unesp UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
CAMPUS DE GUARATINGUETÁ

RAFAEL AUGUSTO CUSTODIO

Controle de Acesso utilizando Arduino, Banco de dados MySql e Labview

Guaratinguetá – SP
2015

Rafael Augusto Custodio

Controle de Acesso utilizando Arduino, Banco de dados MySql e Labview

Trabalho de Graduação apresentado ao Conselho de Curso de Graduação em Engenharia Elétrica da Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, como parte dos requisitos para obtenção do diploma de Graduação em Engenharia Elétrica.

Orientador (a): Prof. Dr. Francisco A. Lotufo

Guaratinguetá – SP
2015

Custodio, Rafael Augusto
C987c Controle de Acesso utilizando Arduino, banco de dados MySQL e LabVIEW. / Rafael Augusto Custodio – Guaratinguetá : [s.n], 2014.
79 f : il.
Bibliografia: f. 45

Trabalho de Graduação em Engenharia Elétrica – Universidade Estadual Paulista, Faculdade de Engenharia de Guaratinguetá, 2014.
Orientador: Prof. Dr. Francisco Antonio Lotufo

1. Arduino (controlador programável) 2. Controle de acesso 3. Controladores programáveis 4. MySQL (Recurso eletrônico) I. Título

CDU 621.381

unesp UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
CAMPUS DE GUARATINGUETÁ

Rafael Augusto Custodio

ESTE TRABALHO DE GRADUAÇÃO FOI JULGADO ADEQUADO COMO PARTE DO REQUISITO PARA A OBTENÇÃO DO DIPLOMA DE
"GRADUADO EM ENGENHARIA ELÉTRICA"

APROVADO EM SUA FORMA FINAL PELO CONSELHO DE CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA
Prof. Dr. Leonardo Mesquita
Coordenador

BANCA EXAMINADORA:

Prof. Dr. Francisco Antônio Lotufo
Orientador/UNESP-FEG

Prof. Dr. José Feliciano Adami
UNESP-FEG

Prof. Victor Gomes Soares de Souza
UNESP-FEG

Janeiro de 2015

AGRADECIMENTOS

Em primeiro lugar agradeço a Deus pela minha vida, minha inteligência, minha família e amigos. Agradeço por todos os desafios que colocou em meu caminho que me fizeram crescer e me desenvolver, como também por todas as vitórias que me ajudou a conquistar,

aos meus pais Angelo e Jocimara, minha irmã Naiara e também à minha namorada Camila, que apesar das dificuldades, nunca deixaram de acreditar e sempre me deram forças pra superar todos os obstáculos, sempre me incentivando a crescer,

ao meu orientador Prof. Francisco Antonio Lotufo que me auxiliou em todas as dificuldades que tive durante o desenvolvimento desse projeto, me incentivando, motivando e instruindo para levar o trabalho ao melhor resultado possível,

ao Prof. Victor Gomes Soares de Sousa pelas suas orientações de melhorias durante o desenvolvimento do projeto,

às repúblicas Dominacana e Jurupinga, onde fiz grandes amigos que sempre me apoiaram nas dificuldades que encontrei no decorrer do curso.

CUSTODIO, R. A. **Controle de acesso com cartão de radiofrequência.** 2015. 79 f. Trabalho de Graduação (Graduação em Engenharia Elétrica) – Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, Guaratinguetá, 2010.

Resumo

A busca por maior confiabilidade nos sistemas de segurança, assim como a gestão de informação desses sistemas está levando a um progresso crescente nos investimentos em novas tecnologias que permitam a implementação de equipamentos com um alto nível de confiabilidade, mas que também tenham um funcionamento ágil e prático. Isso levou as pessoas a voltarem cada vez mais o olhar para os sistemas de automação residencial, empresarial e industrial para a automatização e integração de seus sistemas.

A identificação por rádio frequência é muito difundida atualmente por garantir tanto agilidade na manipulação de dados de registros, quanto na confiabilidade de seu sistema de identificação, que está cada vez mais avançado e menos susceptível a fraudes.

Unida a essa tecnologia, a utilização de banco de dados é sempre muito importante para o armazenamento das informações coletadas, área em que a plataforma *MySQL* é muito utilizada.

Utilizando a plataforma de código aberto *Arduino* para a programação e manipulação do modulo RFID e o software *LabVIEW* para a união de todas essas tecnologias e também para desenvolver uma interface amigável ao usuário, é possível criar um controle de acesso de alta confiabilidade e agilidade de lugares com alta rotatividade de pessoas.

Esse projeto tem como finalidade provar as vantagens de se utilizar todas essas tecnologias trabalhando em conjunto, melhorando assim um sistema falho de segurança de maneira eficaz, ágil e barata.

PALAVRAS-CHAVE: Controle de acesso. RFID. MySQL. LabVIEW. Arduino.

CUSTODIO, R. A. **Access control with radio frequency card.** 2015. 79 f. Graduate Work (Graduate in Electrical Engineering) - Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, Guaratinguetá, 2010.

ABSTRACT

The search for more reliable security systems and information management of these systems is leading to a growing progress in new technologies investments that allow the implementation of equipment with a high level of reliability, but also have an agile and practical operation. This led people to turn increasingly looking for home automation systems, enterprise and industry for the automation and integration of their systems.

The identification by radio frequency is very widespread today for ensuring both agility in handling records data, the reliability of their identification systems, which are increasingly advanced and less susceptible to fraud.

Attached to this technology, the use of the database is always very important for the storage of information collected, the area where the MySQL platform is widely used.

Using the open source Arduino platform for programming and manipulation of RFID module and LabVIEW software for the union of all these technologies and to develop a user-friendly interface, you can create a highly reliable access control and agility places a high turnover of people.

This project aims to prove the advantages of using all these technologies working together, thus improving a flawed system effectively safety, cheaper and quicker.

KEYWORDS: Access Control. RFID. MySQL. LabVIEW. Arduino.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

RFID	*Radio Frequency Identification*
UNESP	Universidade Estadual Paulista "Júlio de Mesquita Filho"
IFF	*Identify Friend or Foe*
RADAR	*Radio Detection and Ragiong*
LF	*Low Frequency*
HF	*High Frequency*
UHF	*Ultra High Frequency*
MIT	*Massachusetts Institute of Technology*
EPC	*Eletronic Product Code*
EEPROM	*Eletrically Erasable Programmable Read-Only Memory*
PDA	*Personal Digital Assistant*
API	*Application Programming Interface*
PWM	*Pulse Width Modulation*
USART	*Universal Synchronous/Asynchronous Receiver-Transmitter*
SPI	*Single Point Injection*
UID	*Unique Identification*
DNS	*Domain Name System*
LED	*Light Emitting Diode*

SUMÁRIO

1 Introdução

A tecnologia RFID (*Radio Frequency Identification)* já existe a mais de 80 anos e nos dias de hoje vem sendo amplamente utilizada em várias camadas da cadeia produtiva social, assim como no cotidiano das pessoas. Essa tecnologia, apesar de não ser nova, vem sendo nos dias de hoje muito estudada em razão de suas diversas características positivas como a agilidade de funcionamento, seu dinamismo e diversas maneiras de garantir a segurança do sistema onde está sendo aplicada (Santini, 2008).

Em diversos casos o RFID é considerado como uma variação do código de barras, mas o RFID parece ser mais que uma simples evolução deste sistema de identificação já tão usado. O RFID vem para facilitar a vida das pessoas em seus diversos aspectos, como na segurança ao substituir a chave de casa e do carro, nos produtos utilizados como em livros e até nas roupas que vestimos, no controle de estoque industriais substituindo o código de barras de forma a ganhar maior agilidade e confiabilidade, dentre outros (Santini, 2008).

1.1 Motivação

Os métodos para garantir a segurança dentro e fora do campus de Guaratinguetá da UNESP sempre foi um assunto de bastante interesse tanto dos alunos quanto dos funcionários e docentes da mesma. O ano de 2012 marcou esse assunto quando casos de furtos contra alunos nos arredores do campus se tornou mais frequente e esses casos tornaram evidente a fragilidade na segurança dos alunos, que se tornou mais crítica quando passou a acontecer também dentro do campus, onde até então todos acreditavam estar razoavelmente seguros.

Algumas medidas foram tomadas para tentar aumentar a segurança como a instalação de câmeras no campus, o aumento de rondas policiais aos arredores em horário de maior movimento e também a impressão de cartões onde contêm os dados do carro e das pessoas que necessitam entrar e sair da universidade de automóvel diariamente. Essa última medida mostrou-se falha no objetivo de controlar a entrada e saída de veículos, já que, em razão do grande fluxo de automóveis, a maioria dos carros passa sem serem checados os cartões corretamente, sendo que alguns entram sem nem possuir o mesmo, além disso, não é feito nenhum registro de entrada e saída de automóveis, fato que aumentaria a confiabilidade do sistema, já que toda a movimentação de carros estaria registrada em um banco de dados.

O uso da tecnologia *RFID* já vem sendo usada em diversos estabelecimentos, como *shoppings*, indústrias, prédios residenciais e executivos, que necessitam um controle de acesso semelhante ao da Universidade. Essa tecnologia é interessante para o atual projeto, pois garante a agilidade de checagem de dados e liberação de entrada e saída, assim como uma confiabilidade de segurança e também um dinamismo, onde a partir dos dados coletados do cartão, é possível tomar inúmeras ações com a ajuda de outras plataformas.

1.2 Objetivo

O projeto desenvolvido tem como finalidade realizar o controle de acesso de carros na UNESP - Campus de Guaratinguetá-SP através da utilização de *tags* RFID. O programa permite registrar novos usuários, controlar o acesso de automóveis na Universidade, registrar dados de entrada e saída dos mesmos, assim como excluir registros ou usuários do sistema.

Para o funcionamento e programação do leitor RFID foi utilizado o Arduino, uma plataforma de codificação aberta e de fácil manipulação, pois sua programação é toda realizada em linguagem C++. O armazenamento de informações é feito pelo MySQL, banco de dados esse amplamente utilizado por possuir também uma programação bem simples, uma alta confiabilidade e também por ser muito dinâmico, permitindo a interação com diversos softwares.

Tanto a interface do programa, quanto a interação entre o Arduino e o banco de dados MySQL é realizada através do LabVIEW. Esse software nos permite a fácil manipulação de todos os recursos utilizados através de uma programação de blocos, gerando uma interface agradável para o operador do programa.

Assim, a ideia base é um maior controle no acesso à UNESP, em particular no Campus de Guaratinguetá, a fim de garantir uma segurança maior para os frequentadores da mesma.

2 Conceitos

2.1 Sistema RFID

Basicamente, o RFID é uma tecnologia que utiliza uma comunicação por radiofrequência, sem fios, para transmitir dados de um dispositivo móvel, como uma simples etiqueta ou um chaveiro (que aqui são chamados simplesmente de *tag*), para um leitor. As etiquetas RFID são hardwares que possuem uma antena e um chip envoltos por algum material, como vidro ou plástico, os quais respondem a sinais remotos de um leitor geralmente conectado a um computador (Santini, 2008).

Um sistema RFID é normalmente composto por dois componentes: as etiquetas, também chamadas de *tag* ou *transponders*, e um leitor. Podem ser divididos em duas categorias: Sistemas Ativos e Sistemas Passivos (Santini, 2008).

A utilização desta tecnologia é muito vasta, podendo ser amplamente estudada e implantada em diferentes setores, de Biblioteconomia a Veterinária; em um contêiner ou numa lata de refrigerantes. Tudo sendo monitorado por leitores e checado via rede, como por exemplo, através da internet. De maneira resumida, é um sistema que transmite dados de um objeto qualquer, através de um meio não guiado, usando ondas de rádio (Santini, 2008).

2.2 Dados Históricos RFID

Se hoje há tanta sofisticação na comunicação por radiofrequência, boa parte do avanço é devido a Sir Robert Alexander Watson-Watt, físico escocês que começou a estudar um método rápido de exibição de sinais de rádio a bordo dos aviões. Trabalhando numa estação meteorológica, tinha a intenção de saber a direção que se dirigia uma tempestade, tendo em vista que os aviões da época não possuíam proteções às atividades atmosféricas. Ainda em 1935, resultante de uma pesquisa do ministério da aviação inglesa de como detectar os aviões inimigos pela artilharia, procurando evitar o fogo amigo (surgimento do IFF - *Identify Friend Or Foe*), nasce o *Radio Detection and Ragiong* (RADAR) (Leal, 2008). Esse sistema baseia-se na reflexão de ondas eletromagnéticas de objetos distantes que permitem sua localização. O primeiro radar, apesar de não ter tido utilidade prática para a época, foi construído na Alemanha por C. Hülsmeyer, em 1904. Este equipamento foi considerado de difícil construção, baixa precisão e ineficiente localização de objetos (Santini, 2008).

O sistema de RADAR foi usado pelos britânicos durante a Segunda Guerra Mundial, pois previam com antecedência os ataques alemães e possuíam a capacidade de saber, com precisão, importantes dados, como a distância e velocidade dos bombardeiros inimigos. Este fato diminuiu muito o número de baixas civis, já que dava tempo para alertar a população a fim de que se protegesse. As potências do eixo, na mesma época, desenvolviam um projeto parecido, porém seu uso era para aumentar a precisão dos tiros (Santini, 2008).

Isso tudo preparou o terreno para de fato se trabalhar com a tecnologia RFID. Harry Stockman foi um dos primeiros, publicando em 1948 *Communications by Means of Reflected Power*, onde descreve a possibilidade do uso da potência refletida como meio de comunicação. Ocorrem na década de 50 os primeiros testes, um período de exploração do RFID, em laboratórios e pequenos dispositivos de rádio. Isso levou a um crescente avanço, sobretudo na década de 60, criando um ambiente propício para a explosão de desenvolvimento da década seguinte, quando várias entidades perceberam o potencial dessa tecnologia. Então começam também os primeiros registros de patentes, mostrando forte interesse comercial no assunto. Em 1973, Mario Cardullo registra a patente de uma etiqueta ativa de RFID e Charles Walton registra um transponder (*TAG*) passivo utilizado para trancas automáticas de portas de um automóvel e, ainda nessa década, surgem os primeiros modelos de identificação de animais por RFID (Leal, 2008).

Até o dado momento, as tags usadas eram as de baixa frequência (LF), 125 kHz, até que as empresas que comercializavam estes sistemas mudaram para os de alta frequência (HF), 13,56 MHz, a qual era irregular. Hoje, estes sistemas são usados em diversas aplicações, como nos controles de acesso e sistemas de pagamentos (Santini, 2008).

No começo dos anos 80, a IBM patenteou os sistemas de Frequência Ultra Alta (UHF), *Ultra High Frequency*, possibilitando que o RFID fizesse leituras a distâncias superiores a dez metros. Hoje, em consequência de problemas financeiros na década de 90, a IBM não é mais detentora da patente, que foi vendida para a Intermec, uma empresa provedora de códigos de barra. O grande crescimento do RFID UHF foi em 1999, quando o Uniform Code Council, EAN International, Protecter & Gable e Gillete fundaram o Auto-Id Center, no MIT, Massachusetts Institute of Technology, berço de vários outros avanços tecnológicos (Santini, 2008).

A pesquisa do Auto-ID Center era mudar a essência do RFID de um pequeno banco de dados móvel para um número de série, o que baixaria drasticamente os custos e transformaria o RFID em uma tecnologia de rede, ligando objetos à Internet através de *tags* (Santini, 2008).

Entre 1999 e 2003, o Auto-ID Center cresceu e ganhou apoio de mais de 100 companhias, além do Departamento de Defesa dos Estados Unidos. Nesta mesma época, foram abertos laboratórios em vários outros países, desenvolvendo dois protocolos de interferência aérea (Classe 1 e Classe 0), o EPC (*Eletronic Product Code*), o qual designa o esquema e a arquitetura de rede para a associação de RFID na Internet. Em 2003, o Auto-ID Center fechou e suas responsabilidades foram passadas para os Auto-ID Labs. Em 2004, a EPC ratificou uma segunda geração de padrões, melhorando o caminho para amplas adoções (Santini, 2008).

Essa tecnologia está em todo o lado, tão comum e rotineira que chega a passar despercebida pelo usuário. No setor empresarial o RFID já esta caminhando ao lado das aplicações de código de barras, ficando cada vez mais conhecida e utilizada, devido ao número de empresas que entraram (e ainda entram) nesse mercado. Essa necessidade advém de nossa sociedade globalizada, carente de informações urgentes e precisas, que podem ser saciadas pelo rastreamento RFID.

No Brasil o Sistema de Identificação, Rastreamento e Autenticação de Mercadorias, conhecido como "Brasil-ID", nasceu de um acordo realizado em 31 de Agosto de 2009 entre o Ministério da Ciência e Tecnologia, a Receita Federal e os Estados da União através de suas Secretarias de Fazenda (Jornal Acrítica, 2011). Esse sistema baseia-se no emprego da tecnologia de Identificação por Radiofrequência para realizar, dentro de um padrão único, a Identificação, Rastreamento e Autenticação de todo tipo de mercadoria em produção e circulação pelo país.

2.3 Elementos Principais Sistema RFID

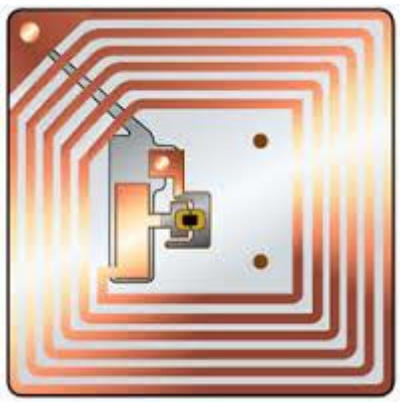
Devido a diversidade de dispositivos utilizados hoje na implementação do RFID em seus diversos setores, seria impossível listar todos os dispositivos necessários. Cada empresa tem suas especificações e preferências para o uso da mesma, e assim, possuí seus softwares e hardwares específicos. Apresenta-se então nessa seção os dispositivos fundamentais que fazem parte de um sistema RFID.

2.3.1 *Transponder* (*tags)*

A palavra *transponder* é um acrônimo para *TRANSmitter/ resPONDER,* porque sua função é transmitir e responder comandos que chegam por radiofrequência. O *transponder*,

RF *tag* ou simplesmente *tag*, é a etiqueta RFID em si. Sua estrutura básica é bem simples: um chip capaz de armazenar informações e uma resistência fazendo o papel de antena, envoltos por algum material como plástico ou silicone, em um determinado formato (chaveiro, etiqueta, cartões, entre outros). O propósito de uma *tag* RFID é a de, fisicamente, anexar dados sobre um objeto (Santini, 2008). A figura 1 apresenta esta *tag*.

Figura 1 – *Tag* RFID



Fonte: (https://electrosome.com).

As *tags* podem ser divididas em três grandes grupos: passivos, ativos e via dupla.

2.3.1.1 *Tag* Passiva

Os transponders passivos não possuem bateria ou qualquer outra fonte de alimentação interna para seu funcionamento, portanto, passam a maior parte do tempo adormecidas. A *tag* desse tipo aproveita a potência fornecida pelo leitor, através da comunicação, e alimenta seus circuitos a fim de transmitir as informações armazenadas. Por esse fato, possuem constituição muito simples e um número reduzido de elementos (SYBASE,2006).

As vantagens desse tipo de *tag* são o tamanho reduzido, o custo reduzido em relação as *tags* ativas assim como sua vida útil longa (superior a 20 anos em condições adequadas de funcionamento) pelo fato de não precisar estar enviando dados a todo o momento e sem necessidade de nenhum tipo de manutenção.

Suas desvantagens estão relacionadas ao fato desse tipo de *tag* não possuir uma fonte de alimentação própria, ou seja, não possui bateria. Por esse motivo estas etiquetas dependem 100% de energização proveniente dos leitores e isto leva a necessidade de ser carregada com energia do leitor para somente depois responder aos comandos enviados por ele. O alcance de

leitura é então limitado devido à quantidade de energia que pode ser absorvida das ondas de RF enviadas pelo leitor. Em determinada distância da antena do leitor, a etiqueta não consegue coletar energia suficiente para ligar o circuito integrado e então fica impossibilitada a comunicação com o leitor.

Essas tag*s* podem possuir, além do identificador, uma memória não volátil EEPROM (*eletrically erasable programmable read-only memory*) para armazenamento de dados. Essa característica é somente para tag*s* passivas de Classe 2, pois as anteriores (Classe 0 e 1) dispunham apenas de número de identificação.

2.3.1.2 *Tag* Ativa

As tag*s* ativas diferem basicamente pelo fato de possuírem uma bateria. Podem possuir um circuito de rádio que lhes permite transmitir o próprio sinal para o leitor, podem ser de leitura e escrita, maior capacidade de memória, tolerância a ruídos e perdas de sinais e uma velocidade de 100 a 200 bytes por segundo de transferência de dados (SYBASE,2006).

Esse tipo de *tag* tem como vantagem o alcance que oferece. Pode ficar ativa continuamente e necessita de pouca potência para se comunicar. Sua memória varia de acordo com a aplicação em que será utilizada. Pelo fato de possuírem alimentação própria, possuem funções adicionais às *tags* passivas.

Sua desvantagem é o maior tamanho em relação as *tags* passivas, pelo fato da necessidade de uma bateria interna. Vida útil sem manutenção reduzida, levando em conta a durabilidade de sua bateria, podendo variar de acordo com a utilização, pois quanto maior o tempo de utilização, menor a vida útil. Possui também um custo elevado.

2.3.1.3 *Tag* Semi-Ativa

As etiquetas semi-ativas possuem baterias como fonte de energia para os circuitos do *microchip*, porém não possuem transmissor de rádio. Portanto, para realizar a comunicação com leitores elas necessitam da energização proveniente deles. A energização do circuito integrado permite a modulação do sinal refletido. A vantagem deste tipo de etiqueta é que não se torna necessário energizar a etiqueta a partir do leitor e assim sendo, podem ser utilizados leitores de potências menores, além do que, uma maior quantidade de dados pode ser armazenada nestas etiquetas.

Etiquetas semi-ativas permitem maiores alcances de leitura em relação às etiquetas passivas, além de possibilitar o acoplamento a sistemas de sensores de ambiente tais como de temperatura, umidade e pressão. Neste tipo de aplicação, os objetos aos quais as etiquetas estão acopladas podem ser monitorados com regularidade. As desvantagens deste tipo de etiqueta são maior custo, tamanhos maiores e vida útil limitada devido à vida útil das baterias. Se a etiqueta é muito estimulada a operar, a durabilidade dela diminui consideravelmente com o tempo.

2.3.2 Antena e Leitor (*Transceiver*)

A antena em um sistema RFID tem como função transformar a energia eletromagnética guiada pela linha de transmissão em energia eletromagnética irradiada e vice-versa. O tipo de antena utilizada, assim como o seu *layout*, são fatores determinantes para o alcance das *tags*, tal qual seu funcionamento e tipo de sistema.

Um leitor, em um sistema RFID tem como medição de desempenho o fato de comunicar-se com as *tags* através da antena e repassar a informação para o software (Santini, 2008).

Os layouts para antena e leitor podem ser de vários tipos, como tipo túnel, portal, portátil, *Smart Shelves*, entre outras, sendo definidos através do tipo de aplicação em que será utilizado o sistema RFID.

2.3.3 Controlador

Outro componente físico é o Controlador, dispositivo responsável por controlar o leitor. Os Controladores podem variar em complexidade, sendo desde um pequeno leitor embarcado em um *palmtop* ou celular até um microcomputador com sistema servidor e várias funcionalidades (Santini, 2008).

2.3.4 Componentes Lógicos

Os componentes lógicos tem a função de fazer todo o controle do sistema RFID. Eles agem desde a comunicação do leitor com a antena até o software instalado no terminal que recebe as informações.

Dentro dos componentes lógicos estão a API (*Application Programming Interface*) e o *middleware.*

A função da API é criar um conjunto de rotinas e padrões para estender as funcionalidades de um sistema, permitindo agregar valor e recurso ao sistema com finalidade de ter o desempenho designado no início do projeto. O API é o que permite que outras aplicações comuniquem-se com o leitor; tem como principal função transformar informações vindas do *middleware* para as *tags* e vice-versa. É na API que são requisitados inventários, monitorada as atividades ou configurados ajustes no leitor (Santini, 2008).

Deve-se, também, controlar a comunicação no subsistema de comunicações. Este é responsável por selecionar o protocolo de comunicação com o *middleware*, como *Ethernet, Bluetooth,* serial ou algum outro do tipo proprietário (Santini, 2008).

O *middleware* é um *software* mediador. Ele é o software responsável por pegar as informações vindas do leitor ou do gerenciador de eventos e transferi-las para um sistema gerenciador de produtos ou um software de controle de estoques ou vendas por exemplo. O *middleware* permite que o programador mova informações entre o sistema RFID e seu banco de dados, sem se preocupar com diferenças de protocolo de comunicação, interfaces de baixo nível, dentre outros. (Santini, 2008).

2.4 Arduino

O Arduino é uma plataforma, de placa única, de prototipagem eletrônica de código aberto que utiliza em sua programação essencialmente a linguagem C/C++. Ele possuí um microcontrolador ATMEGA 328P embarcado que funciona como o coração da plataforma.

Seu ponto forte é o custo acessível para projetistas sem verba para investir em um controlador mais sofisticado e a sua versatilidade e fácil manipulação, razão essa por possuir entradas e saídas analógicas e digitais, EEPROM, porta PWM, porta serial USART, comparador analógico, interrupção externa, porta SPI, software de utilização simples, entre outros recursos para facilitar a vida do programador (www.arduino.cc).

2.5 LabVIEW

O software de projeto gráfico de sistemas NI LabVIEW é a base da plataforma da National Instruments. Com o LabVIEW, você tem um conjunto abrangente de ferramentas que lhe permite desenvolver qualquer aplicação de medição ou controle em muito menos

tempo. O LabVIEW é o ambiente de desenvolvimento ideal para criar inovações, fazer descobertas e obter resultados com maior rapidez (http://brasil.ni.com/).

O LabVIEW é uma ferramenta de controle bastante funcional, permitindo uma fácil programação em blocos com interação entre diversos tipos de softwares e plataformas. Possui uma interface bastante amigável para o usuário, onde é possível organizar os dados de entrada e saída do projeto da melhor forma para a visualização e manipulação destes.

No projeto, foi utilizado LabVIEW versão 12.0, que funciona como o *middleware* do sistema. É pra ele que todas as informações vindas da API são enviadas e é feita então a interação entre a ação desejada pelo usuário, o banco de dados e os fatores externos, como por exemplo a cancela a ser gerenciado seu acionamento.

2.6 MySQL

O MySQL é o banco de dados de código aberto mais popular nos dias de hoje. É um software muito confiável, fácil de ser manipulado e com uma enorme gama de softwares em que é possível fazer interação.

A versão utilizada foi o MySQL 6.1, e é nele que todas as informações do sistema são armazenadas.

3 Desenvolvimento

Nesse capítulo é mostrada toda a parte de programação do sistema, em seus diferentes níveis, detalhadamente.

3.1 Programação do Arduino

O leitor RFID utilizado foi o MFRC522 de 13,56MHz. O motivo dessa escolha é o atendimento a todos os requisitos do sistema, o custo e a facilidade de ser encontrado no mercado brasileiro. Apesar de, no projeto, ser utilizado apenas a função de leitura do módulo, é possível também utilizar a função de escrita em cartões RFID e essas manipulações são possíveis quando a *tag* se encontra a uma distância de no máx 50mm.

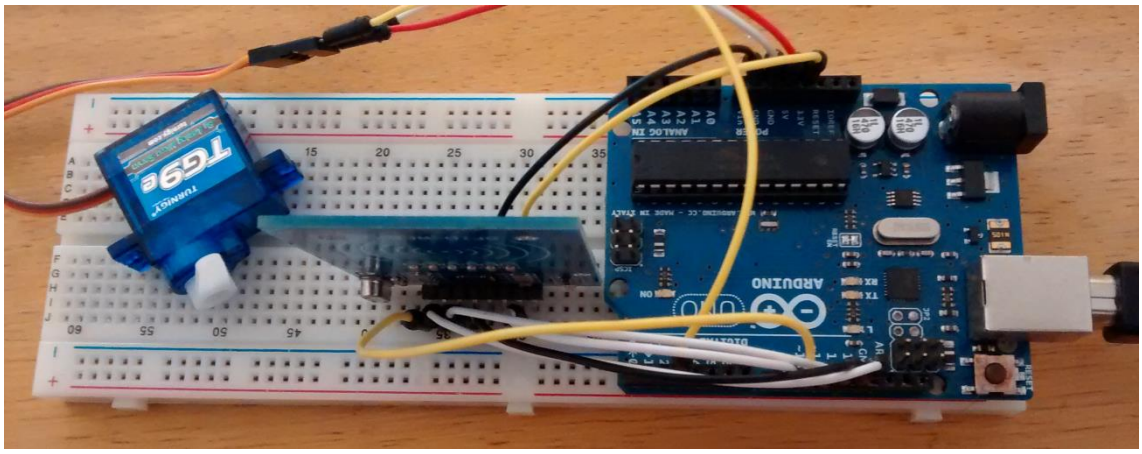
O projeto é desenvolvido para a utilização de *tags* passivas devido a aplicabilidade onde não necessita de leitura a grandes distâncias nem de grande espaço de armazenamento nas *tags* e também devido a sua vida útil ser muito maior, durando toda a vida universitária do estudante quando essa *tag* é mantida em condições favoráveis.

Todo o tratamento de dados é realizado via software *LabVIEW* o que resulta em uma programação do Arduino bastante simples. A comunicação entre o controlador e o *middleware* é feita de forma serial que, apesar de não ser a melhor em termos de velocidade de processamento de dados e distância de comunicação, é bastante prática e de baixo custo para o desenvolvimento do projeto inicial.

No projeto, foi utilizado o Arduino UNO, que funciona como o controlador do sistema RFID. É ele que gerencia todas as rotinas do leitor, que possuí o API do sistema, assim como permite a comunicação entre o *middleware* e o leitor RFID.

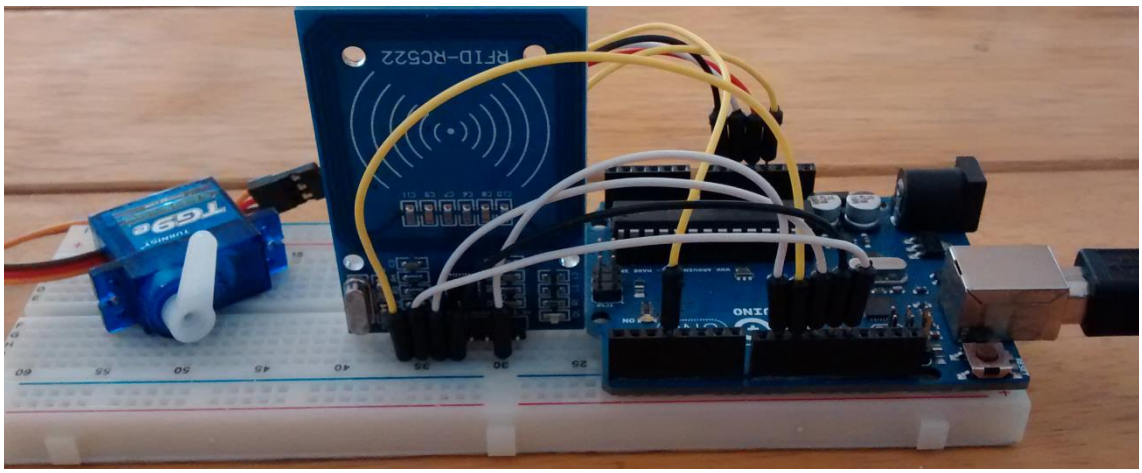
O *layout* do leitor RFID junto com o Arduino é mostrada nas figuras 2 e 3.

Figura 2 – Montagem Arduino e Leitor RFID (vista superior)



Fonte: Autoria própria.

Figura 3 – Montagem Arduino e Leitor RFID (vista frontal)



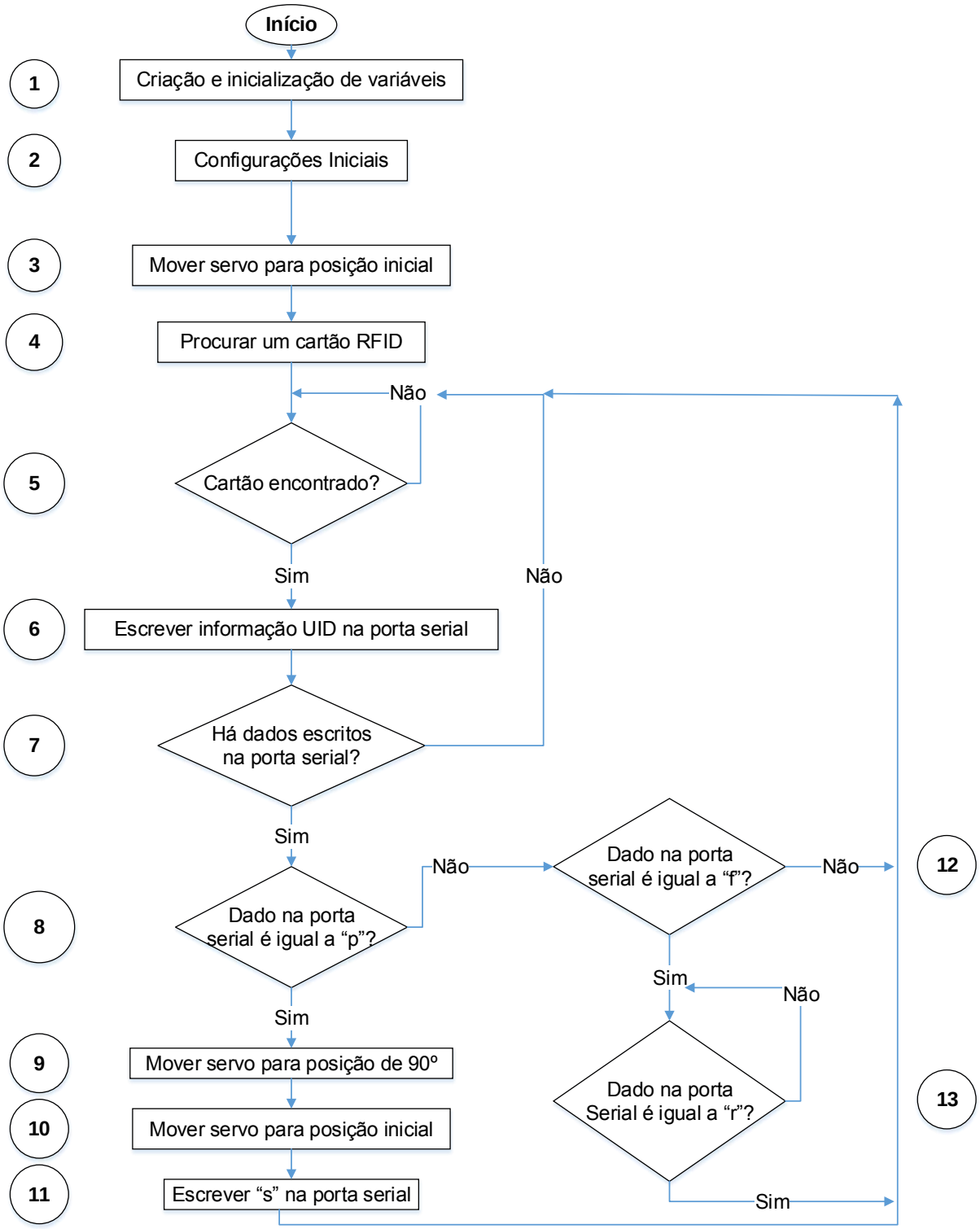
Fonte: Autoria própria.

A biblioteca para a utilização e programação do MFRC522 foi obtida através do site https://github.com/miguelbalboa/rfid. Essa biblioteca nos permite programar toda a parte de gerenciamento de rotinas dentro do Arduino, como também desenvolver o API do sistema RFID.

No apêndice 5 encontra-se todas as etapas de programação do Arduino para o desenvolvimento do projeto.

A figura 4 representa a explicação do funcionamento do programa escrito no Arduino através de um fluxograma.

Figura 4 – Fluxograma Programação em Arduino



Fonte: Autoria própria.

Nota-se que as etapas do programa estão numeradas tanto na programação quanto no fluxograma. Assim, a explicação passo a passo do funcionamento do mesmo pode ser observada a seguir.

1. São iniciadas as variáveis que serão utilizadas tanto para guardar informações essenciais como também as variáveis de apoio.
2. Inicialização da porta serial a 9600bps, inicialização do barramento SPI, inicialização do modulo Mifare RC522, inicialização do servo motor no pino 3.
3. Posicionamento do servo-motor em 0º.
4. O modulo aguarda o posicionamento de um novo cartão a 50mm do leitor.
5. Leitura do UID do novo cartão encontrado.
6. Escrita do UID do novo cartão na porta serial.
7. Verificação da existência de dados escritos na porta serial.
8. Verifica se os dado na porta serial é igual a “p”, que é escrito na porta serial pelo LabVIEW quando o cartão posicionado é válido, ou seja, está cadastrado no bando de dados do sistema.
9. Move o servo motor para a posição de 90º.
10. Move o servo motor para a posição de 0º.
11. Escreve “s” na porta serial, saindo assim do laço.
12. Verifica se o dado na porta serial é igual a “f”. Esta função é necessária quando o cartão posicionado não esta registrado no banco de dados. Caso isso ocorra, o funcionamento do programa só é liberado após o operador ler o aviso e pressionar “OK”, onde então o LabVIEW escreve “r” na porta serial.
13. Verifica se o dado na porta serial é igual a “r”.

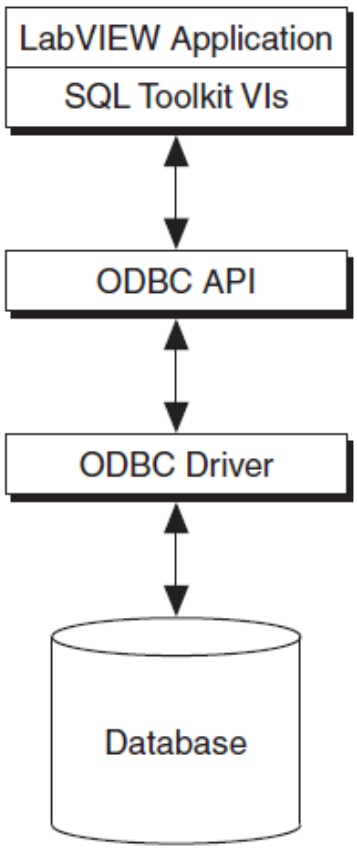
Em nível de hardware, o programa fica em funcionamento ininterruptamente a partir de sua inicialização, lendo a todo o momento qualquer cartão posicionado no leitor, só interrompendo a operação quando o programa é encerrado.

3.2 Banco de dados

Para o armazenamento de registros de entrada e saída e também de usuários cadastrados, utilizou-se o banco de dados MySQL.

Para abrir conexão entre Labview e MySQL é necessário criar um conector ODBC, onde no projeto atual utilizou-se o MySQL ODBC 5.3. Este conector é um padrão de comunicação que permite acessar o banco de dados utilizando as ferramentas certas via software. Esse método consiste em definições em vários níveis da API, um pacote padrão de *drivers,* uma implementação SQL baseada na norma ANSI SQL e um meio para definição e manutenção do DNS (*Domain Name System*). Um DNS é uma forma de rapidamente referenciar um banco de dados específico. Esse DNS tem que ser especificado com um nome único, e isso é feito pelo driver ODBC. É necessário ser definido um único DNS para cada banco de dados em que se deseja trabalhar. Essa comunicação entre MySQL e LabVIEW é feita através do *SQL Toolkit VIs*. Essa comunicação é feita da seguinte forma: O *SQL Toolkit VIs* chama o Microsoft API para ODBC. Então o ODBC se comunica com os drivers específicos do banco de dados que traduzem o comando para o nível de linguagem inferior do banco de dados (*Database Connectivity Toolset User Manual*). Esse processo é mostrado na figura 5.

Figura 5 – Comunicação LabVIEW e MySQL



Fonte: *Database Connectivity Toolset User Manual*. 2001.

Foi criado banco de dados “portaria” e dentro desse banco de dados criou-se a tabela “usuário” e a tabela “registros”, onde toda a edição dessas tabelas foi feita via “*MySQL Workbench 6.1 CE*”. As figuras 6 e 7 mostram os detalhes dessas tabelas.

Figura 6 – Tabela Registros

Field	Type	Null	Key	Default	Extra
DATA	date	NO		NULL	
DATAHORA	datetime	NO		NULL	
ID	varcha...	NO		NULL	
RA	int(6)	YES		NULL	
NOME	tinytext	NO		NULL	
IDADE	int(2)	NO		NULL	
RG	int(9)	NO		NULL	
CARRO	tinytext	NO		NULL	
PLACA	varcha...	NO		NULL	

Fonte: Autoria própria.

A tabela “registros” é usada apenas para armazenar dados de entrada e saída de automóveis na portaria da universidade. Nessa tabela são registradas tanto a data e horário de entrada e saída, como os dados da pessoa que realizou as ações.

Figura 7 – Tabela Usuários

Field	Type	Null	Key	Default	Extra
ID	varcha...	NO	PRI	NULL	
RA	int(6)	YES		NULL	
NOME	tinytext	NO		NULL	
IDADE	int(2)	NO		NULL	
RG	int(9)	NO		NULL	
CARRO	tinytext	NO		NULL	
PLACA	varcha...	NO		NULL	

Fonte: Autoria própria.

A tabela “usuário” é usada para armazenar os dados das pessoas e seus respectivos carros que terão acesso permitido na Universidade.

3.3 LabVIEW

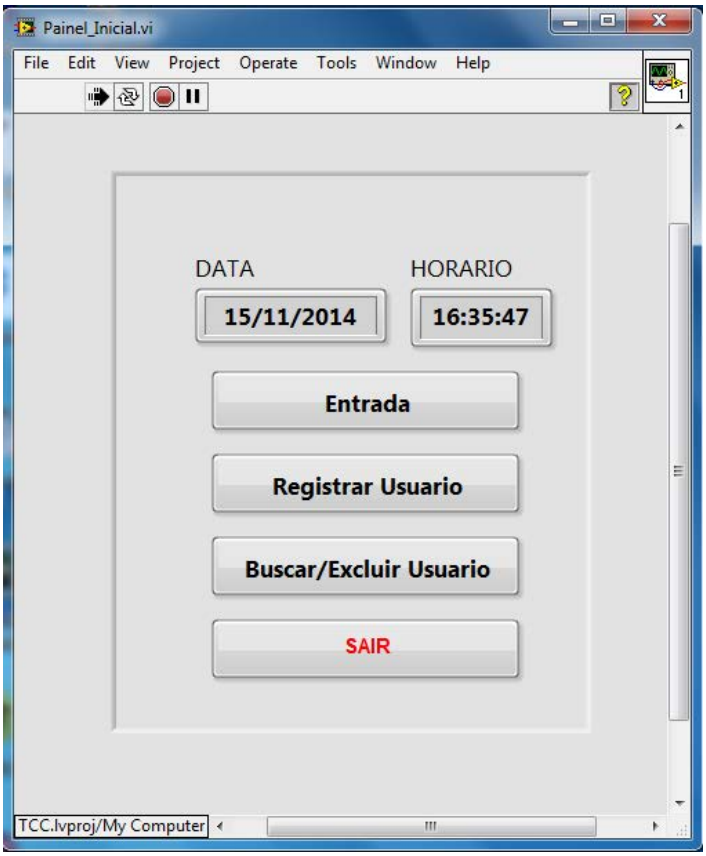
O software LabVIEW funciona como o coração do programa. É nele que se faz todo o tratamento de dados, assim como a comunicação com o Arduino e o MySQL. Toda a parte de programação é mostrada em figuras contidas no apêndice.

No próximo item é explicado o passo-a-passo de toda a programação realizada em LabVIEW e suas interfaces gráficas.

3.3.1 VI Painel Inicial

O painel inicial do programa é o primeiro a ser executado. Nele o usuário escolhe que tipo de ação entre as três possíveis deseja realizar, sendo essas opções a de entrada na portaria, registro de novos usuários, busca ou exclusão de dados e sair. A imagem da interface inicial do programa é mostrada na figura 8.

Figura 8 – Interface Painel Inicial



Fonte: Autoria própria.

Ao se escolher uma das opções do painel, abre-se uma nova janela onde é possível executar a ação escolhida. A data e horário são também mostrados nesse painel, sendo variáveis de fundamental importância no registro de dados de entrada e saída da portaria.

Não é possível executar duas ações simultaneamente, necessitando assim primeiro sair da opção que está em execução para depois escolher uma outra ação.

Em todas as etapas do projeto buscou-se desenvolver um programa com interface amigável para o usuário e de fácil manipulação. Também foi priorizado o tratamento de todos os erros que possam vir a surgir durante a operação do programa, para evitar transtornos na sua aplicação, já que o programa é destinado a segurança da Universidade. A seguir é mostrado a programação em blocos para o funcionamento desse painel. A primeira etapa da programação é mostrada na figura 22, no apêndice 1 .

Para garantir a sequencia de transmissão de dados desse programa, utilizou-se a *Flat sequence structure.* O primeiro bloco dessa sequência contem um *While Loop*, fazendo com que só saia dessa etapa após pressionar o botão "SAIR".

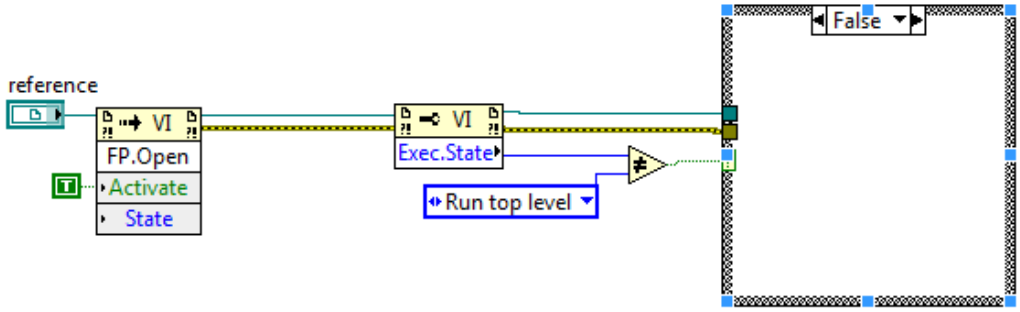
O marcador de horas e data do painel inicial foi feito utilizado a ferramenta *get date/time in seconds*  e para a modelagem dos dados disponibilizados por essa ferramenta, utilizou-se a *format date/time string*  , retornando os dados em formato string.

Dentro do *While Loop* existe um *Event Structure* que é utilizado para o funcionamento dos botões de opção. Ao pressionar um dos botões, muda o estado do *Case Structure* do mesmo para *True* fazendo com que se inicialize o VI dessa opção. A inicialização do VI da opção pressionada é feita através de uma Sub VI nomeada por VI_Salto 

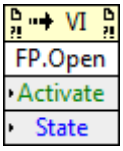
. A figura 23, no apêndice 1 mostra os *Case Structure´s* em estado *True.*

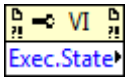
A figura 9 mostra a programação de blocos da Sub VI Salto onde o *Case Structure* está em estado *False.*

Figura 9 – Sub VI Salto_1



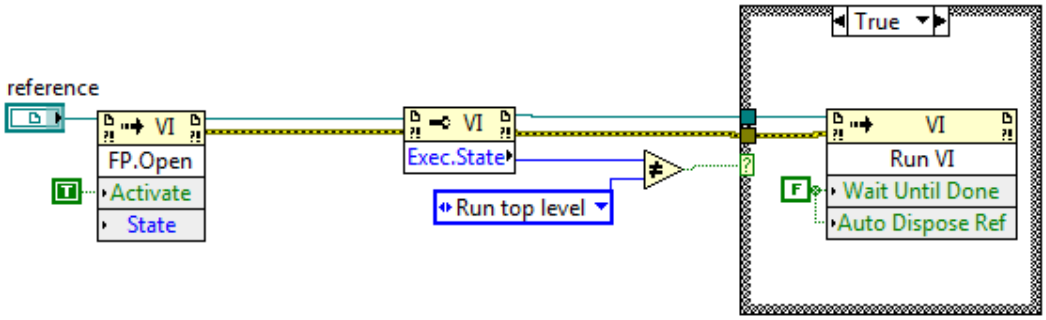
Fonte: Autoria própria.

Nessa etapa de programação, primeiro, a partir de um *Invoke Method: Front Panel: Open* 

é aberto o VI da opção que foi selecionada. Logo após, é feita uma comparação através de um *Property Node* 

fazendo com que, se a janela do VI não está no topo das janelas, ou seja, sobreposta sobre todas as janelas em execução, então o estado do *Case Structure* vai para *True*, como é mostrado na figura 10.

Figura 10 – Sub VI Salto_2



Fonte: Autoria própria.

Dentro do estado *True* desse *Case Structure* existe um *Invoke Method: Run VI* que faz com que o VI seja executado sobreposto a todas as janelas em execução.

A figura 24, no apêndice 1 mostra outro estado do *Event Structure*, que tem finalidade de parar a execução do *While Loop* caso a janela seja fechada.

A figura 25, no apêndice 1 mostra o estado do *Event Structure* para a funcionalidade do botão SAIR. Ao ser pressionado, a execução *While Loop* presente deve ser interrompida. Esse evento acontece quando o botão SAIR muda de estado ou valor.

Quando o Botão SAIR é pressionado, é necessário que além de encerrar a execução do *While Loop,* a janela também seja fechada. Isso acontece a partir da segunda etapa da *Flat Sequence*. Quando pressionado o botão SAIR, o *While Loop* é interrompido, passando assim automaticamente para a segunda etapa do *Flat Sequence.* Nessa etapa, um *Invoke Method:Front Panel:Close* faz com que o *VI* em execução seja fechada.

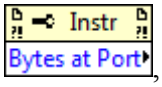
3.3.2 VI Entrada Portaria

O VI Entrada Portaria faz o controle de acesso na portaria. Nesse VI, quando uma pessoa solicita acesso de um carro na Universidade aproximando sua *tag* RFID do leitor, o programa faz a verificação se o usuário está registrado no sistema ou não. Se o usuário está corretamente cadastrado, a cancela é aberta e é registrado a data e horário que o usuário entrou. Se o indivíduo não estiver registrado no sistema, a cancela não abre e o programa gera um aviso para o operador de que a *tag* não está registrada no banco de dados do sistema. A figura 11 mostra a interface do VI Entrada.

Figura 11 – Interface VI Entrada Portaria

Fonte: Autoria própria.

Assim como no VI anterior e nos próximos, usou-se um *Flat Sequence* que, após o botão SAIR ser pressionado, o programa encerre sua execução e a janela seja fechada em seguida. A primeira etapa dessa sequência, etapa essa que ocorrerá nos outros VIs, é a inicialização da porta serial especificando em qual porta está ligada a plataforma Arduino junto com o módulo leitor RFID, utilizando o *VISA Configure Serial Port*  . Também é necessário abrir conexão com o banco de dados, com os valores padrões e fechando as antigas referências, a partir de um arquivo DNS criado, utilizando o *DB Tools Open Conection*  .

Após serem inicializadas a porta serial e a conexão com o banco de dados, abre-se um *While Loop,* e dentro desse ocorrerá a maior parte do andamento do programa. O primeiro passo é a verificação se há bytes escritos na porta serial usando um *Property Node*  , dados esses esperados ser o UID da tag aproximada ao leitor. O estado do *Case Structure,* que aparece em seguida, depende da existência ou não de dados na porta serial.

O LED que aparece na interface é mantido apagado nessa etapa, é aplicado um delay necessário para a verificação da porta serial e também adquirido a hora e data atual do mesmo modo que se fez no VI Painel Inicial.

As figuras 26, 27 e 28, no apêndice 2 mostram o programa no estado em que não há dados escritos na porta serial.

Após ser realizada a ação do *Case Structure,* independente do estado em que se encontra, a tabela registros é recarregada para assim mostrar os dados atualizados da mesma.

Se existe um *UID* escrito na porta serial, o estado do *Case Structure* vai para *true*, sendo assim realizada a verificação se o código está registrado no sistema para a liberação ou não da entrada do veículo na Universidade. O diagrama de blocos desse estado é mostrado nas figuras 29, 30 e 31, no apêndice 2.

Nesse estado do *Case Structure,* é utilizado outro *Flat Sequence* para garantir o andamento sequencial da programação realizada. A primeira etapa dessa sequência faz a leitura da *UID* escrita na porta serial através do *VISA Read*  e em seguida essa informação é atualizada em um bloco de notas, já que esse bloco de notas irá substituir o bloco de notas já existente para tal função. Em seguida, na próxima etapa da sequência, essa informação do bloco de notas é carregada em uma variável local ID. A manipulação dessas informações no bloco de notas é feita através do *Write to Text File*  e do *Read from Text*

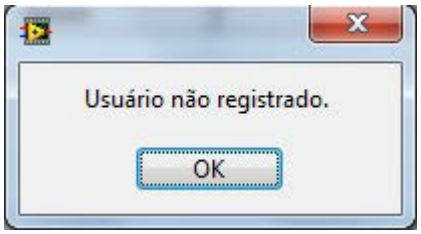
File  . Essa forma de armazenamento e leitura dos dados de entrada é utilizada em todos os *VIs* do programa.

Seguindo para a próxima etapa, é feito a verificação se a informação de entrada escrita no bloco de notas está registrada no sistema. Primeiramente, o *Format Into String*  escreve no *Tools Execute Query*  o comando "select NOME from usuário where ID= "%s";", onde, nesse caso, "%s" é o dado de entrada da ferramenta, ou seja, a variável global ID*,* que nada mais é que selecionar o elemento contido na coluna NOME da tabela usuario onde o elemento da coluna ID é igual aos dados salvos na variável local ID. O comando "%s" é usado quando se usa o *Format Into String*, sendo "%s" substituído pelas entradas (*input)* da ferramenta. Portanto*,* através do *Tools Fetch Element Data*  esse dado é comparado com uma string vazia. Se não houver dados na coluna NOME, então o usuário que solicitou entrada na Universidade não está registrado no sistema, já que essa coluna deve sempre ser preenchida no registro de um novo *UID* na tabela USUARIO*,* e o estado da próxima *Case Structure* continua em *True* como mostrado nas figuras 29, 30 e 31, no apêndice 2. Neste caso, um erro esperado é gerado em razão da ausência de dados tanto na coluna NOME quanto na coluna *ID*, não afetando o funcionamento do programa. Então esse erro é ignorado pelo programa através do *General Error Handler*  .

Caso contrário, se houver dados escritos na coluna NOME o usuário está registrado no sistema e sua entrada na Universidade deve ser então permitida fazendo com que o estado da *Case Structure* mude para *False,* como mostrado na figura 32 e 33, no apêndice 2.

Quando o *Case Structure* está em seu estado *True,* é escrito em todos os indicadores de dados de entrada do painel a mensagem: "Não Registrado". Dentro desse estado existe também uma *Flat Sequence.* Na primeira etapa dessa sequência, é escrito na porta serial a palavra "f", que faz com que o programa no *Arduino* entre em um loop infinito, como mostrado na etapa 12 e 13 da figura 4. Em seguida, na próxima etapa da sequência, aparece a seguinte mensagem para quem está manipulando o programa: "Usuário não registrado", como mostrado na figura 12, só passando para a próxima etapa da sequência após o operador pressionar o botão "OK".

Figura 12 – Mensagem 1

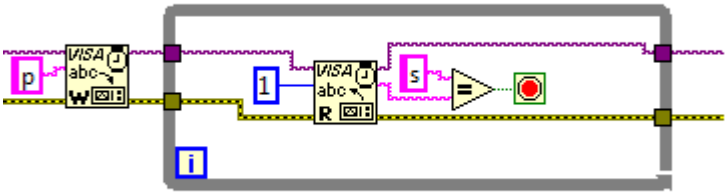


Fonte: Autoria própria.

Na terceira etapa da sequência, é escrito na porta serial a palavra “r”, que por sua vez, faz com que o programa no *Arduino* saia de seu loop infinito dando sequência no programa, assim como em *LabVIEW*, o programa sai do *Case Structure* e da continuidade no andamento.

Se a *tag* aproximada no leitor RFID estiver registrada no sistema, então o *Case Structure* vai para o estado *False.* Nesse estado, primeiramente é carregado todos os dados da tabela USUARIO referentes ao UID da *tag,* através do comando “*select * from usuario where ID= "%s"*;”. Em seguida, os dados de cada coluna são carregados individualmente para posteriormente serem inseridos na tabela registro. Então, através do *Tools Create Parametrized Query*  é enviado o comando “INSERT INTO registros (ID, DATA, DATAHORA, RA, NOME, IDADE, RG, CARRO, PLACA) VALUES(?,?,?,?,?,?,?,?,?);”, onde cada parâmetro, através do *Tools Set Parameter Value*  , é inserido na tabela REGISTROS, juntamente com a data e hora em que a liberação de entrada ou saida foi solicitada. O uso dos pontos de interrogação (?) no comando tem a função de definir o local onde o *Tools Set Parameter Value* irá inserir cada informação. Após isso, o comando é executado pelo *Tools Execute Query*  e as referências são finalizadas através do *Tools Free Object*  . Paralelo a isso, é escrito na porta serial o valor “p” que é o sinal para o *Arduino* acionar o servo motor que abre a cancela. Após ser escrito o valor “p”, é iniciando um *While Loop* e a execução deste só é finalizada após o *Arduino* enviar o valor “s” para a porta serial que significa que a cancela já fechou, como mostrado nos passos 8, 9, 10 e 11 da figura 4. Esse passo, que seria um intertravamento do sistema, é mostrado em programação de blocos do LabVIEW na figura 13 abaixo.

Figura 13 - Intertravamento



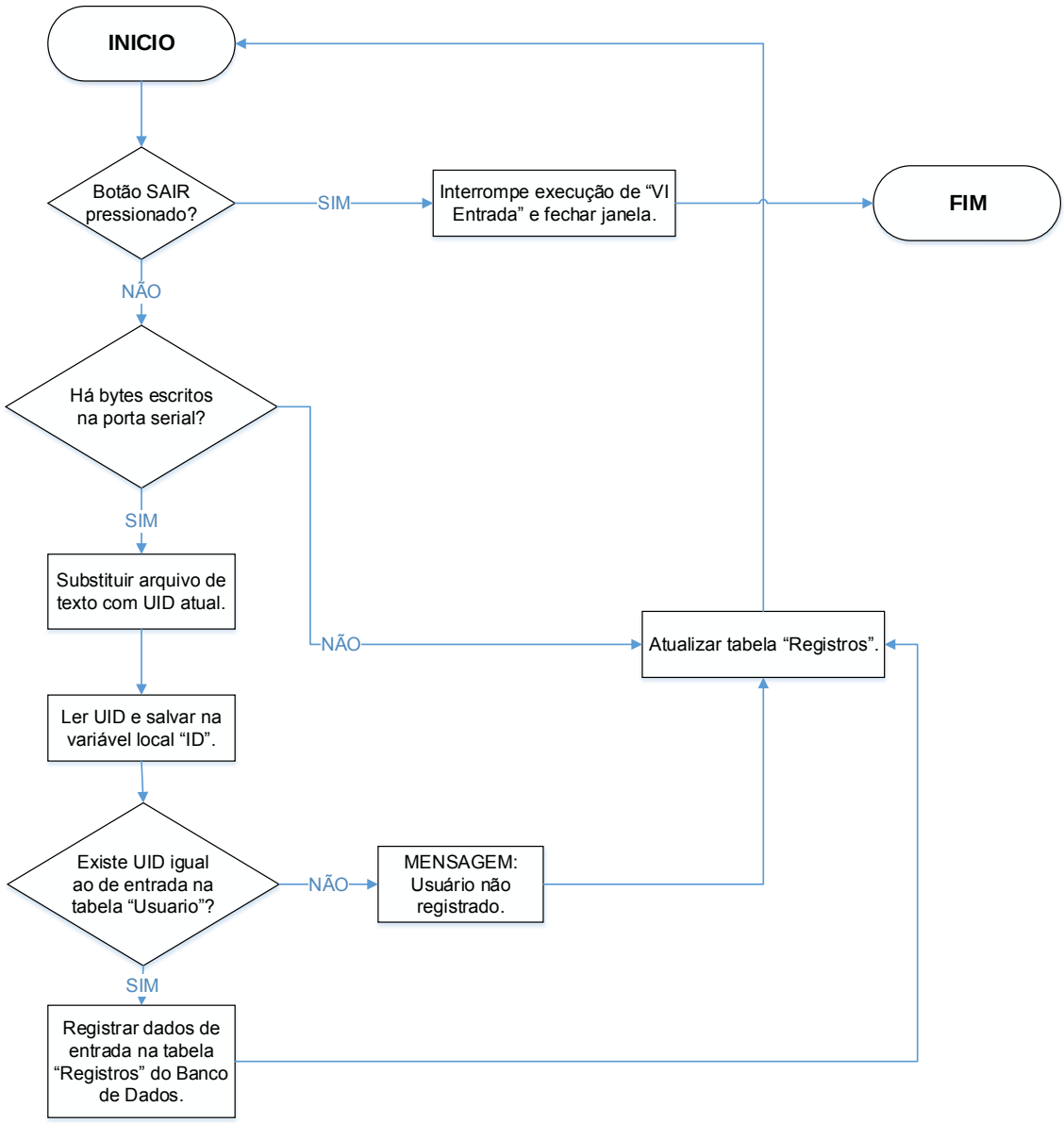
Fonte: Autoria própria.

Além disso, o led de aviso que a *tag* está registrada no sistema é acionada, ficando na cor verde enquanto o *Case Structure* está em execução.

Essas são as condições da programação de blocos do VI Entrada. Quando acionado o botão SAIR o *While Loop* principal é finalizado, são encerradas as referências tanto da conectividade da porta serial quanto do banco de dados, e passa para segunda etapa da *Flat Sequence* que fecha o VI Entrada. Após encerrada as referências, é utilizado o *Clear Errors* para ser eliminado qualquer erro encontrado na execução do *VI.*

O funcionamento do *VI Entrada* é mostrado na figura 14.

Figura 14 – Fluxograma VI Entrada Portaria

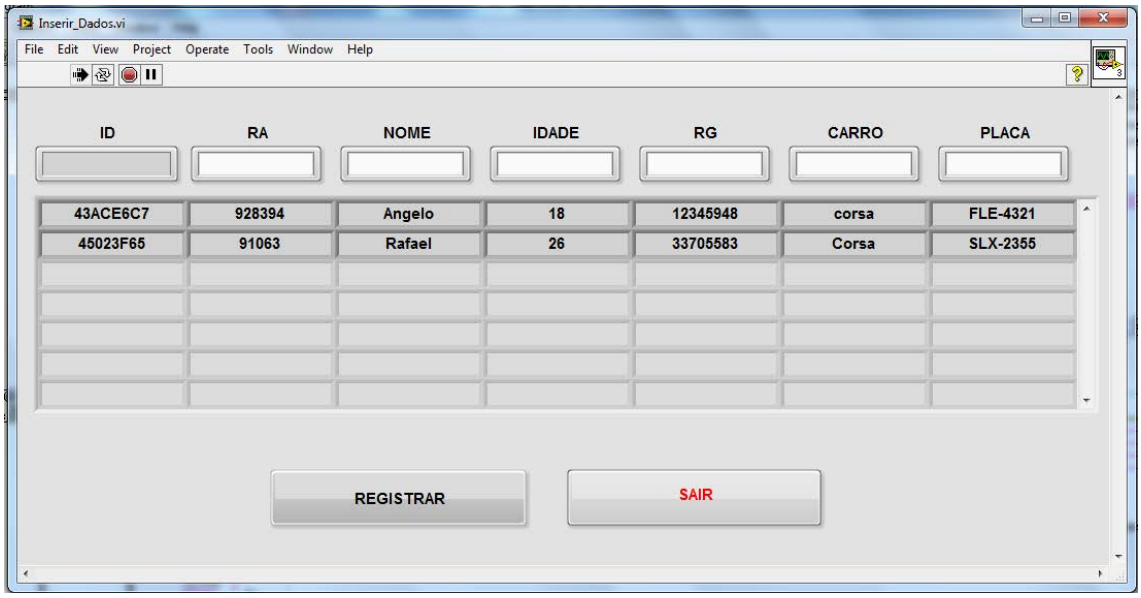


Fonte: Autoria própria.

3.3.3 VI Inserir Dados

O VI Inserir Dados tem a função de registrar novos usuários no banco de dados do sistema para a liberação da entrada e saída dos mesmos. Nele é armazenado todos os dados do usuário referente ao *UID* da *tag* que a pessoa irá utilizar. A figura 15 mostra a interface desse VI.

Figura 15 – Interface VI Inserir Dados



Fonte: Autoria própria.

Nela, a pessoa que está administrando o programa entra com os dados do novo usuário a ser registrado. Nenhum campo deve ficar sem ser preenchido, caso contrário o programa não permite o registro. A descrição da tabela Usuário no MySQL é mostrada na figura 7.

Quando é solicitado o registro, o programa faz uma verificação da existência dos principais parâmetros – ID, RG, RA e Placa – no sistema. Caso algum desses três parâmetros já existirem, então também não é permitido o registro do usuário.

Agora é detalhada toda a parte da programação em blocos do programa em LabView. As figuras 34, 35 e 36, no apêndice 3 mostram o VI Inserir Dados em seu modo de espera, ou seja, seu estado inicial.

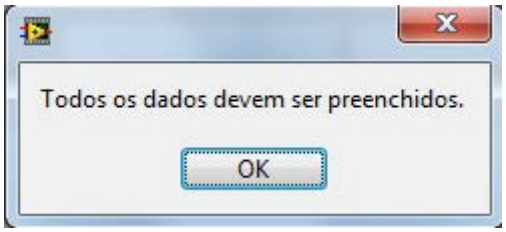
Primeiramente, é estabelecida conexão com o banco de dados portaria no MySQL. Se o botão registrar não é pressionado, o LabVIEW se mantém nesse estado inicial indefinidamente, saindo dele apenas se o botão SAIR for pressionado. Nesse estado a única ação do programa é carregar a tabela Usuario a cada ciclo.

Se o botão registrar é pressionado, então começam as verificações se os dados a serem inseridos são válidos, mudando o *Case Structure* para *True,* como mostra as figuras 37 e 38, no apêndice 3.

A primeira verificação a ser feita é se existe algum campo sem ser preenchido, através de uma verificação utilizando portas *OR* . Se houver ausência de dados em algum dos parâmetros, o próximo *Case Structure* vai para o estado *True* e é mostrada a mensagem

“Todos os dados devem ser preenchidos” para o operador do programa, como mostrado na figura 16, não registrando os dados de entrada.

Figura 16 – Mensagem 2

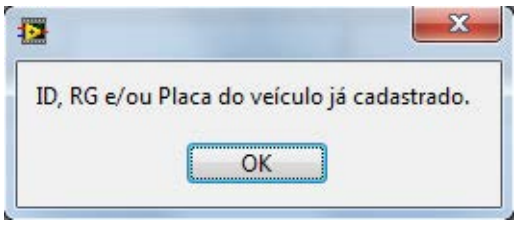


Fonte: Autoria própria.

Caso contrário, o *Case Structure* vai para *False,* onde é feita a verificação se já existe dados armazenados no banco de dados iguais aos principais dados de entrada. A figura 38 e 39, no apêndice 3 mostram essa etapa da programação.

Primeiro, é inserido o código “select ID, RA, RG, PLACA from usuario;” no banco de dados, que pede para retornar todos os dados contidos nos parâmetros ID, RA, RG e Placa da tabela Usuario. Esses dados são reorganizados em um vetor, para a melhor manipulação, utilizando o *Reshape Array*  . Em seguida é feito a busca de cada parâmetro de entrada nesse vetor, utilizando o *Search 1D Array*  . Essa ferramenta retorna o valor *-1* se o dado de busca não é encontrado. Assim, faz-se uma somatória da saída dessa ferramenta nos quatro parâmetros conferidos, se essa soma retornar um valor diferente de *-4* significa que um ou mais parâmetros principais já existem na tabela usuário, fazendo o próximo *Case Structure* ir para *False.* Nesse estado, a mensagem “ID, RG, RA e/ou Placa já cadastrado” é mostrada para o operador do programa, como mostrado na figura 17, e o registro desses dados não é executado. A programação dessa etapa é mostrada na figura 40, no apêndice 3.

Figura 17 – Mensagem 3



Fonte: Autoria própria.

Caso contrário, o próximo *Case Structure* vai para *True*, realizando o registro dos dados de entrada na tabela Usuario. O diagrama de blocos para esse estado é mostrado na figura 41, no apêndice 3.

Quando finalmente chega-se nessa etapa do programa, todas as verificações de dados inválidos foram realizadas, podendo então ser realizado o registro dos dados de entrada. Esse registro é realizado parâmetro por parâmetro através do comando "*INSERT INTO* usuario(ID, RA, NOME, IDADE, RG, CARRO, PLACA) VALUES(?,?,?,?,?,?,?);" que é executado pelo *Tools Create Parametrized Query*  e logo após carregado cada dado pelo *Tools Set Parameter Value*  para ser executado pelo *Tools Execute Query*  .

Esses são todos os passos para realizar a inserção de dados na tabela Usuario. A programação foi feita de forma a garantir que nenhum dado inválido seja inserido na tabela e assim evitar fraudes ou erros.

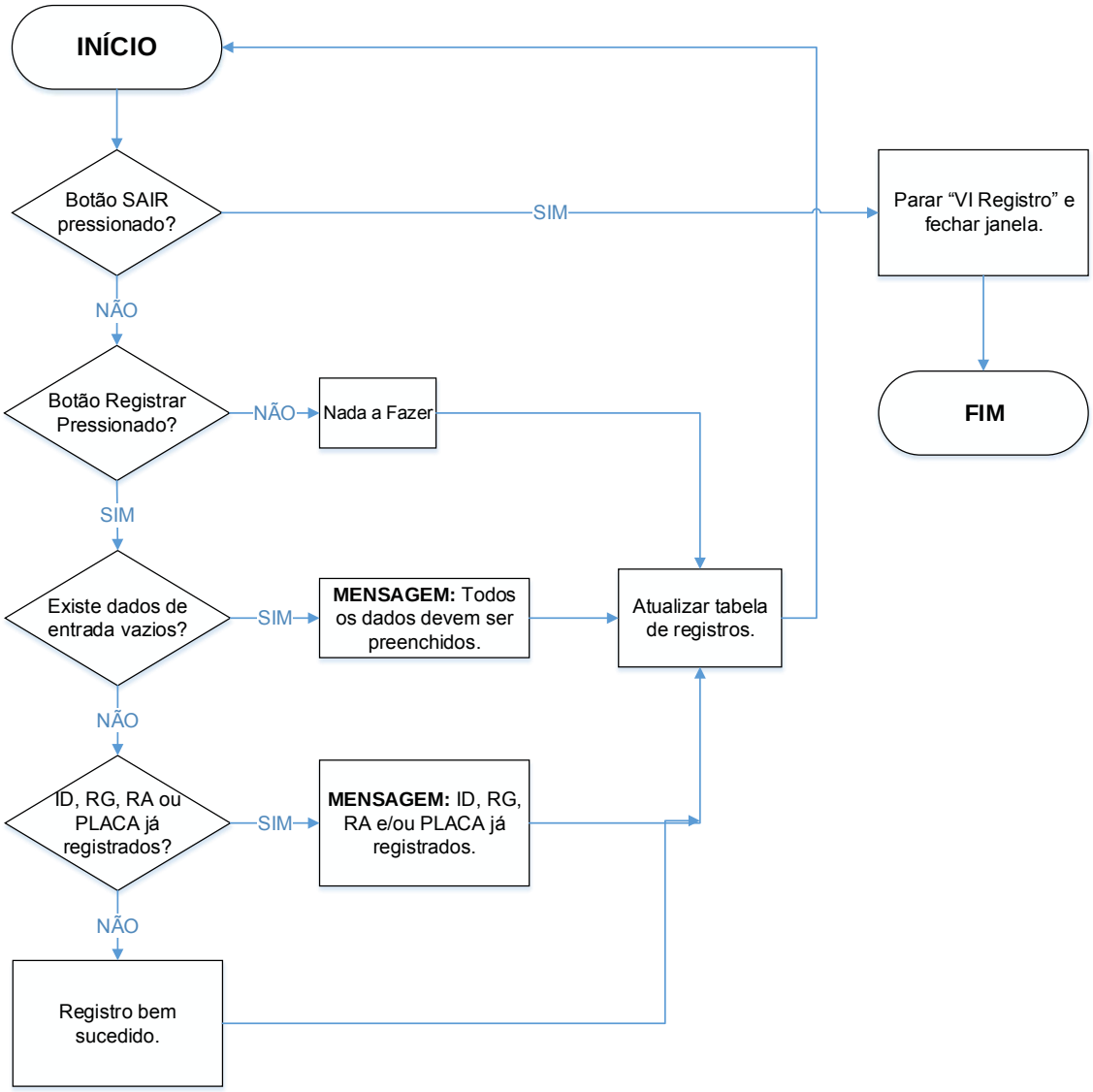
Uma outra etapa que existe ainda nesse VI é para ler o UID da *tag* a ser registrada. Essa etapa é mostrada através de sua programação em LabVIEW na figura 42, no apêndice 3.

Primeiro é verificado se existe bytes escritos na porta serial. Se não houver, o *Case Structure* permanece em *False* e o programa não realiza nenhuma ação. Caso haja bytes na porta serial, então o *Case Structure* vai para *True* como mostrado na figura 43, no apêndice 3.

Essa etapa é idêntica a programação de todos os outros VIs em que se precisa realizar a leitura da UID de alguma *tag*. Primeiro salva-se a UID em um bloco de notas que substitui outro já existente com dados antigos e após isso lê-se essa informação, salvando-a em uma variável local chamada ID.

O fluxograma para a explicação do funcionamento desse VI é mostrado a seguir.

Figura 18 – Fluxograma VI Inserir Dados

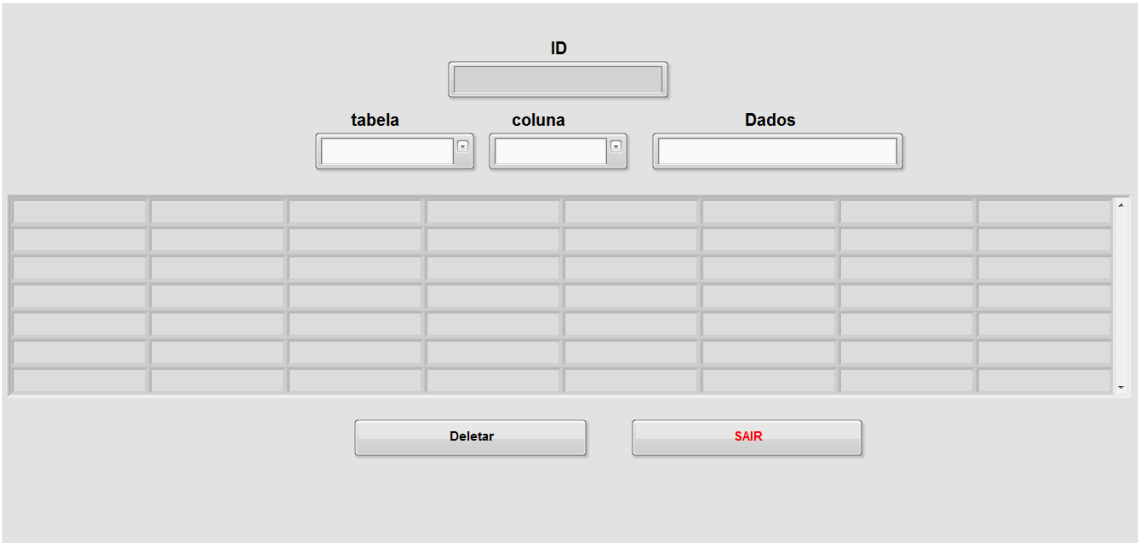


Fonte: Autoria própria.

3.3.4 VI Busca

O *VI Busca* é responsável pela pesquisa de dados tanto na tabela Registros quanto na tabela Usuario. Nele também é possível fazer a exclusão de dados das tabelas. A sua interface é mostrada na figura 19.

Figura 19 – Interface VI Busca



Fonte: Autoria própria.

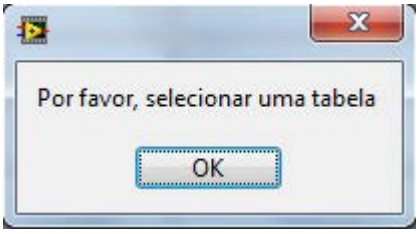
Primeiro passo é escolher a tabela a ser feita a busca. Se os campos Coluna e Dados são deixados em branco é mostrado todos os dados presentes na tabela escolhida. Ao escolher a opção Deletar nessa situação, o programa entende que o operador do programa deseja excluir todos os dados presentes na tabela. Pode-se também escolher dados específicos a serem buscados, como também fazer a exclusão dos mesmos.

As figuras 44 e 45, no apêndice 4 mostram a etapa inicial da programação em bloco desses VIs.

Primeiro o programa faz a verificação se o campo a ser inserido o nome da tabela desejada encontra-se vazio. Se sim, então o *Case Structure* seguinte é levado para o estado *True.* Neste, se o botão Deletar não é pressionado, nada é feito. Caso o mesmo seja pressionado, o *Case Structure* a ele ligado irá para o estado *True* como mostrado na figura 46 do apêndice 4.

Este, em seu estado *True*, escreve a mensagem "Por favor, selecionar uma tabela" ao operador do programa, como mostra a figura 20.

Figura 20 – Mensagem 4



Fonte: Autoria própria.

Se houver dados escritos no campo *Tabela,* então o *Case Structure* vai para *False,* como mostrado nas figuras 47 e 48, no apêndice 4.

Agora é feita a verificação se há dados escritos nos campos Coluna e Dados. Se for verificada a ausência de dados em um dos dois campos, o *Case Structure* seguinte vai para *True.* Neste, é necessário conferir se foi selecionado a tabela Registros. Se sim, então é necessário que seja selecionado todos os campos da tabela para serem mostrados na interface com exceção do campo Data da tabela, pois esse campo é utilizado apenas quando se quer buscar por uma data específica, não sendo necessário ser carregado na interface, já que suas informações já são mostradas no campo Data/Hora. Tal seleção é feita através do comando "select DATAHORA, ID, RA, NOME, IDADE, RG, CARRO, PLACA from %s;", como é mostrado na figura 49, no apêndice 4.

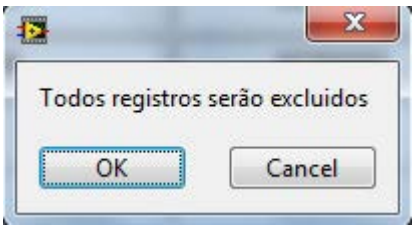
Mas, se foi selecionada a tabela Usuario, então pode ser carregado todos os dados da tabela através do comando "select * from %s;".

Após ser escrito o comando de acordo com a tabela inserida, esses dados são inseridos na interface do programa para que o operador do programa possa visualizar os mesmos.

Se o botão Deletar não é pressionado, nada é feito após os dados serem carregados. O *Case Structure* para quando este é pressionado é mostrado na figura 50, no apêndice 4.

Na primeira etapa da *Flat Sequence,* o operador recebe a mensagem "Todos registros serão excluídos", tendo a opção de pressionar a opção "OK" ou "Cancelar", como mostrado na figura abaixo.

Figura 21 – Mensagem 5



Fonte: Autoria própria.

Se a opção selecionada for *Cancelar* nada é feito e na próxima etapa da *Flat Sequence* o botão *Deletar* é desativado, mas se a opção *OK* for selecionada, então é feita a exclusão de todos os dados contidos na tabela em questão através do comando "delete from %s;". A programação em blocos para essa opção é mostrada na figura 51, no apêndice 4.

Caso seja preenchido tanto o campo *Coluna* quanto o campo *Dados,* o *Case Structure vai para False.* Sua programação é muito parecida ao outro estado, mudando apenas o *Case*

Structure relacionado ao acionamento do botão *Deletar*. Este é mostrado na figura 52, no apêndice 4.

Nesta situação, quando o botão *Deletar* é acionado o dado específico selecionado é deletado sem um prévio aviso do sistema. Essa ação é realizada através do comando "delete from %s where %s ="%s";" especificando a tabela, a coluna e os dados presentes nessa coluna e assim, a linha que possui dados em comum com o requerido é excluída.

No VI busca também existem em paralelo uma programação para que seja realizada a leitura da UID da *tag* atual, etapa essa idêntica às existentes nos outros VIs, como mostra as figuras 53 e 54, no apêndice 4.

4 Conclusão

Durante o desenvolvimento de todo o projeto, foi necessário muito estudo sobre as tecnologias que nele seriam aplicadas. Através desses estudos realizados ficaram evidentes as diversas aplicações e soluções que o sistema RFID vem oferecendo para facilitar o nosso dia-a-dia, aumentar nossa segurança e até mesmo tornar o gerenciamento industrial mais rápido e organizado; sendo totalmente equivocado dizer que o RFID é apenas uma tecnologia criada para substituir o código de barras.

O objetivo do trabalho foi atingido com sucesso, objetivos esses que eram a criação de um sistema de controle de acesso para a circulação de carros na universidade, confiável, de baixo custo e fácil operação. Todas as etapas da programação foram focadas em impedir erros e fraudes no sistema, prezando pela segurança e confiabilidade do sistema. O Arduino tem o custo aproximado de R$100,00 e o módulo leitor RFID de 13,56 MHz tem um custo aproximado de R$90,00 e as *tags* RFID tem um custo aproximado de R$2,00 variando de acordo com a quantidade adquirida, sendo esses os únicos equipamentos externos necessários de investimento, então o sistema mostra-se bastante acessível levando em conta os ganhos com segurança e agilidade que a portaria da UNESP - Campus de Guaratinguetá ganharia.

Portanto, o projeto mostrou ser uma boa solução para os problemas de seguranças enfrentados pela UNESP - Campus de Guaratinguetá, podendo sanar os problemas levando, em conta todos os aspectos de fundamental importância para tornar a sua aplicabilidade viável.

Modificações podem ser feitas para tornar o sistema mais funcional, como adotar uma outra forma de comunicação entre o controlador e o *middleware*, como por exemplo substituir a comunicação serial por uma conexão sem fio. Outra modificação seria aumentar a segurança do sistema contra fraudes de *tags* podendo ser realizada uma criptografia nos cartões para evitar a clonagem dos mesmos. Entretanto, no geral, o sistema está pronto para ser utilizado se considerarmos o seu destino de aplicação onde não tem a necessidade de um investimento em sistemas de segurança tão avançados, sendo muitas vezes caros.

4.1 Sugestões do Autor

Durante testes e apresentações foram apresentadas algumas sugestões de *upgrade* no projeto que melhorariam a sua funcionalidade. Entre elas, algumas estão citadas a seguir.

- Criar um modo de o programa identificar se o registro no sistema é de entrada ou saída da Universidade, assim como uma alternativa para atualizar os dados cadastrais sem haver a necessidade de excluir todos os dados de um usuário para depois arquivar os novos dados.
- Troca do sistema de comunicação entre Arduino e LabVIEW de serial para comunicação sem fio, como por exemplo ethernet ou wireless.
- Implementar um sistema onde possa operar vários sistemas desenvolvido nesse projeto simultaneamente, transmitindo os dados para um computador mestre. Essa implementação se torna necessária devido a necessidade de utilizar o sistema em, no mínimo, quatro cancelas já que a Universidade possui duas portarias.
- Implementar um "*watch dog"* no Arduino para que o sistema consiga se recuperar de qualquer eventual erro que possa ocorrer durante o funcionamento do projeto.
- Armazenamento de foto de cada usuário.
- Utilização de câmera de segurança integrada no sistema para a verificação de compatibilidade entre os dados físicos (carro, cor do carro, placa, pessoa).

BIBLIOGRAFIA CONSULTADA

Arduino Company. Disponível em:
<www.arduino.cc> Acessado em 20 set. 2014.

BERNARDO, CLAUDIO GONÇALVES, A Tecnologia RFID e os Benefícios da Etiqueta Inteligente para os Negócios, IPT/USP, 2004.

GOLDING, PAUL; TENNANT, VANESSA, Using RFID Inventory Reader at the Item-Level in a Library, University of Technology, 2010

HUNT, V. D.; PUGLIA, A.; PUGLIA, M., A Guide To Radio Frequency Identification". Wiley-Interscience, 2007

LABVIEW Database Connectivity Toolkit User Manual. May 2001 Edition. Disponível em:
<http://www.ni.com/manuals/pt/> Acessado em 15 dez. 2014.

LabVIEWing. Disponível em:
<http://www.labviewing.com/> Acessado em 17 ago. 2014.

MySQL . Disponível em:
<http://dev.mysql.com/> Acessado em 30 out. 2014.

National Instruments. Disponível em:
< http://www.ni.com/labview/pt/> Acessado em 30 out. 2014.

NYOMAN ADHIARNA; JAE-JEUNG RHO. Standardization and Global Adoption of Radio Frequency Identification (RFID): Strategic Issues for Developing Countries, IEEE Communication

REFERÊNCIA

Arduino Website. Disponível em:
< http://www.arduino.cc/> Acesso em 09 dez 2014

LEAL, JOSÉ MANUEL PERDIGÃO SILVA, RFID: O Futuro da Gestão de Stocks na Grande Distribuição, Universidade de Lisboa, 2008.

Jornal Acrítica, 2011

Library for MFRC522. 2015. Disponível em:
<https://github.com/miguelbalboa/rfid> Acesso em 15 dez. 2014.

National Instruments Website. Disponível em:
< http://brasil.ni.com/> Acesso em 07 jan 2015.

SANTINI, ARTHUR GAMBIN, RFID: Conceitos, Aplicabilidade e Impactos. 1ª ed. Rio de Janeiro: Ciência Moderna, 2008.

SYBASE, Estudo da Arte, RSC (RFID Solutions Center), 2006

APÊNDICE 1 – IMAGENS PROGRAMAÇÃO LABVIEW - Painel Inicial

Figura 22 – Programação em Blocos VI Painel Inicial 1

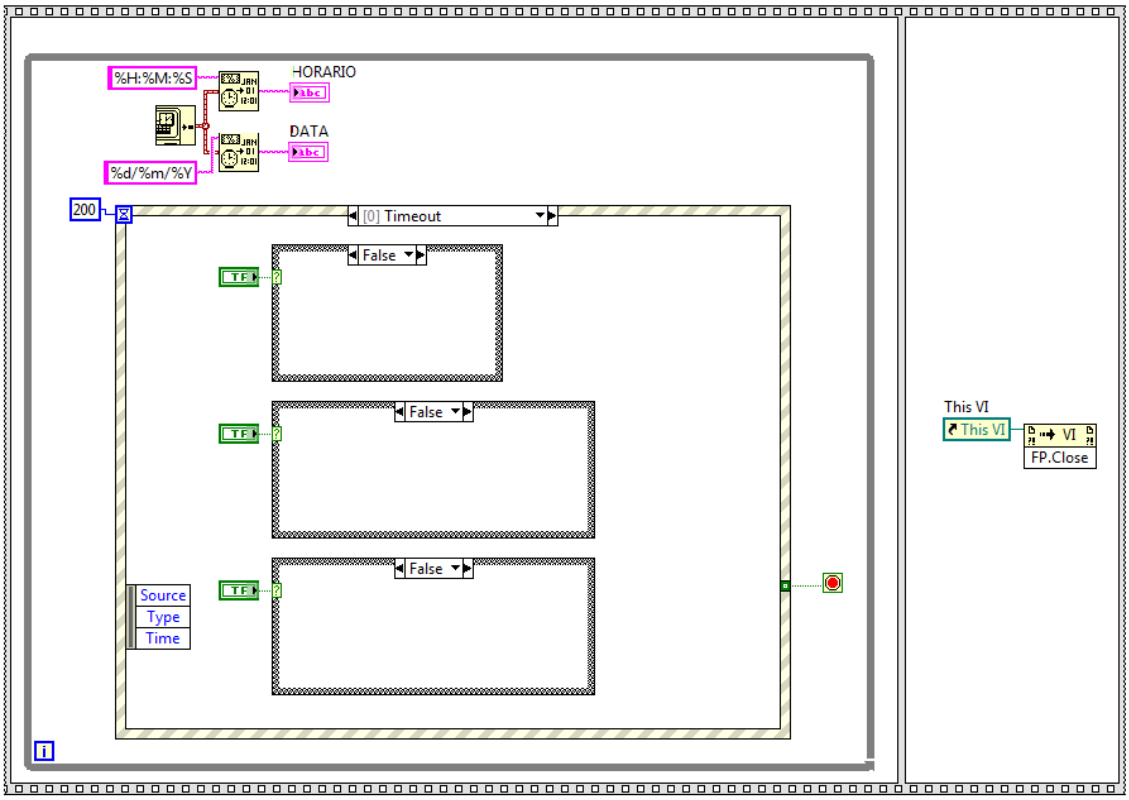


Figura 23 – Programação em Blocos VI Painel Inicial 2

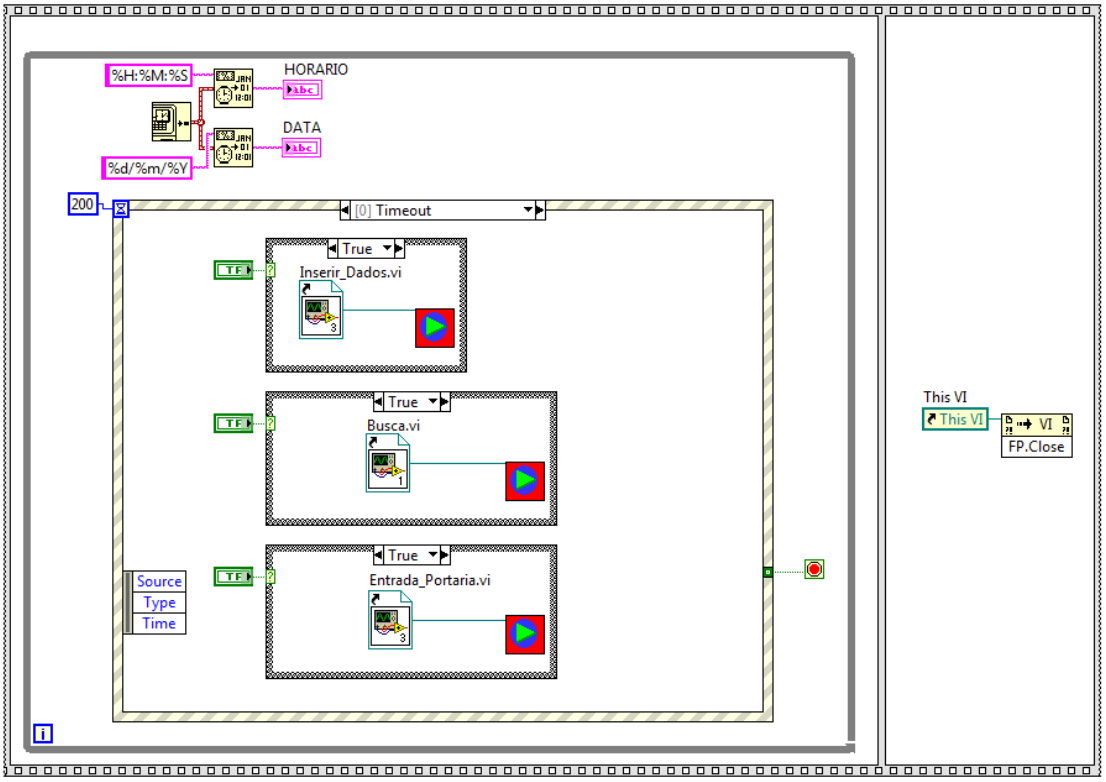


Figura 24 – Programação em Blocos VI Painel Inicial 3

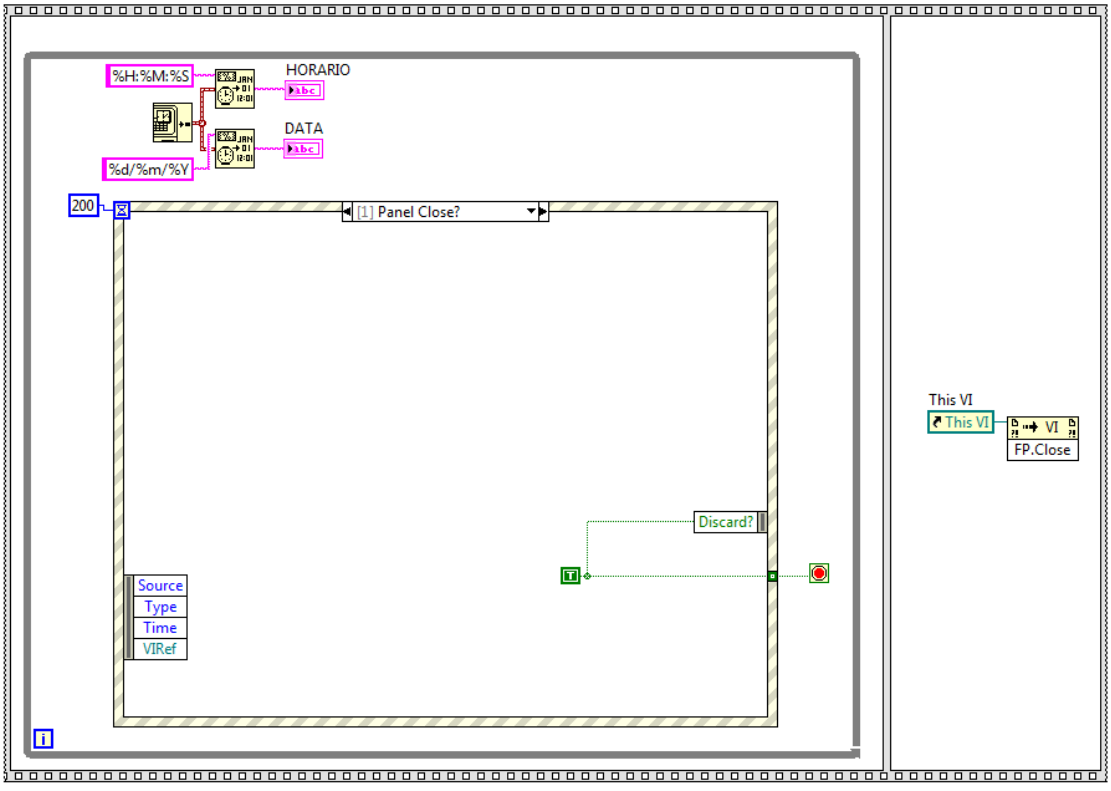
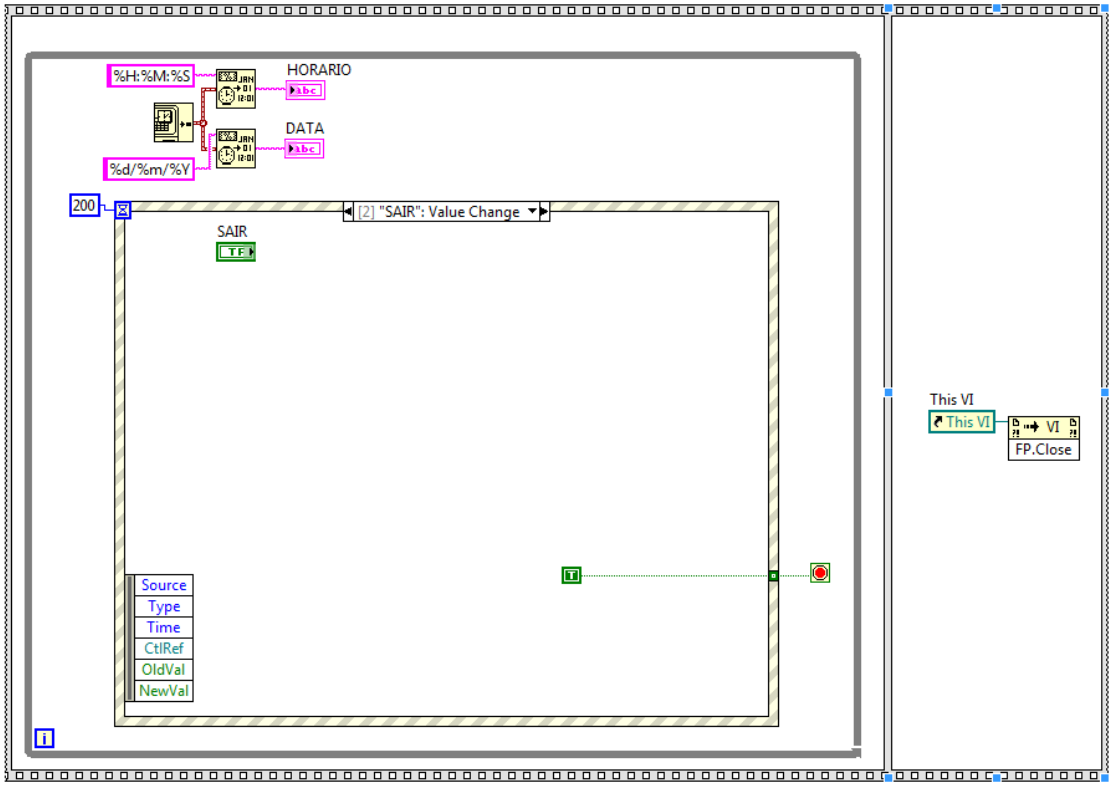


Figura 25 – Programação em Blocos VI Painel Inicial 4



APÊNDICE 2 – IMAGENS PROGRAMAÇÃO LABVIEW – VI Entrada Portaria

Figura 26 – Programação em Blocos VI Entrada Portaria 1

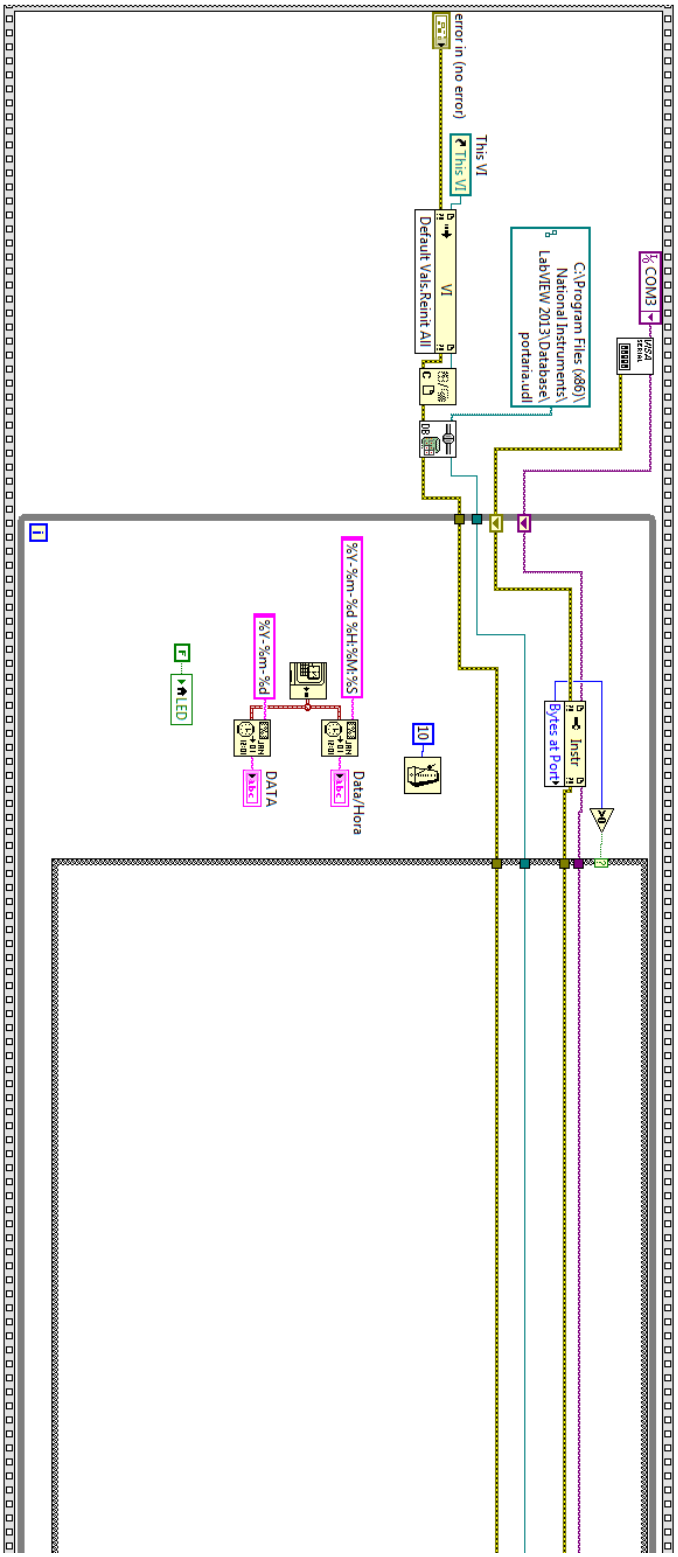


Figura 27 – Programação em Blocos VI Entrada Portaria 2

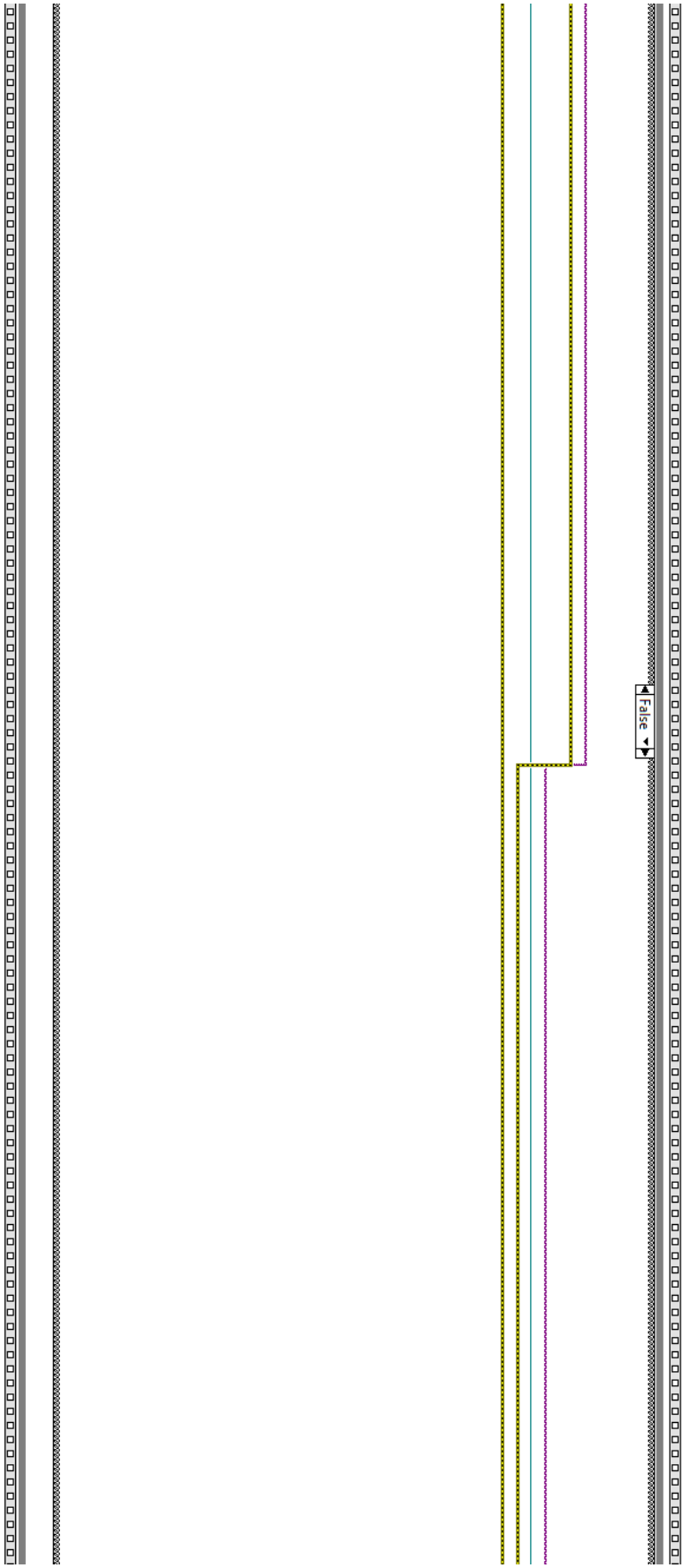


Figura 28 – Programação em Blocos VI Entrada Portaria 3

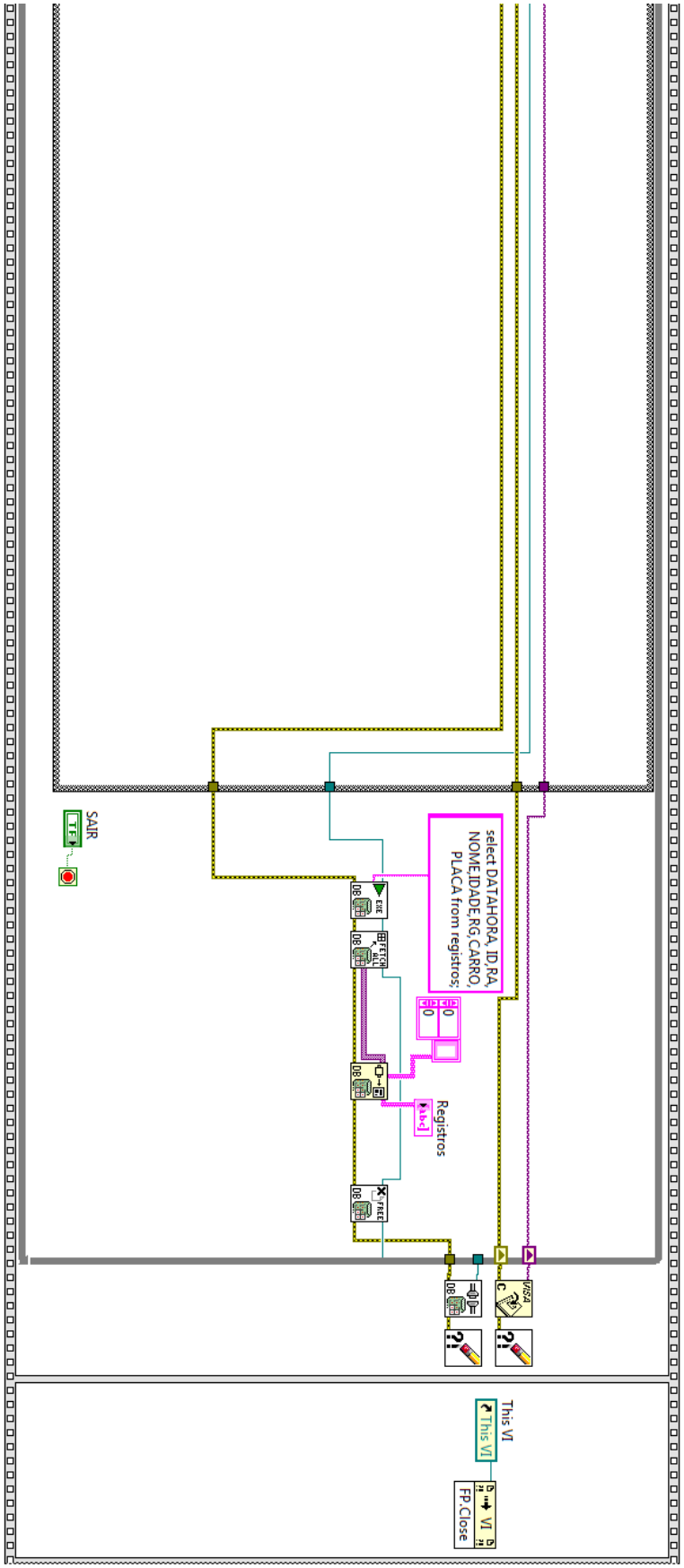


Figura 29 – Programação em Blocos VI Entrada Portaria 4

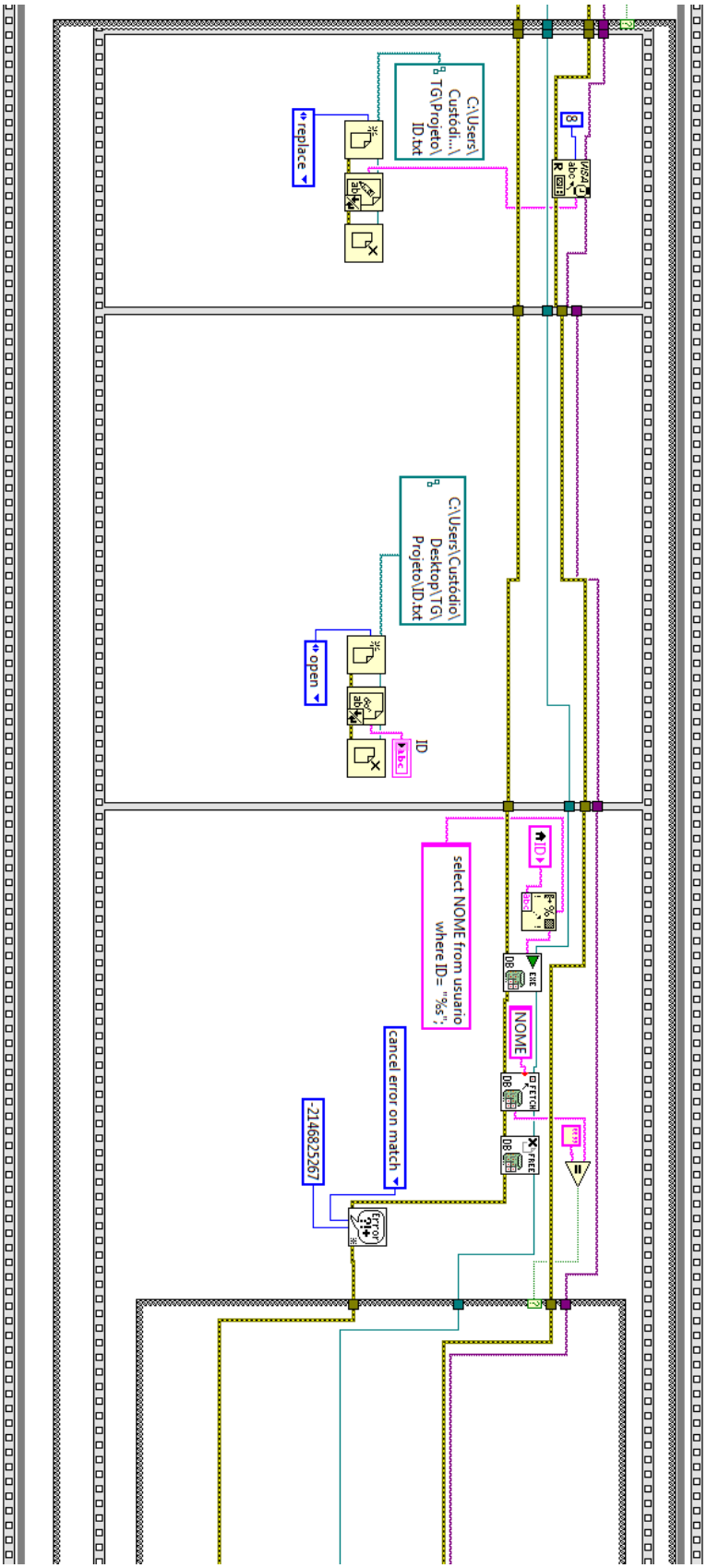


Figura 30 – Programação em Blocos VI Entrada Portaria 5

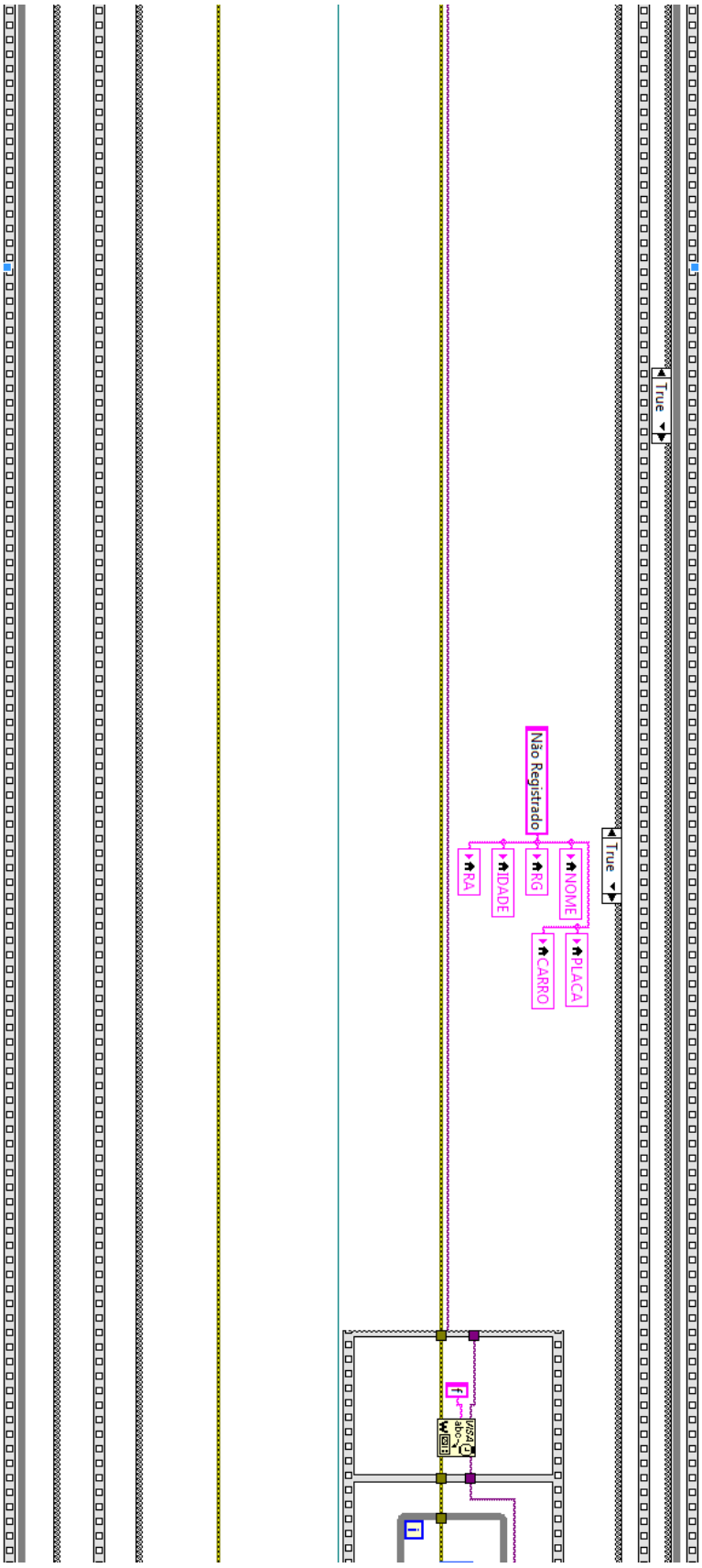


Figura 31 – Programação em Blocos VI Entrada Portaria 6

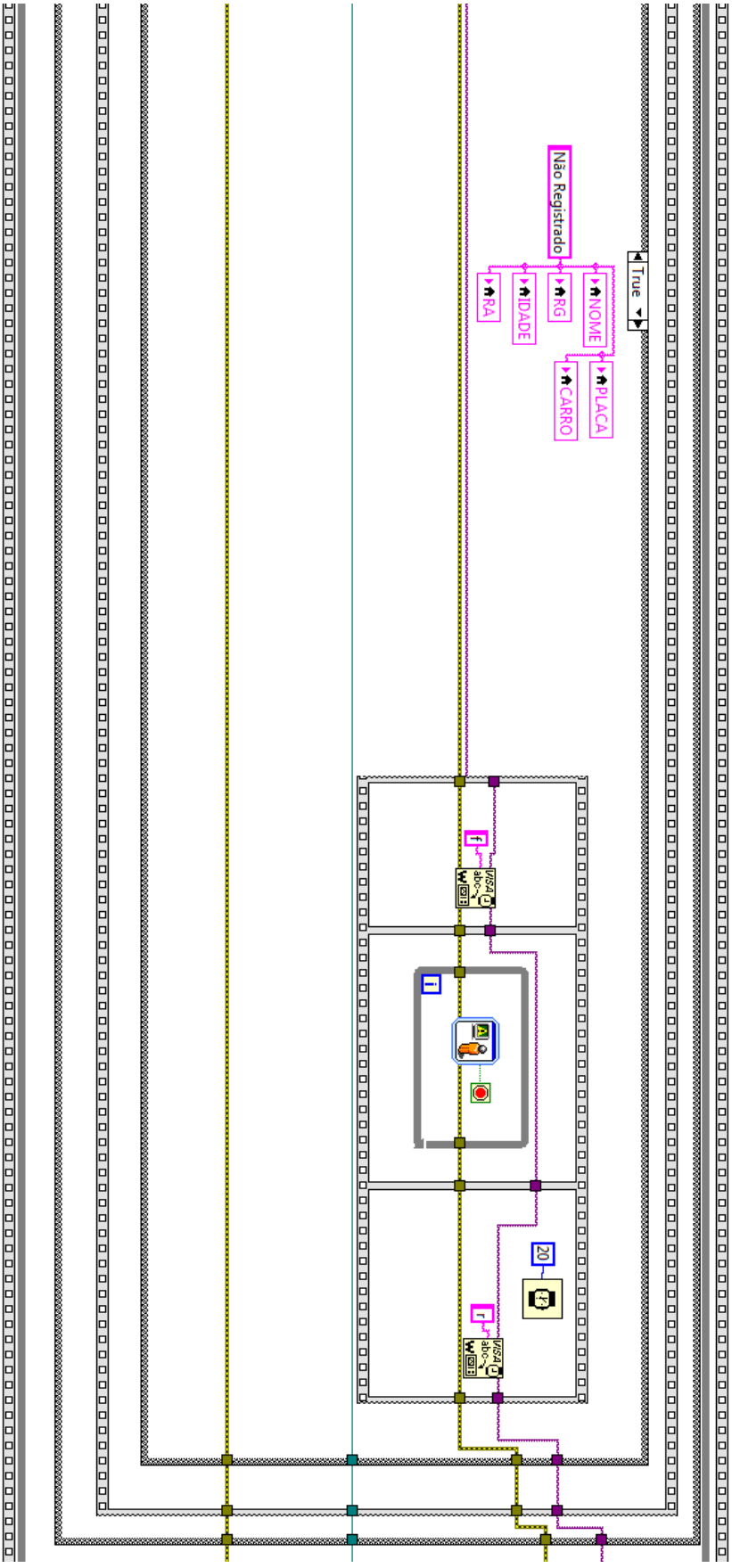


Figura 32 – Programação em Blocos VI Entrada Portaria 7

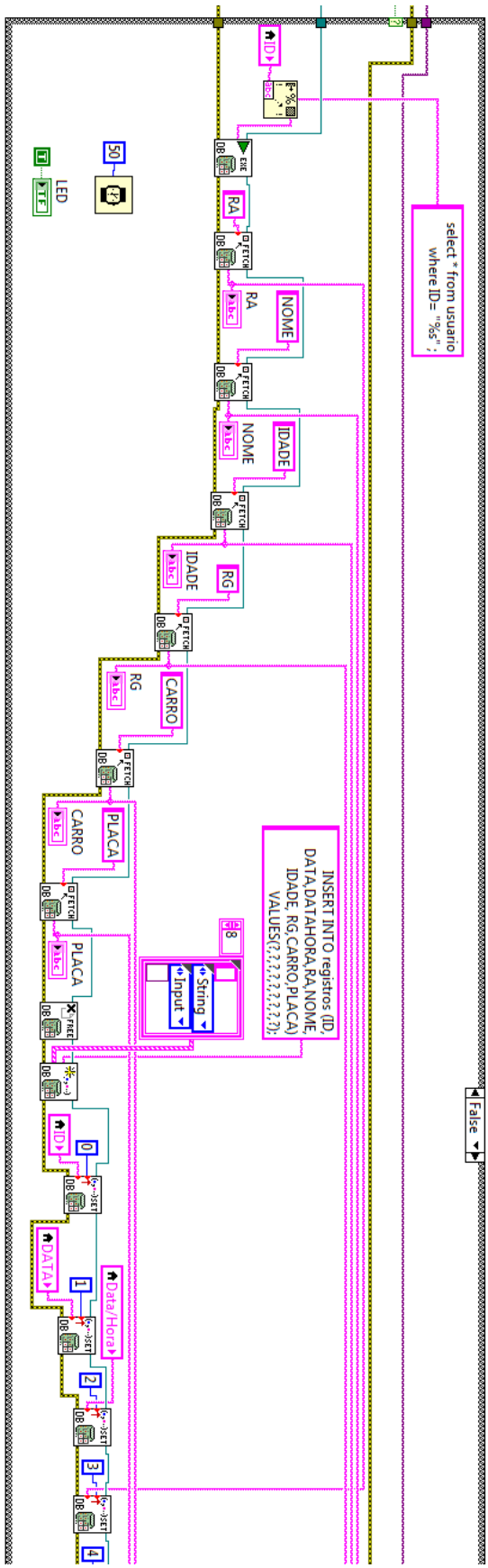
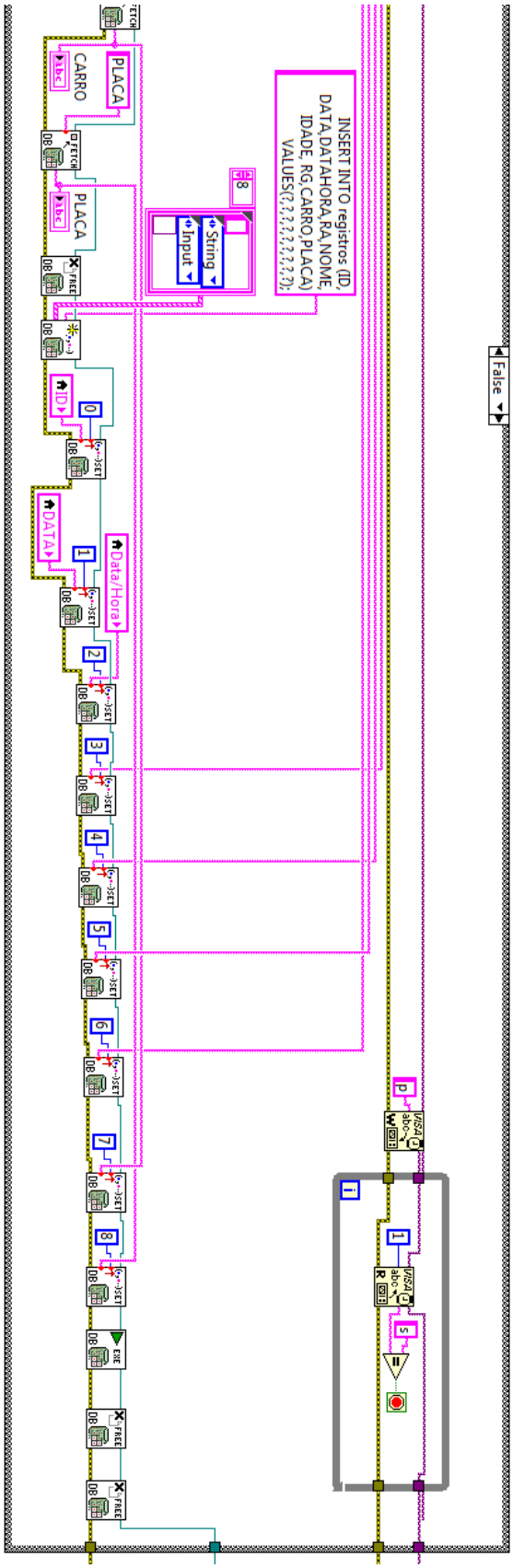


Figura 33 – Programação em Blocos VI Entrada Portaria 8



APÊNDICE 3 – IMAGENS PROGRAMAÇÃO LABVIEW – VI Inserir Dados

Figura 34 – Programação em Blocos VI Inserir Dados 1

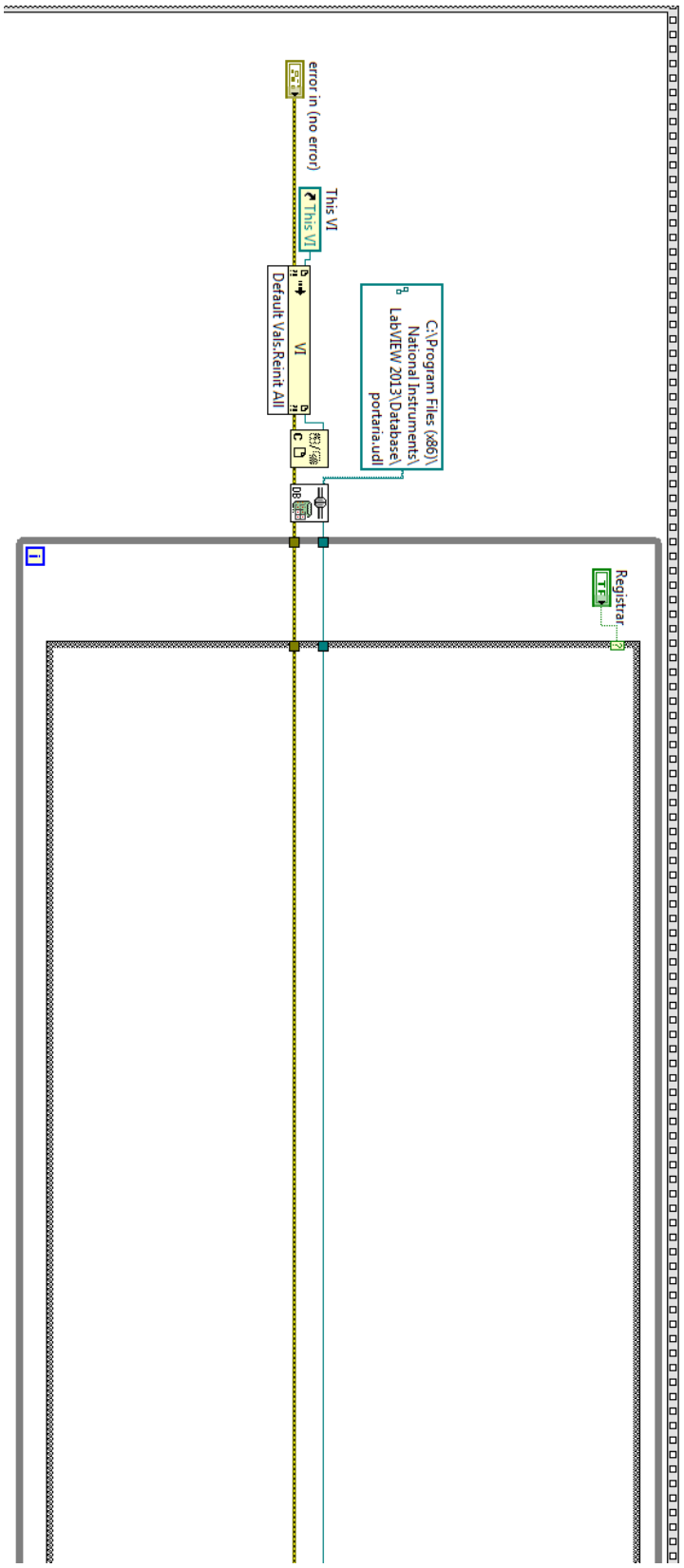


Figura 35 – Programação em Blocos VI Inserir Dados 2

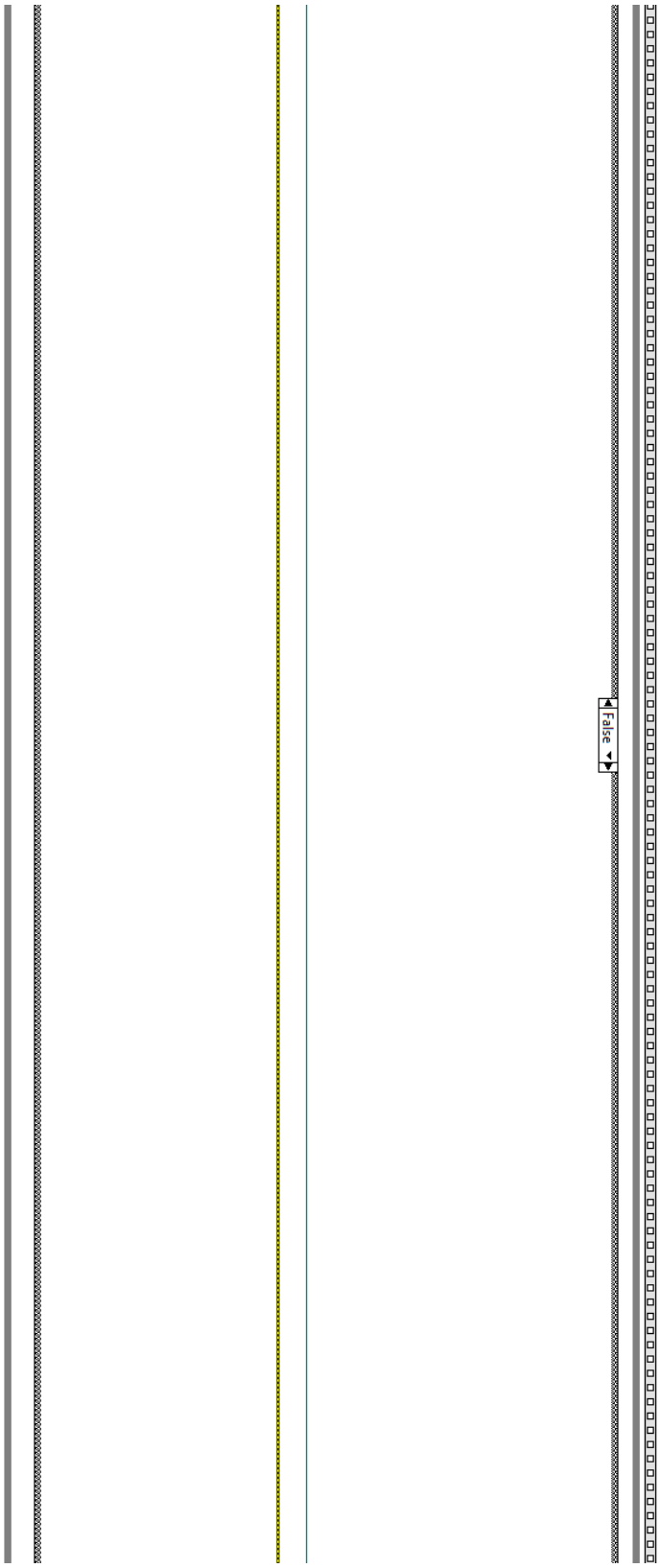


Figura 36 – Programação em Blocos VI Inserir Dados 3

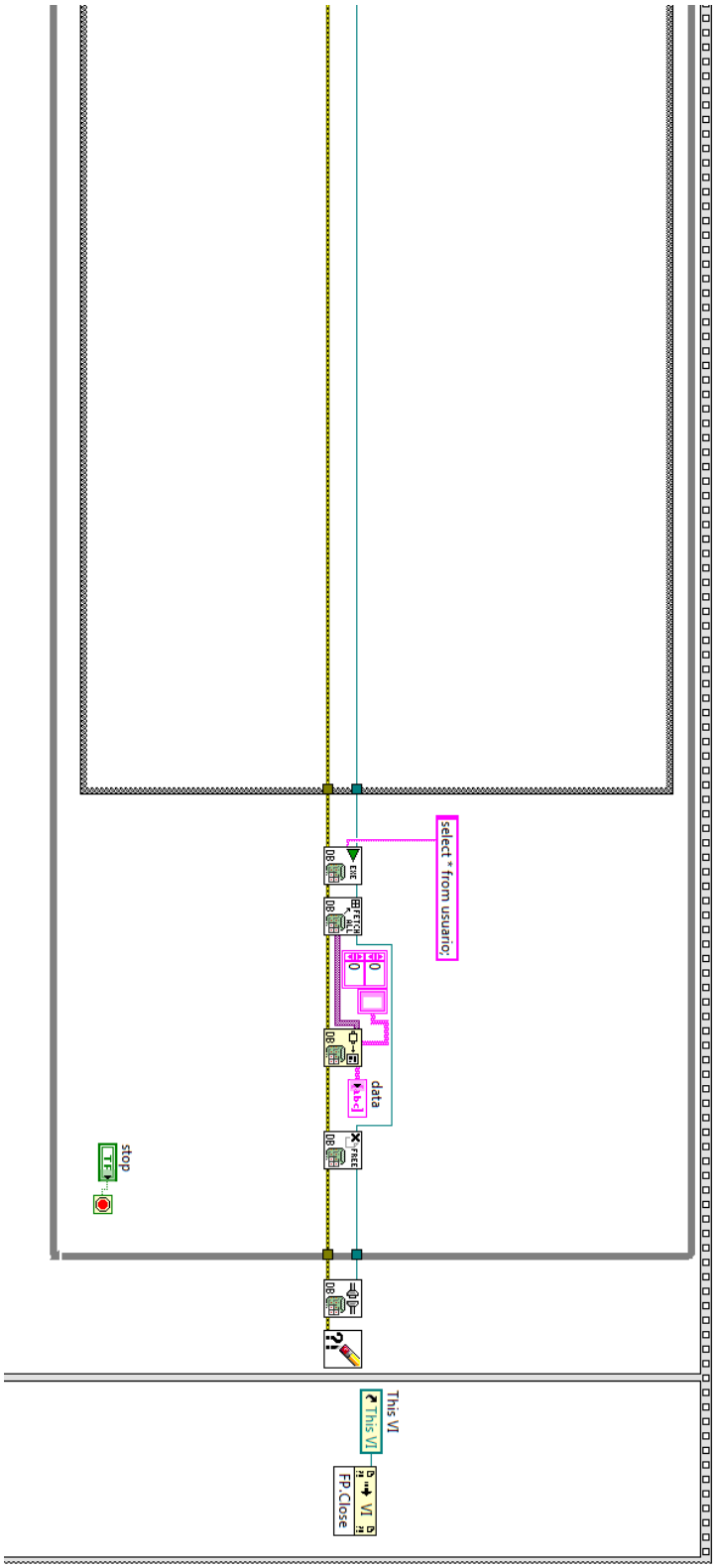


Figura 37 – Programação em Blocos VI Inserir Dados 4

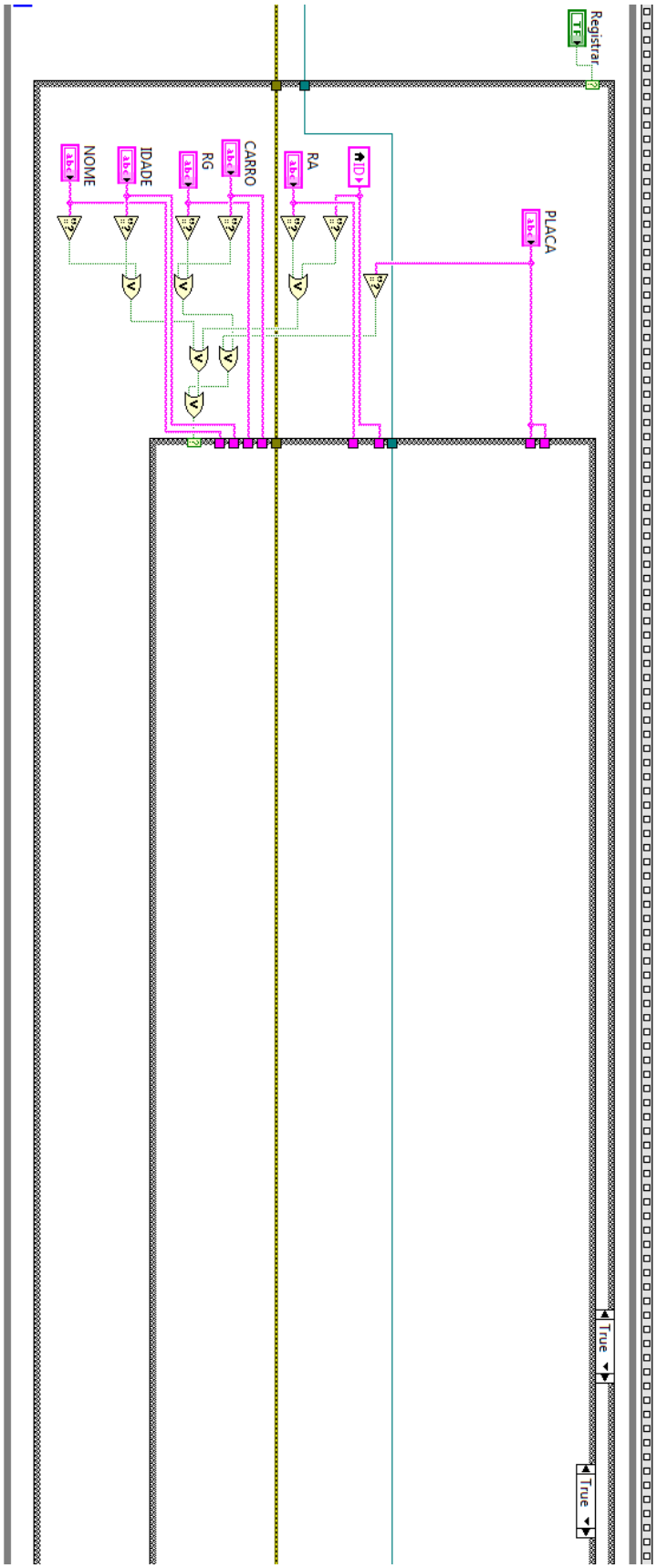


Figura 38 – Programação em Blocos VI Inserir Dados 5

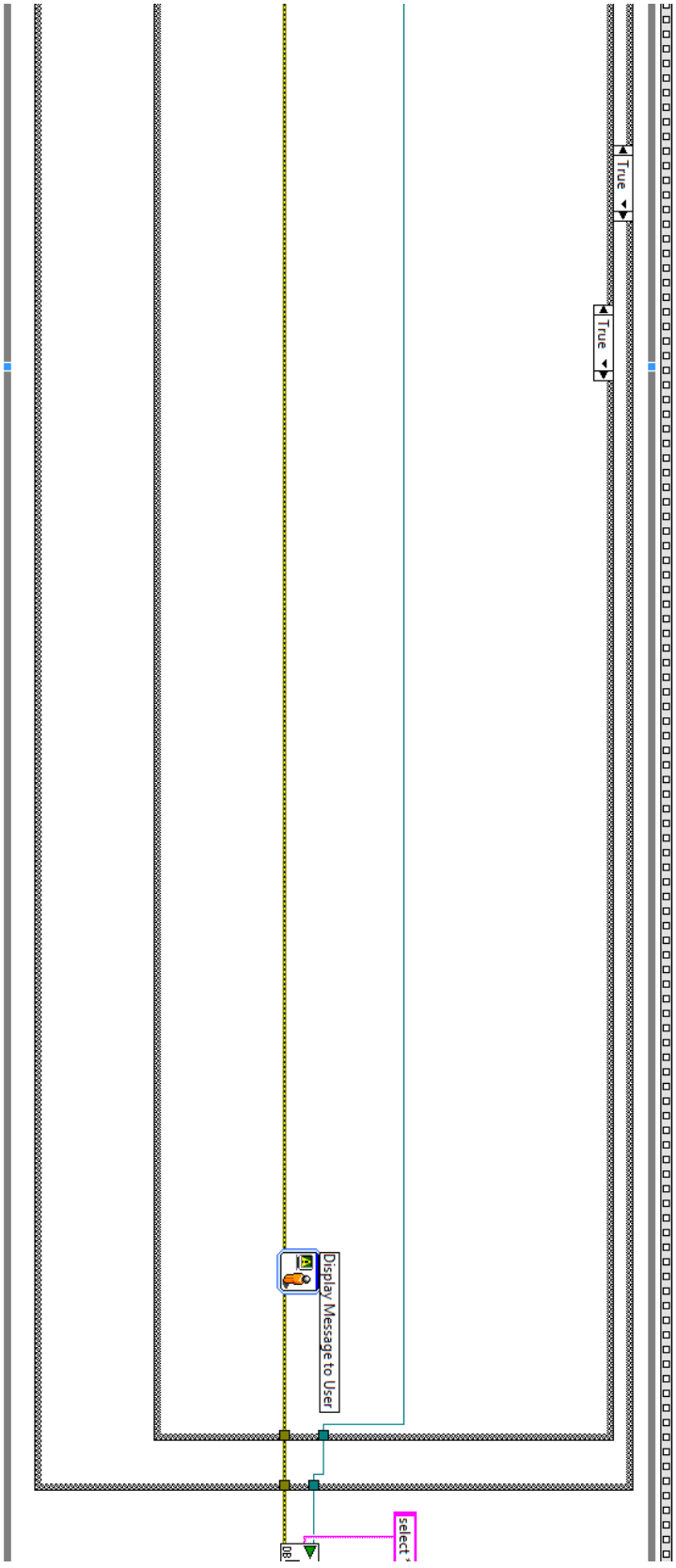


Figura 39 – Programação em Blocos VI Inserir Dados 6

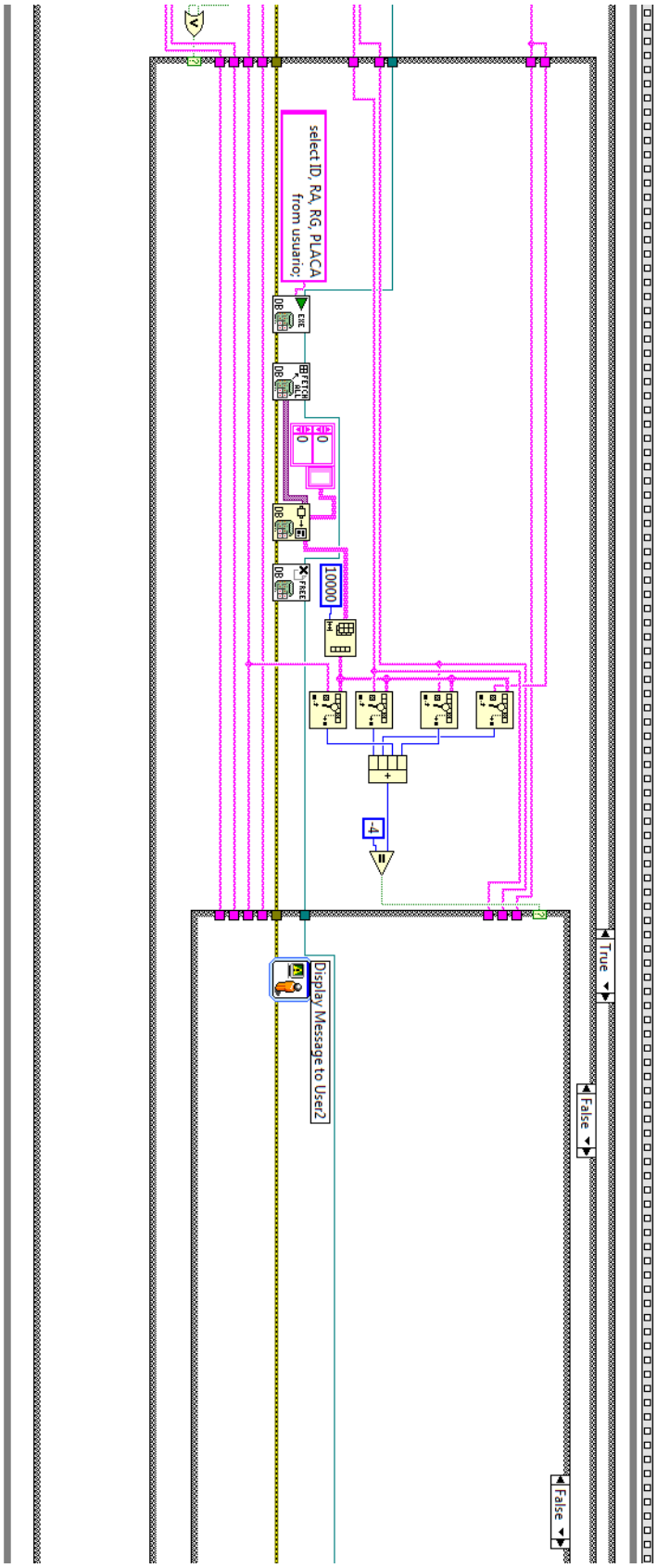


Figura 40 – Programação em Blocos VI Inserir Dados 7

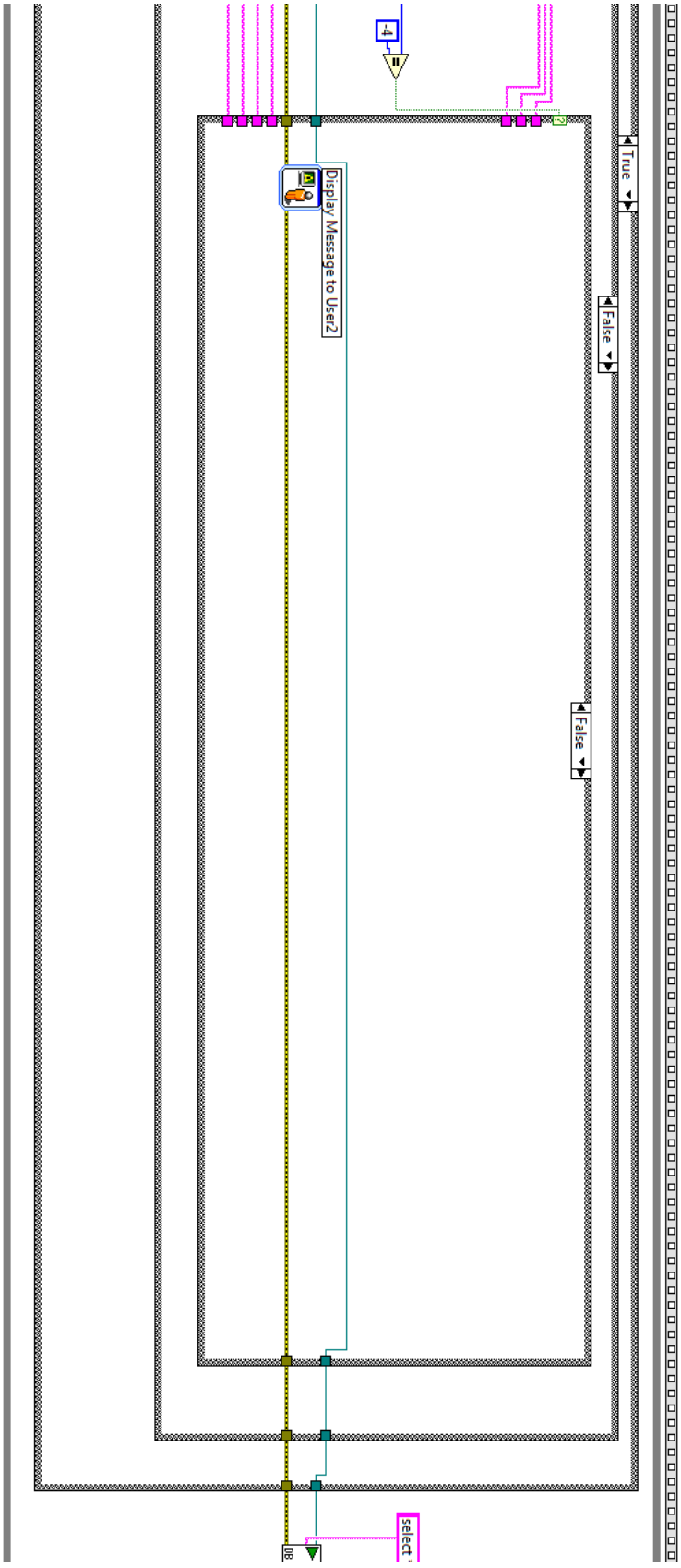


Figura 41 – Programação em Blocos VI Inserir Dados 8

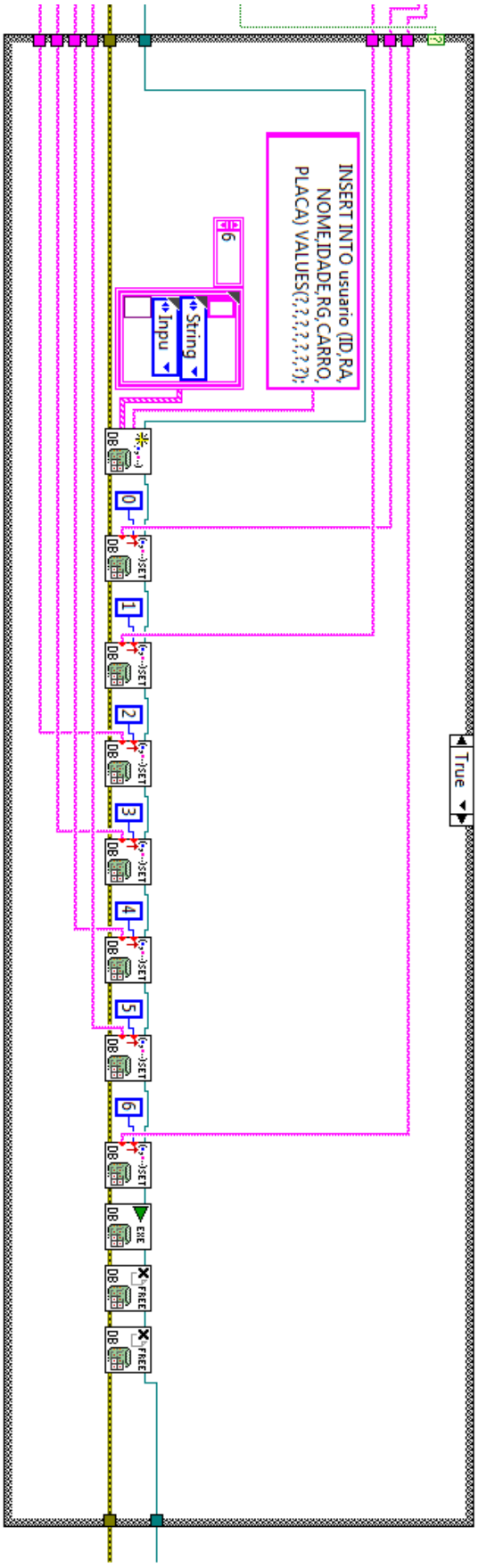


Figura 42 – Programação em Blocos VI Inserir Dados 9

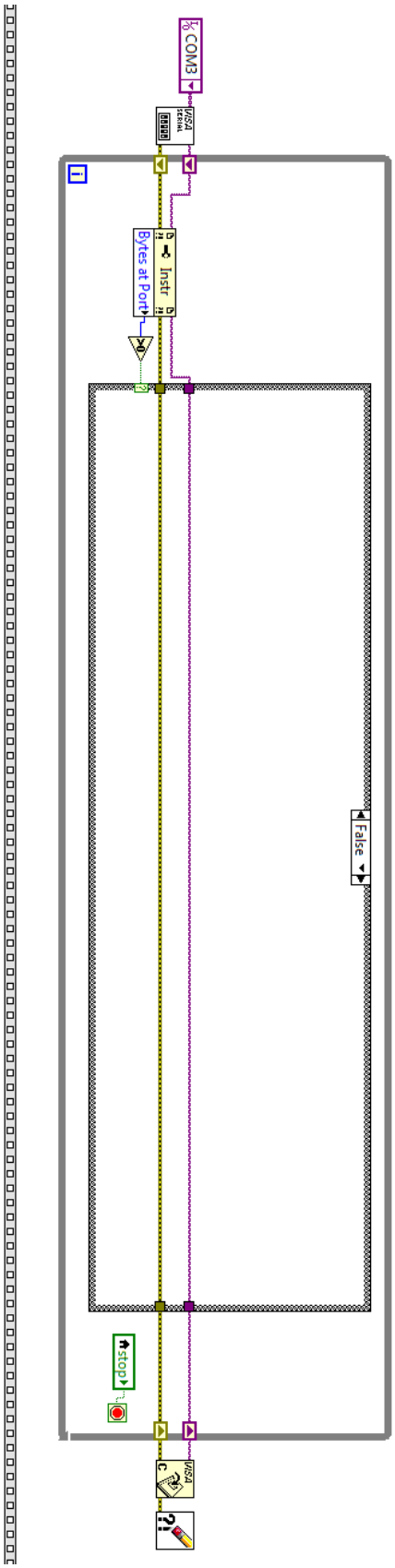
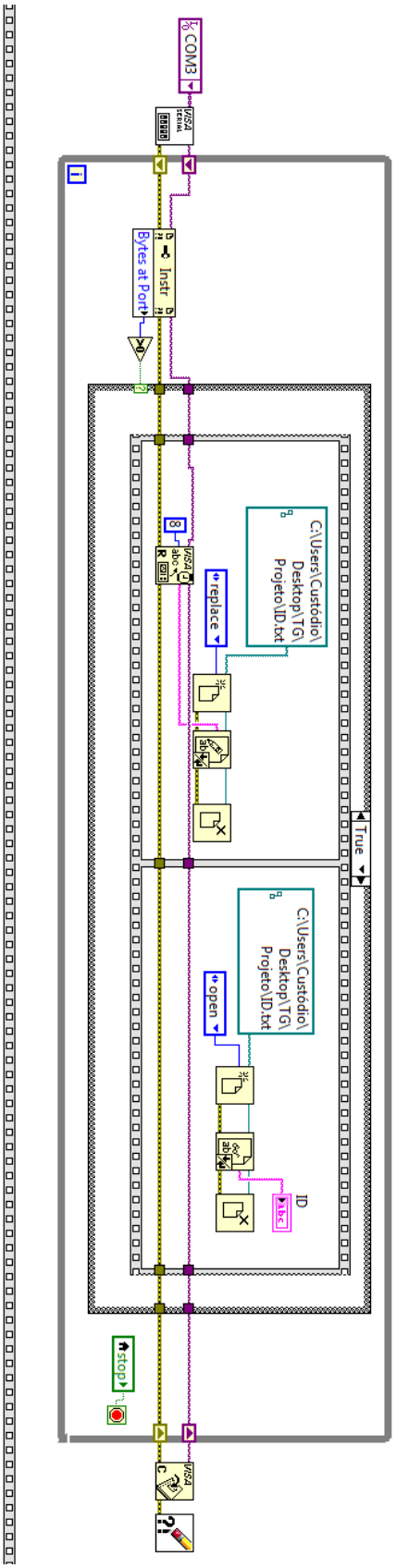


Figura 43 – Programação em Blocos VI Inserir Dados 10



APÊNDICE 4 – IMAGENS PROGRAMAÇÃO LABVIEW – VI Busca

Figura 44 – Programação em Blocos VI Busca 1

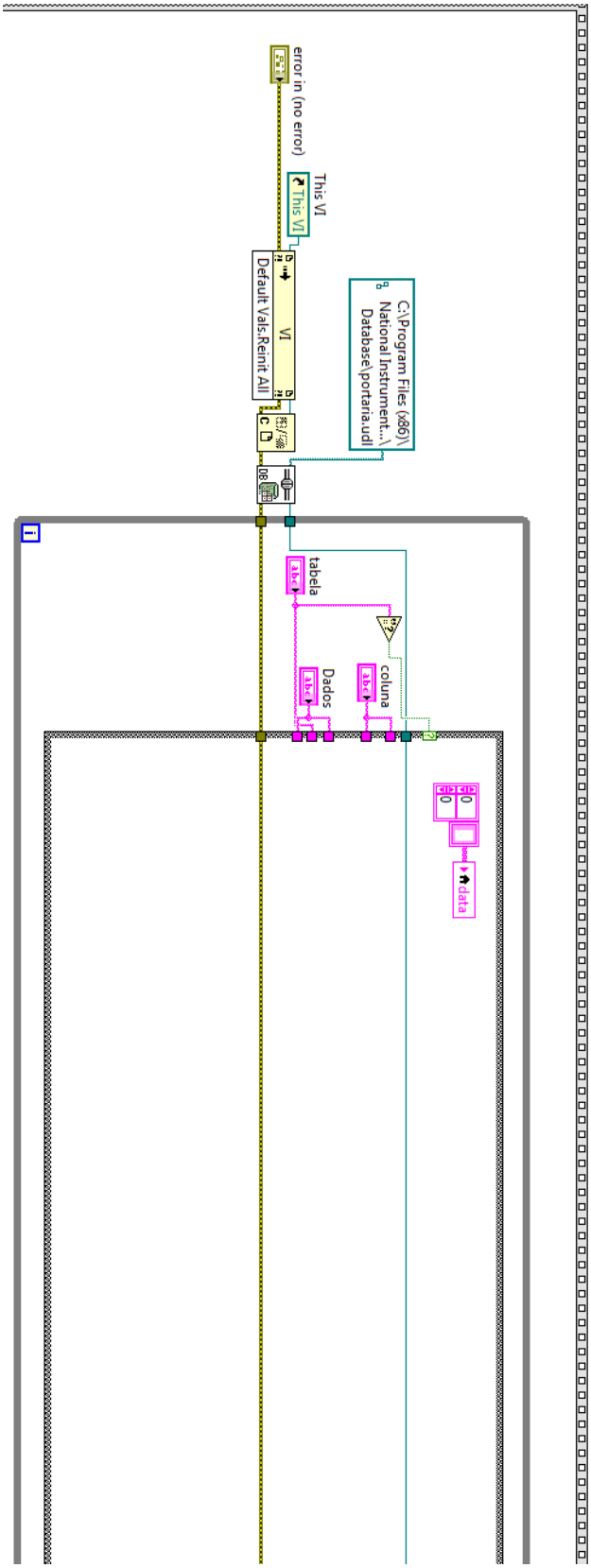


Figura 45 – Programação em Blocos VI Busca 2

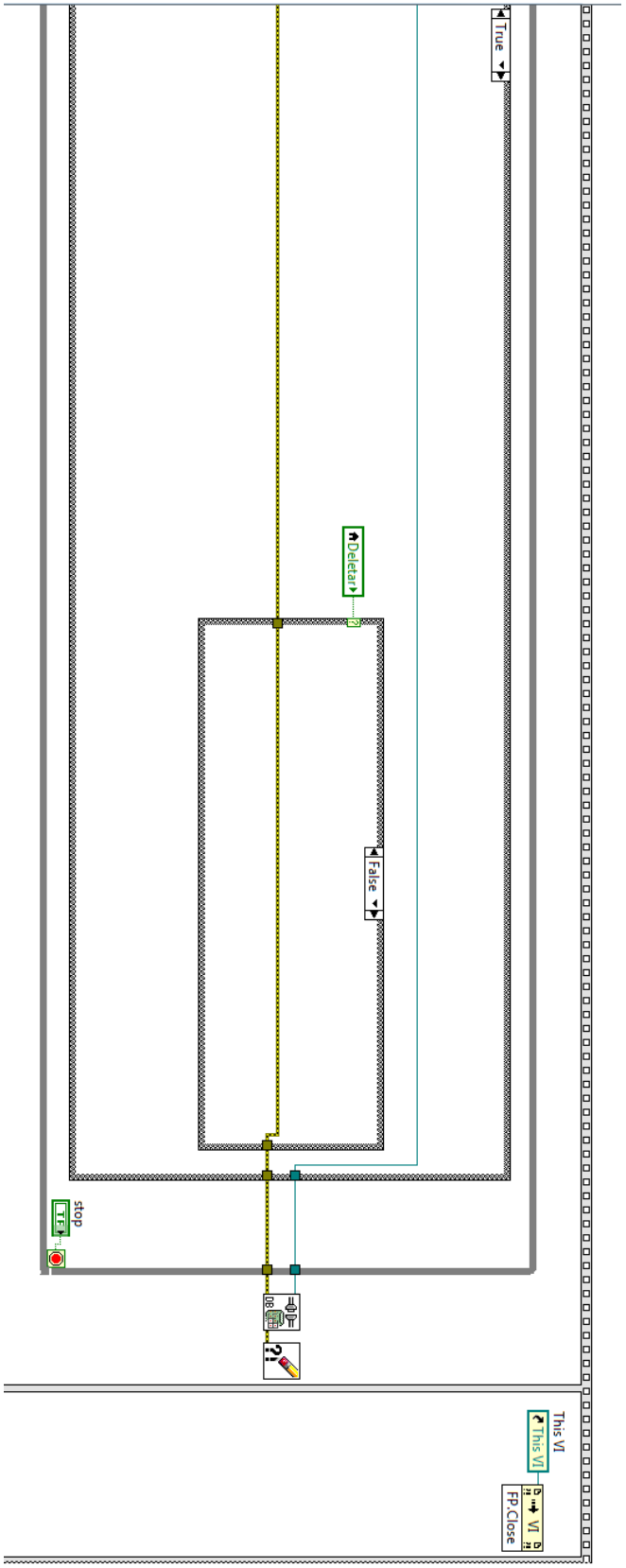


Figura 46 – Programação em Blocos VI Busca 3

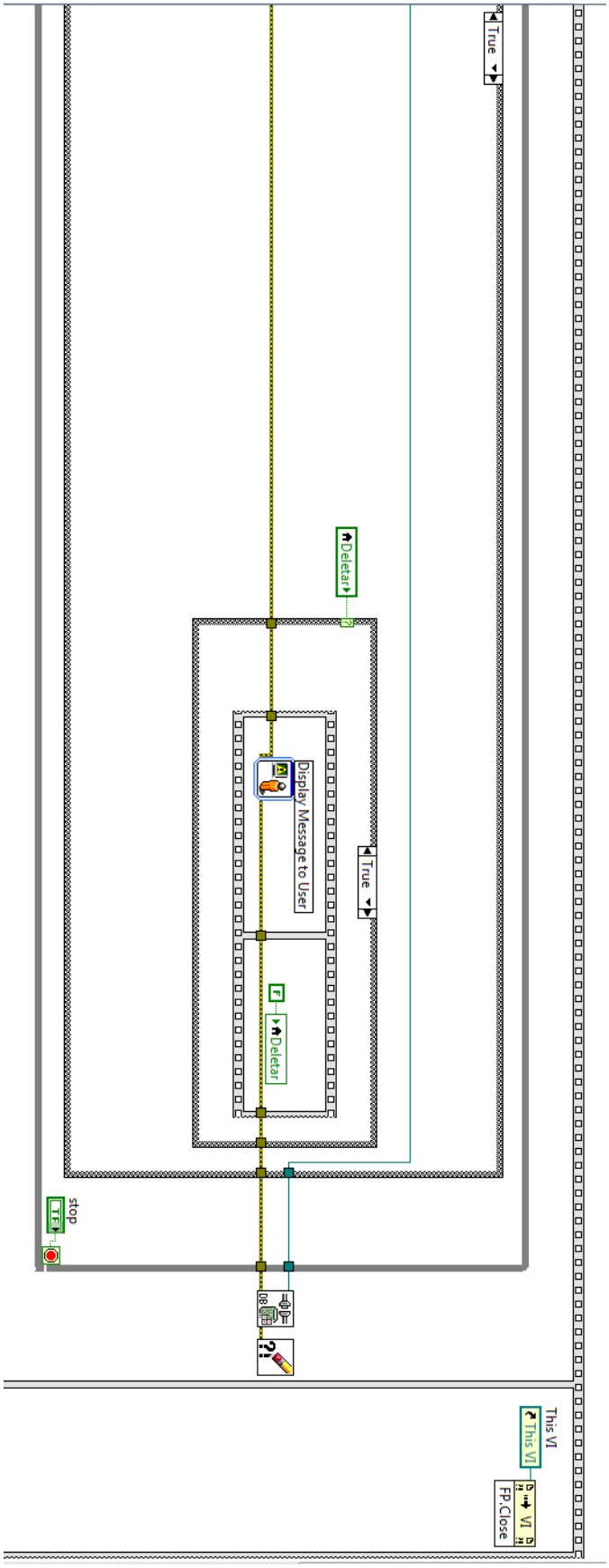


Figura 47 – Programação em Blocos VI Busca 4

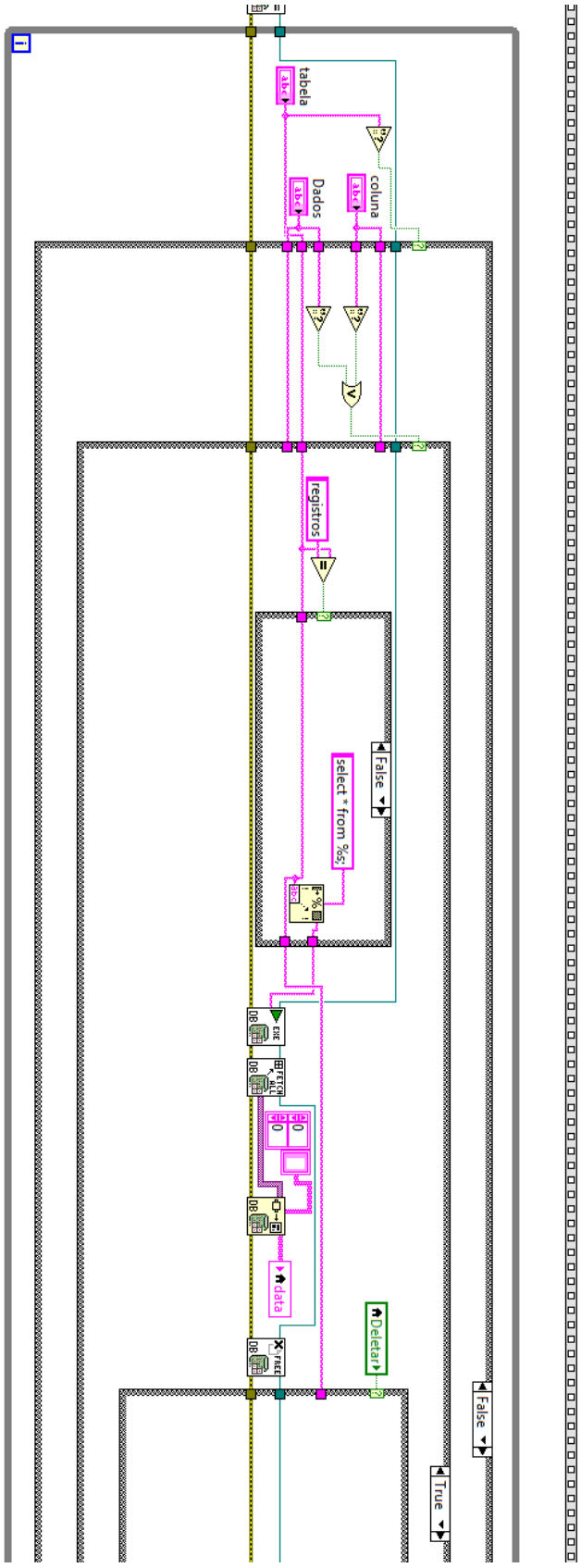


Figura 48 – Programação em Blocos VI Busca 5

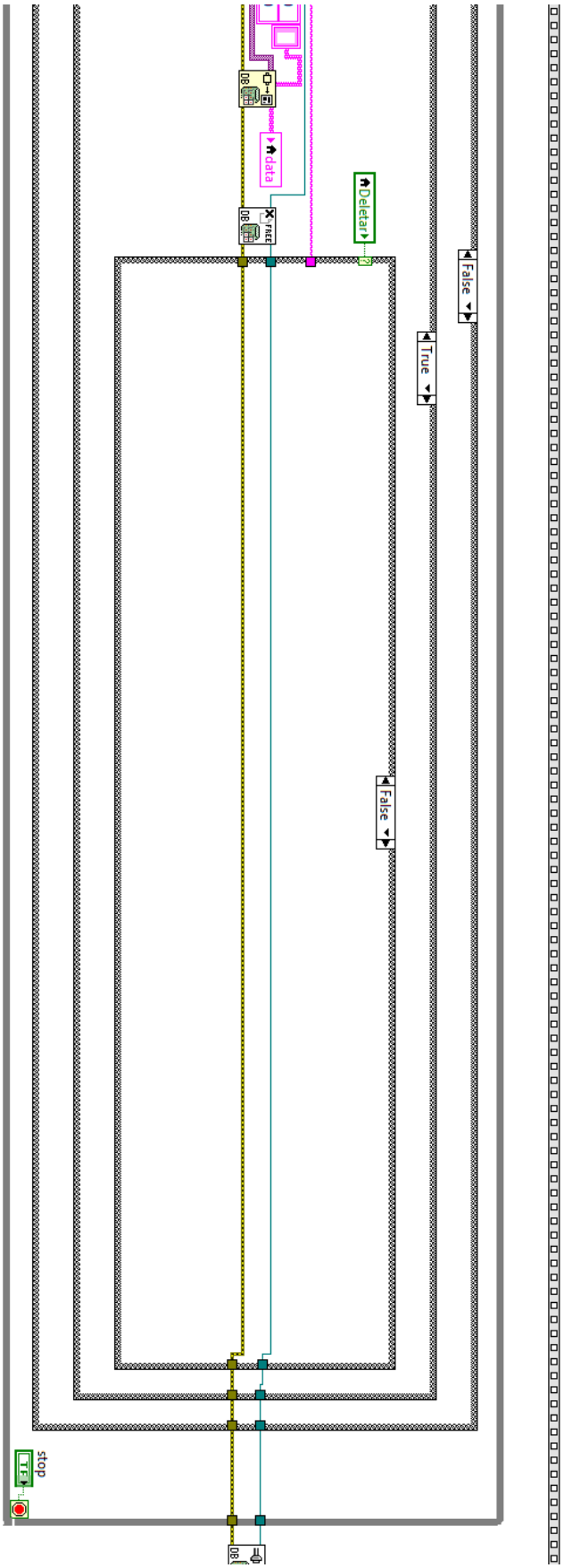


Figura 49 – Programação em Blocos VI Busca 6

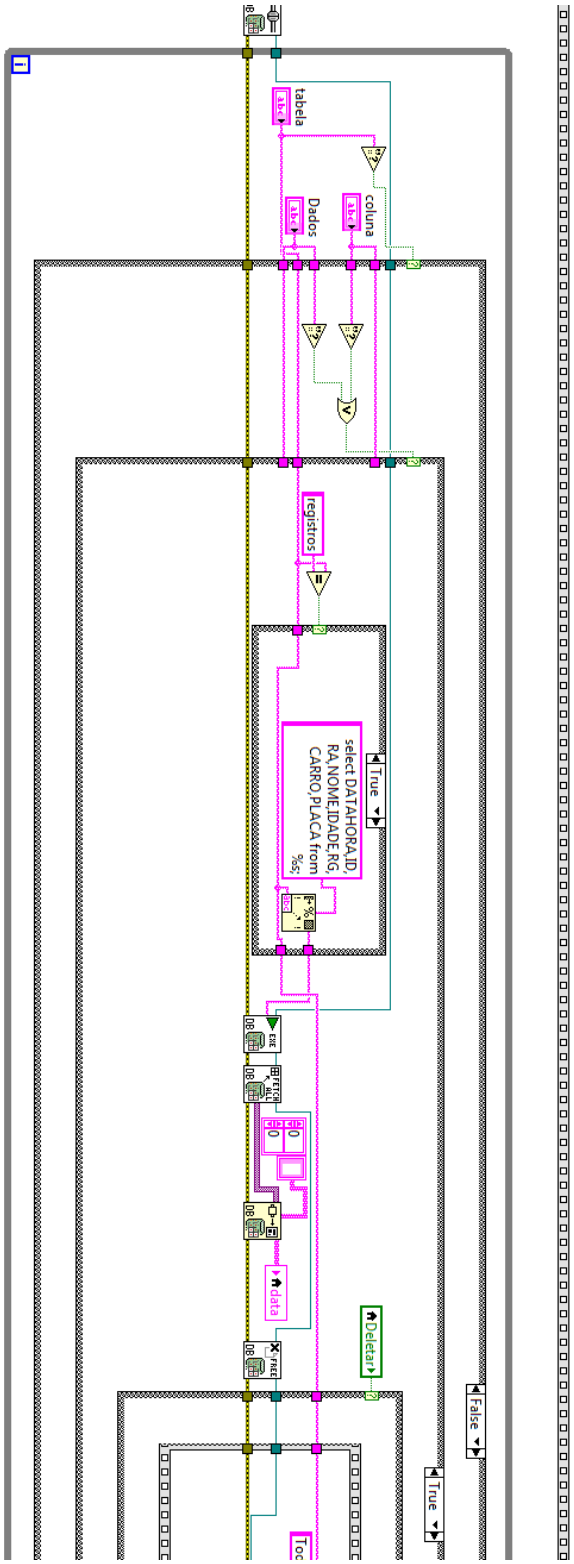


Figura 50 – Programação em Blocos VI Busca 7

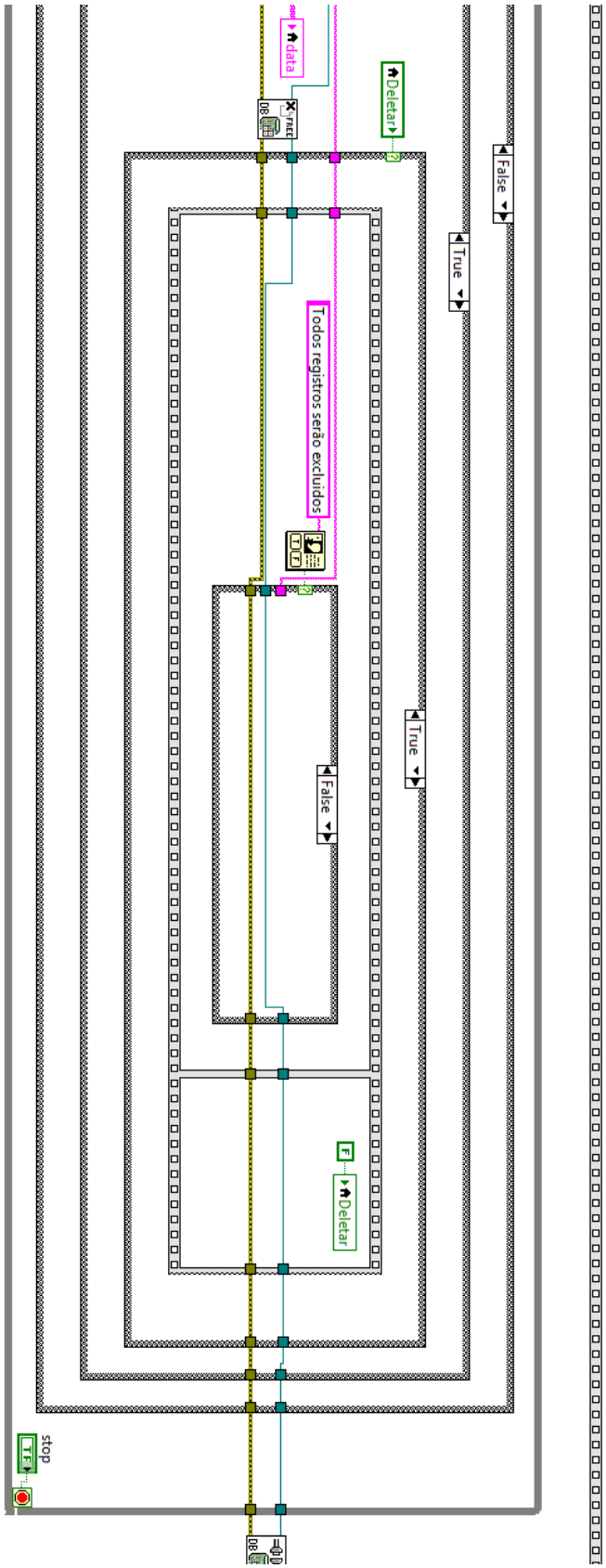


Figura 51 – Programação em Blocos VI Busca 8

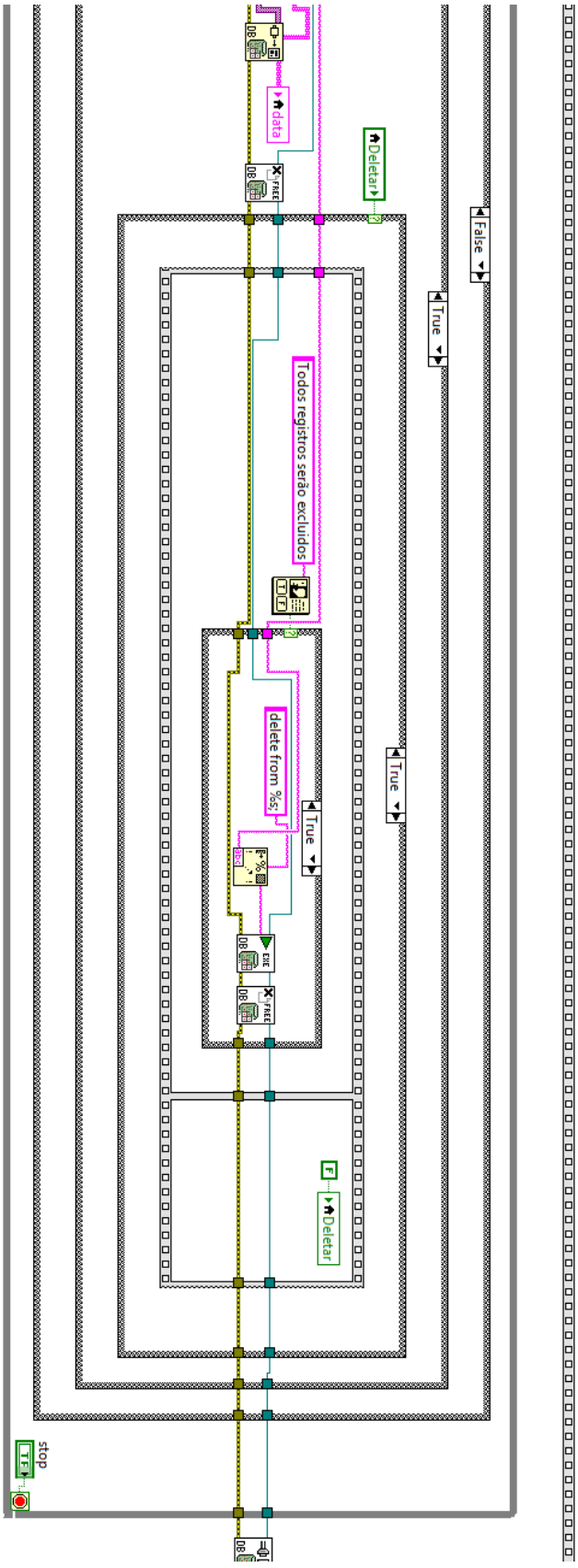


Figura 52 – Programação em Blocos VI Busca 9

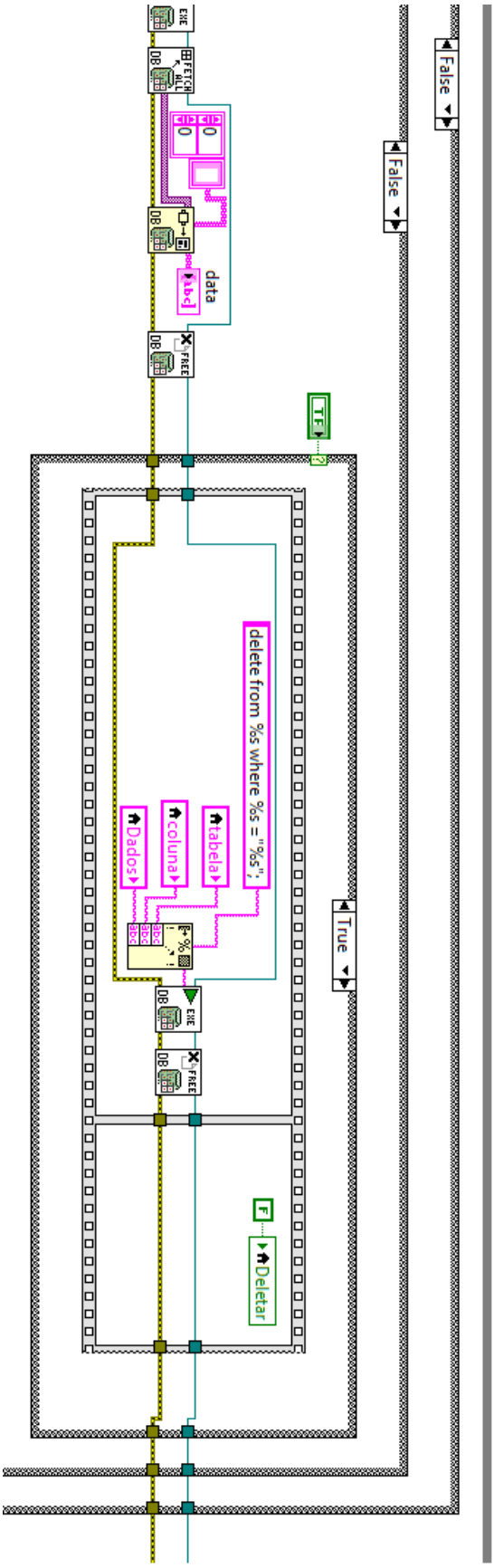


Figura 53 – Programação em Blocos VI Busca 10

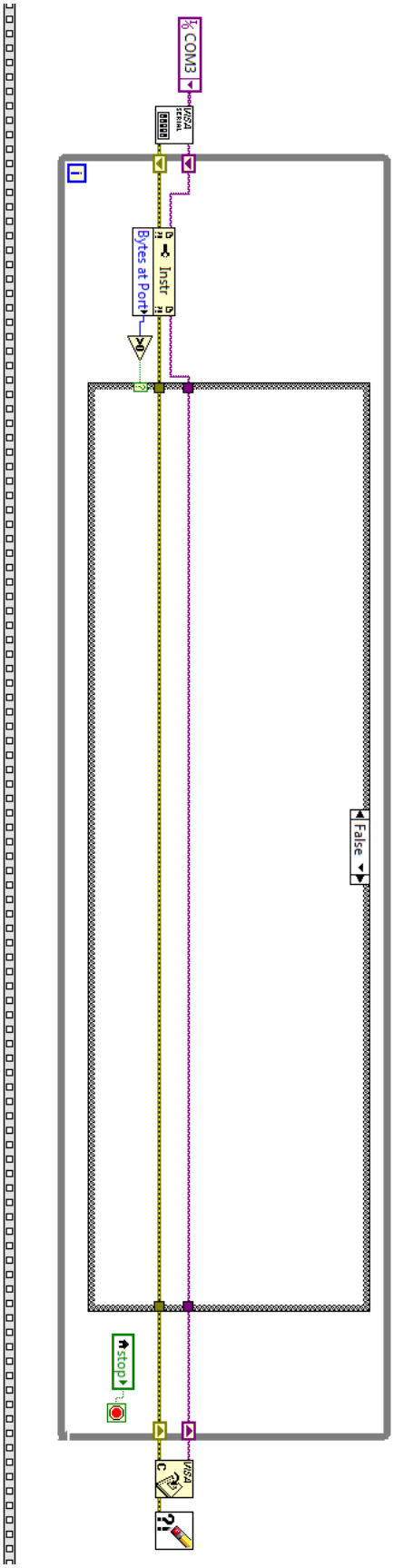
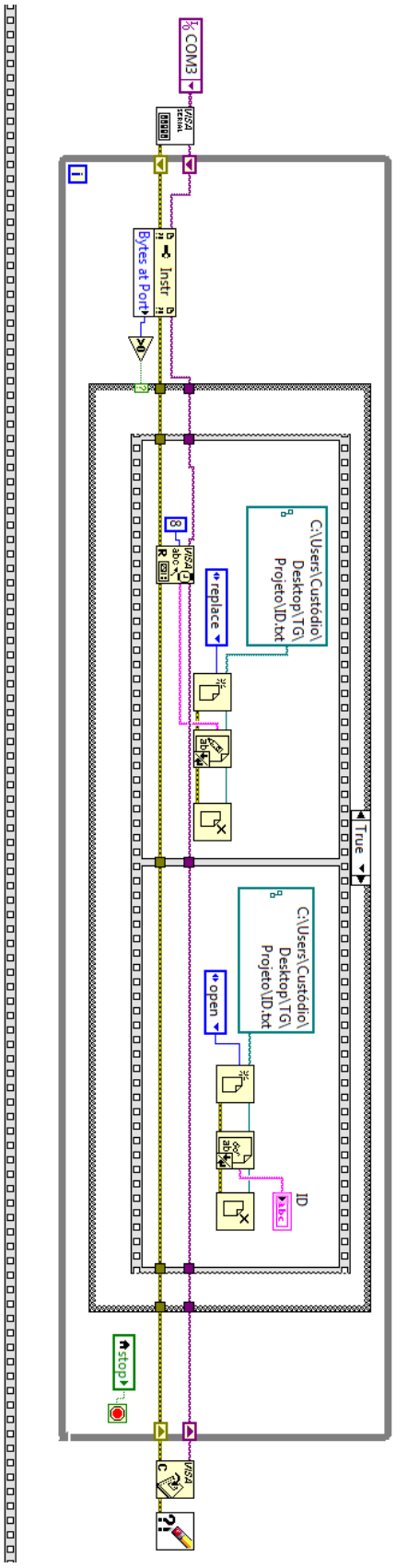


Figura 54 – Programação em Blocos VI Busca 11



APÊNDICE 5 – Programação Arduino

```
#include <SPI.h>
#include <MFRC522.h>
#include <Servo.h>
                                                                    //1
Servo myservo;
int control, pos = 0, a;
char ID[4];
#define SS_PIN 10
#define RST_PIN 9
MFRC522 mfrc522(SS_PIN, RST_PIN);

void setup() {                                                  //2
Serial.begin(9600);
SPI.begin();
mfrc522.PCD_Init();
myservo.attach(3);
}

void loop()
{
myservo.write(pos);                                             //3
if ( ! mfrc522.PICC_IsNewCardPresent())                         //4
{
return;
}
if ( ! mfrc522.PICC_ReadCardSerial())                           //5
{
return;
}
for (byte i = 0; i < mfrc522.uid.size; i++)                     //6
{
Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? "0" : "");
Serial.print(mfrc522.uid.uidByte[i], HEX);
```

```
}
control=0;
delay(200);
if(Serial.available()>0)                                    //7
{
control=Serial.read();
if(control == 'p')                                          //8
{
for(pos = 0; pos < 90; pos += 1)                            //9
{
myservo.write(pos);
delay(15);
}
delay(1000);
for(pos = 90; pos>=1; pos-=1)                               //10
{
myservo.write(pos);
delay(15);
}
Serial.print('s');                                          //11
}
if(control=='f')                                            //12
{
for(control=0;control!='r';control=control)                 //13
{
control=Serial.read();
}
}
}
delay(1000);
}
```