# GRUPO 4

**DOCUMENTAÇÃO**

# GRUPO 4

**DOCUMENTAÇÃO**

Manual da placa desenvolvida durante o período submetido ao prof. Rodrigo De Paula Rodrigues, como requisito parcial para aprovação na disciplina PBLE01 – Co-Design de Produtos Eletrônicos do curso de graduação em Engenharia Eletrônica da Universidade Federal de Itajubá.

PBLE01

*Co-projeto de produtos eletrônicos*

**Manual**

Grupo 4

Rev 1 - Ago/2021

Sumário

1. [Identificação](#_heading=h.gjdgxs) 2
2. [Introdução](#_heading=h.30j0zll) 3
3. [Esquema elétrico](#_heading=h.1fob9te) 4
   1. R[equisitos](#_heading=h.3znysh7) 4
   2. [Representação Do Circuito Em Termos De Blocos Funcionais](#_heading=h.2et92p0) 5
      1. [Circuito De Alimentação 6](#_heading=h.tyjcwt)
      2. [Conector ICSP](#_heading=h.3dy6vkm) 8

[3.2.3 MCP7940N](#_heading=h.1t3h5sf) 9

* + 1. [Amplificadores Operacionais (Amp Op)](#_heading=h.4d34og8) 10
    2. [Comunicação USB](#_heading=h.2s8eyo1) 13
    3. [Teclado de 5 switches](#_heading=h.17dp8vu) 15
    4. [Display LCD](#_heading=h.3rdcrjn) 17
    5. [Outros periféricos](#_heading=h.26in1rg) 18
  1. [Relatório de verificação de erros de projeto](#_heading=h.lnxbz9) 20

1. [Placa de circuito impresso](#_heading=h.35nkun2) 20
   1. [Requisitos](#_heading=h.1ksv4uv) 21
   2. [Desenho da placa de circuito impresso](#_heading=h.44sinio) 22
   3. [Visão tridimensional do circuito projetado](#_heading=h.2jxsxqh) 24
   4. [Relatório de verificação de erros de projeto](#_heading=h.z337ya) 27
2. [Especificações elétricas](#_heading=h.3j2qqm3) 27
   1. [Níveis de alimentação suportados](#_heading=h.1y810tw) 27
   2. [Faixa de níveis de consumo estimadas 28](#_heading=h.4i7ojhp)
   3. [Correntes e tensões máximas de entrada 28](#_heading=h.2xcytpi)
   4. [Correntes e tensões máximas de saída 28](#_heading=h.1ci93xb)
   5. [Características de comunicação 28](#_heading=h.3whwml4)
3. [Programa embarcado de validação 29](#_heading=h.2bn6wsx)
   1. [Requisitos 29](#_heading=h.qsh70q)
   2. [Modelo de operação geral](#_heading=h.3as4poj) 30
   3. [Acesso às funcionalidades](#_heading=h.1pxezwc) 30
      1. [LEDs 31](#_heading=h.49x2ik5)
      2. [Teclado](#_heading=h.2p2csry) 32
      3. [LCD](#_heading=h.147n2zr) 33
      4. [PWM](#_heading=h.3o7alnk) 34
4. [Lista de materiais](#_heading=h.23ckvvd) 35

[9 Anexos (Códigos)](#_heading=h.ihv636) 37

# IDENTIFICAÇÃO

Integrantes do GRUPO 4:

| Nome | Matrícula |
| --- | --- |
| Igor José Marinho Vieira | 2019015342 |
| José Augusto de Souza Melo | 2019005417 |
| Thales Augusto Rodrigues de Campos | 2019005533 |

# INTRODUÇÃO

Este documento apresenta os detalhes do projeto da placa de circuito impresso, desenvolvida durante o período letivo de 2021-1, além de abordar tópicos como o desenvolvimento do software embarcado de validação.

No geral, este documento apresenta, detalhadamente, o funcionamento do esquema elétrico, especificações elétricas da placa de circuito impresso, lista dos componentes eletrônicos usados no projeto, estrutura do programa embarcado de validação, dentre outros detalhes.

Para o desenvolvimento do esquema elétrico foi utilizado o software [*KiCad*](https://www.kicad.org/) *EDA (A Cross Platform and Open Source Electronics Design Automation Suite)* em sua versão 5.1.x para OS Windows. O software [MPLAB® X IDE](https://www.microchip.com/en-us/development-tools-tools-and-software/mplab-x-ide) versão 5.50, foi usado para o desenvolvimento do programa embarcado de validação (firmware).

# ESQUEMA ELÉTRICO

Esta é a primeira seção pratica deste documento, nela será abordado assuntos referentes ao esquema elétrico do projeto. Iniciaremos com uma ideia geral dos requisitos necessários que serão implementados ao projeto. O diagrama elétrico do circuito foi dividido em folhas (*sheets*), tais como: *Alimentação*, *ICSPConnector, MCP7940N, AmpOps, USBSerial, Teclado, LCD* e *LEDs.* Cada um desses subcircuitos serão explicados mais adiante neste documento.

Esta seção abordará todos os cálculos realizados para implementar as entradas diferenciais (discutida na seção 3.3.4, referente ao *sheet* AmpOp), além de apresentar um relatório de erros gerado pela ferramenta KiCad, que aponta possíveis problemas no circuito, caso haja.

# REQUISITOS

Os requisitos do circuito microprocessado estão especificados na tabela abaixo. Todo o projeto do esquema elétrico será baseado nos requisitos apresentados.

| Classe | Requisitos |
| --- | --- |
| Alimentação | Suportar tensão de entrada na faixa de 7 a 12V em corrente contínua; |
| Empregar um conector de alimentação do tipo jack J4; |
| Empregar proteção contra tensão reversa; |
| Empregar um regulador linear com saída de 5V; |
| Empregar um diodo emissor de luz para sinalizar a presença de alimentação na placa. |
| Operação | Empregar microcontrolador PIC18F4550 como unidade de processamento e controle; |
| Possuir uma barra de pinos compatível com a padrão de gravação ICSP para o microcontrolador empregado; |
| Possuir uma chave táctil para reiniciar a operação do microcontrolador. |
| Periféricos e Expansão | Possuir uma barra de expansão que contemple os sinais de referência e de alimentação da placa e quatro entradas analógicas a formarem dois pares de entradas analógicas diferenciais. Tais pares devem ser condicionados de forma a se gerar níveis entre 0V e 3V a serem disponibilizados ao microcontrolador empregado; |

|  | Possuir uma saída analógica formada por sinal em modulação PWM; |
| --- | --- |
| Possuir uma barra de expansão de sinais que contemple os sinais de referência (terra) e de alimentação do circuito assim como os demais pinos não utilizados do microcontrolador; |
| Empregar um relógio de tempo real da família MCP7940. |
| Requisitos adicionais | Empregar um conversor USB-serial da família MCP2200 para |
| se permitir a comunicação serial com a placa desenvolvida. |

Tabela 1 – Tabela de requisitos para o esquema elétrico

Veremos, posteriormente, que alguns requisitos foram adicionados, tais como o uso de um filtro *Butterworth* ligado à saída PWM. Na prática tais requisitos e componentes empregados são definidos pelo projetista em função das necessidades que o projeto demanda.

* 1. **REPRESENTAÇÃO DO CIRCUITO EM TERMOS DE BLOCOS FUNCIONAIS** Podemos deixar um esquema elétrico mais legível e organizado quando criamos blocos funcionais (*sheets*). A imagem apresentada na figura 1 mostra o circuito dividido em vários blocos, cada bloco possui, internamente, um circuito. Esse circuito possui a função descrita no próprio nome do bloco, para fins de organização e legibilidade do esquema elétrico.

Com o intuito de evitar fios entrelaçados, são usados rótulos hierárquicos e rótulos globais. Os rótulos hierárquicos têm por finalidade exibir pinos de conexão, internos ao *sheet,* fora dos blocos funcionais, caso esses pinos precisem de conexão aos outros blocos. Os rótulos globais funcionam como fios, interligando os pinos entre si.

A figura 1 mostra o circuito implementado, seguindo todos requisitos vistos na seção 3.1. Note que foram usados blocos funcionais junto aos rótulos para uma melhor organização.

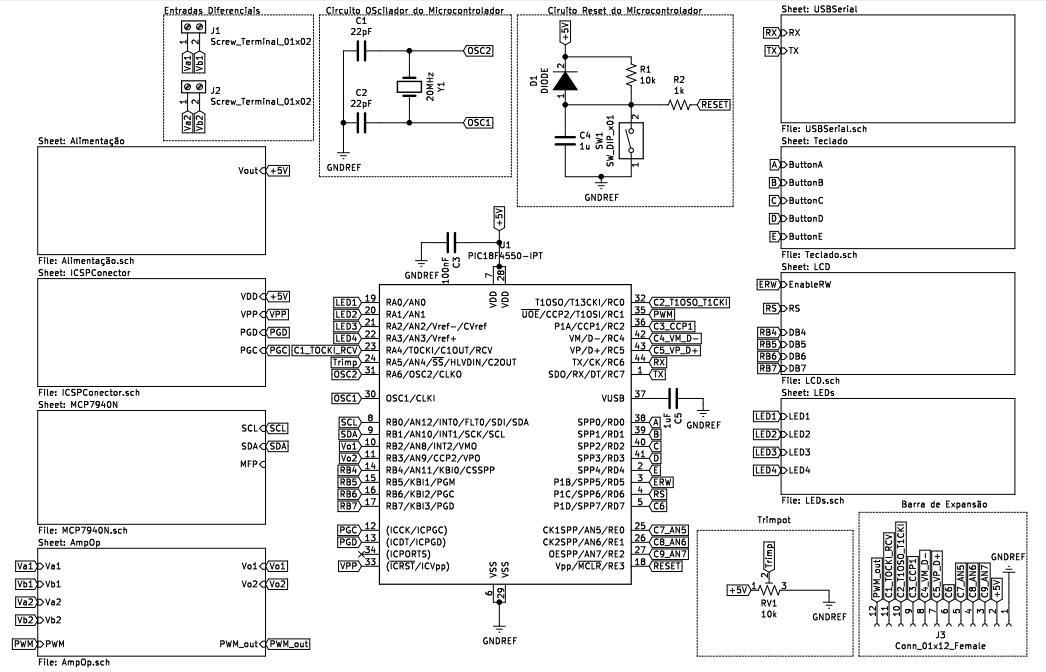


Figura 1 – Visão geral do circuito

Note que nem todos os circuitos estão em um *sheet*, essa decisão foi tomada pois trata-se de circuitos que desempenham funções diretas no microcontrolador, tais como: o circuito do cristal oscilador e o circuito de reset. Outro motivo é a baixa complexidade desses circuitos, sendo possível organizar, visualmente, todo o esquemático de modo que continue legível.

As próximas subseções abordarão, individualmente, a função de cada um dos circuitos presentes em cada *sheet.*

# CIRCUITO DE ALIMENTAÇÃO

O sub-circuito de alimentação, como o nome já diz, é responsável por distribuir alimentação para todo o sistema. Na figura 2 pode ser visto o *sheet* do sub-circuito de alimentação. Este *sheet* possui um pino de conexão, VOut (rótulo hierárquivo),

onde está conectado toda a rede +5V (Volts). Para fins de abstração do esquemático, todo o GND está curto-circuitado pelo componente GNDREF presente na ferramenta KiCAD.



Figura 2 – Bloco funcional de alimentação

O circuito implementado internamente ao *sheet,* apresentado na figura 2, pode ser visto na figura 3.

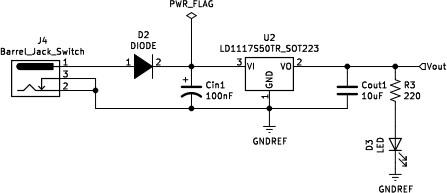


Figura 3 – Circuito de Alimentação

O componente identificado como J4 (figura 4), é um conector que permite a entrada de tensão para o sistema.



Figura 4 – Conector J4

Como não é interessante, em sistemas digitais, uma tensão reversa, a implementação do diodo D2 faz com que a corrente não circule pelo sistema caso haja uma inversão na polaridade da alimentação proveniente do conector J4. O componente eletrônico U2 é um regulador de tensão, LM11750, ele transforma toda tensão entre 5V à 20V em uma saída regulada de 5V, diminuindo as possibilidades de que um ruído chegue aos componentes digitais. Os capacitores presentes na entrada e na saída do regulador de tensão funcionam como filtros, esses capacitores são requeridos pelo LM11750. Para maiores detalhes, acesse o [dataheet](https://www.ti.com/lit/ds/symlink/lm1117.pdf?ts=1628182873040&ref_url=https%253A%252F%252Fwww.google.cl%252F) do LM117.

# CONECTOR ICSP

O conector ICSP (*In-Circuit Serial Programming*) permite a gravação dos softwares embarcados no microcontrolador com o auxílio do [PICkit3](https://ww1.microchip.com/downloads/en/DeviceDoc/51795B.pdf) [Programmer/Debugger](https://ww1.microchip.com/downloads/en/DeviceDoc/51795B.pdf), trata-se de um circuito *programador/debuger* usado para realizar uma comunicação serial de forma a gravar o firmware no PIC18F4550. ICSP é o acrônimo da comunicação entre esses dois dispositivos *In-Circuit Serial Programming™ (ICSP™).* Seu *sheet* é mostrado na Figura 5 e o conector é mostrado na Figura 6. Note que o conector nada mais é do que uma barra de pinos compatível com a conexão do PICkit3 e seu bloco funcional tem como saída os pinos que vão conectados ao microcontrolador.



Figura 5 - Bloco funcional Conector ICSP

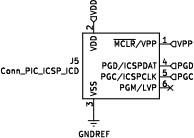


Figura 6 - Conector ICSP

# MCP7940N

Trata-se de um relógio de tempo real (*Real-Time Clock/Calendar -RTCC*). O [**MCP7940N**](http://ww1.microchip.com/downloads/en/devicedoc/20005010f.pdf)usa a interface I2C para se comunicar com o microcontrolador de forma bidirecional. O pino SCL (*Serial Clock*) é usado para sincronizar a transferência de dados enquanto SDA (*Serial Data*) é um pino bidirecional usado para transferir endereços e dados. A saída MFP (*Multifunction Pin*) é um pino de multifunção do RTCC seu uso foi descartado neste projeto. A construção do circuito da Figura 8 foi baseada no circuito modelo que pode ser encontrado no [datasheet](http://ww1.microchip.com/downloads/en/devicedoc/20005010f.pdf). No projeto, não está sendo usado a função de *Backup Supply*, um suprimento de energia para que o módulo opere com o circuito desenergizado, assim o pino VBAT foi conectado ao GND.

Figura 7 - Bloco funcional MCP7940

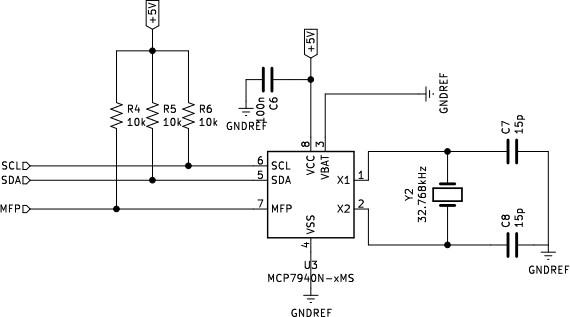


Figura 8 - Ligação do RTCC

# AMPLIFICADORES OPERACIONAIS (AMP OP)

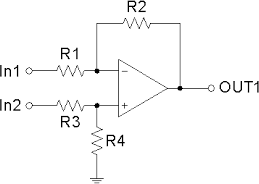
Uma das especificações do projeto seria o uso de amplificadores operacionais para implementar as entradas diferenciais. Tais entradas se referem a 4 entradas de sinais analógicos que serão previamente tratados por amplificadores operacionais antes de serem enviados ao microcontrolador para o devido processamento dos dados. O CI (circuito-integrado) de Amp Op utilizado foi o [LM324DG](https://datasheet.octopart.com/LM324DG-ON-Semiconductor-datasheet-506186.pdf) na configuração subtrator. Segue a baixo os cálculos realizados para projetar o ganho dos AMP OPs:

Figura 9 – Esquemático de um amplificador operacional

Como o amp op está em sua configuração de realimentação negativa, o conceito de curto-circuito virtual pode ser aplicado. As referências aos resistores usados nos cálculos são com base na figura 9.

𝑒+ = 𝑒−

𝑒+ = 𝐼𝑛2

𝐼𝑛1 − 𝑒−



𝑅1

𝑅4

×

𝑅4 + 𝑅3

(𝑒−) − 𝑂𝑢𝑡1

=

𝑅2

𝑂𝑢𝑡

𝑅2

= − (𝐼

− 𝑒−) + 𝑒−

1 𝑅1

𝑛1

𝑂𝑢𝑡

= − 𝑅2 (𝐼

− 𝐼

𝑅4

× ) + 𝐼

𝑅4

×

1 𝑅1

𝑅2

𝑛1

𝑛2

𝑅4

𝑅4 + 𝑅3

𝑅2

𝑛2

𝑅4 + 𝑅3

𝑅4

𝑂𝑢𝑡1 = − 𝑅

𝑅2

𝐼𝑛1 + 𝐼𝑛2 𝑅

+ 𝑅3

𝑅4

×

𝑅1

𝑅2

+ 𝐼𝑛2 ×

𝑅4

+ 𝑅3

𝑂𝑢𝑡1 = − 𝑅

𝐼𝑛1 + 𝐼𝑛2 𝑅

+ 𝑅3

(

𝑅1

+ 1)

𝑅2

𝑅4 𝑅2 + 𝑅1

𝑂𝑢𝑡1 = − 𝑅

𝐼𝑛1 + 𝐼𝑛2 𝑅

(

+ 𝑅3

)

𝑅1

Em nosso projeto R2 = R4 e R1 = R3, assim:

𝑅2

𝑅2 𝑅2 + 𝑅1

𝑂𝑢𝑡1 = − 𝑅

𝑅2

𝑂𝑢𝑡1 = − 𝑅

𝐼𝑛1 + 𝐼𝑛2 𝑅

𝑅2

𝐼𝑛1 + 𝐼𝑛2 𝑅

(

+ 𝑅1

)

𝑅1

𝑂𝑢𝑡1 =

𝑅2



𝑅1

(𝐼𝑛2 − 𝐼𝑛1)

𝑂𝑢𝑡1 =

12𝐾𝛺

20𝐾𝛺 (𝐼𝑛2 − 𝐼𝑛1)

𝑂𝑢𝑡1 = 0,6(𝐼𝑛2 − 𝐼𝑛1)

A fórmula acima, expressa a tensão de saída em função das tensões de entradas. A figura 10 ilustra o *sheet* dos dos amp ops.

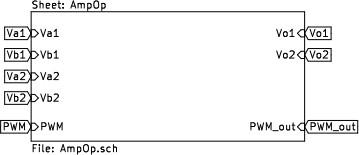


Figura 10 – Sheet do circuito dos amplificadores

Va1, Vb1, Va2 e Vb2 são redes conectadas aos terminais de dois conectores (modelo OSTTA024163). Estes conectores estão identificados como “Entradas Diferenciais”. A figura 11 representa como são esses conectores.



Figura 11 – Modelo dos conectores das entradas diferenciais

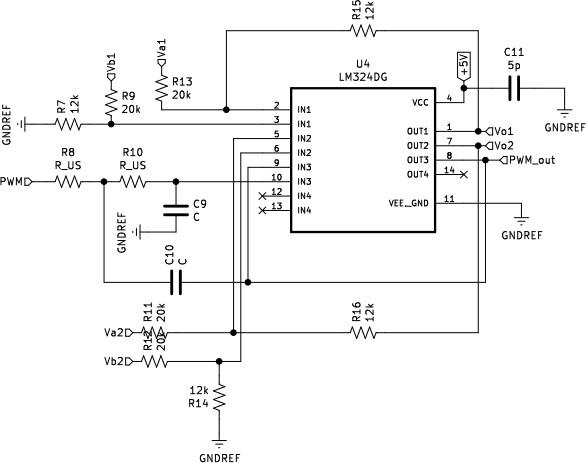


Figura 12 – Circuito das entradas diferenciais

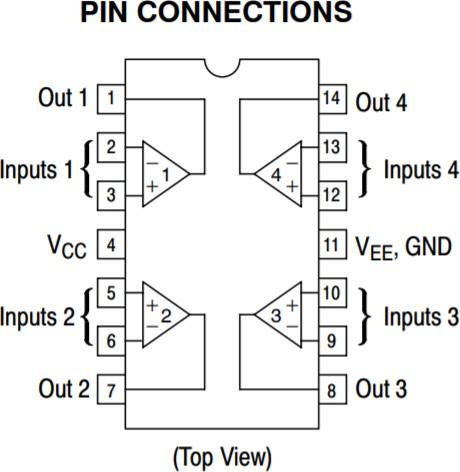
O CI LM324DG possui quatro amplificadores, observe a figura 13 retirada do *datasheet,* para as entradas diferenciais foram usados *inputs-1* e *inputs-2.* Os resistores de 20k e 12k não temos aí fizemos em série 10 10k e 10k e 2k. No filtro butterworth colocamos os valores de 10k nos resistores e 22pF nos capacitores. Note que há a utilização do *inputs-3* como uma entrada para um sinal PWM. De acordo com os requisitos de projeto, visto na seção 3.1, há a necessidade de implementar uma saída PWM (*Pulse Width Modulation*), portanto o rótulo hierárquico *PWM* está conectado diretamente ao pino RC1 do PIC. Para que haja uma “limpeza” no sinal, foi implementado um filtro butterworth de segunda ordem, ilustrado na figura 14.

Figura 13 – Disponibilidade dos pinos no LM324DG

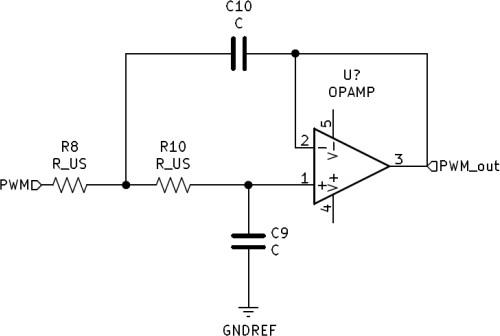


Figura 14 – Filtro Butterworth de segunda ordem

# COMUNICAÇÃO USB

Para a conexão USB, foi usado um transceptor USB com o intuito de diminuir a complexidade do software embarcado. Trata-se do componente [MCP2200](https://ww1.microchip.com/downloads/en/DeviceDoc/22228A.pdf), um conversor serial de USB para o protocolo UART (*Universal Asynchronous Receiver/Transmitter*), que possui uma taxa de 345,6 kB/s, a comunicação RS- 232 e RS-422 são baseados em UART, por exemplo. O *sheet* de MCP2200 é mostrado a seguir na figura 15.

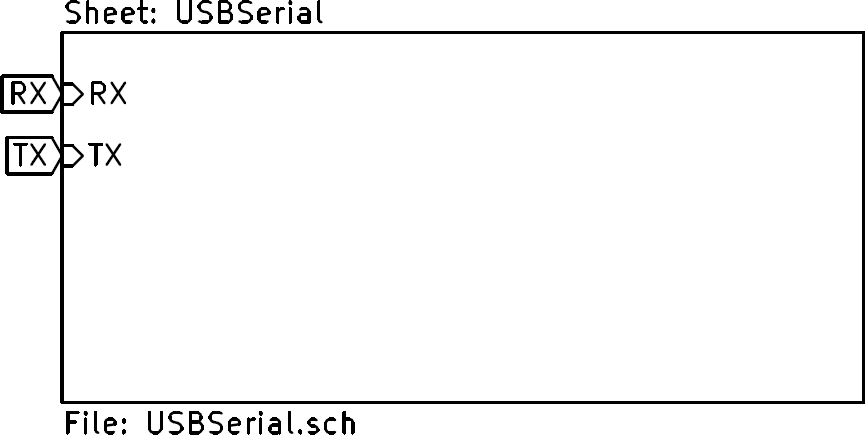


Figura 15 – Sheet do circuito do transceptor USB

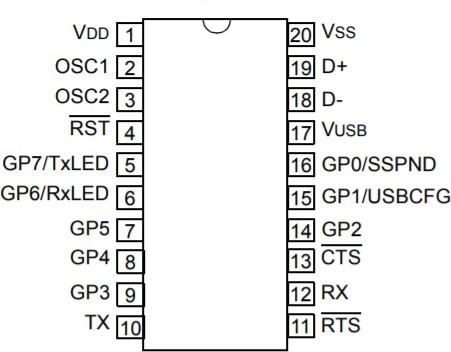


Figura 16 – Pinos do MCP2200

O CI possui oito entradas/saída de uso geral chamadas GPIO (Pinos de GP0 à GP7). Os pinos GP0, GP1, GP6 e GP7 possuem funções alternativas para indicar os status do USB e da comunicação. No projeto, foram usados GP6 (recepção de dados pelo USB) e GP7 (transmissão de dados pelo USB).

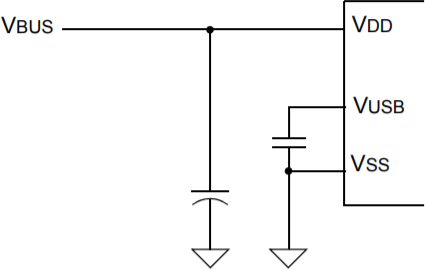
Para o projeto, foi adotado o modo *Bus Power Only,* onde toda a energia é extraída da conexão USB. A figura 17, extraída do *datasheet* ilustra a conexão para esse tipo de operação, note que essa mesma configuração foi usada no projeto (figura 18).

Figura 17 – Modo Bus Power Only

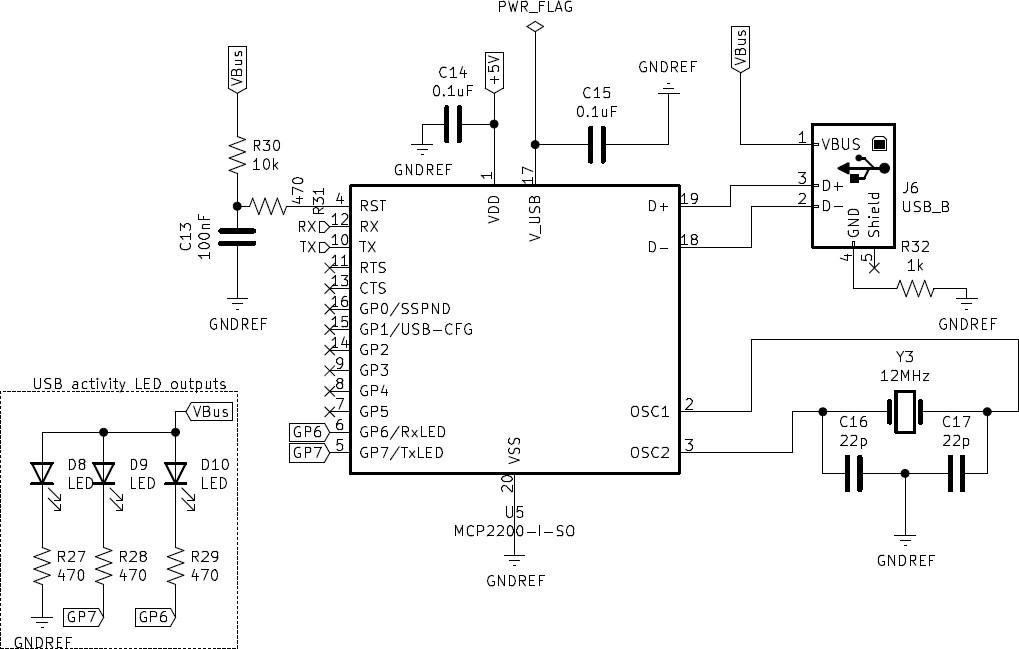


Figura 18 – Circuito interno ao *sheet* USBSerial

J6 representa um conector USB do tipo B (2.0) (figura 19). Note o uso das funções RxLED e TxLED como dito anteriormente. Os pinos RX e TX estão conectados aos pinos *RC6* e *RC7* do microcontrolador, respectivamente. O oscilador de 12 MHz foi escolhido com base no *datasheet*. A velocidade total do USB é, nominalmente, de 12 Mb/s.



Figura 19 – Conector USB B

* + 1. **TECLADO DE 5 *SWITCHES***

Como parte dos requisitos de projeto, mais especificadamente compondo os periféricos, foi implementado um teclado de cinco teclas sem funções previamente definidas, a função de cada tecla fica a critério do firmware. As figuras 20 e 21 referem-se ao *sheet* e ao circuito do teclado, respectivamente.

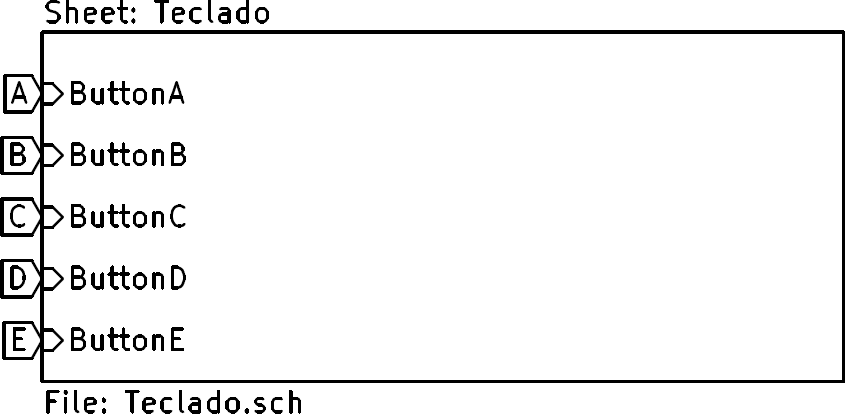


Figura 20 – *Sheet* do teclado de cinco teclas

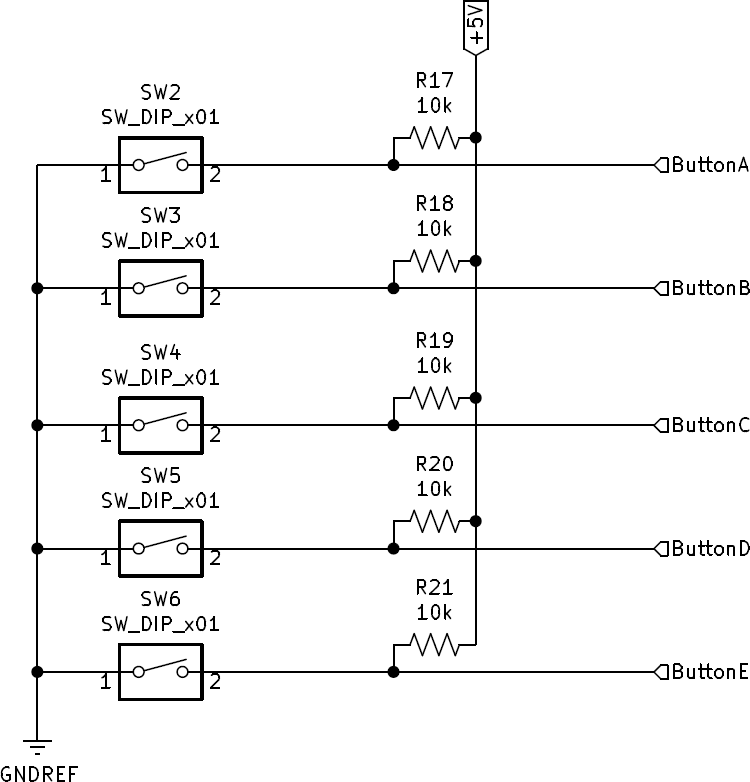


Figura 21 – Circuito referente ao teclado de cinco teclas

Note que há resistores de pull-up presente nos barramentos de cada tecla, portanto esse teclado será ativo em nível lógico baixo, ou seja, 0 V (Volts). As teclas são *switches* tais como ilustrado na figura 22.

Figura 22 – Switch usado no teclado

# DISPLAY LCD

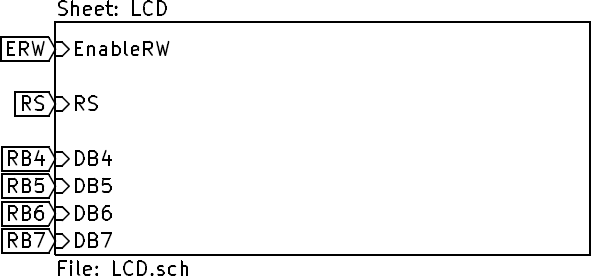
O LCD (*liquid crystal display*) usado no projeto é o HY1602E, possui uma operação igual ao [HD47780](https://www.sparkfun.com/datasheets/LCD/HD44780.pdf) da HITACHI. Esse tipo de LCD funciona em dois modos distintos de operação, são eles o modo 4 bits e o modo 8 bits, detalhados mais adiante. Segue a imagem do *sheet* do circuito.

Figura 23 – Sheet do circuito do display LCD

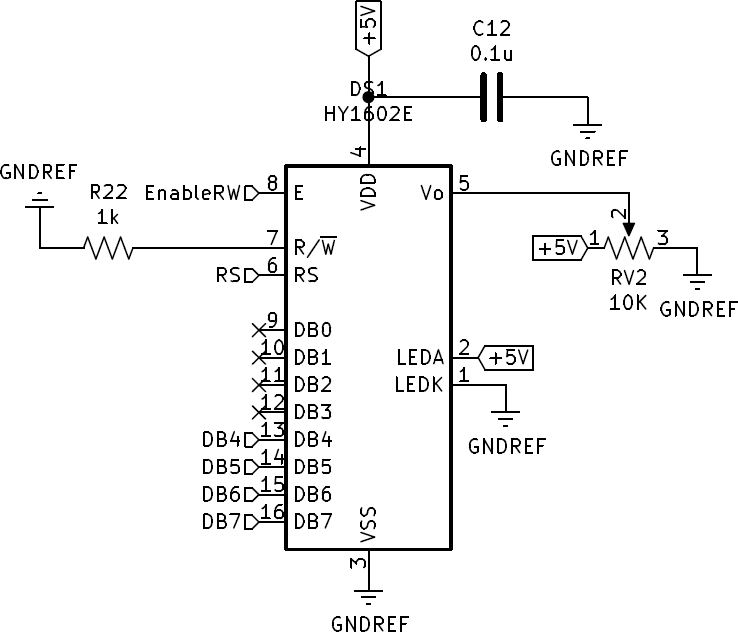


Figura 24 – Circuito do LCD

Para este projeto, usaremos o LCD apenas de modo a inserir dados nele. O pino

𝑅/𝑊̅ é usado para escolher entre leitura (*read*) e escrita (*write*), ao aterrá-lo, estamos desabilitando o modo *read.*

Os pinos *DB0* à *DB7* são usados como entrada de dados ou comandos, portanto, podemos enviar dados de até 1 byte por vez ao display. Para fins de economia, e

também para diminuir a quantidade de pinos usados no PIC, este projeto opera o display em modo 4 bits, note que apenas as entradas de DB4 à DB7 estão sendo usadas.

No esquemático usaram HY1602E e compraram o 1602. Assim foi necessário trocar a pinagem como visto na figura. 16 aterramos e 15 metemos 5V. Aterramos o R22. 9,10,11,12 também foram aterrados.

O funcionamento de escrita, em modo 4 bits, no LCD é simples (como 𝑅/𝑊̅ está

aterrado, não vamos considera-lo). Inicialmente habilita-se *RS* (nível lógico alto), para indicar o envio de dados, depois basta enviar o nibble menos significativo (os 4 últimos bits da palavra), para gravar esse nibble é necessário um pulso de clock no pino *E* (*Enable*), após esse processo é preciso enviar o nibble mais significativo (4 primeiros bits da palavra) e, por fim, enviar outro pulso de clock pelo pino *E.* O exemplo da figura a seguir escreve a letra *A* (0b01000001) no LCD.

Figura 25 – Escrita da palavra *A* (maiúsculo) em modo 4 bits

# OUTROS PERIFÉRICOS

Ainda há alguns periféricos simples para tratar, tais como o *sheet* dos LEDs, o circuito reset do microcontrolador e o *trimpot*. O *sheet* dos LEDs pode ser visto na figura 26, trata-se de 4 LEDs (figura 27) ligados em *RA0*, *RA1*, *RA2* e *RA3* (a função desses LEDs fica por conta do firmware).

Figura 26 – *Sheet* dos LEDs

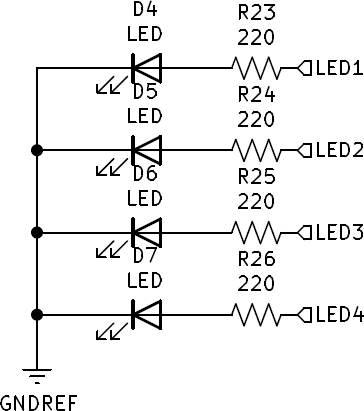


Figura 27 – Circuito dos LEDs

O microcontrolador é resetado através do pino *RE3* que possui a função 𝑀̅̅̅𝐶̅̅𝐿̅̅𝑅̅, responsável por resetar o módulo. Este pino é ativado em nível lógico baixo, ou seja, seu circuito conta com um resistor de pull-up identificado como R1. O capacitor C4 funciona como filtro. O circuito pode ser analisado na figura 28.

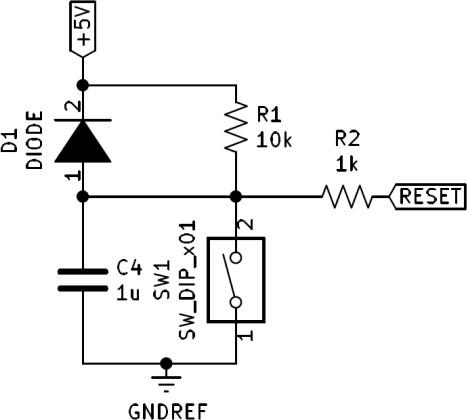


Figura 28 – Circuito reset do microcontrolador

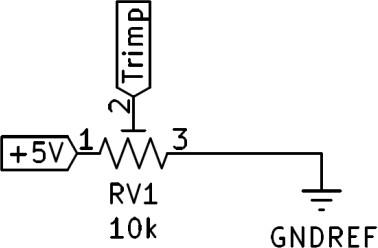
Por último temos o trimpot, ligado ao pino *RA5* (analógico), que foi implementado pois faz parte dos requisitos de projeto. Sua função fica a cargo do firmware.

Figura 30 – Trimpot

# RELATÓRIO DE VERIFICAÇÃO DE ERROS DE PROJETO

O software KiCad, utilizado para o desenvolvimento do esquema elétrico, gera um relatório de erros que apontam possíveis erros ou advertências. Erros são sérios e devem ser tratados, o projeto desenvolvido não possui nenhum erro. As advertências funcionam como alerta que algo pode estar errado, fica a critério dos projetistas decidirem o que fazer.

Para este projeto, o relatório apresentou apenas uma advertência muito específica da ferramenta KiCad, por isso ela foi descartada. Segue o relatório:



# PLACA DE CIRCUITO IMPRESSO

Durante a seção anterior, acompanhamos o desenvolvimento de todo o esquema elétrico do projeto. Nesta seção, veremos o desenvolvimento de toda a placa de circuito impresso e, novamente, será mostrado cada detalhe dessa parte do projeto.

Assim como feito na seção 3.1, iniciaremos analisando os requisitos de projeto para a placa. O desenho da placa será mostrado mais adiante assim como uma visão tridimensional, função disponível na ferramenta KiCad. Na última seção desde capítulo, será apresentado o relatório de verificação de erros de projeto gerado pelo KiCad.

# REQUISITOS

Esta seção especifica os requisitos de projeto para o desenvolvimento da placa. Note que há duas tabelas abaixo, a tabela 2 detalha os requisitos físicos da placa, tais como dimensões e layout, já na tabela 3, vemos os detalhes para a fabricação da placa.

Os softwares, assim como o KiCad, responsáveis por construir PCB (*Printed Circuit Board*), também conhecidas como PCI (Placa de Circuito Impresso), possuem a capacidade de gerar arquivos eletrônicos de fabricação no formato Gerber RS274X, esses arquivos são usados para a confecção física da placa.

Acompanhe, nas tabelas a seguir, todos os requisitos de projeto voltados para o desenvolvimento da placa de circuito impresso.

| Classe | Requisitos |
| --- | --- |
|  | Possuir dimensão de até 8x8 cm2; |
|  | Ser de dupla face e utilizar a face inferior somente como plano |
|  | de terra; |
|  | Possuir capacitores de supressão de tensão entre a |
| Características físicas da placa se circuito impresso | alimentação e o sinal de terra de cada circuito integrado utilizado pelo circuito; |
| Sua face superior deve possuir uma camada de texto (silk) na qual haja a identificação de cada componente eletroeletrônico |
|  | e a identificação do grupo que a desenvolveu; |
|  | Possuir quadro furos de fixação dispostos em seus cantos; |
|  | Possuir identificação de todas as conexões de entrada e de |
|  | saídas presentes na placa. |

Tabela 2 – Tabela de requisitos físicos para a placa de circuito impresso

| Classe | Requisitos |
| --- | --- |
| Espaçamento e dimensões de trilhas e afins | Utilização do formato de arquivos eletrônicos de fabricação Gerber RS274X; |
| Mínima largura para trilhas de sinais de 8 mils; |
| Mínima largura para trilhas de alimentação de 12 mils; |
| Mínimo espaçamento entre trilhas, furos e ilhas de 8 mils |
| Mínimo diâmetro de furo de vias de 12 mils; |
| Mínimo diâmetro de ilhas de vias de 25 mils; |
| Não utilizar microvias. |

Tabela 3 – Tabela de requisitos de produção da placa de circuito impresso

# DESENHO DA PLACA DE CIRCUITO IMPRESSO

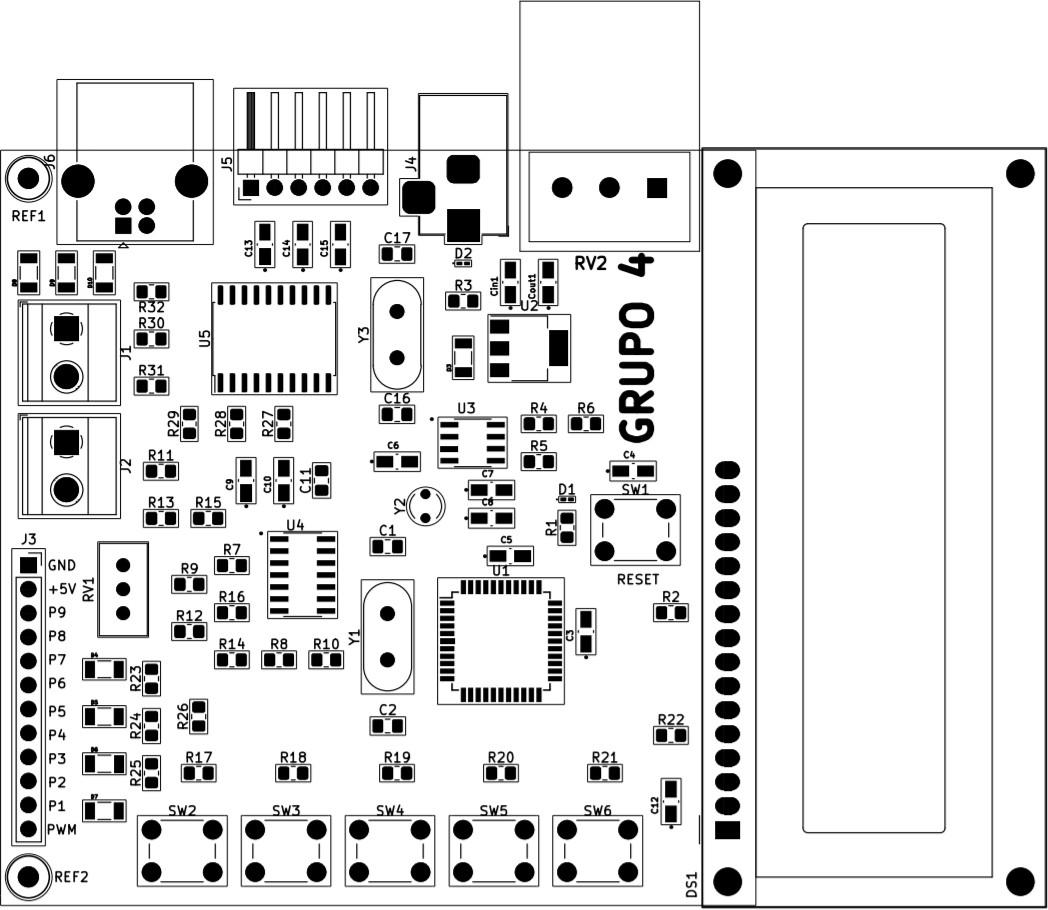


Figura 31 – Disposição dos componentes sobre a placa

Na figura 31 podemos visualizar o desenho da *PCB* gerado pelo KiCad. Note que cada componente possui uma identificação, com isso podemos identifica-lo no esquema elétrico para entendermos melhor sua função.

Segundo os requisitos presente na seção 4.1, a placa deve conter duas faces. A face inferior deve passar toda a rede de terra (*GNDREF*), porém no projeto algumas trilhas de outras redes foram passadas pela face inferior (figura 32). As trilhas da camada superior podem ser visualizadas na figura 33.

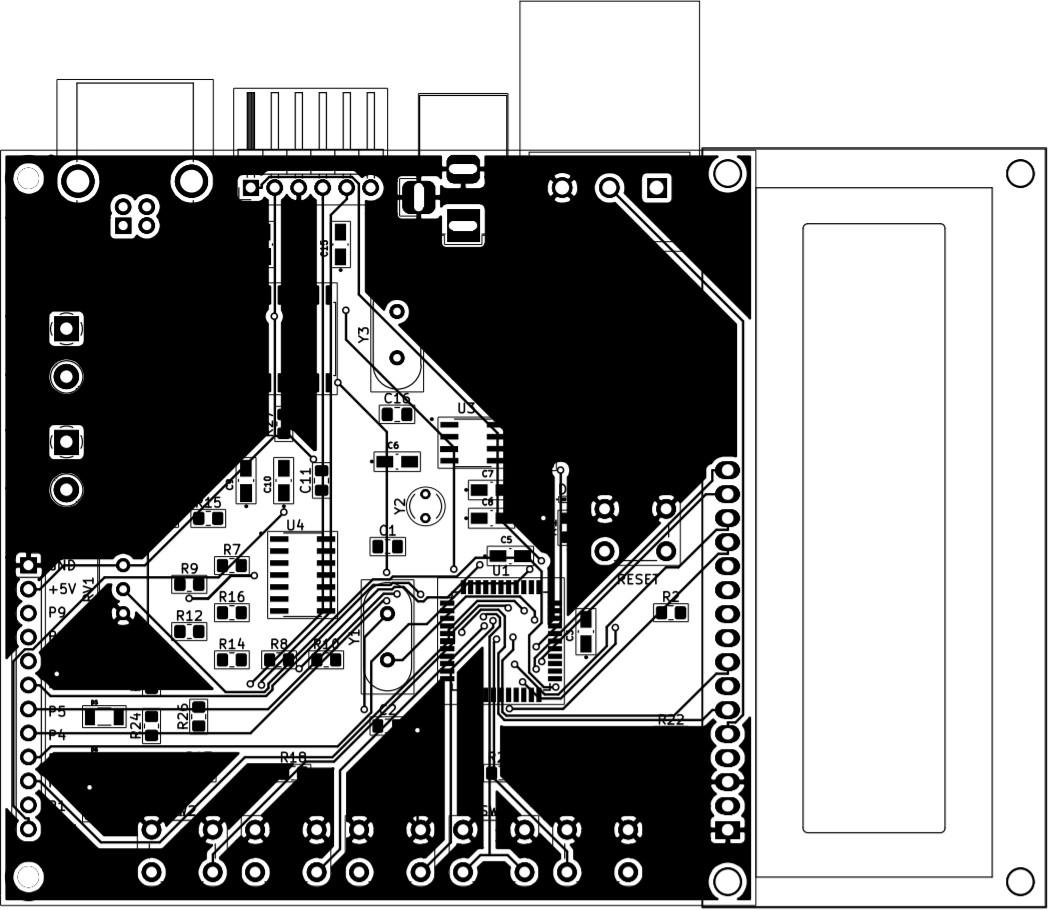


Figura 32 – Exibição da camada inferior da placa

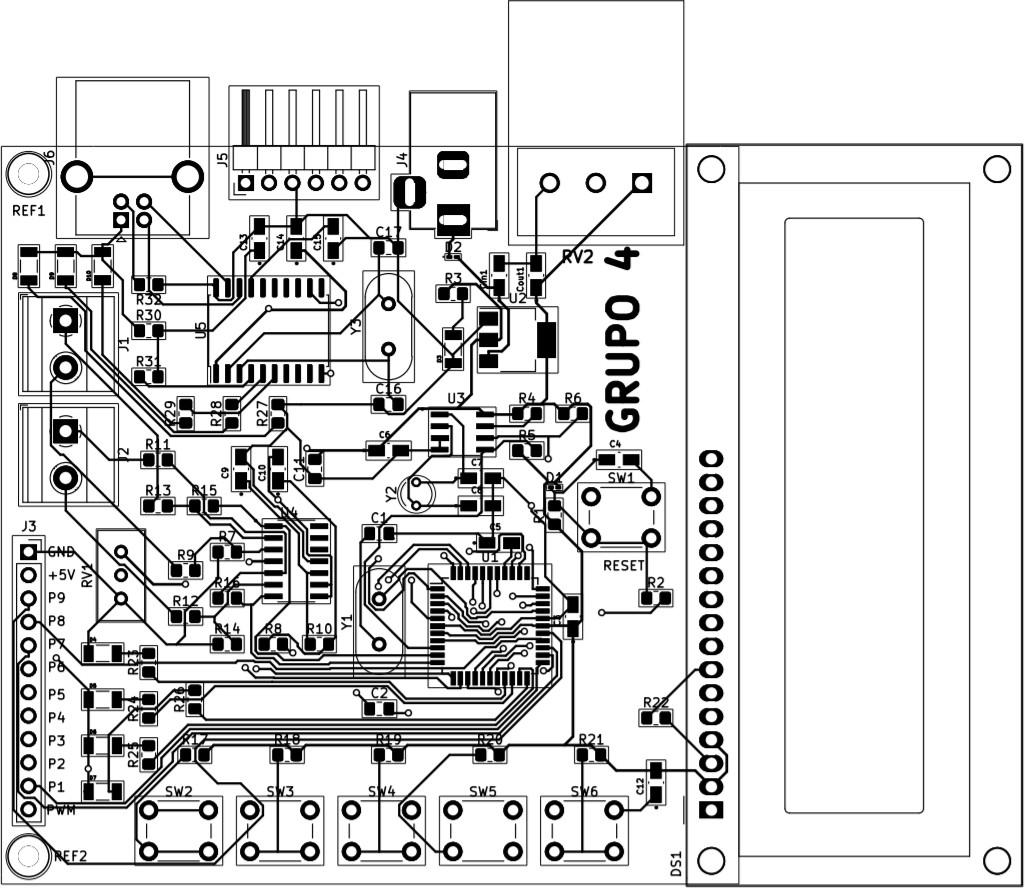


Figura 33 – Exibição da camada superior da placa

# VISÃO TRIDIMENSIONAL DO CIRCUITO PROJETADO

A figura 34 foi dividida em 3 partes, A, B e C para uma melhor apresentação da placa. Dessa forma é possível visualizar a placa em 3 dimensões para ter um ideia de como ficaria o projeto após a confecção. A figura 35 mostra somente a placa, sem os componentes.

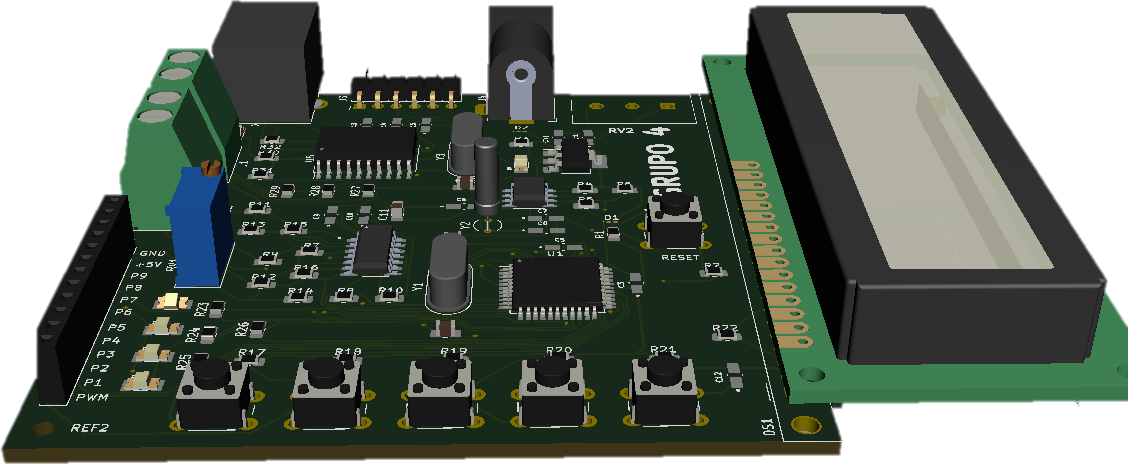


Figura 34A – Visão 3D da placa

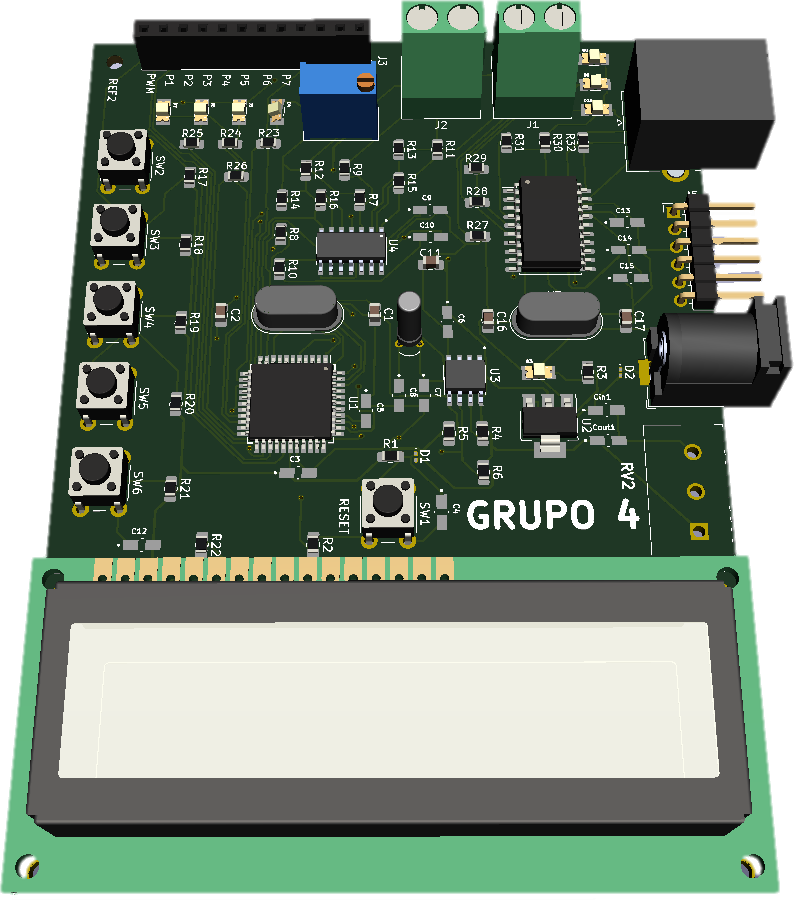


Figura 34B – Visão 3D da placa



Figura 34C – Visão 3D da placa

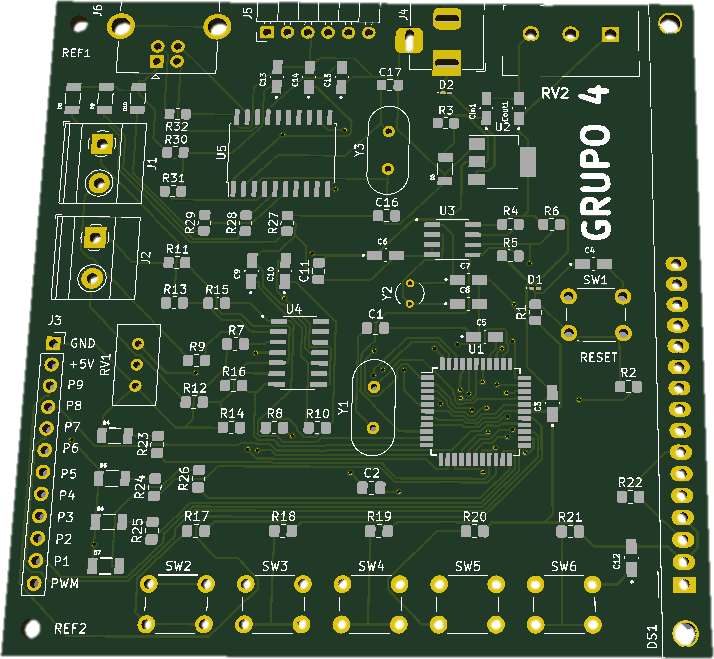


Figura 35 – Placa sem os componentes

# RELATÓRIO DE VERIFICAÇÃO DE ERROS DE PROJETO

Assim como fizemos para o esquema elétrico do projeto, faremos novamente para a placa de circuito impresso. O KiCad também possui ferramenta pra verificar possíveis erros na construção da placa de circuito impresso.

Os erros apontados pela ferramenta são ilhas muito próximas, isso ocorre pois o *footprint* (a pegada) de dois diodos usado na placa estão com ilhas muito próximas. Novamente fica a cargo dos projetistas decidirem como proceder. Para esse projeto podemos ignorar tal problema.



# ESPECIFICAÇÕES ELÉTRICAS

Nesta seção serão abordados os tópicos referentes às especificações elétricas e níveis de consumo da placa.

# NÍVEIS DE ALIMENTAÇÃO SUPORTADOS

O CI LD1117DT50TR, regulador de tensão implementado na alimentação, possui uma faixa de tensão de entrada de 5V – 15V, a uma temperatura de 25°C. Sua saída de 5V é garantida independentemente de sua entrada, desde que seja respeitado os parâmetros do CI. Como visto na seção 3.2.1, a saída do regulador alimenta todo o circuito, portanto podemos considerar a tabela abaixo para a faixa de alimentação do sistema.

| Símbolo | Parâmetro | Min. | Typ. | Max. | Unidade |
| --- | --- | --- | --- | --- | --- |
| VI | Tensão de entrada | 7 |  | 15 | V |

Tabela 4 – Níveis de alimentação da placa

# FAIXA DE NÍVEIS DE CONSUMO ESTIMADAS

| Símbolo | Parâmetro | Min. | Typ. | Max. | Unidade |
| --- | --- | --- | --- | --- | --- |
| PTOT | Potência dissipada | 8,5 |  | 19,5 | W |
| II | Corrente de entrada | 1,2 |  | 1,3 | A |
| IO | Corrente de saída do regulador | 1,2 |  | 1,3 | A |

Tabela 5 – Faixa de níveis de consumo estimadas

# CORRENTES E TENSÕES MÁXIMAS DE ENTRADA

| Símbolo | Parâmetro | Valor | Unidade |
| --- | --- | --- | --- |
| Vinmax | Tensão máxima de entrada | 15 | V |
| Iinmax | Corrente máxima de entrada | 1300 | mA |

Tabela 6 – Correntes e tesões máximos de entrada

# CORRENTES E TENSÕES MÁXIMAS DE SAÍDA

| Símbolo | Parâmetro | Valor | Unidade |
| --- | --- | --- | --- |
| Voutmax | Tensão máxima de saída | 5 | V |
| Ioutmax | Corrente máxima de saída | 1300 | mA |

Tabela 7 – Correntes e tensões máximos de saída

# CARACTERÍSTICAS DE COMUNICAÇÃO

A placa contém os dois tipos de comunicação, paralela e serial. No LCD podemos notar que há a comunicação paralela, já que a informação é passada ao LCD de *nibble* em *nibble*.

Na comunicação USB\_B (2.0) remos o protocolo UART, já que o MCP2200 trabalha dessa forma. A comunicação entre o PIC e o MCP7940N, funciona com o protocolo I²C. A tabela abaixo, retirada das [aulas](https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxybWFhbG1laWRhfGd4OjRiMmY2NzFlYzFlNTFkMGE) do professor Rodrigo Maximiano Antunes De Almeida, mostra as taxas para essas comunicações.

| Protocolo | Taxa (bits/s) | Taxa(bytes/s) |
| --- | --- | --- |
| Serial UART max | 2,7648M | 345,6k |
| I²C | 3,4M | 425k |

Tabela 8 – Taxas de comunicação serial

# PROGRAMA EMBARCADO DE VALIDAÇÃO

Após acompanhar os detalhes com relação ao esquema elétrico e à placa de circuito impresso, daremos uma atenção especial ao *firmware*.

*Firmware* nada mais é do que o software gravado na memória *DRAM* do microcontrolador (PIC18F4550) a fim de atribuir uma função aos componentes da placa. O software desenvolvido para este projeto possui a finalidade de testar alguns requisitos da placa. Os requisitos estão melhor detalhados na próxima seção.

# REQUISITOS

A tabela a seguir apresenta os requisitos de projeto que deve ser seguido ao projetar o software embarcado de validação.

| Classe | Requisitos |
| --- | --- |
| Embarcado de Validação | Detectar e identificar o acionamento do teclado; |
| Testar a exibição de mensagens no visor de cristal líquido; |
| Testar a sinalização de estados por meio dos diodos de sinalização; |
| Estipular e testar a leituras de informações de data e hora; |
| Testar a leitura de sinais pelas entradas diferenciais; |
| Testar a geração da saída analógica; |
| Testar a recepção e o envio de dados pela via serial |
| Criar um menu de opção que faça o uso do LCD e do teclado existente. |

Tabela 9 – Requisitos de projeto para o embarcado de validação

# MODELO DE OPERAÇÃO GERAL

Veremos a seguir o modelo de operação usado no software, ou seja, qual o layout geral usado para o projeto. O diagrama 1, a seguir, representa o funcionamento do sistema embarcado de forma abstrata.

| Configuráçá˜o | Principál | Requisição ao sistema  Resposta do sistema | Bibliotecás |
| --- | --- | --- | --- |
| Bá´sico | | |

Diagrama 1 – Fluxo de dados e funcionamento geral do sistema embarcado

O diagrama anterior demonstra como deve ser o fluxo de dados e a operação do sistema. O software conta com arquivo de configuração, responsável por configurar alguns registros importante do PIC, como habilitar o *Master Clear*. Contamos, também, com um arquivo que contém comandos básicos, além de mapear todos os registros necessários de acordo com o [*datasheet*](https://ww1.microchip.com/downloads/en/devicedoc/39632c.pdf)do microcontrolador. Além disso, o programa principal, conta com uma ampla variedade de bibliotecas, cada uma responsável por gerenciar um módulo contido na placa.

Note, no diagrama 1, que o fluxo de dados ocorre entre o programa principal e as bibliotecas, ou seja, o programa principal solicita um determinado dado a um módulo e este último devolve o dado solicitado ao programa principal.

Mais adiante será apresentado todas as bibliotecas e suas respectivas funções, que podem ser usadas como o programador achar necessário.

# ACESSO ÀS FUNCIONALIDADES

Veremos, detalhadamente, o funcionamento das bibliotecas presentes no *firmware.* Primeiramente faremos uma rápida análise no arquivo de configuração e no arquivo básico.

O arquivo de configuração dispõe de quatro configurações realizadas no microcontrolador.

| Configuração | Descrição | Valor |
| --- | --- | --- |
| *MCLRE* | *Master Clear* habilitado | *ON* |
| *FOSC* | Oscilador com cristal externo HS | *HS* |
| *WDT* | *Watchdog* controlado por software | *OFF* |
| *LVP* | Sem programação em baixa tensão | *OFF* |

Tabela 10 – Configurações realizadas no arquivo *config.h*

Em seguida temos o arquivo *basico.h,* esse arquivo faz o mapeamento de todos os registros necessários (*PORT*, *LATCH*, *TRIS,* entre outros) para o correto funcionamento do programa de acordo com a necessidade do programador. Esse arquivo define, também, algumas operações básicas como mostrado na tabela 11, assim como duas constantes usadas em todo o código: *HIGH* e *LOW*

| Operação | Descrição |
| --- | --- |
| *BitSet* | Habilita um bit (nível lógico alto) |
| *BitClr* | Desabilita um bit (nível lógico baixo) |
| *BitFlp* | Inverte o valor do bit |
| *BitTst* | Testa um bit, retorna 1 caso ele esteja habilitado |

Tabela 11 – Operações descritas no arquivo *basico.h*

# LEDs

Veremos, agora, todas as bibliotecas presentes no projeto. Iniciaremos com a biblioteca dos LEDS. As funções mostradas aqui serão aquelas que estão abertas às outras bibliotecas, ou seja, somente serão analisadas as funções presentes no arquivo *header.*

| **led.h** |
| --- |
| * <<constant>> LED1 : int = 0 * <<constant>> LED2 : int = 1 * <<constant>> LED3 : int = 2 * <<constant>> LED4 : int = 2 |
| + led\_set(led : unsigned char, status : unsigned char) : void  + led\_set\_all() : void  + led\_clr\_all() : void  + led\_tst(led : unsigned char) : int  + led\_init() : void |

Header 1 – LEDs

A biblioteca dos LEDs conta com algumas funções que permitem acendê-los ou apaga-los, além de quatro constantes que definem os pinos do microcontrolador onde cada LED está conectado.

| Função | Descrição |
| --- | --- |
| *led\_set* | Liga um LED, seu primeiro parâmetro indica o led e o segundo  parâmetro indica ligado (*HIGH*) ou desligado (*LOW*). |
| *led\_set\_all* | Liga todos os LEDs. |
| *led\_clr\_all* | Desliga todos os LEDs. |
| *led\_tst* | Testa um LED, seu primeiro parâmetro indica o LED a ser testado. A  função retorna 1 caso o LED esteja aceso e 0 caso esteja apagado. |
| *led\_init* | Configura os pinos do PIC, onde estão ligados os LEDs, como saída.  Essa função deve ser chamada antes de usar qualquer outra função. |

Tabela 12 – Funções da biblioteca dos LEDs

# TECLADO

Para o teclado, foi usado como constantes nomes iniciados com *SW2* até *SW6*, esses nomes foram escolhidos com base no projeto da placa, que imprime exatamente esses nomes na camada *silkscreen.*

| **keyboard.h** |
| --- |
| * <<constant>> SW2 : int = 0 * <<constant>> SW3 : int = 1 * <<constant>> SW4 : int = 2 * <<constant>> SW5 : int = 3 * <<constant>> SW6 : int = 4 |
| + key\_pressed() : unsigned char  + key\_is\_pressed(key : unsigned char) : unsigned char  + keyboard\_init() : void |

Header 2 – Teclado

| Função | Descrição |
| --- | --- |
| *key\_pressed* | Retorna uma constante *int* representando o valor do botão pressionado. Caso não haja botão pressionado, a função retorna  9010. |
| *key\_is\_pressed* | Testa se um botão, passado pelo argumento da função, está  pressionado (retornando 1) ou não (retornando 0). |
| *keyboard\_init* | Configura os pinos, do PIC, ligados aos botões, como entrada de dados. Essa função deve ser chamada antes de usar qualquer outra  função. |

Tabela 13 – Funções da biblioteca do teclado

# LCD

A biblioteca do LCD conta com três comandos pré-definidos como constantes: CLEAR → limpa o LCD

FIRST\_LINE → leva o cursor para o início da primeira linha SECOND\_LINE → leva o cursor para o início da segunda linha

| **lcd.h** |
| --- |
| * <<constant>> CLEAR : int = 1 * <<constant>> FIRST\_LINE : int = 128 * <<constant>> SECOND\_LINE : int = 192 |
| + lcd\_data(data : unsigned char) : void  + lcd\_command(cmd : unsigned char) : void  + lcd\_print(str : char\*) : void  + lcd\_putValue(value : unsigned char) : void  + lcd\_init() : void |

Header 3 – LCD

| Função | Descrição |
| --- | --- |
| *lcd\_data* | Exibe uma palavra de 1 byte no LCD. |
| *lcd\_command* | Envia um comando ao LCD. |
| *lcd\_print* | Envia uma sequência de palavras, 1 byte cada, para ser exibido no  LCD. |
| *lcd\_putValue* | Envia um valor ao LCD. |
| *lcd\_init* | Configura os pinos do PIC conforme a necessidade do LCD. Faz a configuração inicial do display. Essa função deve ser chamada antes  de usar qualquer outra função. |

Tabela 14 – Funções da biblioteca do LCD

# PWM

| **pwm.h** |
| --- |
|  |
| + pwm\_set\_perc(porcento : unsigned char) : void  + pwm\_set\_freq(freq : unsigned char) : void  + pwm\_init() : void |

Header 4 – PWM

| Função | Descrição |
| --- | --- |
| *pwm\_set\_perc* | Altera a porcentagem da modulação de acordo com o que foi passado  pelo parâmetro da função. |
| *pwm\_set\_freq* | Ajusta a frequência da onda. |
| *pwm\_init* | Configura o pino do microcontrolador como saída pwm. Essa função  deve ser chamada antes de usar qualquer outra função. |

Tabela 15 – Funções da biblioteca do PWM

# LISTA DE MATERIAIS

Segue a tabela de todos os componentes utilizados no projeto da placa. Os capacitores e resistores sem valores fazem parte do filtro *butterworth* que não foi projetado.

| Designador | Pacote | Qtd. | Designação |
| --- | --- | --- | --- |
| U1 | TQFP-44\_10x10mm\_P0.8mm | 1 | PIC18F4550-IPT |
| REF2, REF1 | MountingHole\_2.2mm\_M2 | 2 | MountingHole\_2.2mm\_M2 |
| C1, C2 | C\_0805\_2012Metric\_Pad  1.18x1.45mm\_HandSolder | 2 | 22pF |
| C3, C13, Cin1 | CAPMP3216X180N | 3 | 100nF |
| C4 | CAPMP3216X180N | 1 | 1uF |
| C5 | CAPMP3216X180N | 1 | 1uF |
| C6 | CAPMP3216X180N | 1 | 100nF |
| C7, C8 | CAPMP3216X180N | 2 | 15pF |
| C9, C10 | CAPMP3216X180N | 2 | C |
| C11 | C\_0805\_2012Metric\_Pad  1.18x1.45mm\_HandSolder | 1 | 5pF |
| C12 | CAPMP3216X180N | 1 | 0.1u |
| C14, C15 | CAPMP3216X180N | 2 | 0.1uF |
| C16, C17 | C\_0805\_2012Metric\_Pad  1.18x1.45mm\_HandSolder | 2 | 22pF |
| Cout1 | CAPMP3216X180N | 1 | 10uF |
| D1, D2 | D\_0201\_0603Metric\_Pad  0.64x0.40mm\_HandSolder | 2 | DIODE |
| D3, D4, D5, D6,  D7, D8, D9, D10 | RESC1206X70N | 8 | LED |
| DS1 | WC1602A | 1 | HY1602E |
| J1, J2 | TerminalBlock\_Phoenix\_MKDS-1,5-2-  5.08\_1x02\_P5.08mm\_Horizontal | 2 | Screw\_Terminal\_01x02 |
| J3 | PinSocket\_1x12\_P2.54mm\_Vertical | 1 | Conn\_01x12\_Female |
| J4 | BarrelJack\_Horizontal | 1 | Barrel\_Jack\_Switch |
| J5 | PinHeader\_1x06\_P2.54mm\_Horizontal | 1 | Conn\_PIC\_ICSP\_ICD |

| J6 | USB\_B\_OST\_USB-B1HSxx\_Horizontal | 1 | USB\_B |
| --- | --- | --- | --- |
| R1, R4, R5, R6,  R17, R18, R19, R20, R21, R30 | R\_0805\_2012Metric\_Pad 1.20x1.40mm\_HandSolder | 10 | 10kΩ |
| R2, R22, R32 | R\_0805\_2012Metric\_Pad  1.20x1.40mm\_HandSolder | 3 | 1kΩ |
| R3, R23, R24,  R25, R26 | R\_0805\_2012Metric\_Pad  1.20x1.40mm\_HandSolder | 5 | 220Ω |
| R7, R14, R15,  R16 | R\_0805\_2012Metric\_Pad  1.20x1.40mm\_HandSolder | 4 | 12kΩ |
| R8, R10 | R\_0805\_2012Metric\_Pad  1.20x1.40mm\_HandSolder | 2 | R\_US |
| R9, R11, R12,  R13 | R\_0805\_2012Metric\_Pad  1.20x1.40mm\_HandSolder | 4 | 20kΩ |
| R27, R28, R29,  R31 | R\_0805\_2012Metric\_Pad  1.20x1.40mm\_HandSolder | 4 | 470Ω |
| RV1 | Potentiometer\_Bourns\_3296W\_Vertical | 1 | 10kΩ |
| SW1, SW2, SW3,  SW4, SW5, SW6 | SW\_PUSH\_6mm\_H5mm | 6 | SW\_DIP\_x01 |
| U2 | SOT-223-3\_TabPin2 | 1 | LD1117S50TR\_SOT223 |
| U3 | SOIC127P600X175-8N | 1 | MCP7940N-xMS |
| U4 | SOIC127P600X175-14N | 1 | LM324DG |
| U5 | SOIC-20W\_7.5x12.8mm\_P1.27mm | 1 | MCP2200-I-SO |
| Y1 | Crystal\_HC49-4H\_Vertical | 1 | 20MHz |
| Y2 | Crystal\_AT310\_D3.0mm\_L10.0mm\_Vertical | 1 | 32.768kHz |
| Y3 | Crystal\_HC49-4H\_Vertical | 1 | 12MHz |
| RV2 | P160KN0QC15B100K | 1 | 10KΩ |

Tabela 16 – Todos os componentes usados no projeto

# ANEXOS (CÓDIGOS)

Anexo 1 – Main (A)



Anexo 2 – Main (B)

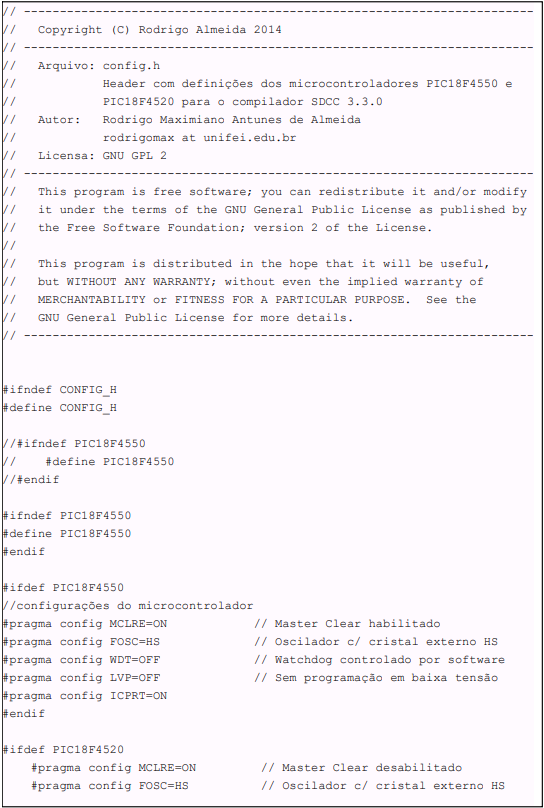


Anexo 3 – Main (C)

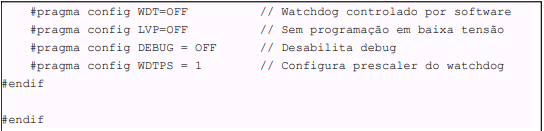


Anexo 4 – Main (D)

Arquivos *header*

**

Anexo 5 – config.h(A)



Anexo 6 – config.h(B)



Anexo 7 – basico.h (A)



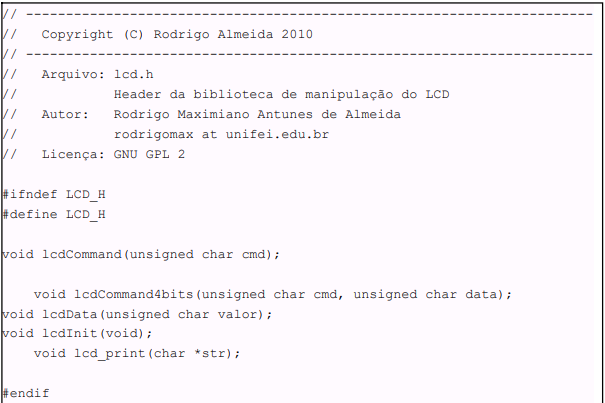
Anexo 8 – basico.h (B)



Anexo 9 – keyboard.h



Anexo 10 – led.h



Anexo 11 – lcd.h



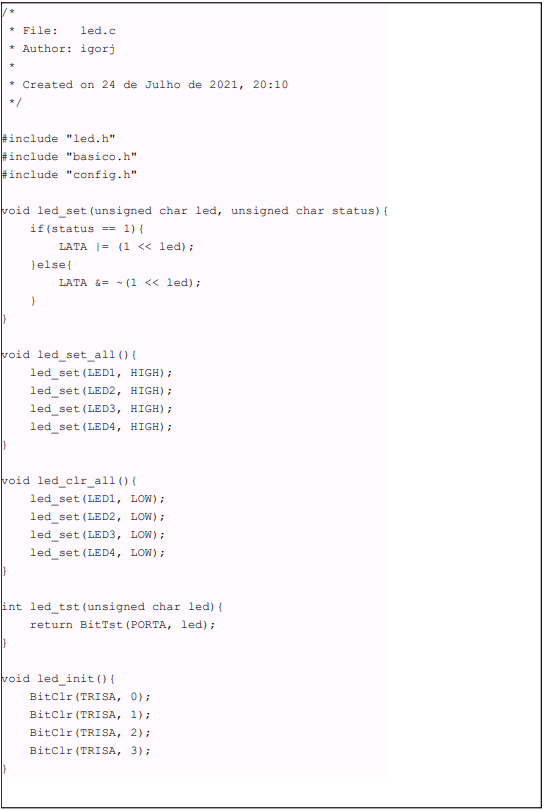
Anexo 12 – pwm.h



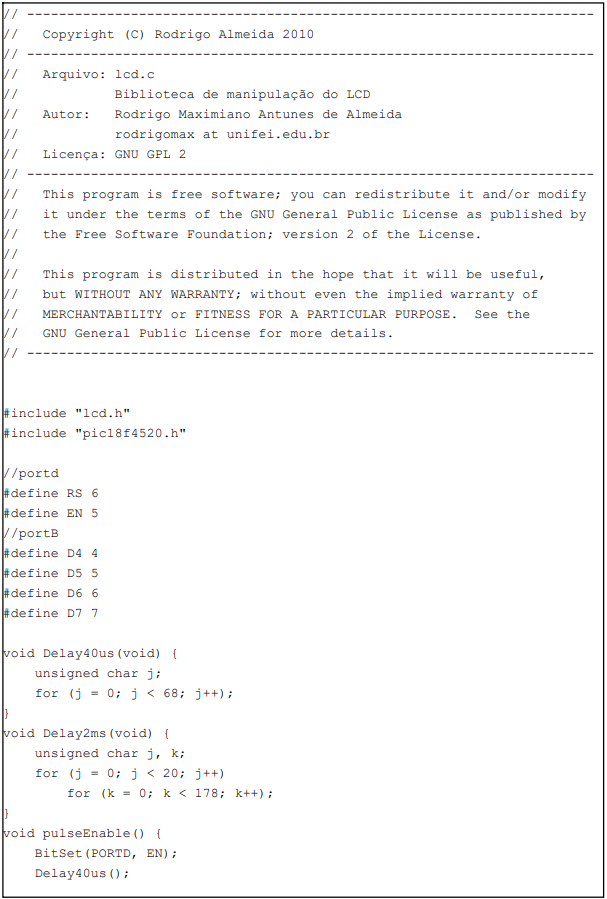
Anexo 13 – rtc.h

Arquivos C

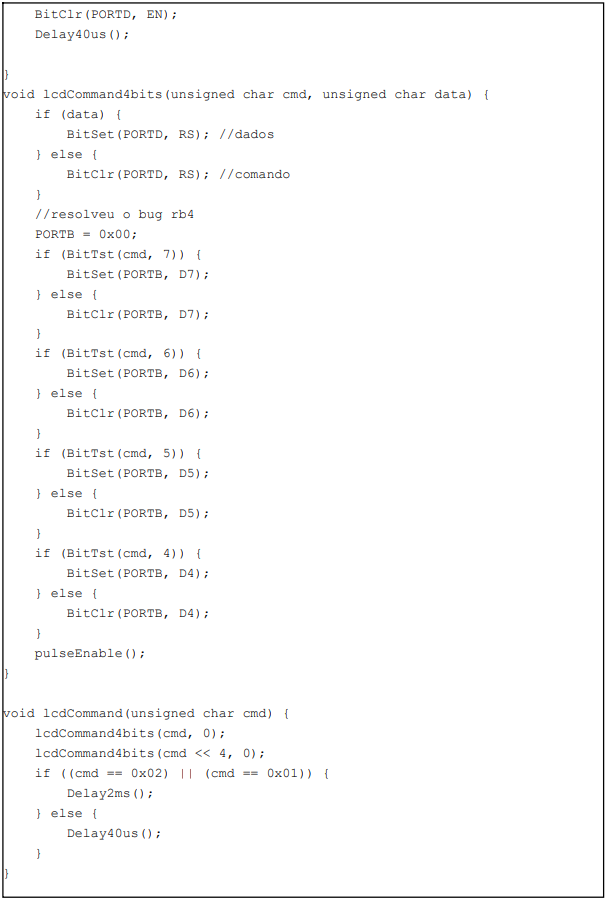
Anexo 14 – keyboard.c



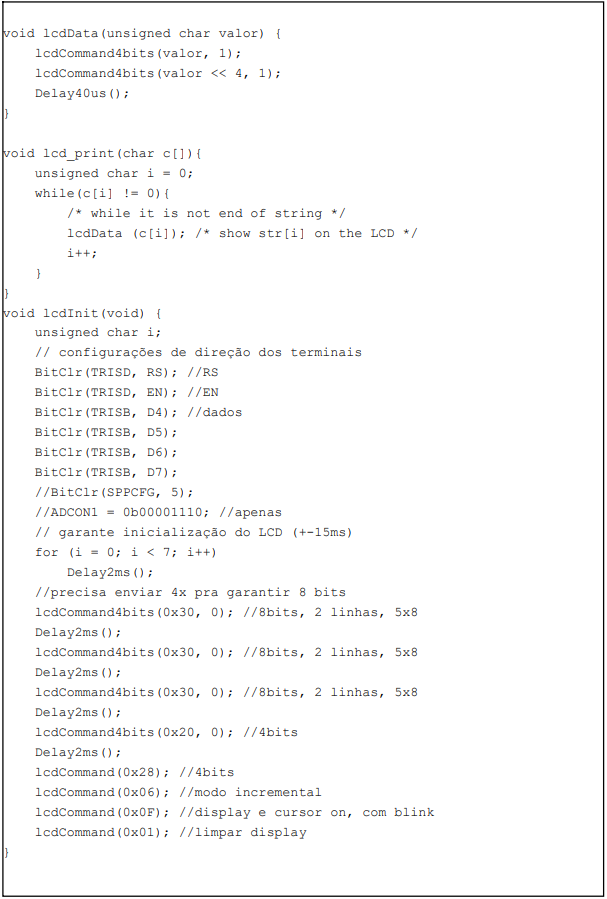
Anexo 15 – led.c



Anexo 16 – lcd.c (A)



Anexo 17 – lcd.c (B)



Anexo 18 – lcd.c (C)



Anexo 19 – pwm.c



Anexo 20 – rtc.c (A)



Anexo 21 – rtc.c (B)



Anexo 22 – rtc.c (C)



Anexo 23 – rtc.c (D)