

Universidade Federal do Pampa

Autor: Alex Veloso da Silveira

DESENVOLVIMENTO DE UMA INTERFACE USB – I/O ANALÓGICA E DIGITAL

Trabalho de Conclusão de Curso II

**BAGÉ
2013**

ALEX VELOSO DA SILVEIRA

DESENVOLVIMENTO DE UMA INTERFACE USB – I/O ANALÓGICA E DIGITAL

Trabalho de Conclusão de Curso apresentado no curso de Graduação em Engenharia de Computação da Universidade Federal do Pampa, UNIPAMPA, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Edson Massayuki Kakuno

Co-orientador: Prof. Dr. Fabrício de Oliveira Ourique

Bagé

2013

ALEX VELOSO DA SILVEIRA

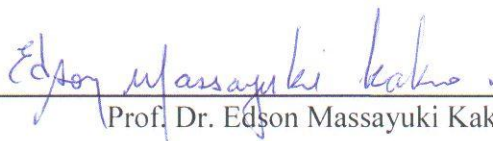
ALEX VELOSO DA SILVEIRA

DESENVOLVIMENTO DE UMA INTERFACE USB – I/O ANALÓGICA E DIGITAL

Trabalho de Conclusão de Curso apresentado no curso de Graduação em Engenharia de Computação da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Computação.

Trabalho de Conclusão de Curso defendido e aprovado em: 11/05/2013

Banca examinadora:



Prof. Dr. Edson Massayuki Kakuno

Engenharia de Computação – UNIPAMPA



Prof. Dr. Milton Roberto Heinen

Engenharia de Computação – UNIPAMPA



Prof. MSc. Érico Marcelo Hoff do Amaral

Engenharia de Computação – UNIPAMPA

Dedico este trabalho à minha avó
Anna Maria Molinari Veloso.

AGRADECIMENTO

Agradeço com grande orgulho aos meus familiares, pois eles se dedicaram e me ajudaram a conquistar essa vitória. Em todos os momentos pude contar com o apoio, a disciplina e os conselhos que me fizeram vencer. Em especial quero agradecer a meus pais, Ricardo e Joana, pois eles são a grande parte das minhas vitórias.

Agradeço a minha namorada fiel, Juliana, pois sempre tem me apoiado, apesar das dificuldades e sempre esteve ao meu lado para me levantar nos momentos em que não conseguia mais caminhar. Agradeço os momentos de compreensão e de abdicção, pois eles são expressos hoje como momentos de sabedoria.

Agradeço em especial ao meu orientador, Edson Massayuki Kakuno pela paciência e companheirismo, não só nestes momentos finais, mas também em todo o período de desenvolvimento deste trabalho.

Agradeço ao professor Pedro Fernando Teixeira Dorneles, pelo apoio e incentivo dado ao longo deste trabalho.

Agradeço ao professor Fabrício de Oliveira Ourique, por ter contribuído neste trabalho e na minha formação.

Agradeço os meus companheiros de turma, por todo apoio e suporte.

Agradeço aos professores do curso da Engenharia de Computação que estão lutando para construção do nosso curso.

Agradeço aos professores do curso da Física, pois mesmo eu sendo de outro curso, me deram todo apoio necessário para o desenvolvimento deste trabalho.

Agradeço em especial os meus colegas da *Falcasoft*, que sempre me apoiaram nos momentos difíceis.

Enfim também sou grato a todos que colaboraram para que eu pudesse completar mais esta etapa de minha formação universitária.

Este trabalho obteve contribuição dos seguintes projetos:

Apoio financeiro parcial do CNPq - FAPERGRS, através do "Edital MCT/CNPq/SECIS/Fundações de Amparo à Pesquisa nº 64/2009 - Espaços Científico-culturais", projeto "Centro de Divulgação de Ciências & Tecnologias da região da Campanha (CDC&TeC)".

Bolsa discente parcial e apoio financeiro parcial do "EDITAL PROEXT 01/2011 – Programa de Apoio a Ações de Formação Continuada", projeto “Utilização do computador, como nova tecnologia, no auxílio na formação técnica e de ciências”.

RESUMO

Neste trabalho é descrito a montagem de uma plataforma de interface universal, com facilidades de entrada e saída analógica e entradas e saídas digitais, bem como a possibilidade de comunicação via USB (*Universal Serial Bus*) com o computador, desta forma é possível instalar o *hardware* específico fora do PC (*Personal Computer*), possibilitando sua utilização em *notebooks*. Com o aumento da facilidade de acesso ao PC, muitos usuários com pouca experiência em programação podem ter dificuldades em desenvolver aplicações em linguagem de baixo nível, como programação de microcontroladores. Neste sentido é proposto o desenvolvimento de pequenas rotinas (leitura das entradas analógicas, leitura das entradas digitais, escrita nas saídas analógicas, escrita nas saídas digitais) que permitam comunicação com o *hardware* e o PC. Foi escolhida a linguagem Python, como linguagem de alto nível, desta forma 'liberando' o usuário final da necessidade de dominar em detalhes as instruções específicas da interface. Para facilitar o acesso ao *hardware* (interface USB, baseado no microcontrolador PIC18F2550 fabricado pela Microchip) foi desenvolvido e disponibilizado o diagrama elétrico e o *layout* da placa de circuito impresso. Visando o acesso de programação de baixo nível aos desenvolvedores interessados em trabalhar no *firmware* da interface, foi desenvolvido um programador para a família PIC18F da Microchip, usando como base o programador PicKit2 da mesma empresa. Como exemplo de aplicação, foi desenvolvido um controlador de temperatura de água utilizando a interface e a linguagem Python.

Palavras-chave

Microcontrolador, PIC18F2550, interface de entrada e saída.

ABSTRACT

Computers today are getting more accessible to everyone and the personal computer is being use in almost all human activities. In some applications, like physics educational laboratories, often the computer is used to control experiments. In such cases, a hardware interface is made necessary. Here, we propose hardware with respective software solution. Aim those that are not expert in computing and low level software programming, we elected these requirement: i) an interface that could read analog and digital signals; ii) an interface that could write analog and digital signals; iii) an interface that communicates with the computer using USB port, because USB is found in almost all computer (desk top and portables); iv) an interface that communicates with a high level software, like Python. In such way that the final users do not need know microprocessor instructions to be able programming the interface. To easily the diffusion of the interface, we make free to everyone the layout of print circuit board of the interface. The interface is based in a microcontroller, PIC18F2550, made by Microchip. Was developed an hardware programmer for PIC18F family, based on PicKit2 from Microchip. The programmer will allow those people that want develop a deeper microcontroller programming of PIC18F to dump his software on the microcontroller and do testes. An application is presented: a temperature water control using the interface and Python language.

Keywords

Microcontroller, PIC18F2550, I/O interface.

LISTA DE ILUSTRAÇÕES

Figura 1 - Modelo básico de um microcontrolador.....	18
Figura 2 – Arquitetura Von-Neumann	19
Figura 3-Arquitetura Harvard	19
Figura 4 - PWM com filtro passa baixa ou saída analógica.	22
Figura 5 - <i>Protoboard</i> ou matriz de contatos	26
Figura 6-Diagrama de pinos do PIC18F2550	27
Figura 7- Programador do microcontrolador PIC18F	29
Figura 8 - Versão preliminar da placa programadora montada em <i>protoboard</i>	30
Figura 9 - Foto da placa programadora do PIC18F da Microchip.....	30
Figura 10-Desenho da placa programadora do PIC18F da Microchip.....	31
Figura 11 - Montagem do PIC18F2550 para testes.....	37
Figura 12 – Placa com componentes mínimos para funcionamento do PIC18F2550.	38
Figura 13 - Desenho da placa de desenvolvimento para o PIC18F2550.	39
Figura 14 - Diagrama elétrico da placa de desenvolvimento PicPampa.	40
Figura 15 -Arquitetura do PicPampa	46
Figura 16 - Dispositivo encontrado no Ubuntu 12.04	47
Figura 17 - Esquemático regulador de tensão.	50
Figura 18 – Foto do circuito regulador de tensão.	51
Figura 19 – Sensor de temperatura LM35.	51
Figura 20 - Fotografia do sensor de temperatura LM35.	52
Figura 21 - Esquemático relé.....	52
Figura 22 - Placa de desenvolvimento PicPampa, com destaque no relé.	53
Figura 23 - Reforçando as conexões das trilhas do Relé.	53

LISTA DE TABELAS

Tabela 1– Descrição dos pinos do barramento ICSP	34
Tabela 2 - Descrição de todos os pinos utilizada na placa PicPampa.	36
Tabela 3 – Código para a leitura das entradas digitais	43
Tabela 4 – Código para acionamento das saídas digitais	43
Tabela 5 – Código para a leitura das entradas analógicas	44
Tabela 6 – Código para o controle das saídas analógicas	44

ABREVIATURAS

Sigla	Significado
ADC	<i>Analog-to-digital converter</i>
BPS	<i>Bits per second</i>
CDC	<i>Communications Device Class</i>
CISC	<i>Complex Instructions Set Computer</i>
CPU	<i>Central Processing Unit</i>
Eagle	<i>Easily Applicable Graphical Unit</i>
EEPROM	<i>Electrically Erasable Programmable Read Only Memory</i>
GPIO	<i>General Purpose Interface Bus (IEEE-488)</i>
HID	<i>Human Interface Device</i>
I/O	<i>Input/Output</i>
ICSP	<i>In Circuit Serial Programmer</i>
IDE	<i>Integrate Development Environment</i>
LS	<i>Low Speed</i>
LVP	<i>Low Voltage Programming</i>
MLA	<i>Microchip Libraries of Applications</i>
MSC	<i>Mass Storage Device</i>
PC	<i>Personal Computer</i>
PCI	<i>Peripheral Component Interconnect</i>
PCMCIA	<i>Personal Computer Memory Card International Association</i>
PIC	<i>Peripheral Interface Controller</i>
PLL	<i>Phase Locked Loop</i>
PROM	<i>Programmable Read Only Memory</i>
PWM	<i>Pulse Width Modulation</i>
PxI	<i>PCI eXtensions for Instrumentation</i>
RAM	<i>Random Access Memory</i>
RISC	<i>Reduced Instructions Set Computer</i>
ROM	<i>Read Only Memory</i>
USART	<i>Universal Synchronous Asynchronous Receiver Transmitter</i>
USB	<i>Universal Serial Bus</i>
WDT	<i>Watchdog Timer</i>

SUMÁRIO

1	INTRODUÇÃO	13
2	OBJETIVOS.....	16
2.1	Geral	16
2.2	Específico.....	16
3	FUNDAMENTAÇÃO TEÓRICA DOS MICROCONTROLADORES	17
3.1	Conceito de microcontrolador	17
3.2	Arquitetura.....	18
3.2.1	Von-Neumann.....	19
3.2.2	Harvard.....	19
3.2.3	Memória de dados.....	20
3.2.4	Memória de programa	20
3.2.5	CISC x RISC	20
3.3	Periféricos.....	21
3.3.1	Portas de entrada e saída digitais.....	21
3.3.2	Modulação por largura de pulso (PWM)	22
3.3.3	Conversor Analógico Digital (ADC).....	23
3.4	USB	23
3.4.1	Etapas de sincronização	24
3.4.2	Classes de dispositivos	24
3.5	Métodos de gravação do <i>firmware</i>	25
3.6	<i>Protoboard</i>	26
3.7	Características do PIC18F2550	27
4	PLACA DO PROGRAMADOR	28
4.1	Metodologia	28
4.2	Resultados e discussões	28
5	PICPAMPA	35
5.1	METODOLOGIA	35
5.2	Resultados e discussões	37
6	FIRMWARE	42
6.1	Metodologia	42
6.2	Resultados e discussões	42
7	ROTINAS DE CONTROLE.....	46

7.1	Metodologia	47
7.2	Resultados e discussões	47
8	APLICAÇÃO	49
8.1	Aquecedor de água.....	49
8.2	Material utilizado.....	49
8.3	Metodologia e implementação	50
8.4	Resultados e discussões	55
9	CONCLUSÕES	56
10	TRABALHOS FUTUROS.....	57
	REFERÊNCIAS.....	58
	APÊNDICE A - MANUAL DE USO DA INTERFACE GRÁFICA	62
	APÊNDICE B - ERRATA	64
	APÊNDICE C - <i>LAYOUT</i> DA PLACA GRAVADORA.	65
	APÊNDICE D - <i>LAYOUT</i> DA PLACA PICPAMPA.....	66
	APÊNDICE E - ROTINAS DE CONTROLE DO PIC PAMAPA	67
	APÊNDICE F - INTERFACE GRÁFICA DE CONTROLE DO PIC PAMAPA	69
	APÊNDICE G – SELEÇÃO DA TENSÃO DE REFERÊNCIA DO ADC	76
	ANEXO A – CARACTRÍSTICA DO PROTOBOARD	77

1 INTRODUÇÃO

A utilização de computadores para automatização de diversos processos em quase todos os ramos da atividade humana (indústrias, comércio, educação, *hobby*) tem crescido de forma significativa. Os benefícios advindos da introdução da computação nestes processos têm incrementado a produtividade e reduzido às possibilidades de falhas. Uma das aplicações que se destacam é a utilização de computadores para monitorar os mais diversos sensores que integram um determinado processo e a partir das informações obtidas enviarem sinais de controle para outros dispositivos, ou gerar relatórios técnicos para posterior análise.

Para fazer a interface entre um computador e o “mundo externo” utilizam-se as chamadas placas de aquisição de dados, que conjugam todos os dispositivos necessários ao interfaceamento entre o PC (*Personal Computer*) e os mais diversos dispositivos externos.

Dentre os vários fabricantes de interfaces, pode-se citar a National Instruments¹ [NI 2012]. Os fabricantes oferecem soluções de *hardware* e *software* proprietários que são de difícil acesso a leigos, tanto do ponto financeiro, como do conhecimento técnico. Os formatos de *hardware* são diversos, indo desde barramentos dedicados (PXI, GPIB, PCMCIA, ISA, PCI, Serial – RS232, Paralela, USB – *Universal Serial Bus*, entre outras) aos padronizados. Muitos destas interfaces não estão mais disponíveis nos computadores pessoais, sendo a interface USB a que está se consolidando como padrão. Hoje em dia alguns notebooks estão vindos somente com a interface física USB.

Utilizando um microcontrolador conectado a USB do computador, é possível captar sinais de entrada, monitorar e controlar um sistema. Para usar um microcontrolador, o código a ser desenvolvido deve ser escrito, testado, e armazenado na memória de dados do microcontrolador. Normalmente o *software* é escrito e compilado em um PC e então carregado para a memória de dados com o código de máquina [HUANG 2005]. Portanto, para que haja uma comunicação eficiente do computador com o mundo externo, através de sinais analógicos, se faz necessário o uso de uma interface de *hardware*, que por sua vez necessita

¹ National Instruments fundada na década de 70 e desde então tem produzido soluções para automação de equipamento de testes e software de instrumentação virtual.

de um *software* específico para que esta interface comunique-se com outros programas e aplicativos.

Os requisitos selecionados para o *hardware* são: ser composto por componentes de fácil acesso, baixo custo e simplicidade de montagem, permitir comunicação com o computador através da porta USB, o que evita a instalação de uma placa dentro do computador e possibilitando o controle através de um *notebook*.

Os requisitos selecionados para o *Software* são: comunicação com o computador simples e transparente ao usuário e deve permitir a integração com outros programas de alto nível (Python por exemplo).

Para a solução de *hardware* é proposto a utilização de um microcontrolador da Microchip² da família 18F, o PIC18F2550 que reúne em um único chip:

1) Encapsulamento DIL (*dual in line*) de 28 pinos, um invólucro padrão que permite fácil manipulação mecânica e permite fácil montagem elétrica;

2) Possibilidade de entradas e saídas digitais, o que permite a conexão de sensores, os quais fornecem sinais pré-processados e na forma digital. A saída digital permite o acionamento de atuadores e/ou sinalizadores para acompanhamento e controle de processos;

3) A entrada analógica permite a comunicação com o mundo analógico, seja lendo sensores e/ou monitorando níveis de tensão elétrica;

4) Duas saídas de pulso modulado (PWM - *Pulse Width Modulation*) o qual permite simular saídas analógica e desta forma controlar cargas de forma linear;

5) Conexão USB, que permite a conexão com a grande maioria dos computador pessoais (portátil ou não), utiliza a alimentação de 5V presente na USB, eliminando a necessidade de uma fonte de alimentação externa. Esta solução permite facilidade em replicação e exige um reduzido conhecimento de eletrônica do usuário.

² Microchip fundado na década de 80 desde então é fabricante de microcontroladores, memórias e componentes analógicos.

Para a solução de *software*, foi implementado rotinas de controle (leitura, escrita, etc.) do *hardware* que será integrado em linguagem *Python* [MATLOFF 2007], programa de alto nível utilizado para a solução em *software* e testes.

Um trabalho correlato é o do Santos [SANTOS 2009] nesse é mostrado uma estratégia de comunicação USB com o PIC18F4550 bem como a confecção artesanal de uma placa de circuito impresso que foi utilizada como placa de desenvolvimento. Porém este mostra apenas a parte do *firmware*, e para o controle da placa de desenvolvimento é necessário uma configuração prévia do *hardware* utilizado.

Este trabalho está organizado da seguinte forma: no Capítulo 2 são listados os objetivos gerais e específicos propostos. O Capítulo 3 apresenta uma fundamentação teórica para o embasamento sobre microcontroladores (conceito e arquitetura), alguns periféricos (PWM, ADC e USB) e principais características do PIC18F2550. Os capítulos 4, 5, 6 e 7 estão estruturados da seguinte forma: inicia com a metodologia utilizada, segue descrevendo o circuito ou funções utilizadas, e finaliza com resultados e discussões. O capítulo 4 trata da concepção e construção de um programador que foi adaptado a partir do programador PicKit2 da Microchip. No capítulo 5 é mostrado o projeto da placa de desenvolvimento (PicPampa). O capítulo 6 apresenta o desenvolvimento e funcionamento do *firmware* embarcado no PIC18F2550. No capítulo 7 são mostrados as rotinas de acesso ao PicPampa. O capítulo 8 apresenta uma aplicação utilizando o PicPampa e o Python, que é um controlador de temperatura de água. No capítulo 9 são listadas as conclusões deste trabalho. E no capítulo 10 são listadas sugestões para trabalhos futuros.

2 OBJETIVOS

2.1 Geral

Desenvolver um sistema de aquisição de dados e controle cujo *hardware* seja simples e de fácil montagem por parte do usuário e de fácil integração a um *software* de programação de alto nível. Tanto a construção e utilização do *hardware* como a utilização do *software* devem ser acessíveis a pessoas com pouca experiência em programação e em eletrônica.

2.2 Específico

- i) Desenvolver, montar e testar uma placa gravadora para a família de microcontroladores PIC18F da Microchip.
- ii) Instalar as ferramentas de *software* necessárias para a comunicação do computador com a placa programadora e comunicação com a placa de desenvolvimento
- iii) Desenvolver, montar e testar uma placa de desenvolvimento (PicPampa).
- iv) Programar rotinas de comunicação via USB entre a placa PicPampa e o Python, programa de alto nível rodando no computador.

3 FUNDAMENTAÇÃO TEÓRICA DOS MICROCONTROLADORES

Nesta seção serão descritos alguns fundamentos teóricos necessários para o leitor iniciante em sistemas digitais. Compõe-se de subsídio para acompanhar a descrição dos desenvolvimentos realizados para se atingir os objetivos citados na seção 2. Para o leitor que possuir conhecimentos básicos de eletrônica digital, caso sinta-se a vontade, pode prosseguir nas seções seguintes.

3.1 Conceito de microcontrolador

O termo microcontrolador é usado para descrever um sistema que tem no mínimo um microprocessador, uma memória de programa, uma memória de dados e dispositivos de entrada e saída chamados de periféricos. Alguns microcontroladores possuem componentes adicionais como temporizadores, contadores, conversores analógicos digitais (ADC -- *Analog-to-digital converter*), PWM, comparadores, interface serial e inclusive alguns com interface *universal serial bus* (USB).

Um microcontrolador se diferencia de um processador pelos componentes nele integrados e também na frequência de *clock*. Enquanto a frequência de um processador chega à ordem de giga-hertz, microcontroladores possuem frequência na ordem de dezenas de mega-hertz. Um sumário com os microcomputadores mais utilizados pode ser encontrado em [W M 2013]. Permitindo consumo energético e custos menores do que um processador. O que o torna atraente para aplicações onde não é necessário muito poder de processamento, como por exemplo: controle de motores, eletrodomésticos, eletrônica embarcada em automóveis e etc, [GUIMARÃES 2010; MICROCHIP ADC 2013].

3.2 Arquitetura

A arquitetura de um microcontrolador é constituída basicamente por um CPU (*Central Processing Unit*), memória de dados, memória de programa, vários periféricos, e um barramento onde todos os periféricos e memórias são interligados [PREDKO,2008], a visão geral da arquitetura é mostrada na Figura 1. Existem duas arquiteturas básicas: *Harvard* e *Von-Neumann*, que serão discutidos a seguir.

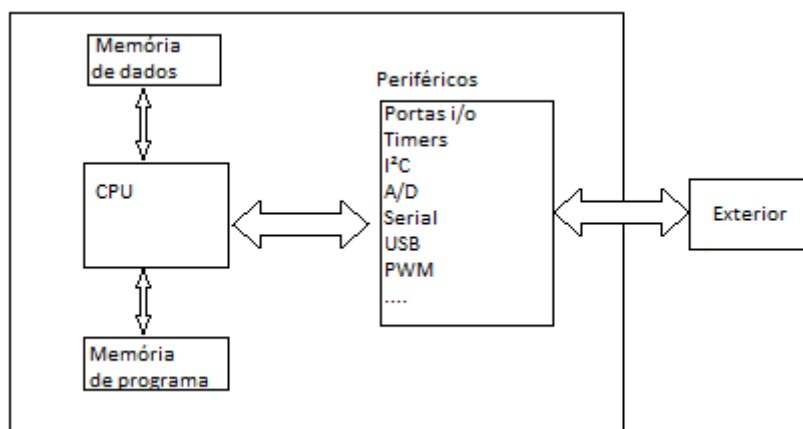


Figura 1 - Modelo básico de um microcontrolador

Fonte: [PREDKO,2008]

3.2.1 Von-Neumann

O modelo de *Von-Neumann* é caracterizado pelas memórias de dados e de programa compartilharem o mesmo barramento para serem acessadas pela CPU, que pode ser observado na Figura 2, no caso dos microcontroladores, acaba limitando a banda de operação. Pois o acesso às duas memórias utiliza a mesma largura de banda de dados [HENNESSY, 2005].

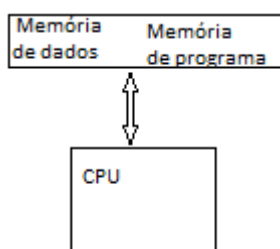


Figura 2 – Arquitetura Von-Neumann

Fonte: [HENNESSY, 2005]

3.2.2 Harvard

O modelo de *Harvard* é caracterizado por possuir barramentos distintos para acesso da memória de programa e a de dados, pode ser observado na Figura 3. Tem a vantagem de permitir que a largura de banda para os barramentos sejam distintas, contudo torna o hardware mais complexo. [MIYADAIRA, 2012]

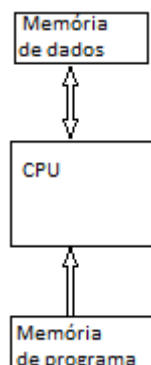


Figura 3-Arquitetura Harvard

Fonte: [HENNESSY, 2005]

3.2.3 Memória de dados

Normalmente é onde fica armazenada variável e constante do sistema, a tecnologia usada nesse tipo de memória normalmente é a RAM (*Random Access Memory*), ela é volátil, portanto a cada vez que a alimentação do sistema é interrompida o conteúdo é perdido. [MIYADAIRA, 2012]

3.2.4 Memória de programa

Local de armazenamento do código de máquina, a tecnologia utilizada é a não volátil, isso significa que os dados não são perdidos quando o sistema for desligado. Normalmente as tecnologias de armazenamento utilizadas são: EEPROM (*Electrically Erasable Programmable Read Only Memory*) e PROM (*Programmable Read Only Memory*).

3.2.5 CISC x RISC

Um microprocessador pode ser do tipo RISC (*Reduced Instruction Set Computer*) ou CISC (*Complex Instruction Set Computer*).

Um microprocessador do tipo CISC é caracterizado por possuir uma quantidade muito grande de instruções na (ordem de centenas), é extremamente versátil, normalmente a arquitetura de memória é a *Von-Neumann*.

Um microprocessador do tipo RISC é caracterizado por possuir o mínimo de instruções, normalmente a arquitetura de memória é a *Harvard*. [BRAHIM, 2012]

3.3 Periféricos

Nessa subseção serão mostrados alguns periféricos encontrados na maioria dos microcontrolador. E também o USB, que não é comum, porém como é usado nesse trabalho este será mostrado.

3.3.1 Portas de entrada e saída digitais

São portas de entrada e saída de dados, cujos valores lógicos podem variar para ‘1’ ou ‘0’. Por essas portas o microcontrolador pode controlar outros dispositivos eletrônicos como *led (light-emitting diode)*, relé, motor e outros, como também monitorar o nível lógico de tensão, nível alto ou baixo. Normalmente uma porta é referente a oito pinos de propósito geral de um microcontrolador. A porta pode assumir várias funções, é necessária a configuração de alguns registradores para se definir a porta como entrada ou saída digital, e qual *bit* da porta acessar, normalmente uma porta tem um *byte*, e cada *bit* é referente a um pino de acesso do microcontrolador [IBRAHIM 2008].

3.3.2 Modulação por largura de pulso (PWM)

O PWM (*pulse width modulation*) consiste em controlar o tempo em que o sinal se mantém em nível alto (*duty cycle* - relação entre o tempo ‘ligado’ e ‘desligado’), ou controlar a largura de pulso, dentro de um determinado período de tempo prefixado (período PWM) [MIYADAIRA, 2012]. Ele é usado para o controle de potência em cargas, por exemplo motores de corrente contínua. Pode ser usado para simular um conversor digital analógico, com o uso de um circuito RC externo, a Figura 4 mostra o diagrama esquemático de um PWM como conversor digital analógico.

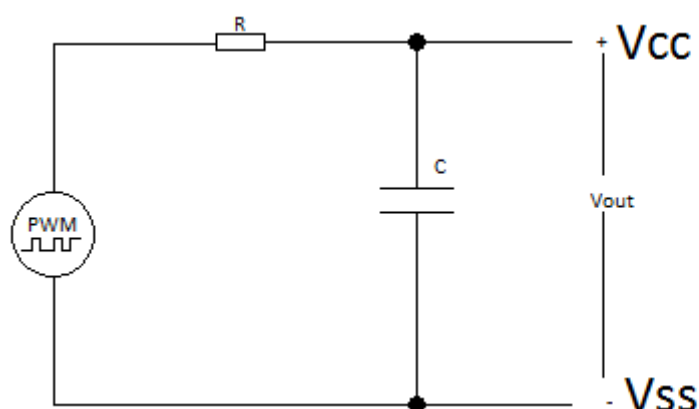


Figura 4 - PWM com filtro passa baixa ou saída analógica.

(Autoria própria)

O sinal analógico vai depender da modulação dos pulsos (PWM), o filtro RC tem a finalidade de filtrar os pulsos PWM. O valor de V_{cc} dependerá da constante RC e do *duty cycle*, quanto menor o *duty cycle*, menor será a tensão de saída, e quando o *duty cycle* se aproxima de 100%, V_{cc} se aproxima do valor máximo de tensão. Uma discussão detalhada pode ser encontrada no “Application Report: SPRA490” [ALTER 1998] da Texas Instruments. Outra abordagem mais simplificada pode ser encontrado em “Arduino’s AnalogWrite – Converting PWM to a voltage” [DANIELS 2011] e um cálculo on-line de filtro pode ser encontrado em [OKAWA 2013].

3.3.3 Conversor Analógico Digital (ADC)

Microcontroladores trabalham com sinais digitais, logo para ter uma representação do mundo exterior (analógico: temperatura, luminosidade, umidade, etc) e poder ser processada é necessário um sensor para transformar uma grandeza física em um sinal elétrico que irá passar por uma conversão analógica digital (ADC). O ADC converte uma grandeza analógica em um valor digital (representação binária) proporcional a ela. Esse valor depende diretamente da resolução e da tensão de referência (V_{ref}) utilizado na conversão, podendo variar de 0 a $[2^{(resolução)}-1]$ unidades.

A qualidade de um ADC não é somente definida pela resolução, mas também pela precisão e acurácia [BALBINOT 2010]. Uma discussão detalhada sobre conversores ADC pode ser encontrada em [ANALOG DEVICES INC. ENGINEERI, 2004].

3.4 USB

O USB (*Universal Serial Bus*) é um dos barramentos mais utilizados para prover comunicação entre periféricos e o PC, como exemplos de periféricos tem-se: impressora, mouse, *web cam* e outros.

Foi projetado para simplificar a comunicação e substituir os inúmeros tipos de cabos e protocolos existentes.

Uma das vantagens do USB, é que ela fornece alimentação para o periférico, evitando o uso de uma fonte externa. A conexão entre o PC e o periférico é formado por um barramento de 4 fios: Dois para alimentação de 5 V e 0V (terra), e mais dois para dados, o D+ e o D-. A comunicação se dá de forma diferencial, utilizando a codificação NRZI (*Non Return to Zero Invert*) [AXELSON 2005].

3.4.1 Etapas de sincronização

Quando um dispositivo USB é conectado ao PC, até que ele seja reconhecido ou instalado pelo sistema operacional, ocorrem cinco etapas. Esse processo é chamado de processo de enumeração.

1. Etapa de conexão: Quando um dispositivo é conectado ao barramento USB, ocorre um desbalanceamento nas linhas D+ e D-, esse desbalanceamento é provocado pelo resistor de *pull-up* do dispositivo, a porta a qual o dispositivo está conectado é temporariamente desligada.
2. Etapa de alimentação: Quando o barramento do *host* (PC), percebe em qual porta que o dispositivo está conectado ele envia um reset durante 10 ms para o barramento, após o *reset* o dispositivo pode começar a drenar até 100 mA do barramento.
3. Etapa padrão: Onde o host começa a recolher informações sobre o dispositivo, e é atribuído um endereço a ele.
4. Etapa de endereçamento: É quando o host atribui um endereço único de sete bits ao dispositivo.
5. Etapa de configuração: Após o host recolher todas as informações necessárias, é que o barramento fornece a energia que é descrita no descritor do dispositivo, a partir dessa etapa o dispositivo está configurado e pronto para ser usado.

3.4.2 Classes de dispositivos

Em uma comunicação USB, dispositivos comuns são agrupados dentro de uma classe padronizada. Isso facilita o reconhecimento de dispositivos padrões tais como mouse, teclado, *pen-drive*, etc, pelo sistema operacional, evitando a necessidade de instalação de um *driver*. A classe padrão utilizada nesse trabalho é a CDC (*communications device class*), ela basicamente emula uma porta serial (RS232) pela USB, a vantagem de se usar essa classe é

que ela é reconhecida como uma porta COM pelo Sistema Operacional. Logo o dispositivo tem a vantagem da conexão *plug and play* da USB, e também da facilidade da comunicação da porta serial.

3.5 Métodos de gravação do *firmware*

Um *firmware* é o conjunto de instruções armazenadas normalmente em memória de programa que serão executadas por algum *hardware*. É possível codificar um *firmware* em linguagem C, desde que o fabricante do microcontrolador forneça um compilador, que contenha uma biblioteca com o código em linguagem de máquina do microcontrolador. É necessário um periférico de gravação conectado entre o PC e o microcontrolador, para a passagem do código de máquina codificado para dentro do microcontrolador.

Alguns microcontroladores têm a capacidade de se reprogramar, ou seja, poder alterar o código de programa, logo tendo uma rotina de controle dentro do microcontrolador, é possível passar um *firmware* de atualização para ele, essa técnica é chamada de *bootloader*. A desvantagem é que o *bootloader* ocupa boa parte da memória de programa, restringindo o tamanho de um *firmware*.

Para cada série de microcontroladores o fabricante oferece solução de gravadores, esses podem ter conexão paralela, serial ou USB com o PC. Nesse trabalho será mostrado o projeto de um gravador, se que se conecta com o PC via USB, e com o microcontrolador via ICSP (*In Circuit Serial Programming*).

3.6 *Proto*board

*Proto*board (ou matriz de contatos) é uma placa com vários furos e conexão interligada e utilizado para montagem ou prototipação de circuitos elétricos. A grande vantagem em se usar uma *proto*board é que não é necessário soldar os componentes, basta encaixar na placa como na descrição da Figura 5, depois do experimento é fácil a retirada dos componentes, e a *proto*board pode ser reutilizada por várias vezes.

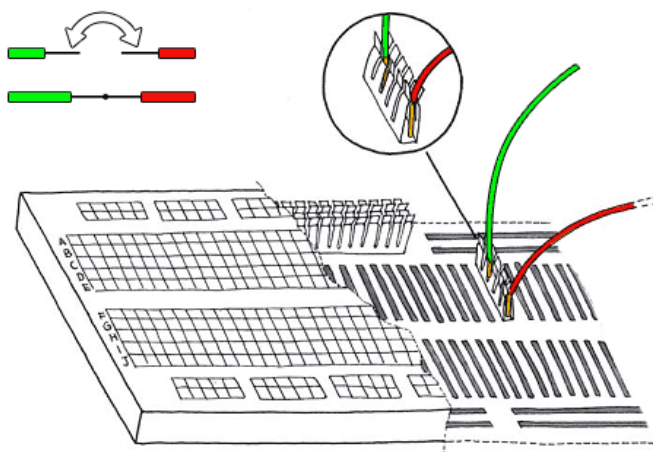


Figura 5 - *Proto*board ou matriz de contatos

Fonte: [Site Arduino]

Deve-se ter um cuidado especial em se utilizar o *proto*board em aplicações que envolvem altas frequências (acima de 10 MHz) ou tempo de transição curto (menores que 100 ns) devido à alta capacitância entre os contatos flexíveis. Também se deve evitar circuitos que envolvam correntes superiores a 3 A, pois pode ocorrer aquecimento devido a forma de conexão dos contatos, conforme ANEXO A.

3.7 Características do PIC18F2550

É um microcontrolador de 8 bits da família 18F da Microchip, formato PDIP e SOIC, de 28 pinos. A base da arquitetura é a *Harvard* com instruções do tipo RISC (Conjunto reduzido de instruções). Pode ser gravado utilizando o método ICSP. Possui 32 kbytes de memória de programa FLASH e 2048 bytes de memória RAM. Ele pode ser alimentado com tensões de 4V a 5,5V. Possui um ADC de 10-bits, e dois PWM de 1 a 10 bits.

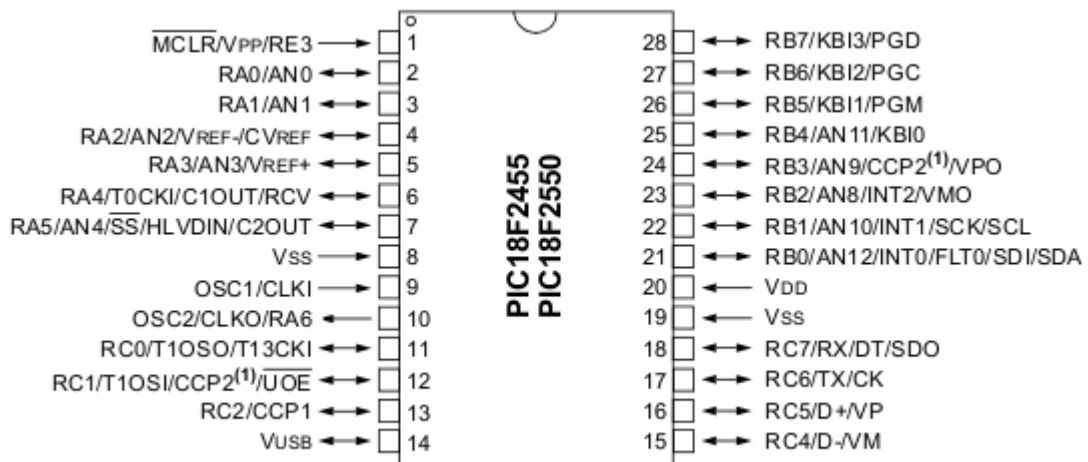


Figura 6-Diagrama de pinos do PIC18F2550

Fonte: [DATASHEET PIC18F2550 - Microchip]

É importante observar que um pino pode ter duas ou mais funções que deve ser definido pelo *firmware* correspondente ao aplicativo desejado, como pode ser observado na Figura 6. Informações mais detalhadas no manual do PIC18F2550 [DATASHEET PIC18F2550 - Microchip].

4 PLACA DO PROGRAMADOR

Para os desenvolvedores interessados em trabalhar com o *firmware* da interface PicPampa, foi desenvolvido um programador, usando como base o PicKit2 da Microchip. Permitindo assim uma maior autonomia na programação do PIC18F2550, evitando-se a dependência com o programador da Microchip. O domínio na construção de um programador, permite oferecer aos usuários interessados em aprofundar os conhecimentos do PIC18F2550, uma possibilidade de *hardware* para a implementação de rotinas em baixo nível desenvolvidas por ele, o usuário. Também contribui no sentido de facilitar a formação de uma comunidade de desenvolvedores para o projeto PicPampa.

4.1 Metodologia

Como a Microchip disponibiliza livremente o diagrama elétrico do PicKit2, foi realizado: um estudo de funcionamento do mesmo, adaptação, quando possível, dos componentes para de montagem de superfície (SMD – *Surface Mounting Device*) para componentes padrão (*through-hole*) disponíveis no mercado nacional, seguido da montagem e teste do protótipo e projeto da placa de circuito impresso.

4.2 Resultados e discussões

Depois de realizado o estudo do diagrama elétrico do PicKit2, foi concebido o projeto do gravador que pode ser visto na Figura 7. Ao longo da discussão dos resultados será realizada uma breve apresentação e descrição do funcionamento do *hardware* do gravador.

Como primeiro teste foi necessário à montagem de um protótipo em *protoboard*, mostrado na Figura 8. Na sequência, a Figura 9 mostra a foto da placa programadora já montada em circuito impresso.

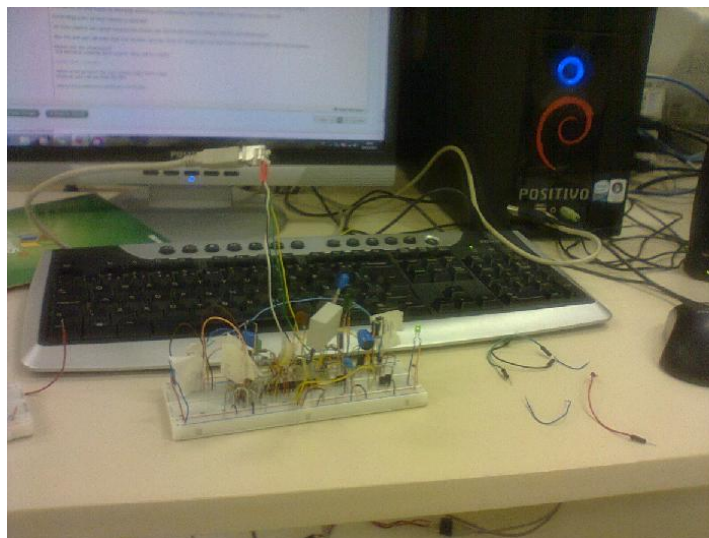


Figura 8 - Versão preliminar da placa programadora montada em *protoboard*.
(Autoria própria)

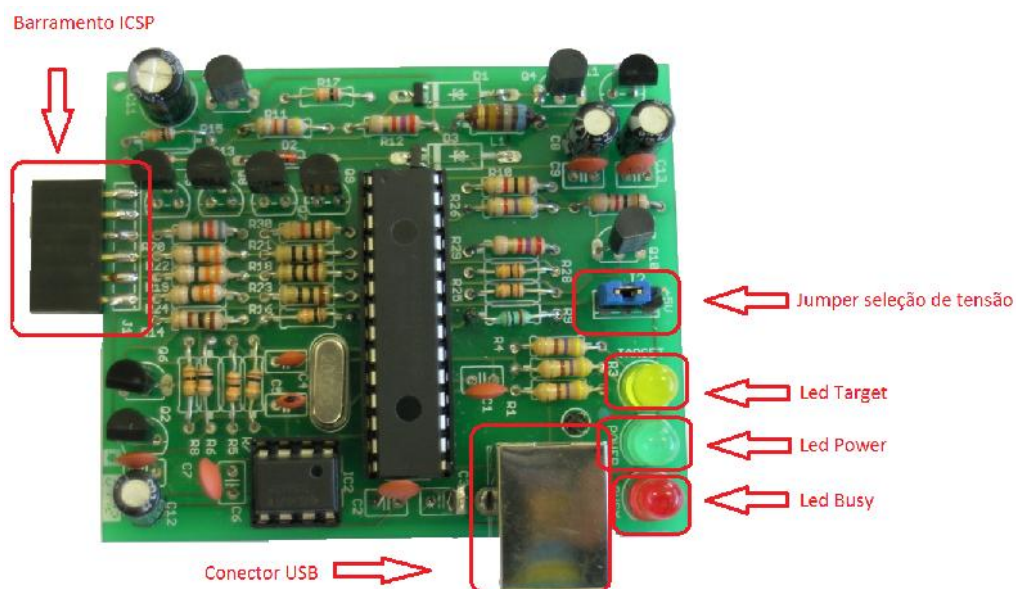


Figura 9 - Foto da placa programadora do PIC18F da Microchip.
(Autoria própria)

Para carregar o *firmware* no microcontrolador é utilizado o barramento ICSP (*In Circuit Serial Programmer*) que é um conector de 6 pinos onde é conectado a placa de

desenvolvimento para a programação do *firmware*. O conector USB é utilizado para alimentação e comunicação com o PC. O *led Power*, verde, indica alimentação (via USB) presente, o *led target*, amarelo, indica que dados estão sendo transferidos para a placa de desenvolvimento e o *led busy*, vermelho, indica que a placa programadora está recebendo dados do computador via USB. O *jumper* seleciona a alimentação do oscilador de 12 V através do regulador de tensão da placa ou através da fonte de 5V diretamente da USB, corresponde ao J2 do esquema de ligações elétricas e está descrito com mais detalhes na descrição do diagrama.

A Figura 10 mostra o desenho da placa programadora que foi desenvolvida utilizando a ferramenta de *software* Eagle, na versão de licença livre (que basicamente limita as dimensões da placa). A placa é de dupla face (camada), sendo que as vias de conexões de furos estão representadas em verde, as trilhas da face superior em vermelho e as trilhas da face inferior em azul. Todos os componentes estão representados também. O *layout* de cada camada pode ser visualizado no Apêndice C - *Layout* da placa gravadora..

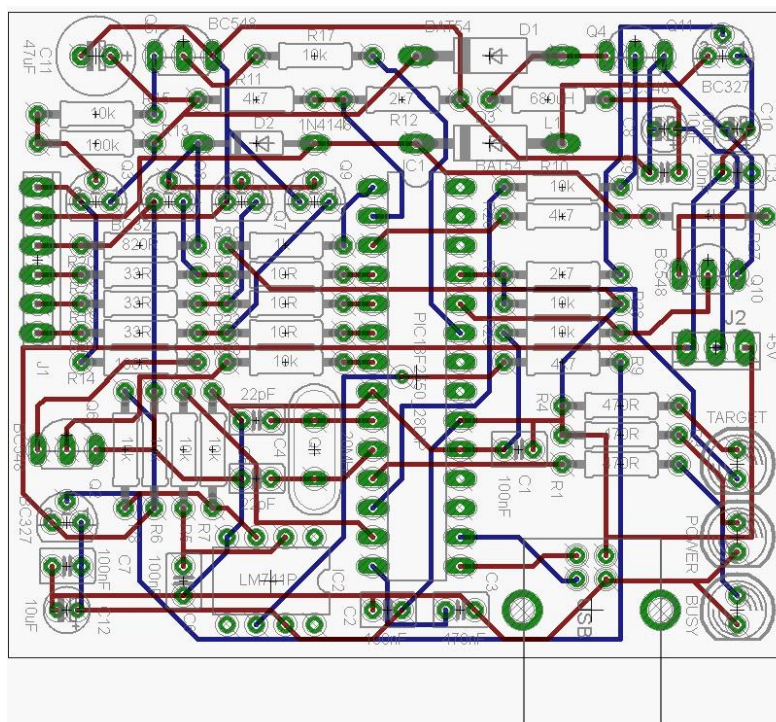


Figura 10-Desenho da placa programadora do PIC18F da Microchip.

(Autoria própria)

A Figura 7 mostra o circuito elétrico que foi desenvolvido a partir do diagrama do Picket2. O programador é baseado no microcontrolador PIC18F2550, o qual gerencia os sinais de escrita e leitura para os microcontroladores da família PIC 12F, 16F e 18F da Microchip, através de barramento de 6 vias (ICSP), proprietário da Microchip, conector J1 da figura 11. A Microchip fornece gratuitamente o *firmware* do programador, bem como ferramentas de software (Mplab e compilador C18) para programação dos PIC's das famílias citadas anteriormente. O programador é conectado ao computador através da interface USB e utiliza alimentação de 5 V do próprio barramento USB. Esta alimentação também é enviada ao microcontrolador que se está programando (“*Target*”). O *firmware* do programador, através de uma tabela, identifica se o “*target*” é alimentado com 3,3 V ou 5V e o *hardware* do programador regula a tensão para o valor correto, a seguir é descrito algumas das etapas executadas pelo programador.

Quando é enviado o sinal de gravação do PIC18F, aparece um potencial de controle no pino 13 (VDD_TGT_ADJ), o que permite o controle de potencial no pino 3 de IC2 de 2,5 V a 5,0 V, e fornece, através de Q2, o potencial V_TGT que alimenta o circuito que libera o potencial VDD_TGT (através de Q11) e alimenta a fonte chaveada que gera 12 V para o pulso de programação, VPP, pino 1 de J1. O V_TGT é controlado através da realimentação formada pelo divisor resistivo (R7 e R8) e o pino 2 de IC2, de aproximadamente metade do valor de V_TGT. Para que o V_TGT passe se chamar VDD_TGT e alimente o “*target*”, os sinais VDD_TGT_P e VDD_TGT_N, pinos 25 e 24 de IC1 respectivamente que habilita Q11 e desabilita Q10 respectivamente. Q10 desabilitado permite que IC1 monitore VDD_TGT através de R26 e pino 3 de IC1 (VDD_TGT_FB). Q11 habilitado libera alimentação ao “*target*” (pino 2 de J1) e desabilita Q7, Q8 e Q9, permitindo a troca de informações, CLK_TGT, DATA_TGT e AUX_TGT, respectivamente, entre o programador e o “*target*”.

Ambos os pinos 24 e 25 de IC1 devem ter nível “zero”, para que tenha potencial no VDD_TGT e acenda o LED TARGET (amarelo). A escolha de D3 é crítica para que o potencial esteja correto. D3 é um diodo *schottky*, com $V_f \sim 0,2$ V. Inicialmente utilizamos o 741 como IC2, contudo observamos que na configuração de fonte de alimentação simples (normalmente é utilizado fonte simétrica) ele não permitia excursão suficiente na tensão de saída para saturar Q2. Idealmente deveríamos utilizar um amplificador operacional “*rail-to-rail*”, contudo optamos por utilizar o TL061 por ser de obtenção relativamente fácil no mercado brasileiro e por ter se mostrado suficiente para a aplicação. O TL061 é um

amplificador operacional de baixo consumo, desenvolvido para aplicações embarcadas, i.e. alimentadas por baterias. Também observamos ser crítica a estabilidade (constância) de V_TGT. No protótipo montado no *proto-board* (Figura 8) foi necessário desconectar de V_TGT a fonte chaveada de 12 V, pois observamos uma leve queda de potencial em V_TGT quando o oscilador de 12 V entrava e isso fazia com que VDD_TGT fosse desabilitado. A alimentação da fonte chaveada foi conectada diretamente ao 5V (VCC) da USB ao invés de conectá-la ao 5V via V_TGT. Esta operação pode ser realizada na placa programadora através da seleção via “*jump*” em J2. É recomendado utilizar a placa programadora com J2 em V_TGT e somente comutar para “+5V” caso encontre dificuldades em proceder com a programação.

Para se obter os 12 V necessários para habilitar a gravação no “*target*”, no VPP_PUMP, pino 12 do pic18F, é gerado um sinal da ordem de 30 kHz, que controla Q4 e gera um pulso de 12 V e largura de 200 ms, aproximadamente. O controle desta tensão se dá através do pino 2 (VPP_FEEDBACK). O pino 23 (VPP_ON) gera um pulso de 5 V que satura Q5 e este Q3, o qual transfere o pulso de 12 V ao pino 1, VPP, do conector de programação J1. O pino 7 de IC1, MCLR_TGT, deve estar no nível “zero”. O pulso de programação, VPP, tem duração de aproximadamente 100 ms e aparece na segunda metade do pulso de 200 ms gerado por Q4. VDD_TGT fica habilitado após a programação do “*target*” e também o LED TARGET.

O sinal de *clock* de 20 MHz pode ser verificado através de um osciloscópio conectado ao pino 10 através de uma ponta de prova de baixa capacitância (x10 -> 10 Mohm e 13 pF), o sinal medido é quase senoidal entre 0V e 4Vp. C4 e C5 não são muito críticos e em alguns casos utilizamos 18pF sem problemas.

Para os testes de sinais foi utilizado o Mplab (Microchip) e carregado o *firmware* versão 2.32, utilizando um PICkit2 da Microchip.

Os sinais do conector J1, 6 pinos (porta ICSP), de programação, são descritos na Tabela 1.

Número do pino	Nome	Descrição
1	VPP	Pulso de 12 V, habilitar gravação.
2	VDD	Alimentação 5V
3	GND	Alimentação 0V
4	DATATGT	Dados do pic de desenvolvimento
5	CLKTGT	<i>Clock</i> do pic de desenvolvimento
6	AUXTGT	Função de <i>debug</i> , mas não utilizado no projeto

Tabela 1– Descrição dos pinos do barramento ICSP

Fonte: [DATASHEET PIC18F2550 - Microchip]

Foram realizados testes de gravação com alguns Pics das famílias: 12F, 16F e 18F, com sucesso em todos eles.

5 PICPAMPA

Para facilitar a utilização do microcontrolador PIC18F2550 como interface para PC via USB, foi desenvolvido uma placa de circuito impresso (PicPampa). A placa oferece possibilidade de funcionalidades na forma de opcionais de *hardware*, como chave de pressão de teste, *led* de teste, amplificador operacional e relé. A montagem destes componentes adicionais é opcional dependendo da necessidade da aplicação.

5.1 METODOLOGIA

A ferramenta utilizada para o projeto da placa PicPampa, foi o Eagle, o diagrama esquemático foi montado de acordo com o *datasheet* do microcontrolador PIC18F2550, o circuito foi projetado para receber gravação pelo barramento ICSP.

Antes do projeto da placa de desenvolvimento foi necessário definir as funções de cada pino do PIC18F2550, pois cada pino pode ter uma ou mais funções. Este é definido pelo *firmware* e a tabela 2 descreve as funções dos pinos selecionados para o PicPampa:

Tipo do barramento	Pino	Função
-	1	VPP ou reset
Entrada Analógica	2	AN0
Entrada Analógica	3	AN1
Entrada Analógica	4	AN2
Entrada Analógica	5	VREF+
Entrada Digital	6	RA4
Entrada Digital	7	RA5
-	8	VSS / GND
-	9	OSC1 Oscilador 20Mhz
-	10	OSC2 Oscilador 20Mhz
Entrada Digital	11	RC0
Saída Analógica	12	CPP2
Saída Analógica	13	CPP1
-	14	VUSB
-	15	D-
-	16	D+
Entrada Digital	17	RC6
Entrada Digital	18	RC7
-	19	VSS / GND
-	20	VDD / VCC
Saída Digital	21	RB0
Saída Digital	22	RB1
Saída Digital	23	RB2
Saída Digital	24	RB3
Saída Digital	25	RB4
Saída Digital	26	RB5
Saída Digital	27	RB6 ou PGC
Saída Digital	28	RB7 ou PGD

Tabela 2 - Descrição de todos os pinos utilizada na placa PicPampa.

(Autoria própria)

O código é executado na IDE Mplab e testada na própria IDE, caso não haja problema, é convertido em linguagem de máquina e utilizando o gravador este é carregado no microcontrolador como *firmware*.

5.2 Resultados e discussões

Para os testes iniciais, foi montado em *protoboard*, conforme a Figura 11 com a configuração mínima para funcionamento do PIC18F2550, suficiente para gravar o código de máquina do *firmware*.

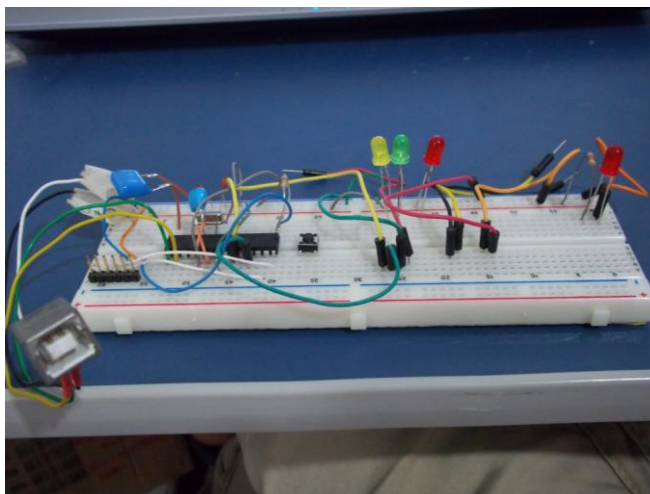


Figura 11 - Montagem do PIC18F2550 para testes.
(Autoria própria)

Após os testes no protoboard, foi produzido a placa de circuito impresso do PicPampa, a Figura 12 mostra a foto da placa montada na configuração mínima para funcionamento. Ela foi desenvolvida utilizando-se a ferramenta de *software* Eagle, na versão de licença livre. A placa é de dupla face, sendo que as vias de conexões de furos estão representadas em verde, as trilhas da face superior em vermelho e as trilhas da face inferior em azul, essas conexões são mostradas na Figura 13. Todos os componentes estão representados, porém para a configuração mínima é necessária uma pequena porção de componentes, conforme descrição elétrica que segue abaixo. A placa ainda possui uma área de prototipagem formada pelo conjunto de ilhas a esquerda da placa, conforme a Figura 12.

Ao centro da placa existe um pequeno bloco de três terminais (imediatamente à esquerda de IC1) em que se deve inserir um “*jump*” no terminal do centro e o da esquerda para gravar o *firmware* no PIC e depois mudar o “*jump*” para terminal central e o terminal da

direita para a utilização da placa de desenvolvimento. Na parte superior e ao centro, pode-se ver o conector de seis pinos (porta ICSP) que se conecta a placa programadora.

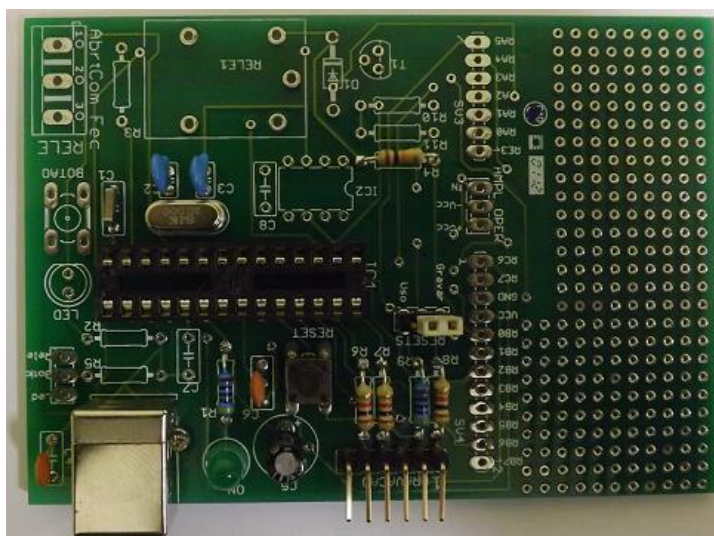


Figura 12 – Placa com componentes mínimos para funcionamento do PIC18F2550.
(Autoria própria)

A configuração mínima para a placa de desenvolvimento compreende os seguintes componentes: IC1 (PIC18F2550), que é o microcontrolador; Q1 (cristal de 20 MHz), C2 = C3 (22 pF) compõe a base de tempo, o cristal é necessário para a estabilidade da comunicação via USB;. Os valores de C2 e C3 não são críticos, podendo ser substituídos por 15 pF ou 18 pF, mesmo o valor do cristal pode ser diferente (12 MHz, por exemplo), sendo necessário a configuração via software. C1 está relacionado com a USB e também não é muito crítico, podendo ser de 0,1 a 0,47 uF. Os resistores R6 = R7 = R8, são resistores de passagens e protegem as linhas de comunicação (VPP, PGD e PGC) com o gravador (PicKit2 da Microchip, por exemplo). Eventualmente estes resistores podem ser omitidos e conectados uma ponte (“*jump*”) nas respectivas posições.

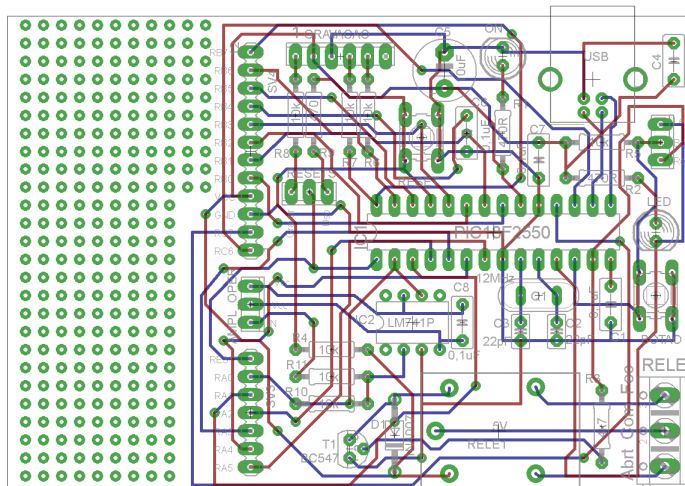


Figura 13 - Desenho da placa de desenvolvimento para o PIC18F2550.

(Autoria própria)

O último circuito necessário para o funcionamento do PIC é o circuito de “reset”: Chave de pressão “reset” + R4 + R9 + C6. Quando a placa é alimentada, C6 está descarregado e força nível lógico zero no pino 1 do IC1, via R9, garantindo o estado de inicialização do PIC ao conectar a fonte. O capacitor C6 é carregado por R4, fazendo com que o sinal de “reset” seja retirado do PIC, através da mudança de nível lógico “zero” para “um”. Ao se pressionar a chave de “reset”, C6 é descarregado novamente e o PIC é levado a condição de inicialização (“reset”). Os conectores SV3 e SV4 permitem acesso às linhas de controle e comandos do PIC. C4, C5 e C7 são filtros da fonte de alimentação. O led “ON” e R1 indicam que a placa está alimentada, esses componentes podem ser omitidos, porém é interessante saber se a placa está ou não alimentada.

Para simulação de comandos simples, foi previsto na placa um “LED” conectado a alimentação positiva através de R2 (terminal RC0), portanto acionado com nível lógico “zero”, que pode ser usado como um dispositivo simples de saída (indicação visual de um nível lógico, “zero” ou lógica invertida, neste caso). O “LED” está conectado ao pino 11 de IC1. Um botão é previsto na placa para simular um sinal de entrada ou uma interação do mundo físico externo com o PIC. O circuito é composto por: “BOTÃO” e R5 conectado a alimentação, que garante nível lógico “um” enquanto a chave não é pressionada. Quando a chave “BOTÃO” é pressionada, leva nível “zero” ao pino 12 de IC1, indicando que um evento ocorreu. Este é conectado no terminal RC1 do PIC.

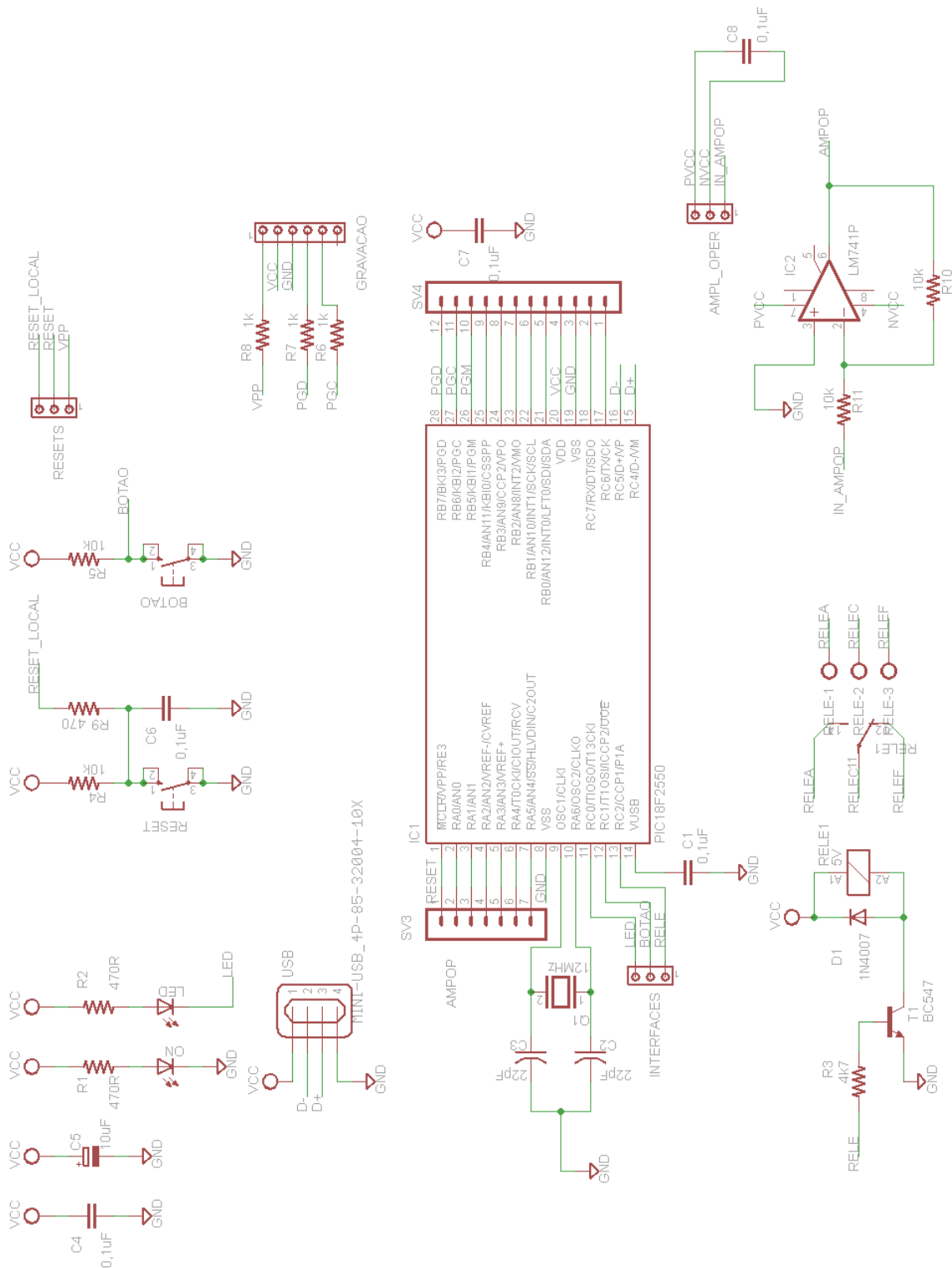


Figura 14 - Diagrama elétrico da placa de desenvolvimento PicPampa.

(Autoria própria)

Outras funções um pouco mais complexas estão previstas na placa de desenvolvimento: caso a aplicação necessite acionar um relé (terminal RC2), para controlar um dispositivo de potência média ou alta, por exemplo, um relé pode ser instalado na placa: “RELE 1” + T1 + R3 + D1. O relé pode ser acionado através de qualquer pino de saída digital dos conectores SV3 e SV4, conectado ao terminal “RELE” de R3. O relé aciona com nível “um”, que satura T1 através de R3, que limita a corrente de base de T1. T1 saturado, faz passar corrente no “RELE 1”, que aciona os contatos do mesmo, permitindo acionar cargas de média e alta potência através dos contatos “RELE C”, contato comum, “RELE F”, contato normalmente fechado e “RELE A”, contato normalmente aberto. Diodo D1, que normalmente está inversamente polarizado, protege T1 quando o relé é desligado. Quando o campo elétrico na bobina colapsa, gera uma diferença de potencial (ddp) de polaridade inversa, que é conduzida através de D1. Se o relé possuir bobina de alta resistência, máximo de 20 mA em 5 V, pode ser conectado diretamente ao PIC, sem a necessidade de T1 e R3, basta conectar a posição do coletor de T1 a um dos pinos de saída de um dos conectores SV3 ou SV4 e neste caso o relé é acionado com nível “zero”.

Caso seja necessário medir sinais com amplitudes da ordem de dezenas ou centenas de milivolts, está previsto na placa de desenvolvimento um amplificador operacional (AmpOp) , IC2, que pode ser o tradicional 741 ou qualquer outro de aplicação específica, que siga a mesma ordem de pinos do 741. O AmpOp está na configuração inversora e os resistores R11 e R10 definem o ganho (G) de tensão, $G = (R10)/(R11)$. É possível configurar o AmpOp no modo não inversor, contudo é necessário uma pequena modificação na placa: interromper o pino 3 do terra, entrada no pino 3 e conectar a entrada IN_AMP para terra, ganho de $[1+(R10/R11)]$. A saída do AmpOp, pino 6, pode ser conectado a uma das entradas analógicas do PIC através de SV3 ou SV4.

No pino 13 (PWM1, terminal RC2 - o mesmo que o BOTÃO), foi configurado e conectado para teste um circuito RC (1 k Ω e 1 μ F) e ajustado para a tensão de 2,5 V. Mediu-se um ruído de 49 mVpp a uma frequência de 60 kHz. Caso queira utilizar o BOTÃO da placa PicPampa, pode-se utilizar o PWM2, terminal RB3, pino 24.

6 FIRMWARE

6.1 Metodologia

O *firmware* foi desenvolvido em C utilizando o compilador C18 da MICROCHIP e a IDE (*Integrate Development Environment*) é o Mplab 8.87. Foi utilizado algumas bibliotecas fornecida pela Microchip para dar suporte a comunicação USB e escolhido a classe CDC, que emula uma porta serial, isso facilita o reconhecimento do dispositivo USB (PicPampa) pelo sistema operacional sem a necessidade de instalação de *driver*.

A estratégia de implementação do *firmware* é feito em cima de uma interrupção de alta prioridade vinda da USB, a cada interrupção, é verificada qual o comando que foi enviado via USB, este é decodificado e processado de acordo com a função requerida (entrada ou saída).

6.2 Resultados e discussões

Foram desenvolvidas quatro rotinas que estão listadas nas tabelas (3, 4, 5 e 6), que são: entrada digital, saída digital, entrada analógica e saída analógica.

Entrada digital: este verifica o estado atual do terminal requisitado, que pode ser RA4, RA5, RC0, RC6 e RC7. Escreve no *buffer* de saída do PIC18F o estado lógico lido, disponibilizando esta informação para o Python. O código correspondente a ser enviado pelo Python em formato ASCII está listado na Tabela 3, que segue.

Sinal em formato ASCII	Terminal Correspondente
J	RA4
K	RA5
O	RC0
P	RC6
Q	RC7

Tabela 3 – Código para a leitura das entradas digitais

Saída digital: verifica o estado atual do terminal (RB0 até RB7) solicitado, muda o estado lógico do terminal selecionado e retorna o estado atual como resposta ao comando, escrevendo no *buffer* de saída do PIC18F. O código correspondente a ser enviado pelo Python em formato ASCII está listado na Tabela 4, que segue.

Sinal em formato ASCII	Terminal Correspondente
A	RB0
B	RB1
C	RB2
D	RB3
E	RB4
F	RB5
G	RB6
H	RB7

Tabela 4 – Código para acionamento das saídas digitais

Leitura analógica: esta rotina realiza 256 leituras, calcula a média entre essas leituras e escreve o resultado correspondente no *buffer* de saída do PIC18F. Retornar uma *String* de 4 posições (“0000” até “1023”), sendo que no Python precisa converter essa *string* em numeral. São quatro entradas analógicas (RA0 até RA3), que pode ser multiplexadas e enviadas ao único ADC de 10 bits. O sinal analógico pode ser lido pelos terminais (RA0, RA1 e RA2). O terminal RA3 é reservado para leitura de um potencial de referência, também é possível utilizar a referencia interna de 5 V (fonte de alimentação via USB), para isso é necessário a modificação no código fonte da leitura analógica, conforme o Apêndice G – Seleção da tensão de referência do ADC. A rotina atual está ajustada para ler uma tensão de referência externa, esta leitura é realizada quando a placa PicPampa é alimentada ou quando é pressionado o

botão de *reset*, somente. Caso queira utilizar a fonte de alimentação externa de 5 V como referência, basta conectá-la ao terminal RA3. O código correspondente a ser enviado pelo Python em formato ASCII está listado na Tabela 5, que segue.

Sinal em formato ASCII	Terminal Correspondente
N0	RA0
N1	RA1
N2	RA2
v	RA3 (Vref)
V	V _{USB}

Tabela 5 – Código para a leitura das entradas analógicas

Saída analógica: O comando aciona uma das duas portas PWM de 10 bits. O comando consiste em enviar um par de caracteres, que são: “P1”, incrementa em uma unidade no PWM1; “p1”, decrementa em uma unidade no PWM 1; “P2”, incrementa em uma unidade no PWM2; “p2”, decrementa em uma unidade no PWM 2. A unidade corresponde a $V_{DD} / 1023$, isto é, caso a tensão (V_{USB}) seja de 5,00 V o valor da unidade corresponde a 4,88mV. Dentro da rotina existe um controle para que o valor superior não exceda 1023 (5 V) e que não seja inferior a 0000 (0 V). Para se obter o sinal analógico é necessário utilizar um filtro RC, por exemplo $R = 1 \text{ K } \Omega$ e $C = 1 \text{ } \mu\text{F}$. O código correspondente a ser enviado pelo Python em formato ASCII está listado na Tabela 6, que segue.

Sinal em formato ASCII	Terminal Correspondente
P1	RC1
P2	RC2

Tabela 6 – Código para o controle das saídas analógicas

Para a comunicação USB, foi escolhida a classe CDC e configurada da seguinte forma:

- BaudRate: 115.200 bps;
- Bit de parada: 1;
- Bit de paridade: sem;
- Número de bits do dado: 8 bits;

A velocidade máxima possível é de 230.400 bps(*bits per second*) utilizando a classe CDC. Na tentativa de se utilizar a velocidade de 230.400 bps, foi encontrado problemas de sincronização na comunicação. Para contornar este problema foi inserindo um tempo máximo de espera (1 ms) na resposta da troca de informações entre o PC e a placa PicPampa, pois foi observado que o Python aguardava indefinidamente uma resposta do microcontrolador. Caso o tempo de espera seja zero (sem o *timeout*), a comunicação é estável até 9.600 bps.

7 ROTINAS DE CONTROLE

Para a comunicação entre o PC e a placa PicPampa, foi necessário desenvolver uma rotina de controle utilizando o Python. A arquitetura geral da interface é mostrada na Figura 15, é possível perceber as várias camadas de software para o controle do periférico.

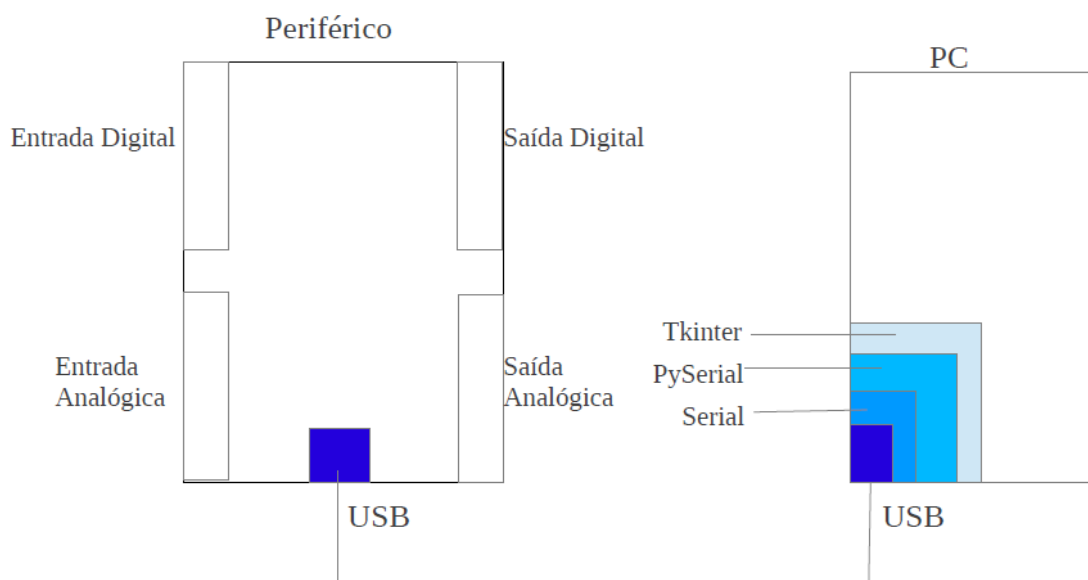


Figura 15 -Arquitetura do PicPampa
(Autoria própria)

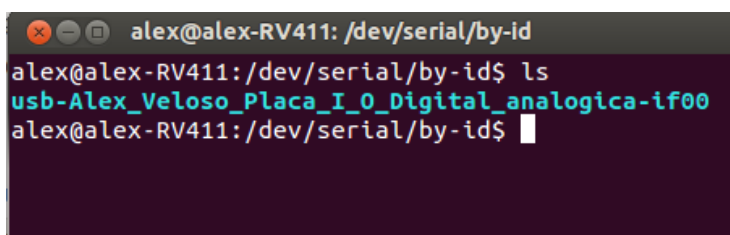
A camada inferior é composta pelo módulo USB que é responsável pela comunicação física e está instalada tanto na placa PicPampa (periférico) como no PC, em seguida encontra-se a camada Serial (RS232), que é o protocolo utilizado para a comunicação. O PySerial é a interface do Python para comunicação serial e o TKinter é a camada para a interface gráfica, opcional.

7.1 Metodologia

Para o desenvolvimento da rotina de comunicação USB, no PC, foi utilizado as funções que estão na biblioteca PySerial e utilizado o Python na sistema operacional Ubuntu 12.04. Para uma verificação rápida de todos os comandos da placa PicPampa foi desenvolvido uma interface gráfica utilizando a biblioteca Tkinter. No sistema operacional Windows é possível realizar a mesma verificação utilizando uma ferramenta da Microchip que é o “Dynamic CDC Demo” que se encontra disponível na página da Microchip [BIBLIOTECA USB 2012].

7.2 Resultados e discussões

Na classe CDC o dispositivo é reconhecido automaticamente pelo Ubuntu, evitando a necessidade de instalação de *driver* ou qualquer outra configuração, na Figura 16, é mostrado o caminho onde o dispositivo pode ser localizado e o nome do dispositivo.



```
alex@alex-RV411: /dev/serial/by-id
alex@alex-RV411:/dev/serial/by-id$ ls
usb-Alex_Veloso_Placa_I_0_Digital_analogica-if00
alex@alex-RV411:/dev/serial/by-id$
```

Figura 16 - Dispositivo encontrado no Ubuntu 12.04

(Autoria própria)

Para a rotina de entrada digital o protótipo da função é “string le_porta_digital(char c)”, onde o parâmetro enviado vai ser o terminal a ser lido segundo a Tabela 3, e o retorno da função será uma string com o estado da porta (0 ou 1).

Para a rotina de saída digital o protótipo da função é “void aciona_porta_digital(char c)”, o argumento é um caractere que é correspondente ao terminal que se deseja comutar o nível lógico. Os oito terminais são iniciados em nível baixo (0V) quando a placa é alimentada

ou quando o botão do *reset* é pressionado. Os sinais foram padronizados em formato ASCII, estão na Tabela 4.

Para a rotina de leitura analógica o protótipo da função é “string le_porta_analogica(char c)”, onde o parâmetro passado será o canal analógico a ser lido segundo a Tabela 5, o retorno será uma string de 4 posições variando de 0 à 1023 correspondendo de 0 volts à tensão de referência.

Para a rotina de controle da saída analógica o protótipo da função é “string aciona_porta_analogica(char c)”, onde o parâmetro enviado deverá ser qual PWM deve ser acionado. Cada vez que o comando incrementar (P1 ou P2) é enviado ao microcontrolador um passo entre 0000 e 1023 é incrementado. Cada vez que o comando decrementar (p1 ou p2) é enviado ao microcontrolador um passo entre 1023 e 0000 é decrementado. O passo corresponde $VDD/1023$. Por exemplo se VDD for igual a 5,00 V um passo corresponderá a 4,88 mV. O comando retornar uma *string* que corresponde ao valor absoluto (posição entre 0000 e 1023) na saída do PWM acionado. A Tabela 6 mostra o código de cada PWM.

Por exemplo, para fazer a comutação de nível lógico de todos os terminais de saída digital é necessário acionar cada terminal individualmente, pois a cada sinal enviado pela rotina de “void aciona_porta_digital()” é comutado o nível lógico do terminal correspondente.

A biblioteca de controle (“controle_pic_pampa.py”) de comunicação, foi codificado para que a cada acesso ao PicPampa, abra uma conexão com a porta USB, espere 1 ms, e feche a porta USB. Foi necessário implementar essa temporização de 1 ms, pois em velocidades de comunicações maiores que 9600 bps, observou-se instabilidade na comunicação entre o PC e o PicPampa.

Todas as quatro funções foram agrupadas em uma única biblioteca (“controle_pic_pampa.py”). Que deve estar no mesmo diretório onde o programa Python é interpretado. É mostrado a implementação desta biblioteca no Apêndice E - Rotinas de Controle do Pic Pamapa.

8 APLICAÇÃO

Nesta seção será mostrado um exemplo de aplicação para a placa de desenvolvimento PicPampa utilizando a biblioteca de rotinas desenvolvida em Python. A aplicação foi definida como um aquecedor de água com temperatura controlada.

8.1 Aquecedor de água

Quando se utiliza um aquecedor elétrico, este aquece a água sem ter controle sobre a temperatura da água, antes de atingir o ponto de ebulição, durante a ebulição a temperatura se mantém constante. Esta aplicação controla o aquecimento da água até a temperatura desejada e estabiliza nesta temperatura, através de um simples processo de desligamento e acionamento automático de uma resistência de aquecimento através do relé instalado na placa PicPampa. O relé aciona um aquecedor conectado a rede elétrica (110 V ou 220 V, conforme especificação do aquecedor).

8.2 Material utilizado

Foram utilizados os seguintes materiais para a aplicação:

- Placa de desenvolvimento PicPampa.
- Computador para controle.
- Aquecedor de elétrico de 600 W.
- Relé de potência com bobina de 5V.
- Diodo 1N4007
- Resistor 4k7
- Transistor bipolar BC547

- Diodo ZENER 2,4V
- Capacitor cerâmico 100nF
- Capacitor eletrolítico de 1uF
- Resistor 1k
- Sensor de temperatura LM35
- 2 Capacitores SMD de 470 nF

8.3 Metodologia e implementação

Para facilitar a montagem do circuito este foi separado em três partes: regulador de tensão, sensor de temperatura e o aquecedor de água que foi montado diretamente na PicPampa.

O circuito regulador de tensão foi necessário, para melhorar a acurácia da tensão de referencia, pois a tensão que vem da USB 5V, não é confiável em termos de acurácia. A saída do regulador foi ligada na porta AN3 que é utilizada como tensão de referência externa. O diagrama esquemático é mostrado na Figura 17 e a respectiva foto na Figura 18, o circuito é alimentado pela saída da PicPampa VDD e GND, e a saída regulada de 2,4 V é ligada na porta AN3 que é utilizada como tensão referencia. Alternativamente, pode-se utilizar um regulador integrado do tipo LM431 ou equivalente.

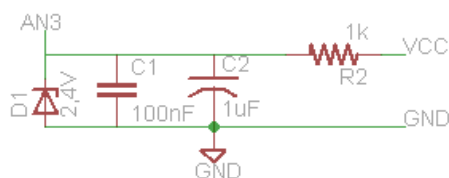


Figura 17 - Esquemático regulador de tensão.

(Autoria própria)

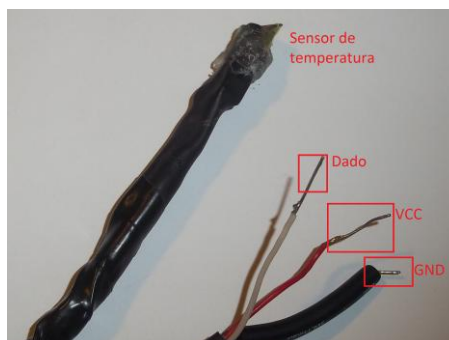


Figura 20 - Fotografia do sensor de temperatura LM35.
(Autoria própria)

O aquecedor elétrico é conectado a um relé que é montado na PicPampa de acordo com a Figura 21 e a foto correspondente na Figura 22. Foi soldado fios por cima da trilha do PicPampa entre a conexão da relé (contatos) com o conector para ligação do aquecedor. Pois como a trilha da placa é estreita esta poderia não ser suficiente para suportar a corrente correspondente aos 600 W do aquecedor elétrico e o fio acrescentado oferece um caminho de baixa resistência para esta corrente. Essa alteração é mostrada na Figura 23.

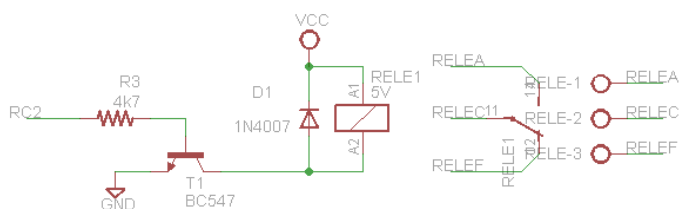


Figura 21 - Esquemático relé.
(Autoria própria)

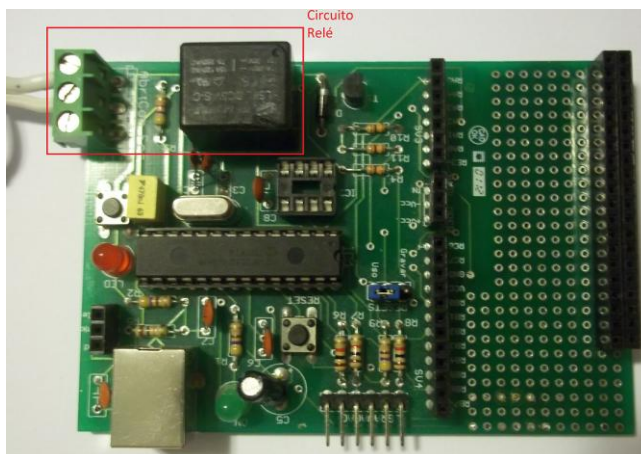


Figura 22 - Placa de desenvolvimento PicPampa, com destaque no relé.
(Autoria própria)

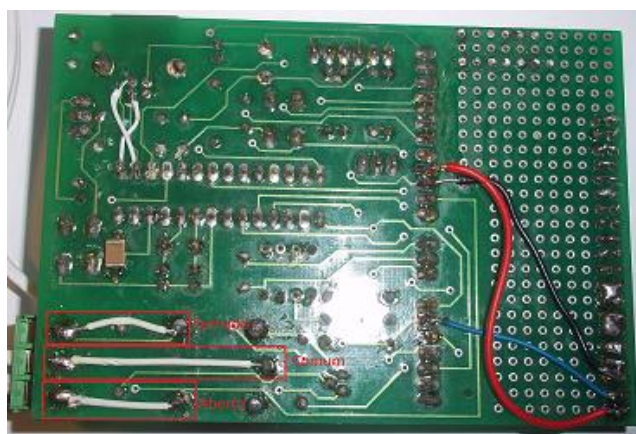


Figura 23 - Reforçando as conexões das trilhas do Relé.
(Autoria própria)

A seguir é mostrado a implementação codificada em Python:

1. `#!/usr/bin/env python #Torna este arquivo executavel.`
2. `from controle_pic_pampa import * #Importa a rotina do PicPampa.`
3. `import time # Biblioteca necessária para a funcao delay.`
4. `def convert_temperatura(temp_str): #Definição de função.`
5. `a = int(temp_str[0:4]) # Converte de string para inteiro.`

```
6.         a = ((a * 2400.0)/1023.0)/100.0 # Formula para converter em um valor de
tensão.
7.         return a # Retorna o valor já convertido.
8.     le_temp = leitura_analogica('N0') # Faz uma leitura analógica.
9.     le_temp = convert_temperatura(le_temp) # Faz uma conversão do valor lido para
temperatura em graus Célsius.
10.    print "Digite a temperatura máxima para o aquecimento:" #Mensagem exibida
no console.
11.    temp_max = raw_input() #Valor digitado no teclado e usado como referência
para a temperatura máxima
12.    aciona_porta_saida('R') #Ativa RELE
13.    while(1): #Lopp
14.        if (temp_max <= le_temp): #compara o valor lido com o valor de referência
15.            aciona_porta_saida('R') #Desativa RELE
16.            while(1):
17.                le_temp = leitura_analogica('N0') #faz uma leitura analógica.
18.                le_temp = convert_temperatura(le_temp) #Faz a conversão em termos
de temperatura.
19.                    print le_temp #imprime a temperatura.
20.                    if ((temp_max - 2) <= le_temp):
21.                        aciona_porta_saida('R') #Ativa RELE
22.                        break;
23.                    aciona_porta_saida('R') #Desativa RELE
24.                    time.sleep(0.5)# espera meio segundo para fazer uma leitura.
25.    le_temp = leitura_analogica('N0') #faz uma leitura analógica.
26.    le_temp = convert_temperatura(le_temp) #Faz a conversão em termos de
temperatura.
27.    print le_temp #imprime a temperatura.
```

8.4 Resultados e discussões

O programa inicia fazendo uma leitura analógica, em seguida converte para o correspondente em temperatura (graus Célsius), depois requisita para o usuário uma temperatura máxima de referência e então o relé é ativado, iniciando o aquecimento. Feito a etapa inicial o programa entra em um loop, faz uma comparação entre a temperatura lida e o valor de referência, caso o valor de referência de temperatura seja menor ou igual à temperatura lida pelo sensor, o relé é desligado. Um segundo *loop* monitora a temperatura e se ela for menor que 2 graus Célsius da temperatura máxima, o relé é ativado novamente e o programa retorna ao primeiro *loop*. Os testes mostraram que a aplicação funcionou como esperando. Foi realizado um teste de estabilidade do programa, no qual o sensor de temperatura ficou monitorando a temperatura ambiente durante 24 horas, sem nenhum problema registrado.

9 CONCLUSÕES

Foi montada com êxito uma placa de programação para o PIC18F utilizando componentes de fácil acesso encontrados no mercado nacional e foram produzidas diversas placas de circuito impresso do programador. A placa programadora funcionou com os microcontroladores das famílias PIC12F, PIC16F e PIC18F da Microchip.

Foi montado uma placa de desenvolvimento (PicPampa) utilizando o microcontrolador PIC18F2550 que oferece facilidades de interface entrada e saída analógica e digital. Em adição a placa oferece facilidades, opcionais, para montagem de alguns periféricos de *hardware* (chave de pressão, *led*, relé e amplificador operacional) e uma área para prototipagem de outros circuitos. Foram produzidas diversas placas de circuito impresso para facilitar o uso da interface.

Foi implementado o *firmware*, para a comunicação via USB, da placa PicPampa com computador. O *firmware* possibilita o acesso aos periféricos internos como ADC, PWM e terminais de entrada e saída digitais do microcontrolador.

Foi desenvolvido uma biblioteca para comunicação do PC com a placa PicPampa que possui funções de acesso aos comandos dos periféricos do microcontrolador para ser utilizada na linguagem Python. Esta biblioteca foi utilizada para escreve um programa de controle de aquecimento de água, através do monitoramento da temperatura. A biblioteca também foi utilizada para o desenvolvimento de uma interface gráfica para o monitoramento dos periféricos do microcontrolador.

Neste trabalho foi utilizado somente *software* de domínio público. A proposta inicial deste trabalho foi apresentada na sessão de pôster no SIEPE que ocorreu em Uruguaiana em 2011. O trabalho na sua fase final foi apresentado na seção oral do SIEPE de 2012 em Bagé com o título “Placa de aquisição de dados com PIC18F2550”. Durante o desenvolvimento deste trabalho foi realizado um *workshop* para montagem das placas programadora e a PicPampa, que contou com 17 participantes. Também durante o desenvolvimento do trabalho, o autor participou do MASTERS (seminário técnico sobre novas tecnologias da Microchip), realizado em novembro de 2012 em São Paulo.

10 TRABALHOS FUTUROS

Explorar outras classes para comunicação USB, como a classe HID (*Human Interface Device*).

Explorar outros periféricos internos como *Timers* e interrupção através de um pino externo.

Escrever rotinas semelhantes para outras linguagens de programação, como por exemplo o *LabView*.

Flexibilizar as rotinas já existentes, por exemplo, implementar na rotina de leitura analógica um parâmetro para a quantidade de leituras pré processadas ou não.

REFERÊNCIAS

ANALOG DEVICES INC. ENGINEERING, ANALOG-DIGITAL CONVERSION, Walt Kester (Editor), Newnes, 2004. Livre para download em: http://www.analog.com/library/analogDialogue/archives/39-06/data_conversion_handbook.html. Acessado em 03/05/2013.

ARDUINO, em:

<<http://www.arduino.cc>> Acessado em 11/11/2012.

AXELSON, Jan. USB Complete: Everything you need to develop custom USB peripherals. 3rd ed. Madison: Lakeview Research, 2005.

BALBINOT, Alexandre, e Brusamarello, Valner João, Instrumentação e Fundamentos de Medidas - Vol. 1, 2a.ed., Ed. LTC, 2010, ISBN: 9788521617549.

BIBLIOTECA USB para o PIC18 em:

<<http://www.microchip.com/pagehandler/en-us/technology/usb/tools/home.html>>

Acessado em 11/11/2012.

BRAHIM, Dogan. Microcontroller Based Applied Digital Control. 1st ed. New York: WILEY, 2006.

C18 Microchip em:

<http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en010014> Acessado em 11/11/2012.

COBERT, Jonathan; RUBINI, Alessandro; HARTMAN, Greg. Linux device drivers. 3rd ed. Sebastopol: O'Reilly, 2005.

DANIELS, David M. Using PWM Output as a Digital-to-Analog Converter on a TMS320C240 DSP - APPLICATION REPORT: SPRA490, 1998. Disponível em: <<http://www.ti.com/lit/an/spra490/spra490.pdf>> . Acessado em 03/05/2013.

Datasheet microcontrolador PIC18F2550 em:

<<http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en00280>>

Acessado em 04/07/2012.

FERRUA, Paulo. Estudo e desenvolvimento de um micro CLP de Baixo custo em um PIC. Vitória: 2007.

GUIMARÃES, Alexandre de Almeida. Eletrônica Embarcada Automotiva, Érica, 2010.

HENNESSY, John L., Patterson, David. [Organização e projeto de computadores: a interface hardware/software](#). 3.ed. Rio de Janeiro: Elsevier, c2005. 484 p.

HUANG, Han-Way. PIC Microcontroller: An introduction to Software & Hardware interfacing. 1st ed. New York: Thonson, 2005.

LUTZ, Mark. Programming Python. 2nd ed. Sebastopol: O'REILLY, 2001.

MATLOFF, Norman. A Quick, Painless Tutorial on the Python Language, University of California, Davis, 2007.

MIYADAIRA, Alberto. Microcontroladores PIC18: Aprenda e programe em linguagem C. 3. Ed. São Paulo: Editora Érica, 2012.

MPLAB em:

<http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469> Acessado em 11/11/2012

MICROCHIP ADC (Application Design Centers), em:

http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1445

Acessado em 03/05/2013.

NI National Instruments em:

<<http://www.ni.com/solutions/>>
Acessado em 10/07/2012).

OKAWA ELECTRIC, em: < <http://sim.okawa-denshi.jp/en/CRlowkeisan.htm>>. Acessado em 03/05/2013.

PAIOTTI, R. As dificuldades de trabalhar com microcontroladores no Brasil. Saber Eletrônica, n. 434, p. 1–5, 03 2009.

PARDUE, Joe. C Programming for Microcontrollers: Featuring ATMEL's AVR Butterfly and the Free WinAVR Compiler. 1st ed. Knoxville: Smiley Micros, 2005.

PONT, Michael. EMBEDDED C. 1st ed. Harlow: Pearson Education. 2005.

PREDKO, Myke. Programming and customizing the PIC Microcontroller. 3rd ed. New York: Mc Graw Hill, 2008.

SANTOS, Leonardo. Sistema de comunicação USB com microcontrolador. Recife: 2009.

SAVANT, C. ; RODEN, Martin; CARPENTER, Gordon. Electronic Design: Circuit and Systems. 2nd ed. Redwood City: The Benjamin Cummings, 1991.

SCHAMBER, Rodrigo. Desenvolvimento de um dispositivo de controle de acesso via protocolo de comunicação USB. Ouro Preto: 2008.

SILVA, Davidson. Sistema de comunicação Bluetooth utilizando Microcontrolador. Recife: 2009.

SOUZA, V. A. Comunicação USB com o PIC. Saber Eletrônica, v. 1, n. 421, p. 1–10, 1 2008.

TANENBAUM, Andrew. Organização Estruturada de Computadores. Editora Prentice Hall, 2007.

WALT, Kester. Analog Devices Inc. Engineering, analog-digital conversion, (Editor), Newnes, 2004. Livre para download em:
<http://www.analog.com/library/analogDialogue/archives/39-06/data_conversion_handbook.html>. Acessado em 03/05/2013.

WM Wikipedia Microcontrolador, em: < <http://es.wikipedia.org/wiki/Microcontrolador>>. Acessado em 03/05/2013.

ZEMBOVICI, Kleiton; FRANCO, Marcelo. Dispositivo para aquisição de sinais e Controle digital via USB. Curitiba; 2009.

APÊNDICE A - MANUAL DE USO DA INTERFACE GRÁFICA

Foi desenvolvida uma interface gráfica para teste e visualização de todos os módulos, para isso é necessário: ter o sistema operacional Ubuntu, ter o modulo PySerial, os arquivos “rotina_pic_pampa.py” e a “interface_pic_pampa.py” que devem estar no mesmo diretório. Para iniciar a interface basta executar o comando Python “interface_pic_pampa.py” dentro do terminal. Na figura A1 é possível observar todas as entradas e saídas digitais, de acordo com as Tabela 3 e Tabela 4.

```
alex@alex-RV411:~$ sudo python interface_pic_pampa.py
0000
0
PORTA RB0
PORTA RB1
PORTA RB2
0712
0713
PORTA RB4
PORTA RB5
PORTA RB6
PORTA RB7
1
1
1
1
1
0
0
0
0
0931
Este software foi desenvolvido por Alex Veloso da Silveira..
```

Figura A1 – Interface gráfica de controle da placa PicPampa
(Autoria própria)

A cada clique no botão, é feito uma comunicação com a PicPampa, e o retorno é impresso na tela. No caso das saídas digitais, é impresso apenas a porta correspondente, no caso de uma entrada é impresso seu estado atual, caso esteja em nível lógico ‘1’ é impresso ‘1’, caso em nível lógico ‘0’ é impresso ‘0’.

Os botões, “LED”, ”RELE” e “botao” são utilizados para o controle dos respectivos componentes (de acordo com o diagrama elétrico da placa PicPampa) que estão montados na placa.

Na figura A2, é possível observar as opções de uso para a leitura analógica, a tensão de referencia padrão vem dos 5V da USB, porém é possível configurar a referência positiva, para o canal 3, que seria correspondente a AN3 segundo a Tabela 5. E fazer a leitura dos canais 0(AN0), 1(AN1) e 2(AN2).

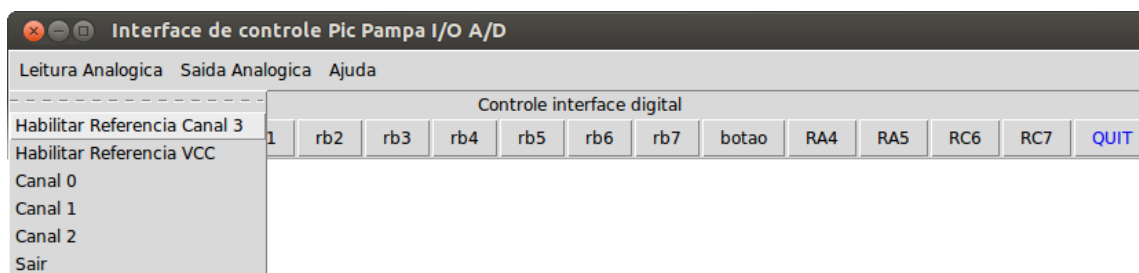


Figura A2 – Menu leitura analógica.

(Autoria própria)

Na figura A3 é possível observar o menu de controle que simula saídas analógicas, pode aumentar o diminuir o nível de tensão com o passo de 488 mV (este passo pode ser configurado no Python), e o valor é impresso na tela. PWM1 é correspondente a CPP1 e PWM2 correspondete a CPP2 segundo a Tabela 6.

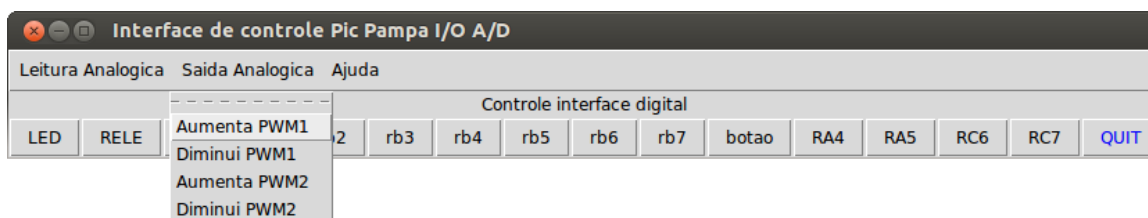


Figura A3 – Menu saída analógica.

(Autoria própria)

APÊNDICE B - ERRATA

Depois de confeccionado as placas de circuito impresso do PicPampa, foi detectado um erro de projeto, as conexões entre o soquete do PIC18F2550 e as linhas de dados D+ e D- da USB estavam trocadas. A correção se dá raspando as trilhas das conexão entre o pino 15 e a linha D+, e entre o pino 16 e a linha D-. Depois basta fazer a conexão usando fios externos entre o pino 15 e a linha D- e entre o pino 16 e a linha D+. Como se pode observar na figura B1. Esta correção já está atualizada no *Layout* atual.

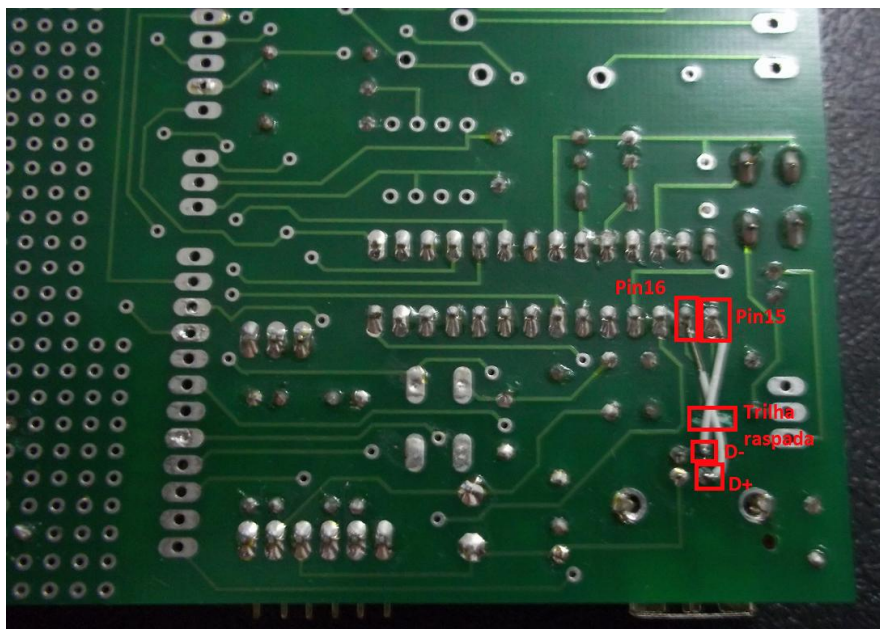
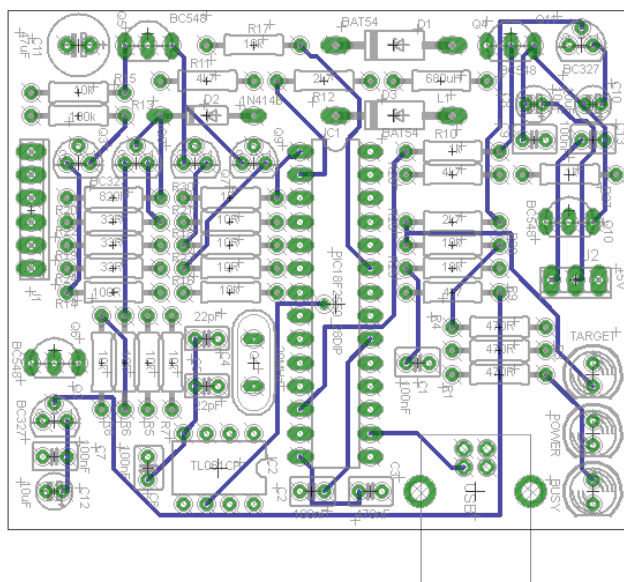


Figura B1 – Linhas D+ e D- trocadas.

(Autoria própria)

APÊNDICE C - LAYOUT DA PLACA GRAVADORA.



APÊNDICE E - ROTINAS DE CONTROLE DO PIC PAMAPA

```
import serial

def aciona_porta_saida(c):

    ser = serial.Serial('/dev/serial/by-id/usb-Alex_Veloso_Placa_I_O_Digital_analogica-
if00', 115200, timeout =0.0001)

    ser.open()

    ser.write(c)

    line = ser.readline()

    ser.close()

    return line

def leitura_analogica(c):

    ser = serial.Serial('/dev/serial/by-id/usb-Alex_Veloso_Placa_I_O_Digital_analogica-
if00', 115200, timeout =0.0001)

    ser.open()

    ser.write(c)

    line = ser.readline()

    ser.close()

    return line

def leitura_digital(c):

    ser = serial.Serial('/dev/serial/by-id/usb-Alex_Veloso_Placa_I_O_Digital_analogica-
if00', 115200, timeout =0.0001)

    ser.open()

    ser.write(c)

    line = ser.readline()

    ser.close()

    return line
```

```
def aciona_saida_analogica(c):  
    ser = serial.Serial('/dev/serial/by-id/usb-Alex_Veloso_Placa_I_O_Digital_analogica-  
if00', 115200, timeout =0.0001)  
  
    ser.open()  
  
    ser.write(c)  
  
    line = ser.readline()  
  
    ser.close()  
  
    return line
```

APÊNDICE F - INTERFACE GRÁFICA DE CONTROLE DO PIC PAMAPA

```
#!/usr/bin/env python

from rotina_porta_saida import *
from Tkinter import *

import time, locale

def pic_pampa_L():

class App:

    def __init__(self, master):

        frame = Frame(master)
        frame.pack()

        self.button = Button(frame, text="QUIT", fg="blue", command=frame.quit)
        self.button.pack(side=RIGHT)

        self.hi_there = Button(frame, text="LED", command=self.led)
        self.hi_there.pack(side=LEFT)

        self.hi_there = Button(frame, text="RELE", command=self.rele)
        self.hi_there.pack(side=LEFT)

        self.portb0 = Button(frame, text="rb0", command = self.rb0)
        self.portb0.pack(side=LEFT)
```

```
self.portb1 = Button(frame, text="rb1",command = self.rb1)
```

```
self.portb1.pack(side=LEFT)
```

```
self.portb2 = Button(frame, text="rb2",command = self.rb2)
```

```
self.portb2.pack(side=LEFT)
```

```
self.portb3 = Button(frame, text="rb3",command = self.rb3)
```

```
self.portb3.pack(side=LEFT)
```

```
self.portb4 = Button(frame, text="rb4",command = self.rb4)
```

```
self.portb4.pack(side=LEFT)
```

```
self.portb5 = Button(frame, text="rb5",command = self.rb5)
```

```
self.portb5.pack(side=LEFT)
```

```
self.portb6 = Button(frame, text="rb6",command = self.rb6)
```

```
self.portb6.pack(side=LEFT)
```

```
self.portb7 = Button(frame, text="rb7",command = self.rb7)
```

```
self.portb7.pack(side=LEFT)
```

```
self.botao = Button(frame, text="botao",command = self.botao)
```

```
self.botao.pack(side=LEFT)
```

```
self.botao = Button(frame, text="RA4",command = self.botao)
```

```
self.botao.pack(side=LEFT)
```

```
self.botao = Button(frame, text="RA5",command = self.botao)
```

```
self.botao.pack(side=LEFT)
```

```
self.botao = Button(frame, text="RC6",command = self.botao)
```

```
self.botao.pack(side=LEFT)
```

```
self.botao = Button(frame, text="RC7",command = self.botao)
```

```
self.botao.pack(side=LEFT)
```

```
def pwm(self):
```

```
    i = 100
```

```
    while(i>0):
```

```
        a = aciona_porta_saida('D')
```

```
        time.sleep(0.001)
```

```
        i=i-1
```

```
def led(self):
```

```
    a = aciona_porta_saida('L')
```

```
    print a
```

```
def rele(self):
```

```
    a =aciona_porta_saida('R')
```



```
print a
```

```
def rb0(self):
```

```
    c = 'A'
```

```
    a = aciona_porta_saida(c)
```

```
    print a
```

```
def rb1(self):
```

```
    c = 'B'
```

```
    a = aciona_porta_saida(c)
```

```
    print a
```

```
def rb2(self):
```

```
    c = 'C'
```

```
    a = aciona_porta_saida(c)
```

```
    print a
```

```
def rb3(self):
```

```
    c = 'D'
```

```
    a = aciona_porta_saida(c)
```

```
    print a
```

```
def rb4(self):
```

```
    c = 'E'
```

```
    a = aciona_porta_saida(c)
```

```
    print a
```

```
def rb5(self):
```

```
    c = 'F'
```

```
    a = aciona_porta_saida(c)
```

```
    print a
```

```
def rb6(self):
```

```
    c = 'G'
```

```
    a= aciona_porta_saida(c)
```

```
    print a
```

```
def rb7(self):
```

```
    c = 'H'
```

```
    a = aciona_porta_saida(c)
```

```
    print a
```

```
def botao(self):
```

```
    c = 'I'
```

```
    a = aciona_porta_saida(c)
```

```
    print a
```

```
def pwm1D():
```

```
    i = 100
```

```
    while(i>0):
```

```
        a = aciona_porta_saida('D')
```

```
        #time.sleep(0.00001)
```

```
        a = ((float(a[0:3])/1023.0)*5)
```

```
        print locale.format("% 1.3f",a,0)
```

```
        i=i-1
```

```
def pwm1d():
```

```
    i = 100
```

```
    while(i>0):
```

```
        a = aciona_porta_saida('d')
```

```
time.sleep(0.00001)

a = ((float(a[0:3])/1023.0)*5)

print locale.format("% 1.3f",a,0)
```

```
i=i-1
```

```
def callback():
```

```
    print "Este software foi desenvolvido por Alex Veloso da Silveira.."
```

```
def canal0():
```

```
    c = 'N'
```

```
    a = aciona_porta_saida(c)
```

```
    print a
```

```
    return a
```

```
root = Tk()
```

```
widget = Label(root)
```

```
widget.config(text='Controle interface digital')
```

```
widget.pack(side=TOP, expand = YES, fill=BOTH)
```

```
root.title('Interface de controle PicPampa I/O A/D')
```

```
app = App(root)
```

```
# create a menu
```

```
menu = Menu(root)
```

```
root.config(menu=menu)
```

```
filemenu = Menu(menu)
menu.add_cascade(label="Leitura Analogica", menu=filemenu)
filemenu.add_command(label="Habilitar Referencia Canal 3", command=callback)
filemenu.add_command(label="Habilitar Referencia VCC", command=callback)

filemenu.add_command(label="Canal 0", command=canal0 )
filemenu.add_command(label="Canal 1", command=canal1 )
filemenu.add_command(label="Canal 2", command=canal2 )

saida = Menu(menu)
menu.add_cascade(label="Saida Analogica", menu=saida)
saida.add_command(label="Aumenta PWM1", command=pwm1D )
saida.add_command(label="Diminui PWM1", command=pwm1d )

saida.add_command(label="Aumenta PWM2", command=pwm2D )
saida.add_command(label="Diminui PWM2", command=pmw1d )
filemenu.add_command(label="Sair", command= callback)

    helpmenu = Menu(menu)
menu.add_cascade(label="Ajuda", menu=helpmenu)
helpmenu.add_command(label="Sobre...", command=callback)
a = "
print a
root.mainloop()
```

APÊNDICE G – SELEÇÃO DA TENSÃO DE REFERÊNCIA DO ADC

```

if (c = 'v'){
    OpenADC(ADC_FOSC_64          //SELEÇÃO DA FONTE DE CLOCK PARA A CONVERSAO A/D. TAD = 64/48M = 1.33Us
    &ADC_RIGHT_JUST //RESULTADO JUSTIFICADO PARA A DIREITA
    &ADC_2_TAD,          //CONFIGURAÇÃO DO TEMPO DE AQUISIÇÃO AUTOMATICO (2*TAD = 2,667us)
    ADC_CH0             //SELECIONA O CANAL 0 (AN0)
    &ADC_INT_OFF        //DESABILITA INTERRUPÇÃO
    &ADC_VREFPLUS_EXT   //VREF+ = VREF+ (PIN AN3)
    &ADC_VREFMINUS_VSS, //VREF-= VSS
    ADC_6ANA);         //Habilita o canal an0 a an5
}
if (c = 'V'){
    OpenADC(ADC_FOSC_64          //SELEÇÃO DA FONTE DE CLOCK PARA A CONVERSAO A/D. TAD = 64/48M = 1.33Us
    &ADC_RIGHT_JUST //RESULTADO JUSTIFICADO PARA A DIREITA
    &ADC_2_TAD,          //CONFIGURAÇÃO DO TEMPO DE AQUISIÇÃO AUTOMATICO (2*TAD = 2,667us)
    ADC_CH0             //SELECIONA O CANAL 0 (AN0)
    &ADC_INT_OFF        //DESABILITA INTERRUPÇÃO
    &ADC_VREFPLUS_VDD   //VREF+ = VDD
    &ADC_VREFMINUS_VSS, //VREF-= VSS
    ADC_6ANA);         //Habilita o canal an0 a an5
}

```

No bloco ‘v’ é selecionado a referência externa, linha (&ADC_VREFPLUS_EXT), e no bloco ‘V’ é selecionado a referência interna (&ADC_VREFPLUS_VDD), que utiliza a fonte de alimentação, $VDD = V_{USB}$.

ANEXO A – CARACTERÍSTICA DO PROTOBOARD

PROTOBOARD MINIPA MP-2420

PROTOBOARD SEM SOLDA

DESCRIÇÃO

O Protoboard é uma ferramenta prática e barata para realizar experiências eletrônicas. Usando o Protoboard você descobrirá que projetar e testar experiências eletrônicas pode ser muito fácil, rápido e mais gostoso tanto para profissionais como para hobistas.

CARACTERÍSTICAS

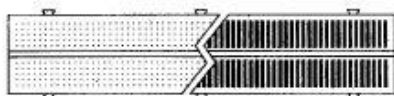
- A conexão dos terminais dos componentes é feita diretamente nos soquetes do protoboard e desconectados facilmente. Devido os componentes não terem sido soldados, eles podem ser usados novamente. Isto economiza tempo e dinheiro.
- Os conjuntos de furos tem a característica importante de permitirem a interconexão de algumas placas, sendo assim um sistema muito versátil.
- Os componentes de montagem podem possuir tamanhos variados e interconectados por fios rígidos (0.3mm – 0.8mm).

CONSTRUÇÃO

Número de Furos: 2420 furos

Faixa Tipo Soquete

As duas colunas estão separadas por um canal central de 7,62mm. E possui 64 grupos de 5 furos de cada lado.

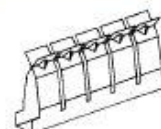


Faixa Tipo Alimentação: São 20 grupos de 25 furos distribuídos em 4 faixas.



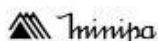
MATERIAL

- Corpo composto de Polímero ABS resistente a 90°C.
- Contatos internos compostos por uma liga de Prata e Níquel.



GERAIS

- Resistência (em 1kHz): < 6mW
- Capacitância (em 1kHz):
Faixa Tipo Soquete: < 2pF
Faixa Tipo Alimentação: < 10pF
- Corrente: 3A (máx)
- Isolação: > 10TΩ (em 500V DC)
- Dimensões: 237(A) x 175(L) x 18.5(P)mm
- Peso: 458g



Minipa Indústria e Comércio Ltda

Al. dos Tupinás, 33 - Planalto Paulista - São Paulo - CEP: 04069-000 - SP - Tel: (11) 5078-1850 - Fax: (11) 5072-2266 - e mail: minipa@minipa.com.br

Sujeito à alterações sem aviso prévio.

<http://www.minipa.com.br/download/manuais/mp-2420/mp-2420-1100.pdf>