SISTEMA DE MONITORAMENTO DE CONSUMO ELÉTRICO DE UMA MINERADORA DE CRIPTOMOEDAS VIA SUPERVISÓRIO

CORREA, Lucas Gabriel[1]
DAL'MAGRO, Fernando Luiz[2]
FILHO, Carlos Roberto da Silva[3]
ANDRADE, Solange Alves C.[4]

RESUMO

Este artigo tem por finalidade apresentar as primeiras etapas no desenvolvimento de um dispositivo capaz de medir os dados de corrente elétrica e tensão e convertê-los nos valores de potência consumida. Estes valores são referentes ao processo de mineração de criptomoedas realizado por uma mineradora de pequeno porte e serão usados para gerar os valores, em Reais, gastos neste processo. O equipamento usa de sensores para a captação dos dados e o uso de um microcontrolador para geri-los e os enviar a um banco de dados desenvolvido pelos autores em *SQL Server*, onde os mesmos serão computados e enviados à um sistema supervisório desenvolvido em linguagem de programação *C#*, dando assim, fácil acesso a estas informações para o usuário da mineradora. Por meio dos resultados que serão exibidos pelo sistema supervisório, o minerador poderá ter um maior controle do lucro real que obteve através da mineração de criptomoedas, haja visto que o usuário terá acesso em tempo real do quanto há sido gasto em tal processo e poderá com base nisso, se posteriormente assim o desejar, mudar o tipo de criptomoedas por ele minerada.

Palavras-chave: Criptomoedas; Mineração; Supervisório; Monitoramento; Gastos.

Lucas Gabriel Correa[1] do Curso de Engenharia de Controle e Automação do Centro Universitário UNISOCIESC, lucasgccorrea@hotmail.com; Fernando Luiz Dal'magro[2] do Curso de Engenharia de Controle e Automação do Centro Universitário UNISOCIESC, fernandoluizdalmagro@gmail.com;
Carlos Roberto da Silva Filho[3]: Mestre, Centro Universitário UNISOCIESC, carlos.silva@unisociesc.com;
Solange Alves C. Andrade[4]: Mestra, Centro Universitário UNISOCIESC, solange@unisociesc.com.br

Joinville – SC, 18 de Junho de 2021.

ABSTRACT

This article presents the first steps in the development of a device capable of measuring electrical current and voltage data and converting them into consumed power values. These amounts refer to the cryptocurrency mining process carried out by a small mining company and will be used to generate the amounts, in reais, spent in this process. The equipment uses sensors to capture data and the use of a microcontroller to manage them and send them to a database developed by the authors in SQL Server, where they will be computed and sent to a supervisory system developed in programming language C#, thus giving easy access to this information for the mining company user. Through the results that will be chosen by the supervisory system, the miner can have greater control of the real profit obtained through the mining of cryptocurrencies, given that the user will have real-time access to how much has been spent on such process and can based on in this, if you later wish, change the type of cryptocurrencies mined by it.

Keywords*: Cryptocurrencies; Mining; Supervisory; Monitoring; Spending.*

1 INTRODUÇÃO

Em 2008, Satoshi Nakamoto, pseudônimo de um programador desconhecido, reinventou a percepção que o mundo dava para a moeda, transformando-a em código de computador. Assim nascia o Bitcoin, a primeira e mais conhecida criptomoeda. Uma moeda digital de código aberto que não depende de uma plataforma de confiança, como a Paypal ou a Mastercard, para fazer o registro das transações online. Após o sucesso da bitcoin, várias outras moedas surgiram e surgem com frequência. Segundo o Clib, uma plataforma que exibe informações sobre criptomoedas, existem mais de 6 mil criptomoedas atualmente.

Há duas formas de adquirir uma criptomoeda, a primeira é através de bolsas de valores conhecidas como exchanges. A segunda forma é através da mineração de criptomoedas, que é como as novas moedas são criadas, meio este que requer um processamento muito alto, realizado por computadores potentes. Usando a bitcoin como exemplo, o computador precisa realizar milhares de cálculos por segundo na tentativa de encontrar a combinação perfeita de uma sequência de informações que torne o bloco de transações de bitcoin compatível com o bloco anterior já existente, sendo que cada bloco possui uma "digital" chamada *hash* que impede que cada bloco encontrado seja duplicado. Ao encontrar a sequência, o minerador recebe uma recompensa em bitcoin para cada bloco que ele minerar.

Como a mineração de criptomoedas funciona através de cálculos complexos realizadas por computadores, os mesmos na maioria dos casos ficam trabalhando 24 horas por dia em potência máxima e somado às técnicas de refrigeração dos componentes que podem ser empregadas, a quantidade de energia elétrica gasta para tal fim pode chegar a valores exorbitantes, muitas vezes inviabilizando financeiramente a prática da mineração.

Segundo o Centro de Finanças Alternativas da Universidade de Cambridge (CCAF), o consumo estimado total de energia gasto, em março de 2021, apenas com o bitcoin está entre 40 e 470 terawatts-hora (TWh) por ano, com uma estimativa média de cerca de 135 terawatts hora. Segundo a Agência de Energia Internacional (IEA) em 2019, a geração de energia mundial corresponde a 25.721 TWh, sendo assim, o consumo na mineração de criptomoedas no cenário atual podendo corresponder a mais de 1,8% da produção mundial no ano de 2019, ou 73,8% da produção do Brasil no mesmo período.

Este projeto visa elaborar um equipamento que possa mensurar o gasto energético de uma mineradora de criptomoedas através de um microprocessador PIC, de modo que os dados sejam disponibilizados via comunicação RS-232 para que possam assim, ser visualizados em um supervisório desenvolvido em C#. Os dados energéticos serão coletados através de um sensor de corrente e tensão acoplados em uma pequena mineradora, a qual utiliza um *desktop* caseiro e de baixo poder de processamento se comparado com equipamentos específicos para o processo, sendo assim, o processo e dados serão limitados apenas ao uso desta mineradora. Com os dados em mãos o usuário poderá verificar os reais ganhos de forma objetiva e clara, pois terá a informação do gasto energético em tempo real.

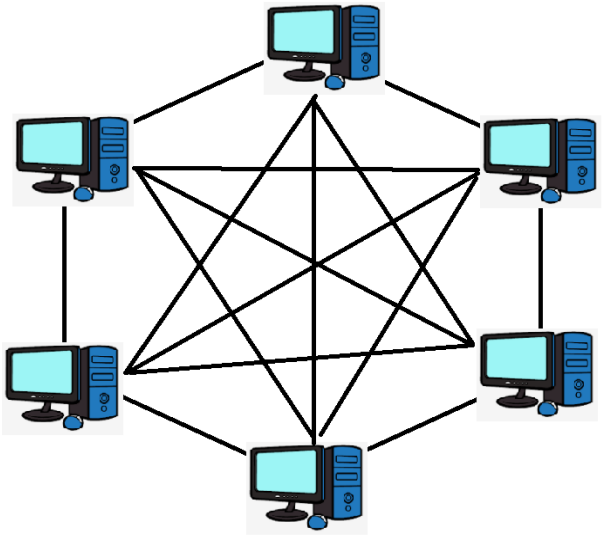
Os principais objetivos específicos do trabalho são constituídos dos seguintes:

a) simulação da placa eletrônica proposta;
b) realizar programação dos microprocessadores via simuladores;
c) elaboração da tela C# para supervisório;
d) criação do banco de dados em SQL Server;
e) desenvolvimento e montagem do dispositivo;
f) análise e testes do sistema de monitoramento;
g) desenvolvimento de um tutorial de como criar um supervisório C# com PIC via RS-232.

2 REVISÃO DE LITERATURA

As criptomoedas são um meio de troca criptografado inteiramente virtual, baseado em uma rede *peer-to-peer* (P2P) onde cada usuário da rede funciona tanto como cliente quanto servidor, podendo enviar ou receber dados de, e para todos, os computadores ligados a rede, não existindo assim, um servidor central ou ponto de controle (Figura 1). Elas utilizam um sistema chamado *blockchain*, um banco de dados sob a forma de livro-razão, que está em posse de todos os usuários do sistema e cada transição de dados deve ser aceita e confirmada por todos os usuários (ANTONOPOULOS, 2014).

Figura 1 - Conexão *peer-to-peer*



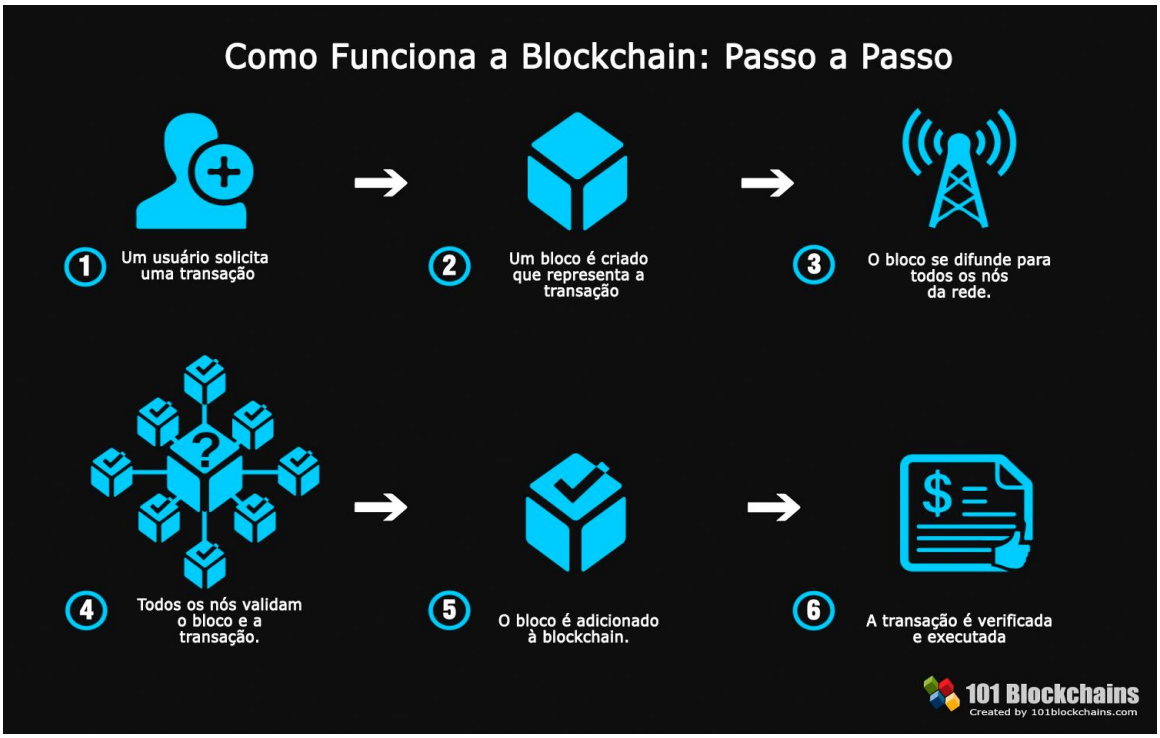
Fonte: Autores (2021)

A *blockchain* é uma cadeia de blocos conectadas por funções *hash*. Estas, são funções matemáticas específicas e calculáveis a partir de um valor de entrada, porém, praticamente impossíveis de encontrar o valor de entrada a partir do valor de saída da função (NARAYANAN *et al*, 2016). Tal característica faz da *blockchain* imutável, uma vez que seus dados são submetidos a esta função, pois necessita de um esforço computacional mínimo para adição de um novo bloco, esforço este, que cresce exponencialmente a cada bloco (AGNER, 2016). Com isso, ao longo do tempo é necessário cada vez mais poder computacional para minerar na mesma taxa e manter a lucratividade estável (KUGLER, 2018).

Cada minerador é um nó da rede *blockchain*, interligado a todos os outros nós da rede. A rede tem um parâmetro de dificuldade que indica o *hash* do bloco esperado por ela, sendo que esta dificuldade é ajustada periodicamente no intuito de manter o tempo médio levado para que algum minerador consiga realizar os cálculos que gere um novo bloco. Cada *hash* tem um número de zeros mínimo para que seja aceito na atualização da *blockchain* e o minerador precisa criar um que representa um número menor ou igual ao bloco atual, isto faz com que a cada nova atualização de bloco o poder computacional necessário para o cálculo seja maior, já que o valor vai diminuindo. O primeiro minerador que criar um *hash* novo recebe uma bonificação X na forma de novas criptomoedas (AGNER, 2016).

Para uma melhor compreensão do conceito citado acima a Figura 2 ilustra o passo a passo do funcionamento de uma rede *blockchain.*

Figura 2 - Funcionamento da *blockchain*



Fonte: 101 Blockchains (2018)

Como os cálculos de mineração geram gastos energéticos consideráveis, o uso de um sistema embarcado, usando um microcontrolador e sensores para monitoramento da energia gasta, podem ser empregados na mineradora. Microcontroladores são pequenos dispositivos “inteligentes”, como eles possuem suas dimensões reduzidas e uma alta capacidade de integração, isso faz com que estejam altamente presentes em equipamentos digitais. Microcontroladores são constituídos basicamente de uma CPU (*Central Processing Unit*), periféricos como, por exemplo, portas SPI e USART, e também de memória, para programa e dados (MIYADARA, 2013).

Para a medição de corrente, existem vários tipos de sensores, como, por exemplo, o Resistor *Shunt* e o Sensor de efeito *Hall*. O sensor do tipo resistor *Shunt* consiste em colocar um resistor de baixa resistência em paralelo com a carga, então é feita a leitura da queda de tensão entre os terminais do dispositivo. O sensor do tipo efeito *HALL*, consiste em dispositivos semicondutores que, através de um campo eletromagnético, sofrem influência, a corrente do circuito passa por uma placa condutora, gerando um

campo magnético, tendo então uma diferença de potencial gerada nas laterais da placa condutora (THOMAZINI; ALBUQUERQUE, 2011).

Para medição de tensão, existem várias formas, sendo uma delas para corrente contínua, utilizar um divisor de tensão resistivo e para corrente alternada, utilizar um divisor resistivo e capacitivo, ou o transformador de potencial. Para utilizar resistores divisores de tensão, temos dois resistores ligados em série, com a tensão a ser medida aplicada nos dois, então, é feita a medida da tensão em apenas um dos resistores, tendo então a amostragem da proporção da razão entre os resistores. O transformador de potencial é um transformador cujo objetivo é diminuir a tensão para a medição/controle, tendo geralmente a saída no secundário de 0 a 12 V, por exemplo, podendo ter de entrada no primário 220 V (THOMAZINI; ALBUQUERQUE, 2011).

O módulo USART (*Universal Synchronous Asynchronous Receiver Transmitter)* é um módulo de I/O presente no microcontrolador, e também pode ser conhecido como SCI (*Serial Communications Interface)*. Ele pode ser configurado com *full-duplex* ou *half-duplex*, sendo que a primeira é uma configuração assíncrona que pode comunicar-se com periféricos, tal como um computador pessoal. Já a segunda opção representa uma configuração síncrona que se comunica com periféricos tais como circuitos integrados e seriais EEPROMs (PALLONES, 2016).

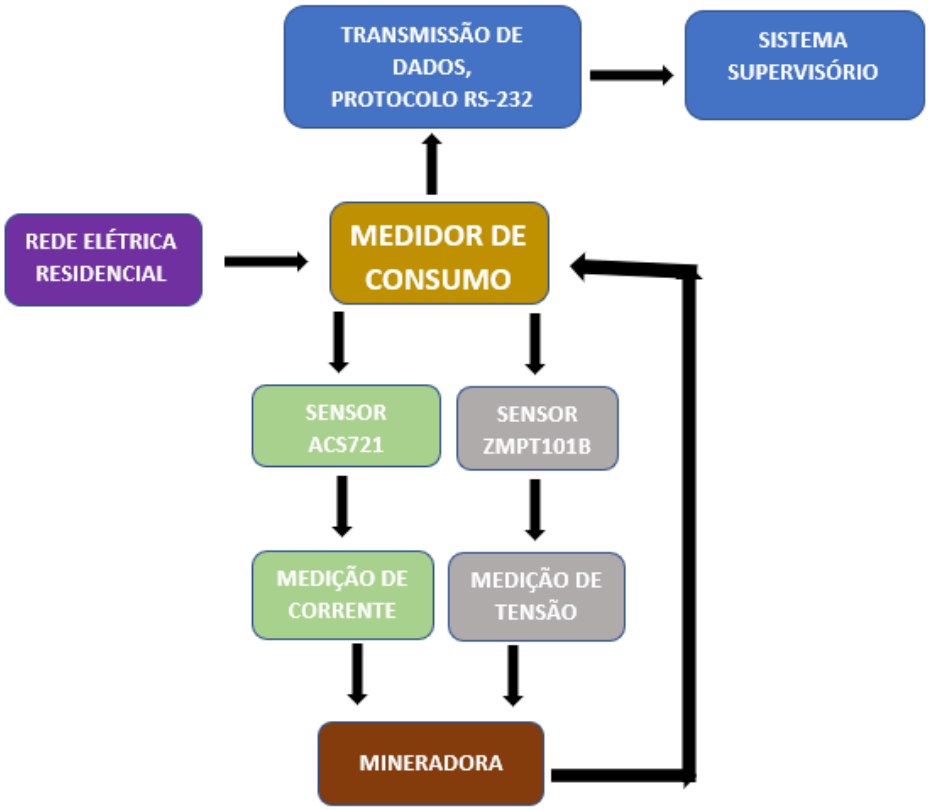
Para que ocorra o monitoramento de variáveis a grandes distâncias, se utiliza na indústria os sistemas supervisórios ou SCADA. Obrigatoriamente é necessário existir uma IHM (interface homem-máquina), onde geralmente fica localizado em uma ou mais centrais de operação, onde pode-se enxergar o status atual do processo (JUNIOR, 2019). Os sistemas de controle de supervisão e aquisição de dados, ou do inglês *Supervisory Control and Data Acquisition* (SCADA), é um sistema que supervisiona uma planta ou processo através de um terminal remoto. Utilizando uma IHM pode-se obter informações em tempo real, permitindo assim, uma melhor tomada de decisões a fim de melhorar os procedimentos realizados. Possuindo tanto conexão com a internet, bem como com redes corporativas, o sistema SCADA permite o gerenciamento de uma ampla gama de dados (MASEDA *et al,* 2021).

3 DESENVOLVIMENTO

O objetivo principal do projeto é desenvolver um dispositivo capaz de mensurar o consumo de energia elétrica para que seja utilizado em uma mineradora de

criptomoedas, realizando cálculos referentes ao consumo energético e enviando os resultados para um sistema supervisório de fácil acesso para o usuário. Como a mineradora em questão é pequena e situada em uma residência doméstica, o dispositivo deverá ter um tamanho que atenda às necessidades do projeto sem que comprometa o ambiente ao redor e que seja de fácil instalação.

Figura 3 - Visão geral do projeto



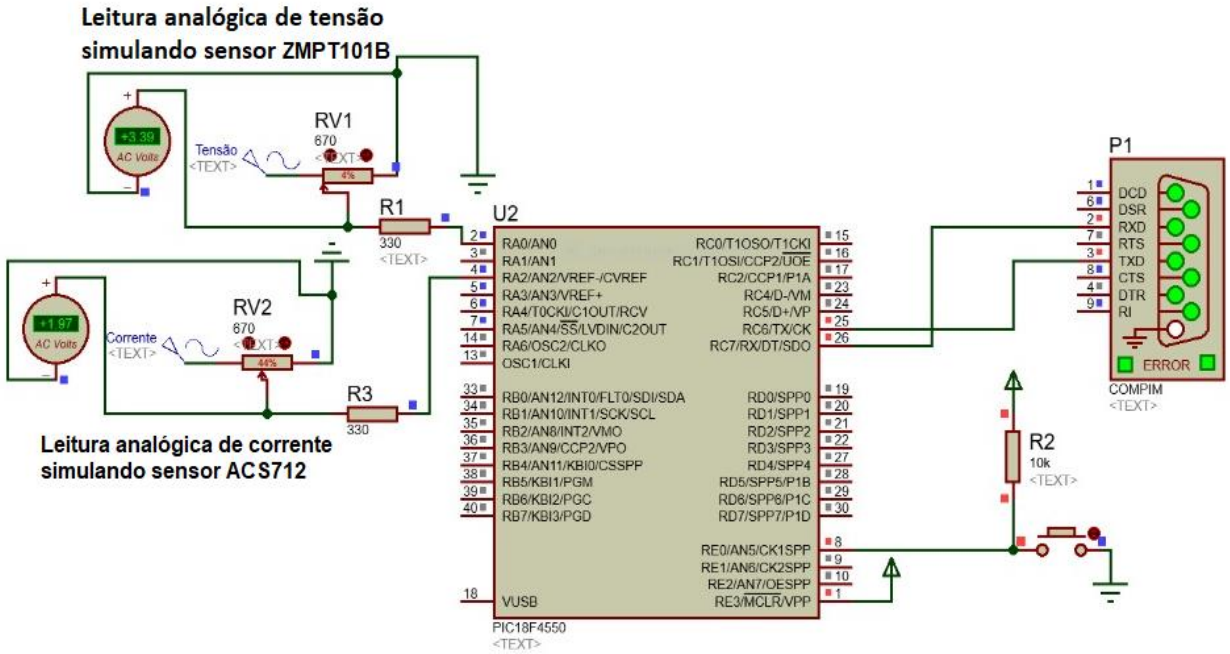
Fonte: Autores (2021)

3.1 SIMULAÇÃO DA TENSÃO, CORRENTE E ENVIO DE DADOS

De forma a garantir um desenvolvimento sem os componentes físicos, a equipe desenvolveu a parte física de forma simulada, tentando garantir uma boa base para aplicação física futuramente.

Para a simulação dos componentes, tal como leitura, conversão analógico-digital e envio de dados via RS-232 para o supervisório, foi utilizado o software PROTEUS, desenvolvido pela *Labcenter Electronics Ltd* como demonstra a Figura 4.

Figura 4 – Simulação da leitura de tensão, corrente e envio de dados



Fonte: Autores (2021)

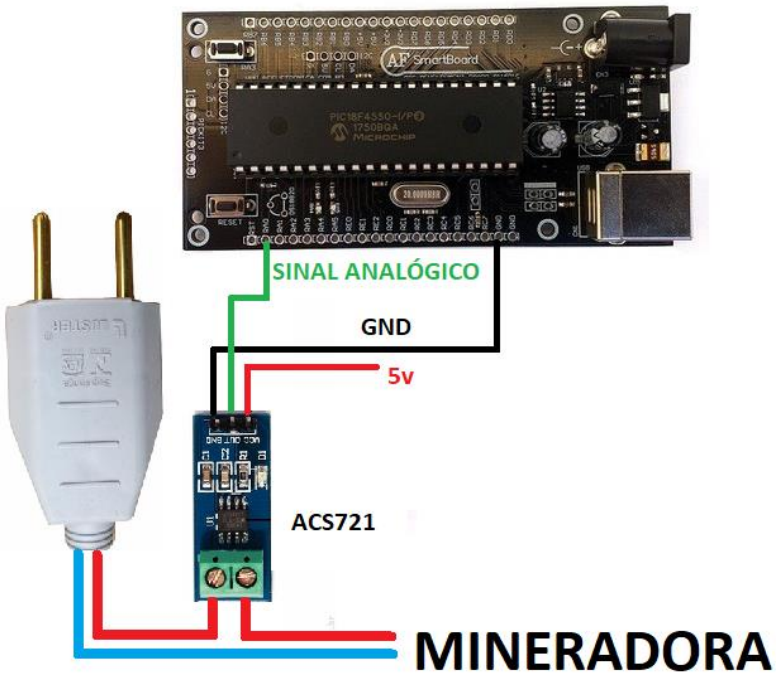
De modo a simular os sensores em uma rede de corrente alternada, foi inserido um sinal senoidal com frequência de 60Hz com sua tensão variando de 0 a 5 Volts, seguidos de potenciômetro, para poder simular como se fosse acontecer realmente na aplicação física.

3.1.1 MEDIÇÃO DE CORRENTE

Para a medição de corrente optou-se pelo sensor ACS712, que é um módulo desenvolvido para a medição de correntes tanto AC quanto DC baseado no Efeito *Hall*, com modelos variando nas faixas de ± 5 A, ± 20 A e ± 30 A.

Podendo trabalhar em temperaturas de -40 °C a 85 °C e com uma precisão de saída de 66 a 185 mV/A, o ACS712 produz tensão analógica de 0 a 5 V o que a torna uma ótima opção ao trabalhar com microcontroladores como PIC, AMR, ATmega, etc. Ele possui um condutor de cobre interno perto da superfície, sendo que quando uma corrente aplicada flui através deste cobre é gerado um campo magnético que é detectado pelo transdutor *Hall* e convertido em uma tensão proporcional na saída. O módulo foi utilizado com uma entrada analógica no microcontrolador (Figura 5), juntamente com uma programação para fazer a tratativa de dados e a conversão de sinal analógico recebido na saída do sensor para um sinal digital.

Figura 5 – Conexão do sensor ACS712 com o PIC18F

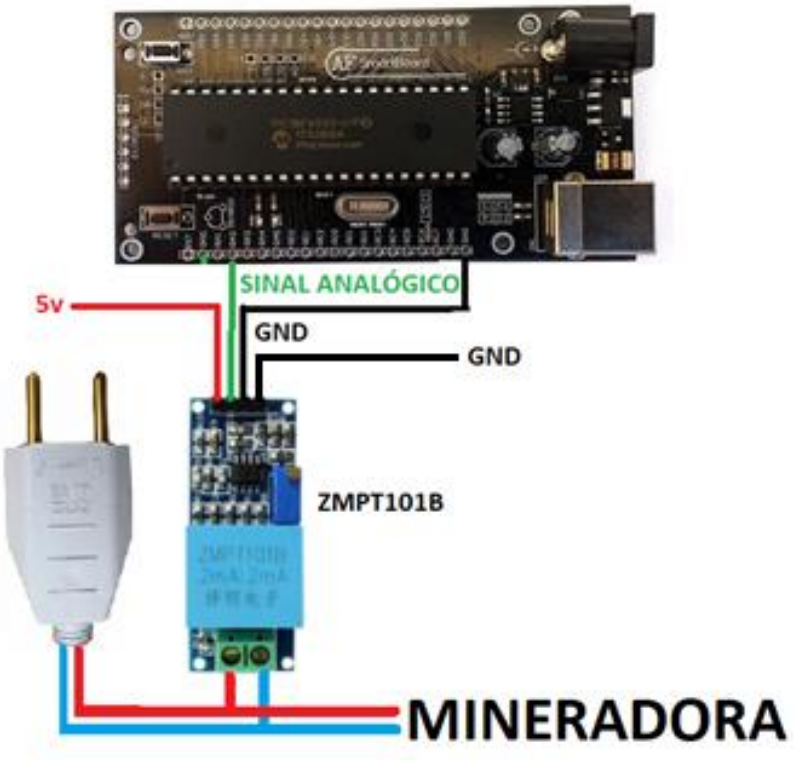


Fonte: Autores (2021)

3.1.2 MEDIÇÃO DE TENSÃO

O ZMPT101B é um transformador de tensão de micro precisão, utilizado para medir tensões AC de 0 a 250 V com frequências na faixa de 50/60 Hz. Ele possui uma relação de tensão de 1:1 entre os enrolamentos primário e secundário, e possui uma corrente de 2 mA em ambos os enrolamentos. Trabalhando em temperaturas que variam de -40 a 70 °C o seu conjunto de componentes permite uma leitura precisa, sendo que sua saída é, no caso deste projeto, um sinal senoidal variando de 0 a 5 V e com um offset de 2,5 V. O módulo escolhido foi acoplado a uma das entradas analógicas do microcontrolador, conforme ilustra a Figura 6, e após realizar as leituras de tensão da mineradora os dados serão enviados para o microcontrolador para que sejam tratados no mesmo.

Figura 6 – Conexão do sensor ZMPT101B com o PIC18F



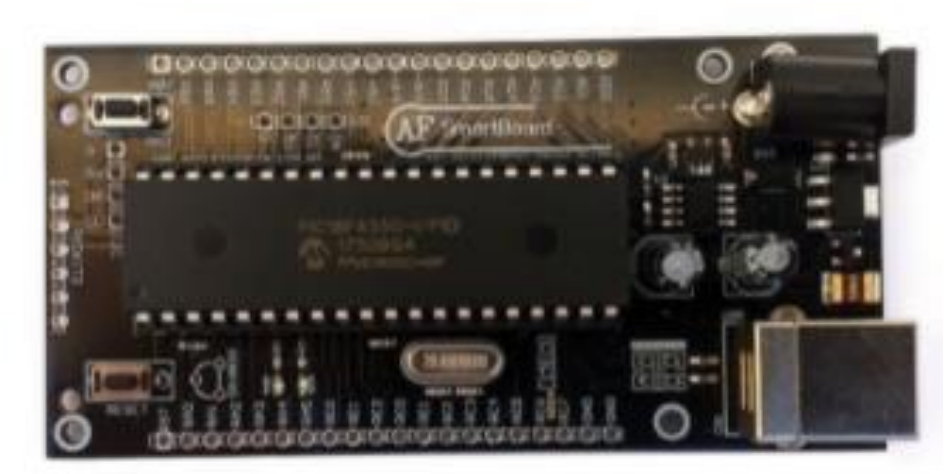
Fonte: Autores (2021)

3.1.3 MICROCONTROLADOR PIC18F4550

O PIC18f4550 (Figura 7) foi o microcontrolador escolhido para a realização deste projeto, pois, além de atender as necessidades do mesmo, foi um componente visto em sala de aula. Ele é um microcontrolador de 8 bits, dotado de 32 Kbytes de memória de programa e 2,048 bytes de memória RAM. Sua alimentação pode variar entre 4 V e 5,5 V, além de conseguir operar em frequências de até 48 MHz e possui também, um oscilador interno de 8 MHz. Ele possui 40 pinos, sendo que deles, 35 podem ser configurados como I/O e periféricos como memória EEPROM de 256 bytes, conversores A/D com tempo de aquisição programável, comparadores analógicos, comunicação USART, TIMERs e um módulo USB.

Com o uso do PIC18F4550 com o compilador XC8 e na plataforma MPLAB, da própria *Microchip*, é possível ler os valores analógicos recebidos dos sensores ACS712 e ZMPT101B, realizar a conversão analógico-digital e posteriormente tratar estes dados da forma correta para enviá-los ao sistema supervisório via protocolo RS-232.

Figura 7 - PIC18F4550



Fonte: Afeeletrônica (2021)

3.2 PROGRAMAÇÂO MICROCONTROLADOR

O diagrama de fluxo da Figura 8 ilustra a lógica de programação utilizada no microcontrolador.

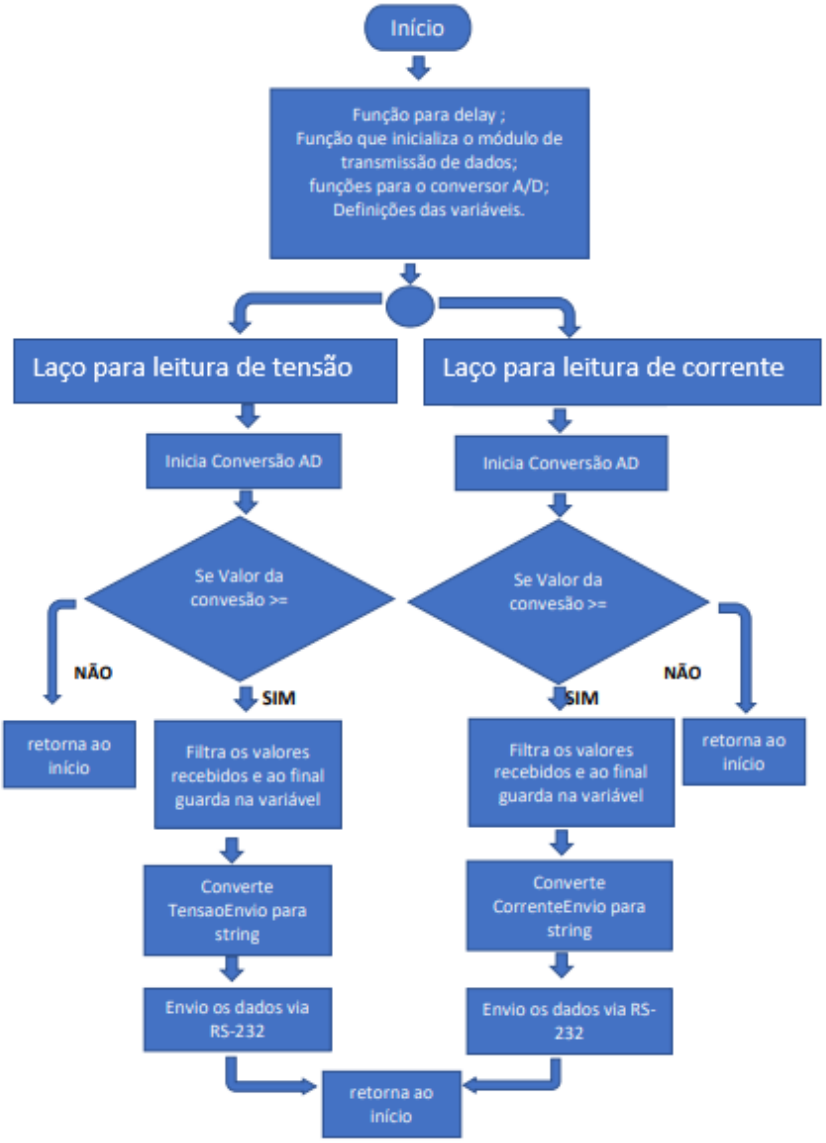
O microcontrolador foi configurado para ter um oscilador interno de oito milhões de *Hertz*, e com esse oscilador, foi configurado para ter uma comunicação com 9600 *Baud Rates.*

Para garantir uma programação mais simplificada, foi utilizado uma função separada, que ao ser chamada e inserida os dados, pode-se obter os valores das entradas analógicas do microcontrolador, já convertidas de zero a mil e vinte e três bits, totalizando mil e vinte e quatro bits.

Devido o sinal recebido pelo microcontrolador ser senoidal com frequência de 60 Hz, foi desenvolvido um código que tem como objetivo, ler por três mil vezes as portar AN0 e setecentas vezes a porta AN2, retirando então os maiores valores lidos dos sensores, podendo então, enviar para o supervisório.

Com os dados de tensão obtidos, é inserido a letra V na frente, com isso, a mensagem para envio via comunicação serial está pronta. Para os dados de corrente é inserido a letra A na frente, e a mensagem está pronta para envio.

Figura 8 – Diagrama de fluxo da programação do microcontrolador



Fonte: Autores (2021)

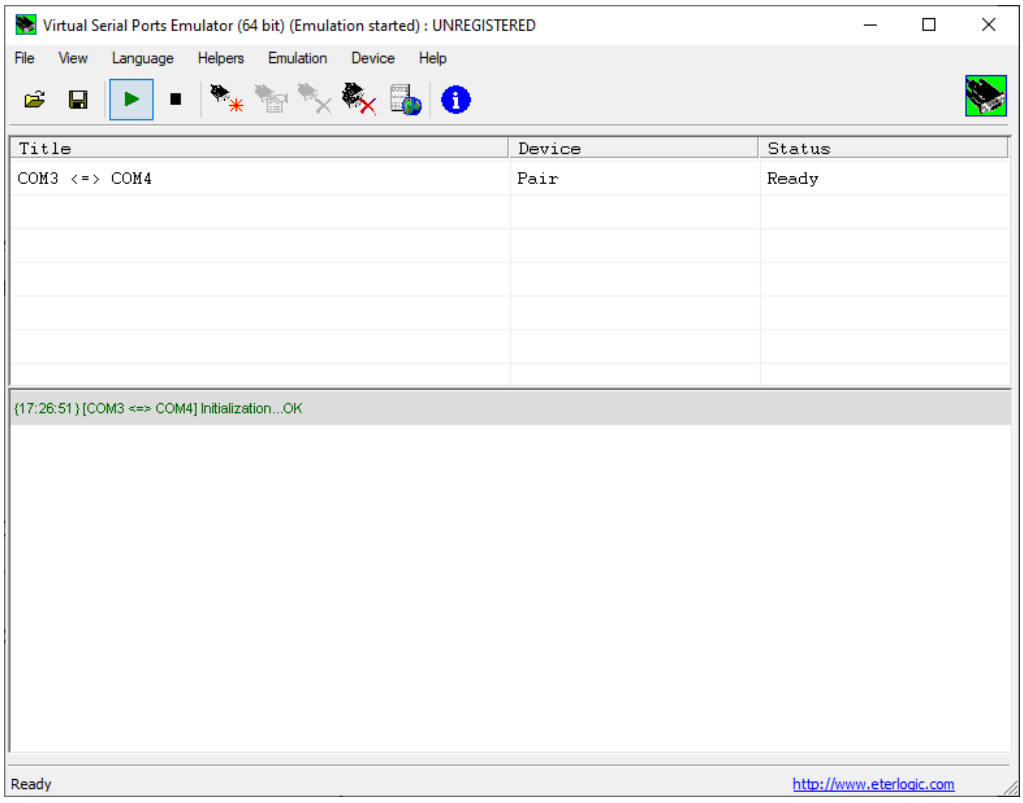
3.3 SISTEMA SUPERVISÓRIO

O sistema supervisório tem como principal objetivo, realizar a comunicação entre o microcontrolador e o Banco de Dados, também como exibir os dados lidos para o usuário. A programação do sistema supervisório consiste em 2 grandes pilares, sendo eles, as comunicações, e a exibição dos dados.

3.3.1 COMUNICAÇÕES

O supervisório inicialmente necessita ser conectado com o PIC através das portas COM, para isso, é necessário a utilização do software *Virtual Serial Port Emulator*, que tem como objetivo criar ás portas COM em modo *bridge*, fazendo uma ponte de comunicação entre as duas COM criadas, conforme demonstra a Figura 9.

Figura 9 – Criação das portas COM utilizando o software *Virtual Serial Port Emulator*



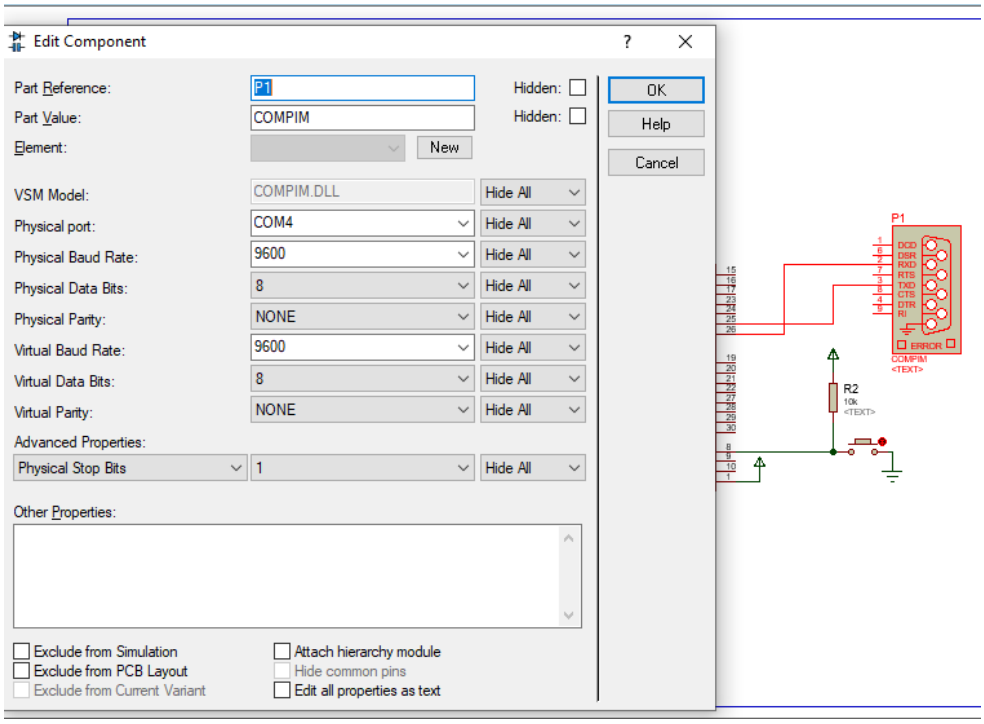
Fonte: Autores (2021)

De forma a ter a simulação dos componentes integrada com o sistema supervisório, fez-se necessário garantir a comunicação entre o simulador do microcontrolador (Proteus) e o supervisório, para isso, no software proteus, foi utilizado o componente COMPIM (Figura 10), que tem como objetivo, escrever de forma serial em umas das portas COM existentes no computador. Foi configurado então no campo *Physical port* a porta COM3, criada na etapa anterior.

Após inserir uma das portas COM no simulador, deve-se adicionar a configuração da comunicação serial, sendo esta, de velocidade 9600 *Baud Rate*, *Data Bits* 8, paridade *None* e *Stop Bit* 1.

Caso seja necessário alterar a velocidade de comunicação (*Baud Rate*), é preciso alterar o código do microcontrolador, pois o *Baud Rate* escolhido é através de cálculos considerando a velocidade do oscilador do microcontrolador.

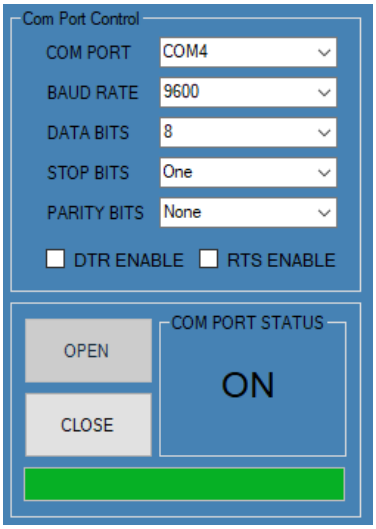
Figura 10 – Configuração das portas COM criadas utilizando o software Proteus



Fonte: Autores (2021)

Após, no software supervisório, realiza-se a mesma configuração da etapa anterior de comunicação conforme Figura 11.

Figura 11 – Configuração da porta COM no supervisório



Fonte: Autores (2021)

Inserida a porta COM diferente do software Proteus, aperta-se OPEN, e caso a comunicação seja estabelecida com sucesso, na aba *COM PORT STATUS* será exibido o status *ON*, conforme Figura 11.

A comunicação com o banco de dados é feita internamente nas linhas de códigos do supervisório, conforme na Figura 12.

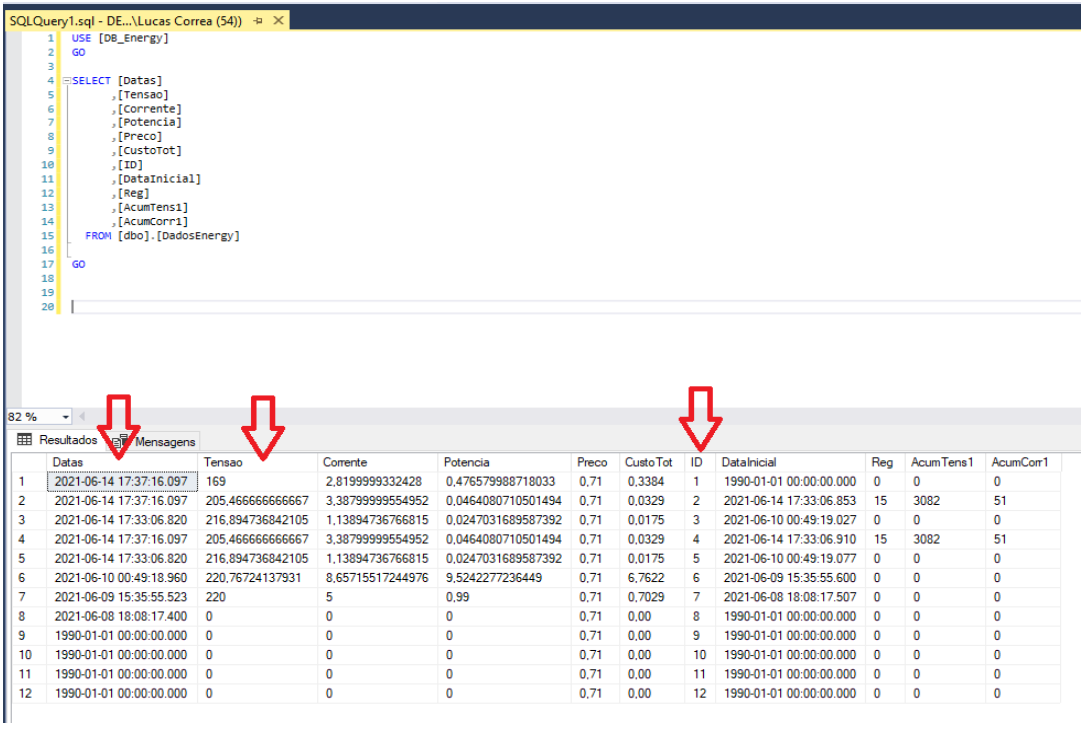
Figura 12 – Trecho da programação para comunicação com o banco de dados

```
144  SqlConnection sql = new SqlConnection("Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=DB_Energy;Data Source=DESKTOP-3SHV045\\SQLEXPRESS");
145
146  SqlCommand command = new SqlCommand("update DadosEnergy Set Datas = @Datas, Tensao = @Tensao where ID = 1", sql);
147
148     command.Parameters.Add("@Datas", SqlDbType.DateTime).Value = DateTime.Now;
149     command.Parameters.Add("@Tensao", SqlDbType.Real).Value = Tensaoaux;
150
151     sql.Open();
152     command.ExecuteNonQuery();
153     sql.Close();
```

Fonte: Autores (2021)

A linha 144 presente no código na figura 12, representa a inserção dos dados de segurança para estabelecer a comunicação com o banco de dados, já na linha 146 é inserido o comando que deseja-se enviar para o banco *SQL Server*, sendo o @Datas e @Tensão, variáveis internas do C# que vão ser atribuídas para as colunas de nome Datas e Tensão no supervisório, respetivamente, na linha que contem ID = 1, conforme mostra a Figura 13.

Figura 13 – Amostragem de dados Datas e Tensão no banco de dados



```
SQLQuery1.sql - DE...\Lucas Correa (54))
1  USE [DB_Energy]
2  GO
3
4  SELECT [Datas]
5        ,[Tensao]
6        ,[Corrente]
7        ,[Potencia]
8        ,[Preco]
9        ,[CustoTot]
10       ,[ID]
11       ,[DataInicial]
12       ,[Reg]
13       ,[AcumTens1]
14       ,[AcumCorr1]
15   FROM [dbo].[DadosEnergy]
16
17 GO
18
19
20
```

82 %

Resultados | Mensagens

	Datas	Tensao	Corrente	Potencia	Preco	CustoTot	ID	DataInicial	Reg	AcumTens1	AcumCorr1
1	2021-06-14 17:37:16.097	169	2,8199999332428	0,476579988718033	0,71	0,3384	1	1990-01-01 00:00:00.000	0	0	0
2	2021-06-14 17:37:16.097	205,466666666667	3,38799999554952	0,0464080710501494	0,71	0,0329	2	2021-06-14 17:33:06.853	15	3082	51
3	2021-06-14 17:33:06.820	216,894736842105	1,13894736766815	0,0247031689587392	0,71	0,0175	3	2021-06-10 00:49:19.027	0	0	0
4	2021-06-14 17:37:16.097	205,466666666667	3,38799999554952	0,0464080710501494	0,71	0,0329	4	2021-06-14 17:33:06.910	15	3082	51
5	2021-06-14 17:33:06.820	216,894736842105	1,13894736766815	0,0247031689587392	0,71	0,0175	5	2021-06-10 00:49:19.077	0	0	0
6	2021-06-10 00:49:18.960	220,76724137931	8,65715517244976	9,5242277236449	0,71	6,7622	6	2021-06-09 15:35:55.600	0	0	0
7	2021-06-09 15:35:55.523	220	5	0,99	0,71	0,7029	7	2021-06-08 18:08:17.507	0	0	0
8	2021-06-08 18:08:17.400	0	0	0	0,71	0,00	8	1990-01-01 00:00:00.000	0	0	0
9	1990-01-01 00:00:00.000	0	0	0	0,71	0,00	9	1990-01-01 00:00:00.000	0	0	0
10	1990-01-01 00:00:00.000	0	0	0	0,71	0,00	10	1990-01-01 00:00:00.000	0	0	0
11	1990-01-01 00:00:00.000	0	0	0	0,71	0,00	11	1990-01-01 00:00:00.000	0	0	0
12	1990-01-01 00:00:00.000	0	0	0	0,71	0,00	12	1990-01-01 00:00:00.000	0	0	0

Fonte: Autores (2021)

Nas linhas 148 e 149 da Figura 12, são atribuídos os valores para o @Datas e @Tensão vindas da comunicação serial, já na linha 151 habilita-se a comunicação e a linha 152 é responsável em enviar o comando para o *SQL Server*, e a linha 153 é responsável por encerrar a comunicação com o banco de dados. Para os dados relacionados a Corrente é realizado o mesmo procedimento citado anteriormente.

Para a verificação da integridade dos dados trocados entre o microcontrolador e o supervisório, é necessário realizar uma verificação da mensagem recebida pela segunda parte, sendo que este processo consiste em verificar o tamanho da mensagem, e se ela começa com o caractere inicial correto.

Os dados de tensão são constituídos por onze caracteres, sendo o caractere inicial representado pela letra V. Já os dados de corrente possuem nove caracteres, iniciando pela letra A. Após a verificação dos dados lidos, essas leituras são enviadas para o banco de dados. A Figura 14 mostra um trecho do algoritmo cuja função é realizar tal processo e verificar a leitura de Tensão e envio dos dados lidos para o banco de dados.

Figura 14 – Algoritmo de verificação de tensão e envio dos dados

```
if (dataIN.StartsWith("V"))
{

    Tensaoaux = null;
    Tensaoaux = dataIN;

    if (Tensaoaux.Length == 11)
    {

        Tensaoaux = Tensaoaux.ToString().Substring(1, (Tensaoaux.Length - 8));
        label9.Text = null;
        label9.Text = Tensaoaux;


        SqlConnection sql = new SqlConnection("Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=DB_Energy;Data Source=DESKTOP-3SHV045\\SQLEXPRESS");

        SqlCommand command = new SqlCommand("update DadosEnergy Set Datas = @Datas, Tensao = @Tensao where ID = 1", sql);

            command.Parameters.Add("@Datas", SqlDbType.DateTime).Value = DateTime.Now;
            command.Parameters.Add("@Tensao", SqlDbType.Real).Value = Tensaoaux;

            sql.Open();
            command.ExecuteNonQuery();
            sql.Close();

        this.dadosEnergyTableAdapter.Fill(this.dB_EnergyDataSet.DadosEnergy);
        dataGridView1.Columns[1].DefaultCellStyle.Format = "N2";
        dataGridView1.Columns[2].DefaultCellStyle.Format = "N2";
        dataGridView1.Columns[3].DefaultCellStyle.Format = "N2";

    }
}
```

Fonte: Autores (2021)

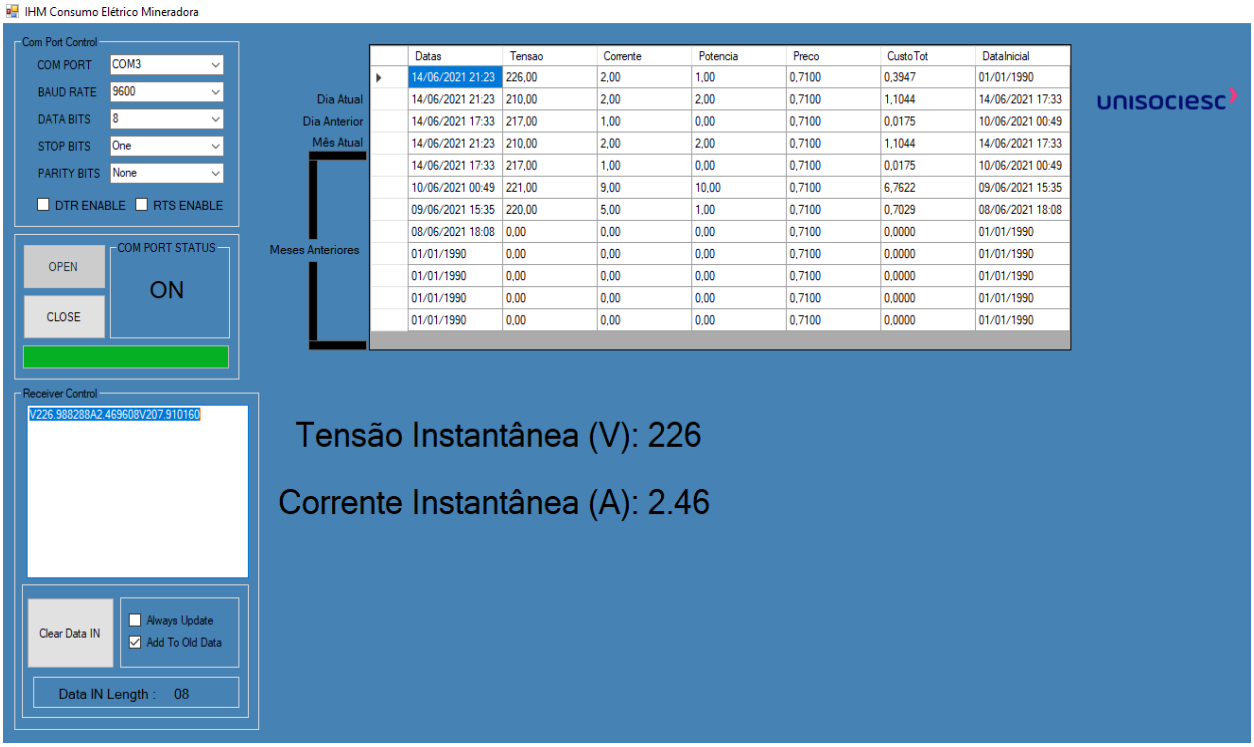
Para o envio dos dados de corrente é realizado um processo de forma similar ao descrito acima para os dados de tensão. Após os dados de corrente serem registrados no banco de dados, é feita a chamada pelo supervisório, do script dentro do *SQL Server* responsável em fazer todos os cálculos.

3.3.2 EXIBIÇÃO DOS DADOS

Para exibir os dados, é feito uma consulta da tabela do banco de dados dentro do supervisório através do *DataGridView*, uma função presente no *software Visual Studio*. O banco de dados presente no computador, é um banco de dados local, onde a conexão é mais facilmente estabelecida.

Os dados a serem exibidos no supervisório resumem-se na Figura 15, sendo eles, no canto esquerdo os dados que envolvem configuração e monitoramento da comunicação serial RS-232, e na parte central os dados energéticos.

Figura 15 – Supervisório e exibição dos dados



Fonte: Autores (2021)

A Figura 16 mostra uma representação de modo a facilitar o entendimento do fluxo de informações trocadas entre o microcontrolador, o sistema supervisório e o banco de dados.

Figura 16 – Representação do fluxo de dados entre os sistemas

Fonte: Autores (2021)

3.4 BANCO DE DADOS SQL SERVER

Para gerir todos os dados recebidos, dentro do *SQL Server* existem alguns scripts responsáveis em realizar os cálculos necessários, juntamente com a organização da tabela a ser visualizada no supervisório, conforme mostrada na Figura 15.

Após a leitura da corrente por meio do supervisório, o próprio supervisório envia uma linha de comando para o banco de dados, comando este, responsável por realizar os cálculos e arrumar a tabela, conforme Figura 16.

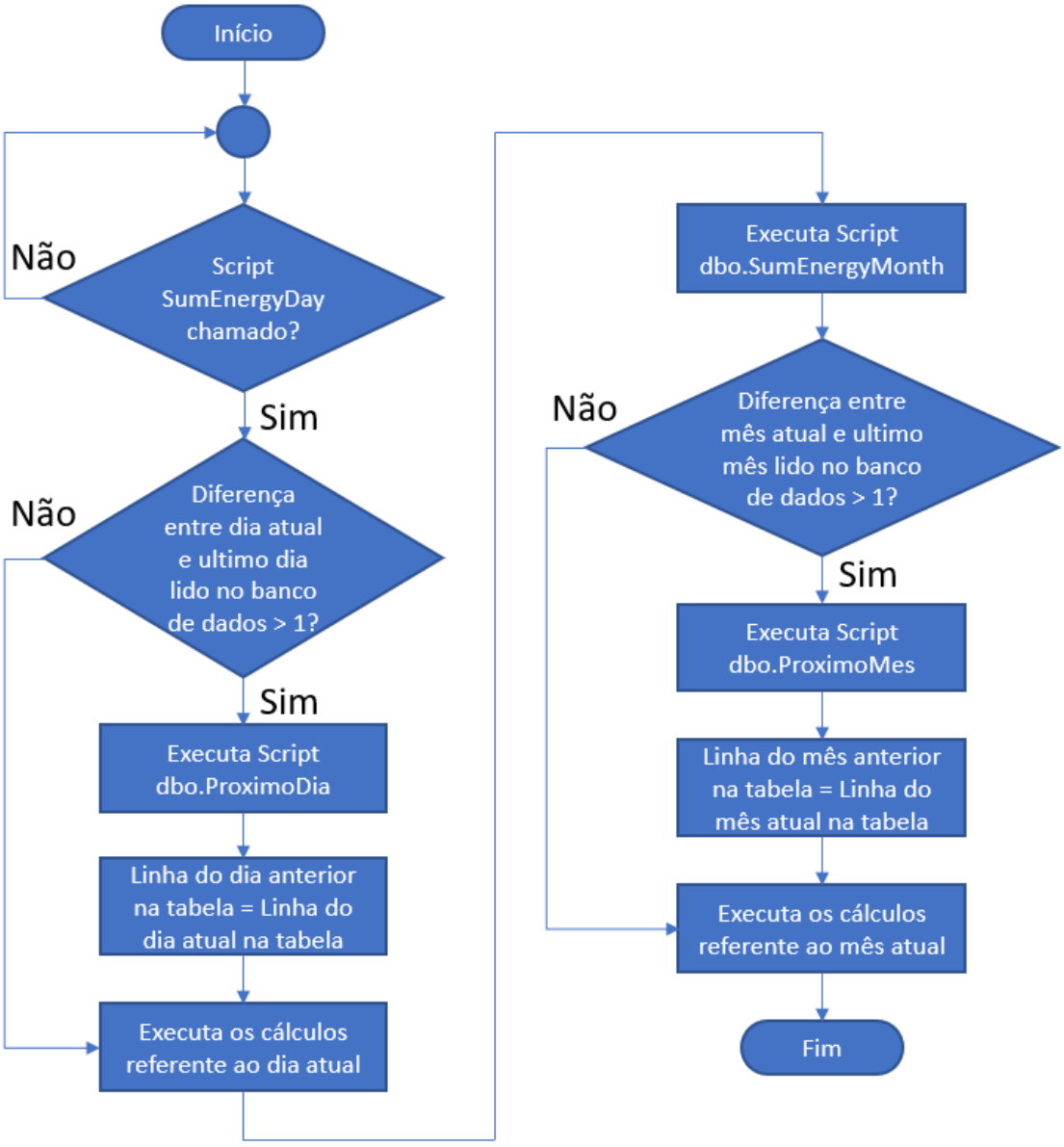
Figura 16 – Comando enviado pelo supervisório

```
SqlCommand commandProcedure1 = new SqlCommand("EXEC dbo.SumEnergyDay", sql);

sql.Open();
commandProcedure1.ExecuteNonQuery();
sql.Close();
```

Fonte: Autores (2021)

Dentro do banco de dados existem 4 *scripts* responsáveis por realizar os cálculos e organiza-los na tabela de dados. Existe um *script* cujo nome é dbo.SumEnergyDay, que ao ser chamado pode desencadear a execução dos outros três *scripts* presentes no banco de dados, conforme mostrado no fluxograma da Figura 17.

Figura 17 – Fluxograma do *script* dbo.SumEnergyDay



Fonte: Autores (2021)

Antes de executar qualquer cálculo, é preciso saber a diferença de dias entre a data da última medição registrada na tabela, com o horário do sistema, para isso, utilizou-se de uma função do SQL Server chamada DATEDIFF, que tem com objetivo retornar a diferença entre duas datas, sendo que a resposta da função pode ser escolhida entre minutos, segundos, dias, meses, anos, etc. A linha de código responsável por tal ação, pode ser vista na Figura 18.

Figura 18 – Algoritmo responsável por verificar a diferença entre datas

```
/*Verifica as datas para ver se é um novo dia*/

UPDATE DadosEnergy
SET Datas = (SELECT Datas from dbo.DadosEnergy where ID = 1) Where ID = 2;

SET @TOTALDAY4 = (SELECT DataInicial from [dbo].[DadosEnergy] where ID = 2)
SET @TOTALDAY6 = (SELECT Datas from [dbo].[DadosEnergy] where ID = 2)
SET @TOTDAY = DATEDIFF( DD , @TOTALDAY4 , @TOTALDAY6 )

IF (@TOTDAY != 0)
    EXEC dbo.ProximoDia;
```

Fonte: Autores (2021)

Caso a variável @TOTDAY seja diferente de zero, o programa entende que é um novo dia, executando então o *script* dbo.ProximoDia, responsável por atualizar os dados do Dia Atual e do Dia Anterior no supervisório, conforme Figura 19.

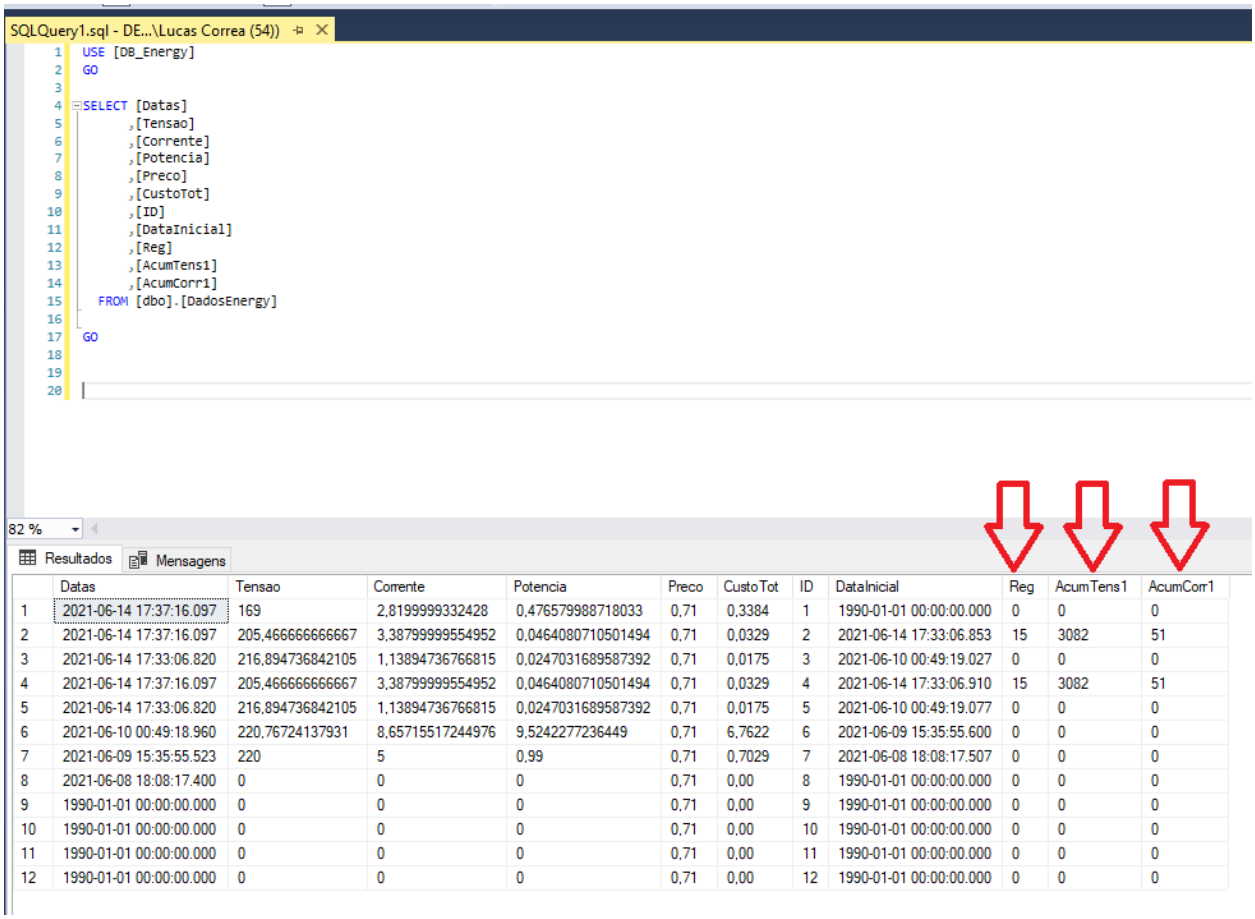
Figura 19 – Algoritmo para atualização de dados do Dia Atual e do Dia Anterior

```
SET @DATAATUAL = SYSDATETIME()

SET @REGNEXT = 0

UPDATE DadosEnergy

SET Reg = @REGNEXT where ID = 2

SET @DATASNEXT = (SELECT Datas from dbo.DadosEnergy where ID = 2)
SET @TENSAONEXT = (SELECT Tensao from dbo.DadosEnergy where ID = 2)
SET @CORRENTENEXT = (SELECT Corrente from dbo.DadosEnergy where ID = 2)
SET @POTENCIANEXT = (SELECT Potencia from dbo.DadosEnergy where ID = 2)
SET @PRECONEXT = (SELECT Preco from dbo.DadosEnergy where ID = 2)
SET @CUSTOTOTNEXT = (SELECT CustoTot from dbo.DadosEnergy where ID = 2)
SET @DATAINICIALNEXT = (SELECT DataInicial from dbo.DadosEnergy where ID = 2)

UPDATE DadosEnergy

SET Datas = @DATASNEXT, Tensao = @TENSAONEXT, Corrente = @CORRENTENEXT, Potencia = @POTENCIANEXT, Preco = 0.71, CustoTot = @CUSTOTOTNEXT, DataInicial = @DATAINICIALNEXT Where ID = 3;

UPDATE DadosEnergy

SET Datas = SYSDATETIME(), Tensao = 0, Corrente = 0, Potencia = 0, Preco = 0.71, CustoTot = 0, DataInicial = SYSDATETIME(), Reg = 0, AcumCorr1 = 0, AcumTens1 = 0 Where ID = 2;
```

Fonte: Autores (2021)

De modo a manter uma melhor precisão para os dados de tensão e corrente, existem duas colunas responsáveis em acumular todas as medidas recebidas, sendo elas *AcumTens1* para tensão e *AcumCorr1* para corrente. Existe também, uma terceira coluna chamada *Reg*, que contém a quantidade de vezes que tais dados foram acumulados, conforme mostrado na Figura 20.

Figura 20 – Colunas para amostragem dos dados acumulados



SQLQuery1.sql - DE...\Lucas Correa (54))

```
USE [DB_Energy]
GO

SELECT [Datas]
      ,[Tensao]
      ,[Corrente]
      ,[Potencia]
      ,[Preco]
      ,[CustoTot]
      ,[ID]
      ,[DataInicial]
      ,[Reg]
      ,[AcumTens1]
      ,[AcumCorr1]
  FROM [dbo].[DadosEnergy]

GO
```

82 %

Resultados | Mensagens

	Datas	Tensao	Corrente	Potencia	Preco	CustoTot	ID	DataInicial	Reg	AcumTens1	AcumCorr1
1	2021-06-14 17:37:16.097	169	2,8199999332428	0,476579988718033	0,71	0,3384	1	1990-01-01 00:00:00.000	0	0	0
2	2021-06-14 17:37:16.097	205,466666666667	3,38799999554952	0,0464080710501494	0,71	0,0329	2	2021-06-14 17:33:06.853	15	3082	51
3	2021-06-14 17:33:06.820	216,894736842105	1,13894736766815	0,0247031689587392	0,71	0,0175	3	2021-06-10 00:49:19.027	0	0	0
4	2021-06-14 17:37:16.097	205,466666666667	3,38799999554952	0,0464080710501494	0,71	0,0329	4	2021-06-14 17:33:06.910	15	3082	51
5	2021-06-14 17:33:06.820	216,894736842105	1,13894736766815	0,0247031689587392	0,71	0,0175	5	2021-06-10 00:49:19.077	0	0	0
6	2021-06-10 00:49:18.960	220,76724137931	8,65715517244976	9,5242277236449	0,71	6,7622	6	2021-06-09 15:35:55.600	0	0	0
7	2021-06-09 15:35:55.523	220	5	0,99	0,71	0,7029	7	2021-06-08 18:08:17.507	0	0	0
8	2021-06-08 18:08:17.400	0	0	0	0,71	0,00	8	1990-01-01 00:00:00.000	0	0	0
9	1990-01-01 00:00:00.000	0	0	0	0,71	0,00	9	1990-01-01 00:00:00.000	0	0	0
10	1990-01-01 00:00:00.000	0	0	0	0,71	0,00	10	1990-01-01 00:00:00.000	0	0	0
11	1990-01-01 00:00:00.000	0	0	0	0,71	0,00	11	1990-01-01 00:00:00.000	0	0	0
12	1990-01-01 00:00:00.000	0	0	0	0,71	0,00	12	1990-01-01 00:00:00.000	0	0	0

Fonte: Autores (2021)

Com os valores acumulados, é possível obter a média da tensão e corrente, fazendo $AcumTens1/Reg$ e $AcumCorr/Reg$, conforme demonstra a Figura 21. Após esses cálculos, pode-se exibir os dados de potência e custo total de energia gasta para o dia atual e mês atual no supervisório, como é mostrado na figura anterior.

Figura 21 – Algoritmo para cálculos de tensão, corrente, potência e custo total

```
/* Calculo tempo maquina rodando*/

SET @AUXHOUR1 = (SELECT DataInicial from [dbo].[DadosEnergy] where ID = 2)
SET @AUXHOUR2 = (SELECT Datas from [dbo].[DadosEnergy] where ID = 2)
SET @Tothour = DATEDIFF( MI , @AUXHOUR1 , @AUXHOUR2 )

/*Execução dos calculos ID 2*/

SET @CALCReg = (SELECT Reg from dbo.DadosEnergy where ID = 2) + 1

SET @AUXSUMTENSAO = ((SELECT AcumTens1 from [dbo].[DadosEnergy] where ID = 2) + (SELECT Tensao from [dbo].[DadosEnergy] where ID = 1))
SET @AUXSUMCURRENT = ((SELECT AcumCorr1 from [dbo].[DadosEnergy] where ID = 2) + (SELECT Corrente from [dbo].[DadosEnergy] where ID = 1))
SET @DIVCURRENT = (@AUXSUMCURRENT / @CALCReg)
SET @DIVTENSAO = (@AUXSUMTENSAO / @CALCReg)
SET @AUXPottot = ((((@DIVCURRENT*@DIVTENSAO) / 1000) * (@Tothour / 60))
SET @AUXCustotot2 = ((((@DIVCURRENT*@DIVTENSAO) / 1000) * (@Tothour / 60) * (SELECT Preco from dbo.DadosEnergy where ID = 2))
```

Fonte: Autores (2021)

Na Figura 21, entre as linhas 69 e 71, há o cálculo da diferença de datas da primeira medida menos a data da última medida, sendo a resposta desse cálculo em minutos, nos dando quanto tempo ficou ligado o software de medição no dia. Entre as linhas 75 e 82, executamos os cálculos de acúmulo de tensão e corrente, os cálculos de tensão média, corrente média, potência acumulada em *KWatts*/h e custo total em reais, esses mesmos cálculos são feitos tanto para o dia atual, quanto para o mês atual.

4. RESULTADOS E DISCUSSÕES

Com a parte de simulação concluída, deu-se início a parte de execução, onde consiste em fazer a montagem elétrica dos componentes, realizar a análise dos resultados obtidos através das leituras de tensão e corrente com o PIC, comparando com os valores de consumo energético já entregues pelo computador, e por último, realizar um tutorial mostrando como fazer um supervisório em C# comunicar via RS232 com um PIC.

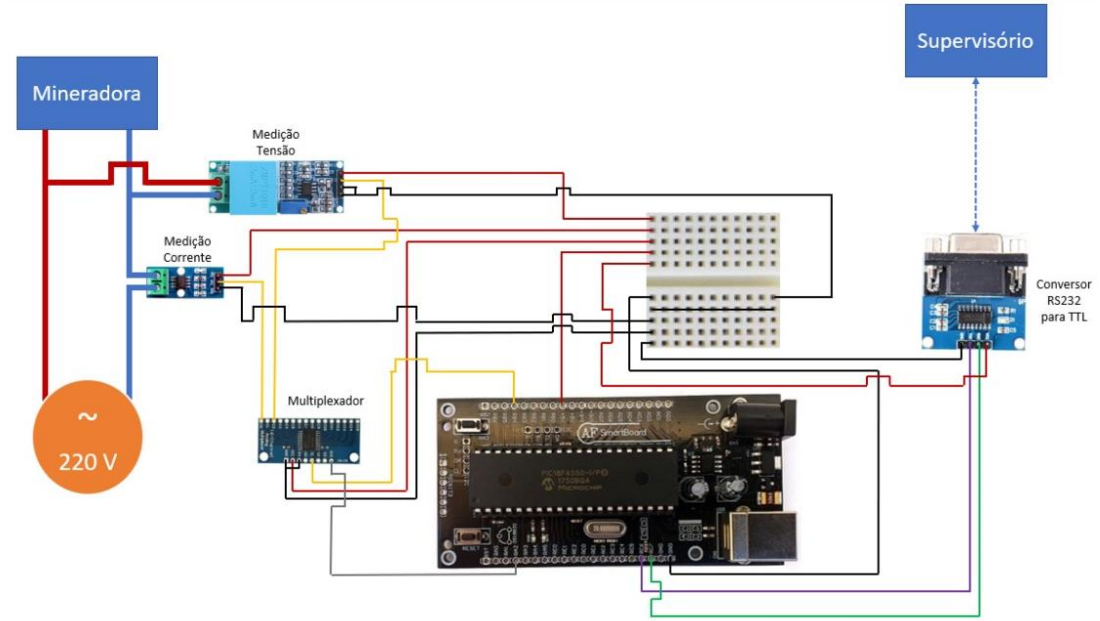
4.1 DESENVOLVIMENTO E MONTAGEM DO DISPOSITIVO

Nesta etapa, concluída a parte de simulação e foi iniciado a parte física, onde foram encontradas algumas dificuldades.

Para elaboração do medidor de consumo, foi utilizado o Kit de desenvolvimento PIC18F4550 desenvolvido pela AF SmartBoard, juntamentamente com os sensores ZMPT101B para medição de tensão e o sensor ACS712 para medição de corrente.

Inicialmente foi idealizado utilizar os canais AN0 e AN2 do PIC para realizar as respectivas ligações dos sensores, contudo, ao fazer a soldagem dos pinos nas placas, e realizado os primeiros testes simulando uma tensão de 3.3 V nas entradas, observou-se que o canal AN0 estava com um sinal de 5 V fixo no terminal, isso ocorreu com outros pinos exceto o AN2, então precisando utilizar 2 portas para leitura, sendo que, apenas a AN2 funcionou corretamente, foi utilizado um multiplexador comercial de 16 canais Cd74hc4067, para que possa ler os dois sensores no canal AN2, a nova configuração das ligações mudou, ficando conforme Figura 22.

Figura 22 – Esquema de ligação dos componentes



Fonte: Autores (2021)

Para as medições serem enviadas para o supervisório, foi utilizado um conversor de serial para TTL, baseado no chip MAX232, onde tem como entrada de dados os respectivos pinos RC6 e RC7 TX e RX do PIC, enviando via DB9 os valores já em TTL de modo que o computador reconheça como sendo uma porta COM, além do MAX232, foi necessário um cabo DB9 macho para USB macho.

Após fazer o download do software idealizado nas simulações para o PIC, o mesmo travou, debugando o código, observou-se que as funções presentes na biblioteca STDLIB.H, estavam travando o PIC. Para que o software funcione, foi realizado algumas modificações no código de modo em que, em vez de realizar a leitura adc, e converter esses dados para uma *array* de *char* e assim então enviar para o supervisório, foi idealizado converter os registradores de leitura adc de inteiro para binário, deixando que o supervisório se encarregue de converter de binário para *double*. Essa modificação otimizou muito o código deixando-o mais enxuto e mais rápido, essa modificação no código do PIC pode ser vista na figura 23.

Figura 23 – Função PIC de conversão Decimal para Binário

```
void dec2bin(int c)
{
    int i = 0;
    for(i = 11; i >= 0; i--){
        if((c & (1 << i)) != 0)
        {

        UART_Write('1');

        }else{

        UART_Write('0');

        }
    }
}
```

Fonte: Autores (2021)

Com essa alteração no código, agora enviando binário e não mais char, as conversões de binário para float entregues pelo PIC via serial, são realizadas dentro do supervisório, reduzindo a programação do PIC, realizando as medições de forma mais rápida. Essa alteração de leitura no supervisório, pode ser vista na Figura 24.

Figura 24 – Função C# de conversão Binário para String e Double

```
if (Tensaoaux.Length == 13)
{
    Tensaoaux = Tensaoaux.Trim('V');
    Tensaoaux = Convert.ToInt32(Tensaoaux, 2).ToString();
    Double Tensaodou = Convert.ToDouble(Tensaoaux);
    Tensaodou = Tensaodou * 0.2443;
    String Tensaotex = Convert.ToString(Tensaodou);
    label9.Text = null;
    label9.Text = Tensaotex;
```

Fonte: Autores (2021)

Depois de realizados os testes em bancada, foi feito o encapsulamento do mesmo tornando-o compacto e de fácil instalação. Ficaram para fora do dispositivo dois cabos USB, sendo um de alimentação e outro da comunicação serial, um conector de tomada macho, onde deve ser ligado na rede elétrica, e o conector fêmea, onde deve ser ligado a mineradora, conforme demonstrado na Figura 25.

Figura 25 – Dispositivo de medição de consumo elétrico

Fonte: Autores (2021)

4.2 ANÁLISE E TESTES DO SISTEMA DE MONITORAMENTO

Para realizar as análises e teste, vai ser utilizado os dados obtidos através do medidor de consumo elétrico, com os *datasheet* dos equipamentos, como também os dados de consumo fornecido pelo computador.

Conforme já esperado, para fazer o cálculo do consumo elétrico da mineradora, não se pode pegar o *datasheet* dos equipamentos e utilizá-los para calcular o KW/h gasto, já que para o processo de mineração, não se usa 100% do processador e memória do computador , outro caso parecido com esse, é referente ao principal equipamento, as placas de vídeo. Para garantir a maior eficiência possível, deve-se utilizar softwares que alteram as frequências das memórias, e da *GPU*, tanto como também, limitar o *power draw* da placa de vídeo.

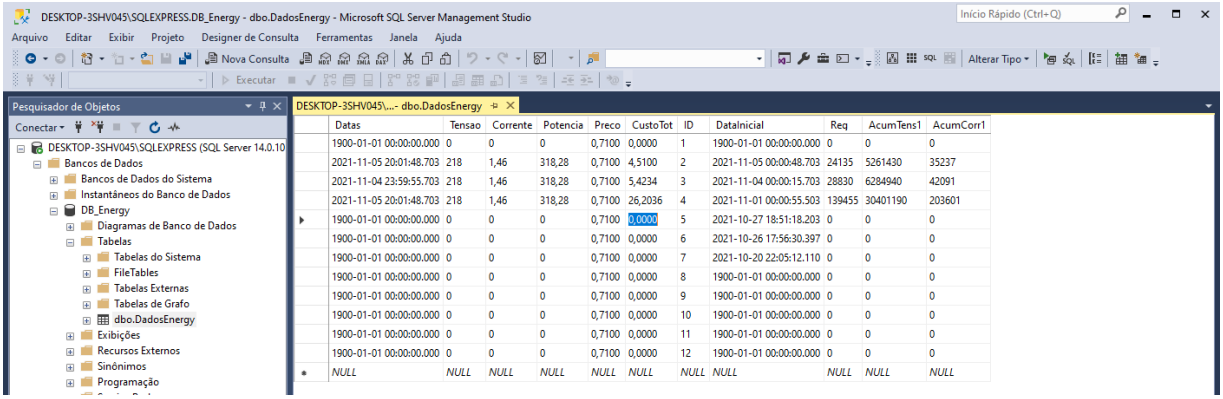
Tabela 1 – Medições e levantamento dos dados de consumo

Equipamentos	Potência Registrada Softwares (Watts)	Datasheet dos Equipamentos (Watts)	Medidor de Consumo
RX5500XT (Placa de Vídeo)	60	130	
GTX 1060 (Placa de Vídeo)	78	120	
RTX 3060 (Placa de Vídeo)	110	170	
Processador	5	58	
SSD	N/A	0,279	
Memória RAM	N/A	67	
SOMATÓRIA WATTS	253	545,279	318
Média de Tensão Volts			218
Média de Corrente Amperes			1,46

Fonte: Autores (2021)

Conforme a Tabela 1 apresenta, os dados de consumo disponibilizado pelo computador não são suficientes para calcular o KW/h, isso se da porque o mesmo software não disponibiliza os dados de consumo de todos os equipamentos principais do computador, quando comparado as medições entre os dados disponibilizados pelo computador com os dados do medidor de consumo, observa-se que existe uma diferença de aproximadamente 65 Watts, essa diferença fica ainda maior quando utilizado os dados disponibilizados pelos datasheets dos equipamentos, já que eles disponibilizam o valor com 100% de carga. Os dados do consumo das leituras do Medidor de Consumo, foram extraídos diretamente do banco de dados onde o software escreve, conforme Figura 26.

Figura 26 – Banco de Dados Referente a 5 dias de medições



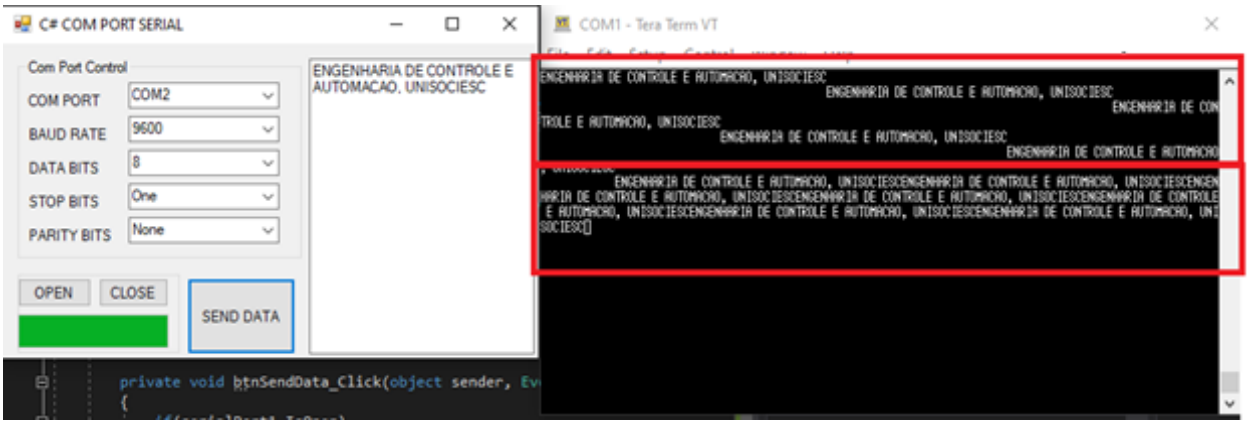
Fonte: Autores (2021)

4.3 DESENVOLVIMENTO DE UM TUTORIAL DE COMO CRIAR UM SUPERVISÓRIO C# COM PIC VIA RS-232

A criação do tutorial aqui apresentado foi elaborada utilizando o software Visual Studio 2017, para as simulações de portas seriais e para simular o terminal de conexão do computador, que serão necessárias para a demonstração de alguns passos, foi utilizado o software VPSE (Visual Port Serial Emulator), também necessário para etapas demonstradas mais a frente, foi utilizado o software Tera Term e, por último, utilizamos o software PROTEUS para simular o microcontrolador PIC e o envio de dados na etapa final do tutorial.

A primeira etapa documento consiste em demonstrar a criação e funcionamento do mesmo mecanismo utilizado neste projeto, mas como o intuito de enviar dados ao invés de recebe-los, mostrando assim, a versatilidade da linguagem C# em conjunto com o software Visual Studio e abrindo margem para projetos futuros que o leitor possa vir a ter. A Figura 27 mostra o funcionamento desta primeira etapa no envio de dados, sendo a parte em destaque o software Tera Term simulando o receptor.

Figura 27 – Envio de dados do supervisório para o *software Tera Term*



Fonte: Autores (2021)

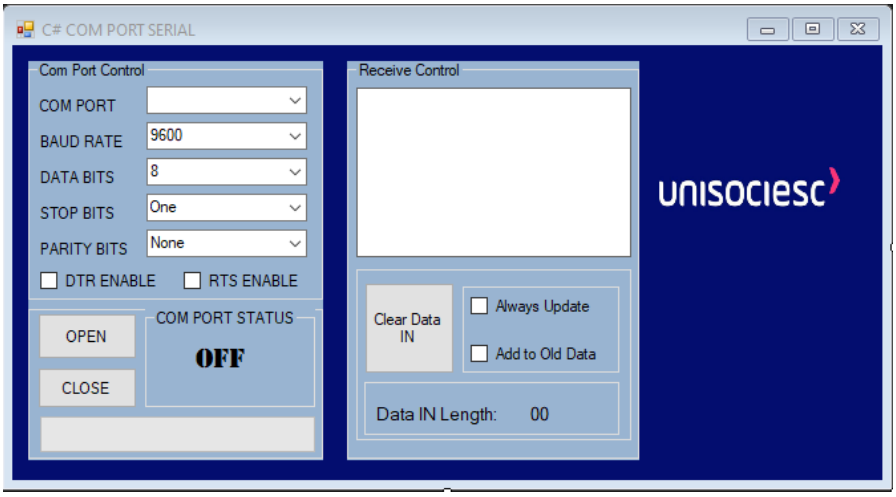
A segunda etapa consiste no objetivo proposto, sendo ele os passos para a criação de um supervisório capaz de receber dados via RS232 e exibi-los em tela. Os passos iniciais de configuração para a conexão entre as duas portas são os mesmos da etapa anterior, caso o leitor opte por fazê-los de forma separada ou não tenha utilidade para alguma delas.

Utilizando as ferramentas citadas e o passo-a-passo descrito no tutorial, o leitor estará apto a criar um supervisório capaz de receber e exibir em tela dados de um

microcontrolador PIC, mas os mesmos passos poderiam ser adaptados para microcontroladores de outros fabricantes. Ao final o leitor terá uma caixa de texto onde a mensagem será exibida, assim como o tamanho da mensagem recebida e poderá, dentre outras coisas, alterar a forma da exibição dos dados.

Vale apontar que o tutorial não mostra a criação fidedigna do supervisório apresentado no artigo, haja visto que o mesmo utiliza de outras ferramentas como um banco de dados desenvolvido em *SQL Server*, mas as ferramentas presentes no *Visual Studio* que são mostradas nesta etapa são as mesmas utilizadas na criação do supervisório deste projeto. A Figura 28 mostra como ficará o *layout* do supervisório ensinado nesta segunda etapa.

Figura 28 – Supervisório final da segunda etapa do tutorial



Fonte: Autores (2021)

5. CONCLUSÃO

Durante o desenvolvimento do trabalho, foi possível adquirir diversos conhecimentos voltados a banco de dados *SQL Server* e supervisório desenvolvidos em C#, utilizando o *software Visual Studio*, como também a programação do microcontrolador utilizado pela equipe.

A simulação da placa proposta pela equipe foi de grande vália para a aplicação prática, pois, os resultados obtidos atenderam todas as expectativas dos autores, mostrando, assim, que o projeto está adequado para ser montado fisicamente.

Com a programação em execução, pode-se observa que ela foi capaz de obter bons resultados de forma eficiente, garantido que a equipe consiga obter os dados para

o supervisório, contudo, por mais que a comunicação seja feita virtualmente, dentre todas as mensagens enviadas para o supervisório, uma certa percentagem não chega da maneira correta, com isso, fica um ponto de observação para aplicação física, podendo gerar dificuldades.

O supervisório de maneira geral, conseguiu tanto receber os dados via comunicação com o microcontrolador, quanto trata-los de maneira que possa garantir o recebimento da mensagem correta, como também, mostrou ter uma grande capacidade de integração com o banco de dados, tanto para exibir, quanto para comunicar-se com o mesmo.

A utilização do banco de dados *SQL Server*, mostrou ser uma ferramenta muito poderosa no que diz respeito a tratar dados, com ele foi possível armazenar todas as informações necessárias, como também a execução de todos os cálculos, pode-se observar que a mesma tem um grande potencial para grandes aplicações voltadas a integração com sistemas supervisórios, dando mais autonomia para o programador.

Com o trabalho desenvolvido, pode-se abrir portas mostrando a capacidade dos softwares gratuitos *SQL Server* e *Visual Studio*, mostrando-se capazes de gerir um sistema supervisório de pequeno porte, observou-se também que o *software Visual Studio* tem a capacidade para comunicação *ModBus*, abrindo possibilidades para aplicações de maior porte, no que diz respeito a sistemas supervisórios.

No que diz respeito a confecção do dispositivo, pode-se observar que só a simulação não é suficiente para validar um projeto, quando colocado em prática, a placa do microcontrolador desenvolvida pela empresa AFSmartBoard demonstrou ter poucas informações disponíveis, dificultando o *debug* de alguns erros que ocorreram no desenvolvimento. Por isso, um dos upgrades possíveis é utilizar um PIC solto com um gravador separado, podendo você fazer toda a ligação do PIC e então debugando melhor o código.

As análises realizadas pelas medições foram coerentes, se mostraram eficientes e cumprem com o principal objetivo, que é demonstrar o consumo juntamente com os valores de tensão e corrente da rede. Como um upgrade futuro, uma análise de maior tempo de aquisição e comparações com o clima e o quanto minerou no dia, com o consumo, podem mostrar melhor a eficiência da mineradora.

A elaboração do tutorial desenvolvido como etapa deste projeto pretende ser de grande ajuda para futuros leitores, caso os mesmos sintam interesse na plataforma Visual Studio e na linguagem de programação C#, além é claro, do tema principal

proposto. Com o desenvolvimento desta etapa os autores tiveram um aprofundamento maior no tema, haja visto que ensinar é a melhor forma de consolidar um aprendizado. Criar o tutorial mostrou-se um processo desafiador, pois, muitos detalhes precisavam ser mostrados para que este se tornasse um meio de aprendizado simples e de fácil entendimento para todos que no futuro o lerem.

REFERÊNCIAS BIBLIOGRÁFICAS

ANTONOPOULOS, Andreas M. **Mastering Bitcoin:** unlocking digital crypto-currencies. 1. ed. Sebastopol: O'Reilly Media, 2014.

AGNER, Marco. **Bitcoin Para Programadores.** 1. ed. Rio de Janeiro: ITSRIO, 2018.

BROWN, Gavin, WHITTLE, Richard. **Algorithms, Blockchain & Cryptocurrency –** implications for the future of the workplace. 1. ed. Bingley: Emerald Publishment Limited, 2020.

JUNIOR, Ervaldo Garcia. **Introdução a sistemas de supervisão, controle e aquisição de dados: SCADA**. 1. ed. Rio de Janeiro: Alta Books, 2019.

KUGLER, Logan. **Why Cryptocurrencies Use So Much Energy:** and what to do about it. Nova York, vol. 61, n. 07, p. 15-17, julio, 2018. Disponível em https://doi.org/10.1145/3213762>. Acesso em: 03 mai. 2021.

MASEDA, Javier F. *et al.* Sensors Data Analysis in Supervisory Control and Data Acquisition (SCADA) Systems to Foresee Failures with an Undetermined Origin. **Sensors,** Basiléia, vol. 21, n. 2762, abril, 2021. Disponível em <https://doi.org/10.3390/s21082762>. Acesso em: 2 jun. 2021.

MICROCONTROLLERSLAB. **UART Communication with Pic Microcontroller** - programming in MPLAB XC8. Disponível em <UART Communication PIC Microcontroller, Programming in MPLAB XC8 (microcontrollerslab.com)>. Acesso em: 20 de mar. 2021.

MICROCHIP. **PIC18F2455/2550/4555/4550 Data Sheet.** Chandler: Microchip Technology Inc, 2006.

MICROCHIP. **MPLAB XC8 C Compiler User's Guide.** Chandler: Microchip Tecnology inc, 2012.

MICROSOFT. **Tutoriais do Visual Studio –** C#. 2019. Disponível em <Desenvolvimento em C# com o Visual Studio - Visual Studio | Microsoft Docs> Acesso em: 08-22 mai. 2021.

MICROSOFT. **Visão Geral das Ferramentas do SQL**. 2020. Disponível em <Visão geral das ferramentas do SQL - SQL Server | Microsoft Docs> Acesso em: 05-12 abr. 2021.

MIYADAIRA, Alberto Noboru. **Microcontroladores PIC18:** aprenda e programe em linguagem C. 4. ed. São Paulo: Érica, 2013.

NARAYANAN, Arvind *et al.* **Bitcoin and Cryptocurrency Technologies**: a comprehensive introduction. Princeton: Princeton University Press, 2016.

PALLONES, Mark. Getting The Most Out of Peripherals.**Eletronics World.** Londres. vol 122, p. 22-25, maio, 2016. Disponível em <May 2016 – Electronics World>. Acesso em: 25 mai. 2021.

THOMAZINI, Daniel. ALBUQUERQUE, Pedro U. B. **Sensores Industriais:** fundamentos e aplicações. 8. ed. São Paulo: Érica, 2011.

CAMBRIDGE, Univesity of. **Cambridge Bitcoin Eletricity Consumption Index.** Disponível em <Cambridge Bitcoin Electricity Consumption Index (CBECI)> Acesso em: 25 mar. 2021.

NVIDIA. **Geforce RTX 3060 e RTX 3060 TI**. 2021. Disponível em <https://www.nvidia.com/pt-br/geforce/graphics-cards/30-series/rtx-3060-3060ti/>. Acesso em: 12 out. 2021.

NVIDIA. **Geforce GTX 1060**. 2021. Disponível em <https://www.nvidia.com/pt-br/geforce/products/10series/geforce-gtx-1060/>Acesso em: 12 out. 2021.

AMD. **AMD Radeon™ RX 5500 XT**. 2021. Disponível em <https://www.amd.com/pt/products/graphics/amd-radeon-rx-5500-xt> Acesso em: 12 out. 2021.

INTEL. **Processador Intel® Celeron® G5925 (4 M de cache, 3,60 GHz)**. 2021. Disponível em
< https://www.intel.com.br/content/www/br/pt/products/sku/201887/intel-celeron-processor-g5925-4m-cache-3-60-ghz/specifications.html>
Acesso em: 12 out. 2021.

OUTERVISION. **OuterVision® Power Supply Calculator**. 2021. Disponível em
<https://outervision.com/power-supply-calculator>
Acesso em: 12 out. 2021.