UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
UNIDADE ACADÊMICA DE ENGENHARIA ELÉTRICA

RELATÓRIO DE ESTÁGIO SUPERVISIONADO

Rafael Soares do Egito

Campina Grande
2014

RAFAEL SOARES DO EGITO

Relatório de Estágio submetido à Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Engenharia Elétrica.

RELATÓRIO DE ESTÁGIO SUPERVISIONADO

Aprovado em ___/___/___

Professor Avaliador

Universidade Federal de Campina Grande

Avaliador

Professor Doutor George Acioli Júnior

Universidade Federal de Campina Grande

Orientador

Campina Grande

2014

Agradecimentos

A Deus, pela Sua soberania que me permitiu chegar até aqui. Que Ele seja glorificado acima de tudo.

À minha mãe, por tantos anos de trabalho árduo para me dar a oportunidade de realizar este sonho, me apoiando em todos os momentos e nunca me deixando desistir.

Ao meu irmão, pelos seus vários puxões de orelha que me fizeram ser um profissional excelente como ele.

Ao meu pai, que sempre se orgulhou de mim e me trouxe motivação para seguir nesta caminhada.

À minha namorada Rayanne, que me apoiou e me suportou nas minhas dificuldades e momentos de desânimo.

Aos meus amigos que, sem dúvida, foram imprescindíveis não apenas nas horas de estudos, mas também nas de lazer.

"Todas as coisas foram feitas
por intermédio dele, e, sem ele,
nada do que foi feito se fez"

João 1:3

Sumário

1 Introdução

Relatório referente ao estágio realizado na empresa Suna Engeharia, localizada em Campina Grande, Paraíba, no período de 10 de julho de 2014 a 17 de novembro de 2014, totalizando 450 horas. São apresentadas todas as atividades desenvolvidas no período, tal como uma breve apresentação da empresa, mostrando seus principais trabalhos, nos quais serão aplicados o trabalho desenvolvido pelo aluno.

São apresentadas, de forma clara e objetiva, as etapas da implementação do padrão de comunicação entre os dispositivos que formam a rede formada entre os dispositivos de controle e os supervisórios, de maneira semelhante a uma rede industrial. A rede é formada por microcontroladores Arduino atuando como controladores, um computador Raspberry Pi atuando como servidor e PCs como supervisórios.

2 Apresentação da Empresa

A Suna Engenharia é uma empresa jovem, fundada no ano de 2012, de caráter inovador, comprometida com valores éticos e ambientais. A mesma nasceu com o desejo de romper o ciclo vicioso de combate à seca, mostrou que o problema das áreas áridas é, basicamente, a falta de tecnologia adequada ao convívio com a seca. Assim a empresa tem como seu maior objetivo possibilitar o convívio do homem do campo com a seca. Mas para possibilitar esse convívio cada propriedade ou mesmo um grupo de propriedades deveria se unir de maneira a um sistema integrado de geração de renda, como apresentado na Figura 1 a seguir.

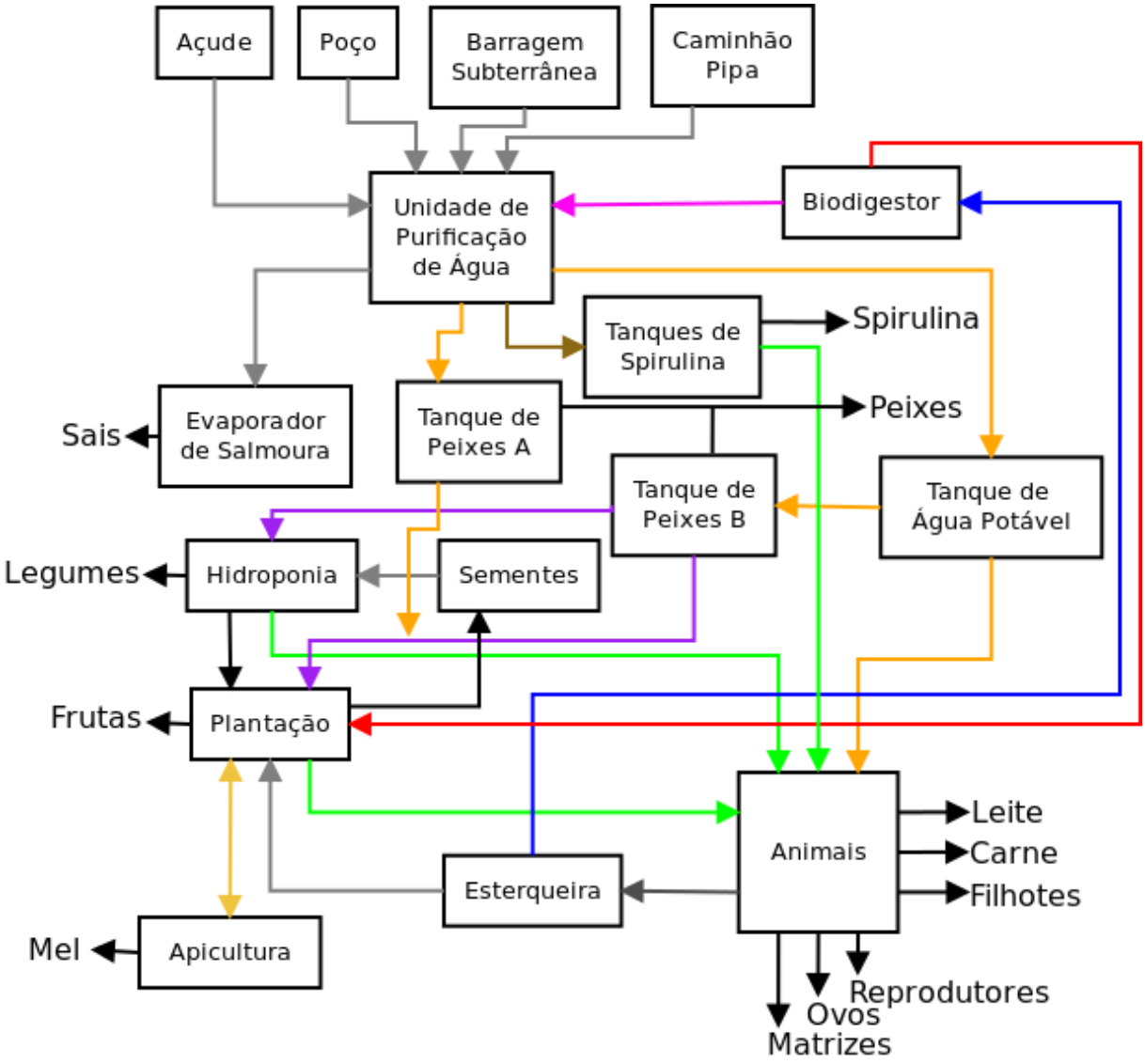

Figura 1 - Sistema integrado para geração de renda

Para que fosse possível a criação desse sistema diversos problemas tinham que ser solucionados. O mais importante deles era a Unidade de Purificação de Água. Foi pensando

nisso que fundador da empresa o engenheiro químico Sebastião de Araújo Coutinho criou o Concentrador de Sólidos Solúveis.

2.1 Concentrador de Sólido Solúvel

O Concentrador de Sólidos Solúveis é um equipamento que promove a separação de um líquido dos sólidos dissolvidos através do processo de umidificação e desumidificação com reaproveitamento do calor latente de vaporização.

O reaproveitamento de calor latente de vaporização reduz significativamente os custos de operação do equipamento, sendo possível e recomendado o uso de fontes de energias alternativas tais como solar, geotérmica entre outras para fornecimento de calor a evaporação reduzindo, desta forma, os custos de operação.

Uma das grandes qualidades desse equipamento é a sua versatilidade uma vez que através da "concentração" de sólidos e dependendo de em qual processo o mesmo é utilizado ele pode ter diversos nomes entre concentrador e dessalinizador. O mesmo pode ser utilizado com concentrador nos processos de produção de soda caustica, café solúvel, suco de frutas e substâncias sensíveis a temperatura. E como dessalinizador, onde se necessita concentrar os sais da água. Na Figura 2 é apresentada uma representação desse equipamento funcionando como um dessalinizador.

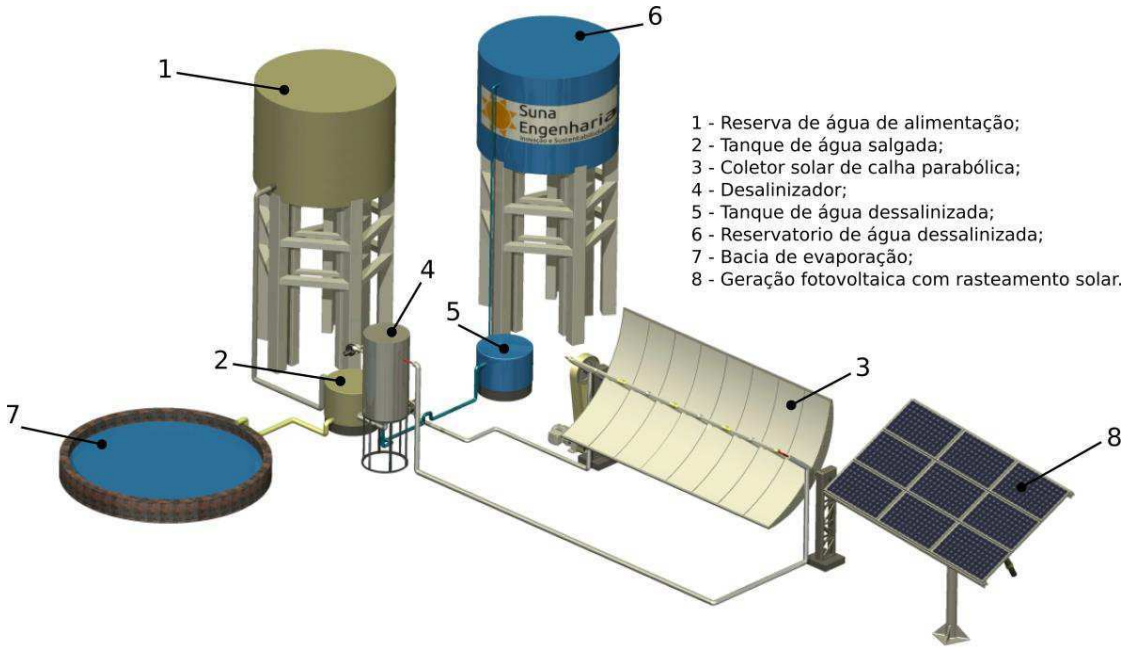

Figura 2 - Coletor solar funcionando com um dessalinizador

Esse equipamento funcionando como um dessalinizador surgiu com o intuito de se substituir a dessalinização por Osmose Reversa que embora seja a opção mais conhecida apresenta elevado custo ambiental devido a grande quantidade de efluente salino produzido o

que tem um reflexo direto nos seus elevados custos e por isso mesmo só é utilizado para purificação de água para o consumo humano. Já o dessalinizador, como apresentado na Figura 2, possui uma elevada produção (quando comparado com o processo de osmose reversa) além de apresentar como subproduto do processo apenas uma lama salina que entre outras coisas pode até servir como uma fonte se sais minerais para animais.

Durante a fase de projeto do Concentrador de Sólidos Solúveis percebeu-se que em muitas situações seria necessário que componentes como o soprador de ar e a bomba funcionassem em localidades onde não tem rede elétrica. Com isso seria necessário a inserção de fontes de energia não convencional como, por exemplo, a energia solar para mover esses componentes. Mas isso iria incorrer em um grande problema inerente a essas maquinas (geralmente monofásicas) que é o fato de que a corrente necessária para partir essas máquinas serem geralmente até 6 vezes maiores do que a sua corrente nominal. Dessa maneira seria necessário superdimensionar o sistema o que encareceria o projeto. Uma possível solução seria colocar um componente muito conhecido pelas indústrias que é um *softstarter*. O problema era que esse componente só existia para máquinas trifásicas e a maioria das máquinas utilizadas no dessalinizador são monofásicas. Com isso a empresa desenvolveu o seu próprio *softstarter*.

2.2 Softsarter Monofásico

As aplicações de energia solar fotovoltaicas necessitam que, em seu dimensionamento, sejam consideradas variáveis como insolação média, radiação solar global média e a carga a ser fornecida. Quando trata-se de sistemas de bombeamento movido a energia solar fotovoltaica, é necessário considerar uma carga extra, maior que a carga nominal do motor da bomba. Essa obrigatoriedade deve-se a presença do pico de potência na partida do motor elétrico da bomba que em alguns casos pode ser mais de seis vezes o valor da potência nominal.

Esse comportamento é inerente aos motores elétricos, sua existência causa consumo excessivo de potência na partida, obriga ao sobre dimensionamento dos cabos de alimentação. Para reduzir esse pico existem os inversores de frequência e chaves de partida estática para motores trifásicos apenas.

No intuito de resolver alguns desses problemas, a Suna Engenharia e a Teslatec desenvolveram um dispositivo para eliminar os picos de potência da partida de motores elétricos monofásicos - SSM-736. Sua utilização contribui para redução do consumo elétrico

de motores que partem diversas vezes ao dia e por consequência também contribui para redução dos impactos ambientais.

O desenvolvimento do SSM-736 foi motivado pelos seguintes desafios:

- Substituir motores elétricos especiais de 12 Vcc por motores elétricos monofásicos de 220 Vca em projetos de energia solar devido ao alto investimento e alto custo de manutenção.
- Eliminar os picos de corrente característicos dos motores indutivos que podem danificar os inversores de potência utilizados em aplicações de energia solar.
- Inexistência de *softstarter* para motores monofásicos.
- Economia de energia com motores monofásicos industriais e residenciais que partem diversas vezes ao dia, com significativa redução da vibração desses motores durante a sua partida e a parada.

Atualmente o protótipo encontra-se em testes nas dependências da Suna Engenharia.

2.2.1 Aplicações para o Softstarter Monofásico

Sistema de Bombeamento com Energia Solar

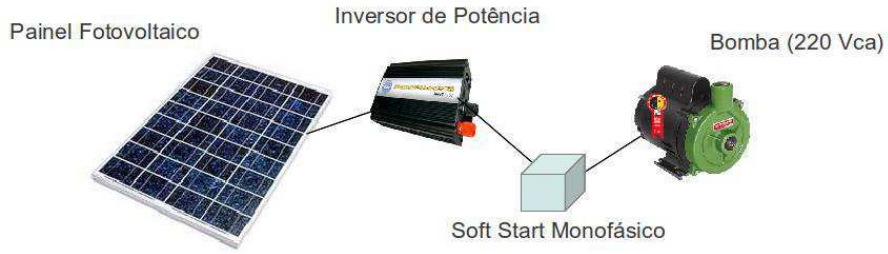

Figura 3 - Sistema de bombeamento com energia solar

Os sistemas de bombeamento com uso de energia solar fotovoltaica, como o apresentado na Figura 3, apresentam características adequadas para o uso do *softstarter* monofásico tais como:

- Elevado custo de investimento inicial.
- Superdimensionamento da geração elétrica.
- Elevado pico de potência na partida.
- Reduzida vazão bombeada, aproximadamente 0,9 m^3/h em 10 mca.
- Utilização de motores especiais de 12 Vcc.

Com a utilização do *softstarter* monofásico é possível obter:

- Menor custo de implantação e manutenção.
- Redução da potência do painel solar.
- Redução de componentes dedicados.
- Utilização de motores comuns, 220 Vca, para acionamento das bombas.
- Elevada vazão bombeada, aproximadamente 3.000 m3/h em 10 mca.

Motores Industriais

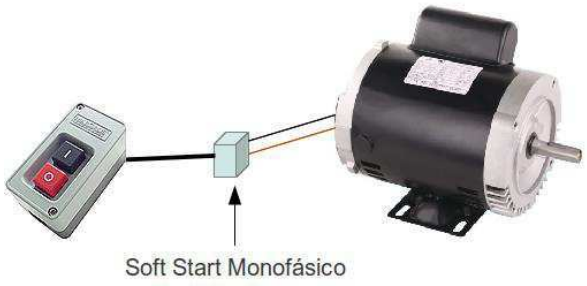

Figura 4 - Uso de *softstarter* monofásico

Os motores industriais, como o apresentado na Figura 4, são adequados ao uso dos *softstarter* monofásico pelos seguintes motivos:

- Elevado número de partidas e paradas em um dia.
- Elevado consumo elétrico na partida.
- Afundamento na tensão da rede durante a partida.
- Com a utilização do softstarter monofásico é possível obter:
- Redução do consumo elétrico na partida.
- Eliminação do afundamento de tensão durante a partida.

Assim, observa-se que dois dos principais problemas já foram solucionados. Com isso, é possível colocar o dessalinizador para funcionar. Mas ainda existem alguns problemas a serem solucionados entre eles o baixo rendimento das atuais células fotovoltaicas, que é a principal fonte de energia elétrica para o dessalinizador. Assim procura-se minimizar esse problema com a inserção de um rastreador solar (projeto também desenvolvido) que tem por objetivo reduzir as perdas de energia por reflexão da luz solar quando comparado com o painel fotovoltaico estático.

2.3 Áreas de Atuação

A Suna Engenharia conta com profissionais qualificados para executar os seguintes serviços:

Pesquisa e Desenvolvimento

- Estudo da viabilidade técnica e econômica;
- Criação de protótipo.

Patentes

- Busca;
- Redação.

Sistemas Fotovoltaicos

- Dimensionamento;
- Montagem;
- Manutenção.

Automação

- Auditoria de malhas de controle;
- Sintonia de controladores;
- Auditoria de válvula de controle utilizando o software de gerenciamento de malhas de controle da Matrikon;
- Configuração do SDCD CS3000 da Yokogawa;
- Configuração do Batch3000 da Yokogawa;
- Configuração do historiador de dados de processos InfoPlus21;
- Programação de CLP.

Instrumentação

- Calibração de instrumentos de campo e de laboratório;
- Configuração e parametrização de controladores, transdutores e conversores;
- Manutenção e calibração de válvulas de controle;
- Manutenção de cromatógrafo e analisador de líquido e de gás de processo.

Elétrica e Eletrônica

- Montagem e manutenção de painéis de comando;
- Manutenção de sistemas controlados por CLPs;
- Manutenção de retificadores e UPSs;
- Manutenção de sistemas de Proteção Catódica;
- Manutenção elétrica predial.

3 Atividades Desenvolvidas

3.1 Suporte às Atividades Desenvolvidas

No início de suas atividades na empresa, o aluno deu suporte às atividades desenvolvidas pela empresa com conhecimentos de Engenharia Elétrica. A empresa desenvolve vários projetos que utilizam dispositivos elétricos, eletrônicos (microprocessadores, sensores, etc). Como único membro da equipe que é da área de Engenharia Elétrica, o aluno pôde participar de reuniões para discutir viabilidade técnica e econômica dos dispositivos a serem adquiridos pela empresa.

3.1.1 Análise de viabilidade de sensores

O aluno ficou encarregado de realizar uma pesquisa de mercado a fim de tomar a decisão de quais marcas e modelos dos seguintes sensores deveriam sem comprados:

- Termômetro digital;
- Higrômetro;
- Voltímetro/amperímetro digital;
- Medidor de pH;
- Termostato;
- Controlador de rotação de motor DC.

Foi utilizado o seguinte critério para a tomada da decisão de qual produto a empresa deveria adquirir: comparam-se os principais parâmetros de cada produto e apresentam-se para a empresa três relatórios de orçamento com os produtos que possuem, respectivamente, os parâmetros elevados, medianos e baixos, ficando a cargo da empresa a tomada de decisão final na compra dos produtos. Os principais critérios avaliados (que se aplicam a todos os produtos listado) foram:

- Resolução;
- Temperatura de operação;
- Umidade de operação;
- Faixa de alimentação.

Além dos critérios específicos de cada produto.

3.1.2 Análise de viabilidade de lâmpada UV purificadora

Em certa aplicação da empresa foi projetada a colocação de uma lâmpada ultravioleta purificadora para limpeza do ar que entra em um laboratório com controle de impurezas. O problema é que a lâmpada poderia causar um aumento na temperatura do ar e, se fosse o caso, seria necessário o projeto de um sistema de resfriamento do ar. Sendo assim, foi necessária a investigação sobre lâmpadas ultravioleta, que ficou a cargo do aluno.

Na investigação, foi concluído que esse tipo de lâmpada trabalha na faixa de 40°C. Comparando com outros tipos de lâmpada, esse valor é bem baixo, concluindo-se então que não seria necessário nenhum tipo de resfriamento. O gráfico da Figura 5 apresenta a potência para lâmpadas UV, que está entre as mais baixas do espectro.

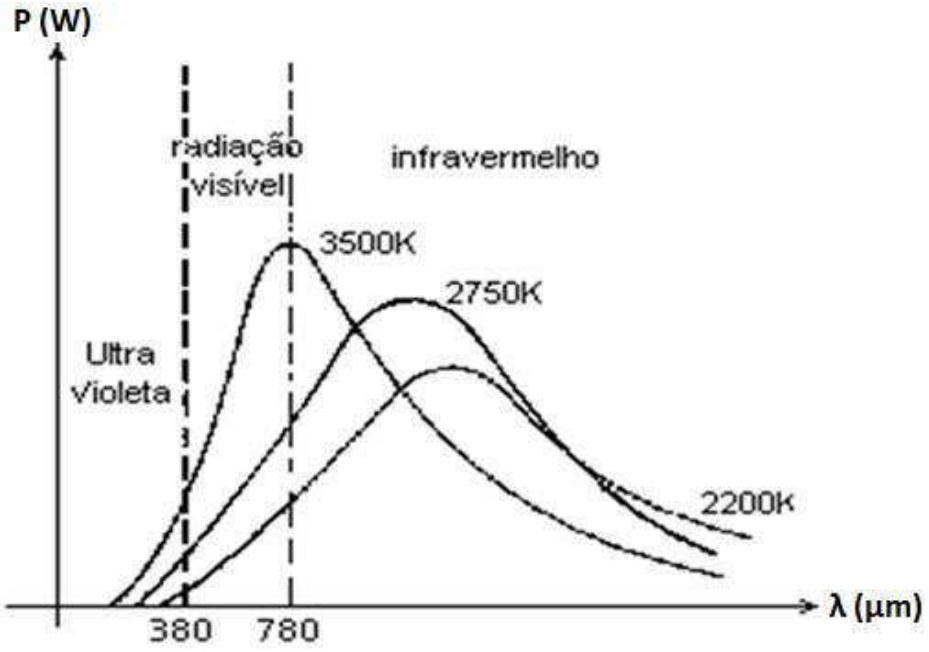

Figura 5 - Gráfico de potências de lâmpada *versus* comprimento de onda

3.1.3 Problema com associação de termopares

Foi apresentado ao aluno o problema típico de imprecisão da medição com termopares. Uma pesquisa na literatura levou a diversas soluções para este problema. O aluno concluiu, juntamente com o engenheiro chefe que, para o caso da empresa, a melhor solução é a associação de dois termopares, sendo que um deles deve ser imerso em água a 0 °C (como ilustrado na Figura 6). A escolha desse método é devida à sua simplicidade de execução e ao fato de a necessidade da medição externa para garantir a água a 0 °C não é um problema para este caso específico.

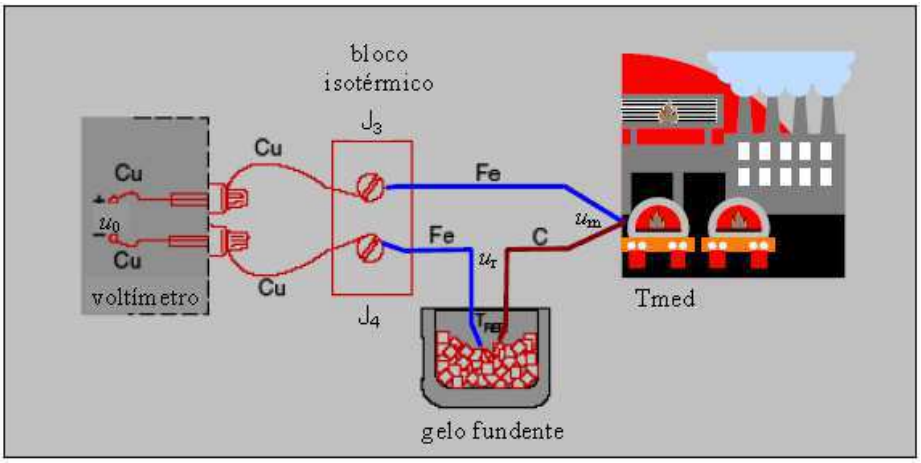

Figura 6 - Associação de termopares para cancelar a imprecisão na medição

Este tipo de medição consiste nos seguintes passos:

- Utilizam-se dois termopares, um imerso em água a 0 °C e outro no local a ser medido;
- Associam-se os termopares juntando as pontas de mesmo metal;
- As duas pontas que restaram são as que serão ligadas no voltímetro.

3.2 Automatização de uma Máquina de Impacto por Queda de Peso

A empresa desenvolveu uma máquina de impactopor queda de peso, como mostrado na Figura 7, para o LPI do Departamento de Engenharia Mecânica da UFCG. A máquina deveria captar variações de velocidade e aceleração da deformação de uma amostra ao impactar com o peso da máquina.

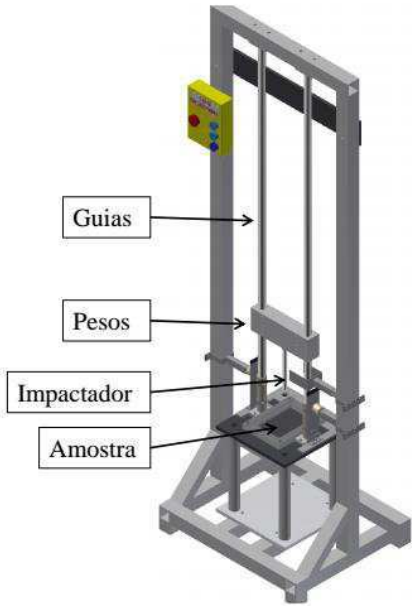

Figura 7 - Máquina de impacto

O projeto exigia a automatização do processo que consistia em um acelerômetro (para a medição da aceleração do impactador, de onde seriam extraídos todos os outros dados úteis do processo), um sistema de aquisição, um sistema antirrebote e um *software* que fosse capaz de receber os dados do experimento, calcular os valores das variáveis envolvidas e disponibilizá-los em formato que fosse possível de ser interpretado por um *software* de planilhas (e.g., Microsoft Excel, LibreOffice Calc etc).

A fim de familiarizar-se com o ambiente de desenvolvimento Qt (que será melhor explicado posteriormente neste relatório), o aluno foi responsável por implementar no *software* os algoritmos responsáveis pelos cálculos das variáveis envolvidas no processo, além de apresentar os resultados em gráficos. Os algoritmos envolviam, inclusive, comunicação serial com o Arduino presente no sistema de aquisição da máquina.

As equações implementadas foram as seguintes:

- Cálculo da velocidade final

$$V = \sqrt{2gh} \tag{1}$$

onde

g = aceleração da gravidade, em m/s^2

h = altura, em m

- Cálculo da energia de impacto

$$E_{impacto} = mgh \tag{2}$$

onde

m = massa, em kg

- Velocidade de impacto

$$V_i = \frac{W_{12}}{t_2 - t_1} + g\left(t_i - \frac{t_2 + t_1}{2}\right) \tag{3}$$

onde

V_i = velocidade de impacto, em m/s

W_{12} = distância entre os dois sensores de posição, em m/s

t_1 = instante de passagem pelo primeiro sensor, em s

t_2 = instante de passagem pelo segundo sensor, em s

g = aceleração da gravidade, em m/s^2

- Cálculo da velocidade durante o ensaio

$$V(t) = V_i + gt - \int_0^t \frac{F(t)}{m} dt \quad (4)$$

$$V(t) = V_i + gt - \int_0^t \frac{m \cdot a(t)}{m} dt \quad (5)$$

$$V(t) = V_i + gt - \int_0^t a(t)\, dt \quad (6)$$

onde

$V(t)$ = velocidade do impactador no instante t, em m/s

t = tempo de duração do ensaio, em s

$F(t)$ = força medida no instante t, em N

$a(t)$ = aceleração medida pelo acelerômetro, em m/s^2

- Deformação da amostra durante o ensaio

$$\delta(t) = \delta_i + V_i t + \frac{gt^2}{2} - \int_0^t \left(\int_0^t \frac{F(t)}{m} dt \right) dt \quad (7)$$

$$\delta(t) = \delta_i + V_i t + \frac{gt^2}{2} - \int_0^t \left(\int_0^t a(t)\, dt \right) dt \quad (8)$$

onde

$\delta(t)$ = deformação do impactador no instante t, em m

δ_i = deformação de referência no instante $t = 0$, em m

- Energia absorvida durante o impacto

$$E_a(t) = \frac{m(v_i^2 - v(t)^2)}{2} + mg\delta(t) \quad (9)$$

onde

$E_a(t)$ = energia absorvida no tempo t, em J

- Energia de impacto

$$E_i = \frac{mV_i^2}{2} \quad (10)$$

A Figura 8 apresenta a tela principal do *software* desenvolvido, na qual é possível ver a grade onde são apresentadas as amostras e os valores das variáveis calculadas.

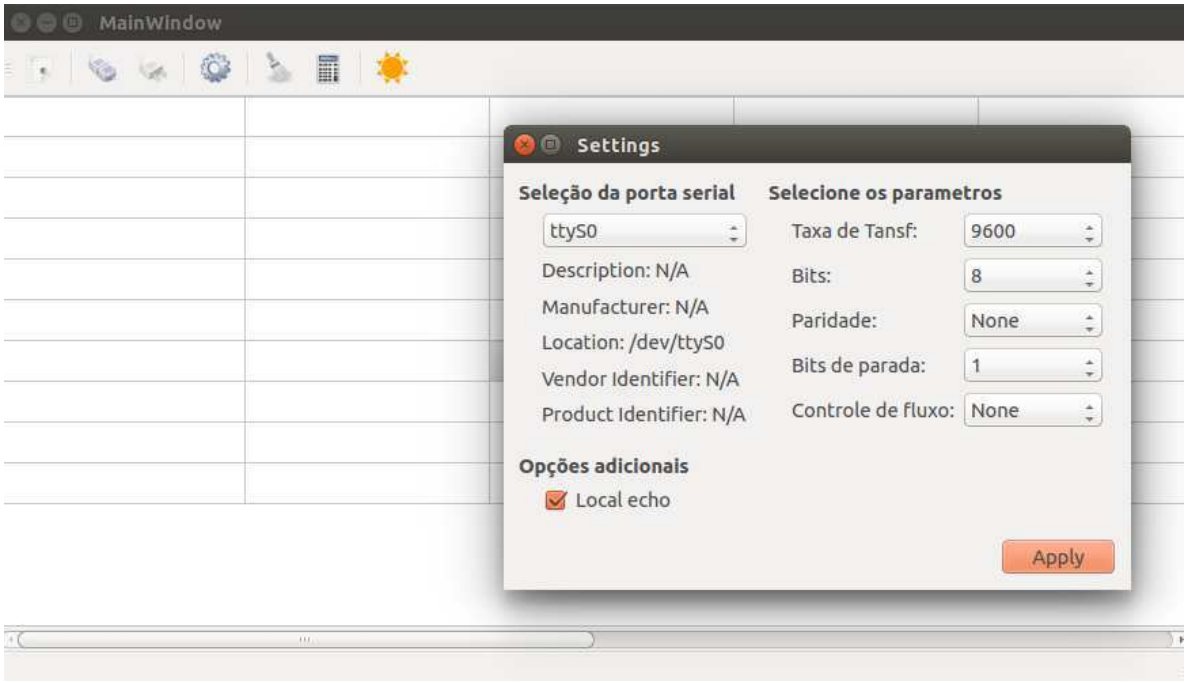

Figura 8 - Tela principal do *software* desenvolvido

3.3 Desenvolvimento do Padrão de Comunicação da Empresa

Foi desenvolvido o padrão de comunicação que será utilizado entre os dispositivos que são utilizados nos produtos desenvolvidos pela empresa. O padrão consiste basicamente em um conjunto de funções computacionais que serão utilizadas em cada nível da arquitetura da rede de dispositivos, a fim de criar um protocolo exclusivo da empresa, garantindo que a comunicação será realizada da maneira adequada e com agilidade e também que nenhum outro dispositivo não credenciado seja adicionado à rede, além de facilitar a tarefa de programação da comunicação entre os níveis.

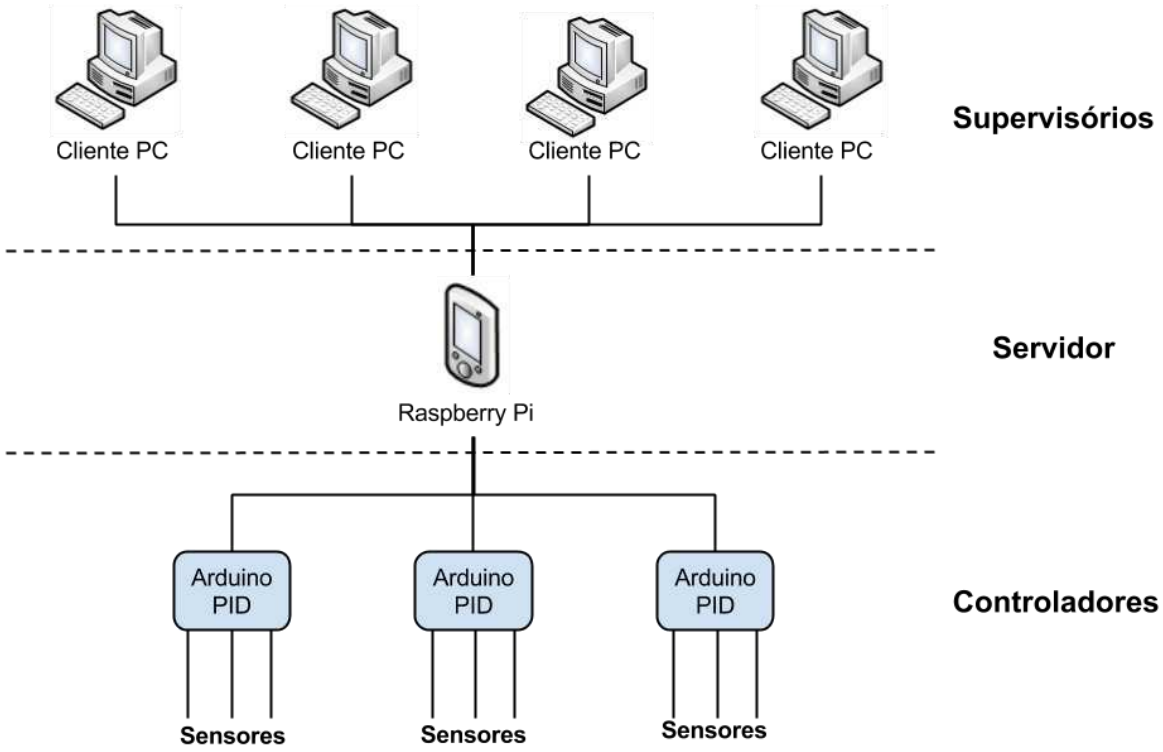

Figura 9 - Arquitetura da rede

A rede, como apresentado na Figura 9, é divida em três camadas: a dos supervisórios, a do servidor e a dos controladores. A camada de supervisórios é composta de PCs que farão a supervisão dos dados de todos os controladores. O servidor é um dispositivo Raspberry Pi e é o responsável por gerenciar a transmissão de dados entre os controladores e os supervisórios. Os controladores são implementados em Arduinos que se comunicam com o servidor através de conexão serial.

3.3.1 Qt e Qt Creator

Os padrões utilizados nas camadas de servidor e supervisório foram desenvolvidas utilizando a plataforma Qt, que é um *framework* multiplataforma para desenvolvimento de interfaces gráficas em C++ criado pela empresa norueguesa Trolltech. Com ele é possível desenvolver aplicativos e bibliotecas uma única vez e compilá-los para diversas plataformas sem que seja necessário alterar o código-fonte. Foi utilizada a IDE Qt Creator que é uma IDE multiplataforma que traz consigo a Qt SDK, que permite o desenvolvimento de *software* em Qt utilizando interface gráfica (de maneira semelhante ao Visual Studio).

O Qt foi escolhido pela possibilidade de gerar o *software* com interface gráfica e, principalmente, por ser multiplataforma, permitindo facilmente a migração entre os sistemas

operacionais Linux e Windows, além da possibilidade de programar o Raspberry Pi em um PC comum e apenas compilar o código no mesmo.

3.3.2 Qwt

Qwt (sigla do inglês para *Qt Widgets for Technical Applications*) é um conjunto de *widgets* Qt, componentes de interface de usuário e classes que são úteis primeiramente para programas de aplicação técnica. Além de plotagem 2D, o Qwt fornece escala, cursores, mostradores tipo *dial*, mostradores tipo termômetro, círculos e botões para controle ou exibição de valores, *arrays* ou intervalos do tipo *double*. As figuras 10 e 11 apresentam exemplos de *widgets* criados utilizando Qwt.

Fonte: http://qwt.sourceforge.net/sinus.png

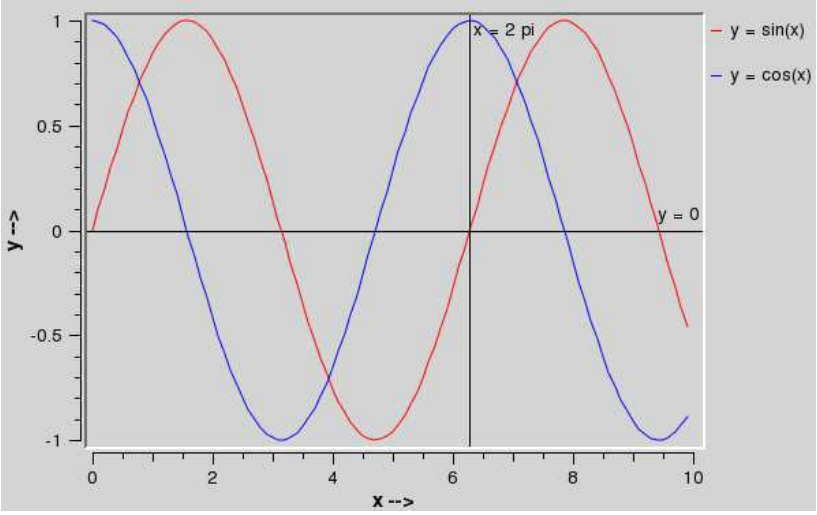

Figura 10 - Exemplo de gráfico gerado utilizando Qwt

Fonte: http://www.digitalfanatics.org/projects/qt_tutorial/pngs/11/1.png

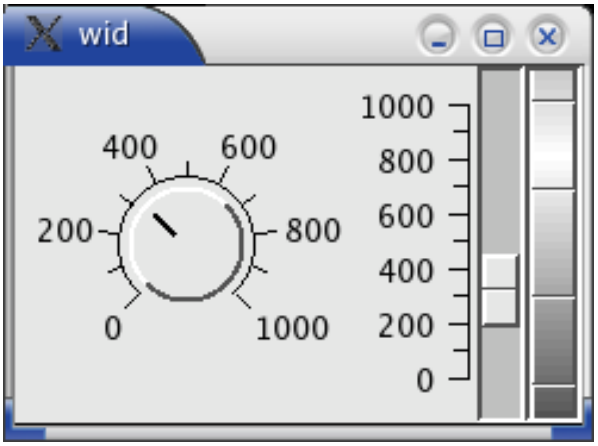

Figura 11 - Exemplo de controle gerado utilizando Qwt

Os *widgets* fornecidos pela ferramenta Qwt foram uma ferramenta poderosa de muita utilidade na aplicação do padrão criado, em especial na camada de supervisórios, na qual é de extrema importância a apresentação clara dos dados, de modo que os mesmos sejam facilmente interpretados.

3.3.3 Arduino

Arduino é uma plataforma de prototipagem eletrônica de *hardware* livre e de placa única projetada com um microcontrolador Atmel AVR com suporte de entrada/saída embutido, uma linguagem de programação padrão, a qual tem origem em Wiring, e é essencialmente C/C++. O objetivo do projeto é criar ferramentas que são acessíveis, com baixo custo, flexíveis e fáceis de usar por qualquer pessoa. Principalmente para aqueles que não teriam alcance aos controladores mais sofisticados e de ferramentas mais complicadas. A Figura 12 apresenta um dos modelos mais comuns de Arduino, o Uno.

O Arduino foi escolhido como plataforma para a camada de controladores pela capacidade de processamento de seu microcontrolador ser bastante satisfatória para as aplicações de controle que são realizadas pela empresa. O Arduino também é facilmente programável, o que o destaca entre outros microcontroladores.

No padrão criado, o Arduino deve responder a três comandos que serão enviados pela porta serial:

1. Se for enviado o comando `r#`, o Arduino responderá para quem enviou a solicitação com os valores de todas as variáveis armazenadas nele naquele instante seguindo o seguinte padrão:

   V1;XXX;V2;XXX;V3;XXX;[...];V4;XXX

   onde V*i* são os nomes das variáveis e XXX os respectivos valores
2. Se for enviado o comando `w#` seguido de V1;XXX, o valor XXX será atribuído à variável V1 no Arduino e o valor das variáveis são novamente retornados
3. Se for enviado o comando `z#`, o Arduino responderá para quem enviou a solicitação com uma sequência de três caracteres, de modo que o primeiro será a letra A seguida de dois algarismos cuja soma é 10. Este comando serve para que o Arduino seja identificado como um autorizado a participar da rede Suna.

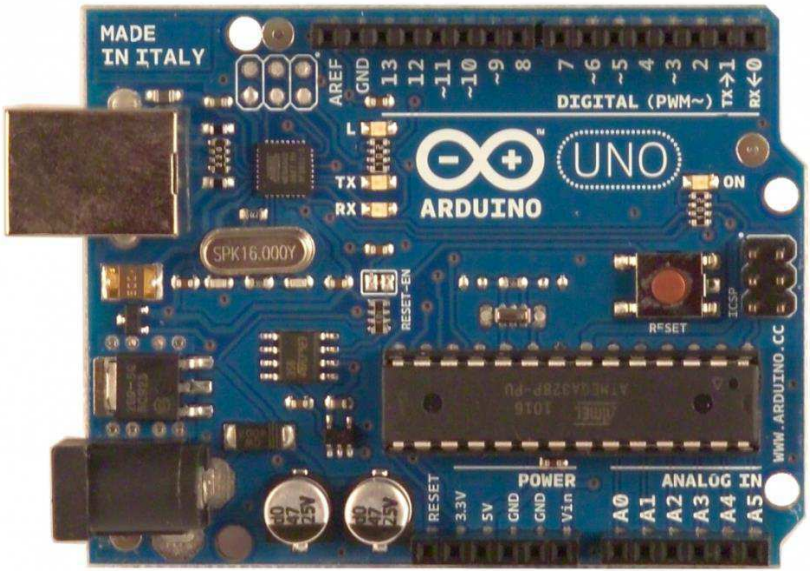

Figura 12 – Arduino Uno

3.3.4 Raspberry Pi

Raspberry Pi é um computador do tamanho de um cartão de crédito, que se conecta a um monitor de computador ou TV, e usa um teclado e um mouse padrão, desenvolvido no Reino Unido pela Fundação Raspberry Pi. Todo o *hardware* é integrado numa única placa. O computador é baseado em um *System on Chip* (SoC) Broadcom BCM2835, que inclui um processador ARM1176JZF-S de 700 MHz, GPU VideoCore IV e 512 MB de memória RAM em sua última revisão. O projeto não inclui uma memória não-volátil – como um disco rígido – mas possui uma entrada de cartão SD para armazenamento de dados.

O Raspberry Pi foi escolhido como plataforma para a camada de servidor porque ele pode facilmente ser programado como um computador comum que utiliza um sistema operacional baseado em Linux, além de ser barato e de pequeno porte, o que facilita bastante ao conectá-lo ao restante do sistema a ser monitorado, reduzindo o custo de produção.

O Raspberry Pi possui uma capacidade limitada de processamento se comparado com um PC comum. Por esse motivo, após vários testes de desempenho, foi concluído que ele deve ser utilizado com a função dedicada de administrar as conexões de Arduinos e PCs supervisórios e gerenciar os dados que são transmitidos entre essas outras duas camadas sem fazer manipulação nos mesmos.

As principais funções implementadas no servidor são:

- addConnection(): responsável por adicionar uma conexão *socket* com cada supervisório conectado a ele;

```
void Host::addConnection()
{
    QtcpSocket* connection = nextPendingConnection();
    connections.append(connection);
    Qbuffer* buffer = new Qbuffer(this);
    buffer->open(QIODevice::ReadWrite);
    buffers.insert(connection, buffer);
    connect(connection, SIGNAL(disconnected()), SLOT(removeConnection()));
    connect(connection, SIGNAL(readyRead()),    SLOT(receiveMessage()));

    escreverLog(tr("Cliente conectado: ") + connection->peerAddress().toString());
}
```

- removeConnection(): responsável por remover a conexão *socket* ao detectar a ausência de um dos supervisórios;

```
void Host::removeConnection()
{
    QtcpSocket* socket = static_cast<QtcpSocket*>(sender());
    Qbuffer* buffer = buffers.take(socket);
    buffer->close();
    buffer->deleteLater();
    connections.removeAll(socket);
    socket->deleteLater();

    escreverLog(tr("Cliente desconectado: ") + socket->peerAddress().toString());
}
```

- receiveMessage(): responsável por receber e interpretar a mensagem recebida pelo supervisório e interpretá-la e transmiti-la para os Arduinos;

```
void Host::receiveMessage()
{
    QtcpSocket* socket = static_cast<QtcpSocket*>(sender());
    QbyteArray ba = socket->readAll();
    Qstring msg = Qstring::fromAscii(ba.data());
    escreverLog(msg);

    if(msg == "LER_VARIAVEIS")
    {
        QbyteArray dados;

        foreach(QserialPort *qsp, listSerial)
        {
            qsp->write("r#");
            if(qsp->waitForReadyRead(timeout1))
                if(qsp->canReadLine())
                    dados=qsp->readLine();
        }

        socket->write(dados);
        escreverLog("dados: " + dados);
    }
    else if(msg.at(0) == 'w')
    {
        QbyteArray dados;

        QbyteArray ba1 = msg.toLocal8Bit();
        const char *c = ba1.data();

        foreach(QserialPort *qsp, listSerial)
        {
```

```
            qsp->write©;
            if(qsp->waitForReadyRead(timeout1))
                if(qsp->canReadLine())
                    dados=qsp->readLine();
        }

        socket->write(dados);
    }
    else if(msg.at(0) == 'a') // TESTE
    {
        QbyteArray dados;

        QbyteArray ba1 = msg.toLocal8Bit();
        const char *c = ba1.data();

        foreach(QserialPort *qsp, listSerial)
        {
            qsp->write©;
            if(qsp->waitForReadyRead(timeout1))
                if(qsp->canReadLine())
                    dados=qsp->readLine();
        }

        escreverLog(Qstring::fromAscii(dados.data()));
        socket->write(dados);
    }
}
```

- conectarSerial(): responsável por adicionar uma conexão *serial* com cada Arduino conectado a ele;

```
void Host::conectarSerial()
{
    serial = new QserialPort(this);
    serial->setPortName("ttyUSB0");
    serial->open(QIODevice::ReadWrite);
    serial->setBaudRate(115200);
    serial->setDataBits(QserialPort::Data8);
    serial->setParity(QserialPort::NoParity);
    serial->setStopBits(QserialPort::OneStop);
    serial->setFlowControl(QserialPort::NoFlowControl);
    verificaArduino();
    listSerial.append(serial);

    escreverLog(tr("Arduinos conectados: ") + Qstring::number(listSerial.size()));
}
```

- verificaArduino()

```
bool Host::verificaArduino()
{
    QbyteArray ba;
    Qstring msg;

    for(int i = 0; i < 5; i++)
    {
        serial->write("z#");

        if (serial->waitForBytesWritten(2000/*timeout1*/)) {
            if(serial->waitForReadyRead(2000/*timeout1*/)) {
                if(serial->canReadLine())
                    ba=serial->readLine();
```

```
                else escreverLog("=➔ canReadLine() == false"); }
            else escreverLog("=➔ waitForReadyRead() == false"); }
        else escreverLog("=➔ waitForBytesWritten() == false");

        msg = Qstring::fromAscii(ba.data());
        escreverLog(tr("Mensagem recebida: ") + msg);
    }

    Qchar verificadorArduino = 'A';

    if(msg.at(0) == verificadorArduino)
    {
        QstringList strList;
        strList = msg.split(verificadorArduino, Qstring::SkipEmptyParts);

        int  i, i1, i2;
        i = strList.at(0).toInt();
        i1 = i/10;
        i2 = i%10;

        if(i1+i2 == 10) return true;
        else return false;
    }
    else return false;
}
```

- escreverLog(): responsável por escrever em um arquivo .txt o log de várias ocorrências no programa

```
void Host::escreverLog(Qstring logString)
{
    QdateTime dateTime;
    Qfile logFile("logHost.txt");

    logFile.open(QIODevice::ReadWrite | QIODevice::Text);
    /*QbyteArray baRead = */logFile.readAll();
    Qstring dt = dateTime.currentDateTime().toString();
    Qstring linha = "\n<" +  dt + "> " + logString;
    QbyteArray ba/* = baRead*/;
    ba.append(linha.toLocal8Bit());
    char *send = ba.data();
    logFile.write(send);
}
```

3.3.5 Supervisório

O programa supervisório é implementado em PCs comuns e é o responsável pelo maior esforço computacional da rede, uma vez que é ele quem seleciona dentro de todas as variáveis aquelas que são importantes, além de realizar tarefas utilizando interface gráfica.

As principais funções implementadas no servidor são:

- toggleConnection(): responsável por conectar ou desconectar o supervisório do servidor

```
void Cliente::toggleConnection()
{
    if (socket->state() == QabstractSocket::UnconnectedState)
    {
        socket->connectToHost(ipAddress, PORTA);
        escreverLog(tr("Cliente conectado ao servidor ") + socket-
>peerAddress().toString());
    }
    else
    {
        socket->disconnectFromHost();
        escreverLog(tr("Cliente desconectado do servidor ") + socket-
>peerAddress().toString());
    }
}
```

- lerVariaveis(): responsável por enviar o comando de leitura das variáveis para o servidor

```
void Cliente::lerVariaveis()
{
        socket->write("LER_VARIAVEIS");
}
```

- escreverVariaveis(): responsável por enviar o comando de escrita de uma variável para o servidor

```
void Cliente::escreverVariaveis()
{
    QbyteArray ba1 = "w" + ui->txtEscrever->text().toLocal8Bit() + "#";
    const char *c = ba1.data();
    socket->write©;
}
```

- receiveMessage(): responsável por receber o nome e o valor de todas as variáveis

```
void Cliente::receiveMessage()
{
    qint64 bytes = buffer->write(socket->readAll());
    buffer->seek(buffer->pos() – bytes);

    while (buffer->canReadLine())
    {
        Qstring line = buffer->readLine();
        ui->txtLer->setText(line);
        criarListas(line);
    }
}
```

- escreverLog(): responsável por escrever em um arquivo .txt o log de várias ocorrências no programa

```
void Cliente::escreverLog(Qstring logString)
{
    QdateTime dateTime;
    Qfile logFile("logClient.txt");
```

```
    logFile.open(QIODevice::ReadWrite | QIODevice::Text);
    QbyteArray baRead = logFile.readAll();
    Qstring dt = dateTime.currentDateTime().toString();
    Qstring linha = "\n<" +  dt + "> " + logString;
    QbyteArray ba;
    ba.append(linha.toLocal8Bit());
    char *send = ba.data();
    logFile.write(send);
}
```

3.3.6 Elaboração de resumo técnico do padrão

Após o término do desenvolvimento do padrão, foi elaborado um resumo técnico que apresenta as especificações que já foram aqui mostradas. O resumo também mostra detalhes do funcionamento do padrão e observa que as vantagens do protocolo TCP foram incorporadas pelo padrão, tornando desnecessária a implementação de *handshaking* ou de sistemas de prioridades de uso do barramento, por exemplo.

3.3.7 Desenvolvimento de um programa supervisório

Na fase final do estágio, foi desenvolvido um programa supervisório que serviria de base para os próximos a serem desenvolvidos.

A Figura 13 apresenta a tela principal do supervisório. A Figura 14 apresenta a tela de configuração da conexão com o servidor. O programa pode gerar quantos gráficos forem necessários, como apresentado na Figura 15. Esses gráficos podem ser mostrados em janela ou abas, como apresentado na Figura 16. Na Figura 17 é possível ver que cada gráfico pode ser configurado de acordo com a espessura e cor da linha, título do gráfico, título dos eixos, etc. Além disso, nesta janela também é possível selecionar quais variáveis serão plotadas em cada gráfico.

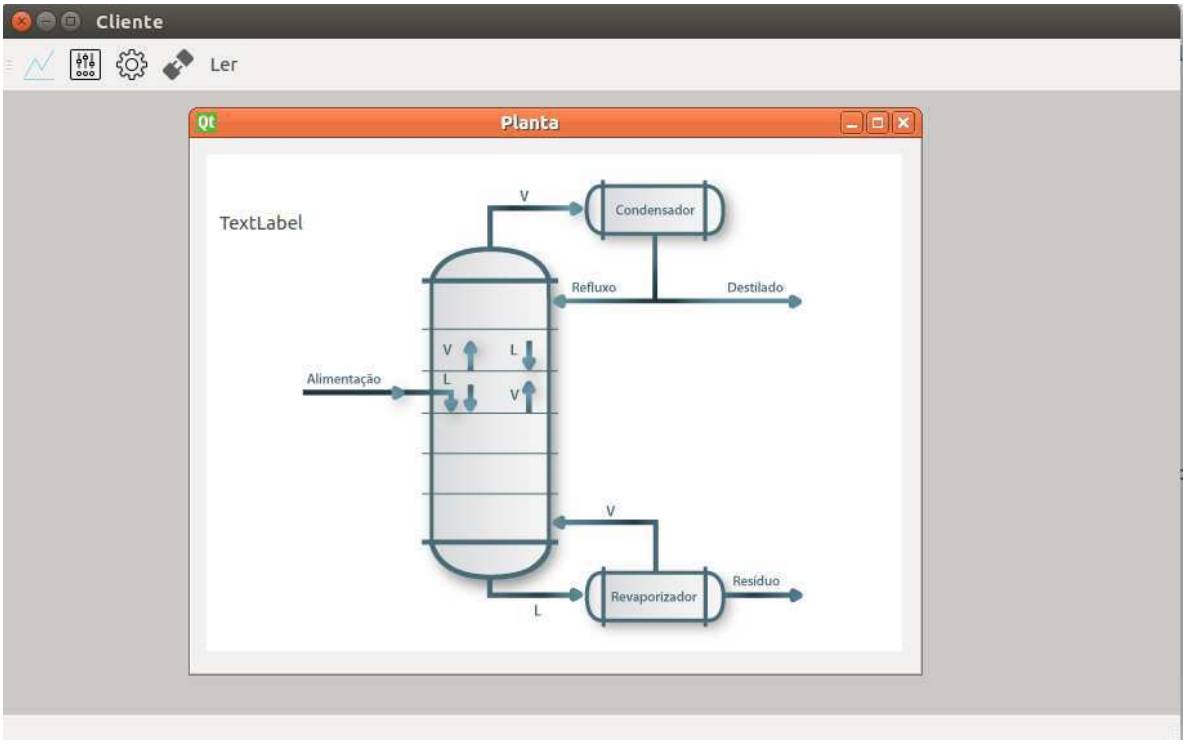

Figura 13 - Tela principal do supervisório

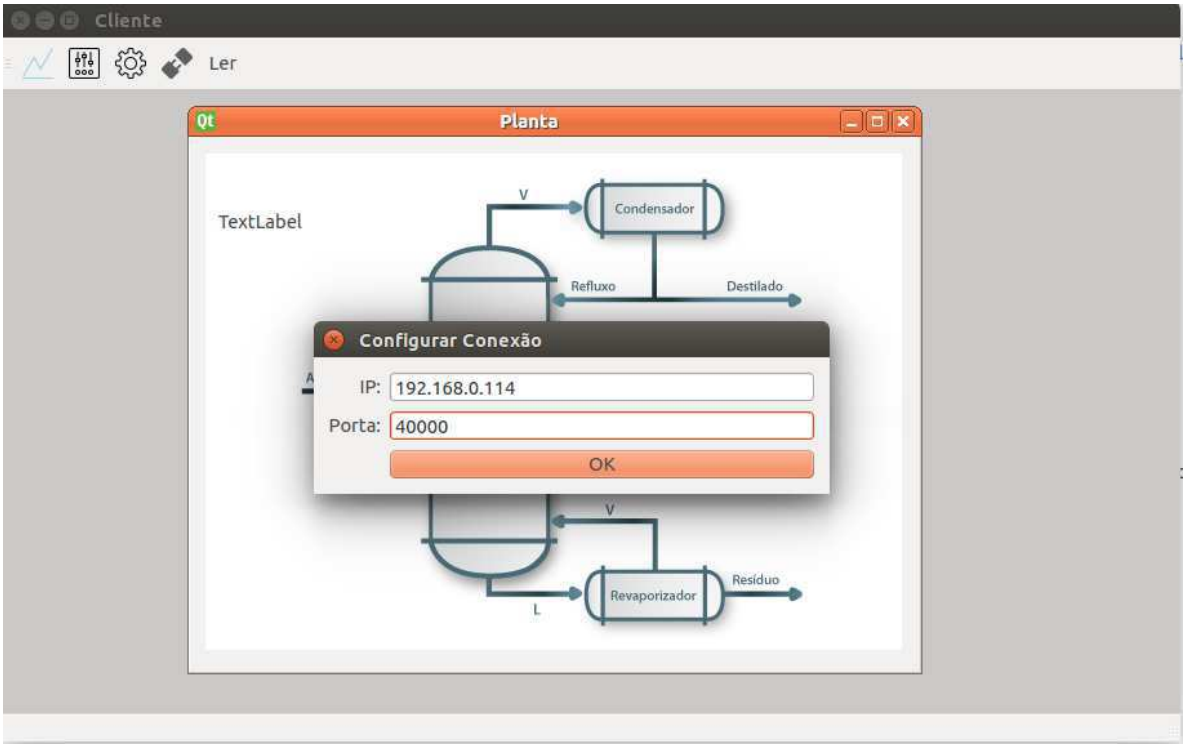

Figura 14 - Janela de configuração da conexão com o servidor

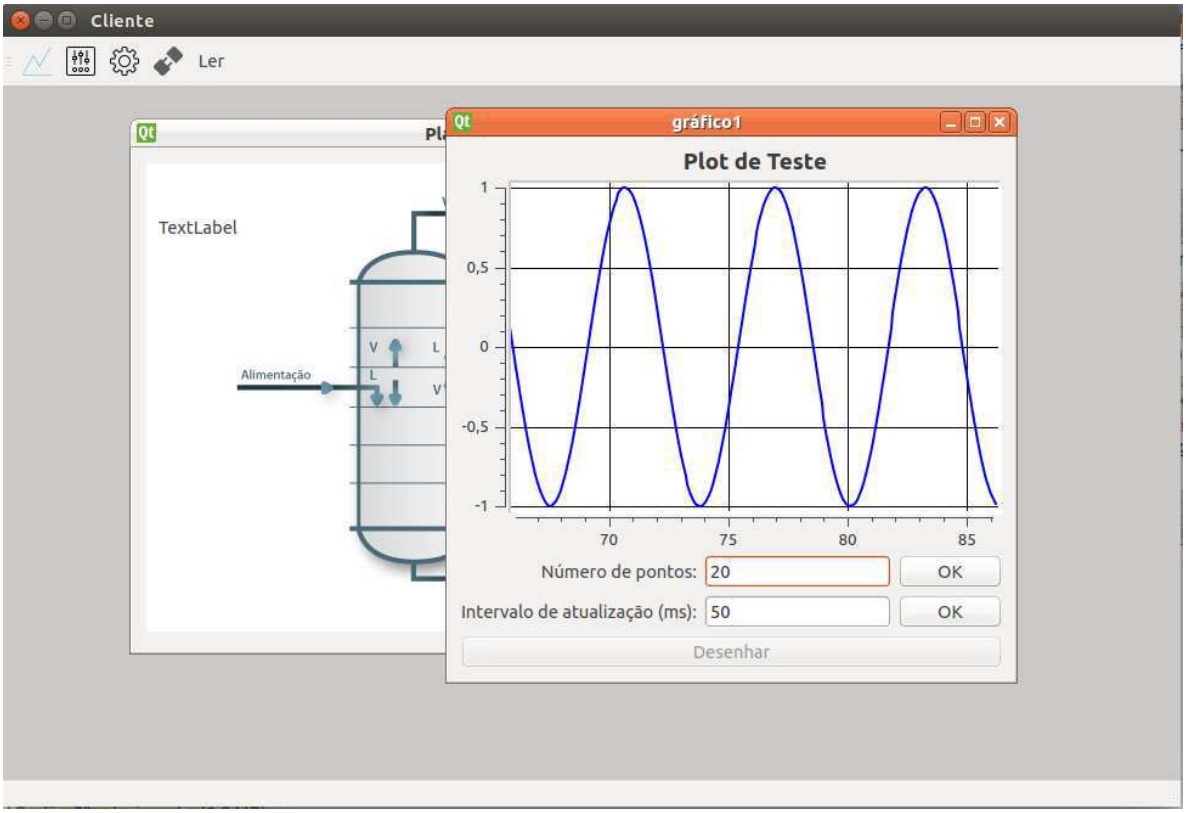

Figura 15 - Um dos gráficos que podem ser gerados

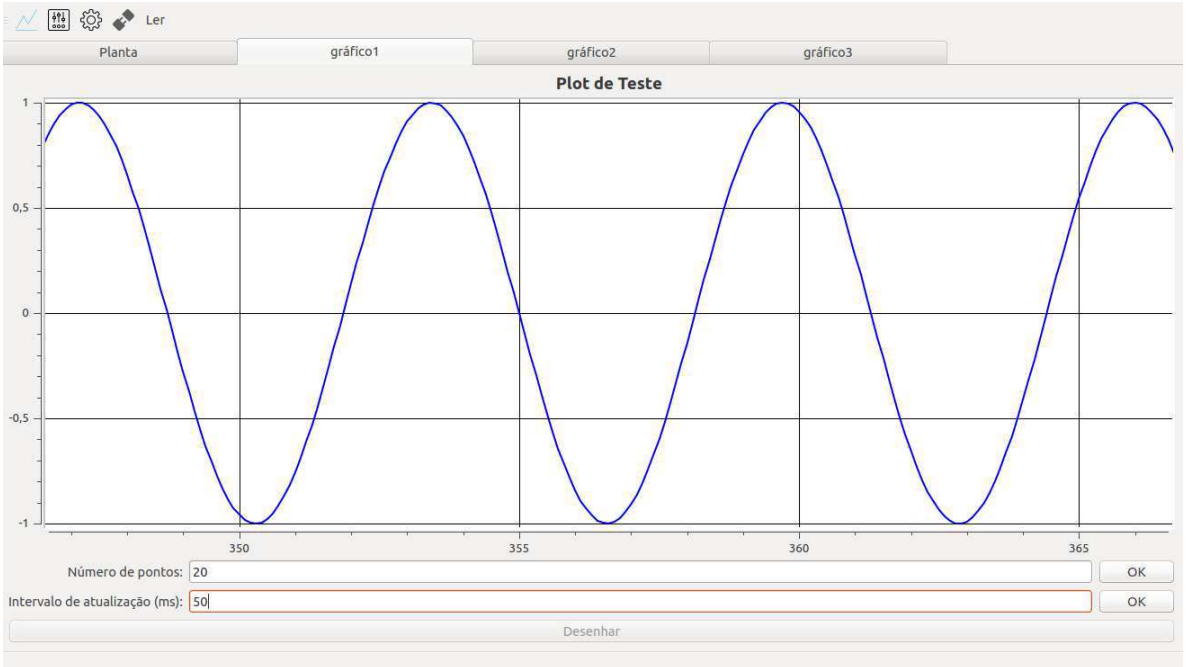

Figura 16 - O programa pode apresentar os gráficos em janela ou abas

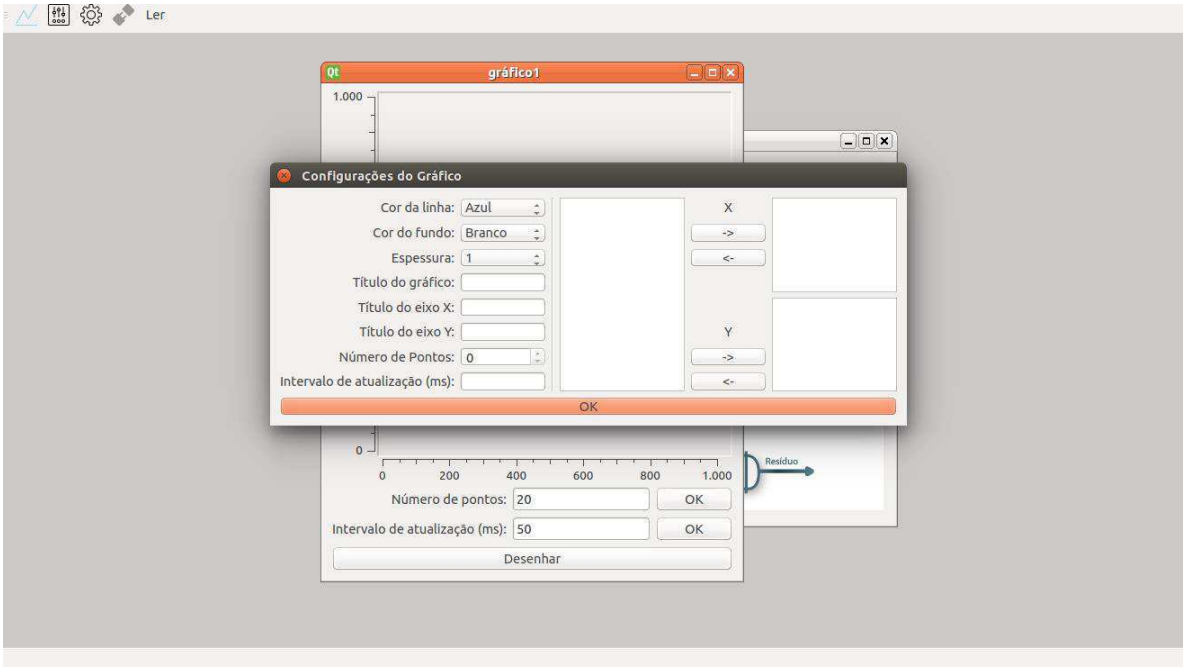

Figura 17 - Janela de configuração do gráfico

4 Conclusão

No período de estágio foram realizadas diversas atividades que permitiram ao aluno desenvolver os conhecimentos adquiridos durante o curso. Foram desenvolvidas também habilidades que não são exploradas durante o curso como a análise de custos e de viabilidade. Além disso, foi uma oportunidade de adquirir conhecimentos em ferramentas novas para o aluno, como a Qt, o Qwt e o Raspberry Pi. O estágio, portanto, pode ser classificado como uma experiência satisfatória.

Além disso, foi também uma oportunidade de se relacionar com profissionais de outras Engenharias, permitindo o aprimoramento da capacidade de trabalho em equipe e da multidisciplinaridade, além de trocar informações com pessoas que possuem diferentes tipos de conhecimento. Por todos estes aspectos, o estágio foi uma experiência bastante construtiva para a vida profissional do aluno.

O estágio também foi uma oportunidade para perceber que os conhecimentos adquiridos durante o curso são apenas uma base para o que é realmente necessário no mercado de trabalho. Foi necessário um esforço para adquirir certos conhecimentos que não são desenvolvidos no curso.

5 Referências Bibliográficas

- COUTINHO, Sebastião Araújo. Apresentação da Suna Engenharia. 2014. Disponível em: <http://sunaengenharia.com.br/index.php/pt/a-empresa/nossa-historia>. Acesso em: 07 dez. 2014.
- COUTINHO, Sebastião Araújo. Concentrador de Sólidos Solúveis. 2014. Disponível em: <http://sunaengenharia.com.br/index.php/pt/projetos/concentrador-de-solidos-soluveis>. Acesso em: 07 dez. 2014.
- COUTINHO, Sebastião Araújo. Softstarter monofásico. 2014. Disponível em: <http://sunaengenharia.com.br/index.php/pt/projetos/softstart-monofasico>. Acesso em: 07 dez. 2014.
- Potência Elétrica. 2014. Disponível em: <http://fisicaevestibular.com.br/exe_elt_2.htm>. Acesso em: 07 dez. 2014.
- GÓMEZ, Luis Alberto. Definição, Leis Básicas e Circuitos a Termopar. Florianópolis: Ecv, 2014.
- Qwt. 2014. Disponível em: <http://qwt.sourceforge.net/>. Acesso em: 07 dez. 2014.
- Arduino. 2014. Disponível em: <http://arduino.cc/>. Acesso em: 07 dez. 2014.