DANIEL KÖRBES

TRAÇADOR DE PERFIL TÉRMICO PARA USO NA INDÚSTRIA ELETRÔNICA

FLORIANÓPOLIS, 2011

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE ESPECIALIZAÇÃO EM DESENVOLVIMENTO DE PRODUTOS ELETRÔNICOS

DANIEL KÖRBES

TRAÇADOR DE PERFIL TÉRMICO PARA USO NA INDÚSTRIA ELETRÔNICA

Trabalho de Conclusão de Curso submetido ao Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina como parte dos requisitos para obtenção do título de Especialista em Desenvolvimento de Produtos Eletrônicos

Professor Orientador: Everton Luiz Ferret dos Santos, M. Eng.

FLORIANÓPOLIS, 2011

K841t Korbes, Daniel
Traçador de perfil térmico para uso na indústria eletrônica [monografia] / Daniel Korbes ; orientador, Everton Luiz Ferret dos Santos. – Florianópolis, SC, 2011.
1 v. : il. Tabs. Gráficos.

Monografia de especialização (Eletrônica) – Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina. Departamento Acadêmico de Eletrônica. Curso de Especialização em Desenvolvimento de Produtos Eletrônicos.
Inclui referências.

1. Eletrônica. 2. Soldagem eletrônica. 3. Controle de processosl. I. Santos, Everton Luiz Ferret dos. II. Título
CDD: 621.381

Sistema de Bibliotecas Integradas do IFSC
Biblioteca Dr. Hercílio Luz – Campus Florianópolis
Catalogado por: Elaine Santos da Silva CRB 14/1182
Rose Mari Lobo Goulart CRB 14/277

TRAÇADOR DE PERFIL TÉRMICO PARA USO NA INDÚSTRIA ELETRÔNICA

DANIEL KÖRBES

Este trabalho foi julgado adequado para obtenção do Título de Especialista em Desenvolvimento de Produtos Eletrônicos e aprovado na sua forma final pela banca examinadora do Curso de Pós-Graduação em Desenvolvimento de Produtos Eletrônicos do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis, 13 de Dezembro de 2011

Banca Examinadora:

_____

Everton L. F. Santos, M.Eng.

_____

Charles Borges de Lima, Dr. Eng.

_____

Leandro Schwarz, M. Eng.

Dedico esse trabalho a minha família, especialmente a meus pais Patrício e Roswitha.

AGRADECIMENTOS

Agradeço inicialmente a minha família. Meus pais, Patrício e Roswitha, pelo apoio incondicional, suporte afetivo e financeiro nas horas de aperto. A meu irmão André pelas lições de programação visual e aos amigos Juliano Fusinato e Ederson Ferronato pela grande contribuição na confecção do protótipo e grandes incentivadores do trabalho. A todos os amigos que nas boas e más horas foram “ouvidos” e “palavras”.

“The first requisite for success is to develop the ability to focus and apply your mental and physical energies to the problem at hand - without growing weary. Because such thinking is often difficult, there seems to be no limit to which some people will go to avoid the effort and labor that is associated with it....”

Thomas Alva Edison

“O primeiro requisito para o sucesso é desenvolver a capacidade de concentrar e aplicar as suas energias físicas e mentais para o problema em questão - sem cansar. Porque tal pensamento é muitas vezes difícil, parece não haver limite para o que algumas pessoas façam para evitar o esforço e o trabalho que está associado com ele....” (Tradução aproximada)

Resumo

Esse trabalho tem por objetivo apresentar um equipamento que auxilie os profissionais da indústria de montagem eletrônica moderna na etapa de ajustes do processo de soldagem automatizada. São apresentados alguns conceitos de metalurgia básica aplicada ao tema, os problemas envolvidos nos processos de soldagem eletrônica, soluções conhecidas que envolvam o controle da temperatura do processo e o desenvolvimento do equipamento em si. O equipamento é microcontrolado e trás funções equivalentes aos similares do mercado, porém com foco em baixo custo e facilidade de uso. Tal equipamento é acompanhado de um software para PC desenvolvido unicamente para essa aplicação.

Palavras-chave: Soldagem eletrônica. Microcontradores. Controle de Processos.

Abstract

This paper aims to present a device that assists professionals in the modern electronics assembly industry in the settings stage of the automated welding process. We introduce some concepts of metallurgy applied to the theme, the problems involved in the process, known solutions involving the temperature control process and the development of the equipment itself. This equipment uses a microcontroller and had functions as the similar ones, but with a focus on low cost and ease of use. Such equipment is accompanied by a PC software developed exclusively for this application .

Keywords: Welding. Microcontrollers. Process Control.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1. INTRODUÇÃO GERAL

A eletrônica tem possibilitado uma rápida evolução -em todas as áreas do conhecimento. Medicina, mecânica, administração dentre outras fazem uso de aparelhos eletrônicos para desenvolvimento do conhecimento. A própria eletrônica vale-se dessa evolução.

Componentes menores que consomem menos energia e que realizam suas funções em menor tempo possibilitam a criação de tecnologias cada vez melhores. E para que essas tecnologias sejam inseridas, difundidas e aproveitadas pelo mercado consumidor é necessário o aperfeiçoamento dos processos de montagem da indústria eletrônica.

Processos de fabricação mais rápidos, baratos e eficientes são a chave para a difusão de qualquer tecnologia. A montagem de eletrônicos ganhou grande impulso na difusão com a adoção de processos automatizados de inserção e soldagem nos anos 60 e 70. Com o surgimento dos componentes SMD, no final da década de 70, a indústria modificou seus padrões e os processos ganharam ainda mais velocidade.

A montagem de circuitos eletrônicos envolve várias etapas, podendo-se citar:

- Preparação de componentes.
- Inserção de componentes.
- Soldagem.
- Revisão e retrabalho.

Para que esses processos sejam bem sucedidos é necessário o controle de várias condições, dentre elas estão a precisão e repetitibilidade do posicionamento na inserção de componentes, a velocidade e a temperatura nas etapas de soldagem.

O objetivo principal deve trabalho é o desenvolvimento de um equipamento para levantamento das curvas de temperatura nos fornos e máquinas de solda automatizadas no processo de soldagem em indústrias eletrônicas.

O trabalho estará organizado da seguinte forma, no capítulo um será feita uma revisão sobre metalurgia, envolvendo metais, ligas e junções. No capítulo dois será explorado o

processo de montagem na indústria eletrônica sendo tecidos comentários sobre as etapas da montagem e possíveis problemas causados por más condições nos processos.

O capítulo três trata do projeto de hardware do equipamento proposto. A escolha dos circuitos, componentes e a programação serão os tópicos abordados.

O capítulo quatro tratará do software criado para a comunicação e processamento dos dados obtidos pelo dispositivo projetado. O capítulo cinco apresenta os testes e os resultados experimentais obtidos com o dispositivo.

Finalmente no capítulo seis são apresentadas as conclusões deste trabalho.

2. METALURGIA BÁSICA DAS SOLDAS

2.1. Introdução

Definem-se metais como elementos que possuem características distintas como brilho, dureza, maleabilidade, ductibilidade e que são usualmente bons condutores de calor e eletricidade, conforme [2]. Os átomos desses elementos são arranjados de formas definidas e em patrões repetitivos em todas as direções, o que leva ao surgimento de estruturas chamadas cristais.

Quando várias dessas estruturas são unidas é o marco para o surgimento de estruturas ditas policristalinas. A forma como essas estruturas se apresentam varia conforme as condições de resfriamento do metal líquido. Quando o metal líquido é resfriado vários núcleos cristalinos se formam e quando estes se encontram formam chamados grãos. Por causa do crescimento aleatório dos núcleos cristalinos os grãos têm configurações cristalinas imprevisíveis. Através dos processos metalúrgicos chamados recozimento é possível alterar as características dos grãos e das estruturas.

Usualmente a maioria dos metais não é usada em seus estados puros, segundo [1]. A combinação entre metais e de metais com outros elementos gera ligas, as quais apresentam características diferenciadas de seus elementos de origem. A maioria das ligas é formada através de processos de solução. Os metais são fundidos e quando em estado líquido, misturados. Os metais podem então dissolver-se um no outro, assim como água e álcool e formar uma liga de fase única. Nesse caso diz-se que os metais tem solubilidade total e cada parte analisada da mistura será igual.

No caso dos metais não se diluírem completamente é dito que a liga tem solubilidade parcial, como água e óleo. Logo partes diferentes da liga terão concentrações diferentes de cada metal. Ainda há a possibilidade de essa separação ocorrer durante o processo de resfriamento da liga, o que gera ligas diferentes conforme o processo de resfriamento aplicado à liga líquida. A esse fenômeno dá-se o nome de compostos

intermetálicos. Esses compostos são únicos e sua formação não segue qualquer regra de formação química, como outros materiais.

Os compostos intermetálicos podem ser definidos como materiais de fase única tendo uma pequena gama de componentes de simples proporções estequiométricas. Eles podem ser de natureza metálica ou de compostos de ligações iônicas.

2.2. Ligas

As ligas são composições de metais, ou metais e seus aditivos. Nas soldagens o elemento que realiza a fixação usualmente é uma liga. Essa liga precisa ser forte para que o processo seja bem sucedido.

Dois métodos são importantes no fortalecimento de ligas nas soldagens e principalmente nas soldagens que usam estanho-chumbo, a principal liga para soldagens na indústria eletrônica, sendo conhecidos como encruamento (ou endurecimento pela deformação a frio) e a adição de elementos na rede cristalina. O encruamento é o processo onde a rede cristalina é deformada por processos físicos, geralmente pressões extremas. Na adição pequenas quantidades de outros elementos são adicionadas a liga, criando imperfeições na rede cristalina do metal.

No caso do encruamento, se forem feitos tratamentos posteriores na liga, tal como o recozimento, o encruamento perde seu efeito, assim descrito em [2]. O recozimento é um processo usado para aliviar as tensões internas das redes cristalinas, fazendo com que voltem ao seu estado original. Esse processo é feito aquecendo-se o material a temperaturas adequadas e mantendo-a por certo período. Logo após o resfriamento do material é controlado. Para muitas ligas de soldagem a temperatura de recozimento está abaixo ou próxima das temperaturas ambiente. Assim, o encruamento torna-se ineficiente para o endurecimento das ligas de solda, novamente como descrito em [2].

Tal problema não existe quando feita a adição de elementos na rede cristalina. O recozimento e nenhum outro tratamento afetarão a dureza da liga. O fortalecimento das ligas é um efeito muito desejado para operações de soldagem, visto que as soldas podem unir peças de materiais iguais, o que poderia ser considerado uma imperfeição na rede cristalina do ponto de vista da peça final, ou unir materiais diferentes, formando então uma intrincada e complexa rede de uniões e camadas Intermetálicas.

Para que as características das ligas fiquem claramente representadas faz-se uso do chamado diagrama de fases, exibido na Figura 1. Esse diagrama representa uma liga comum nas soldagens de componentes eletrônicos que contém chumbo e estanho. O diagrama descreve o comportamento da liga em função da variação das quantidades de cada um dos componentes e da temperatura da mesma, como definido por [5].

Na região “A”, a liga encontra-se em estado sólido independente de sua composição. Na região “B”, a liga apresenta-se líquida. É visível que aqui a composição da liga influencia na temperatura de fusão.

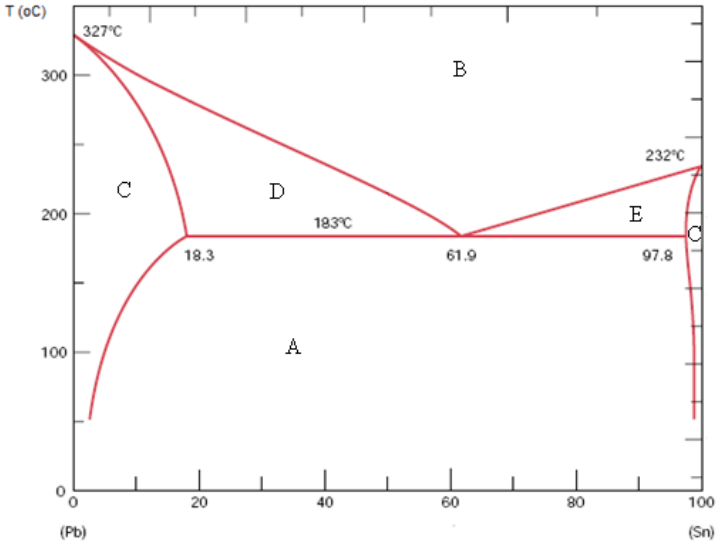


Figura 1 - Diagrama de fase Liga Estanho-Chumbo.

As áreas “C”, “D” e “E” são especiais. As áreas “C” demarcam o comportamento aproximado dos materiais em condições próximas de sua pureza. Nessas áreas quanto maior a temperatura e quando cruzando pela linha vermelha, encontram-se os pontos de fusão, 327 ºC para o chumbo e 232 ºC para o estanho. Nas áreas “D” e “E”, o material apresenta cristais sólidos flutuando no líquido do material em equilíbrio.

O ponto notável nesse diagrama ocorre para uma liga com composição aproximada de 62% de estanho e 38% de chumbo, na temperatura de 183ºC. Esse ponto define-se como ponto eutético e a liga como liga eutética. As principais características desse ponto são a baixa temperatura de fusão, abaixo de ambos componentes da liga separadamente, e a passagem completa do estado sólido para o estado líquido da liga, informação disponível em [2].

Os diagramas de fase das ligas podem apresentar mais de um ponto eutético, dependendo unicamente da liga analisada, e o material pode se apresentar como uma mistura com múltiplas fases, conforme [5].

2.3. Ligas Estanho-Chumbo

A humanidade conhece o estanho e o chumbo separadamente desde a antiguidade. Ambos são relativamente abundantes na natureza, porém associados a outros elementos.

O estanho é encontrado na natureza usualmente associado ao oxigênio, formando um mineral chamado cassiterita. Seu refino e processamento são relativamente simples, passando por estágios de flotação e purificação com adição de ácidos. A redução final, para aumento da concentração, dá-se através da queima com carvão em grandes fornalhas.

O chumbo apresenta-se na natureza comumente associado ao enxofre formando o mineral galena ou ainda quando também associado ao oxigênio forma-se sulfato de chumbo, porém também é possível sua associação com carbono gerando carbonato de chumbo. O processo para obtenção de

chumbo puro passa por etapas de fundição em fornalhas e posterior redução para retirada de impurezas.

As ligas formadas por esses metais são praticamente sinônimos de solda ao redor do planeta. Apesar de existirem muitas outras ligas e aditivos a ser acrescida nas mesmas a maior parte das ligas de solda em uso são de estanho-chumbo.

Conforme descrito anteriormente, a adição de outros elementos na liga altera algumas de suas características básicas. Nas ligas de estanho-chumbo os elementos mais comumente adicionados são cobre, bismuto, magnésio e zinco [2].

Para soldas na indústria eletrônica prefere-se o uso da liga eutética, porém há aplicações variadas em outras áreas, como hidráulica, mecânica geral e automotiva, iluminação e até mesmo para a produção de jóias, para uma variada gama de proporções na liga e de tipos de aditivos.

2.4. A Química dos Fluxos

Os fluxos são basicamente catalisadores das reações químicas envolvidas nos processos de soldagem. Merecem atenção especial nesse trabalho devido a sua grande necessidade prática de utilização [1].

As duas principais tarefas dos fluxos são limpar as superfícies a ser soldadas e mantê-las limpas, e atuar na tensão superficial dos metais a serem soldados de forma a garantir espalhamento uniforme da liga de solda.

Caso fosse possível garantir de outra forma a limpeza das superfícies e uma condição atmosférica satisfatória para soldagem, o fluxo se tornaria dispensável. Isso seria de grande valia para os processos pois os fluxos têm condições particulares de trabalho.

A maioria deles tem necessidade de controle da temperatura do ambiente para sua ativação (ou perfeita atuação) e desativação (tornar-se inócuo ao meio), eles são agentes ácidos que podem levar a danos futuros caso não sejam desativados apropriadamente e ainda podem ter efeito contrário, prejudicando a soldagem caso não sejam corretamente utilizados.

Assim como as ligas metálicas usadas nas soldas, os fluxos também são compostos. Partes integrantes dos fluxos são os veículos e os ativadores. Veículos são substâncias que diluem os ativadores e servem para facilitar sua distribuição. Um exemplo clássico de veículo nos fluxos é o álcool isopropílico, ou isopropanol.

Os ativadores são os agentes químicos propriamente ditos que farão o trabalho de limpeza da superfície e a manterão limpa até o momento de soldagem, segundo [2].

Para que tanto o veículo quanto os ativadores tenham desempenho ótimo é necessário que o fluxo atinja sua estabilidade térmica. A estabilidade térmica pode ser definida como a temperatura ideal para que o veículo do fluxo evapore e que o ativador tenha energia suficiente para realizar sua tarefa e logo após evaporar sem deixar resíduos na placa.

2.5. A Soldagem

Quando dois metais são unidos por um terceiro e é criada uma continuidade metálica tendo como resultado duas interfaces ditas intermetálicos, as quais são formadas pelo terceiro metal que adere aos outros dois metais, caracteriza-se uma soldagem. Na Figura 2 é ilustrada essa situação.

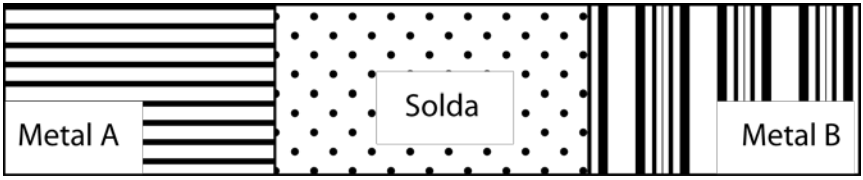


Figura 2 - Ilustração da definição de soldagem

Os metais A e B são ditos metais-base. São os metais que precisam ser unidos no processo. A solda é o material adesivo que une os metais-base. A região de fronteira entre os metais A e B e a solda é o lugar aonde são formados os intermetálicos.

A correta formação da zona de intermetálicos é o resultado de um processo de soldagem correto. Caso a zona de intermetálicos seja fracamente criada ela se torna o ponto fraco na união dos metais, resultando em baixa condutividade elétrica e baixa resistência mecânica.

Para que haja a criação correta da zona de intermetálicos é necessária a existência de parâmetros suficientemente corretos no momento da soldagem. A inexistência de contaminantes nas áreas a serem unidas, o equilíbrio termodinâmico dos metais e da solda e o equilíbrio das forças que atuam sobre a área são fatores que influenciam no processo.

Considerando a área livre de contaminantes (através do uso de fluxos adequados, atmosferas controladas ou quaisquer outros meios) e o ponto de equilíbrio termodinâmico atingido (não havendo troca de energia térmica entre o meio e os materiais) resta analisar as forças envolvidas no processo. Essas forças são geradas pelo atrito entre os diferentes materiais, a coesão dos próprios materiais e a forma das áreas a serem unidas.

Aqui cunha-se o termo “molhabilidade” (do inglês *wetting,* também definido em [1]), definido como a capacidade de espalhamento da solda fluida pelas áreas as quais deve unir. Quando as forças envolvidas no processo geram superam a tensão superficial do sistema a molhabilidade da área é total, resultando em grande superfície coberta pela solda.

2.6. Conclusão

Os tópicos envolvidos no processo metalúrgico da soldagem são de grande complexidade e com diversos parâmetros a serem controlados. Para a obtenção de uma correta soldagem é necessário que as áreas estejam limpas, com energia suficiente (temperatura adequada) e feitas de materiais próprios a possibilidade de união.

A questão envolvida pela energia disponível para o processo (calor) é de grande importância. O calor fornecido é usado para ativação dos fluxos usados na preparação das superfícies, na redução das tensões superficiais dos materiais e na obtenção do ponto eutético da liga de soldagem.

Os materiais corretos também devem ser empregados. Deve-se ter atenção aos metais-base a serem soldados e utilizar outro metal, ou liga, que possibilite a formação de um intermetálico adequado, com características elétricas e mecânicas condizentes com os metais base.

3. INDÚSTRIA ELETRÔNICA

A indústria eletrônica tem evoluído rápida e significativamente. Comparados aos seus primórdios, nas décadas de 60 e 70, a capacidade produtiva, os métodos utilizados e os tipos de bens industrializados muito evoluíram. Passou-se de uma indústria praticamente artesanal, com pequena demanda, de bens produzidos quase que individualmente para consumidores locais ou regionais, para uma escala de produção global.

As primeiras conexões eram feitas através do processo "*wire wrapping*". Consistia em pequenas hastes puntiformes as quais eram enrolados os fios que fariam as ligações para outras hastes e componentes, conforme padrões apresentados em [9].

A Figura 3 ilustra uma dessas conexões. Conforme a referência consultada, não é necessária a aplicação de solda afinal o fio é helicoidalmente pré-formado garantido conexão mecânica e elétrica estável.

Figura 3 - Exemplo de conexão "*Wire Wrapping*"

Essas conexões ainda são praticadas quando se visa uma montagem temporária para testes. Na Figura 4, temos um exemplo da utilização do "*wire wrapping*".

Junto com as técnicas "*wire wrapping*" iniciaram os componentes THT, acrônimo de "*Through Hole Technology*". Nessas conexões os componentes são soldados nas placas, usualmente no lado contrário ao seu posicionamento no substrato. Existem furos na placa pelos quais os terminais dos componentes passam, possibilitando assim sua soldagem no lado oposto.

Essa tecnologia possibilitou um maior povoamento das placas de circuito e melhor apresentação geral dos produtos. Apesar de estar em uso desde a década de 1950, atualmente ainda é uma tecnologia utilizada dada a difusão dos

componentes, o tamanho dos mesmos (facilita a substituição e o "*hobismo*" de eletrônica) e a não necessidade de maquinário específico para a montagem de lotes compostos de centenas de unidades.

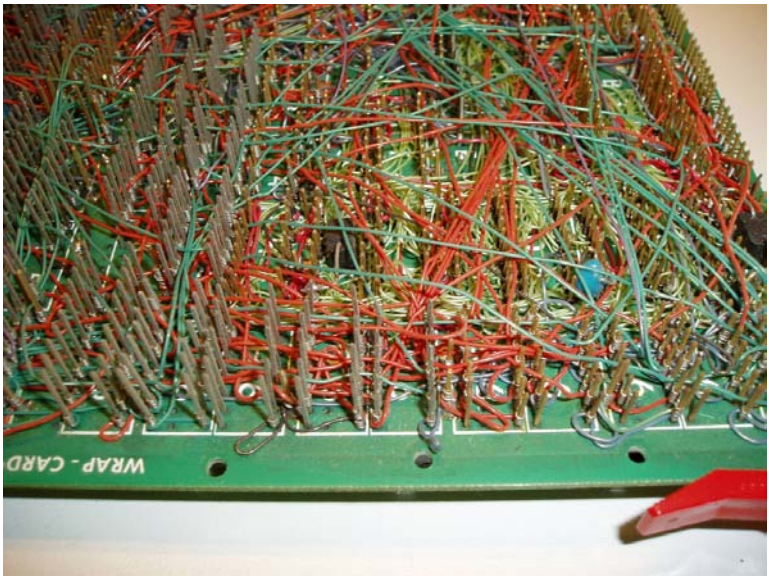


Figura 4 - Placa de um computador Z80 Europeu. Fonte: Wikipedia, "*Wire Wrap*".

As montagens baseadas em THT apesar de ter vários pontos a favor possuem também alguns problemas, tais como a baixa velocidade de montagem das placas apesar do uso de maquinário específico e dimensão limitada dos componentes (os padrões criados não possibilitam mais reduções). Para obtenção de menores dimensões a THT está sendo substituída desde os anos 70 pela SMT, "*Surface Mount Technology*".

Os projetos em SMT incluem componentes menores, com maior precisão de valores, características construtivas melhores (padronização aceita internacionalmente) e maior rapidez para inserção dos componentes nas placas, conforme [8]. Ao contrário do que ocorre com a THT, os componentes são soldados no mesmo lado onde estão posicionados na placa. Isso também possibilita maior densidade de componentes.

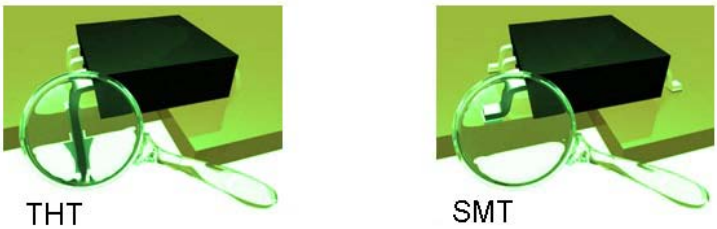


Figura 5 - Diferença entre montagens THT e SMT. Fonte: www.ebertech.com.br

Porém, seu uso demanda a utilização de maquinário específico para preparação das placas a serem soldadas, inserção de componentes, inspeção e retrabalho.

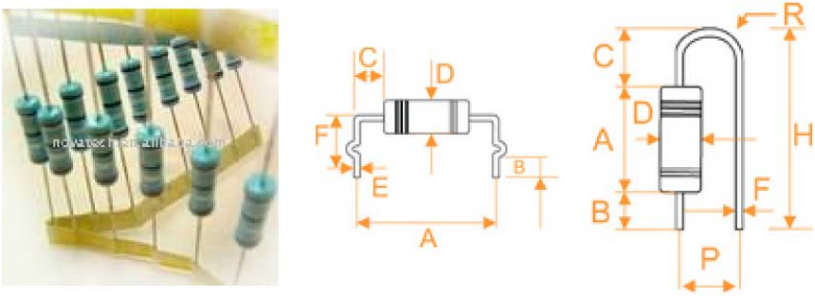
Além disso, tecnologias de dimensões cada vez menores começam a impossibilitar, ou ao menos dificultar, as montagens de "hobistas" e possíveis manutenções em equipamentos danificados.

Na Figura 5 é apresentada a diferença entre THT e SMT.

3.1. Preparação

Antes que a soldagem dos componentes ocorra vários processos são realizados a fim de garantir sucesso na montagem como um todo. A separação de componentes e posicionamento na linha de montagem (seja automática ou manual) é comum em qualquer tecnologia utilizada (THT ou SMT). A diferença aparece na apresentação e acondicionamento dos componentes de cada tecnologia.

Nas montagens THT manuais é usual os componentes serem pré-formados antes da apresentação na linha de montagem. Isso envolve a remoção dos componentes de seus carretéis de acondicionamento, conformação e ajuste do tamanho de seus terminais. Na Figura 6a é a apresentação padrão de resistores THT e nas Figura 6b e 6c exemplos de pré-formações.



(a) (b) (c)

Figura 6 - (a) Apresentação dos resistores em carretel. (b) e (c) Exemplos de pré-formação. Fonte: Omtec.

Essas pré-formações podem ser feitas com maquinário apropriado, no caso resistores, diodos, capacitores e demais componentes que sejam comercializados em fitas ou carretéis. Componentes comercializados a granel serão processados manualmente.

Nas montagens THT automatizadas e SMT não são necessárias essas pré-formações previamente. No caso da THT, a própria insersora realiza essa tarefa e no caso SMT realmente é desnecessário pois os componentes não possuem terminais e tem dimensões padronizadas, assumido por [10].

Outro componente da montagem que necessita cuidados são as placas de circuito impresso (PCI ou do inglês PCB, *Printed Circuit Board*). Dependendo do local de armazenamento é necessária uma limpeza da placa para facilitar a ação do fluxo durante a soldagem.

3.2. Montagem

A montagem é caracterizada pela inserção e soldagem dos componentes em um substrato. A inserção pode ser manual ou automatizada, dependendo da capacidade produtiva desejada. Feita a inserção, deve ocorrer a soldagem dos componentes ao substrato. Essa soldagem também pode ser manual ou automatizada.

Aqui serão apenas tratados os casos que envolvam as partes automatizadas do processo, mesmo que etapas manuais estejam envolvidas em outro período.

Uma vez preparados para a inserção, os componentes THT são movimentados para sua posição na PCI. Após a inserção automatizada as placas seguem para a inserção manual de componentes maiores, fora da faixa de trabalho das máquinas, e finalmente para a máquina de soldagem automática.

As placas montadas com SMT primeiramente passam pelo processo de deposição de pasta de solda ou deposição de adesivo. Tanto a pasta quanto o adesivo servirão como fixação mecânica do componente à placa, porém a pasta de solda ainda realiza a conexão elétrica entre trilha e componente.

Após a etapa de deposição os componentes são inseridos também pelas máquinas e após isso todo o conjunto segue para o forno de refusão, calibrado para fusão da pasta de solda ou cura do adesivo.

Após o forno, essas placas podem voltar à etapa de deposição para que mais componentes sejam posicionados no outro lado da PCI, e o processo se repete, adentrar uma linha de inserção de componentes THT, ou serem enviadas para a máquina de solda automática. Tudo isso dependendo da densidade e variedade de componentes que povoam a placa. Finalmente, após a inserção e soldagem de todos os componentes a placa montada é enviada para revisão e retrabalho caso necessário.

Esses são os passos, em linhas gerais, nas linhas de montagens de placas eletrônicas. Os equipamentos utilizados serão discriminados na próxima seção.

Tanto no processo THT quanto SMT, o fator temperatura é de vital importância. Seja no processo de cura do adesivo ou de fusão da pasta de solda em SMT, ou na rampa de aquecimento no processo de solda automatizada em THT é necessário o controle da quantidade de energia transferida em forma de calor.

3.3. Equipamentos

Como já citado, a primeira etapa da montagem SMT é a deposição de pasta ou adesivo sobre a PCI. Essa deposição pode ser feita por dispensadoras, ou dosadoras, ou por transferência através de estêncil. As dispensadoras usualmente são acopladas como módulos auxiliares nas máquinas de insersão de componentes.

Na Figura 7 está um exemplo de dispensadora. Na região destacada “a” está o depósito de pasta de solda e em “b” o bico de injeção do mecanismo.

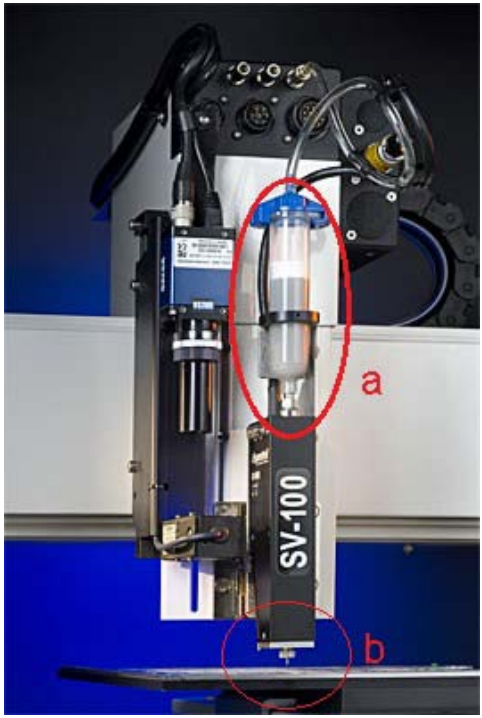


Figura 7 - Exemplo de dispensadora. Região “a” destaca o reservatório de pasta de solda (podendo ser substituído por adesivo). Região “b” destaca o bico de injeção. Fonte: ASYMTEK.

A maioria das máquinas de insersão trabalha baseada num sistema de eixos cartesianos onde são definidas as posições das ilhas (os contatos) de cada tipo de componente. De posse das coordenadas de cada ilha, a máquina deposita uma pequena quantidade de pasta, calibrada pelo operador da máquina, em cada ilha. Esse método tem pequenas desvantagens como a velocidade de deposição (cada ponto é depositado individualmente) e constante necessidade de renovação do reservatório (baixo volume disponível).

O método para deposição usando estêncil é comumente chamado de "*stencil printer*". O processo ocorre quando a pasta de solda ou adesivo é pressionado através de pequenas aberturas feitas numa tela apropriada. As aberturas correspondem aos pontos onde existem ilhas na placa, ou posição para deposição de adesivo. A Figura 8 ilustra um estêncil para essa finalidade.

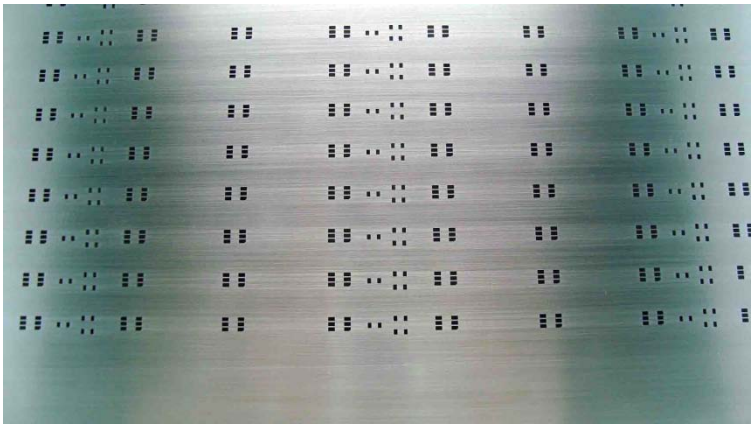


Figura 8 - Exemplo de estêncil usado para deposição. Fonte: Placompel.

O processo que faz uso do estêncil tem muitas semelhanças com as técnicas de "*silk screen*", usadas para decoração de camisetas e outras peças de vestuário. Na Figura 9é apresentada uma impressora semi-automática para uso com essa técnica de deposição.

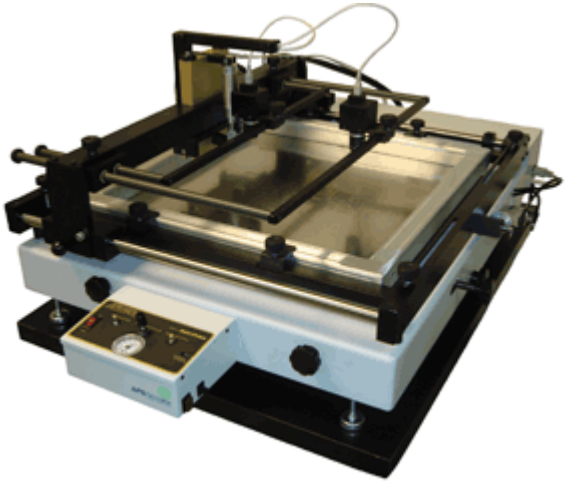


Figura 9 - Impressora semi-automática. Fonte: APS Gold.

Após a deposição de pasta ou de adesivo é feita a insersão dos componentes. As máquinas de insersão coletam os componentes de suas embalagens de acondicionamento e posicionam em lugares previamente indicados.

A remoção e inserção dos componentes são feitas com o uso de ações pneumáticas. Pressão negativa é gerada para remoção dos componentes de suas embalagens. Para seu posicionamento, essa pressão negativa é removida ou até mesmo invertida para uma pequena pressão positiva, de modo que os terminais do componente sejam englobados pela pasta depositada anteriormente ou que o corpo do mesmo tenha contato suficiente com o adesivo.

As posições são dadas por coordenadas, como num plano cartesiano, e dependendo da máquina pode ter redundância através de um sistema ótico. O sistema ótico funciona através do reconhecimento de padrões previamente configurados na máquina.

A Figura 10 traz alguns detalhes de um modelo de insersora para componentes SMT. A figura 10a ilustra a cabeça de insersão e no detalhe a câmera usada no sistema de realimentação ótico. Na Figura 10b uma visão geral da mesa de insersão.

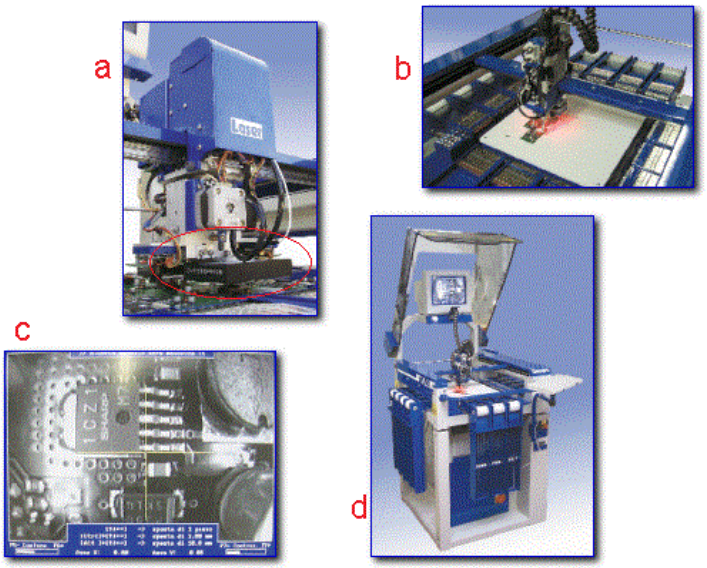


Figura 10 - Exemplo de máquina insersora SMT. Fonte: TWS.

Na Figura 10c é mostrada a tela do sistema de realimentação ótica e finalmente na Figura 10d o aspecto da máquina.

Completa a inserção dos componentes na placa, o conjunto segue para o forno de refusão, para cura do adesivo ou fusão da pasta de solda. Os fornos de refusão possuem algumas tecnologias diferentes empregadas para a transmissão de energia. São elas emissão de Infravermelho ou convecção de calor.

Na emissão de infravermelho, lâmpadas especiais são posicionadas dentro das câmaras do forno e configuradas a emitir doses quantificadas de radiação. Ao atingir o conjunto (placa com componentes) a radiação é absorvida e transformada em calor. O problema dessa técnica consiste na variedade de materiais utilizados no conjunto. Cobre, fibras e silício entre outros tem diferentes taxas de absorção de radiação, fazendo assim com que existam diferenças de temperatura na placa gerando “pontos quentes” que levam a maior estresse térmico os componentes e mesmo o substrato da placa.

Os fornos que trabalham com convecção de calor usam elementos resistivos e ventiladores para a transmissão de energia. Em cada câmara do forno existe um elemento e ventiladores para geração de um fluxo de ar que atinge a placa. Na Figura 11 exemplifica um forno de refusão.

Figura 11 - Exemplo de forno de refusão. Fonte: Madell Technology Corporation.

Para evitar o estresse térmico que ocorre nos fornos infravermelhos, os fornos por convecção possuem diversas áreas de pré-aquecimento e um área onde se atinge o pico de temperatura, que é a área onde ocorre a fusão da pasta de solda.

Após essa área, existem ainda zonas para resfriamento da placa. Tanto a recepção de muita energia (aquecimento) quanto à remoção da mesma (resfriamento) de forma muito abrupta geram o dito estresse térmico na placa. Por isso a necessidade de áreas de aquecimento e resfriamento nesse equipamento.

O processo também pode incluir algum componente THT, requisitando assim soldagem em outro tipo de maquinário. Assim, essa solda pode ser feita de modo manual, em tanque

estático (semi-automático ou automático) ou máquina de onda. Na Figura 12 é mostrada uma máquina de solda de onda.

Figura 12 - Máquina de solda automática. Fonte: Dentec.

3.4. Processos de Soldagem

Apresentadas as etapas da montagem e o principal maquinário utilizado para o processo globalmente, volta-se o foco para o processo de soldagem propriamente dito.

Como tratado no capítulo 1, a soldagem é um processo metalúrgico para união de dois ou mais metais fazendo uso de um terceiro elemento também metálico. Esse elemento pode ser uma liga ou metal puro. Para que o processo seja executado com sucesso é necessária a transmissão de energia suficiente para que o elemento que realizará a união possa criar camadas intermetálicas com os metais base (que são os metais a serem unidos) fortes o suficiente, comparáveis aos metais base.

A transmissão de energia ocorre através do aquecimento das partes envolvidas no processo, metais base e elemento de

solda. O maquinário apresentado na sessão anterior necessita controlar a temperatura e velocidade com as quais o conjunto placa e componentes passa por cada uma de suas áreas. Tanto os fornos de refusão quanto as máquinas de solda possuem áreas de pré-aquecimento para que a energia seja transferida de forma gradual sem gerar estresse para a placa e para os componentes.

Nos processos SMT, a pasta de solda depositada previamente contém os elementos necessários para a soldagem. Os principais elementos são o fluxo e a liga metálica existente. Existem vários tipos de pasta, definidos pela granulação, composição da liga metálica e tipo de fluxo. Para cada tipo de pasta o fabricante define métodos de armazenagem e preparação para deposição. Em linhas gerais as pastas devem ser armazenadas refrigeradas e a preparação envolve re-aquecimento lento (acima de 8 horas dependendo do fabricante) e a re-mixagem da pasta.

Na soldagem THT por máquina fluxo e liga estão separados. Ao adentrar a máquina de solda, a primeira estação aplica o fluxo. Essa aplicação pode ser por spray ou com espuma. Na aplicação por spray, pequenos bicos de injeção pulverizam o fluxo diretamente contra a placa. Na aplicação com espuma o fluxo é aerado a ponto de formar uma espuma densa o suficiente para espalhar o mesmo na placa. A Figura 13 traz uma ilustração da aplicação em espuma.

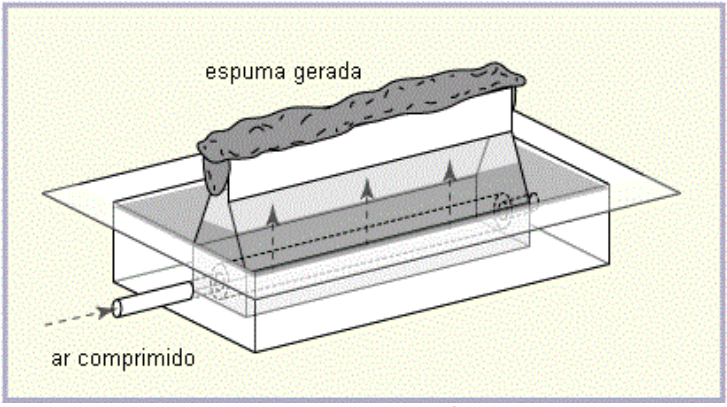


Figura 13 - Aplicador de fluxo através de espuma. Fonte: www.ami.ac.uk

Ambas a tecnologias, SMT e THT, precisam de aquecimento gradual como já citado. Para isso os fornos de refusão e as máquinas de soldagem têm várias áreas de pré-aquecimento. Não só apenas a temperatura deve ser considerada, mas também o tempo de exposição da placa em cada área, configurado pela velocidade com a qual a mesma atravessa a área. Na Figura 14 é apresentado um modelo idealizado da curva de temperatura do forno de refusão.

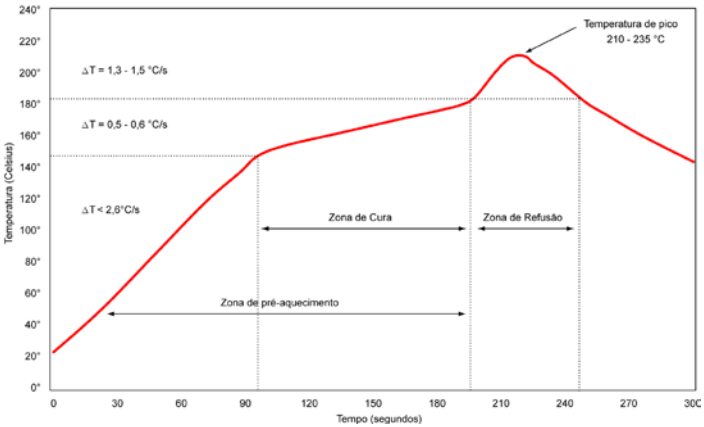


Figura 14 - Exemplo idealizado de curva de aquecimento de forno de refusão.

Para as máquinas de solda THT o perfil da curva é praticamente o mesmo, porém devem ser feitas leituras conforme a placa a ser soldada, uma vez que a densidade de componentes na placa tem influência direta na absorção de energia.

Cada uma das áreas tem suas funções e necessidades. A “Zona de pré-aquecimento” tem a função de transmitir a maior parte da energia necessária para o conjunto a ser soldado. Porém, sem excesso, pois também é necessário tempo para a atuação do fluxo e o mesmo evaporar. Caso isso não seja levado em consideração, o fluxo pode não ter tempo de limpar os contatos e facilitar a molhabilidade da solda, gerando soldas ruins. O fluxo também pode formar bolhas e explodir, deslocando os componentes.

A “Zona de Cura” tem esse nome pois é nessa área que ocorre a evaporação do fluxo ou o início da cura do adesivo, quando do uso desses. Um detalhe sobre o uso de adesivos é que sua curva é mais plana no topo. Para os adesivos não é necessário um pico de energia (calor), apenas que se mantenha um nível de temperatura por um período determinado pelo fabricante para a total cura do adesivo.

Considerando que o fluxo foi devidamente espalhando e ativado pelo calor, chega-se a “Zona de Refusão”. Aqui um pico de energia é dado para a pasta, fazendo com que a o metal da liga funda e forme as camadas intermetálicas, realizando a soldagem propriamente dita. Na Figura 15é exibido um perfil de solda ideal.

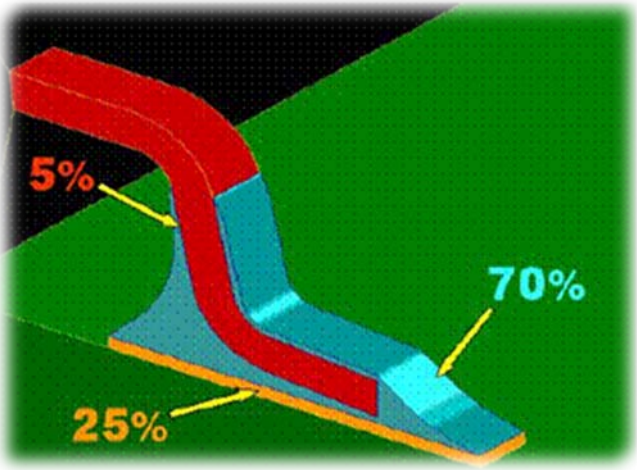


Figura 15 - Perfil de solda ideal quando a deposição de material da liga.
Fonte: http://smd-on-line.com.

A “Zona de Refusão” nos processos que envolvem THT ou soldagem por máquina de onda é a zona onde há o contato do conjunto placa e componentes com a liga já fundida. Esse contato pode se dar de três meios, onda plana, onda dupla ou soldagem seletiva.

O sistema de onda simples é exibido na Figura 16. Uma bomba é usada para impulsionar a liga já liquefeita para cima e para dentro de um bocal instalado no tanque de solda. Esse sistema gera uma onda controlada da liga fundida, comumente utilizado quando somente são utilizados componentes THT.

A parte da onda que se desloca contra o movimento da placa tende a ajudar a soldagem, forçando o contato entre os pontos de solda e a liga. A solda que flui no mesmo sentido que a placa ajuda a remover excessos de solda dos terminais e ainda empurra as oxidações de volta para a superfície do tanque.

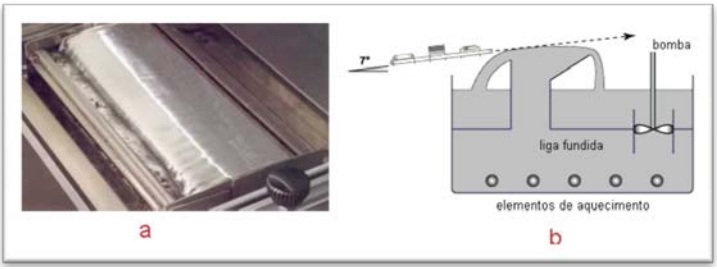


Figura 16 - Sistema de soldagem de onda plana, também conhecido como onda simples ou onda lambda. (a) Foto e (b) Corte explicativo. Fonte: www.ami.ac.uk.

O sistema de dupla onda funciona basicamente pelo mesmo princípio que o de onda plana. Bombas impelem a liga fundida para dentro de bocais na superfície do tanque. A Figura 17 apresenta o sistema. Esse sistema introduz uma onda com superfície turbulenta, ideal para aplicações onde os componentes SMT são colados no lado de solda da placa.

A onda turbulenta tem maior energia cinética que a onda plana, assim consegue impulsionar gotas de material em “pontos cegos” da onda plana. Essa por sua vez permanece nesse sistema com as mesmas funções do sistema de onda simples.

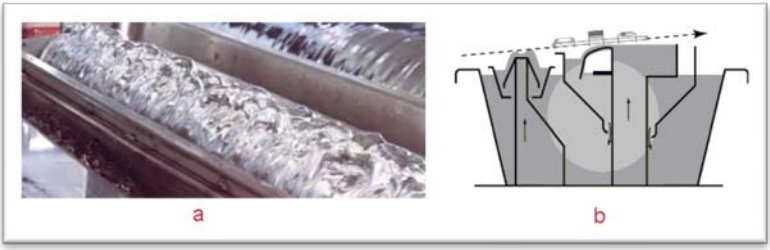


Figura 17 - Sistema Dupla Onda. (a) Foto e (b) Corte explicativo. Fonte: www.ami.ac.uk.

A soldagem seletiva é utilizada quando existem componentes sensíveis ao possível estresse térmico gerado pelos métodos anteriores e a quantidade de pontos que

necessitam soldagem é pequena. A Figura 18 ilustra esse sistema.

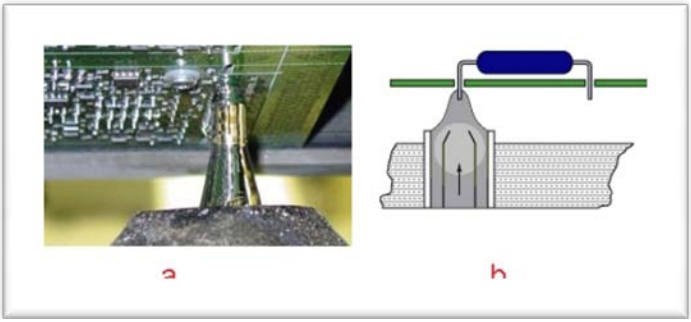


Figura 18 - Sistema de solda seletiva. Fonte: Saline Lectronics.

Nesse sistema um pequeno bocal se desloca, abaixo da placa, pelos pontos a serem soldados. A liga fundida é também impulsionada por bombas através do bocal. Na figura 18a uma máquina em atividade e n figura 18b um corte explicativo do sistema.

3.5. Revisão e Pós-Processamento

Após o processo de montagem é comum que exista uma etapa de revisão para combater possíveis falhas no processo. Os principais defeitos encontrados no processo com SMT são curtos entre terminais, falha na refusão, vazios de solda, solda granulada, componentes levantados, trincas ou fissuras na solda e aparecimento de bolas de solda. A figura 19 traz exemplos desses defeitos.

Os curtos entre terminais podem ter várias origens, tais como alinhamento deficiente entre estêncil e PCI, estêncil com resíduos de pasta de solda, parâmetros de Impressão incorretos, excesso de pasta de solda na abertura do estêncil, escorrimento da pasta de solda, terminais dos componentes desalinhados ou avariados, posicionamento do componente, ilhas sub dimensionados.

As falhas na refusão são dadas principalmente por qualidade da pasta de solda comprometida, excesso de oxidação nas ilhas e componentes e perfil térmico incorreto.

Os vazios de solda têm principais causas perfil térmico incorreto, especialmente no pré-aquecimento e pré-refusão. Não ocorre uma evaporação correta do solvente do fluxo e seleção incorreta da pasta de solda.

O aparecimento de solda granulada após a refusão deve-se a má qualidade da PCI, movimentos durante o resfriamento (trepidação de esteira transportadora) ou quantidade insuficiente de fluxo.

Os componentes levantados e Tombstoning são frequentes quando o depósito de pasta de solda é irregular, ou seja, existem variações da pressão aplicada durante impressão.

Outras causas comuns são vibração durante o resfriamento, tempo muito longo acima da fase líquida, durante a refusão, ilhas usadas como dissipadores térmicos, terminais contaminados ou oxidados, um lado do componente é soldado antes do outro, geometria desigual das ilhas, as ilhas e/ou terminais estão desnivelados.

A soldagem inconstante, a máscara de solda está avançando sobre a ilha - diminuindo assim a superfície de soldagem ou a máscara de solda está sob o componente também são motivos para aparecimento de componentes levantados.

As fissuras, ou trincas na soldas, aparecem quando o perfil térmico é incorreto – a taxa de aquecimento do componente é muito íngreme, a temperatura do componente é muito alta, a taxa de resfriamento é muito longa ou ocorre crescimento exagerado da camada intermetálica.

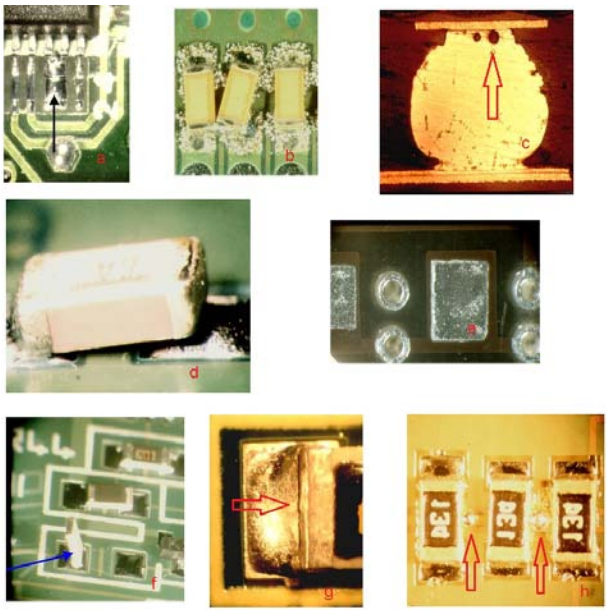


Figura 19 - Falhas comuns no processo de soldagem SMT. (a) Curto entre terminais, (b) Falha na refusão, (c) Vazio de solda, (d) Componentes levantados, (e) Solda granulada, (f) *Tombstoning* – Componente levantado na horizontal, (g) Trincas e (h) Bolas de solda. Fonte: http://smd-on-line.com.

Os principais problemas nos processos envolvendo THT são a falha na solda por inexistência de solda, curtos entre terminais e levantamento dos componentes (o terminal sai do seu furo). Os principais motivos são o perfil térmico incorreto, oxidações severas nas ilhas e/ou terminais e projetos mal executados.

Todos esses problemas devem ser eliminados por ações corretivas em diversas etapas do processo. Porém uma pequena quantidade é aceita por norma, devendo então ser retrabalhada através de equipamentos adequados. O tradicional ferro de solda, acompanhado de estações de soldagem controlada, microscópios, lentes de aumento, estação de *reballing* – no caso de componentes BGA - e máquinas de soldagem seletiva são essenciais nessa tarefa.

As informações contidas nessa seção foram extraídas principalmente de [8] e [10]

3.6. Conclusão

A indústria de montagem eletrônica avança conforme a demanda do mercado por equipamentos eletrônicos aumenta. E nos tempos atuais, nunca essa demanda esteve tão grande. A proliferação de “*gadgets*” de todos os tipos na última década e a modernização de processos de fabricação nas décadas anteriores foram determinantes para que esses produtos pudessem se espalhar pelo globo.

O aumento da automação nos processos de montagem foi fator essencial para que tudo isso acontecesse. Porém junto com a automação vieram problemas que necessitaram estudo e trabalho para estabilização desses processos.

4. PROJETO DE HARDWARE

Nesse capítulo encontra-se o foco do trabalho, a determinação e criação de um equipamento para rastreamento das curvas de temperatura das máquinas de soldagem na indústria de montagem eletrônica.

O requisito básico para esse projeto é a facilidade, praticidade e robustez do equipamento aliados a um custo compatível para o mercado a que se destina.

Tendo em vista equipamentos semelhantes importados, a variedade de configurações disponíveis é grande. Equipamentos com capacidade de avaliação da curva de temperatura diretamente no equipamento, transferência de dados *on-line* para um computador, possibilidade para até 25 sondas, capacidade para avaliação de períodos de até 24h são algumas características disponíveis.

O equipamento aqui apresentado conta com as seguintes características:

1. Capacidade para até quatro sondas de temperatura operando simultaneamente.
2. Capacidade de armazenamento de dados de temperaturas limitada em 10 minutos por sonda, ou 4800 pontos, não volátil até a próxima varredura.
3. Amostragem de temperatura a cada 0,5 s com precisão de 0,25 °C.
4. Uso de sondas tipo Termopar K.
5. Capacidade de comunicação com um PC via interface USB.

4.1. Eescolha de Circuitos e Definição de Componentes

Inicia-se a determinação dos circuitos através do tratamento dos sinais oriundos das sondas de temperatura. Conhecida a temperatura do ponto eutético da liga de estanho-chumbo usada no processo como 183 °C, limita-se o número de sondas disponíveis para essa aplicação.

Sondas baseadas em semicondutores (NTC, PTC e demais sensores, a exemplo LM35) serão eliminadas, pois nessa faixa de temperatura existe a degradação das junções dos semicondutores, gerando leituras incoerentes.

As sondas recomendadas para aferição de temperaturas nessa faixa são os termopares. Essas sondas baseiam-se no efeito Seebeck, o qual explica que uma pequena variação de tensão surge quando existe variação de temperatura em uma junção de metais (junção bimetálica). O termopar mais difundido é do tipo K (cromel/alumel).

Sabe-se que as sondas Termopar K têm faixa de atuação de -200 °C até 1200 °C são de baixo custo relativo e têm comportamento definido. A tensão em seus terminais varia na ordem de 48 µV/°C, assim esse sinal precisa ser tratado e amplificado, de modo a conter a informação de temperatura correta.

Comparados a outros tipos de sensores os termopares tem uma grande faixa de atuação, porém sua exatidão é pequena. Na faixa de medição entre -200 °C e 800 °C a exatidão é de apenas 1°C. Acima dessa temperatura pode-se ter erros de até 7,5 °C. Apesar da pequena faixa de operação (-55 °C a 150°C) o sensor LM35 tem exatidão de no mínimo 0,4 °C. Ainda há o efeito de junção fria, ligado ao efeito bimetálico entre conectores, trilhas e conectores do termopar na placa.

Para compensar esse efeito e facilitar a leitura das temperaturas será usado o circuito integrado MAX6675 da Maxis®, cujas funções incluem a leitura, discretização e digitalização da temperatura das sondas com precisão de 0,25 °C, compensação de junção fria e tratativa dos dados serial (SPI). Apesar do alto custo desses CI, sua utilização facilita o roteamento da placa, a tratativa de hardware e software para compensação da junção fria e praticamente equivale ao custo quando do uso de amplificadores de instrumentação, a exemplo INA129, para tratamento do sinal dos termopares ou de CI similares, tal como o AD595.

De acordo com os requisitos de hardware listados para o armazenamento dos dados de quatro sondas no tempo máximo de 10 minutos, sendo que cada temperatura coletada é descrita por um byte, são necessários 4800 endereços de um byte cada. Para armazenamento dos dados recolhidos será usada uma

memória com comunicação SPI de 128K x 8 bits, AT25128B da Atmel®.

A interface USB será criada pelo integrado FT232RL da FTDI®. Esse componente conecta-se a um controlador através do protocolo UART e emula no PC uma porta serial. Desse modo toda a complexidade do protocolo USB fica a cargo do próprio FT232RL e a comunicação passa a ser serial entre o controlador e o PC. Esse componente também traz a vantagem do uso de poucos componentes e a presença de um regulador de baixa queda (*low drop voltage*) interno possibilitando a alimentação de todos os circuitos através da porta USB.

O controlador utilizado será o ATMEGA8-16P, também da Atmel®. Esse controlador de 8bits tem os recursos necessários para a aplicação (capacidade de processamento, comunicação SPI, comunicação UART), aliado a um custo baixo e grande disponibilidade de bibliotecas de programação.

Na Figura 20 é exibido um diagrama de blocos do hardware a ser criado. A existência do barramento SPI facilita o roteamento das trilhas.

4.2. Esquemáticos

O esquemático do projeto foi feito na suíte de desenvolvimento PROTEUS, que engloba o gerador de esquemáticos ISIS e o "*CAD*" ARES para desenvolvimento de PCI´s, está disponível no anexo 1. Foram levados em consideração vários aspectos industriais na confecção do protótipo, como custo de inserção de componentes, posicionamento de componentes para facilitar a revisão da solda, dimensões da PCI entre outros.

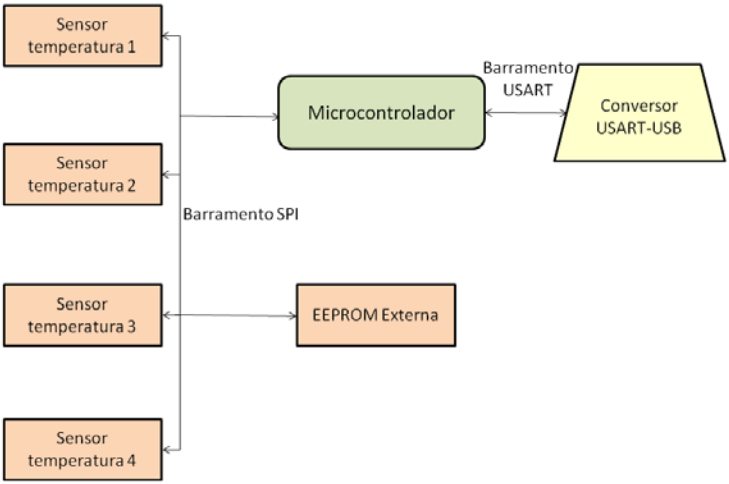


Figura 20 - Diagrama de blocos do hardware.

O uso de barramentos diminuiu a quantidade de pinos usados do microcontrolador e deixou o esquemático simples. A o barramento de comunicação SPI é dotada de filtros de alta frequência e de proteções de sobrecorrente em todos os pinos. A ligação USART entre o microcontrolador e o FT232RL e entre este e o conector USB é feita conforme recomendação do fabricante.

A alimentação para esse equipamento deve possibilitar sua mobilidade. Calculando o consumo em suas piores condições têm-se os resultados exibidos na tabela 1.

Tabela 1 – Consumo nas piores condições (Vcc=3,0 V)

Componente	Quantidade	Consumo unitário	Consumo total
Atmega8P-16P	1	48 mA	48 mA
MAX6675	4	50 mA	200 mA
FT232RL	1	24 mA	24 mA
AT25128	1	46 mA	46 mA
		Total	318 mA

Esse total é pouco superior ao comumente fornecido pelas baterias CR-2032 (uso comum para manter dados das BIOS de placas-mãe). Pela facilidade e disponibilidade usaram-se no protótipo duas pilhas AA.

4.3. Roteamento e Desenho da Placa de Circuito Impresso

Considerando que o protótipo deve suportar as temperaturas elevadas atingidas dentro dos fornos de refusão ou das máquinas de solda, uma proteção envolverá todo o hardware. Assim o posicionamento dos componentes tais como o conector USB, os interruptores de configuração e os conectores dos termopares são críticos.

As trilhas foram dimensionadas de modo que a máquina que confeccionou o protótipo pudesse operar fora de sua região crítica, onde não existe precisão para confecção das mesmas.

A proximidade entre os conectores USB e o integrado FT232RL e dos conectores dos termopares a seus respectivos MAX6675 foi considerada. Quanto menores essas trilhas, menor a incidência de ruídos na medição de temperatura e na comunicação USB. Também as trilhas de comunicação SPI e USART foram desenhadas para que a incidência de ruído fosse minimizada. O desenho final da placa após o roteamento está disponível no anexo 2.

Na Figura 21 é exibida uma foto do protótipo montado.

4.4. Proteção do Protótipo

Considerando o ambiente extremo (temperaturas acima de 150 ºC) onde o protótipo será testado é necessária a utilização de uma proteção. Para essa finalidade foi usada uma caixa metálica de inox e uma camada de manta anti-chama isolante térmica.

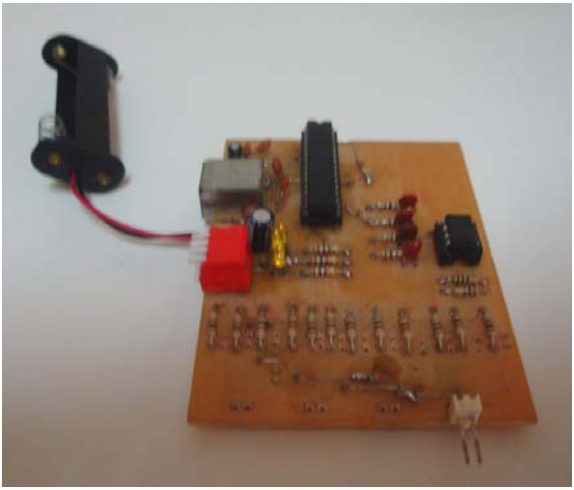


Figura 21 - Foto do protótipo montado.

Caso essas proteções não sejam usadas as baterias alcalinas, que são em sua maioria projetadas para trabalhar em temperaturas baixas (-18 a 55 ºC), sofrerão danos podendo explodir ou vazar. Também as soldas do próprio protótipo sofrem processos de cristalização, o que pode levar ao surgimento de funcionamento incorreto do mesmo.

A caixa de inox foi preferida dada sua não emissão de resíduos em altas temperaturas, não gerando contaminação na liga de solda. A manta isolante foi usada para isolação da temperatura. Na Figura 22 são exibidas as proteções utilizadas.

Figura 22 - Caixa inox e manta isolante utilizada como proteção térmica.

4.5. **Desenvolvimento de Firmware**

O firmware para o hardware descrito acima foi desenvolvido no programa AVR Studio 5.0. Esse software é gratuito e possibilita a programação dos microcontroladores em linguagem C (o código está disponível no anexo 3). Várias partes do código são originadas em exemplos de códigos disponibilizadas pela fabricante do microcontrolador. O fluxograma apresentado na Figura 23 representa o código do firmware.

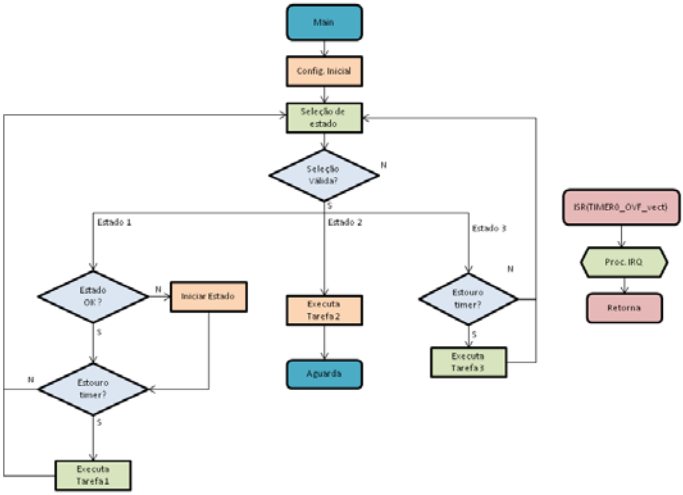


Figura 23 - Fluxograma do firmware.

O código começa com a definição de entradas e saídas das portas e configuração das comunicações USART e SPI. Ficaram definidas como saídas os pinos conectados aos LED´s, que serão usados como indicadores dos estados correntes e de alertas de funcionamento e como entradas os pinos destinados às configurações dos estados.

A porta USART foi configurada com "*baud rate"* de 9600 kbps, 8 bits, sem paridade e sem bit de parada. A ausência de bits de controle é compensada pela criação de um protocolo de comunicação que faz o controle dos dados transmitidos.

A comunicação SPI foi definida com o controlador como mestre do canal, com frequência de 62500 Hz (f_{osc}/16). Com essa frequência, tanto a memória quanto os sensores de temperatura e o controlador operam longe de seus limites máximos na temporização da comunicação, consumindo assim menos recursos do controlador e energia das baterias.

Assim, foram definidas três configurações de trabalho possíveis frente as posições das chaves de configuração:

- Estado 1: leitura da temperatura dos sensores e armazenamento individual das mesmas.
- Estado 2: transferência dos dados armazenados.
- Estado 3: leitura da temperatura e transferência imediata, sem armazenamento.

Antes das etapas onde as leituras são executadas, são feitas verificações de quantas sondas estão conectadas na placa. Assim, são utilizados menos espaços na memória e menos dados são trafegados, tanto na placa, quanto para o computador.

Os estados 1 e 2 foram desenvolvidos para o uso principal do protótipo, imaginando seu uso corriqueiro como produto final. Assim suposto, fez-se desses estados um código suficiente para seu funcionamento sem necessidade de interações constantes com o usuário final.

O estado 3 foi proposto como um estágio para calibração e testes, podendo ser usado em outras funções que não as principais descritas neste trabalho.

O código composto pensando na modularização das tarefas. Tarefas que são recorrentes e fundamentais para o funcionamento do produto, tais como o envio de dados através da USART ou SPI, foram transformadas em funções específicas e separadas, porém continuam compartilhando funções ainda mais elementares como configuração de registradores de comunicação.

4.6. Comunicação

Para a comunicação entre a placa protótipo e o computador foi criado um protocolo de comunicação que tem por objetivos facilitar a interpretação dos dados pelo software do computador e garantir a integridade dos dados transmitidos sem que a complexidade do protocolo impedisse seu uso no microcontrolador.

Assim, foi criada uma estrutura de três bytes ditos “A”, “B” e CRC. Os bytes “A” e “B” inicialmente contém o número de sensores em atividade e a quantidade de amostras feitas por cada um dos sensores. Essa informação é chamada de “header”.

Após a transmissão do “header” os bytes “A” e “B” passam a conter os 12 bits de informação da temperatura lida pelos conversores MAX6675 e informações sobre a qual sensor de temperatura a leitura em transmissão pertence.

O byte CRC é uma informação gerada através de operações conhecidas baseada nos bytes “A” e “B”. Ele tem por função criar um valor que represente de alguma forma os bytes “A” e “B”. Dessa maneira o receptor dos dados (no caso o computador) pode verificar a integridade dos dados transmitidos refazendo a operação conhecida com os bytes recebidos e comparando o valor calculado com o CRC também recebido. Se os valores forem iguais é menor a probabilidade que os dados recebidos estejam corrompidos.

O algoritmo para cálculo do CRC foi desenvolvido baseado nas informações disponíveis em [12] e é exibido no anexo 3, dentro da função “CRC”.

4.7. Custos

Um dos requisitos do projeto é que o mesmo tenha baixo custo em relação aos equipamentos já disponíveis. Um levantamento feito no mercado mostrou um preço médio de US$ 800 para equipamentos com funções similares as aqui propostas.

A tabela 2 exibe os valores para os componentes utilizados no protótipo, sendo os valores obtidos através de consultas “online” em distribuidor.

Tabela 2 – Valores dos componentes para o protótipo

Componente	Custo Unitário (US$)	Custo Total (US$)
MAXIM MAX6675	6,82	27,28
FTDI FT232RL	2,65	2,65
Atmel ATMEGA8-16P	3,66	3,66
Atmel AT25128B	0,86	0,86
Conector USB tipo B	1,08	1,08
Conjunto 4 chaves DIP	1,79	1,79
Porta-pilha	1,8	1,8
Miscelânia (Resistores, LEDs, capacitores, conectores genéricos)	5	5
Termopar tipo K (400 °C)	8,3	33,2
PCI	13,8	13,8
Caixa Inox	31,66	31,66
Manta termo-isolante	12,20	12,20
	Total	135,12

Os valores exibidos são referentes a compras em grandes quantidades (embalagens fechadas de componentes) variando de 1000 a 4000 unidades. O custo da PCI é apenas da placa de cobre nu, não está incluso o custo de usinagem do protótipo.

O custo do termopar, da caixa inox e da manta termo–isolante foram obtidos em lojas da região em moeda local (Real) e convertidos para dólar, com taxa de câmbio de 1,8.

4.8. Conclusão

O uso de termopares, especificamente os tipo K, atende aos requisitos do projeto plenamente. Sua faixa de operação de temperatura é compatível com o uso e seu custo é baixo quando comparado a outros tipos de sondas de temperatura.

A utilização dos CIs MAX6675 faz com que a tratativa do sinal do termopar seja facilitada. O próprio circuito integrado amplifica o sinal, discretiza, quantifica, faz a compensação de junta fria e transmite em protocolo SPI. Tudo isso num único componente de tamanho reduzido, o que auxilia a obtenção de um desenho de placa compacto.

A memória EEPROM externa também utiliza protocolo SPI em sua comunicação, algo extremamente conveniente uma vez que esse protocolo já está em uso e pode ser facilmente expansível para novos componentes.

O uso do controlador Atmel Atmega8-16P é conveniente por tratar-se de dispositivo bem difundido e fácil programação, possuir o hardware embarcado necessário (comunicação SPI e USART) e ter espaço em memória de programa suficiente para a aplicação.

5. PROJETO DO SOFTWARE

Para que os dados recolhidos pelo equipamento sejam úteis são necessários meios que possibilitem sua visualização e avaliação. Assim, foi proposta a criação de um software que recebe os dados recolhidos pelo produto, faz uma tratativa dos mesmos e os exibe de maneira que estejam claros e de fácil percepção.

Tal software tem como pré-requisitos:

- Possibilidade de retenção dos dados recebidos.
- Disponibilizar a visualização das temperaturas em gráficos de linha.
- Calcular o gradiente de temperatura dados os intervalos das amostras.
- Exibir a temperatura máxima registrada e o tempo de exposição.

Além disso, é necessário que o software seja de fácil instalação e uso, abranja grande gama de plataformas na qual possa ser instalado e seja gratuito. Para esses requisitos foi escolhida a linguagem de programação Microsoft® C#, através do ambiente de programação Visual Studio 2010 Express®. Dessa forma consegue-se a facilidade de instalação e abrangência necessárias e a gratuidade devido a não necessidade de pagamento sobre a plataforma utilizada.

5.1. Programação

A linguagem C# tem sua origem na linguagem C, amplamente difundida, e teve em seus degraus de evolução a linguagem C++.Assim como a linguagem C++, a linguagem C# também é orientada a objetos e apresenta diferentemente da C++, a programação orientada a eventos. A orientação a eventos na programação é o equivalente ao tratamento de interrupções na programação de hardwares. Isso simplifica a programação,

uma vez que para pequenas tratativas os códigos se tornam enxutos.

Além das vantagens supracitadas, a linguagem C# apresenta ainda:

- Suporte total a plataforma Microsoft .NET® (suporte a ferramentas de internet).
- Maior eficácia do "*Garbage* Colector" (limpa a memória).
- Facilidade de interação com Javascript.

Porém, por ser não ser uma plataforma compilada e sim interpretada, tarefas executadas por um programa escrito em C# tem um tempo de execução maior quando comparado ao mesmo programa escrito em C++. Além disso, algumas bibliotecas e programas antigos podem necessitar de reengenharia.

A exibição do código completo é desnecessária e não trará vantagem devido às características da linguagem, como heranças de classes e polimorfismos entre as mesmas, e por se tratar de programação visual grande parte do conteúdo é destinado a criação do ambiente do programa. Porém o anexo 4 traz as principais classes utilizadas.

Assim como no firmware desenvolvido para o protótipo, a comunicação recebeu cuidados especiais, sendo o protocolo desenvolvido também aqui qplicado. Usando a vantagem da programação orientada a eventos, o recebimento dos dados ficou atrelado a um evento. Assim, quando um dado é identificado na porta serial é disparado o evento "*DataReceived*", o qual recebe os dados através da classe "*SerialPort*".

Os dados recebidos são tratados em uma máquina de estados, a qual toma as decisões sobre o arranjo e informações contidas no mesmo. A máquina de estados em questão é a classe "*StateMachine*".

Caso o pacote recebido for o cabeçalho do protocolo, "*header*", é disparado um novo evento chamado "*OnHeaderReceived*", o qual retira do pacote os dados referentes ao número de amostras e quantidade de sensores.

Se o pacote recebido for uma amostra o evento é disparado, chamado "*OnTemperatureReceived*". Esse evento trata o pacote retirando as informações sobre qual sensor está enviando o dado e qual a temperatura registrada. Junto com

essa tratativa da comunicação, esse evento também calcula as estatísticas pertinentes até o instante do recebimento da amostra.

Quando todas as amostras forem recebidas, dispara-se o evento “OnReceptionCompleted”. Esse evento indica que todas as amostras previstas no cabeçalho foram recebidas e encerra a comunicação, disponibilizando os dados para visualização com o desenho de um gráfico e exibição das estatísticas

O processo de comunicação começa quando o usuário clica no botão “Iniciar”, na tela principal do programa. Isso dispara a ação de carregar a biblioteca de comunicação, abrindo a porta serial selecionada e ficar no aguardo do primeiro evento “*DataReceived*”.

Para o usuário as informações sobre a comunicação são informadas na tela principal do programa. Uma vez que um “*header*” válido é recebido, o espaço onde o gráfico e as estatísticas se localizam é limpo e fica no aguardo da finalização da comunicação.

5.2. Resultados

A tela principal do programa é exibida na Figura 24. No canto superior esquerdo são disponibilizadas as configurações da porta serial necessárias para o funcionamento da comunicação. São vistas caixas de seleção para o nome da porta, “*baud rate*”, o tamanho do quadro, a paridade, presença de bit de parada e “*timeout*” (parada por falta de recebimento de dados).

Apesar de todas essas opções estarem disponíveis na primeira versão do protótipo, alvo desse trabalho, conta apenas com a comunicação a 9600 kbps, quadro de 8 bits, sem paridade e sem bit de parada.

O nome da porta e o “*timeout*” são definidos pelo usuário conforme indicação do sistema operacional sobre qual porta foi configurada para o chip de comunicação utilizado no protótipo.

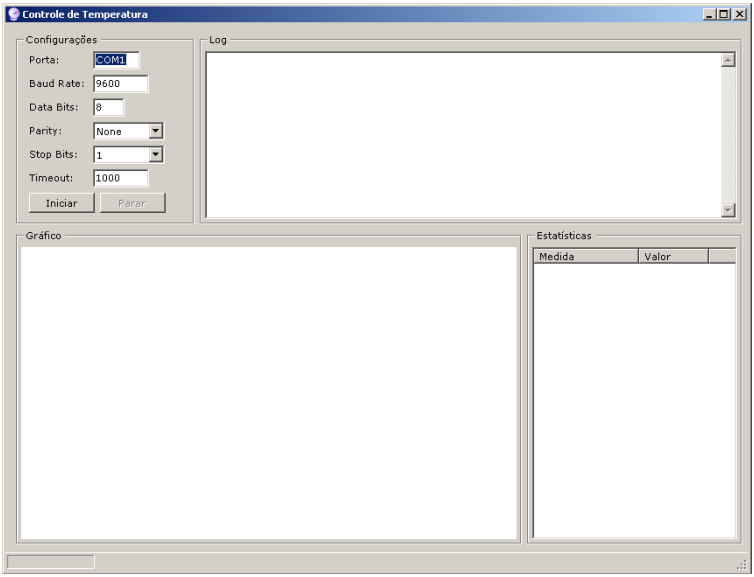


Figura 24 - Tela principal do programa para exibição dos dados registrados.

Na Figura 25 é exibida a tela do programa principal após a recepção completa dos dados aleatórios registrados pelo protótipo. Percebe-se na barra inferior a exibição do gradiente de temperatura referente à reta destacada no gráfico. Também na barra inferior é destacada a posição do cursor no gráfico.

A reta foi definida com dois cliques sobre a curva de dados. Isso torna simples a geração e avaliação dos gradientes de interesse do usuário.

As temperaturas máximas de cada sensor e o tempo no qual ela aparece são exibidos na parte inferior direta da tela.

Na parte superior, a direita, é exibido o registro da comunicação. Nesse registro constam os dados já tratados da comunicação entre o protótipo e o software. Assim, pode-se ver a qual sensor cada temperatura pertence e em que data o dado foi transferido.

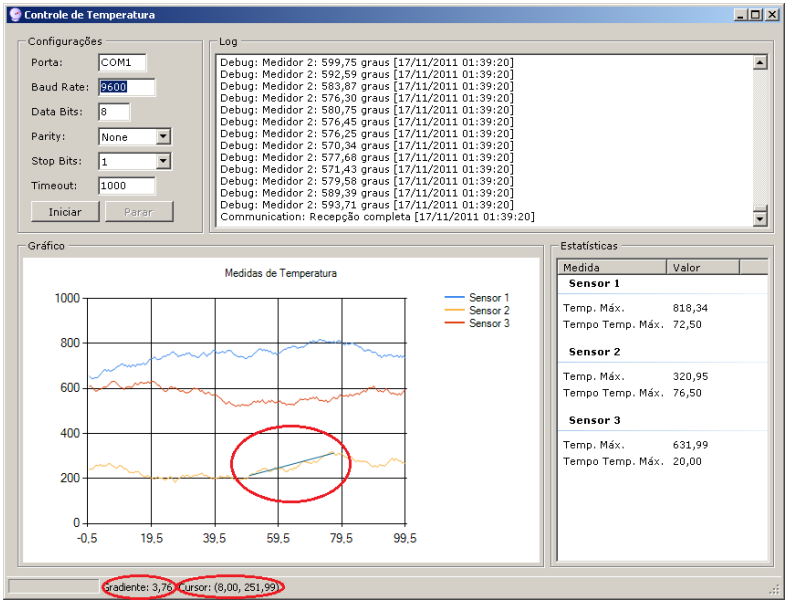


Figura 25 - Tela principal com dados recebidos e dados calculados.

5.3. Conclusão

Como proposto, o software desenvolvido para o computador atende os requisitos. Não é necessário que o usuário faça intervenções complexas para efetuar o recebimento das amostras de temperatura e os dados essenciais, pois o gradiente de temperatura e temperatura máxima, podem ser obtidos com cliques sobre o gráfico.

A ferramenta de programação, *Visual Studio Express C#,* é muito versátil e possibilita alterações rápidas do código. A vasta diversidade de bibliotecas de funções prontas ajuda no rápido desenvolvimento do código e a disponibilidade da comunidade de desenvolvedores ajuda programadores iniciantes.

Outro ponto forte é a linguagem de programação em si. A orientação a eventos facilita a programação e faz com que alterações sejam facilmente executadas. Porém o código acaba se tornando mais intrincado e de complexo entendimento caso a documentação seja falha.

6. RESULTADOS EXPERIMENTAIS

Para a obtenção dos resultados experimentais o protótipo foi posto em funcionamento para obter os dados de temperatura e levantamento do perfil térmico em condições reais. A base da experimentação foi uma máquina de soldagem de dupla onda do fabricante APSGold/Novastar, modelo Spartan 8D.

Nesse teste foi usada apenas uma sonda de temperatura, posicionada no centro da placa de circuito avaliada. A utilização de apenas uma sonda para avaliação justifica-se pela baixa densidade de componentes na placa, o que facilita o aquecimento uniforme da placa, e suas pequenas dimensões.

Na Figura 26 é exibida a configuração de teste do protótipo, na qual o mesmo está acondicionado dentro na caixa protetora e envolvido pela manta isolante.

Figura 26 - Protótipo acondicionado dentro da caixa protetora com a manta isolante.

6.1. Testes

O principal teste avalia o funcionamento das funções 1 e 2, que são efetuar a leitura das temperaturas tratadas pelos CIs MAX6675 e gravá-las na EEPROM externa; na função 2, ler as

temperaturas armazenadas nessa EEPROM externa e transferi-las para o PC.

Para testar essas funções um termopar tipo K foi conectado ao protótipo e a extremidade da junção bimetálica foi fixa na placa de teste através de fita adesiva especial resistente a temperatura. Foi feita a parametrização da máquina para a placa de teste, ressaltando-se que a máquina já se encontrava calibrada para essa placa. Assim o resultado esperado possui um perfil térmico adequado.

A placa de teste foi presa ao "*pallet*" de transporte da máquina de solda e o protótipo, já acondicionado dentro de sua caixa protetora, também foi preso a um "*pallet*". Ambos "*pallets*" foram acomodados na esteira transportadora da máquina de solda e passaram por todas as etapas do ciclo de soldagem da máquina de dupla onda, incluindo a exposição ao fluxo e as ondas de solda, para que o perfil fosse traçado por inteiro.

Ao final do teste foram avaliadas as condições da capa protetora do protótipo e da placa de teste, junto com as curvas geradas pelo software.

A função 3 faz a leitura da temperatura dos sensores e transmite diretamente pela USART, sem armazenamento de informações. Esse teste é dinâmico, conforme informações são recebidas pelo software, o mesmo atualiza o gráfico gerado na tela. Nessa função fica desabilitada a exibição da temperatura máxima recebida e do gradiente até que a recepção de dados seja finalizada (clicando-se no botão "Parar").

6.2. Relatórios Gerados

Nessa seção os dados recolhidos pelo protótipo serão interpretados e identificados os pontos principais da curva de temperatura. Na Figura 27 é exibida a tela principal do programa. Em destaque tem-se o pico de temperatura, obtido quando a sonda de temperatura, conectada a placa de teste, atinge as ondas de solda. Percebem-se dois picos entre um curto vale que representa o espaçamento das ondas de solda.

Como citado no capítulo 3, são nesses momentos que a energia transferida nas zonas anteriores de pré-aquecimento

deve atingir seu valor máximo, fazendo com que soldagem seja executada de maneira desejada.

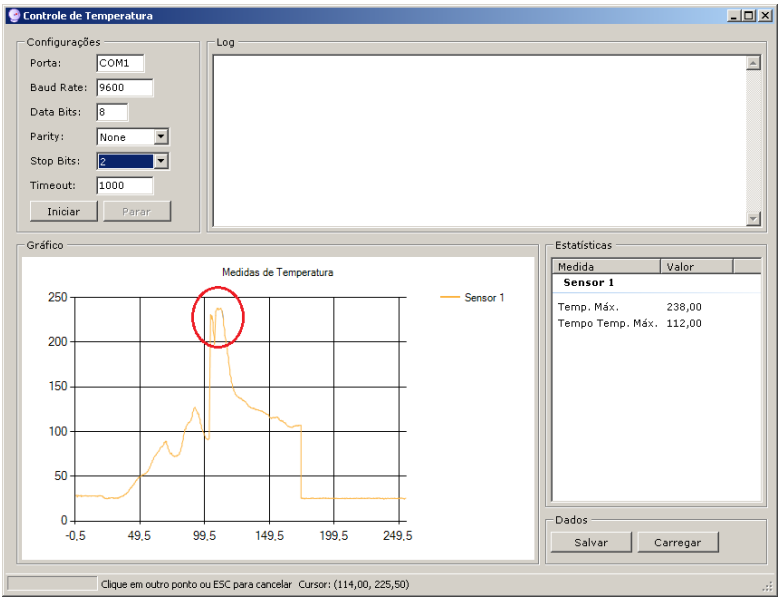


Figura 27 - Tela principal destaque para o pico de temperatura.

Na Figura 28 é destacada a primeira zona de pré-aquecimento existente na máquina onde o teste foi executado. O gradiente de temperatura é exibido na barra inferior.

O gradiente é calculado usando a equação 1:

$$T = \frac{\Delta T}{\Delta t} = \frac{T_2 - T_1(°C)}{t_2 - t_1(s)} \qquad (1)$$

Onde T_x são as temperaturas das amostras e t_x são os intervalos onde as amostras foram feitas.

É recomendado que os gradientes de temperatura das zonas de pré-aquecimento não sejam muito grandes. A transferência de muita energia (aquecimento muito rápido) pode fazer com que o substrato da placa de circuito sofra deformação fazendo com que os componentes não tenham a fixação mecânica ideal e a conexão elétrica fique prejudicada.

Outro fator para que o pré-aquecimento seja controlado é a ativação do fluxo. Parte da energia transferida nesses períodos é absorvida pelo fluxo que usa esse calor para executar a limpeza dos terminais e ilhas onde a solda irá se depositar e para evaporar o solvente do fluxo.

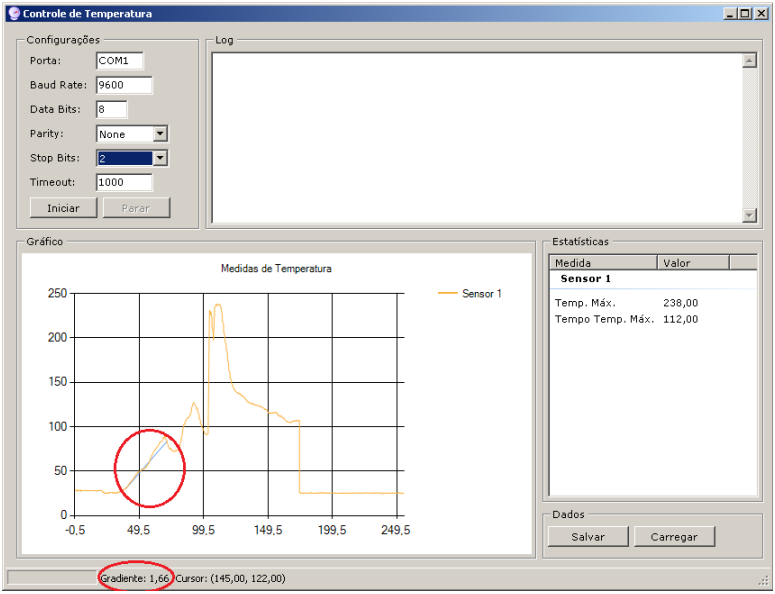


Figura 28 - Tela principal destaque para a primeira zona de pré-aquecimento.

O gradiente de temperatura deve ser mantido entre 2 a 3 °C/s. Porém esse gradiente depende da velocidade com a qual a placa percorre os estágios da máquina, da capacidade da máquina de gerar calor, da capacidade de absorção do substrato e das características da própria placa.

Na Figura 29 é exibida a tela do programa, com destaque para o gradiente de temperatura da segunda zona de pré-aquecimento.

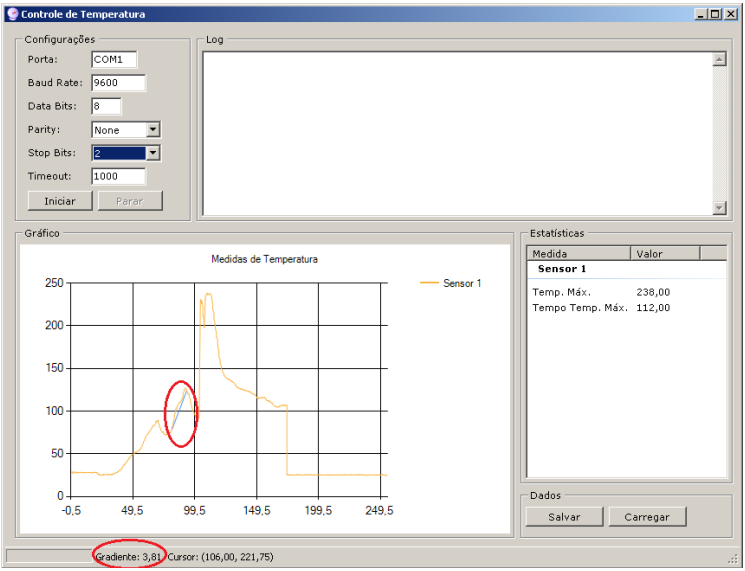


Figura 29 - Tela principal destaque para a segunda zona de pré-aquecimento.

Do mesmo modo que a energia das zonas de pré-aquecimento é transferida para a placa de circuito impresso e para o fluxo a fim de que os processos que envolvem esses componentes (soldagem e limpeza de substrato) sejam corretamente realizados, também é necessária a remoção dessa energia após a etapa de deposição da liga de solda no substrato. Assim, também nessa etapa deve ser controlada a temperatura, porém de modo inverso.

O resfriamento incorreto da placa também acarreta danos ao substrato, gerando deformações, e as soldas que caso forem resfriadas muito rapidamente podem não formar as camadas intermetálicas corretamente.

Na Figura 30 é destacada a área de resfriamento da placa após a soldagem dos componentes.

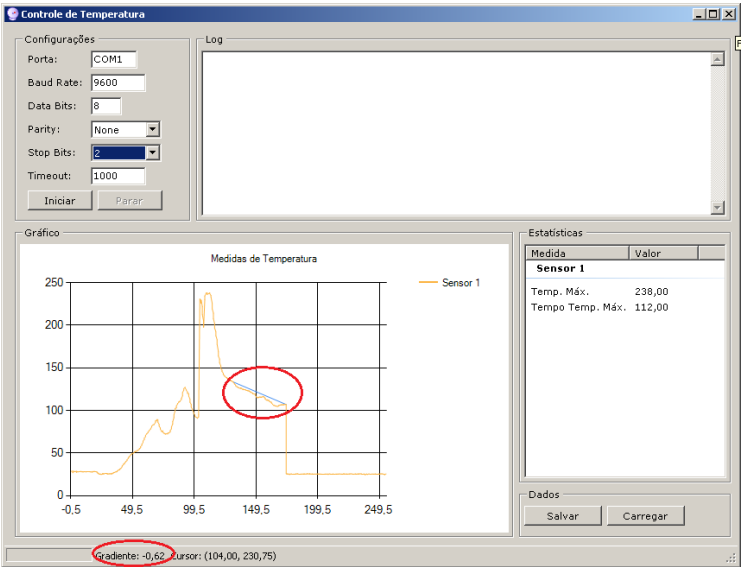


Figura 30 - Tela principal destaque para a área de resfriamento.

6.3. Proteção do protótipo

Para assegurar a eficácia das proteções do protótipo foi também avaliado o comportamento térmico da caixa metálica e da manta isoladora. O conjunto caixa, manta e protótipo foi mantido na zona de refusão de um forno de refusão, onde a temperatura atinge 230 °C e lá foi mantido por cinco minutos (trezentos segundos).

O gráfico da Figura 31 ilustra o comportamento da temperatura dentro da caixa. Percebe-se que a curva de aquecimento tem um gradiente pequeno até aproximadamente duzentos e dez segundos, margeando os 40 °C.

Após duzentos e dez segundos a taxa de aquecimento interno da caixa aumenta consideravelmente. Esse parâmetro é de fundamental importância dado que a fonte de energia do protótipo são pilhas alcalinas e a sugestão de uso das mesmas recomenda não extrapolar a temperatura de 55 °C dado os riscos de explosão e vazamento.

Isso prova a eficácia da proteção do protótipo, uma vez que o tempo de permanência nessa câmara de aquecimento é superior ao tempo total do clico de soldagem por refusão e a temperatura da câmara é a maior do ciclo, o que força maior transmissão para o protótipo.

Essa seria a situação de pior esforço térmico, pois o ciclo completo de solda tem gradientes de temperatura menores nas zonas de pré-aquecimento anteriores.

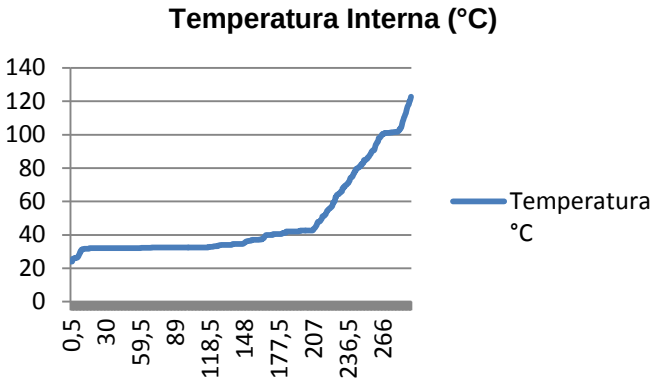


Figura 31 – Gráfico do comportamento da temperatura interna da proteção do protótipo

6.4. Conclusão

O experimento mostrou-se válido pois através do protótipo e dos dados obtidos foi possível a avaliação e interpretação de uma curva de temperatura.

De posse dos dados verifica-se que a máquina na qual as medições foram feitas está com um perfil térmico adaptado para a placa proposta.

7. CONCLUSÃO GERAL

A indústria eletrônica cresceu muito movida principalmente pela grande demanda por bens de consumo como eletrodomésticos, carros e equipamentos diversos do nosso dia-a-dia. Essa crescente demanda fez com que a indústria melhorasse seus processos para que pudesse continuar competitiva.

Melhoria de processos resume-se a redução de desperdícios levando a aumento da eficiência e maior competitividade. Também essa competitividade fez com que a velocidade dos processos aumentasse, não somente sua eficiência, fazendo com que etapas da produção como a soldagem se tornasse críticas.

Nesse âmbito esse trabalho trás um estudo básico sobre a metalurgia envolvida no processo de soldagem, a apresentação dos equipamentos mais comuns nos processos e apresenta uma solução para o controle de um parâmetro fundamental no processo de soldagem, a temperatura.

O protótipo aqui apresentado compôs-se de uma placa de aquisição de dados criada visando baixo custo, robustez de processo e facilidade de uso, e um software dedicado ao tratamento das informações colhidas pela placa de aquisição.

Como possíveis melhorias e trabalhos futuros propõem-se o aumento do número de sondas para aferição da temperatura, disponibilização da troca de alguns parâmetros como tempo entre amostras, ou número de amostras por segundo, a avaliação de outros meios para transferência dos dados coletados (Bluetooth, zig-bee, wi-fi dentre outros) para a parte física.

Para o software pode-se considerar melhorias na interface incluindo uma ferramenta para medição do tempo de exposição a certas faixas de temperatura e a visualização de mais gradientes de temperatura por gráfico exibido.

8. BIBLIOGRAFIA

[1] LAU, J.H. **Electronics manufacturing: with lead-free, halogen-free, and conductive-adhesive material**. McGraw-Hill. 2003.

[2] MANKO, Howard H. **Solders and soldering Materials, Design, Production and Analysis for reliable bonding**. McGraw-Hill. 1964.

[3] LIMA, Charles Borges de, **Técnicas de Projetos Eletrônicos com os Microcontroladores AVR**. Ed. do autor. 2010.

[4] SCHILDT, Herbert. **C Completo e Total**. São Paulo. Makron Books. 1996.

[5] CALLISTER, Willian D. **Ciência e Engenharia de Materiais: uma introdução.** LTC. 2007

[6] GURGACZ, G.; NASCIMENTO, Z. **Metodologia do Trabalho Científico com**
Enfoque nas Ciências Exatas. Joinville. SOCIESC, 2007.

[7] NORMAS PARA APRESENTAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO – TCC. Curso Superior de Tecnologia em Sistemas Eletrônicos. Florianópolis: CEFET, 2008.

[8] Seção Tutoriais, <http://smd-on-line.com/>

[9] “NASA WORKMANSHIP STANDARDS”, disponível em <http://workmanship.nasa.gov/lib/insp/2%20books/links/sections/files/301.pdf>

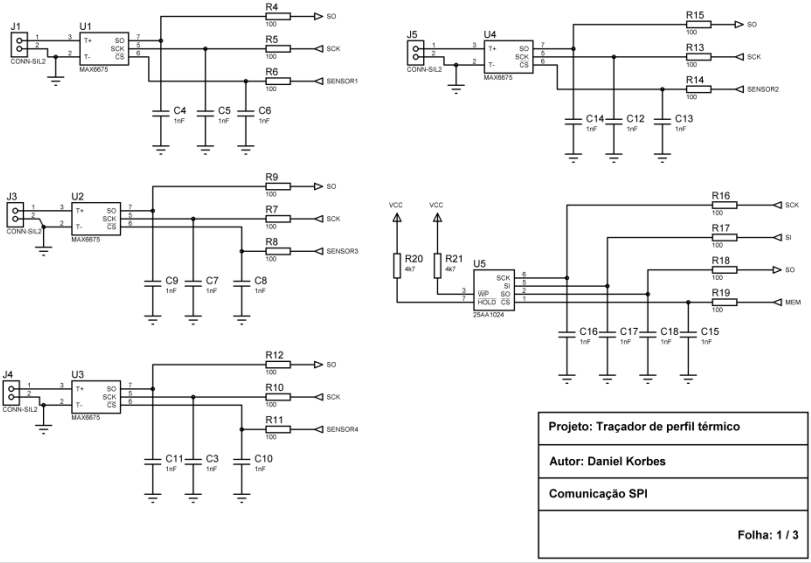
[10] Association Connecting Electronics Industries. **IPC J-STD-003B**: *Solderability Tests for Printed Boards*, 2003.

[11] AVRFREAKS.NET. Consulta a fórum de discussão. Disponível em: http://www.avrfreaks.net/index.php?name=PNphpBB2&file=viewforum&f=11&sid=a63212cbee238a7cecf932e304228d34

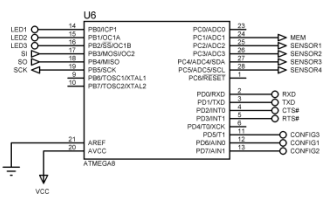
[12] On-line CRC calculation and free library, http://www.lammertbies.nl/comm/info/crc-calculation.html

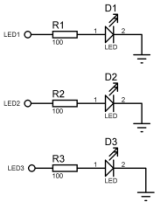
ANEXOS

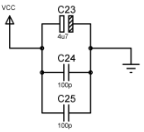
Anexo 1
Esquemático

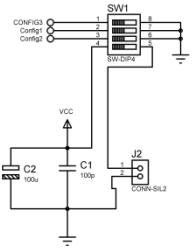


Projeto: Traçador de perfil térmico
Autor: Daniel Korbes
Comunicação SPI
Folha: 1 / 3

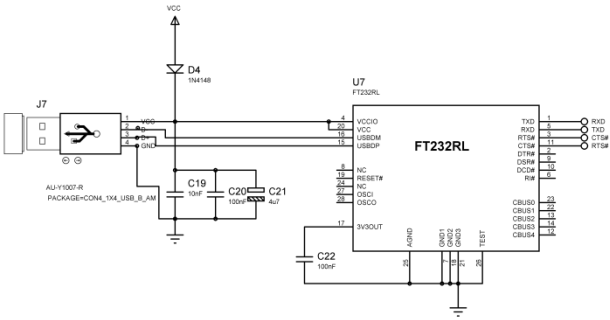








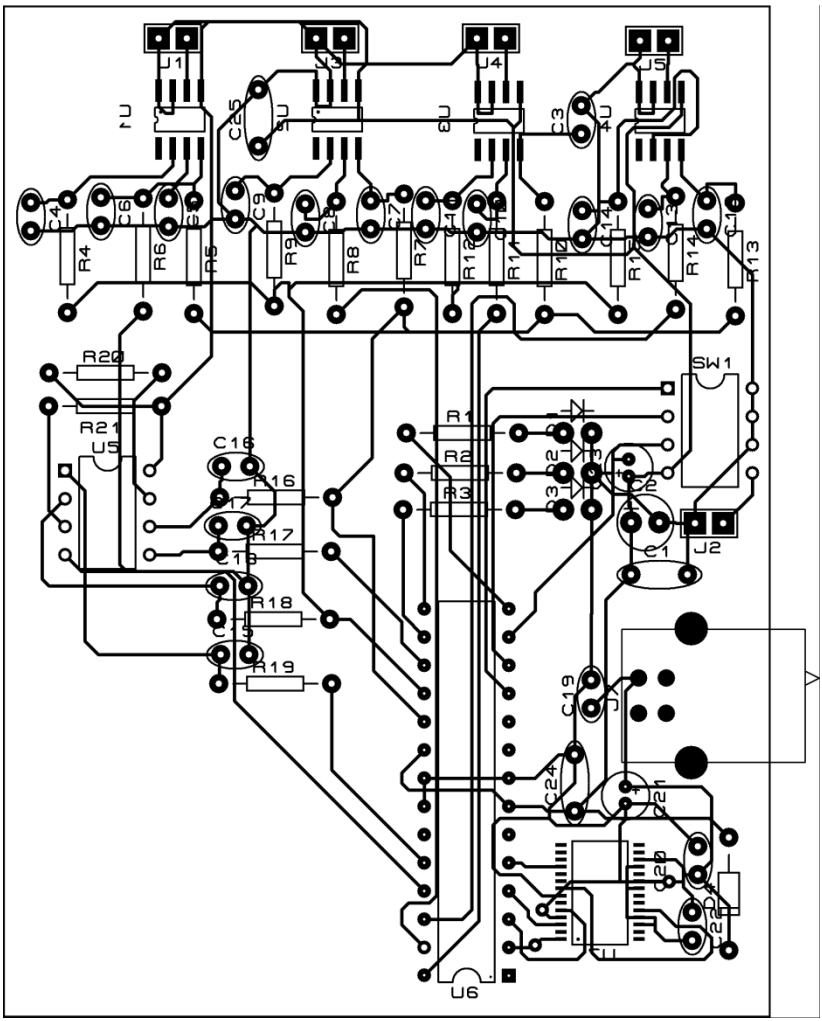
Projeto: Traçador de perfil térmico
Autor: Daniel Korbes
Miscelânea
Folha: 2 / 3



Projeto: Traçador de perfil térmico
Autor: Daniel Korbes
Comunicação USART - USB
Folha: 3 / 3

Anexo 2

Layout da placa


J1
J3
J4
J5
C4
C6
C5
C9
C8
C7
C3
C14
C13
C11
R4
R6
R5
R9
R8
R7
R12
R11
R10
R15
R14
R13
R20
R21
C16
C5
R16
R17
C15
R18
C17
R19
R1
R2
R3
D2
D3
SW1
C2
C1
J2
C19
J6
C24
C21
C20
C22
U6

Anexo 3
Código fonte firmware

```
/*
 * Thermal.c
 *
 * Created: 12/09/2011 18:58:17
 *  Author: Daniel Korbes
 */

#define F_CPU 1000000UL
#define set_bit(adress,bit) (adress|=(1<<bit))
#define clr_bit(adress,bit) (adress&=~(1<<bit))
#define tst_bit(adress,bit) (adress&(1<<bit))
#define cpl_bit(adress,bit) (adress^=(1<<bit))

//Portas IO comum
#define LED1   0
#define LED2   1
#define LED3   2
#define SW1             PD5
#define SW2             PD6
#define SW3             PD7

//Portas SS barramento SPI
#define sensor1 PC2
#define sensor2 PC3
#define sensor3 PC4
#define sensor4 PC5
#define memoria PC1

//Operações para a memória SPI
#define ler 0x03
#define escrever 0x02
#define prot 0x04
#define hab 0x06
#define wrsr 0x01

//Definições USART
#define FOSC 1000000 //1843200 //clock speed
#define BAUD 9600
#define MYUBRR 5 //FOSC/16/BAUD-1
```

```
#include <avr/io.h>
#include "delay.h"
#include "eeprom.h"
#include <avr/interrupt.h>
#include <avr/crc16.h>

unsigned long int temp=0, endereco=0, nr_sensores,
contador_amostras=0;
unsigned int status=0, config=0, aux_isr=0, tempo_amostra=0,
i=0, erro_com=0;
char estado=0;

/* Definição dos bits de STATUS

0 0 0 0 . 0 0 0 0
| | | |   | | | |
| | | |       | | |     Inicialização do estado 1 (0 - não, 1 - sim)
| | | |   | | Inicialização do estado 2 (0 - não, 1 - sim)
| | | |   | Inicialização do estado 3 (0 - não, 1 - sim)
| | | |
| | | Presença sensor 1 (1 - sim, 0 - não)
| | Presença sensor 2 (1 - sim, 0 - não)
| Presença sensor 3 (1 - sim, 0 - não)
Presença sensor 4 (1 - sim, 0 - não)

 Definição dos bits de CONFIG

0 0 0 0 . 0 0 0 0
| | | |   | | | |
| | | |       | | |     Controle comunicação (aguardando USART? 1 -
sim, 0 - não)
| | | |   | |
| | | |   |
| | | |
| | Salvam a quantidade de sensores presentes ao iniciar a leitura
(combinação binaria de 2 bits)
| |
|

*/
```

```
ISR(TIMER0_OVF_vect) //interrupção do T/C0
{
	cpl_bit(PORTB,LED1);
	TCNT0=0x0B;
	aux_isr++;
	if (aux_isr>2)
	{
		aux_isr=0;
		tempo_amostra=1;
	}
	if (tst_bit(config,0))
	{
		erro_com++;
	}
}

void IO_init(void)
{
	DDRB=0x2F;
	PORTB=0x28;
	DDRC=0x3E;
	PORTC=0xC0;
	DDRD=0x00;
	PORTD=0xE0;
}

void USART_Init(unsigned int ubrr)
{
UBRRH = (unsigned char)(ubrr>>8); //Set baud rate
UBRRL = (unsigned char)ubrr;
UCSRB = (1<<RXEN)|(1<<TXEN); //Enable receiver and
transmitter
UCSRC = (1<<URSEL)|(1<<USBS)|(3<<UCSZ0); //Set frame
format: 8data, 2stop bit
}

void USART_Transmit( unsigned char data )
{
```

```
while ( !( UCSRA & (1<<UDRE)) ); //Wait for empty transmit
buffer
UDR = data; //Put data into buffer, sends the data
}

unsigned char USART_Receive( void )
{
set_bit(config,0);
while ((UCSRA & (1<<RXC)) == 0) //Wait for data to be received
{
        if (erro_com>=10)
        {
                clr_bit(config,0);
                erro_com=0;
                break;
        }
}
return UDR; //Get and return received data from buffer
}

int CRC(unsigned long int dado)
{
        unsigned int crc_calc;
        crc_calc = (int)((dado & 0x00FF) + crc_calc) % 256;
        crc_calc = (int)(((dado & 0xFF00)>>8) + crc_calc) % 256;
        return crc_calc;
}

void USB_Comm(unsigned long int data)
{
        unsigned int data_low, data_high, crc_trasmit,
crc_receive, ack=0;
        data_low = data;
        data_high = data >> 8;
        crc_trasmit=CRC(data);
        while(!ack)
        {
                USART_Transmit(data_high);
                USART_Transmit(data_low);
                USART_Transmit(crc_trasmit);
```

```
                crc_receive=USART_Receive();
                if (crc_receive==0x06)
                {
                        ack=1;
                        crc_receive=0;
                }
        }
}

void SPI_Master_Init( )
{
//DDRB = (1<<PB5)|(1<<PB3); //Ajusta MOSI e SCK como saída,
demais pinos entrada
SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0); //Habilita SPI,
Mestre, taxa de clock ckl/16
set_bit(PORTC,sensor1);
set_bit(PORTC,sensor2);
set_bit(PORTC,sensor3);
set_bit(PORTC,sensor4);
set_bit(PORTC,memoria);
}

void SPI_Master_Transmit(char dado)
{
SPDR = dado;                            //Inicia a transmissão
while(!(SPSR & (1<<SPIF))); //Espera a transmissão ser
completada
}

void Read_temp(char sensor)
{
        clr_bit(PORTC,sensor);
        _delay_us(1);
        SPI_Master_Transmit(0x00);
        temp = SPDR << 8;
        SPI_Master_Transmit(0x00);
        temp |= SPDR;
        set_bit(PORTC,sensor);
}
```

```
void Write_mem(unsigned long int address, unsigned int dado)
{
	unsigned int low_address, high_address;
	unsigned int low_dado, high_dado;
	low_address = address;
	high_address = address >> 8;
	low_dado = dado;
	high_dado = dado >> 8;
	clr_bit(PORTC,memoria);
	_delay_us(1);
	SPI_Master_Transmit(hab);
	set_bit(PORTC,memoria);
	_delay_us(1);
	clr_bit(PORTC,memoria);
	_delay_us(1);
	SPI_Master_Transmit(escrever);
	SPI_Master_Transmit(high_address);
	SPI_Master_Transmit(low_address);
	SPI_Master_Transmit(high_dado);
	set_bit(PORTC,memoria);
	_delay_ms(5);
	address++;
	low_address = address;
	high_address = address >> 8;
	clr_bit(PORTC,memoria);
	_delay_us(1);
	SPI_Master_Transmit(hab);
	set_bit(PORTC,memoria);
	_delay_us(1);
	clr_bit(PORTC,memoria);
	_delay_us(1);
	SPI_Master_Transmit(escrever);
	SPI_Master_Transmit(high_address);
	SPI_Master_Transmit(low_address);
	SPI_Master_Transmit(low_dado);
	set_bit(PORTC,memoria);
	_delay_ms(5);
}

long int Read_mem(long int address)
```

```
{
	unsigned int low_address, high_address;
	unsigned long int temp_mem;
	low_address = address;
	high_address = address >> 8;
	clr_bit(PORTC,memoria);
	_delay_us(1);
	SPI_Master_Transmit(ler);
	SPI_Master_Transmit(high_address);
	SPI_Master_Transmit(low_address);
	SPI_Master_Transmit(0x00);
	set_bit(PORTC,memoria);
	temp_mem = SPDR;
	temp_mem = temp_mem << 8;
	address++;
	low_address = address;
	high_address = address >> 8;
	clr_bit(PORTC,memoria);
	_delay_us(1);
	SPI_Master_Transmit(ler);
	SPI_Master_Transmit(high_address);
	SPI_Master_Transmit(low_address);
	SPI_Master_Transmit(0x00);
	set_bit(PORTC,memoria);
	temp_mem = temp_mem | SPDR;
	return temp_mem;
}
int main(void)
{
	IO_init();				// Inicializações
	clr_bit(PORTB,LED1);
	clr_bit(PORTB,LED2);
	clr_bit(PORTB,LED3);
	aux_isr=0;
	tempo_amostra=0;
	contador_amostras=0;
	nr_sensores=0;
	USART_Init (MYUBRR);
	SPI_Master_Init();
	TCNT0=0x0B;
```

```
        TCCR0 = (1<<CS02) | (1<<CS00); //T/C0 com prescaler
de 1024, a 1 MHz gera uma interrupção a cada 244 ms +/-
        TIMSK = 1<<TOIE0; //habilita a interrupção do T/C0
        sei();
        temp=0x415A;
   while(1)
   {
                if (!tst_bit(PIND,SW1))
                {
                        _delay_ms(5);
                        if (!tst_bit(PIND,SW1))
estado=estado|0x01;             // Testa SW1 (gravação de
valores)
                        else estado=estado&0x06;
                }
                if (!tst_bit(PIND,SW2))
                {
                        _delay_ms(5);
                        if (!tst_bit(PIND,SW2))
estado=estado|0x02;             // Testa SW2 (transmissão de
valores)
                        else estado=estado&0x05;
                }
                if (!tst_bit(PIND,SW3))
                {
                        _delay_ms(5);
                        if (!tst_bit(PIND,SW3))
estado=estado|0x04;             // Testa SW3 (testes e calibração)
                        else estado=estado&0x03;
                }

                switch (estado)
                {
                        case 0x01:
                                cpl_bit(PORTB,LED2);
                                                // Faz com que os
valores de temperatura sejam gravados na EEPROM externa
                                if (!tst_bit(status,0))
                                                //testa a primeira
passagem por aqui, a inicialização
```

```
{
        Read_temp(sensor1);
                //descobre quais sensores estão conectados
        if (!tst_bit(temp,2))
        {
                set_bit(status,4);
                nr_sensores++;
        }

        else clr_bit(status,4);

        Read_temp(sensor2);
        if (!tst_bit(temp,2))
        {
                set_bit(status,5);
                nr_sensores++;
        }

        else clr_bit(status,5);

        Read_temp(sensor3);
        if (!tst_bit(temp,2))
        {
                set_bit(status,6);
                nr_sensores++;
        }

        else clr_bit(status,6);

        Read_temp(sensor4);
        if (!tst_bit(temp,2))
        {
                set_bit(status,7);
                nr_sensores++;
        }

        else clr_bit(status,7);
```

```
Write_mem(0xfffc,nr_sensores);
endereco=0x0000; // aponta a memoria para o início
set_bit(status,0); // marca a inicialização
        }
else
        {
        if(tempo_amostra==1)
        {
                if (tst_bit(status,4))
                {
                Read_temp(sensor1);
                clr_bit(temp,15);
                clr_bit(temp,14);
                Write_mem(endereco,temp);
                endereco=+2;
        }
        if (tst_bit(status,5))
        {
        Read_temp(sensor2);
        clr_bit(temp,15);
        set_bit(temp,14);
        Write_mem(endereco,temp);
        endereco=+2;
        }
        if (tst_bit(status,6))
        {
        Read_temp(sensor3);
        set_bit(temp,15);
        clr_bit(temp,14);
        Write_mem(endereco,temp);
        endereco=+2;
        }
        if (tst_bit(status,7))
        {
        Read_temp(sensor4);
        set_bit(temp,15);
        set_bit(temp,14);
        Write_mem(endereco,temp);
                                endereco=+2;
```

```
        }
        tempo_amostra=0;
        contador_amostras++;
        Write_mem(0xfffe,contador_amostras);
        }
}
break;
case 0x02:
        // Os dados gravados na EEPROM externa são
transmitidos
contador_amostras=Read_mem(0xfffe);
nr_sensores=Read_mem(0xfffc);
USB_Comm(contador_amostras||(nr_sensores<<14));
        for (endereco=0x0000 ; endereco <= contador_amostras *
nr_sensores ; endereco+=2)
        {
        cpl_bit(PORTB,LED3);
        USB_Comm(Read_mem(endereco));
        }
        while (1)
        {
        }
break;
case 0x04:
        // Lê os dados dos sensores e já transmite, sem
armazenar
        cpl_bit(PORTB,LED2);
        cpl_bit(PORTB,LED3);
        if (tempo_amostra==1)
        {
        Read_temp(sensor1);
        if (!tst_bit(temp,2))
        {
        clr_bit(temp,15);
        clr_bit(temp,14);
        set_bit(temp,12);
        USB_Comm(temp);
        }
```

```
            Read_temp(sensor2);
            if (!tst_bit(temp,2))
            {
            clr_bit(temp,15);
            set_bit(temp,14);
            set_bit(temp,12);
            USB_Comm(temp);
            }
            Read_temp(sensor3);
            if (!tst_bit(temp,2))
            {
            set_bit(temp,15);
            clr_bit(temp,14);
            set_bit(temp,12);
            USB_Comm(temp);
            }
            Read_temp(sensor4);
            if (!tst_bit(temp,2))
            {
            set_bit(temp,15);
            set_bit(temp,14);
            set_bit(temp,12);
            USB_Comm(temp);
            }
            tempo_amostra=0;
            }
            break;
            default:          // Pisca LED´s avisando configuração
errada
            clr_bit(PORTB,LED2);
            clr_bit(PORTB,LED3);
            estado=0x00;
                        }
      }
}
```

Anexo 4

Fonte Software

StateMachine.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO.Ports;
using System.Diagnostics;

namespace Communication
{
   class StateMachine : IDisposable
   {
      enum States
      {
         Byte1Header,
         Byte2Header,
         CRCHeader,
         Byte1Sample,
         Byte2Sample,
         CRCSample
      }

      SerialPort _port;
      Dispatcher _disp;
      ILog _logger;
      SerialPortConfiguration _config;

      const byte ACK = 0x06;
      const byte ENQ = 0x05;

      Stopwatch _timer = new Stopwatch();
      States _state = States.Byte1Header;
      private Dictionary<int, int> _sampleCounter = new
Dictionary<int, int>();
      byte[] _header = new byte[3];
      byte[] _sample = new byte[3];

      public StateMachine(ILog logger, SerialPortConfiguration
config, Dispatcher disp)
```

```
        {
            if (logger == null)
                throw new ArgumentNullException("config");

            if (config == null)
                throw new ArgumentNullException("config");

            if (disp == null)
                throw new ArgumentNullException("disp");

            _logger = logger;
            _disp = disp;
            _disp.StateMachine = this;
            _config = config;
            _disposed = false;

            _port = new SerialPort(config.PortName,
config.BaudRate, config.Parity, config.DataBits, config.StopBits);
            _port.DataReceived += new
SerialDataReceivedEventHandler(port_DataReceived);
            _port.Open();
        }

        #region Message Receiving
        private void port_DataReceived(object sender,
SerialDataReceivedEventArgs e)
        {
            byte[] buffer = new byte[10];
            int read = _port.Read(buffer, 0, 10);

            for (int i = 0; i < read; i++)
            {
                ProcessByte(buffer[i]);
            }
        }

        private void ProcessByte(byte p)
        {
            if (_state == States.Byte1Header)
                _timer.Restart();
```

```
            else
            {
                if (_timer.ElapsedMilliseconds > _config.Timeout)
                {
                    _logger.Write("Timeout", LogType.Communication);
                    _state = States.Byte1Header;
                    _timer.Restart();
                }
            }

            _logger.Write(string.Format("Received byte 0x{0:X} in
state {1:G}", p, _state), LogType.Debug);

            switch (_state)
            {
                case States.Byte1Header:
                    _header[0] = p;
                    _state = States.Byte2Header;
                    break;

                case States.Byte2Header:
                    _header[1] = p;
                    _state = States.CRCHeader;
                    break;

                case States.CRCHeader:
                    _header[2] = p;
                    if (ProcessHeader())
                        _state = States.Byte1Sample;
                    else
                        _state = States.Byte1Header;
                    break;

                case States.Byte1Sample:
                    _sample[0] = p;
                    _state = States.Byte2Sample;
                    break;

                case States.Byte2Sample:
                    _sample[1] = p;
```

```
                _state = States.CRCSample;
                break;

            case States.CRCSample:
                _sample[2] = p;
                if (ProcessSample())
                    _state = SamplesComplete ? States.Byte1Header
: States.Byte1Sample;
                else
                    _state = States.Byte1Sample;
                break;

            default:
                _logger.Write("Invalid state", LogType.Debug);
                _state = States.Byte1Header;
                break;
        }
    }
    #endregion

    private bool SamplesComplete { get; set; }

    public int Meters { get; set; }

    public int Samples { get; set; }

    private bool ProcessSample()
    {
        // Check CRC
        byte crc = CalculateCRC(_sample, 2);
        if (crc != _sample[2])
        {
            SendRequestToRepeat();
            return false;
        }

        // Get data
        int meter = ((_header[0] & 0xC0) >> 6);
        int temp_raw = ((_header[0] & 0x0F) << 8) | _header[1];
```

```
            double temp = (int)(temp_raw / 4) + (temp_raw % 4) *
0.25;

            // Notify the controller
            SendConfirmation();

            // Notify the user
            _disp.OnTemperatureReceived(meter, temp);

            // Log the message
            LogMessage("Rx", _header);

            // Update counters
            if (_sampleCounter.ContainsKey(meter))
                _sampleCounter[meter] = _sampleCounter[meter] + 1;
            else
                _sampleCounter[meter] = 1;

            // Check if we're done
            if (_sampleCounter.Count == Meters)
            {
                bool done = true;
                foreach (var item in _sampleCounter)
                {
                    if (item.Value != Samples)
                    {
                        done = false;
                        break;
                    }
                }
                if (done)
                {
                    SamplesComplete = true;
                    _disp.OnReceptionCompleted();
                }

            }

            return true;
        }
```

```
private bool ProcessHeader()
{
    // Check CRC
    byte crc = CalculateCRC(_header, 2);
    if (crc != _header[2])
    {
        SendRequestToRepeat();
        return false;
    }

    // Get data
    Meters = ((_header[0] & 0xC0) >> 6) + 1;
    Samples = ((_header[0] & 0x0F) << 8) | _header[1];

    // Reset counters
    _sampleCounter.Clear();
    SamplesComplete = false;

    // Notify the controller
    SendConfirmation();

    // Notify the user
    _disp.OnHeaderReceived(Meters, Samples);

    // Log the message
    LogMessage("Rx", _header);

    return true;
}

private byte CalculateCRC(byte[] buffer, int count)
{
    if (count > buffer.Length || count < 0)
        throw new ArgumentOutOfRangeException("count");

    byte crc = 0;

    for (int i = 0; i < count; i++)
        crc = (byte)((buffer[i] + crc) % 256);
```

```
            _logger.Write(string.Format("O CRC para {0} é 0x{1:X}",
StringFromBuffer(buffer, count), crc), LogType.Debug);

            return crc;
        }

        private void SendConfirmation()
        {
            byte[] buffer = new byte[1] { ACK };
            _port.Write(buffer, 0, 1);
            LogMessage("Tx", buffer);
        }

        private void SendRequestToRepeat()
        {
            byte[] buffer = new byte[1] { ENQ };
            _port.Write(buffer, 0, 1);
            LogMessage("Tx", buffer);
        }

        private void LogMessage(string type, byte[] buffer)
        {
            _logger.Write(string.Format("{0}: {1}", type,
StringFromBuffer(buffer)), LogType.Communication);
        }

        private string StringFromBuffer(byte[] buffer)
        {
            return StringFromBuffer(buffer, buffer.Length);
        }

        private string StringFromBuffer(byte[] buffer, int count)
        {
            if (count > buffer.Length || count < 0)
                throw new ArgumentOutOfRangeException("count");

            List<string> ls = new List<string>();
            for (int i = 0; i < count; i++)
                ls.Add(string.Format("0x{0:X}", buffer[i]));
```

```
            return string.Join(" ", ls);
        }


        #region IDisposable Members

        private bool _disposed;
        public void Dispose()
        {
            Dispose(true);
            GC.SuppressFinalize(this);
        }

        protected virtual void Dispose(bool disposing)
        {
            // If you need thread safety, use a lock around these
            // operations, as well as in your methods that use the
resource.
            if (!_disposed)
            {
                if (disposing)
                {
                    if (_port != null)
                        _port.Dispose();
                }

                // Indicate that the instance has been disposed.
                _port = null;
                _disposed = true;
            }
        }

        ~StateMachine()
        {
            // Simply call Dispose(false).
            Dispose(false);
        }

        #endregion
```

```
        internal void Close()
        {
            _port.Close();
            _port.Dispose();
            _port = null;
        }
    }
}
```

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace ControleTemperatura
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new frmMain());
        }
    }
}
```

TestDispatcher.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using Communication;
using System.Threading;

namespace ControleTemperatura
{
   internal class TestDispatcher: IDispatcher
   {
      #region IDispatcher Members

      public event
EventHandler<TemperatureReceivedEventArgs>
TemperatureReceived;

      public event EventHandler<HeaderReceivedEventArgs>
HeaderReceived;

      public event EventHandler ReceptionCompleted;

      public void Close()
      {
         _t.Dispose();
         _t = null;
      }

      #endregion

      internal void OnTemperatureReceived(int meter, double
temperature)
      {
         if (TemperatureReceived != null)
            TemperatureReceived(this, new
TemperatureReceivedEventArgs() { Meter = meter, Temperature
= temperature });
      }

      internal void OnHeaderReceived(int nMeters, int nSamples)
      {
         if (HeaderReceived != null)
```

```
            HeaderReceived(this, new
HeaderReceivedEventArgs() { Meters = nMeters, Samples =
nSamples });
        }

        internal void OnReceptionCompleted()
        {
            if (ReceptionCompleted != null)
                ReceptionCompleted(this, new EventArgs());
        }

        private Timer _t;

        public TestDispatcher()
        {
            _t = new Timer(OnTimer, null, 0, 30000);
        }

        private void OnTimer(object stateInfo)
        {
            int meters = 3;
            int samples = 200;
            Random rand = new
Random(DateTime.Now.Millisecond);
            OnHeaderReceived(meters, samples);
            for (int i = 0; i < meters; i++)
            {
                double temp = rand.NextDouble() * 700;
                for (int j = 0; j < samples; j++)
                {
                    temp = temp + (float)(rand.NextDouble() * 20.0 -
10.0);
                    OnTemperatureReceived(i, temp);
                }
            }
            OnReceptionCompleted();
        }

        #region IDisposable Members
```

```
        private bool _disposed;
        public void Dispose()
        {
            Dispose(true);
            GC.SuppressFinalize(this);
        }

        protected virtual void Dispose(bool disposing)
        {
            // If you need thread safety, use a lock around these
            // operations, as well as in your methods that use the
resource.
            if (!_disposed)
            {
                // Indicate that the instance has been disposed.
                _disposed = true;
            }
        }

        ~TestDispatcher()
        {
            // Simply call Dispose(false).
            Dispose(false);
        }

        #endregion
    }
}
```