Controle e Supervisão de Sistema de Automação Utilizando Computação em Nuvem

José Antonio Toledo Júnior * **Guilherme Gomes da Silva** **

* *Rod. MG 443 KM 7, Ouro Branco - MG, 36420-000, Brasil (e-mail: jatoledojunior@gmail.com)*
** *Rod. MG 443 KM 7, Ouro Branco - MG, 36420-000, Brasil (e-mail: guilhermegomes@ufsj.edu.br)*

Abstract: Today's most commercialized control and supervisory systems for industrial automation already support TCP/IP communication for the development of remote schemas and data traffic over the Internet. SCADA systems are, in large majority, software that is installed locally in reserved computers, which makes it impossible to use them far from these places. This work then proposes the development of a complete control and supervision system of a simplified industrial plant with the following characteristics: the supervisory system must be accessible from any computer connected to the Internet without installation of specific software; the control system must be modular and expandable; and the data management, including the storage and availability of them, must be done through the cloud computing without real-time constraints.

Resumo: Os sistemas atuais mais comercializados de controle e supervisão para automação industrial já suportam comunicação TCP/IP para desenvolvimento remoto e tráfego de dados pela Internet. Mesmo assim, os sistemas SCADA são, em grande maioria, softwares proprietários que necessitam estar instalados localmente em computadores reservados, o que inviabiliza o uso dos mesmos longe desses locais. Este trabalho vem, então, propor o desenvolvimento de um sistema completo de controle e supervisão de uma planta industrial simplificada com as seguintes características: o sistema supervisório deve ser acessível de qualquer dispositivo conectado à Internet sem necessidade de instalação de softwares específicos; o sistema de controle deve ser de caráter modular e expansível; e a tratativa dos dados, incluindo armazenamento e disponibilização dos mesmos, deve ser feita através de computação em nuvem sem restrições de tempo real.

Keywords: Industrial Automation; Cloud Computing; Control System; Supervisory System.
Palavras-chaves: Automação Industrial; Computação em Nuvem; Sistema de Controle; Sistema Supervisório.

1. INTRODUÇÃO

A automação é a operação de uma máquina ou grupo de máquinas, local ou remotamente, com a mínima interferência do operador humano. Isso significa ter um mecanismo de atuação própria para fazer a ação requerida em tempo determinado ou em resposta a certas condições.

O PLC (*Programmable Logic Controller*) é o coração da automação industrial. Trata-se de um equipamento eletrônico, digital, microprocessado, que pode controlar um processo ou uma máquina e ser programado ou reprogramado rapidamente. Diferentemente desse controlador, o sistema supervisório, ou SCADA (*Supervisory Control And Data Acquisition*), é um conjunto de hardware e software que permite ao operador ter acesso a informações do processo de forma clara e objetiva.

Em aplicações industriais, as informações do SCADA normalmente são providas pelo PLC e devem ser apresentadas em formato padronizado e amigável, permitindo uma eficiente interação com o processo (Vilela and Vidal, 2003). Um exemplo de arquitetura da automação clássica está apresentada na Figura 1.

De acordo com McHugh (2003), a Internet é baseada no protocolo TCP/IP (*Transmission Control Protocol / Internet Protocol*) de um conjunto de protocolos de comunicação. Diante dessa realidade, os sistemas atuais mais comercializados de controle e supervisão para automação industrial já suportam esse tipo de comunicação para desenvolvimento remoto e tráfego de dados pela Internet.

Mesmo assim, os sistemas SCADA são, em grande maioria, softwares proprietários que necessitam estar instalados localmente em computadores reservados. Isso inviabiliza o uso longe dos locais específicos, necessitando de pessoas para informar por rádio as condições das variáveis e do processo para os mantenedores e operadores que estão no chão-de-fábrica realizando alguma atividade.

DOI: 10.17648/sbai-2019-111109

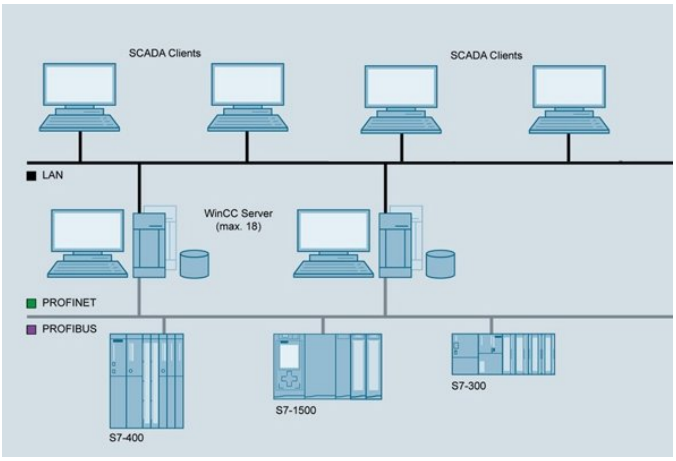

Figura 1. Exemplo de arquitetura de automação clássica. Fonte: (Siemens, n.d.)

Este trabalho vem, então, propor o desenvolvimento de um sistema completo de controle e supervisão de uma planta industrial simplificada utilizando tecnologias não tradicionais que contornem as dificuldades apresentadas. O projeto deverá ter as seguintes características:

- O sistema supervisório deve ser acessível de qualquer dispositivo conectado à Internet sem necessidade de instalação de softwares específicos;
- O sistema de controle deve ser de caráter modular e expansível;
- A tratativa dos dados, incluindo armazenamento e disponibilização dos mesmos, deve ser feita através de computação em nuvem sem restrições de tempo real.

Serão utilizados um Arduino, simulando o PLC que adquire e controla os dados, e um Raspberry para realizar a troca deles com o banco de dados, o serviço GAS (*Google App Script*) de computação em nuvem e o sistema supervisório criado através de programação web.

2. REVISÃO BIBLIOGRÁFICA

A computação em nuvem, também chamada de *Utility Computing* ou *Cloud Computing*, surge da necessidade de construir infraestruturas de TI (*Tecnologia da Informação*) complexas, onde os usuários têm que realizar instalação, configuração e atualização de sistemas de software. Em geral, os recursos de computação e hardware são propensos a ficarem obsoletos rapidamente e a utilização de plataformas computacionais de terceiros é uma solução inteligente para contornar o problema. Na computação em nuvem os recursos de TI são fornecidos como um serviço, permitindo que os usuários o acessem sob demanda e paguem apenas pelos recursos recebidos, independente de localização e sem a necessidade de conhecimento sobre a tecnologia utilizada.

O objetivo é fornecer componentes básicos como armazenamento, processamento e largura de banda de uma rede como uma "mercadoria" através de provedores especializados com um baixo custo por unidade utilizada. Usuários não precisam se preocupar com escalabilidade e backups, pois a capacidade de armazenamento fornecida é praticamente infinita e o provedor é responsável por substituir os dados em tempo hábil por meio de réplicas caso os componentes falhem. O serviço propõe fornecer disponibilidade total, isto é, os usuários podem ler e gravar dados a qualquer momento, sem nunca serem bloqueados; os tempos de resposta são quase constantes e não dependem do número de acessos simultâneos, do tamanho do banco de dados ou de qualquer parâmetro do sistema (Sousa et al., 2011).

2.1 Modelos de Serviços

Sousa et al. (2011) define três modelos de serviços no ambiente de computação em nuvem:

SaaS (*Software as a Service*): Proporciona sistemas de software disponíveis por meio da Internet e acessíveis a partir de vários dispositivos através de uma interface *thin client*, como um navegador web. O usuário não administra ou controla a infraestrutura subjacente, incluindo rede, servidores, sistema operacional ou armazenamento, somente configurações específicas. O GAS é um exemplo de serviço SaaS.

PaaS (*Platform as a Service*): Fornece sistema operacional, linguagens de programação e ambientes de desenvolvimento para as aplicações, auxiliando a implementação de sistemas de software. Assim, como no SaaS, o usuário não administra ou controla a infraestrutura subjacente, mas tem controle sobre as aplicações implantadas e, possivelmente, as configurações de aplicações hospedadas nesta infraestrutura. O GAE (*Google App Engine*) e Microsoft Azure são exemplos de serviços PaaS.

IaaS (*Infrastructure as a Service*): Em geral, o usuário não administra ou controla a infraestrutura da nuvem, mas tem controle sobre os sistemas operacionais, armazenamento, aplicativos implantados e, eventualmente, seleciona componentes de rede, tais como firewalls. O Amazon EC2 (*Elastic Compute Cloud*) é um exemplo de serviço IaaS.

2.2 Google App Script

Criado e mantido pelo Google, o GAS busca integrar funcionalidades do Google Apps através de scripts desenvolvidos em JavaScript. A linguagem conta com serviços do G Suite, que inclui Calendar, Document, Drive, Gmail, Maps, Spreadsheet, entre outros. O principal objetivo é automatizar tarefas que integrem um ou mais serviços do Google (Vieira, 2014).

Um recurso interessante do GAS é desenvolver scripts que apresentam um *endpoint* na rede por meio do protocolo HTTP (*Hypertext Transfer Protocol*), isto é, respondem a requisições do tipo *get* e *post*. Com esse mecanismo é possível criar uma API (*Application Programming Interface*), ou seja, um aplicativo web para envio ou captura de dados. Uma aplicação cliente faz uma chamada HTTP para o GAS que faz o tratamento da requisição e devolve a resposta ao cliente (FazerLab, n.d.).

3. METODOLOGIA

A planta escolhida foi de um sistema de limpeza de torre de resfriamento, utilizada para remover particulados

DOI: 10.17648/sbai-2019-111109

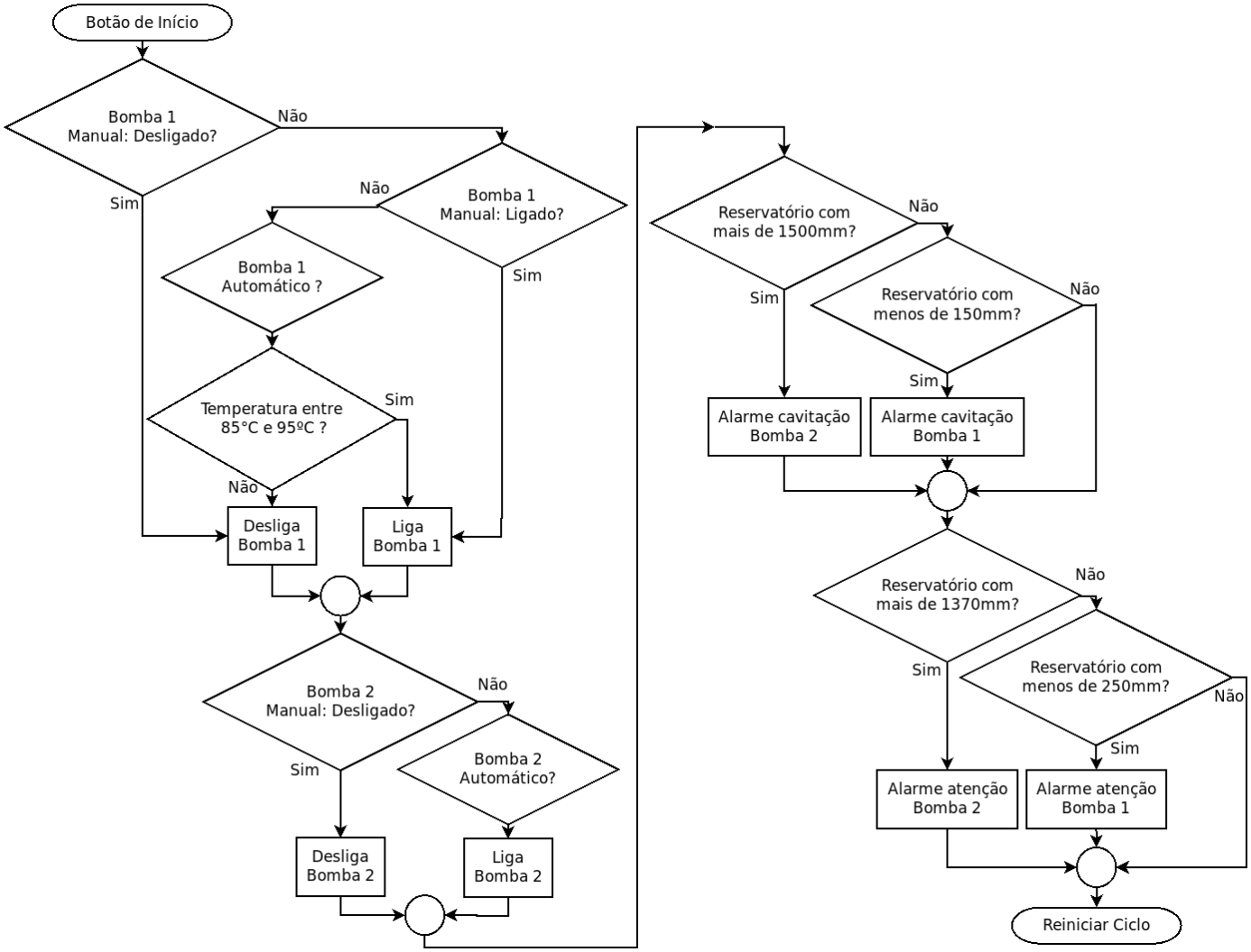

Figura 2. Fluxograma de funcionamento normal do sistema.

encrustados em trocadores de calor. A limpeza é realizada através de um óleo específico mantido num reservatório horizontal a uma temperatura próxima de 90°C devido à serpentina com vapor superaquecido, controlada por uma válvula automática. O óleo do reservatório é bombeado para a torre através da bomba 1 e retorna devido à bomba 2.

O funcionamento do sistema segue os fluxogramas apresentados nas Figuras 2, 3 e 4, importantes para definir quais dados serão controlados e exibidos, principalmente de setores críticos.

Definidos os fluxogramas, o esquemático do sistema de controle é apresentado na Figura 5 e a comunicação entre as partes do projeto deve ser baseada na estrutura da Figura 6.

4. RESULTADOS E DISCUSSÕES

No tópico 4.1 estão definidas as tags do processo e a lógica de controle do Arduino seguindo os fluxogramas elaborados; nos tópicos 4.2 e 4.3 são discursadas as rotinas do Raspberry para troca de dados com o Arduino e o GAS, respectivamente; no tópico 4.4 encontra-se informações

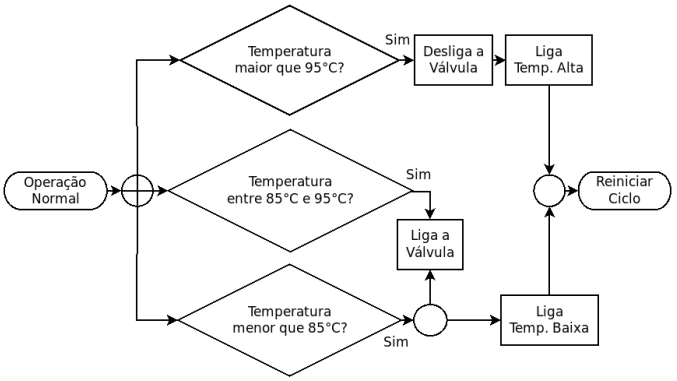

Figura 3. Fluxograma de controle de temperatura do óleo.

sobre o funcionamento do banco de dados e os aplicativos web desenvolvidos; e por fim o tópico 4.5 apresenta a interface supervisória da planta.

4.1 Sistema de Controle com o Arduino

A Tabela 1 contém apenas as variáveis de controle do processo, enquanto a Tabela 2 contém as de controle e

DOI: 10.17648/sbai-2019-111109

Tabela 1. Mapa de variáveis apenas de controle.

Nome	Endereço	Tipo	Descrição
LOS_EMERGENCY	A9	Bool	Sinaleiro sonoro de emergência
POS_EMERGENCY	A7	Bool	Botoeira de emergência
FBK_OK_SYSTEM	M0	Bool	Verificação de sistema ok
KYV_PUMP1_START	D40	Bool	Relé de acionamento da bomba 1
KYV_PUMP2_START	D41	Bool	Relé de acionamento da bomba 2
SNS_R_OIL_LEVEL	D44/45	Float	Sensor ultrassônico nível do óleo reservatório
SNS_R_OIL_TEMPERATURE	A5	Int	Sensor potenciômetro temperatura do óleo reservatório

Tabela 2. Mapa de variáveis de controle e armazenamento.

Nome	Endereço	Tipo	Descrição
COM_MODE_PUMP1	QW0	Int	Comando modo de operação bomba 1
COM_MODE_PUMP2	QW1	Int	Comando modo de operação bomba 2
COM_START	Q0.0	Bool	Comando de partida do processo
FBK_PUMP1_ALARM	I0.0	Bool	Alarme de possível falha da bomba 1
FBK_PUMP1_FAULT	I0.1	Bool	Falha da bomba 1
FBK_PUMP1_START	I0.2	Bool	Comando de acionamento da bomba 1
FBK_PUMP2_ALARM	I1.0	Bool	Alarme de possível falha da bomba 2
FBK_PUMP2_FAULT	I1.1	Bool	Falha da bomba 2
FBK_PUMP2_START	I1.2	Bool	Comando de acionamento da bomba 2
FBK_T_OIL_LEVEL	IW0	Float	Cálculo nível de óleo torre
FBK_R_OIL_LEVEL	IW1	Float	Cálculo nível de óleo reservatório
FBK_R_OIL_TEMPERATURE	IW2	Float	Cálculo temperatura do óleo reservatório
FBK_SUPERHEATED_STEAM	I2.0	Bool	Válv. controle de vapor superaquecido reservatório
FBK_HLEVEL	I2.1	Bool	Nível alto de óleo no reservatório
FBK_HTEMP	I2.2	Bool	Temperatura alta do óleo no reservatório
FBK_LLEVEL	I2.3	Bool	Nível baixo de óleo no reservatório
FBK_LTEMP	I2.4	Bool	Temperatura baixa do óleo no reservatório
FBK_EMERGENCY	I3.0	Bool	Sinal de emergência

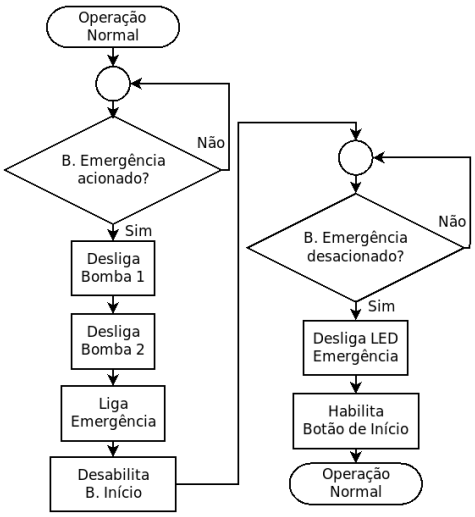

Figura 4. Fluxograma de controle de emergência.

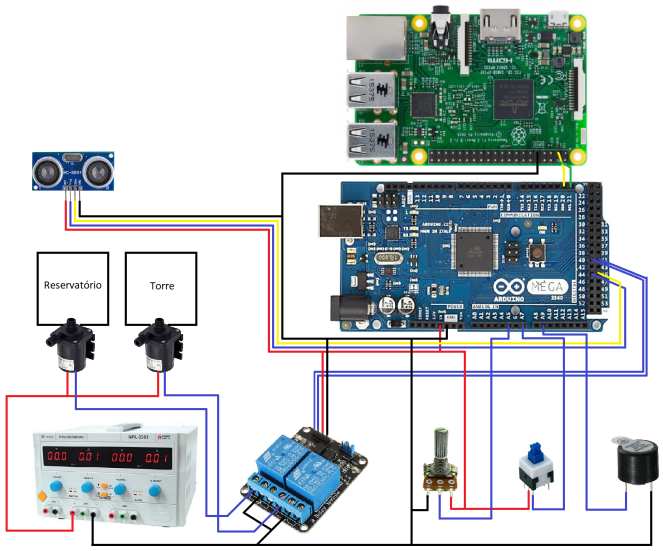

Figura 5. Esquemático do sistema de controle.

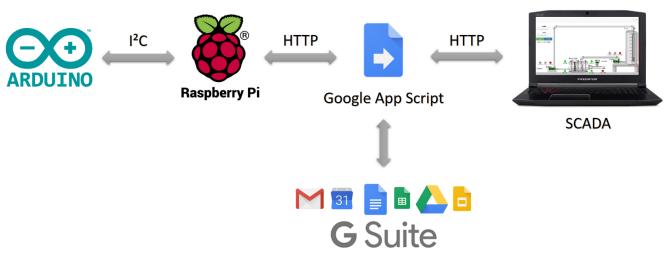

Figura 6. Esquemático de comunicação.

armazenamento. A partir delas, toda a lógica foi feita em 4 rotinas no Arduino:

main_routine(): Inicia o processo ao pressionar o botão de início no supervisório e faz o gerenciamento da situação emergência, não permitindo ligar as bombas em caso de anormalidade.

acionamento(): Controla as bombas dependendo de seus modos de funcionamento (Automático, Manual - Ligado e Manual - Desligado). Elas podem entrar em falha e desligar com o objetivo de evitar cavitação, sendo relacionadas somente quando a condição normal de operação for restabelecida. A bomba 1 desarma para nível do reservatório

DOI: 10.17648/sbai-2019-111109

abaixo de 150mm e a bomba 2 para nível da torre abaixo de 72mm.

reservatorio(): Controla o reservatório no que tange ao nível, obtido pelo sensor ultrassônico, e à temperatura, simulada pelo potenciômetro. Assim é possível acionar indicadores de nível e de temperatura alto/baixo, além de abrir/fechar a válvula de vapor superaquecido e colocar a bomba 1 em estado de alarme caso o óleo esteja abaixo de 250mm.

torre(): Controla o nível da torre de resfriamento, que é calculado através da conservação de massa do sistema, ou seja, a quantidade de óleo que sai do reservatório entra na torre e vice-versa. A rotina gera um alarme na bomba 2 quando o nível na torre está abaixo de 164mm.

4.2 Troca de Dados Arduino-Raspberry

As informações fluem entre Arduino e Raspberry via comunicação I^2C (*Inter-Integrated Circuit*), sendo o Raspberry o mestre e o Arduino o escravo.

No Arduino a comunicação é aberta com Wire.begin() e as escutas por Wire.onReceive() e Wire.onRequest(), passando as funções de receber e enviar dados, respectivamente, enquanto no Raspberry é feita pela classe I2CProtocol implementada. O construtor dessa classe recebe o endereço hexadecimal do Arduino e define o barramento através da função smbus.SMBus(), além do tamanho e offset para o vetor de dados. Dois métodos são criados, read_block_data() e write_block_data(), para respectivamente receber e enviar um bloco de informações.

Os equipamentos trocam um vetor de até 32 bytes do tipo char contendo o valor da variável, que deve ser tratado posteriormente. As tags são transmitidas em uma sequência previamente acordada, sendo necessário realizar modificações no receptor caso haja alguma modificação no emissor.

Todas as variáveis da Tabela 2 são enviadas do Arduino para o Raspberry, porém apenas as três primeiras vão do Raspberry para o Arduino e, para esse caso específico, a informação é agrupada e transmitida em apenas um vetor de dados.

4.3 Troca de dados Raspberry-GAS

As variáveis que foram recebidos do Arduino precisam ser historiadas no banco de dados na nuvem do Google e acessadas pelo supervisório. Dados que devem ser atualizados no banco são passados através do método *post*, enquanto a captura de informação é feita pelo método *get*. Qualquer modificação no banco necessita de usuário e senha com permissão, informações essas passadas no corpo da requisição e codificadas com Base64.

Ao iniciar o Raspberry uma sincronização é feita pela classe SyncVariables para obter do banco de dados todas as variáveis. Elas serão salvas numa lista de objetos Variable, que tem os seguintes atributos: name, description, tag, type, eng_unit, l_limit, h_limit, ll_limit, hh_limit e last_value. Após isso, a cada 2 segundos o programa pega do banco apenas o último valor das variáveis de comando do Arduino, através da classe GetRequestPLC, e salva nesse banco os últimos valores das variáveis de controle e armazenamento, através da classe PostRequestPLC.

Todas as classes criadas que tratam de requisições HTTP são inicializadas com a URL (*Uniform Resource Locator*) do respectivo aplicativo web, e toda a informação trocada é feita através do objeto JSON (*JavaScript Object Notation*). Para evitar que a *thread* fique travada durante a chamada, utilizou-se programação paralela disponível pela biblioteca *threading*.

4.4 Banco de Dados e Aplicativos Web com o GAS

Com uma conta Google criou-se uma planilha denominada db.gsheet com 3 abas:

credentials: Contém usuário, senha, e-mail, data de criação e grupo das contas cadastradas.

variables_info: Contém os dados das variáveis utilizadas no sistema SCADA. Os últimos valores das tags do modo de funcionamento das bombas e do botão de Iniciar Processo são atualizados pelo sistema supervisório, enquanto o restante através do processo de controle.

variables_hist: Contém verticalmente o histórico das variáveis, sendo novos dados inseridos na linha superior e alimentados unicamente pelo sistema de controle.

Um script do Google pode ser criado individualmente ou estar associado a uma planilha. Foram utilizados 5 scripts individuais:

O script **sync_variables** envia num vetor as informações das variáveis do banco quando uma chamada for feita. Da mesma forma, **requests_plc** e **requests_scada** têm métodos para enviar e receber os últimos valores das variáveis, incluindo a função de autenticação. Foram implementados também **hist_requests_scada**, que envia o histórico de valores da variável com o nome passado por parâmetro na requisição, e *auth_user*, que é responsável por receber o usuário e senha pelo método *post* e verificar a permissão dele.

Ao publicar os scripts como aplicativos web, basta ter acesso à URL do aplicativo que os métodos doGet() e doPost() responderão às requisições que chegarem, enviando ou recebendo dados de acordo com a lógica desenvolvida e a permissão de cada usuário. É possível ainda que um script seja programado para executar uma determinada função de tempos em tempos.

Através dos serviços do G Suite, com poucas linhas de código, é possível mandar e-mail alertando sobre uma parada inesperada do processo, enviar um relatório de produção para a gerência, armazenar todos os documentos técnicos em uma pasta compartilhada, entre outras tarefas que automatizam e tratam melhor os dados. No projeto, três variáveis são analisadas (FBK_PUMP1_FAULT, FBK_PUMP2_FAULT e FBK_EMERGENCY) e, caso necessário, um e-mail é enviado para os usuários do grupo de administradores dizendo que o processo foi interrompido devido àquela variável.

DOI: 10.17648/sbai-2019-111109

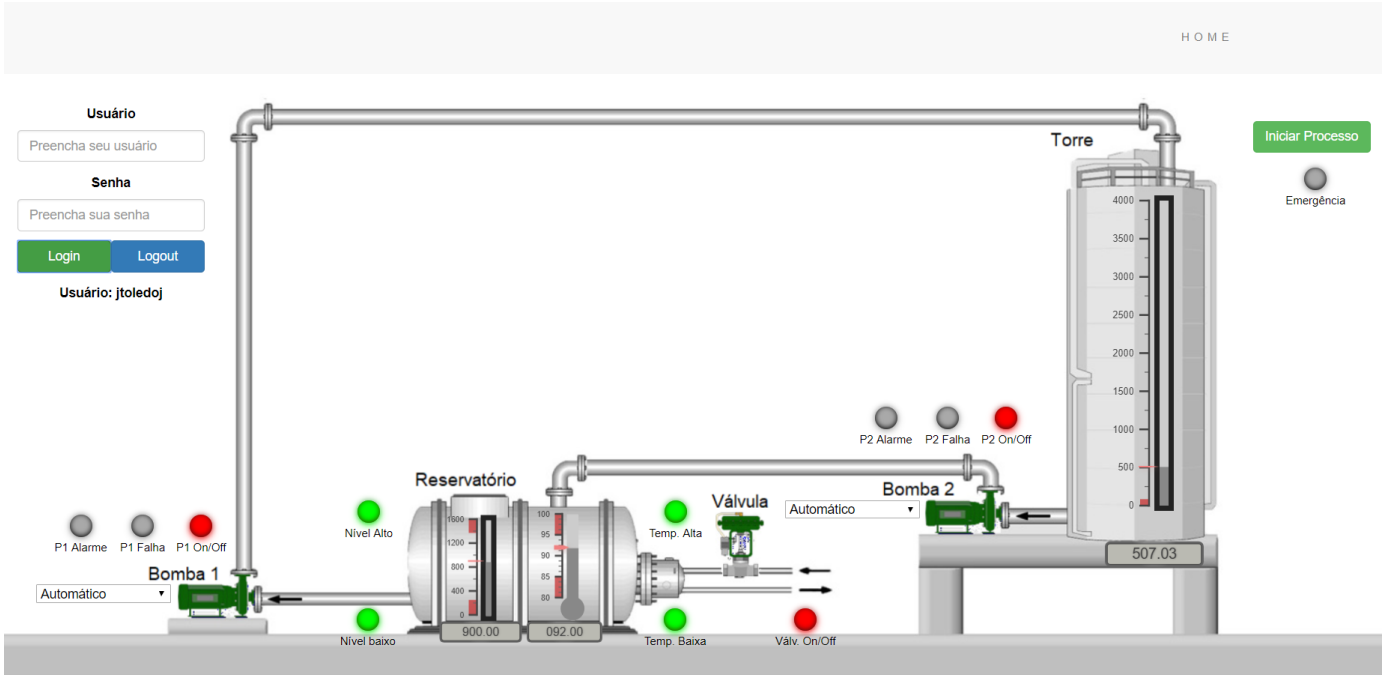

Figura 7. Sinótico para desktop do sistema proposto.

4.5 Sistema de Supervisão via Web

O sistema supervisório está exibido na Figura 7. O reservatório, a torre, as bombas e o encanamento são elementos de uma imagem de fundo estática, enquanto o restante é dinâmico com o processo.

Os campos de autenticação são formados pela tag <form> e somente com usuários logados no sistema que o botão de início e os modos de funcionamento das bombas surtem efeito. O botão de Login chama a função login() do JavaScript que realiza a requisição, através de jQuery, para o aplicativo auth_user, exibindo êxito ou falha ao logar via pop-up com a função alert(). O usuário fica ativo até o mesmo pressionar o botão Logout, o qual chama a função logout().

Os LEDs são tags estilizadas com as classes implementadas. Para cada valor das variáveis, a classe atual de estilização é removida e outra é adicionada. O padrão de cores para os indicadores é vermelho se o bit for verdadeiro (1) e verde se falso (0); para os alertas é amarelo intermitente se verdadeiro (1) e cinza se falso (0); e para as falhas ou emergência é vermelho intermitente se verdadeiro (1) e cinza se falso (0);

Os gauges de nível e temperatura são tags <canvas> desenhados através da biblioteca Canvas-Gauge, onde são instanciados e parametrizados os LinearGauges. Ao clicar neles um gráfico criado através da biblioteca Chart.js é exibido abaixo do sinótico. Para isso, a função atualiza_grafico() é chamada recebendo o nome da variável do gauge que foi selecionado e uma requisição para o aplicativo hist_requests_scada é feita.

As funções recebe_dados() e envia_dados() são executadas a cada 1 segundo a fim de atualizar o valor das variáveis no supervisório e enviar os parâmetros setados. O sistema é totalmente responsivo, como mostra a Figura 8, o que permite a utilização fácil e rápida no chão-de-fábrica ou em uma viagem a negócios, por exemplo.

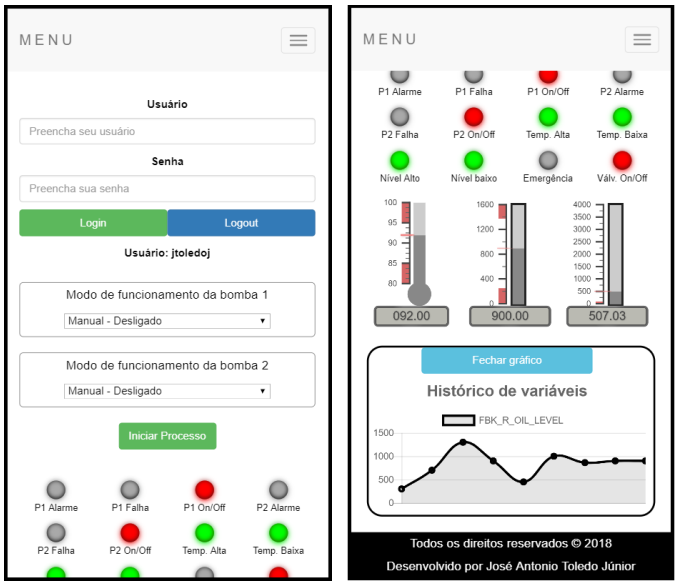

Figura 8. Sinótico para mobile do sistema proposto.

Para entrar na aplicação a partir de qualquer dispositivo conectado à Internet é necessário hospedá-la em um provedor de sites, como a HostGator e a Uol Host, onde um endereço IP real e URL são definidos para acesso.

5. CONCLUSÃO

O presente trabalho integrou diversas tecnologias e conceitos de programação e automação para obter seus resultados. Assimilar tantas ferramentas, linguagens e metodologias distintas é um processo difícil e demorado, mas foi sendo individualmente dominado e funcionou muito

DOI: 10.17648/sbai-2019-111109

bem em conjunto com as partes. Todo o arquivo fonte do projeto pode ser baixado em (Github, 2018).

A ideia do trabalho de desenvolver um sistema completo de controle e supervisão de uma planta industrial simplificada cumpre os objetivos propostos permitindo: 1. supervisão através de qualquer dispositivo conectado à Internet necessitando apenas do navegador web; 2. controle flexível do processo com o Arduino e Raspberry; 3. tratativa dos dados através de computação em nuvem, sem nenhuma restrição de tempo real, utilizando as diversas ferramentas disponíveis pelo GAS.

As partes do projeto são mutualmente independentes, ou seja, caso for necessário utilizar um PLC industrial no lugar do conjunto Arduino-Raspberry, o serviço de computação em nuvem e o sistema supervisório continuarão válidos desde que o PLC consiga realizar todas as operações da forma aqui descrita.

O processo industrial escolhido, mesmo que simplificado, contém intertravamentos para não permitir danos aos equipamentos, além de funções de controle de operação normal, temperatura e situações de emergência integradas.

O Arduino se mostrou eficiente no processamento dos dados de controle e o Raspberry realizou com facilidade a troca de informações com o banco de dados na nuvem. Fica proposto de melhoria para o sistema definir um protocolo de transmissão de dados com a comunicação I^2C que elimine a necessidade de conhecer a sequência das informações enviadas entre as duas plataformas.

Os recursos oferecidos pelo GAS têm grande utilidade na indústria a fim de automatizar tarefas rotineiras, porém a tentativa de substituição de um banco de dados tradicional por uma planilha do Google não foi a melhor escolha, pois é gasto um tempo razoavelmente grande para se escrever um dado na planilha - aproximadamente 250ms.

O sistema supervisório teve seu sinótico muito fidedigno ao processo real e é responsivo a diversos dispositivos. A utilização de editores de site é uma ótima oportunidade para criar várias páginas web de supervisão, desde que a ferramenta de edição seja flexível e permita acessar o código fonte.

A questão da segurança da informação trafegada não foi o foco deste projeto, mas é de fundamental importância para um sistema real. Aqui todos os dados de autenticação foram passados no corpo da requisição com apenas uma codificação Base64, dessa forma fica de sugestão para melhorias o aumento de complexidade na segurança dos dados e a utilização de padrões de autenticação já bem definidos na comunidade de desenvolvedores, como o SHA-256.

REFERÊNCIAS

FazerLab (n.d.). *Dados em tempo real com planilha do Google Docs*. Available at: <http://wp.me/p67Df7-cI>. [Accessed 20 Nov. 2018].

Github (2018). *Automacao industrial*. Available at: <https://github.com/JTOLEDOJ/Automacao-Industrial/>. [Accessed 22 Nov. 2018].

McHugh, D. (2003). Implementing tcp/ip in scada systems. 66. Available at: <https://research.thea.ie/handle/20.500.12065/398>. [Accessed 20 Nov. 2018].

Siemens (n.d.). *WinCC/Server*. Available at: <https://w3.siemens.com/mcms/human-machine-interface/en/visualization-software/scada/wincc-options/wincc-server/pages/default.aspx>. [Accessed 04 Dec. 2018].

Sousa, F.R.C., Moreira, L.O., Macêdo, J.A.F., and Machado, J.C. (2011). Gerenciamento de dados em nuvem: conceitos, sistemas e desafios. 40. Available at: <http://200.17.137.109:8081/novobsi/Members/josino/fundamentos-de-banco-de-dados/2012.1/Gerenciamento_Dados_Nuvem.pdf>. [Accessed 21 Nov. 2018].

Vieira, J.C. (2014). Linguagem de programação para nuvem: experiências com google app script. 62. Available at: <https://pt.scribd.com/document/377361848/TG364-Julio-Cesar-Vieira>. [Accessed 21 Nov. 2018].

Vilela, P.S.C. and Vidal, F.J.T. (2003). Automação industrial. 5. Available at: <https://www.dca.ufrn.br/~affonso/FTP/DCA447/trabalho1/trabalho1_19.pdf>. [Accessed 20 Nov. 2018].

DOI: 10.17648/sbai-2019-111109