UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE ESPECIALIZAÇÃO EM INTERNET DAS COISAS

GUILHERME THIEMANN

SISTEMA SUPERVISÓRIO DE INFRAESTRUTURA HOSPITALAR

MONOGRAFIA DE ESPECIALIZAÇÃO

CURITIBA
2020

GUILHERME THIEMANN

SISTEMA SUPERVISÓRIO DE INFRAESTRUTURA HOSPITALAR

Monografia de Especialização, apresentada ao Curso de Especialização em Internet das Coisas, do Departamento Acadêmico de Eletrônica – DAELN, da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Especialista.

Orientador: Prof. Dr. Guilherme Luiz Moritz

CURITIBA
2020

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Curitiba

Diretoria de Pesquisa e Pós-Graduação
Departamento Acadêmico de Eletrônica
Curso de Especialização em Internet das Coisas

TERMO DE APROVAÇÃO

SISTEMA SUPERVISÓRIO DE INFRAESTRUTURA HOSPITALAR

por

GUILHERME THIEMANN

Esta monografia foi apresentada em 20 de Fevereiro de 2020 como requisito parcial para a obtenção do título de Especialista em Internet das Coisas. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. Dr. Guilherme Luiz Moritz
Orientador

_____
Prof. M. Sc. Danillo Leal Belmonte
Membro titular

_____
Prof. M. Sc. Omero Francisco Bertol
Membro titular

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso -

AGRADECIMENTOS

Agradeço a todos que, de alguma forma, contribuíram para realização desse trabalho pelo apoio e insistência para terminá-lo, em especial a Frida.

RESUMO

THIEMANN, Guilherme. **Sistema supervisório de infraestrutura hospitalar**. 2020. 56 p. Monografia de Especialização em Internet das Coisas, Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná. Curitiba, 2020.

Este trabalho apresenta uma implementação de monitoramento da infraestrutura e utilidades hospitalares, baseado em internet das coisas e na real necessidade encontrada no dia-a-dia pela manutenção e operação da infraestrutura de um hospital de grande porte. O monitoramento das utilidades em tempo real através de sistema supervisório.

Palavras-chave: IOT.Infraestrutura. Utilidades. Monitoramento.

ABSTRACT

THIEMANN, Guilherme. **Hospital infrastructure supervisory system**. 2020. 56 p. Monografia de Especialização em Internet das Coisas, Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná. Curitiba, 2020.

This work presents an implementation of monitoring of hospital infrastructure and utilities, based on internet of things and the real need found in the day-to-day for the maintenance and operation of the infrastructure of a large hospital. Monitoring of utilities in real time through a supervisory system.

Keywords: IOT. Infrastructure. Utilities. Monitoring.

LISTA DE ILUSTRAÇÕES

Figura 1 – Sensor de nível Eicos LA26M-40 11
Figura 2 – Sensor ultrassônico JSN-SR04T 12
Figura 3 – Sensor de temperatura full gauge MT-511Ri 13
Figura 4 – Placa de desenvolvimento ESP32 14
Figura 5 – Diagrama das conexões 17
Figura 6 – Imagem do aplicativo MQTT Dash parametrizado para monitorar a infraestrutura 19
Figura 7 – Imagem do grupo do Telegram 20
Figura 8 – Imagem do alarme de água da hemodiálise em nível baixo 22

Fotografia 1 – Televisão instalada na manutenção projetando o dashboard 18

Gráfico 1 – Comparativo de nível de caixa de água 23

LISTA DE SIGLAS

ANVISA	Agência Nacional de Vigilância Sanitária
BOT	Bot, diminutivo de *robot*, também conhecido como *Internet bot*
IOT	*Internet of Things*
MQTT	*Message Queue Telemetry Transport*

SUMÁRIO

1 INTRODUÇÃO 8
1.1 CONTEXTUALIZAÇÃO 8
1.2 PROBLEMA 9
1.3 OBJETIVOS 9
1.3.1 Objetivo Geral 9
1.3.2 Objetivos Específicos 9
1.4 JUSTIFICATIVA 10
1.5 ESTRUTURA DO TRABALHO 10
2 HARDWARE 11
2.1 SENSORES PARA ÁGUA E ESGOTO 11
2.2 SENSORE DE TEMPERATURA 12
2.3 SENSORES DIVERSOS 13
2.4 MICRO CONTROLADOR ESP32 14
2.5 RASPBERRY PI3 MODEL B+ 15
2.6 MODEM 4G COM ROTEADOR WI-FI 15
3 DESENVOLVIMENTO 16
3.1 SENSORES APLICADOS 16
3.2 MONTAGEM DA ESTRUTURA DE CONEXÃO E PROCESSAMENTO 17
4 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS 21
4.1 BENEFÍCIOS OBSERVADOS 21
4.1.1 Tratamento de Água da Hemodiálise 21
4.1.2 Monitoramento do Sistema de Aquecimento de Água 22
4.1.3 Estado dos Geradores 23
4.2 CONHECIMENTO DO PERFIL DA INFRAENTRUTURA 23
5 CONCLUSÃO 25
REFERÊNCIAS 26
APÊNDICE A - Programa do ESP32: Caixa da Água 27
APÊNDICE B - Programa do ESP32: Central com Telegram 36

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

O processo de monitoramento de utilidades e da infraestrutura é de suma importância para operação e manutenção de qualquer edificação. O trabalho desenvolvido aborda essa temática de monitoramento de uma edificação hospitalar.

A importância desse monitoramento refletirá na rápida ação da manutenção dando início, caso necessário, as ações de contingência, de acordo com a situação encontrada. Serão monitoras as seguintes utilidades: águas, energia elétricas, ar medicinal e equipamentos.

As utilidades tem um papel fundamental no funcionamento de um hospital, a água tem um papel muito importante em vários processos: limpeza geral, lavagem de roupa de cama, roupa e campo cirúrgico, limpeza de material cirúrgico, produção de comida, esterilização de material, água para hemodiálise e higiene dos pacientes (ANVISA, 2012). Há outras aplicações secundarias para a água que não foram citadas mas já é possível entender a criticidade de uma falta total ou parcial dessa utilidade e o quanto implicará no funcionamento de um hospital.

O ar medicinal é gerado através de compressores de ar comprimido e posteriormente filtrado atendendo as normas sanitárias e armazenado em pulmões para equalização da pressão e interligada diretamente a rede de consumo. O ar medicinal tem uma aplicação geral, sendo utilizado desde as emergências com a nebulização até a uti no auxilio respiração (ANVISA, 2016).

A energia elétrica é uma das mais críticas e utilizada em todo os processos do hospital. Há moto geradores para suprir a falta de energia elétrica por parte da concessionária. Os geradores são equipamentos que necessitam de manutenção rigorosa pois a falha desses equipamentos poderá acarretar sérios prejuízos aos pacientes (ANVISA, 2016).

O trabalho foca no monitoramento dessas utilidades citadas, de forma simples e de baixo custo, monitorando níveis de reservatórios de caixas d’água, temperatura de água, funcionalidade de equipamentos e sistemas com o envio de mensagem de caso algum sistema esteja fora dos padrões determinados.

1.2 PROBLEMA

O trabalho realizado foi em cima de um hospital infantil de grande porte em Curitiba, monitorando as utilidades por ele utilizado.

O sistema de abastecimento de água é composto por sistemas distintos: água fornecida pela concessionária e outra retirada de um poço artesiano. Esse sistema composto fornece ao hospital 25.000m³/mês de água potável.

A falta total ou parcial de água gera enormes transtornos, parando de imediato a central de esterilização de matérias, hemodiálise, lavanderia, cozinha e limpeza geral serão afetadas.

O sistema energético e de ar medicinal fornecem os suprimentos aos equipamentos de suporte a vida, e o monitoramento é essencial para uma rápida ação caso apresente falha.

Com a criticidade apresentada a proposta de um sistema de monitoramento de falhas desses sistemas foi desenvolvido, esperando observar os desvios de comportamento e sinalizar quando algo está fora da normalidade.

1.3 OBJETIVOS

Nesta seção são apresentados os objetivos geral e específicos do trabalho, relativos ao problema anteriormente apresentado.

1.3.1 Objetivo Geral

Aplicar os conhecimentos adquiridos em IOT como o sensoriamento sem fio das utilidades e infraestrutura hospitalar e reportar em menor tempo possível a manutenção dos problemas detectados.

1.3.2 Objetivos Específicos

Para atender ao objetivo geral, neste trabalho de conclusão de curso, os seguintes objetivos específicos serão abordados:

- Monitorar o nível de cisternas e caixas da água (Apêndice A);
- Temperatura de água de banho do paciente;
- Estado dos geradores;

- Nível de água da hemodiálise;
- Estado do sistema de ar medicinal;
- Interface com a manutenção via mensagem de Telegram (Apêndice B);
- Painel de monitoramento na manutenção desses sistemas.

1.4 JUSTIFICATIVA

A atividade hospitalar é muito complexa tendo em vista todas as atividades desenvolvidas desde a lavanderia ao centro cirúrgico. A parada por problema técnicos, desabastecimento ou falta de alguma utilidade impacta diretamente em toda a cadeia dos serviços oferecido no hospital. Minimizar essas paradas é muito importante para garantir um atendimento adequado e constante, sem intermitência.

É essencial para o hospital em questão o sistema desenvolvido ser baixo custo e pouca intervenção na estrutura existente, fácil implementação.

1.5 ESTRUTURA DO TRABALHO

Esta monografia de especialização está dividida em 5 (cinco) seções. Nesta primeira seção foi introduzido o assunto tema do trabalho e também foram abordados a motivação e os objetivos geral e específicos da pesquisa, a justificativa e a estrutura geral do trabalho.

Já na segunda seção: “Hardware”, serão abordados os hardwares aplicados no desenvolvimento do trabalho.

A seguir na terceira seção: “Desenvolvimento”, serão abordados o desenvolvimento e os softwares aplicados ao projeto.

Na quarta seção: “Apresentação e Análise dos Resultados”, tendo como base os resultados obtidos com a implantação do projeto.

Por último na quinta seção: “Conclusão”, serão retomados a pergunta de pesquisa e os seus objetivos e apontado como foram solucionados, respondidos, atingidos, por meio do trabalho realizado. Além disto, serão sugeridos trabalhos futuros que poderiam ser realizados a partir do estudo realizado.

2 HARDWARE

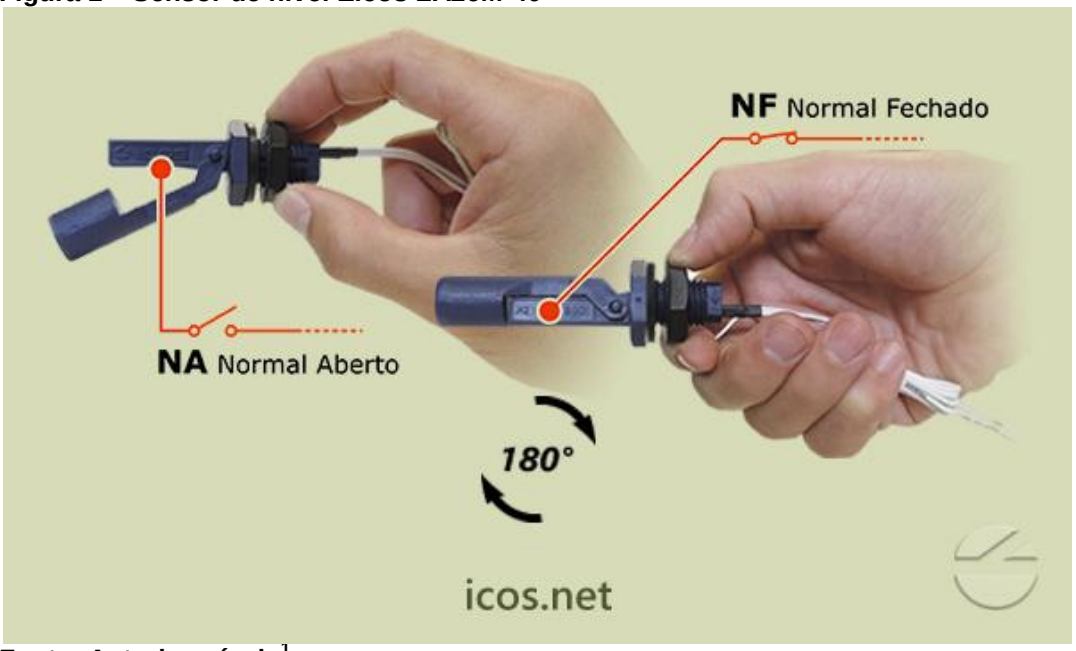
A seguir será descrito os hardware e sensores aplicados ao projeto descriminando a sua aplicação e funcionalidade no sistema. Começando pelos sensores até hardware de processamento.

2.1 SENSORES PARA ÁGUA E ESGOTO

Os sensores para medir nível de água são de dois tipos e tem a finalidade de saber o nível de água nos reservatórios que estão inseridos.

O sensor de nível da Figura 1, indica apenas se há ou não nível de água no local instalado, permanecendo fixo em uma posição. Este sensor tem a mesma funcionalidade de um interruptor, com apenas dois estados, fechado ou aberto.

Figura 1 – Sensor de nível Eicos LA26M-40



Fonte: Autoria própria[1].

O sensor da Figura 2 trata-se de um sensor ultrassônico e foi utilizado para monitorar o nível das caixas d'água, por meio de ultrassom.

[1] Imagem realizada no site do fabricante. Disponível em: <http://www.eicos.com.br/sensor-de-nivel/montagem-lateral/LA26M-40/>. Acesso em: 17 dez. 2019.

Figura 2 – Sensor ultrassônico JSN-SR04T

Fonte: Autoria própria[2].

O sensor JSN-SR04T 2.0, apresentado novamente na Figura 2, têm as seguintes características técnicas:

- Tensão de trabalho 5V;
- Consumo de 5 a 30 mA;
- Frequência do ultrassom de 40kHz
- Resistente a água;
- Detecção de 25cm à 2 metros.

O processo deve realizar a instalação fixa, acima do nível máximo da lâmina d'água.

2.2 SENSORE DE TEMPERATURA

O sensor de temperatura utilizado, apresentado na Figura 3, foi o MT-511Ri do Fabricante Full Gauge Controls.

[2] Imagem realizada no site da loja de informática. Disponível em: <dhttps://www.usinainfo.com.br/1011193-thickbox_default/sensor-ultrassonico-jsn-sr04t-a-prova-d-agua-modulo-para-arduino.jpg>. Acesso em: 18 dez. 2019.

Figura 3 – Sensor de temperatura full gauge MT-511Ri

Fonte: Autoria própria[3].

A funcionalidade deste sensor é medir a temperatura, programando o sensor para atuar conforme a temperatura desejada, a utilização foi no reservatório de água quente que atende a água de banho dos pacientes e cozinha, a temperatura parametrizada foi de 40ºC, ou seja, quando a temperatura ficar abaixo de 40ºC a saída do sensor ficará fechada.

2.3 SENSORES DIVERSOS

Os sensores dos geradores, ar comprimido e de energia da concessionária são sinais de seus controladores específicos inerentes dos equipamentos de controle, a captação desses sinais foram através de contatos secos, On/OFF.

Para os geradores foram pegos 3 sinais de cada gerador há 2 geradores: falha crítica, não critica e gerador em operação.

O ar medicinal temos dois sinais, um de falha no compressor e outro de baixa pressão na rede

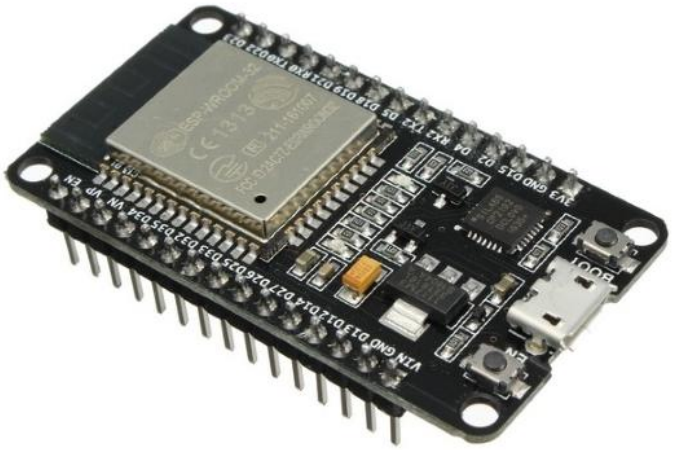
O sensor que monitora a rede de energia elétrica é um contato seco que está ligado ao transformador de energia do hospital, que quando a falta de energia o contato abre.

[3] Imagem realizada no site do fabricante. Disponível em: <https://www.fullgauge.com.br/public/uploads/downloads/mt511ri.jpg>. Acesso em: 10 dez. 2019.

2.4 MICRO CONTROLADOR ESP32

O microcontrolador ESP32, apresentado na Figura 4, foi utilizado para realizar as leituras dos contatos seco de todos os sensores aplicados e fazer a correlação da altura medida pelo sensor de ultrassom em nível dos reservatórios. Este microcontrolador foi utilizado para fazer a comunicação dos eventos para a manutenção.

Figura 4 – Placa de desenvolvimento ESP32



Fonte: Autoria própria[4].

O microcontrolador ESP32 foi utilizado para realizar as leituras dos contatos seco de todos os sensores aplicados e fazer a correlação da altura medida pelo sensor de ultrassom.

As principais características técnicas do microcontrolador ESP32 são:

- CPU: Xtensa® Dual-Core 32-bit LX6;
- ROM: 448 KBytes;
- RAM: 520 Kbytes;
- Portas GPIO: 11;
- Wireless padrão 802.11 b/g/n;
- Tensão de operação: 4,5 ~ 9V.

[4] Imagem realizada no site da loja de informática. Disponível em: <https://www.filipeflop.com/produto/modulo-wifi-esp32-bluetooth/>. Acesso em: 10 dez. 2019.

2.5 RASPBERRY PI3 MODEL B+

O Raspberry Pi3 é um hardware que é um mini computador que foi instalado o sistema operacional Raspbian, será utilizado para realizar o processamento do das trocas de informações entre as aplicações, nele está instalado o sistema operacional Rapbian, a escolha desse hardware foi devido ao baixo custo, possibilidade de instalação dos softwares necessários para a aplicação e WI-FI integrado.

2.6 MODEM 4G COM ROTEADOR WI-FI

Foi utilizado um modem 4g com roteador WI-FI embutido para conexão com a internet. Através desse acesso à internet que serão enviadas as mensagens do Telegram e acesso do aplicativo MQTT instalados nos smartfone da manutenção. Através desse roteador serão conectados os microcontroladores ESP32 e os Raspberry.

3 DESENVOLVIMENTO

3.1 SENSORES APLICADOS

O projeto de monitoramento das utilidades e infraestrutura do hospital iniciou-se com a implantação dos sensores nos seguintes locais:

- 2 cisternas de 25.000L cada;
- 4 caixas de água com 25.000 l cada;
- 1 Reservatório de 300 l água de osmose para uso na hemodiálise;
- 1 reservatório de água pluvial e esgoto;
- 1 reservatório de 300 l água pura para esterilização de material;
- 1 Boiler de 3000l de água quente para banho;
- 2 geradores de 500KVA;
- 1 sistema de geração de ar medicinal;
- Subestação de energia 2700kVA.

Para monitoramento dos níveis dos reservatórios de água utilizou-se o sensor de nível apresentado novamente na Figura 1, colocou-se sensores na metade do nível da caixa e denominou como nível baixo, e outro sensor próximo a caixa vazia, denominando caixa vazia. As caixas com esse sistema são 2 cisternas e nas 4 caixas de água. As demais apenas 1 sensor ajustado ao meio da caixa, alertando o nível anormal.

Os sensores dos geradores são contatos secos retirados do controlador dos geradores, extraído as seguintes informações:

- Alarme não critico;
- Alarme crítico;
- Gerador em Operação.

Os alarmes não críticos, tem por definição da manutenção do hospital, problemas no equipamento que devem ser verificados, mas não afetam o gerador de entrar em funcionamento caso necessário, como por exemplo resistência de aquecimento do bloco do motor. Já os alarmes críticos, são defeito que impedirá o funcionamento do gerador devendo ser tratado imediatamente. O alerta de gerador em operação é apenas uma sinalização.

O sensor de temperatura utilizado para monitoramento de água quente está parametrizado para atuar caso a temperatura fique abaixo de 40ºC.

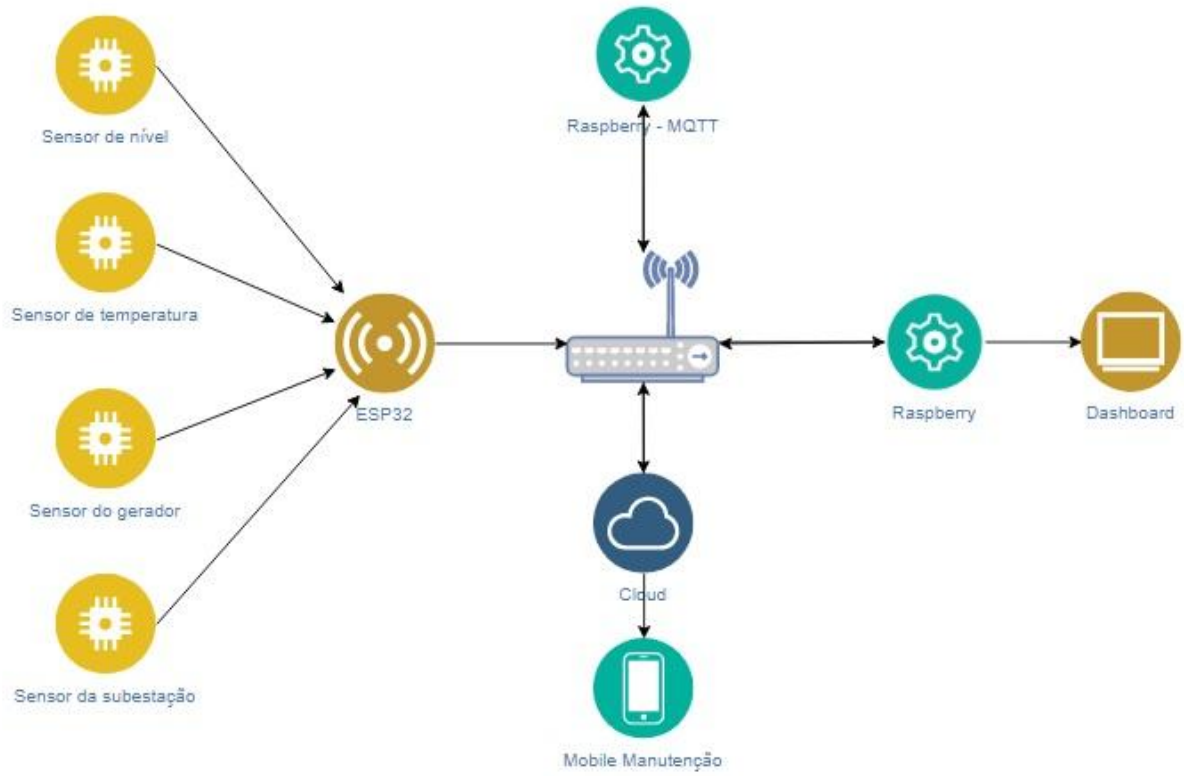
O sistema de ar medicinal é monitorado pelo controlador dos compressores utilizando-se apensa um contato seco, relé NA/NF, caso o compressor entre em falha ou a pressão da rede fique abaixo de 4bar.

Deve-se utilizar sempre a terceira pessoa do singular na elaboração do texto, mantendo-se a forma impessoal com o uso do pronome SE.

3.2 MONTAGEM DA ESTRUTURA DE CONEXÃO E PROCESSAMENTO

O sistema está conectado via protocolo 802.11n entre o ESP32, raspberry e o roteador, como pode ser observado na Figura 5. A conexão entre os sensores e o ESP32 é física, a conexão do Mobile é via rede de telefonia móvel.

Figura 5 – Diagrama das conexões



Fonte Autoria própria.

No raspberry utilizou-se sistema operacional Raspbian, com a utilização do software MQTT mosquitto. O software MQTT tem a função de comunicação entre o

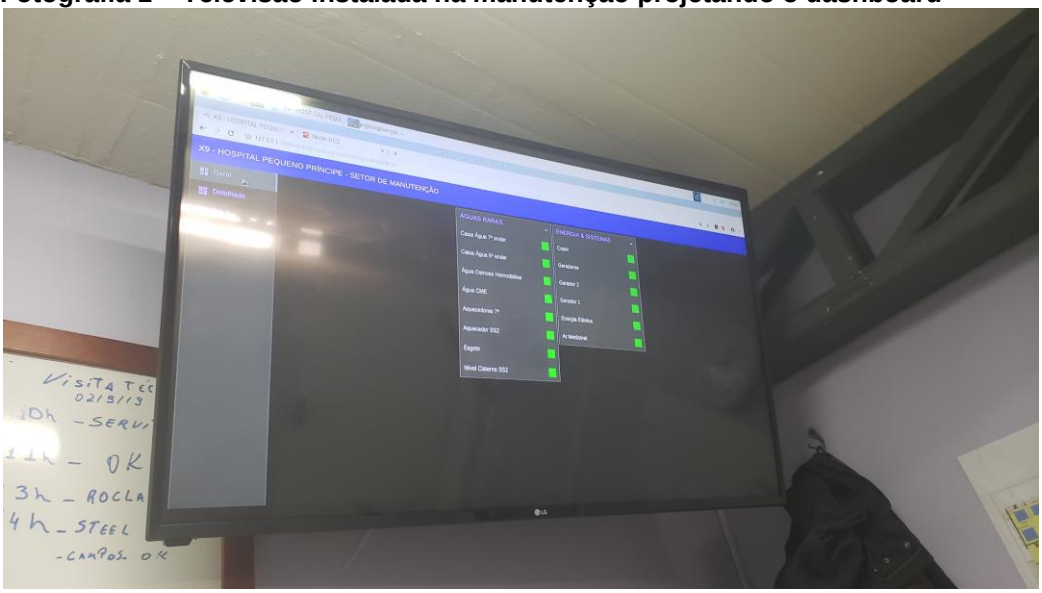
ESP32 com o Dashboard e com o mobile através do protocolo da IBM MQTT esse protocolo opera pelo sistema de *publish/subscribe*.

O protocolo MQTT fornece um método leve de executar mensagens usando um modelo de publicação / assinatura. Isso o torna adequado para mensagens da Internet das Coisas, como sensores de baixa potência ou dispositivos móveis, como telefones, computadores incorporados ou micro controladores (FELIPE, 2020).

Os sensores publicam no servidor, já o dashboard e o aplicativo MQTT Dash estão inscritos nos tópicos de interesse.

Já o outro raspberry tem papel de prover um dashboard para visualização local e remota. Este conectado via WI-FI ao roteador e conectado via HDMI (*High-Definition Multimedia Interface)* com um televisor de 43” instalados na sala de coordenação da manutenção do hospital, como pode ser observado na Fotografia 1. O dashboard foi criado utilizando o programa Node-RED, ferramenta de programação em bloco e de fácil utilização (NODE-RED, 2019). Foi utilizado o serviço de domínio gratuito para gerar um domínio publico para acessar remotamente esse dashboard, a configuração foi realizada no raspberry.

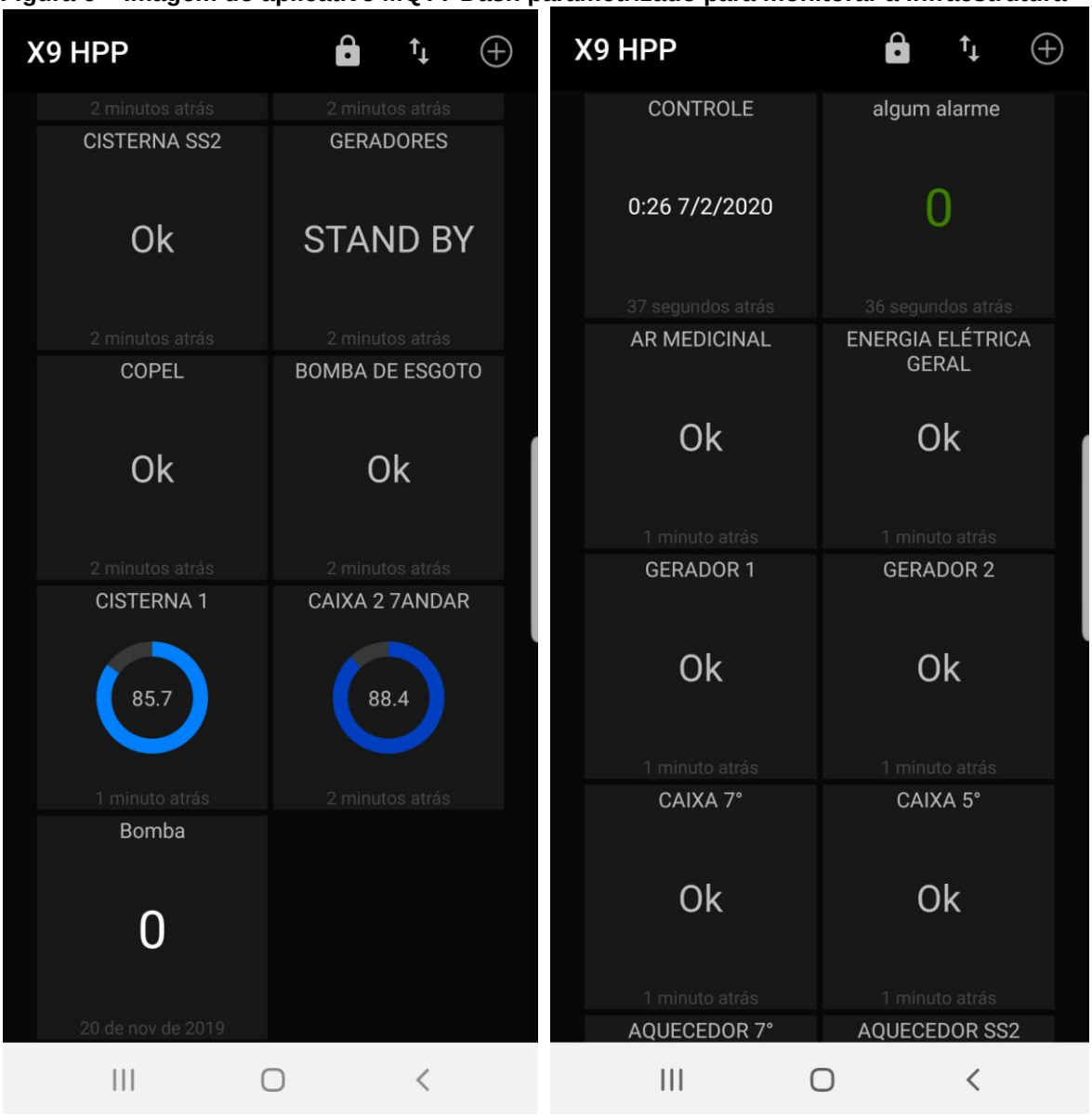
Fotografia 1 – Televisão instalada na manutenção projetando o dashboard



Fonte: Autoria própria.

Nos smartfones, da equipe de manutenção, instalou-se o aplicativo MQTT dash, no qual foi parametrizado o endereço do servidor do MQTT, de forma que toda a fez que ocorrer uma publicação esta informação estará disponível para conferencia nos celulares parametrizados.

Figura 6 – Imagem do aplicativo MQTT Dash parametrizado para monitorar a infraestrutura



Fonte Autoria própria.

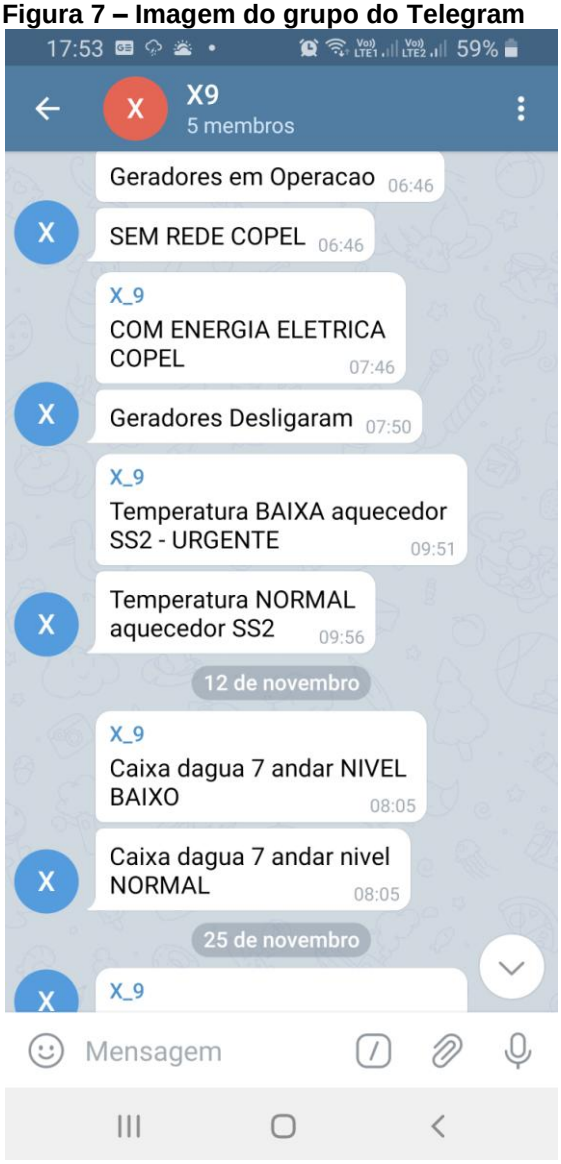
Uma das funcionalidades implementadas é a mensagem via Telegram no ESP32 (Apêndice B). Foi utilizada biblioteca especificas para essa função. Criou-se um BOT, após a criação do BOT foi criado um grupo no qual foram incluídos coordenadores, assim que ocorrer alguma alteração importante na infraestrutura o ESP32 envia uma mensagem para este grupo criado informando o problema. Caso o problema persista a mensagem é reenviada a cada 20min até que o problema seja resolvido. Quando o estado volta a normalidade é enviada outra mensagem ao grupo informando que está normalizado o problema.

O Telegram é um aplicativo de mensagens móvel e desktop baseado em nuvem com foco em segurança e velocidade (TELEGRAM, 2019a) .

> Bots são aplicativos de terceiros que são executados dentro do Telegram. Os usuários podem interagir com bots enviando-lhes mensagens, comandos e solicitações inline. Você controla seus bots usando solicitações HTTPS para nossa API bot (TELEGRAM, 2019b).

As mensagens apresentadas na Figura 7 foram de eventos que ocorreram no dia 09 de Novmento de 2019, sábado, com início 06 horas 46 minutos, no qual ocorreu a falta de energia da concessionária, os geradores entraram em operação comunicação imediata à manutenção sobre a falha.

Figura 7 – Imagem do grupo do Telegram



Fonte Autoria própria.

4 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

4.1 BENEFÍCIOS OBSERVADOS

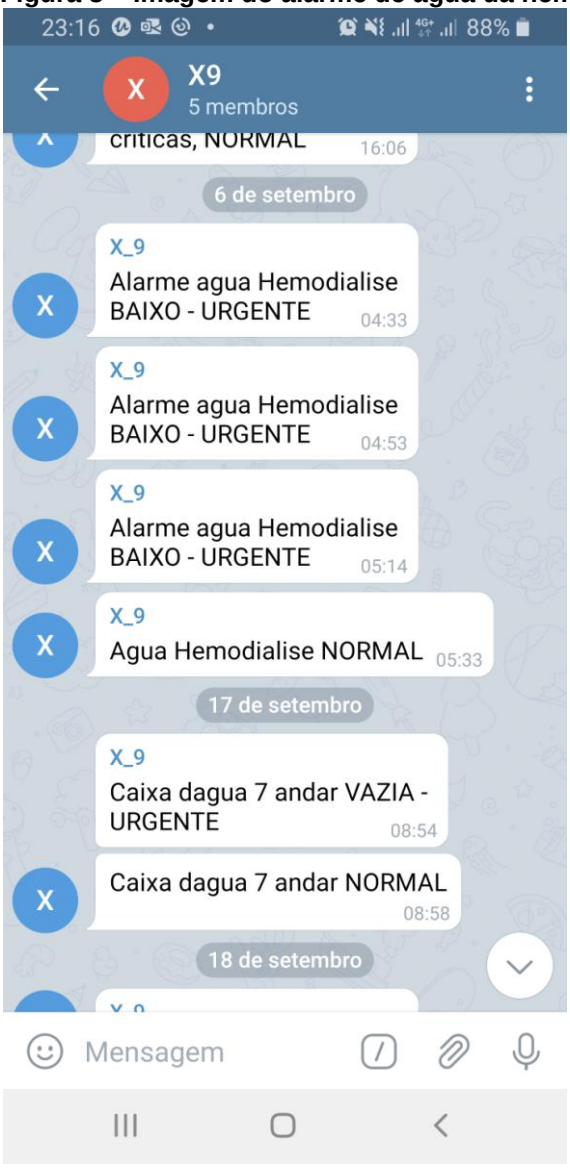
Com a implementação do monitoramento da infraestrutura e utilidades observou uma mudança na cultura e nas ações da manutenção com relação ao tratamento dos problemas e alarmes apontados pelo projeto.

4.1.1 Tratamento de Água da Hemodiálise

Antes da implantação do projeto ocorria com frequência o auto desligamento da osmose reversa por motivos desconhecidos ocasionando a falta de agua para a realização de hemodiálise nos pacientes. Esse problema geralmente ocorria após um final de semana e observado somente na segunda-feira, gerando um transtorno para a equipe de nefrologia e pacientes.

Após a implantação do monitoramento quando o nível ultrapassava o mínimo estabelecido aciona o plantão da manutenção. Na Figura 8 é possível observar um evento que ocorreu durante a madrugada do dia 6 de Setembro de 2019 às 04 horas e 33 minutos. O plantonista recebeu a mensagem via Telegram do evento e iniciou o deslocamento até o hospital, após exatamente 1 hora o problema foi resolvido, sem gerar nenhum impacto a pacientes. Antes do monitoramento esse problema seria observado tardiamente ocasionando imensos transtornos.

Figura 8 – Imagem do alarme de água da hemodiálise em nível baixo



Fonte Autoria própria.

4.1.2 Monitoramento do Sistema de Aquecimento de Água

Com o monitoramento do sistema de aquecimento de água foi possível eliminar as faltas de água quente para banho de pacientes. O sistema de monitoramento evitou desde a implantação mais de 30 situações de temperatura baixa para banho.

Os principais problemas que geraram essas possíveis baixas temperaturas foram: válvula reguladora de pressão de gás natural obstruída, quebra de aquecedor de passagem de água, curto circuito em disjuntor de alimentação dos aquecedores e bombas de circulação de água, queima de bomba d’água, filtro obstruído da entrada de água do sistema.

4.1.3 Estado dos Geradores

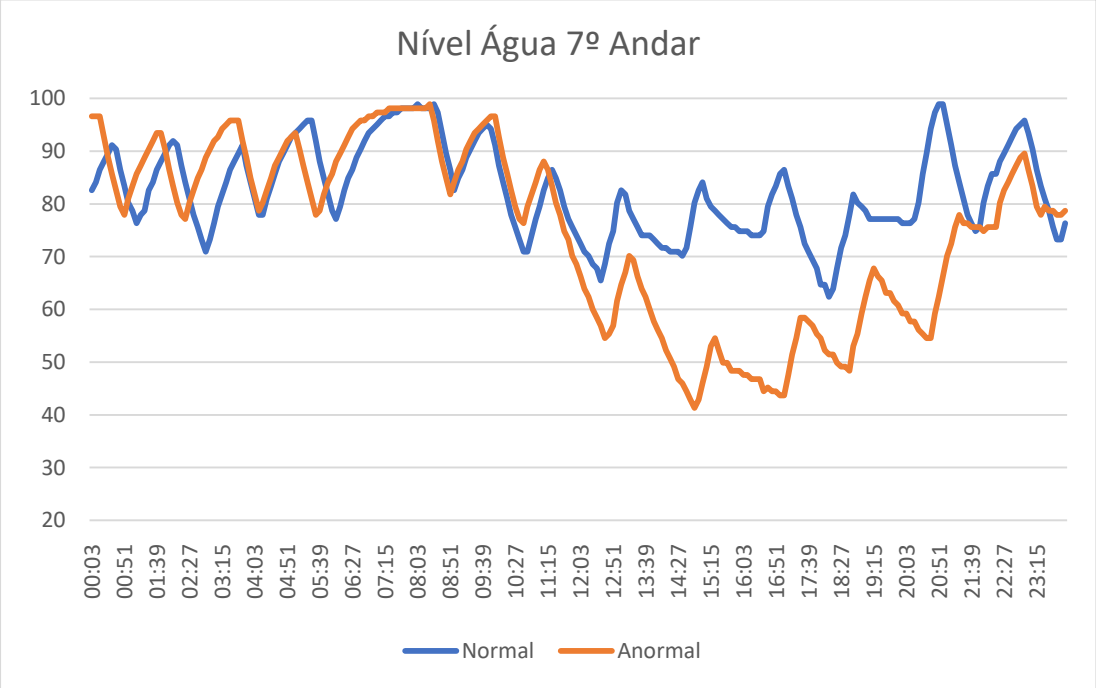
O principal benefício de monitorar o estado dos geradores é a informação no exato momento que ocorre o evento deste a implantação foram 4 eventos de oscilação de abastecimento de energia da concessionária de energia, no qual os geradores entram e abastecem o hospital. Foram 2 eventos não críticos por queima da resistência de aquecimento do bloco do motor do gerador, e 1 evento critico ocasionado por má conexão nos bornes do controlador.

4.2 CONHECIMENTO DO PERFIL DA INFRAENTRUTURA

O principal conhecimento adquirido com a analise de dados foi o perfil de consumo de água e o comportamento do nível de água das caixas elevadas. O monitoramento envia o estado do nível da caixa a cada 6 minutos, ou seja 10 por horas, 240 por dia.

No Gráfico 1 há duas curvas do nível da mesma caixa d'água em dias diferentes, uma série denominada normal e outra anormal. Observa-se o afundamento maior no nível anormal em relação a outra série.

Gráfico 1 – Comparativo de nível de caixa de água



Fonte: Autoria própria.

A curva anormal foi devida ao baixo nível de água na cisterna ocasionado pela menor vazão de entrada de água, devido ao baixo nível, as bombas d’água de recalque desligaram para não trabalharem a vazio, assim que recuperaram o nível voltavam a funcionar. Com esses dados, levantados pelo programa apresentado no Apêndice A, inicia uma nova visão da infraestrutura do hospital, avaliar possíveis problemas apenas analisando a característica da curva de nível da caixa d’água.

5 CONCLUSÃO

O trabalho desenvolvido gerou muitos pontos positivos de segurança na operação da infraestrutura e utilidades hospitalar. Os resultados conquistados implementando os conhecimentos em IOT com monitoramento através de sensores via rede sem fio foi de grande importância, foram prevenidas varias faltas de água, tanto potável, aquecida e do tratamento para hemodiálise, evitando transtornos imensos.

A maior dificuldade de utilizar o ultrassom para monitoramento do nivel foi a escolha correta do sensor, que no início foi instalado um sensor sem proteção adequado que ocasionou a deterioração por corrosão causada pelo cloro presente dentro dos reservatórios de água, o sensor durava apenas 7 dias, com a aquisição do modelo já descrito já ultrapassou 4 meses sem nenhum problema.

O monitoramento dos geradores e do sistema de ar medicinal informando o estado que estão via Telegram gerou uma tranquilidade para a equipe de manutenção, pois a qualquer mudança de estado há o envio dessa informação para todos os responsáveis pelo equipamento.

O envio de mensagem ao grupo da manutenção diminuiu o tempo de atendimento principalmente das ocorrências emergenciais fora do horário comercial. Como o sistema monitora a infraestrutura o chamado não depende mais de uma sinalização humana. Essa sinalização humana nem sempre é imediata, quando observada um tempo já passou, aumentando em muito o tempo morto para tratamento da ocorrência.

O tema que foi proposto implementar foi alcançado plenamente, e abriu a possibilidade de expandir essa ideia de monitoramento para demais setores e serviços dentro do hospital, tais como: monitoramento das condições ambientais com temperatura do ar, umidade; temperatura de condicionamento de medicamentos dentro das faixas de controle.

REFERÊNCIAS

ANVISA. **Segurança do paciente em serviços de saúde**: limpeza e desinfecção de superfícies. Agência Nacional de Vigilância Sanitária (ANVISA). Copyright© 2012 Agência Nacional de Vigilância Sanitária. Brasília: 2012. Disponível em: <https://www.riscobiologico.org/lista/20140128_01.pdf>. Acesso em: 20 dez. 2019.

ANVISA. **Segurança no ambiente hospitalar**. Agência Nacional de Vigilância Sanitária (ANVISA). Copyright© 2016 Agência Nacional de Vigilância Sanitária. Brasília: 2016. Disponível em: <http://portal.anvisa.gov.br/documents/33852/271855/Seguran%C3%A7a+no+ambiente+hospitalar/473c5e32-025a-4dc2-ab2e-fb5905d7233a>. Acesso em: 20 dez. 2019.

FELIPE, Nilton. **Mosquitto Eclipse – Um broker MQTT de software livre**. Diário de Nilton Felipe, publicado em: 21 abr. 2018. Disponível em: <https://niltonfelipe.wordpress.com/2018/04/21/mosquitto-eclipse-um-broker-mqtt-de-software-livre/>. Acesso em: 21 dez. 2019.

NODE-RED. Node-RED**: Low-code programming for event-driven applications** Node-RED é um projeto da OpenJS Foundation. Copyright© 2019 OpenJS Foundation. Disponível em: <https://nodered.org/>. Acesso em: 15 dez. 2019.

TELEGRAM. **Telegram**: A new era of messaging. 2019a. Disponível em: <https://telegram.org/> Acesso em: 21 dez. 2019.

TELEGRAM. **Bots: An introduction for developers**. 2019b. Disponível em: <https://core.telegram.org/bots> Acesso em: 20 dez. 2019.

APÊNDICE A - Programa do ESP32: Caixa da Água

```
#include "esp_system.h"
#include <time.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>

#define BOTtoken "***************"  //

#define ID_MQTT  "X9_2"
#define Username "x9"
#define Password "****"
#define WillTopic "HPP/LAST_WILL/X9"
#define WillQoS 1
#define WillRetain 1
#define WillMessage "X9_teste OFF"

bool bott = 0 ;

const unsigned int TRIG_PIN=27;
const unsigned int ECHO_PIN=14;

const unsigned int TRIG1_PIN=13;
const unsigned int ECHO1_PIN=12;

WiFiClientSecure client;
UniversalTelegramBot bot(BOTtoken, client);

const char* BROKER_MQTT = "192.168.10.144";
//const char* BROKER_MQTT = "*****.mooo.com"; //URL do broker MQTT que se deseja utilizar
int BROKER_PORT = 1883; // Porta do Broker MQTT

WiFiClient espClient;
PubSubClient MQTT(espClient);

const char* ssid     = "X_9";const char* password = "******";

long timezone = 1;
byte daysavetime = 1;
struct tm tmstruct ;

int short minuto[2];
int short hora[2];
char* txtsms="";
char* TOPICO_PUBLISH="";
#define TOPICO_SUBSCRIBE "#"
```

```
int k=0;
int  i = 0;
char controle[30]="";
String Scontrole="";

//gpio to use to trigger delay
const int wdtTimeout = 30000;  //time in ms to trigger the watchdog
hw_timer_t *timer = NULL;

void IRAM_ATTR resetModule() {
  ets_printf("reboot\n");
  esp_restart();
}

void initWiFi();
void initMQTT();
void reconectWiFi();
void mqtt_callback(char* topic, byte* payload, unsigned int length);
void VerificaConexoesWiFIEMQTT(void);
void InitOutput(void);
void leitura();
void Controle();
void Enviarsms(char* txt);


void setup() {
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  pinMode(TRIG1_PIN, OUTPUT);
  pinMode(ECHO1_PIN, INPUT);

  Serial.begin(115200);
  initWiFi();
  Serial.println("Contacting Time Server");

  configTime(3600*timezone,  daysavetime*3600,  "time.nist.gov",  "0.pool.ntp.org",
"1.pool.ntp.org");
  struct tm tmstruct ;
  int j=0;
  while(i==0){
    delay(1000);
    tmstruct.tm_year = 0;
    getLocalTime(&tmstruct, 5000);
    Serial.print(".");
    if((tmstruct.tm_year+1900)>2018)
      i=1;
    if(j==5){
       Serial.println("Restart");
       delay(2000);
```

```
        ESP.restart();
       }
       j++;
  }
  i=0;
  j=0;
  Serial.println("");
  Serial.printf("\nNow is : %d-%02d-%02d
%02d:%02d:%02d\n",(tmstruct.tm_year)+1900,( tmstruct.tm_mon)+1,
tmstruct.tm_mday,tmstruct.tm_hour -5 , tmstruct.tm_min, tmstruct.tm_sec);
  Serial.println("");

  Serial.println("Feito");

  i=0;
   while (i < 2) {
    minuto[i] = 0;
    hora[i] = 0;
    i = i + 1;
  }

  InitOutput();
  initMQTT();

  timer = timerBegin(0, 80, true);                  //timer 0, div 80
  timerAttachInterrupt(timer, &resetModule, true);  //attach callback
  timerAlarmWrite(timer, wdtTimeout * 1000, false); //set time in us
  timerAlarmEnable(timer);
  i=0;
  hora[0]=-1;
  minuto[0]=-1;

 }

void loop() {
  timerWrite(timer, 0); //reset timer (feed watchdog)
  //garante funcionamento das conexões WiFi e ao broker MQTT
  VerificaConexoesWiFIEMQTT();
  //envia o status de todos os outputs para o Broker no protocolo esperado
  //keep-alive da comunicação com broker MQTT
  MQTT.loop();
  Controle();

  if(minuto[0] != minuto[1]){
    minuto[0] = minuto[1];
    //if(k>=5){
    leitura();
   // k=0;
```

```
   Serial.printf("\nNow is : %d-%02d-%02d
%02d:%02d:%02d\n",(tmstruct.tm_year)+1900,( tmstruct.tm_mon)+1,
tmstruct.tm_mday,tmstruct.tm_hour -5 , tmstruct.tm_min, tmstruct.tm_sec);

  }
}

void Enviarsms(char* txt){
  bott=0;
  Serial.println("Enviando SMS, um momento...");//354323177 guilherme
  Serial.println(txt);
  delay(200);
  bott =  bot.sendMessage("354323177", txt); // Grupo x9 bot.sendMessage
  Serial.print("Valor de Bott:");
  Serial.println(bott);
  delay(200);
    if(bott==0){
       Serial.println("Restart");
       delay(2000);
       ESP.restart();
    }
}

void VerificaConexoesWiFIEMQTT(void)
{
    if (!MQTT.connected())
        reconnectMQTT(); //se não há conexão com o Broker, a conexão é refeita

     reconectWiFi(); //se não há conexão com o WiFI, a conexão é refeita
}

void EnviaEstadoOutputMQTT(char* TOPICO_PUBLISH,char* valor)
{

    MQTT.publish(TOPICO_PUBLISH, valor,true);

    Serial.println(" enviado ao broker!");
    Serial.print(TOPICO_PUBLISH);
    Serial.print(" : ");
    Serial.println(valor);
    delay(100);
    return ;
}
```

```
void initWiFi()
{
    delay(10);
    Serial.println("------Conexao WI-FI------");
    Serial.print("Conectando-se na rede: ");
    Serial.println(ssid);
    Serial.println("Aguarde");

    reconectWiFi();
}

void reconectWiFi()
{
    //se já está conectado a rede WI-FI, nada é feito.
    //Caso contrário, são efetuadas tentativas de conexão
    if (WiFi.status() == WL_CONNECTED)
        return;

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED)
    {
        delay(100);
        Serial.print(".");
    }

    Serial.println();
    Serial.print("Conectado com sucesso na rede.... ");
    Serial.print(ssid);
    Serial.println("IP obtido: ");
    Serial.println(WiFi.localIP());
}

void initMQTT()
{
    MQTT.setServer(BROKER_MQTT, BROKER_PORT);     //informa qual broker e
porta deve ser conectado
    MQTT.setCallback(mqtt_callback);              //atribui função de callback (função
chamada quando qualquer informação de um dos tópicos subescritos chega)
}

//Função: função de callback
//        esta função é chamada toda vez que uma informação de
//        um dos tópicos subescritos chega)
//Parâmetros: nenhum
//Retorno: nenhum
void mqtt_callback(char* topic, byte* payload, unsigned int length)
{

}
```

```
void reconnectMQTT()
{
    while (!MQTT.connected())
    {
        Serial.print("* Tentando se conectar ao Broker MQTT: ");
        Serial.println(BROKER_MQTT);
        if (MQTT.connect(ID_MQTT, Username, Password, WillTopic, WillQoS, WillRetain, WillMessage)) //if (MQTT.connect(ID_MQTT, Username, Password, WillTopic, WillQoS, WillRetain, WillMessage))
        {
            Serial.println("Conectado com sucesso ao broker MQTT!");
            //MQTT.subscribe(TOPICO_SUBSCRIBE);
        }
        else
        {
            Serial.println("Falha ao reconectar no broker.");
            Serial.println("Havera nova tentatica de conexao em 2s");
            delay(2000);
        }
    }
}

void InitOutput(void)
{
    //IMPORTANTE: o Led já contido na placa é acionado com lógica invertida (ou seja,
    //enviar HIGH para o output faz o Led apagar / enviar LOW faz o Led acender)
  //  pinMode(D0, OUTPUT);
   // digitalWrite(D0, HIGH);
}


void leitura(void){
  int caixa1[3];
  int caixa2[3];
  float mcaixa1=0;
  float mcaixa2=0;
  int dist[12];
  int dist1[12];

  for(int i=0;i<3;i++){
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

 const unsigned long  duration= pulseIn(ECHO_PIN, HIGH);
 int distance= duration/29/2;
```

```
 if(duration==0){
   Serial.println("Warning: no pulse from sensor");
   }
  else{
      Serial.print("CAIXA 1 DISTANCIA:");
      Serial.print(distance);
      Serial.println(" cm");
  }
delay(200);

  digitalWrite(TRIG1_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG1_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG1_PIN, LOW);

 const unsigned long  duration1= pulseIn(ECHO1_PIN, HIGH);
 int  distance1= duration1/29/2;
 if(duration1==0){
   Serial.println("Warning: no pulse from sensor");
   }
  else{
      Serial.print("CAIXA 2 DISTANCIA:");
      Serial.print(distance1);
      Serial.println(" cm");
  }

      Serial.print(distance1);
      Serial.print("   ");
      Serial.println(distance);

      delay(200);
      caixa1[i]=distance;
      caixa2[i]=distance1;
  }

  if((caixa1[0]<=caixa1[1] && caixa1[1]<=caixa1[2]) || (caixa1[2]<=caixa1[1] &&
caixa1[1]<=caixa1[0] ))
  mcaixa1=caixa1[1];

  else if((caixa1[1]<=caixa1[0] && caixa1[0]<=caixa1[2]) || (caixa1[2]<=caixa1[0] &&
caixa1[0]<=caixa1[1] ))
  mcaixa1=caixa1[0];

  else if((caixa1[0]<=caixa1[2] && caixa1[2]<=caixa1[1]) || (caixa1[1]<=caixa1[2] &&
caixa1[2]<=caixa1[0] ))
  mcaixa1=caixa1[2];
```

```
 if((caixa2[0]<=caixa2[1] && caixa2[1]<=caixa2[2]) || (caixa2[2]<=caixa2[1] && caixa2[1]<=caixa2[0] ))
  mcaixa2=caixa2[1];

  else if((caixa2[1]<=caixa2[0] && caixa2[0]<=caixa2[2]) || (caixa2[2]<=caixa2[0] && caixa2[0]<=caixa2[1] ))
  mcaixa2=caixa2[0];

  else if((caixa2[0]<=caixa2[2] && caixa2[2]<=caixa2[1]) || (caixa2[1]<=caixa2[2] && caixa2[2]<=caixa2[0] ))
  mcaixa2=caixa2[2];

  mcaixa2=mcaixa2*(-0.778)+163.49;
  mcaixa1=mcaixa1*(-0.778)+163.49;

  Scontrole+=mcaixa1;
  Scontrole.toCharArray(controle, Scontrole.length() + 1);
  Scontrole="";
  EnviaEstadoOutputMQTT("HPP/SS2/CX_AGUA/3",controle);
  delay(200);

  Scontrole+=mcaixa2;
  Scontrole.toCharArray(controle, Scontrole.length() + 1);
  Scontrole="";
  EnviaEstadoOutputMQTT("HPP/SS2/CX_AGUA/4",controle);
  delay(200);
}

void Controle(void){
getLocalTime(&tmstruct, 5000);

if((tmstruct.tm_hour -5)==-5)
hora[1]=19;
else if((tmstruct.tm_hour -5)==-4)
hora[1]= 20;
else if((tmstruct.tm_hour -5)==-3)
hora[1]= 21;
else if((tmstruct.tm_hour -5)==-2)
hora[1]= 22;
else if((tmstruct.tm_hour -5)==-1)
hora[1]= 23;
else{
hora[1]= tmstruct.tm_hour -5;
}
minuto[1]=tmstruct.tm_min;

i=0;
```

```
if (minuto[i] > 59)
  {
    minuto[i]=minuto[i]-59;
    hora[i]=hora[i]+1;
  }
  if (hora[i] > 23)
  {
    hora[i]=0;
  }

  i=0;
 char alarme[3];
if(minuto[0]!= minuto[1]){

  if((tmstruct.tm_hour -5)==-5)
  Scontrole+= 19;
  else if((tmstruct.tm_hour -5)==-4)
  Scontrole+= 20;
  else if((tmstruct.tm_hour -5)==-3)
  Scontrole+= 21;
  else if((tmstruct.tm_hour -5)==-2)
  Scontrole+= 22;
  else if((tmstruct.tm_hour -5)==-1)
  Scontrole+= 23;
  else
  Scontrole+= (tmstruct.tm_hour -5);

  Scontrole+= ":";
  Scontrole+= tmstruct.tm_min;
  Scontrole+= " ";
  Scontrole+= tmstruct.tm_mday;
  Scontrole+= "/";
  Scontrole+= tmstruct.tm_mon+1;
  Scontrole+= "/";
  Scontrole+= tmstruct.tm_year+1900;
  Scontrole.toCharArray(controle, Scontrole.length() + 1);
  Scontrole="";
 ////////////////////// EnviaEstadoOutputMQTT("HPP/CONTROLE/teste",controle);
  delay(200);
  //minuto[0]=minuto[1];
}
 }
```

APÊNDICE B - Programa do ESP32: Central com Telegram

```
#include "esp_system.h"
#include <time.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>

#define BOTtoken "**************************"  // your Bot Token (Get from Botfather)
#define ID_MQTT  "X9_central"
#define Username "x9"
#define Password "robox9"
#define WillTopic "HPP/LAST_WILL/X9"
#define WillQoS 1
#define WillRetain 1
#define WillMessage "X9_CENTRAL OFF"

bool bott = 0 ;
int alarmes =0;

WiFiClientSecure client;
UniversalTelegramBot bot(BOTtoken, client);

const char* BROKER_MQTT = "******.mooo.com"; //URL do broker MQTT que se deseja utilizar
int BROKER_PORT = 1883; // Porta do Broker MQTT

WiFiClient espClient;
PubSubClient MQTT(espClient);

#define VERMELHO  "2"
#define AMARELO  "1"
#define VERDE  "0"

const char* ssid     = "X_9";const char* password = "******";

long timezone = 1;
byte daysavetime = 1;
struct tm tmstruct ;

int short minuto[21];
int short hora[21];
char* txtsms="";
char* TOPICO_PUBLISH="";
#define TOPICO_SUBSCRIBE "#"

const int Paquecedorsubsolo = 13;
const int Paquecedorsetimo =  12;
const int Pcaixaaguaquintoa = 14;
```

```
const int Pcaixaaguaquintov =  27;
const int Pcaixaaguacmev =  33;
const int Pcisternasubsologerala = 5;
const int Pcisternasubsologeralv =  35;
const int Pcaixaaguasetimoa = 34;
const int Pcaixaaguasetimov =  15;
const int Posmosehemov =  2;
const int Pcompressorar =  4;
const int Pesgotoss2v = 16;
const int Pgerador1a =  17;
const int Pgerador1v =  32;
const int Pgerador2a =  18;
const int Pgerador2v =  19;
const int Pgerador =  21;
const int Penergia =  22;
const int Pcopel =  23;

bool aqsub = 0 ;
bool aqset = 0 ;
bool caguaquia = 0 ;
bool caguaquiv = 0 ;
bool caguacmev = 0 ;
bool cisubgeralv = 0 ;
bool cisubgerala = 0;
bool caguasetv = 0 ;
bool caguaseta = 0 ;
bool oshemo = 0 ;
bool ar=0 ;
bool esgotoss2 = 0;
bool gerador1a = 0;
bool gerador1v = 0;
bool gerador2a = 0;
bool gerador2v = 0;
bool gerador = 0;
bool energia = 0;
bool copel = 0;

bool flag[21];
int  i = 0;
char controle[30]="";
String Scontrole="";

// gpio to use to trigger delay
const int wdtTimeout = 10000;  //time in ms to trigger the watchdog
hw_timer_t *timer = NULL;
```

```
void IRAM_ATTR resetModule() {
  ets_printf("reboot\n");
  esp_restart();
}

void initWiFi();
void initMQTT();
void reconectWiFi();
void mqtt_callback(char* topic, byte* payload, unsigned int length);
void VerificaConexoesWiFIEMQTT(void);
void InitOutput(void);
void leitura();
void Controle();
void Enviarsms(char* txt);
void leitura_gpio();

void setup() {
    Serial.begin(115200);
    initWiFi();
    Serial.println("Contacting Time Server");

    configTime(3600*timezone,  daysavetime*3600,  "time.nist.gov",  "0.pool.ntp.org",
"1.pool.ntp.org");
    struct tm tmstruct ;
    int j=0;
    while(i==0){

      delay(1000);
      tmstruct.tm_year = 0;
      getLocalTime(&tmstruct, 5000);
      Serial.print(".");
      if((tmstruct.tm_year+1900)>2018)
        i=1;
      if(j==5){
          Serial.println("Restart");
          delay(2000);
          ESP.restart();
        }
        j++;
    }
    i=0;
    j=0;
    Serial.println("");
    Serial.printf("\nNow                is                :                %d-%02d-%02d
%02d:%02d:%02d\n",(tmstruct.tm_year)+1900,(                tmstruct.tm_mon)+1,
tmstruct.tm_mday,tmstruct.tm_hour -5 , tmstruct.tm_min, tmstruct.tm_sec);
    Serial.println("");

    Serial.println("Feito");
```

```
  pinMode(Paquecedorsubsolo, INPUT);
  pinMode(Paquecedorsetimo, INPUT);
  pinMode(Pcaixaaguaquintoa, INPUT);
  pinMode(Pcaixaaguaquintov, INPUT);
  pinMode(Pcaixaaguacmev, INPUT);
  pinMode(Pcisternasubsologeralv, INPUT);
  pinMode(Pcisternasubsologerala, INPUT);
  pinMode(Pcaixaaguasetimov, INPUT);
  pinMode(Pcaixaaguasetimoa, INPUT);
  pinMode(Posmosehemov, INPUT);
  pinMode(Pcompressorar, INPUT);
  pinMode(Pesgotoss2v, INPUT);
  pinMode(Pgerador1a, INPUT);
  pinMode(Pgerador1v, INPUT);
  pinMode(Pgerador2a, INPUT);
  pinMode(Pgerador2v, INPUT);
  pinMode(Pgerador, INPUT);
  pinMode(Penergia, INPUT);
  pinMode(Pcopel, INPUT);
  i=0;
   while (i < 20) {
    flag[i] = 0;
    minuto[i] = 0;
    hora[i] = 0;
    i = i + 1;
  }

    flag[19]=0;
    InitOutput();
    initMQTT();

    timer = timerBegin(0, 80, true);                  //timer 0, div 80
    timerAttachInterrupt(timer, &resetModule, true);  //attach callback
    timerAlarmWrite(timer, wdtTimeout * 1000, false); //set time in us
    timerAlarmEnable(timer);
    i=0;
    hora[19]=-1;
 }

void loop() {
  timerWrite(timer, 0); //reset timer (feed watchdog)
  //garante funcionamento das conexões WiFi e ao broker MQTT
  VerificaConexoesWiFIEMQTT();
  //envia o status de todos os outputs para o Broker no protocolo esperado
  //keep-alive da comunicação com broker MQTT
  MQTT.loop();
  leitura_gpio();
  Controle();

  if(hora[19] != hora[20]){
```

```
    flag[19]=1;
    leitura();
    hora[19]=hora[20];

    Serial.printf("\nNow is : %d-%02d-%02d
%02d:%02d:%02d\n",(tmstruct.tm_year)+1900,( tmstruct.tm_mon)+1,
tmstruct.tm_mday,tmstruct.tm_hour -5 , tmstruct.tm_min, tmstruct.tm_sec);
    Serial.print("saindo do if flag[19[]:");
    Serial.println(flag[19]);
  }
}

void Enviarsms(char* txt){
  bott=0;
  Serial.println("Enviando SMS, um momento...");//354323177
  Serial.println(txt);
  delay(200);
  bott =  bot.sendMessage("-388472861", txt); // Grupo x9 bot.sendMessage
  Serial.print("Valor de Bott:");
  Serial.println(bott);
  delay(200);
    if(bott==0){
       Serial.println("Restart");
       delay(2000);
       ESP.restart();
    }

  flag[19] = 0;
}

void VerificaConexoesWiFIEMQTT(void)
{
    if (!MQTT.connected())
        reconnectMQTT(); //se não há conexão com o Broker, a conexão é refeita

     reconectWiFi(); //se não há conexão com o WiFI, a conexão é refeita
}

void EnviaEstadoOutputMQTT(char* TOPICO_PUBLISH,char* valor)
{

    MQTT.publish(TOPICO_PUBLISH, valor,true);

    Serial.println(" enviado ao broker!");
    Serial.print(TOPICO_PUBLISH);
    Serial.print(" : ");
    Serial.println(valor);
    delay(100);
    return ;
}
```

```
void initWiFi()
{
    delay(10);
    Serial.println("------Conexao WI-FI------");
    Serial.print("Conectando-se na rede: ");
    Serial.println(ssid);
    Serial.println("Aguarde");

    reconectWiFi();
}

void reconectWiFi()
{
    //se já está conectado a rede WI-FI, nada é feito.
    //Caso contrário, são efetuadas tentativas de conexão
    if (WiFi.status() == WL_CONNECTED)
        return;

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED)
    {
        delay(100);
        Serial.print(".");
    }

    Serial.println();
    Serial.print("Conectado com sucesso na rede.... ");
    Serial.print(ssid);
    Serial.println("IP obtido: ");
    Serial.println(WiFi.localIP());
}

void initMQTT()
{
    MQTT.setServer(BROKER_MQTT, BROKER_PORT);     //informa qual broker e
porta deve ser conectado
    MQTT.setCallback(mqtt_callback);              //atribui função de callback (função
chamada quando qualquer informação de um dos tópicos subescritos chega)
}
  void mqtt_callback(char* topic, byte* payload, unsigned int length)
{
}
```

```
void reconnectMQTT()
{
    while (!MQTT.connected())
    {
        Serial.print("* Tentando se conectar ao Broker MQTT: ");
        Serial.println(BROKER_MQTT);
        if (MQTT.connect(ID_MQTT, Username, Password, WillTopic, WillQoS,
WillRetain, WillMessage)) // connect (clientID, ,   )
        {
            Serial.println("Conectado com sucesso ao broker MQTT!");
            //MQTT.subscribe(TOPICO_SUBSCRIBE);
        }
        else
        {
            Serial.println("Falha ao reconectar no broker.");
            Serial.println("Havera nova tentatica de conexao em 2s");
            delay(2000);
        }
    }
}

void InitOutput(void)
{
    //IMPORTANTE: o Led já contido na placa é acionado com lógica invertida (ou
seja,
}

void leitura(void){
  if(aqsub==HIGH)
    EnviaEstadoOutputMQTT("HPP/SS2/AQUECEDOR",VERMELHO);
  else
    EnviaEstadoOutputMQTT("HPP/SS2/AQUECEDOR",VERDE);

    if(aqset==HIGH)
    EnviaEstadoOutputMQTT("HPP/7/AQUECEDOR",VERMELHO);
  else
    EnviaEstadoOutputMQTT("HPP/7/AQUECEDOR",VERDE);

  if(caguaquia==HIGH)
    EnviaEstadoOutputMQTT("HPP/5/CX_AGUA",AMARELO);
  else
    EnviaEstadoOutputMQTT("HPP/5/CX_AGUA",VERDE);


  if(caguaquiv==HIGH)
    EnviaEstadoOutputMQTT("HPP/5/CX_AGUA",VERMELHO);
  else
    EnviaEstadoOutputMQTT("HPP/5/CX_AGUA",VERDE);
```

```
if(caguacmev==HIGH)
  EnviaEstadoOutputMQTT("HPP/T/CX_AGUA/CME",VERMELHO);
 else
  EnviaEstadoOutputMQTT("HPP/T/CX_AGUA/CME",VERDE);
 if(cisubgeralv==HIGH)
  EnviaEstadoOutputMQTT("HPP/SS2/CX_AGUA",VERMELHO);
 else
  EnviaEstadoOutputMQTT("HPP/SS2/CX_AGUA",VERDE);

 if(caguasetv==HIGH)
  EnviaEstadoOutputMQTT("HPP/7/CX_AGUA",VERMELHO);
 else{
  if(caguaseta==HIGH)
  EnviaEstadoOutputMQTT("HPP/7/CX_AGUA",AMARELO);
  else
  EnviaEstadoOutputMQTT("HPP/7/CX_AGUA",VERDE);
 }

 if(oshemo==HIGH)
  EnviaEstadoOutputMQTT("HPP/T/OS_AGUA",VERMELHO);
 else
  EnviaEstadoOutputMQTT("HPP/T/OS_AGUA",VERDE);

 if(ar==HIGH)
  EnviaEstadoOutputMQTT("HPP/SS2/AR",VERMELHO);
 else
  EnviaEstadoOutputMQTT("HPP/SS2/AR",VERDE);

 if(esgotoss2==HIGH)
  EnviaEstadoOutputMQTT("HPP/SS2/ESGOTO",VERMELHO);
 else
  EnviaEstadoOutputMQTT("HPP/SS2/ESGOTO",VERDE);

 if(gerador1v==HIGH)
  EnviaEstadoOutputMQTT("HPP/T/GERADOR/1",VERMELHO);
 else{
  if(gerador1a==HIGH)
  EnviaEstadoOutputMQTT("HPP/T/GERADOR/1",AMARELO);
  else
  EnviaEstadoOutputMQTT("HPP/T/GERADOR/1",VERDE);
 }

 if(gerador2v==HIGH)
  EnviaEstadoOutputMQTT("HPP/T/GERADOR/2",VERMELHO);
 else{
  if(gerador2a==HIGH)
  EnviaEstadoOutputMQTT("HPP/T/GERADOR/2",AMARELO);
  else
  EnviaEstadoOutputMQTT("HPP/T/GERADOR/2",VERDE);
```

```
  }
   if(gerador==HIGH)
    EnviaEstadoOutputMQTT("HPP/T/GERADOR",AMARELO);
  else
    EnviaEstadoOutputMQTT("HPP/T/GERADOR",VERDE);

  if(energia==HIGH)
    EnviaEstadoOutputMQTT("HPP/SS1/ENERGIA",AMARELO);
  else
    EnviaEstadoOutputMQTT("HPP/SS1/ENERGIA",VERDE);

  if(copel==HIGH)
    EnviaEstadoOutputMQTT("HPP/SS1/COPEL",AMARELO);
  else
    EnviaEstadoOutputMQTT("HPP/SS1/COPEL",VERDE);
Serial.println(minuto[19]);
}

void Controle(void){
getLocalTime(&tmstruct, 5000);
if((tmstruct.tm_hour -5)==-5)
hora[20]=19;
else if((tmstruct.tm_hour -5)==-4)
hora[20]= 20;
else if((tmstruct.tm_hour -5)==-3)
hora[20]= 21;
else if((tmstruct.tm_hour -5)==-2)
hora[20]= 22;
else if((tmstruct.tm_hour -5)==-1)
hora[20]= 23;
else{
hora[20]= tmstruct.tm_hour -5;
}
minuto[20]=tmstruct.tm_min;

i=0;
alarmes=0;
while (i < 19) {
  alarmes=flag[i]+alarmes;
  if (minuto[i] > 59)
  {
    minuto[i]=minuto[i]-59;
    hora[i]=hora[i]+1;
  }
  if (hora[i] > 23)
  {
    hora[i]=0;
  }
  if (flag[i] == 1 && (hora[20] == hora[i] && minuto[20]== minuto[i] ))
  {
```

```
    flag[i]=0;
  }
  i = i + 1;
  }
  i=0;
 char alarme[3];
if(minuto[19]!= minuto[20]){
  if((tmstruct.tm_hour -5)==-5)
  Scontrole+= 19;
  else if((tmstruct.tm_hour -5)==-4)
  Scontrole+= 20;
  else if((tmstruct.tm_hour -5)==-3)
  Scontrole+= 21;
  else if((tmstruct.tm_hour -5)==-2)
  Scontrole+= 22;
  else if((tmstruct.tm_hour -5)==-1)
  Scontrole+= 23;
  else
  Scontrole+= (tmstruct.tm_hour -5);
  Scontrole+= ":";
  Scontrole+= tmstruct.tm_min;
  Scontrole+= " ";
  Scontrole+= tmstruct.tm_mday;
  Scontrole+= "/";
  Scontrole+= tmstruct.tm_mon+1;
  Scontrole+= "/";
  Scontrole+= tmstruct.tm_year+1900;
  Scontrole.toCharArray(controle, Scontrole.length() + 1);
  Scontrole="";
  EnviaEstadoOutputMQTT("HPP/CONTROLE",controle);
  delay(200);
  Scontrole+= alarmes;
  Scontrole.toCharArray(alarme, Scontrole.length() + 1);
  Scontrole="";
  EnviaEstadoOutputMQTT("HPP/ALARMES",alarme);
  minuto[19]=minuto[20];
}

 }
void leitura_gpio(){
  delay(10);
  aqsub = digitalRead(Paquecedorsubsolo);
  delay(10);
  aqset = digitalRead(Paquecedorsetimo);
  delay(10);
  caguaquia = digitalRead(Pcaixaaguaquintoa);
  delay(10);
  caguaquiv = digitalRead(Pcaixaaguaquintov);
  delay(10);
  caguacmev = digitalRead(Pcaixaaguacmev);
```

```
  delay(10);
  cisubgeralv = digitalRead(Pcisternasubsologeralv);
  delay(10);
  cisubgerala = digitalRead(Pcisternasubsologerala);
  delay(10);
  caguasetv = digitalRead(Pcaixaaguasetimov);
  delay(10);
  caguaseta = digitalRead(Pcaixaaguasetimoa);
  delay(10);
  oshemo = digitalRead(Posmosehemov);
  delay(10);
  ar = digitalRead(Pcompressorar);
  delay(10);
  esgotoss2 = digitalRead(Pesgotoss2v);
  delay(10);
  gerador1v = digitalRead(Pgerador1v);
  delay(10);
  gerador1a = digitalRead(Pgerador1a);
  delay(10);
  gerador2v = digitalRead(Pgerador2v);
  delay(10);
  gerador2a = digitalRead(Pgerador2a);
  delay(10);
  gerador = digitalRead(Pgerador);
  delay(10);
  energia = digitalRead(Penergia);
  delay(10);
  copel = digitalRead(Pcopel);

////////////////////////////////////////////////////////////////////
  if (aqsub == HIGH && flag[0] == 0) {
    delay(2);
    if(digitalRead(Paquecedorsubsolo) == aqsub)
      {
        hora[0]=hora[20];
        minuto[0]=minuto[20]+10;

        Serial.println("");
        Serial.printf("\nNow is : %d-%02d-%02d
%02d:%02d:%02d\n",(tmstruct.tm_year)+1900,( tmstruct.tm_mon)+1,
tmstruct.tm_mday,tmstruct.tm_hour -5 , tmstruct.tm_min, tmstruct.tm_sec);
        Serial.println("");

        txtsms = "Temperatura BAIXA aquecedor SS2 - URGENTE";

        Enviarsms(txtsms);

       Serial.println("Temperatura BAIXA aquecedor SS2 - URGENTE");
        flag[0] = 1;
```

```
        Serial.println(flag[0]);
        EnviaEstadoOutputMQTT("HPP/SS2/AQUECEDOR",VERMELHO);
      }
  }

  if (aqsub == LOW && flag[0] == 1) {
    delay(2);
    if(digitalRead(Paquecedorsubsolo)==aqsub)
      {
        txtsms = "Temperatura NORMAL aquecedor SS2";
        Serial.println("");
        Serial.printf("\nNow is : %d-%02d-%02d
%02d:%02d:%02d\n",(tmstruct.tm_year)+1900,( tmstruct.tm_mon)+1,
tmstruct.tm_mday,tmstruct.tm_hour -5 , tmstruct.tm_min, tmstruct.tm_sec);
        Serial.println("");
        Serial.println("Temperatura NORMAL");
        Enviarsms(txtsms);
        flag[0] = 0;
        Serial.println(flag[0]);
        EnviaEstadoOutputMQTT("HPP/SS2/AQUECEDOR",VERDE);
      }
  }

  ////////////////////////////////////////////////////////////////////
  if (aqset == HIGH && flag[1] == 0) {
    delay(2);
    if(digitalRead(Paquecedorsetimo) == aqset)
      {
        hora[1]=hora[20];
        minuto[1]=minuto[20]+20;
        txtsms = "Temperatura BAIXA aquecedor 7 andar - URGENTE";
        Serial.println("");
        Serial.printf("\nNow is : %d-%02d-%02d
%02d:%02d:%02d\n",(tmstruct.tm_year)+1900,( tmstruct.tm_mon)+1,
tmstruct.tm_mday,tmstruct.tm_hour -5 , tmstruct.tm_min, tmstruct.tm_sec);
        Serial.println("");
        Serial.println("Temperatura NORMAL");

        Serial.println("Temperatura BAIXA aquecedor 7 andar - URGENTE");
        flag[1] = 1;
        Enviarsms(txtsms);
        EnviaEstadoOutputMQTT("HPP/7/AQUECEDOR",VERMELHO);
      }
  }

  if (aqset == LOW && flag[1] == 1) {
    delay(2);
    if(digitalRead(Paquecedorsetimo) == aqset)
      {
        txtsms = "Temperatura NORMAL aquecedor 7 andar";
```

```
      Serial.println("");
      Serial.printf("\nNow is : %d-%02d-%02d
%02d:%02d:%02d\n",(tmstruct.tm_year)+1900,( tmstruct.tm_mon)+1,
tmstruct.tm_mday,tmstruct.tm_hour -5 , tmstruct.tm_min, tmstruct.tm_sec);
      Serial.println("");
      Serial.println("Temperatura NORMAl aquecedor 7 andar ");
      flag[1] = 0;
      Enviarsms(txtsms);
      EnviaEstadoOutputMQTT("HPP/7/AQUECEDOR",VERDE);
    }
 }

 ////////////////////////////////////////////////////////////////////////////daquiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii
 if (caguaquia == HIGH && flag[2] == 0) {
  delay(2);
  if(digitalRead(Pcaixaaguaquintoa) == caguaquia)
    {
      hora[2]=hora[20];
      minuto[2]=minuto[20]+20;
      txtsms = "Caixa dagua 5 andar NIVEL BAIXO - URGENTE ";
      Enviarsms(txtsms);
      flag[2] = 1;
      EnviaEstadoOutputMQTT("HPP/5/CX_AGUA",AMARELO);
    }
 }
 if (caguaquia == LOW && flag[2] == 1) {
  delay(2);
  if(digitalRead(Pcaixaaguaquintoa) == caguaquia)
    {
      txtsms = "Caixa dagua 5 andar NORMAL";
      Enviarsms(txtsms);
      flag[2] = 0;
      EnviaEstadoOutputMQTT("HPP/5/CX_AGUA",VERDE);
    }
 }

 ////////////////////////////////////////////////////////////////////////
 if (caguaquiv == HIGH && flag[3] == 0) {
  delay(2);
  if(digitalRead(Pcaixaaguaquintov) == caguaquiv)
    {
      hora[3]=hora[20];
      minuto[3]=minuto[20]+20;
      txtsms = "Caixa dagua 5 andar VAZIA - URGENTE";
      Enviarsms(txtsms);
      flag[3] = 1;
      EnviaEstadoOutputMQTT("HPP/5/CX_AGUA",VERMELHO);
    }
 }
```

```
if (caguaquiv == LOW && flag[3] == 1) {
    delay(2);
   if(digitalRead(Pcaixaaguaquintov) == caguaquiv)
      {
        txtsms = "Caixa dagua NORMAL 5 andar";
        Enviarsms(txtsms);
        flag[3] = 0;
        EnviaEstadoOutputMQTT("HPP/5/CX_AGUA",VERDE);
      }
  }

  ////////////////////////////////////////////////////////////////////
  if (caguacmev == HIGH && flag[5] == 0) {
    delay(2);
    if(digitalRead(Pcaixaaguacmev) == caguacmev)
      {
        hora[5]=hora[20];
        minuto[5]=minuto[20]+20;
        txtsms = "Caixa Osmose CME NIVEL BAIXO - URGENTE";
        Enviarsms(txtsms);
        flag[5] = 1;
        EnviaEstadoOutputMQTT("HPP/T/CX_AGUA/CME",VERMELHO);
      }
  }
  if (caguacmev == LOW && flag[5] == 1) {
    delay(2);
    if(digitalRead(Pcaixaaguacmev) == caguacmev)
      {
        txtsms = "Caixa Osmose CME NIVEL NORMAL";
        Enviarsms(txtsms);
        flag[5] = 0;
        EnviaEstadoOutputMQTT("HPP/T/CX_AGUA/CME",VERDE);
      }
  }

  ////////////////////////////////////////////////////////////////////
  if (cisubgeralv == HIGH && flag[6] == 0) {
    delay(2);
    if(digitalRead(Pcisternasubsologeralv) == cisubgeralv)
      {
        hora[6]=hora[20];
        minuto[6]=minuto[20]+20;
        txtsms = "Cisterna VAZIA SS2 - URGENTE";
        Enviarsms(txtsms);
        flag[6] = 1;
        EnviaEstadoOutputMQTT("HPP/SS2/CX_AGUA",VERMELHO);
      }
  }
```

```
if (cisubgeralv == LOW && flag[6] == 1) {
    delay(2);
    if(digitalRead(Pcisternasubsologeralv) == cisubgeralv)
      {
        txtsms = "Cisterna SS2 NORMAL";
        Enviarsms(txtsms);
        flag[6] = 0;
        EnviaEstadoOutputMQTT("HPP/SS2/CX_AGUA",VERDE);
      }
  }

  ////////////////////////////////////////////////////////////////////
  if (caguasetv == HIGH && flag[7] == 0) {
    delay(2);
    if(digitalRead(Pcaixaaguasetimov) == caguasetv)
      {
        hora[7]=hora[20];
        minuto[7]=minuto[20]+20;
        txtsms = "Caixa dagua 7 andar VAZIA - URGENTE";
        Enviarsms(txtsms);
        flag[7] = 1;
        EnviaEstadoOutputMQTT("HPP/7/CX_AGUA",VERMELHO);
      }
  }
  if (caguasetv == LOW && flag[7] == 1 && flag[8] == 0) {//ver aqui se vai funcionar
    delay(2);
    if(digitalRead(Pcaixaaguasetimov) == caguasetv)
      {
        txtsms = "Caixa dagua 7 andar NORMAL";
        Enviarsms(txtsms);
        flag[7] = 0;
        EnviaEstadoOutputMQTT("HPP/7/CX_AGUA",VERDE);
      }
  }

////////////////////////////////////////////////////////////////////
  if (caguaseta == HIGH && flag[8] == 0 && flag[7] == 0) {
    delay(2);
    if(digitalRead(Pcaixaaguasetimoa) == caguaseta)
      {
        hora[8]=hora[20];
        minuto[8]=minuto[20]+20;
        txtsms = "Caixa dagua 7 andar NIVEL BAIXO";
        Enviarsms(txtsms);
        flag[8] = 1;
        EnviaEstadoOutputMQTT("HPP/7/CX_AGUA",AMARELO);
      }
  }
```

```
if (caguaseta == LOW && flag[8] == 1 && flag[7] == 0) {
    delay(2);
    if(digitalRead(Pcaixaaguasetimoa) == caguaseta)
      {
        txtsms = "Caixa dagua 7 andar nivel NORMAL";
        Enviarsms(txtsms);
        flag[8] = 0;
        EnviaEstadoOutputMQTT("HPP/7/CX_AGUA",VERDE);
      }
  }

  ////////////////////////////////////////////////////////////////
  if (oshemo == HIGH && flag[9] == 0) {
    delay(2);
    if(digitalRead(Posmosehemov) == oshemo)
      {
        hora[9]=hora[20];
        minuto[9]=minuto[20]+20;
        txtsms = "Alarme agua Hemodialise BAIXO - URGENTE";
        Enviarsms(txtsms);
        flag[9] = 1;
        EnviaEstadoOutputMQTT("HPP/T/OS_AGUA",VERMELHO);
      }
  }

  if (oshemo == LOW && flag[9] == 1) {
    delay(2);
    if(digitalRead(Posmosehemov) == oshemo)
      {
        txtsms = "Agua Hemodialise NORMAL";
        Enviarsms(txtsms);
        flag[9] = 0;
        EnviaEstadoOutputMQTT("HPP/T/OS_AGUA",VERDE);
      }
  }

  ////////////////////////////////////////////////////////////////
  if (ar == HIGH && flag[10] == 0) {
    delay(2);
    if(digitalRead(Pcompressorar) == ar)
      {
        hora[10]=hora[20];
        minuto[10]=minuto[20]+20;
        txtsms = "Ar Comprimido SS2 FALHA no SISTEMA - URGENTE";
        Enviarsms(txtsms);
        flag[10] = 1;
        EnviaEstadoOutputMQTT("HPP/SS2/AR",VERMELHO);
      }
  }
```

```
  if (ar == LOW && flag[10] == 1) {
    delay(2);
    if(digitalRead(Pcompressorar) == ar)
      {
        txtsms = "Ar comprimido SS2 NORMAL";
        Enviarsms(txtsms);
        flag[10] = 0;
        EnviaEstadoOutputMQTT("HPP/SS2/AR",VERDE);
      }
  }

  ////////////////////////////////////////////////////////////////////
  if (esgotoss2 == HIGH && flag[11] == 0) {
    delay(2);
    if(digitalRead(Pesgotoss2v) == esgotoss2)
      {
        hora[11]=hora[20];
        minuto[11]=minuto[20]+20;
        txtsms = "Esgoto SS2 nivel ALTO - URGENTE";
        Enviarsms(txtsms);
        flag[11] = 1;
        EnviaEstadoOutputMQTT("HPP/SS2/ESGOTO",VERMELHO);
      }
  }

  if (esgotoss2 == LOW && flag[11] == 1) {
    delay(2);
    if(digitalRead(Pesgotoss2v) == esgotoss2)
      {
        txtsms = "Esgoto SS2 nivel NORMAL";
        Enviarsms(txtsms);
        flag[11] = 0;
        EnviaEstadoOutputMQTT("HPP/SS2/ESGOTO",VERDE);
      }
  }

////////////////////////////////////////////////////////////////////
  if (gerador1a == HIGH && flag[12] == 0) {
    delay(2);
    if(digitalRead(Pgerador1a) == gerador1a)
      {
        hora[12]=hora[20];
        minuto[12]=minuto[20]+20;
        txtsms = "Gerador 1 falha nao critica";
        Enviarsms( txtsms);
        flag[12] = 1;
        EnviaEstadoOutputMQTT("HPP/T/GERADOR/1",AMARELO);
      }
  }
```

```
 if (gerador1a == LOW && flag[12] == 1 && flag[13] == 0) {//testeeeeeeeeeee
    delay(2);
    if(digitalRead(Pgerador1a) == gerador1a)
     {
       txtsms = "Gerador 1 sem falhas nao criticas, NORMAL";
       Enviarsms( txtsms);
       flag[12] = 0;
       EnviaEstadoOutputMQTT("HPP/T/GERADOR/1",VERDE);
    }
 }

 ////////////////////////////////////////////////////////////////
 if (gerador1v == HIGH && flag[13] == 0) {
   delay(2);
   if(digitalRead(Pgerador1v) == gerador1v)
     {
       hora[13]=hora[20];
       minuto[13]=minuto[20]+20;
       txtsms = "Urgente Falha critica  Gerador 1";
       Enviarsms(txtsms);
       flag[13] = 1;
       EnviaEstadoOutputMQTT("HPP/T/GERADOR/1",VERMELHO);
    }
 }

 if (gerador1v == LOW && flag[13] == 1 && flag[12] == 0) { //testeeeeeeee
   delay(2);
   if(digitalRead(Pgerador1v) == gerador1v)
     {
       txtsms = "Gerador 1 Normal";
       Enviarsms( txtsms);
       flag[13] = 0;
       EnviaEstadoOutputMQTT("HPP/T/GERADOR/1",VERDE);
     }
 }

////////////////////////////////////////////////////////////////
 if (gerador2a == HIGH && flag[14] == 0) {
   delay(2);
   if(digitalRead(Pgerador2a) == gerador2a)
     {
       hora[14]=hora[20];
       minuto[14]=minuto[20]+20;
       txtsms = "Gerador 2  falha nao critica";
       Enviarsms( txtsms);
       flag[14] = 1;
       EnviaEstadoOutputMQTT("HPP/T/GERADOR/2",AMARELO);
    }
 }
```

```
  if (gerador2a == LOW && flag[14] == 1 && flag[15] == 0) {
    delay(2);
    if(digitalRead(Pgerador2a) == gerador2a)
      {
        txtsms = "Gerador 2 sem falhas nao criticas, NORMAL";
        Enviarsms( txtsms);
        flag[14] = 0;
        EnviaEstadoOutputMQTT("HPP/T/GERADOR/2",VERDE);
     }
  }

  ////////////////////////////////////////////////////////////////////
  if (gerador2v == HIGH && flag[15] == 0) {
    delay(2);
    if(digitalRead(Pgerador2v) == gerador2v)
      {
        hora[15]=hora[20];
        minuto[15]=minuto[20]+20;
        txtsms = "Urgente Falha critica Gerador 2";
        Enviarsms(txtsms);
        flag[15] = 1;
        EnviaEstadoOutputMQTT("HPP/T/GERADOR/2",VERMELHO);
     }
  }

  if (gerador2v == LOW && flag[15] == 1 && flag[14] == 0) {
    delay(2);
    if(digitalRead(Pgerador2v) == gerador2v)
      {
        txtsms = "Gerador 2 Normal";
        Enviarsms( txtsms);
        flag[15] = 0;
        EnviaEstadoOutputMQTT("HPP/T/GERADOR/2",VERDE);
      }
  }

////////////////////////////////////////////////////////////////////
  if (energia == HIGH && flag[17] == 0) {
    delay(2);
    if(digitalRead(Penergia) == energia)
      {

        hora[17]=hora[20]+1;
        minuto[17]=minuto[20];
        txtsms = "SEM ENERGIA ELETRICA ";
        Enviarsms( txtsms);
        flag[17] = 1;
        EnviaEstadoOutputMQTT("HPP/SS1/ENERGIA",VERMELHO);
    }
  }
```

```
  if (energia == LOW && flag[17] == 1) {
    delay(2);
    if(digitalRead(Penergia) == energia)
      {
        txtsms = "COM ENERGIA ELETRICA";
        Enviarsms(txtsms);
        flag[17] = 0;
        EnviaEstadoOutputMQTT("HPP/SS1/ENERGIA",VERDE);
     }
  }

  if (gerador == HIGH && flag[16] == 0) {
    delay(2);
    if(digitalRead(Pgerador) == gerador)
      {
      hora[16]=hora[20];
      minuto[16]=minuto[20]+20;
      txtsms = "Geradores em Operacao";
      Enviarsms(txtsms);
      flag[16] = 1;
      EnviaEstadoOutputMQTT("HPP/T/GERADOR",AMARELO);
      }
  }

  if (gerador == LOW && flag[16] == 1) {
    delay(2);
    if(digitalRead(Pgerador) == gerador)
      {
      txtsms = "Geradores Desligaram";
      Enviarsms(txtsms);
      flag[16] = 0;
      EnviaEstadoOutputMQTT("HPP/T/GERADOR",VERDE);
      }
  }

 if (copel == HIGH && flag[18] == 0) {
    delay(2);
    if(digitalRead(Pcopel) == copel)
      {
        hora[18]=hora[20];
        minuto[18]=minuto[20]+10;
        txtsms = "SEM REDE COPEL ";
        Enviarsms(txtsms);
        flag[18] = 1;
        EnviaEstadoOutputMQTT("HPP/SS1/COPEL",VERMELHO);
      }
  }
```

```
 if (copel == LOW && flag[18] == 1) {
    delay(2);
    if(digitalRead(Pcopel) == copel)
      {
        txtsms = "COM ENERGIA ELETRICA COPEL";
        Enviarsms(txtsms);
        flag[18] = 0;
        EnviaEstadoOutputMQTT("HPP/SS1/COPEL",VERDE);
      }
  }
}
```