**Programa Institucional de Bolsas de Iniciação Científica – PIBIC**

**RELATÓRIO PARCIAL**

# **Matrícula:**.\_\_\_\_\_\_\_.\_\_\_\_\_\_\_-\_\_\_\_ **C.R.:** \_\_\_\_\_\_\_\_

Curso: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DADOS DO ORIENTADOR

Nome:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Matrícula Siape:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_CPF:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Endereço:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Bairro:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Cidade:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_UF:\_\_\_\_\_\_\_\_\_\_\_\_\_\_CEP:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_E-mail:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-\_\_\_\_\_\_\_\_\_ Telefone 1: (\_\_\_\_\_)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Telefone 2: (\_\_\_\_\_)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. IDENTIFICAÇÃO DO PROJETO
2. Título do Projeto:\_Automação de Sistemas Elétricos de Potência com Dispositivos Eletrônicos Inteligentes\_\_\_\_\_\_\_\_\_
3. Local de Realização (Unidade/Instituto/Departamento/Laboratorio):\_Laboratório de Eletrônica Básica\_UFF\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
4. Endereço:\_Rua Passo da Pátria 156, Bloco D\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
5. Bairro:São Domingos\_\_\_\_\_\_\_\_\_\_Cidade:Niterói\_\_\_\_\_\_\_\_ \_\_UF:RJ\_\_\_CEP:\_24210-240\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   * + - 1. DADOS DO BOLSISTA
         2. Nome:Lucas Abdalla Menezes\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
         3. Matrícula:\_116.038.019\_\_\_\_\_\_\_\_\_CPF:171.509.847-11\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_CR:\_\_6.9\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
         4. Curso/Departamento/Instituto:\_Egenharia Elétrica / TEE – Departamento de Engenharia Elétrica / UFF – Universidade Federal Fluminense \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
         5. Endereço:Rua Santa Rosa 91, Apto. 906\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
         6. Bairro:Santa Rosa\_\_\_\_\_Cidade:\_Niterói\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_UF:\_RJ\_\_\_\_\_\_CEP:24240-225\_\_\_\_\_\_\_\_\_\_\_ E-mail:\_lucasabdalla@id.uff.br\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Telefone 1: (\_21\_\_)971475033\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Telefone 2: (\_\_\_\_\_)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

##### 

**1. TÍTULO**

**AUTOMAÇÃO DE SISTEMAS ELÉTRICOS DE POTÊNCIA COM DISPOSITIVOS ELETRÔNICOS INTELIGENTES**

**2. INTRODUÇÃO**

**Veja o projeto e aproveite algum texto dali. Use também o que você já escreveu para o artigo.**

**3. METODOLOGIA**

Etapa 1:

A primeira etapa do projeto foi destinada ao estudo de bibliografias recomendadas pelo orientador, sobre alguns aspectos iniciais que viriam a ser encontrados durante o desenvolvimento da pesquisa. Foram explorados os seguintes temas em artigos científicos, Teses e trabalhos de conclusão de curso de graduação: Técnicas para medição de corrente de maneira não intrusiva em sistemas de distribuição de energia elétrica e tipos de algoritmos da inteligência computacional para identificação de padrões. (Falta Indicar as Referências).

Etapa 2:

Nessa etapa, foi iniciado o desenvolvimento do artigo sobre o uso de sistemas embarcados de baixa custo para o monitoramento não intrusivo de cargas ou NILM (do inglês Non-Intrusive Load Monitoring). Além do estudo de outras bibliografias para investigar os requisitos mínimos de software e hardware do sistema para o tratamento dos dados, de maneira que a proposta de reconhecimento de cargas não fosse afetada.

Etapa 3:

Na terceira etapa, foram estudadas plataformas computacionais de baixo custo disponíveis no mercado e sua aderência aos requisitos mínimos de software e hardware escolhidos na etapa anterior. Foi feito também, o projeto do circuito para medições de corrente, incluindo a seleção de componentes para montagem do circuito.

Etapa 4:

Nessa etapa, foi feita a montagem de um protótipo do circuito projetado, e iniciada a elaboração de um código que possa ser reaproveitado para os demais dispositivos que serão testados. Esse código é responsável por fazer a aferição de corrente de uma malha do circuito, e então faz o cálculo do seu valor RMS para cada instante, dado um certo tempo de amostragem.

**4.RESULTADOS**

**4.1 Elaboração do ambiente de testes.**

Após estudados métodos de extração dos valores de corrente, o primeiro passo foi montar uma esquemática do ambiente de testes. Na Figura X, está ilustrado um diagrama com os elementos utilizados no ambiente de testes.

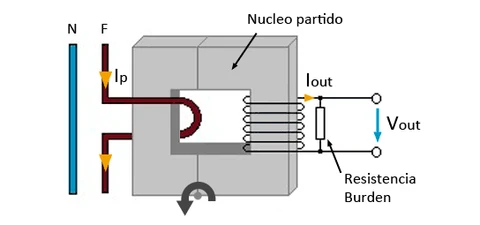
INCLUIR AQUI UMA FIGURA COM O DIAGRAMA DAS TRÊS PARTES E SUA INTERLIGAÇÃO

Após a montagem do esquema anterior, ele funciona da seguinte forma: Uma corrente atravessa o TC, que por sua vez, produz uma tensão no seu enrolamento secundário, que é proporcional ao fluxo enlaçado gerado por essa corrente. Essa tensão está contida no intervalo de [-1,1] V. O microcontrolador utilizado suporta apenas valores de tensão positivos entre 0 e +3,3V na sua porta de leitura para o conversor analógico -digital (ADC). Portanto, foi criado um circuito, que fosse capaz de produzir um deslocamento do nível de tensão CC (Offset), que garanta que essa tensão seja estritamente positiva, e dentro destes limites. Nesse caso, considerando a maior variação possível do sensor, ficará entre +0,65 e +2,65] V. Depois disso, foi dado início ao desenvolvimento de um código que fosse capaz de ler a tensão na porta do ESP32 (porta 34, no nosso caso), e que fizesse a conversão desse valor para equivalência em Ampéres RMS que atravessa o TC. A descrição dos três elementos será feita nas seções a seguir.

**- Um sensor de corrente não-intrusivo - transformador de corrente (TC) tipo janela;**

  
Figura 1: TC tipo janela não intrusivo

O esquema de funcionamento é descrito por Figura 5.

  
*Figura : Esquema de funcionamento do TC*

|  |  |
| --- | --- |
|  |  |

Onde,

- Ir é a corrente produzida no enrolamento secundário;

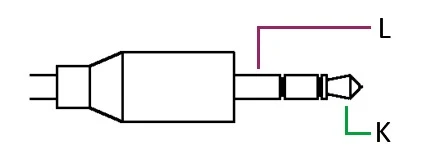
- Ip é a corrente correspondente ao número de condutores que atravessam o núcleo magnético do TC;

-VP é a tensão no terminal primário;

-VS é a tensão produzida na Resistência Burden;

-NP /Ns , representa o relação de transformação;

A tensão de saída fornecida pela TC, que é proporcional à corrente que o atravessa, é entregue por meio de um conector Jack 3,5 mm, que é muito comum no campo de áudio [6]. Onde a tensão de saída é dada pela diferença de potencial entregue pelos contatos, “L” e “K” do conector, representada pela Figura 3.

  
Figura 3: Conector Jack 3,5 mm

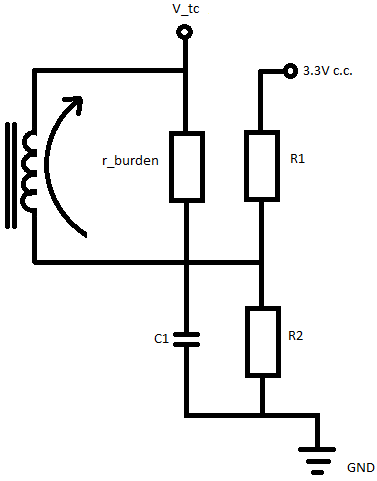
**Circuito de condicionamento do sinal do sensor – deslocamento de nível CC (Corrente Contínua)**

4.1.1 Elaboração e montagem do circuito de deslocamento de nível CC.

Uma vez que a maioria dos microcontroladores só suporta tensão CC na sua entrada, para a criação do circuito, é necessário saber qual a corrente máxima que o TC é capaz de suportar sem que a sua saturação magnética seja atingida, a fim de tornar o sinal de saída estritamente positivo. Com os dados nominais do TC, temos que para uma corrente máxima de 6A, esse produz uma tensão de 1V. Dessa forma, foi montado um circuito que gera, pelo menos, 1Volt de Offset do sinal de entrada, já que essa seria a tensão máxima que o TC consegue reproduzir.

Seguindo [7]

foi possível chegar no esquema da Figura 4.

  
Figura 4: Esquema do Circuito de Offset

Onde alimentamos o circuito com uma tensão de (3.3 V c.c.) para um offset de 1.65V (3.3 V /2). A tensão de saída V\_tc é exatamente a tensão produzida pelo secundário do TC, deslocada de 1,65 V positivamente. Essa portanto, assume valores de (1.65 – 1) V até (1.65V +1) V. Definindo o intervalo [0.65 V, 2.65 V] dependendo do valor instantâneo de corrente demandado pela carga.

4.1.2 Componentes utilizados

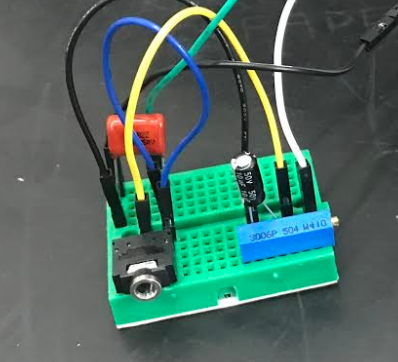
- 1 Trimpot de 500kOhms, separada igualmente entre R1 e R2;

- 1 Capacitor para filtrar sinais de alta frequência que chegam no circuito;

- 1 Conector p2 para receber o sinal de saída do TC;

- 1 Fonte CC de 3.3 V proveniente da própria placa de desenvolvimento ESP32 DEVKIT V4;

O circuito projetado foi montado em um protótipo, utilizando uma matriz de contatos ou protoboard. A Figura X ilustra o protótipo do circuito de condicionamento de sinais montado.

  
Figura 5: Circuito para o deslocamento CC do Sinal

**- Uma plataforma computacional de baixo custo que será responsável por receber e tratar o sinal de saída do circuito montado.**

Para início da bancada de testes, foi utilizado um ESP32 [8]. pelas suas características um pouco mais robustas que outros sistemas computacionais de baixo custo na mesma faixa de preço (como Arduino). Para as principais diferenças entre os diposítivos, tendo como base o Datasheet dos fabricantes [8] e [9] temos:

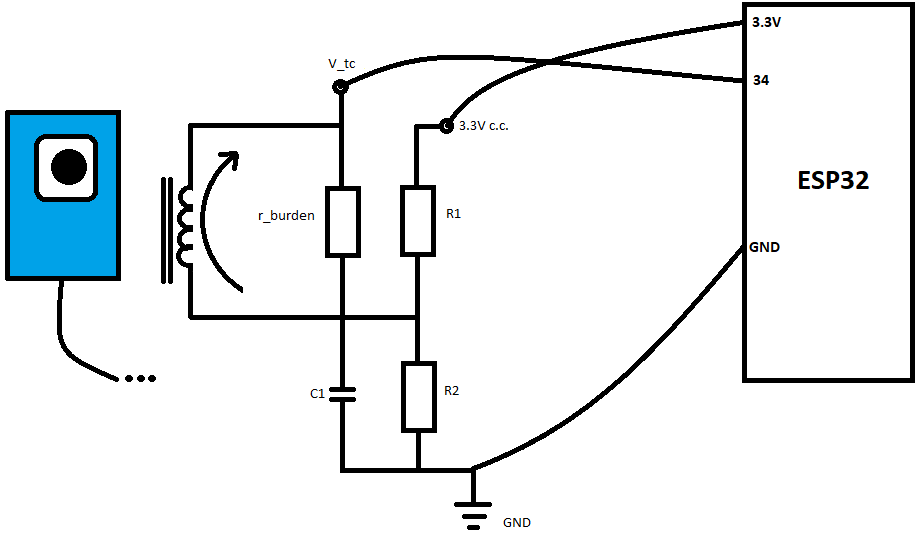
|  |  |  |
| --- | --- | --- |
| Descrição | Arduino UNO | ESP32 |
| Pinos de I/O | 23 pinos com 6PWM | 34 pinos com 16 PWM |
| Memória RAM/SRAM | 2 kB | 520 kB |
| Temporizadores | 3 Timers de 16-bits e dois de 8bits | 4 timres de 64-bits |
| Memória Flash | 32 kB | 4 MB |
| Memória ROM/EEPROM | 4 kB | 448 kB |
| Wifi | Não possui | Possui |
| Frequência de Operaçao | 0 – 16 MHz | 80 – 240 MHz |

Unidades do Arduino UNO podem ser encontradas no Brasil por cerca de R$70,00 cada. Já o ESP32, está disponível por cerca de R$40,00 cada.

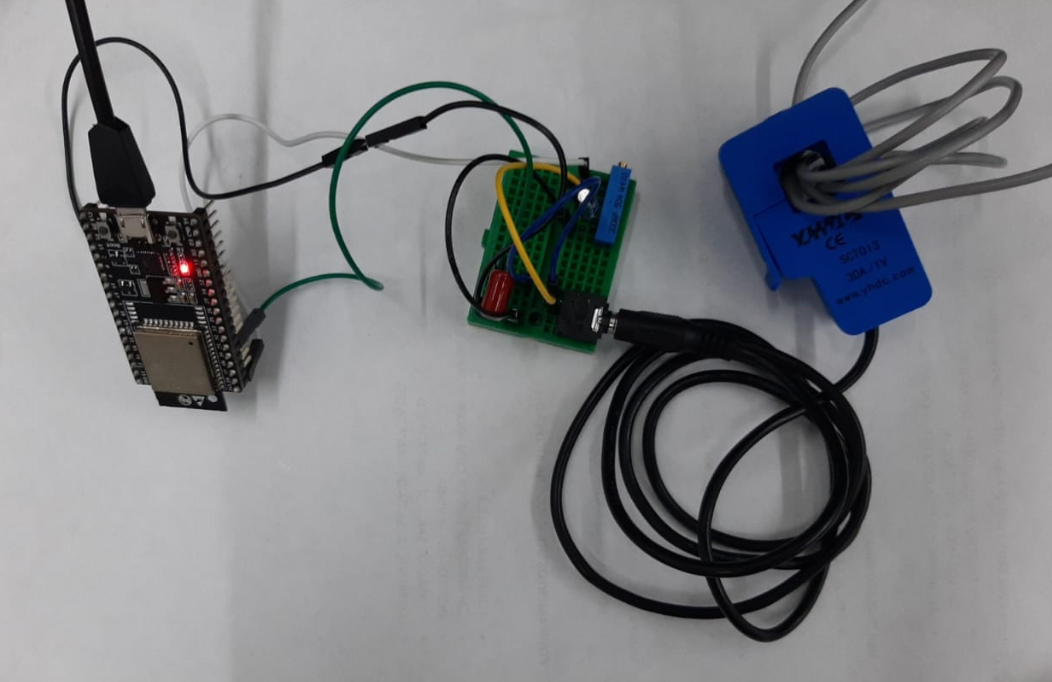
. Uma foto do dispositivo utilizado, montado em uma placa de desenvolvimento ESP32 DEVKIT V4 [9] se encontra na Figura 6..

  
Figura 6: ESP32 DEVKIT V4

Juntando as três partes isoladas num diagrama, temos:



Reproduzindo, temos:

  
Figura 7: Bancada de testes

**4.2 Produção de um código para o ESP32 que faça a leitura da corrente que passa pelo TC.**

**4.2.1 Idealização do Código**

Começando o desenvolvimento do código, o desafio se dá em identificar o quão rápido pode ser a frequência de amostragem, uma vez que o código seja escrito da forma mais otimizada possível.

O ESP32, consegue receber códigos de *Micro-Python* ou *C/C++,* como os demais microcontroladores do mercado. Utilizando Micro-Python, foi verificado que esse era muito inconstante e pouco confiável para tempos exatos de amostragem. Um simples código como o seguinte, gerava valores muitos distintos para cada valor da variável TEMPO\_DA\_ITERACAO, variando de [200,500] us. Isso ocorre, por falta de otimização na conversão de Micro-Python para C.

***VALORES\_LIDOS = [];***

***TEMPO\_PERCORRIDO = 0;***

***TEMPO\_DA\_ITERACAO;***

***PARA I de 0 ATÉ 255 FAÇA PASSO 1***

***VALORES\_LIDOS[I] = LER\_PORTA\_34();***

***TEMPO\_DA\_ITERACAO = TEMPO\_ATUAL – TEMPO\_PERCORRIDO;***

***ESCREVER(TEMPO\_DA\_ITERACAO);***

***TEMPO\_PERCORRIDO = TEMPO\_ATUAL;***

***FIM PARA;***

O mesmo código anterior escrito em C, ainda que gerando grandes variações, mostrou-se mais eficiente, sempre entregando valores dentro do intervalo de [50, 200] us. Para a identificação das cargas, os intervalos entre amostras devem quase que constantes, para que esses não interfiram no resultado. Dessa forma, foi necessário a utilização de outra ferramenta para aquisição dos dados, além de uma simples iteração usando FOR.

Para isso, foi utilizado um timer/cronômetro do próprio ESP32, o uso de Interrupção e Região Crítica do código. O timer é responsável simplesmente por armazenar o tempo decorrido desde que o dispositivo foi ligado. A interrupção faz como que o processador pause a sua tarefa atual, e a atenda o dispositivo que chamou a interrupção. Já a Região Crítica de um código, garante que apenas aquele trecho de código, seja executada naquele instante, tornando a operação sobre recurso compartilhado atômica.

Então, esse código funcionaria da seguinte forma: O ESP32 é ligado, o timer é disparado, e uma função de Interrupção é associada ao timer, para que a cada 100 usus essa função seja chamada. Quando essa é chamada, o código entra numa Região Crítica, e garante que apenas a leitura da porta digital esteja sendo executada naquele instante, otimizando o processo de leitura. Depois, que alguns valores fossem armazenados, uma função para o cálculo RMS dos valores obtidos seria chamada, e o timer reiniciado para recomeçar o processo. Além disso, o último valor RMS calculado fica disponível num servidor web, hospedado no próprio ESP32, e que pode ser acessado por protocolo HTTP.

**4.2.2 Produção do código**

A fim de otimizar e permitir o bom funcionamento do código, os seguintes artifícios foram usados:

Definir constantes com ***#define***, para reservar um espaço em memória que não precisa ser acessado.

***#define ADC\_SAMPLES 256***

***#define I\_RMS\_VEC\_SIZE 512***

Declaração dos timers que serão utilizados. O timer0 é responsável por pegar as amostras a cada 100 us e armazená-las num vetor de 256 espaços. O timer1 é responsável por resetar o timer0, a cada 500 ms.

***hw\_timer\_t\* timer0 = NULL;***

***hw\_timer\_t\* timer1 = NULL;***

Declaração das variáveis com o modificador ***volatile,*** para evitar otimizações indesejadas do compilador. E uso de **DRAM\_ATTR** para garantir que a variável vá para a RAM que tem acesso mais rápido que a Flash. Declaração de um semáforo que identifica se no momento está ocorrendo um evento de interrupção. Uso de **uint32\_t**, para garantir 32 bits de memória para a variável.

***volatile SemaphoreHandle\_t timerSemaphore;***

***volatile DRAM\_ATTR float i\_rms;***

***volatile DRAM\_ATTR float i\_rms\_data[I\_RMS\_VEC\_SIZE];***

***volatile DRAM\_ATTR uint32\_t i\_rms\_data\_idx = 0;***

***volatile DRAM\_ATTR uint32\_t isrCounter = 0;***

***volatile DRAM\_ATTR uint32\_t adc\_data[ADC\_SAMPLES];***

***volatile DRAM\_ATTR uint32\_t sampletime\_us[ADC\_SAMPLES];***

***volatile uint32\_t buffer\_adc\_data[ADC\_SAMPLES];***

***volatile uint32\_t buffer\_sampletime\_us[ADC\_SAMPLES];***

Associando a função ***onTimer()*** ao **timer0** que é chamada a cada 100 us. Associando a função ***funcaoTimer1()***, ao **timer1**, que é chamado a cada 500 ms.

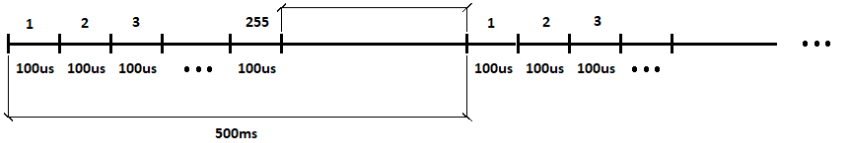
***timerAttachInterrupt(timer0, &onTimer, true);***

***timerAlarmWrite(timer0, 100, true);***

***timerAttachInterrupt(timer1, &funcaoTimer1, true);***

***timerAlarmWrite(timer1, 500000, true);***

O esquema fica como demonstrado a seguir:



Função ***onTimer()***, que é chamada pela Interrupção, e entra em uma Região Crítica para fazer a leitura da porta 34 do controlador.

**void IRAM\_ATTR onTimer(){**

**portENTER\_CRITICAL\_ISR(&timerMux);**

**adc\_data[isrCounter] = analogRead(34);**

**sampletime\_us[isrCounter]=micros();**

**isrCounter++;**

**portEXIT\_CRITICAL\_ISR(&timerMux);**

**xSemaphoreGiveFromISR(timerSemaphore, NULL);**

**}**

Função ***funcaoTimer1()***, que é chamada pela Interrupção, a cada 500 ms para resetar o **timer0**.

***void IRAM\_ATTR funcaoTimer1(){***

***timerRestart(timer0);***

***}***

Função ***calcularRms()***, que é chamada após o reset do **timer0**. Ela recebe o vetor de 255 valores coletados pela função ***onTimer()***, e faz um *padding* no início e fim do vetor, a fim de deixar apenas 167 amostras (um ciclo completo 100u \* 167 = 1/60), para o cálculo do valor RMS.

***float calcularRms(volatile uint32\_t arr[]) {***

***float square = 0;***

***float aux;***

***for (int i = 44; i <= 210 ;i++) {***

***aux = ((arr[i]/1845.0) - 1.0) \* 6.0 \* 1.637;***

***square += aux\*aux;***

***}***

***return sqrt(square/167);***

***}***

Função que retorna o último valor RMS calculado, ao acessar <https://IP.DO.DISPOSITIIVO/i_rms>.

***server.on("/i\_rms", handle\_OnConnect\_i\_rms);***

***void handle\_OnConnect\_i\_rms\_data() {***

***String ptr = "";***

***for (int i=0;i<I\_RMS\_VEC\_SIZE;i++)***

***{***

***ptr += String(i\_rms\_data[i],3);***

***if (i<(I\_RMS\_VEC\_SIZE-1))***

***ptr += ",";***

***}***

***ptr += String(" ") + String(i\_rms\_data\_idx);***

***server.send(200, "text/plain", ptr);***

***}***

MOSTRAR E EXPLICAR A EQUAÇÃO DE CONVERSÃO DO VALOR EM CONTAGENS DO ADC PARA AMPERES.

**5. PRODUÇÃO TÉCNICO-CIENTÍFICA**

Como citado na segunda etapa da parte de Metodologia desse documento, está sendo redigido um Artigo Científico, a partir de pesquisas feitas em base de dados como IEEE Xplore [10]. Esse artigo tem como intuito testar o funcionamento do monitorador de carga que está sendo produzido, e a partir do resultados encontrados, sugerir futuras aplicações da técnica.

**6. CONCLUSÕES**

**7. REFERÊNCIAS BIBLIOGRÁFICAS**

[6] <https://www.poweruc.pl/blogs/news/non-invasive-sensor-yhdc-sct013-000-ct-used-with-arduino-sct-013>

[7] <https://learn.openenergymonitor.org/electricity-monitoring/ct-sensors/interface-with-arduino>

[8] <https://www.espressif.com/en/products/hardware/esp32/overview>

[9] <https://www.farnell.com/datasheets/1682209.pdf>

[10] <https://ieeexplore.ieee.org/Xplore/home.jsp>

**8. AUTO-AVALIAÇÃO DO ALUNO**

FAÇA UMA DISSERTAÇÃO AUTO-REFLEXIVA SOBRE O QUE APRENDEU E O QUE EXPERIENCIOU SOBRE O TEMA E SOBRE O PROCESSO DE PESQUISA.