REVISTA CONEXÃO ELETRÔNICA

MONITORAMENTO VIRTUAL DE ATIVOS EM MOTORES ELÉTRICOS

Andre Soares Bispo[1]; Ronaldo Vieira de Souza Júnior[1]; Richard Vieira do Espirito Santo[2,7]; André Aparecido Leal de Almeida[3,7]; Luan Estevam Francisco[4,7]; Lucas Nuud Táparo[5,7]; Thiago Raniel[6,7]*

[1] Graduando em Engenharia Elétrica, Faculdade Integradas de Três Lagoas – AEMS; [2] Esp. em Engenharia de Software – Universidade Estácio de Sá; [3] Esp. em Segurança Cibernética – IGTI; [4] Graduação em Sistemas para Internet – IFMS; [5] Esp. em Engenharia de Dados – UNOPAR; [6] Mestre em Engenharia Elétrica – UNESP; [7] Docente das faculdades integradas de Três Lagoas – FITL/AEMS
* autor correspondente: thiago.raniel@gmail.com

RESUMO
Com os conceitos da indústria 4.0, trataremos sobre o Monitoramento de motores elétricos industriais, utilizando o Arduíno Uno, Raspberry Pi 3, software Domotic Z e o software Callmebot. Através dos dados adquiridos e tratados, o enfoque será a maior operacionalidade e confiabilidade dos ativos para que por meio dessas anomalias seja possível realizar estudos de prevenção de falhas indesejadas e possíveis soluções, garantido assim uma melhor vida útil dos motores e a redução considerável no tempo dos equipamentos parados, diminuindo as perdas nas produções e auxiliando para que seja de maior agilidade as identificações das falhas nos equipamentos, sendo assim possível realizar a supervisão destes por meios de plataformas online, possibilitando o processamento de informações a longa distancias, no qual será utilizando um conjunto pequenos equipamentos que informará algum dados, essas informações serão enviadas em algumas plataformas, variando e podendo ser enviadas para um Whats App ou até mesmo via e-mail, podendo conter nas mensagens de falhas as informações de quais falhas estão ativas e sendo implementado um texto de apoio possibilitando a localização do equipamento e qual o métodos adotará para que seja realizado o serviço de forma ágil, segura e eficaz, sendo algumas dessas falhas do tipo de temperatura, trip, motor desligado ou ligado dentre outros.

PALAVRAS-CHAVE: monitoramento de ativos; comunicação virtual; microprocessadores.

1 INTRODUÇÃO

A primeira revolução industrial (segunda metade do século XVIII até metade do século XIX) caracteriza-se pela introdução da máquina a vapor, que usa água e vapor para mecanizar a produção, antes essencialmente artesanal. A segunda revolução industrial (meados do século XIX até metade do século XX) caracteriza-se pelo advento da energia elétrica, o que facilita as linhas de produção em massa. A terceira revolução industrial, desenvolvida na segunda metade do século XX, caracteriza-se pela implantação de componentes eletrônicos e tecnologia que permite automação dos processos produtivos. A quarta revolução industrial, iniciada na primeira década do século XXI, caracteriza-se pela digitalização da produção que possibilita a personalização da produção em massa. Esta utiliza internet ubíqua e móvel, sensores menores e mais poderosos e inteligência artificial, que possibilitam mudanças profundas na forma de produção e consumo, desencadeando o desenvolvimento de novos modelos de

negócios (DELOITTE, 2014).

O monitoramento eficaz é de grande importância para assegurar a funcionalidade, eficiência de equipamentos de uso específico e melhoria da operacionalidade de ativos. Atualmente, isto é possível com a utilização de microcontroladores (Arduino), microprocessadores (Raspberry), softwares (IDE, DomoticZ e Callmebot) e plataformas digitais (WhatsApp, e-mail etc.)

2 OBJETIVOS

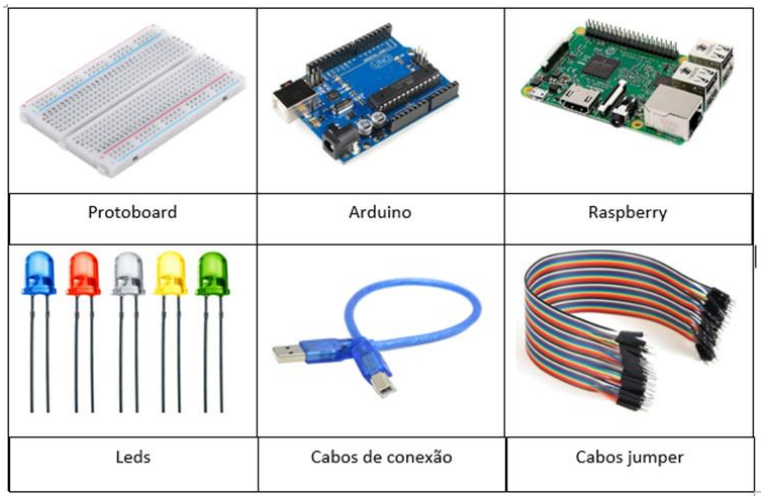
O objetivo deste estudo consiste em desenvolver um projeto para realizar monitoramento virtual à distância em motores elétricos dentro das indústrias de modo a evitar paradas não desejadas por longos períodos e consequente minimização das perdas na produção.

3 MATERIAL E MÉTODOS

Os fundamentos teóricos basearam-se em artigos científicos nacionais e internacionais, indexados em plataformas como Google acadêmico, Scielo, manuais de fabricantes, algoritmos e scripts.

O projeto constituiu-se de um microcontrolador Arduino uno, um microprocessador Raspberry pi 3, três softwares (IDE, Domoticz e callMeBot), protoboard, leds, resistores, cabos para comunicações, cabos conectores e o aplicativo WhatsApp (Figura 1).

Figura 1. Materiais necessários para a montagem do projeto.



Fonte: Extraído de: Protoboard – Casa da Robótica, s.d.; Arduino – filipeflop, raspberry – saravati, leds – foxlux, Cabo de conexão – aliexpress, Jumper – zoiid.

O desenvolvimento do projeto constituiu-se de (i) programação do microcontrolador Arduino uno; (ii) programação do software online Domoticz; (iii) aquisição do código API e (iv) comunicação entre Domoticz e aplicativo WhatsApp.

3.1 Programação do microcontrolador Arduino uno

O processo da programação do Arduino, realizado pelo software IDE, baseou-se na linguagem C/C++ com variáveis definidas de leitura de entrada e saída, cujo algoritmo está mostrado no Anexo 1.

Após a programação, realizou-se um teste de verificação, no qual se fez a leitura dos níveis lógicos, variáveis e condições. Como não se encontrou nenhuma avaria relacionada à elaboração da escrita, estando de forma correta as condições de programação, seguiu-se para a compilação (carregamento das

informações do software para o microcontrolador Arduino uno).

3.2 Programação do *software online* Domoticz

A programação do *software online* Domoticz no microprocessador Raspberry é necessária para torná-lo apto a receber informações do Arduino e enviar as mensagens ao aplicativo WhatsApp.

Inicialmente, definiram-se alguns parâmetros para capacitar o Domoticz a processar as informações advindas do Arduino por meio de layers com os respectivos nomes das possíveis falhas geradas no equipamento, tais como motor desligado, motor ligado, alta temperatura e/ou trip do motor. Para a entrega das notificações recebidas via WhatsApp ao usuário, necessitou a aquisição do código API.

3.3 Aquisição do código API

A comunicação do Domoticz com o WhatsApp foi possibilitada pelo software CallMe bot, um tipo de mensageiro intermediário simples, de fácil utilização, disponível como número para WhatsApp (necessidade da aquisição do código API) e gratuito.

O API é obtido pelo WhatsApp por meio das etapas (i) salvamento do número de comunicação do callMe bot (+34 644 17 94 64); (ii) envio da mensagem com a frase “I allow callmebot to send me messages”; (iii) resposta do CallMe Bot com a mensagem “CallMe Bot API Activated for”, número do contato do usuário e número do código API para a comunicação entre Domoticz e CallMe bot.

3.4 Comunicação entre Domoticz e aplicativo WhatsApp

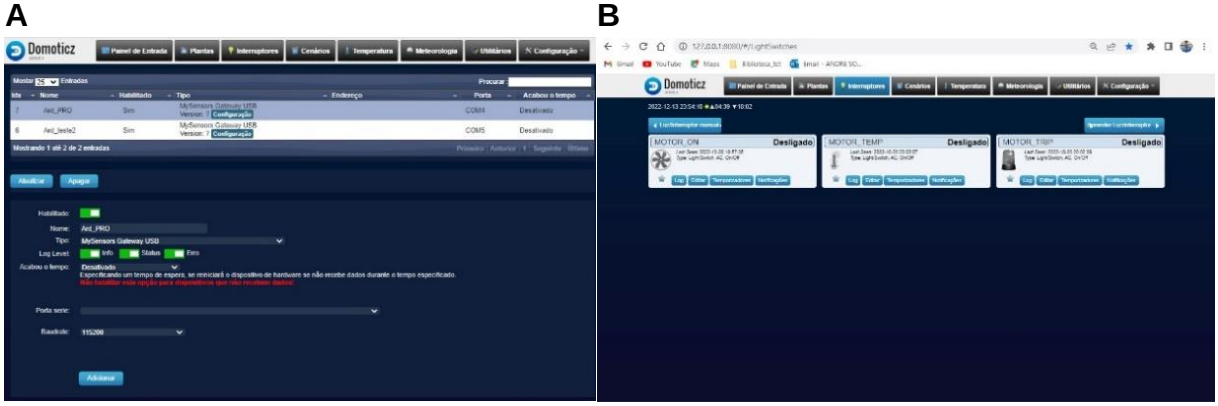
A comunicação do Domoticz com o Whatsapp foi possível pela configuração do primeiro (criação de switches com as possíveis falhas) para a inserção de mensagens com a notificação da(s) falha(s), conforme mostrado na Figura 2.

Para possibilitar o Domoticz a enviar mensagens do status do motor, adicionou-se uma mensagem no switch, conforme segue https://api.callmebot.com/whatsapp.php?source=dom&apikey=986262&phone=+códigopaís-DDDnúmerotelefone96203263&text=Temperatura_alta. A título de observação, não se deve ter espaço entre código do país, DDD e número do telefone do usuário. Assim, foi possível obter o alarme técnico desejado para o ativo monitorado de forma virtual via Whatsapp (Figura 3).

Após programação do microcontrolador Arduino e configurações do software Domoticz no microprocessador Raspberry, realizou-se a montagem do projeto.

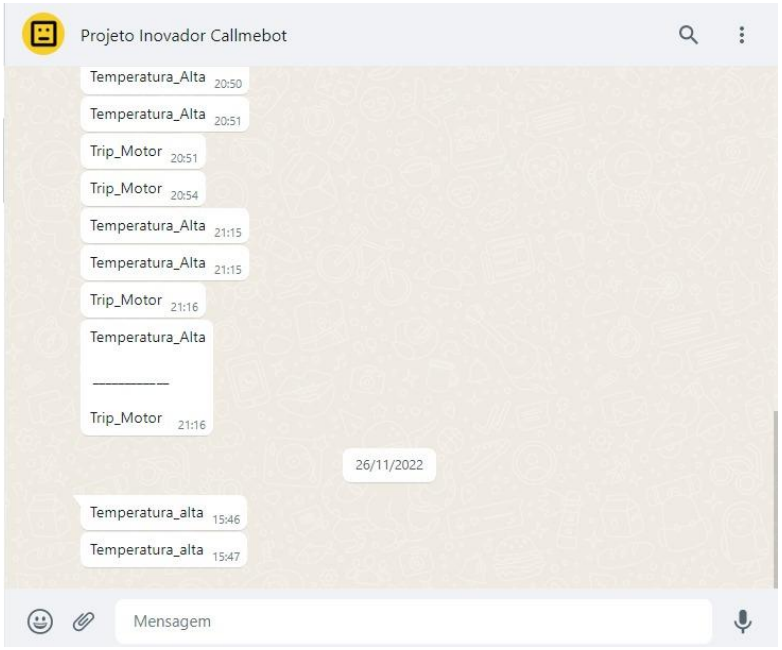
Figura 2. Telas de configuração do *Software* Domoticz. A. Configuração do hardware. B. Configuração dos alarmes.

A **B**



Fonte: Extraído de Software Domoticz, s.d.

Figura 3. Monitoramento do ativo via Whatsapp.



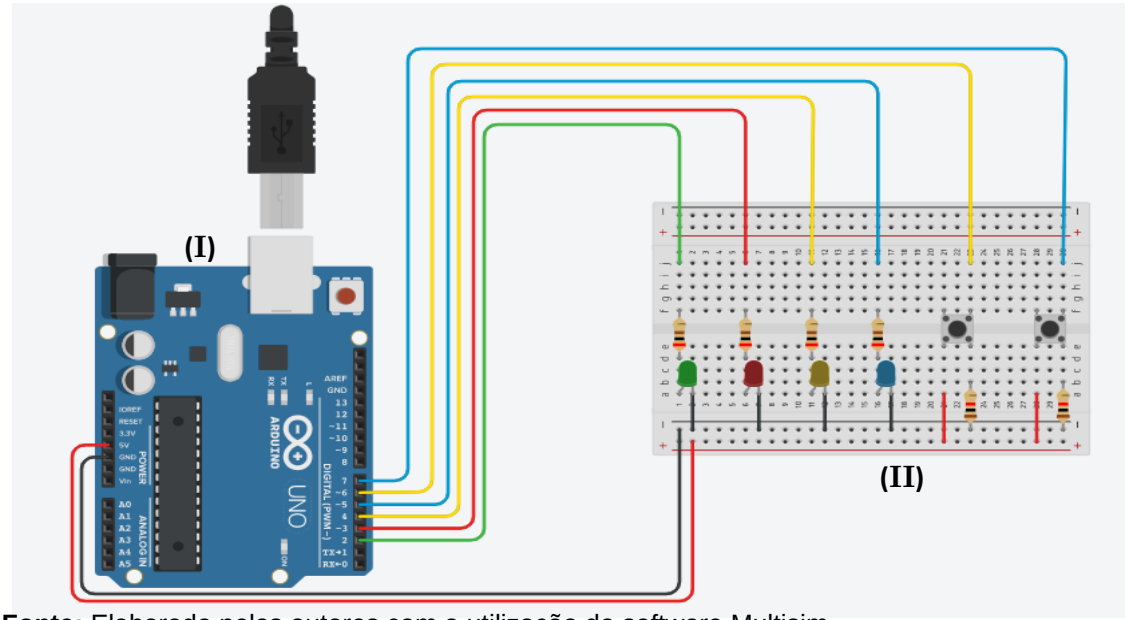
Fonte: Elaborado pelos autores.

4 RESULTADOS E DISCUSSÃO

O circuito desenvolvido (Figura 4) consistiu em um microcontrolador Arduino Uno (Figura 4I) e um protoboard. Este é responsável pelo circuito eletrônico, contém quatro leds de cores diferentes que indicam condições específicas (normais ou falhas) do equipamento, resistores e botões (Figura 4II). Os pinos digitais 2, 3, 4 e 5 são pinos de saídas e acionam os respectivos leds (protoboard) a acenderem, enquanto os pinos 6 e 7 recebem sinais externos do equipamento e os envia aos pinos de saídas, conforme a anomalia detectada.

O led verde (OFF) indica motor desligado; o vermelho (ON) indica motor ligado; amarelo indica alta temperatura e o azul indica trip do motor (Quadro 1).

Figura 4. Representação esquemática do layout do circuito eletrônico para monitoramento virtual à distância.



Fonte: Elaborado pelos autores com a utilização do software Multisim.

Quadro 1. Indicação das cores dos leds, conforme a condição do motor.

Leds	Condição do motor
Verde (pino 2)	Desligado
Vermelho (pino 3)	Ligado
Amarelo (pino 4)	Alta temperatura
Azul (pino 5)	Trip

Fonte: Elaborado pelos autores.

Em condições normais de funcionamento do equipamento, o led vermelho se mantém aceso, ou seja, a máquina está ligada. Caso aconteça alguma anomalia, os pinos digitais 6 e/ou 7 (Arduino) recebem sinais externos do equipamento e os manda para o pino digital 2 (Arduino), e este, instantaneamente, aciona o desligamento do motor e acende o led verde (apto para manutenção segura). Isto garante a segurança da equipe de manutenção, uma vez que o religamento do equipamento é realizado pelo próprio colaborador por meio do software Domoticz.

Em situações de temperatura máxima de trabalho, o pino 6 (Arduino) recebe este sinal externo e o envia, instantaneamente aos pinos 2 e 4 (Arduino). O primeiro aciona o led verde (como mencionado acima) e o pino 4 aciona o led amarelo (alta temperatura) e envia a mensagem de falha ao aplicativo WhatsApp, via software Domoticz.

Em situações de trip de motor, o pino 7 (Arduino) recebe este sinal externo e o envia, instantaneamente aos pinos 2 e 5 (Arduino). O primeiro aciona o led verde (como mencionado acima) e o pino 5 aciona o led azul (trip de motor) e envia a mensagem de falha ao aplicativo WhatsApp, via software Domoticz.

A simulação da falha de temperatura foi realizada pela utilização de um botão; toda vez que este era acionado, o nível lógico da porta 6 mudava de estado e comutava o pino 4 para nível lógico alto e acendia o led amarelo, indicando no painel o aumento de temperatura. A simulação da falha de trip do motor (sobretensão, subtensão e sobrecorrente) também foi realizada pela utilização de um botão; toda vez que este era acionado, o nível lógico da porta 7 mudava de estado e comutava o pino 5 para nível lógico alto e acendia o led azul, indicando no painel o trip específico. O link <https://www.youtube.com/watch?v=9iHJzuRyASs> mostra as duas simulações.

Todas as falhas ocorridas no motor são notificadas ao software Domoticz, o qual aciona um sistema de intertravamento do motor. O religamento é possível somente após o restabelecimento das condições ideais de funcionamento do motor e redefinir este parâmetro no sistema operacional acima citado.

5 CONCLUSÕES

A metodologia adotada para o desenvolvimento deste projeto teve por objetivo obter um sistema de baixo custo para o monitoramento dos ativos industriais usando o Raspberry, Arduino, Domoticz e CallmeBot. O software e o hardware obtidos foram muito satisfatórios para esta implementação, pois além de fácil implementação foi possível um monitoramento eficaz e em tempo real do ativo, diminuindo o tempo de parada do processo industrial. Para este projeto, foi proposto o monitoramento de um motor elétrico, que é um equipamento muito importante para os processos industriais. Assim, para diminuir o tempo de parada do processo industrial e melhorar a agilidade da manutenção industrial, quando o sistema detecta a falha de um motor, o eletricista recebe em seu celular em tempo real qual o ativo falhou e qual o tipo da falha. Para trabalhos futuros, sugere-se que seja implementado uma API no Telegram (menor custo em relação ao Whatsapp) e que na mensagem de alarme seja anexado um arquivo em PDF com o Procedimento Operacional para a manutenção do ativo que apresentou a falha. Com isso, pode-se diminuir ainda mais o tempo de parada na manutenção de ativos, além disso,

padroniza-se a manutenção industrial e garante mais segurança no trabalho durante a execução da tarefa. Portanto, no final deste estudo, obteve-se uma forma simples de se monitorar os motores garantindo uma boa performance e vida útil destes equipamentos.

REFERÊNCIAS

AIRES, R. W. A.; MOREIRA, F. K.; FEIRE, P. S. Indústria 4.0: competências requeridas aos profissionais da quarta revolução industrial, 2017. Disponível em: <https://proceeding.ciki.ufsc.br/index.php/ciki/article/view/314/153>. Acesso em: 10 abr. 2022

ARDUINO. Disponível em: <https://www.filipeflop.com/categoria/arduino/placas-arduino/>. Acesso em: 2 jun. 2022.

CABO DE CONEXÃO. Disponível em <https://pt.aliexpress.com/item/32948960337.html>. Acesso em: 2 jun. 2022.

CALLMEBOT. API gratuita para enviar mensagens para o Whatsap, 2021. Disponível em: <https://www.callmebot.com/>. Acesso em: 11 abr. 2022

DOMOTICZ. Control at your fingertips, 2015. Disponível em: <https://domoticz.com/>. Acesso em: 9 abr. 2022

JUMPER. Disponível em: <https://www.zoiid.com.br/MLB-2124390119-cabo-wire-jumper-40-fios-macho-macho-30cm-protoboard-arduino-_JM>. Acesso em: 2 jun. 2022.

LEDS. Disponível em: <https://www.foxlux.com.br/blog/foxlux-2/como-funcionam-as-luzes-de-led/>. Acesso em: 2 jun. 2022.

OLHAR DIGITAL. Raspberry Pi: o que é, para que serve e como comprar, 2019. Disponível em: <https://olhardigital.com.br/2019/02/18/noticias/raspberry-pi-o-que-e-para-que-serve-e-como-comprar/>. Acesso em: 10 abr. 2022.

PROTOBOARD. Disponível em: <https://www.casadarobotica.com/prototipagem-e-ferramentas/prototipagem/protoboard/protoboard-400-furos-pontos>. Acesso em: 2 jun. 2022.

THOMSEM, A. O que é Arduíno, 2014. Disponível em: <https://www.filipeflop.com/blog/o-que-e-arduino/>. Acesso em: 11 abr. 2022

RASPBERRY. Disponível em: <https://www.saravati.com.br/raspberry-pi-3-model-b>. Acesso em: 2 jun. 2022.

Anexo 1.

```
// Enable debug prints to serial monitor
#define MY_DEBUG

// Set LOW transmit power level as default, if you have an amplified NRF-module and
// power your radio separately with a good regulator you can turn up PA level.
//#define MY_RF24_PA_LEVEL RF24_PA_LOW

// Enable serial gateway
#define MY_GATEWAY_SERIAL

// Define a lower baud rate for Arduinos running on 8 MHz (Arduino Pro Mini 3.3V & Sense-Bender)
#if F_CPU == 8000000L
#define MY_BAUD_RATE 38400
#endif

// Enable inclusion mode
#define MY_INCLUSION_MODE_FEATURE

// Set inclusion mode duration (in seconds)
#define MY_INCLUSION_MODE_DURATION 60

// Set blinking period
#define MY_DEFAULT_LED_BLINK_PERIOD 300
#include <MySensors.h>

#define CHILD_ID_G 1
#define CHILD_ID_R 2
#define CHILD_ID_Y 3
#define CHILD_ID_B 4

MyMessage msg_G(CHILD_ID_G, V_TRIPPED);
MyMessage msg_R(CHILD_ID_R, V_TRIPPED);
MyMessage msg_Y(CHILD_ID_Y, V_TRIPPED);
MyMessage msg_B(CHILD_ID_B, V_TRIPPED);

#define RELAY_PIN_G 2 // LED VERDE
#define RELAY_PIN_R 3 // LED VERMELHO
#define RELAY_PIN_Y 4 // LED AMARELO
#define RELAY_PIN_B 5 // LED AZUL#define NUMBER_OF_RELAYS 4
// Total number of attached relays
#define RELAY_ON 1  // GPIO value to write to turn on attached relay
#define RELAY_OFF 0 // GPIO value to write to turn off attached relay

uint8_t value_G = RELAY_OFF;
uint8_t value_R = RELAY_OFF;
uint8_t value_Y = RELAY_OFF;
uint8_t value_B = RELAY_OFF;
int BT_S = 6; //botao sensor temperatura
int BT_T = 7; //botao trip
byte ST_LED_G = LOW;
byte ST_LED_Y = LOW;
byte ST_LED_B = LOW;
byte ST_LED_R = LOW;
byte ST_BT_S = 0;
byte ST_BT_T = 0;
byte ST_ALARM = 0;
void before()
```

```
{
//LED VERDE
  for (int sensor=1, pin=RELAY_PIN_G; sensor<=NUMBER_OF_RELAYS; sensor++, pin++)
{
    // Then set relay pins in output mode
    pinMode(pin, OUTPUT);
    // Set relay to last known state (using eeprom storage)
    digitalWrite(pin, loadState(sensor)?RELAY_ON:RELAY_OFF);
  }
 //LED VERMELHO
  for (int sensor=2, pin=RELAY_PIN_R; sensor<=NUMBER_OF_RELAYS; sensor++, pin++) {
    // Then set relay pins in output mode
    pinMode(pin, OUTPUT);
    // Set relay to last known state (using eeprom storage)
    digitalWrite(pin, loadState(sensor)?RELAY_ON:RELAY_OFF);
  }
  //LED AMARELO
  for (int sensor=3, pin=RELAY_PIN_Y; sensor<=NUMBER_OF_RELAYS; sensor++, pin++) {
    // Then set relay pins in output mode
    pinMode(pin, OUTPUT);
    // Set relay to last known state (using eeprom storage)
    digitalWrite(pin, loadState(sensor)?RELAY_ON:RELAY_OFF);
  }
 //LED AZUL
  for (int sensor=4, pin=RELAY_PIN_B; sensor<=NUMBER_OF_RELAYS; sensor++, pin++) {
    // Then set relay pins in output mode
    pinMode(pin, OUTPUT);
    // Set relay to last known state (using eeprom storage)
    digitalWrite(pin, loadState(sensor)?RELAY_ON:RELAY_OFF);
  }
MOTOR_OFF();
}
void setup()
{
pinMode(BT_S , INPUT);  // pinMode = INPUT vai receber o sinal do BT
pinMode(BT_T , INPUT);
}
void presentation()
{
 // Send the sketch version information to the gateway and Controller
  sendSketchInfo("TCC -  Inovador", "ANDRE E RONALDO");
// LED VERDE
  for (int sensor=1, pin=RELAY_PIN_G; sensor<=NUMBER_OF_RELAYS; sensor++, pin++)
{
    // Register all sensors to gw (they will be created as child devices)
    present(sensor, S_LIGHT);
    present(CHILD_ID_G, S_LIGHT);
  }

// LED VERMELHO
  for (int sensor=2, pin=RELAY_PIN_R; sensor<=NUMBER_OF_RELAYS; sensor++, pin++) {
    // Register all sensors to gw (they will be created as child devices)
    present(sensor, S_LIGHT);
    present(CHILD_ID_R, S_LIGHT);
  }
 // LED AMARELO
   for (int sensor=3, pin=RELAY_PIN_Y; sensor<=NUMBER_OF_RELAYS; sensor++, pin++)
{
    // Register all sensors to gw (they will be created as child devices)
    present(sensor, S_LIGHT);
```

```
    present(CHILD_ID_Y, S_LIGHT);
  }
  // LED AZUL
   for (int sensor=4, pin=RELAY_PIN_B; sensor<=NUMBER_OF_RELAYS; sensor++, pin++)
{
    // Register all sensors to gw (they will be created as child devices)
    present(sensor, S_LIGHT);
    present(CHILD_ID_B, S_LIGHT);
}
}
 void loop()
{
delay(50);
ST_BT_S = digitalRead(BT_S); //Realizo  leitura do botao
if (ST_BT_S == 1)
 {
  delay(50);
  TEMP_ALTA();
  delay(100);
 }

ST_BT_T = digitalRead(BT_T); //Realizo  leitura do botao
if (ST_BT_T == 1)
 {
  delay(50);
  TRIP();
  delay(100);
 }
}

// Função de inicio do motor
void MOTOR_OFF(){
  ST_ALARM = 0;
 digitalWrite(RELAY_PIN_G, 1);
 digitalWrite(RELAY_PIN_R, 0);
 value_R = 0;
 send(msg_R.set(value_R)); // envia mensagem para o domoticz para        delisgar o motor
digitalWrite(RELAY_PIN_Y, 0);
 value_Y = 0;
send(msg_Y.set(value_Y)); // envia mensagem para o domoticz para delisgaro sinal de temp
digitalWrite(RELAY_PIN_B, 0);
value_B = 0;
send(msg_B.set(value_B)); // envia mensagem para o domoticz para delisgar o sinal de trip

// Função de reconhecimento do alarme de temperatura
void TEMP_ALTA(){
    digitalWrite(RELAY_PIN_Y, 1); // ativa o led de temperatura
    ST_LED_Y = 1;
    value_Y = RELAY_ON;
    send(msg_Y.set(value_Y)); // envia mensagem para o domoticz o valor de temperatura
    delay(100);
    value_R = RELAY_OFF;
    send(msg_R.set(value_R)); // envia mensagem para o domoticz para delisgar o motor
    delay(100);
    digitalWrite(RELAY_PIN_R, 0);// desativa o led vermelho
    digitalWrite(RELAY_PIN_G, 1);// ativo o led verde
}
// Função de reconhecimento do alarme de trip
void TRIP(){
    digitalWrite(RELAY_PIN_B, 1); // ativa o led de temperatura
```

```
    ST_LED_B = 1;
    value_B = RELAY_ON;
    send(msg_B.set(value_B)); // envia mensagem para o domoticz o valor de trip
    delay(100);
    value_R = RELAY_OFF;
    send(msg_R.set(value_R)); // envia mensagem para o domoticz para delisgar o motor
    delay(100);
    digitalWrite(RELAY_PIN_R, 0); // desativa o led vermelho
    digitalWrite(RELAY_PIN_G, 1); // ativo o led verde
}
void MOTOR_ALARM(){
    delay(100);
    value_R = RELAY_OFF;
    send(msg_R.set(value_R)); // envia mensagem para o domoticz para delisgar o motor
    digitalWrite(RELAY_PIN_R, 0);// desativa o led vermelho
    digitalWrite(RELAY_PIN_G, 1);// ativo o led verde
    ST_ALARM = 0;
}
void receive(const MyMessage &message)
{
  // Atuação remota do domoticz
if (message.getType()==V_STATUS) {
    // Leitura do botao remoto

    // ----------------------------------
    // FUNCIONAMENTO DO MOTOR REMOTO
    // ----------------------------------
    if(message.getSensor() == 2){ // IDENTIFICAO DO BOTAO REMOTO

    if (ST_LED_Y == 1 or ST_LED_B == 1) //verifico a situacao do motor
    {
       ST_ALARM = 1;
    }
    else{
      // VERIFICAÇÃO DO SINAL DO BOTAO (SE = 1 OU 0)
     if( message.getBool()?RELAY_ON:RELAY_OFF == 1) {
        digitalWrite(RELAY_PIN_R,  message.getBool()?RELAY_ON:RELAY_OFF);  //  ATIVO
LED VERMELHO (MOTOR LIGADO)
        ST_LED_R = 1;
        value_R = 1;
        ST_LED_G = 0;
        digitalWrite(RELAY_PIN_G, 0); // DESATIVO LED VERDE
        value_G = 0;
      }
      else{
         ST_LED_R = 0;
         value_R = value_R == RELAY_ON ? RELAY_OFF: RELAY_ON; // DESATIVO LED
VERMELHO
         ST_LED_G = 1;
         digitalWrite(RELAY_PIN_G,  message.getBool()?RELAY_ON:RELAY_OFF);  //  ATIVO
LED VERDE (MOTOR DESLIGADO)
         value_G = value_G == RELAY_ON ? RELAY_OFF: RELAY_ON;
      }
    }
    }
 // ----------------------------------
    // FUNCIONAMENTO DO ALARME DE TEMPERATURA REMOTO E RECONHECI-
MENTO
    // ----------------------------------
 else if (message.getSensor() == 3){ // IDENTIFICAO DO BOTAO REMOTO
```

```
      if( message.getBool()?RELAY_ON:RELAY_OFF == 1) {
       TEMP_ALTA(); // BUSCO A FUNCAO DE TEMPERATURA
      }
      else{
        ST_LED_Y = 0;
        value_Y = value_Y == RELAY_ON ? RELAY_OFF: RELAY_ON;
        digitalWrite(RELAY_PIN_Y, message.getBool()?RELAY_ON:RELAY_OFF);   // DESA-
TIVO LED AMARELO
      }
    }

    // ------------------------------------
    // FUNCIONAMENTO DO ALARME DE TRIP REMOTO E RECONHECIMENTO
    // ------------------------------------

    else if (message.getSensor() == 4){ // IDENTIFICAO DO BOTAO REMOTO

      if( message.getBool()?RELAY_ON:RELAY_OFF == 1) {
       TRIP();  // BUSCO A FUNCAO DE TRIP
      }
      else{
        ST_LED_B = 0;
        value_B = value_B == RELAY_ON ? RELAY_OFF: RELAY_ON;
        digitalWrite(RELAY_PIN_B,  message.getBool()?RELAY_ON:RELAY_OFF);  //  DESA-
TIVO LED AZUL
      }
    }

    // Store state in eeprom
   if (ST_ALARM == 1){
   MOTOR_ALARM(); // Situação do motor em interlock de falha
   }
   else{
    saveState(message.getSensor(), message.getBool());
    // Write some debug info
    Serial.print("Incoming change for sensor:");
    Serial.print(message.getSensor());
    Serial.print(", New status: ");
    Serial.println(message.getBool());
   }
  }
}
```