Automação e Supervisão de Processos 1
Projeto 2
Monta Cargas (Fevereiro 2018)

Alexandre A. Trindade, Thiago de Morais R. Santos, Hugo L. C. Monteiro

***Resumo*—O projeto do Monta Cargas, consiste em simular um elevador de cinco andares, o qual possui somente painel externo de chamada para cada andar. Requisitos que fazem parte desta solução incluem sensor de fim de curso para indicar passagem por cada andar, sensor ótico para indicar se a porta do elevador está bloqueada e sensor de temperatura para indicar o momento de acionar a ventilação. O objetivo consiste em montar o sistema de controle no microcontrolador e através do protocolo de comunicação Modbus Serial, o supervisório ScadaBR foi escolhido para visualização gráfica do funcionamento do sistema e também envio de comandos. A metodologia seguida foi de planejar partes do projeto, verificar se o código e montagem está correta, e posteriormente testar a comunicação com o supervisório. A visualização feita no supervisório permite atestar que os problemas e detalhes anunciados neste projeto foram levados em consideração na solução proposta.**

***Palavras-chave*—Protocolo Modbus®, Arduino Mega 2560, CLP, ScadaBR, supervisório, monta cargas, elevador.**

***Abstract*—The Monta Cargas project consists of a five-floor elevator simulator, which has an external call panel for each floor. The requirements of this project include, end-of-travel sensor to indicate passage through each floor, optical sensor to indicate when the elevator door is blocked and temperature sensor to indicate when to turn on the ventilation. The purpose is to configure the control system with microcontroller and through the Modbus Serial communication protocol, the ScadaBR supervisor was chosen to graphically display the system operation and also send commands. The methodology followed consists in designing separate parts of this project, verifying that the code and assembly are correct, and then communicating with the supervisor. The research done with ScadaBR enables through the graphical representation check that all the specifications and demands were taken into account in the proposed solution.**

***Index Terms*—Modbus® Protocol, Arduino Mega 2560, PLC (Programmable Logic Controller), ScadaBR, supervisory, monta cargas, elevator.**

I. Introdução

ESTE projeto parte do tema de comparação de plataformas, o Controlador Lógico Programável (CLP) e microcontrolador. O CLP é mais robusto e mais caro, utilizado em aplicações práticas finais, o microcontrolador é mais acessível, permite simular sistemas de investimento alto, não tem finalidade de ser utilizado em aplicações práticas finais para usuários. O problema enunciado: elevador "monta-cargas" possui somente painel externo de chamada, sensor de fim de curso indica a passagem por cada andar; ao abrir a porta do elevador, o tempo de espera é de 10 segundos, o sensor ótico indica se há bloqueio, e não permite que a porta se feche bloqueada, ao identificar um objeto, o tempo de espera da porta do elevador para ser fechada é reiniciado. A temperatura deve ser monitorada em °C, o usuário manipula o valor que ativa a ventilação. Quando parado em um dos cinco andares, o botão do andar correspondente abri o elevador. Primeiramente foi instalado e configurado o supervisório ScadaBR (Aquisição de Dados e Controle do Supervisório), *software*-livre, gratuito e de código fonte aberto. Seu funcionamento depende de outros dois programas instalados Máquina Virtual Java (JVM) e o servidor web Apache Tomcat®. O protocolo de comunicação adotado foi o Modbus® Serial (estrutura de troca de mensagens usada para comunicação tipo mestre/escravos ou cliente/servidor entre dispositivos inteligentes) [1].

A plataforma utilizada para montagem, Arduino Mega 2560, é característica geral de microcontroladores a facilidade de se cometer erros de montagem, e na programação, tendo em vista esta perspectiva, as funcionalidades atribuídas ao elevador foram divididas em etapas e implementadas sequencialmente. De maneira que as dificuldades encontradas tinham seu campo de erro dentro da etapa em fase de teste.

A maioria das variáveis envolvidas são binárias, como estado da porta, bloqueio da passagem do elevador, os botões de chamada, o sensor de fim de curso, etc. O valor de cada uma das variáveis citadas anteriormente assume portanto os valores "0" ou "1", analisar e monitorar uma quantidade pequena de variáveis somente pelo valor numérico ou booleano ("TRUE", "FALSE") não constitui dificuldade, porém, quando o número de variáveis se torna expressivo, fica confuso e difícil interpretar cada uma delas. Neste projeto são mais de vinte registradores, a vantagem na utilização do supervisório torna-se notável, tanto no processo de encontrar os erros específicos no circuito montado, e principalmente pelos recursos de representação gráfica, que incluem *gifs*, imagens, gráficos, botões entre outros recursos.

Os resultados e avaliação dos mesmos foram testados via supervisório, utilizamos a representação gráfica do ScadaBR como *output*, e meio o qual avaliamos cada mudança feita, até atingir o exato movimento ou ação requerida na ficha projeto.

Artigo recebido 05 de março de 2018; revisado 05 de março de 2018; aceito 05 de março de 2018. Data da publicação 05 de março de 2018; data da versão atual 05 de março de 2018.

Alexandre A. Trindade, Thiago de Morais R. Santos, e Hugo L. C. Monteiro são alunos atualmente do departamento IV, Coordenação de Engenharia de Controle e Automação, Instituto Federal de Goiás, Câmpus Goiânia, Rua 75, nº 46, Centro, Goiânia-GO. CEP: 74055-110, Brasil. (e-mail: alexandre.alves.trindade@gmail.com).

II. Fundamentação Teórica

Elevadores do tipo monta carga, como da Fig. 1, proporcionam agilidade no transporte, atendem edifícios comerciais e residenciais, é encontrado no mercado com capacidade média de 10 a 300kg. Comércios que utilizam desses elevadores incluem restaurantes, açougue, supermercados, bares e lancherias, escritórios, casas de alimento. As empresas vendem os elevadores, citando as vantagens de sua utilização como rapidez na montagem e instalação; baixo consumo de energia; baixo nível de ruído; dispensa casa de máquinas e poço; praticidade e segurança [2], [3],[4].

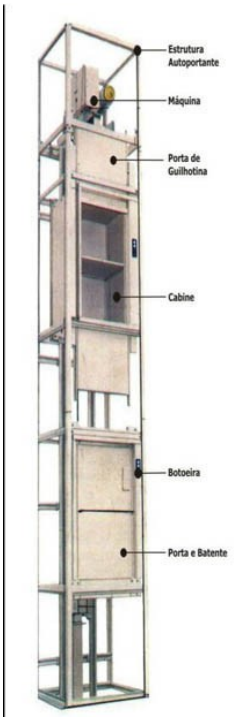


Fig. 1. Elevador monta-cargas [5].

A. Arduino Mega 2560

O microcontrolador Arduino, Fig. 2, foi criado em 2005 com o intuito de interagir em projetos escolares de forma a ter orçamento menor que outros sistemas de prototipagem disponíveis naquela época. Hoje é um dos mais conhecidos e utilizados, sua placa consiste em microcontrolador Atmel AVR de 8 bits. Possui 54 pinos entradas/saídas digitais – 14 podem usados como saída PWM. 16 entradas analógicas, 4 UARTs – portas seriais de *hardware* – oscilador de cristal de 16MHz, conexão USB, uma entrada de alimentação, uma conexão ICSP e um botão de *reset* [6],[7],[8].

O ATmega2560, Fig. 3, fornece quatro portas de comunicação serial UARTs para TTL (5V). Um chip FTDI FT232RL direciona uma destas portas para a conexão USB e os drivers FTDI (que acompanham o *software* do Arduino) fornecem uma porta com virtual para *softwares* no computador.

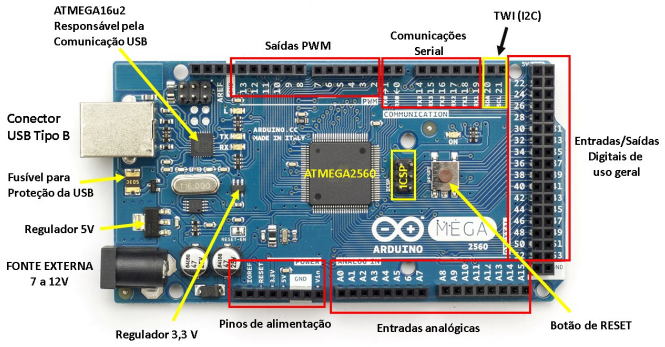


Fig. 2. Arduino Mega 2560 [9].

Fig. 3. ATmega2560 [9].

A alimentação da placa Arduino Mega 2560, pode ser feita tanto pela USB, como por uma alimentação externa como ilustrado na Fig. 4. Servindo como interface USB para a comunicação com o computador tem-se o microcontrolador ATMEL ATMEGA16U2, Fig. 5. O envio e recepção de dados da placa para o computador são indicados pelos *leds* TX e RX controlados pelo software do ATMEL ATMEGA16U2 [9].

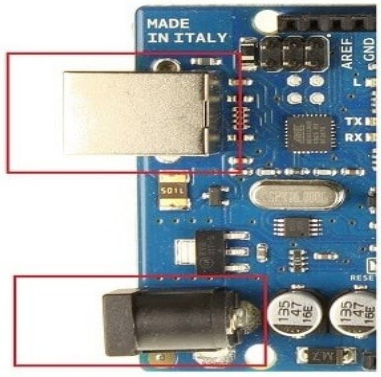


Fig. 4. Alimentação do Arduino Mega 2560 [9].

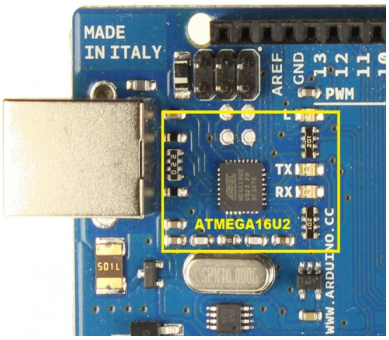


Fig. 5. ATMEL ATMEGA16U2 [9].

TABELA I
RESUMO DO MICROCONTROLADOR [8]

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

B. ScadaBR

SCADA - Aquisição de Dados e Controle do Supervisório – sigla bastante explicativa da utilidade do ScadaBR, a parte de aquisição de dados, de uma planta ou sistema controlado envia as variáveis – temperatura, pressão, etc – para o ScadaBR via protocolo de comunicação, as variáveis são os registradores, o supervisório os salva como *data points.* Desempenha também função de controle, envia comandos diretamente para o sistema controlado.

Este supervisório tem boa parte de sua aplicação nas indústrias, sendo aplicado com CLPs. O objetivo principal do ScadaBR é propiciar interface de alto nível ao operador, em tempo real, este sistema é configurável, e por ser gratuito adotar um grande número de protocolos de comunicação - evidenciados na Fig. 6 - ser confiável e manuseio e utilização ser considerada acessível em seu nível de dificuldade, torna o SCADA extremamente popular.

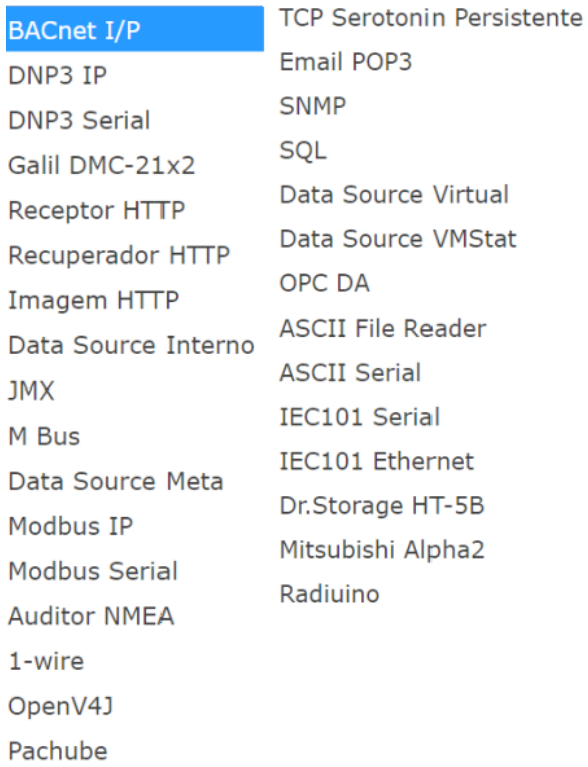


Fig. 6. Protocolos de comunicação adotados pelo ScadaBR.

C. Modbus Serial

O protocolo Modbus foi criado na década de 1970 pela Modicon, para comunicação de dispositivos mestre/escravo e cliente/servidor. A Modicon foi adquirida pela Schneider, os direitos sobre o protocolo foram liberados pela Oganização Modbus. Sua utilzação é feita principalmente em instrumentos e equipamentos de laboratório; automação residencial e automação de navios. Sua implementação é considerada simples, pode ser utilizado com os seguintes padrões de meio físico: RS232, RS485, Ethernet TCP/IP (Modbus TCP) [10].

Na Fig. 7 o *gateway* estabelece comunicação entre dois tipos de Modbus, o serial em RS485 e o TCP/IP em *ethernet*. Neste caso o mestre é o CLP, envia e recebe dados dos escravos – inversor de frequência, Interface Homem Máquina (IHM), controlador de temperatura, interface I/O remota Modbus [10].

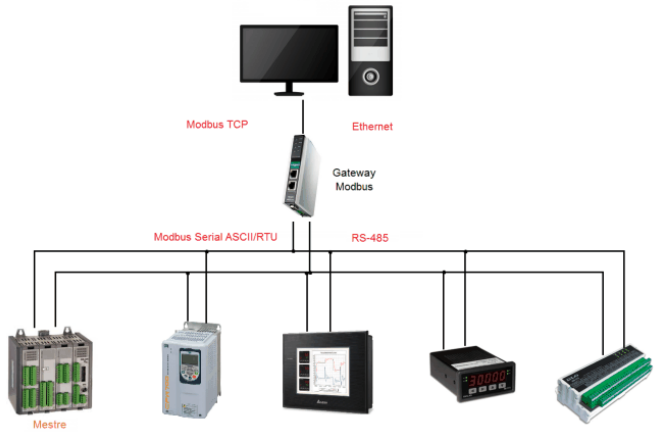


Fig. 7. Exemplo de rede com protocolo Modbus [10].

O padrão Modbus adota dois métodos de transmissão, o ASCII (*American Standard Code for Information*) Mode e o RTU Mode. Conforme o método selecionado os parâmetros da porta de comunicação serial são modificados como *baud rate*, paridade, etc. No modo de transmissão ASCII cada byte de caractere, é enviado dois caracteres sem geração de erros, no RTU (*Remote Terminal Unit*) cada mensagem de 8 bits contém dois caracteres hexadecimais de 4 bits [11].

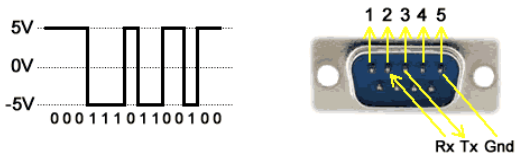


Fig. 8. Representação envio de dados protocolo Modbus [11].

Os dados são enviados em série de zero e um, os bits, os zeros são enviados no valor de tensão positivo e um valor negativo, esquema da Fig. 8. Transmissão típica é de 9600 *baud rate* (bits por segundo) [11].

TABELA II
FUNÇÕES NO PROTOCOLO MODBUS [10]

Código da função	Descrição
1	Leitura de bloco de bits do tipo coil(saída discreta).
2	Leitura de bloco de bits do tipo entradas discretas.
3	Leitura de bloco de registradores do tipo holding.
4	Leitura de bloco de registradores do tipo input.
5	Escrita em um único bit do tipo coil(saída discreta).
6	Escrita em um único registrador do tipo holding.
7	Ler o conteúdo de 8 estados de exceção.
8	Prover uma série de testes para verificação da comunicação e erros internos.
11	Obter o contador de eventos.
12	Obter um relatório de eventos.
15	Escrita em bloco de bits do tipo coil(saída discreta).
16	Escrita em bloco de registradores do tipo holding.
17	Ler algumas informações do dispositivo.
20	Ler informações de um arquivo.
21	Escrever informações em um arquivo.
22	Modificar o conteúdo de registradores de espera através de operações lógicas.
23	Combina ler e escrever em registradores numa única transação.
24	Ler o conteúdo da fila FIFO de registradores.
43	Identificação do modelo do dispositivo.

No protocolo Modbus o mestre especifica o tipo de função ou serviço ao escravo (leitura, escrita, etc). Cada função é utilizada para acessar tipo diferente de dado, como na Tabela 2.

D. Sensores

O sensor de temperatura empregado neste projeto é o LM35. Este sensor é um circuito integrado de precisão que apresenta uma saída de tensão linear proporcional à temperatura em que ele se encontrar no momento, tendo em sua saída um sinal de 10mV para cada Grau Célsius de temperatura. Não é necessário calibração externa para que o sensor trabalhe com exatidão dentro da faixa de temperatura entre –55 ºC e 150 ºC. Ele pode ser usado de duas formas, com alimentação simples ou simétrica, dependendo do que se desejar como sinal de saída, mas independentemente disso, a saída continuará sendo de 10mV/ºC, em cada uma dessas duas formas de alimentação a temperatura máxima e mínima medida com exatidão, é diferente. A Fig. 9 mostra o sensor LM35, Fig. 10 e Fig. 11 mostram o esquema de modo básico e escala completa de leitura de temperatura, em (1) o código para conversão do valor analógico para graus Celsius [12].

$$\frac{5 * \left(analogRead\left(SENSOR_{TEMP}\right)\right) * 100}{1024} \quad (1)$$

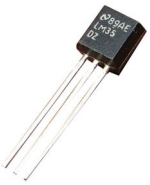


Fig. 9. Sensor de temperatura LM35 [12].

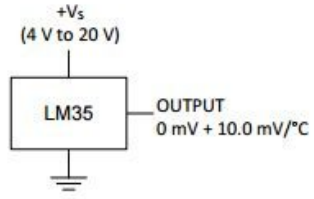


Fig. 10. Modo básico – (2ºC a 150ºC) [13].

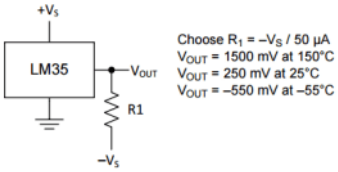


Fig. 11. Modo escala completa – (-55ºC a 150ºC) [13].

O sensor ótico utilizado para impedir fechamento da porta do elevador quando está bloqueada é o TCRT5000, visto na Fig. 12, sensor reflexivo que inclui emissor infravermelho e fototransistor em pacote com chumbo que bloqueia a luz visível. O emissor infravermelho (cor azul) e o transistor IR (fototransistor – cor preta), são separados por uma barreira. Quando algum objeto se aproxima do sensor, parte da luz emitida pelo emissor é refletida pelo objeto para o fototransistor. A distância de detecção dependerá do coeficiente de reflexão do objeto a ser detectado, o esquema elétrico e funcionamento são representados na Fig. 13 [14].

Fig. 12. TCRT5000 [14].

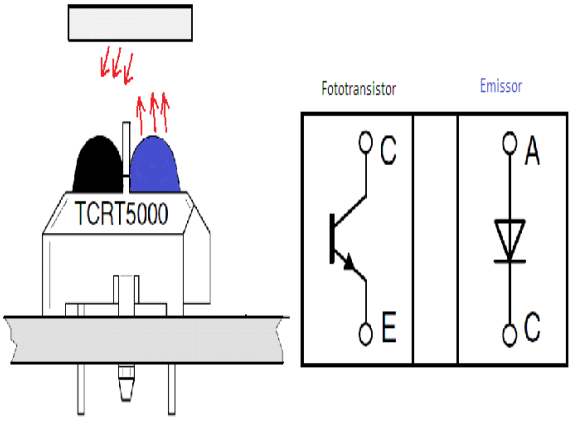


Fig. 13. Esquema elétrico e funcionamento [15].

III. Instalação ScadaBR

Para instalar o ScadaBR no sistema operacional Windows, os seguintes passos foram seguidos:

1) Instalar Java Virtual Machine (versão 1.6 ou superior);
2) Instalar TomCat versão 7;
3) Instalar o ScadaBR;
4) Comunicação com o Arduino.

Atenção é requerida em cada um destes passos, instalação feita de modo inapropriado possivelmente resultará em erros de funcionamento do supervisório. Deste modo deve ser verificado se o PC é de 32 ou 64 bits; o servidor TomCat é recomendado ser instalado como administrador do sistema; a porta de conexão HTTP foi definida como 8085 [1].

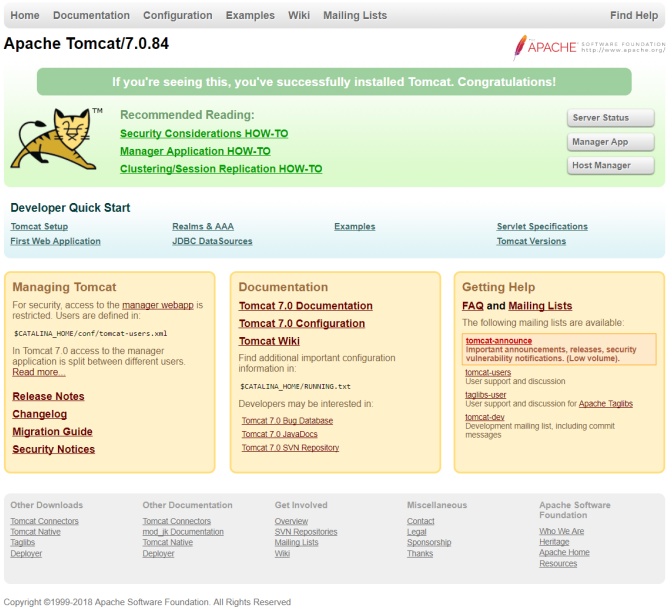


Fig. 14. Página inicial localhost:8085.

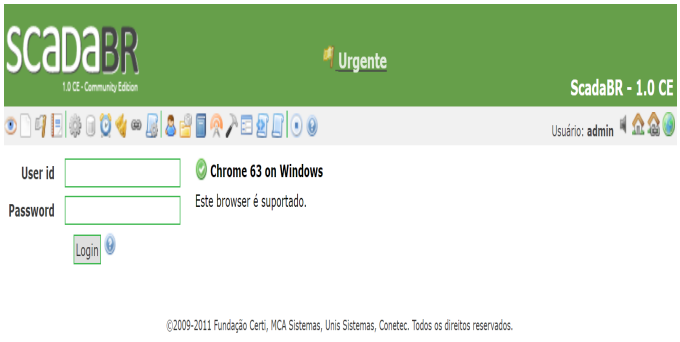


Fig. 15. ScadaBR localhost:8085/ScadaBR.

Como verificação se o ScadaBR foi instalado e configurado corretamente, com o servidor TomCat ligado, inicia-se um dos navegadores de internet disponíveis e ao enviar na barra de endereço *localhost:8085* se o Apache TomCat tiver sido instalado e configurado certo a Fig. 14 é mostrada no navegador, em seguida é enviado *localhost:8085/ScadaBR,* e a Fig. 15 é mostrada, basta agora iniciar sessão digitando *login* e senha.

IV. Teste de Comunicação com o Arduino

Para este projeto o modo de comunicação foi o Modbus RTU, e a biblioteca deste protocolo deve ser carregada no Arduino para funcionamento do ScadaBR. Primeiramente criamos o *data source* (LM35) como representado na Fig. 16 com o protocolo Modbus Serial.

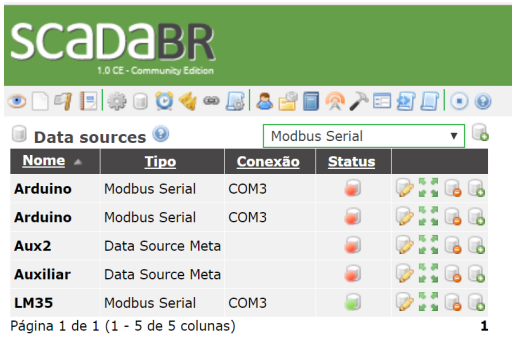


Fig. 16. Data sources.

As variáveis a serem monitoradas ou controladas pelo ScadaBR são definidas como registradores no código fonte, e são criadas uma a uma como *data points* no supervisório. Na Fig. são 25 registradores criados e monitorados neste projeto. O ANDAR_ATUAL recebe cada andar o qual o elevador está posicionado; os botões são o painel externo de cada andar; CHAMADA foi utilizada para registrar de maneira fixa no painel qual foi o andar requisitado; *door* recebe a saída do sensor ótico; FIMDECURSO foi utilizado como método de controle da abertura e fechamento da porta do elevador; *mais* e *menos* incrementam ou decrementam o *setpoint* da temperatura por 1 °C; RELE_A é a saída do estado do relé; TEMP123 é o sensor de temperatura e VENTILADOR desliga ou desliga a ventilação conforme a temperatura está acima ou abaixo do *setpoint.*

Data points

Nome	Tipo de dado	Status	Escravo	Faixa	Offset (baseado em 0)
AA	Numérico		1	Registrador holding	13
ANDAR_ATUAL	Numérico		1	Registrador holding	12
BOTAO1	Binário		1	Registrador holding	2/0
BOTAO2	Binário		1	Registrador holding	3/0
BOTAO3	Binário		1	Registrador holding	4/0
BOTAO4	Binário		1	Registrador holding	5/0
BOTAO5	Binário		1	Registrador holding	6/0
BOTAO6	Binário		1	Registrador holding	16/0
CHAMADA1	Binário		1	Registrador holding	7/0
CHAMADA2	Binário		1	Registrador holding	8/0
CHAMADA3	Binário		1	Registrador holding	9/0
CHAMADA4	Binário		1	Registrador holding	10/0
CHAMADA5	Binário		1	Registrador holding	11/0
door	Binário		1	Registrador holding	14/0
FIMDECURSO1	Binário		1	Registrador holding	18/0
FIMDECURSO2	Binário		1	Registrador holding	19/0
FIMDECURSO3	Binário		1	Registrador holding	20/0
FIMDECURSO4	Binário		1	Registrador holding	21/0
FIMDECURSO5	Binário		1	Registrador holding	22/0
mais	Binário		1	Registrador holding	24/0
menos	Binário		1	Registrador holding	25/0
porta	Binário		1	Registrador holding	15/0
RELE_A	Binário		1	Registrador holding	17/0
setpoint	Numérico		1	Registrador holding	23
TEMP123	Numérico		1	Registrador holding	0
VENTILADOR	Binário		1	Registrador holding	1/0

Fig. 17. Data points.

Watch list — lm35list

LM35 - TEMP123	36.13ºC	18:59:06
LM35 - VENTILADOR	1	18:59:06
LM35 - BOTAO1	0	18:59:06
LM35 - BOTAO2	0	18:59:06
LM35 - BOTAO3	0	18:59:06
LM35 - BOTAO4	0	18:59:06
LM35 - BOTAO5	0	18:59:06
LM35 - CHAMADA1	1	18:59:06
LM35 - CHAMADA2	0	18:59:06
LM35 - CHAMADA3	0	18:59:06
LM35 - CHAMADA4	0	18:59:06
LM35 - CHAMADA5	0	18:59:06
LM35 - ANDAR_ATUAL	1.0	18:59:06
LM35 - AA	0.0	18:59:06
LM35 - door	1	18:59:06
LM35 - porta	1	18:59:06

Fig. 18. Watch list.

Nas Fig. 16 e Fig. 17 o *data source* LM35 e os *data points* estão ativados, na Fig. 18 *Watch list* mostra o valor atual de cada registrador, os tipos de registradores neste projeto são binários ou numéricos. A temperatura indicada pelo sensor de temperatura foi de 36,13 °C.

V. Montagem

A Fig. 19 mostra como os sensores, botões e o relé foram montados no *protoboard* e suas conexões com o Arduino Mega 2560. A lista de materiais utilizados:

1) 6 botoeiras;
2) 1 sensor ótico TCRT5000;
3) 1 sensor de temperatura LM35;
4) 3 resistores de 10kΩ;
5) 4 resistores de 1kΩ;
6) 1 resistor de 470Ω;
7) 1 Arduino Mega 2560;
8) 1 cabo USB para Arduino;
9) *Jumpers;*
10) 1 módulo relé de um canal.

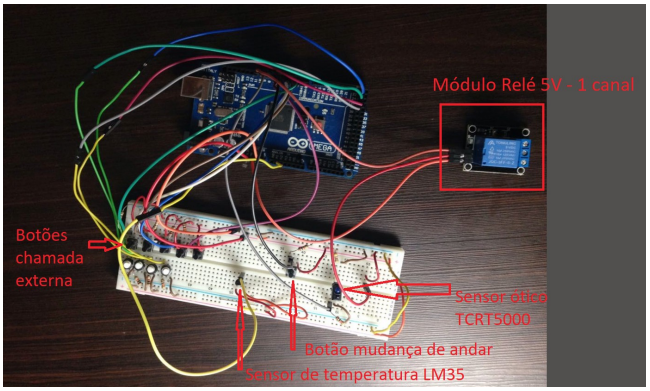


Fig. 19. Montagem do projeto.

VI. Fluxograma Código Fonte

O fluxograma visto na Fig. 20 auxilia a compreensão da programação que foi feita neste projeto. Primeiramente são declaradas as variáveis e logo em seguida os registradores que serão utilizados. Os pinos são definidos como *input* ou *output*, para leitura das variáveis neste processo os registradores recebem os sinais de saída dos botões e sensores. Se a porta estiver fechada e algum botão de um painel externo for pressionado, o LED no painel de chamada é aceso, as variáveis de controle da porta do elevador são modificadas. Resumidamente, essas variáveis citadas permitem que ao chegar ao andar de destino a porta abra uma vez, como o código é em *loop*, as variáveis de controle da abertura da porta tem a função de ativar ou desativar os parâmetros do *if* com o objetivo de evitar erros, e aberturas em andares não requisitados antes da chegada ao andar destino.

Se o andar de destino for maior que o andar atual, ao ser pressionado o botão de mudança de andar, a variável andar atual é incrementada em um, se não, é decrementada em um. Essa simulação é de um sensor fim de curso utilizada em elevadores. Se o elevador chega ao seu destino, a porta é aberta por 10s, como especificado, neste tempo se houver bloqueio no sensor ótico o tempo de abertura é *resetado,* ao detectar ausência de objeto realiza nova contagem de 10s até fechar a porta.

Em seguida declaramos o registrador que recebe o valor analógico da saída do sensor ótico, e é verificado se a temperatura está acima ou abaixo do *setpoint,* o que acionará ou desligar o relé, responsável pelo sistema de ventilação. O supervisório contém dois botões *mais* e *menos*, são definidos no final do *loop*, tem a função de modificar o *setpoint* de temperatura, o programa então retorna para verificar se há nova chamada no painel externo.

Fig. 20. Fluxograma da programação.

VII. RESULTADOS

O recurso de representação do supervisório foi utilizado para demonstrar o funcionamento do algoritmo implementado. Na Fig. 21 o painel de chamada mostra que o andar 3 foi requisitado o elevador. A Fig. 22 mostra o início da movimentação do elevador que passa pelo andar 2 logo em seguida Fig. 23 e em seguida chega ao seu destino Fig. 24. Na Fig. 24 a porta é aberta por 10s, e então é fechada, no período em que a porta é aberta, foi adotado neste projeto que outros comandos de chamada são bloqueados neste tempo. A Fig. 25 mostra o LED de luz vermelha quando a porta é bloqueado, ao pressionar o botão de aumento do *setpoint,* o novo valor é mostrado no supervisório, Fig. 26.

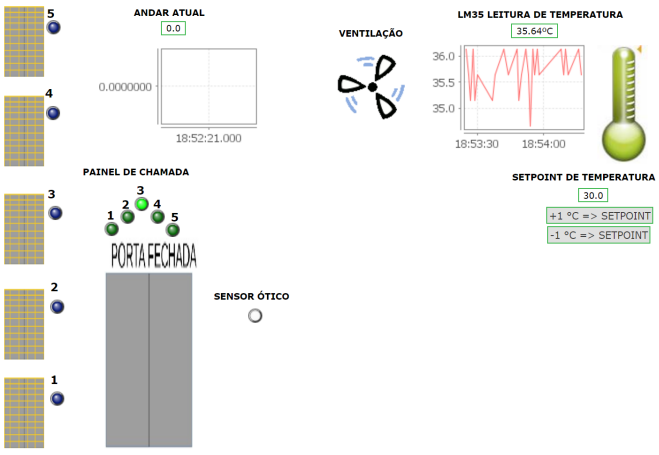


Fig. 21. Início da simulação, o andar 3 é requisitado no painel de chamada.

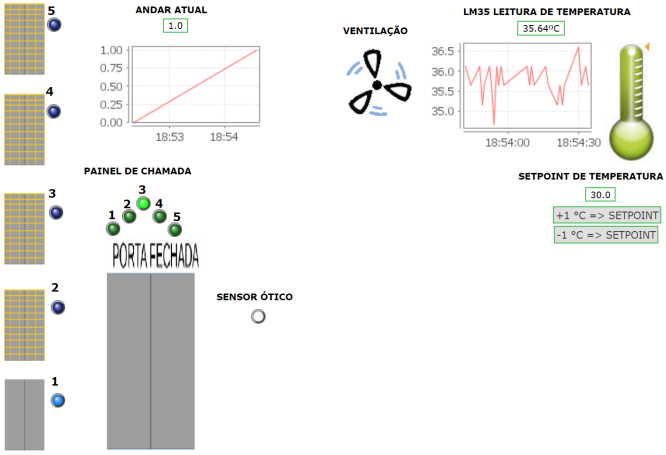


Fig. 22. Início da movimentação do elevador, passa no primeiro andar.

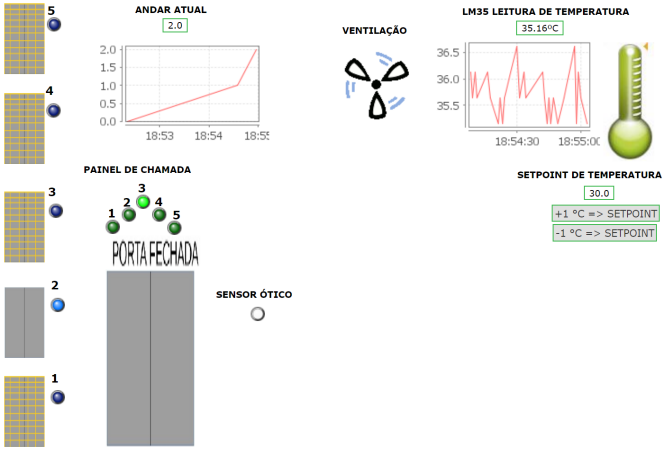


Fig. 23. Passa em seguida pelo segundo andar.

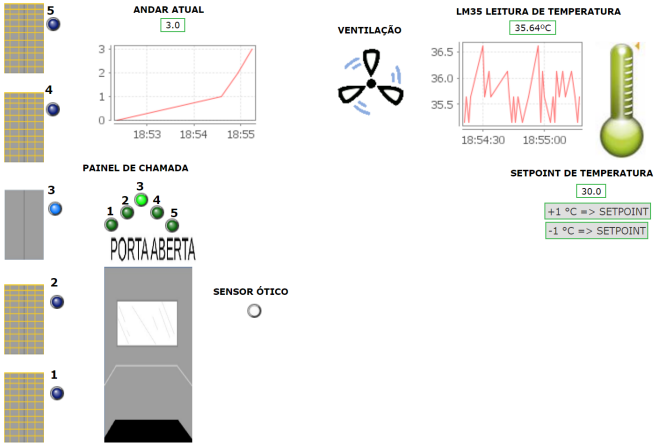


Fig. 24. Ao chegar no andar de destino a porta é aberta.

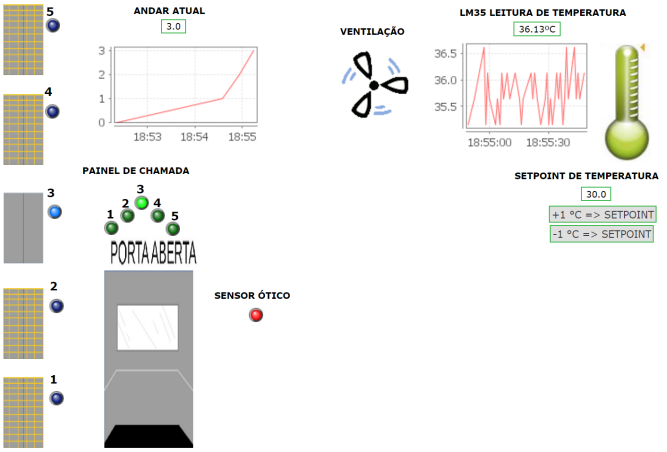


Fig. 25. Led sinaliza que a porta está bloqueada.

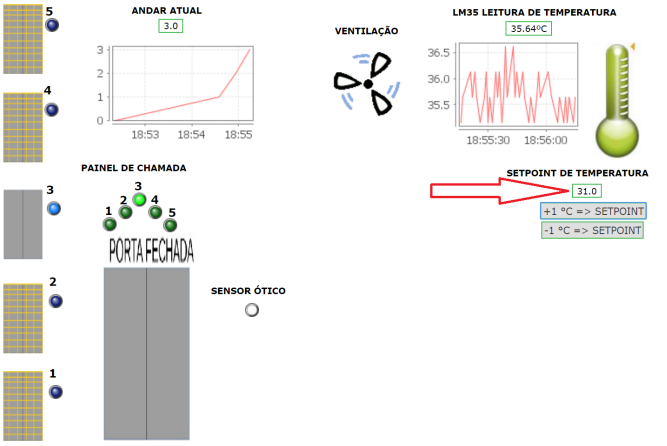


Fig. 26. Botão muda o *setpoint* da temperatura para 31ºC.

VIII. Conclusão

Na realização deste projeto, foi notado, contextualizado na comparação com o mesmo desafio no CLP, a programação em Ladder, implementada em forma de lógica sequencial e programação em blocos é notadamente mais fácil do que programação em linguagem C para o Arduino, linguagem de programação de alto nível, para o mesmo desafio, houve quantidade superior de *bugs* foi necessário mais tempo para propor soluções para cada etapa a ser implementada, correções até finalmente obter êxito. O aspecto custo, o projeto com o Arduino é extremamente barato, já os equipamentos para este projeto utilizando o CLP são mais caros, e é necessário maior cuidado com a segurança das pessoas envolvidas e também para não danificar os componentes, a tensão de trabalho de 220VCA em comparação com 5VCC no projeto atual.

Neste projeto foi prezado a qualidade da solução proposta para cada detalhe do problema proposto, obteve-se êxito na movimentação do elevador, obedecendo ao comando fim de curso que controla cada etapa até chegar ao destino; a porta abre por 10 segundos e então é fechada, ao pressionar o botão do andar atual é novamente aberta, e se houver bloqueio o tempo é *resetado*; foi utilizado um relé na montagem indicando ativação da ventilação, o *setpoint* é controlado via supervisório e os gráficos da movimentação do elevador e variação da temperatura são mostrados.

Portanto, citando todas essas etapas que foram implementadas separadamente, fazem parte de única solução demonstrada neste artigo e por meio de teste utilizando o ScadaBR comprova sucesso na implementação deste projeto.

Referências

[1] E. P. Vasconcelos Filho, "Introdução aos Supervisórios ScadaBR". [Online] Disponível: https://academicoweb.ifg.edu.br/uploads/MATERIAIS_AULAS/200550-ASP_1_-_2017.2_-_Avalia%C3%A7%C3%A3o_3_-_Projeto_02.pdf, acesso 05 Mar. 2018.

[2] *Monta-Carga (até 300kg). [Online] Disponível: http://www.adelevadores.com.br/produtos/monta-carga/, acesso 05 Mar. 2018.*

[3] *Monta-Cargas. [Online] Disponível: http://www.thyssenkruppelevadores.com.uy/pt-BR/produto/monta-cargas/, acesso 05 Mar. 2018.*

[4] *Elevador monta carga. [Online] Disponível: http://www.engetax.com.br/produtos/monta-carga/, acesso 05 Mar. 2018.*

[5] *[Online] Disponível: http://www.adelevadores.com.br/wp-content/gallery/monta/imadedd.jpg, acesso 5 Mar. 2018.*

[6] *O que é Arduino? [Online] Disponível: https://www.filipeflop.com/blog/o-que-e-arduino/, acesso 5 Mar. 2018.*

[7] *Arduino Mega2560 R3 [Online] Disponível: https://multilogica-shop.com/arduino-mega2560-r3, acesso 5 Mar. 2018.*

[8] *Arduino Mega 2560 Datasheet [Online] Disponível: https://www.robotshop.com/media/files/pdf/arduinomega2560datasheet.pdf, acesso 5 Mar. 2018.*

[9] *Arduino Mega 2560 [Online] Disponível: https://www.embarcados.com.br/arduino-mega-2560/, acesso 5 Mar. 2018.*

[10] *Protocolo Modbus: Fundamentos e Aplicações [Online] Disponível: https://www.embarcados.com.br/protocolo-modbus/, acesso 5 Mar. 2018.*

[11] *Modbus [Online] Disponível: https://www.citisystems.com.br/modbus/, acesso 5 Mar. 2018.*

[12] *LM35 [Online] Disponível: https://i2.wp.com, acesso 5 Mar. 2018.*

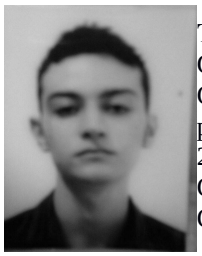
[13] *TEXAS INSTRUMENTS [Online] Disponível: http://www.ti.com/, acesso 5 Mar. 2018.*

[14] *[Online] Disponível: https://uge-one.com, acesso 5 Mar. 2018.*

[15] *[Online] Disponível: https://www.vishay.com/, acesso 5 Mar. 2018.*

Alexandre A. Trindade nasceu em Brasília – DF em 1994. Graduou do ensino médio em 2012 pelo Colégio da Polícia Militar de Goiás – Hugo de Carvalho Ramos. No mesmo ano iniciou o bacharelado em Engenharia de Controle e Automação no Instituto Federal de Goiás Câmpus Goiânia. Nos anos de 2013 a 2014 participou do programa de pesquisa Jovens Talentos para Ciência da CAPES. Contribuiu para desenvolvimento de aparelho de medição do tempo de resposta de indivíduos, coleta e análise de dados. No período 2014 a 2015 foi aluno do Ciências Sem Fronteiras na Universidade de Nevada, Reno realizou estágio voluntário no UNR Robotics Research Lab. Contribuiu em projetos com o robô PR2, utilizando ROS (*Robot Operating System*) e BAXTER. Durante os anos 2016 e 2017 foi bolsista de Iniciação Científica, na área de Engenharia Elétrica, Desenvolvimento de Metodologia para Identificação de Sistemas.

Hugo L. C. Monteiro nasceu em Goiânia – Go em 1990, graduou do ensino médio em 2008 pelo Colégio Einstein em Goiânia. Trabalhou de forma CLT, 3 anos na área de TI pela empresa High Tech e em 2011 ingressou na faculdade de Engenharia de Controle e Automação, pelo Instituto Federal de Goiás. Em 2012 foi admitido pela empresa Leitura MegaStore como consultor de informática durante 3 anos, no ano seguinte foi contratado pelo Itáu/Rede como Consultor de Negócios duração de 8 meses.



Thiago de Morais R. Santos nasceu em Goiânia GO em 1991. Em 1996 se mudou para Senador Canedo onde formou no ensino médio em 2008 pelo Colégio Estadual Pedro Xavier Teixeira. Em 2012 iniciou o Bacharelado em Engenharia de Controle e Automação no Instituto Federal de Ciência e Tecnologia de Goiás.