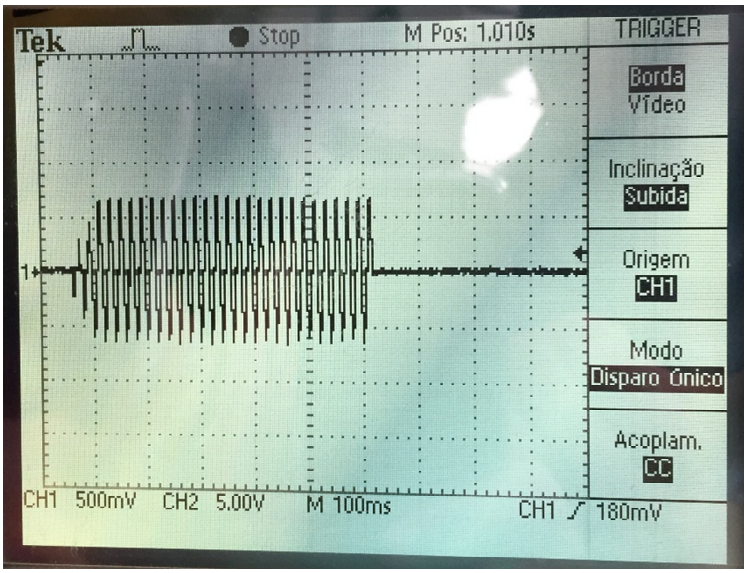


Figura 65 - Sinal de tensão AC da sonda de corrente.

A Figura 65 mostra o sinal da sonda de corrente visualizado no osciloscópio.

A resistência de amostragem a colocar no transformador de intensidade para monitorizar a intensidade da corrente durante uma soldadura terá de ter um valor superior (neste momento tem 1.5 Ohm) à implementada atualmente, para que o sinal de tensão gerado pelo transformador de intensidade possua um valor próximo de 5V para assim ser diretamente retificado pelo circuito "TDAQ" sem haver a necessidade de ajustar o valor de tensão posteriormente.

Obtidos os resultados e as conclusões procederam-se às alterações em conformidade. Encontrando-se assim o protótipo perto da sua versão final. Inseriram-se os circuitos em caixas de teste, efectuaram-se todas as ligações necessárias e procederam-se a testes em laboratório de modo a tentar simular ao máximo o cenário de soldadura real.

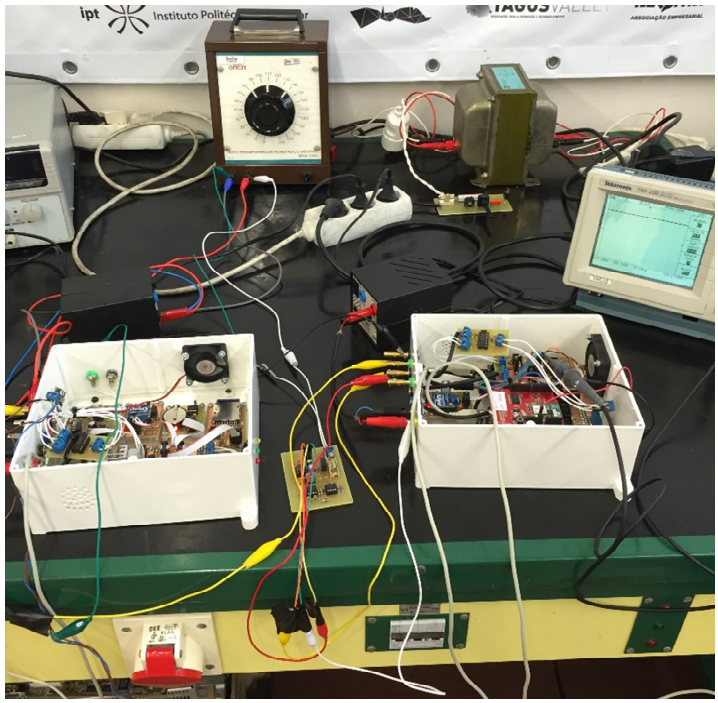


Figura 66 - Novo protótipo.

A Figura 66 mostra o cenário implementado em laboratório.

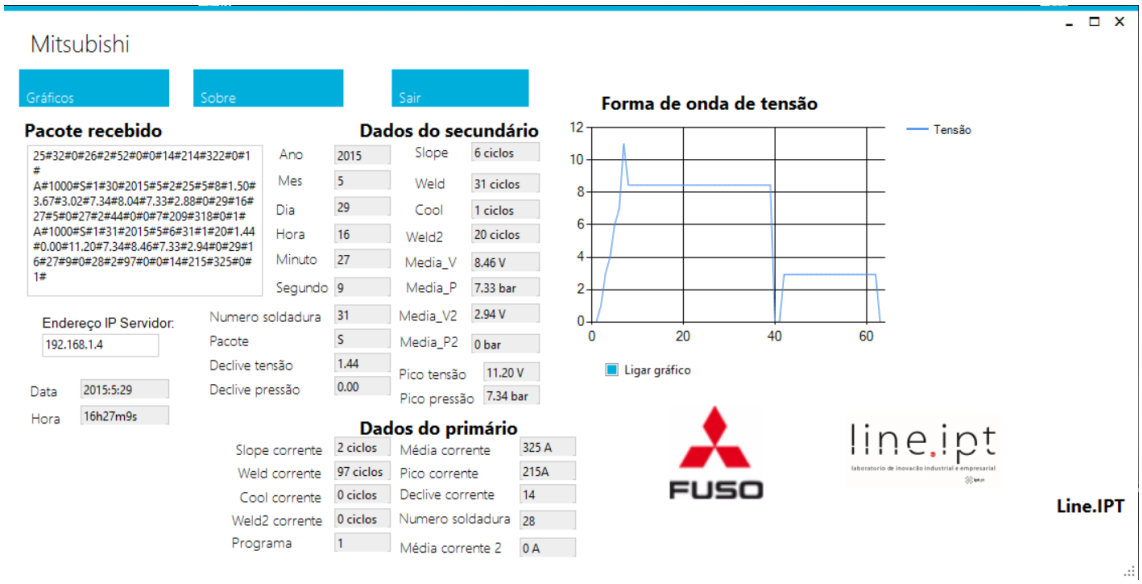


Figura 67 - Aplicação Scada em funcionamento.

Até este ponto a aplicação "Scada" já recebe todos os dados do primário e secundário por soldadura estando assim bem sincronizada (Figura 67).

2.2.11 V2 Schmitt trigger com isolamento de sinal

O acesso aos sinais do botão de soldadura da máquina em ambos os módulos faz com que as massas entre os módulos estejam ligadas entre si podendo isso causar interferências e comportamentos irregulares devido à queda de tensão nos cabos com a distância dos mesmos deixando também de existir isolamento galvânico. Para isso otimizou-se o circuito "Schmitt trigger" para isolar a massa do sinal em ambos os módulos recorrendo a "optocouplers" na entrada do sinal no circuito. Deste modo a massa do circuito fica isolada da massa do sinal (Figura 68).

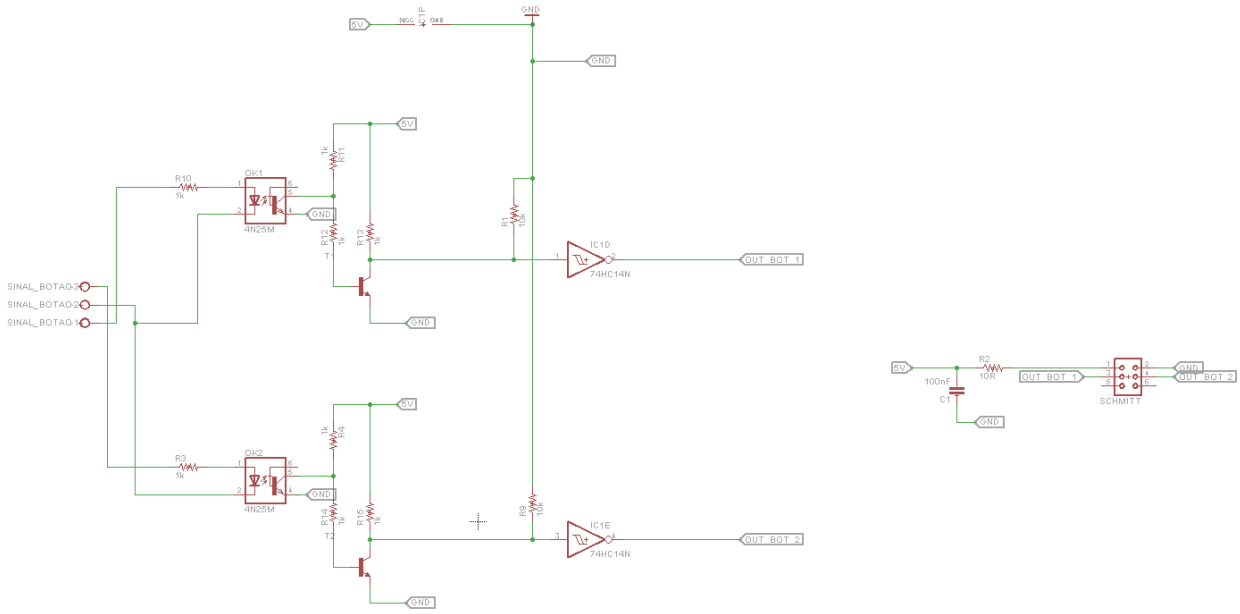


Figura 68 - Esquemático da versão 2 do circuito "Schmitt Trigger".

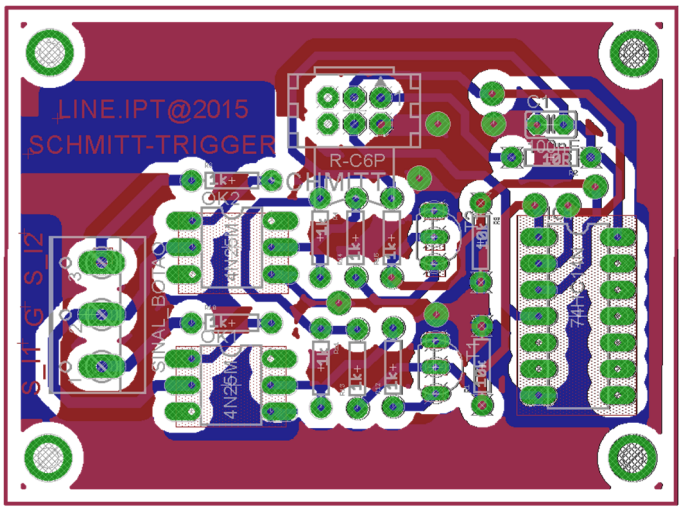


Figura 69 - PCB desenvolvida do "Schmitt Trigger".

A Figura 69 mostra a PCB do circuito desenvolvido.

Ao testar este circuito no terreno verificou-se que em alguns controladores a ligação ao "optocoupler" ativava inadvertidamente a máquina de soldadura. Ao observar o problema com mais detalhe verificou-se que o "optocoupler" utilizado ("4N25") consome cerca de 60mA para entrar em funcionamento, foi portanto necessário substituir estes por outros que consomem menos corrente. A escolha recaiu no "optocoupler" "HCPL-4731" que é "ultra-low current" necessitando no mínimo de 40uA de corrente para funcionar. Para adquirir a amostra do sinal de tensão com menos corrente possível realizaram-se os seguintes cálculos para a obtenção da resistência de entrada do "optocoupler":

$$V_{entrada} = R_{entrada} * I\_entrada \quad (4)$$

Para obter um valor de corrente de 50uA (deixando uma margem dos 40uA mínimos necessários):

$$11V = R_{entrada} * 50uA \quad (5)$$

$$R_{entrada} = \frac{11V}{50uA} \quad (6)$$

$$R_{entrada} = 220K \quad (7)$$

Com este valor de corrente consegue-se monitorizar com sucesso o valor de tensão do botão de soldadura sem perturbar o funcionamento do controlador devido ao valor elevado da resistência de entrada.

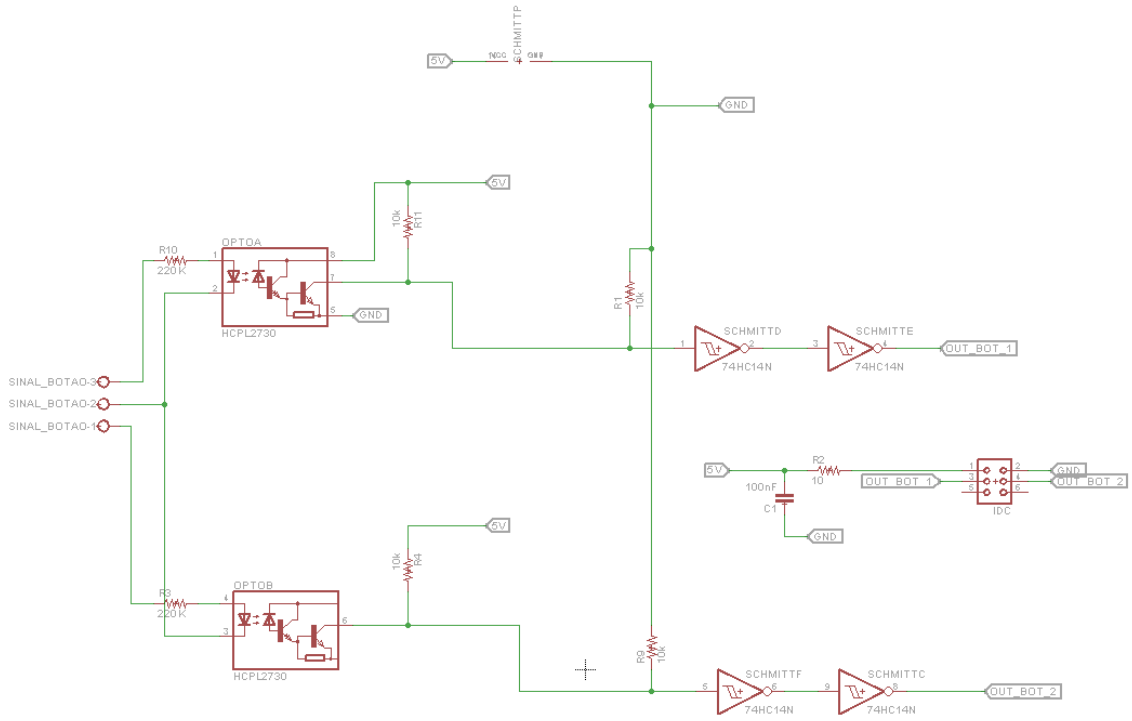


Figura 70 - Esquemático do circuito "Schmitt Trigger" final.

A Figura 70 mostra o esquemático do circuito "Schmitt Trigger" final com os "optocoupler" de "ultra-low current". Deve-se ainda referir que este "optocoupler" tem dois canais de entrada, evitando assim colocar dois "optocoupler" para ambos os sinais do botão de soldadura.

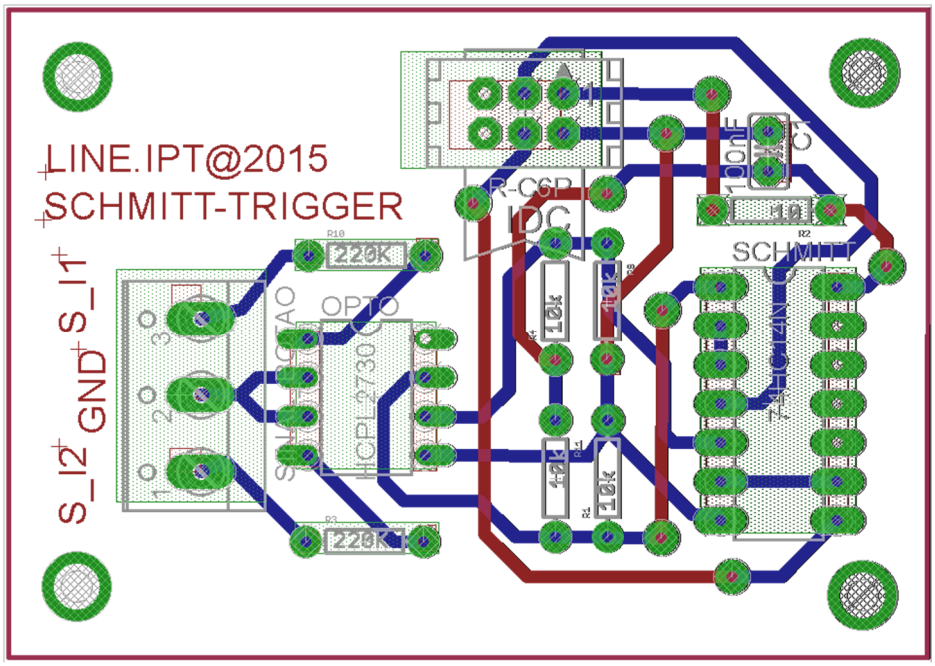


Figura 71 - PCB desenvolvida do circuito "Schmitt Trigger final".

A Figura 71 mostra a PCB desenvolvida do circuito “Schmitt Trigger” final.

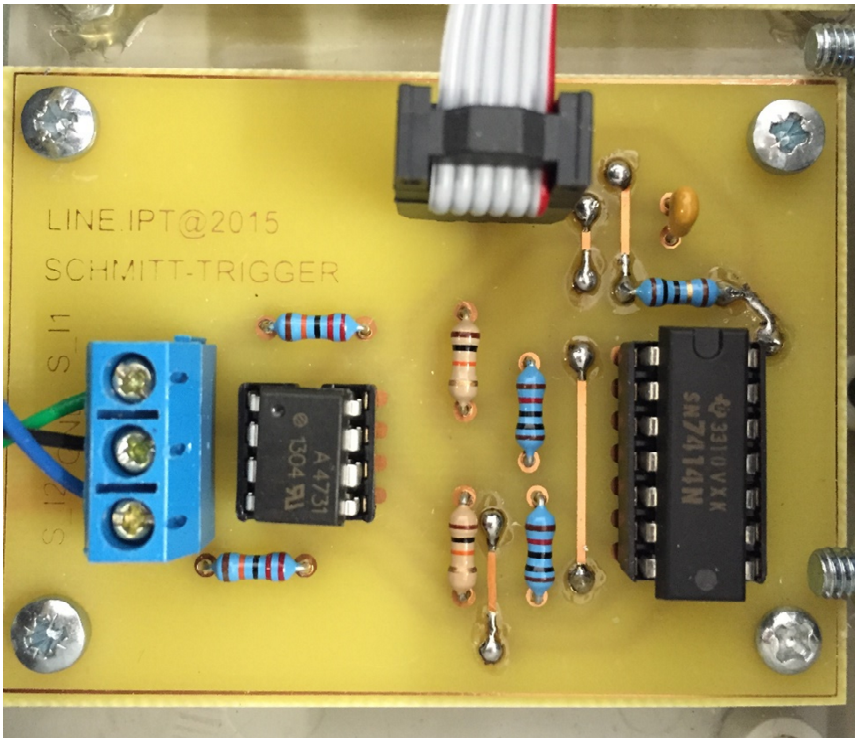


Figura 72 - "PCB” desenvolvida.

A Figura 72 mostra a “PCB” desenvolvida e montada na caixa dos protótipos.

2.2.12 Driver e indicação luminosa

Para realizar a indicação luminosa ao operador do estado da soldadura recorreu-se a leds "rgb" que com a cor verde e vermelha dão a indicação da soldadura "OK/not OK" (Figura 73).

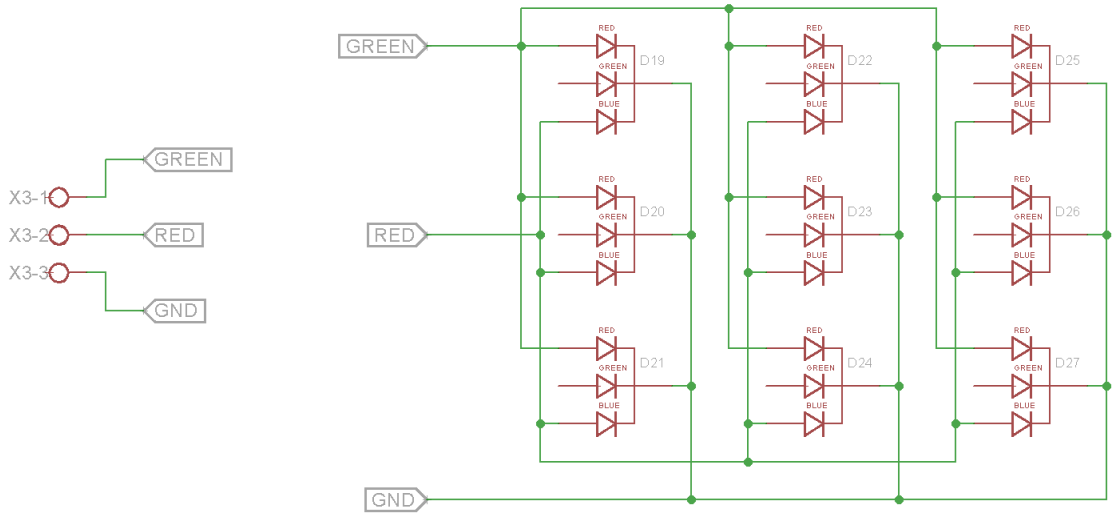


Figura 73 - Esquemático do módulo dos leds "rgb".

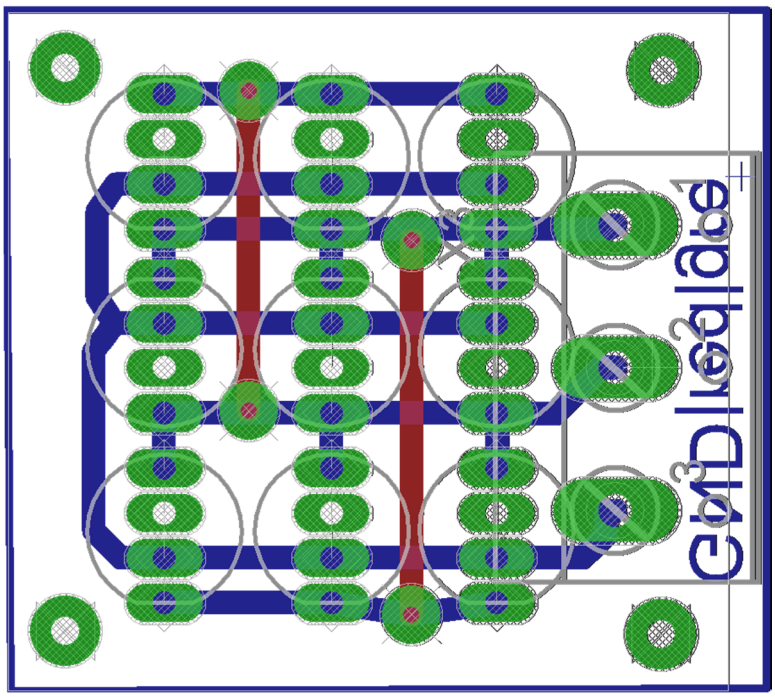


Figura 74 - "PCB" desenvolvida.

A Figura 74 mostra a "PCB" desenvolvida para o circuito dos leds "rgb".

Como driver para os leds desenvolveu-se um pequeno circuito com transístores para ligar/desligar os leds através da utilização das portas digitais do microcontrolador.

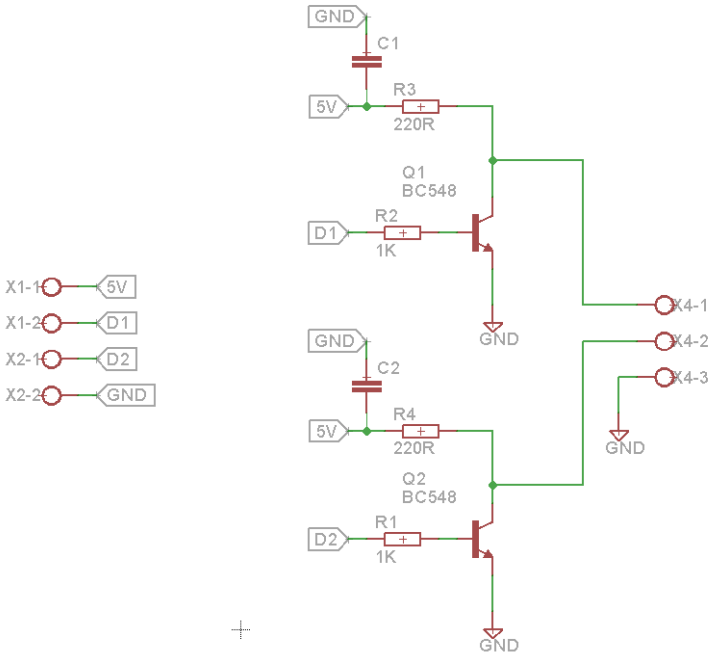


Figura 75 - Esquemático do driver dos leds.

A Figura 75 mostra o circuito do driver para os "leds".

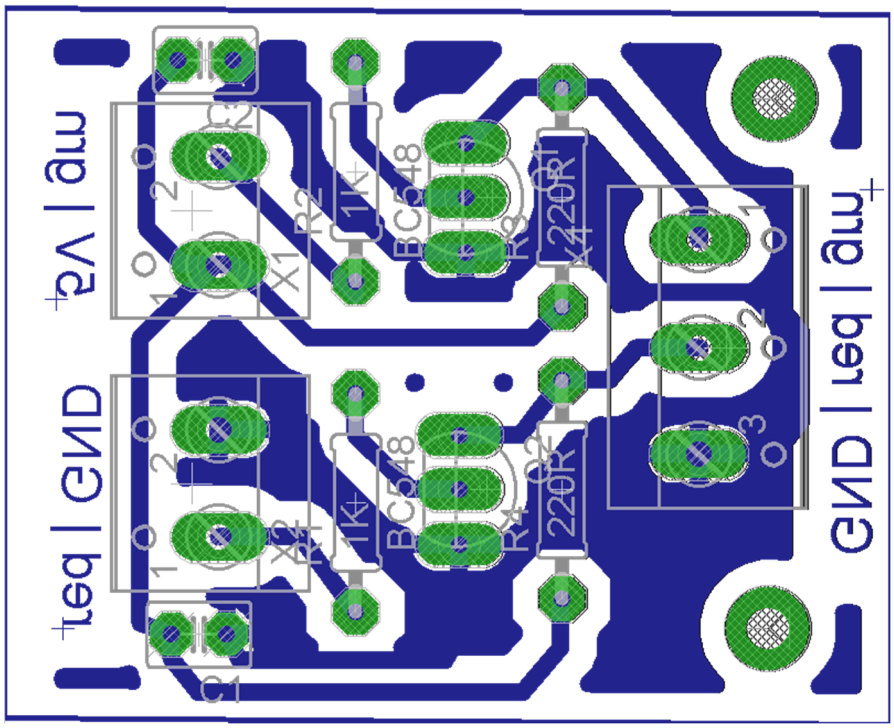


Figura 76 - PCB desenvolvida.

A Figura 76 mostra a “PCB” desenvolvida em “software”.

Como suporte para o módulo dos leds desenvolveu-se uma caixa utilizado a nova impressora 3D adquirida pelo LINE sendo assim possível prender a caixa à máquina de soldadura protegendo ao mesmo tempo os leds de danos (Figura 77 e Figura 78).

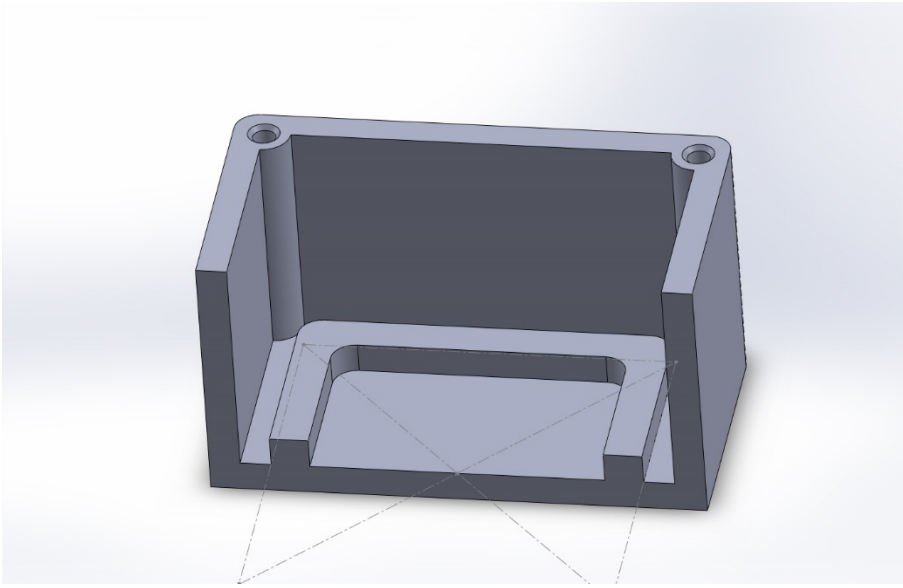


Figura 77 - Caixa desenvolvida em “Solidworks” com recorte ao centro.

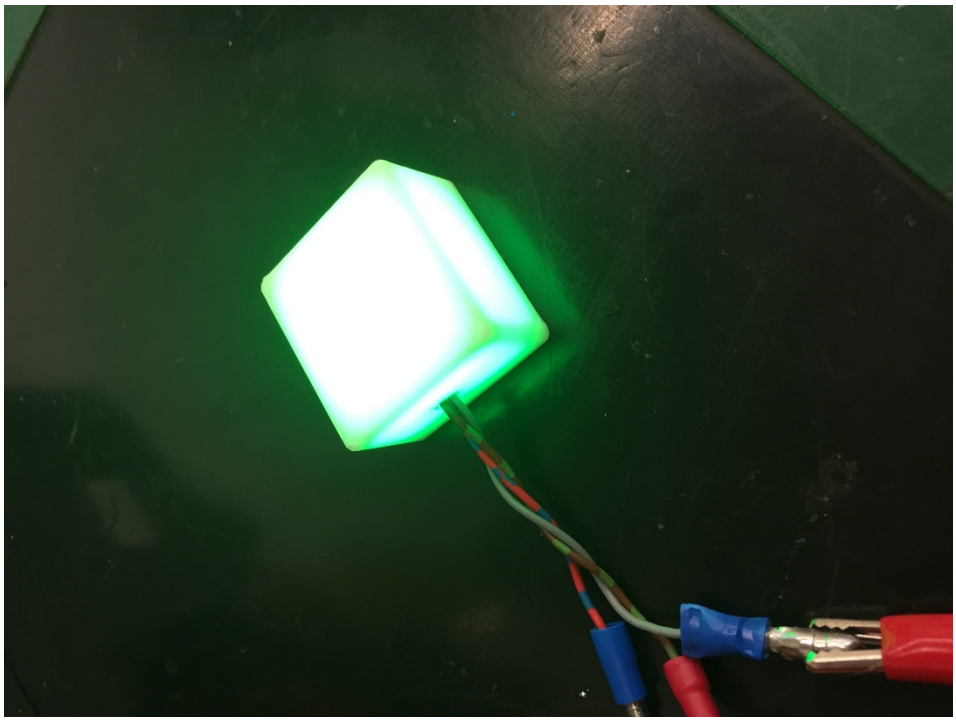


Figura 78 - Caixa com leds montados.

2.2.13 Aplicação “SCADA” desenvolvida em “Python”

Devido a alguns problemas na aplicação “Scada” desenvolvida em visual basic, nomeadamente na amostragem de valores nos gráficos e na falta de opções para personalizar os mesmos, foi necessário desenvolver uma aplicação “SCADA” numa linguagem alternativa. Para isso recorreu-se à linguagem de programação “Python” nomeadamente a versão 2.7.10. Como “IDE” de desenvolvimento utilizou-se o “Pycharm” (Figura 80) que oferece um ambiente de programação bastante completo. Para a amostragem das formas de onda da soldadura utilizou-se o módulo “Matplotlib” que oferece um leque vasto de configurações e personalização dos gráficos à imagem do “Matlab”. Para ambiente gráfico da aplicação escolheu-se a plataforma “Qt4” que tem uma grande compatibilidade com a linguagem “Python” devido ao módulo “Pyqt”. Esta interface oferece um ambiente gráfico moderno e foi personalizada com recurso à ferramenta “Qt creator”.

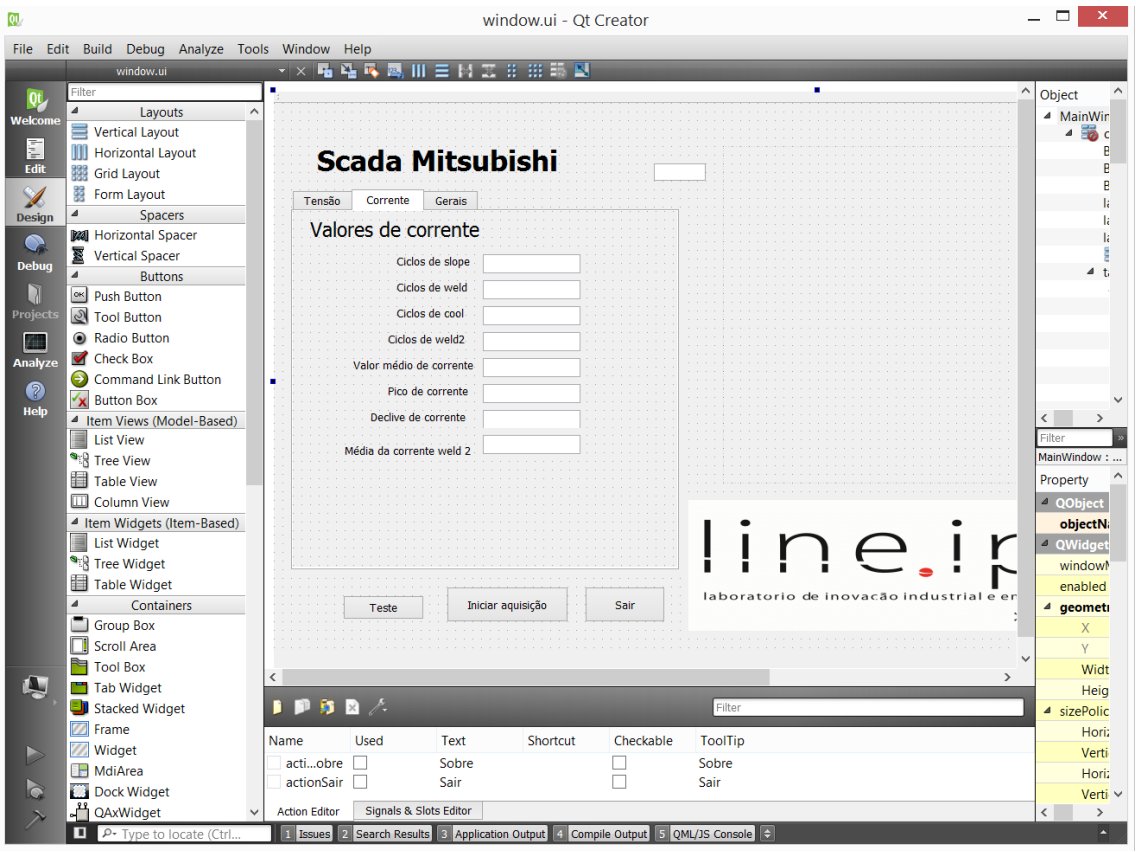


Figura 79 - Qt creator.

A interface gráfica é desenvolvida inicialmente nesta aplicação (Figura 79) que gera um ficheiro do tipo “.ui” que é posteriormente importado no código “Python”.

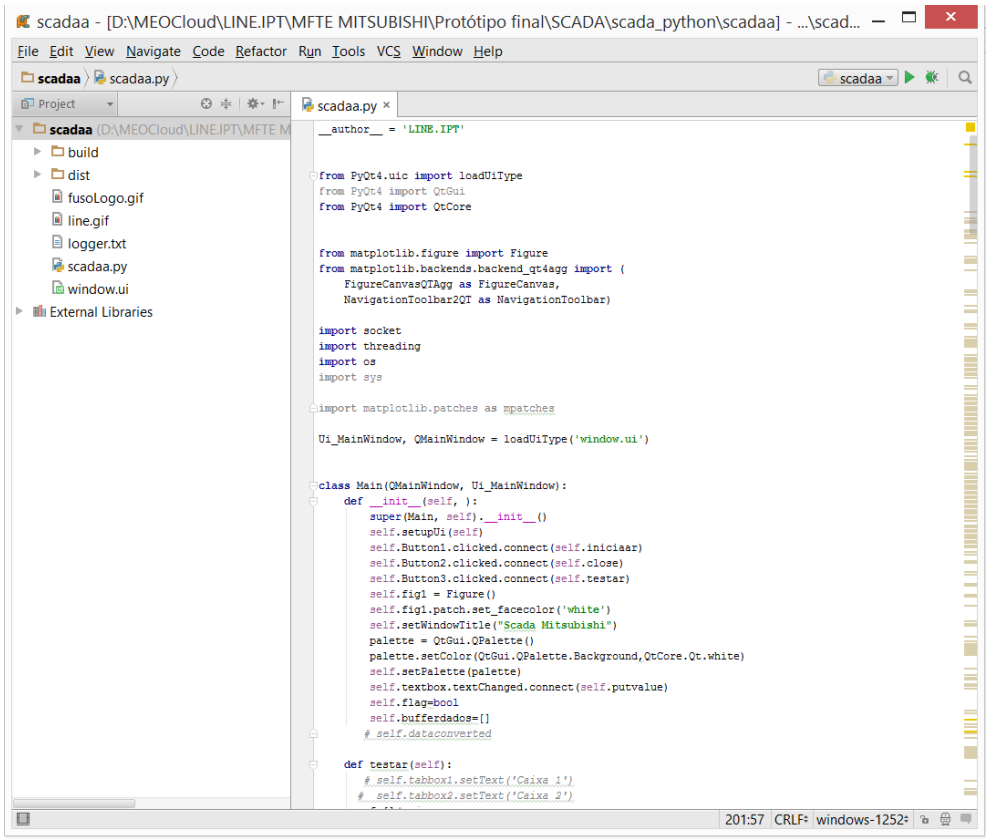


Figura 80 - Aplicação python desenvolvida na plataforma "Pycharm".

A Figura 80 mostra a interface de programação “Pycharm”.

Tal como a aplicação em “Visual Basic”, esta comunica com o protótipo via “TCP/IP” através de um “socket” de comunicação. A aplicação “Python” funciona como servidor da rede e o protótipo como cliente. A aplicação recebe os dados e coloca os valores nos campos específicos. Posteriormente esta guarda esse pacote de dados num ficheiro de texto “.txt” para ser importado para uma base de dados.

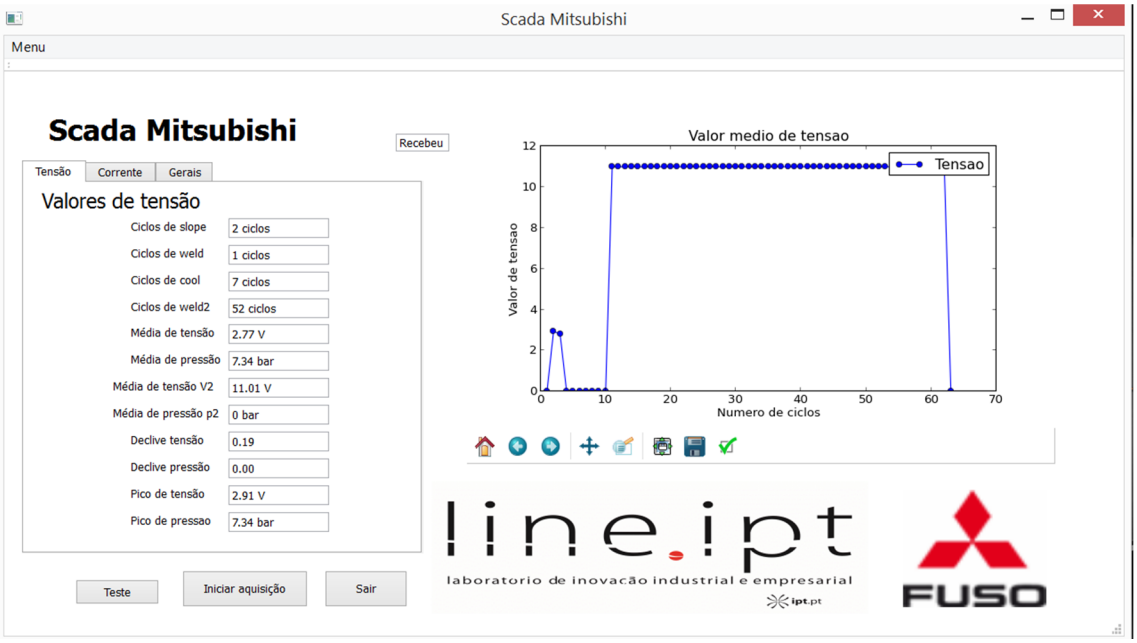


Figura 81 - Aplicação Scada em funcionamento.

Esta aplicação (Figura 81) apresenta um aspeto mais funcional do que a aplicação desenvolvida anteriormente, bem como gráficos mais personalizáveis e com mais informação. Para facilitar o "debug" e análise da aquisição de dados dos protótipos desenvolveu-se uma aplicação ("script") para aquisição de dados via "Série" e amostragem em "tempo real" dos valores adquiridos. Para a comunicação serial utilizou-se o módulo "Pyserial", para amostragem gráfica utilizou-se também o módulo "Matplotlib". Para a amostragem dos dados em "tempo real" utilizou-se o módulo "DrawNow". É importante ainda referir que amostragem nunca é totalmente em "tempo real", sendo portanto esta uma visualização mais rápida e eficiente dos valores (Figura 82).

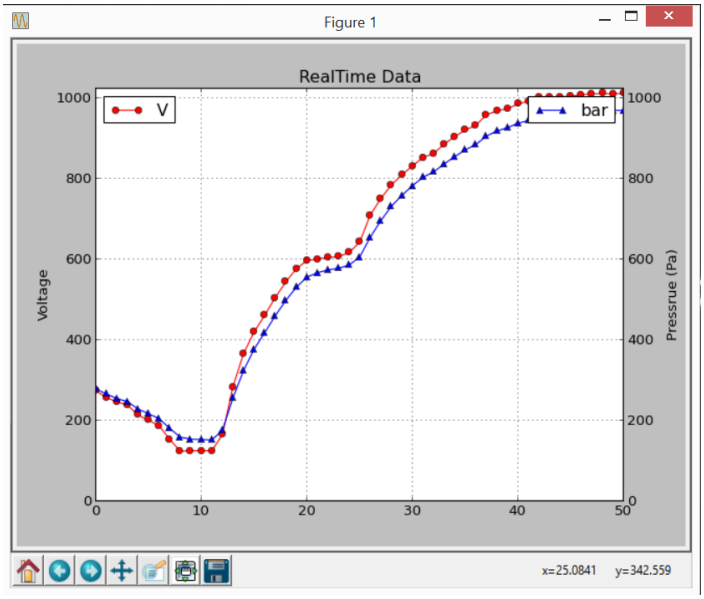


Figura 82 - Aplicação de amostragem em "tempo real".

Deste modo a análise e "debug" são bastante mais rápidos e fáceis pois não é necessário recorrer ao "Matlab" que era a ferramenta anteriormente usada.

2.2.14 Algoritmo otimizado

Com todas as ultimas implementações no sistema foi também necessário adaptar e modificar o algoritmo de forma a implementar as ultimas funcionalidades. Foram criadas duas interrupções externas para identificar os dois programas de soldadura e para dar início ao processo de aquisição de soldadura. De referir que todo o código referente às funcionalidades "criticas" (interrupções externas e timers) foi desenvolvido acedendo diretamente aos registos de configuração como se se tratasse de um microcontrolador "puro" da "Atmel". Deste modo tem-se um controlo total de todas as configurações que se pretende, bem como um aumento de desempenho do microcontrolador. Os pacotes de dados foram otimizados e corrigidos visto que já se tem bem definido o numero de variáveis a enviar. Todo o algoritmo foi revisto e otimizado para corrigir "bugs" e melhorar o funcionamento global. A velocidade do "baudrate" do "Xbee" foi também aumentada para se ter uma comunicação mais rápida.

A Figura 83 mostra o fluxograma simplificado do funcionamento dos módulos "Slave" do primário e secundário. Representa-se também na Figura 84 o fluxograma simplificado do funcionamento do módulo "Master". Este espera pela receção dos pacotes de dados de ambos os "Slave", junta-os num só, incrementa o número de soldadura e envia para a

aplicação “SCADA”. Nesta fase, em laboratório pode-se concluir que as “falsas” soldaduras foram eliminadas e a dessincronização dos dados foi também eliminada. O algoritmo de gravação de dados para cartão “SD” quando falham as comunicações foi otimizando passando apenas a gravar num ficheiro “.txt” todas as soldaduras em que ocorreu erro de envio, para posteriormente poderem ser enviadas para a base de dados.

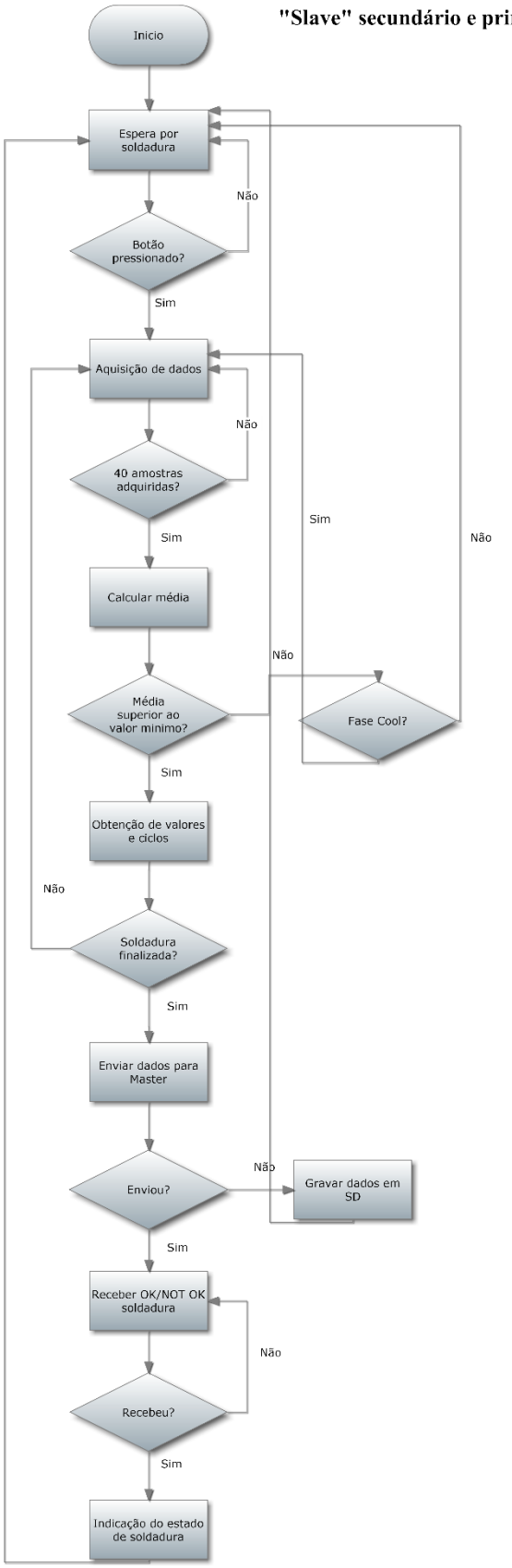


Figura 83 - Fluxograma do módulo slave do secundário e do slave do primário.

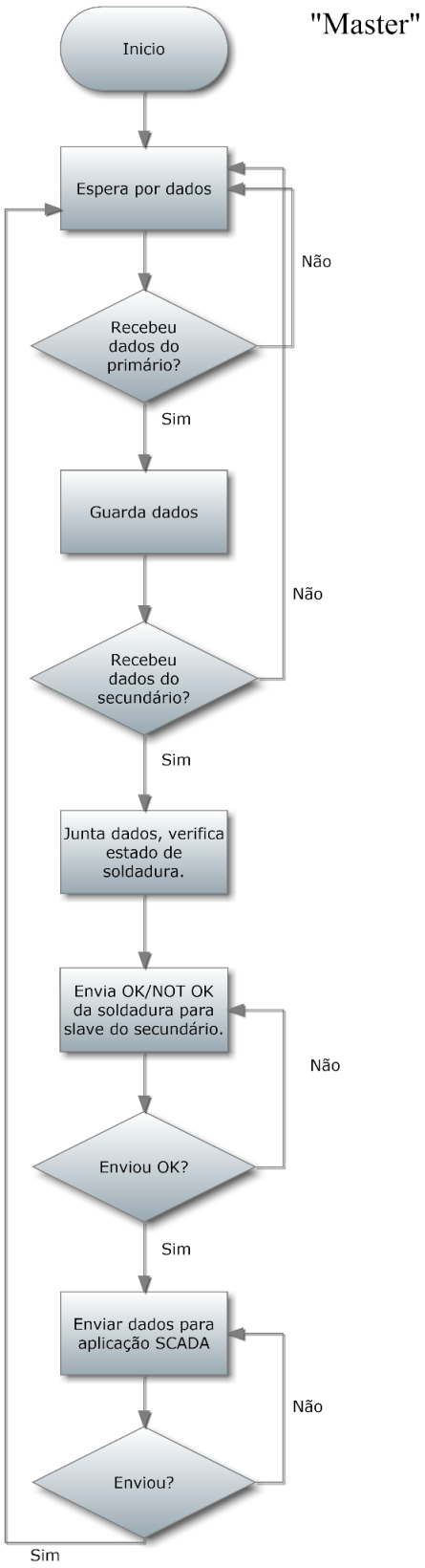


Figura 84 - Fluxograma do módulo "Master".

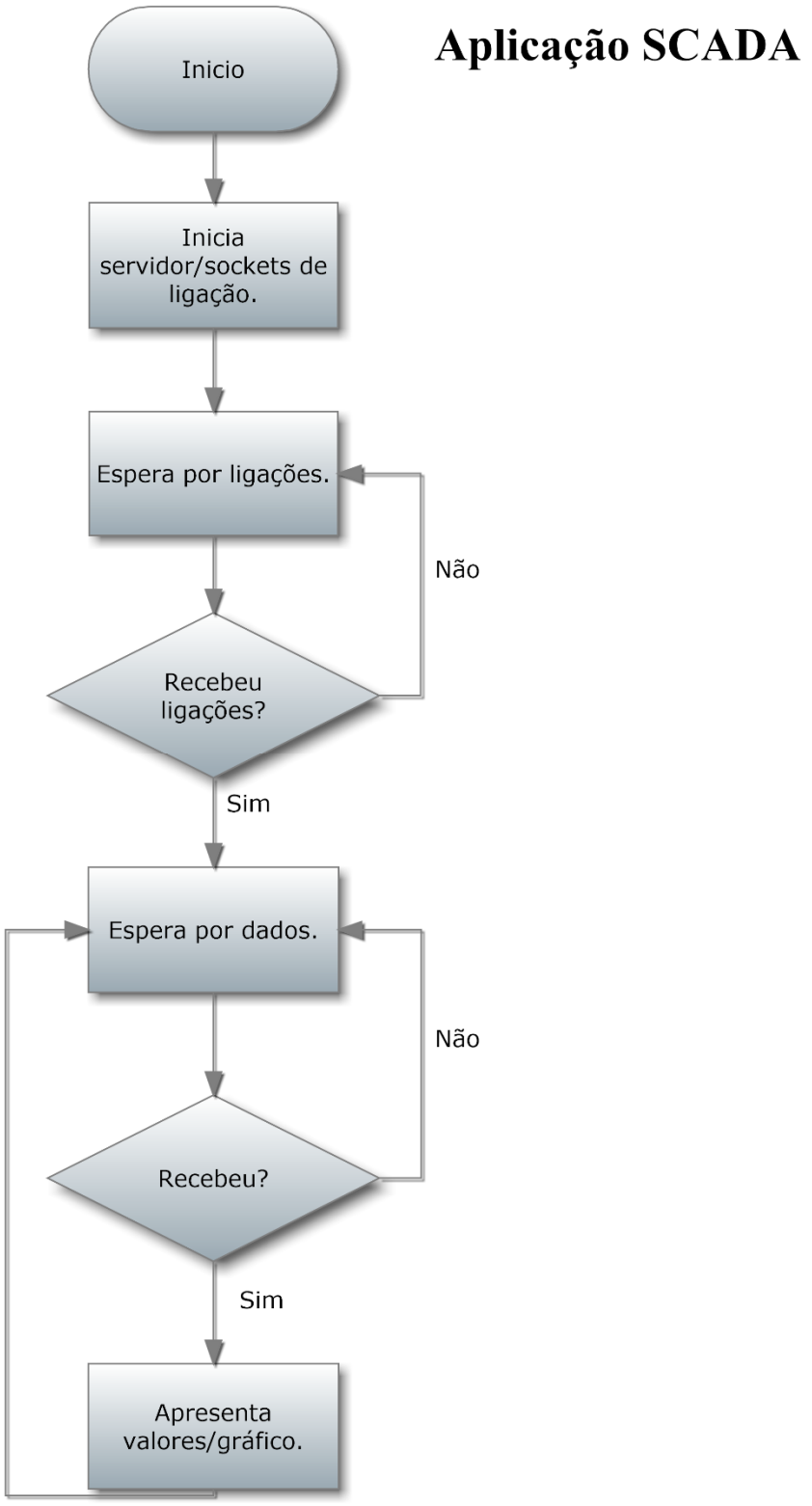


Figura 85 - Aplicação "SCADA".

A Figura 85 apresenta o fluxograma simplificado do funcionamento da aplicação "SCADA". Assim que a tecla de início de aquisição é pressionada são inicializados dois processos

individuais (threads), um para cada "socket" de comunicação que se encontra estabelecido para cada protótipo. Assim que as comunicações estiverem estabelecidas, sempre que a aplicação receber dados esses são apresentados na interface gráfica, sendo ao mesmo tempo guardados num ficheiro de texto.

2.2.15 Finalização de dois protótipos para testes

Como o desenvolvimento em laboratório para esta fase está finalizado, acordou-se com a Mitsubishi em implementar dois protótipos finais em duas máquinas de soldadura. O objectivo é efetuar um teste durante vários dias em ambiente real de produção. Pretende-se assim verificar o comportamento do protótipo bem como a fiabilidade dos dados adquiridos. Deste modo montaram-se e contruíram-se dois protótipos com todas as novas otimizações e prontos a serem testados.

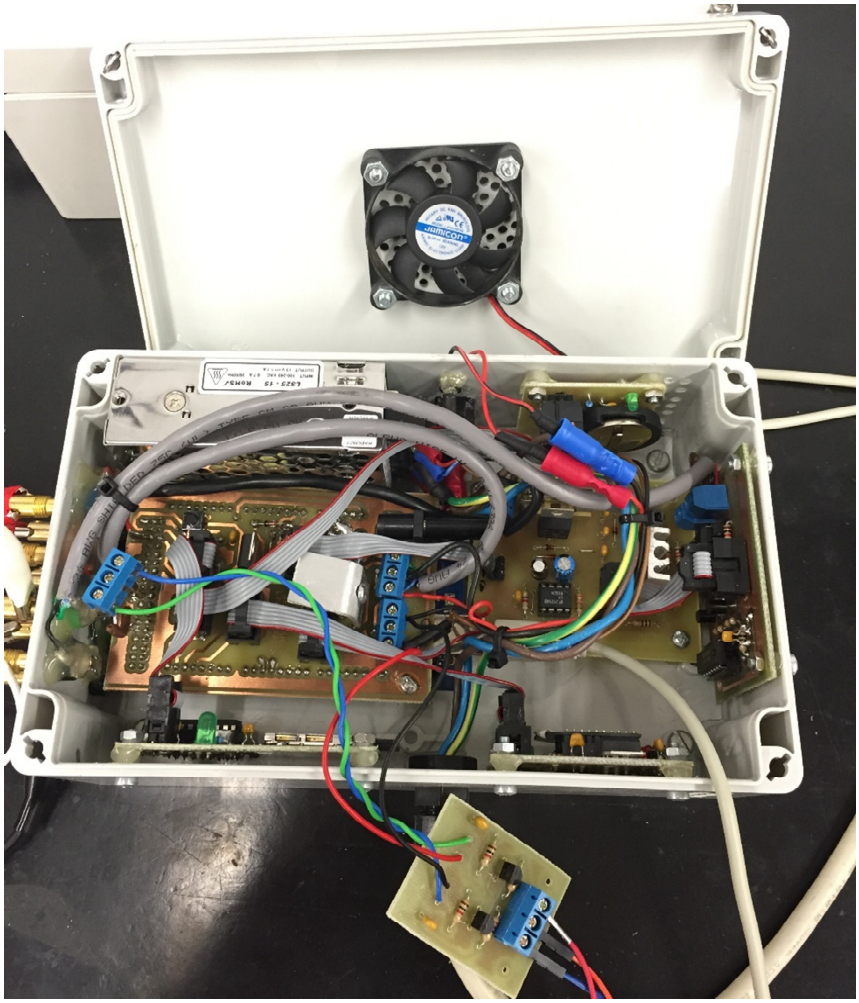


Figura 86 - Módulo "slave" do secundário.

A Figura 86 mostra o interior do módulo “slave” do secundário.

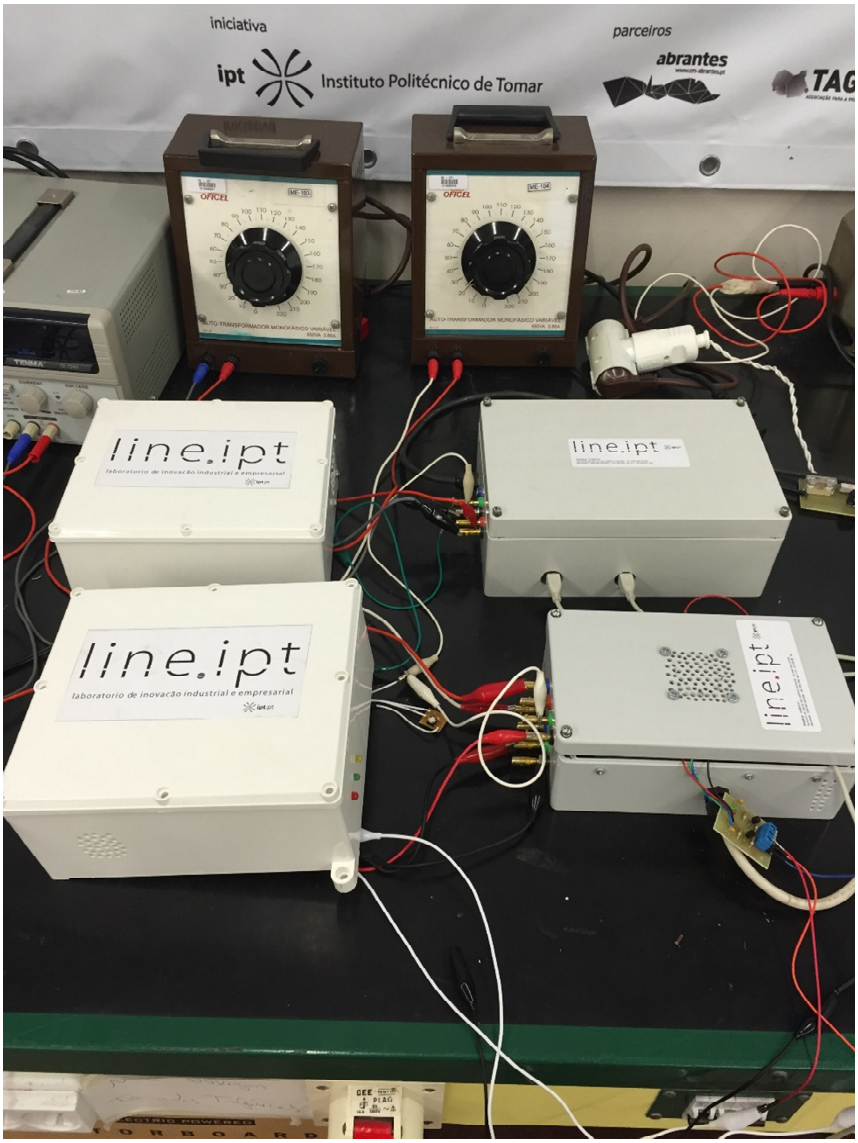


Figura 87 - Os dois protótipos finalizados.

A Figura 87 mostra os dois protótipos finalizados e montados.

É importante referir que o sensor de pressão para o segundo protótipo foi diferente do primeiro (“Aplisens”) devido às dimensões serem superiores às pretendidas pela empresa. Desta vez a escolha recaiu sobre a “Schneider” nomeadamente no modelo “XMLP010BD21V” (

Figura 88) que comparativamente com o da “Aplisens” possui exatamente o mesmo funcionamento e o mesmo nível de sinais, mas com dimensões bastante mais reduzidas.

Figura 88 - Sensor de pressão "Schneider".

Figura 89 - Módulo "SLAVE" do secundário do protótipo 1 instalado.

A Figura 89 mostra o módulo “SLAVE” do Protótipo 1 instalado na máquina e em funcionamento.

Figura 90 - Módulo "SLAVE" do secundário do protótipo 2 instalado.

A Figura 90 mostra o módulo “SLAVE” do protótipo 2 em funcionamento e instalado na máquina.

Figura 91 - Módulo "MASTER" do protótipo 1 instalado.

AFigura 91 mostra o módulo “MASTER” em funcionamento e instalado.

Figura 92 - Módulo "MASTER" do protótipo 2 instalado.

A Figura 92 mostra o módulo “MASTER” do protótipo 2 em funcionamento e instalado.

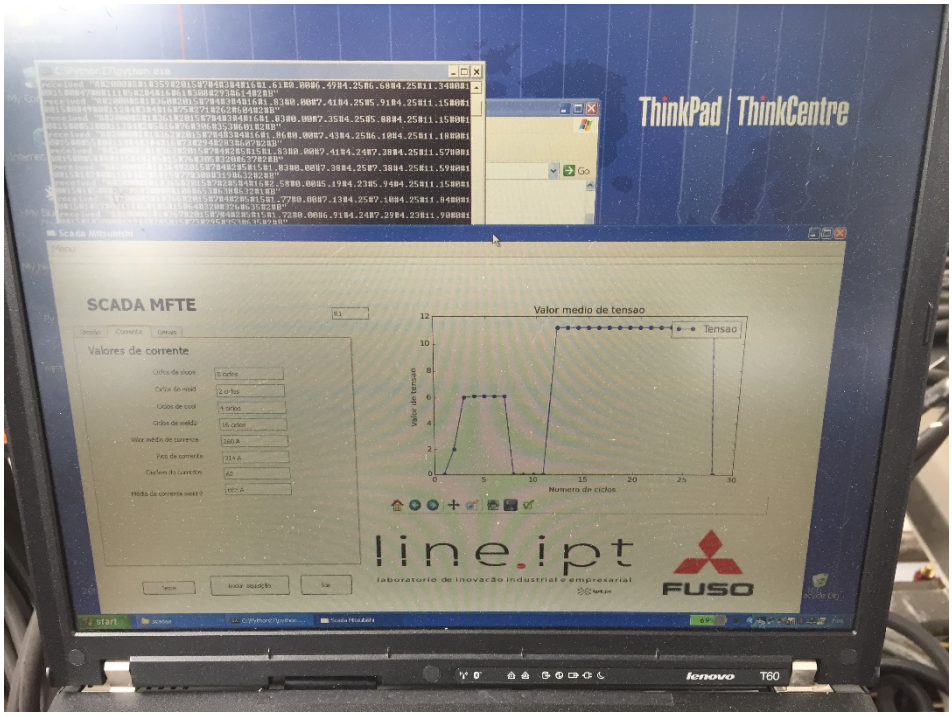


Figura 93 - Aplicação "SCADA" em funcionamento.

A Figura 93 mostra a aplicação “SCADA” em funcionamento no PC da “MFTE”.

Quando tudo estiver testado e aprovado vão-se desenvolver as placas finais (Figura 94), nomeadamente o módulo do microcontrolador de modo a não utilizar plataformas de desenvolvimento da “Arduino” que não estão otimizadas para a situação.

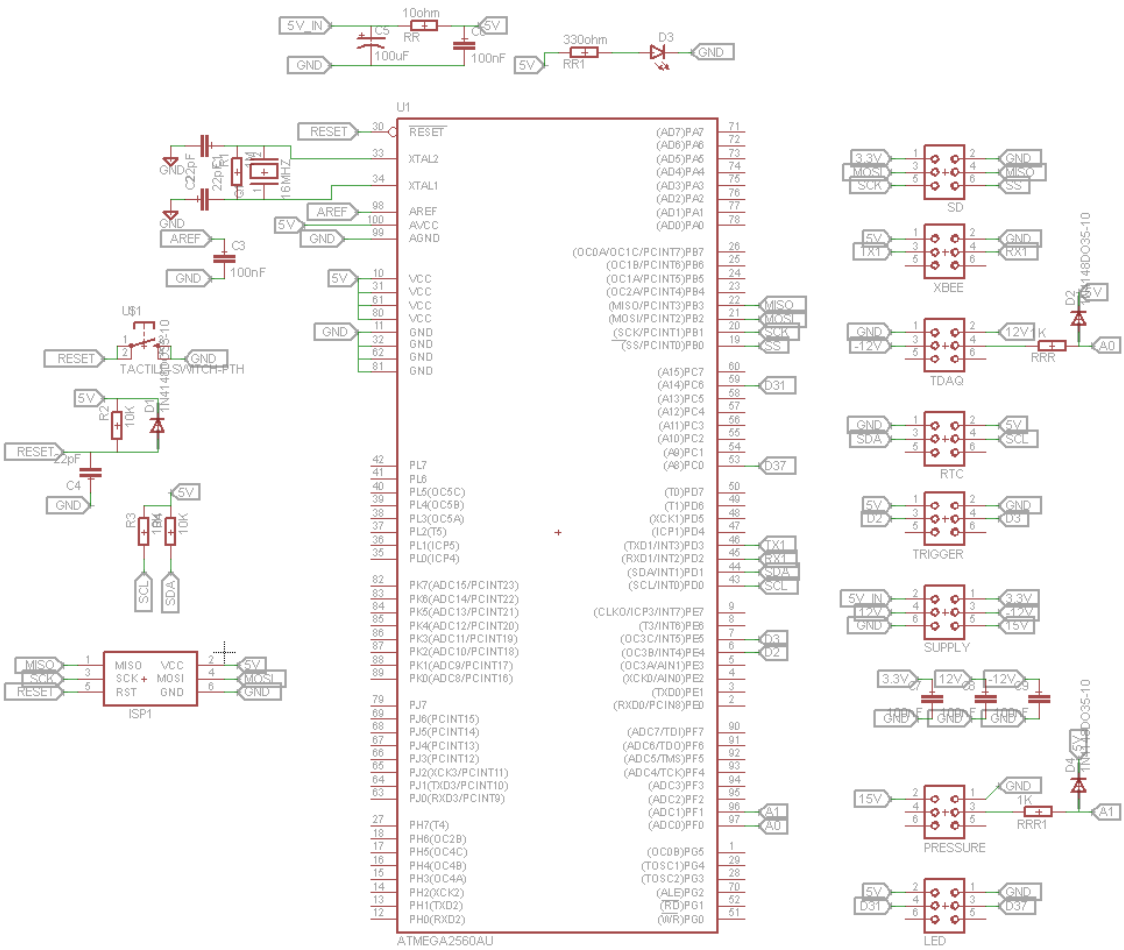


Figura 94 - Esquemático do módulo microcontrolador final.

A nível de alimentação irá utilizar-se um transformador externo de 15V/1A da marca "Nimo" de modo a que não estejam fontes dentro das caixas evitando assim a elevação de temperatura excessiva que poderá ocorrer no interior (Figura 95).

Figura 95 - Conversor a ser utilizado no protótipo final.

Dentro da caixa irá desenvolver-se um circuito com recurso a conversores DC-DC da marca "Traco Power" (Figura 96) que são bastante mais eficientes do que os tradicionais reguladores de tensão. Pretende-se colocar três conversores do tipo "Buck" para os valores de tensão necessários (±12V, 5V e 3.3V). Deste modo a eficiência geral de todo o sistema será superior o que irá aumentar o tempo de vida do sistema devido à redução da temperatura do sistema. Estas modificações finais têm como base não só o correto funcionamento do sistema mas também o compromisso em entregar à empresa um sistema robusto, eficiente e otimizado ao máximo para cumprir da melhor maneira as suas funções.

Figura 96 - Conversor DC-DC com isolamento galvânico da "Traco Power".

Para a construção final das caixas dos protótipos irá continuar-se com a temática da modularização, ou seja não desenvolver uma placa final com todas as funcionalidades mas

sim vários módulos ligados entre si, o que irá facilitar a manutenção em caso de avaria (só será necessário substituir o módulo com defeito em vez de todo o sistema) bem como a integração de novas funcionalidades no futuro.

Até este ponto pode-se fazer um balanço positivo relativamente ao desenvolvimento do projeto. A interação com as máquinas de soldadura é "total" não se observando qualquer tipo de interferência com o funcionamento normal da mesma. As funcionalidades pretendidas encontram-se implementadas, faltando apenas calibrações no que toca à análise da qualidade da soldadura para assim se poder comparar os resultados do protótipo com testes mecânicos destrutivos até se obter o melhor resultado final.

3 Henriques & Henriques

Um segundo projecto desenvolvido durante o estágio foi o projeto "HIPERBAR - Sistema de Controlo para Câmara Hiperbárica". Este tem como objetivo o desenvolvimento de um Caderno de Encargos para a instalação dos sistemas de controlo para uma câmara Hiperbárica tendo em vista a sua homologação e certificação.

Com este projeto pretende-se que a Henriques & Henriques obtenha "know how" relativo ao controlo deste tipo de sistemas, de forma a que possa ser aplicado em qualquer câmara Hiperbárica, para qualquer tipo de utilização: sistemas de segurança dos trabalhadores (minas, por exemplo), médica/terapêutica, estética/beleza, lazer.

O projeto enquadra:

- Análise da legislação relativa à homologação de câmaras hiperbáricas (entidades envolvidas, ensaios, testes exigidos, etc.);

- Análise do estado da arte, nomeadamente no que diz respeito ao processo que se pretende controlar, às variáveis críticas, aos sistemas de segurança, etc.;

- Análise da concorrência (ao nível das aplicações medicinais, estéticas, etc.);

- Desenvolvimento do caderno de encargos relativo ao sistema elétrico e eletrónico para o controlo integrado;

- Desenvolvimento do caderno de encargos relativo ao Software;

- Desenvolvimento do caderno de encargos relativo à homologação do produto nas várias vertentes funcionais.

O desenvolvimento do projeto foi estruturado pelas seguintes fases:

- 1ª Fase: Elaboração de relatório relativo ao estado da arte e análise da concorrência;
- 2ª Fase: Elaboração de relatório relativo à homologação do produto;
- 3ª Fase: Análise e desenvolvimento de protótipos para sustentarem o caderno de encargos;
- 4ª Fase: Elaboração do caderno de encargos final.

O projeto terá a duração de 1 ano sendo as quatro fases divididas aproximadamente com o mesmo intervalo de tempo ao longo desse ano, sendo a Fase 2 a que apresenta a maior duração como mostra a Tabela 2.

Tabela 2 - Calendarização das atividades do Projeto

Mês	1	2	3	4	5	6	7	8	9	10	11	12
1ª Fase:												
2ª Fase:												
3ª Fase:												
4ª Fase:												

3.1 Fase 1

O objetivo da primeira fase é garantir que a Henriques & Henriques obtenha o "know-how" necessário para a construção dos sistemas de eletrónica e controlo que envolvem o funcionamento de uma câmara hiperbárica, de forma que o caderno de encargos possa ser utilizado para qualquer tipo de aplicação.

3.2 Aplicações de camaras Hiperbáricas

As camaras hiperbáricas têm um vasto leque de possíveis aplicações, fazendo-se assim portanto referência às mais comuns e as que podem ser resultado do desenvolvimento deste projeto:

- Medicina: As câmaras hiperbáricas desenvolvidas para medicina são utilizadas em diversos tratamentos, nomeadamente no âmbito da medicina de urgência, reanimação em pacientes politraumatizados, nas intoxicações, no tratamento de infeções, síndromes neurológicas, ortopedia e cirurgia geral. Estas podem ser rígidas (monolugar, duplolugar ou multilugar), portáteis (monolugar) ou flexíveis (apenas se referiu a sua existência, uma vez não ser o mercado pretendido pela empresa);
- Mergulho: As câmaras hiperbáricas utilizadas em "mergulho" têm aplicação essencialmente para transporte de mergulhadores que sofreram um acidente durante a prática de mergulho (existindo a possibilidade de haver acompanhamento de um

médico durante o transporte) e por mergulhadores cujos trabalhos exigem longos períodos submersos, existindo uma relação entre a profundidade de mergulho e o tempo de permanência, que pode condicionar a saúde do mergulhador, sendo necessário que, ao ultrapassar esses limites o mergulhador retorne à superfície seguindo os procedimentos da "Tabela de Descompressão"(Ref[10]) de modo a reequilibrar o seu corpo à pressão normal e assim evitar uma embolia gasosa. Estas podem ser utilizadas para descompressão, recompressão, saturação ou transporte de pacientes.

- Refúgio: Em minas assim como em túneis profundos, ou em profundidades onde a pressão seja elevada, este tipo de câmaras hiperbárica tem aplicação durante a deslocação de modo a "ambientar" a pressão, assim como em tratamento de urgência aos trabalhadores e servir também como refúgio em caso de emergência, devido a alterações de pressão, qualidade do oxigénio ou outra circunstância, como desabamento, onde os trabalhadores podem permanecer em segurança, durante um certo período de tempo. Estas são apenas do tipo rígido podendo ser utilizadas em projetos de tuneis, exploração de minérios ou processos químicos.

3.3 Características dos vários tipos de câmaras Hiperbáricas

3.3.1 Características gerais das camaras medicinais

Devido ao elevado número de tabelas para os diversos tipos de camaras colocou-se apenas uma (Tabela 3) a título de exemplo.

Tabela 3 - Especificações gerais

Especificações	Perry	Sechrist/Oxy Heal	Revitalair	Enviromental	Baroxhbo	Hipertech	Haux-Life
Pressões máximas de operação	2,07 Bar	2,07 Bar	---------- ------	3,04 Bar	2 - 4 Bar	3 Bar	2,06 Bar

Pressões máximas de teste	------	--------	----------	----------	------	4 Bar	3,10 Bar
Temperaturas de Operação	Entre 0º e 38ºC	Entre 10º e 38ºC	----------		------	----------	----------
Humidade de operação	----------	Entre 30 e 90% (a 25ºC)	----------	Entre -18º e 30ºC	------	--------	---------
Pressões de fornecimento de O_2	3,45 – 6,2 Bar	3,45 - 4,83 Bar	----------	----------	------	----------	----------
Fluxos de fornecimento de O_2	850 – 1130 lpm	----------	140lpm	----------	------	---------	----------
Pressões de fornecimento de gás hospitalar	----------	----------	----------	Entre 4,46 e 7,91 Bar	------	----------	----------
Taxas de ventilação	125 – 385 lpm	80 – 400 lpm	----------	----------	------	----------	--------
Taxas de mudança de pressão	0,07 – 0,34 bar por minuto	0,07 – 0,34 Bar por minuto	----------	Entre 1,05 e 1,35 Bar/min	------	----------	---------

Taxas de depressão de emergência	De 2,07 para 0 Bar em 100-120 segundos	De 2,07 para 0 Bar em 60-119 segundos	----------	De 3,04 para 1 Bar em 120 segundos	------	----------	---------
Alimentação elétrica	-------	---------	220V, 50-60Hz, UPS disponível	----------	----------	----------	De acordo com os requisitos do cliente, UPS disponível
Duração de compressões	-------	----------	----------	Total em menos de 3 minutos	----------	----------	----------

3.4 Requisitos para licenciamento de camaras Hiperbáricas

Para o correto funcionamento de uma câmara hiperbárica é necessário garantir o controlo de três grandezas fundamentais:

- Pressão;

- Temperatura;

- Oxigénio.

Para além do controlo tem de se garantir que a gama de valores destas grandezas se encontra em conformidade com as normas e legislação em vigor, para que se cumpram todos os requisitos de segurança e deste modo ser possível a homologação do produto.

3.5 Organismos de inspeção

Para garantir que a camara hiperbárica cumpre todos os requisitos é necessário recorrer a organismos que estejam acreditados para poderem executar inspeções em equipamentos sob pressão. De acordo com o "IPAC" (Diretório de Entidades Acreditadas) os organismos de inspeção sectorial (ISO/IEC 17020) para equipamentos sob pressão (ESP) são os seguintes:

- I0001 - Instituto Tecnológico do Gás;
- I0002 - Instituto de soldadura e qualidade / Equipamentos Sob Pressão e Equipamentos de transporte de mercadorias perigosas;
- I0006 - Bureau Veritas Rinave – Sociedade Unipessoal Lda;
- I0009 - SGS Portugal – Sociedade Geral de Superintendência, SA / Serviços industriais;
- I0016 - GASMED - Inspecção e Análise de Projectos de Gás, Lda;
- I0030 - TELECERT - Certificações Técnicas, Lda;
- I0032 - Hotgas - Inspecções e Calibrações, Lda;
- I0033 - REDINSPAL - Consultoria e Inspeções Técnicas, Lda;
- I0038 - SPFT - Sociedade Portuguesa Fiscalização Técnica, Lda;
- I0074 - EQS - Serviços de Engenharia, Qualidade e Segurança, Lda;
- I0075 - QTEC – Qualynspect, Lda;
- I0086 - TUV Rheinland Portugal, Inspecções Técnicas, Lda / Industrie.

3.6 Normas e legislação para Portugal

Para licenciamento de camaras hiperbáricas em território nacional é necessário seguir os seguintes decretos de lei:

3.6.1 Decreto de lei 211/99 – Equipamentos sob pressão e métodos de controlo desses recipientes.

O decreto de lei 211/99 aplica-se a equipamentos sob pressão com uma pressão máxima admissível (PS) superior a 0,5 bar.

3.6.2 Decreto de lei 90/2010 – Licenciamento de equipamentos sob pressão

O licenciamento e homologação de qualquer equipamento sob pressão (camaras hiperbáricas) terá de seguir o decreto de lei 90/2010. Este decreto de lei engloba o funcionamento base de uma camara hiperbárica. A especificação numa determinada área, nomeadamente a medicina terá de englobar para além deste decreto de lei outras normas.

3.7 Normas e legislação internacional

Para ser permitido o licenciamento de uma camara a nível internacional terá de se seguir as normas adicionais apresentadas à frente.

3.7.1 EN13445 – Unfired pressure vessels

A norma "EN13445 – Unfired pressure vessels" disponibiliza regras para desenho, fabrico e inspeção de camaras pressurizadas, sendo composta por sete partes:

- EN 13445-1: Geral;
- EN 13445-2: Materiais;
- EN 13445-3: Desenho;
- EN 13445-4: Fabricação;
- EN 13445-5: Inspeção e teste;
- EN 13445-6: Requisitos para desenho e fabrico de camaras pressurizadas e componentes pressurizados construídos a partir de ferro fundido e grafite esferoidal;
- EN 13445-8: Requisitos adicionais para camaras pressurizadas de alumínio e ligas de alumínio.

3.7.2 EN14931 – Pressure vessels for human occupancy

A norma "EN14931 – Pressure vessels for human occupancy" disponibiliza as regras para camaras pressurizadas de ocupação humana multilugar que podem ser utilizadas em terapia hiperbárica. Estas regras englobam o desempenho, requisitos de segurança e testes.

3.7.3 ISO 13485 – Dispositivos médicos, sistemas de gestão de qualidade

A norma "ISO 13485 – Dispositivos médicos, sistemas de gestão de qualidade" representa requisitos na área de sistemas de gestão de qualidade para desenho e fabrico de dispositivos médicos. Se a camara hiperbárica for desenvolvida para aplicação em áreas médicas terá de seguir esta norma.

3.8 Diretivas Europeias

3.8.1 PED (97/23/EC) – Pressure equipement directive

A diretiva "PED (97/23/EC)" estabelece os padrões para o desenvolvimento e fabrico de equipamentos sob pressão assim como os requisitos a serem cumpridos para uma inspeção que verifique a conformidade.

3.8.2 Diretiva 93/42/EEC – Dispositivos Médicos

A diretiva "93/42/EEC" europeia é aplicável aos dispositivos médicos e respetivos acessórios, estando descritos todos requisitos necessários à sua conceção, construção, materiais e colocação no mercado, sendo portanto necessária no caso de as camaras hiperbáricas serem utilizadas para fins medicinais.

3.9 Tabela resumo de legislação necessária

A Tabela 4 resume as normas e legislação necessária para a certificação de uma camara hiperbárica.

Tabela 4 - Tabela de resumo de normas e legislação.

Legislação Portuguesa	Decreto de lei 211/99 – Equipamentos sob pressão e métodos de controlo desses recipientes.
	Decreto de lei 90/2010 – Licenciamento de equipamentos sob pressão.
Normas e legislação internacional	EN13445 – Unfired pressure vessels.
	EN14931 – Pressure vessels for human occupancy.
	ISO 13485 – Dispositivos médicos, sistemas de gestão de qualidade.
Diretivas europeias	PED – Pressure equipement directive.
	Diretiva 93/42/EEC – Dispositivos Médicos.

3.10 Estado de arte e tecnologia

Relativamente à parte da escolha/desenvolvimento de tecnologia de controlo analisaram-se os diferentes fabricantes de camaras de modo a perceber-se que tipos de tecnologias estes utilizam.

Uma camara hiperbárica é um sistema complexo ao nível de controlo e monitorização devido ao elevado número de componentes necessários para o seu funcionamento (sensores, atuadores, etc). Para o controlo de cada parâmetro é necessária uma solução integrada de automação. A título de exemplo, para o caso do controlo do nível de oxigénio são necessários sensores para monitorizar o seu nível, atuadores para aumentar ou diminuir o nível de oxigénio (compressores ou bombas de ar) e um componente de processamento (autómato ou microcontrolador) que interligue os outros componentes de modo a formar uma solução integrada de controlo.

Averiguaram-se também algumas tecnologias patenteadas existentes que possam ser utilizadas numa camara hiperbárica.

3.11 Patentes

Para este tipo de tecnologia verificou-se que existe um elevado número de patentes assim como de tecnologias.

Resumiu-se na Tabela 5 um grupo de tecnologias que eventualmente poderiam ser utilizadas mas que se encontram patenteadas.

Tabela 5 - Tecnologias patenteadas

Tecnologias Patenteadas	
Designação	**Descrição**
US4633859 (A) (1987-01-06)	Sistema de controlo gás inerte ambiental.
US2005178387 (A1) (2005-08-18)	Sistema de terapia de oxigénio hiperbárica.
US2010059059 (A1) (2010-03-11)	Sistema de controlo de câmara hiperbárica.
WO2014101548 (2014-07-03)	Método de controlo de pressão para turbina ventiladora.
WO2013053090 (2013-04-18)	Sistema de controlo e monotorização computacional integrado.
WO2013116324 (2013-08-08)	Sistemas e métodos de controlo de sessões de tratamento com oxigénio hiperbárico.
US4448189 (A) — 1984-05-15	Válvula de fluidos e meios para ajudar a controlar o funcionamento da câmara.
US3676563 (A) — 1972-07-11	Misturas gasosas para respirar acima da pressão atmosférica normal.
US3754133 (A) — 1973-08-21	Lâmpada para uso em ambientes com pressão elevada.

US3984673 (A) — 1976-10-05	Sistema de iluminação externa para câmaras hiperbáricas.
US 20040261796 A1	Sistema de controlo e /ou monitorização para camaras hiperbáricas.

3.12 Problemas e opções sugeridas

As empresas fornecedoras no mercado das camaras hiperbáricas têm na sua posse a tecnologia, mas esta é para uso próprio e não se encontra disponível para utilização por outros.

Os sistemas de controlo para camaras hiperbáricas têm de seguir requisitos bastante rigorosos.

A sugestão fornecida será inicialmente incorporar tecnologia já existente no mercado que apresente um grau de robustez e eficiência elevado para assim se cumprir os requisitos exigidos nas normas e legislação.

Após o protótipo estar concluído recomenda-se iniciar-se o desenvolvimento de alguns sistemas com vista à otimização dos produtos existentes e à criação de tecnologia inovadora que sirva de mais-valia para garantir uma vantagem competitiva relativamente aos outros fabricantes.

3.13 Análise de informação

Apresentando como objetivo, o estado da arte e análise da concorrência das tecnologias atuais de Sistema de controlo para Câmara Hiperbárica, iniciou-se por analisar o tipo de câmaras que cada empresa vende, assim como as normas e certificados relacionados com estas. Também se efetuou uma análise comparativa entre câmaras do mesmo tipo de cada empresa. Concluiu-se que, de forma geral, estas vendem para todo o mundo e as câmaras cumprem diversas normas.

Alcançadas estas etapas, efetuou-se uma pesquisa das tecnologias patenteadas com o objetivo de encontrar tecnologias existentes que possam auxiliar o desenvolvimento do caderno de encargos.

De seguida efetuou-se uma análise às normas, diretivas e à identificação de algumas organizações de regulamentação e normalização. Chegou-se à conclusão que para o mercado Norte-americano, as câmaras tem de obedecer aos requisitos de **ASME/PVHO-1** (regula a construção de equipamentos sob pressão), **ASME/PVHO-2** (estabelece critérios e orientações técnicas para a operação e manutenção de PVHOs incluindo as suas janelas de acrílico), por forma a serem aprovadas pela **FDA**.

Para o mercado Europeu, as câmaras têm de obedecer aos requisitos da norma **ISO 13485** (desenvolvimento e construção de equipamentos médicos), que proporciona um primeiro passo para a conformidade com os requisitos regulamentares europeus como "**European Medical Devices Directive 93/42/EEC**" e "**Pressure Equipment Directive (97/23/EC)**". Para o caso particular de câmaras multilugar, é necessário também cumprir os requisitos da norma **EN14931**. O livro "Handbook on Hyperbaric Medicine" (Ref.11) é uma fonte com grande volume de informação sobre este tipo de câmaras.

Os padrões que estabelecem a segurança em equipamentos sob pressão da união europeia, são muito próximos dos padrões dos EUA, o que permite à maioria das agências de inspeção internacional fornecer os serviços de verificação e certificação para avaliar o cumprimento das diferentes diretivas sobre equipamentos sob pressão.

Efetuou-se uma análise à **prEN14931:2006,** com o objetivo de adquirir informações sobre os sistemas envolvidos e as variáveis de monotorização presentes nos sistemas hiperbáricos de câmaras multilugar para terapia.

Por fim analisou-se também os requisitos a serem cumpridos pela norma **EN 13445** (contém os requisitos para o desenvolvimento, fabrico e inspeção de equipamentos sob pressão). Alguns dos requisitos são por exemplo a existência de:

- Pelo menos 2 compartimentos;
- Sistemas de compressão, descompressão, ventilação, gás de tratamento separados;
- Sistemas de ar comprimido, iluminação interior e de emergência, janelas em acrílico, medidas anti-incêndio, meios de comunicação entre o interior e exterior da câmara;
- Sistemas de alarme visuais e sonoros, fontes de energia de emergência, sistemas de tubagem, válvulas e acessórios, videovigilância, relógio com reserva de energia, instalação elétrica exterior de acordo com **IEC 60364-7-7710;**

- Dispositivos de segurança em todas as aberturas e fechaduras.
- Sistemas redundantes em todos os sistemas.

As variáveis a controlar e monitorizar são:

- A compressão; a descompressão; a ventilação; o sistema de comunicação; e a ativação dos sistemas de anti-incêndio.
- A pressão de cada compartimento, a pressão do ar comprimido armazenado, a pressão de cada gás, a temperatura, a taxa da ventilação da câmara e antecâmara, e a temperatura interior de cada compartimento.
- A concentração de oxigénio de cada compartimento, a pressão parcial de dióxido de carbono dentro da câmara, a fonte de pressão do sistema de respiração, o estado do sistema de anti-incêndio de cada câmara, e o estado da fonte de alimentação do sistema elétrico.

3.14 Conclusões Fase 1

Verificou-se que existe um elevado número de tecnologias patenteadas, mas na impossibilidade de as testar não se consegue saber se funcionam corretamente ou não e se cumprem os níveis de robustez e normas exigidas. Portanto nesta primeira fase não se sugere o desenvolvimento de raiz de sistemas de controlo, mas sim uma incorporação de tecnologias existentes.

Após o desenvolvimento, análise e estudo do protótipo, caso se considere conveniente, recomenda-se iniciar a otimização dos sistemas existentes ou mesmo proceder ao desenvolvimento de novos sistemas proprietários que possam ter vantagem sobre os dos outros fabricantes.

Sugeriu-se a troca de ordem das fases 2 e 3, ou seja testar e instalar um protótipo funcional antes do desenvolvimento do caderno de encargos para se garantir que cumpre todos os requisitos exigidos para o licenciamento de uma camara hiperbárica e assim valida-lo.

A nível nacional a entidade principal para certificação de equipamentos médicos é a "Infarmed".

De modo a a ter contacto com uma situação real visitou-se uma camara hiperbárica em funcionamento no Hospital da Marinha. Com a orientação de médicos e técnicos especialistas foi possível ver e analisar todos os sistemas em funcionamento de modo a fazer uma relação com as normas e legislações observadas anteriormente.

Para além dos sistemas referidos anteriormente, os quais foram observados em funcionamento e numa situação real de tratamento, concluiu-se que para além da robustez necessária destes sistemas, são necessárias também medidas rigorosas de segurança ao nível do vestuário e dos objetos pessoais que se podem levar para o interior da camara. O próprio material de roupa pode desencadear um explosão no interior causando danos catastróficos tanto para o equipamento como para os ocupantes.

Durante uma sessão de tratamento é necessário que um dos ocupantes seja um enfermeiro para dar assistência aos pacientes caso seja necessário. Existe também uma antecâmera que permite igualar a pressão com a da camara principal para dar acesso ao interior sem ser necessário desligar todo o sistema.

Para a fase 2 do projeto irá ser feita uma análise detalhada das normas acima mencionadas de modo a que o caderno de encargos possa ser finalizado. A H&H escolheu as camaras de fins medicinais como a aplicação para a qual se vai desenvolver o caderno de encargos pois estas são as mais detalhadas e complexas ao nível de normas e legislação devido a estarem ligadas à área saúde.

4 PneuSines

Atualmente o desempeno das jantes de veículos automóveis é executado por operadores especializados com auxílio de máquinas-ferramentas próprias. O nível de automação destes equipamentos é ainda diminuto, sendo o conhecimento e experiência do operador decisivos para a qualidade final do serviço e duração do processo.

Uma máquina que realize autonomamente todo o processo de desempeno de jantes, com elevados parâmetros de qualidade, será um avanço tecnológico significativo no sector, não só porque diminuirá o tempo de operação, libertando o operador, como possibilitará a manutenção dos parâmetros de qualidade através da redução do erro e consequente aumento da consistência dos resultados. Para a PneuSines, a aquisição desta tecnologia representa não só um aumento da qualidade e eficiência do serviço, como também uma oportunidade de um novo mercado/negócio, pela comercialização destes novos equipamentos.

O projeto "Automatização de Máquina de Desempeno de Jantes" tem o objetivo desenvolver um sistema de automatização de todos os processos/operações da tarefa de desempeno de jantes. O sistema desenvolvido terá por base uma máquina-ferramenta já existente, sendo neste equipamento que serão aplicadas e integradas as soluções tecnológicas e será a partir da solução final que serão definidos os critérios e parâmetros para a construção de uma nova máquina automática de desempeno de jantes.

O projeto é estruturado da seguinte forma:

1ª Fase: Estudos prévios

Nesta fase pretende-se adquirir conhecimento acerca do processo de desempanagem (funcionalidades, operações, restrições, parâmetros).

2ª Fase: Desenvolvimento de soluções tecnológicas

Nesta fase pretende-se aplicar os conhecimentos adquiridos na fase anterior para o desenvolvimento das soluções de automatização das tarefas/operações, assim como integrar/incorporar as tecnologias na máquina-ferramenta de teste.

3ª Fase: Testes dos Protótipos de desenvolvimento

Nesta fase pretender-se-á efetuar testes dos protótipos desenvolvidos, aferição dos sistemas, análise e correção dos desvios/erros.

4ª Fase: Integração dos sistemas e desenho da solução final

Nesta última fase é pretendida a integração dos automatismos e dos sistemas de controlo desenvolvidos. Desenvolvimento do "software", interface de controlo da máquina e testes finais.

No final do projeto, ter-se-á o desenho do sistema global do sistema de automatização de uma máquina de desempanagem de jantes, tendo em vista a construção de um protótipo para comercialização, numa fase seguinte de investimento.

O projeto terá a duração de 12 meses, sendo apresentada na Tabela 6 a calendarização dos trabalhos.

Tabela 6 - Calendarização das atividades do Projeto.

Fase	Meses											
	1	2	3	4	5	6	7	8	9	10	11	12
1ª Fase: Estudos prévios												
2ª Fase: Desenvolvimento de soluções tecnológicas												
3ª Fase: Testes dos protótipos de desenvolvimento												
4ª Fase: Integração dos sistemas e desenho da solução final												

4.1 Levantamento inicial

Antes de iniciar qualquer tipo de desenvolvimento houve um esforço no sentido de tentar perceber o método de desempenagem de uma jante. Na Pneusines o processo é manual e feito por um operador que controla um motor que faz rodar a jante para auxiliar o processo de desempenagem.

Inicialmente a jante é colocada no seu suporte (Figura 97):

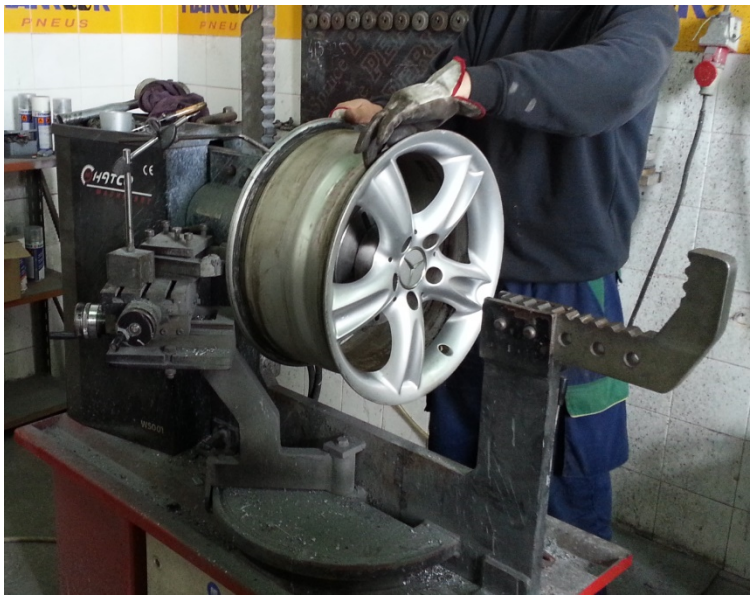


Figura 97 - Jante no suporte.

A jante é presa num eixo rotativo, manualmente o operador vai rodando a jante pra identificar as zonas com empeno. Em termos de localização os empenos nas jantes ocorrem em 90% dos casos nas abas laterais e no interior, sendo que os restantes 10% dizem respeito à quebra da jante caso em que é necessário solda-la para a reparar. O protótipo irá ser desenvolvido para tratar o grupo dos 90% que são as situações mais frequentes.

A fase seguinte consiste na colocação de um medidor que fica encostado à jante, o operador vai rodando e quando o medidor "raspar" na jante é sinal que aquela zona está empenada (Figura 98).

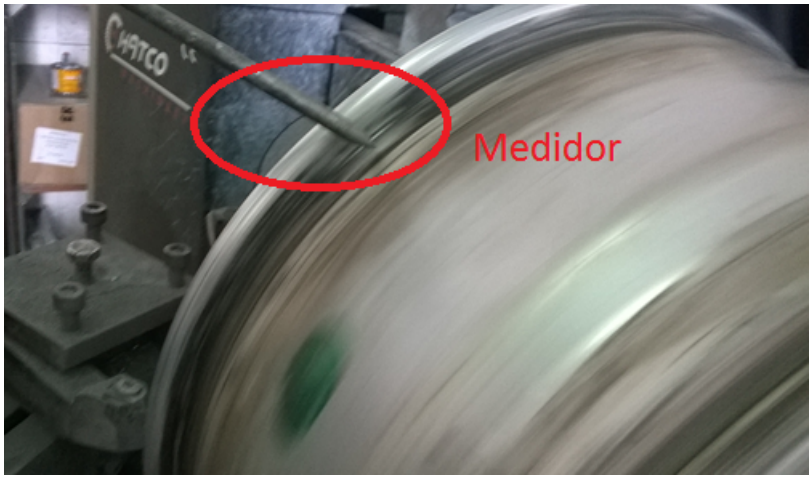


Figura 98 - Medidor posicionado para identificar empenagens.

De seguida é colocado um cilindro pneumático no interior da jante (Figura 99) para fazer força no sentido oposto de modo a retificar aquela zona. Para compensar a deformação o operador com um martelo executa duas pancadas em ambas as extremidades da zona que foi deformada.

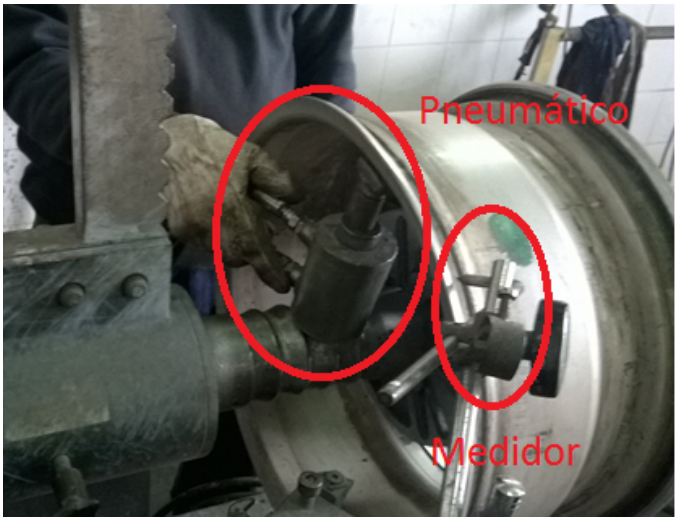


Figura 99 - Pneumático no interior da jante e medidor.

Este processo é repetido as vezes que forem necessárias até a jante estar totalmente uniforme. É um processo lento, sujeito a falhas humanas visto que o medidor é posicionado manualmente. Se o posicionamento não estiver correto a desempenagem pode não ficar correta.

A empresa decidiu que o trabalho em causa deveria ter como ponto de partida uma máquina disponível, a qual deveria servir de base para o posterior desenvolvimento da nova máquina, tal como o representado na Figura 100.

Figura 100 - Máquina pretendida para ponto de partida.

Esta máquina já contém um motor interno "AC" para fazer rodar a jante e um suporte para esta. Foi também pedido o desenvolvimento de um suporte universal para a maior parte do tipo de jantes.

Em termos de desenvolvimento/adaptação da máquina existente, de modo a que o processo seja automático irá ser necessário um motor para rodar a jante, um encoder para determinar a posição precisa da jante, um sensor para se identificar as zonas empenadas e algo que faça a força para endireitar a jante (atuador). Para controlar todo o processo irá introduzir-se um autómato com capacidade para ler os valores dos sensores, atuar os motores e atuadores e com possibilidade de comunicação "wireless" para comunicar com uma aplicação para monitorização.

Os atuadores terão de ser posicionados para executarem dois movimentos, um na vertical para o interior, e outro na horizontal para as abas.

4.2 Solução mecânica

Após concluída a aprendizagem do processo de desempenagem (funcionalidades, operações, restrições, parâmetros, etc.) iniciou-se o estudo para o desenvolvimento da solução mecânica de automatização do processo a aplicar e para isso simularam-se várias opções para chegar a uma mais viável.

O protótipo irá ser desenvolvido a partir de uma máquina convencional, fornecida pela empresa, e em acordo com o seguinte: a estrutura superior da máquina fornecida pela

empresa irá ser alterada de forma a se colocar um novo conjunto de rotação de jante, constituído por um veio de aço CK45 suportado por duas chumaceiras (componentes identificados na Figura 105). A sua rotação irá ser feita a partir de um conjunto motor-redutor ligado a um autómato, que por leitura de um sensor de medida de raio de jante irá colocar as zonas empenadas da jante para a zona superior (90º) em relação à horizontal.

Depois desta operação um sistema de cilindro hidráulico, suportado por dois guiamentos lineares e um sistema de fuso, movimenta o cilindro para a zona a intervencionar, exercendo pressão até corrigir a zona com empeno. Na série de figuras da Figura 101 à Figura 104 é possível visualizar a máquina de diferentes ângulos.

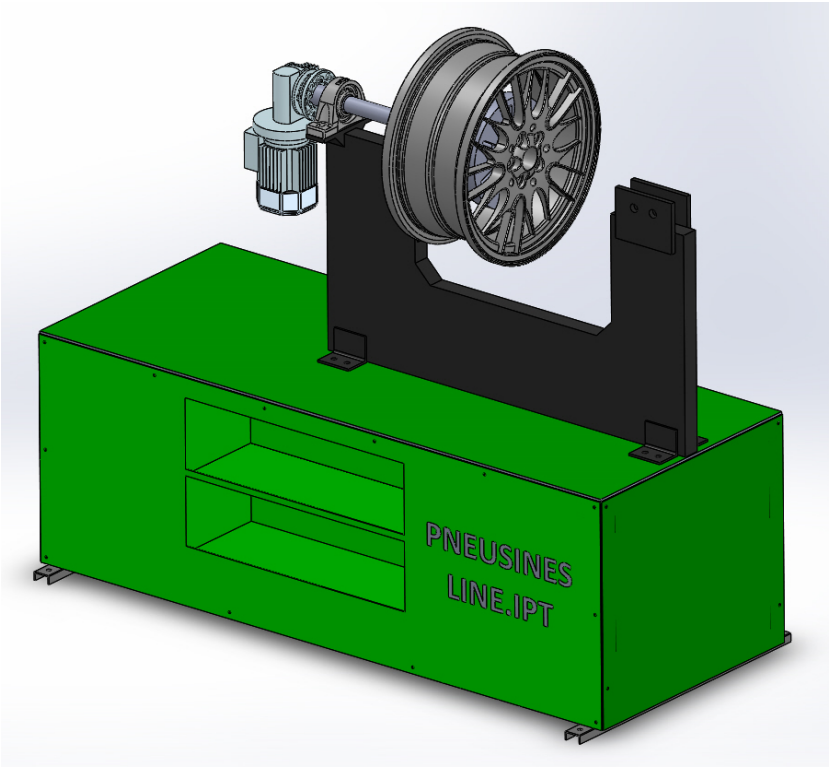


Figura 101 - Protótipo da máquina.

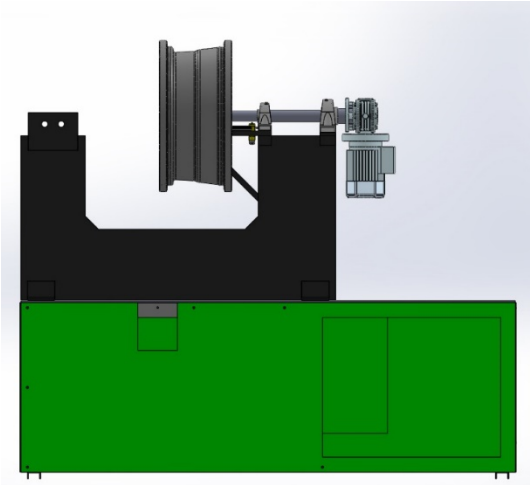


Figura 102 - Vista lateral.

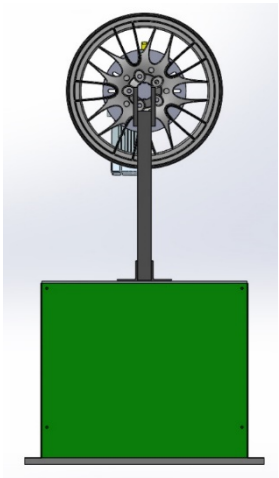


Figura 103 - Vista frontal.

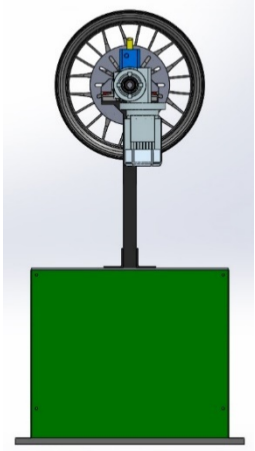


Figura 104 - Vista traseira.

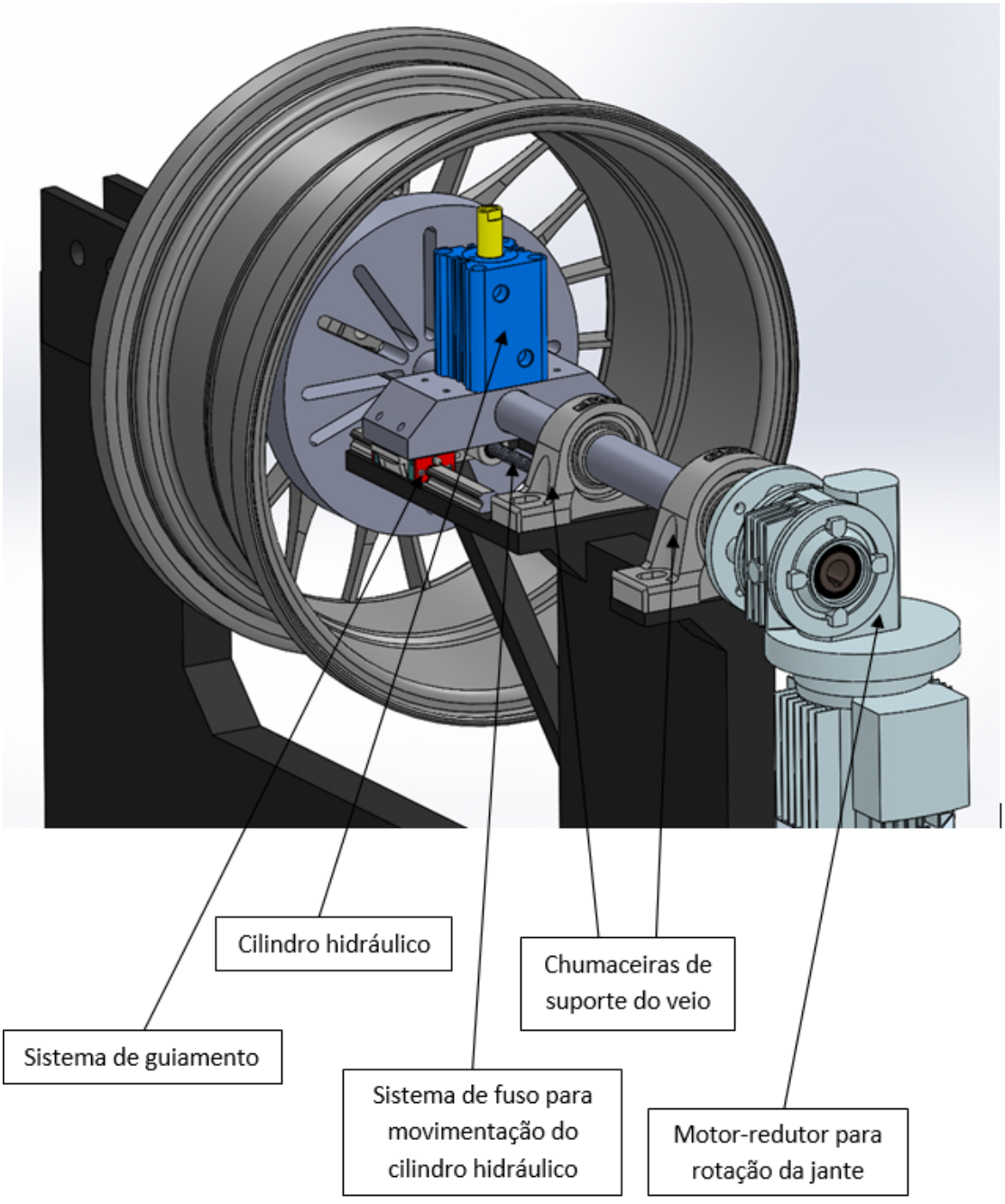


Figura 105 - Interior do protótipo.

Na Figura 105 é possível visualizar os diferentes constituintes do protótipo e a respetiva identificação.

4.3 Automação

Com base na ideia mecânica desenvolvida planeou-se a parte de automação da máquina.

Para a identificação dos empenos decidiu-se a utilização de um sensor com base em medição laser de grande precisão. Para rodar a jante e movimentar o pneumático optou-se por utilizar servomotores.

Para a automatização do protótipo irá adotar-se uma solução Siemens que fornece todos os componentes necessários ao desenvolvimento do projeto, nomeadamente uma arquitetura adequado ao controlo de servomotores. A solução escolhida foi o "Starter Kit S7-1200" da "Siemens" composto pelos componentes base para o desenvolvimento do projeto (autómato, software de programação e consola gráfica para monitorização do processo).

Como módulo principal optou-se por um "StarterKit" com o autómato S7-1200 (que contém o autómato, software de programação e consola gráfica (Ref: 6AV6651-7DA01-3AA4) sendo a melhor opção pois este inclui todos os componentes necessários ao inicio do desenvolvimento, nomeadamente o cabo de programação, o software e consola gráfica (Figura 106).

Figura 106 - Starterkit Siemens escolhido.

Para alimentar o autómato escolheu-se o módulo de alimentação recomendado para esta gama (tensão de alimentação de 24V) com a Ref: 6EP1332-1SH71.

Para o controlo da posição e atuação dos servomotores é necessário um módulo de expansão com saídas digitais (DI/O), que serve de interface com o autómato. Para uma maior versatilidade optou-se por uma solução de saídas em relé, o que, ao garantir o isolamento galvânico torna o sistema mais robusto. Este módulo contém também entradas digitais, o

que permite adicionar sensores digitais ou botões de interface, caso seja necessário (Ref: 6ES7223-1PL32-0XB0) (Figura 107).

Figura 107 - Módulo DI/O.

Como irá ser utilizado um sensor de medição de raio da jante, a solução de automação necessitará de processar sinais analógicos, para isso escolheu-se um módulo de expansão com tecnologia de aquisição de dados analógicos. Este módulo possui várias entradas que permitem acrescentar futuramente outros sensores, caso tal seja necessário (Ref: 6ES7231-4HD32-0XB0) (Figura 108).

Figura 108 - Módulo de entradas analógicas.

A consola gráfica permite monitorizar em tempo real todo o processo, bem como fazer a interação com o operador da máquina, como por exemplo o início da desempanagem, paragem ou finalização do processo.

Será também introduzido no protótipo um conjunto de componentes secundários que tornam todo o sistema mais eficiente, nomeadamente botões manuais para testes e manutenções, sensores indutivos para determinar se a jante se encontra na posição correta, indicadores luminosos e sonoros de processo, entre outros.

Assim que a parte mecânica do projeto estiver concluída será com estes equipamentos que se irá desenvolver toda a parte de automação da máquina até chegar a um produto final que seja satisfatório.

De referir que não se entrou em detalhes aprofundados da parte mecânica pois essa parte será desenvolvida pelo membro da equipa com as funções da Engenharia Mecânica.

5 Niepoort

O projeto da empresa Niepoort "Sistema de Monitorização Fitossanitária" para Vitivinicultura Biológica, tem como principal objetivo o desenvolvimento de um sistema de monitorização fitossanitária aplicado à vitivinicultura biológica, baseado em sensores e estações agrometeorológicas. O sistema efetua a recolha de diferentes dados que, integrados por um "software" especificamente desenvolvido, permite a gestão preventiva e sustentada da vinha. Este sistema foi desenvolvido de forma a ser possível monitorizar tempo real e à distância as condições agrometeorológicas da vinha. Esta estação de monitorização é constituída por um Datalogger da marca "CAMPBELL" que é bastante conceituada nesta área oferecendo a robustez necessária.

De referir que neste projeto se incidiu unicamente sobre a parte do sistema de alimentação visto que foi esse o serviço que foi requisitado ao LINE.

5.1 Grandezas medidas

A estação meteorológica tem como objetivo monitorizar as seguintes grandezas:

- Temperatura do ar;
- Humidade relativa;
- Velocidade do vento;
- Radiação solar;
- Precipitação;
- Folha molhada;
- Direção do vento.

A monitorização destas grandezas permite otimizar a interação com as vinhas para assim melhorar a produção e otimizar os recursos existentes.

5.2 Constituição das estações instaladas

As estações meteorológicas instaladas têm como componente principal o "datalogger" "Campbell" que tem incorporado um conversor para gerir a alimentação do "datalogger" e o carregamento da bateria. Para além do datalogger, esta estação tem um módulo de

comunicação "GSM" 3G para comunicar com o servidor que armazena a informação. Para a alimentação da estação está incluída uma bateria de chumbo que armazena energia para suportar os períodos de fraca luminosidade. Para o fornecimento de energia está incorporado um painel solar. Para analisar todas as variáveis acima mencionadas estão incorporados os diversos sensores. A Tabela 7 apresenta resumidamente os constituintes:

Tabela 7 - Componentes da estação de monitorização

Componentes		
Item	**Descrição**	**Fornecedor**
1.	Datalogger CR200 com painel de ligações	CAMPBELL
2.	Painel Solar, 10 Watt.	Phaesun
3.	Fonte de alimentação por bateria recarregável.	Phaesun
4.	Caixa branca, estanque.	CAMPBELL
5.	Suporte	CAMPBELL
6.	Sensor de temperatura do ar e humidade relativa.	CAMPBELL
7.	Escudo de radiação para sensor CS215.	CAMPBELL
8.	Sensor de velocidade do vento.	EMLTD
9.	Sensor de radiação (piranómetro).	CAMPBELL
10.	Base de nivelamento para sensor CS300.	CAMPBELL
11.	Braço de montagem para sensores	BRPi
12.	Sensor de precipitação (udómetro ou pluviómetro).	Pronamic
13.	Sensor de humectação	CAMPBELL

14.	Modem GSM	Wavecom

5.3 Sistema de alimentação das estações

Pretendeu-se dimensionar um sistema de alimentação otimizado para a estação de monitorização, de modo a que esta funcione eficientemente e de modo autónomo.

Para fornecer energia ao sistema instalou-se um painel solar fazendo uso de energias renováveis e aproveitando a abundância de energia solar da região.

Para armazenar a energia obtida instalou-se uma bateria de chumbo para alimentar o sistema quando o painel não está a fornecer energia.

Para fornecer uma tensão estável (a tensão do painel solar varia com base intensidade solar) instalou-se um conversor para gerir tanto a tensão de carga da bateria como a tensão de alimentação do "datalogger".

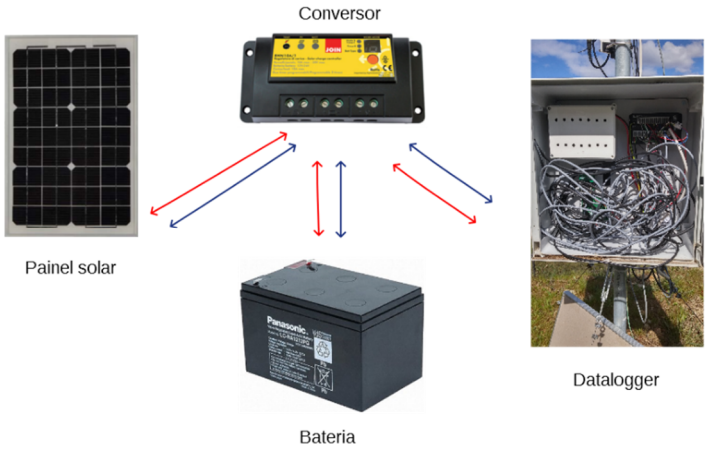


Figura 109 - Esquema geral de funcionamento.

A Figura 109 mostra o esquema geral de funcionamento do sistema. O painel recebe a radiação solar e fornece uma tensão variável para o conversor. O conversor regula a tensão de carga da bateria bem como a tensão de alimentação da estação, fazendo a gestão entre alimentação do painel solar e da bateria quando o painel não fornece energia.

O consumo energético da estação é decomposto em dois componentes principais, o "datalogger" e o "modem" "3G". O datalogger tem três estados de funcionamento:

- Estado de repouso - Não adquire dados nem comunica, consumindo neste estado aproximadamente 4mA.
- Estado de aquisição - Neste estado o "datalogger" adquire dados e guarda na memória. Consome aproximadamente 9mA.
- Estado de comunicação - Neste estado o "datalogger" transmite a informação adquirida e consome nesta operação aproximadamente 9mA.

O "modem" "3G" tem apenas dois estados de funcionamento:

- Estado de repouso - Neste estado o "modem" não comunica e consome cerca de 20mA.
- Estado de comunicação – Neste estado o "modem" envia a informação e consome cerca de 120mA.

Em repouso a estação meteorológica consome no total aproximadamente:

$$I_{repouso} = I_{datalogger} + I_{modem} \quad \textbf{(8)}$$

$$I_{repouso} = 4mA + 20mA \quad \textbf{(9)}$$

$$I_{repouso} = 24mA \quad \textbf{(10)}$$

Durante comunicações a estação meteorológica consome no total aproximadamente:

$$I_{com} = I_{datalogger} + I_{modem} \quad \textbf{(11)}$$

$$I_{com} = 9mA + 120mA \quad \textbf{(12)}$$

$$I_{com} = 129mA \quad \textbf{(13)}$$

Devido à inacessibilidade do equipamento (localizado no Douro) e à falta de mais informação em respeito aos tempos de funcionamento dos diferentes estados de operação

optou-se por sobre dimensionar os componentes do sistema de alimentação para garantir o correto funcionamento de todo o sistema.

Considerando que a estação se encontra vinte e três horas em repouso e uma hora em atividade no total durante um dia e funcionamento, o consumo energético é o seguinte:

$$C_{energetico} = t_{repouso} * I_{repouso} * V_{painel} + t_{com} * I_{com} * V_{painel} \quad \textbf{(14)}$$

$$C_{energetico} = 23h * 0.024A * 12V + 1h * 0.129A * 12V \quad \textbf{(15)}$$

$$C_{energetico} = 8.172\ Wh \quad \textbf{(16)}$$

A nível de produção de energia com recurso ao painel solar, considerando um coeficiente de irradiância de 1.6KWH/m^2 por dia (Ref[12]) para a situação de menor irradiância (Dezembro), e um angulo de inclinação de montagem do painel de 45Cº (orientação sudoeste) calculou-se a seguinte produção média de energia diária:

$$E_{produzida} = Coeficiente * P_{painel} * 1000 \quad \textbf{(17)}$$

$$E_{produzida} = 1.6KWHm^2 * 0.02KW * 1000 \quad \textbf{(18)}$$

$$E_{produzida} = 32\ Wh \quad \textbf{(19)}$$

Deste modo a energia produzida diárimente pelo painel solar é aproximadamente quatro vezes superior à energia consumida pela estação, garantindo assim que o sobredimensionamento dos componentes garante o correto funcionamento de todo o sistema.

5.3.1 Bateria

A bateria instalada possui a referência "LC-RA1212PG" da marca "Panasonic" e apresenta uma tensão de 12V e 12Ah de capacidade. Esta bateria permite armazenar energia suficiente

para alimentar a estação durante os períodos noturnos e de baixa luminosidade garantindo assim o seu funcionamento continuo (Figura 110).

Figura 110 - Bateria.

5.3.2 Painel Solar

O painel solar instalado tema referência "MS-20P" da marca "Sumsol" tem uma potência máxima de 20W, tensão máxima de 17.5V e uma corrente máxima de 1,15A. As células são do tipo policristalina e tem dimensões de 539*366*25 mm (Figura 111).

Figura 111 - Painel solar escolhido.

5.3.3 Conversor

O conversor instalado tem a referência "RHN10A" (Figura 112) da marca "Join", o qual permite uma tensão de entrada (tensão do painel solar) de 12V a 24V, uma corrente máxima de 10A e apresenta dois canais de saída, um para a bateria e outra para alimentar o datalogger. Este conversor garante uma gestão eficiente do carregamento da bateria e da alimentação do datalogger independentemente das variações da tensão gerada pelo painel solar (Figura 112).

Figura 112 - Conversor escolhido.

5.3.4 Resultado final

O resultado final foi um sistema de alimentação capaz de fornecer energia durante grandes períodos em que o painel pode, eventualmente, não captar energia solar.

Figura 113 - Estação meteorológica.

A Figura 113 mostra a estrutura da estação de monitorização.

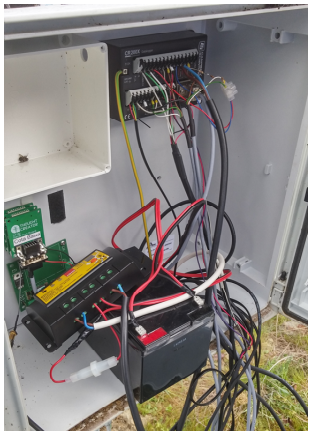


Figura 114 - Ligações do sistema.

A Figura 114 mostra as ligações internas da estação de monitorização.

6 Gestão de projetos e outras atividades

6.1 Gestão de projetos

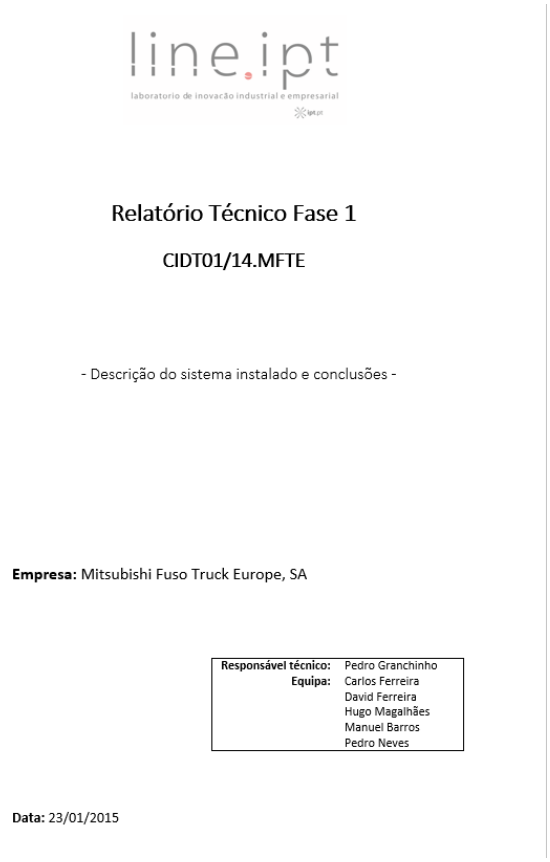
O trabalho desenvolvido no diariamente no LINE, para além de todo o desenvolvimento técnico dos projetos tem também uma componente de gestão diária de tudo o que envolve um projeto, ou seja, identificar o material necessário e desenvolver os orçamentos necessários para a requisição desse material aos respetivos fornecedores. Tem-se ainda de executar as diferentes etapas dos projetos cumprindo com as datas pré-definidas e mantendo sempre um contacto constante com a empresa a que o projeto diz respeito e gerir eficientemente os orçamentos do projeto.

A gestão tem de ser realizada em duas vertentes:

- Desenvolver o projeto com o orçamento que foi disponibilizado para esse fim. Para isso é necessário uma racionalização dos componentes e fazer um bom planeamento inicial para tentar prever situações que possam ocorrer numa situação futura.
- Desenvolver o protótipo com um preço final que seja aceitável para a empresa que o requisitou. Este ponto é importante porque se o protótipo final resultar num custo bastante elevado não se torna viável a sua implementação ou replicação caso seja necessário. Para isso o protótipo tem de estar otimizado ao máximo para ter um custo o mais reduzido possível mantendo as suas funcionalidades.

6.1.1 Relatórios técnicos

Todo o trabalho desenvolvido para um projeto tem de ser explicado e analisado para que essa informação possa ser acedida futuramente no LINE. As características desse projeto têm também de ser relatadas e justificadas à empresa a que diz respeito, para isso é necessário o desenvolvimento de relatórios técnicos, relatórios esses que podem dizer respeito ao projeto completo ou às diferentes fases de desenvolvimento de modo a dar o "feedback" às empresas acerca do progresso do projeto. Estes relatórios tem de ser objetivos e coerentes salvaguardando ao mesmo tempo os interesses do LINE ao nível de tecnologias desenvolvidas mantendo sempre o "template" de relatórios técnicos do LINE.


line.ipt
laboratorio de inovação industrial e empresarial

Relatório Técnico Fase 1

CIDT01/14.MFTE

- Descrição do sistema instalado e conclusões -

Empresa: Mitsubishi Fuso Truck Europe, SA

Responsável técnico:	Pedro Granchinho
Equipa:	Carlos Ferreira
	David Ferreira
	Hugo Magalhães
	Manuel Barros
	Pedro Neves

Data: 23/01/2015

Figura 115 - Capa de um relatório técnico.

A Figura 115 apresenta a capa de um dos relatórios técnicos entregue à "Mitsubishi".

6.1.2 Requisições de material

Durante o desenvolvimento de um projeto, quando há a necessidade de adquirir novos componentes ou equipamentos, é necessário elaborar uma requisição de material. Essa requisição de material contém a descrição dos componentes a adquirir, a referência, a quantidade e o preço. Para cada requisição de material é necessário identificar três fornecedores diferentes para os mesmos componentes. Esta situação é exigida pelo IPT de modo a procurar obter as melhores cotações possíveis.

Para os fornecedores com loja "online" não é necessário pedir uma fatura pró-forma, para todos os outros essa fatura é obrigatória. No final o fornecedor que garantir o menor preço será a quem é atribuída a compra. O material selecionado é escolhido de forma autónoma de acordo com as necessidades do projeto.

Após a finalização da requisição esta é entregue à parte da gestão do LINE num ficheiro "Excel" (Figura 116) com a lista de material do fornecedor com o preço mais baixo e os "PDF" para os três orçamentos dos fornecedores.

	Data:		9-mar-2015		
				X	CI&DT nº 01/14 (MFTE)
	Destinado a:		Projecto I&D - empresa		
	Fornecedor 3:	Farnell			
	Critério de escolha:				
			Alguns componentes não encontrados nos outros fornecedores		
	Quant.	Código Fornecedor	Designação do Material/ Serviço	Custo Unit.	Custo Total
	1	3204984	MEGA 3204984 PCB, FR4, 457X305, DS	53,53 €	**53,53 €**
	1	3204947	MEGA 3204947 PCB, FR4, 305X457, SS	51,15 €	**51,15 €**
	1	2440091	PRO POWER PP000307 MULTICORE, 2 COND, 0.22MM2, 25M, 440V	23,51 €	**23,51 €**
	1	2343916	PRO POWER-4156-TRI RATED WIRE, 22AWG, BLUE, 100M	8,84 €	**8,84 €**
	1	2343914	PRO POWER-4155-TRI RATED WIRE, 22AWG, BLACK, 100M	8,84 €	**8,84 €**
	1	2343919	PRO POWER-4159-TRI RATED WIRE, 22AWG, RED, 100M	8,84 €	**8,84 €**
	1	2343920	PRO POWER-4160-TRI RATED WIRE, 22AWG, ORANGE, 100M	8,84 €	**8,84 €**
	1	2343917	PRO POWER-4157-TRI RATED WIRE, 22AWG, BROWN, 100M	8,84 €	**8,84 €**
	1	2343924	PRO POWER-4164-TRI RATED WIRE, 22AWG, GRN/YEL, 100M	8,84 €	**8,84 €**
	4	1016345	Caixa BERNSTEIN CT-762	15,77 €	**63,08 €**
				Sub-Total:	244,31 €
				IVA (23%):	56,19 €
				Portes:	9,00 €

Figura 116 - Exemplo de requisição de material.

Para além das típicas requisições de material colaborou-se ativamente na escolha dos equipamentos e no desenvolvimento de um caderno de encargos para um concurso publico para a obtenção das novas máquinas a serem adquiridas pelo LINE para equipar o novo laboratório situado em Abrantes no "Tecnopolo". Estes equipamentos consistem numa linha de produção de "PCB's" e todo o resto do material (mobiliário, equipamentos de bancada, consumíveis, componentes, baterias de ião de lítio, motores elétricos para carro elétrico, entre muitos outros). Todo este trabalho foi desenvolvido durante várias semanas pois o caderno de encargos necessitou de todas as características detalhadas de todos os equipamentos para ser lançado.

6.2 Montagem de um protótipo desenvolvido para a Martifer

O projeto desenvolvido pela equipa anterior do LINE para Martifer consistiu no desenvolvimento de um protótipo de aquisição de dados de uma máquina de soldadura utilizada nas instalações da referida empresa. Este protótipo adquire as varáveis durante uma soldadura (tensão "DC", intensidade de corrente e velocidade do fio), converte-as para as

respetivas unidades e envia para uma aplicação “SCADA” desenvolvida no PC em “Visual Basic” para posteriormente exportar 3sses dados para uma base de dados.

Numa fase inicial do estágio foi pedido para se construir e montar um protótipo idêntico ao que foi desenvolvido para o projeto da Martifer de modo a servir de maquete de apresentação para as empresas.

A montagem deste protótipo serviu também como aprendizagem e aquisição de conhecimento de todo o processo de desenvolvimento deste protótipo, o qual se espera que vá para servir de base para projetos futuros.

Figura 117 - Exterior do protótipo da Martifer.

A Figura 117 mostra o exterior do protótipo construído para o projeto da Martifer.

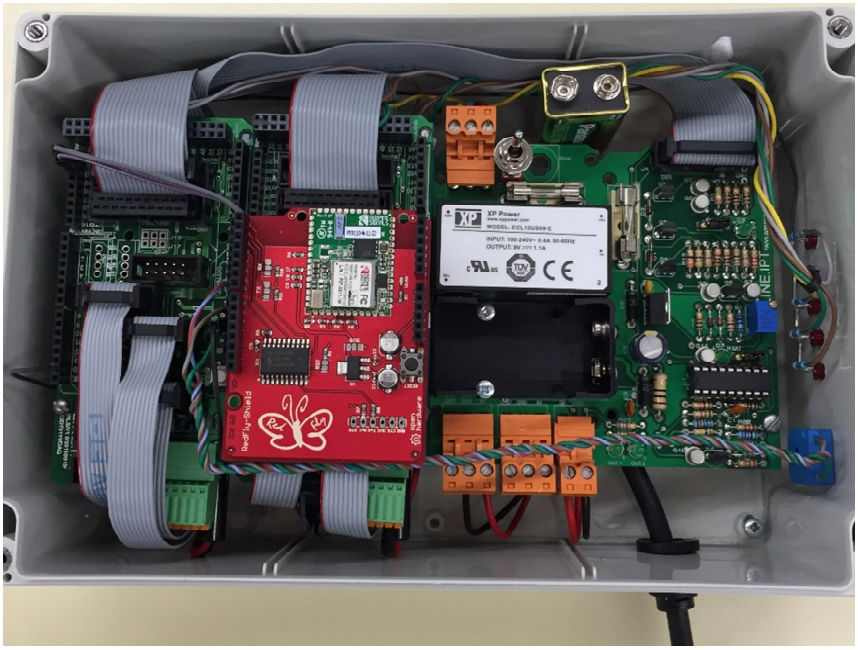


Figura 118 - Vista dos circuitos internos do protótipo.

A Figura 118 mostra o interior do protótipo construído.

6.3 Desenvolvimento de PCB's

O desenvolvimento de "PCB's" (placa de circuito impresso) é uma constante em todos os projetos que envolvam circuitos eletrónicos. Este desenvolvimento é feito em três fases:

- Fase de "software" com a aplicação "Eagle" da "Cadsoft", em que é desenvolvido o esquema das ligações eletrónicas do circuito para posteriormente se desenhar a placa em que os componentes e as ligações elétricas ("pista") são posicionados estrategicamente.

- Criação da placa física, em que inicialmente o circuito é impresso em folha "acetato" (impressão de fotolitos). Essa impressão é colocada nas faces exteriores de uma placa fotossensível e de seguida colocada com equipamento de luz "UV" para que o desenho seja "impresso" na placa (Figura 119).

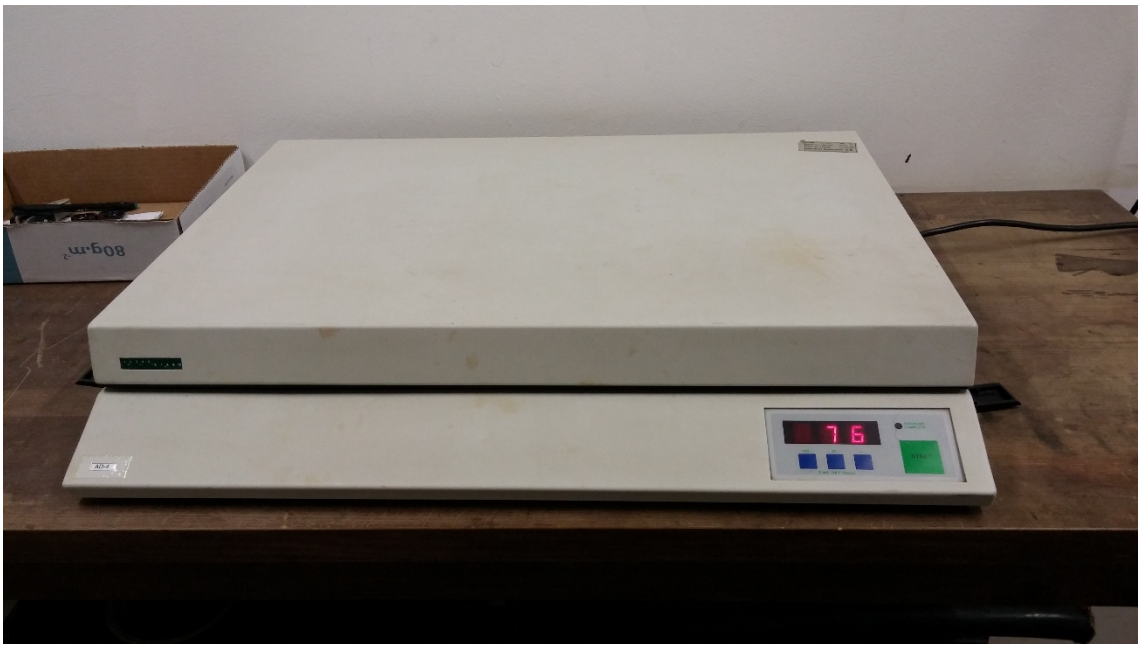


Figura 119 - Impressora "UV".

A exposição à luz ultravioleta sensibiliza o verniz existente na superfície da placa. De seguida existe um processo químico onde a "PCB" é colocada num líquido revelador (PCB developer) que retira as parcelas de verniz que foram anteriormente expostas à luz, o que permite revelar as pistas da placa. Após essa revelação a placa é colocada num tanque com um ataque químico para retirar todo o cobre que não se encontra protegido por verniz, ficando apenas as pistas que se desenharam (Figura 120).

Figura 120 - Tanque de Percloreto de ferro.

- Por ultimo é necessário furar a placa com um berbequim e utilizar uma "broca" de acordo com o diâmetros dos "Pads" desenhados, limpar a placa com álcool para retirar a película de verniz, colocar os componentes, soldar e no final voltar a proteger todo o sistema com uma camada de verniz.

6.4 Orientação de estagiários

Uma das tarefas desempenhadas durante o estágio foi a orientação de quatro estagiários do CET de Eletrónica e Biomédica da escola de Abrantes.

O estágio teve a duração de treze semanas (aproximadamente dois meses e meio), durante estas semanas os estagiários observaram o trabalho que se desenvolveu diariamente e executaram tarefas próprias dentro da área de eletrónica.

Inicialmente forneceu-se uma introdução sobre princípios básicos de eletrónica, sensores e microcontroladores com montagens práticas. Explicou-se como interagir com o "software" de programação "Arduíno", "software" de desenvolvimento de pcb's, "software" de simulação "Spice", "software" de desenvolvimento de circuitos "Eagle", "Fritzing", "Matlab" e aplicações "SCADA".

Para complementar as montagens desenvolvidas pelos estagiários mostrou-se como trabalhar com equipamento genérico de laboratório nomeadamente fontes de alimentação, geradores de sinal, osciloscópio e multímetros. Exemplificou-se o processo de criação de "PCB's" desde o desenvolvimento em "software", impressão da placa física, furação, soldadura e testes de funcionamento.

Com o intuito de funcionar como projeto final do estágio desenvolveu-se em conjunto com os estagiários um circuito que emite um sinal (onda quadrada) com uma frequência variável. Este circuito teve como objetivo a emissão de um sinal sonoro que afastasse animais específicos. Como os diferentes animais são suscetíveis a gamas de frequência bastante distintas, desenvolveu-se o circuito para uma frequência ajustável e que permitia no máximo aproximadamente 60kHz com base nos componentes disponíveis no laboratório.

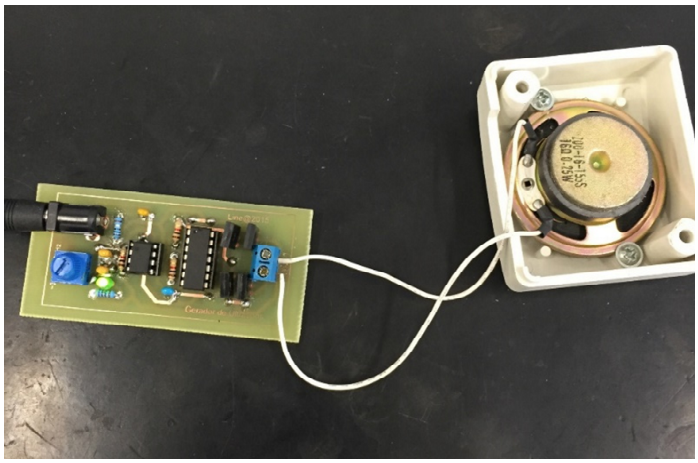


Figura 121 - Circuito desenvolvido.

A Figura 121 mostra o circuito desenvolvido a alimentar um altifalante.

6.5 Formação em impressora 3D

Com a aquisição da impressora 3D profissional "Fortus 250mc" da "Stratasys" por parte do "Line" foram dados aos funcionários 3 dias de formação com um técnico especializado da empresa que forneceu a impressora.

A formação envolveu todo o funcionamento físico da impressora, como operar, substituir material e fazer uma correta manutenção da máquina (Figura 122).

Figura 122 - Impressora 3D.

A máquina adquirida utiliza como material de impressão "ABS" e material de suporte "PLA" ambos da marca "Stratasys".

Em termos de funcionamento inicialmente é impressa uma camada de "ABS" na plataforma para fixar as primeiras camadas à base de suporte e posteriormente é impressa a base da camada de suporte. Esta camada permite a fixação da peça de modo a não sofrer deslocações tornando a qualidade final superior.

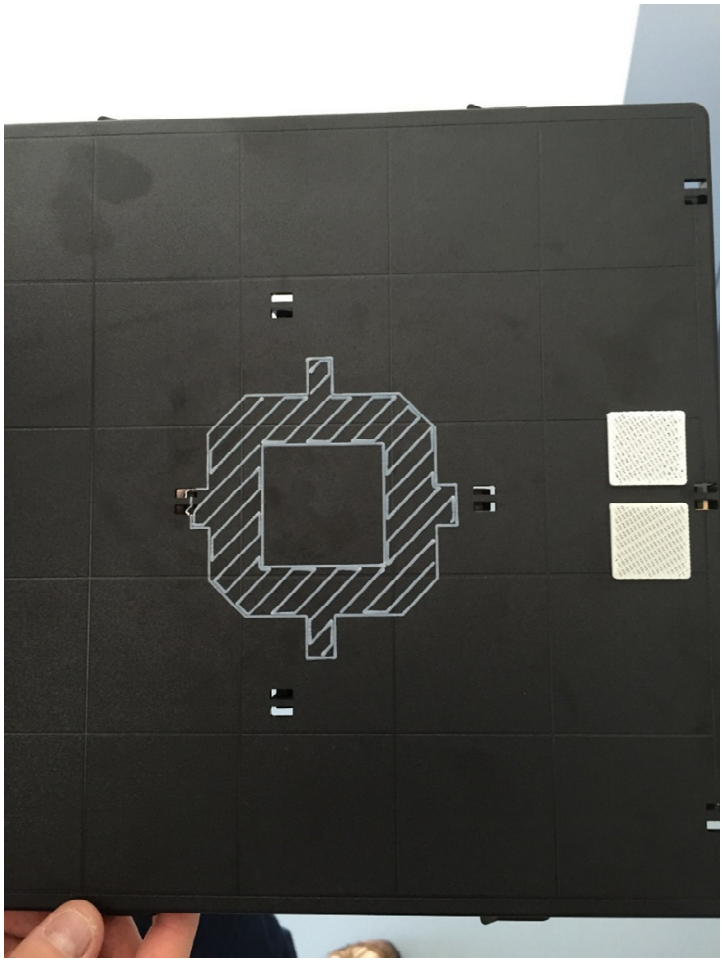


Figura 123 - Plataforma de impressão com camada de "ABS".

Para cada impressão é necessário uma plataforma nova (Figura 123) pois a máquina antes de iniciar o processo de impressão faz uma calibração e verificação para identificar se a plataforma contém defeitos que possam implicar que a peça não seja impressa corretamente.

A formação incidiu não só no funcionamento e manuseamento da máquina mas também sobre o "software" que esta impressora utiliza, nomeadamente o "Insight" (Figura 124).

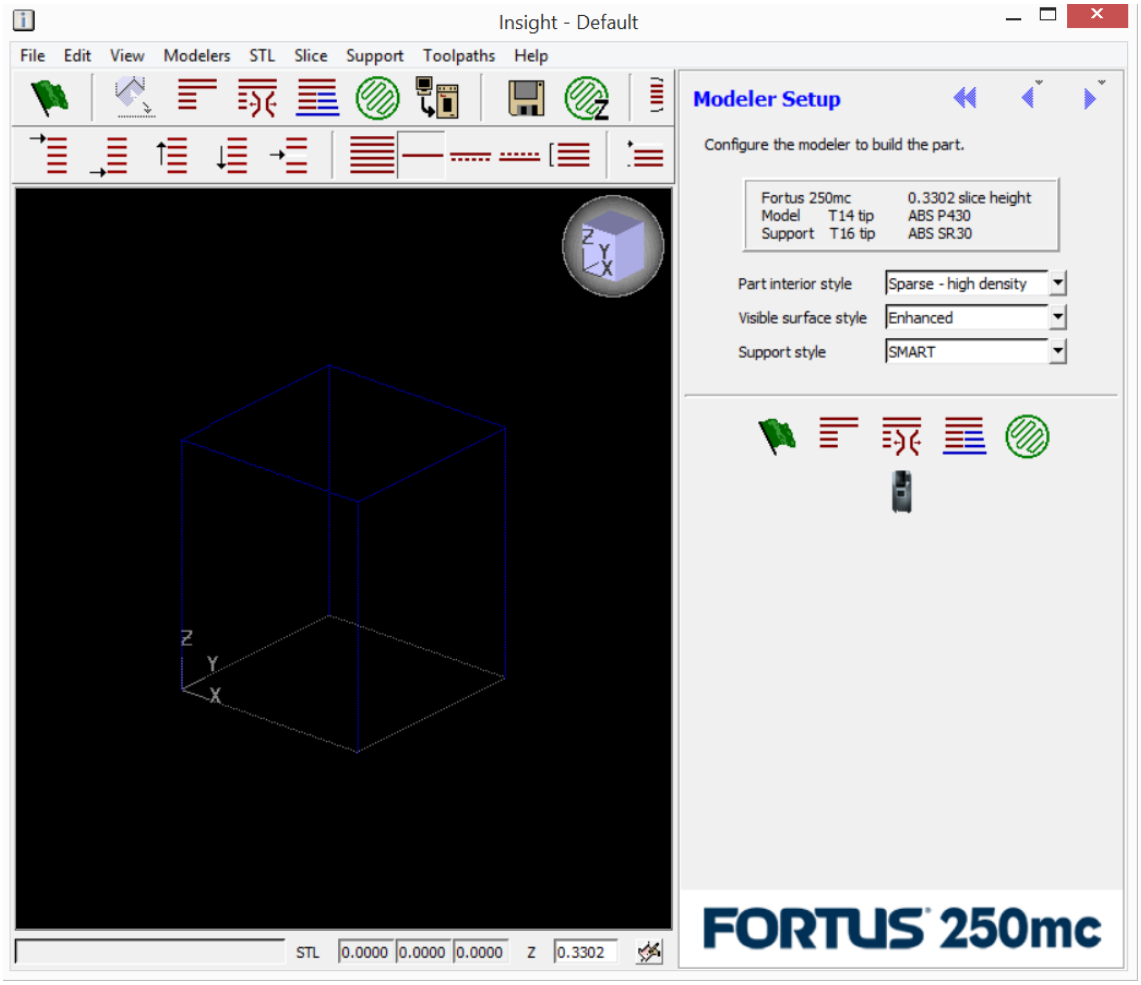


Figura 124 - Software Insight.

O "software" permite a importação da peça que se pretende criar e todas as configurações para a impressão da peça, nomeadamente inclinação, orientação, densidade, quantidade de suporte utilizado posição na plataforma, etc. Por fim quando todas as configurações forem escolhidas e geradas é apresentado um relatório contendo a duração da impressão, volume de "ABS" e de "PLA" utilizado.

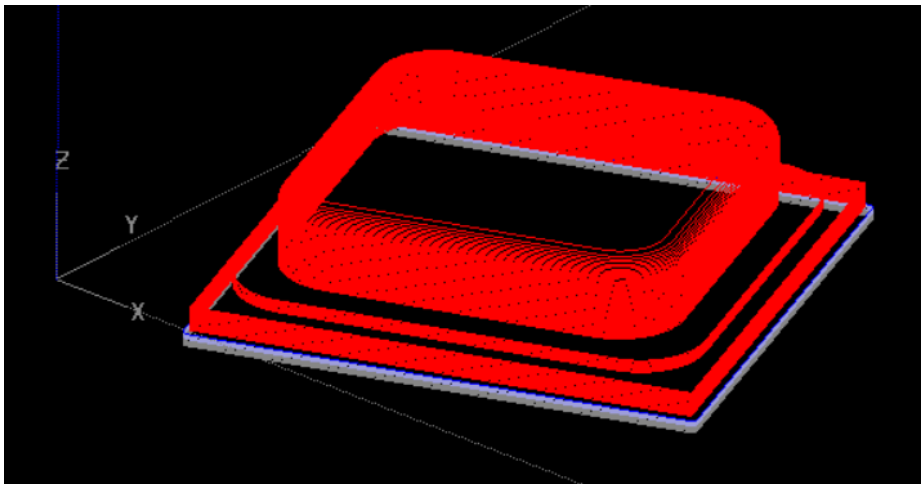


Figura 125 - Recorte de uma peça em camadas.

A Figura 125 mostra o recorte de uma peça.

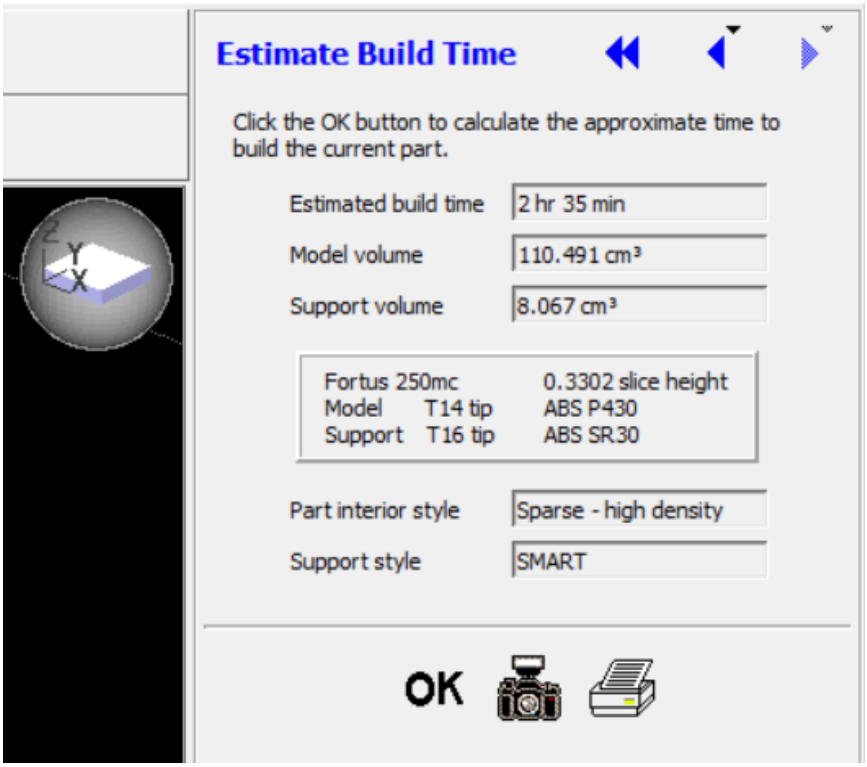


Figura 126 - Relatório da peça a imprimir.

Posteriormente a peça é enviada para a máquina para impressão via comunicação "Ethernet" (Figura 126).

Após a conclusão da formação foram impressas peças com acompanhamento do técnico para se perceber na prática o funcionamento da máquina (Figura 127).

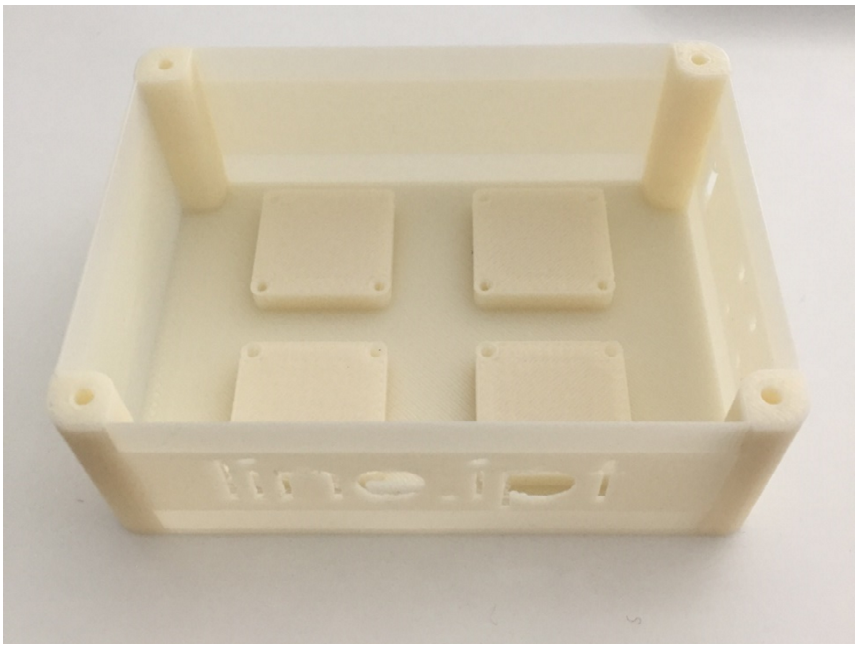


Figura 127 - Peça impressa na impressora 3D.

6.6 Seminário "Sagitron"

No dia 10 de Março de 2015 presenciou-se o seminário da empresa "Sagitron" que visou mostrar e apresentar todos os novos produtos, equipamentos e funcionalidades desenvolvidas pela "Microchip" nos últimos tempos. Este seminário foi de uma importância extrema pois foram dados a conhecer todos os novos equipamentos recentemente lançados no mercado e os que futuramente irão ser lançados. Aumentou-se assim o conhecimento dos componentes em causa, assim como das funcionalidades novas para deste modo todos os protótipos que serão desenvolvidos no futuro sejam tecnologicamente avançados e eficientes tanto em termos energéticos como em execução de tarefas.

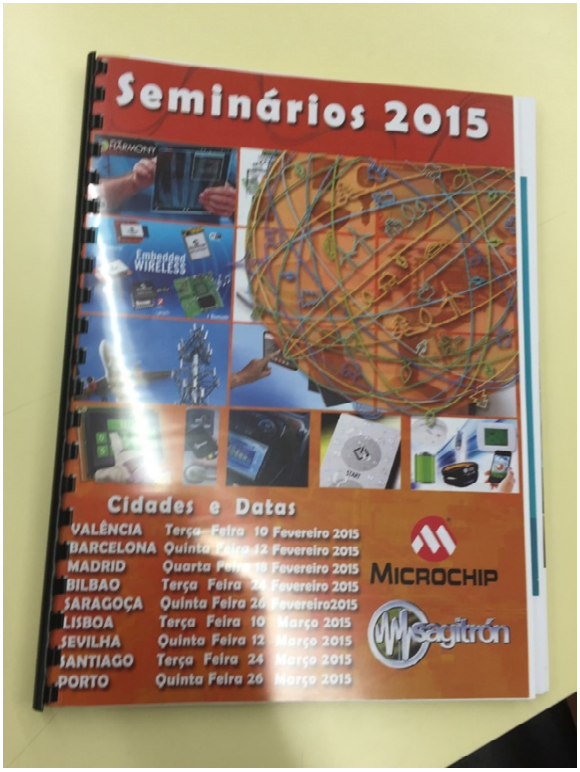


Figura 128 - Apresentação do seminário.

A Figura 128 mostra o manual fornecido durante o seminário.

6.7 Curso "Harmony"

No dia 5 de Abril de 2015 marcou-se presença num curso com a duração de um dia sobre a nova plataforma de desenvolvimento e programação de microcontroladores "Harmony" da "Microchip". Este curso foi ministrado pela empresa "Sagitron". Numa primeira fase foi apresentada a tecnologia, arquitetura e funcionamento dos novos microcontroladores "PIC23MZ" que podem operar a uma frequência de até 200MHz. Nesta fase foi também apresentado todo o "software" de programação "Harmony" e todas as suas funcionalidades e potencialidades. Este "software" de desenvolvimento vem introduzir um novo paradigma de programação de microcontroladores em que toda a programação é 100% modular. Ou seja, todos os algoritmos e configurações que se façam para um determinado microcontrolador da família PIC32 é 100% funcional para qualquer desses microcontroladores e adaptado imediatamente para qualquer micro sem qualquer tipo de modificação no código, mantendo-se todas as funcionalidades intactas. Isto é possível visto que esta plataforma usa "Plibs" que são ficheiros que têm todas as informações de todos os

microcontrolador e permite a exportação do código para outro Micro sem qualquer alteração no algoritmo.

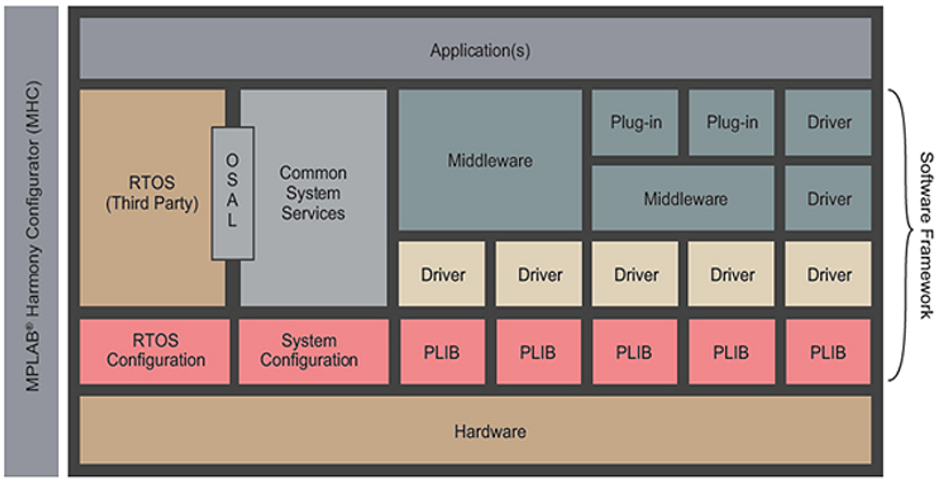


Figura 129 - Arquitetura Harmony.

A Figura 129 mostra o esquema geral da arquitetura "Harmony".

A plataforma "Harmony" disponibiliza também ferramentas avançadas para todas as configurações de funcionalidades de microcontroladores (temporizadores, interrupções, etc.). Estas ferramentas permitem uma configuração gráfica e "user-friendly" o que vem diminuir bastante o tempo de desenvolvimento de projetos pois o código é gerado automaticamente com base nestas configurações.

Numa segunda parte do curso foi realizada uma abordagem prática com recurso à plataforma de desenvolvimento "PIC32MZ" (Figura 130).

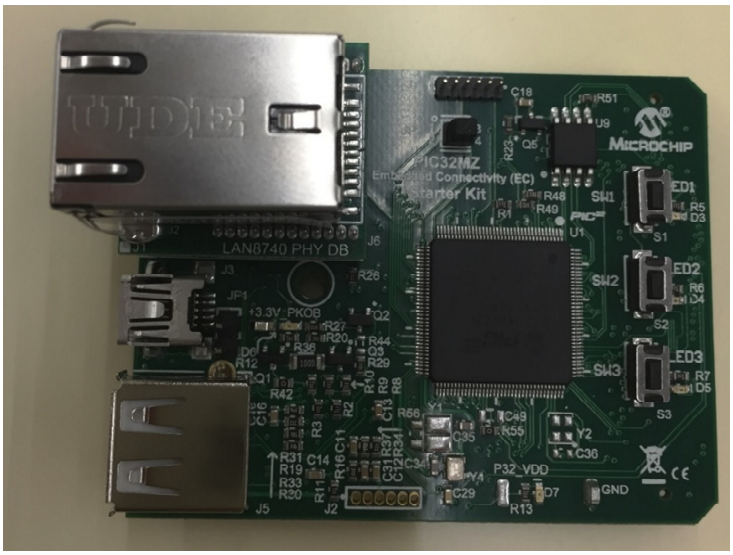


Figura 130 - PIC32MZ.

Foram executados exercicios práticos "passo-a-passo" (Figura 131) que demonstraram todas as capacidades tanto do microcontrolador como do "software" de programação.

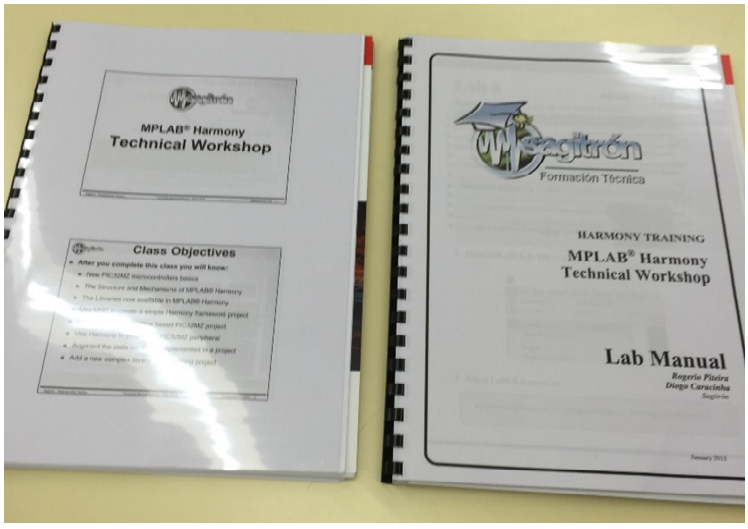


Figura 131 - Apresentação e Manual do curso.

6.8 Palestra "Arduino Day"

No dia 27 de Março de 2015 o autor participou como orador no evento "Arduino Day 2015" com a exposição de dois temas através do LINE.IPT, "Atmel Studio" (Figura 132) e "Arduino em Ambiente Industrial" (Figura 133 e Figura 133).

Figura 132 - Capa da apresentação "Atmel Studio".

Figura 133 - Capa da apresentação "Arduino em ambiente industrial".

Com estes temas pretendeu-se expor conhecimentos e experiências adquiridas durante os desenvolvimentos para os projetos das empresas.

6.9 Seminário "Sustentabilidade"

No dia 25 de Março de 2015 assistiu-se ao evento "Workshop de Sustentabilidade: Empresas e I&D" (Figura 134).

Figura 134 - Programa do "Workshop sustentabilidade".

6.10 Protótipo de exposição "Maquete LINE"

Com o interesse das empresas em conhecer as tecnologias desenvolvidas pelo LINE houve a necessidade de mostrar as potencialidades ao nível de desenvolvimento de protótipos, microcontroladores, aquisição de dados e redes. Foi assim pedido por parte do LINE que se procedesse ao desenvolvimento de um protótipo que servisse de exposição para potenciais clientes.

Decidiu-se monitorizar os seguintes parâmetros:

- Temperatura;
- Luminosidade;
- Tensão de referência de um potenciómetro.

A temperatura será monitorizada com recurso a um sensor "LM35" que varia a tensão com base na temperatura. A luminosidade será monitorizada com recurso a um "LDR". A tensão de referência do potenciómetro será monitorizada pela variação do mesmo.

Como microcontrolador escolheu-se um "Atmega328P-PU" da "Atmel" pois estes já se encontram em armazém no LINE e são de formato "DIP" o que facilita a construção do circuito. Para as comunicações "wireless" escolheu-se a "breakout-board" "CC300" pois tem um custo reduzido e implementa todas as funcionalidades necessárias ("Web Client") para comunicação com um servidor. Para simular atuações introduziu-se um botão do tipo "switch" e um "LED" para o cliente experimentar as funcionalidades do protótipo e observar as atuações no protótipo ou numa aplicação "SCADA" através de comunicações "wireless". Todos estes componentes serão colocados numa caixa construída para levar às empresas. Será colocado também um "LCD" "40*4" que permita observar em tempo real a monitorização das variáveis. Desenvolveu-se então um circuito com todas estas funcionalidades cujo esquema elétrico se encontra representado na Figura 135.

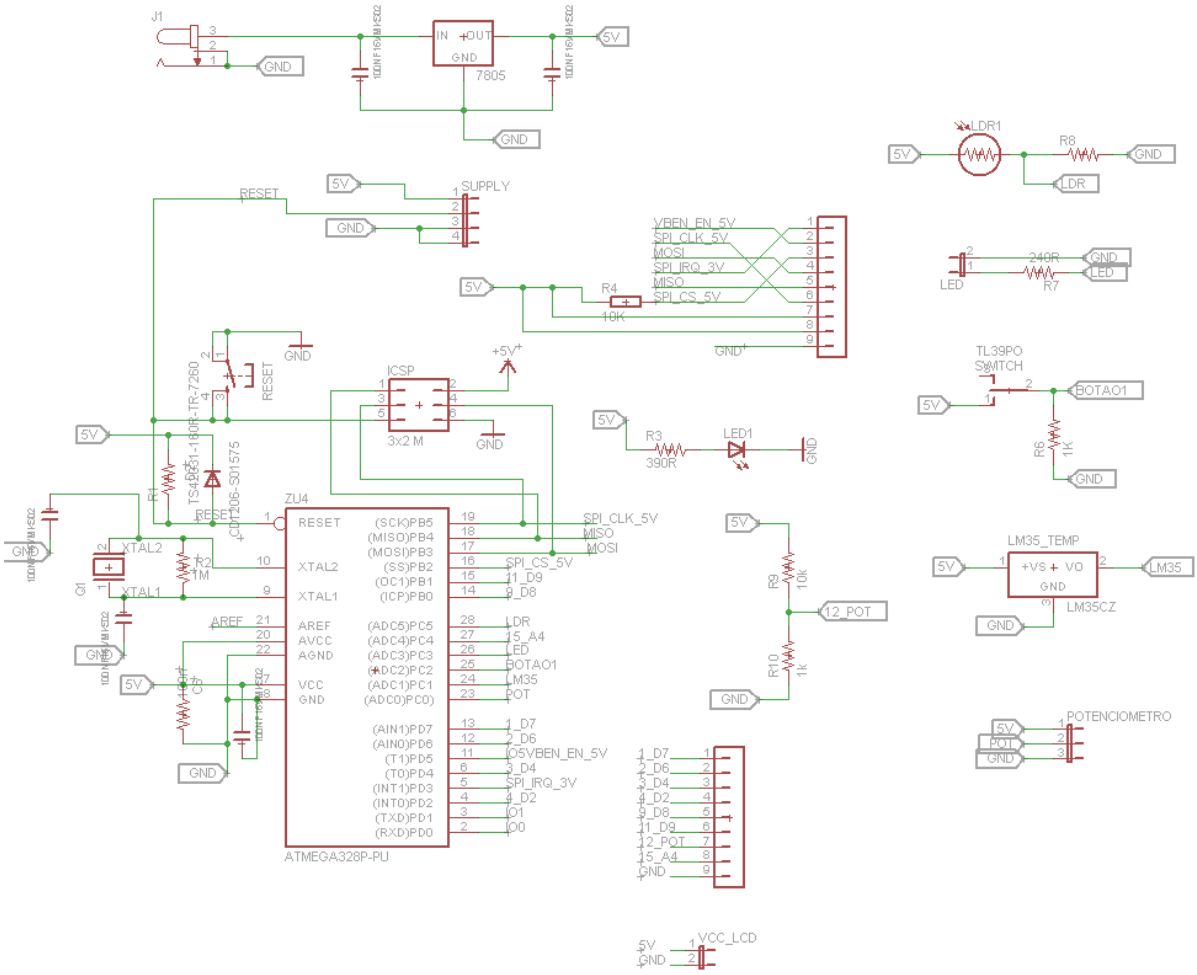


Figura 135 - Esquemático do circuito desenvolvido.

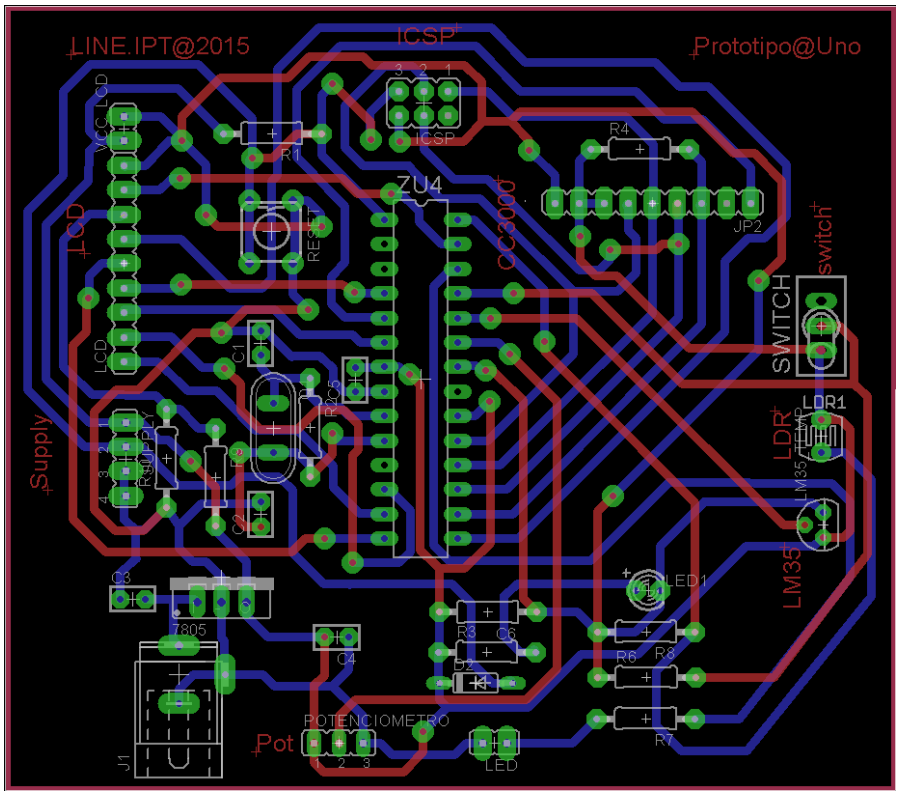


Figura 136 – PCB desenvolvida.

A Figura 136 mostra a PCB desenvolvida do circuito principal do protótipo.

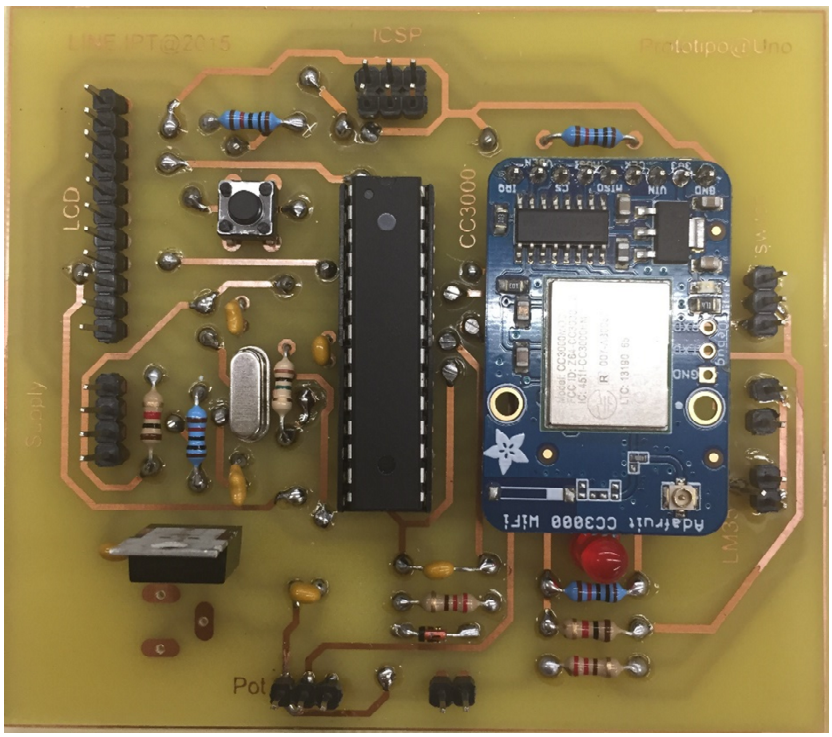


Figura 137 - Placa desenvolvida e construída.

A Figura 137 mostra a "PCB" construída para o protótipo.

Para a alimentação deste sistema desenvolveu-se um circuito (Figura 138) com base numa fonte comutada "AC/DC" da "XP POWER" que tem como entrada a tensão da rede e como saída 9V de tensão DC, tensão esta que vai alimentar o regulador de tensão de 5V no circuito. Esta fonte é soldada diretamente na "PCB".

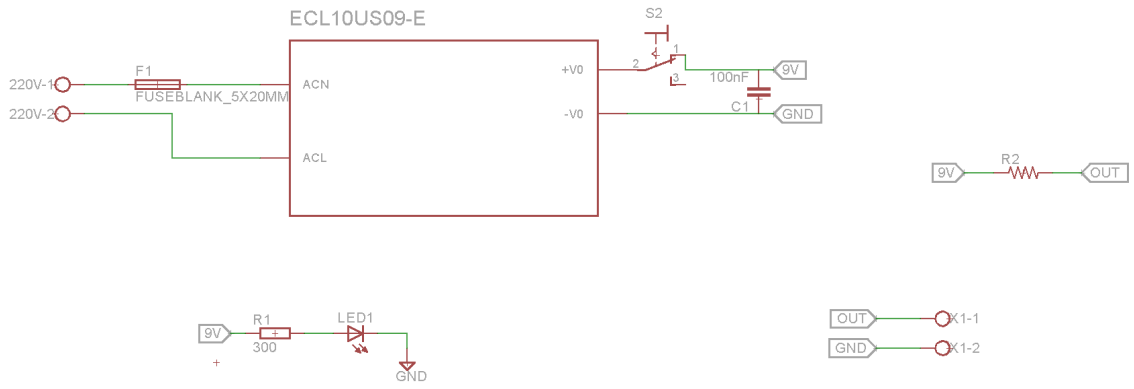


Figura 138 - Esquemático do circuito.

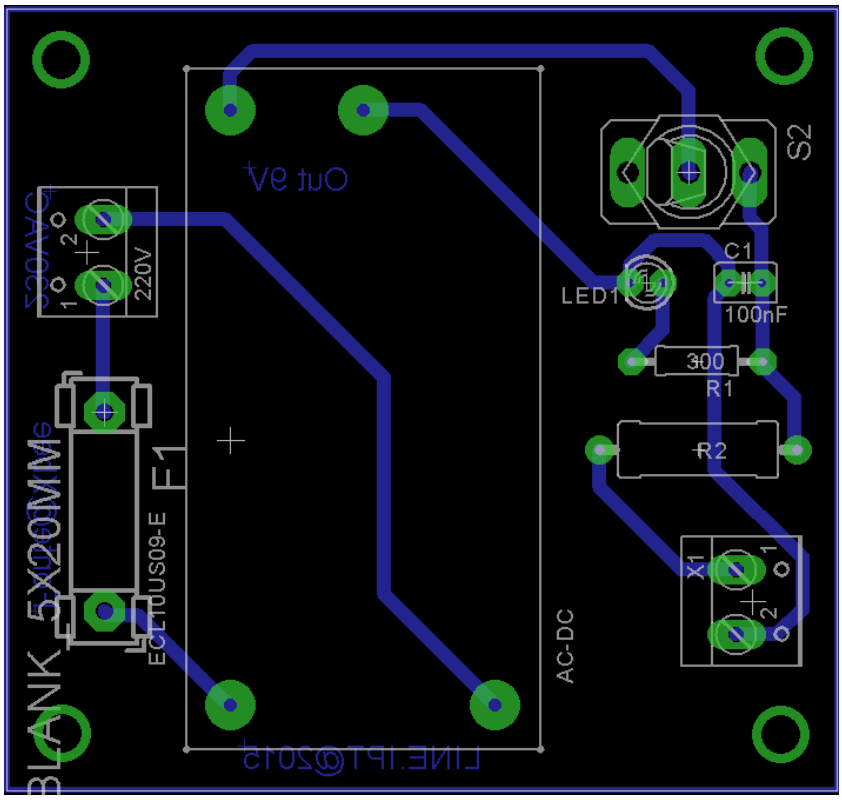


Figura 139 - Ficheiro ".brd" do circuito.

A Figura 139 mostra o ficheiro ".brd" do circuito desenvolvido.

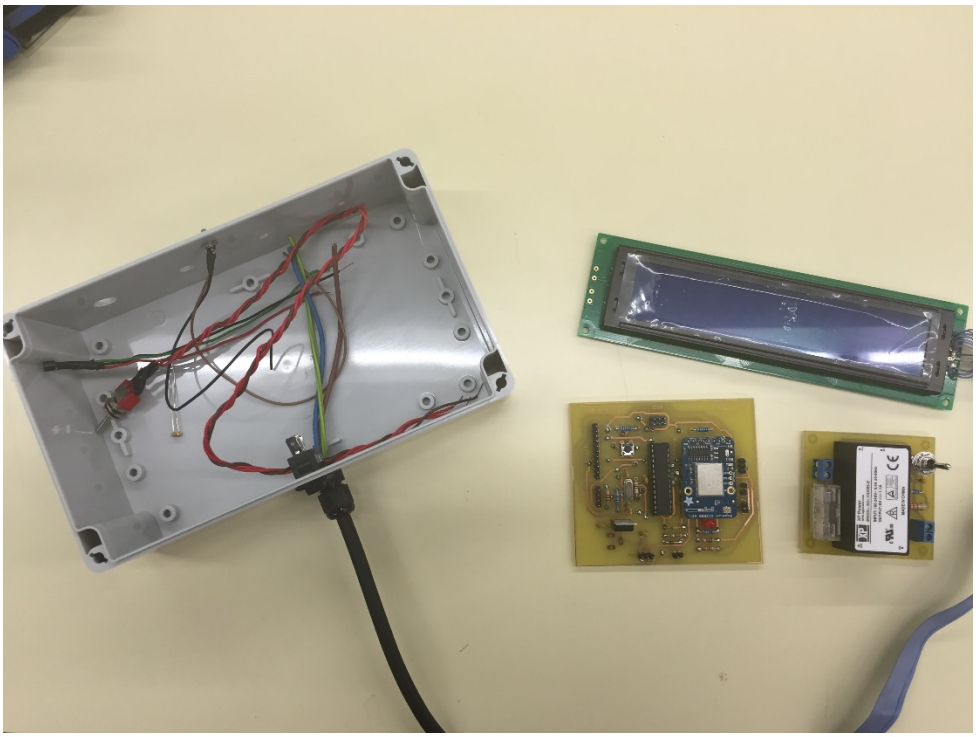


Figura 140 - Componentes do sistema.

A Figura 140 mostra os diferentes componentes do sistema.

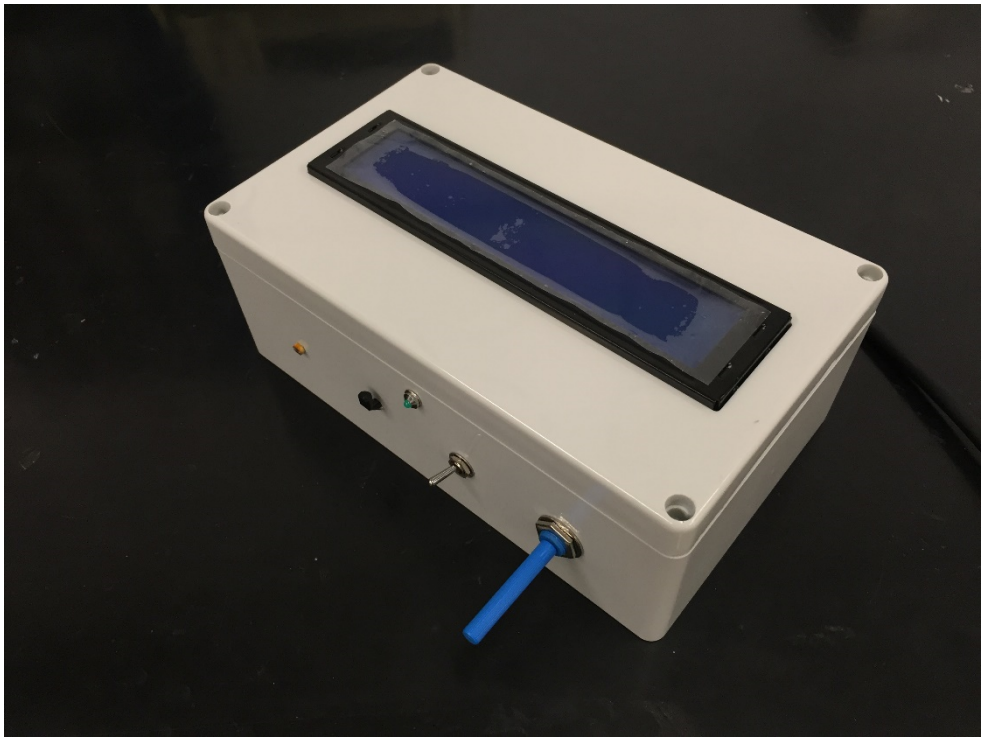


Figura 141 - Protótipo final.

A Figura 141 mostra o protótipo construído.

7 Conclusão

O LINE.IPT encontra-se envolvido numa vasta área de aplicações que vão desde a Engenharia Eletrotécnica, Mecânica, Informática até à Gestão, tal como se pôde observar nos capítulos anteriores.

Iniciou-se este trabalho apresentando uma visão geral do trabalho desenvolvido durante o estágio, os projetos principais e todas as outras tarefas que são desenvolvidas diariamente. Apresentou-se o trabalho desenvolvido para o projeto da Mitsubishi, nomeadamente a investigação acerca do funcionamento das máquinas de soldar e dos seus controladores que permitiu desenvolver todo o sistema para adquirir os sinais durante uma soldadura (tensão de soldadura, corrente e pressão), analisá-los, armazená-los e para indicar o estado da soldadura. Implementaram-se comunicações "wireless" para a interação entre ambos os módulos de cada protótipo bem como com a aplicação "SCADA" em funcionamento no PC. Todos os dados adquiridos durante os testes foram armazenados num ficheiro de texto ".txt" para serem analisados e exportados para uma base de dados.

O trabalho seguinte consistiu na apresentação do projeto realizado para a H&H. Este consistiu numa pesquisa de toda a legislação e normas necessárias que envolvem a normalização do funcionamento de uma Camara Hiperbárica. Esta investigação permitiu concluir-se que as Camaras hiperbáricas com aplicações médicas são as mais complexas a nível de requisitos exigidos para a homologação. Assim foi sugerido à H&H a escolha desta opção pois o caderno de encargos final a ser desenvolvido irá abranger todas as diferentes aplicações das camaras hiperbáricas devido ao facto dos requisitos para as aplicações medicinais envolverem a esmagadora maioria das outras aplicações. Na próxima fase serão analisadas em detalhe todas as normas selecionadas para o desenvolvimento do caderno de encargos final.

O projeto para a Pneusines envolveu uma forte componente de desenvolvimento mecânico pois toda a estrutura da máquina terá de ser modificada. Relativamente à parte de automação todo o sistema foi projetado para garantir uma expansibilidade futura a nível de funcionalidades comprovada com os equipamentos adquiridos para esse fim.

A primeira tarefa desenvolvida no início do estágio (construção de um protótipo desenvolvido para o projeto da Martifer) teve um forte impacto a nível de metodologia de

trabalho pois foi o primeiro contacto com as estratégias de desenvolvimento de projetos seguidas no LINE servindo de ponto de partida para todo o trabalho seguinte desenvolvido.

Os relatórios técnicos, gestão de orçamentos, gestão de projetos permitiram ir um pouco além da engenharia, entrando num campo que não é tipico nesta área. Desenvolver projetos não contempla apenas a parte técnica, sendo fundamental a "gestão" o que implica uma grande cooperação dentro de uma equipa de trabalho para que todas as partes trabalhem em sincronia de modo a que o resultado final seja o desejado.

Todas as formações e "Workshops" vieram complementar os conhecimentos adquiridos durante o desenvolvimento dos projetos contribuindo com novas ideias, componentes e metodologias de trabalho nomeadamente o curso "Harmony" da "Microhip" ministrado pela Sagitron. A formação incidiu sobre a nova impressora 3D da "Stratasys", adquirida pelo LINE. Esta veio dar um conhecimento bastante técnico sobre uma área que atualmente é muito debatida, mas que ao nível de precisão e de funcionalidades fornecidas por esta máquina não é comum, ganhando assim uma vantagem competitiva relativamente a outras empresas que executam impressões com máquinas de precisão e qualidade de produto final inferior.

Por fim, o protótipo de exposição "Maquete LINE" desenvolvido veio corrigir uma vulnerabilidade por parte do LINE que era a exposição das tecnologias desenvolvidas e potencialidades de projetos. Obteve-se assim um protótipo físico para dar a conhecer às empresas um leque de funcionalidades que poderão ser exploradas nas mais diversas áreas de aplicação.

Este estágio levou a um grande crescimento profissional do autor pois foi o primeiro contacto de trabalho real nesta área. O trabalho desenvolvido diariamente no LINE veio mostrar a grande complexidade que é lidar com empresas, nomeadamente na área de automação industrial, em que os produtos desenvolvidos exigem um elevado grau de robustez e fiabilidade. Lidar com vários projetos simultaneamente veio revelar-se ser um grande desafio, pois estão envolvidas áreas de aplicações diferentes e muito distintas o que leva a um contacto simultâneo com diferentes tecnologias envolvidas, nomeadamente eletrónica, microcontroladores, instrumentação e sensores, programação de software e interfaces, telecomunicações entre muitos outros. Outro grande desafio foi garantir que os protótipos desenvolvidos sejam financeiramente atrativos e viáveis para a sua implementação e

replicação pois não basta ter um grande orçamento disponível para o desenvolvimento se o preço do produto final não for viável.

Devido ao grande volume de informação do trabalho desenvolvido, a tarefa de escrita e seleção da informação foi muito exigente para manter o relatório com um número de páginas na ordem da centena e meia, pelo que em muitos apenas se apresenta o circuito/programa final desenvolvido sem apresentar todos os processos de desenvolvimento, dando assim uma ideia de simplicidade que em muitos dos casos não corresponde à realidade. O objetivo foi apresentar a maior quantidade de informação possível sem que o texto se tornasse "pesado" do ponto de vista da sua leitura.

8 Referências Bibliográficas

[1] Andrej Lebar, Luka Selak, Rok Vrabic, Peter Butala, "Online Monitoring, Analysis, and Remote Recording of Welding Parameters to the Welding Diary", University of Ljubljana, Faculty of Mechanical Engineering, Slovenia, 2012.

[2] Dawei Zhao, Yuanxun Wang, Suning Sheng, Zongguo Lin, "Real time monitoring weld quality of small scale resistance spot welding for titanium alloy", Department of Mechanics, School of Civil Engineering and Mechanics, Huazhong University of Science and Technology, Wuhan 430074, China, 2013.

[3] Yanhua Ma, Pei Wu, Chuanzhong Xuan, Yongan Zhang, and He Su, "Review on Techniques for On-Line Monitoring of Resistance Spot Welding Process", College of Mechanical and Electrical Engineering, Inner Mongolia Agricultural University, Hohhot 010018, China, 2013.

[4] LI Ru-xiong, "Analyzing system of electric signals in spot welding process", Department of Mechanical and Electrical, Jingdezhen Ceramic Institute Jingdezhen, China, 2012.

[5] Kenneth V. Cartwright, PHD Thesis, "Determining the effective or RMS voltage of various waveforms without calculus", School of Sciences and Technology, College of the Bahamas, 2007.

[6] Jeremy Blum, "Exploring Arduino: Tools and Techniques for Engineering Wizardry", WILEY, 2013.

[7] Jorge Penedo, Ana Madahil, Ana Ferreira, Ana Nunes, Jorge Pereira, José Venâncio, Maria Maia, Paula Alves, "Carta de Equipamentos Pesados em Saude", Despacho N.º 3484/2013, publicado em Diário da Républica N.º 45, 2ª Série, 5 de Março de 2013.

[8] IPQ, "Norma Portuguesa EN ISO 13485", 2014.

[9] Paul Scherz, Simon Monk, "Practical Electronics for Inventors 3rd Edition", McGraw-Hill, 2013.

[10] http://mergulhobarcelos.planetaclix.pt/tabelas_de_descompressao.htm, 2015.

[11] D. Mathieu, "Handbook on Hyperbaric Medicine", Springer, 2006.

[12] http://solarelectricityhandbook.com/solar-irradiance.html, 2015

9 Anexos

Para além dos anexos apresentados nas seguintes páginas encontram-se também em formato digital no "cd" entregue, anexos extra para os diferentes projetos apresentados na tese, com explicações mais detalhadas de alguns circuitos e figuras bem como de opções e decisões tomadas durante o desenvolvimento.

Anexo 1:

Folha de parâmetros de soldadura:

REV:2

TIMER OBARA(Antigo)	PISTOLA 1		PISTOLA 2	
1008	AX207023 CM627801		C-45-301 (A) CM627101	
PROGRAMA	SOLENOIDE 1		SOLENOIDE 2	
	SW1	SW3	SW2	SW4
SQUEEZE	25	30	20	0
SLOPE	3	3	3	0
WELD 1	2	0	0	0
COOL	5	0	0	0
WELD 2	11	5	7	0
HOLD	15	15	15	0
OFF	5	5	5	0
HEAT 1	3000	0	0	0
HEAT 2	10000	10000	8800	0
PULSE	0	0	0	0
TURN-RATIO	19		19	
CUR-LIMIT +	5%		5%	
CUR-LIMIT -	5%		5%	
VOLT FACTOR	1		1	
ELEC.FORCE(Kgf)	(B) 300		225	
	0		0	
ELECT.DIAMET.	4,5		4,5	

LETRA ALTERAÇÃO	ALTERACÃO	DATA	ASSIN.
	Novo Lay-Out Soldadura TD	12-Abr-05	
A	Transferência de Pinça do transformador 1015 para 1008	20-Jul-05	
B	Em Pistola1 E.force de 360 para 300.	07-Fev-06	

Cópia para Divisão de Produção

Anexo 2:

Programa do módulo "slave" do secundário implementado - slave_s_P1_V3.ino:

```
#include <avr/io.h>
#include <avr/interrupt.h>
//#include "digitalWriteFast1.h"
#include <XBee.h>
#include <Wire.h>
#include "RTClib.h"
#include <SdFat.h>
boolean flag_transmite = false; //indica que a soldadura acabou
volatile boolean flag0;
boolean flag_sd = false;
int ano, mes, dia, hora, minuto, segundo;
///////////////sd
const uint8_t chipSelect = 53;
const uint32_t SAMPLE_INTERVAL_MS = 200;
#define FILE_BASE_NAME "A"
#define sdpackage_size 23
uint16_t sdpackage[sdpackage_size] = { 0 };
SdFat sd;
SdFile file;
uint32_t logTime;
const uint8_t BASE_NAME_SIZE = sizeof(FILE_BASE_NAME) - 1;
char fileName[13] = FILE_BASE_NAME "00.txt";
RTC_DS1307 rtc;
#define TAMANHO 40 //comprimento dos arrays para amostras
#define TAMANHO_M 100 //comprimento dos arrays para medias
//#define MIN 180//valor minimo para se considerar desligado
#define MIN 30  //valor minimo para se considerar desligado
#define SLOPE 20//diferenca de tens�o media minimo para considerar que est� na fase slope
#define COOL 16//numero de ciclos maximo para ainda considerar fase cool
#define ID_MAQUINA 1 //Identifica��o da maquina
```

```
#define SIZE_FRAME 43//26 //tamanho do pacote de transmiss◆o dos slaves
#define BAUDRATE_XBEE 57600 //velocidade de transmiss◆o com o master
#define BAUDRATE_WI 115200//velocidade de transmiss◆o com a porta serie para debug
/****************Carateres especiais para rece◆◆o de dados do slave prim◆rio************************/
#define ESCAPE_FRAME 47 //  '/'
#define START_FRAME 65 //  'A'
////////////////////////////////////////////////////////////////////////////////////////////////xbee para enviar dados de soldadura
XBee xbee = XBee();
unsigned long start = millis();
XBeeResponse response = XBeeResponse();
Rx16Response rx16 = Rx16Response();
uint8_t payload[SIZE_FRAME] = {
        0 };
Tx16Request tx = Tx16Request(0x1884, payload, sizeof(payload));
TxStatusResponse txStatus = TxStatusResponse();
////////////////////////////////////////////////////////////////////////////////////////////////xbee para receber ok/not ok soldadura
XBee xbee2 = XBee();
XBeeResponse response2 = XBeeResponse();
Rx16Response rx162 = Rx16Response();
uint8_t payload2[2] = {
        0 };
Tx16Request tx2 = Tx16Request(0x1884, payload2, sizeof(payload2));
TxStatusResponse txStatus2 = TxStatusResponse();
//////////////////////////////////////////////////////
unsigned int value = 0;
volatile boolean debug = false;
//////////////////////////////////////////////////////
//buffer que guardam as amostragens de um ciclo
long buf_v[TAMANHO];
long buf_i[TAMANHO];
//buffers que guardam as medias por ciclo
```

```
int buf_media_v[TAMANHO_M];
int buf_media_i[TAMANHO_M];
boolean flag_weld_on = false;//indica a fase soldadura
boolean flag_weld_ant = false;//guarda o estado anterior de "flag_weld_on"
boolean flag_slope_on = false;//indica a fase "slope"
int ind_buf = 0;//index dos buffers "buf_v,buf_i"
int media_ant = 0;//variavel auxiliar para guardar a media do ciclo anterior
int slope = 0;//numero de ciclos da fase slope
int weld = 0;//numero de ciclos da fase weld
int weld2 = 0;//numero de ciclos da fase weld2
int contador_s = 0;//contador de ciclos da fase COOL
int cool = 0;
byte erro = 0;
int first_v = 0;//variavel auxiliar para guardar o primeiro valor
int first_i = 0;//variavel auxiliar para guardar o primeiro valor
int pico_v = 0; //tensao no ultimo ciclo da fase slope
int pico_i = 0; //corrente no ultimo ciclo da fase slope
int media_final_v = 0;//tensao media durante a fase weld
int media_final_i = 0;//corrente media durante a fase weld
int media_final_v2 = 0;//tensao media durante a fase weld
int media_final_i2 = 0;//corrente media durante a fase weld
int declive_v = 0;//declive de tens□o durate a fase slope
int declive_i = 0;//declive de corrente durate a fase slope
unsigned long n_soldadura = 0;//contador de soldaduras efetuadas desde o arranque do
arduino
long v_lastcycle = 0;
long p_lastcycle = 0;
//int cont_amostr=0;//contador de numero amostras num ciclo
boolean ss = false;//para alternar o estado do led para indica□□o de nova amostra
byte buff_trans[SIZE_FRAME + 1];
unsigned long xpto = 0;
unsigned long time = 0;
unsigned long tesr = 0;
byte rdia = 0;
```

```
unsigned long time_ant = 0;
long xpto_v = 0;
long xpto_i = 0;
long media_slope_v = 0;
long media_slope_i = 0;
long media_weld_v = 0;
long media_weld_i = 0;
long media_weld_v2 = 0;
long media_weld_i2 = 0;
long media_ant_v = 0;
long media_ant_i = 0;
long media_actu_v = 0;
long media_actu_i = 0;
boolean status_receive = false;
unsigned int state = 0;
unsigned long actual = 0;
volatile unsigned long anterior = 0;
volatile int count = 0;
////////////////////////////////////////////////////////////////
volatile boolean wtd = false;
int fases = 0; //0=em espera1, 1=slope soldadura1,2=soldadura1, 3=espera2, ,
4=soldadura2, 5=fim soldadura
//fun��o que guarda as medias do ciclo atual
void setup() {
  pinMode(31,OUTPUT);
  pinMode(37,OUTPUT);
  D31_HIGH();//apagar leds
  D37_HIGH();
  // put your setup code here, to run once:
Serial.begin(9600);
    //  cli();

      Serial1.begin(BAUDRATE_XBEE);
          xbee.setSerial(Serial1);
```

```
        xbee2.setSerial(Serial1);
if (!sd.begin(chipSelect, SPI_SIXTEENTH_SPEED)) sd.initErrorHalt("nao abriu sd");
        ////////////////////////////////////rtc
#ifdef AVR
        Wire.begin();
#else
        Wire1.begin(); // Shield I2C pins connect to alt I2C bus on Arduino Due
#endif
        rtc.begin();
        if (!rtc.isrunning()) {
                Serial.println("RTC is NOT running!");
                // following line sets the RTC to the date & time this sketch was compiled
                //rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
                // This line sets the RTC with an explicit date & time, for example to set
                // January 21, 2014 at 3am you would call:
                //rtc.adjust(DateTime(2015, 3, 13, 17, 50, 0));
        }
        ////////////////////////////////////
        ////////////////interrup□ao para leitura de dados frequencia de 2k
        cli();//desativa interrup□□es
        /////Timer3///////////
        TCCR3A = 0;//reset ao timer //registos com valores de configura□□es
        TCCR3B = 0;//reset ao timer //registos com valores de configura□□es
        OCR3A = 124;//valor para overflow do timer
        TCCR3B |= (1 << WGM12);//liga modo ctc
        TCCR3B |= (1 << CS31) | (1 << CS30);//prescaler de 64
        //TCCR3B |= (1 << CS32);//prescaler de 256
        TIMSK3 |= (1 << OCIE3A);//liga o timer para modo de compara□ao de
interrup□ao
        ////////////////////////////////////////////interrupcao para watchdog do botao de soldadura
        //set timer1 interrupt at 1Hz
        TCCR1A = 0;// set entire TCCR1A register to 0
        TCCR1B = 0;// same for TCCR1B
        TCNT1 = 0;//initialize counter value to 0
```

```
        // set compare match register for 1hz increments
        OCR1A = 15624;// = (16*10^6) / (1*1024) - 1 (must be <65536)
        // turn on CTC mode
        TCCR1B |= (1 << WGM12);
        // Set CS10 and CS12 bits for 1024 prescaler
        TCCR1B |= (1 << CS12) | (1 << CS10);
        // enable timer compare interrupt
        TIMSK1 |= (1 << OCIE1A);
        ////////////////////////




        sei();
        ////////////////////////////////int4 e int5 bits diretos controlo por botao pino D2(int4) e
D3(int5)
        EICRB = 0b00001111 ;//ascendente para int 4 e 5
        //EICRB = 0x0F;//em hex
        EIMSK = 0b00110000 ;//int 4 e 5
delay(2000);
}
void loop() {
        while (1){
                //-------------------------------------------------------------------------------------
-------------------------------------------
                if (flag_transmite == true){
  //Serial.println("flag de enviar a 1");
update_time();
xbee.send(tx);//transmite os novos dados
//debug_value();
ack_receive();
if (status_receive == true){//flag para receção do ok/not ok da soldadura do master
ok_receive();
}
if (flag_sd == true){
```

```
cria_frame_sd();
writesd();
flag_sd = false;
flag_transmite = false;
}
flag_transmite = false;
}//flag trnsmite
else{
flag_transmite = false;
}
                delay(50);
        }
}
void ack_receive(){
                        xbee.readPacket();
                        if (xbee.getResponse().isAvailable()) {
                                if (xbee.getResponse().getApiId() ==
TX_STATUS_RESPONSE) {

        xbee.getResponse().getZBTxStatusResponse(txStatus);
                                        if (txStatus.getStatus() == SUCCESS) {//reccebeu
confirmação de pacote recebido
                                                status_receive = true;
                                // Serial.println("recebeu ack");
                                                //delay(50);
                                        }
                                        if (!(txStatus.getStatus() == SUCCESS)) {
                                                Serial.println("NOT SUCCESS");
                                                flag_sd = true;
                                        }
                                }
                        }
                        else if (xbee.getResponse().isError()) {
                                Serial.println("ERROR");
                                flag_sd = true;
```

```
                    }
                    else{
                        Serial.println("TIMEOUT");
                        flag_sd = true;
                    }
}
void ok_receive(){
                        xbee2.readPacket(500);
                        if (xbee2.getResponse().isAvailable()) {
                            if (xbee2.getResponse().getApiId() ==
RX_16_RESPONSE) {

        xbee2.getResponse().getRx16Response(rx162);
                                state = (int)rx162.getData(0);//rececao se
soldadura esta ou nao ok
                                if (state == 10){
                                    //ativar led de operador
                                    //Serial.println("Ok");
                            D37_HIGH();//quando flag0 voltar a 1
                            D31_LOW();//apaga leds
                                }
                                if (state == 20){
                                    //ativar led de operador
                                    //Serial.println("NOTOk");
                             D37_LOW();
                             D31_HIGH();
                                }
                                //status_receive=false;
                                //flag_transmite=false;
                                //sei();
                             }
                        }
                        else{
                            status_receive = true;
                        }
```

```
                    status_receive = false;
                    flag_transmite = false;
}
void debug_value(){
                if (weld2>0){
                    // Serial.println("Numero de soldaduras");
                    // Serial.println(n_soldadura);
                     Serial.println("Tensão2");
                    Serial.println(media_final_v2*(28.4/1024.0));
                     Serial.println(ano);
                     Serial.println(mes);
                     Serial.println(dia);
                     Serial.println(hora);
                     Serial.println(minuto);
                     Serial.println(segundo);
                    //digitalWriteFast(45, HIGH);
                }
                else{
                    //Serial.println("Numero de soldaduras");
                    //Serial.println(n_soldadura);
                    Serial.println("media tensao final");
                    Serial.println(media_final_v*(28.4 / 1024.0));
                    Serial.println("Pressao");
                    Serial.println(media_final_i);
                    //Serial.println("media tensao ultimo ciclo");
                    //Serial.println(v_lastcycle*(28.4/1024.0));
                    Serial.println(ano);
                    Serial.println(mes);
                    Serial.println(dia);
                    Serial.println(hora);
                    Serial.println(minuto);
                    Serial.println(segundo);
                }
}
```

```
void update_time(){
                        DateTime now = rtc.now();
                        ano = now.year();
                        mes = now.month();
                        dia = now.day();
                        hora = now.hour();
                        minuto = now.minute();
                        segundo = now.second();
}
void D37_HIGH(){
PORTC |= (1<<PC0);//high luz verde
}
void D31_HIGH(){
PORTC |= (1<<PC6);//high luz vermelha
}
void D37_LOW(){
PORTC &= ~(1<<PC0);//low
}
void D31_LOW(){
PORTC &= ~(1<<PC6);//low
}
ISR(INT4_vect){//interrup??o externa para detetar mudan?a de flanco
        flag0 = true;
        //Serial.println("programa 1");
}
ISR(INT5_vect){
        //Serial.println("programa 2");
        flag0 = true;
}
////////////////////////////////////////////////////////////////////////////
ISR(TIMER3_COMPA_vect){//interrupcao de tempo para leitura de sensores
        if (flag0 == true && flag_sd == false && flag_transmite == false){// se o botao de
soldadura for pressionado
                wtd = true;
```

```
            D31_HIGH();//apagar leds
            D37_HIGH();
                if (count>3 && wtd == true){//watchdog de 3 segundos se o botão for
pressionado e nao houver tensão
                        flag0 = false;
                        wtd = false;
            D31_HIGH();
            D37_HIGH();
                        count = 0;
                }
                //Serial.println("Disparou");
                save_amostra();
                if (fases == 5){//terminou a soldadura
                //      Serial.println("terminou soldadura");
                        n_soldadura++;
                        //v_lastcycle=media_ant_v;
                        media_ant_v = 0;
                        media_ant_i = 0;
                        contador_s = 0;
                        fases = 0;
                        if (flag_transmite == false){
                                cria_frame();
                                flag0 = false;
                        //      Serial.println("vai enviar");
                        }
                        flag_transmite = true;
              //  Serial.print("flag transmite ativada");
                        ////////////////////flag0=false;
                }
        }
        else{
                //digitalWriteFast(45,LOW);
        }
}
```

```
//////////////////////////////
ISR(TIMER1_COMPA_vect){  //timer 1 para watchdog do botao de soldadura 1 segundo
        if (flag0 == true){
                count++;
        }
        else{
                count = 0;
        }
}
void save_amostra(){
        //Serial.println("Entrou na funcao");
        static long aux_v, aux_i;
        aux_v = analogRead(A0);//leitura de tensao
        aux_i = analogRead(A1);//leitura de pressao
//Serial.println(aux_v);
//Serial.println(aux_i);
        ss = ~ss;
        xpto_v += (long)pow(aux_v, 2);//somatório das leituras de tensão
        xpto_i += (long)pow(aux_i, 2);//somatorio das leituras de pressao
        ind_buf++;//incrementa index
        if (ind_buf >= 40){    //apos 40 amostras guarda a media
                ind_buf = 0;
//Serial.println("vai calcular media");
                calcula_media();
        }
}
void calcula_media(){
//  Serial.println("calcula media");
        //digitalWriteFast(8,HIGH);
        media_actu_v = sqrt(xpto_v / TAMANHO);//guarda a media de tensao;
        media_actu_i = sqrt(xpto_i / TAMANHO);//guarda a media de pressao;
        xpto_v = 0;//limpa buffer do ciclo
        xpto_i = 0;//limpa buffer do ciclo
//Serial.println(media_actu_v);
```

```
        flag_weld_ant = flag_weld_on;
        if (media_actu_v<MIN){// se o valor RMS for muito baixo, desliga-se a
"soldadura"
                //Serial.println("Valor rms muito baixo");
                flag_weld_on = false;
                if (contador_s<COOL){//n□o incrementa mais ciclos
                        contador_s++;
                }
                if (fases == 3 && contador_s == COOL){
                        wtd = false;
                        fases = 5;
                }
        }
        else{
                wtd = false;
                flag_weld_on = true;
        }
        if (flag_weld_on == true){
//  Serial.println("fases");
                switch (fases) {
                case 0:
                        contador_s = 0;
                        first_v = media_ant_v;//new
                        first_i = media_ant_i;//new
                        fases = 1;
                        weld = 0;
                        slope = 1;
                        //Serial.println("case0");
                        break;
                case 1:
                        if (media_actu_v - media_ant_v <= SLOPE){    //acabou a fase slope
                                declive_v = (media_ant_v - first_v) / (slope + 1);
                                declive_i = (media_ant_i - first_i) / (slope + 1);
                                slope++;
```

```
                        media_weld_v = media_actu_v;
                        media_weld_i = media_actu_i;
                        fases = 2;
                        pico_v = media_ant_v;
                        pico_i = media_ant_i;
                           //      Serial.println("case1");
                    }
                    else{//incrementa o numero de ciclos da fase slope
                        slope++;
                    }
                    break;
                case 2:
                    media_weld_v += media_actu_v;
                    media_weld_i += media_actu_i;
                    weld++;
//Serial.println("case2");
                    break;
                case 3:
                    if (contador_s<COOL){
                         // Serial.println("case3");
                        fases = 4;
                        weld2 = 1;
                        cool = contador_s;
                        media_weld_v2 = media_actu_v;
                        media_weld_i2 = media_actu_i;
                    }
                    break;
                case 4:
//Serial.println("case4");
                    media_weld_v2 += media_actu_v;
                    media_weld_i2 += media_actu_i;
                    weld2++;
                    break;
                default:;
```

```
                }
        }
        else{
                switch (fases) {
                case 2:
                        fases = 3;
                        weld2 = 0;
                        cool = 0;
                        media_weld_v2 = 0;
                        media_weld_i2 = 0;
                        if (weld>0){
                                media_final_v = media_weld_v / (weld + 1);
                                media_final_i = media_weld_i / (weld + 1);
                                //v_lastcycle=media_ant_v;
                                //p_lastcycle=media_actu_i;
  //  Serial.println("A");
                        }
                        break;
                case 4:
                        fases = 5;
                        if (weld2>0){//calcula a 2 soldadura
//Serial.println("B");
                                media_final_v2 = media_weld_v2 / (weld2);
                                media_final_i2 = media_weld_i2 / (weld2);
                        }
                        break;
                default:;
                }
        }
        media_ant_v = media_actu_v;
        media_ant_i = media_actu_i;
}
#define error(msg) error_P(PSTR(msg))
void error_P(const char* msg) {
```

```
        sd.errorHalt_P(msg);
}
void writesd(){
if (!file.open("test.txt", O_RDWR | O_CREAT | O_AT_END)) {
 sd.errorHalt("opening test.txt for write failed");
 }
        for (uint8_t i = 0; i < sdpackage_size; i++) {
                if (i == 0){//caracter A
                        file.write(sdpackage[i]);
                        file.write("#");
                }
                if (i == (sdpackage_size - 1)){//caracter B
                        file.write(sdpackage[i]);
                }
                else{
                        file.print(sdpackage[i], DEC);
                        file.write("#");
                }
        }
        file.println();
     file.close();
    // Serial.println("Terminou escrita");
}
void cria_frame(){
        payload[0] = START_FRAME;//A
        payload[1] = ID_MAQUINA;//id maquina (existem 2 por transformador)
        payload[2] = (int)((n_soldadura >> 24) & 0xFF);
        payload[3] = (int)((n_soldadura >> 16) & 0xFF);
        payload[4] = (int)((n_soldadura >> 8) & 0XFF);
        payload[5] = (int)((n_soldadura & 0XFF));
        payload[6] = (int)((ano >> 8) & 0XFF);
        payload[7] = (int)(ano & 0XFF);
        payload[8] = (int)((mes >> 8) & 0XFF);
        payload[9] = (int)(mes & 0XFF);
```

```
        payload[10] = highByte(slope);
        payload[11] = lowByte(slope);
        payload[12] = highByte(weld);
        payload[13] = lowByte(weld);
        payload[14] = highByte(cool);
        payload[15] = lowByte(cool);
        payload[16] = highByte(weld2);
        payload[17] = lowByte(weld2);
        payload[18] = highByte(declive_v);
        payload[19] = lowByte(declive_v);
        payload[20] = highByte(declive_i);
        payload[21] = lowByte(declive_i);
        payload[22] = highByte(pico_v);
        payload[23] = lowByte(pico_v);
        payload[24] = highByte(pico_i);
        payload[25] = lowByte(pico_i);
        payload[26] = highByte(media_final_v);
        payload[27] = lowByte(media_final_v);
        payload[28] = highByte(media_final_i);
        payload[29] = lowByte(media_final_i);
        payload[30] = highByte(media_final_v2);
        payload[31] = lowByte(media_final_v2);
        payload[32] = highByte(media_final_i2);
        payload[33] = lowByte(media_final_i2);
        payload[34] = (int)((dia >> 8) & 0XFF);
        payload[35] = (int)(dia & 0XFF);
        payload[36] = (int)((hora >> 8) & 0XFF);
        payload[37] = (int)(hora & 0XFF);
        payload[38] = (int)((minuto >> 8) & 0XFF);
        payload[39] = (int)(minuto & 0XFF);
        payload[40] = (int)((segundo >> 8) & 0XFF);
        payload[41] = (int)(segundo & 0XFF);
        payload[42] = erro;
}
```

```
//cria pacote para ser transmitido para o sd
void cria_frame_sd(){
	sdpackage[0] = START_FRAME;//A
	sdpackage[1] = ID_MAQUINA;//id maquina (existem 2 por transformador)
	sdpackage[2] = n_soldadura;
	sdpackage[3] = ano;
	sdpackage[4] = mes;
	sdpackage[5] = dia;
	sdpackage[6] = hora;
	sdpackage[7] = minuto;
	sdpackage[8] = segundo;
	sdpackage[9] = slope;
	sdpackage[10] = weld;
	sdpackage[11] = cool;
	sdpackage[12] = weld2;
	sdpackage[13] = declive_v;
	sdpackage[14] = declive_i;
	sdpackage[15] = pico_v;
	sdpackage[16] = pico_i;
	sdpackage[17] = media_final_v;
	sdpackage[18] = media_final_i;
	sdpackage[19] = media_final_v2;
	sdpackage[20] = media_final_i2;
	sdpackage[21] = erro;
	sdpackage[22] = 'B';
}
```

Anexo 3:

Programa do módulo "slave" do primário implementado - slave_P_P1_V3.ino:

```
#include <avr/interrupt.h>
#include <Wire.h>
#define DEBUG //para debug
#define TAMANHO 40 //comprimento dos arrays para amostras
#define TAMANHO_M 100 //comprimento dos arrays para medias
//#define MIN 150  //valor minimo para se considerar desligado
#define MIN 30
#define SLOPE 20//diferenca de tens□o media minimo para considerar que est□ na fase slope
#define COOL 16//numero de ciclos maximo para ainda considerar fase cool
#define ID_MAQUINA 1 //Identifica□□o da maquina
#define SIZE_FRAME 25//26 //tamanho do pacote de transmiss□o dos slaves
#define BAUDRATE_PRI 38400 //velocidade de transmiss□o com o master
#define BAUDRATE_WI 115200//velocidade de transmiss□o com a porta serie para debug
/****************Carateres especiais para rece□□o de dados do slave prim□rio**********************/
#define ESCAPE_FRAME 47 //  '/'
#define START_FRAME 65 //  'A'
//buffer que guardam as amostragens de um ciclo
long buf_v[TAMANHO];
//buffers que guardam as medias por ciclo
int buf_media_v[TAMANHO_M];
boolean flag_weld_on = false;//indica a fase soldadura
boolean flag_weld_ant = false;//guarda o estado anterior de "flag_weld_on"
boolean flag_slope_on = false;//indica a fase "slope"
boolean flag_transmite = false; //indica que a soldadura acabou
volatile boolean flag0;
int ano, mes, dia, hora, minuto, segundo;
int ind_buf = 0;//index dos buffers "buf_v,buf_i"
int media_ant = 0;//variavel auxiliar para guardar a media do ciclo anterior
```

```
int slope = 0;//numero de ciclos da fase slope
int weld = 0;//numero de ciclos da fase weld
int weld2 = 0;//numero de ciclos da fase weld2
int contador_s = 0;//contador de ciclos da fase COOL
int cool = 0;
byte erro = 0;
int first_v = 0;//variavel auxiliar para guardar o primeiro valor
int pico_v = 0; //tensao no ultimo ciclo da fase slope
int media_final_v = 0;//tensao media durante a fase weld
int media_final_v2 = 0;//tensao media durante a fase weld
int declive_v = 0;//declive de tens�o durate a fase slope
unsigned long n_soldadura = 0;//contador de soldaduras efetuadas desde o arranque do arduino
//int cont_amostr=0;//contador de numero amostras num ciclo
byte buff_trans[SIZE_FRAME + 1];
unsigned long xpto = 0;
//fun��o que guarda as novas amostras
long xpto_v = 0;
long media_slope_v = 0;
long media_weld_v = 0;
long media_weld_v2 = 0;
long media_ant_v = 0;
long media_actu_v = 0;
volatile int count = 0;
volatile int programa = 0;
void save_amostra(){
        static long aux_v;
        aux_v = analogRead(A0);//corrente
        xpto_v += (long)pow(aux_v, 2);
        ind_buf++;//incrementa index
        if (ind_buf >= 40){    //apos 40 amostras guarda a media
                ind_buf = 0;
                calcula_media();
        }
```

```
}
//SLOPE
int fases = 0; //0=em espera1, 1=slope soldadura1,2=soldadura1, 3=espera2, , 4=soldadura2, 5=fim soldadura
//fun��o que guarda as medias do ciclo atual
volatile boolean wtd = false;
void calcula_media(){
	media_actu_v = sqrt(xpto_v / TAMANHO);//guarda a media de tens�o;
	xpto_v = 0;//limpa buffer do ciclo
	flag_weld_ant = flag_weld_on;
	if (media_actu_v<MIN){// se o valor RMS for muito baixo, desliga-se a "soldadura"
		//Serial.println("Valor rms muito baixo");
		flag_weld_on = false;
		if (contador_s<COOL){//n�o incrementa mais ciclos
			contador_s++;
		}
		if (fases == 3 && contador_s == COOL){
			wtd = false;
			fases = 5;
		}
	}
	else{
		wtd = false;
		flag_weld_on = true;
	}
	if (flag_weld_on == true){
		switch (fases) {
		case 0:
			contador_s = 0;
			first_v = media_ant_v;//new
			fases = 1;
			weld = 0;
			slope = 1;
```

```
            //Serial.println("Slope");
            break;
        case 1:
            if (media_actu_v - media_ant_v <= SLOPE){    //acabou a fase slope
                declive_v = (media_ant_v - first_v) / (slope + 1);
                slope++;
                media_weld_v = media_actu_v;
                fases = 2;
                pico_v = media_ant_v;
                //          Serial.println("FASE2");
            }
            else{//incrementa o numero de ciclos da fase slope
                slope++;
            }
            break;
        case 2:
            //Serial.println(slope);
            media_weld_v += media_actu_v;
            weld++;
            break;
        case 3:
            if (contador_s<COOL){
                //   Serial.println("FASE4");
                fases = 4;
                weld2 = 1;
                cool = contador_s;
                media_weld_v2 = media_actu_v;
            }
            break;
        case 4:
            media_weld_v2 += media_actu_v;
            weld2++;
            break;
        default:;
```

```
                }
        }
        else{
                switch (fases) {//acabou soldadura-separar de soldadura com pre soldadura
                case 2:
                        fases = 3;
                        weld2 = 0;
                        cool = 0;
                        media_weld_v2 = 0;
                        if (weld>0){
                                media_final_v = media_weld_v / (weld + 1);
                        }
                        break;
                case 4:
                        fases = 5;
                        if (weld2>0){//calcula a 2 soldadura
                                media_final_v2 = media_weld_v2 / (weld2);
                        }
                        break;
                default:;
                }
        }
        media_ant_v = media_actu_v;
}
////////////////////////////////////////////////////////////
ISR(INT4_vect){//interrupcao externa para detetar mudan?a de flanco
        flag0 = true;
        //Serial.println("programa1");
        programa = 1;
}
ISR(INT5_vect){//interrupcao externa para detetar mudan?a de flanco
        flag0 = true;
        //Serial.println("programa2");
        programa = 2;
```

```
}
//////////////////////////////////////////////////////////////
ISR(TIMER3_COMPA_vect){//interrup��o de tempo para leitura de sensores
        if (flag0 == true && flag_transmite == false){// se o botao de soldadura for
pressionado
                wtd = true;
                if (count>3 && wtd == true){
                        flag0 = false;
                        wtd = false;
                        count = 0;
                }
                save_amostra();
                if (fases == 5){//terminou a soldadura
                        //Serial.println("Fase5");
                        n_soldadura++;
                        media_ant_v = 0;
                        contador_s = 0;
                        fases = 0;
                        if (flag_transmite == false){
                                cria_frame();
                                flag0 = false;
                                //Serial.println("FIM");
                        }
                        flag_transmite = true;
                }
        }
        else{
                //digitalWriteFast(45,LOW);
        }
}
ISR(TIMER1_COMPA_vect){  //timer 1 para watchdog do botao de soldadura 1 segundo
        if (flag0 == true){
                count++;
        }
```

```
        else{
                count = 0;
        }
}
void setup() {
        Serial.begin(9600);
        Serial3.begin(BAUDRATE_PRI);
        cli();//desliga interrup��es
        /////Timer3////////////// interrup�ao de 2 em 2 ms
        TCCR3A = 0;//reset ao timer //registos com valores de configura��es
        TCCR3B = 0;//reset ao timer //registos com valores de configura��es
        OCR3A = 124;//valor para overflow do timer
        TCCR3B |= (1 << WGM12);//liga modo ctc
        TCCR3B |= (1 << CS31) | (1 << CS30);//prescaler de 64
        //TCCR3B |= (1 << CS32);//prescaler de 256
        TIMSK3 |= (1 << OCIE3A);//liga o timer para modo de compara�ao de
interrup�ao
        ////////////////////////////////////////////////////
        ////////////////////////////////////////////interrupcao para watchdog do botao de soldadura
        //set timer1 interrupt at 1Hz
        TCCR1A = 0;// set entire TCCR1A register to 0
        TCCR1B = 0;// same for TCCR1B
        TCNT1 = 0;//initialize counter value to 0
        // set compare match register for 1hz increments
        OCR1A = 15624;// = (16*10^6) / (1*1024) - 1 (must be <65536)
        // turn on CTC mode
        TCCR1B |= (1 << WGM12);
        // Set CS10 and CS12 bits for 1024 prescaler
        TCCR1B |= (1 << CS12) | (1 << CS10);
        // enable timer compare interrupt
        TIMSK1 |= (1 << OCIE1A);
        EIMSK = 0b00000000;
        ////////////////////////////////int4 e int5 bits diretos controlo por botao pino D2(int4) e
D3(int5)
```

```
	EICRB = 0b00001111;//ascendente para int 4 e 5
	EIMSK = 0b00110000;//int 4 e 5
	sei();//liga interrup��es
	////////////////////////////////////
	delay(2000);
}
void loop() {
	if (flag_transmite == true){//flag de finalização de soldadura
		//debug_value();//para debug
	    send_data();//envia dados
	}
}
void debug_value(){
		Serial.println("Media final");
		Serial.println(media_final_v);
}
void send_data(){//enviar dados para master
		Serial3.write('A');//send data
		//Serial.println('A');
		for (int i = 1; i<SIZE_FRAME; i++){
			if ((buff_trans[i] == START_FRAME || buff_trans[i] ==
ESCAPE_FRAME)){//carater especial, tem de ser "escapado"
				buff_trans[i] = buff_trans[i] ^ 0x20;
			}
			Serial3.write(buff_trans[i]);
			//Serial.println(buff_trans[i]);
		}
		//Serial.println("enviou");
		flag_transmite = false;
}
//cria o pacote para ser transmitido para o master
void cria_frame(){
	buff_trans[0] = START_FRAME;//A
	buff_trans[1] = ID_MAQUINA;//id maquina (existem 2 por transformador)
```

```
        buff_trans[2] = (int)((n_soldadura >> 24) & 0xFF);
        buff_trans[3] = (int)((n_soldadura >> 16) & 0xFF);
        buff_trans[4] = (int)((n_soldadura >> 8) & 0XFF);
        buff_trans[5] = (int)((n_soldadura & 0XFF));
        buff_trans[6] = highByte(slope);
        buff_trans[7] = lowByte(slope);
        buff_trans[8] = highByte(weld);
        buff_trans[9] = lowByte(weld);
        buff_trans[10] = highByte(cool);
        buff_trans[11] = lowByte(cool);
        buff_trans[12] = highByte(weld2);
        buff_trans[13] = lowByte(weld2);
        buff_trans[14] = highByte(declive_v);
        buff_trans[15] = lowByte(declive_v);
        buff_trans[16] = highByte(pico_v);
        buff_trans[17] = lowByte(pico_v);
        buff_trans[18] = highByte(media_final_v);
        buff_trans[19] = lowByte(media_final_v);
        buff_trans[20] = highByte(programa);
        buff_trans[21] = lowByte(programa);
        buff_trans[22] = highByte(media_final_v2);
        buff_trans[23] = lowByte(media_final_v2);
        buff_trans[24] = erro;
}
```

Anexo 4:

Programa do módulo "master" implementado - Master_P1_v3.ino:

```
#include <XBee.h>
#include <RedFlyClient.h>
#include <SdFat.h>
#define ID_TRANSFORMADOR "2000" //Identifica��o do transformador
#define ID_MAQUINA1 1//Identifica��o da maquina 1
#define ID_MAQUINA2 2//Identifica��o da maquina 2
#define SIZE_FRAME 43//26 //tamanho do pacote de transmiss�o dos slaves
#define SIZE_FRAMEP 25
//#define NETWORK_WI "TP-LINK_POCKET_3020_A00F8E"  //dd-wrt  //NETLINE2 //nome da rede WIFI
#define NETWORK_WI "LINE"  //dd-wrt  //NETLINE2 //nome da rede WIFI
#define PASS_NET_WI ""  //Password da rede WIFI
#define BAUDRATE_WI 115200 //velocidade de transmiss�o com o PC
#define BAUDRATE_XBEE 57600//9600 //velocidade de transmiss�o com o slave secund�rio
#define BAUDRATE_PRI 38400 //velocidade de transmiss�o com o slave prim�rio
#define PORTA_PC 50002  //numero da porta do PC
/****************Carateres especiais para rece��o de dados do slave prim�rio************************/
#define ESCAPE_FRAME 47      //  '/'
#define START_FRAME 65 //  'A'
//Configura��o do Xbee
XBee xbee = XBee();
XBeeResponse response = XBeeResponse();
Rx16Response rx16 = Rx16Response();
uint8_t payload[SIZE_FRAME] = {
        0 };
Tx16Request tx = Tx16Request(0x1244, payload, sizeof(payload));    //configura endere�o Xbee de destino e tamanho do pacote
TxStatusResponse txStatus = TxStatusResponse();
//////////////////////////////////////////////////
XBee xbee2 = XBee();
```

```
XBeeResponse response2 = XBeeResponse();
Rx16Response rx162 = Rx16Response();
uint8_t payload2[2] = {
        0 };
Tx16Request tx2 = Tx16Request(0x1244, payload2, sizeof(payload2));    //configura endere
o Xbee de destino e tamanho do pacote
TxStatusResponse txStatus2 = TxStatusResponse();
////////////////////////////////////////
/*********************************Defini
o do cliente************************/
/*********************************Dados de rede*********************************/
byte ip[] = {
        192, 168, 0, 103 }; //IP do shield (cliente)
byte gateway[] = {
        192, 168, 0, 100 }; //ip from gateway/router
byte netmask[] = {
        255, 255, 255, 0 }; //netmask
byte dnsserver[] = {
        192, 168, 0, 100 }; //ip from dns server
byte server[] = {
        192, 168, 0, 101 }; //IP do servidor ao qual se vai ligar
/*********************************Defini
o do cliente************************/
RedFlyClient client(server, 50002); //Indica
o do servidor e da porta  qual se ir
ligar
/****************Fun
es para envio dos dados por porta s
rie******************/
void debugout(char *s)  {
        RedFly.disable();
        Serial.print(s);
        RedFly.enable();
}
void debugoutln(char *s){
        RedFly.disable();
```

```
        Serial.println(s);
        RedFly.enable();
}
int count = 0; //guarda o index do pacote "frame" do prim□rio
byte frame[SIZE_FRAME + 1]; //guarda o pacote vindo do slave do prim□rio
boolean flag_completa = false;
boolean flag_esc = false;
//fun□□o que junta os 2 bytes num unico numero
int inteiro(byte high_b, byte low_b){
        return  ((int)high_b << 8) + low_b;
}
uint8_t ret;//variavel auxiliar para o teste da comunica□□o com o PC
//strings para optimizar a velocidade de transmiss□o com o PC
char pacote_P[10] = { "A#"ID_TRANSFORMADOR"#P#" };
char pacote_S[10] = { "A#"ID_TRANSFORMADOR"#S#" };
void setup() {
        //pinMode(41,OUTPUT);
        //digitalWriteFast(41,LOW);
        Serial.begin(115200);
        Serial1.begin(BAUDRATE_XBEE);
        Serial3.begin(BAUDRATE_PRI);
        xbee.setSerial(Serial1);
        xbee2.setSerial(Serial1);
        delay(1000); //delay de 1 segundo para dar tempo de inicializar as portas de
comunica□□o
        //inicializa□ao wireless////////////////////////////////////////////////////////////////////
        ret = RedFly.init(BAUDRATE_WI, HIGH_POWER); //LOW_POWER
MED_POWER HIGH_POWER
        if (ret)
        {
                debugoutln("INIT ERR");
        }
        else
        {
```

```
            RedFly.scan();
            ret = RedFly.join(NETWORK_WI, PASS_NET_WI,
INFRASTRUCTURE);//join network
            if (ret)
            {
                  debugoutln("JOIN ERR");
            }
            else
            {
                  ret = RedFly.begin(ip, dnsserver, gateway, netmask);//set ip config
                  if (ret)
                  {
                        debugoutln("BEGIN ERR");
                        RedFly.disconnect();
                  }
                  else
                  {
                        if (client.connect(server, PORTA_PC))//liga-se ao servidor
                        {
                              ;
                        }
                  }
            }
      }
      debugoutln("FIM SETUP");
      sei();
}
//flag para indica��o de recep��o de dados novos
boolean new_dados_P = false;
boolean new_dados_S = false;
boolean new_dados_S2 = false;
boolean flag_ok = false;
unsigned int slope = 0;
unsigned int weld = 0;
```

```
unsigned int cool = 0;
unsigned int weld2 = 0;
float media_v = 0;
float media_v2 = 0;
float media_p = 0;
float media_p2 = 0;
//////////////////////parametros para comparar
unsigned int p_slope = 0;
unsigned int p_weld = 0;
unsigned int p_cool = 0;
unsigned int p_weld2 = 0;
unsigned long p_media_v = 0;
unsigned long p_media_v2 = 0;
unsigned long p_media_p = 0;
unsigned long p_media_p2 = 0;
boolean checked=false;
unsigned long n_soldadura_master=0;
//////////////////////
/*********************************Buffer*********************************
*****/
char buf[466];
void loop() {
        while (1){
                //obufstream bout(buf, sizeof(buf));
/////////////////////////leitura de dados slave secund�rio////////////////////////////
slave_s_receive();
/////////////////////////////////////////////////////////////////////leitura de dados do slave do primario
slave_p_receive();
/////////////////////////construção do pacote para scada////////////////////////////
if (flag_completa == true && new_dados_S == true){
  n_soldadura_master++;
wifi_package_create();
}
/////////////////////////////////////////////////////////////////////envio de ok /not ok da sodadura
```

```
if (flag_ok){
checked=false;
weld_check();//verifica se soldadura é ok/notok
if(checked==true){
ok_send();//envia ok/not ok
}
}
//////////////////////////transmiss�o de dados para PC//////////////////////////////
if (new_dados_P == true && new_dados_S2 == true /*  new_dados_S==true*/){  //se cumpriu todas as etapas
server_send_data();//envia dados para servidor
}
                delay(50);
        }
}
void server_send_data(){
                        /*******************Medi��o da qualidade do sinal da rede wireless*************************/
                        if (RedFly.getrssi()>80)
                        {
                                debugoutln("rssi>80");
                        }
                        else{
                                /************************Testa o estado da liga��o com o servidor***************************/
                                if ((!client.connected() || client.status()))//Se a liga��o estiver desligada reinicia o processo de conex�o
                                {
                                        debugoutln("DISCONNECTED");
                                        client.stop();
                                        RedFly.disconnect();
                                        ret = RedFly.init(BAUDRATE_WI, HIGH_POWER);
                                        if (!ret)
                                        {
                                                RedFly.scan();
```

```
                                        ret = RedFly.join(NETWORK_WI,
PASS_NET_WI, INFRASTRUCTURE);
                                        if (!ret)
                                        {
                                                ret = RedFly.begin(ip, dnsserver,
gateway, netmask);
                                                if (!ret)
                                                {
                                                        client.connect(server,
PORTA_PC);
                                                        if (client.connected()){

        debugoutln("CONNECTED");
                                                        }
                                                }
                                                else{
                                                        debugoutln("BEGIN ERR");
                                                }
                                        }
                                        else{
                                                debugoutln("JOIN ERR");
                                        }
                                }
                                else{
                                        debugoutln("INIT ERR");
                                }
                        }
                        /**********************Envio dos dados para o
servidor*************************/
                        if (client.connected()){//se existir conec��o com servidor
do PC
                                //digitalWriteFast(46,HIGH); //indica��o de
transmiss�o para o PC
                                client.write(buf);//Envio do dados lidos do ficheiro
                                if (new_dados_P == true){//limpa a flag
                                        new_dados_P = false;
```

```
                                        }
                                        if (new_dados_S == true){//limpa a flag
                                                new_dados_S = false;
                                        }
                                        if (new_dados_S2 == true){//limpa a flag
                                                new_dados_S2 = false;
                                        }
                                }
                        }
}
void ok_send(){
                        xbee2.send(tx2);//envio do ok/not ok
                        xbee2.readPacket(500);
                        if (xbee2.getResponse().isAvailable()) {///receção de ack
                                if (xbee2.getResponse().getApiId() ==
TX_STATUS_RESPONSE) {

        xbee2.getResponse().getZBTxStatusResponse(txStatus2);
                                        if (txStatus2.getStatus() == SUCCESS) {//reccebeu
confirma��o de pacote recebido
                                                Serial.println("Ok");
                                                // digitalWriteFast(37, HIGH);
                                        checked=false;
                                                flag_ok = false;
                                                new_dados_S2 = true;
                                                //status_receive=true;
                                                //delay(50);
                                        }
                                        if (!(txStatus2.getStatus() == SUCCESS)) {
                                                Serial.println("NOT SUCCESS");
                                                //flag_sd=true;
                                        }
                                }
                        }
                        else if (xbee2.getResponse().isError()) {
```

```
                        Serial.println("ERROR");
                        //flag_sd=true;
                }
                else{
                        Serial.println("TIMEOUT");
                        //xbee.send(tx);
              checked=false;
                        flag_ok = false;
                        new_dados_S2 = true;
                }
}
void weld_check(){
                if (slope == p_slope && weld == p_weld && cool == p_cool &&
weld2 == p_weld2 && media_p == p_media_p){
                        payload2[0] = (int)10;
checked=true;
                }
                else
                {
                        payload2[0] = (int)20;
checked=true;
                }
}
void slave_p_receive(){
        static byte aux;
        while (Serial3.available()>0){
                //Serial.println("comunicou");
                aux = Serial3.read();
                if (flag_completa == true){//regeita novos dados at? que o pacote
atual seja processado
                        continue;
                }
                if (aux == START_FRAME){//se ? o valor de "inicio"
                        count = 0;
```

```
                        }
                        if (aux != START_FRAME && count == 0){//se o pacote est�
n�o est� sincronizado
                                count = 0;
                                continue;
                        }
                        if (aux == ESCAPE_FRAME){//se � o valor "escape"
                                flag_esc = true;
                                continue;
                        }
                        if (flag_esc == true){//converte o byte "escapado"
                                flag_esc = false;
                                aux = aux ^ 0x20;
                        }
                        frame[count] = aux;//grava o byte
                        count++;
                        if (count >= SIZE_FRAMEP){//pacote completo
                                count = 0;
                                flag_completa = true;
                                Serial.println("recebeu primario");
                                continue;
                        }
                }
}
void slave_s_receive(){
                xbee.readPacket();
                if (xbee.getResponse().isAvailable()) {
                        if (xbee.getResponse().getApiId() == RX_16_RESPONSE) {
                                //digitalWriteFast(41,HIGH);
                                if (new_dados_S == false /* && new_dados_P==false &&
new_dados_S2==false */){//rejeito os novos pacotes at� o atual ser processado
                                        //digitalWriteFast(41,HIGH);
                                        xbee.getResponse().getRx16Response(rx16);
                                        //cria o pacote de dados a ser enviado para o PC
```

```
                                        Serial.println("Recebeu do secundario");
                                        // Serial.println(media_p*0.009765625,DEC);
                                        new_dados_S = true;
                                        //flag_ok=true;//flag para envio do ok not ok da
soldadura para o slave do secundario
                                }
                        }
                }
}
void wifi_package_create(){
obufstream bout(buf, sizeof(buf));
                        //////////slave secundario////////////////
                        bout << pacote_S << (int)rx16.getData(1) << "#";
                        bout << (((unsigned long)rx16.getData(2) << 24)//numero de
soldadura
                                + ((unsigned long)rx16.getData(3) << 16)
                                + ((unsigned long)rx16.getData(4) << 8)
                                + ((unsigned long)rx16.getData(5))) << "#";
                        bout << (((unsigned int)rx16.getData(6) << 8)//ano
                                + ((unsigned int)rx16.getData(7))) << "#";
                        bout << (((unsigned int)rx16.getData(8) << 8)//mes
                                + ((unsigned int)rx16.getData(9))) << "#";
                        slope = inteiro(rx16.getData(10), rx16.getData(11));//ciclos de slope
                        bout << slope << "#";
                        weld = inteiro(rx16.getData(12), rx16.getData(13));//ciclos de weld
                        bout << weld << "#";
                        cool = inteiro(rx16.getData(14), rx16.getData(15));//ciclos de cool
                        bout << cool << "#";
                        weld2 = inteiro(rx16.getData(16), rx16.getData(17));//ciclos de weld
2
                        bout << weld2 << "#";//weld2
                        bout << ((unsigned long)inteiro(rx16.getData(18),
rx16.getData(19))*(28.4 / 1024)) << "#";//declive de tensão (10/901=0.009765625)
                        bout << ((unsigned long)inteiro(rx16.getData(20),
rx16.getData(21))*0.009765625) << "#";//declive de pressão
```

```
                    bout << ((unsigned long)inteiro(rx16.getData(22),
rx16.getData(23))*(28.4 / 1024)) << "#";//pico tensão
                    bout << ((unsigned long)inteiro(rx16.getData(24),
rx16.getData(25))*0.009765625) << "#";//pico de pressão
                    media_v = ((float)inteiro(rx16.getData(26), rx16.getData(27)))*(28.4
/ 1024.0);
                    bout << ((unsigned long)inteiro(rx16.getData(26),
rx16.getData(27)))*(28.4 / 1024.0) << "#";//media tensão
                    media_p = (float)inteiro(rx16.getData(28),
rx16.getData(29))/*(10/901)*/;
                    bout << ((unsigned long)inteiro(rx16.getData(28),
rx16.getData(29))*0.009765625) << "#";//10-valor maximo de pressao em bar
                    media_v2 = ((float)inteiro(rx16.getData(30),
rx16.getData(31)))*(28.4 / 1024.0);       //901- sensor de pressao da sinal maximo de 4.4v
                    bout << ((unsigned long)inteiro(rx16.getData(30),
rx16.getData(31)))*(28.4 / 1024.0) << "#";//media v2
                    media_p2 = ((float)inteiro(rx16.getData(32),
rx16.getData(33)))*0.009765625;
                    bout << (unsigned long)inteiro(rx16.getData(32),
rx16.getData(33))*(10 / 901) << "#";//media pressao 2
                    bout << inteiro(rx16.getData(34), rx16.getData(35)) << "#";//dia
                    bout << inteiro(rx16.getData(36), rx16.getData(37)) << "#";//hora
                    bout << inteiro(rx16.getData(38), rx16.getData(39)) << "#";//minuto
                    bout << inteiro(rx16.getData(40), rx16.getData(41)) <<
"#";//segundo
                    bout << (int)rx16.getData(SIZE_FRAME - 1) << "#"; //"#B#";//erro
pacote 23
                    /////////slave primario///////////////
                /*      bout << (((unsigned long)frame[2] << 24)//numero de soldadura
slave primario
                                + ((unsigned long)frame[3] << 16)
                                + ((unsigned long)frame[4] << 8)
                                + ((unsigned long)frame[5])) << "#"; */
            bout << n_soldadura_master << "#";
                    bout << inteiro(frame[6], frame[7]) << "#";//slope corrente
                    bout << inteiro(frame[8], frame[9]) << "#";//weld corrente
                    bout << inteiro(frame[10], frame[11]) << "#";//cool corrente
                    bout << inteiro(frame[12], frame[13]) << "#";//weld2 corrente
```

```
                bout << inteiro(frame[14], frame[15]) << "#";//declive corrente
                bout << inteiro(frame[16], frame[17]) << "#";// pico corrente
                bout << inteiro(frame[18], frame[19]) << "#";//media corrente
                bout << inteiro(frame[22], frame[23]) << "#";//media corrente weld2
pacote 32
                bout << inteiro(frame[20], frame[21]) << "#B";//programa
                // bout << (int)frame[SIZE_FRAMEP-1]<<"#B";
                flag_ok = true;//flag para envio do ok not ok da soldadura para o
slave do secundario
                new_dados_P = true;
                flag_completa = false;
}
```

Anexo 5:

Aplicação "SCADA" desenvolvida em Python - scadaa.py:

```
__author__ = 'LINE.IPT'

from PyQt4.uic import loadUiType
from PyQt4 import QtGui
from PyQt4 import QtCore

from matplotlib.figure import Figure
from matplotlib.backends.backend_qt4agg import (
    FigureCanvasQTAgg as FigureCanvas,
    NavigationToolbar2QT as NavigationToolbar)

import socket
import threading
import os
import sys

import matplotlib.patches as mpatches

Ui_MainWindow, QMainWindow = loadUiType('window.ui')


class Main(QMainWindow, Ui_MainWindow):
    def __init__(self, ):
        super(Main, self).__init__()
        self.setupUi(self)
        self.Button1.clicked.connect(self.iniciaar)
        self.Button2.clicked.connect(self.close)
        self.Button3.clicked.connect(self.testar)
        self.fig1 = Figure()
        self.fig1.patch.set_facecolor('white')
        self.setWindowTitle("Scada Mitsubishi")
        palette = QtGui.QPalette()
        palette.setColor(QtGui.QPalette.Background,QtCore.Qt.white)
        self.setPalette(palette)
        self.textbox.textChanged.connect(self.putvalue)
        self.flag=bool
        self.bufferdados=[]

    def testar(self):
        f=[]#criar array
        f2=[0,1,2,3,4]
        asd="5#10#20#30#40"
        print asd
        convertido=asd.split("#")
        print convertido
        print convertido[0]
        print convertido[1]
        print convertido[2]
        print convertido[3]
        valor1=convertido[0]
        valor2=convertido[1]
        valor3=convertido[2]
        valor4=convertido[3]
```

```
        for i in range(5):
            f.append(convertido[i])#mete valores para o array
        print "valor de f"
        print f
        print valor1
        print valor2
        print valor3
        print valor4
        ax1f1 = self.fig1.add_subplot(111)
        ax1f1.hold(False)
        #tem de ser imprimido em lista-linha
        #ax1f1.plot(f,'-o',label='Tensao')
        ax1f1.plot(f2,f,marker='o',label='Tensao')
        ax1f1.set_xlabel('Numero de ciclos')
        ax1f1.set_ylabel('Valor de tensao')
        ax1f1.set_title('Valor medio de tensao')
        #ax1f1.plot(valor3,'o-',label='Tensao')
        ax1f1.legend()
       # red_patch = mpatches.Patch(color='red', label='Tensao')
        #ax1f1.legend(handles=[red_patch])
        self.canvas.draw()
        self.tb1.setText(str(valor1))
        self.tb2.setText(str(valor2))
        self.tb3.setText(str(valor3))
        #escrever para ficheiro de texto
        text_file = open("logger.txt", "a") #a--apend   w--write
        text_file.write("Pacote: %s\n" % asd)
        text_file.close()

    def iniciaar(self):
        threading.Thread(target=self.startserver).start()
        threading.Thread(target=self.startserver2).start()

    def startserver(self):#a correr numa thread a parte
         sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
         server_address = ('', 50002)
         print >>sys.stderr, 'starting up on %s port %s' % server_address
         sock.bind(server_address)
         sock.listen(1)
         while True:
            # espera por ligacao
           print >>sys.stderr, 'waiting for a connection'
           connection, client_address = sock.accept()
           if self.flag==1:
            connection.close()
            break

           try:
             print >>sys.stderr, 'connection from', client_address
            # print >>sys.stderr, 'connection from', client_address2
             while True:
              data = connection.recv(120)
              print >>sys.stderr, 'received "%s"' % data
              #print data
              if data:
               self.textbox.setText('R1')
               dataconverted=data.split('#')
               self.bufferdados[:]=[] # limpa lista ou array
               if dataconverted[0]=='A':
```

```
                    text_file = open("logger.txt", "a") #a--apend   w--
write
                    #text_file.write("Pacote: %s\n" % data)
                    text_file.write(data+"\n")
                    text_file.close()
                    i=0
                    for i in range(len(dataconverted)):
                      self.bufferdados.append(dataconverted[i])#mete
valores para o array
                self.fill()#funcao para desenhar no ecra e escrever para
textboxes
                i=0
            finally:
          # fecha ligacao
              connection.close()

    def startserver2(self):#a correr numa thread a parte
         sock2 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
         server_address2 = ('', 50003)
         print >>sys.stderr, 'starting up on %s port %s' %
server_address2
         sock2.bind(server_address2)
         sock2.listen(1)
         while True:
           print >>sys.stderr, 'waiting for a connection'
           connection2, client_address2 = sock2.accept()
           #connection2, client_address2 = sock.accept()
           if self.flag==1:
            connection2.close()
            break

           try:
             print >>sys.stderr, 'connection from', client_address2
            # print >>sys.stderr, 'connection from', client_address2

             while True:
              data2 = connection2.recv(120)#recebe dados
             # data2 = connection2.recv(120)
              print >>sys.stderr, 'received "%s"' % data2
              #print >>sys.stderr, 'received "%s"' % data2
              #print data
              if data2:
               self.textbox.setText('R2')
                #print >>sys.stderr,
               dataconverted2=data2.split('#')
               self.bufferdados[:]=[] # limpa lista ou array
               if dataconverted2[0]=='A':
                    text_file = open("logger.txt", "a") #a--apend   w--
write
                    #text_file.write("Pacote: %s\n" % data)
                    text_file.write(data2+"\n")
                    text_file.close()
                    i=0
                    for i in range(len(dataconverted2)):
                      self.bufferdados.append(dataconverted2[i])#mete
valores para o array


               self.fill()#funcao para desenhar no ecra e escrever para
```

```
textboxes
                i=0
            finally:
        # fecha ligacao
            connection2.close()

    def fill(self):
        buffergraf=[]
        buffercount=[]
        self.tb1.setText(str(self.bufferdados[5]))#ano
        self.tb2.setText(str(self.bufferdados[6]))#mes
        self.tb3.setText(str(self.bufferdados[19]))#dia
        self.tb4.setText(str(self.bufferdados[20]))#hora
        self.tb5.setText(str(self.bufferdados[21]))#minuto
        self.tb6.setText(str(self.bufferdados[22]))#segundo
        self.tb7.setText(str(self.bufferdados[7])+' ciclos')#slope
        self.tb8.setText(str(self.bufferdados[8])+' ciclos')#weld
        self.tb9.setText(str(self.bufferdados[9])+' ciclos')#cool
        self.tb10.setText(str(self.bufferdados[10])+' ciclos')#weld2
        self.tb11.setText(str(self.bufferdados[15])+' V')#mv
        self.tb12.setText(str(self.bufferdados[16])+' bar')#mp
        self.tb13.setText(str(self.bufferdados[17])+' V')#mv2
        self.tb14.setText(str(self.bufferdados[18])+' bar')#mp2
        self.tb15.setText(str(self.bufferdados[24]))#numero de soldadura
        self.tb16.setText(str(self.bufferdados[1]))#id maquina
        self.tb17.setText(str(self.bufferdados[11]))#declive tensao
        self.tb18.setText(str(self.bufferdados[12]))#declive pressao
        self.tb19.setText(str(self.bufferdados[13])+' V') #pico tensao
        self.tb20.setText(str(self.bufferdados[14])+' bar') #pico pressao

        self.tbsc.setText(str(self.bufferdados[25])+' ciclos') #slope
corrente
        self.tbwc.setText(str(self.bufferdados[26])+' ciclos') #weld
corrente
        self.tbcc.setText(str(self.bufferdados[27])+' ciclos') #cool
corrente
        self.tbw2c.setText(str(self.bufferdados[28])+' ciclos') #weld2
corrente
        self.tbmc.setText(str(self.bufferdados[31])+' A') #media corrente
        self.tbpc.setText(str(self.bufferdados[30])+' A') #pico corrente
        self.tbdc.setText(str(self.bufferdados[29])) #declive corrente
        #self.tbnsc.setText(str(self.bufferdados[24])) #numero de
soldadura
        self.tbmc2.setText(str(self.bufferdados[32])+' A') #media
corrente weld 2
        self.tbprog.setText(str(self.bufferdados[33])) #programa de
sodladura selecionado
        count=0
        for j in range((int(self.bufferdados[7])-1)):
            buffergraf.append(float(self.bufferdados[11])*j) #slope
            count=count + 1
            buffercount.append(count)

        buffergraf.append(self.bufferdados[13]) #pico
        count=count + 1
        buffercount.append(count)

        for k in range(int(self.bufferdados[8])):
            buffergraf.append(self.bufferdados[15]) #weld
```

```
            count=count + 1
            buffercount.append(count)

        for l in range(int(self.bufferdados[9])):
            buffergraf.append(0)                    #cool
            count=count + 1
            buffercount.append(count)

        for z in range(int(self.bufferdados[10])):
            buffergraf.append(self.bufferdados[17])  #weld2
            count=count + 1
            buffercount.append(count)

        buffergraf.append(0)
        count=count + 1
        buffercount.append(count)
        ax1f1 = self.fig1.add_subplot(111)
        ax1f1.hold(False)
        #tem de ser imprimido em lista-linha
        #ax1f1.plot(f,'-o',label='Tensao')
        ax1f1.plot(buffercount,buffergraf,marker='o',label='Tensao')
        ax1f1.set_xlabel('Numero de ciclos')
        ax1f1.set_ylabel('Valor de tensao')
        ax1f1.set_title('Valor medio de tensao')
        #ax1f1.plot(valor3,'o-',label='Tensao')
        ax1f1.legend()
       # red_patch = mpatches.Patch(color='red', label='Tensao')
        #ax1f1.legend(handles=[red_patch])
        self.canvas.draw()
        for v in range(len(buffergraf)):
            buffercount[v] = 0
            buffergraf[v] = 0

        count = 0

    def addmpl(self):
       # fig1 = Figure()
       # fig1.hold(False)
        self.canvas = FigureCanvas(self.fig1)
        self.mplvl.addWidget(self.canvas)
        #self.canvas.draw()
        self.toolbar = NavigationToolbar(self.canvas,
        self.mplwindow, coordinates=True)
        self.mplvl.addWidget(self.toolbar)

    def rmmpl(self,):
        self.mplvl.removeWidget(self.canvas)
        self.canvas.close()
        self.mplvl.removeWidget(self.toolbar)
        self.toolbar.close()

    def plot(self):
        #fig1 = Figure()
        #fig1.hold(False)
        ax1f1 = self.fig1.add_subplot(111)
        ax1f1.hold(False)
        ax1f1.plot(np.random.rand(5),'o-',label='Tensao')
        ax1f1.legend()
       # red_patch = mpatches.Patch(color='red', label='Tensao')
```

```
        #ax1f1.legend(handles=[red_patch])
        self.canvas.draw()
        self.putvalue()
        #self.addmpl()

    def putvalue(self,vall):
        self.textbox.setText(str(vall))

    def close(self):
       self.flag=1
       os._exit(1)

    def createMenusAndToolbars(self):
        fileMenu = self.menuBar().addMenu('Menu')
        fileMenu.addAction('Quit', self.close)

if __name__ == '__main__':
    import sys
    from PyQt4 import QtGui
    import numpy as np
    app = QtGui.QApplication(sys.argv)
    main = Main()
    #main.plot()
    main.addmpl()
    main.createMenusAndToolbars()
    main.show()
    #main.iniciaar()
    sys.exit(app.exec_())
```

Anexo 6:

Programa desenvolvido em Python para aquisição de dados em tempo real - realtimedata.py:

```
__author__ = 'LINE.IPT'

import serial
import numpy
import matplotlib.pyplot as plt
from drawnow import *

voltage= []
pressure=[]
arduinoData = serial.Serial('com4', 115200)
plt.ion() #modo interativo matplotlib
cnt=0

def makeFig(): #funcao para plot
    plt.ylim(0,1023)                                 #gama de valores
    plt.title('RealTime Data')      #titlo
    plt.grid(True)                                   #grid on
    plt.ylabel('Voltage')                             #label
    plt.plot(voltage, 'ro-', label='V')       #plot da temperatura
    plt.legend(loc='upper left')                     #local da legenda
    plt2=plt.twinx()                                 #criar segundo valor
para mostrar
    plt.ylim(0,1023)
    plt2.plot(pressure, 'b^-', label='bar')
    plt2.set_ylabel('Pressrue (Pa)')

    plt2.ticklabel_format(useOffset=False) #nao auto escalar eixo dos y
    plt2.legend(loc='upper right')

while True:
    while (arduinoData.inWaiting()==0): #espera ate receber dados
        pass
    arduinoString = arduinoData.readline() #read line
    dataArray = arduinoString.split(',')   #separa pacote de dados por
linha
    v = float( dataArray[0])            #converter para float
    P = float( dataArray[1])
    voltage.append(v)                        #adiciona o valor a lista para
mostrar no grafico
    pressure.append(P)
    drawnow(makeFig)                         #chama a funcao drawnow para
mostrar em tempo real o grafico
    plt.pause(.000001)                       #pausa para nao crashar o
drawnow
    cnt=cnt+1
    if(cnt>50):                              #contagem so para amostrar 50
pontos
        voltage.pop(0)
        pressure.pop(0)
```

Anexo 7:

Programa do arduino para comunicação com a aplicação realtimedata.py - realtimedata.ino:

```
float voltage;
float pressure;
void setup(){
Serial.begin(115200);
}
void loop() {
  while(1){
voltage=analogRead(0);
pressure=analogRead(1);
Serial.print(voltage);
Serial.print(" , ");
Serial.println(pressure);
delay(250);
}
}
```

Anexo 8:

Programa da aplicação "SCADA" desenvolvido em "Visual Basic" - Form1.vb:

```
Imports System.Net
Imports System.Net.Sockets
Imports System.Threading
Imports System.IO
Imports System.DateTime

Public Class Form1
    Inherits MetroFramework.Forms.MetroForm
    Dim contador_erro As Integer = 0 'contador de pedidos de leitura do socket, para
prevenir do overflow deste
    Dim shut_flag As Boolean = False ' flag para pedido de encerramento da aplicação
    Dim tcpListener As New TcpListener(50002) 'objeto de ligação de rede
    Dim erro As Boolean = False 'variavel auxiliar para a ligação de rede
    Dim file1 As System.IO.StreamWriter 'variavel que guarda os dados para o
ficheiro
    Dim count_file As Integer = 0 'numero de pacotes guardados no ficheiro atual
    Dim count_n_file As Integer = 0 'index do ficheiro atual
    Dim teste(50) As String 'guarda dados individuis do pacote
    Dim array_teste As Integer = 0 'variavel para contar o numero de frames
imprimidos na caixa de texto
    Dim old_date As String ' varivel auxiliar para guardar a data

    Delegate Sub SetTextCallback(ByVal [text] As String)

    'A primeira função a ser chamada no arranque
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim IPHost As IPHostEntry
        IPHost = Dns.GetHostByName(Dns.GetHostName())
        tbIP.Text = IPHost.AddressList(0).ToString()
        'cria um nova thread separada para a função Main
        ThreadPool.QueueUserWorkItem(Sub(value)
                                         main_function()
                                     End Sub)
    End Sub

    'Função Main, é responsavel pelo tratamento de dados em paralelo com a GUI
    Private Sub main_function()
        'Me.BackColor = Color.Blue
        Dim handlerSocket As Socket
        Thread.CurrentThread.Name = " main_function "
        Try
            tcpListener.Start()  'Inicializa a ligação de rede
        Catch ex As System.Net.Sockets.SocketException
            erro = True
            Console.WriteLine("Erro 001")
        End Try
        While erro = False 'enquanto não houver erro de ligação de rede
            Try
                If tcpListener.Pending() And contador_erro < 5 Then 'se existem
dados e não há muitos pedidos pendentes
                    Console.WriteLine("loop1")
                    handlerSocket = tcpListener.AcceptSocket()
                    If handlerSocket.Connected And shut_flag = False Then 'se
aceitou o socket e não há pedido para encerrar o programa
```

```
                        contador_erro += 1
                        Console.WriteLine("loop2 {0}", contador_erro)
                        'cria um nova thread separada para tratamento dos dados do
socket
                        ThreadPool.QueueUserWorkItem(Sub(value)
                                                         stream_read(handlerSocket)
                                                     End Sub)
                    End If
                End If
            Catch ex As System.Net.Sockets.SocketException
                Console.WriteLine("Erro 002")
            End Try

        End While
    End Sub

    'devolve uma string com a data atual
    Private Function date_to_str()
        Return (Date.Now.Year.ToString() + "_" + Date.Now.Month.ToString() + "_" +
Date.Now.Day.ToString())
    End Function

    'função que faz o tratamento dos dados recebidos
    Private Sub stream_read(ByVal connection As Socket)
        Dim aux() As String
        Thread.CurrentThread.Name = " stream_read "
        Dim data As String = Nothing
        If connection.Connected = False Then ' se a ligação foi abaixo
            Console.WriteLine("connection crash")
            contador_erro -= 1
            Exit Sub
        End If
        Dim networkStream As NetworkStream = New NetworkStream(connection)
        Dim blockSize As Int16 = 10240
        Dim thisRead As Int16
        Dim dataByte(blockSize) As Byte
        While connection.Connected And shut_flag = False ' enquanto houverem dados
no socket e não há pedido para encerrar o programa
            Try
                thisRead = networkStream.Read(dataByte, 0, blockSize) ' lê os dados
            Catch ex As System.IO.IOException
                connection.Close()
                connection = Nothing
                thisRead = Nothing
                contador_erro -= 1
                Console.WriteLine("Erro 003")
                Exit Sub
            End Try
            If thisRead > 0 And shut_flag = False Then 'se existem dados e não há
pedido para encerrar o programa
                If file1 Is Nothing Then 'após receber o primeiro pacote, abre/cria
um novo ficheiro
                    Try 'se o ficheiro já existir no computador, vai acrescentar
dados a este
                        file1 =
My.Computer.FileSystem.OpenTextFileWriter(Application.StartupPath & "\" +
date_to_str() + "__0.txt", True)
                        old_date = date_to_str()
                        count_file = 0
                    Catch ex As Exception
```

```
                        Console.WriteLine("Erro 004")
                        Exit Sub
                    End Try
                End If
                data = System.Text.Encoding.ASCII.GetString(dataByte, 0, thisRead)
                aux = data.Split("B")
                For dime As Integer = 0 To aux.GetUpperBound(0)
                    If count_file = 6000 Or Not old_date.CompareTo(date_to_str())
Then 'se chegou aos 6000 pacotes, ou se o dia mudou troca de ficheiro
                        If Not old_date.CompareTo(date_to_str()) Then ' se o dia
mudou
                            count_n_file = 0
                            old_date = date_to_str()
                        Else
                            count_n_file += 1
                        End If
                        'If count_file = 6000 Then 'se chegou aos 6000 pacotes,
troca de ficheiro
                        'count_file = 0
                        'count_n_file += 1
                        file1.Flush()
                        file1.Close()
                        Try ' cria o novo ficheiro para os dados não serem longos no
mesmo ficheiro
                            file1 =
My.Computer.FileSystem.OpenTextFileWriter(Application.StartupPath & "\" +
date_to_str() + "__" + count_n_file.ToString + ".txt", True)
                        Catch ex As Exception
                            Console.WriteLine("Erro 005")
                            Exit Sub
                        End Try
                    End If
                    If (aux(dime) <> "") Then 'se existem dados escreve-os no
ficheiro
                        file1.WriteLine(aux(dime))
                        count_file += 1
                    End If
                    Me.SetText(aux(dime))
                Next
            Else 'se não existem dados
                Exit While
            End If
        End While
        contador_erro -= 1
        connection.Disconnect(True)
    End Sub

    'função que atualiza os dados na caixa de texto e atualiza os gráficos
    Private Sub SetText(ByVal [text] As String)
        Dim serie_gv As String
        Dim serie_gi As String
        Dim serie_flag As Boolean
        Dim dime As Integer
        Dim declivep As Integer
        Dim slopep As Integer
        Dim pico As Integer
        Try
            If Me.TextBox1.InvokeRequired Then 'se é necessário invocar a caixa de
texto
                Dim d As New SetTextCallback(AddressOf SetText)
```

```
                Me.Invoke(d, New Object() {[text]})
            Else 'se não é necessário invocar a caixa de texto
                array_teste += 1
                If (array_teste >= 15) Then 'se já foram  imprimidos 15 frames no
ecra, limpa a caixa de texto
                    Me.TextBox1.Clear()
                    array_teste = 0
                End If
                Me.TextBox1.AppendText([text] + vbCr)
                teste = text.Split("#")
                If teste(2) = "S" Then
                    tbdata.Text = teste(5).ToString + ":" + teste(6).ToString + ":"
+ teste(19).ToString
                    tbhora.Text = teste(20).ToString + "h" + teste(21).ToString +
"m" + teste(22).ToString + "s"
                    Tb2.Text = teste(5) 'ano
                    Tb3.Text = teste(6) 'mes
                    Tb4.Text = teste(19) 'dia
                    Tb5.Text = teste(20) 'hora
                    Tb6.Text = teste(21) 'minuto
                    Tb7.Text = teste(22) 'segundo
                    Tb8.Text = teste(7) + " ciclos" 'slope
                    Tb9.Text = teste(8) + " ciclos" 'weld
                    Tb10.Text = teste(9) + " ciclos" 'cool
                    Tb11.Text = teste(10) + " ciclos" 'weld2
                    Tb12.Text = teste(15) + " V" 'mv
                    Tb13.Text = teste(16) + " bar" 'mp
                    Tb14.Text = teste(17) + " V" 'mv2
                    Tb15.Text = teste(18) + " bar" 'mp2
                    tbns.Text = teste(4) 'numero soldadura
                    tbps.Text = teste(2) 'pacote primario ou secundario
                    tbdt.Text = teste(11) 'declive tensao
                    tbdp.Text = teste(12) 'declive pressao
                    tbpt.Text = teste(13) + " V" 'pico tensao
                    tbpp.Text = teste(14) + " bar" 'pico pressão


                    tbsc.Text = teste(25) + " ciclos" 'slope corrente
                    tbwc.Text = teste(26) + " ciclos" 'weld corrente
                    tbcc.Text = teste(27) + " ciclos" ' cool corrente
                    tbw2c.Text = teste(28) + " ciclos" ' weld2 corrente
                    tbmc.Text = teste(31) + " A" ' media da corrente
                    tbpc.Text = teste(30) + "A" 'pico corrente
                    tbdc.Text = teste(29) 'declive corrente
                    tbnsc.Text = teste(24) 'numero de soldadura
                    tbmc2.Text = teste(32) + " A" ' media da corrente weld2
                    tbprog.Text = teste(33)  'programa de soldadura selecionado

                    'Console.WriteLine(teste.GetUpperBound(0))
                    ' If teste.GetUpperBound(0) = 19 Then 'se o pacote está inteiro
                    ' If teste(0).CompareTo("A") Then 'se o pacote está sincronizado

                    'imprime os dados novos no gráfico consuante o estado das
CheckBoxes
                    'If (serie_flag = True And Form2.CheckBox1.Checked = True) Or
(serie_flag = False And Form2.CheckBox2.Checked = True) Or (Form2.CheckBox1.Checked
= True And Form2.CheckBox2.Checked = True) Then
                    If (MetroCheckBox1.Checked = True) Then
                        Chart1.Series.Clear()
                        Chart1.Series.Add("Tensão")
```

```
                        Chart1.Series("Tensão").ChartType =
DataVisualization.Charting.SeriesChartType.Line
                        ' Form2.TextBox1.Text = "Número da soldadura: " + teste(4)
                        'calculo dos pontos Y para visualização no gráfico
                        declivep = Val(teste(7)) - 1
                        For dime = 0 To declivep Step 1 'fase slope
                            slopep = Val(teste(11)) * dime
                            Chart1.Series("Tensão").Points.AddY(slopep)

'Form2.Chart1.Series(serie_gi).Points.AddY(Convert.ToInt32(teste(11)) * dime)
                        Next dime
                        pico = Val(teste(13))
                        Chart1.Series("Tensão").Points.AddY(pico) 'pico
                        'Form2.Chart1.Series(serie_gi).Points.AddY(teste(13))
                        For dime = 0 To Val(teste(8)) 'fase weld
                            Chart1.Series("Tensão").Points.AddY(Val(teste(15)))

'Form2.Chart1.Series(serie_gi).Points.AddY(Convert.ToInt32(teste(15)))
                        Next
                        For dime = 0 To Val(teste(9)) 'fase cold
                            Chart1.Series("Tensão").Points.AddY(0)

'Form2.Chart1.Series(serie_gi).Points.AddY(Convert.ToInt32(0))
                        Next
                        For dime = 0 To Val(teste(10)) 'fase weld2
                            Chart1.Series("Tensão").Points.AddY(Val(teste(17)))

'Form2.Chart1.Series(serie_gi).Points.AddY(Convert.ToInt32(teste(17)))
                        Next
                        Chart1.Series("Tensão").Points.AddY(0)

'Form2.Chart1.Series(serie_gi).Points.AddY(Convert.ToInt32(0))
                    End If
                End If

                ' If teste(26) = "P" Then
                'tbsc.Text = teste(31) + " ciclos" 'slope corrente
                '  tbwc.Text = teste(32) + " ciclos" 'weld corrente
                '  tbcc.Text = teste(33) + " ciclos" ' cool corrente
                '  tbw2c.Text = teste(34) + " ciclos" ' weld2 corrente
                '  tbmc.Text = teste(39) + " A" ' media da corrente
                ' tbpc.Text = teste(37) + "A" 'pico corrente
                ' tbdc.Text = teste(35) 'declive corrente
                ' tbnsc.Text = teste(28) 'numero de soldadura
                ' End If
                ' End If
            End If
        Catch ex As Exception
            Console.WriteLine("Erro 006")
            Exit Sub
        End Try
    End Sub


    'mostra a janela dos  gráficos
    Private Sub GráficosToolStripMenuItem_Click(sender As Object, e As EventArgs)
        Form2.Show()
    End Sub

    'mostra a janela "sobre"
```

```
    Private Sub SobreToolStripMenuItem_Click(sender As Object, e As EventArgs)
        Form3.Show()
    End Sub

    'Fecha o programa
    Private Sub Close_All()
        shut_flag = True
        If Not file1 Is Nothing Then 'se houve dados novos durante a execução do
programa, fecha o ficheiro
            Try
                file1.Flush()
                file1.Close()
            Catch ex As Exception
                Console.WriteLine("Erro 007")
            End Try
        End If
        Console.WriteLine("saiu close all")
        Thread.Sleep(1000)
        System.Windows.Forms.Application.Exit()

    End Sub

    'pedido para fechar o programa
    Private Sub SairToolStripMenuItem_Click(sender As Object, e As EventArgs)
        'cria thread para close
        If shut_flag = False Then
            ThreadPool.QueueUserWorkItem(Sub(value)
                                             Close_All()
                                         End Sub)

        End If

    End Sub

    'pedido para fechar o programa
    Private Sub btnClose_Click(sender As Object, e As EventArgs)
        'cria thread para close
        If shut_flag = False Then
            ThreadPool.QueueUserWorkItem(Sub(value)
                                             Close_All()
                                         End Sub)

        End If

    End Sub


    'coloca programa em primeiro plano quando cliquado no Ícone da Área de
Notificação
    Private Sub NotifyIcon1_MouseDoubleClick(sender As Object, e As MouseEventArgs)
Handles NotifyIcon1.MouseDoubleClick
        Me.Show()
        Me.WindowState = FormWindowState.Normal
        NotifyIcon1.Visible = False
    End Sub


    'coloca o programa no Ícones da Área de Notificação
    Private Sub Form1_Resize(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.Resize
```

```
        If Me.WindowState = FormWindowState.Minimized Then
            NotifyIcon1.Visible = True
            Me.Hide()
            NotifyIcon1.BalloonTipText = "Programa Minimizado"
            NotifyIcon1.ShowBalloonTip(500)
        End If
    End Sub

    Private Sub Tb3_TextChanged(sender As Object, e As EventArgs)

    End Sub

    Private Sub MetroTile1_Click(sender As Object, e As EventArgs) Handles
MetroTile1.Click
        Form2.Show()
    End Sub

    Private Sub MetroTile2_Click(sender As Object, e As EventArgs) Handles
MetroTile2.Click
        Form3.Show()
    End Sub

    Private Sub MetroTile3_Click(sender As Object, e As EventArgs) Handles
MetroTile3.Click
        'cria thread para close
        If shut_flag = False Then
            ThreadPool.QueueUserWorkItem(Sub(value)
                                             Close_All()
                                         End Sub)

        End If
    End Sub

    Private Sub TextBox1_Click(sender As Object, e As EventArgs) Handles
TextBox1.Click

    End Sub

    Private Sub MetroTextBox6_Click(sender As Object, e As EventArgs) Handles
Tb5.Click

    End Sub

    Private Sub MetroTextBox4_Click(sender As Object, e As EventArgs) Handles
Tb7.Click

    End Sub
End Class
```

Programa da aplicação “SCADA” desenvolvido em “Visual Basic” - Form3.vb:

```
Public Class Form3
    Inherits MetroFramework.Forms.MetroForm
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Me.Close()
    End Sub

    Private Sub Form3_Load(sender As Object, e As EventArgs) Handles MyBase.Load
```

```
    End Sub
End Class
```

Anexo 9:

Programa para inserção de parâmetros via "keypad" – inserção_v2.ino:

```
#include <LiquidCrystal.h>
#include <Keypad.h>
inline void digitalWriteDirect(int pin, boolean val){//fast digital write
  if(val) g_APinDescription[pin].pPort -> PIO_SODR = g_APinDescription[pin].ulPin;
//SODR:Registo HIGH
  else    g_APinDescription[pin].pPort -> PIO_CODR = g_APinDescription[pin].ulPin;
//CODR:Registo low
}
inline int digitalReadDirect(int pin){ //fast digital read
  return !!(g_APinDescription[pin].pPort -> PIO_PDSR & g_APinDescription[pin].ulPin);
//PDSR:Registo de leitura digital
}
inline int pinModeOutputDirect(int pin){ //fast pinmodo output
g_APinDescription[pin].pPort -> PIO_OER = g_APinDescription[pin].ulPin; //OER:
registo de configuração como saida
}
const byte ROWS = 4; // Four rows
const byte COLS = 3; // Three columns
// Define the Keymap
char keys[ROWS][COLS] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}
};
//pinos no teclado da esquerda para a direita (26-38)
// Connect keypad ROW0, ROW1, ROW2 and ROW3 to these Arduino pins.
byte rowPins[ROWS] = { 28, 38, 36, 32 };
// Connect keypad COL0, COL1 and COL2 to these Arduino pins.
byte colPins[COLS] = { 30, 26, 34};
// Create the Keypad
Keypad kpd = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
```

```
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(11, 10, 5, 4, 3, 2);
int button=7;
int pot=5;
int led1=22;
int led2=23;
int prog=6;
int daq=7;
int progstate=0;
int daqstate=0;
void setup() {
  Serial.begin(9600);
  pinMode(button,INPUT);
  pinMode(7,INPUT);
  pinMode(6,INPUT);
  pinModeOutputDirect(led1);
  pinModeOutputDirect(led2);
  digitalWriteDirect(led1,LOW);
  digitalWriteDirect(led2,LOW);
  lcd.begin(16, 2);
  lcd.clear();
  lcd.setCursor(0, 1);
}
int state=0;
int laststate=0;
int ON=0;
int OFF=1;
int count=0;
char um=0;
char dois=0;
char parametro[3]={0};
char parametro2[3]={0};
char fase='I';
char key;
```

```
char key2;
char key3;
char slope[3]={0};
char weld1[3]={0};
char cool[3]={0};
char weld2[3]={0};
char hold[3]={0};
char heat1[3]={0};
char heat2[3]={0};
int valslope=0;
int valweld1=0;
int valcool=0;
int valweld2=0;
int valhold=0;
int valheat1=0;
int valheat2=0;
boolean flag1=true;
boolean flag2=false;
boolean flag3=false;
void resetparametro(char parametroo[2], char fasee){//reset aos parametros
parametroo[0]=0;
parametroo[1]=0;
parametroo[2]=0;
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("reset");
fase=fasee;
delay(1000);
flag1=true;
flag2=false;
flag3=false;
}
int conc=0;
int i=0;
```

```
int total=0;
void loop() {
  while(1){
 ////////////////////modo de configuração//////////////////
    if(digitalReadDirect(prog)==HIGH && digitalReadDirect(daq)==LOW){
    digitalWriteDirect(led1,LOW);
    digitalWriteDirect(led2,HIGH);
      switch (fase){
        case'I'://inicialização
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Init de para");
        key=0;
        key2=0;
        key3=0;
        delay(2000);
        flag1=true;
        fase='A';
        break;
        case 'A'://Slope
      if(flag1){//insercao do primeiro valor
      lcd.setCursor(0, 0);
      lcd.print("Slope:          ");
      lcd.setCursor(1, 1);
        key = kpd.getKey();
        if(key!='*'&& key!='#' && key != NO_KEY)//primeiro numero
        {
        slope[0]=key;
        lcd.print(key);
        flag1=false;
        flag2=true;
        }
        if(key=='*'){//reset
          resetparametro(slope,'A');
```

```
      }
      }
     if(flag2){//inserção do segundo valor
      key2 = kpd.getKey();
      if(key2!='*' && key2!='#' && key2 != NO_KEY){//segundo numero
      slope[1]=key2;
      lcd.print(key2);
      flag2=false;
      flag3=true;
      }
      if(key2=='*'){//reset
        resetparametro(slope,'A');
       }
      }
       if(flag3){//tecla cardinal para confirmar valor inserido
      key3 = kpd.getKey();
      if(key3!='*' && key3!='#' && key3 != NO_KEY){//caso se insira mais de dois
numeros
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("2 numeros");
      delay(2000);
      resetparametro(slope,'A');
      flag3=false;
      flag1=true;
      }
      if(key3=='*'){//reset
      resetparametro(slope,'A');
      }
      if(key3=='#'){//Proximo parametro
        lcd.clear();
        key=0;
        key2=0;
        key3=0;
```

```
  //flag1=false;
  flag2=false;
  flag3=false;
  fase='B';
  flag1=true;
 }
  }
 break;
 case 'B'://segundo parametro
 if(flag1){
 lcd.setCursor(0, 0);
 lcd.print("Weld1          ");
 lcd.setCursor(1, 1);
 key = kpd.getKey();
 if(key!='*'&& key!='#' && key != NO_KEY)
 {
 weld1[0]=key;
 lcd.print(key);
 flag1=false;
 flag2=true;
 }
 if(key=='*'){
  resetparametro(weld1,'B');
 }
 }
if(flag2){
 key2 = kpd.getKey();
 if(key2!='*' && key2!='#' && key2 != NO_KEY){
 weld1[1]=key2;
 lcd.print(key2);
 flag2=false;
 flag3=true;
 }
 if(key2=='*'){
```

```
  resetparametro(weld1,'B');
  }
 }
 if(flag3){
key3 = kpd.getKey();
if(key3!='*' && key3!='#' && key3 != NO_KEY){//Caso se insira mais de dois numeros
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("2 numeros");
delay(2000);
resetparametro(weld1,'B');
flag3=false;
flag1=true;
}
if(key3=='*'){//Reset
resetparametro(weld1,'B');
}
if(key3=='#'){//Proximo valor
  lcd.clear();
  key=0;
  key2=0;
  key3=0;
  flag2=false;
  flag3=false;
  fase='C';
  flag1=true;
}
  }
break;
////////////////////////////////////////////////////////////////////////////////
case 'C'://Terceiro parametro
if(flag1){
lcd.setCursor(0, 0);
```

```
      lcd.print("Cool           ");
      lcd.setCursor(1, 1);
      key = kpd.getKey();
      if(key!='*'&& key!='#' && key != NO_KEY)
      {
      cool[0]=key;
      lcd.print(key);
      flag1=false;
      flag2=true;
      }
      if(key=='*'){
        resetparametro(cool,'C');
      }
      }
     if(flag2){
      key2 = kpd.getKey();
      if(key2!='*' && key2!='#' && key2 != NO_KEY){
      cool[1]=key2;
      lcd.print(key2);
      flag2=false;
      flag3=true;
      }
      if(key2=='*'){
        resetparametro(cool,'C');
        }
       }
       if(flag3){
      key3 = kpd.getKey();
      if(key3!='*' && key3!='#' && key3 != NO_KEY){//Caso se insira mais de dois
numeros
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("2 numeros");
      delay(2000);
```

```
resetparametro(cool,'C');
flag3=false;
flag1=true;
}
if(key3=='*'){//Reset
resetparametro(cool,'C');
}
if(key3=='#'){//Proximo valor
  lcd.clear();
  key=0;
  key2=0;
  key3=0;
  flag2=false;
  flag3=false;
  fase='D';
  flag1=true;
}
  }
break;
////////////////////////////////////////////////////////////////////////////
case 'D'://quarto parametro
if(flag1){
lcd.setCursor(0, 0);
lcd.print("Weld2           ");
lcd.setCursor(1, 1);
key = kpd.getKey();
if(key!='*'&& key!='#' && key != NO_KEY)
{
weld2[0]=key;
lcd.print(key);
flag1=false;
flag2=true;
}
if(key=='*'){
```

```
        resetparametro(weld2,'D');
      }
      }
     if(flag2){
      key2 = kpd.getKey();
      if(key2!='*' && key2!='#' && key2 != NO_KEY){
      weld2[1]=key2;
      lcd.print(key2);
      flag2=false;
      flag3=true;
      }
      if(key2=='*'){
        resetparametro(weld2,'D');
       }
      }
      if(flag3){
      key3 = kpd.getKey();
      if(key3!='*' && key3!='#' && key3 != NO_KEY){//Caso se insira mais de dois
numeros
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("2 numeros");
      delay(2000);
       resetparametro(weld2,'D');
      flag3=false;
      flag1=true;
      }
      if(key3=='*'){//Reset
       resetparametro(weld2,'D');
      }
      if(key3=='#'){//Proximo valor
        lcd.clear();
        key=0;
        key2=0;
```

```
        key3=0;
        flag2=false;
        flag3=false;
        fase='E';
        flag1=true;
      }
       }
      break;
////////////////////////////////////////////////////////////////////////////////////////
      case 'E'://Quinto parametro
      if(flag1){
      lcd.setCursor(0, 0);
      lcd.print("Hold            ");
      lcd.setCursor(1, 1);
      key = kpd.getKey();
      if(key!='*'&& key!='#' && key != NO_KEY)
      {
      hold[0]=key;
      lcd.print(key);
      flag1=false;
      flag2=true;
      }
      if(key=='*'){
        resetparametro(hold,'E');
      }
      }
     if(flag2){
      key2 = kpd.getKey();
      if(key2!='*' && key2!='#' && key2 != NO_KEY){
      hold[1]=key2;
      lcd.print(key2);
      flag2=false;
      flag3=true;
      }
```

```
      if(key2=='*'){
        resetparametro(hold,'E');
       }
      }
      if(flag3){
      key3 = kpd.getKey();
      if(key3!='*' && key3!='#' && key3 != NO_KEY){//Caso se insira mais de dois
numeros
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("2 numeros");
      delay(2000);
       resetparametro(hold,'E');
      flag3=false;
      flag1=true;
      }
      if(key3=='*'){//Reset
       resetparametro(hold,'E');
      }
      if(key3=='#'){//Proximo valor
        lcd.clear();
        key=0;
        key2=0;
        key3=0;
        flag2=false;
        flag3=false;
        fase='F';
        flag1=true;
      }
       }
      break;
////////////////////////////////////////////////////////////////
      case 'F'://Quinto parametro
      if(flag1){
```

```
      lcd.setCursor(0, 0);
      lcd.print("Heat 1           ");
      lcd.setCursor(1, 1);
      key = kpd.getKey();
      if(key!='*'&& key!='#' && key != NO_KEY)
      {
      heat1[0]=key;
      lcd.print(key);
      flag1=false;
      flag2=true;
      }
      if(key=='*'){
        resetparametro(heat1,'F');
      }
      }
     if(flag2){
      key2 = kpd.getKey();
      if(key2!='*' && key2!='#' && key2 != NO_KEY){
      heat1[1]=key2;
      lcd.print(key2);
      flag2=false;
      flag3=true;
      }
      if(key2=='*'){
         resetparametro(heat1,'F');
        }
       }
       if(flag3){
      key3 = kpd.getKey();
      if(key3!='*' && key3!='#' && key3 != NO_KEY){//Caso se insira mais de dois
numeros
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("2 numeros");
```

```
        delay(2000);
         resetparametro(heat1,'F');
        flag3=false;
        flag1=true;
        }
        if(key3=='*'){//Reset
         resetparametro(heat1,'F');
        }
        if(key3=='#'){//Proximo valor
          lcd.clear();
          key=0;
          key2=0;
          key3=0;
          flag2=false;
          flag3=false;
          fase='G';
          flag1=true;
        }
          }
        break;
////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
        case 'G'://Setimo parametro
        if(flag1){
        lcd.setCursor(0, 0);
        lcd.print("Heat 2            ");
        lcd.setCursor(1, 1);
        key = kpd.getKey();
        if(key!='*'&& key!='#' && key != NO_KEY)
        {
        heat2[0]=key;
        lcd.print(key);
        flag1=false;
        flag2=true;
```

```
      }
      if(key=='*'){
        resetparametro(heat2,'G');
      }
      }
     if(flag2){
      key2 = kpd.getKey();
      if(key2!='*' && key2!='#' && key2 != NO_KEY){
      heat2[1]=key2;
      lcd.print(key2);
      flag2=false;
      flag3=true;
      }
      if(key2=='*'){
       resetparametro(heat2,'G');
        }
       }
       if(flag3){
      key3 = kpd.getKey();
      if(key3!='*' && key3!='#' && key3 != NO_KEY){//Caso se insira mais de dois
numeros
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("2 numeros");
      delay(2000);
      resetparametro(heat2,'G');
      flag3=false;
      flag1=true;
      }
      if(key3=='*'){//Reset
      resetparametro(heat2,'G');
      }
      if(key3=='#'){//Proximo valor
        lcd.clear();
```

```
          key=0;
          key2=0;
          key3=0;
          flag2=false;
          flag3=false;
          fase='H';
          flag1=true;
        }
         }
        break;
////////////////////////////////////////////////////////
       case 'H': //fase final
      lcd.clear();
      delay(1000);
      lcd.setCursor(0, 0);
      lcd.print(slope[0]);
      lcd.setCursor(6, 0);
      lcd.print(slope[1]);
      lcd.setCursor(0, 1);
      lcd.print(weld1[0]);
      lcd.setCursor(6, 1);
      lcd.print(weld1[1]);
valslope=atoi(slope);//concatenação dos valores inseridos
valweld1=atoi(weld1);
valcool=atoi(cool);
valweld2=atoi(weld2);
valhold=atoi(hold);
valheat1=atoi(heat1);
valheat2=atoi(heat2);
      lcd.setCursor(8, 0);
      lcd.print(slope);
      lcd.setCursor(8, 1);
      lcd.print(weld1);
      delay(5000);
```

```
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Trocar modo de func");
      delay(5000);
        break;
        default:
        fase='I';
        break;
        }
  }
 ////////////////////////////////////////////////////////
//////////////////////modo de aquisição//////////////////
if(digitalReadDirect(prog)==LOW && digitalReadDirect(daq)==HIGH){
   lcd.clear();
   lcd.setCursor(0, 0);
    lcd.print("DAQ           ");
    digitalWriteDirect(led1,HIGH);
    digitalWriteDirect(led2,LOW);
    delay(1000);
    }
  /// ////////////////////////////////////////////////////////
  //////////////////////modo sem configuração
    if(digitalReadDirect(prog)==LOW && digitalReadDirect(daq)==LOW){
      lcd.clear();
      fase='INI';
    lcd.print("Escolha Config      ");
    digitalWriteDirect(led1,HIGH);
    digitalWriteDirect(led2,HIGH);
    delay(1000);
    }
    }
}
```

Anexo 10:

Script em Matlab para análise de parâmetros adquiridos de uma soldadura - analisedadosv2.m:

```
clc
clear all
%% Import data from text file.
% Script for importing data from the following text file:
%
%    E:\drop\Dropbox\LINE_TMR (1)\MFTE
%    MITSUBISHI\Visitas\15-01-2015\matlab2\dados_2014_12_19__0.txt
%
% To extend the code to different selected data or a different text file,
% generate a function instead of a script.

% Auto-generated by MATLAB on 2015/01/16 11:09:51

%% Initialize variables.
filename = 'dados_2014_12_19__0.txt';
delimiter = '#';

%% Format string for each line of text:
%   column1: text (%s)
%   column2: double (%f)
%   column3: text (%s)
%   column4: double (%f)
%   column5: double (%f)
%   column6: double (%f)
%   column7: double (%f)
%   column8: double (%f)
%   column9: double (%f)
%   column10: double (%f)
%   column11: double (%f)
%   column12: double (%f)
%   column13: double (%f)
%   column14: double (%f)
%   column15: double (%f)
%   column16: double (%f)
%   column17: double (%f)
%   column18: double (%f)
%   column19: double (%f)
% For more information, see the TEXTSCAN documentation.
formatSpec = '%s%f%s%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%[^\n\r]';

%% Open the text file.
fileID = fopen(filename,'r');

%% Read columns of data according to format string.
% This call is based on the structure of the file used to generate this
% code. If an error occurs for a different file, try regenerating the
code
% from the Import Tool.
dataArray = textscan(fileID, formatSpec, 'Delimiter', delimiter,
'ReturnOnError', false);

%% Close the text file.
```

```
fclose(fileID);

%% Post processing for unimportable data.
% No unimportable data rules were applied during the import, so no post
% processing code is included. To generate code which works for
% unimportable data, select unimportable cells in a file and regenerate
the
% script.

%% Allocate imported array to column variable names
Initi = dataArray{:, 1};
ID_transformador = dataArray{:, 2};
P_S = dataArray{:, 3};
ID_maquina = dataArray{:, 4};
n_sold = dataArray{:, 5};
time = dataArray{:, 6};
slope = dataArray{:, 7};
weld = dataArray{:, 8};
cool = dataArray{:, 9};
weld2 = dataArray{:, 10};
declive_v = dataArray{:, 11};
declive_i = dataArray{:, 12};
pico_v = dataArray{:, 13};
pico_i = dataArray{:, 14};
media_final_v = dataArray{:, 15};
media_final_i = dataArray{:, 16};
media_final_v2 = dataArray{:, 17};
media_final_i2 = dataArray{:, 18};
erro = dataArray{:, 19};

%% Clear temporary variables
clearvars filename delimiter formatSpec fileID dataArray ans;
j=1;
s=1;
cprimario=0;
tprimario=0;
csecundario=0;
tsecundario=0;
for i=1:length(P_S)%separação dos pacotes do primario do secundario
  if (strcmp(P_S(i),'P')==1)%se encontrar um P pertence ao primario
     cprimario(j,1)=media_final_i2(i);%guarda os valores em variaveis
auxiliares
     tprimario(j,1)=media_final_v2(i);
     j=j+1;
  end

    if (strcmp(P_S(i),'S')==1)%se encontrar um s pertence ao secundario
     %csecundario(s,1)=media_final_i2(i);
     tsecundario(s,1)=media_final_v2(i);
     s=s+1;
    end

end
%%%%grafico tensao primario e secundario %%%%%%%%%%%%%%%5
stsecundario=1/100:1/50:length(tsecundario)/50-1/100;
stprimario=1/100:1/50:length(tprimario)/50-1/100;
linestring='-o';
figure(1)
```

```
subplot(2,1,1)
plot(stsecundario,tsecundario,linestring,'MarkerFaceColor','b'),grid On;
title('Dados recolhidos do valor médio de tensão(secundário)');
legend('Valor de tensão média secundário','Location','NorthEast');
xlabel('Vector de tempo');
ylabel('Escala do porto analógico');
subplot(2,1,2)
plot(stprimario,tprimario,linestring,'MarkerFaceColor','g'),grid On;
title('Dados recolhidos do valor médio de tensão(primário)');
legend('Valor de tensão média primário','Location','NorthEast');
xlabel('Vector de tempo');
ylabel('Escala do porto analógico');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%grafico corrente primario%%%%%%%%%%%%%%%%
scsecundario=1/100:1/50:length(csecundario)/50-1/100;
scprimario=1/100:1/50:length(cprimario)/50-1/100;
figure(2)
plot(scprimario,cprimario,linestring,'MarkerFaceColor','g'),grid On;
title('Dados recolhidos do valor medio de corrente(primário)');
legend('Corrente','Location','NorthEast');
xlabel('Vector de tempo');
ylabel('Escala do porto analógico');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%grafico rms todos os valores%%%%%%%%%%%
gweld2=0;
gcool=0;
nfigura=3;
cria_grafico( slope,weld,gweld2,
gcool,declive_v,pico_v,media_final_v,media_final_v2,nfigura,'Todos os
valores' );
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%grafico rms para slope <5 (valores do secundário)
%%%%%%%%%%%%%%%%%%%%%%%%%
i=0;
j=1;
aslope=0;
aweld=0;
adeclivev=0;
apicov=0;
amfv=0;
amfv2=0;
count=0;
count2=0;
for i=1:length(slope)%percorre todos os pacotes

  if (strcmp(P_S(i),'S')==1)
    if (slope(i)<5)%slope inferior a 4
     aslope(j,1)=slope(i);%guarda os valores em variaveis auxiliares
     aweld(j,1)=weld(i);
     adeclivev(j,1)=declive_v(i);
     apicov(j,1)=pico_v(i);
     amfv(j,1)=media_final_v(i);
     amfv2(j,1)=media_final_v2(i);
     j=j+1;
     count=count+1;%variavel para contar numero de soldaduras com weld<4
  end
  end
  count2=count2+1;%variavel para contar numero de soldaduras
end
nfigura=4;
```

```
cria_grafico( aslope,aweld,gweld2,
gcool,adeclivev,apicov,amfv,amfv2,nfigura,'Tensao rms secundario' );
disp('Numero de soldaduras com weld<4');
disp(count)
disp('Numero de soldaduras');
disp(count2)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%grafico rms para slope <5 (valores do primario)
%%%%%%%%%%%%%%%%%%%%%%%%%
i=0;
j=1;
aslope=0;
aweld=0;
adeclivev=0;
apicov=0;
amfv=0;
amfv2=0;
count=0;
count2=0;
for i=1:length(slope)%percorre todos os pacotes

  if (strcmp(P_S(i),'P')==1)
    if (slope(i)<5)%slope inferior a 4
        if(media_final_v(i)>100)
     aslope(j,1)=slope(i);%guarda os valores em variaveis auxiliares
     aweld(j,1)=weld(i);
     adeclivev(j,1)=declive_v(i);
     apicov(j,1)=pico_v(i);
     amfv(j,1)=media_final_v(i);
     amfv2(j,1)=media_final_v2(i);
     j=j+1;
     count=count+1;%variavel para contar numero de soldaduras com weld<4
        end
  end
  end
  count2=count2+1;%variavel para contar numero de soldaduras
end
nfigura=5;
cria_grafico( aslope,aweld,gweld2,
gcool,adeclivev,apicov,amfv,amfv2,nfigura,'Tensao rms primario' );
disp('Numero de soldaduras com weld<4');
disp(count)
disp('Numero de soldaduras');
disp(count2)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Script em Matlab para análise de parâmetros adquiridos de uma soldadura - cria_grafico.m:

```
function [  ] = cria_grafico( slope,weld,gweld2,
gcool,declive_v,pico_v,media_final_v,media_final_v2,nfigura,nome )
q=1;
gslope=0;%reset a variaveis
gweld=0;
gdeclive=0;
i=0;
w=0;
grafico=0;
gpico=0;
f=0;
u=0;
gv1=0;
gv2=0;
linestring='-o';


for i=1:length(slope)%%ciclo para criar o grafico para todas as medidas

gslope=slope(i);
gweld=weld(i);
gweld2=0;
gcool=0;
gdeclive=declive_v(i);
gpico=pico_v(i);
gv1=media_final_v(i);
gv2=media_final_v2(i);

for w=0:gslope
   grafico(w+1)=w*gdeclive;%parte do declive
end
w=w+1;
grafico(w)=gpico;%parte do pico
f=w;%para plot do grafico

for u=w+1:gweld+w
    grafico(u)=gv1; %parte weld inicial
end

for w=u+1:gcool+w %parte do cool
    grafico(w)=0;
end

for u=w+1:gweld2+w %segunda parte da soldadura
    grafico(u)=gv2;
end

grafico(length(grafico)+1)=0;%fecha o grafico

    tgrafico=1/100:1/50:length(grafico)/50-1/100;%vetor de tempo para
amostragem
    figure(nfigura)
    subplot(2,1,1)
    hold on
```

```
    plot(tgrafico,grafico,linestring,'MarkerFaceColor','b'),grid
on;%grafico rms
    subplot(2,1,2)
    hold on
    plot((f-1)/50+1/100,gpico,'og','MarkerFaceColor','g'),grid
on;%grafico pico

end

subplot(2,1,1)
xlim([0 0.8]);%configuracoes dos graficos
ylim([0 800]);
title(nome);
legend('Tensão RMS','Location','NorthEast');
xlabel('Segundos');
ylabel('Escala do porto analógico do micro');
subplot(2,1,2)
xlim([0 0.8]);
ylim([0 800]);
title(nome);
legend('Pico_v','Location','NorthEast');
xlabel('Segundos');
ylabel('Escala do porto analógico do micro');
hold off
```

Anexo 11:

Script em Matlab para aquisição de dados em "tempo real" de uma soldadura - Readsold.m:

```
%%
%clear all
clc
%Definição da porta de comunicação a utilizar
%nporta=input('Introduza o número da porta COM a utilizar: ');
%porta=['COM' num2str(nporta)]
baudr=115200;
porta=['COM' num2str(7)];
%obj1 = instrfind('Type', 'serial', 'Port', porta, 'Tag', '')
obj1 = serial(porta,'BaudRate',baudr, 'InputBufferSize', 10000);

% Se não existir o objecto para a porta escolhida cria o objecto
% Se já existir fecha a porta e utiliza o objecto já existente

%Verifica se porta está ocupada
aux= instrfind
if isempty(obj1)
    fprintf('Porta Livre \n')
    obj1 = serial(porta,'BaudRate',baudr, 'InputBufferSize', 10000);
else
    fprintf('Porta aberta \n')
    fclose(obj1);
    obj1 = serial(porta,'BaudRate',baudr, 'InputBufferSize', 10000);
end

% Abre a porta escolhida
fopen(obj1);
obj1

%%
% Comunicação RS232 - com protocolo
% Ciclo infinito até botão "quit" ser pressionado
CANAL=1;
dados=zeros(1,4100);
i=1;
count=0;
fase=0;
data2=0;
init=0;
if(exist('file_t','var')==0)
    file_t=0;
end
file_t=file_t+1;
while (i<=4100)
    data =fread(obj1, 1,'uint8');
    if(init==2)
        data2 =fread(obj1, 1,'uint8');
    end
    if (data == 10 && init==0)                          %1° Byte SYNC
            disp('inicio');
            i=1;
            init=1;
```

```
    else if(data== 5 && init==1)                          %2° Byte SYNC
            disp('sincronizado');
            init=2;
    else
        if(init==2)
           dados(i)=data*256+data2;
           i=i+1;
        end
        end
    end
end
fclose(obj1);
disp('FIM');
dados_temp = cat(2,dados(4041:4080), dados(1:4040));
dados_temp1= cat(2,zeros(1,40),dados_temp);

rms_v=RMS3(dados_temp1);
```

Script em Matlab para aquisição de dados em "tempo real" de uma soldadura - RMS3.m:

```
function [xpto_u] = RMS(data_t)
colorstring = 'gyrcmykbgk';
linestring='-o';
t=0:1/(40*50):(length(data_t)-1)/(40*50);
figure(2)
plot(t,data_t,'-b');
xlim([0 0.86]);
hold on

for ii=0:9
    if(ii<8)
        data_t2= cat(2,zeros(1,ii*5),data_t);
    else
        data_t2= cat(2,data_t,zeros(1,ii*5));
    end
    %data_t2= cat(2,zeros(1,ii*5),data_t);
    rms_var=0;
    %rms_var2=0;
    xpto_var=0;
    j=1;
    contador_s=0;
    COOL_const=16;
    SLOPE_const=20;
    MIN_const=22;
    fases=0;
    flag_weld_on=0;
    %flag_weld_ant=0;
    %first_v=0;
    %media_actu_v=0;
    media_ant_v=0;
    slope=0;
    cool=0;
    weld=0;
    weld2=0;
    first_v=0;
    pico_v=0;
    declive_v=0;
    media_weld_v=int64(0);
    media_weld_v2=0;
    media_final_v=0;
    media_final_v2=0;
    desfasamento=0;
    zerosw=0;
    count1=0;

    for i=1:length(data_t)+ii*5
        xpto_var=xpto_var+data_t2(i)*data_t2(i);
        count1=count1+1;
        if(ii<8)
            if(fases==0 && data_t2(i)<1)
                if (zerosw==0)
                    desfasamento=desfasamento+1;
                end
            else if(fases==0 && data_t2(i)>=1)
                    if(desfasamento>0 && zerosw==0)
```

```
                    desfasamento=desfasamento-1;
                end
                zerosw=1;
            end
        end
    end
    if (mod(count1,40)==0 && i>1)%faz media
        if(desfasamento>0)
            count1=40-desfasamento;
            desfasamento=0;
            rms_var(1)=0;
            j=2;
            continue
        end
        rms_var(j)=int16(sqrt(xpto_var/40));
        media_actu_v=int16(rms_var(j));
        xpto_var=0;
        j=j+1;
        count1=0;
        zerosw=0;
        if(media_actu_v<MIN_const)
            flag_weld_on=0;
            if(contador_s<COOL_const)
                contador_s=contador_s+1;
            end
            if(fases==3 && contador_s==COOL_const)
                fases=5;
            end
        else
            flag_weld_on=1;
        end
        if(flag_weld_on==1)
            switch (fases)
                case {0}
                    contador_s=0;
                    first_v=media_ant_v;
                    fases=1;
                    weld=0;
                    slope=0;
                case {1}
                    if(int16(media_actu_v-media_ant_v)<=SLOPE_const)
                        declive_v=int16((media_ant_v-
first_v)/(slope+1));
                        slope=slope+1;
                        media_weld_v=media_actu_v;
                        fases=2;
                        weld=1;
                        pico_v=media_ant_v;
                    else
                        slope=slope+1;
                    end
                case {2}
                    media_weld_v=media_weld_v+media_actu_v;
                    weld=weld+1;
                case {3}
                    if(contador_s<COOL_const)
                        fases=4;
                        weld2=1;
                        cool=contador_s;
```

```
                            media_weld_v2=media_actu_v;
                        end
                    case {4}
                        media_weld_v2=media_weld_v2+media_actu_v;
                        weld2=weld2+1;
                end
            else
                switch (fases)
                    case{0}
                        desfasamento=0;
                    case {2}
                        fases=3;
                        weld2=0;
                        cool=0;
                        media_weld_v2=0;
                        if(weld>0)
                            media_final_v=int16(media_weld_v/(weld));
                        end
                    case {4}
                        fases=5;
                        if(weld2>0)
                            media_final_v2=int16(media_weld_v2/(weld2));
                        end
                end
            end
            media_ant_v=media_actu_v;
        end
    end
    t1=1/(2*50):1/(50):((length(rms_var)-1)/(50))+(1/(2*50));
    if(ii==0)
        figure(2);
        plot(t1,rms_var,cat(2,linestring,colorstring(ii+1)));
        title('Dados recolhidos do valor médio de tensão(secundário)');
        legend('Valor de tensão AC retificado','Valor médio de
tensão','Location','NorthEast');
        xlabel('Vector de amostras');
        ylabel('Valor analógico');
        xlim([0 0.86]);
        figure (3)
        hold on
    end
    if(ii==4 || ii==0 || ii~=10)
        subplot(2,1,1)
        hold on
        plot(t1,rms_var,cat(2,linestring,colorstring(ii+1)));
        xlim([0 0.86]);
        subplot(2,1,2)
        hold on
        graf_d=0;

        for yy=0:slope
            graf_d(yy+1)=yy*declive_v;
        end
        yy=yy+1;
        graf_d(yy)=pico_v;
        for zz=yy+1:weld+yy
            graf_d(zz)=media_final_v;
        end
        for yy=zz+1:cool+zz
```

```
            graf_d(yy)=0;
        end
        for zz=yy+1:weld2+yy
            graf_d(zz)=media_final_v2;
        end
        zz=zz+1;
        graf_d(zz)=0;
        t2=1/100:1/50:length(graf_d)/50-1/100;
        plot(t2,graf_d,cat(2,linestring,colorstring(ii+1)));
        xlim([0 0.86]);
    end
    if(ii==0)
        slope
        cool
        weld
        weld2
        first_v
        pico_v
        declive_v
        media_final_v
        media_final_v2
    end
end
hold off

%%
xpto_u=0;
end
```

Anexo 12:

Algoritmo do microcontrolador para aquisição de dados em "tempo real" de uma soldadura com Matlab - slave_secundario_mablab.ino:

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include "digitalWriteFast1.h"
//#define DEBUG //para debug
#define LED1  30
#define LED2  46
#define LED3  48
#define TAMANHO 40 //comprimento dos arrays para amostras
#define TAMANHO_M 100 //comprimento dos arrays para medias
#define MIN 300  //valor minimo para se considerar desligado
#define SLOPE 20//diferenca de tensão media minimo para considerar que está na fase slope
#define COOL 16//numero de ciclos maximo para ainda considerar fase cool
#define ID_MAQUINA 1 //Identificação da maquina
#define SIZE_FRAME 35//26 //tamanho do pacote de transmissão dos slaves
#define BAUDRATE_XBEE 38400 //velocidade de transmissão com o master
#define BAUDRATE_WI 115200//velocidade de transmissão com a porta serie para debug
/*****************Carateres especiais para receção de dados do slave primário************************/
#define ESCAPE_FRAME 47 //  '/'
#define START_FRAME 65 //  'A'
int teste_ind=0;
int teste_sol=0;
int count_r=0;
int fases=0;
//buffer que guardam as amostragens de um ciclo
long buf_v[TAMANHO];
boolean flag_transmite=false; //indica que a soldadura acabou
int ind_buf=0;//index dos buffers "buf_v,buf_i"
//int cont_amostr=0;//contador de numero amostras num ciclo
boolean ss=false;//para alternar o estado do led para indicação de nova amostra
```

```
byte buff_trans[SIZE_FRAME+1];
unsigned long xpto=0;
unsigned long xpto_v=0;
int contador_teste=0;
void save_amostra(){
  static long aux_v,aux_i;
    aux_v=analogRead(A0);//tensao
   // aux_i=analogRead(A1);//corrente
  digitalWriteFast(LED1,ss);
  ss=~ss;
  if(fases==0){
    buf_v[ind_buf]=aux_v;
    xpto_v+=aux_v;
    ind_buf++;//incrementa index
    if(ind_buf>=40){    //apos 40 amostras guarda a media
      if(xpto_v>MIN){
        contador_teste=0;
        fases=1;
      }
      ind_buf=0;
      xpto_v=0;
    }
  }else if(fases==1){
    if(contador_teste<4000){
      if(contador_teste==0){
        Serial.write(10);
        Serial.write(5);
      }
      //Serial.print(aux_v);
      Serial.write((aux_v&0xFF00)>>8);
      Serial.write(aux_v&0x00FF);
    }
    contador_teste++;
    if(contador_teste>=4000){
```

```
      fases=0;
      contador_teste=0;
      flag_transmite=true;
    }
  }
}
ISR(TIMER3_COMPA_vect){//interrupção para leitura dos sensores
  if(flag_transmite==false){
    save_amostra();
  }
}
void setup() {
  cli();//desliga interrupções
  pinMode(A0, INPUT);//entrada analogica 1
  pinMode(A1, INPUT);//entrada analogica 2
  pinMode(LED1,OUTPUT);//led que indica nova amostagem
  digitalWriteFast(LED1,LOW);
  /////Timer3////////////
  TCCR3A=0;//reset ao timer //registos com valores de configurações
  TCCR3B=0;//reset ao timer //registos com valores de configurações
  OCR3A=124;//valor para overflow do timer
  TCCR3B |= (1<<WGM12);//liga modo ctc
  TCCR3B |= (1 << CS31) | (1 << CS30);//prescaler de 64
  //TCCR3B |= (1 << CS32);//prescaler de 256
  TIMSK3 |= (1<<OCIE3A);//liga o timer para modo de comparaçao de interrupçao
  Serial.begin(BAUDRATE_WI);
  delay(1000);//delay de 1 segundo para dar tempo de inicializar as portas de comunicação
  //flag_transmite=false;
  sei();//liga interrupções
  Serial.println("FIM SETUP");
}
int programa=0; //0=SW1, 1=SW3,  no caso so slave primário também: 3=SW2,4=SW4
void loop() {
  static int i,j,k;
```

```
  if(flag_transmite==true){
    for(i=0;i<40;i++){
      Serial.write((byte)0);
      Serial.write((byte)0);
    }
    for(i=0;i<40;i++){
      Serial.write((buf_v[i]&0xFF00)>>8);
      Serial.write(buf_v[i]&0x00FF);
    }
    for(i=0;i<40;i++){
      Serial.write((byte)0);
    }
    delay(5000);
    flag_transmite=false;
  }
  delay(100);
}
```