Centro de Engenharia Elétrica e Informática

Curso de Graduação em Engenharia Elétrica

PLATENY DE BRITO PONCHET

AVALIAÇÃO DO TRANSFORMADOR DE CORRENTE SCT-013 APLICADO EM MEDIDOR ELETRÔNICO DE POTÊNCIA

Campina Grande, Paraíba
Setembro de 2016

PLATENY DE BRITO PONCHET

AVALIAÇÃO DO TRANSFORMADOR DE CORRENTE SCT-013 APLICADO EM MEDIDOR ELETRÔNICO DE POTÊNCIA

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Eletrônica

Orientador:
Prof. Francisco das Chagas Fernandes Guerra

Campina Grande, Paraíba
Setembro de 2016

PLATENY DE BRITO PONCHET

AVALIAÇÃO DO TRANSFORMADOR DE CORRENTE SCT-013 APLICADO EM MEDIDOR ELETRÔNICO DE POTÊNCIA

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Eletrônica

Aprovado em / /

Professor Avaliador
Universidade Federal de Campina Grande
Avaliador

Prof. Francisco das Chagas Fernandes Guerra
Universidade Federal de Campina Grande
Orientador, UFCG

Dedico este trabalho aos meus pais, Severina Mouzinho de Brito Souza e Cláudio Ponchet de Souza, que sempre lutaram para que eu tivesse um futuro digno.

AGRADECIMENTOS

Agradeço a Deus, em primeiro lugar, pela minha vida e pelos dons que me foram confiados, sem os quais não teria tido êxito nesta caminhada.

Agradeço também aos meus pais, Severina e Cláudio, por sempre terem se dedicado, se esforçado e, apesar de todas as dificuldades, sempre me proporcionaram a melhor educação e liberdade para me dedicar somente a meus estudos.

Agradeço também a minha família e amigos, todos aqueles que me ajudaram de alguma forma a vencer todos os obstáculos que enfrentei na vida. E em especial, agradeço a minha namorada, Anna Prycilla, por toda paciência e apoio, a Tchai Oliveira por todas as palavras de incentivo e força, e a Nelson por toda ajuda.

Agradeço ao Professor Tarso Vilela pela contribuição na formatação deste trabalho.

Agradeço também ao Professor Chagas, por ter aceitado me orientar neste trabalho de conclusão de curso, ter me dado o apoio, sanado minhas dúvidas e disponibilizado o laboratório e os equipamentos para meus experimentos.

“Não perca mais tempo com o que não faz a vida valer a pena.”

Mensagem Brasil.

RESUMO

No mundo atual, o uso excessivo da energia elétrica leva a uma situação de esgotamento das fontes de energia e motiva a busca por alternativas eficientes tanto na questão do consumo quanto na questão ecológica. Nesta busca são necessários equipamentos cuja função é voltada a avaliação da eficiência. Abordando o caso do consumo residencial, medidores de energia convencionais dão lugar a medidores modernos com inclusão de elementos de eletrônica digital e de comunicação, os chamados *Smart Meters*, os quais guardam informações detalhadas do consumo e da potência. Este trabalho foi voltado a avaliação do transformador de corrente SCT-013 da *Dechang Electronic* como sensor de corrente de um *Smart Meter*, executando experimentos com o transformador trabalhando isoladamente e implementando um *Smart Meter* para teste do transformador aplicado em um sistema digital.

Palavras-chave: Sistema Embarcado, *Smart Meter*, Transformador de Corrente.

ABSTRACT

In a modern world powered by electricity, overuse leads to a shortage of energy sources and motivates the search for efficient. In this search, equipments whose function is to evaluate the efficiency are needed. In the residential case, conventional power meters, which mostly measure only consumption to generate the famous account of electricity, give rise to modern meters with inclusion of digital electronics and communication elements, called *Smart Meters*, which keep detailed information of consumption and power. This work was aimed at evaluating the SCT-013 current transformer of Dechang Electronic as a current sensor of a Smart Meter, running experiments with the transformer working alone and implementing a Smart Meter for transformer test applied to a digital system.

Keywords: Embedded System, Smart Meter, Current Transformer.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADC – *Analog-Digital Converter* – Conversor analógico-digital

f - Frequência

FFT – *Fast Fourier Transform* – Transformada Rápida de Fourier

fp – Fator de potência

FS – *File System* – Sistema de Arquivos

I^2C – *Inter Integrated Circuit*

IDE – *Integrated Development Environment* – Ambiente de Desenvolvimento Integrado

LSB – *Least Significant Bit – Bit* Menos Significativo

MSB – *Most Significant Bit – Bit* Mais Significativo

P – Potência Ativa

PMIC – Power Management Integrated Circuit – Circuito Integrado Medidor de Potência

Q – Potência Reativa

RMS – Root Mean Square – Raiz do valor quadrático médio

RTC – Real Time Clock – Relógio de Tempo Real

SD – Secure Digital Card - Cartão SD

SPI – Serial Peripheral Interface Bus

SS – *Slave Select* – Referente a comunicação *SPI*

TC – Transformador de Corrente

UART – *Universal asynchronous receiver/transmitter*

SUMÁRIO

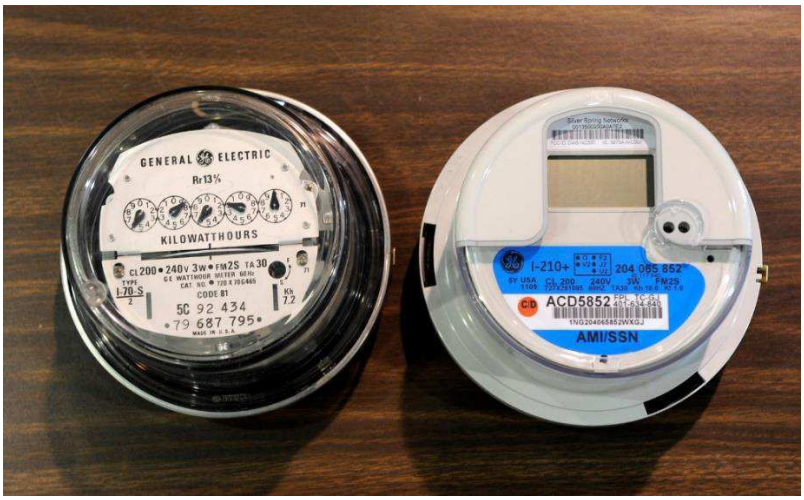
1 INTRODUÇÃO

O mundo moderno é movido pelo uso intensivo de energia, sendo esta oriunda de diversas fontes: eletricidade, gasolina, álcool, gás natural, etc. O usuário, seja ele residencial ou industrial, utiliza aparelhos que convertem estas formas de energia (elétrica, química, etc.) em outras, tais como: mecânica, térmica, luminosa, sonora, etc. Neste processo de conversão há sempre uma parcela de energia que é perdida para o meio ambiente. Lâmpadas incandescentes, por exemplo, transformam energia elétrica em luz e calor (quem já teve a experiência de tocar em uma lâmpada incandescente após alguns minutos de uso sabe disto - a temperatura da superfície é muito alta). O caso da lâmpada incandescente é um dos mais notáveis, pois a grande maioria da energia é transformada em calor (cerca de 92%), e apenas uma pequena parcela é transformada em luz, que é o objetivo da lâmpada. Neste contexto já podemos falar no termo eficiência energética sendo a relação entre a quantidade energia empregada em uma atividade e aquela disponibilizada para sua realização.

Na busca pela sustentabilidade (utilizar recursos para satisfazer as necessidades presentes sem comprometer as gerações futuras), a eficiência energética tomou também outro significado nos dias de hoje: a forma como se utiliza a energia. Computador ligado sem estar em uso, lâmpadas, ventiladores, condicionadores de ar ligados em locais onde não há pessoas, entre outras situações, também são denominadas causas de ineficiência energética.

Para entender melhor estas situações são necessários equipamentos para avaliação da energia consumida. Medidores de energia que antes eram utilizados pelo usuário final, em sua grande maioria, apenas para determinar qual o preço que ele pagaria pelo serviço, aos poucos vão dando lugar para os chamados *Smart Meters*, medidores que além de medir o consumo também fornecem outras características que ajudam na avaliação da eficiência energética do local. Algoritmos computacionais, com os dados enviados pelos *Smart Meters*, já conseguem identificar equipamentos mal utilizados e ineficientes.

Figura 1 - Medidor Antigo e Smart Meter



Fonte: Site PalmBeachPost[1]

A responsabilidade dos engenheiros eletricistas neste ambiente é implementar os *Smart Meters*, utilizando tecnologias atuais tais como: processadores, chips digitais dedicados, conversores analógico-digital, sensores para coletar as grandezas elétricas necessárias, redes de comunicação sem fio, etc. Na busca pela melhor solução, este trabalho de conclusão de curso tratou da análise do transformador de corrente SCT-013 como um sensor de corrente elétrica não invasivo e da aplicação dele em um medidor de potência.

Figura 2 - TC SCT-013-000

Fonte: Site Filipeflop[2].

[1] Disponível em: https://cmgpbpopinionzone.files.wordpress.com/2014/09/photo-fpl-smart-meter-analog.jpg. Acesso em Setembro de 2016.
[2] Disponível em: http://s3.amazonaws.com/img.iluria.com/product/2025CD/618515/450xN.jpg. Acesso em Setembro de 2016.

2 OBJETIVOS

2.1 OBJETIVO GERAL

Avaliação das características do TC SCT-013 como sensor de corrente isolado e implementação de um *Smart Meter* que o use para avaliação de uma aplicação.

2.2 OBJETIVOS ESPECÍFICOS

- Avaliação do TC:
  - Atenuação e Linearidade
  - Consequência do aumento da carga aplicada
  - Defasagem entre entrada e saída
- *Smart Meter*
  - Projeto e Produção
  - Coleta de medições e validação

3 FUNDAMENTAÇÃO TEÓRICA

3.1 POTÊNCIA EM SISTEMAS NÃO-SENOIDAIS

Na grande maioria das situações de fornecimento de energia elétrica para realização de trabalho, ela é dada sob a forma de tensões e correntes alternadas, variando no tempo de forma periódica. Em relação a tensão, fornecida como sinal senoidal, interferências causam distorções e erros ao aplicar modelos senoidais. Em relação a corrente a situação é mais agravante, visto que com o advento de equipamentos eletrônicos, por exemplo, o sinal de corrente passou a ser bem mais alterado e divergente do senoidal. A gama de problemas associados é enorme: determinação de potência nominal de um equipamento para seu funcionamento, determinação do consumo de energia, projeto de transformadores, projeto de geradores, instalações elétricas, etc. Este tópico apresenta alguns conceitos referentes a potência em sistemas com sinais senoidais distorcidos.

Valor Eficaz

O valor eficaz de uma função periódica é definido como a raiz quadrada do valor médio da função ao quadrado (**NILSSON; RIEDEL, 2009**), ou seja:

$$f_{rms} = \sqrt{\frac{1}{T}\int_{t_0}^{t_0+T} f^2(t)dt} \qquad (1)$$

Valor RMS é o valor correspondente contínuo do sinal alternado que dissiparia a mesma potência aplicado a uma determinada carga.

No caso discreto, temos:

$$f_{rms} = \sqrt{\frac{1}{N}\sum_N f_k^2} \qquad (2)$$

Potência Instantânea

A potência instantânea em algum ponto de um circuito, em qualquer instante de tempo *t,* é dada pela multiplicação entre a tensão *v* e a corrente *i* neste instante de tempo, ou seja:

$$p = vi \qquad (3)$$

Dado o sentido convencional da corrente, se ela estiver no sentido da elevação de tensão, a equação acima deve apresentar um sinal negativo. A unidade da potência é expressa em watts (W) (*Volt* (V) vezes *Ampere* (A)).

Potência Média

Como foi dito anteriormente, a tensão e a corrente nos sistemas considerados neste trabalho variam no tempo e podem ser escritas como:

$$v = v(t)$$
$$i = i(t)$$

Sendo assim, a potência instantânea também varia no tempo e é dada por:

$$p(t) = v(t) * i(t) \qquad (4)$$

Como o sinal é periódico, a potência média nada mais é do que "a média das potências instantâneas durante um período ou, em forma de equação,

$$P = \frac{1}{T}\int_{t_0}^{t_0+T} p(t)dt \qquad (5)$$

Onde T é o período da função." (**NILSSON, RIEDEL, 2009**)

No caso discreto, toma-se uma média das amostras:

$$P = \frac{1}{N}\sum_N p_k = \frac{1}{N}\sum_N v_k i_k \qquad (6)$$

Esta potência também é chamada muitas vezes de *potência ativa* por ser a potência que é convertida em trabalho (convertida para outra forma de energia).

Potência Aparente

A potência aparente é definida como a multiplicação entre o valor *RMS* da tensão e o valor *RMS* da corrente:

$$|S| = V_{rms} * I_{rms} \qquad (7)$$

A unidade da potência aparente é VA (Volt-ampère).

Fator de Potência

A definição primordial de fator de potência é a seguinte:

$$fp = \frac{P}{|S|} \qquad (8)$$

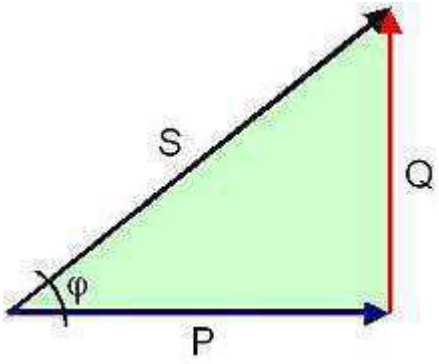
No caso senoidal, esta definição resume-se a:

$$fp = \cos(\theta_v - \theta i) \qquad (9)$$

Potência Reativa

Tendo conhecimento da potência ativa e da potência aparente, a potência reativa (potência relacionada a energia elétrica que não é convertida em outra forma de energia, periodicamente permutada entre a carga e a fonte) pode ser obtida através do triângulo de potências, mostrado na Figura 3.

Figura 3 - Triângulo de Potências



Fonte: Site Wikipedia.[3]

$$Q = \sqrt{S^2 - P^2} \qquad (10)$$

Onde,

S é $a\ potência\ aparente$

P é $a\ potência\ ativa$

Q é $a\ potência\ reativa$

$cos\phi$ é $o\ fator\ de\ potência$

Esta é uma forma de calcular considerando todas as distorções presentes nos sinais. Nesse contexto, esta componente de energia é denominada de potência não-ativa, representada por N, segundo especificado na Norma IEEE 1459-2010. (**ANEEL**)

Outra forma de calcular a potência reativa é pelo método de deslocamento no tempo, descrito a seguir.

Considerando inicialmente um caso senoidal, onde a tensão e a corrente variam de forma senoidal, a função potência instantânea em função do tempo pode ser expressa como: (**NILSSON, RIEDEL, 2009**)

[3] Disponível em: https://upload.wikimedia.org/wikipedia/commons/thumb/3/32/TrianguloDePotencia2.jpg/220px-TrianguloDePotencia2.jpg. Acesso em Setembro de 2016.

$$p(t) = \frac{V_m I_m}{2}\cos(\theta_v - \theta_i) + \frac{V_m I_m}{2}\cos(\theta_v - \theta_i)\, cos2\omega t - V_m I_m sen(\theta_v - \theta_i) sen(2\omega t)$$

(11)

Onde:

$V_m e\, I_m$ são a tensão e a corrente de pico, respectivamente
θ_v e θ_i são a fase da tensão e da corrente, respectivamente
ω é a frequência fundamental do sistema, em rad/s

Pode-se observar que p(t) pode ser escrito como:

$$p(t) = P + Pcos2\omega t - Qsen2\omega t \quad (12)$$

Onde P é outra definição para a potência média ou ativa, que foi apresentada anteriormente, e Q é chamada de potência reativa.

É fácil observar uma relação entre a potência ativa e a potência reativa:

$$Q(\theta_v, \theta_i) = \frac{V_m I_m}{2} sen(\theta_v - \theta_i) \qquad (13)$$

$$P(\theta_v, \theta_i) = \frac{V_m I_m}{2} cos(\theta_v - \theta_i) \qquad (14)$$

Como $sen(x) = \cos(x - \frac{\pi}{2})$, temos:

$$Q(\theta_v, \theta_i) = \frac{V_m I_m}{2} cos\left(\theta_v - \theta_i - \frac{\pi}{2}\right) = P\left(\theta_v - \frac{\pi}{2}, \theta_i\right) \qquad (15)$$

Manipulando as equações 4,5 e 15, temos que:

$$p(t) = v(t) * i(t) \quad (4)$$

$$p(\theta) = v(\theta) * i(\theta) \quad (16)$$

$$P = \frac{1}{T}\int_{t_0}^{t_0+T} p(t)dt = \frac{1}{2\pi}\int_0^{2\pi} p(\theta)d\theta = \frac{1}{2\pi}\int_0^{2\pi} v(\theta)i(\theta)d\theta \qquad (17)$$

$$Q = P\left(\theta_v - \frac{\pi}{2}, \theta_i\right) = \frac{1}{2\pi}\int_0^{2\pi} v\left(\theta - \frac{\pi}{2}\right) i(\theta)d\theta \qquad (18)$$

No caso discreto:

$$Q = \frac{1}{N}\sum_N v'_k i_k \qquad (19)$$

Onde, *v'* é o sinal de tensão atrasado em 90°.

A unidade da potência reativa é var (volt-ampère-reativo).

3.2 HARMÔNICOS EM SISTEMAS ELÉTRICOS

"A distorção de tensão e corrente é analisada matematicamente através dos estudos das ondas não senoidais periódicas. Nestas condições, sabe-se que qualquer onda que possua em seu conteúdo distorções ou frequências com amplitude diferente da fundamental, pode ser decomposta de acordo com a série de *Fourier*, em uma componente de mesma frequência que a da onda resultante distorcida, que é chamada de *onda fundamental*, e em outras ondas senoidais de frequências múltiplas da fundamental, que, como em acústica, receberam a denominação de *harmônicas*. A ferramenta matemática utilizada no cálculo desses índices de amplitude e ângulo das harmônicas é denominada de FFT (*Fast Fourier Transformer*) ou Transformada Rápida de *Fourier*." (**GARCIA**)

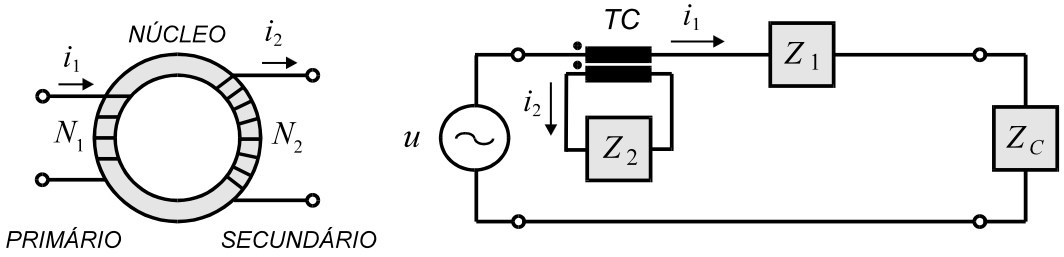
Ou seja, um sinal de tensão ou corrente com distorções geradas por equipamentos não lineares e interferências pode ser decomposto em uma soma de outros sinais. A componente que tem a mesma frequência fundamental é chamado de componente fundamental e as componentes de frequências múltiplas da fundamental são chamadas de harmônicas. A partir delas, obtidas pela ferramenta FFT, é possível tirar conclusões sobre a distorção do sinal em relação a uma senóide perfeita.

3.3 TRANSFORMADORES DE CORRENTE

"Transformadores de corrente (TCs) destinam-se a alimentar instrumentos de medição, proteção ou controle em sistemas elétricos. Reduzem a corrente do sistema de potência a um valor adequado aos instrumentos, de modo a haver uma relação fixa entre os valores instantâneos correspondentes das ondas de corrente de saída e de entrada, com diferenças de fase mínimas possíveis entre as mesmas. Também promovem isolação elétrica entre os instrumentos e o sistema de potência." (GUERRA)

A sua construção e a forma como ele deve ser ligado é mostrado na Figura 4:

Figura 4 - Transformador de Corrente e Forma de Ligação



Fonte: Notas de Aula do Professor Francisco das Chagas Fernandes Guerra.

Dado o número de espiras do enrolando primário (N_1) e do enrolamento secundário (N_2), considerando um núcleo com permeabilidade magnética muito alta, a relação entre as correntes do primário e do secundário é dada por:

$$\frac{I_1}{I_2} = \frac{N_2}{N_1} \qquad (20)$$

3.4 *POWER MANAGEMENT INTEGRATED CIRCUIT – PMIC*

Os *PMICs* são circuitos integrados cuja funcionalidade é voltada a gerência de potência em um determinado circuito. Em geral, os *PMICs* possuem as seguintes características:

- Vários canais diferencias de ADC;
- Cálculos internos de grandezas como: potências, valores eficazes, frequência, energia, etc.;
- Erro abaixo de 1% para os cálculos internos;
- Baixo consumo de potência (corrente na faixa de mA para uma alimentação de 5 V ou 3.3 V contínua);
- Interface de comunicação digital (variando de acordo com o chip, podendo ser *SPI*, I2C, UART, etc.);
- Tamanho pequeno, feitos em tecnologia SMD.

Em relação aos cálculos, muitos dos *PMICs* já possuem a funcionalidade *True-RMS*, que é o fato de não considerar sempre o caso senoidal, dando mais exatidão as medições. Alguns ainda calculam considerando o caso senoidal e em outros, a forma de cálculo pode ser configurada.

Figura 5 - *PMIC* CS5490

Fonte: Site Octopart.[4]

[4] Disponível em: http://sigma.octopart.com/41460041/image/Cirrus-Logic-CS5490-ISZ.jpg. Acesso em Setembro de 2016.

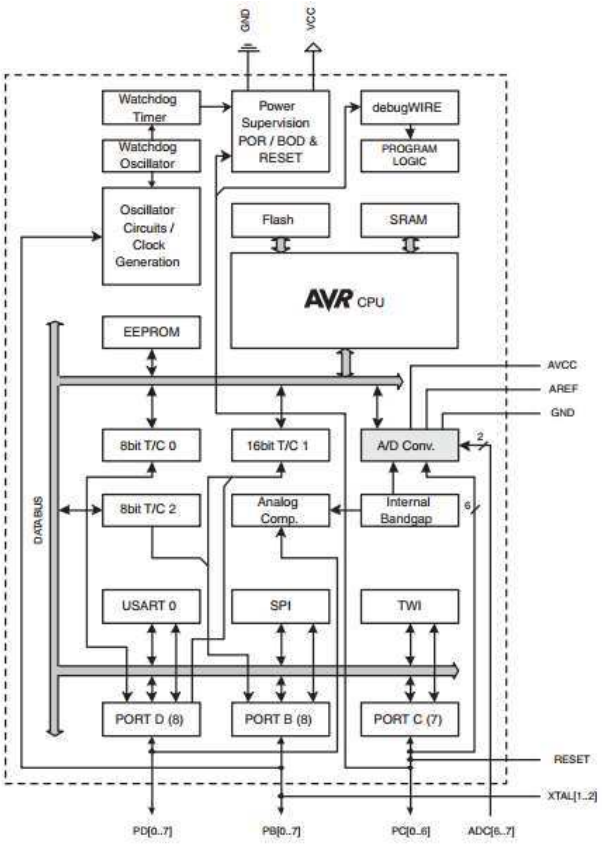
3.5 MICROCONTROLADORES

Microcontrolador é um sistema eletrônico digital independente (SoC, *System on Chip*), contendo um processador, memória e periféricos. Eles são projetados de forma a, na maioria dos projetos de sistemas embarcados, para coloca-los em funcionamento basta adicionar um *software*.

Possuem memória volátil de alta velocidade de acesso (memória *RAM*, os valores são perdidos depois de retirada da fonte) e memória não volátil para armazenamento de programas (memória *Flash* ou *EEPROM* nos microcontroladores mais atuais, onde os dados continuam gravados mesmo depois de retirada da fonte).

Na Figura 6 encontra-se um diagrama de blocos de um microcontrolador, o ATmega328 da empresa *Atmel*.

Figura 6 - Diagrama de Blocos do ATmega328



Fonte: *Datasheet* do ATmega328.

No diagrama de blocos acima é possível visualizar os componentes de um microcontrolador: interfaces de comunicação (*USART*, *SPI* e *TWI*), comparadores analógicos, *ADC*, memória *EEPROM*, circuito gerador de *clock*, memória *flash*, memória *SRAM*, *CPU*, etc.

É comum encontrar no mercado as chamadas *Dev Boards* (placas de desenvolvimento), sistemas embarcados formados por microcontrolador, circuito externo necessário ao funcionamento do microcontrolador, interface *USB* para fácil comunicação e programação, *LEDs*, entradas de áudio, saídas de vídeo, placas de comunicação *wireless*, barramento de pinos macho e fêmea, chaves, etc. Isso facilita e acelera mais os projetos, deixando o microcontrolador, de fato, independente.

Figura 7 - ATmega328p-AU

Fonte: Site Instituto Digital.[5]

Figura 8 - Arduino UNO

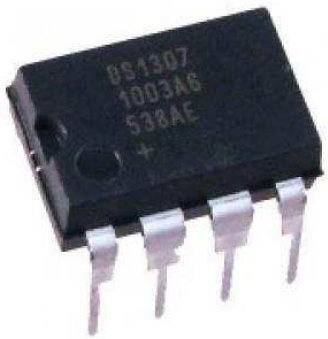
Fonte: Site Comphaus.[6]

[5] Disponível em: http://s3.amazonaws.com/img.iluria.com/product/F462C/238A49/450xN.jpg. Acesso em Setembro de 2016.
[6] Disponível em: http://comphaus.com.br/home/wp-content/uploads/2014/03/Arduino-Uno-R3-Front.jpg. Acesso em Setembro de 2016.

3.6 RELÓGIO DE TEMPO REAL

Relógio de tempo real, do inglês *Real Time Clock*, é um circuito integrado cuja função é manter o controle do tempo atual. O DS1307 por exemplo, da empresa *Maxim Integrated*, possui consumo muito baixo quando operando como escravo de um processador (unidades de mA no máximo) e possui também uma entrada para bateria, fazendo com que ele mantenha o controle do tempo mesmo sem alimentação de uma fonte externa (nesse modo o consumo é menor que 500 nA).

Figura 9 - Relógio de Tempo Real DS1307



Fonte: Site Webtrônico.[7]

3.7 SECURE DIGITAL CARD

Secure Digital Cards, ou comumente chamados de cartão *SD*, são pequenos cartões de memória não-voláteis, utilizados geralmente em sistemas embarcados como

[7] Disponível em: http://www.webtronico.com/image/cache/data/produtos/ds1307-500x500.jpg. Acesso em Setembro de 2016.

espaço de armazenamento extra. Possuem capacidades de criptografia e de direitos autorais. Existem 4 famílias atualmente:

- *SD*;
- *SDSC*: Capacidade padrão, até 2 GB de memória;
- *SDHC*: Capacidade alta, até 32 GB de memória;
- *SDXC*: Capacidade elevada, até 2 TB de armazenamento;

Cartões *SD* possuem os chamados Sistemas de Arquivos, que é uma forma de organização, armazenamento e codificação dos dados em um meio de armazenamento. Sendo assim, o *software* do microcontrolador ou um sistema operacional, sabendo interpretar o sistema de arquivos de um cartão *SD*, pode ler os dados e gravar dados no mesmo.

Existe também o cartão chamado *microSD*, que é uma versão em menor do tamanho dos cartões *SD*.

Figura 10 - Cartão SDHC 4 GB SanDisk

Fonte: Site Sandisk.[8]

Figura 11 - Micro SDHC 32 GB SanDisk

Fonte: Site Sandisk.[9]

[8] Disponível em: https://www.sandisk.com/content/dam/sandisk-main/en_us/portal-assets/product-images/retail-products/Blue_SDHC_Class4_Front_4GB.png. Acesso em Setembro de 2016.
[9] Disponível em: https://www.sandisk.com.br/content/dam/sandisk-main/en_us/portal-assets/product-images/retail-products/microSD_SDHC_Class4_32GB-retina.png.thumb.319.319.png. Acesso em Setembro de 2016.

3.8 COMUNICAÇÃO SERIAL

Comunicação serial é uma transferência de dados binária (palavras formadas apenas por zeros e uns) de forma serial, ou seja, um *bit* por vez. É usada para comunicação do microcontrolador com os periféricos necessários ao funcionamento do sistema embarcado, como por exemplo: *PMIC*, relógio de tempo real, cartão *SD*, etc. Pode ser síncrona (com o auxílio de um sinal de alta frequência chamado *Clock* que varia entre 0 e 1) ou assíncrono (sem a utilização do sinal *Clock*).

A seguir encontram-se brevemente descritas, de forma simples, 3 formas de comunicação serial: a *UART*, a I^2C e a *SPI*.

UART

UART é uma transmissão assíncrona, pois o transmissor não gera nenhum *clock* de sincronia para o receptor. Ao invés disso, o transmissor e o receptor combinam entre si uma velocidade de comunicação anteriormente. Um *bit* chamado *Start Bit* é adicionado no início de cada palavra binária que será enviada para avisar ao receptor que um dado está prestes a ser enviado. No final um *bit* de paridade pode ser acrescentado a palavra, *bit* que diz se a quantidade de *bits* com valor 1 é par ou ímpar, a depender da definição, se é paridade par ou ímpar (não é obrigatório). Ao final, é adicionado um *End Bit* para representar o fim do envio da palavra.

Em geral, em microcontroladores é uma comunicação a dois fios, chamados de RX e TX.

I^2C

I2C é uma forma de comunicação serial multimestre, onde vários periféricos podem ser conectados ao barramento e podem conversar entre si, apenas referenciando o

endereço do destinatário. Possui apenas duas linhas bidirecionais: A *SCL* (*Serial Clock*) e a *SDA* (*Serial Data*), ambos inativos com nível lógico alto.

Ao iniciar uma transmissão, o mestre muda o nível de *SDA* de 1 para 0, avisando aos escravos que uma comunicação será iniciada. Em sequência, o mestre inicia a geração do *clock* em *SCL* e começa a enviar os 7 *bits* de endereço do destinatário pelo *SDA*, e após isso um *bit* de R/W (leitura ou escrita). Ao terminar de enviar os 8 *bits*, o mestre passa o controle de *SDA* ao escravo para que ele mude o nível logico do mesmo de 0 para 1. Caso isso aconteça, o mestre continua enviando normalmente os dados no caso da escrita, ou recebendo normalmente no caso de leitura.

SPI

A *SPI* é uma forma de comunicação serial que permite a comunicação do mestre com vários periféricos. Possui quatro fios:

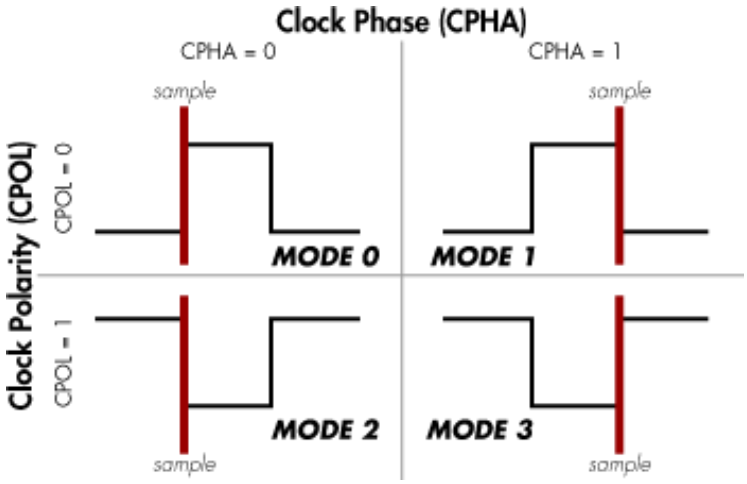
- *MOSI* (*Master Output, Slave Input*): saída de dados do mestre, também conhecido como *SDI* (*Serial Data In*) do lado do escravo;
- *MISO* (*Master Input, Slave Output*): entrada de dados do mestre, também conhecido como *SDO* (*Serial Data Out*) do lado do escravo;
- *SCK* (*Serial Clock*): *Clock* serial;
- *SS* (*Slave Select*): Seleção do escravo.

Nesta forma de comunicação, todos os escravos compartilham os fios *MOSI*, *MISO* e *SCK*, mas cada um possui seu *SS* individual junto ao mestre. Esta é uma desvantagem da comunicação *SPI*, quanto maior o número de escravos, maior será o número de portas digitais utilizadas para representar os sinais *SS*.

Quando o mestre quer iniciar a transmissão, ele muda o nível lógico do *SS* do escravo que ele deseja se comunicar e, a depender do modo de operação *SPI*, ele começa

a enviar ou receber os dados *bit* a *bit* enquanto gera o *clock* de sincronia. Os modos *SPI* são mostrados na Figura 12:

Figura 12 - Modos de Operação *SPI*



Fonte: Site Totalphase.[10]

Cada modo se refere ao momento de aquisição do dado serial (subida ou descida do *clock*) e ao nível inativo do *clock* (0 ou 1).

[10] Disponível em: http://www.totalphase.com/support/article_attachments/200063856/spi-modes.png. Acesso em Setembro de 2016.

4 ATIVIDADES E RESULTADOS

Neste tópico, para cada objetivo específico é apresentada a metodologia, os resultados e conclusões parciais.

4.1 ATENUAÇÃO E LINEARIDADE DO SCT-013

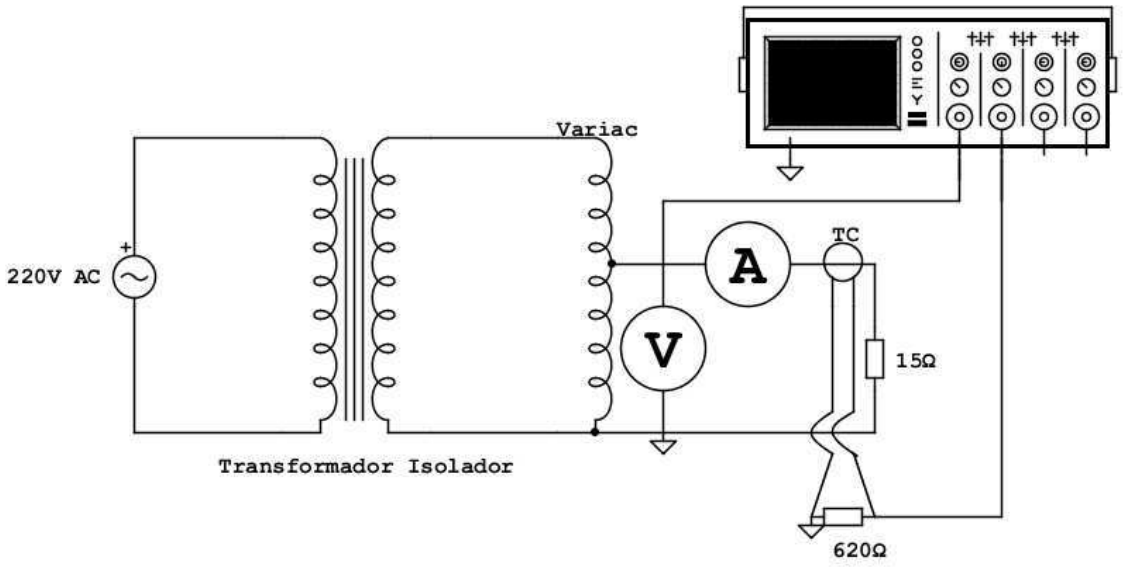
Segundo o *datasheet* do TC SCT-013, sua saída em corrente é linear com a seguinte relação:

$$100\ A{:}\,50\ mA$$

O objetivo deste passo foi avaliar se esta informação é verdadeira após a aplicação de uma carga, observando a atenuação e a linearidade.

Para isso, foi montado em laboratório o circuito é mostrado na Figura 13.

Figura 13 - Bancada de Testes do TC



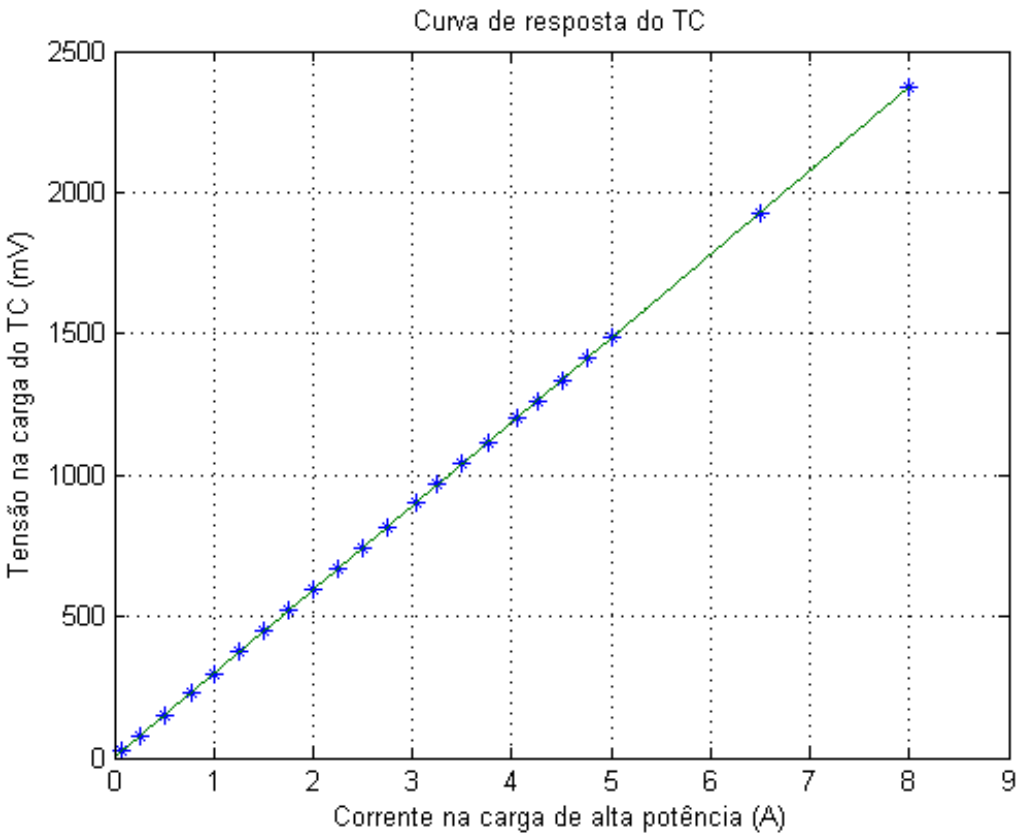
Fonte: Elaborada pelo autor no aplicativo FidoCadJ.

Um *variac*, autotransformador com tensão de saída variável, foi utilizado como fonte de tensão alternada variável de forma a tornar possível a aplicação de correntes alternadas de diversas magnitudes no primário do TC. Para isso, uma carga de alta potência (de resistência 15 Ω) foi aplicada na saída do *variac*. O circuito foi alimentado pela rede elétrica local do laboratório (220 V AC) e isolado por um transformador isolador.

Uma vez montado o circuito na bancada de testes, foram adicionados: multímetro operando como voltímetro para controle da tensão de saída do *variac*; multímetro funcionando como amperímetro para controle da corrente na carga de alta potência; o TC para avaliação, com uma carga de 620 Ω na saída, convertendo a saída em corrente para uma saída em tensão; um osciloscópio digital moderno, o DSO9024H da empresa *Agilent*, para avaliação das formas de onda da saída do *variac* e da saída do TC.

Foram tomados 20 valores de corrente situados entre 0 e 5 A, e dois valores acima deste intervalo (6.5 A e 8 A). A partir disso, foram feitas leituras da corrente na carga de alta potência e da saída em tensão do TC. Cada ponto individual foi plotado em um gráfico e, após verificação da linearidade, uma reta foi aproximada. A ferramenta computacional para auxílio nessa tarefa foi o *Matlab*. O gráfico resultante é mostrado na Figura 14.

Figura 14 - Curva Atenuação e Linearidade do TC



Fonte: Elaborada pelo autor no aplicativo Matlab..

Pela análise do gráfico pode-se observar que a saída do TC, para uma carga de 620 Ω aplicada e para os pontos de corrente de entrada avaliados, é de fato linear. Além disso, ainda com o *Matlab* verificou-se o coeficiente angular da reta, que nada mais é que o ganho do sistema TC + Resistor 620 Ω (entrada em A e saída em mV):

$$G_{620\Omega} = 296.5771$$

Utilizando a *Lei de Ohm*, onde:

$$V = RI \qquad (21)$$

Verifica-se que o Resistor aplica um ganho de 620 na corrente de saída do TC, então o ganho do TC na verdade é (entrada em A e saída em mA):

$$G = \frac{G_{620\Omega}}{620} = 0.4784$$

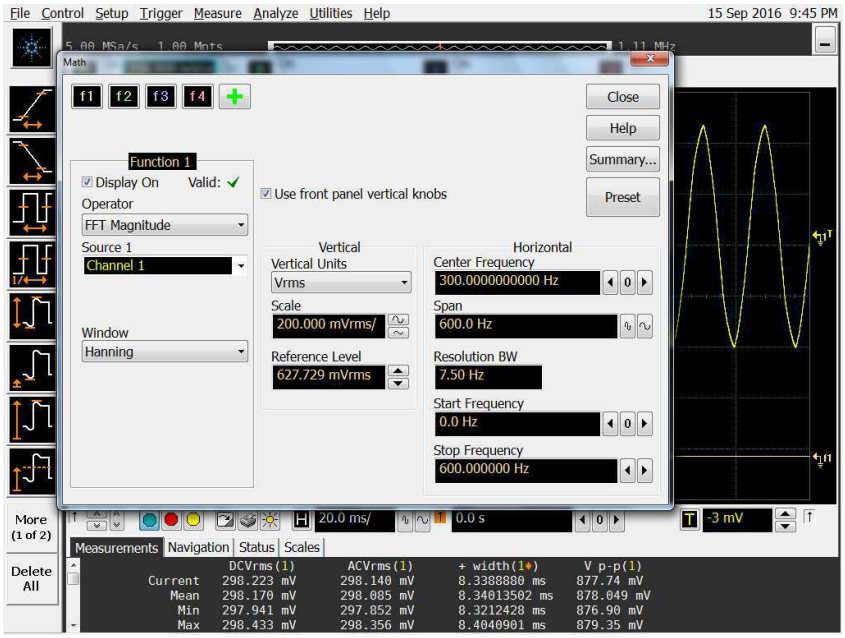
Isso implica que, para uma corrente de 1 A na entrada do TC, a saída é 0.4784 mA. Por consequência, para uma corrente de 100 A, a corrente na saída do TC é de 47.84 mA, verificando a especificação do SCT-013 que é 100 A:50 mA.

4.2 CONSEQUÊNCIA DO AUMENTO DA CARGA APLICADA

Outras cargas foram aplicadas na saída do transformador de corrente para verificar se há alguma alteração na forma de onda de saída. A montagem da bancada continuou a mesma descrita no tópico anterior. Com o auxílio do osciloscópio DSO9024H, a tensão de saída do TC, para uma entrada de corrente de 1 A, foi avaliada tanto no tempo como no domínio da frequência (utilizando a ferramenta *FFT* do osciloscópio).

Na Figura 15, as configurações do osciloscópio DSO9024H:

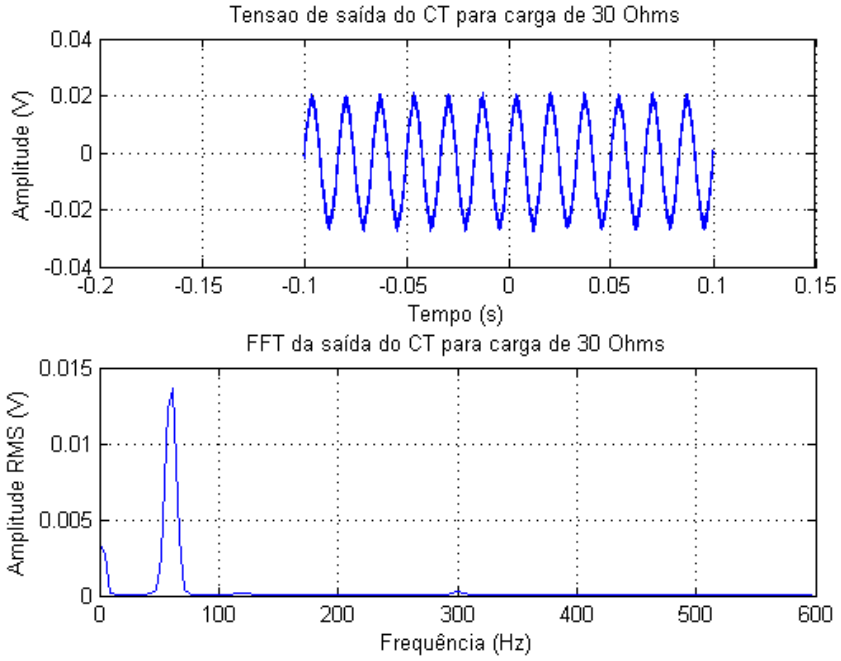
Figura 15 - Configurações FFT do DSO9024H



Fonte: *Printscreen* do aplicativo no Sistema Operacional Windows.

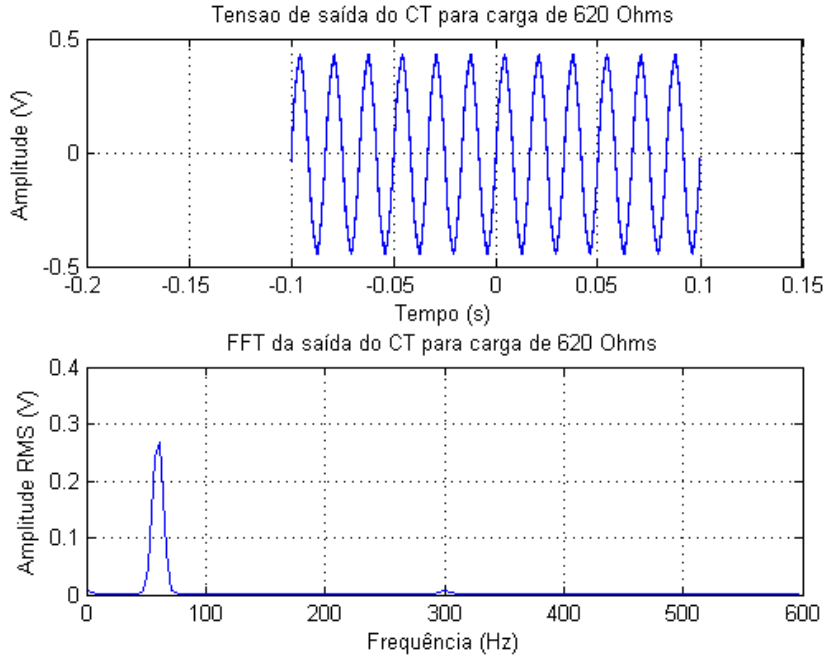
Os dados digitais foram salvos em tabelas (arquivos com extensão .csv) e com auxílio da ferramenta *Matlab*, foram plotados para avaliação.

Figura 16 - Saída do TC para uma carga de 30 Ω



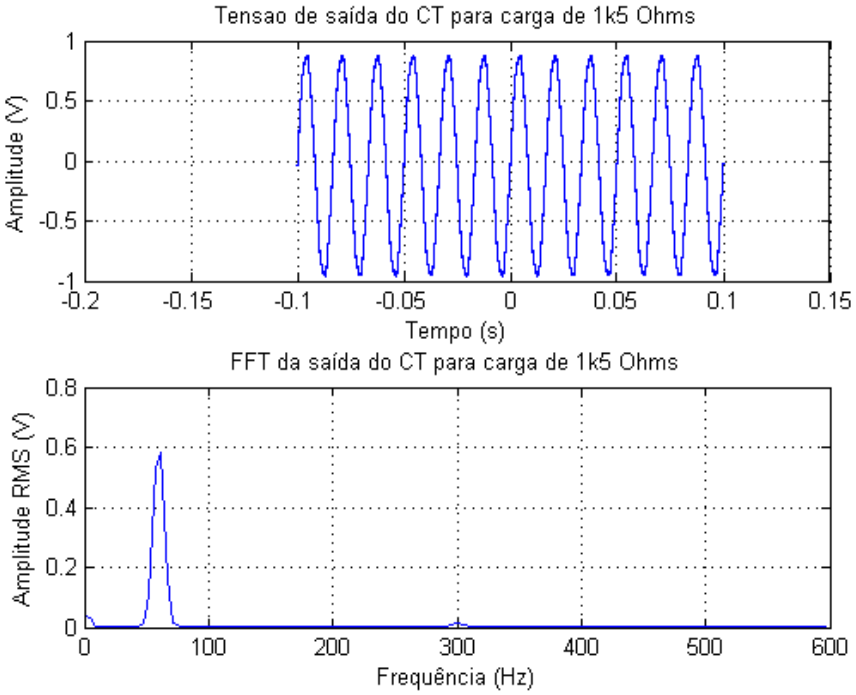
Fonte: Elaborada pelo autor no aplicativo Matlab.

Figura 17 - Saída do TC para uma carga de 620 Ω



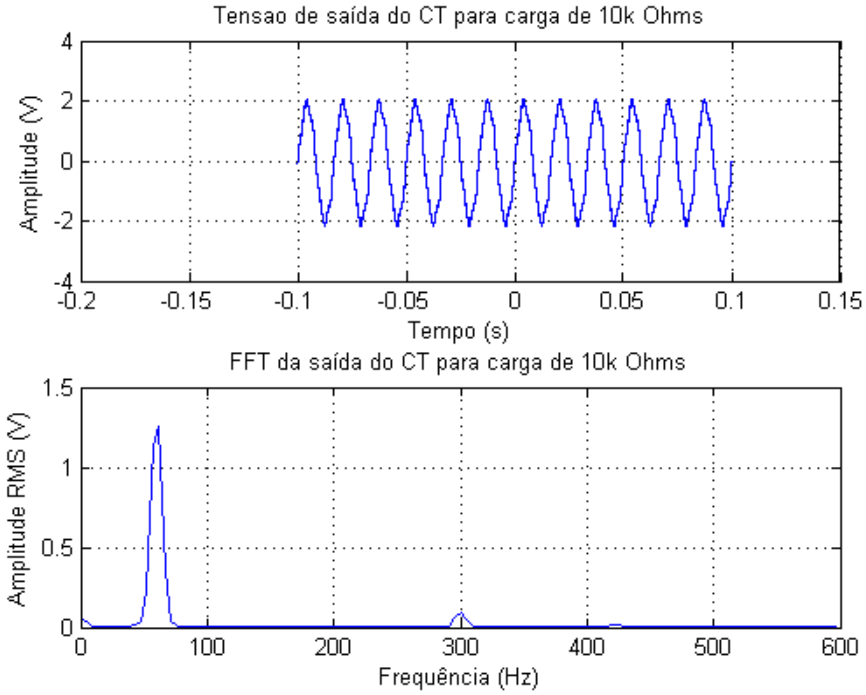
Fonte: Elaborada pelo autor no aplicativo Matlab.

Figura 18 - Saída do TC para uma carga de 1.5 kΩ



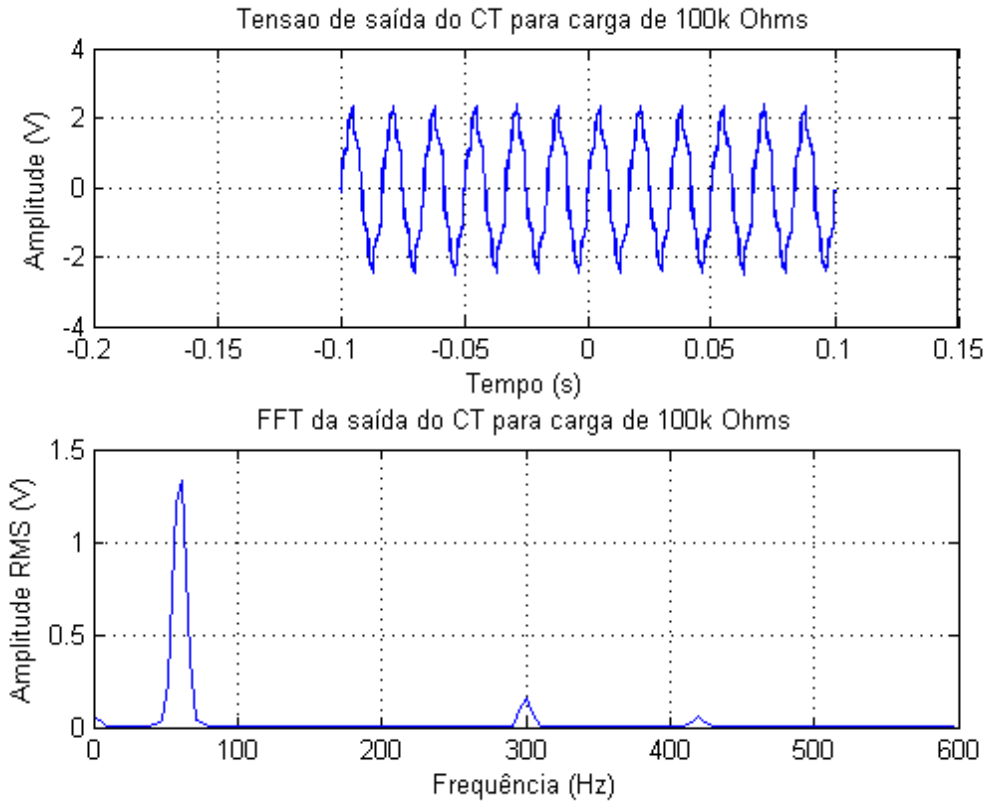
Fonte: Elaborada pelo autor no aplicativo Matlab.

Figura 19 - Saída do TC para uma carga de 10 kΩ



Fonte: Elaborada pelo autor no aplicativo Matlab.

Figura 20 - Saída do TC para uma carga de 100 kΩ



Fonte: Elaborada pelo autor no aplicativo Matlab.

Fazendo inicialmente uma análise no ganho, para uma corrente de 1A aplicada, a corrente na saída é:

$$I_o = 1A * G = 0.4784\ mA$$

Logo, as saídas em tensão para as 5 situações deveriam ser, utilizando a Lei de Ohm, em teoria:

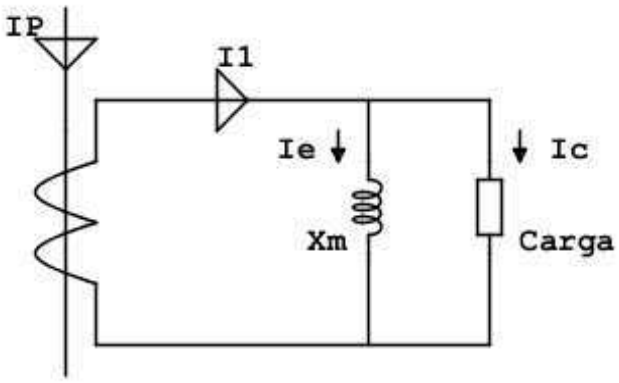
$$V_{30R} = 14.4\ mV$$
$$V_{620R} = 296.6\ mV$$
$$V_{1k5} = 717.6\ mV$$
$$V_{10k} = 4.784\ V$$
$$V_{100k} = 47.84\ V$$

Comparando os valores teóricos com os valores medidos, observa-se que a partir da carga com resistência 1.5 kΩ surge um erro na saída, na carga de 10 kΩ este erro é bem mais evidente e na carga de 100 kΩ o erro se torna ainda mais claro pela distorção do sinal, vista tanto no domínio do tempo como verificada no domínio da frequência (componentes harmônicas em 300 Hz e 420 Hz, quinta e sétima harmônica, respectivamente).

Este erro é dado pelo fato da permeabilidade magnética do núcleo não ser tão grande a ponto de poder ser desprezada para qualquer carga aplicada.

Um modelo para o TC, levando em consideração a permeabilidade magnética finita, é mostrado na Figura 21:

Figura 21 - Modelo do TC



Fonte: Elaborada pelo autor no aplicativo FidoCadJ.

Onde:

I_p é *a corrente primária*

I_1 é *a corrente secundária*

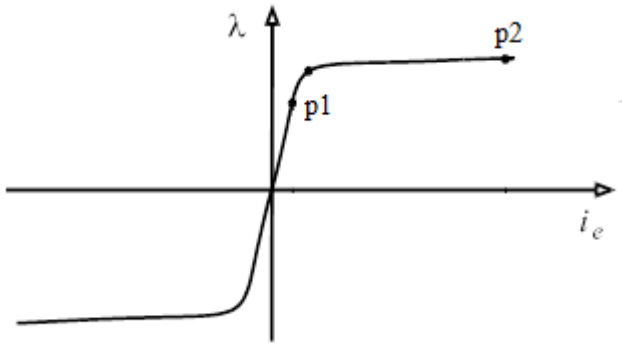
I_e é *a corrente de excitação magnética*

I_c é *a corrente na carga*

X_m é *a reatância de magnetização*

Além disso, o transformador de corrente possui um comportamento histerético na curva fluxo versus corrente de excitação, curva esta que pode ser aproximada por uma reta. Nessa região, diz-se que o transformador encontra-se na *região não saturada*. Porém, para níveis elevados de corrente de excitação, o transformador entra na região chamada *região de saturação*, onde mesmo para elevados valores de corrente, o fluxo permanece constante.

Figura 22 - Curva Fluxo Versus Corrente de Exciação do Núcelo do TC

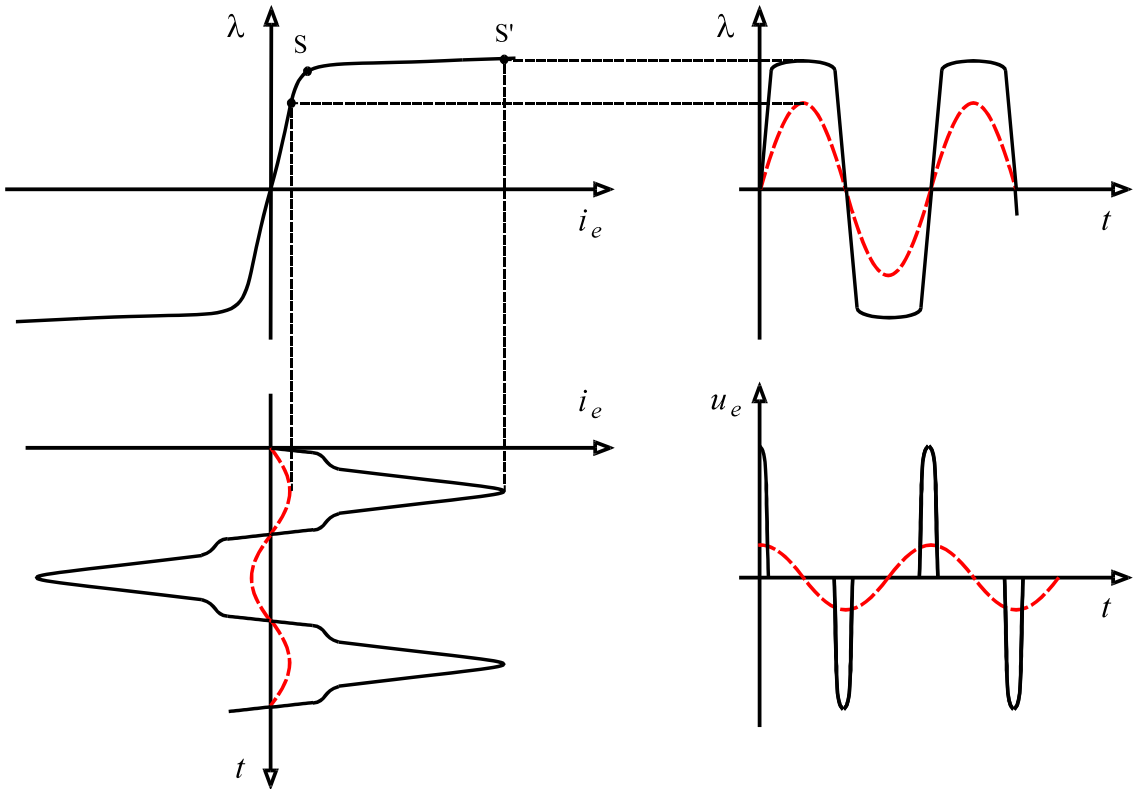


Fonte: Notas de aula do Professor Chagas.

Na figura 21 observa-se que o ponto p1 encontra-se na região não saturada e o ponto p2 encontra-se na região de saturação.

O problema do aumento da carga se dá justamente pela entrada do TC na região de saturação. Como a corrente primária induz uma corrente fixa no secundário do TC (atuando como fonte de corrente independente, dependendo apenas do número de espiras no primário e no secundário) e a corrente secundária é dada por $I_1 = I_e + I_c$ = cte., um aumento na carga leva a uma diminuição da corrente I_c e a um aumento da corrente de excitação I_e, levando o transformador a entrar na região de saturação, como é mostrado nos gráficos na Figura 23.

Figura 23 - Influência da impedância da carga no funcionamento do TC



Fonte: Notas de aula do Professor Chagas.

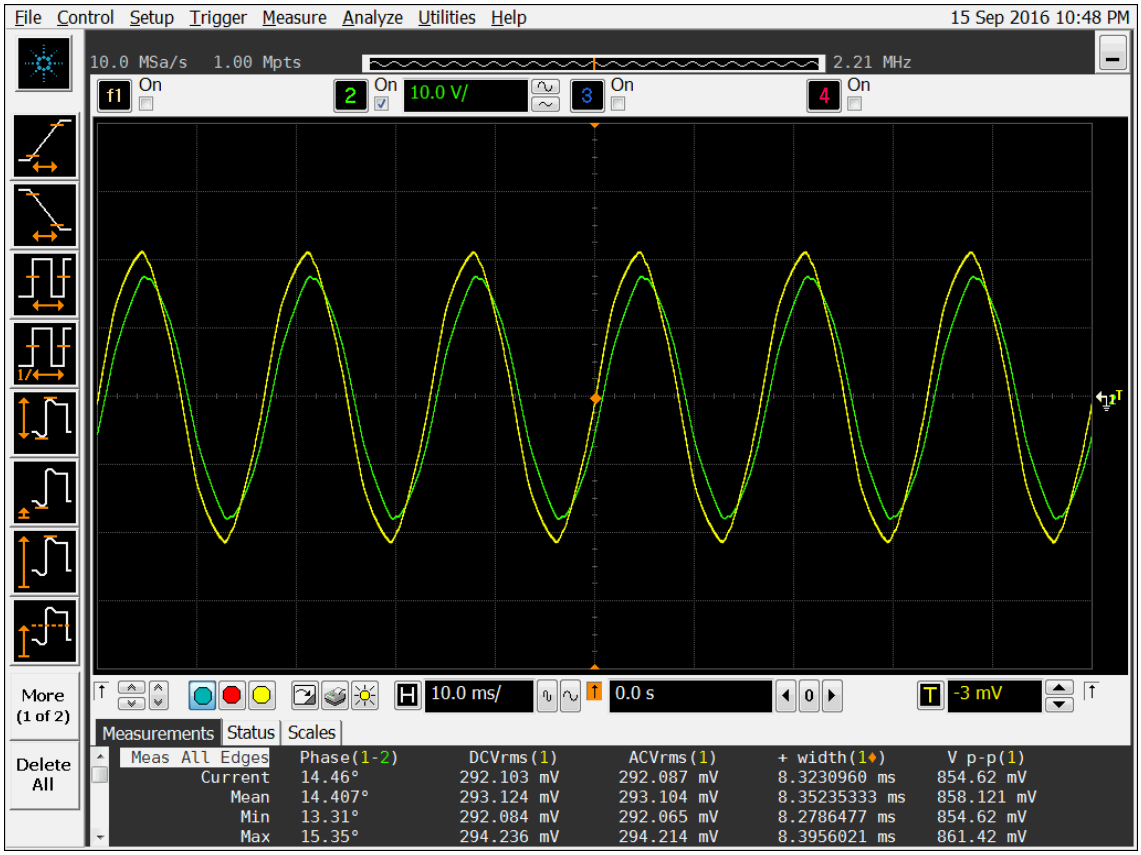
A tensão u_e é chamada de tensão de excitação secundária e, dada a definição de tensão como a derivada do fluxo no tempo, partindo do ponto posicionado na região não-saturada, sendo o fluxo uma função senoidal, a tensão de excitação secundária também será uma senoide. Já no ponto situado na região de saturação, observamos uma distorção no sinal de tensão dado o fluxo limitado. Isso explica a distorção e a limitação na tensão de saída ao aplicarmos cargas com resistência acima de 1 kΩ no TC SCT-013. Como conclusão, recomenda-se que sejam usadas cargas com o menor valor de resistência possível.

4.3 DEFASAGEM ENTRE ENTRADA E SAÍDA

Ainda utilizando a mesma bancada de testes mostrada no tópico 4.1, foi observada e medida no osciloscópio a tensão na carga puramente resistiva aplicada a saída do variac

(sinal verde, representando a corrente no primário do TC, dado que em uma carga puramente resistiva, a fase da corrente e da tensão são as mesmas) e a tensão de excitação secundária do TC (sinal amarelo), mostrada na Figura 24.

Figura 24 - Defasagem entre Entrada e Saída do TC



Fonte: *Printscreen* do aplicativo no Sistema Operacional Windows.

Sendo assim, observou uma diferença de fase de 14.46° para o resistor de 620 Ω, onde o sinal de saída encontra-se adiantado em relação a entrada. O resultado deste teste exige uma calibragem de fase no medidor que utilizar este TC.

4.4 PROJETO E PRODUÇÃO DO *SMART METER*

Uma vez feita a análise do TC SCT-013, o próximo passo foi aplica-lo em um *Smart Meter* mas para isso houve um desenvolvimento deste medidor anteriormente aos testes do TC.

Escolha do *PMIC* e Circuito Externo de Funcionamento

O primeiro passo do projeto do *Smart Meter* foi selecionar o *PMIC* a ser usado para facilitar a aquisição dos dados e os cálculos das grandezas necessárias a validação do TC. Após pesquisas, o *chip* selecionado foi o M90E36A da empresa *Atmel* por possuir um número de canais ADC suficientes para uma futura migração do *Smart Meter* para uma versão de medição de potência trifásica. O M90E36A possui as seguintes características:

- Alimentação única de 3.3 V;
- Interface *SPI* de 4 fios;
- Encapsulamento de baixo custo TQFP48;
- 7 Conversores analógico-digitais diferenciais de 24 *bits* (3 para medição de tensão e 4 para medição de corrente);
- Cálculos de: potência ativa, reativa e aparente; tensão e corrente *RMS*; frequência; *análise fourier* para tensão e corrente; distorção harmonica total + ruído; fator de potência.

As especificações elétricas dos canais *ADC* encontram-se na Tabela 1.

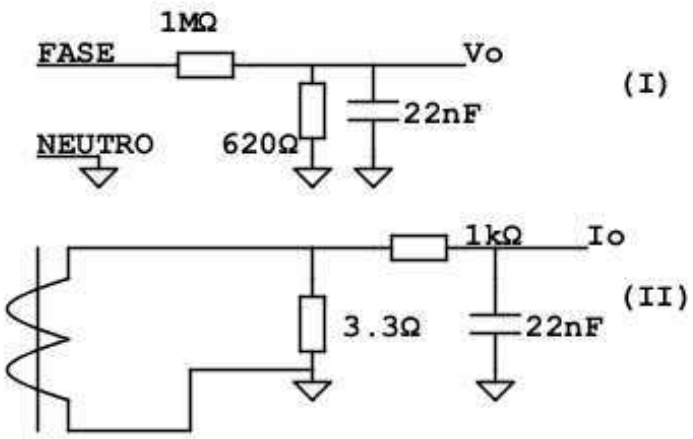
Tabela 1 - Especificações do ADC do M90E36A

Parâmetro	**Mínimo**	**Típico**	**Máximo**	**Unidade**
Tensão Diferencial de Entrada	0.12		720	mVrms
Tensão da Entrada Analógica	GND-300		VDD-1200	mV
Impedância de Entrada		120		kΩ

A partir das especificações dos canais analógicos, foram projetados os circuitos de atenuação de tensão e corrente de forma a suportar tensões de 110 V ou 220 V e considerando uma corrente máxima na entrada de 100 A.

Os circuitos de atenuação são mostrados na Figura 25.

Figura 25 - Circuitos de Atenuação para o M90E36A



Fonte: Elaborada pelo autor no aplicativo FidoCadJ.

A partir dos circuitos podem ser obtidos os ganhos de cada um (desprezando os capacitores de 22 nF, usados para filtrar altas frequências, onde sua influência no ganho e na fase será corrigida por meio da calibragem do medidor). Sendo assim:

- Para o circuito I, atenuador da tensão:

$$V_o = V_i * \frac{620}{1M+620} = (6.1962e-4) * V_i \qquad (22)$$

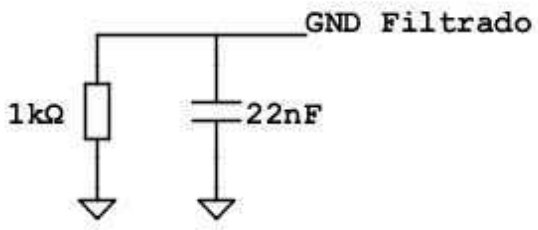
Ou seja para uma tensão de entrada de 220 Vrms, a saída será de 0.1363 Vrms com pico de 192 mV.

- Para o circuito II, atenuador de corrente:

Sabendo que a corrente máxima na saída do TC é de 50 mArms (para uma corrente de 100 Arms no primário), o resistor de resistor de 3.3 Ω faz com que a saída máxima do atenuador de corrente seja de I_{omax}=3.3*50m=165 mVrms com pico de 233.34 mV.

Como os canais *ADC* do M90E36A são diferenciais, as saídas dos atenuadores foram colocadas nas entradas positivas dos canais *ADC* e nos canais negativos foram aplicadas referências filtradas, como é mostrado na Figura 26.

Figura 26 - Referência (GND) filtrada



Fonte: Elaborada pelo autor no aplicativo FidoCadJ.

Projetados os circuitos de atenuação, foi feito um circuito para teste do mesmo em *Protoboard*, contendo todos os elementos externos suficientes para seu funcionamento. Para a leitura dos dados foi utilizado o microcontrolador Arduino Uno como mestre na comunicação *SPI*.

O circuito completo do teste do M90E36A encontra-se em anexo.

Os pinos 37, 38, 39 e 40 do M90E36A, *CS (SS)*, *SCLK*, *SDO* e *SDI* foram conectados as entradas digitais 10, 13, 12 e 11 do Arduino UNO, entradas dedicadas a comunicação *SPI*. O Arduino também ficou responsável pela alimentação do M90E36A. (O Arduino utilizado tem uma saída de 3.3 V regulada e tem também uma chave que configura ele para trabalhar em 3.3 V, atendendo as especificações elétricas do M90E36A, mostradas na Tabela 2).

Tabela 2 - Especificações Elétricas DC do M90E36A

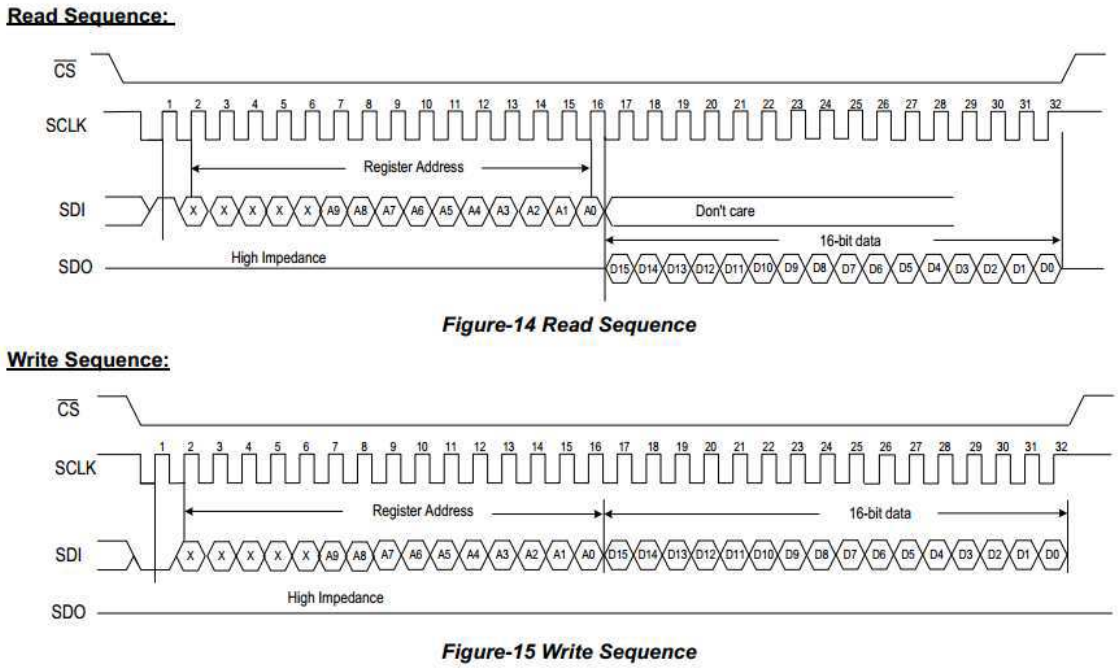
Parâmetro	**Min**	**Tip**	**Max**	**Unidade**
AVDD	2.8	3.3	3.6	V
DVDD	2.8	3.3	3.6	V
VDD18		1.8		V
Nível Lógico Alto	2.4		VDD	V
Nível Lógico Baixo			0.8	V

Comunicação do M90E36A

O M90E36A possui uma gama de registradores de configuração e de armazenamento de dados, todos de tamanho fixo de 16 *bits*. Uma vez conectado o Arduino, antes do início da escrita do código foi necessário estudar como ler e escrever em seus registradores. O processo encontra-se a seguir:

- Muda o nível lógico do CS (SS) de 1 para 0, selecionando o M90E36A e iniciando uma comunicação *SPI* descrita no tópico 3.8. Após isso, o primeiro *bit* a ser enviado define se é um processo de leitura ou escrita (1 para leitura e 0 para escrita). Após o envio do primeiro *bit*, os próximos 15 *bits* representam o endereço do registrador de interesse, enviando primeiro o MSB até o LSB. Se o processo for de leitura, o M90E36A retornará os 16 *bits* de dados referentes ao registrador selecionado, do MSB ao LSB. Se o processo for de escrita, o mestre envia os 16 *bits* de dados que deseja escrever, do MSB até o LSB. Após termino, elevar o nível do CS (SS) novamente.

Figura 27 - Sequência de Leitura e Escrita no M90E36A



Fonte: *Datasheet* do M90E36A.

Após domínio da sequência de escrita e leitura no M90E36A, fez-se necessário um estudo mais detalhado sobre os registradores. O processo de configuração e calibragem do M90E36A encontra-se no próximo tópico.

Processo de Configuração e Calibragem do M90E36A

Antes de iniciar qualquer processo, recomenda-se dar um comando de *reset* no M90E36A, comando que leva o chip a sua configuração padrão inicial. Existem duas formas de executar um *reset*:

- Via *hardware*: Baixar o nível lógico do pino 41 por no mínimo 5 ms (40 ms no máximo) e subir novamente o nível lógico;
- Via *software*: Escrever 0x789A no registrador *SoftReset* (endereço 0x00).

O M90E36A possui registradores chamados *Start*. Os utilizados neste projeto são:

- *ConfigStart*: Relacionado a configuração;
- *AdjStart*: Relacionado a calibragem das aquisições;
- *CalStart*: Relacionado a calibragem dos cálculos;

Ao escrever 0x5678 (hexadecimal) em um registrador do tipo *Start*, o processo ao qual ele está associado inicia.

A cada um dos registradores *Start* está associado um conjunto de registradores, os que guardam as configurações de fato. Na Figura 28, Figura 29 e Figura 30, encontra-se a lista de registradores associados tanto ao *ConfigStart*, *AdjStart* e *CalStart*.

Figura 28 - Registradores associados ao *ConfigStart*.

Register Address	Register Name	Read/Write Type	Functional Description	Power-on Value and Comments
		Configuration Registers*		
30H	ConfigStart	R/W	Calibration Start Command	6886H
31H	PLconstH	R/W	High Word of PL_Constant	0861H
32H	PLconstL	R/W	Low Word of PL_Constant	C468H
33H	MMode0	R/W	HPF/Integrator On/off, CF and all-phase energy computation configuration	0087H
34H	MMode1	R/W	PGA gain configuration	0000H
35H	PStartTh	R/W	Active Startup Power Threshold. 16 bit unsigned integer, Unit: 0.00032 Watt	0000H.
36H	QStartTh	R/W	Reactive Startup Power Threshold. 16 bit unsigned integer, Unit: 0.00032 var	0000H
37H	SStartTh	R/W	Apparent Startup Power Threshold. 16 bit unsigned integer, Unit: 0.00032 VA	0000H
38H	PPhaseTh	R/W	Startup power threshold (for |P|+|Q| of a phase) for any phase participating Active Energy Accumulation. Common for phase A/B/C.	0000H 16 bit unsigned integer, Unit: 0.00032 Watt/var
39H	QPhaseTh	R/W	Startup power threshold (for |P|+|Q| of a phase) for any phase participating ReActive Energy Accumulation. Common for phase A/B/C.	0000H 16bit unsigned integer, Unit: 0.00032 Watt/var
3AH	SPhaseTh	RW	Startup power threshold (for |P|+|Q| of a phase) for any phase participating Apparent Energy Accumulation. Common for phase A/B/C.	0000H 16 bit unsigned integer, Unit: 0.00032 Watt/var
3BH	CS0	R/W	Checksum 0 Checksum register.	421CH (calculated value after reset)

Fonte: *Datasheet* do M90E36A.

Figura 29 - Registradores associados ao *AdjStart*.

Register Address	Register Name	Read/Write Type	Functional Description	Power-on Value
60H	AdjStart	R/W	Measurement Calibration Startup Command	6886H
61H	UgainA	R/W	Phase A Voltage RMS Gain	CE40H
62H	IgainA	R/W	Phase A Current RMS Gain	7530H
63H	UoffsetA	R/W	Phase A Voltage RMS Offset	0000H
64H	IoffsetA	R/W	Phase A Current RMS Offset	0000H
65H	UgainB	R/W	Phase B Voltage RMS Gain	CE40H
66H	IgainB	R/W	Phase B Current RMS Gain	7530H
67H	UoffsetB	R/W	Phase B Voltage RMS Offset	0000H
68H	IoffsetB	R/W	Phase B Current RMS Offset	0000H
69H	UgainC	R/W	Phase C Voltage RMS Gain	CE40H
6AH	IgainC	R/W	Phase C Current RMS Gain	7530H
6BH	UoffsetC	R/W	Phase C Voltage RMS Offset	0000H
6CH	IoffsetC	R/W	Phase C Current RMS Offset	0000H
6DH	IgainN	R/W	Sampled N line Current RMS Gain	7530H
6EH	IoffsetN	R/W	Sampled N line Current RMS Offset	0000H
6FH	CS3*	R/W	Checksum 3	8EBEH

Fonte: *Datasheet* do M90E36A.

Figura 30 - Registradores associados ao *CalStart*.

Register Address	Register Name	Read/Write Type	Functional Description	Power-on Value
			Calibration Registers	
40H	CalStart	R/W	Calibration Start Command	6886H
41H	POffsetA	R/W	Phase A Active Power Offset	0000H
42H	QOffsetA	R/W	Phase A Reactive Power Offset	0000H
43H	POffsetB	R/W	Phase B Active Power Offset	0000H
44H	QOffsetB	R/W	Phase B Reactive Power Offset	0000H
45H	POffsetC	R/W	Phase C Active Power Offset	0000H
46H	QOffsetC	R/W	Phase C Reactive Power Offset	0000H
47H	GainA	R/W	Phase A Active/Reactive Energy calibration gain	0000H
48H	PhiA	R/W	Phase A calibration phase angle	0000H
49H	GainB	R/W	Phase B Active/Reactive Energy calibration gain	0000H
4AH	PhiB	R/W	Phase B calibration phase angle	0000H
4BH	GainC	R/W	Phase C Active/Reactive Energy calibration gain	0000H
4CH	PhiC	R/W	Phase C calibration phase angle	0000H
4DH	CS1*	R/W	Checksum 1	0000H

Fonte: *Datasheet* do M90E36A.

Observando a lista de registradores associados aos registradores *Start*, observa-se que para cada processo está presente um registrador chamado CS (*Checksum*). Antes de finalizar um determinado processo, o mestre deve escrever em CS um código que é gerado a partir dos valores de todos os outros registradores associados. O M90E36A, por sua vez, recalcula frequentemente este valor e compara com o que foi escrito. Caso haja alguma diferença, isso representa uma mudança não desejada na configuração e um *bit* no

registrador chamado *SysStatus0* é mudado de 0 pra 1 (o *bit* 14 no caso da configuração, 12 no caso da calibragem de aquisição e o *bit* 8 no caso da calibragem dos cálculos).

As equações que geram o valor do registrador *CheckSum* são mostradas a seguir:

- O *byte* menos significativo é dado por:

$$L_B = MOD(H_1 + H_2 + H_3 + \cdots + L_1 + L_2 + L_3 + \cdots, 2^8) \qquad (23)$$

Onde H e L são, respectivamente, os *bytes* mais significativos e menos significativos dos registradores associados. A função *MOD* representa o resto da divisão do primeiro parâmetro pelo segundo.

- O *byte* mais significativo é dado por:

$$H_B = H_1\ XOR\ H_2\ XOR\ H_3\ XOR\ \ldots XOR\ L_1\ XOR\ L_2\ XOR\ \ldots \qquad (24)$$

Após cálculo e escrita do *CheckSum*, escreve-se 0x8765 no registrador *Start* associado ao processo para finaliza-lo.

Leitura das Grandezas Calculadas

A Tabela 3 mostra o formato de codificação das grandezas calculadas. A lista completa com os endereços encontra-se em anexo.

Tabela 3 - Grandezas calculadas M90E36A

Tipo	**Formato e resolução**
Tensao RMS	Sem sinal. Formado por dois registradores com 32 *bits* no total. Despreza-se o ultimo *byte* e a resolução é 0.01/256 V. Valor máximo: 655.35 V
Corrente RMS	Sem sinal. Formado por dois registradores com 32 *bits* no total. Despreza-se o último *byte* e a resolução é 0.001/256 A. Valor máximo: 65.535 A
Potência Ativa/Reativa	Complemento de 2. Formado por dois registradores com 32 *bits* no total. Despreza-se o último *byte* e a resolução é 1/256 (W/var). Valor máximo (módulo): 32767 (W/var).
Fator de Potência	Registrador de 16 *bits*. Resolução: 0.001. Valor máximo (módulo): 1

Roteiro de Configuração e Calibragem do M90E36A

Uma vez compreendido o funcionamento do M90E36A, o roteiro de inicialização e calibragem foi montado e é apresentado a seguir:

Configuração:

- Escreve-se 0x5678 no registrador *ConfigStart* (endereço 0x30);
- Configuração do registrador *MMode0* (endereço 0x33):

Tabela 4 - Configuração do registrador MMode0

Bit	**Nome**	**Valor**	**Configuração**
15, 14	-	-	Reservados
13	*I1I3Swap*	0	Fase A corresponde ao *ADC* número 1.
12	*Freq60Hz*	1	Frequência da rede: 60Hz
11	*HPFOff*	0	Filtro passa-altas habilitado.
10	*didtEn*	0	Integrador para sensor de corrente desabilitado.
9	*001LSB*	0	Configuração *default* de energia. (Não-usado)
8-0	-	0x087	Configuração trifásica (Mantidos os valores *default*)

  - Valor a ser escrito: 0x1087;
- Calcula e escreve o *Checksum*;
- Escreve-se 0x8765 no registrador *ConfigStart*;

Calibragem das aquisições:

- Escreve-se 0x5678 no registrador *AdjStart* (endereço 0x60);
- Tensão:
  - Aplica a tensão nominal no circuito de atenuação e faz a leitura de seu valor *RMS* por um medidor de referência (U_{real});
  - Faz a leitura do valor *RMS* calculado pelo M90E36A, dado o ganho padrão dele de 52800 (U_{medido}).
  - Calcula o ganho correto, pela equação:

$$G = \frac{U_{real}}{U_{medido}} * 52800 \qquad (25)$$

  - Escreve o valor resultado no registrador de ganho correspondente (*Ugain*).

- O processo de calibragem para a corrente é semelhante ao da tensão mas com um problema em relação ao valor máximo. Como foi mostrado na tabela 3, o valor máximo *RMS* é 65.535 A, o que é um problema visto que o medidor foi projetado para 100 A. Visto isso, foi necessária uma manipulação dos ganhos. O processo completo encontra-se a seguir:
    - Aplica-se uma corrente de referência no intervalo especificado no momento do projeto do atenuador. No caso do TC SCT-013 que é especificado para correntes de entrada entre 0-100 A, usa-se qualquer corrente nessa faixa de valores (I_{real});
    - Faz a leitura do valor *RMS* calculado pelo M90E36A, dado o ganho padrão dele de 30000 (I_{medido}).
    - Como a corrente máxima é especificação do M90E36A e não pode ser configurada, no cálculo do ganho usa-se a corrente de referência atenuada por uma constante K:

$$G = \frac{\frac{I_{real}}{K}}{I_{medido}} * 30000 \qquad (26)$$

    - Escreve o valor resultado no registrador de ganho correspondente (*Igain*);
    - Desta forma, o valor fornecido pelo M90E36A não será o valor real, será o valor real atenuado por K. Ou seja, quando o M90E36A der como resultado 65.535 A, o resultado real é 65.535*K. Isso terá implicações também nas potências, como é mostrado a seguir:

$$P_{real} = P_{medido} * K \qquad (27)$$
$$Q_{real} = Q_{medido} * K \qquad (28)$$

- Calcula e escreve o *Checksum*;
- Escreve-se 0x8765 no registrador *AdjStart*;

Calibragem do cálculo:

- A calibragem feita em relação aos cálculos é justificada na diferença de fase entre a entrada e a saída nos circuitos de atenuação. Sendo assim, aplica-se uma carga com fator de potência conhecido e baseado nela corrige-se a fase.
- O M90E36A possui registradores chamados *Phi* que aplicam atrasos no sinal de tensão ou de corrente para correção da fase. Como foi visto no tópico 4.3, a saída do atenuador de corrente é adiantada em relação a entrada, sendo necessário aplicar um atraso no canal de corrente. Sendo assim, o procedimento é o seguinte:
    - Escreve-se 0x5678 no registrador *CalStart* (endereço 0x40);
    - Faz-se a leitura da potência ativa e reativa na carga de referência. No caso particular deste projeto, fez-se a leitura das potências em uma lâmpada incandescente considerando que a potência reativa real seja 0 e que a corrente seja puramente senoidal;
    - Sabendo que a diferença de fase por ser calculada, a partir do triângulo de potências, pela equação:

$$\phi = tg^{-1}\left(\frac{Q}{P}\right) \qquad (29)$$

    - Uma vez calculada a diferença de fase é feita a conversão dela para atraso de tempo pela seguinte equação:

$$360° \rightarrow \frac{1}{60}s$$
$$\phi \rightarrow t$$
$$t = \frac{\phi}{21600}s \qquad (30)$$

    - O registrador *Phi* aplica atrasos no sinal de corrente com base em número de ciclos de 2.048MHz, ou seja, a conversão é feita pela equação:

$$1\ ciclo \rightarrow \frac{1}{2.048e^6}\,s$$

$$N_c \rightarrow t$$

$$N_c = t * 2.048e^6\ ciclos \qquad (31)$$

- Escreve o valor resultado no registrador de fase correspondente (*Phi*);
- Calcula e escreve o *Checksum*;
- Escreve-se 0x8765 no registrador *AdjStart*;

Resultados Numéricos da Calibragem

Tensão *RMS*:

$$U_{real} = 225.2\ V$$

$$U_{medido} = 377.8740\ V$$

$$G_U = \frac{U_{real}}{U_{medido}} * 52800 = 31467\ (ganho\ de\ tensão)$$

Corrente *RMS* (utilizando um micro-ondas como referência de corrente):

$$I_{real} = 6.5\ A$$

$$K = 10$$

$$I_{medido} = 1.0141\ A$$

$$G_I = \frac{\frac{I_{real}}{K}}{I_{medido}} * 30000 = 19229\ (ganho\ de\ corrente)$$

Fase (utilizando uma lâmpada incandescente de 100 W como referência):

$$P = 96.58\ W$$

$$Q = 7.98\ var$$

$$\phi = tg^{-1}\left(\frac{Q}{P}\right) = 4.7234°$$

$$t = \frac{\phi}{21600} = 2.1868e^{-4}\ s$$

$$N_c \cong 448\ ciclos\ (atraso\ no\ sinal\ de\ corrente)$$

O medidor de referência utilizado foi um multímetro *Hikari True RMS* (10 A AC máximo).

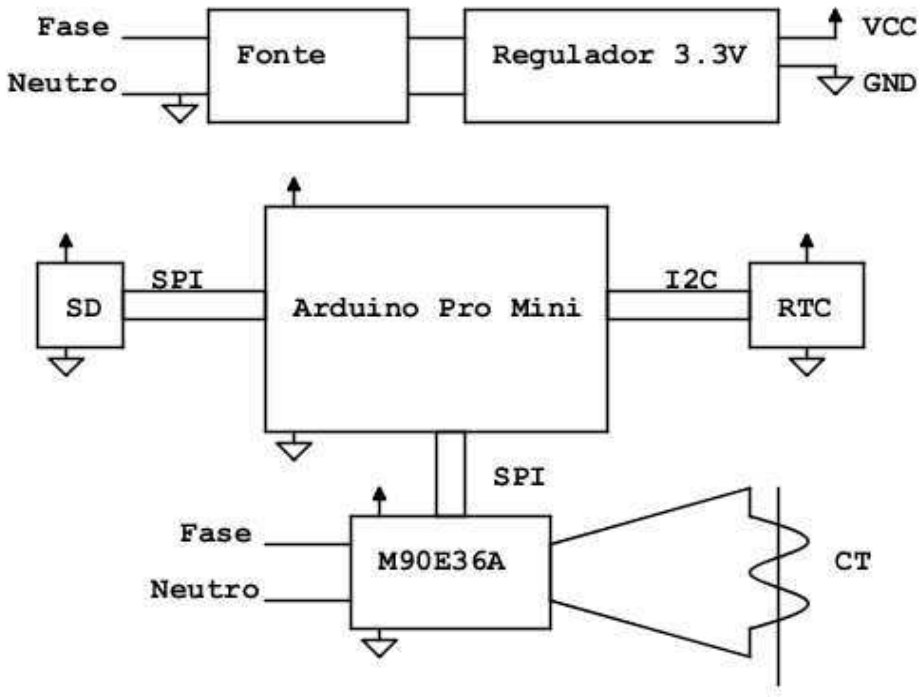
Arquitetura do *Smart Meter*

O *Smart Meter* foi projetado possuindo as seguintes características:

- Medição de potência com o M90E36A;
- Relógio de tempo real com o auxílio de uma placa de circuito impresso possuindo o DS1307;
- Armazenamento de dados com o auxílio de uma placa com entrada para cartão *SD*;
- Alimentação com fonte de tensão contínua 5 V regulada para 3.3 V;
- Arduino Pro-Mini, com o atmega328p, como microcontrolador principal.

A arquitetura é mostrada na Figura 31.

Figura 31 - Arquitetura *Smart Meter*



Fonte: Elaborada pelo autor no aplicativo FidoCadJ.

Pela figura observa-se as comunicações: Arduino Pro Mini, como microcontrolador principal, comunicando-se com o cartão *SD* via *SPI*, com o M90E36A via *SPI* também e com o *RTC* via *I2C*. O atmega328p, microcontrolador do Arduino Pro Mini, possui em sua arquitetura barramentos *SPI* e *I2C* implementados em *hardware*. Basicamente, o *Smart Meter* é o circuito de teste do M90E36A (em anexo) com o acréscimo dos módulos *SD* e *RTC*.

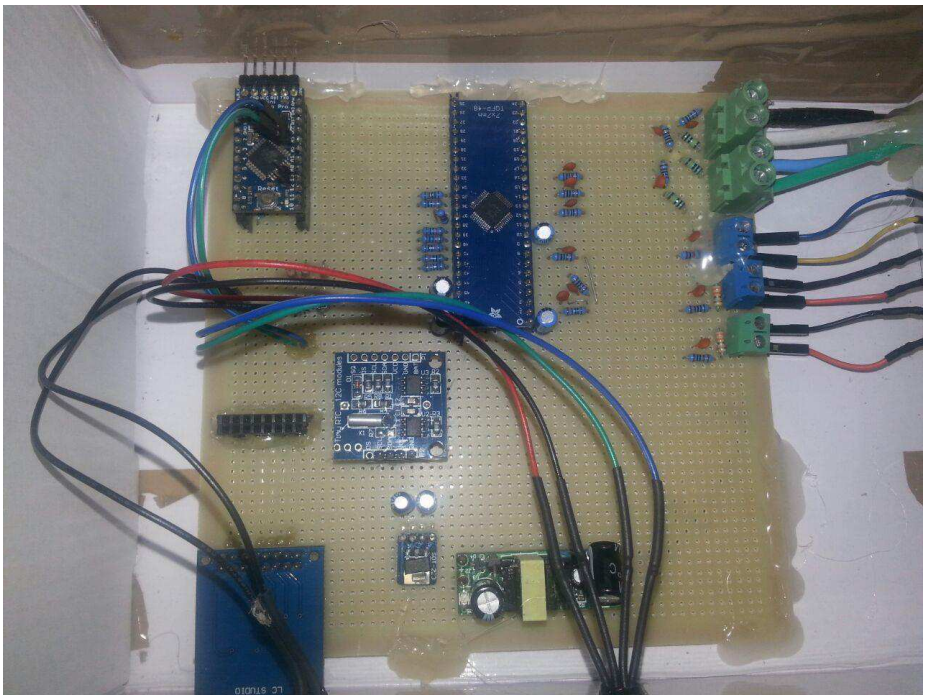
Além destes módulos, foi colocado também um *LED RGB* (3 cores) para indicação do status do medidor (A cor vermelha representando a etapa de configuração, a cor verde simbolizando que o medidor está pronto e a cor azul representando a etapa de coleta de medições). Foi colocado também um botão para início e fim da coleta.

Na Figura 32 e na Figura 33, fotos reais do *Smart Meter* concluído, colocado em caixa de papel devido a limitações financeiras.

Figura 32 - *Smart Meter*

Fonte: Elaborada pelo autor.

Figura 33 - *Smart Meter* Internamente



Fonte: Elaborada pelo autor.

Software do *Smart Meter*

O *software* para o Arduino Pro Mini foi feito utilizando o aplicativo para *Windows* Arduino IDE: plataforma de edição, compilação e *download* de *software* no microcontrolador.

Na Tabela 5 a seguir encontram-se as bibliotecas utilizadas bem como algumas de suas funções.

Tabela 5 - Bibliotecas utilizadas no Smart Meter

Biblioteca	**Descrição**	**Funções e Classes Utilizadas**
SdFat	Biblioteca para acesso de leitura e escrita em sistemas de arquivos FAT16/FAT32 em cartões *SD*.	Classe SdFat: Representa o cartão *SD*. cardBegin(): Inicializa cartão. cardSize(): Retorna tamanho do cartão. fsBegin(): Inicializa FS. rmRfStar(): Apaga todos os arquivos. ls(): Lista os arquivos do cartão. Classe SdFile: Representa um arquivo. open(): Abre arquivo. close(): Fecha arquivo. write(): Escreve em arquivo.
RTClib	Biblioteca para manipulação do *RTC*.	begin(): Inicializa o *RTC*. isRunning(): Testa se esta funcionando. adjust(): Ajusta a hora no *RTC*. now(): Retorna a hora atual.
SPI	Biblioteca com os métodos da comunicação *SPI*.	begin(): Inicializa a comunicação. beginTransaction(): Inicializa uma transação. transfer16(): Envia 16 *bits*. endTransaction(): Finaliza uma transação.

Como há apenas um barramento *SPI* no Arduino Pro Mini e dois módulos *SPI* (Módulo *SD* e o M90E36A), o controle foi feito normalmente pelo sinal de *SS*, uma porta digital do Arduino representando o *SS* do *SD* e outra porta digital representando o *SS* do M90E36A.

Como não há biblioteca para o M90E36A, funções tiveram que ser implementadas e estão apresentadas na Tabela 6.

Tabela 6 - Funções do M90E36A

Função	**Descrição**
pmicSetup()	Inicializa o M90E36A.
write16()	Escreve 16 *bits* em um registrador.
read16()	Lê 16 *bits* de um registrador.
csCalculator()	Calcula o CS.
realRms()	Retorna os valores RMS.
activePower()	Retorna a potência ativa.
reactivePower()	Retorna a potência reativa.
apparentPower()	Retorna a potência aparente.
powerFactor()	Retorna fator de potência.
checksumStatus()	Retorna status do Checksum.
measurementCalibration()	Inicia ou finaliza a calibragem de aquisição.
energyCalibration()	Inicia ou finaliza a calibragem de cálculos.
measurementGainCalibration()	Calibra os ganhos das medições.

O código completo encontra-se em anexo.

4.5 COLETA DE MEDIÇÕES E VALIDAÇÃO

Uma vez concluídos o *Smart Meter* e os testes com o TC SCT-013, alguns equipamentos foram medidos: micro retifica, geladeira, microondas, sanduicheira e ventilador. O canal utilizado para medição foi o canal 3 do M90E36A.

As grandezas calculadas foram:

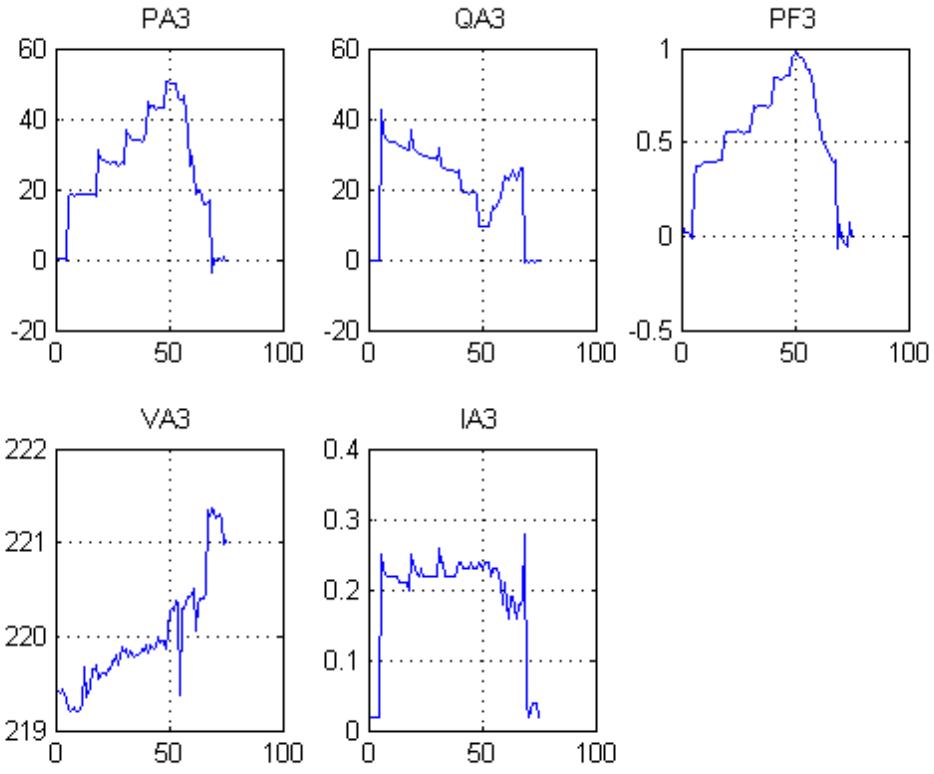
- PA3: Potência ativa do canal 3, medida em Watt;
- QA3: Potência reativa do canal 3, medida em var;
- PF3: Fator de potência, grandeza adimensional;
- VA3: Tensão RMS do canal 3, medida em Volt;
- IA3: Corrente RMS do canal 3, medida em Ampere.

As grandezas foram plotadas em função do tempo e comparadas com as medições do medidor de referência (multímetro *Hikari True RMS*).

Micro Retifica

A micro retifica foi avaliada sem carga, aumentando a velocidade e depois diminuindo. Na Figura 34, encontram-se as curvas feitas com o auxílio do *Matlab* com os dados coletados pelo *Smart Meter*.

Figura 34 - Curvas Micro Retifica



Fonte: Elaborada pelo autor no *Matlab*.

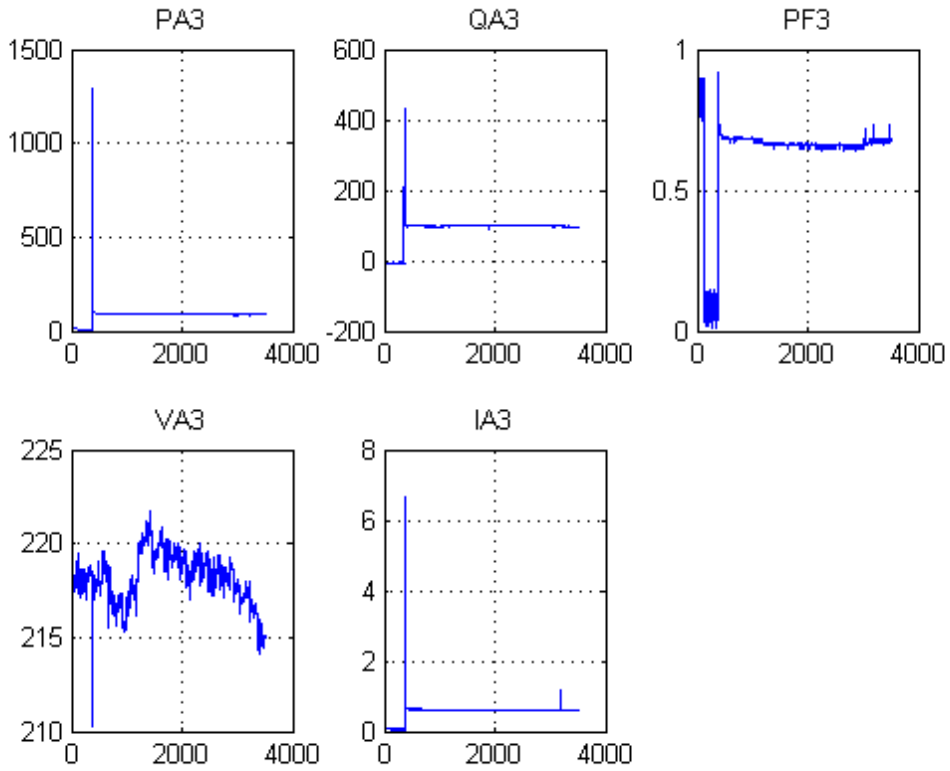
A corrente medida no medidor de referência variava entre 0.21 e 0.23 A.

É possível observar os degraus de aumento e diminuição da velocidade na curva de potência ativa.

Geladeira

Na Figura 35, as curvas referentes a geladeira.

Figura 35 - Curvas da Geladeira



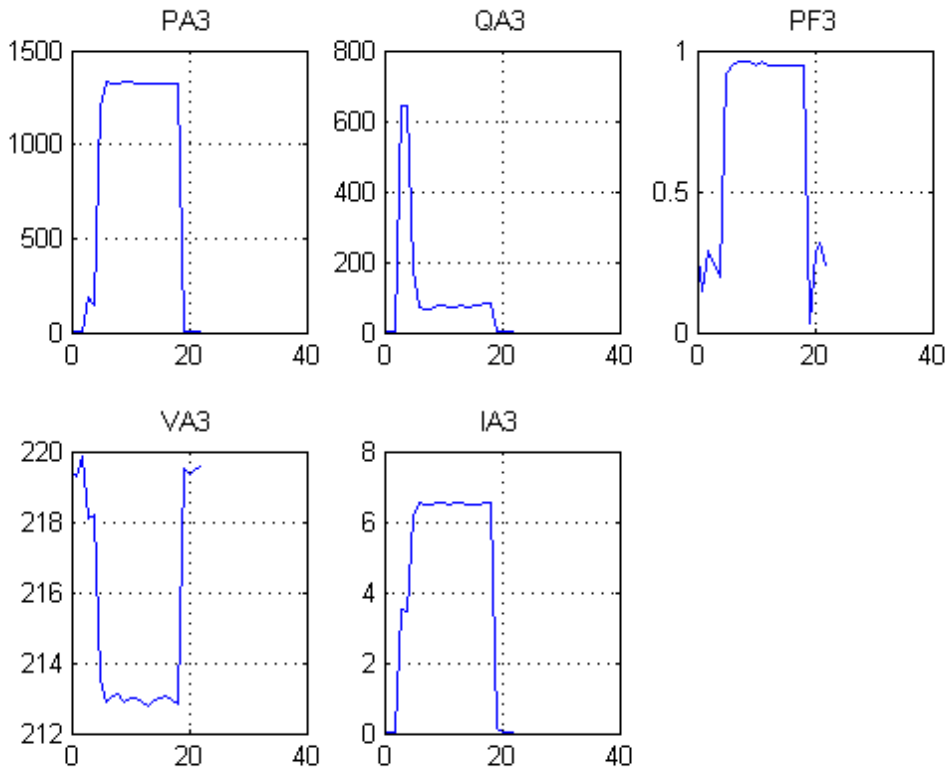
Fonte: Elaborada pelo autor no *Matlab*.

Pela análise das curvas observa-se claramente o pico de partida do motor do compressor da geladeira. Tanto o medidor de referência quanto o *Smart Meter* deram como resultado 0.61 A de corrente RMS.

Microondas

O microondas foi avaliado sem nenhuma carga, apenas em funcionamento na potência máxima. Na Figura 36, as curvas referentes ao microondas.

Figura 36 - Curvas do Micro-ondas



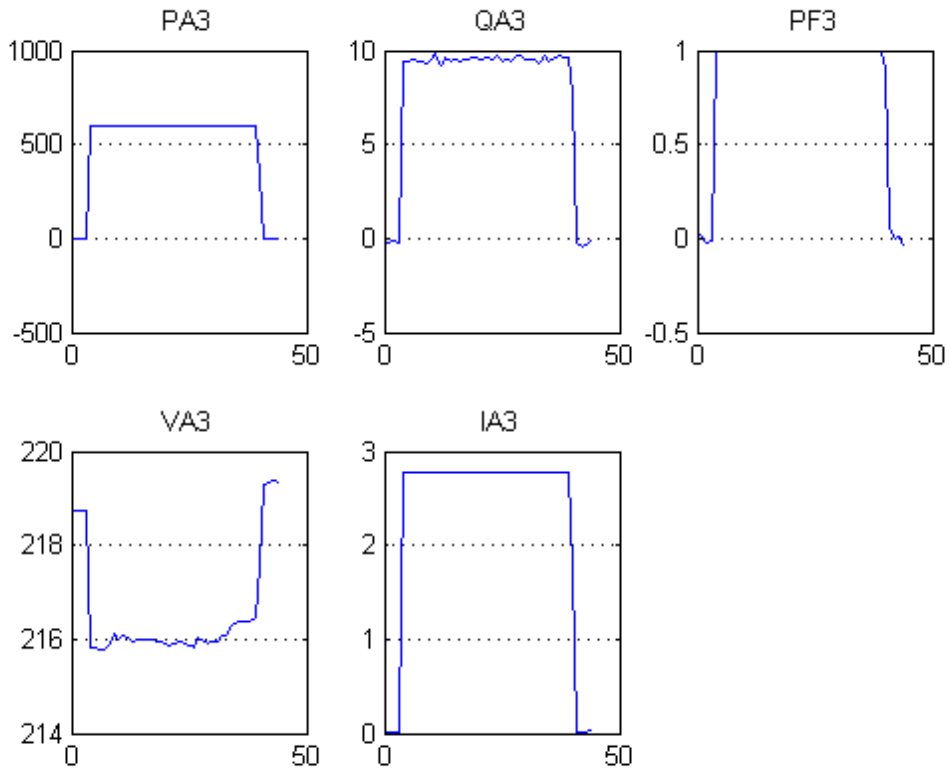
Elaboradas pelo autor no Matlab.

Pela potência mais elevada, observa-se uma queda de tensão bem maior que nas outras situações. Tanto o medidor de referência quanto o *Smart Meter* deram como resultado 6.5 A de corrente *RMS*.

Sanduicheira

A sanduicheira foi avaliada sem nenhuma carga, apenas em funcionamento na potência máxima. Na Figura 37, as curvas referentes a sanduicheira:

Figura 37 - Curvas Sanduicheira



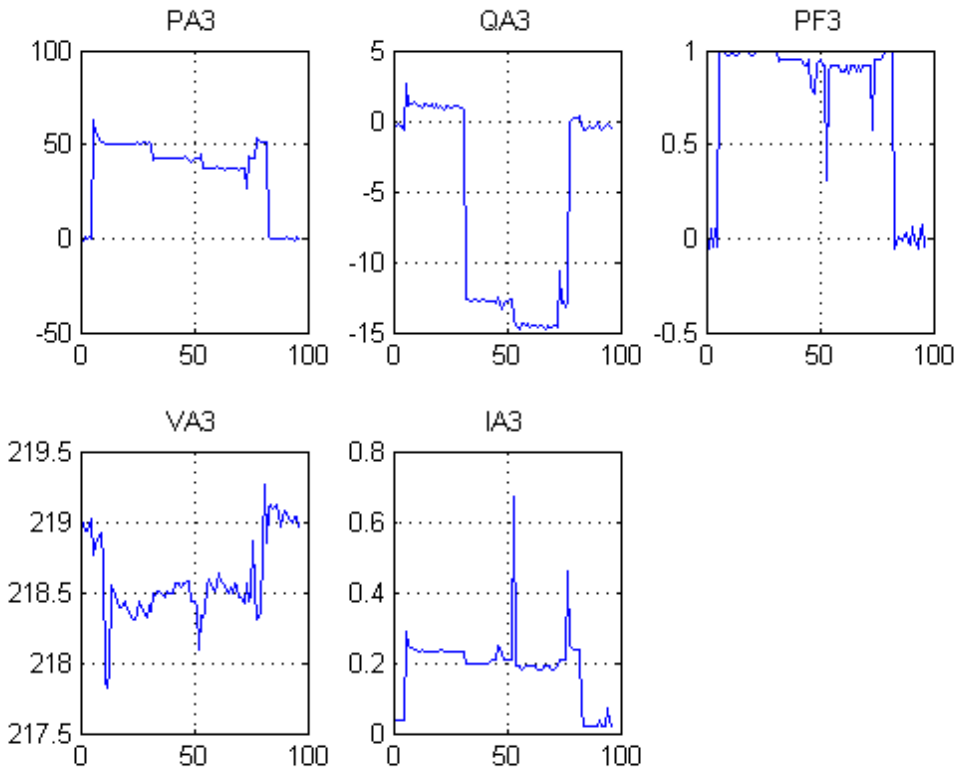
Fonte: Elaborada pelo autor no Matlab.

Como a sanduicheira pode ser aproximada por uma carga puramente resistiva, que aquece e fornece calor ao alimento, o fator de potência deve dar 1, o que foi verificado nas curvas fornecidas pelo *Smart Meter*. Tanto o medidor de referência quanto o *Smart Meter* deram como resultado 2.8 A de corrente *RMS*.

Ventilador

O ventilador foi avaliado diminuindo e aumentando sua velocidade. O resultado encontra-se nas curvas apresentadas na Figura 38.

Figura 38 - Curvas Ventilador



Fonte: Elaborada pelo autor no Matlab.

Pela curva de potência ativa é possível observar a queda de potência ao diminuir a velocidade do ventilador. O medidor de referência deu como resultado 0.21-0.24 A para a corrente *RMS*, o que pode ser verificado no gráfico da corrente fornecido pelo *Smart Meter*.

5 TRABALHOS FUTUROS

No desenvolvimento de projetos sempre há trabalhos futuros, melhorias, otimizações, etc. A proposta deste trabalho de conclusão de curso se ateve a avaliação do TC, implementação de um *Smart Meter* e medições de corrente com o mesmo, atividades que foram concluídas. Mas ficou claro também que ainda há muito trabalho a ser feito, como por exemplo:

- Validação das potências com um medidor de potência verificado;
- Validação do fator de potência;
- Medição de harmônicas com o M90E36A;
- Cálculo de energia consumida;
- Rotinas de autocalibrarem;
- Testes do *Smart Meter* em ambiente trifásico;
- Avaliação do TC SCT-013 para valores mais elevados de corrente;
- Desenvolvimento de uma placa de circuito impresso;

6 CONCLUSÃO

Este trabalho de conclusão de curso tratou do desenvolvimento de um produto destinado à medição de potência de forma digital, utilizando um *hardware*, compacto, em que se obtém simplicidade no processo e excelente grau de exatidão nos resultados. Mostrou passos desde o projeto do *hardware*, incluindo escolhas de componentes, ferramentas, testes e avalições, tomando as conclusões necessárias para o progresso, até o desenvolvimento do *software*.

Em relação aos objetivos deste trabalho, foi possível concluir que o transformador de corrente SCT-013 pode muito bem ser utilizado como parte de um medidor de potência, sendo um sensor de corrente não invasivo, auxiliando nas medições de forma segura e precisa, devido a sua isolação e linearidade. Foi possível concluir também que o *chip* M90E36A, assim como outros chips da família *PMIC*, facilitam as soluções de medição de potência: uma alternativa barata, pequena, baixo consumo e poderosa na questão computacional.

Como última atividade do curso de graduação em Engenharia Elétrica serviu como revisão dos conhecimentos, consolidação e aplicação. Inúmeras disciplinas foram utilizadas neste trabalho: Cálculo, álgebra vetorial, circuitos lógicos, circuitos elétricos, eletrônica, sistemas elétricos, arquiteturas de sistemas digitais, técnicas de programação, instalações elétricas, etc. O curso de Engenharia Elétrica da Universidade Federal de Campina Grande fornece uma base sólida e conhecimentos avançados para atuação do engenheiro formado no mercado de trabalho, sendo capaz de se adaptar a atividades de empresas ou de encabeçar seus próprios projetos.

REFERÊNCIAS

NILSSON, James W.;RIEDEL, Susan A. *Circuitos Elétricos*. 8. Ed. São Paulo : Pearson Prentice Hall, 2009.

GUERRA, F. C. F. *Notas de aula: Aspectos Básicos dos Transformadores de Corrente*. 28p. Campina Grande, 2011.

IEEE. *IEEE Standard Definitions for the Measurement of Electric Power Quantities Under Sinusoidal, Nonsinusoidal, Balanced, or Unbalanced Conditions*. IEEE Std 1459-2010. Mar. 2010.

ANEEL. *Nota Técnica n° 0083/2012-SRD/ANEEL*. 19p. Jun. 2012.

ANEEL. *Procedimentos de Distribuição de Energia Elétrica no Sistema Elétrico Nacional – PRODIST. Módulo 8 – Qualidade de Energia Elétrica*. 62p. Jan. 2010.

HEATH, Steve. *Embedded Systems Design*. 2. Ed. Newnes, 2003.

SPARKFUN. *I2C*. Disponível em: https://learn.sparkfun.com/tutorials/i2c. Acesso em Setembro de 2016.

FREEBSD. *Serial and UART Tutorial*. Disponível em: https://www.freebsd.org/doc/en/articles/serial-uart/. Acesso em Setembro de 2016.

WIKIPEDIA. *Secure Digital Card*. Disponível em: https://pt.wikipedia.org/wiki/Secure_Digital_Card. Acesso em Setembro de 2016.

SUHETT, Marcos Riva. *Análise de Técnicas de Medição de Potência Reativa em Medidores Eletrônicos*. Dissertação (Mestrado em Engenharia Elétrica). UFRJ, 2008.

GARCIA, Flávio Resende. *Harmônicos em Sistemas Elétricos de Potência*. 50p.

ATMEL. *Atmel M90E36A Datasheet*. 87p. California, 2015.

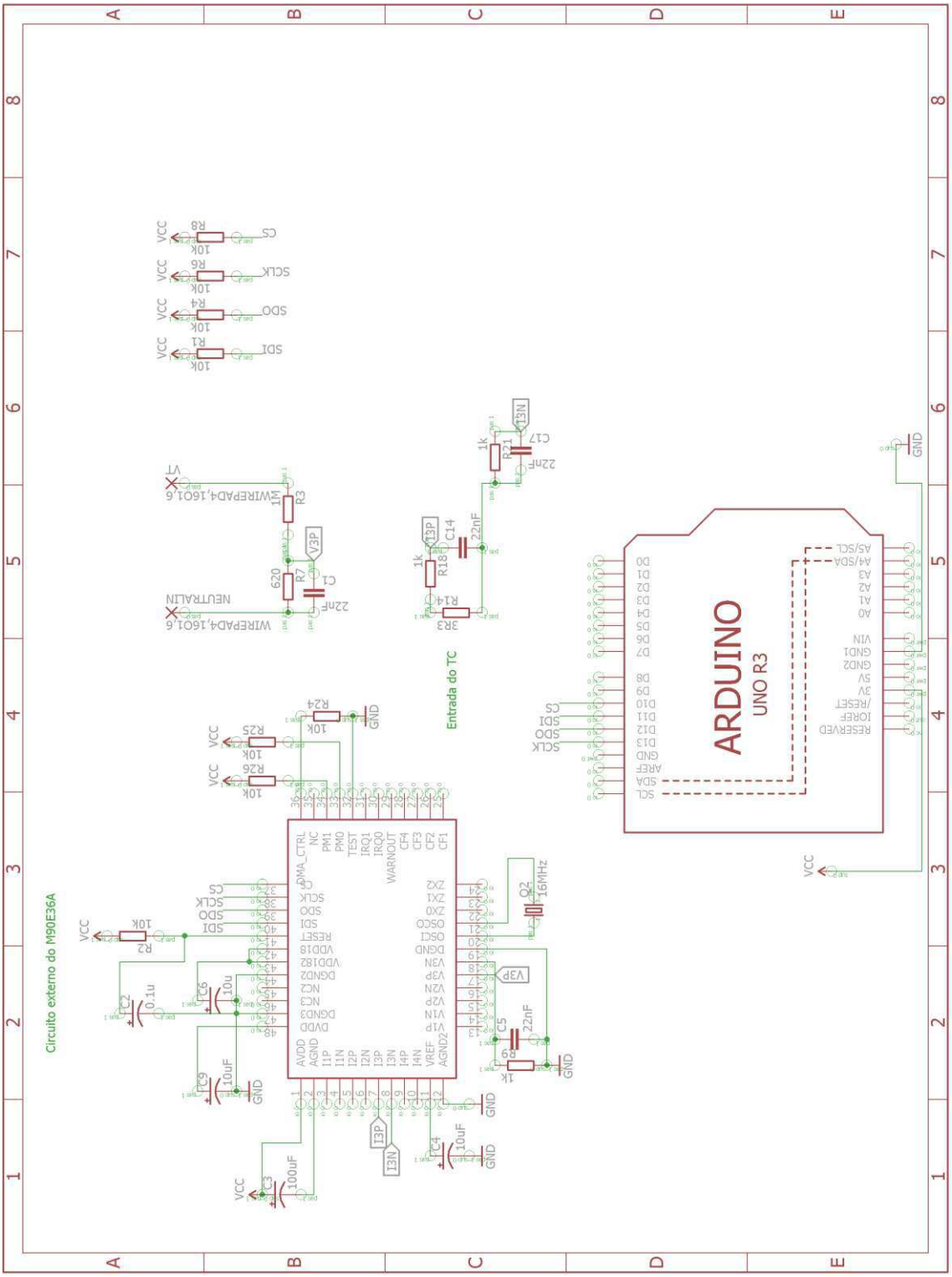
ATMEL. *ATmega48A/PA/88A/PA/168A/PA/328/P Datasheet*. 660p. California, 2015.

MAXIM INTEGRATED. *DS1307 Datasheet*. 14p. California. 2015.

BEIJING YAOHUADECHANG ELECTRONIC. *SCT-013-000 Datasheet*. 1p. 2011.

ANEXO A – CIRCUITO DE TESTE DO M90E36A

Figura 39 - Circuito de Teste do M90E36A



Fonte: Elaborada pelo autor no aplicativo *Eagle*.

ANEXO B – LISTA DE GRANDEZAS CALCULADAS PELO M90E36A

Figura 40 - Grandezas calculadas pelo M90E36A Pg. 1

B0H	PmeanT	R	Total (all-phase-sum) Active Power
B1H	PmeanA	R	Phase A Active Power
B2H	PmeanB	R	Phase B Active Power
B3H	PmeanC	R	Phase C Active Power
B4H	QmeanT	R	Total (all-phase-sum) Reactive Power
B5H	QmeanA	R	Phase A Reactive Power
B6H	QmeanB	R	Phase B Reactive Power
B7H	QmeanC	R	Phase C Reactive Power
B8H	SAmeanT	R	Total (Arithmetic Sum) apparent power
B9H	SmeanA	R	phase A apparent power
BAH	SmeanB	R	phase B apparent power
BBH	SmeanC	R	phase C apparent power
BCH	PFmeanT	R	Total power factor
BDH	PFmeanA	R	phase A power factor
BEH	PFmeanB	R	phase B power factor
BFH	PFmeanC	R	phase C power factor
C0H	PmeanTLSB	R	Lower word of Total (all-phase-sum) Active Power
C1H	PmeanALSB	R	Lower word of Phase A Active Power
C2H	PmeanBLSB	R	Lower word of Phase B Active Power
C3H	PmeanCLSB	R	Lower word of Phase C Active Power
C4H	QmeanTLSB	R	Lower word of Total (all-phase-sum) Reactive Power
C5H	QmeanALSB	R	Lower word of Phase A Reactive Power
C6H	QmeanBLSB	R	Lower word of Phase B Reactive Power
C7H	QmeanCLSB	R	Lower word of Phase C Reactive Power
C8H	SAmeanTLSB	R	Lower word of Total (Arithmetic Sum) apparent power
C9H	SmeanALSB	R	Lower word of phase A apparent power
CAH	SmeanBLSB	R	Lower word of phase B apparent power
CBH	SmeanCLSB	R	Lower word of phase C apparent power

Fonte: *Datasheet* do M90E36A.

Figura 41 - Grandezas calculadas pelo M90E36A Pg. 2

D0H	PmeanTF	R	Total active fundamental power
D1H	PmeanAF	R	phase A active fundamental power
D2H	PmeanBF	R	phase B active fundamental power
D3H	PmeanCF	R	phase C active fundamental power
D4H	PmeanTH	R	Total active harmonic power
D5H	PmeanAH	R	phase A active harmonic power
D6H	PmeanBH	R	phase B active harmonic power
D7H	PmeanCH	R	phase C active harmonic power
D8H	IrmsN1	R	N Line Sampled current RMS
D9H	UrmsA	R	phase A voltage RMS
DAH	UrmsB	R	phase B voltage RMS
DBH	UrmsC	R	phase C voltage RMS
DCH	IrmsN0	R	N Line calculated current RMS
DDH	IrmsA	R	phase A current RMS
DEH	IrmsB	R	phase B current RMS
DFH	IrmsC	R	phase C current RMS
E0H	PmeanTFLSB	R	Lower word of Total active fundamental Power
E1H	PmeanAFLSB	R	Lower word of phase A active fundamental Power
E2H	PmeanBFLSB	R	Lower word of phase B active fundamental Power
E3H	PmeanCFLSB	R	Lower word of phase C active fundamental Power
E4H	PmeanTHLSB	R	Lower word of Total active harmonic Power
E5H	PmeanAHLSB	R	Lower word of phase A active harmonic Power
E6H	PmeanBHLSB	R	Lower word of phase B active harmonic Power
E7H	PmeanCHLSB	R	Lower word of phase C active harmonic Power
E9H	UrmsALSB	R	Lower word of phase A voltage RMS
EAH	UrmsBLSB	R	Lower word of phase B voltage RMS
EBH	UrmsCLSB	R	Lower word of phase C voltage RMS
EDH	IrmsALSB	R	Lower word of phase A current RMS
EEH	IrmsBLSB	R	Lower word of phase B current RMS
EFH	IrmsCLSB	R	Lower word of phase C current RMS

Fonte: *Datasheet* do M90E36A.

ANEXO C – *SOFTWARE* DO *SMART METER*

MAIN.ino

```
#include <SPI.h>
#include <SdFat.h>
#include <Wire.h>
#include "RTClib.h"

#define M90_CHIP_SELECT 2
#define M90_RESET 3
#define RED 4
#define GREEN 5
#define BLUE 6
#define CONTROL_BUTTON 9
#define SD_CHIP_SELECT SS

//M90
#define N 10
#define FACTOR 0.10
#define URES 0.0000390625
#define IRES 0.00000390625
#define POWRES 0.00390625
#define TOTALPOWRES 0.015625
#define PFRES 0.001
#define H_RES 0.006103515625
#define UF_RES 0.032656
#define IF_RES 0.0032656
#define F_RES 0.009767252604167
#define START 1
#define END 0

#define CALIBRATE_VOLTAGE 0
#define CALIBRATE_CURRENT 0

enum CHANNELS {UA, UB, UC, IA = 4, IB, IC, TOTAL = -1};
enum CHECKSUM_NUMBER {CONFIG, ENERGY, FUNDAMENTAL_HARMONIC,
MEASUREMENT};

SdFat sd;
SdFile F;

//Clock
RTC_DS1307 rtc;

// serial output steam
ArduinoOutStream cout(Serial);

// global for card size
uint32_t cardSize;

// control button previous state
boolean prevState = HIGH;

boolean blueState = HIGH;
```

```
boolean wrongWayFlag = LOW;

void setup() {
   Serial.begin(9600);
   cout << "Serial iniciada." << endl;
   pinMode(M90_CHIP_SELECT, OUTPUT);
   digitalWrite(M90_CHIP_SELECT, HIGH);
   pinMode(SD_CHIP_SELECT, OUTPUT);
   digitalWrite(SD_CHIP_SELECT, HIGH);

   pinMode(CONTROL_BUTTON, INPUT);

   if (! rtc.begin()) {
      cout << F("Falha no RTC.") << endl;
      return;
   }
   cout << "RTC Iniciado" << endl;

   if (! rtc.isrunning()) {
      cout << F("RTC não esta respondendo.") << endl;
      return;
   }
   cout << "RTC funcionando." << endl;

   if(!pmicSetup()) {
      cout << F("Falha no M90E36A.") << endl;
      return;
   }
   cout << "Medidor Inicializado." << endl;

   cout << "Calibrando..." << endl;
   measurementCalibration(START);

   #if CALIBRATE_VOLTAGE == 1
      cout << "Ganho UA: " << measurementGainCalibration(UA,225.2) << endl; //31467
      cout << "Ganho UB: " << measurementGainCalibration(UB,225.2) << endl; //31337
      cout << "Ganho UC: " << measurementGainCalibration(UC,225.2) << endl; //31379
   #else
      write16(0x61,31467); //UA Gain
      write16(0x65,31337); //UB Gain
      write16(0x69,31379); //UC Gain
   #endif

   #if CALIBRATE_CURRENT == 1
      cout << "Ganho IA: " << measurementGainCalibration(IA,6.5) << endl; //19161
      cout << "Ganho IB: " << measurementGainCalibration(IB,6.5) << endl; //19054
      cout << "Ganho IC: " << measurementGainCalibration(IC,6.5) << endl; //19148
   #else
      write16(0x62,19229); //IA Gain
      write16(0x66,19121); //IB Gain
      write16(0x6A,19216); //IC Gain
   #endif

   cout << "Resultado da calibragem: " << measurementCalibration(END) << endl;

   energyCalibration(START);

   //Compensação do erro de fase
   write16(0x48,448);
   write16(0x4A,448);
```

```
    write16(0x4C,448);
    energyCalibration(END);

    pinMode(RED, OUTPUT);
    pinMode(GREEN, OUTPUT);
    pinMode(BLUE, OUTPUT);
    digitalWrite(RED, HIGH);
    digitalWrite(GREEN, LOW);
    digitalWrite(BLUE, LOW);

    if (!sd.cardBegin(SD_CHIP_SELECT, SPI_HALF_SPEED)) {
        cout << F("Falha SD.") << endl;
        digitalWrite(RED, LOW);
        return;
    }
    cardSize = sd.card()->cardSize();

    if (cardSize == 0) {
        digitalWrite(RED, LOW);
        return;
    }
    if (!sd.fsBegin()) {
        digitalWrite(RED, LOW);
        return;
    }

    cout << "SD Inicializado." << endl;
    cout << F("Lista dos arquivos no SD:\n");
    sd.ls("/", LS_R);

    rmRfStarTest();

    prevState = digitalRead(CONTROL_BUTTON);

    delay(500);
}

//------------------------------------------------------------------------------
void loop() {
    digitalWrite(RED, LOW);
    digitalWrite(GREEN, HIGH);

    cout << "Pressione o botão para iniciar";
    cout << endl;

    while(!negedge()) delay(400);

    if(!prepareNextFile()) {
        digitalWrite(RED, HIGH);
        cout << F("Erro ao preparar arquivo.") << endl;
        return;
    }

    digitalWrite(GREEN, LOW);
    digitalWrite(BLUE, blueState);

    dftStart();
    delay(1000);

    //While principal de coleta de assinaturas
```

```
    while(!negedge()) {
      cout << "inicio: " << millis(); //apagar
      blueState = !blueState;

      logData();
      if(wrongWayFlag == HIGH) {
        digitalWrite(BLUE, LOW);
        digitalWrite(RED, blueState);
      }
      else {
        digitalWrite(BLUE, blueState);
        digitalWrite(RED, LOW);
      }

      if (!F.sync() || F.getWriteError()) {
        F.close();
        if(!prepareNextFile()) {
          digitalWrite(RED, LOW);
          cout << F("Erro ao preparar arquivo.") << endl;
          return;
        }
        cout << "Escrevendo em: ";
        F.printName();
      }
      dftStart();

      cout << "fim: " << millis(); //apagar
      delay(800);
    }
    //end

    digitalWrite(BLUE, HIGH);
    digitalWrite(RED, HIGH);
    digitalWrite(GREEN, HIGH);
    delay(1000);
    digitalWrite(BLUE, LOW);
    digitalWrite(RED, HIGH);
    digitalWrite(GREEN, LOW);

    blueState = LOW;

    F.close();
    delay(1000);
}

void writeHeader(uint32_t timestamp) {
    F.write(String(timestamp).c_str());
    F.write(" ");
}

void logData() {
    DateTime now = rtc.now();
    float AP;
    wrongWayFlag = LOW;

    writeHeader(now.unixtime()); // 1
    F.write(String(activePower(TOTAL)).c_str()); // 2
    F.write(" ");
    F.write(String(activeFundamentalPower(TOTAL)).c_str()); // 3
    F.write(" ");
```

```
F.write(String(activeHarmonicPower(TOTAL)).c_str()); // 4
F.write(" ");
AP = activePower(UA);
if(AP < -5) wrongWayFlag = HIGH;
F.write(String(AP).c_str()); // 5
F.write(" ");
F.write(String(activeFundamentalPower(UA)).c_str()); // 6
F.write(" ");
F.write(String(activeHarmonicPower(UA)).c_str()); // 7
F.write(" ");
AP = activePower(UB);
if(AP < -5) wrongWayFlag = HIGH;
F.write(String(AP).c_str()); // 8
F.write(" ");
F.write(String(activeFundamentalPower(UB)).c_str()); // 9
F.write(" ");
F.write(String(activeHarmonicPower(UB)).c_str()); // 10
F.write(" ");
AP = activePower(UC);
if(AP < -5) wrongWayFlag = HIGH;
F.write(String(AP).c_str()); // 11
F.write(" ");
F.write(String(activeFundamentalPower(UC)).c_str()); // 12
F.write(" ");
F.write(String(activeHarmonicPower(UC)).c_str()); // 13
F.write(" ");

//REACTIVE POWER
F.write(String(reactivePower(TOTAL)).c_str()); // 14
F.write(" ");
F.write(String(reactivePower(UA)).c_str()); // 15
F.write(" ");
F.write(String(reactivePower(UB)).c_str()); // 16
F.write(" ");
F.write(String(reactivePower(UC)).c_str()); // 17
F.write(" ");

//APARENT POWER
F.write(String(aparentPower(TOTAL)).c_str()); // 18
F.write(" ");
F.write(String(aparentPower(UA)).c_str()); // 19
F.write(" ");
F.write(String(aparentPower(UB)).c_str()); // 20
F.write(" ");
F.write(String(aparentPower(UC)).c_str()); // 21
F.write(" ");

//POWER FACTOR
F.write(String(powerFactor(TOTAL)).c_str()); // 22
F.write(" ");
F.write(String(powerFactor(UA)).c_str()); // 23
F.write(" ");
F.write(String(powerFactor(UB)).c_str()); // 24
F.write(" ");
F.write(String(powerFactor(UC)).c_str()); // 25
F.write(" ");

//VOLTAGE RMS
F.write(String(realRms(UA)).c_str()); // 26
F.write(" ");
```

```
F.write(String(fundamental(UA)*UF_RES).c_str()); // 27
F.write(" ");
F.write(String(harmonic(UA,2)*H_RES).c_str()); // 28
F.write(" ");
F.write(String(harmonic(UA,3)*H_RES).c_str()); // 29
F.write(" ");
F.write(String(harmonic(UA,4)*H_RES).c_str()); // 30
F.write(" ");
F.write(String(harmonic(UA,5)*H_RES).c_str()); // 31
F.write(" ");
F.write(String(realRms(UB)).c_str()); // 32
F.write(" ");
F.write(String(fundamental(UB)*UF_RES).c_str()); // 33
F.write(" ");
F.write(String(harmonic(UB,2)*H_RES).c_str()); // 34
F.write(" ");
F.write(String(harmonic(UB,3)*H_RES).c_str()); // 35
F.write(" ");
F.write(String(harmonic(UB,4)*H_RES).c_str()); // 36
F.write(" ");
F.write(String(harmonic(UB,5)*H_RES).c_str()); // 37
F.write(" ");
F.write(String(realRms(UC)).c_str()); // 38
F.write(" ");
F.write(String(fundamental(UC)*UF_RES).c_str()); // 39
F.write(" ");
F.write(String(harmonic(UC,2)*H_RES).c_str()); // 40
F.write(" ");
F.write(String(harmonic(UC,3)*H_RES).c_str()); // 41
F.write(" ");
F.write(String(harmonic(UC,4)*H_RES).c_str()); // 42
F.write(" ");
F.write(String(harmonic(UC,5)*H_RES).c_str()); // 43
F.write(" ");

//CURRENT RMS
F.write(String(realRms(IA)).c_str()); // 44
F.write(" ");
F.write(String(fundamental(IA)*IF_RES/FACTOR).c_str()); // 45
F.write(" ");
F.write(String(harmonic(IA,2)*H_RES).c_str()); // 46
F.write(" ");
F.write(String(harmonic(IA,3)*H_RES).c_str()); // 47
F.write(" ");
F.write(String(harmonic(IA,4)*H_RES).c_str()); // 48
F.write(" ");
F.write(String(harmonic(IA,5)*H_RES).c_str()); // 49
F.write(" ");
F.write(String(realRms(IB)).c_str()); // 50
F.write(" ");
F.write(String(fundamental(IB)*IF_RES/FACTOR).c_str()); // 51
F.write(" ");
F.write(String(harmonic(IB,2)*H_RES).c_str()); // 52
F.write(" ");
F.write(String(harmonic(IB,3)*H_RES).c_str()); // 53
F.write(" ");
F.write(String(harmonic(IB,4)*H_RES).c_str()); // 54
F.write(" ");
F.write(String(harmonic(IB,5)*H_RES).c_str()); // 55
F.write(" ");
```

```
    F.write(String(realRms(IC)).c_str()); // 56
    F.write(" ");
    F.write(String(fundamental(IC)*IF_RES/FACTOR).c_str()); // 57
    F.write(" ");
    F.write(String(harmonic(IC,2)*H_RES).c_str()); // 58
    F.write(" ");
    F.write(String(harmonic(IC,3)*H_RES).c_str()); // 59
    F.write(" ");
    F.write(String(harmonic(IC,4)*H_RES).c_str()); // 60
    F.write(" ");
    F.write(String(harmonic(IC,5)*H_RES).c_str()); // 61
    F.write(" ");
    F.write(String(neutralCurrentRms()).c_str()); // 62
    F.write(" ");

    //FREQUENCY
    F.write(String(frequency()).c_str()); // 63
    F.println();
    return;
}

boolean negedge() {
    int count = 0;
    boolean currentState;

    for(int i=0;i<10;i++) {
        currentState = digitalRead(CONTROL_BUTTON);
        if(currentState == HIGH)
            count++;
        else count--;
        delay(10);
    }

    if(count<=0)
        currentState = LOW;
    else currentState = HIGH;

    if(currentState != prevState) { //edge detected
        prevState = currentState;
        if(currentState == LOW) //negedge test
            return true;
        else return false;
    }
    return false;
}

boolean prepareNextFile() {
    DateTime now = rtc.now();
    String fileName = String(now.unixtime()) + ".log";

    if (!F.open(fileName.c_str(), O_CREAT | O_WRITE | O_EXCL)) {
        cout << F("error file.open") << endl;
        return false;
    }
    return true;
}

void rmRfStarTest() {
    int8_t cnt = 0;
```

```
    digitalWrite(GREEN, HIGH);
    digitalWrite(BLUE, HIGH);

    while(!digitalRead(CONTROL_BUTTON)) {
      delay(100);
      cnt++;
      if(cnt == 10) {
        cout << F("Erasing SD Card...") << endl;
        sd.vwd()->rmRfStar();
        break;
      }
    }

    digitalWrite(GREEN, LOW);
    digitalWrite(BLUE, LOW);

    return;
}
```

M90E36A.ino

```
uint16_t measurementGainCalibration(int channel, float ref) {
    uint8_t i;
    uint16_t gain;
    float meas, attenuatedRef;

    for(meas=0,i=0;i<N;i++) {
      meas += rms(channel);
      delay(500);
    }

    cout << "Valor medido (SOMATORIO): " << meas;
    meas = meas/N;
    cout << "Valor medido: " << meas;
    cout << "Valor referencia: " << ref;
    switch(channel) {
      case UA: case UB: case UC:
        meas = meas*URES;
        gain = 52800*ref/meas;

        cout << meas << " " << ref << " " << gain << endl;
        write16(0x61 + channel*4, gain);
        break;
      case IA: case IB: case IC:
        meas = meas*IRES;
        attenuatedRef = ref*FACTOR;

        gain = 30000*attenuatedRef/meas;

        cout << meas << " " << attenuatedRef << " " << gain << endl;
        write16(0x62 + (channel-4)*4, gain);
        break;
    }

    return gain;
}

bool measurementCalibration(bool opt) {
    switch(opt) {
      case 1:
```

```
            write16(0x60, 0x5678);
            return 1;
        case 0:
            write16(0x6F, csCalculator(0x61, 0x6E));
            write16(0x60, 0x8765);
            return !checksumStatus(MEASUREMENT);
    }
}

bool energyCalibration(bool opt) {
    switch(opt) {
        case 1:
            write16(0x40, 0x5678);
            return 1;
        case 0:
            write16(0x4D, csCalculator(0x41, 0x4C));
            write16(0x40, 0x8765);
            return !checksumStatus(ENERGY);
    }
}

bool pmicSetup() {
    SPI.begin();
    pinMode(M90_CHIP_SELECT, OUTPUT);
    digitalWrite(M90_CHIP_SELECT, HIGH);
    pinMode(M90_RESET, OUTPUT);
    digitalWrite(M90_RESET, LOW);
    delay(20);
    digitalWrite(M90_RESET, HIGH);
    delay(400);
    //Software Reset
    write16(0x00, 0x789A);
    delay(100);

    //Configuration
    write16(0x30, 0x5678);
    write16(0x33, 0x1087);
    write16(0x3B, csCalculator(0x31, 0x3A));
    write16(0x30, 0x8765);

    delay(1000);
    if(read16(0x33) != 0x1087) return false;
    return !checksumStatus(CONFIG);
}

void dftStart() {
    write16(0x1D1, 0x0001);
    return;
}

uint16_t csCalculator(byte startRegister, byte endRegister) {
    int i;
    byte lowCsByte = 0, highCsByte = 0;
    uint16_t aux;

    for(i=startRegister; i<=endRegister; i++) {
        aux = read16(i);
        lowCsByte += (byte)(aux>>8) + (byte)(aux&(0x00FF));
        highCsByte = highCsByte ^ (byte)(aux>>8) ^ (byte)(aux&(0x00FF));
    }
```

```
    return ((uint16_t)lowCsByte) + (((uint16_t)highCsByte)<<8);
}

bool checksumStatus(int number) {
    switch(number) {
        case CONFIG:
            return bitRead(read16(0x01),14);
        case ENERGY:
            return bitRead(read16(0x01),12);
        case FUNDAMENTAL_HARMONIC:
            return bitRead(read16(0x01),10);
        case MEASUREMENT:
            return bitRead(read16(0x01),8);
    }
}

uint16_t read16(uint16_t address) {
    uint16_t readData;

    SPI.beginTransaction(SPISettings(1000000, MSBFIRST, SPI_MODE3));
    digitalWrite(M90_CHIP_SELECT, LOW);

    SPI.transfer16(0x8000 | address);
    readData = SPI.transfer16(0x0000);

    digitalWrite(M90_CHIP_SELECT, HIGH);
    SPI.endTransaction();

    delay(1);
    return readData;
}

void write16(uint16_t address, uint16_t data) {
    SPI.beginTransaction(SPISettings(1000000, MSBFIRST, SPI_MODE3));
    digitalWrite(M90_CHIP_SELECT, LOW);

    SPI.transfer16(address);
    SPI.transfer16(data);

    digitalWrite(M90_CHIP_SELECT, HIGH);
    SPI.endTransaction();
    delay(1);
    return;
}

float frequency() {
    return read16(0xF8)*F_RES;
}

uint32_t rms(int8_t channel) {
    return (((uint32_t)read16(0xD9+channel))<<8) + (read16(0xE9+channel)>>8);
}

float neutralCurrentRms() {
    uint32_t readData = 0;
    readData = ((uint32_t)read16(0xDC))<<8;

    return readData*IRES/FACTOR;
}
```

```
float realRms(int8_t channel) {
   switch(channel) {
      case UA: case UB: case UC:
         return rms(channel)*URES;
      case IA: case IB: case IC:
         return rms(channel)*IRES/FACTOR;
   }
}

uint16_t harmonic(int8_t channel, int8_t order) {
   if(order < 2 || order > 32)
      return -1;
   else {
      switch(channel) {
         case UA: case UB: case UC:
            return read16(0x160+channel*0x20+(order-2));
         case IA: case IB: case IC:
            return read16(0x100+(channel-4)*0x20+(order-2));
      }
   }
}

float activeHarmonicPower(int8_t channel) {
   uint32_t readData = 0;
   float ahp;
   readData = (((uint32_t)read16(0xD5+channel))<<8) + (read16(0xE5+channel)>>8);
   ahp = twosComplementToDecimal(readData,24);

   return (channel==TOTAL)?(ahp*TOTALPOWRES/FACTOR):(ahp*POWRES/FACTOR);
}

float activeFundamentalPower(int8_t channel) {
   uint32_t readData = 0;
   float afp;
   readData = (((uint32_t)read16(0xD1+channel))<<8) + (read16(0xE1+channel)>>8);
   afp = twosComplementToDecimal(readData,24);

   return (channel==TOTAL)?(afp*TOTALPOWRES/FACTOR):(afp*POWRES/FACTOR);
}

uint16_t fundamental(int8_t channel) {
   switch(channel) {
      case UA: case UB: case UC:
         return read16(0x1C1+channel*2);
      case IA: case IB: case IC:
         return read16(0x1C0+(channel-4)*2);
   }
}

float activePower(int8_t channel) {
   uint32_t readData = 0;
   float ap;
   readData = (((uint32_t)read16(0xB0+channel+1))<<8) + (read16(0xC0+channel+1)>>8);
   ap = twosComplementToDecimal(readData,24);

   return (channel==TOTAL)?(ap*TOTALPOWRES/FACTOR):(ap*POWRES/FACTOR);
}

float reactivePower(int8_t channel) {
```

```
    uint32_t readData = 0;
    float rp;
    readData = (((uint32_t)read16(0xB4+channel+1))<<8) + (read16(0xC4+channel+1)>>8);
    rp = twosComplementToDecimal(readData,24);

    return (channel==TOTAL)?(rp*TOTALPOWRES/FACTOR):(rp*POWRES/FACTOR);
}

float aparentPower(int8_t channel) {
    float ap;
    ap = (((uint32_t)read16(0xB8+channel+1))<<8) + (read16(0xC8+channel+1)>>8);
    return (channel==TOTAL)?(ap*TOTALPOWRES/FACTOR):(ap*POWRES/FACTOR);
}

float powerFactor(int8_t channel) {
    float pf;
    pf = twosComplementToDecimal(read16(0xBC+channel+1), 16)*PFRES;
    return pf;
}

int32_t twosComplementToDecimal(uint32_t complement, const int Nbits) {
    int32_t output;
    output = -1;
    output *= (int32_t)bitRead(complement,Nbits-1);
    output *= powOfTwo(Nbits-1);

    for(int8_t i=0; i<=Nbits-2;i++) {
        output += ((int32_t)bitRead(complement, i))*powOfTwo(i);
    }

    return output;
}

int32_t powOfTwo(int8_t expoent) {
    return ((int32_t)1)<<expoent;
}
```