UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Matheus Chaves Rocha

Uso dos protocolos MQTT e S7 para o aperfeiçoamento da automação na indústria de sacolas plásticas

Florianópolis
2022

Matheus Chaves Rocha

Uso dos protocolos MQTT e S7 para o aperfeiçoamento da automação na indústria de sacolas plásticas

Relatório final da disciplina DAS5511 (Projeto de Fim de Curso) como Trabalho de Conclusão do Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Santa Catarina em Florianópolis.
Orientador: Prof. Marcelo De Lellis Costa De Oliveira, Dr.
Supervisor: Guilherme Cornelli, Eng.

Florianópolis
2022

Ficha de identificação da obra

A ficha de identificação é elaborada pelo próprio autor.
Orientações em:
http://portalbu.ufsc.br/ficha

Matheus Chaves Rocha

Uso dos protocolos MQTT e S7 para o aperfeiçoamento da automação na indústria de sacolas plásticas

Esta monografia foi julgada no contexto da disciplina DAS5511 (Projeto de Fim de Curso) e aprovada em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação

Florianópolis, 12 de dezembro de 2022.

Prof. Hector Bessa Silveira, Dr.
Coordenador do Curso

Banca Examinadora:

Prof. Marcelo De Lellis Costa de Oliveira, Dr.
Orientador
UFSC/CTC/DAS

Guilherme Cornelli, Eng.
Supervisor
GreyLogix Brasil

Prof. Rodrigo Lange, Dr.
Avaliador
IFRS

Prof. Eduardo Camponogara, Dr.
Presidente da Banca
UFSC/CTC/DAS

Este trabalho é dedicado aos meus amigos e família.
Em especial ao meu pai, Rubens Rocha, e à minha mãe,
Regina Rocha, que me acompanharam e deram suporte
em todas as etapas dessa jornada, tanto nas boas
quanto nas ruins.

AGRADECIMENTOS

Primeiramente, gostaria de agradecer à Universidade Federal de Santa Catarina por me proporcionar um ensino de qualidade e diversas experiências que levarei para a vida toda. Gostaria também de agradecer à Greylogix Brasil por ter acreditado em mim e me dado a oportunidade de desenvolver um produto do início ao fim. Um agradecimento especial ao Renato Leal por me inserir no time e confiar no meu potencial desde o começo. Agradeço também ao Rafael Gonçalves pelo apoio desde o início do desenvolvimento da UPC e, por último, gostaria de agradecer ao Guilherme Cornelli, meu atual supervisor, por todo o suporte e confiança nessa jornada.

RESUMO

O processo para a produção de sacolas plásticas não evoluiu de maneira significativa desde a chegada ao Brasil, em 1949. Tendo três áreas principais: impressão, bobinadeira e corte e solda, o processo de produção de sacolas plásticas acaba sendo demorado devido aos apontamentos manuais feitos pelos operadores. Durante a produção, o operador deve preencher a quantidade produzida, os tempos de paradas que as máquinas tiveram e as perdas. Isso acaba sendo impreciso, com pouca confiabilidade e aberta a erros humanos. A GreyLogix Brasil juntamente com a Projedata, desenvolveram um produto para que ocorra a automatização dessas três áreas e a redução de erros na produção. Surgiu, assim, a unidade de controle e processamento de dados (UPC) que se comunica com o ERP da Projedata, o Autoflex. A UPC consiste em um CLP e um *gateway* IOT, ambos Siemens. Para haver comunicação entre os equipamentos da UPC é utilizado o protocolo S7, sendo um protocolo nativo da Siemens. Para haver comunicação entre a UPC e o Autoflex é utilizado o MQTT, pois consome poucos dados e é possível gerar tópicos em paralelo sem que haja conflito entre eles. As principais vantagens do produto são a contagem automática da produção, apontamento de paradas da máquina, apontamento de *setups* da máquina, rastreabilidade nas etiquetas geradas, configurações rápidas de máquinas na UPC e redução no tempo em que os relatórios de produção são enviados para o supervisor, sendo que, antes do produto, levavam em torno de 24 horas e, com a UPC, é possível ver em tempo real no Autoflex. Com isso, o produto conseguiu gerar rastreabilidade de todas as áreas e criou confiabilidade nos dados.

Palavras-chave: MQTT. S7. IOT. CLP LOGO.

ABSTRACT

The process of producing plastic bags has not evolved significantly since arriving in Brazil in 1949. Having three main areas: printing, winder and cutting and welding, the plastic bag production process ends up being time consuming due to the manual reports made by operators. During production, the operator must fill in the quantity produced, the downtimes that the machines had and the losses. This turns out to be inaccurate, unreliable and open to human error. GreyLogix Brasil together with Projedata developed a product to automate these three areas and reduce errors in production, thus creating the data control and processing unit (UPC) that communicates with Projedata's ERP, Autoflex . The UPC consists of a PLC and an IOT gateway, both Siemens. For communication between the UPC equipment, the S7 protocol is used, which is a native Siemens protocol. For communication between the UPC and Autoflex, MQTT is used, as it consumes little data and it is possible to generate topics in parallel without conflict between them. The main advantages of the product are automatic production counting, machine stoppages recordings, machine setups, traceability in generated labels, quick machine configurations in the UPC and reduction in the time in which production reports are sent to the supervisor, whereas before the product it took around 24 hours and with the UPC it is possible to see it in real time in Autoflex. As a result, the product managed to generate traceability in all areas and created reliability in the data.

Keywords: MQTT. S7. IOT. PLC LOGO.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Com a inovação da indústria 4.0, tem-se visto cada vez mais setores da indústria se modernizando tecnologicamente. Observa-se o aparecimento exponencial de tecnologias voltadas à comunicação e armazenamento de informações, tudo com o objetivo de maior interconexão entre todas as áreas da empresa.

Com a chegada ao Brasil, em 1949, a indústria do plástico não conseguiu acompanhar o processo de modernização devido à sua baixa margem de lucro. Isso acarretou em um baixo nível de automatização, especialmente nas áreas de corte e solda, impressão e bobinadeira.

Figura 1 – Máquina de corte e solda de sacola plástica

Fonte: (EXPORTAÇÃO, 2022)

A área de impressão tem como objetivo imprimir o logo da empresa na sacola plástica. Já a bobinadeira é a etapa do processo que une dois rolos plásticos formando uma bobina e, por último, a etapa de corte e solda corta a bobina em vários pedaços de sacola plástica, sendo esta a última fase antes do envio para o cliente.

Mesmo sendo etapas diferentes do processo de fabricação de sacolas plásticas, o funcionamento e os erros que acontecem no corte e solda, impressão e bobinadeira são os mesmos.

Figura 2 – Impressora flexográfica para sacolas plásticas

Fonte: (MAQUINAS, 2022)

Pela falta de automatização na indústria de plástico, toda a coleta de dados da produção para criação de relatórios é feita de forma manual, isso acaba trazendo diversos problemas, como: erro na contagem de produção (em torno de 5%), tempos de paradas de máquinas menores do que realmente aconteceram, erros na geração de ordens de produção, atrasos no desenvolvimento das etiquetas, um tempo de espera de 24 horas para que os supervisores obtivessem os relatórios. Tudo isso gerava uma deficiência em saber o que estava acontecendo, de fato, na produção naquele momento e tendo muitos espaços para desinformações e perdas de *insights* por falta de dados. Além disso, por não haver rastreabilidade entre os setores, caso um cliente retornasse um produto devido à qualidade ruim da sacola, não era possível descobrir de qual bobina as sacolas vieram e, em consequência, não poderiam retirar do estoque

as sacolas feitas por ela.

Sabendo desses problemas, a GreyLogix Brasil, junto com a Projedata, iniciou o desenvolvimento de uma unidade de processamento e coleta de dados (UPC), a qual faz a automatização dos processos ditos anteriormente, de forma a aumentar a agilidade e a confiabilidade da produção, além de padronizar o processo, apresentando-o em tempo real e criar uma rastreabilidade entre os setores.

Este projeto foi iniciado como um *proof of concept* (POC) na matéria de estágio obrigatório, sendo utilizado apenas um controlador lógico programável (CLP) Siemens que se comunicava via MQTT com o *software*, da Projedata, *Autoflex*. Este *software* tem o objetivo de configurar os parâmetros para o cadastro das máquinas monitoradas, apresentar os dados coletados pelo CLP e passá-los para um sistema integrado de gestão empresarial (ERP). Com isso, a POC tinha como função o controle de produção, produtividade e informações de tempo de parada e estado atual das máquinas.

Atualmente, o projeto evoluiu para a utilização de um CLP de menor porte e um IOT para controle da lógica e troca de dados via protocolos S7 e MQTT. Essa decisão foi tomada porque em vez de utilizar um CLP de grande porte, que foi desenvolvido para cálculos rápidos e lógicas complexas, porém não lida bem com *strings* e protocolos de comunicação como o MQTT, utilizou-se um CLP menor para realizar apenas os cálculos de contagem e adicionar o *gateway* IOT 2050 da *Siemens* para fazer o processamento e tratamento das *strings* e se comunicar com o *Autoflex*.

Além da questão de comunicação, a nova versão tende a facilitar e aumentar a confiança nos apontamentos de paradas da máquina, na criação de etiquetas rastreáveis para os lotes de sacolas, na organização das ordens de produção e geração dos lotes, bem como facilitar o processo de *setup* da máquina e configuração das UPC's através de um *web-server*. Tem-se então um produto que se comunica desde a base da fábrica até a parte mais alta da gerência, por meio do qual é possível trocar e guardar dados de forma eficiente, confiável, automática e com baixo gasto de processamento.

1.1 OBJETIVO E METODOLOGIA

Sendo a GreyLogix Brasil uma empresa de tecnologia e tendo um time específico de inovação e desenvolvimento de produtos, é natural o interesse em desenvolver uma solução voltada à indústria 4.0. É importante ressaltar que este trabalho é uma continuação do relatório de estágio do graduando, onde as mudanças foram feitas para transformar a POC em um produto confiável e otimizado. Entre o estágio obrigatório e o PFC foi iniciado o desenvolvimento do projeto, porém este está sendo finalizado juntamente com a matéria de fim de curso. Sendo assim, o objetivo desse projeto de fim de curso (PFC) é apresentar, de forma detalhada, as melhorias, tanto em hardware quanto em software, da UPC. O código foi alterado e novas funcionalidades foram implementadas de acordo com a solicitação dos consumidores. Também serão apre-

sentados os resultados obtidos e os *feedbacks* dos clientes, para os quais o produto está sendo implementado.

Para conseguir atingir os objetivos descritos anteriormente, serão detalhados os softwares necessários para o projeto. Será utilizado um *gateway* inteligente IoT da *Siemens*, o IOT2050, no qual será executado o software *Node-Red*. Haverá também um CLP *Siemens* Logo, cuja programação será feita por meio do software *Logo Soft Comfort*. Além disso, será utilizado um *broker* MQTT para o auxílio de tráfego de mensagens.

Serão apresentados, no decorrer do projeto, diagramas e fluxogramas para representar o processo e todas as possíveis tomadas de decisões da UPC. Além disso, serão introduzidos os conceitos dos protocolos MQTT e S7 que são usados na solução. Serão detalhados os produtos usados para a criação da UPC.

1.2 GREYLOGIX BRASIL

A GreyLogix Brasil foi criada em 2007 por seus fundadores Renato Leal e Rafael Gonçalves. Esta surgiu a partir da empresa alemã, Bilfinger Greylogix GmbH, na qual o Renato e o Rafael haviam trabalhado no ano de 2005. A partir de sua origem, a GreyLogix Brasil participou de diversos projetos internacionais e, com a entrada dos diretores Marcírio Fernandes e Antônio Tiburske, ampliou-se o escopo no qual a empresa trabalha, abordando agora, em seu portfólio, soluções para engenharia elétrica (básica e detalhada), engenharia de software, instrumentação, controle e montagem de painéis elétricos. Sendo assim, atualmente a GreyLogix Brasil trabalha com soluções completas.

Contendo 8 unidades espalhadas pelo Brasil, a empresa atua em diversas áreas, sendo elas: alimentos e bebidas; energia e meio ambiente; química e farmacêutica; papel e celulose; água e afluentes; automobilística e metalmecânica; álcool e açúcar.

Sempre tentando inovar em suas soluções e trazer o que há de mais moderno para seus clientes, em 2017 a GreyLogix Brasil ganhou o prêmio da ABDI (Agência Brasileira de Desenvolvimento Industrial) como umas das empresas mais inovadoras do país. Em 2015, a empresa ganhou também o prêmio de uma das melhores empresas para se trabalhar pelo GPTW. No mesmo ano, recebeu o certificado da ISO9001 em projetos de engenharia e montagem de painéis.

1.3 ESTRUTURA DO DOCUMENTO

Este documento está dividido em 5 capítulos, neles serão descritos o funcionamento da UPC e das ferramentas utilizadas por ela, a visão dos clientes sobre o produto e uma conclusão sobre o trabalho.

No capítulo 2, é apresentada uma fundamentação teórica, nela são apresentados os protocolos S7 e MQTT usados na solução, também é discutido sobre as ferramentas de programação do IOT e do CLP, o *node-red* e o *logo soft comfort*, sucessivamente.

No capítulo 3, é descrito o desenvolvimento do produto. São apresentados fluxogramas, diagrama UML de sequência, o modo que as mensagens são trocadas entre os equipamentos e todas as possíveis decisões que a UPC pode passar.

No capítulo 4, é realizada uma análise de resultados: vantagens e desvantagens, impactos dos resultados nos clientes, feitos a partir de *feedbacks* dos supervisores.

No último capítulo, é feita uma conclusão do PFC, apontando as motivações, problemas e resultados obtidos, assim como as perspectivas futuras do projeto.

2 FERRAMENTAS E MÉTODOS

Neste capítulo, serão explicados os conceitos dos protocolos MQTT e S7, e também será discutido sobre as ferramentas de programação *Node-Red* e *Logo Soft Comfort*.

2.1 PROTOCOLO MQTT

O protocolo *Message Queuing Telemetry Transport* (MQTT) foi criado nos anos 90 pela IBM. Sendo do padrão OASIS, ele foi projetado para transportar mensagens muito leves no método *publish/subscribe*, perfeito para conectar dispositivos remotos utilizando uma largura de banda mínima de rede, ideal para o desenvolvimento de tecnologia de internet das coisas, podendo ter centenas de clientes remotos conectados em um único servidor (VALERIE LAMPKIN, 2012). Hoje em dia, o MQTT é amplamente utilizado em diversas indústrias, tais quais: automotiva, manufatura, telecomunicações, petróleo e gás, etc. Algumas vantagens do protocolo MQTT são (VALERIE LAMPKIN, 2012):

- Estende a conectividade além dos limites da empresa para dispositivos inteligentes;
- Oferece opções de conectividade otimizadas para sensores e dispositivos remotos;
- Fornece dados relevantes para qualquer ativo inteligente de tomada de decisão que possa usá-lo;
- Permite escalabilidade massiva de implantação e gerenciamento de soluções.

Conceitos do MQTT

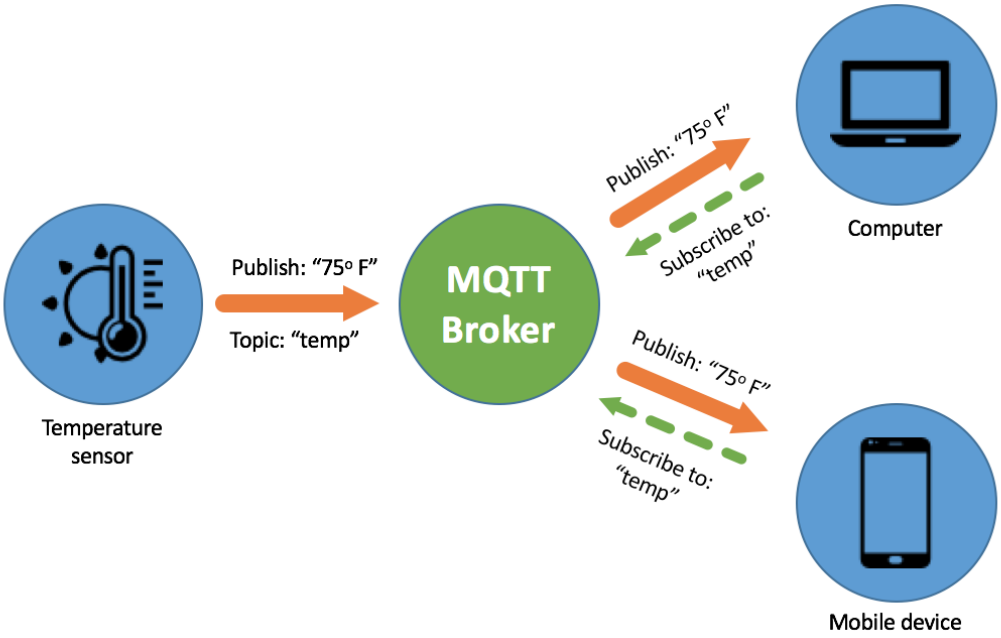
Publish-Subscribe

O protocolo MQTT segue o princípio do modelo *publisher/subscriber*, segundo o qual os clientes podem se inscrever em tópicos criados no servidor e, assim, receber qualquer mensagem que for publicada nele. Por ser uma comunicação bilateral, os clientes também podem publicar mensagens em tópicos e todos que estiverem inscritos neles receberão.

Broker

O *broker* é um conceito importante do protocolo MQTT, sendo por ele onde trafegam todas as mensagens e tópicos criados. O *broker* é o servidor que controla o envio de mensagens, checa a conectividade com os clientes, define as autenticações

Figura 3 – Comunicação MQTT



Fonte: (PINTO, 2019)

e libera a conexão com possíveis inscritos. Vê-se um exemplo, na Figura 3, onde ele está distribuindo para os dois clientes, inscritos no tópico "temp", a mensagem que foi publicada. Portanto, nota-se que o cliente inscrito e o publicador nunca se conectam diretamente, de modo que o remetente e o receptor são desacoplados (VALERIE LAMPKIN, 2012).

Inscrições e Tópicos

As mensagens no protocolo MQTT são publicadas em tópicos, que podem ser considerados como áreas reservadas para o envio e recebimento de mensagens. Por sua vez, os clientes podem se inscrever nesses tópicos; uma vez inscritos, eles recebem tudo que neles for publicado. Os tópicos podem possuir subtópicos; caso o inscrito queira receber todos os subtópicos possíveis, ele pode utilizar o operador curinga '#'.

Níveis de Qualidade de Serviço

O MQTT prevê três níveis diferentes para entregar a mensagem. Quanto maior o nível, mais esforço o *broker* tem que fazer para garantir que os inscritos a recebam. Com isso, aumenta também a quantidade de dados utilizados, aumentando assim, a latência no canal de comunicação. Os níveis que podem ser utilizados são:

- QoS 0 (No máximo uma vez): Chamada de *"fire and forget"*, este nível de qualidade de serviço envia apenas uma vez a mensagem, não checa se ela foi recebida ou não e não armazena a mensagem no *broker*. Caso o cliente inscrito no tópico esteja desconectado, a mensagem será descartada. Sendo assim, o QoS 0 é o mais rápido (MQ, 2022).

Figura 4 – QoS 0

Fonte: (HIVEMQ TEAM, 2015)

- QoS 1 (Pelo menos uma vez): Sendo este o modo de transferência padrão, caso o remetente não receba a confirmação de recebimento da mensagem, ele continuará a enviando. Ela deve ser armazenada tanto no remetente quanto no destinatário, até que o sinal PUBACK seja emitido, alertando que a mensagem foi entregue pelo menos uma vez. Após isso, a mensagem é deletada (MQ, 2022).

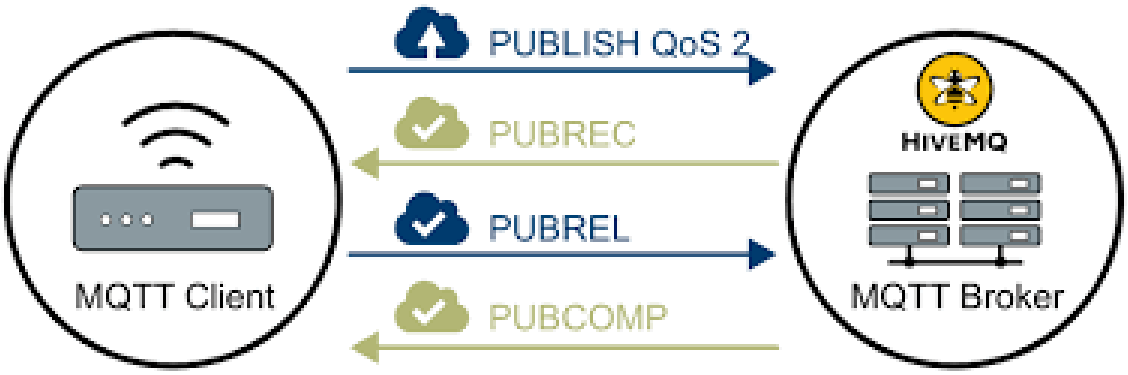
Figura 5 – QoS 1

Fonte: (HIVEMQ TEAM, 2015)

- QoS 2 (Exatamente uma vez): A mensagem será entregue exatamente uma vez. Sendo ela armazenada no remetente e no destinatário, esta será excluída até

que o processo de confirmação seja finalizado. O QoS 2 é o mais lento de todos, porém o mais seguro, pois são necessários ao menos dois pares de transmissões entre o remetente e o destinatário. No primeiro par de transmissões, o emissor transmite a mensagem e recebe a confirmação do receptor de que a armazenou. Se o remetente não receber uma confirmação, a mensagem será enviada novamente com o sinalizador DUP definido até que uma confirmação seja recebida. No segundo par de transmissões, o emissor informa ao receptor que pode concluir o processamento da mensagem “PUBREL”. Se o remetente não receber uma confirmação da mensagem “PUBREL”, esta será enviada novamente, até que uma confirmação seja recebida. O remetente exclui a mensagem salva ao receber a confirmação. O receptor pode processar a mensagem na primeira ou segunda fase, desde que não a reprocesse (MQ, 2022).

Figura 6 – QoS 2



Fonte: (HIVEMQ TEAM, 2015)

Estrutura da Mensagem

O protocolo MQTT funciona em cima da camada TCP/IP. A mensagem consiste em três partes, sendo elas: *header* fixo, *header* variável e *payload*. O *header* variável pode assumir valores de 2 a 5 bytes, porém o fixo contém apenas 2 bytes, o que pode ser visto na Tabela 1. Os quatro últimos bits representam o tipo de mensagem, como pode ser visto na Tabela 2. Os primeiros quatro bits representam as *flags* para cada *MQTT Control Packet type*, visto na Tabela 3 (BANKS, 2019).

Já o *header* variável pode estar na mensagem em determinados *MQTT Control Packet types* e pode conter até 2 bytes, sendo eles mostrados na Tabela 4 (BANKS, 2019). Por fim, tem-se o *payload*, que é a mensagem em si. O *payload* só está presente em alguns *MQTT Control Packet types*, como mostrado na Tabela 5 (BANKS, 2019).

Tabela 1 – *Header* MQTT

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type				Flags specific to each MQTT Control Packet type			
byte 2...	Remaining Length							

Fonte:(BANKS, 2019)

Tabela 2 – *MQTT Control Packet types*

Name	Value	Direction of flow	Description
Reserved	0	Forbidden	Reserved
CONNECT	1	Client to Server	Connection request
CONNACK	2	Server to Client	Connect acknowledgment
PUBLISH	3	Client to Server or Server to Client	Publish message
PUBACK	4	Client to Server or Server to Client	Publish acknowledgment (QoS 1)
PUBREC	5	Client to Server or Server to Client	Publish received (QoS 2 delivery part 1)
PUBREL	6	Client to Server or Server to Client	Publish release (QoS 2 delivery part 2)
PUBCOMP	7	Client to Server or Server to Client	Publish complete (QoS 2 delivery part 3)
SUBSCRIBE	8	Client to Server	Subscribe request
SUBACK	9	Server to Client	Subscribe acknowledgment
UNSUBSCRIBE	10	Client to Server	Unsubscribe request
UNSUBACK	11	Server to Client	Unsubscribe acknowledgment
PINGREQ	12	Client to Server	PING request
PINGRESP	13	Server to Client	PING response
DISCONNECT	14	Client to Server or Server to Client	Disconnect notification
AUTH	15	Client to Server or Server to Client	Authentication exchange

Fonte:(BANKS, 2019)

Last Will Testament

O *last will testament* tem como função o envio de uma mensagem pré-programada quando um cliente ou publicador perde a conexão. A mensagem e o tópico para o qual ela será enviada já é registrado no *broker* quando o cliente se conecta a ele. Assim, quando a conexão entre eles é perdida, o *broker* encaminha para o tópico definido a mensagem (VALERIE LAMPKIN, 2012).

Tabela 3 – Flag Bits

MQTT Control Packet	Fixed Header flags	Bit 3	Bit 2	Bit 1	Bit 0
CONNECT	Reserved	0	0	0	0
CONNACK	Reserved	0	0	0	0
PUBLISH	Used in MQTT v5.0	DUP	QoS		RETAIN
PUBACK	Reserved	0	0	0	0
PUBREC	Reserved	0	0	0	0
PUBREL	Reserved	0	0	1	0
PUBCOMP	Reserved	0	0	0	0
SUBSCRIBE	Reserved	0	0	1	0
SUBACK	Reserved	0	0	0	0
UNSUBSCRIBE	Reserved	0	0	1	0
UNSUBACK	Reserved	0	0	0	0
PINGREQ	Reserved	0	0	0	0
PINGRESP	Reserved	0	0	0	0
DISCONNECT	Reserved	0	0	0	0
AUTH	Reserved	0	0	0	0

Fonte:(BANKS, 2019)

2.2 PROTOCOLO S7

O protocolo S7 é um protocolo de comunicação proprietário da *Siemens* que facilita a comunicação entre os equipamentos da marca e é compatível com as redes Profinet, Profibus, MPI (SIEMENS, 2022a). Neste projeto, utilizam-o através de cabos ethernet RJ-45. O protocolo depende do ITOT (*ISO transport service on top of TCP*) (MARSHALL ROSE, 1987) e da porta TCP 102 para ser implementado (KLEINMANN; WOOL, 2014). O protocolo funciona como mestre/escravo, em que é necessária uma solicitação e uma resposta a ela.

O número máximo de comunicações em paralelo e o tamanho da mensagem é negociável durante o período de *setup*. É possível escrever e ler variáveis através do S7 (MIRU, 2016). Além disso, o protocolo permite múltiplos requisitos ao mesmo tempo (KLEINMANN; WOOL, 2014).

O protocolo S7 é dividido em três partes, sendo: *header*, parâmetros e dados. O *header*, como pode ser visto na Figura 7, tem o tamanho de 10 a 12 bytes, contendo (MIRU, 2016):

- *Protocol ID*: representa uma constante com valor de 0x32;
- Tipo da mensagem: este podendo ser requisito de trabalho, *acknowledge*, *ackno-*

Tabela 4 – *MQTT Control Packets* que contêm o identificador

.

MQTT Control Packet	Packet Identifier field
CONNECT	NO
CONNACK	NO
PUBLISH	YES (If QoS > 0)
PUBACK	YES
PUBREC	YES
PUBREL	YES
PUBCOMP	YES
SUBSCRIBE	YES
SUBACK	YES
UNSUBSCRIBE	YES
UNSUBACK	YES
PINGREQ	NO
PINGRESP	NO
DISCONNECT	NO
AUTH	NO

Fonte:(BANKS, 2019)

wledge data ou *user data*;

- Reservado: constante com valor 0x000;
- Referência *protocol data unit* (PDU): é gerado pelo mestre e é incrementado a cada transmissão, servindo para correlacionar a solicitação com a resposta;
- Tamanho do parâmetro: dita o tamanho da seção dos parâmetros;
- Tamanho do dado: dita o tamanho da seção dos dados;
- *Error class* e *error code*: apenas presentes em mensagens do tipo *acknowledge data*, representando os erros que podem ocorrer.

A seção “Parâmetro” traz consigo informações sobre qual ação será tomada, seja para ler ou escrever a variável. Além disso, nessa consta o endereço de onde a variável está e, encontram-se nela os seguintes dados (MIRU, 2017):

- Código da função: constante que representa se é escrita ou leitura;
- Contagem do item: número de requisitos de item;
- Requisito de item: onde é apresentado o endereço atual da variável e seu tamanho.

Tabela 5 – *MQTT Control Packets* que contêm *payload*

.

MQTT Control Packet	Payload
CONNECT	Required
CONNACK	None
PUBLISH	Optional
PUBACK	None
PUBREC	None
PUBREL	None
PUBCOMP	None
SUBSCRIBE	Required
SUBACK	Required
UNSUBSCRIBE	Required
UNSUBACK	Required
PINGREQ	None
PINGRESP	None
DISCONNECT	None
AUTH	None

Fonte:(BANKS, 2019)

Tabela 6 – Camadas do Protocolo S7

.

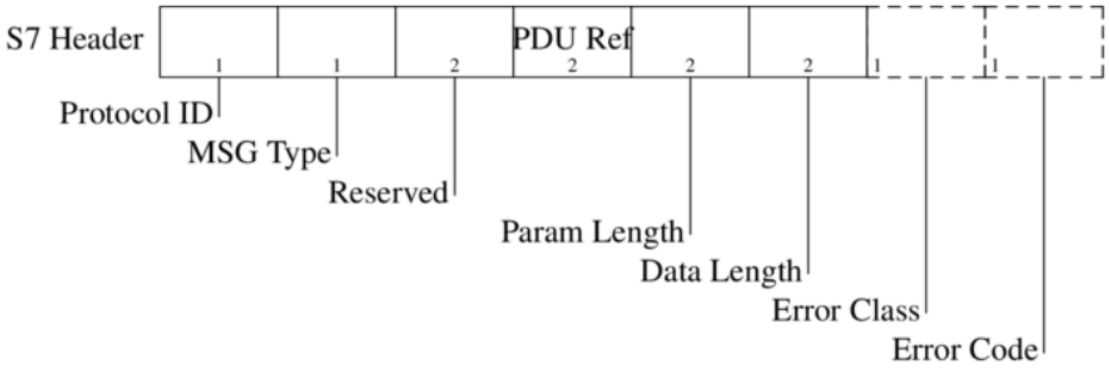
	OSI layer	Protocol
7	Application Layer	S7 communication
6	Presentation Layer	S7 communication
5	Session Layer	S7 communication
4	Transport Layer	ISO-on-TCP (RFC 1006)
3	Network Layer	IP
2	Data Link Layer	Ethernet
1	Physical Layer	Ethernet

Fonte: (WIRESHARK, 2020)

Por fim, a seção “Dados” pode variar de acordo com o tipo de mensagem, sendo escrita ou leitura. Nessa seção, há o *Data Item*, que contém o valor da variável que está sendo apontada na seção de “Parâmetros” (MIRU, 2017).

Como o S7 é um protocolo nativo da *Siemens*, o endereçamento pode ser feito

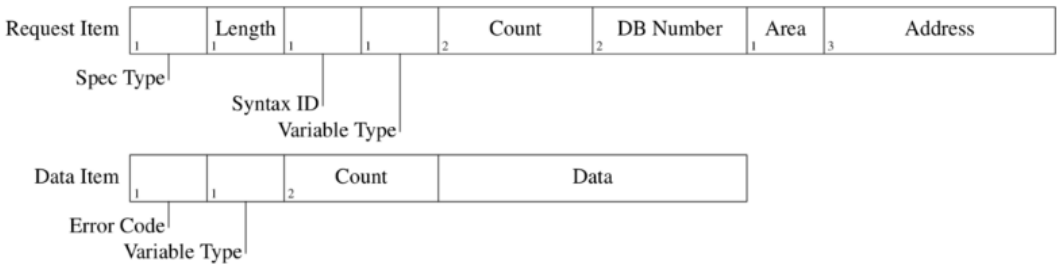
Figura 7 – *Header* S7



Fonte: (MIRU, 2017)

a partir dos endereços das variáveis (DB's), marcadores, *inputs*, *outputs*, contadores e *timers* encontrados nos CLP's.

Figura 8 – Requisito e dado de item S7

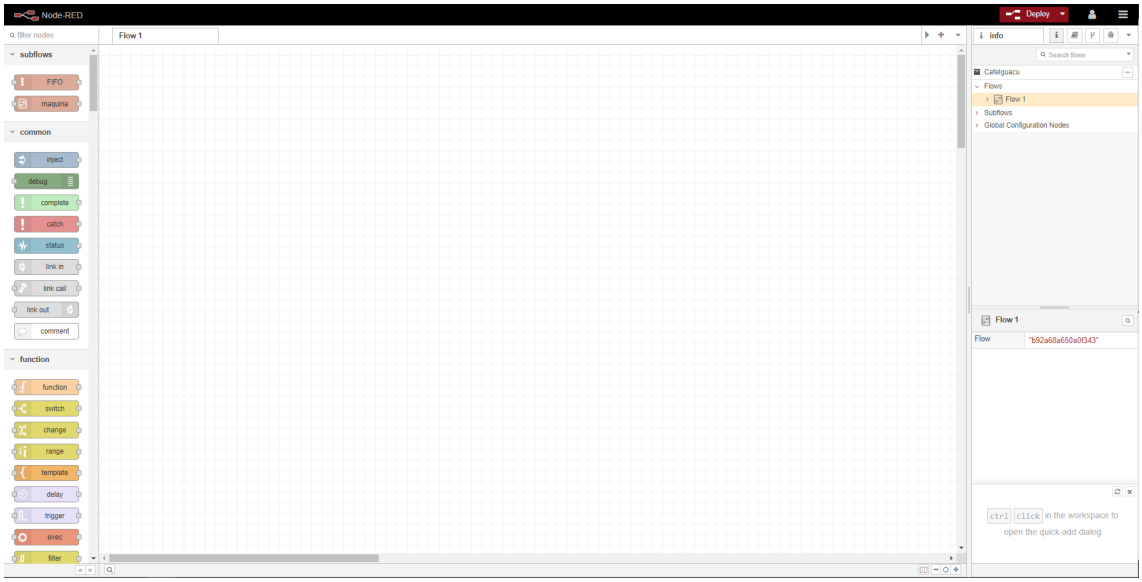


Fonte: (MIRU, 2017)

2.3 *NODE-RED*

O *Node-Red*, desenvolvido pela IBM, é um *software* de programação visual baseado em fluxos que tem como função conectar dispositivos de *hardware*, API's e serviços *online*. Pode-se abrir o editor através da *web*, conforme ilustrado na Figura 9.

Figura 9 – Editor Node-Red



Fonte: Arquivo Pessoal

Conceitos do *Node-Red*

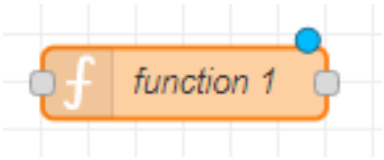
Node

O *node* (Figura 10) é um bloco básico para a criação dos fluxos, que é acionado quando recebe um objeto do um outro *node* que o antecede, ou quando recebe informações de eventos externos, como mensagens vindas de protocolos de comunicação, API's, *timers*, etc. Um *node* processa o objeto recebido e pode passar adiante a mensagem alterada ou não (FOUNDATION, 2022). Os *nodes* podem ser oriundos de bibliotecas, como *nodes* de protocolo de comunicação; podem ser do próprio sistema, como *delays*; e também podem ser nodes de função, em que é possível escrever códigos em JavaScript (THULUVA *et al.*, 2020).

Flow

O *flow* é representado como uma guia no espaço do editor do *node-red* e serve para organizar o código, podendo ter diversas instâncias que se comunicam.

Figura 10 – *Node*

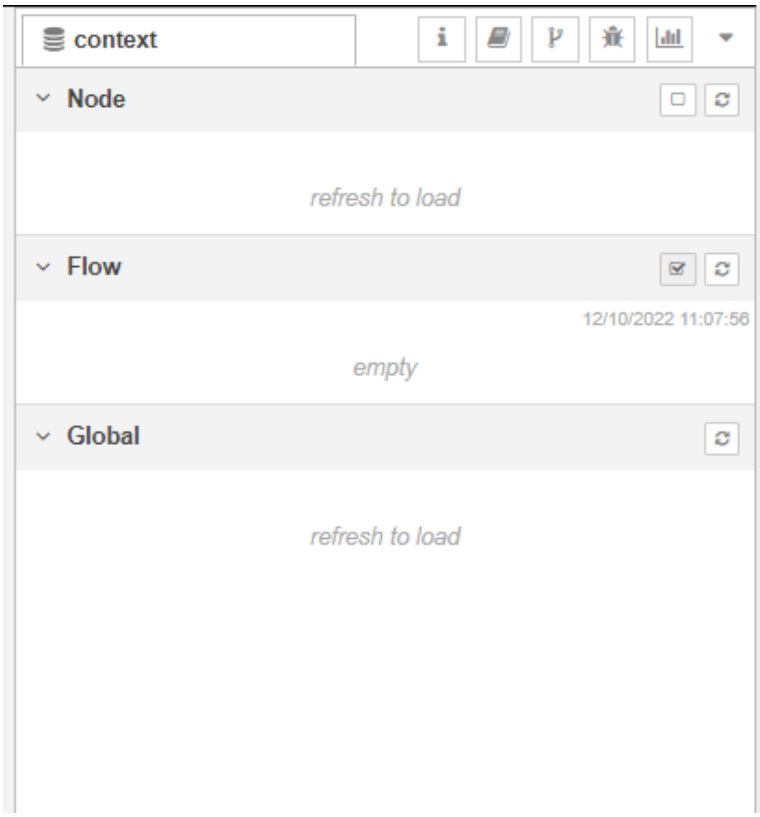


Fonte: Arquivo Pessoal

Context

O *context* (Figura 11) representa a base de dados do *node-red*. É possível guardar os valores de variáveis de maneira volátil ou persistente utilizando o *context*. Pode-se salvar os dados apenas dentro de um *node*, dentro de um *flow* ou de maneira global. O padrão é que o *context* salve apenas de maneira volátil, mas com algumas configurações é possível ativar o modo persistente (FOUNDATION, 2022).

Figura 11 – *Context Data*



Fonte: Arquivo Pessoal

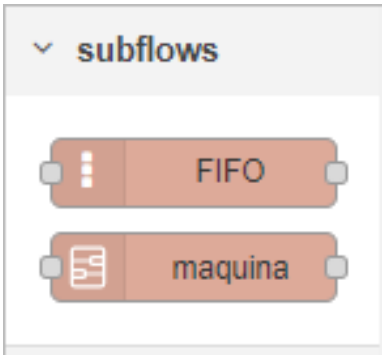
Mensagens

A mensagem é passada de *node* em *node* dentro de um *flow*, consistindo em um objeto JavaScript que pode ter várias propriedades. Por padrão, a propriedade *payload* é a que contém as informações mais úteis, mas é possível adicionar tópicos e outras propriedades de maneira dinâmica.

Subflow

O *subflow* (Figura 12) é uma junção de vários *nodes* dentro de um único *node*. Com ele, pode-se utilizar o conceito de instâncias, onde cada *subflow* contém o mesmo código, porém cada instância tem seu próprio *context* e pode agir de maneira independente. Sempre que houver uma alteração em um *subflow*, suas instâncias são atualizadas (FOUNDATION, 2022).

Figura 12 – *Subflow*



Fonte: Arquivo Pessoal

Web-Server

O *node-red* suporta um *web-server* atrelado a ele, sendo assim, pode-se criar *dashboards* interativos com o sistema no *back-end*. É possível criar abas dentro do *web-server* para ser mais *user friendly* e também separar em *containers* os objetos colocados dentro dele, deixando a tela responsiva. Pode-se utilizar a biblioteca do próprio *node-red* para criar o *dashboard* ou utilizar um *node* de programação HTML para isso.

2.4 *LOGO SOFT COMFORT*

Para programar o CLP da *Siemens*, Logo 12/24RCEo, é utilizado o *software* da Siemens *Logo Soft Comfort*. Sua linguagem de programação pode variar entre *Ladder* ou FBD, com a qual é possível criar *user-defined blocks*. Estes são códigos

desenvolvidos pelo usuário que permitem a criação de instâncias. Por ser um CLP simples, existem algumas limitações, dentre elas: o *software* lida apenas com valores inteiros, há um limite de blocos por código, há um limite de 24 entradas, sendo 4 entradas rápidas (aceitam variações de 50ms) e as outras 20 entradas conseguem captar variações de no máximo 200ms. É importante ressaltar que a comunicação entre o *software* e o CLP é através de um cabo de rede ethernet sobre a camada TCP/IP. De acordo com a *Siemens* (SIEMENS, 2011), tem-se as seguintes funcionalidades no *software*:

- Criação gráfica *offline* do código com *Ladder* ou com diagrama de funções (FBD);
- Simulação do código;
- Criar e imprimir um plano de vista geral do código;
- Salvaguarda dos dados do código no disco rígido ou em outro meio;
- Comparação de alterações atuais do código com o antigo;
- Transferência do código: do LOGO! para PC e do PC para LOGO!;
- Leitura do contador de horas de serviço;
- Ajuste da hora;
- Mudança de hora Verão/Inverno.

Conceitos do *Logo Soft Comfort*

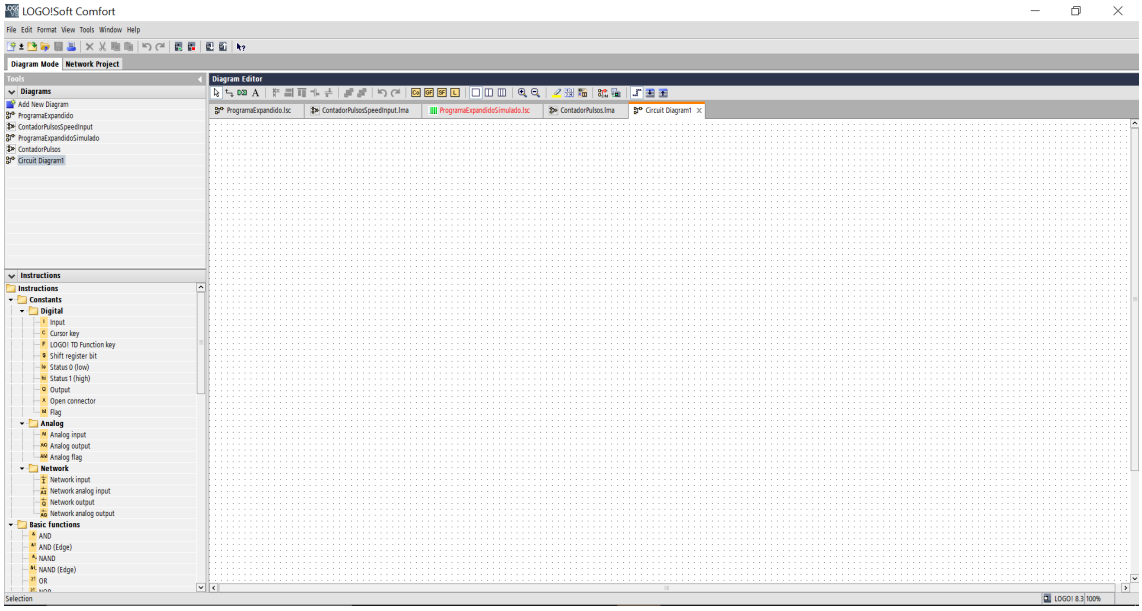
Diagrama de Circuito

No diagrama de circuito é desenvolvida a lógica para a programação do CLP. Como mencionado anteriormente, o *Logo Soft Comfort* suporta dois tipos de linguagens, *Ladder* e FBD (SIEMENS, 2022b). É possível selecionar o tipo de linguagem escolhida na criação de um diagrama de circuito, como aparece na Figura 13.

User-Defined Block

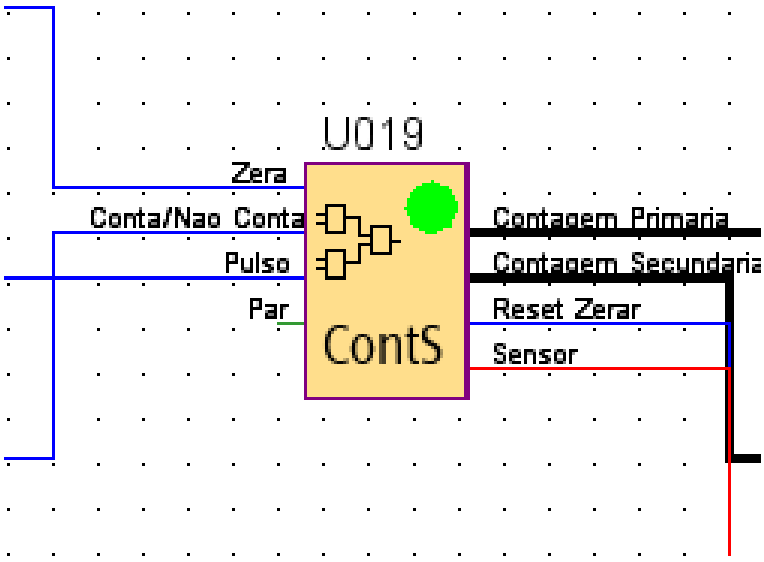
A *user-defined block* é uma ferramenta que auxilia na otimização dos códigos repetidos dentro do *software*, através dela pode-se criar um código base e transformá-lo em um único bloco com várias instâncias. Quando ocorrerem atualizações no código base, todas as instâncias são atualizadas.

Figura 13 – *Logo Soft Comfort*



Fonte: Arquivo Pessoal

Figura 14 – *User-Defined Block*



Fonte: Arquivo Pessoal

3 DESENVOLVIMENTO

Neste capítulo, será detalhado o funcionamento do produto. Serão apresentados diagramas de rede, diagrama UML de sequência e fluxograma de estados para melhor entendimento do sistema.

3.1 REQUISITOS DO PROJETO

Finalizando a POC e percebendo que era de grande valia o que havia sido desenvolvido, foi necessário readaptar o projeto para deixá-lo mais seguro, rápido e de baixo custo. Com isso, foi estudado novamente quais ferramentas e *hardwares* utilizar para que o produto se tornasse melhor. Inicialmente, foi notado que, utilizando apenas um CLP de grande porte para fazer a comunicação, coleta dos dados e seu processamento, era muito limitante, tendo em vista que eram utilizadas muitas *strings* e isso consumia muita memória do CLP. Além disso, os blocos para conexão no protocolo MQTT eram instáveis. Sendo assim, foi escolhido reduzir o tamanho do CLP para um de pequeno porte, em que este teria a única função de controlar a contagem das batidas.

Para fazer o processamento dos dados e trabalhar com a comunicação MQTT foi escolhido um *gateway* IOT 2050 da *Siemens*, porque é possível embarcar nele o *software node-red*, possibilitando o trabalho com funções na linguagem *JavaScript*, o que facilitou o tratamento de strings e também uma maior confiabilidade na comunicação com o protocolo MQTT.

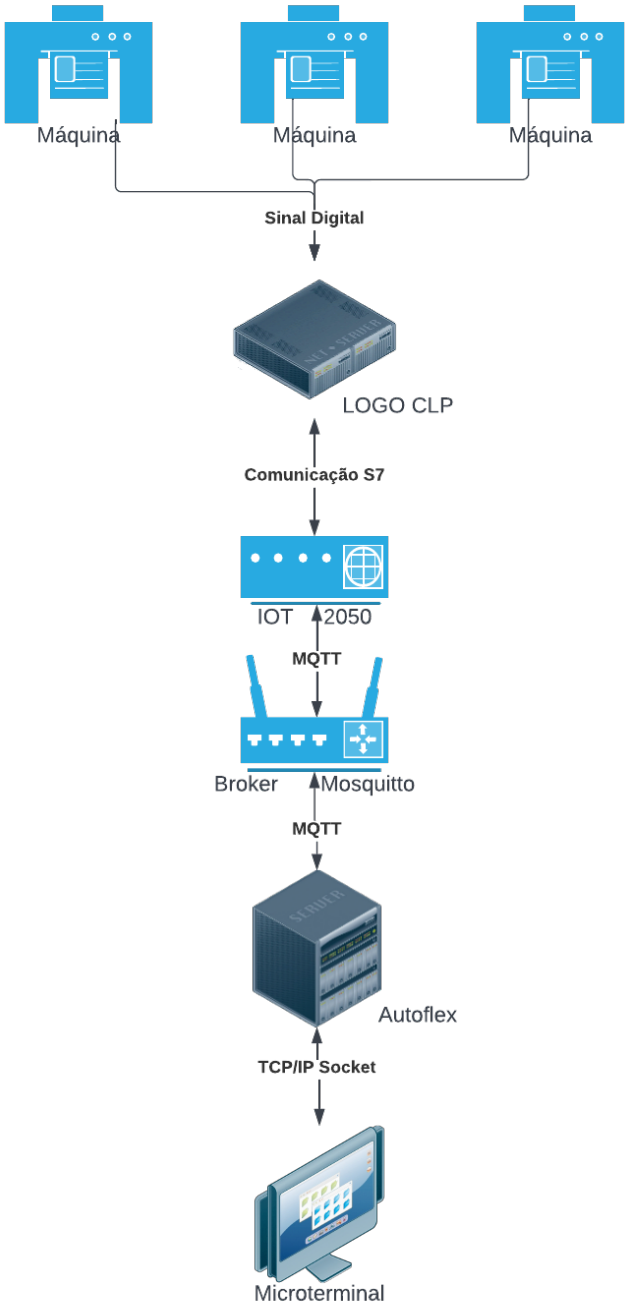
A decisão sobre os protocolos de comunicação se basearam no baixo custo para a transferência de dados. O MQTT, como será discutido adiante, tem a facilidade de ser dinâmico e independente, sendo possível um *broker* tratar diversos tópicos ao mesmo tempo sem que ocorram conflitos. O protocolo S7 foi selecionado por ser um protocolo nativo da *Siemens* e, como tanto o CLP, quanto o *gateway* são *Siemens*, a comunicação entre os dois acaba se tornando rápida e segura.

3.2 DIAGRAMA DA UPC

Como mostrado nos capítulos anteriores, a unidade de processamento e coleta de dados (UPC) consiste em um CLP e um IOT que fazem a coleta e processamento das máquinas de diferentes áreas da indústria de plástico. A Figura 15 apresenta o diagrama de comunicação entre os equipamentos.

Através do CLP é feita a captura dos sinais digitais produzidos pelas máquinas. Ocorre no Logo uma contagem dos sinais obtidos, quando a máquina está programada e não está em modo *setup*. A cada 2 segundos é enviada a contagem atual para o IOT 2050, pelo protocolo de comunicação S7. Dentro do IOT ocorre o processamento dos

Figura 15 – Diagrama de Comunicação



Fonte: Arquivo Pessoal

dados e o envio das mensagens, através do protocolo MQTT, para o *broker mosquitto* que está configurado internamente na rede do cliente. Em seguida, o *broker* encaminha a mensagem para o Autoflex que, por sua vez, encaminha através de um *socket* TCP/IP para o micro terminal, localizado ao lado da máquina e onde o operador consegue interagir.

3.3 ESTRUTURA DAS MENSAGENS

MQTT

Para facilitar a comunicação entre o IOT 2050 e o sistema ERP Autoflex, optou-se por padronizar as mensagens, seus significados e seus tópicos. Será discorrido inicialmente sobre os tópicos e suas diferenças.

Foi utilizado o tópico das mensagens MQTT para diferenciar o tipo de informação que está sendo enviado. Como será explicado mais adiante, existem 9 tipos diferentes de ações que podem ser enviadas entre a UPC e o Autoflex. Cada ação é diferenciada por um tópico e, com isso, a comunicação foi modularizada e um tópico foi mantido independente do outro. A estrutura do tópico é formada pela seguinte configuração:

- Posição 0: representa o fluxo da mensagem, de onde e para onde está indo, podendo sair da UPC e ir para o Autoflex (greylogix->autoflex) ou o caminho contrário (autoflex->greylogix);
- Posição 1: esta posição representa o evento deste tópico, ou seja, qual é a função daquela mensagem. Pode ser de programação, desprogramação, máquina parada, parada, produtividade, produção lote, registro equipamento, LWT ou *alert-danger*;
- Posição 2: representa em qual versão o código se encontra;
- Posição 3: esta posição mostra qual o código do cliente de acordo com o que foi cadastrado no Autoflex;
- Posição 4: no último elemento do tópico é apresentado o número do recurso, sendo único para cada máquina;
- Tópico completo: autoflex->greylogix/programacao/v1.1/1/502.

O *payload* é a parte onde constam as informações de fato, o que deseja-se enviar para o recurso. Como dito anteriormente, cada evento tem um tópico específico e aqui serão destrinchados cada um deles e o que há em seus *payloads*. A mensagem contém diversas informações dentro dela e, cada uma destas, é separada por ";". Nos exemplos do *payload*, os parâmetros são colocados na sequência que são explicados, e ao lado do nome de cada parâmetro será indicado seu *data type*. Porém, vale ressaltar que, quando o *payload* é enviado, ele vira uma grande *string*; os *data types* colocados abaixo são como eles são tratados dentro da UPC e dentro do Autoflex, ou seja, ocorre uma transformação da string para o *data type* desejado.

Programação

Para iniciar o recurso é necessário que o Autoflex encaminhe uma mensagem para a UPC programando a máquina em questão. Essa mensagem ditará quais os parâmetros daquele recurso. Sendo eles:

- Empresa (inteiro): o número da empresa;
- OP (int): o código da ordem de produção que será executada;
- Etapa (int): o código da etapa do processo;
- Sequência Etapa (int): código da sequência da etapa de produção;
- Recurso (int): o número da máquina em questão;
- Usuário (int): o código do operador que está monitorando o processo;
- Quantidade Programada (float): o total de sacolas para a produção;
- Unidade (string): a unidade onde são medidos os cortes;
- Data início programação (string): a hora e data em que se iniciará a produção;
- Data fim programação (string): a previsão de término da produção;
- Tempo máquina (char): o objetivo é ditar se a produtividade será medida em minutos ou horas;
- Tempo *setup* (inteiro): o tempo que será levado para fazer o *setup* do recurso;
- Quantidade lote envio (float): dita a contagem necessária para a UPC enviar um lote para o Autoflex. Esse lote é vinculado apenas para a quantidade de sacolas contadas na UPC antes de enviar para o Autoflex, e não aos lotes reais que estão sendo empacotados;
- Fator de conversão (float): tem a mesma função que a unidade. Ajuda a converter os lotes que serão enviados para o Autoflex. Caso seja necessário enviar um lote de 100 e o fator de conversão for 1000, envia-se o valor 0.2;
- Quantidade pistas (int): este elemento informa quantas pistas há no recurso, ou seja, qual a contagem feita em um sinal digital;
- Reiniciar data início (string): ao ser iniciada a produção em uma OP nova, a data de início da produção é salva, representando a data e a hora em que a bobina começou a ser utilizada. Caso ocorra de ser inserido uma nova bobina na produção, a data e a hora são reiniciadas, ajudando assim na rastreabilidade;

- Segundos Iniciar Parada (int): este elemento tem como função configurar por quantos segundos a máquina deve estar parada para ser enviado o evento “Parada” para o Autoflex;
- Troca OP (char): esta variável sinaliza se ocorrerá uma troca da ordem de produção (OP), ou não;
- Modo *setup* (int): sinaliza em qual estado está o *setup*: caso o valor seja 0, o modo *setup* está como indefinido, podendo estar ativo ou não. Quando o valor for 1, está sendo sinalizado que o sistema está entrando no modo *setup* e, por fim, quando a UPC recebe 2, é sinalizado que o modo *setup* foi encerrado;
- Motivo *setup* (int): é passado o código do motivo do *setup*, que será repassado para a mensagem de parada;
- Rastreia consumo (string): neste item é passado o código das bobinas utilizadas, sendo possível rastrear qual produção é de qual bobina;
- Segundos reativar produção (int): este parâmetro configura quantos segundos precisam ser decorridos para que o recurso saia do modo parado para o modo ativo;
- Leituras reativar produção (int): em conjunto com o parâmetro anterior, para o recurso retornar para ativo, ele deve exceder a quantidade de sinais digitais configurados nesse parâmetro em menos tempo que o configurado no parâmetro “segundos reativar produção”.
- Segundos cálculo produtividade (int): por fim, este parâmetro dita de quanto em quanto tempo a produtividade será enviada para o Autoflex;
- Exemplo de *payload*: 1;260229;500;4;502;3523370957;5357.14286;MIL;2021-08-11 00:00:00.000;2021-08-11 00:50:00.000;M;0;100;1000.0;1.0;2.0;N;30;N;0;3;303;10;2;60.

Desprogramação

Assim como o evento anterior, a desprogramação é uma mensagem enviada do Autoflex para a UPC e tem como objetivo desprogramar o recurso, ou seja, zerar os parâmetros que haviam sido colocados nela.

- Empresa (int): o número da empresa;
- OP (int): o código da ordem de produção que será desprogramada;
- Etapa (int): o código da etapa do processo;

- Sequência Etapa (int): código da sequência da etapa de produção;
- Recurso (int): o número da máquina em questão;
- Exemplo de *payload*: 1;156839;500;4;502.

Registro Equipamento

A partir deste tópico, as mensagens seguem o fluxo (greylogix->autoflex). O objetivo desse evento é para registrar o ID da UPC que está vinculada ao recurso, assim, quando o equipamento for desligado e a mensagem *last will testament* (será explicado mais à frente) for enviada, o sistema saberá quais recursos estavam registrados e, com isso, poderá redirecionar a mensagem para as sessões dos usuários (*front-end*).

- Empresa (int): o número da empresa;
- Recurso (int): o número da máquina em questão;
- ID (string): o ID criado para a UPC;
- Data atual UPC (string): a data em que a mensagem está sendo enviada;
- Versão UPC (string): a versão do código que está na UPC;
- Exemplo de *payload*: 1;502;IOT151611QRWOCDMDF;2022-10-31 18:12:33.139;v3.0

Máquina Parada

Este evento tem como objetivo identificar o estado atual do recurso, sendo ele parado ou ativo.

- Valor (char): podendo ter valor ‘S’, ‘N’ ou ‘T’, representa se a máquina está parada, ativa ou em *setup*, respectivamente;
- Exemplo de *payload*: S.

Parada

O objetivo deste evento é enviar o início e fim das paradas. Para considerar uma nova parada (data de início), o equipamento deverá aguardar o recurso ir para máquina parada igual a “S” e esperar a quantidade de segundos predefinida na programação.

- Empresa (int): o número da empresa;
- OP (int): o código da ordem de produção que será desprogramada;
- Etapa (int): o código da etapa do processo;

- Sequência Etapa (int): código da sequência da etapa de produção;
- Recurso (int): o número da máquina em questão;
- Motivo parada (int): código provindo da programação que sinaliza qual o motivo da parada;
- Data hora início (string): data e hora do início da parada;
- Usuário início (int): usuário que iniciou a parada;
- Data hora fim (string): data e hora em que a parada foi finalizada;
- Usuário fim (int): usuário que finalizou a parada;
- Exemplo de *payload*: 1;156839;500;4;502;3;2020-06-22 10:12:13.250;3523370957; 2020-06-22 12:05:13.560;3523370957.

Produção-Lote

Este evento tem o objetivo de fazer a contagem da produção, sendo assim, na programação será enviado o tamanho do lote de recebimento e a UPC, ao chegar na quantidade predefinida, envia a informação.

- Empresa (int): o número da empresa;
- OP (int): o código da ordem de produção que será desprogramada;
- Etapa (int): o código da etapa do processo;
- Sequência Etapa (int): código da sequência da etapa de produção;
- Recurso (int): o número da máquina em questão;
- Usuário (int): quem estava operando o sistema;
- Data hora início (string): data e hora em que a bobina começou a ser usada;
- Data hora registro (string): data e hora em que o lote foi enviado;
- Quantidade lote (float): a produção necessária para formar um lote;
- Unidade (string): unidade de conversão, assim como explicado no tópico “Programação” no parâmetro unidade;
- Peso lote (float): o peso de cada lote. Parâmetro atualmente configurado como zero;
- Tara lote (float): a tara da balança. Parâmetro atualmente configurado como zero;

- Rastreia consumo (string): código que identifica qual bobina está sendo utilizada;
- Exemplo de *payload*: 1;156839;500;4;502;3523370957;2022-09-08 08:03:18.657; 2022-09-08 10:34:06.127;0.2;MIL;0;0;303.

Produtividade

O evento de produtividade fornece a projeção da produção em um minuto ou em uma hora. O tempo para esse evento ser enviado varia de acordo com a "programação" e, com esse valor, é efetuado um cálculo para saber a projeção da produção.

- Valor (float): a projeção da produção;
- Unidade (string): segue o mesmo conceito do parâmetro "unidade" da produção-lote;
- Tempo máquina (string): mostra se a projeção é de um minuto ou de uma hora;
- Exemplo *payload*: 0.055;MIL;MIN.

Alert Danger

Este evento tem o objetivo de enviar um alerta para o micro terminal com um destaque em vermelho, representando erro e perigo. Dentre os erros cadastrados estão: número de pistas inexistente e fator de conversão fora do permitido.

- Valor (string): texto contendo o erro ocorrido;
- Exemplo de *payload*: O número de pista está nulo.

Last Will Testament

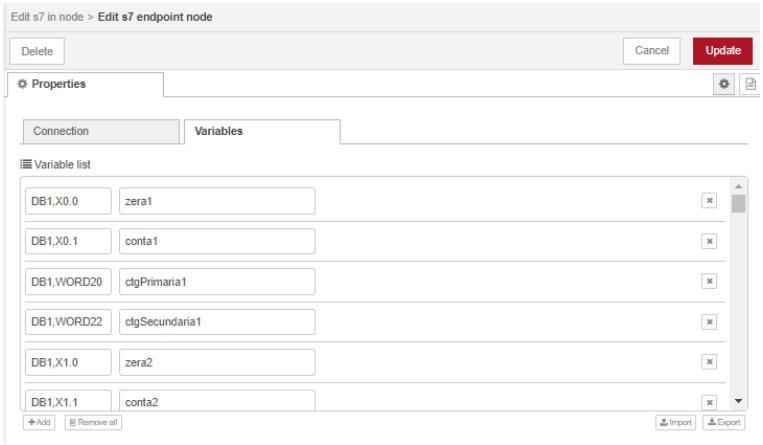
O objetivo deste evento é informar aos inscritos no tópico que houve uma desconexão. Caso a UPC perca conexão com o *broker*, este enviará uma mensagem já pré-programada ao Autoflex informando a queda de conexão.

- Valor (string): a mensagem configurada;
- Exemplo de *payload*: equipamento "IOT151611QRWOCDMDF" desconectado! Pode haver perda de dados. Por favor, contactar responsável.

Comunicação S7

O protocolo S7 segue uma estrutura diferente do MQTT. Por ser uma comunicação entre equipamentos *Siemens*, o IOT deve apontar para os endereços das variáveis que estão localizadas no CLP. Para o projeto, foi escolhida a forma de puxar todas as variáveis selecionadas de uma única vez a cada dois segundos, ou seja, a cada dois segundos, o IOT solicita ao CLP o envio de todas as variáveis configuradas em sua lista, como é visto na Figura 16.

Figura 16 – Lista de variáveis do S7



Fonte: Arquivo Pessoal

As variáveis trocadas com cada máquina configurada no CLP são: contagem primária e secundária, zera, conta e led. Será explicado mais adiante o significado de cada uma delas.

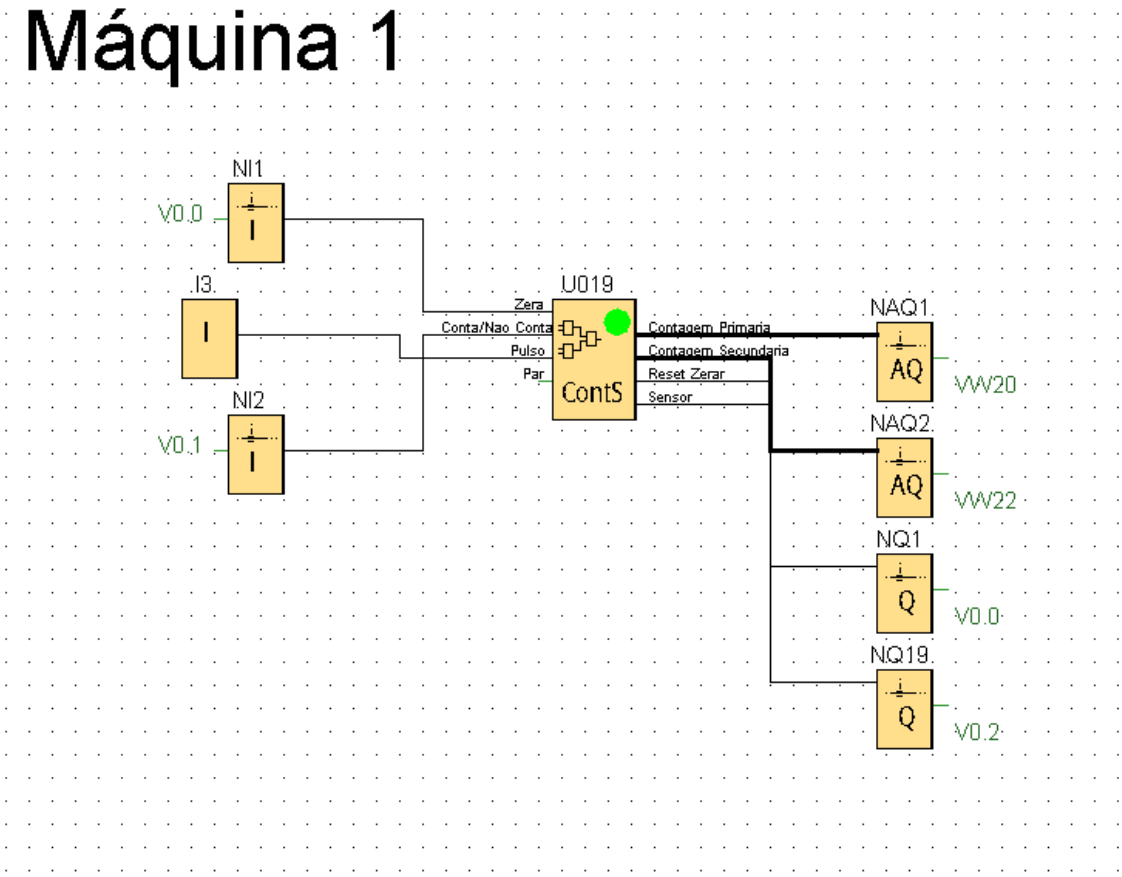
3.4 ESTRUTURA DO CÓDIGO

Logo Soft Comfort

O código desenvolvido dentro do CLP tem como objetivo fazer a contagem dos sinais digitais recebidos para determinada máquina. Cada CLP consegue contabilizar 18 máquinas simultaneamente e de maneira independente. O CLP tem uma comunicação bidirecional com o IOT, ele recebe o valor de 2 variáveis e envia 3 para o IOT. Como pode-se ver na Figura 17, o código principal é um *user-defined block* (UDF) e cada máquina possui o seu próprio, com suas variáveis de entrada e saída.

Como pode-se ver no diagrama de funcionamento do UDF, Figura 18, tem-se 3 entradas em cada instância. O sinal digital é o sinal provindo do recurso em questão, a variável “conta” é um *booleano* que indica se o sistema deve contabilizar o sinal digital ou não e, por fim, a variável “zera” representa se as contagens já feitas devem

Figura 17 – Código de uma máquina



Fonte: Arquivo Pessoal

ser reiniciadas. A saída do UDF são as contagens já feitas e o sinal se a máquina está ativa ou não. Como o *Logo* aceita apenas números inteiros, indo até 32767, quando o primeiro contador atinge o limite, ele é reiniciado e o contador secundário é incrementado.

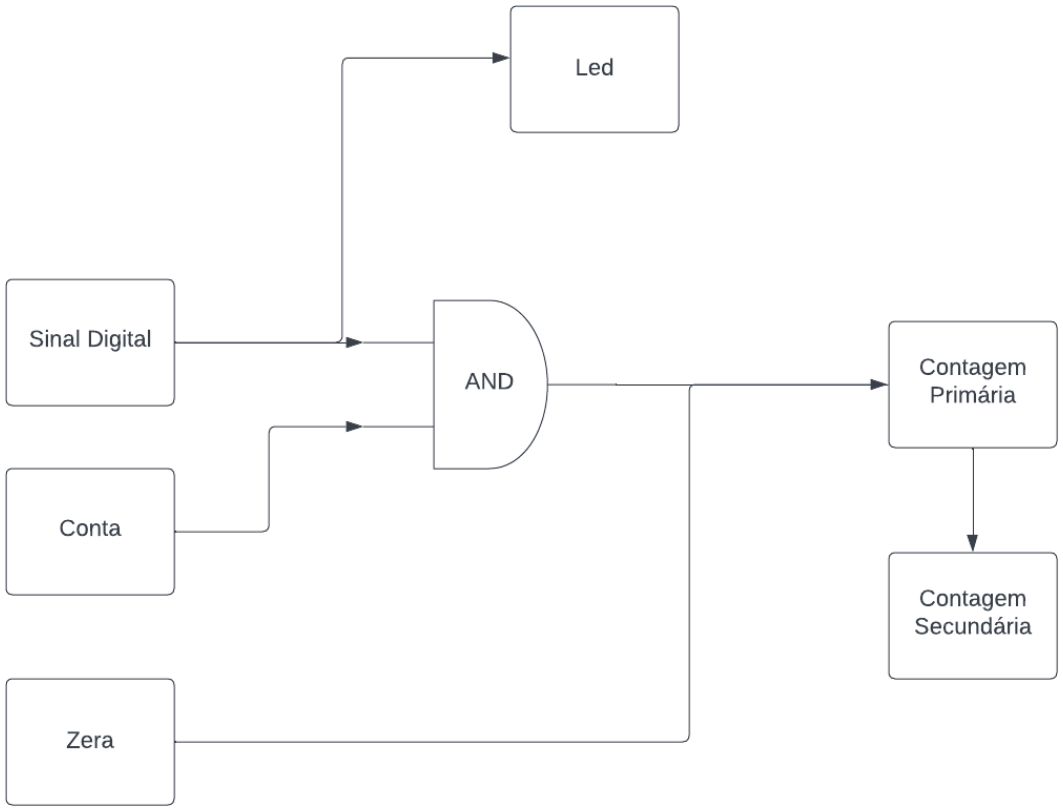
Node-Red

Dentro do IOT é onde ocorre todo o processamento dos dados, criação e tratamento dos eventos. Foi utilizada a ferramenta *subflow* para criar instâncias do mesmo código para todas as 18 máquinas. Nesta parte, o código será apresentado na sequência em que a programação é recebida, processada e enviada para o Autoflex. A Figura 19 mostra a representação de cada *flow* e para onde vão os dados tratados deste *flow*.

Inicialização

O primeiro *flow* presente no código é o de inicialização (Figuras 20 e 21). Nele, a variável denominada “comodato” é ativada e tem como objetivo indicar ao sistema

Figura 18 – Diagrama lógico do UDF



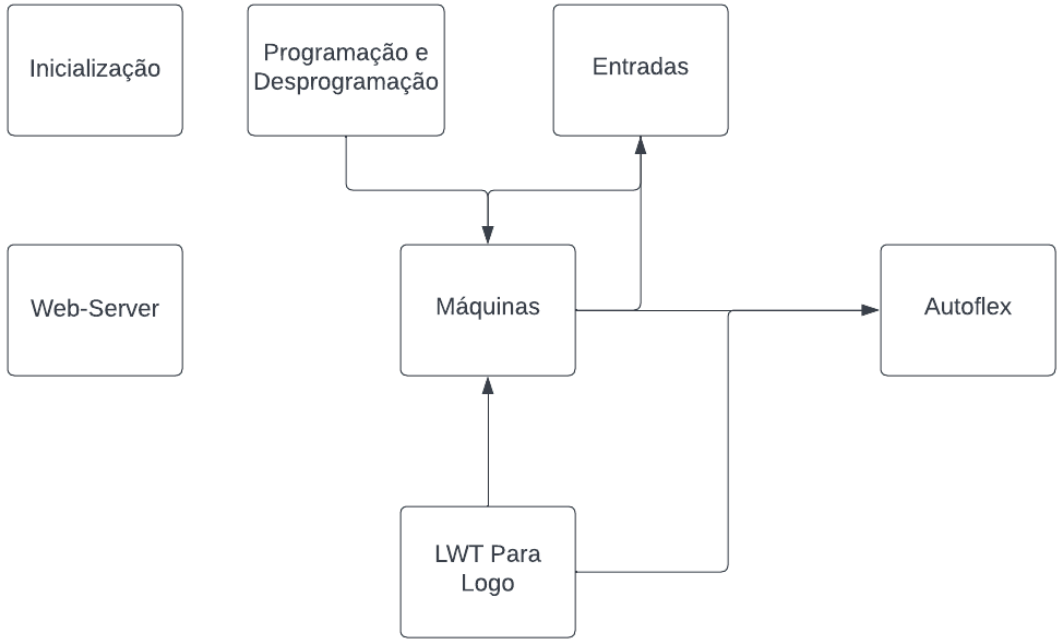
Fonte: Arquivo Pessoal

que o IOT em questão pode ser usado até determinada data, já definida no programa. O comodato é checado diariamente até o dia de vencimento do contrato. Após o vencimento, todo o sistema é bloqueado e não há mais processamento dos dados. Além disso, nesse *flow* são inicializados todos os *arrays* que serão utilizados no banco de dados, tais como os nomes das máquinas, as entradas do CLP que estão configuradas para quais instâncias de *subflow*, a quantidade de máquinas já cadastradas e o ID do IOT em questão.

Programação e Desprogramação

No *flow* “Programação e Desprogramação” (Figuras 22 e 23) é onde são processadas as mensagens recebidas do Autoflex. Com dois *nodes* de entrada do protocolo MQTT recebem-se, através dos tópicos de programação e desprogramação, as mensagens de configuração de cada recurso sendo utilizado. Após passar pelo nó do MQTT, a mensagem passa por uma função que checa se o comodato está ativo e, em seguida, passa por uma segunda função, a qual checa em que instância o recurso configurado

Figura 19 – Fluxograma dos *flows*



Fonte: Arquivo Pessoal

está, e assim o direciona para o *subflow* correto. O recurso e a instância do *subflow* são interligados pelo *web-server*, que será discutido posteriormente.

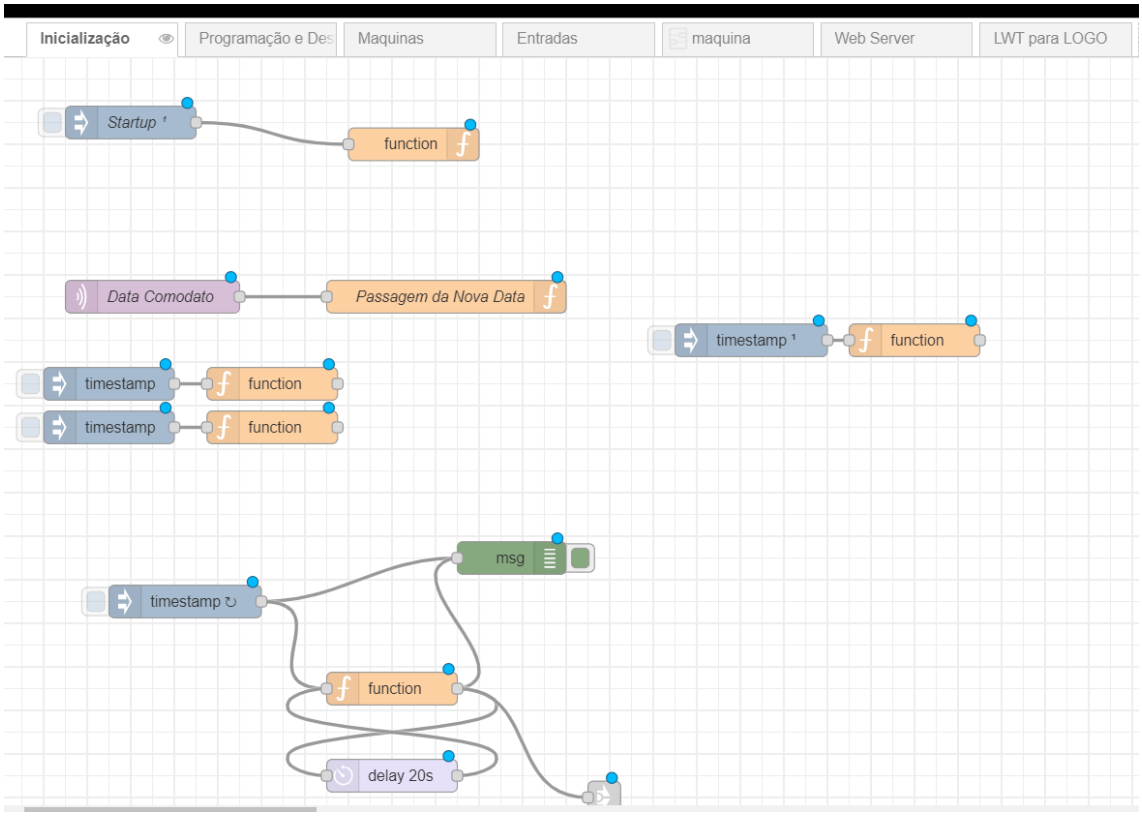
Entradas

Neste *flow* (Figuras 24, 25 e 26) são recebidos e enviados os dados que passam pelo protocolo S7. Como dito anteriormente, o nó S7 tem uma lista de variáveis que é solicitada ao CLP a cada 2 segundos. Após receber os valores, em formato JSON, ele é passado para uma função denominada “Direcionamento”, cujo objetivo é separar a mensagem JSON em 18 mensagens específicas, uma para cada recurso. Em seguida, cada uma das mensagens segue para uma função que representa sua entrada digital no CLP, e então ocorre um segundo redirecionamento para sua instância do *subflow* correta. Além de entrada do S7, nesse *flow* tem-se as suas saídas, em que são recebidos, das instâncias do *subflow*, os valores das variáveis “conta” e “zera”, que são direcionados para os recursos corretos dentro do CLP.

Máquinas

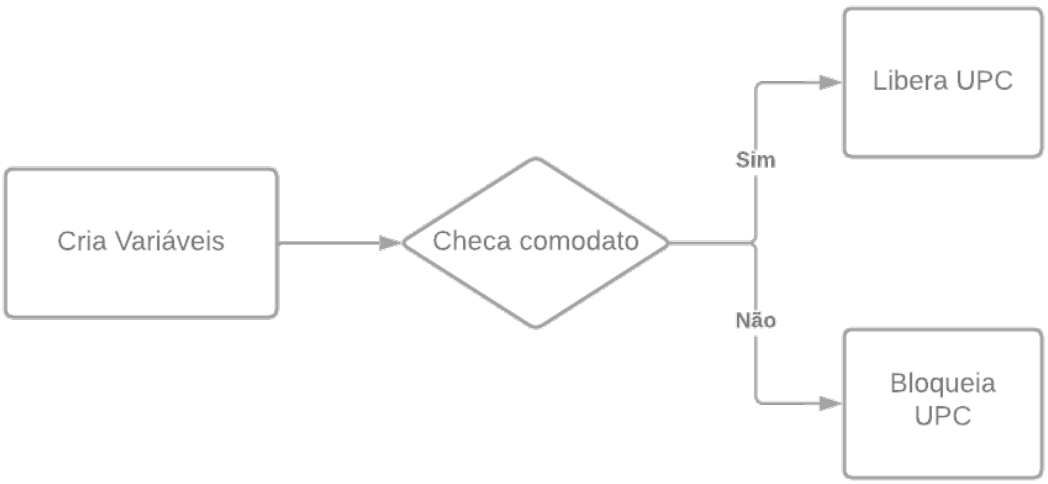
Neste *flow* (Figuras 27 e 28) é onde ocorre o processamento das 18 instâncias do *subflow*, tratamento dos dados e o envio das mensagens para o Autoflex através do protocolo MQTT. Aqui será destrinchado o funcionamento do *subflow*.

Figura 20 – *Flow* Inicialização



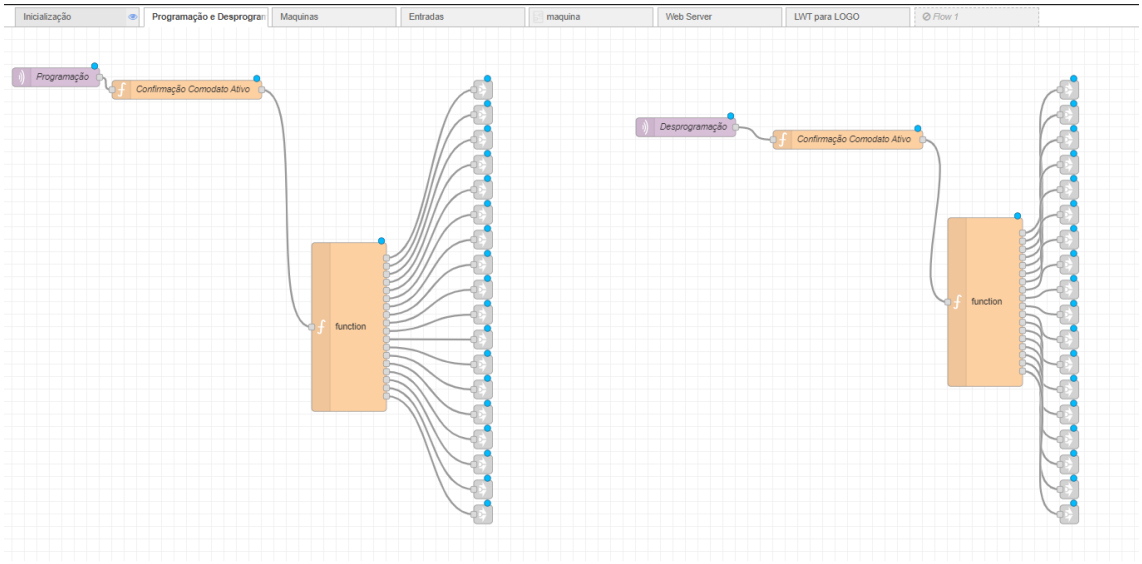
Fonte: Arquivo Pessoal

Figura 21 – Fluxograma da inicialização



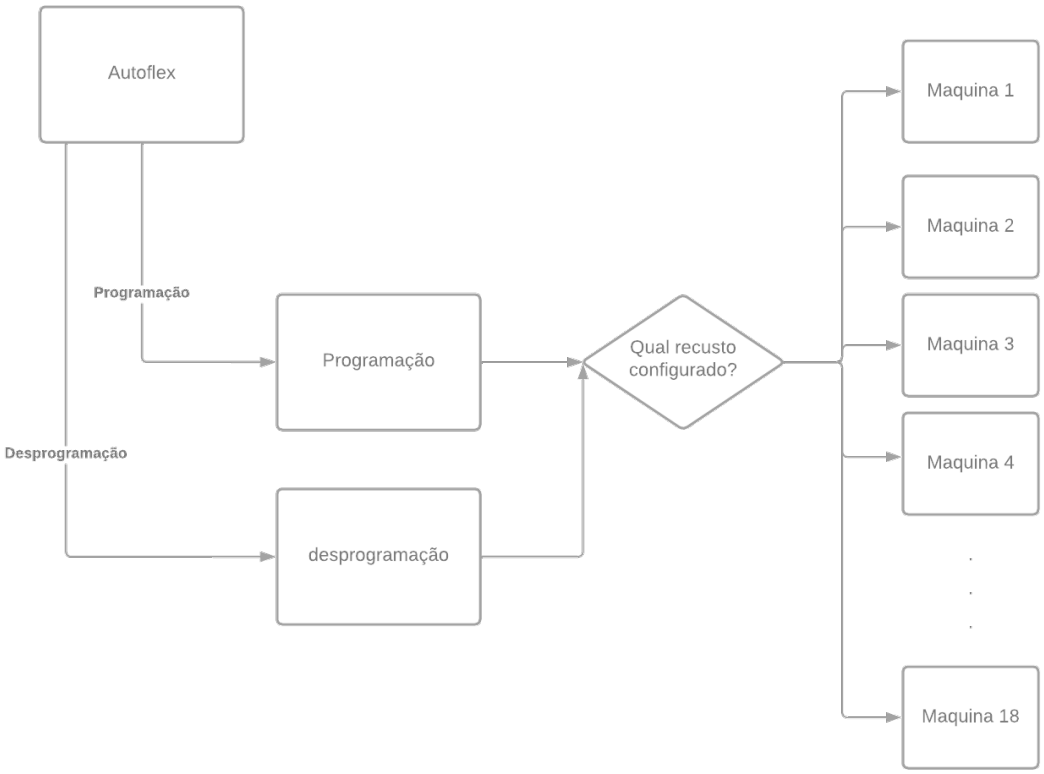
Fonte: Arquivo Pessoal

Figura 22 – *Flow* Programação e Desprogramação



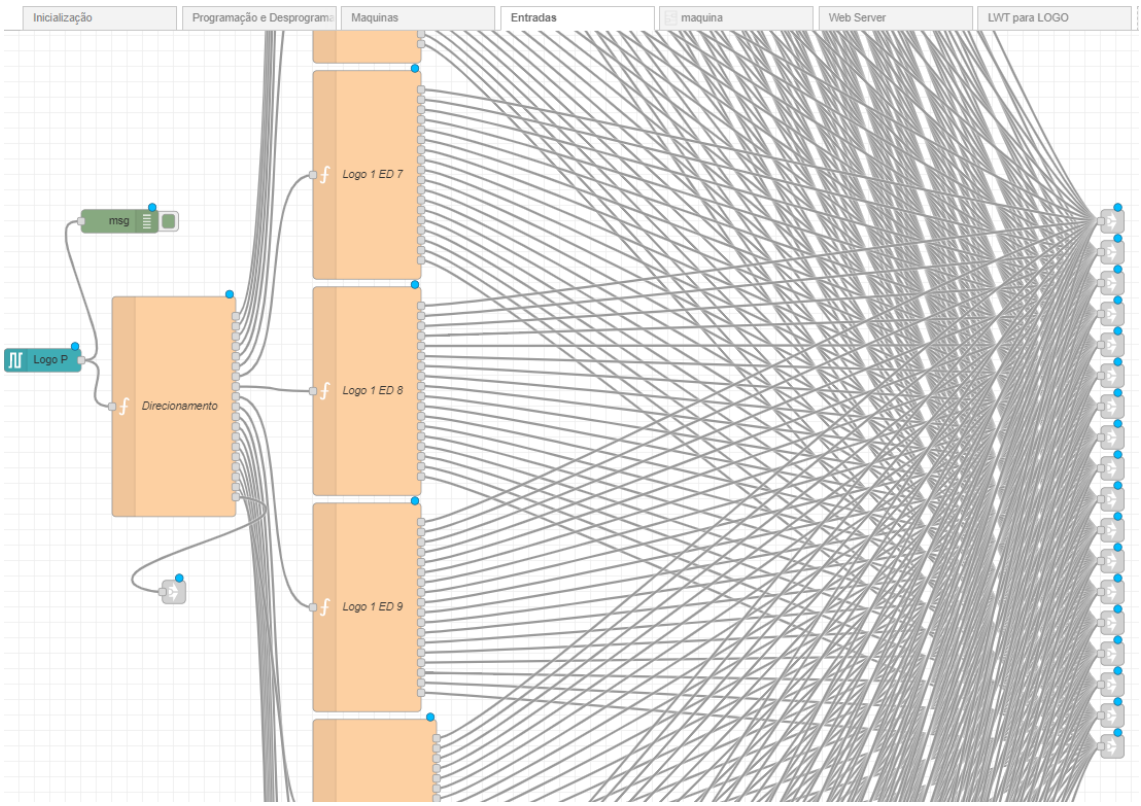
Fonte: Arquivo Pessoal

Figura 23 – Fluxograma da programação e desprogramação



Fonte: Arquivo Pessoal

Figura 24 – *Flow* Entradas

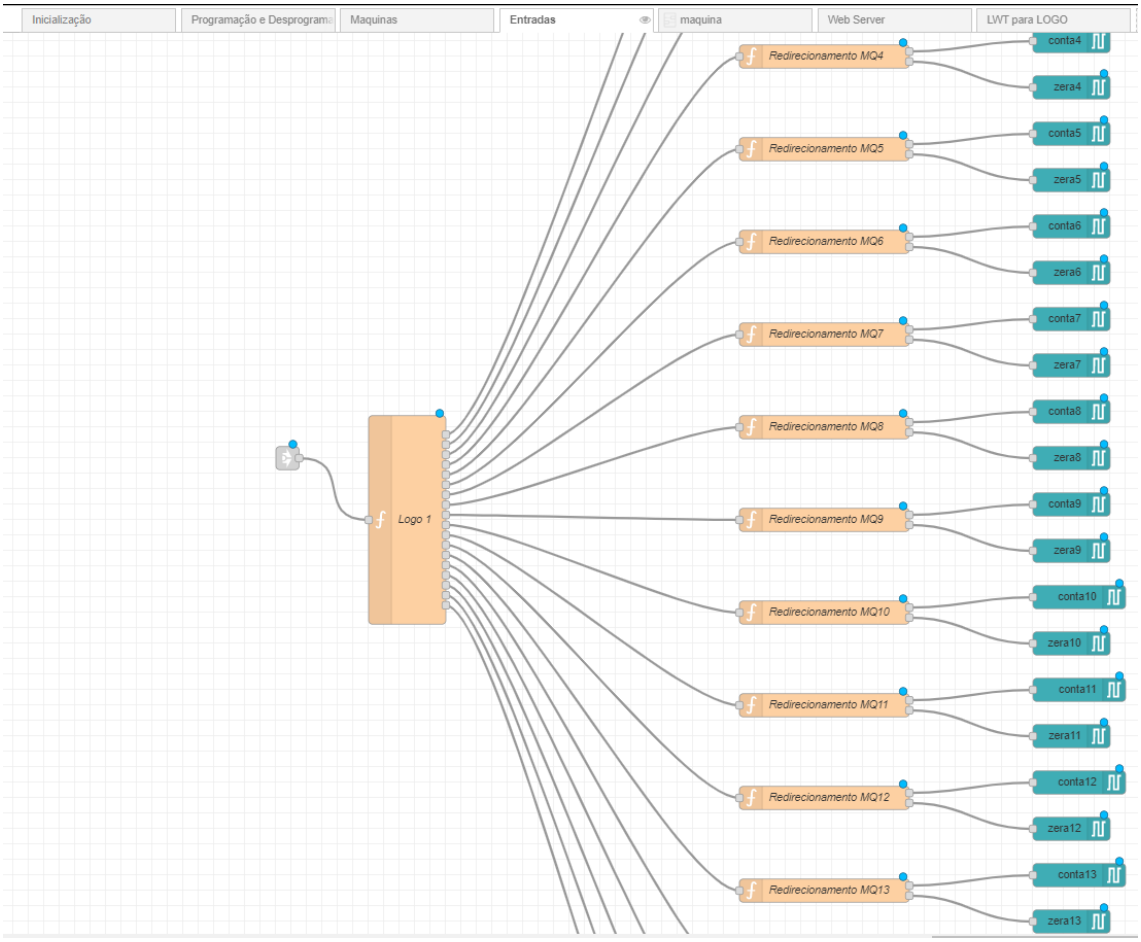


Fonte: Arquivo Pessoal

Quando a instância recebe uma programação, separa-se a mensagem em suas respectivas variáveis e, após isso, passa-se por uma função “ProgramaCheck” em que é checado o estado em que o IOT está. Como será visto adiante, o programa funciona a partir de estados, em que cada um deles dita o evento que ocorrerá a seguir. Após o IOT definir o estado em que se encontra, ele segue enviando a mensagem “Registro Equipamento” para o Autoflex e sinaliza para o CLP as variáveis “conta” e “zera”. Após o recurso receber sua programação, aquela instância está pronta para realizar as capturas e processamentos dos dados.

Como visto anteriormente, a cada dois segundos o *flow* “Entradas” envia a quantidade total de sinais digitais registrados em cada máquina. Chegando essa informação no *subflow*, ela é direcionada para a função “Tratamento Pulsos”, onde ocorre o tratamento da contagem e a divisão dos lotes. Quando a máquina está produzindo e a função sinaliza que um lote foi finalizado, é enviada uma mensagem para a função seguinte, “Produção-Lote” que, por sua vez, cria o *payload* MQTT necessário para envio para o Autoflex. Na função “Tratamento Pulsos“ também é checado se uma máquina está ativa ou parada. Caso a contagem recebida na mensagem atual seja maior que a anterior, a máquina está produzindo. Se a contagem anterior for igual à atual, isso representa que a máquina está parada, então é disparado um *timer* de 5 segundos.

Figura 25 – *Flow* Entradas



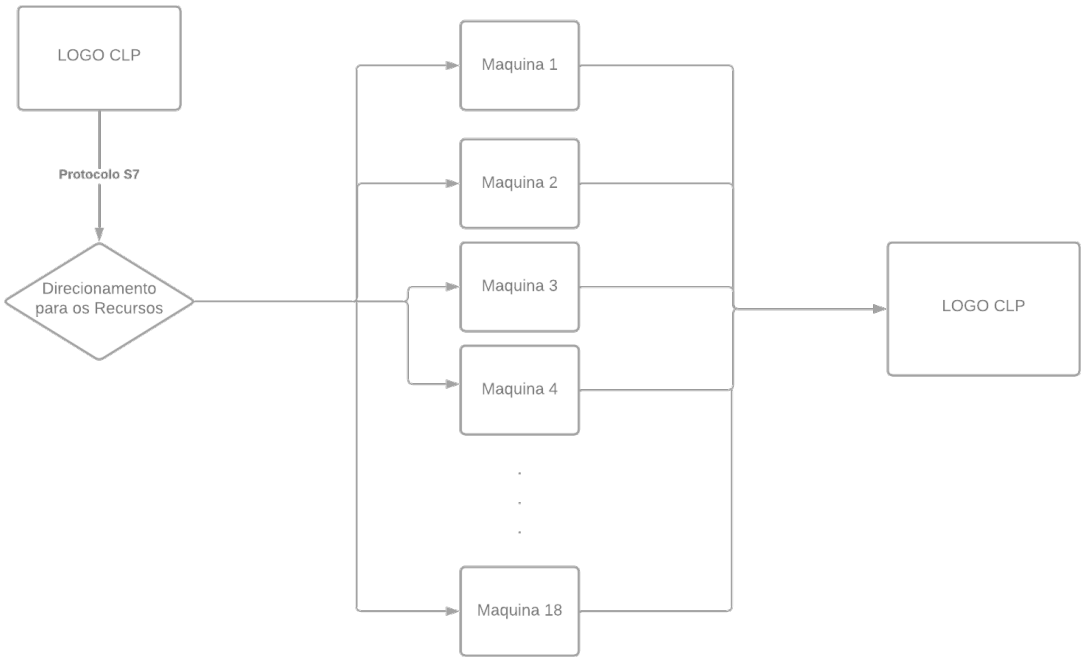
Fonte: Arquivo Pessoal

Caso esse *timer* finalize antes de receber uma mensagem com uma contagem total maior, o recurso entra no estado de “máquina-parada”, como discutido anteriormente. Além disso, quando o recurso se encontra parado e recomeça a aumentar a contagem, a máquina só se tornará ativa novamente se ele conseguir passar a contagem do parâmetro “leituras reativar produção” em um tempo menor que o parâmetro “segundos reativar produção”; caso não consiga, a contagem gerada nesse período é descartada e o recurso se mantém parado.

Quando é enviada uma mensagem para o Autoflex sinalizando que a máquina está parada, é iniciado um *timer* para que ocorra o evento “Parada” e, se a máquina continuar parada quando esse *timer* finalizar, é gerada uma mensagem para o Autoflex com a parada iniciada. Quando o recurso voltar à ativa, esta parada é finalizada e enviada para o Autoflex.

Quando a contagem do *flow* “Entradas” é recebida, essa também é direcionada para a função “Tratamentos Pulsos Produtividade”, onde nela ocorre o cálculo da produtividade a ser enviada. Tem-se um *timer* que gera o envio da produtividade

Figura 26 – Fluxograma da entrada



Fonte: Arquivo Pessoal

de acordo com o recebido na programação através do parâmetro “segundos cálculo produtividade”.

Por fim, quando recebe-se uma desprogramação proveniente do *flow* “Programação e Desprogramação”, envia-se uma *flag* para o sistema sinalizando que é necessário enviar toda a produção remanescente e finalizar qualquer parada que esteja em aberto.

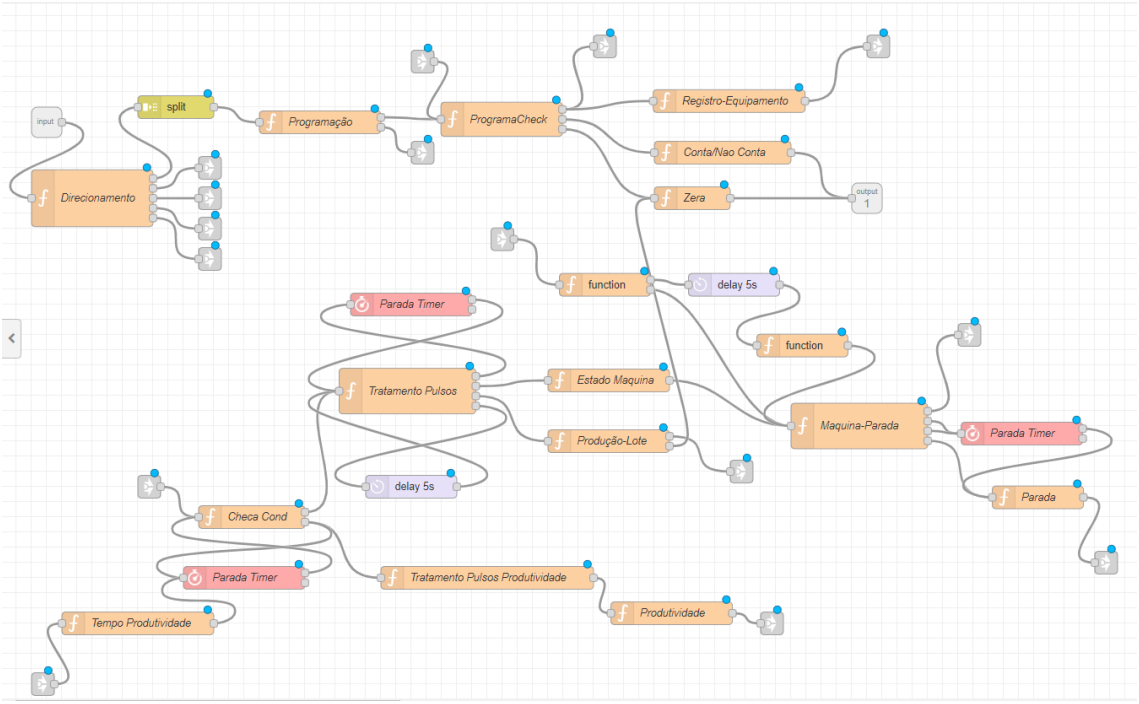
Web-Server

No *flow* “*Web-Server*” é feita a lógica de criação e remoção dos recursos (Figuras 29 e 30), estado dos sensores (Figura 31) e configuração dos IP’s da UPC, do *broker* e o endereço MAC do IOT (Figura 32). No *web-server* também há uma aba que mostra o *status* de todas as máquinas, se estão ativas ou paradas.

Quando um recurso é criado, é necessário que sejam passadas algumas informações, tais quais:

- Entrada: esse item dita qual entrada digital do CLP está sendo usada;
- Saída: dita qual saída digital do CLP está sendo utilizada;
- Máquina interna: representa qual *subflow* será utilizado para esse recurso;

Figura 27 – *Subflow* Máquina



Fonte: Arquivo Pessoal

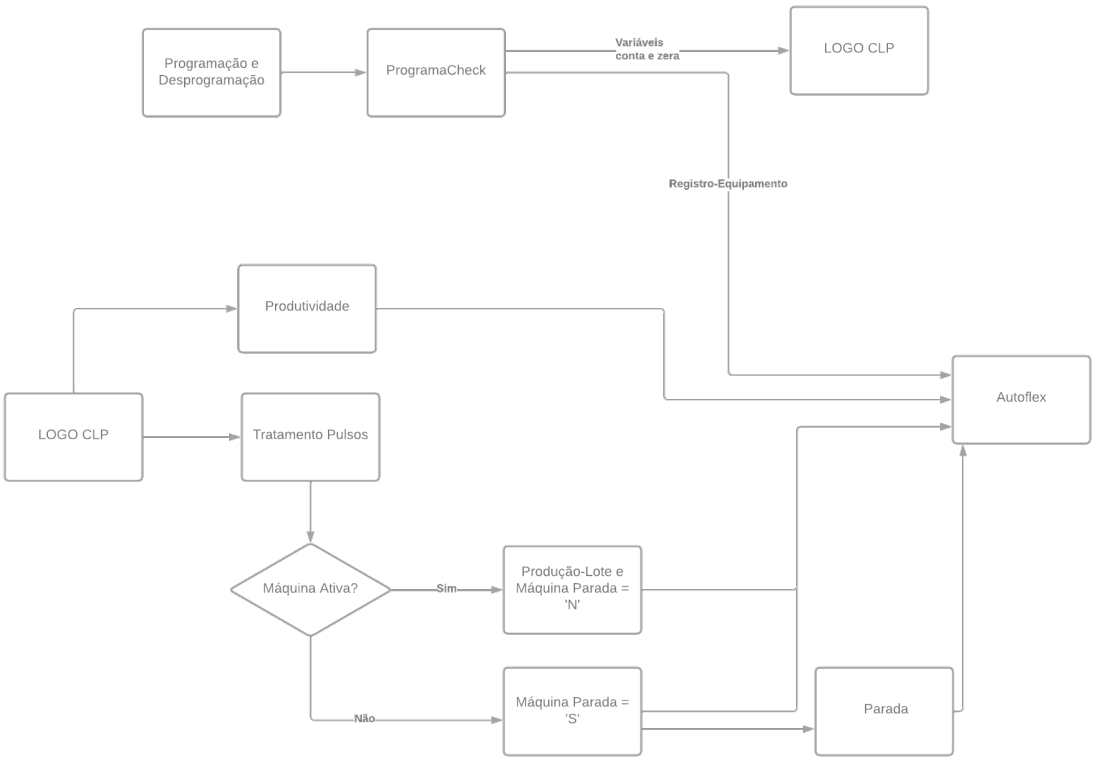
- Logo: representa qual CLP será utilizado, pois futuramente haverá mais de um CLP se comunicando com o IOT, sendo possível a utilização de múltiplas máquinas simultaneamente;
- Empresa/Recurso: mostra o código da empresa e do recurso em questão.

Terminando a configuração do recurso, o sistema já cria os tópicos MQTT de programação e desprogramação desta máquina e os associa a um *subflow* específico. Então, toda vez que uma mensagem é recebida no *flow* “Programação e Desprogramação”, ele compara o tópico da mensagem com as que estão salvas no sistema, sabendo assim para onde direcionar a mensagem recebida.

LWT para Logo

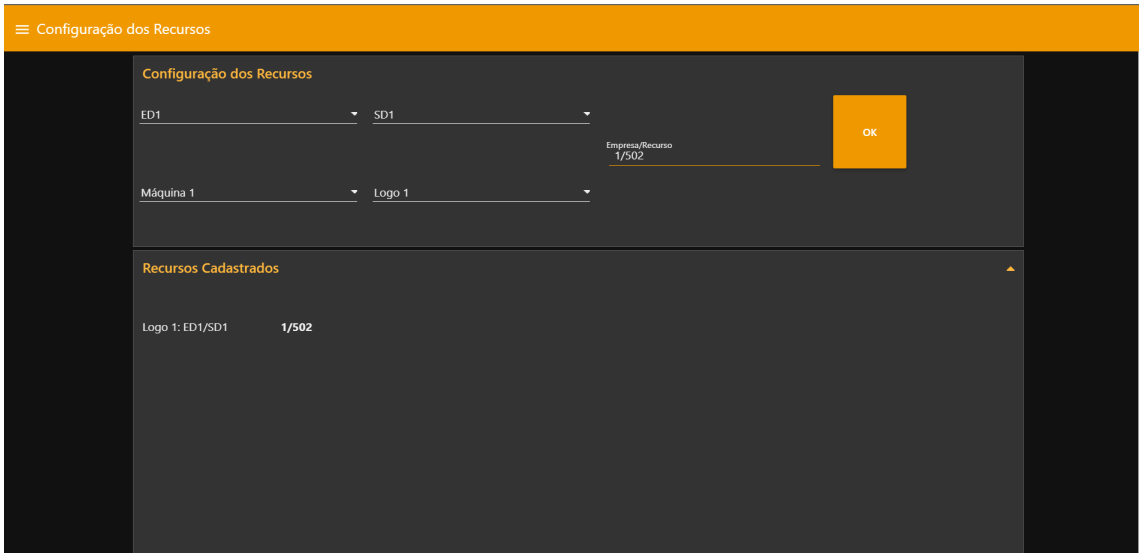
Este último *flow* (Figura 33) tem a função de sinalizar ao Autoflex caso ocorra uma perda de comunicação com o CLP. A cada 30 segundos é enviado um sinal ao Logo e, caso ele não retorne o sinal, é enviada uma mensagem para o Autoflex, através do MQTT, avisando ao sistema que o CLP não está mais respondendo. No lado do CLP também é notado que não foi recebido o sinal, e então um LED, presente na frente da caixa, começa a piscar vermelho, sinalizando que é aquela UPC que sofreu perda de comunicação.

Figura 28 – Fluxograma da máquina



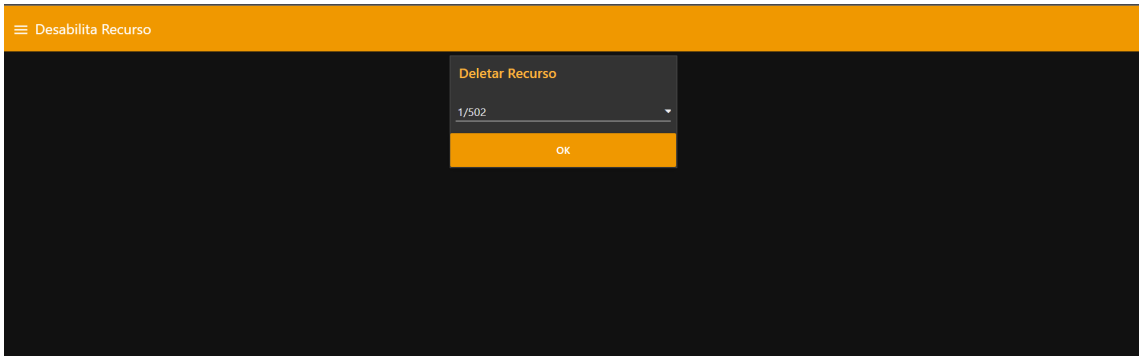
Fonte: Arquivo Pessoal

Figura 29 – Configuração do Recurso



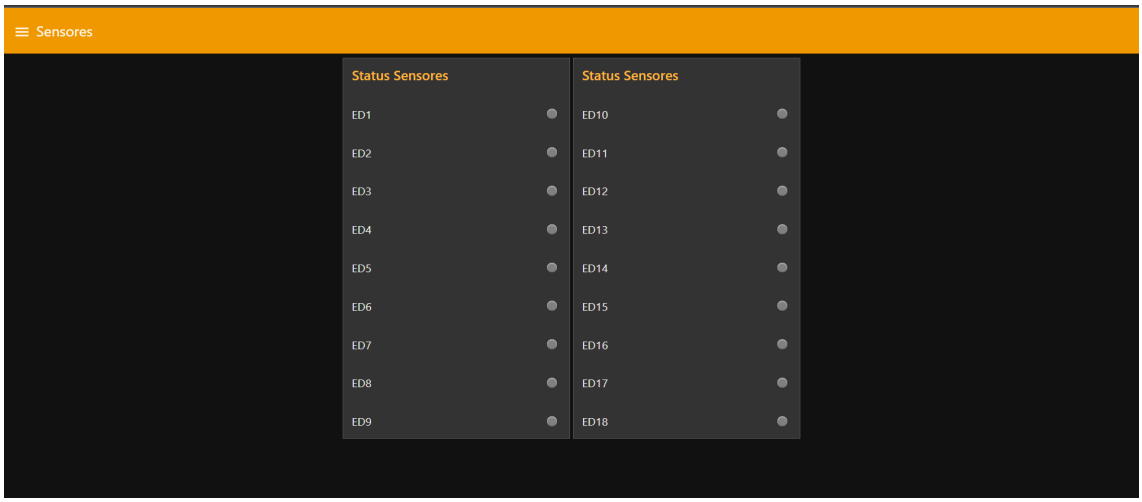
Fonte: Arquivo Pessoal

Figura 30 – Desabilita o Recurso



Fonte: Arquivo Pessoal

Figura 31 – *Status* dos sensores



Fonte: Arquivo Pessoal

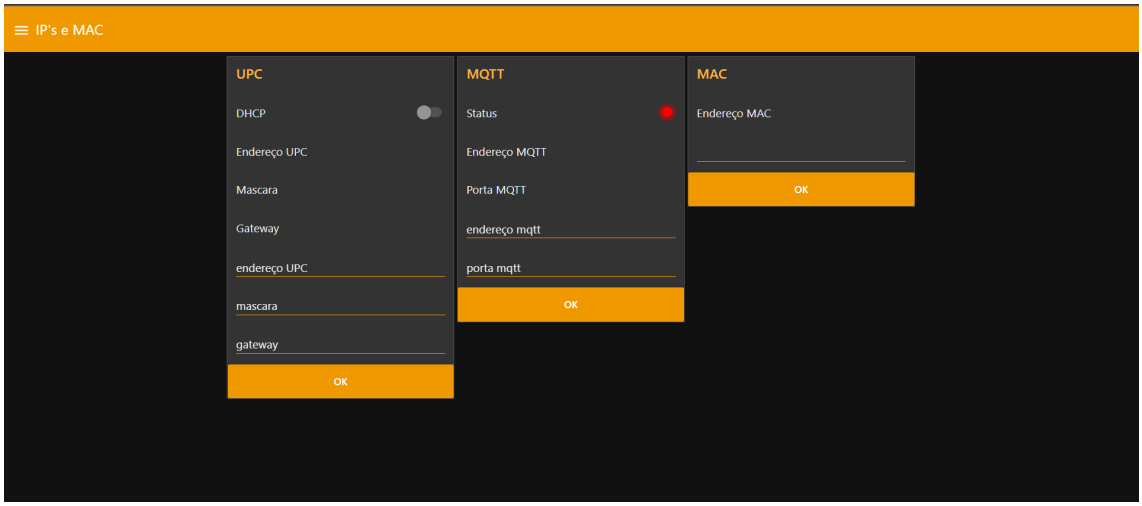
3.5 ESTADOS E SEQUÊNCIA DE AÇÕES DA UPC

Nesta seção, serão detalhadas as sequências de ações que a UPC pode tomar. As Figuras 34 e 35 apresentam o fluxograma de estados, que representa qual será o próximo estado que a UPC entrará após finalizar o seu atual, e o diagrama UML de sequência da UPC, respectivamente.

Ciclo Básico

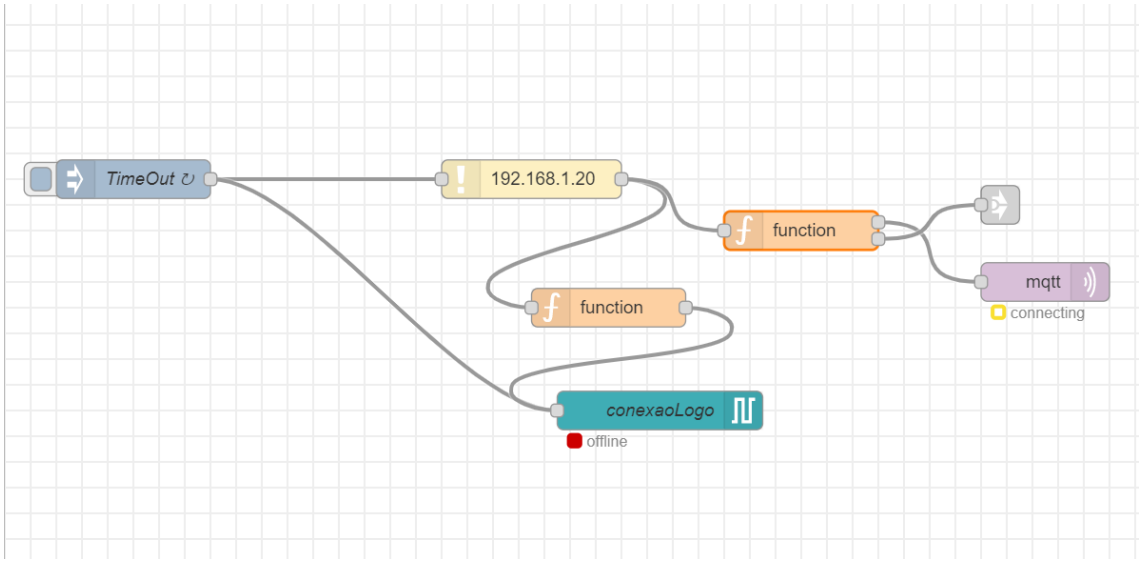
O estado 0 consiste na operação normal do sistema, considerando que já tem uma programação rodando na UPC. Como a UPC já tem uma OP em andamento e, imaginando que o recurso está ativo, este encontra-se no estado 0 da máquina de estados. Sendo assim, ele enviará a produção-lote conforme é atingida a quantidade

Figura 32 – IP's e MAC



Fonte: Arquivo Pessoal

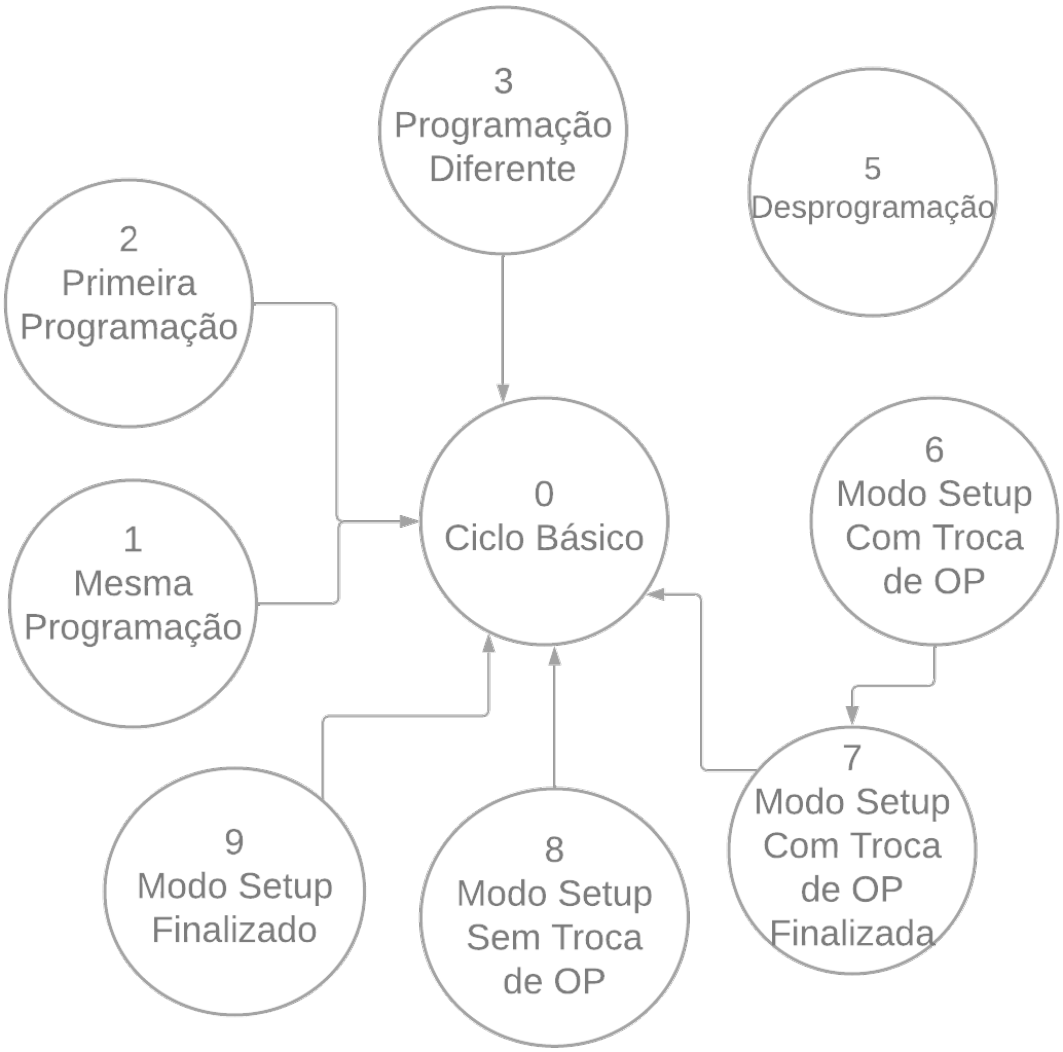
Figura 33 – *Flow* LWT para Logo



Fonte: Arquivo Pessoal

de produção correta. A produtividade é enviada de acordo com o tempo programado no parâmetro "segundos cálculo produtividade", originada do tópico programação, com a projeção relacionada ao minuto ou a hora, dependendo do parâmetro "tempo máquina", também configurado pelo tópico programação. Caso a máquina pare, a UPC contará 5 segundos e enviará o tópico máquina-parada com valor "S". Considerando que o recurso continue parado pelo tempo configurado na programação, será enviada a parada de início com a data e hora em que a máquina parou inicialmente. Quando a máquina retomar a produção, a UPC enviará o tópico máquina-parada com valor "N" e

Figura 34 – Fluxograma de Estados



Fonte: Arquivo Pessoal

a parada fim com a data e hora de início da parada e de fim.

Mesma Programação

Quando a mesma programação é novamente recebida na UPC entra-se no estado 1. Imaginando que a UPC tem uma programação rodando, caso a UPC receba um programa no qual os 5 primeiros parâmetros são iguais a que está no sistema, considera-a como uma programação da mesma OP. Entrando nesse estado, o sistema retornará o estado atual da máquina, reenviará o registro-equipamento e atualizará o restante dos parâmetros que vão na programação. Ao finalizar, o sistema retorna ao ciclo básico.

Primeira Programação

O estado 2 representa a programação inicial que um *subflow* recebe após ser desprogramado. Quando isso ocorre, o sistema salva a data em que o recurso foi configurado para ser usado na produção-lote, desbloqueia a contagem no CLP e envia o estado atual da máquina e do registro-equipamento. Por fim, retorna ao ciclo básico.

Programação Diferente

O sistema entra no estado 3 quando há uma troca de OP, ou seja, quando o sistema recebe um programa no qual um dos 5 primeiros parâmetros está diferente da programação atual. Ao entrar no estado 3, o sistema envia a produção remanescente, zera a contagem que estava sendo feita, checa se há uma parada em aberto e, se positivo, finaliza a parada. Além disso, o sistema envia o estado atual da máquina, o registro-equipamento e cria uma nova data de início para ser usada na produção-lote. Finalmente, a UPC retorna ao ciclo básico.

Desprogramação

É iniciado o estado 5 quando recebe-se uma desprogramação para um recurso que estava em funcionamento, a qual representa a desativação do recurso. Ao receber a desprogramação, o sistema bloqueia e zera a contagem do CLP, envia a produção-lote remanescente e, caso o recurso tenha uma parada ativa, esse finalizará a parada. O *subflow* só poderá ser utilizado novamente quando o sistema receber uma programação para aquele recurso.

Modo *Setup* Com Troca de OP

Os estados 6 e 7 estão conectados. O estado 6 representa o início do *setup*, que ocorre quando a programação contendo a mesma OP que está no sistema é recebida, porém com os parâmetros “troca OP” como “S” e “modo *setup*” como “1”. Já o estado 7 representa o final do *setup*, quando a programação com a nova OP é recebida. Quando recebe-se a programação indicando um *setup* com troca de OP, o sistema, primeiramente, bloqueia a contagem no CLP e salva a data em que o *setup* iniciou. Ao receber a programação com a nova OP, o sistema envia o evento “máquina-parada” com o valor de “T”, envia o registro-equipamento e inicializa uma parada com a data salva anteriormente e o motivo, obtido através da programação. Nesse estado, a contagem da produção, produtividade e do estado atual da máquina não é enviada para o Autoflex.

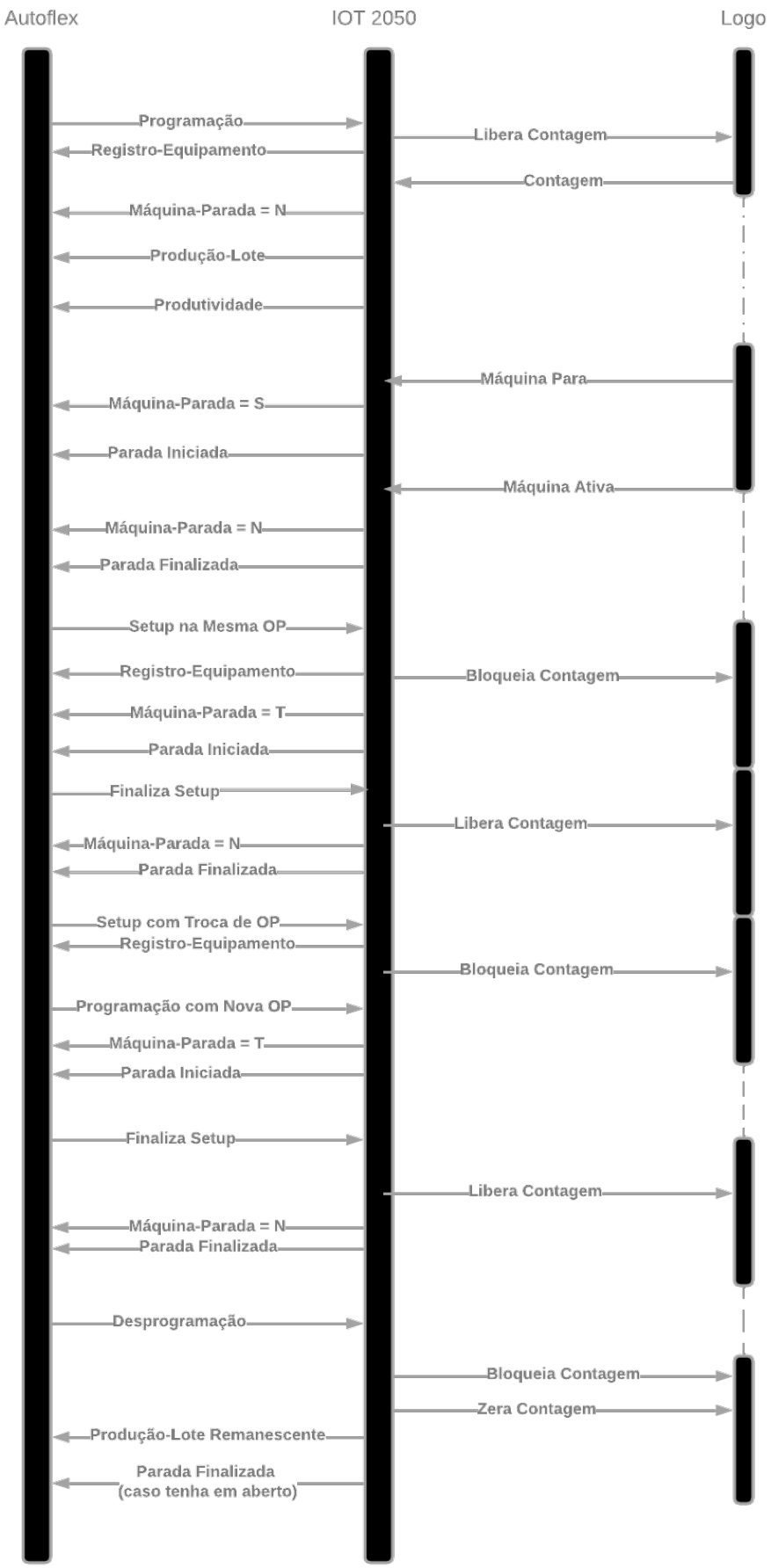
Modo *Setup* Sem Troca de OP

O sistema entra no estado 8 quando ocorre um *setup* na mesma OP em que o *subflow* está. Isso acontece quando a UPC recebe uma programação contendo o parâmetro “troca OP” com valor “N” e o “modo *setup*” como “1”. Com isso, o sistema envia a “máquina-parada” como “T”, envia o registro-equipamento, bloqueia a contagem no CLP e inicia uma parada com o motivo salvo do parâmetro “motivo setup”. Caso o sistema entre em *setup* e o recurso já esteja com uma parada ativa, a UPC reenvia aquela parada alterando apenas o campo “motivo parada”. Assim como nos estados 6 e 7, o sistema não envia mais produção, produtividade e o estado atual da máquina.

Modo *Setup* Finalizado

Quando o sistema recebe uma programação contendo o parâmetro “modo *setup*” com valor “2” o *subflow* entra no estado 9. Isso representa que o *setup* foi finalizado, sendo assim, a instância em questão finalizará a parada atual e retornará o estado atual da máquina. Caso a máquina esteja parada, uma nova parada será criada tendo a mesma data e hora de finalização do *setup*. Por fim, retorna-se ao estado 0.

Figura 35 – Diagrama UML de Sequência



4 RESULTADOS

A versão do projeto descrito neste documento é a 3.0, tendo sido aplicada, até o momento, para apenas um cliente. Foi possível conversar com os supervisores da empresa contratante, tendo sido realizado um questionário para entender o sentimento a respeito do produto e de suas melhorias.

1. **É perceptível uma mudança no tempo de produção?**

   Sim, com a automatização da contagem, apontamento de paradas, envios de produção e criação de etiquetas houve uma redução no tempo em que os operadores levam para fazer o empacotamento dos lotes. Por o sistema ser em tempo real, o supervisor não demorava mais um dia inteiro para receber o relatório da produção, ele conseguia ter acesso a esses dados quando necessário.

2. **Há confiabilidade no sistema e nos dados?**

   Sim, o sistema consegue informar o tempo exato em que houve paradas, mostra a produção correta das máquinas, os horários de início e fim de cada produção e cria uma rastreabilidade nos produtos que ajuda a encontrar problemas ocasionais.

3. **Quais alterações do produto impactaram mais?**

   O aumento de máquinas rodando simultaneamente, a possibilidade de utilizar a mesma UPC para diferentes áreas da produção e a possibilidade de configuração dos recursos e da UPC através do *web-server* foram as melhorias mais citadas, pois ajudaram na redução de manutenções solicitadas pelo cliente e trouxeram maior flexibilidade. Além disso, foi comentado sobre os parâmetros “segundos reativar produção” e “leituras reativar produção” que estes ajudaram a deixar mais fiel a contagem da produção. Também houve uma redução no tempo que a UPC levava para se ativar, isso ocorreu devido à utilização de *subflows* em vez de um *flow* por recurso, o que diminuiu o processamento do sistema.

4. **Qual a percepção sobre o sistema?**

   De maneira geral, os supervisores elogiaram o produto quanto à confiança no sistema, sua flexibilidade e a geração de produção automática. A redução do tempo para as informações chegarem às mãos dos supervisores foi fundamental para agilizar a obtenção de *insights* e possíveis atrasos e problemas que poderiam ocorrer. Um ponto negativo apresentado foi a necessidade de ter um computador conectado à UPC quando era necessário o acesso remoto para manutenção do produto. Isso está sendo estudado e atualizações futuras trarão uma conexão VPN direta com a UPC, descartando a necessidade do cliente para acesso remoto.

5. **Qual a percepção com a segurança do produto?**

   Pelo fato de todo cliente ter um *broker* MQTT rodando internamente na fábrica, a possibilidade de ataques externos não existe, pois toda a comunicação permanece no interior da fábrica. Com isso, houve uma aprovação dos clientes em relação à segurança do projeto. Após passar por extensivos testes, a contagem de produção também foi aprovada e os consumidores se sentem satisfeitos com o resultado final.

De maneira geral, o projeto foi bem aceito. Atualmente, a empresa conta com mais de 20 clientes com a versão 2.7 rodando e, grande parte das manutenções realizadas provêm de questões de instalação do produto e não de *software*. Com a versão 3.0 sendo bem sucedida, já estão sendo agendadas as atualizações nos clientes restantes.

5 CONCLUSÃO

Por ser um setor de pouco desenvolvimento tecnológico, a indústria de plástico precisava de um sistema que conseguisse trazer agilidade e confiabilidade para todas as etapas de criação de sacolas plásticas: extrusão, impressão e corte e solda. Com esse problema em mãos, a *Greylogix*, em parceria com a *Projedata*, desenvolveu o produto apresentado nesse documento, com o objetivo de trazer a automatização da coleta dos dados produzidos pelos setores e, através dos protocolos S7 e MQTT, os enviar para um sistema ERP.

Utilizando equipamentos *Siemens*, por serem robustos e bem testados, foi criada a unidade de controle e processamento (UPC), dentro da qual há um CLP para a captura dos sinais digitais produzidos pelas máquinas e um *gateway* IOT para processar os dados e envio de forma padronizada para o sistema ERP. A comunicação entre os equipamentos *Siemens* é feita através do protocolo S7 e, para se comunicar com o sistema da *Projedata*, o *Autoflex*, foi utilizado o protocolo MQTT.

Sendo um produto desenvolvido a partir de uma prova de conceito (POC), foram feitas diversas mudanças para que sua versão final fosse atingida, estando dentre elas: a alteração de um CLP mais robusto para um com menor robustez que atua junto com um *gateway* IOT, tendo cada um uma função específica no processo de captura dos dados; o aumento na quantidade de máquinas que podem ser utilizadas simultaneamente; a facilidade de um *web-server* para a configuração dos recursos, IP's do *broker* MQTT e da própria UPC; utilização de *subflows* para a redução do processamento e adição de funcionalidades solicitadas pelos clientes.

A partir dos *feedbacks* dos supervisores, foi possível concluir que o projeto foi um sucesso. Os objetivos propostos foram atingidos e houve a redução do tempo necessário para que os relatórios chegassem à parte mais alta da gerência. Além disso, foram trazidas maior confiabilidade para a contagem da produção e informações de parada e *setup*, criação de uma rastreabilidade do produto, bem como uma maior agilidade nas trocas de informações e a capacidade de ter 18 recursos rodando simultaneamente.

O projeto encontra-se na versão 3.0, entretanto, ainda há o que melhorar. Nas versões subsequentes o foco será desenvolver uma maneira de ter acesso remoto ao equipamento sem a necessidade de interação com o usuário, facilitando, assim, a manutenção e atualização de versão do código. Além disso, será estudada a possibilidade de adicionar funcionalidades que façam a UPC se comunicar com a máquina e mandar certos comandos para ela, tais como: desligar, aumentar ou reduzir a velocidade.

REFERÊNCIAS

BANKS, Andrew. **MQTT Version 5.0**. https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html, mar. 2019.

EXPORTAÇÃO, Brasia Comercial Imporação e. **Corte e Solda Fundo e Lateral 900**. 2022. Disponível em: https://www.brasialtda.com.br/produtos/corte-e-soldas/corte-e-solda-fundo-e-lateral-900.

FOUNDATION, OpenJS. **Node-RED Concepts**. https://nodered.org/docs/user-guide/conceptsnode, 2022.

HIVEMQ TEAM, he. Quality of Service (QoS) 0,1, 2 MQTT Essentials: Part 6. **https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels**/, 2015.

KLEINMANN, Amit; WOOL, Avishai. Accurate Modeling of the Siemens S7 SCADA Protocol for Intrusion Detection and Digital Forensics. **Journal of Digital Forensics, Security and Law**, v. 9, jan. 2014. DOI: 10.15394/jdfsl.2014.1169.

MAQUINAS, WV. **Impressora flexográfica para sacolas plasticas**. 2022. Disponível em: https://www.wvmaquinas.com.br/impressora-flexografica-sacolas-plasticas.

MARSHALL ROSE, Dwighat Cass. **RFC1006: ISO transport services on top of the TCP: Version 3**. https://dl.acm.org/doi/book/10.17487/RFC1006: RFC Editor, 1987.

MIRU, Gyorgy. **The Siemens S7 Communication - Part 1 General Structure**. Jan. 2016. Disponível em: http://gmiru.com/article/s7comm/.

MIRU, Gyorgy. **The Siemens S7 Communication - Part 2 Job Requests and Ack Data**. Jun. 2017. Disponível em: http://gmiru.com/article/s7comm-part2/.

MQ, IBM. **Qualities of service provided by an MQTT client**. https://www.ibm.com/docs/en/ibm-mq/8.0?topic=concepts-qualities-service-provided-by-mqtt-client, Setembro 2022.

PINTO, Pedro. MQTT: Protocolo de comunicação para pequenos dispositivos móveis. **https://pplware.sapo.pt/tutoriais/networking/mqtt-protocolo-de-comunicacao/**, 2019.

SIEMENS. **CPU-CPU Communicationwith SIMATIC Controllers**. https://cache.industry.siemens.com/dl/files/908/78028908/att_32073/v1/78028908_SIMATIC_Comm_DOKU_v21_e.pdf, set. 2022a.

SIEMENS. **LOGO!Soft Comfort Online Help**. https://cache.industry.siemens.com/dl/files/807/100782807/att_924632/v1/Help_en-US_enUS.pdf, jan. 2022b.

SIEMENS. **SIMATIC Manual de instruções do LOGO!** https://cache.industry.siemens.com/dl/files/461/16527461/att_82574/v1/Logo$_pt$.*pdf*, 2011.

THULUVA, Aparna; ANICIC, Darko; RUDOLPH, Sebastian; ADIKARI, Malintha. Semantic Node-RED for rapid development of interoperable industrial IoT applications. **Semantic Web**, v. 11, out. 2020. DOI: 10.3233/SW-200405.

VALERIE LAMPKIN, Weng Tat Leong. **Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry**. https://www.redbooks.ibm.com/redbooks/pdfs/sg248054.pdf: IBM Redbooks, 2012.

WIRESHARK. **S7 Communication (S7comm)**. Wireshark. Ago. 2020. Disponível em: https://wiki.wireshark.org/S7comm.