CENTRO UNIVERSITÁRIO DE BRASÍLIA - UniCEUB

CURSO DE ENGENHARIA DE COMPUTAÇÃO

MÁRCIO DE ANDRADE MELLO

SISTEMA DE CONTROLE REMOTO DE TRANCA DE PORTA VIA TECNOLOGIA GSM

Orientador: Prof. José Julimá Bezerra Junior

Brasília

Novembro, 2010

MÁRCIO DE ANDRADE MELLO

SISTEMA DE CONTROLE REMOTO DE TRANCA DE PORTA VIA TECNOLOGIA GSM

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.
Orientador: Prof. José Julimá Bezerra Junior

Brasília

Novembro, 2010

MÁRCIO DE ANDRADE MELLO

SISTEMA DE CONTROLE REMOTO DE TRANCA DE PORTA VIA TECNOLOGIA GSM

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.
Orientador: Prof. José Julimá Bezerra Junior

Este Trabalho foi julgado adequado para a obtenção do Título de Engenheiro de Computação, e aprovado em sua forma final pela Faculdade de Tecnologia e Ciências Sociais Aplicadas - FATECS.

_____

Prof. Abiezer Amarilia Fernandez
Coordenador do Curso

Banca Examinadora:

_____

Prof. M.Sc. José Julimá Bezerra Junior
Orientador

_____

Prof. nome, titulação.
Instituição

_____

Prof. nome, titulação.
Instituição

_____

Prof. nome, titulação.
Instituição

DEDICATÓRIA

Dedico este trabalho aos meus pais Silene Magali de Andrade e Márcio Antônio da Costa Mello que sempre se dedicaram ao máximo em me dar tudo do melhor.

AGRADECIMENTOS

Primeiramente agradeço aos meus pais por tudo que fizeram por mim em minha vida. Agradeço ao meu tio Thyrso Villela que me ajudou muito com a elaboração da monografia e me aconselhou bastante durante o processo de escrita. Agradeço ao colega Felipe Souto por me ajudar na construção do protótipo. Ao meu orientador, José Julimá, e ao corpo docente do UniCEUB pelos anos de conhecimento adquiridos. Agradeço, em especial, à minha prima Luciana Melo e aos meus amigos Maria Luiza, Diego Soares e Hoa Lim por todo apoio que me deram com meus problemas pessoais. E, por fim, agradeço à Grazielly Sousa, ao lado de quem pude viver os melhores anos de minha vida. A todos, o meu mais sincero obrigado!

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

RESUMO

Motivado pelo crescente aumento da violência, principalmente a relacionada à invasão de residências e empresas, este trabalho apresenta o desenvolvimento de um sistema capaz de controlar remotamente, via chamada de telefone celular usando a tecnologia GSM, a abertura da tranca de uma porta. Esse sistema é capaz de armazenar informações sobre os horários de abertura da tranca de uma porta juntamente com as do solicitante dessa abertura e enviá-las ao administrador desse sistema via mensagem SMS. Para tanto, foram usados um kit de comunicação GSM/GPRS, um computador e um servo motor para a simulação de abertura da tranca e desenvolvidos um programa para a interação entre o celular do usuário e o sistema e um *drive* de controle para o motor que abre a tranca da porta. Para o armazenamento das informações foi utilizado o sistema gerenciador de banco de dados MySQL. A ideia do sistema é simples: existem dois tipos de usuários, o usuário comum, que para abrir a tranca da porta realiza uma ligação de um celular GSM, e o administrador, que gerencia todo o sistema, seja por meio de uma ligação de celular, seja pelo recebimento de informações via mensagem SMS dos eventos. Para a tranca da porta ser aberta, o número do telefone do usuário e os horários nos quais ele pode efetuar a abertura da tranca devem estar cadastrados no sistema. Foram feitos vários testes visando testar a robustez do sistema a falhas. Os testes mostraram que o sistema funcionou como projetado, ou seja, usuários foram capazes de abrir remotamente a tranca da porta nos horários permitidos, permissões de abertura da tranca foram negadas quando usuários não cadastrados no sistema ou que fizeram chamadas em horários não permitidos tentaram abrir a tranca da porta e o administrador recebeu todas as informações referentes aos eventos ocorridos. O sistema se portou de forma mais rápida que a esperada. Seu tempo de resposta médio foi calculado em 8 segundos e em todos testes o administrador foi informado dos eventos ocorridos em menos de um minuto via mensagem SMS. O sistema funcionou sem apresentar nenhum tipo de erro em todos os cerca de 80 testes realizados, superando com folga o requisito mínimo de atender 30 solicitações sem falhas.

Palavras Chave: controle remoto de tranca; tecnologia GSM.

ABSTRACT

In this work it is presented a system developed to remotely control a door's lock through a GSM mobile phone call. This project was motivated by the increasing criminal rate in our society, namely the unauthorized access to residences and commercial buildings. The system concept is simple. There are two types of users: the ordinary one, who can open the door's lock by means of a GSM mobile phone call, and the administrator, who can manage the whole system either by a GSM mobile phone call or by sending SMS to the system. In order to remotely open the door, the user´s GSM cell phone number, along with the time period allowed to access the system, must be inserted into the system`s database. The system stores, in a MySQL database, information regarding the door´s opening procedure, including data about the user responsible for sending such a command and the time tag for the command sent. It sends these data to the system´s administrator via SMS. The system encompasses a GSM/GPRS communication kit, a computer, and a servo motor to emulate the door´s opening and locking. A computer program to control the whole system was developed, which includes the handling of the interaction between system and users and a drive to control the servo motor responsible for opening the door. Several tests were performed in order to test the performance of the system and to check its tolerance to failures. The tests showed that the system met the design goals: registered users had permission to remotely open a door´s lock via a GSM cell phone, at the time period allocated for that, non-registered users have their permission denied, and the system administrator was informed of all such events. The system responded faster than expected, with a mean response time of 8 seconds, and in all tests the administrator was informed of all events in less than one minute by an SMS. It worked as designed in all the nearly 80 tests performed, easily surpassing the minimum specified number of 30 calls without failure.

Subject Readings: Door`s lock remote control; GSM technology.

CAPÍTULO 1 – INTRODUÇÃO

Este capítulo mostra uma visão geral deste trabalho. Ele apresenta a motivação, o objetivo e os resultados esperados do projeto e mostra como este trabalho foi dividido em capítulos.

1.1. Motivação

Atualmente vivemos em um mundo em que a informação e a velocidade de acesso a essa informação são importantíssimos. Nossas atividades vêm sendo integradas aos novos recursos tecnológicos que surgem. A grande proposta para o futuro é incluir portabilidade a esses recursos.

Os aparelhos celulares são hoje o grande exemplo de avanço nessa área. Hoje, na terceira geração, esses aparelhos nos permitem acessar a internet, ouvir músicas, enviar arquivos e mensagens de texto, usar recursos de geoposicionamento por satélite (GPS) etc. Esses recursos podem ser utilizados praticamente em qualquer lugar, bastando estar dentro da área de cobertura da operadora do celular.

Outra grande preocupação atual é com relação à segurança. As pessoas saem de casa com medo e não sabem se, ao voltarem, encontrarão sua casa da mesma forma que a deixaram. Os índices de violência urbana estão em constante aumento. Os meios de comunicação em massa mostram histórias violentas cada vez mais comuns no cotidiano.

Mas esse não é um problema enfrentado apenas pelas pessoas comuns. Muitas empresas também possuem problemas relacionados à segurança. Para contornar essa situação, as empresas definem políticas de segurança interna e várias vezes a preocupação diz respeito ao acesso a uma determinada sala, ou depósito, por exemplo.

Em 2009, um acontecimento teve repercussão nacional: devido a falhas em um processo de segurança, não o patrimônio, mas informações sigilosas foram roubadas para proveito de outrem. Esse foi o caso do vazamento da prova do Exame Nacional do Ensino Médio (ENEM). Alega-se que um funcionário não autorizado entrou na sala onde estavam sendo armazenadas as provas e se apossou de uma delas com o gabarito.

O processo de apuração foi demorado, milhares de alunos de todo o país foram prejudicados, houve muita frustração por parte desses, os cofres públicos

sofreram, pois foi necessário fazer nova licitação para a elaboração da prova e, principalmente, a credibilidade do maior exame nacional para avaliar estudantes do Ensino Médio foi bastante contestada (O GLOBO, 2009).

Analisando esse caso surgem algumas perguntas básicas: Será que se a instituição mantivesse maior segurança isso teria acontecido? Será que se ficasse registrado quem entrou na sala isso não inibiria a ação do infrator? Será que se existisse um sistema que informasse um administrador no momento de abertura da porta, ele não poderia tomar providências imediatas que impossibilitariam tantos prejuízos?

Tendo por base essas questões e procurando integrar a questão de portabilidade, surgiu a ideia para este projeto: desenvolver um sistema que use uma chamada de um telefone celular para abrir uma tranca de uma porta e um sistema capaz de controlar e armazenar informações referentes às aberturas dessa tranca.

1.2. Objetivos

1.2.1. Objetivo geral

Este trabalho tem por objetivo aumentar o nível de segurança de um determinado local controlando remotamente a tranca de uma porta via tecnologia GSM (Sistema Global para Comunicações Móveis).

1.2.2. Objetivo específico

Desenvolver um sistema que permita o controle remoto da tranca de uma porta utilizando-se de tecnologia do GSM. Neste sentido, será apresentado um protótipo que simule a tranca de uma porta controlada remotamente pelo sistema.

1.3. Resultados Esperados

Espera-se que o sistema só permita a abertura da tranca da porta por números de celular com cadastro no banco de dados do sistema e que estejam dentro do seu horário permitido. O sistema deverá informar ao usuário se a solicitação de abertura da porta foi aceita ou não e informar ao administrador do sistema sobre essa solicitação em até um minuto; e deverá abrir a tranca em até 10 segundos após a primeira chamada da solicitação. O sistema deverá ser capaz de reconhecer o administrador e dar permissão exclusiva a ele para os comandos SMS. O sistema deverá armazenar as informações, no banco de dados, sobre todas as solicitações de abertura de tranca

realizadas. O sistema deverá, também, se manter operacional sem que ocorram erros por, no mínimo, 30 solicitações de abertura de tranca.

1.4. Não Contempla o Escopo do Projeto

O escopo desse projeto não contempla a questão de falha por baixo sinal de rede GSM, ou por falta de créditos para realização de chamadas de usuários. Não pode ser atribuído a ele, igualmente, problemas de comunicação com uma operadora de telefonia móvel. Ainda, por questões financeiras, não contempla problemas de alimentação elétrica do sistema, ficando a sugestão de que para um melhor uso desse sistema ele tenha alimentação própria e independente da rede elétrica comercial.

1.5. Estrutura do Trabalho

Além deste capítulo introdutório, esta monografia está dividida em mais cinco capítulos, que serão brevemente descritos a seguir.

O segundo capítulo trata da descrição do projeto. Nele é mostrado um esquema de todo o projeto sendo detalhada cada parte que o engloba. O *hardware* utilizado no projeto, juntamente com suas especificações, e os motivos que levaram à sua escolha também são mostrados neste capítulo.

O terceiro capítulo apresenta o desenvolvimento deste projeto. É discutida a lógica do programa que controla todo o sistema e mostrado um fluxograma que ilustra o seu funcionamento.

O quarto caítulo trata da operação do sistema desenvolvido, mostrando a forma como os usuários comuns e o administrador devem se portar perante o sistema.

O quinto capítulo apresenta os testes realizados e os resultados obtidos. São discutidas possíveis falhas do sistema e como ele responde a essas falhas.

O sexto capítulo é a conclusão deste trabalho. Ela trata da comparação entre os resultados esperados com os resultados obtidos, apresenta propostas para melhorias futuras do projeto e sugere possíveis aplicações do sistema desenvolvido em outros campos.

CAPÍTULO 2 – DESCRIÇÃO DO PROJETO

Neste capítulo é feita uma descrição detalhada do projeto, apresentando seus principais componentes e as formas como eles interagem entre si. São mostrados os tipos de usuários que podem operar o sistema e apresentado o diagrama de blocos do sistema, assim como as integrações entre os componentes.

2.1. Diagrama de Blocos do Projeto

A Figura 2.1 apresenta o diagrama de blocos deste projeto no qual podem ser observados os principais componentes inseridos neste trabalho.

O quadro representando o Sistema de Controle da Tranca é a parte desenvolvida neste trabalho: foram criados o programa, o banco de dados e o *drive* do motor e feita a integração entre eles.

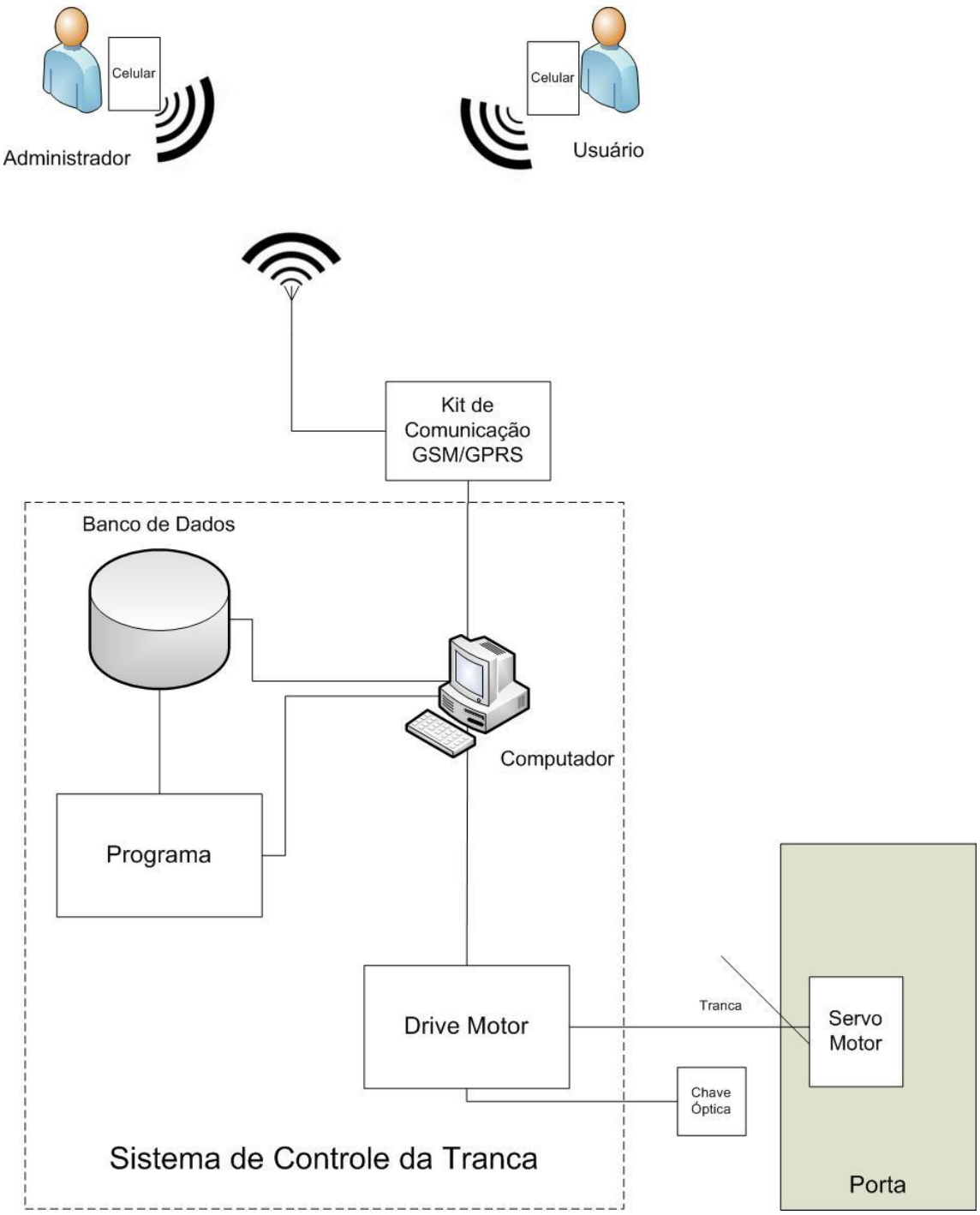


Figura 2.1 – Visão geral do projeto.

Como dito anteriormente, existem dois tipos de usuários: o comum e um administrador. Estes tipos de usuários possuem permissões diferentes para operar o sistema.

O usuário comum possui apenas a permissão para abrir a tranca em um horário específico. Para tanto, esse usuário realiza uma ligação para o número de

operação do sistema, número esse referente ao cartão SIM inserido no kit de comunicação GSM/GPRS. O kit GSM/GPRS recebe essa ligação e informa ao sistema o número que está chamando. O sistema, através do computador, aciona o programa desenvolvido repassando os dados referentes ao número solicitante e o horário de solicitação.

O programa, ao analisar as informações recebidas, consulta o banco de dados para verificar as permissões para o usuário solicitante. O programa analisa as informações do usuário cadastradas no banco de dados. Depois de verificadas as permissões, o sistema registra, neste mesmo banco de dados, as informações sobre a referida solicitação incluindo se a permissão foi negada ou concedida.

Após obter a resposta do programa, o sistema analisa esta resposta da seguinte forma:

- Caso a resposta seja de permissão concedida, o sistema acionará, através do computador, o *drive* do motor e este por sua vez enviará um sinal ao servo motor para a abertura da tranca, que ficará aberta por 15 segundos. Logo após, o programa gera duas mensagens SMS que serão enviadas ao kit GSM/GPRS que as transmitirá para o administrador e para o usuário solicitante da abertura da tranca.
- Caso a resposta seja de permissão negada, o sistema então irá verificar no banco de dados se esta solicitação já foi realizada na última hora. Em caso afirmativo, ele irá apenas ignorar esta solicitação; em caso negativo, o sistema acionará o programa que irá gerar duas mensagens, uma para o administrador e outra para o usuário solicitante informando da negação da solicitação.

Além deste comando de abertura da tranca por ligação, o administrador ainda tem a possibilidade de enviar outros dois comandos via SMS. Um para verificar aposição da tranca, se ela está aberta ou fechada, e o outro para modificar esta posição. Apenas o administrador pode manter a tranca aberta por tempo indeterminado.

Independentemente do comando enviado pelo administrador, o modo como o sistema irá se portar será parecido. Em todos os casos, o sistema, através do computador, acionará o programa que, por sua vez, irá verificar as informações do solicitante no banco de dados, para certificar-se de que ele é realmente o administrador.

Basicamente, após essa verificação, o computador aciona o *drive* do motor conforme o comando enviado pelo administrador inicialmente.

2.2. Tipos de Usuários

O projeto desenvolvido teve como objetivo contribuir para se obter maior segurança no controle ao acesso de um determinado local. Foram desenvolvidos dois tipos de usuários nesse projeto: o administrador e os usuários comuns.

2.2.1. Administrador

O administrador é o responsável pelo controle de todo o sistema. Ele será o responsável pelo cadastro dos usuários comuns no banco de dados. É o único com acesso aos comandos via SMS para verificar a posição da tranca da porta ou para modificar esta posição entre aberta e fechada.

O administrador será informado, via mensagem SMS, toda vez que houver uma solicitação de abertura da porta oriunda de um usuário comum, desde que não seja uma solicitação repetida e negada dentro do espaço de tempo de uma hora, e se esta solicitação de um usuário comum foi concedida ou não.

Ele ficará responsável pela manutenção da linha telefônica do número de operação do sistema e por outros aspectos de manutenção do sistema, tais como a do motor da tranca, a atualização de cadastros etc.

2.2.2. Usuários comuns

Estes usuários poderão apenas realizar um tipo de comando: abrir a tranca por um período de 15 segundos. Eles possuirão cadastro de permissão com o horário permitido para abrir a porta. Para poder solicitar a abertura da tranca, basta que estes usuários liguem para o número do sistema.

A solicitação desse usuário será concedida ou não dependendo da situação de seu registro. Em todos os casos, este usuário irá receber uma mensagem SMS informando sobre o estado da solicitação, desde que essa solicitação já não tenha sido negada dentro de um período de uma hora. Caso essa solicitação seja negada devido ao

fato de o horário ser indevido, ele será informado do motivo da negação da solicitação. Caso a solicitação seja negada pelo fato de este usuário não estar cadastrado, ele será informado que não possui cadastro.

O usuário comum é responsável pela manutenção de seu número de telefone cadastrado no sistema. Caso ocorra qualquer tipo de problema com esse número, o usuário comum deverá contatar o administrador para que este possa tomar as medidas cabíveis à situação.

Como este projeto visa restringir e controlar o acesso de pessoas a um determinado local (uma única porta), o sistema foi planejado para atender um número restrito de usuários. Assim sendo, o sistema se restringe ao cadastro de, no máximo, 10 usuários. O sistema, na atual configuração, não se adéqua a locais de alto fluxo de pessoas, pois o administrador não teria como controlar muitas informações ao mesmo tempo e ainda poderia ocorrer um gargalo na porta em que se encontra a tranca. No entanto, nada impede que alterações futuras sejam feitas no sistema para que ele se adéque a outras situações que exijam controlar várias portas e o acesso de várias pessoas a um determinado local. Tais situações são exploradas no Capítulo 6.

2.3. *Hardware*

Esta seção aborda o *hardware* envolvido neste projeto. Antes de tratar especificamente desse *hardware,* serão abordados alguns tópicos sobre o GSM, o SMS e o cartão SIM e as razões que levaram às escolhas desses padrões de tecnologia para esse projeto.

Global System for Mobile Communications, ou Sistema Global para Comunicações Móveis (GSM: originalmente *Groupe Special Mobile*), é uma tecnologia móvel e o padrão mais popular para telefones celulares do mundo. Telefones GSM são usados por mais de um bilhão de pessoas em mais de 200 países.

Por ser bastante comum, a presença do sistema GSM faz com que o *roaming* internacional seja muito comum por meio de "acordos de *roaming*" entre operadoras de telefonia móvel. O GSM diferencia-se muito de seus antecessores sendo que o sinal e os canais de voz são digitais, o que significa que o GSM é visto como um sistema de telefonia celular de segunda geração (2G). Esse fato também significa que a comunicação de dados foi acoplada ao sistema logo no início. O GSM é um padrão aberto desenvolvido pela 3GPP.

O GSM possui uma série de características que o distinguem dentro do universo das comunicações móveis. Nascido nos anos 80 e fruto de uma cooperação sem precedentes dentro da Europa, o sistema partilha elementos comuns com outras tecnologias utilizadas em telefones celulares, como a transmissão ser feita de forma digital e a utilizar células.

Do ponto de vista do consumidor, a vantagem principal do GSM é o oferecimento de serviços novos com baixos custos. Por exemplo, a troca de mensagens de texto foi originalmente desenvolvida para o GSM. A vantagem para as operadoras tem sido o baixo custo de infraestrutura causada por competição aberta. A principal desvantagem é que o sistema GSM é baseado na rede TDMA (*Time Division Multiple Access*), que é considerada menos avançada que a concorrente CDMA (*Code Division Multiple Access*). O desempenho dos aparelhos celulares atualmente é muito similar, mas se comparados aos aparelhos da primeira geração são bem superiores, apesar disso, o sistema GSM tem mantido compatibilidade com os telefones GSM da primeira geração. Ao mesmo tempo, o sistema GSM continua a ser desenvolvido com o lançamento do sistema GPRS. Além disso, a transmissão de dados em alta velocidade foi adicionada no novo esquema de modulação EDGE. A versão de 1999 do padrão introduziu índices relativamente altos de transmissão de dados, e é normalmente referida como 3G.

O método utilizado pelo GSM para gerir as frequências é uma combinação de duas tecnologias: o TDMA e o FDMA (*Frequency Division Multiple Access*). O FDMA divide os 25 MHz disponíveis de frequência em 124 canais com uma largura de 200 kHz e uma capacidade de transmissão de dados da ordem de 270 kbps. Uma ou mais dessas frequências é atribuída a cada estação base e dividida novamente, em termos de tempo, utilizando o TDMA, em oito espaços de tempo (*time slots*). O terminal utiliza um *time slot* para recepção e outro para emissão. Eles se encontram separados temporalmente para que o celular não possa receber e transmitir sinais ao mesmo tempo. Essa divisão de tempo também é chamada de *full-rate*. As redes também podem dividir as frequências em 16 espaços, processo designado como *half-rate*, mas a qualidade da transmissão é inferior.

A voz é codificada de uma forma complexa, de forma que erros na transmissão possam ser detectados e corrigidos. Em seguida, a codificação digital da voz é enviada nos espaços de tempo, cada um com uma duração de 577 milissegundos e uma capacidade de 116 bits codificados. Cada terminal deve possuir uma agilidade de

frequência, podendo deslocar-se entre os espaços de tempo utilizados para envio, recepção e controle dentro de um quadro completo. Ao mesmo tempo, um celular verifica outros canais para determinar se o sinal é mais forte e mandar a transmissão para eles, caso a resposta seja afirmativa (CERNE, 2010).

Devido ao fato de o GSM ser a tecnologia mais utilizada para as comunicações móveis atualmente ele foi escolhido para o desenvolvimento deste projeto.

O GSM utiliza-se de cartões SIM, ou *SIM Card,* que são circuitos impressos do tipo *smart card.* O nome SIM deriva da expressão inglesa *Subscriber Identity Module* que em português significa módulo de identificação do assinante. Esses cartões SIM armazenam o número do telefone e podem armazenar agendas telefônicas, mensagens SMS, informações do usuário, entre outros. Possuem duas versões e vários tamanhos diferentes. Atualmente, o mais utilizado é o cartão SIM de 128 kB pertencente à versão 2.0. Servem também para identificar um número na rede GSM, assim como fazer a autenticação nessa rede.

Já o SMS (Serviço de Mensagens Curtas ou *Short Message Service*) é um serviço disponível em telefones celulares digitais que permite o envio de mensagens curtas (até 255 caracteres em GSM e 160 em CDMA) entre estes equipamentos e entre outros dispositivos de mão como *palm* e *handheld*, e até entre telefones fixos (linha fixa).

O SMS originalmente foi projetado como parte do GSM, mas está agora disponível num vasto leque de redes, incluindo redes 3G. Já se discute e se planeja sua evolução através do MMS (*Multimedia Messaging Service*). Com o MMS, os usuários podem enviar e receber mensagens não mais limitados aos 255 caracteres do SMS, bem como podem enriquecê-las com recursos audiovisuais, como imagens, sons e gráficos.

O primeiro SMS foi enviado em dezembro de 1992 de um computador pessoal (PC) a um telefone celular da rede GSM da Vodafone no Reino Unido. O termo “torpedo” é utilizado no Brasil para designar o nome das mensagens escritas que são enviadas para e pelo celular (CERNE, 2010).

Observando esses conceitos fica mais fácil o entendimento sobre o kit de comunicação GSM/GPRS, componente utilizado nesse trabalho.

2.3.1. Kit de comunicação GSM/GPRS

O kit de comunicação GSM/GPRS é quem faz a comunicação entre o sistema e o usuário neste projeto. Esse kit já vem com circuitos integrando o módulo GSM ao cartão SIM, além de oferecer comunicação no padrão RS232 através de uma porta serial. Este kit é alimentado por uma fonte de 12 V com corrente mínima de 1 A.

Dentro do kit podemos observar os seguintes elementos: módulo SIM340Z, conector para o cartão SIM, entrada DC de 12 V, dois *leds*, um botão para ligar e desligar o kit e uma porta de comunicação serial RS-232. A Figura 2.2 apresenta o kit de comunicação GSM/GPRS.

Figura 2.2 – Kit de comunicação GSM/GPRS.
FONTE: Cerne Tecnologia.

Pode-se observar a entrada serial no canto inferior esquerdo da placa (7), a entrada DC de 12 V no canto inferior direito (1) e acima da entrada DC estão os dois *leds* que indicam se o kit está sendo utilizado (2). Após os *leds*, no canto superior direito, observa-se o *slot* para acoplar o cartão SIM (3); ao seu lado, observa-se o componente mais importante do kit, o módulo SIM340 (4). Logo abaixo do módulo encontra-se o botão de ligar e desligar (5). Por fim entre a entrada serial e a entrada DC se encontra um Max 232 (6) que converte os sinais de 5 V para 3,3 V.

2.3.2. Módulos GSM

Os módulos GSM são utilizados em manutenções remotas, transporte, logística, sistemas de tráfego, *gateways* celulares, *vending machines*, sistemas de segurança e de saúde, entre outros. Esses módulos geralmente possuem um padrão de comunicação serial que utilizam os comandos AT. Isso possibilita sua aplicação em diferentes projetos e, ainda, a flexibilização de projetos, uma vez que, possuindo um certo padrão, esses módulos podem ser trocados sem maiores alterações do projeto em que foi inserido.

Existem vários módulos GSM diferentes no mercado, uns com mais recursos que outros, com preços variados e com possibilidades de variadas aplicações. Por exemplo: primeiramente, esse projeto seria sobre uma monitoração de rede elétrica em um módulo GSM seria usado para informar ao usuário quando faltasse energia nesta rede elétrica.

2.3.3. Módulo SIM340

Desenvolvido para o mercado mundial, o módulo SIM340 é uma ferramenta quad-band GSM/GPRS que trabalha nas frequências GSM 850 MHz, GSM 900 MHz, DCS 1800 MHz e PCS1900 MHz. Ele possui um tamanho pequeno (40mm x 33mm x 2.85 mm) e, portanto, pode ser incluído em praticamente qualquer aplicação, tais como: *smartphones*, PDAs, entre outros serviços de telefonia móvel. Na Figura 2.3 pode-se observar o Módulo SIM340 (SIMCOM, 2006).

Figura 2.3 – Módulo SIM340
FONTE: Cleapseletroeletronica

No Quadro 2.1 podem ser observadas algumas especificações do módulo SIM340.

Quadro 2.1 – Especificações do módulo SIM340

Alimentação	3,4 a 4,5 V
Temperatura de Operação	-20º C a 60º C
Peso	8 g
Frequências de Operação	Quad-Band (850, 900, 1800 e 1900 MHz)

Devido ao fato de este módulo ter sido usado com êxito em outros projetos, além de ser mais popular do que os módulos pesquisados, ele foi escolhido para este trabalho. Não foi encontrada nenhuma empresa que vendesse esse módulo SIM340 separadamente. Desta forma, o módulo foi obtido juntamente com o kit de comunicação GSM/GPRS.

2.3.4. Servo motor

Os servos motores possuem uma grande aplicabilidade e funcionalidade, que se estendem desde o setor da robótica de pequeno porte até as indústrias e seus dispositivos automáticos. Para movimentos de rotação e movimentos lineares os servos motores são os motores elétricos mais indicados. São utilizados principalmente em aplicações de posicionamento. Eles transformam os sinais elétricos recebidos do computador em movimentos de rotação ou em deslocamentos lineares precisos no motor. Existem vários modelos de servos motores, com diferentes tipos de motores e com diferentes processos de realimentação (FRANCISCO, apud MELO, 2010).

A Figura 2.4 apresenta o servo motor utilizado na realização deste trabalho.

Figura 2.4 – Servo motor

O servo motor utilizado neste trabalho é o HXT900 da empresa HEXTRONIK. Suas principais características são descritas no Quadro 2.2.

Quadro 2.2 – Características do servo motor

Torque	1,60 kg/cm
Velocidade	0,12 s/60º
Peso	9,1 g
Dimensões	Comprimento - 21 mm Largura – 12 mm Altura – 22 mm

Fonte: SERVODATABASE, 2010.

O servo motor é ligado ao computador pela porta paralela. Desta forma, o sistema aciona o *drive* do motor que envia os sinais elétricos pela porta paralela. A comunicação com a porta paralela é descrita na Seção 2.4.

O controle do ângulo de rotação do servo motor é dado pela duração do impulso elétrico que o motor recebe (FRANCISCO, apud MELO, 2010).

A escolha do servo motor para este projeto se deu por alguns motivos: simular de maneira simples a abertura e o fechamento de uma tranca, já que, devido às suas características, ele permite um controle específico sobre a sua posição de giro, o que permite a verificação sobre um possível posicionamento da tranca juntamente com o auxílio de uma chave óptica.

2.3.5. Chave óptica

As chaves ópticas são componentes que consistem em um emissor de luz (um LED infravermelho, por exemplo) e um receptor que, dependendo da aplicação do dispositivo, pode ser um fotodiodo, foto-transistor, foto-diac, foto-disparador, entre outros. São dispositivos que envolvem o uso da luz como meio de transmissão de sinais de controle e informações. São bastante comuns nas aplicações modernas e podem ser encontrados em inúmeras versões.

O acionamento das chaves ópticas é feito por algum tipo de objeto que se interponha ao feixe de luz que vai do elemento transmissor (LED) ao elemento receptor (que pode variar conforme a aplicação). A luz do elemento emissor incide no elemento

sensor através de uma abertura. Quando um objeto se interrompe ao feixe de luz fenda, um sinal de comando é produzido no sensor (BRAGA, 2010).

A chave óptica foi utilizada neste trabalho para verificar se a tranca estava de fato trancando a porta. Ela se localiza exatamente na posição de encaixe da tranca. O sistema analisa se o feixe de luz está sendo interrompido no momento em que a tranca estiver na posição fechada. Caso a tranca não esteja trancando a porta, apesar de estar na posição fechada, ou seja, caso a tranca se feche com a porta aberta, o sistema passa a realizar ligações para o administrador de 30 em 30 segundos até que a situação seja resolvida. A Figura 2.5 apresenta a chave óptica utilizada no trabalho.

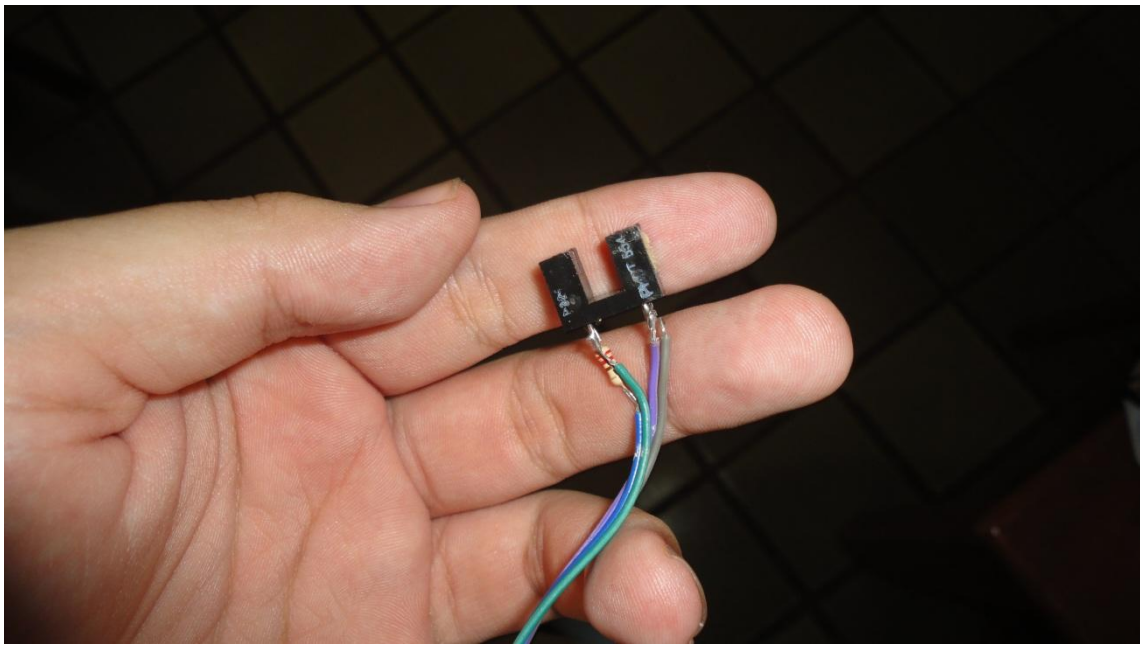


Figura 2.5 – Chave óptica

2.3.6. Computador

Em termos de *hardware*, o computador é a peça central desse projeto e é o responsável por realizar a integração de todo o sistema. O computador armazena o programa que se comunica com o kit GSM/GPRS, o banco de dados, que contém os cadastros e registros, e o *drive* do motor, que realiza a abertura ou o fechamento da tranca.

É de fundamental importância para esse projeto que o tempo de resposta do sistema não seja muito aleatório. A segurança depende disto. O computador não é o

responsável pelo atraso do sistema e, portanto, pode-se utilizar um computador com configurações bem básicas. O Quadro 2.3 apresenta as características do computador utilizado neste projeto.

Quadro 2.3 - Computador utilizado

Componentes	Requisitos
Processador	Frequência de 1,0 GHz
Memória	256 MB
HD	40 GB
Porta Serial	Uma porta serial disponível
Porta Paralela	Porta paralela disponível
Sistema Operacional	Windows XP ou superior

Conforme dito anteriormente, não é necessário o uso de um computador com grande capacidade de processamento e poderia até ser um computador de menor capacidade do que este utilizado. O grande responsável pelo atraso do tempo no sistema é o tempo de abertura e fechamento da tranca. Não se pode reduzir muito esse tempo, pois ele depende mais do ser humano (tempo de passagem pela porta) do que da própria máquina.

O computador requer a presença de uma porta serial e uma porta paralela livres para se conectar os componentes. A Figura 2.6 apresenta as conexões do computador com os componentes.

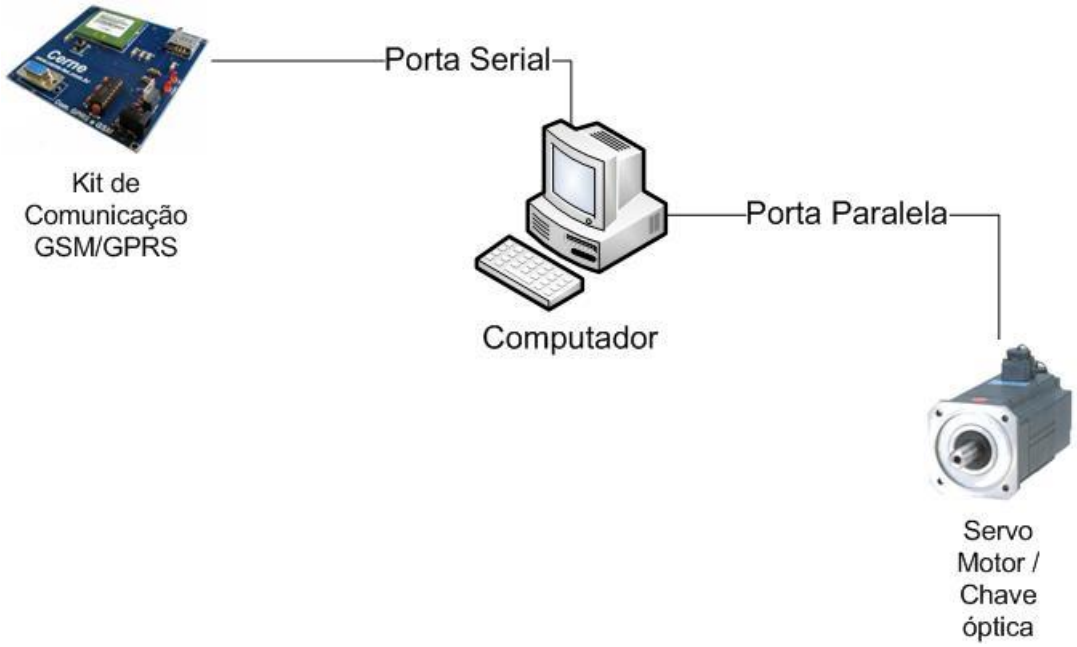


Figura 2.6 – Conexões do computador

2.3.6.1. Porta Serial

A porta serial é uma porta de comunicação com periféricos externos ao computador. Era muito utilizada para se conectar modems, *mouses*, algumas impressoras, *scanners*, entre outros.

Também conhecida como RS-232, a comunicação serial faz a transmissão de dados informando os bits sequencialmente, ou seja, é enviado um bit de dados de cada vez.

O padrão RS-232 original possuía 25 fios diferentes, porém a IBM (*International Business Machines*), ao utilizar este padrão para o seu projeto IBM-PC, modificou a quantidade de fios para 9 (WIKIPEDIA, 2010).

Neste trabalho, a porta serial é utilizada para fazer a comunicação entre o sistema e o kit de comunicação GSM/GPRS. Através dela são enviados os comandos para a operação do kit.

2.3.6.1.1. Comunicação síncrona

Neste tipo de comunicação, o emissor e o receptor da mensagem sincronizam os bits iniciais através de um *clock*. Não existem os bits de parada.

É mais suscetível a erros por conta deste sincronismo. Uma vez quando a sincronia entre ambos é perdida, é necessário começar uma nova comunicação entre ambos.

2.3.6.1.2. Comunicação assíncrona

A comunicação assíncrona não exige que o emissor e o receptor estejam em sincronismo, no entanto, ela exige que sejam enviados bits de começo e de parada para o seu correto funcionamento. Esses bits são inseridos no começo e no final de cada mensagem enviada indicando o seu começo (start bit), e o seu final (*stop bit*) (ISAAC, 2009).

Este tipo de comunicação é mais simples de ser implementada do que a comunicação assíncrona. Ela exige códigos menos complexos e por este motivo foi escolhida para a comunicação entre o computador e o Kit GSM/GPRS deste projeto.

2.3.6.2. Porta paralela

A porta paralela é uma interface de comunicação entre um computador e um periférico. A IBM foi uma das primeiras empresas a desenvolver a porta paralela como uma maneira de ligar uma impressora a um computador. A porta paralela não é usada apenas para se comunicar com a impressora: pode-se desenvolver um circuito eletrônico e acoplá-lo a essa porta e, através de um programa específico, enviar-lhe sinais digitais, por meio dessa porta, para controlá-lo.

Diferentemente da comunicação serial, na comunicação em paralelo os grupos de bits são transferidos simultaneamente, em geral, *byte* a *byte*, através de diversas linhas condutoras dos sinais. Dessa forma, como vários bits são transmitidos simultaneamente a cada ciclo, a taxa de transferência de dados (*throughput*) é alta (ROGERCOM, 2010).

As portas paralelas podem ser utilizadas para o controle de circuitos eletrônicos, ou motores. Neste trabalho essa porta é a responsável por fazer a integração do servo motor e da chave óptica com o sistema de controle da tranca.

2.4. Integração do Kit de Comunicação GSM/GPRS com o Sistema

O kit de comunicação GSM/GPRS está ligado diretamente ao computador através de uma porta serial. Como o computador está sobre controle do sistema, ele será comandado para enviar os sinais para o kit.

Os comandos que forem enviados estão todos dentro do programa desenvolvido. O kit de comunicação GSM/GPRS trabalha com comandos AT. Estes comandos AT são melhores descritos no capítulo 3 deste trabalho. O kit vem com uma antena para poder se comunicar com as redes GSM/GPRS e assim pode se comunicar com os telefones celulares dos usuários do sistema.

O kit é responsável basicamente por receber ligações, identificar os números e enviar as mensagens SMS geradas pelo sistema para os usuários. Após receber uma ligação ou mensagem SMS, o kit gera uma interrupção no programa através da comunicação com o sistema. O programa verifica que tipo de comando é e repassa essa informação para o sistema. Identificado o comando, o programa tomará as medidas referentes ao comando.

2.5. Integração do Programa com o Banco de Dados

O programa do sistema por várias vezes fará chamadas ao banco de dados. Neste banco estão contidas as informações referentes ao cadastro dos usuários e as solicitações realizadas.

O programa toda vez que é acionado pelo sistema se comunica com o banco de dados verificando o registro do solicitante. O banco verifica a situação desse registro e retorna para o programa. No Apêndice A é mostrado todo o código do programa e assim pode-se analisar de uma forma mais detalhada como são feitas essas chamadas.

2.6. Integração do *Drive* do Motor com o Servo Motor

Foi desenvolvido um *drive* para controlar o motor. Esse *drive* é responsável por verificar a posição do motor e se preciso alterá-la. Ele envia os sinais elétricos ao motor pela porta paralela do computador. O servo motor por sua vez recebe

os comandos do drive e então altera sua posição, abrindo ou fechando a tranca. No Capítulo 3 deste trabalho é mostrada de forma mais detalhada esta integração.

CAPÍTULO 3 – DESENVOLVIMENTO DO PROJETO

O objetivo deste projeto é contribuir para o aumento da segurança de um determinado lugar, mais especificamente controlar a abertura de uma porta. Sendo assim, ele visa gerenciar o controle das pessoas que entram e saem deste lugar.

Para isso, foi desenvolvido um sistema que monitora e gerencia tal controle. Primeiramente, nesse sistema será identificado o solicitante e a forma de solicitação que ele está fazendo. Após essa verificação, o sistema realiza a operação de acordo com o comando recebido.

Para tanto, foi criado um banco de dados do tipo MySQL, foi desenvolvido um programa para cadastro e verificação de usuários e ainda um *drive* de motor para controlar o servo motor que controla a tranca.

3.1. Linguagem de Programação

Existem hoje várias linguagens de programação que são classificadas devido à sua proximidade com a linguagem humana. Quanto maior for o nível da linguagem, maior a sua proximidade com a nossa linguagem cotidiana, geralmente com o inglês. Por outro lado, quanto menor o nível da linguagem, maior a sua proximidade com a linguagem de máquina, em que dados e instruções são tratados diretamente pela máquina em nível binário.

As linguagens de alto nível, porém, necessitam que suas instruções sejam compiladas para linguagens de máquina para assim poderem ser interpretadas.

A linguagem de baixo nível Assembly é bastante utilizada na programação de microcontroladores e traz algumas vantagens na sua utilização, entre elas: maior controle sobre o *hardware*, economia de código e memória do programa, além do tempo de execução dos programas em Assembly geralmente serem menores do que os de programas escritos em linguagem de alto nível.

Por outro lado, linguagens de alto nível possuem maior portabilidade, ou seja, programas feitos nestas linguagens podem ser utilizados em diferentes tipos de máquina, enquanto que o Assembly tem que ser feito um programa diferente para cada máquina. Além disso, os programas escritos em linguagens de alto nível são muito mais fáceis de serem interpretados por programadores (NICOLOSI, 2004).

Levando em conta essas vantagens da programação em linguagem de alto nível e pensando em flexibilizar o projeto para futuras aplicações foi feita a opção pela linguagem C para esse projeto. Além disso, há o fato de o C possuir várias bibliotecas, o que o torna ainda mais atraente.

3.2. Comandos AT

Os comandos AT foram originados de uma linguagem específica de comandos para modems, chamada *Hayes Command Set*, que se tornou um padrão para controle de modems, entre outros dispositivos. A sigla AT é um mnemônico de *Attention*, que significa atenção em português, sendo utilizada como prefixo para a maioria dos comandos utilizados.

Eles são comandos geralmente utilizados para controlar celulares e modems configuráveis, para a realização de chamadas, para verificações de funcionalidades e atributos de modems, para envio de mensagens, para ligações, entre outros.

Alguns dispositivos podem ter suas restrições a alguns comandos desse padrão. Os terminais e módulos GSM possuem suporte a praticamente todos os comandos do padrão AT e, geralmente, possuem alguns comandos extras para funções específicas do módulo com o intuito de facilitar a sua utilização pelo usuário (ISAAC, 2009).

3.3. Banco de Dados MySQL

O MySQL é um SGBD (sistema de gerenciamento de banco de dados) que utiliza a linguagem SQL (Linguagem de Consulta Estruturada, do inglês *Structured Query Language*) como interface. É um *software* com licença livre. Ganhou grande popularidade devido a esse fato. Ele busca velocidade, robustez e facilidade de uso. A base sob a qual o MySQL foi construído é formada por um conjunto de rotinas que foram utilizadas em ambiente de produção com alta demanda. Apesar de o MySQL

estar sempre em desenvolvimento, este sistema já oferece um conjunto de funções altamente útil. Funciona com a maioria dos sistemas operacionais, incluindo Linux, Unix (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) e Windows (MILANI, 2008).

É um banco de dados do tipo relacional, ou seja, ele é composto por tabelas e estas são compostas por linhas e colunas. As tabelas dentro do banco de dados podem estar relacionadas ou não, podendo muitas vezes uma informação contida em uma tabela completar a informação contida em outra tabela.

3.3.1. Características do MySQL

Algumas características interessantes do MySQL e que podem ser destacadas para o seu uso neste trabalho são:

- Portabilidade (é suportado em mais de 20 plataformas diferentes);
- Compatibilidade (existem *drivers* ODBC, JDBC e NET e módulos de interface para diversas linguagens de programação, como Delphi, Java, C/C++, Visual Basic, Python, Perl, PHP, ASP e Ruby);
- Excelente desempenho e estabilidade;
- Pouco exigente quanto a recursos de *hardware*;
- Facilidade de uso;
- É um *software* livre;
- É o mais popular SGBD livre do mercado;
- Replicação facilmente configurável;
- Interfaces gráficas (MySQL Toolkit) de fácil utilização cedidas pela MySQL Inc (MySQL, 2010).

O MySQL foi escolhido para este projeto por ser um software livre, de fácil utilização e de tempo de resposta considerado adequado para a solução criada, além de possibilitar uma maior flexibilidade do projeto devido ao fato de poder ser utilizado em várias plataformas diferentes.

3.3.2. Banco de dados do sistema

O banco de dados desenvolvido para esse sistema é de certa forma simples. Ele possui quatro tabelas: tb_usuarios, tb_solicitacoes, tb_log_adm e tb_solicitacoes_invalidas. A Figura 5.3 apresenta o modelo relacional do banco de dados criado para o sistema.

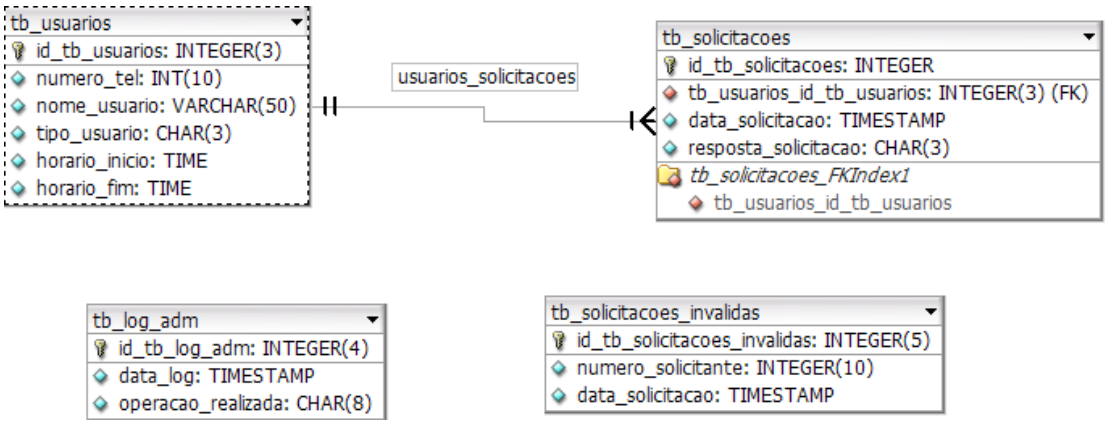


Figura 5.3 – Banco de dados do sistema

Observa-se que existe uma relação apenas entre as tabelas tb_usuarios e a tb_solicitaçoes no formato 1:n. A tabela tb_log_adm não se relaciona com nenhuma outra tabela do banco de dados pelo fato de ela ser reservada para os eventos realizados pelo administrador. Como o administrador é único, não é necessário criar um campo referenciando quem é o administrador. A tabela tb_solicitacoes_invalidas é referente às solicitações de números não cadastrados no sistema ou operações realizadas por usuários sem a devida permissão. Como a principal ideia é armazenar os eventos, e pode acontecer de os números não serem cadastrados no sistema, essa tabela também não se referencia com nenhuma outra.

3.3.2.1. Tabela tb_usuarios

A tabela tb_usuarios é a responsável por armazenar o cadastro de todos os usuários do sistema. Possui os campos numero_tel e nome_usuario referentes às informações pessoais do usuário. Pode-se, dependendo da necessidade de onde o sistema for empregado, criar novos campos referentes às informações dos usuários, tais

como, RG, endereço, CPF, entre outros, sem causar grandes modificações no sistema. Por se tratar apenas de um protótipo, esse nível de detalhamento referente aos usuários não se faz necessário.

Além desses campos, temos os campos referentes aos registros dos usuários no sistema, sendo eles: tipo_usuario, horario_inicio e horário_fim. O campo tipo_usuario serve para diferenciar os usuários comuns do administrador do sistema. Já os campos horario_inicio e horario_fim são referentes, respectivamente, ao inicio da permissão do usuário para poder abrir a tranca e ao término dessa permissão. O sistema não prevê intervalos no horário bem como não faz distinção entre os dias da semana, ficando, portanto, a sugestão de melhoria para projetos futuros.

3.3.2.2. Tabela tb_solicitacoes

Esta tabela armazena as informações referentes a todas as solicitações realizadas pelos usuários do sistema. Por esse motivo, ela se relaciona com a tabela tb_usuario possuindo a chave estrangeira tb_usuario_id_tb_usuario. Os campos data_solicitacao e resposta_solicitacao armazenam as informações de quando foi feita a solicitação e se esta foi atendida ou não.

3.3.2.3. Tabela tb_log_adm

É a tabela responsável por armazenar todos os eventos gerados pelo administrador do sistema. Ela não se referencia com nenhuma outra tabela por se tratar exclusivamente dos eventos do administrador. Como o administrador é único nesse sistema não se faz necessário armazenar as informações referentes a quem está realizando a operação. Dependendo da necessidade em que for inserido, pode-se desenvolver o sistema de forma a permitir a existência de mais de um administrador. Contudo, teriam que ser feitas algumas modificações consideráveis no sistema. Logo, fica outra sugestão de melhoria para futuros projetos.

3.3.2.4. Tabela tb_solicitacoes_invalidas

Foi criada com o intuito de se armazenar solicitações de números não cadastrados no sistema, ou solicitações de usuários comuns tentando realizar outra operação que não seja a abertura da tranca por 15 segundos. Pode ser utilizada de forma a procurar possíveis ataques ao sistema. Armazena as informações referentes ao número que está fazendo a solicitação, bem como a data e o horário da solicitação.

3.4. Programa Desenvolvido

Usando os recursos da linguagem de programação C e os recursos do banco de dados MySQL, foi desenvolvido um programa para poder atender as necessidades das propostas deste projeto. A lógica do programa é descrita na Figura 3.1.

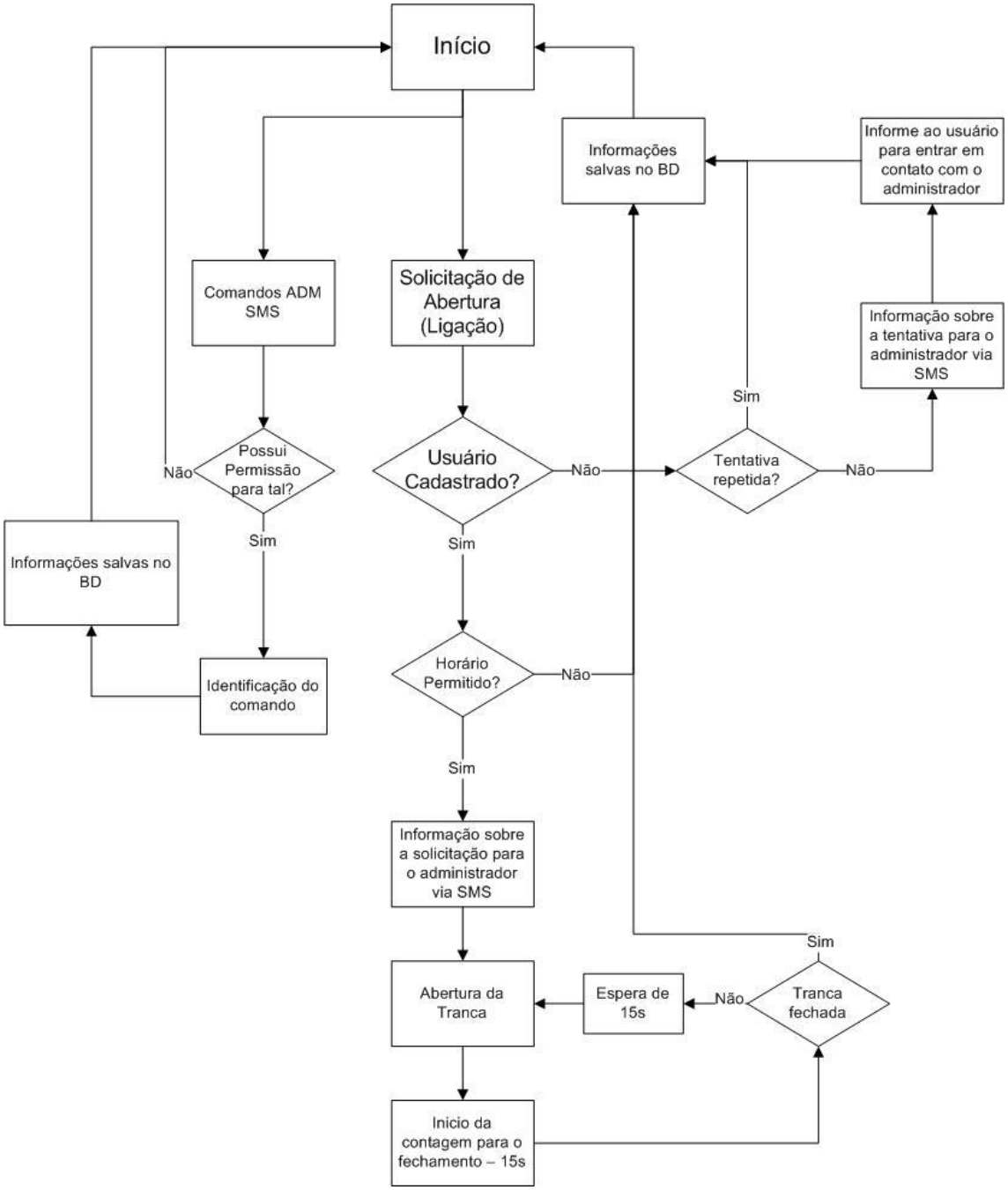


Figura 3.1 – Fluxograma do programa

Como visto anteriormente, existem dois tipos de usuário para este sistema. Procurando deixar menos complicada a explicação, será abordado um tipo de usuário por vez.

O primeiro a ser abordado será o usuário comum. Conforme já citado antes, este usuário tem a possibilidade de realizar apenas um comando: solicitar a abertura da tranca de uma porta por uma ligação oriunda de um telefone celular GSM. Realizada esta ligação, o sistema irá verificar se este usuário é cadastrado e autorizar ou não a abertura da tranca.

Se o usuário não for cadastrado, com o intuito de evitar ataques massivos ao sistema, este verifica se essa tentativa já foi realizada recentemente. Não sendo repetida, o sistema gera uma mensagem SMS para o administrador. Nesta mensagem será enviado o número que fez a solicitação de abertura da tranca e informado que este usuário não é cadastrado no sistema. Após enviada essa mensagem, o sistema gera outra mensagem SMS, só que agora para o solicitante informado que ele não possui permissão para abrir a tranca. O sistema volta então para o estado inicial, aguardando o recebimento de um comando. Caso o usuário faça essa tentativa, que foi negada, repetitivamente, o sistema passa a ignorar esse solicitante até que esse usuário esteja corretamente cadastrado e sua solicitação esteja no horário correto.

Sendo o usuário cadastrado, o sistema então passa a verificar se o horário da solicitação é um horário permitido para aquele usuário. Não sendo o horário permitido, o sistema volta a verificar se a solicitação é repetida e tomará as mesmas decisões que já foram descritas anteriormente. Se o solicitante estiver dentro do horário permitido, será gerada uma mensagem SMS para o administrador informando que a tranca está sendo aberta pelo solicitante. O sistema aciona o drive do motor que faz com que a tranca se abra. É disparado um *timer* de 15 segundos para o fechamento da tranca. Após a tranca ser fechada, o sistema salva no banco de dados as informações referentes ao horário de abertura da tranca e ao usuário que a realizou. Finalizado este processo, o sistema novamente volta ao estado inicial.

Analisando o administrador do sistema, percebe-se que ele possui este mesmo comando dos usuários comuns. Quando for realizada uma chamada para a abertura da tranca, o sistema se comporta semelhantemente ao usuário comum. É notada a diferença de comportamento do sistema quando esse usuário realiza um comando via SMS.

Este comando pode ser para verificar a posição da tranca ou ainda para mudar a posição da tranca. Se o comando for de verificação do estado da tranca, o sistema primeiramente verifica as permissões do solicitante. Depois dessa verificação, o sistema verifica a posição da tranca e a informa ao administrador. Se o administrador modificar a posição da tranca de fechada para aberta, esta ficará aberta por tempo indeterminado até que o administrador envie o comando de fechar.

As informações referentes a essas solicitações serão todas armazenadas no banco de dados do sistema e o sistema volta ao estado inicial, aguardando novos comandos.

A Figura 3.2 mostra o caminho realizado pelos comandos utilizados no sistema.

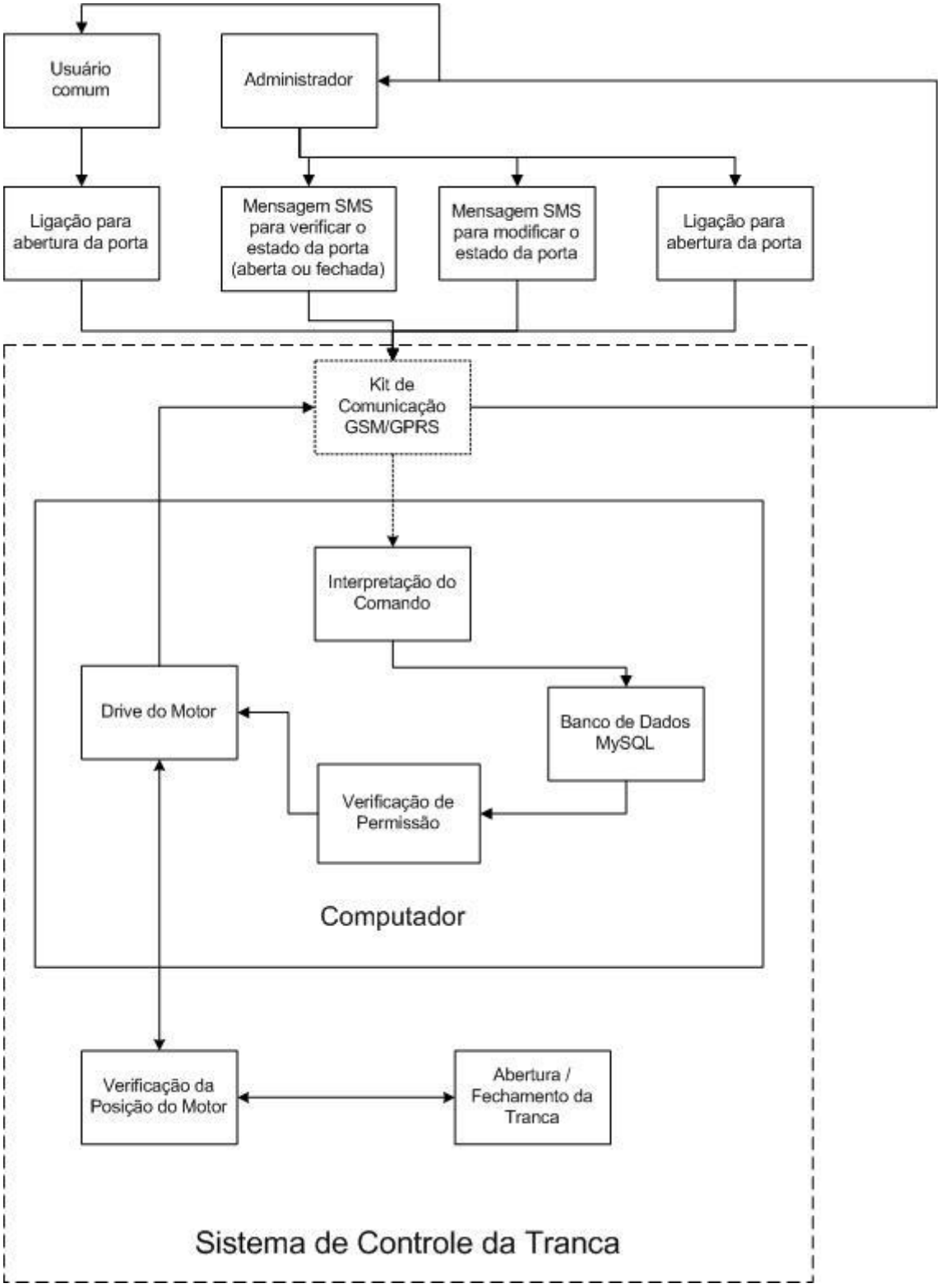


Figura 3.2 – Caminho dos comandos do sistema

3.4.1. Lógica do programa

O programa desenvolvido para o sistema tem como objetivo principal aguardar recebimento de comandos através do kit de comunicação GSM/GPRS, sendo estes ligações ou mensagens SMS. Primeiramente o programa configura a comunicação com a porta serial através da função “AbrePorta()”.

Pode-se observar que a função configura a porta COM5 a uma taxa de 115200 kbps. Ainda observa-se que é necessário acesso exclusivo para essa porta. O restante desta função bem como todo o programa pode ser encontrado no Apêndice A deste trabalho.

Após realizar a configuração da porta, o programa faz uma pequena configuração no módulo GSM e então passa a aguardar por recebimento de caracteres.

Quando um caractere é recebido o programa verifica se este é referente a uma ligação ou a uma nova mensagem SMS. Caso esteja recebendo uma ligação, o módulo envia a *string* “RING”, caso seja uma nova mensagem SMS a *string* enviada é “+CMTI”, ambas enviadas pela porta serial.

O programa identifica qual foi a *string* recebida e a trata conforme a lógica do sistema. Terminando todo o tratamento, o programa volta ao estado inicial aguardando um novo caractere ser recebido na porta. A Figura 5.2 apresenta um pequeno trecho do programa.

```
    armazenar_numero_adm();
    limpa_memoria();
    printf("Aguardando Comandos\n");
    while(1)
    {
        limpar_comando(comando_at);
        limpar_rxstring();
        espera_caractere();
        if(strncmp(rxstring,"RING",4)==0)
        {
            chamada_usuario();
        }
        else if(strncmp(rxstring,"+CMTI",5)==0)
        {
            comandos_adm();
        }
    limpar_rxstring();
    }
    Sleep(500);
    FechaPorta();
    FreeLibrary(hLib);

    return 0;

}

int AbrePorta()
{
    DCB dcb;
    BOOL fSuccess;
    char *pcCommPort = "COM5";
    int baudRate = 115200;
    hCom = CreateFile( pcCommPort,
                       GENERIC_READ | GENERIC_WRITE,
                       0,    // a porta deve ser aberta com acesso exclusivo
                       NULL, // sem atributos de segurança
                       OPEN_EXISTING,
```

Figura 5.2 – Identificação de chamada pelo programa

Neste trecho o programa observa-se o *loop* em que o programa fica aguardando o recebimento de novos dados pela porta serial. A função "espera_caractere()" é a responsável por identificar as *strings* de dados recebidas e armazená-las na variável "rxstring".

A função chamada usuário é uma das principais funções do programa. É ela que faz a verificação de cadastros de usuários, bem como de seus horários de permissão.

CAPÍTULO 4 – OPERAÇÃO DO SISTEMA

Neste capítulo será descrita detalhadamente a forma de operação do sistema. Portanto, serão mostrados todos os passos a que o usuário deverá estar atento para o manuseio correto do sistema.

4.1. Definição do Administrador

O primeiro passo para operar o sistema é definir quem será o seu administrador. Este usuário deverá ser cadastrado no banco de dados na tabela tb_usuarios como adm no campo tipo_usuario. No campo telefone deverá ser inserido o número do celular no qual o administrador deseja receber as informações sobre o sistema.

Para exemplificar, é mostrado um exemplo com os números e usuários que foram utilizados nos testes. O administrador desse sistema será "administrador". O tipo de usuário, portanto, fica definido como adm e o telefone será 8401-1602. A Figura 4.1 apresenta este cadastro do administrador.

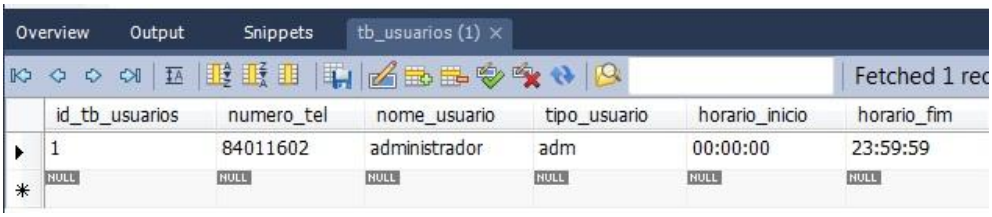


Figura 4.1 – Cadastro do administrador

4.2. Definindo o Número de Operação do Sistema

O usuário deve adquirir um cartão SIM, exclusivo para o sistema, da operadora de telefonia móvel de sua preferência, bem como o tipo de plano de sua preferência. Será de responsabilidade do usuário a manutenção dos créditos ou os pagamentos das contas deste número.

Continuando com o exemplo anterior, o número definido para esse sistema é o 82365638. Portanto, o cartão SIM inserido no kit de comunicação GSM/GPRS deverá ser o deste número.

4.3. Definindo os Usuários Comuns

Após a definição do administrador e do número do sistema, são cadastrados os usuários comuns do sistema. São definidos o nome, telefone para acesso, tipo de usuário e horário permitido. Quem deve realizar estes cadastros é o administrador, e estes cadastros, diferentemente dos cadastros de administrador e de número do sistema, podem ser realizados após o sistema estar em operação.

Para o cadastramento de usuários, o administrador deverá abrir o banco de dados e cadastrar os usuários. A Figura 4.2 apresenta o cadastramento de usuários.

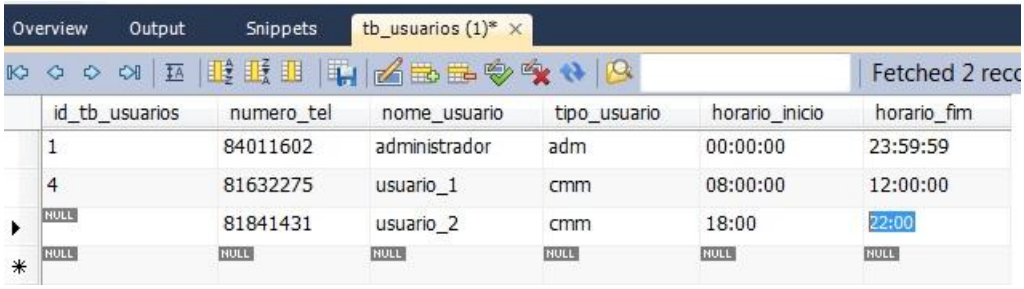


Figura 4.2 – Cadastro de usuários comuns

O próximo passo é fornecer as seguintes informações: nome do usuário, número de telefone, tipo de usuário, horário do começo da permissão para abrir a tranca da porta e horário do fim dessa permissão. No exemplo, serão cadastrados dois usuários: usuario_1 e usuario_2. O Quadro 4.1 mostra os registros desses usuários.

Quadro 4.1 – Usuários cadastrados

Nome	Telefone	Tipo de usuário	Começo da permissão	Fim da permissão
Administrador	84011602	adm	-	-
usuario_1	81632275	cmm	08:00	12:00
usuario_2	81841434	cmm	18:00	22:00

Conforme visto no Quadro 4.1, o sistema possui três usuários, incluindo o administrador. O usuario_1 tem permissão para a abertura da tranca da porta das 08:00 às 12:00. Já o usuario_2 possui permissão para essa abertura das 18:00 às 22:00. O sistema trabalha com uma tolerância de 5 minutos em relação ao tempo de cadastro, ou seja, o tempo real de permissão de abertura da tranca para o usuario_1, por exemplo, é das 07:55 às 12:05.

É de responsabilidade dos usuários e do administrador que os seus horários estejam sincronizados com o horário do sistema, que é definido pelo computador do sistema.

4.4. Realizando uma Ligação para a Abertura da Tranca

Após a realização do cadastro do usuário, esse agora possui a permissão de abrir a tranca da porta no horário a ele determinado. Para tanto, basta efetuar uma ligação do número de telefone cadastrado em seu nome para o número de operação do sistema.

Portanto, se o usuario_1 quiser abrir a tranca às 10:00, basta utilizar o celular cadastrado no sistema em seu nome, no caso 81632275, e realizar uma chamada para 82365638. A Figura 4.3 exibe uma chamada sendo realizada para o sistema.

Figura 4.3 – Ligação de solicitação de abertura da tranca

O sistema reconhecerá e desligará a ligação. Em seguida, fará as verificações de permissão desse usuário. Como o usuario_1 possui registro e permissão para esse horário, ele receberá uma mensagem SMS com o seguinte texto: “Permissão verificada, a tranca será aberta”. O administrador também receberá uma mensagem SMS, em até um minuto, informando: “O usuario_1 solicitou abertura da tranca às 10:00 e a tranca foi aberta”. A tranca então é aberta por um período de 15 segundos. A Figura 4.4 apresenta as mensagens recebidas pelo usuário e pelo administrador na respectiva ordem.

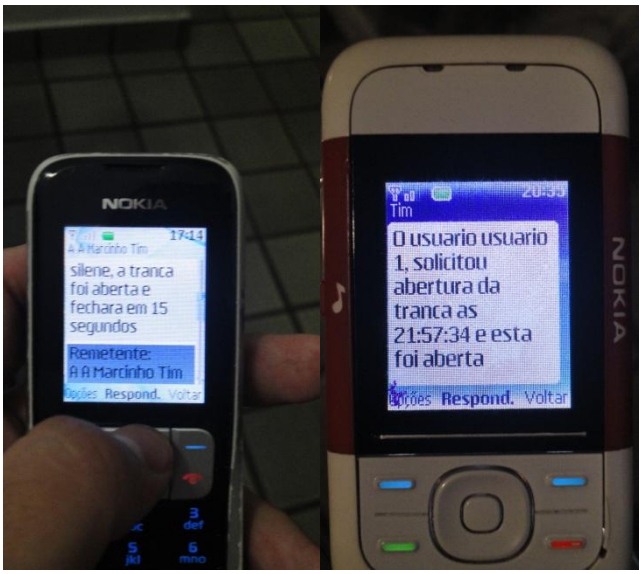


Figura 4.4 – Mensagens recebidas após confirmação de cadastro

Caso o usuario_1 tente realizar uma abertura fora do horário permitido, além de a tranca não ser aberta ele irá receber a seguinte mensagem: “Você não possui autorização para abrir a tranca neste horário”. O administrador, por sua vez, receberá a mensagem: “O usuario_1 está solicitando abertura da tranca fora do seu horário permitido”. A Figura 4.5 apresenta as mensagens recebidas pelo usuário e pelo administrador.

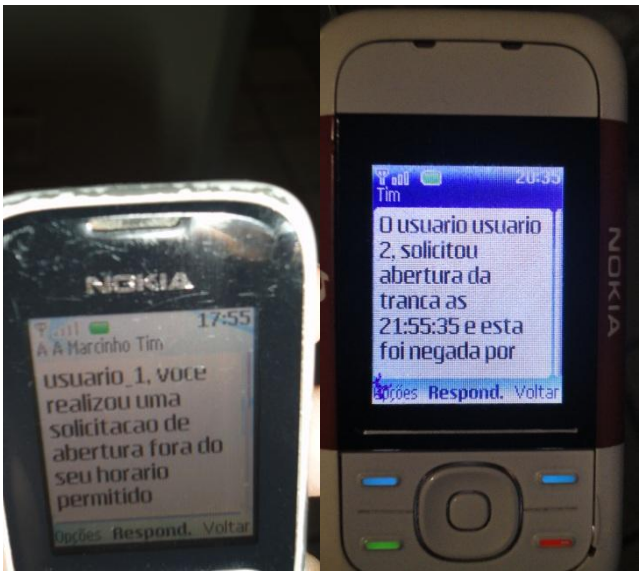


Figura 4.5 – Mensagens recebidas após a negação de solicitação

Se um usuário não cadastrado fizer a ligação para o número de operação do sistema, por exemplo, o número 55556666 ligar às 14:00, o sistema irá retornar uma mensagem para esse número informando: “Você não possui permissão para a abertura

da tranca. Entre em contato com o administrador no número 84011602, caso deseje se cadastrar no sistema". E uma mensagem será enviada ao administrador: "O número 55556666, que não possui cadastro, solicitou abertura da tranca às 14:00".

Visando não saturar o administrador do sistema com muitas mensagens, como no caso de possíveis ataques em massa de ligações para o número do sistema, se o número que efetuar a ligação não possuir cadastro e já tiver realizado outra ligação dentro de um período de 1h, o sistema não irá gerar mensagens nem para o solicitante nem para o administrador, porém irá registrar a solicitação na tabela tb_solicitacoes no banco de dados, para posterior análise por parte do administrador.

4.5. Os comandos SMS do Administrador

O administrador do sistema é o único que possui outros comandos além do comando de abertura da tranca. São três comandos que podem ser realizados via SMS: "posicao", "abrir" e "fechar".

Todos os comandos deverão ser escritos em letras minúsculas e exatamente da forma como foram apresentados, ignorando-se as aspas, para poderem funcionar corretamente.

4.5.1. O comando "posicao"

Este comando é de uso exclusivo do administrador do sistema. Para utilizá-lo, o administrador deve enviar um torpedo para o número de operação do sistema com a seguinte mensagem: "posicao". A Figura 4.6 apresenta este comando sendo utilizado.

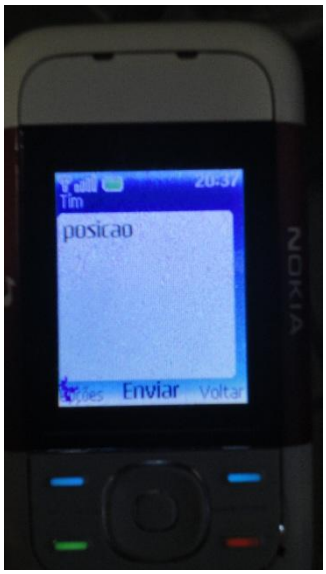


Figura 4.6 – Envio do comando “posicao”

O sistema então verificará a posição em que se encontra a tranca e retornará essa informação ao administrador. Estando a tranca aberta, será enviada uma mensagem para o administrador dizendo “Tranca aberta”, ou, caso esteja fechada, “Tranca fechada”. A Figura 4.7 apresenta as respostas ao comando “posicao”.

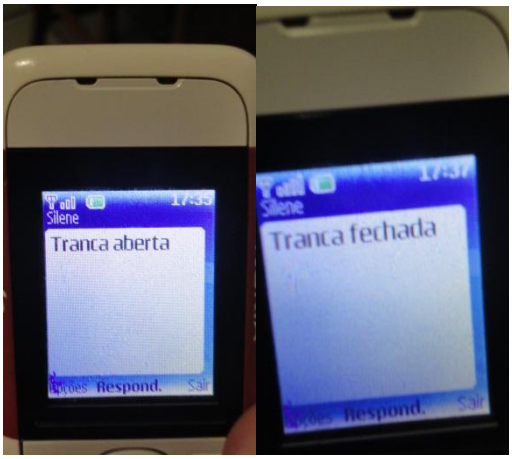


Figura 4.7 – Resposta ao comando “posicao”

Um usuário comum ou outro não cadastrado que enviarem esse comando para o sistema terão seus comandos ignorados. Nenhuma mensagem será enviada a eles.

4.5.2. O comando “abrir”

Este comando, como o próprio nome diz, serve para o administrador abrir a tranca. Apesar de ser parecido com o comando comum dos usuários, este comando,

exclusivo do administrador, se diferencia pelo fato de abrir a tranca por tempo indeterminado.

Semelhantemente ao comando “posicao”, para realizar este comando basta que o administrador envie uma mensagem SMS para o número de operação do sistema com a palavra “abrir”. Será gerada uma resposta SMS: “Tranca Aberta”. Na Figura 4.8 é representado o comando.

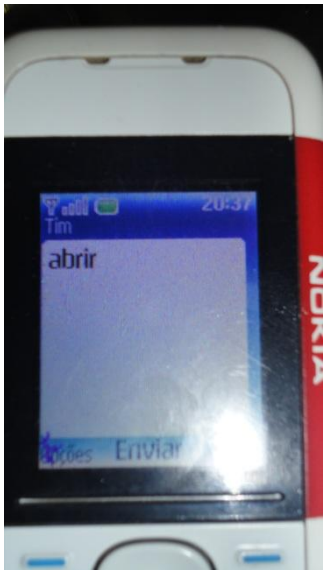


Figura 4.8 – Comando “abrir”

Este comando é gravado na tabela tb_log_adm no banco de dados. Outros usuários, que não seja o administrador, que enviar esse comando ao sistema terão suas solicitações ignoradas. A Figura 4.9 apresenta essa tabela com os registros de solicitação.

id_tb_log_adm	data_log	operacao_realizada
1	2010-11-15 12:55:39	fechar
2	2010-11-15 12:55:55	posicao
3	2010-11-15 13:35:12	abrir
4	2010-11-17 18:31:38	abrir
5	2010-11-17 18:36:30	posicao
6	2010-11-17 18:37:08	abrir
7	2010-11-17 18:37:35	fechar
NULL	NULL	NULL

Figura 4.9 – Tabela tb_log_adm

4.5.3. O comando “fechar”

É o comando que trabalha juntamente com o comando “abrir”. Ele é o responsável por fechar a tranca quando essa for aberta pelo comando abrir. É de uso exclusivo do administrador. Funciona da mesma forma que os comandos anteriores: basta que o administrador envie uma mensagem SMS para o número de operação do sistema dizendo: “fechar”. A resposta para esse comando é por SMS: “Tranca fechada”. A Figura 4.10 apresenta o comando.

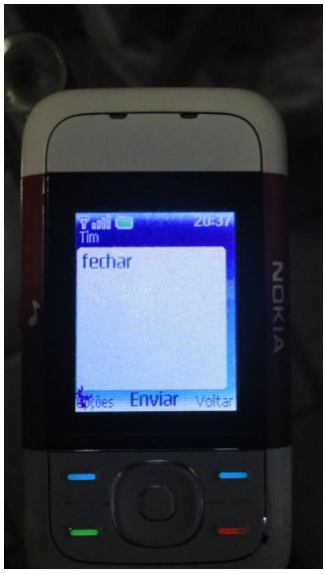


Figura 4.10 – Comando “fechar”.

Assim como os outros comandos por SMS, o comando “fechar” também ignora solicitações feitas por usuários que não seja o administrador. Esses comandos são registrados na tabela tb_log_adm no banco de dados.

4.5.4. Chamada de tranca aberta

O sistema possui uma chave óptica que facilita a identificação da situação da porta. Para a segurança não basta a tranca estar apenas na posição fechada, é preciso que ela esteja realmente trancando a porta.

Se o sistema, por meio dessa chave, identifica que a porta está aberta em um momento em que ela deveria estar fechada, o sistema realiza ligações de 30 em 30 segundos para o administrador. Essa é a única situação em que o sistema realizará uma ligação e ela é exclusiva para o administrador. Com essa chamada, o administrador tem

a possibilidade de agir mais rapidamente caso a porta fique aberta mesmo com a tranca fechada.

4.6. SMS Ignorados

Quaisquer mensagens SMS recebidas pelo número de operação do sistema diferentes dos três comandos anteriormente apresentados (“posicao”, “abrir” e “fechar”) serão ignoradas e excluídas da memória do cartão SIM.

Serão desconsiderados também comandos escritos de forma errada, ou que contenham mais informação do que a solicitada.

Mensagens encaminhadas por qualquer número que não seja o do administrador, mesmo que seja o exato comando, também serão ignoradas.

CAPÍTULO 5 – TESTES DO SISTEMA

Neste capítulo são mostrados os testes realizados para verificar o funcionamento do sistema e descrito o protocolo de falhas desenvolvido para esses testes.

5.1. Protocolo de Testes

O sistema foi testado para impedir que usuários não cadastrados, bem como usuários cadastrados mas que façam ligações em horários não permitidos, abram a tranca. Foi feito o teste para verificar se o administrador será sempre informado das solicitações e do estado da tranca.

Foram feitos testes para verificar o tempo de resposta do sistema e verificar se esse tempo estava dentro do tempo de um minuto máximo estipulado. Para tanto, os seguintes testes foram realizados:

- Verificar se o sistema está cadastrando corretamente os usuários;
- Verificar as análises de permissões;
- Verificar o comando abrir do administrador;
- Verificar o comando fechar do administrador;
- Verificar se o servo motor identifica a posição da tranca;
- Verificar o comando “posicao” do administrador;
- Contar o tempo de resposta médio.

5.2. Implementação dos Testes

Foram utilizados nos testes um servo motor para simular uma tranca sendo aberta e fechada, um módulo GSM e um computador com sistema operacional Windows XP.

5.2.1. Kit de comunicação GSM/GPRS

Primeiramente, foi testado o kit de comunicação GSM/GPRS. Foi analisado o seu correto funcionamento, bem como realizados os testes dos comandos AT que foram utilizados no programa desenvolvido.

Os testes foram realizados no programa HyperTerminal do Windows XP. Eles consistiram em enviar os comandos AT ao kit e verificar se os comandos eram reconhecidos, interpretados e executados. Foram testados os comandos AT utilizados por esse trabalho. Nenhum comando testado apresentou erro.

Após realizar os testes do kit, foi desenvolvido o trecho do programa que atua juntamente com o kit. Foi feita uma pesquisa a fim de descobrir como enviar os comandos pela porta serial do computador através da linguagem C. A Figura 5.1 apresenta o trecho do programa responsável por abrir a comunicação com a porta serial.

```
int AbrePorta() {
   DCB dcb;
   BOOL fSuccess;
   char *pcCommPort = "COM1";
   int baudRate = 115200;
   hCom = CreateFile( pcCommPort,
                      GENERIC_READ | GENERIC_WRITE,
                      0,    // must be opened with exclusive-access
                      NULL, // no security attributes
                      OPEN_EXISTING, // must use OPEN_EXISTING
                      0,    // not overlapped I/O
                      NULL);  // hTemplate must be NULL for comm devices

   if (hCom == INVALID_HANDLE_VALUE)
   {
       // Handle the error.
       printf ("Nao foi possivel conectar ao Modem GSM\nPorta: %s\nVelocidade: %dbps",pcCommPort,baudRate);
       return(1);
   }
   //Timeouts

    COMMTIMEOUTS  timeOuts;

    timeOuts.ReadIntervalTimeout          = 0xFFFFFFFF;
    timeOuts.ReadTotalTimeoutConstant     = 0;
    timeOuts.ReadTotalTimeoutMultiplier   = 0;
    timeOuts.WriteTotalTimeoutConstant    = 5000;
    timeOuts.WriteTotalTimeoutMultiplier  = 0;

    SetCommTimeouts(hCom, &timeOuts);
```

Figura 5.1 – Abrindo comunicação com a porta serial

O trecho do programa abre a comunicação com a porta serial COM1 do computador a uma velocidade de 11520 kbps. Caso o programa não encontre nenhum erro e realize com sucesso a comunicação com a porta serial, ele retorna o valor 0. O restante da função pode ser encontrado no Apêndice A deste trabalho.

5.2.2. *Drive* do motor

Após testar o kit de comunicação GSM/GPRS foi testado o funcionamento do servo motor. Ele se liga ao computador através da porta paralela. Os primeiros testes

realizados não obtiveram muito sucesso, pois aparentemente a corrente que estava sendo enviada pela porta paralela não conseguia modificar corretamente a posição do motor.

Para o controle do motor foi desenvolvido o *drive* do motor. Esse *drive* se preocupa em verificar e modificar a posição do servo motor. Foi necessário inserir a biblioteca inpout32.dll para a comunicação com a porta paralela ser realizada.

A Figura 5.2 apresenta o trecho do *drive* do motor responsável por alterar a posição do motor e depois retorná-la à posição original.

```
    Sleep(5);
    outportb(0x378,0x00);
    Sleep(15);
}*/
    while(1) {
                 Sleep(2000);
      var=240;
      int i,j;
      for (i=0; i<5;i++) {
                outportb(0x378,0x01);
          for (j=0;j<var;j++) {
              outportb(0x80,0x00);
          }
          outportb(0x378,0x0);
          Sleep(60);
    }
    Sleep(15000);
         var=470;
      for (i=0; i<5;i++) {
                outportb(0x378,0x01);
          for (j=0;j<var;j++) {
              outportb(0x80,0x00);
          }
          outportb(0x378,0x0);
          Sleep(60);
    }
}

    FreeLibrary(hLib); //Libera memória alocada pela DLL.
    return(0);
}
```

Figura 5.2 – Trecho do *drive* do motor

Pode-se observar nesse trecho de programa que o *drive* envia os sinais pelos endereços 0x378 e 0x80. A intensidade do sinal enviado, em volts, é o que determina a posição do motor. Essa intensidade é determinada pela variável var. Os valores da variável var foram determinados através de testes realizados. Foi preciso ser

ajustada manualmente a posição do motor e ir alterando o valor da variável var até se obter as duas posições desejadas. O restante do código referente ao *drive* do motor é encontrado no Apêndice B deste trabalho.

A função “Sleep” determina, em milissegundos, o tempo em que o programa fica em repouso. Na Figura 6.2 observa-se que antes de o *drive* do motor retornar para a posição inicial ele faz uma pausa de 15s. Este é o tempo que a tranca permanece aberta após uma ligação de um usuário comum.

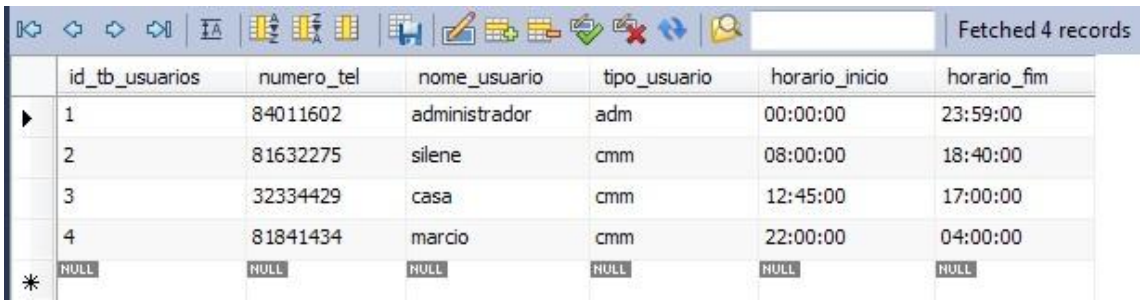
5.2.3. Cadastramento dos usuários

O primeiro teste realizado, após o sistema estar desenvolvido, foi no sentido de verificar se este estava cadastrando corretamente os usuários. Para essa verificação o banco de dados foi aberto e foram verificadas suas tabelas para conferir se os dados estariam corretos.

Foram cadastrados os seguintes usuários:

- usuario_1, com permissão das 08:00 às 12:00 e com o telefone 81632275;
- usuario_2, com permissão das 18:00 às 23:00 e com o telefone 81841434.

A Figura 6.1 apresenta a imagem do banco verificando esses cadastros.



id_tb_usuarios	numero_tel	nome_usuario	tipo_usuario	horario_inicio	horario_fim
1	84011602	administrador	adm	00:00:00	23:59:00
2	81632275	silene	cmm	08:00:00	18:40:00
3	32334429	casa	cmm	12:45:00	17:00:00
4	81841434	marcio	cmm	22:00:00	04:00:00
NULL	NULL	NULL	NULL	NULL	NULL

Figura 6.1 – Tabela de usuários

5.2.4. Permissões

O segundo passo foi testar se as solicitações estavam sendo corretamente atendidas conforme o cadastro. Foi verificado novamente no banco de dados se o

sistema estava registrando as operações corretamente. A Figura 6.2 apresenta a tabela de solicitações dos usuários.

id_tb_solicitacoes	tb_usuarios_id_tb_usuarios	data_solicitacao	resposta_solicitacao
1	1	2010-11-15 12:28:02	nao
2	1	2010-11-15 12:39:33	sim
3	1	2010-11-15 12:41:18	sim
4	3	2010-11-15 12:43:53	sim
5	1	2010-11-15 13:34:29	sim
6	3	2010-11-15 13:36:28	sim
7	1	2010-11-17 15:44:24	sim
8	3	2010-11-17 15:50:03	sim
9	3	2010-11-17 17:27:59	nao
10	3	2010-11-17 17:42:50	nao
11	3	2010-11-17 18:25:30	nao
12	1	2010-11-17 18:32:14	sim
13	3	2010-11-17 18:37:48	nao
14	2	2010-11-17 18:38:38	nao
15	2	2010-11-17 18:39:23	nao
16	2	2010-11-17 18:41:16	sim
17	4	2010-11-17 18:46:04	nao
18	4	2010-11-17 23:45:57	sim
19	4	2010-11-18 00:01:05	sim
20	1	2010-11-18 16:12:02	sim

Figura 6.2 – Tabela de solicitações dos usuários

5.2.5. Comando “abrir” do administrador

O teste consistiu em duas partes: a primeira parte foi testar se o administrador consegue realizar corretamente este comando. A segunda parte foi testar se usuários que não fossem o administrador poderiam executar esse comando. Os resultados deste teste são apresentados no Quadro 5.1.

Quadro 5.1 – Resultado do teste para o comando “abrir”

Número do Telefone	Tipo de usuário	Posição da tranca após o comando
84011602	Administrador	Aberta
81632275	Comum	Fechada
81841434	Não Cadastrado	Fechada

Após esse teste foi observado que o sistema ignora corretamente usuários que não são o administrador e usuários não cadastrados no sistema.

5.2.6. Comando “fechar” do administrador

Os testes realizados para este comando foram os mesmos para o comando abrir, diferenciando-se apenas pelo fato de que para esse teste a tranca foi posicionada aberta forçadamente. O Quadro 5.2 apresenta os resultados do teste.

Quadro 5.2 – Resultado do teste para o comando “fechar”

Número do telefone	Tipo de usuário	Posição da tranca após o comando
84011602	Administrador	Fechada
81632275	Comum	Aberta
81841434	Não Cadastrado	Aberta

Da mesma forma que o comando “abrir”, o comando “fechar” apresentou um correto funcionamento. Ele respondeu apenas ao usuário “administrador”.

5.2.7. Posicionamento da tranca

É importante para o sistema conseguir identificar o posicionamento da tranca. Os testes realizados nesse sentido verificaram se os sensores ópticos estavam funcionando corretamente e ainda se o servo motor estava fechando e abrindo totalmente a tranca.

Para a correta verificação, primeiro foi testado se o motor estava se posicionando corretamente nas posições: aberta e fechada. Inicialmente observou-se que o motor não assumia a posição correta para nenhuma das duas posições. Foi necessário ajustar através de testes do tipo, tentativa e erro, os valores da variável var no *drive* do motor. Essa é a variável responsável por determinar a posição de giro do motor.

Uma vez que foram terminados os ajustes, os testes passaram a ser com a chave óptica. A verificação para isso foi de certa forma simples de se fazer. A função "inport()" verifica o pino da porta paralela em que a chave óptica envia sinais para o computador. Quando é recebido um sinal elétrico por esse pino, o valor retornado por essa função se modifica. Ou seja, Quando a chave óptica identifica algo entre o seu emissor de luz e seu receptor, ela gera um pequeno sinal. Os testes, portanto, serviram para ajustar os valores que serviram como identificação da posição do motor.

5.2.8. O Comando "posicao" do administrador

Após verificar se o sistema conseguiria identificar a posição da tranca, o teste para esse comando se tornou simples. Bastou apenas fazer a mesma verificação que foi feita nos comandos abrir e fechar. Foi verificado se apenas o administrador teria acesso a esse comando. No Quadro 5.3 são apresentados os resultados do teste.

Quadro 5.3 – Resultado do teste para o comando "posicao".

Número do telefone	Tipo de usuário	Recebeu resposta
84011602	Administrador	Sim
81632275	Comum	Não
84488402	Não Cadastrado	Não

Da mesma forma que os outros comandos do administrador, o comando "posicao" demonstrou que estava com a lógica correta.

5.2.9. Tempo médio de resposta

Para poder se obter um tempo de resposta médio, foram calculados os tempos de resposta para diferentes operadoras de telefonia móvel. Foram realizadas 10 medições para cada operadora. O Quadro 5.4 apresenta os tempos de resposta do sistema calculados para cada operadora.

Quadro 5.4 – Tempo de resposta do sistema.

Operadora	Tempo de resposta
Oi	8s
Tim	7s
NEXTEL	Sem resposta

Obtendo-se os tempos médios por operadoras, o tempo médio do sistema foi calculado como sendo a média desses tempos. Ficou atribuído o tempo médio de resposta em 8 segundos para o sistema.

Neste teste foi identificado que celulares pertencentes à operadora NEXTEL não conseguem se comunicar com o módulo GSM SIM340. Igualmente, o módulo também não consegue se comunicar com os celulares com cartões SIM da NEXTEL. Foram realizadas pesquisas e descoberto que a operadora NEXTEL trabalha com dois tipos de padrão de comunicação móvel: rádio e TDMA. Como o projeto propõe-se a trabalhar com a tecnologia GSM, não é do escopo deste tratar da questão da comunicação com essa operadora.

De toda forma, o tempo de resposta do sistema foi considerado satisfatório, pois além de nenhuma outra operadora testada apresentar esse problema, todas as solicitações abriram a tranca 10 segundos após a primeira chamada da solicitação e enviaram a mensagem SMS para o administrador em até 1 minuto. Contudo, é importante ressaltar que este tempo de resposta não leva em conta exclusivamente o desempenho do sistema, mas também o desempenho das redes das operadoras de telefonia móvel, o que em determinadas situações de congestionamento da rede, pode causar atrasos.

5.3. Protocolo de Testes para Identificar Falhas do Sistema

A fim de se testar possíveis falhas no sistema, foram pensadas diversas situações diferentes. Buscando-se imaginar o maior número de falhas possíveis, foram realizados os seguintes testes:

- Usuário não cadastrado fazendo uma ligação para abertura da tranca;
- Usuário cadastrado, mas fazendo a ligação fora do horário permitido;

- Tolerância de atraso ou adiantamento (5 minutos antes do tempo cadastrado);
- Tentativa de um usuário comum ou não cadastrado realizar os comandos do administrador;
- Solicitações negadas repetidas;
- Dois usuários tentando realizar uma ligação ao mesmo tempo;
- Forçar uma pequena abertura na porta.

5.3.1. Usuário não cadastrado realizando uma ligação para abrir a tranca

Este teste de falha é o mais simples de ser realizado. Basta usar qualquer número de telefone que não esteja cadastrado no sistema e realizar uma ligação para o número de operação do sistema.

Foram feitos testes simples, usando um telefone não cadastrado e logo após cadastrando-o. E foi verificado como o sistema se portou. O Quadro 5.5 demonstra o resultado deste teste.

Quadro 5.5 – Teste de cadastro.

Número do solicitante	Cadastrado no sistema	Resposta da solicitação
84011602	NÃO	NÃO
84011602	SIM	SIM
81632275	NÃO	NÃO
81632275	SIM	SIM
81841434	NÃO	NÃO
81841434	SIM	SIM

Foi observado um adequado comportamento do sistema, tanto no que se diz respeito a negar solicitações de usuários não cadastrados, como em autorizar usuários com permissão.

Da mesma forma, foi observado que todas as solicitações foram armazenadas de forma correta no banco de dados.

5.3.2. Usuário cadastrado e fora do seu horário permitido

Bastante parecido com o teste anterior, esse teste visa observar se o sistema irá conceder a permissão para alguém fora do seu horário. Aproveitando-se dos resultados obtidos no teste anterior e visando maior agilidade no processo dos testes, apenas foram alterados os horários cadastrados dos registros anteriormente testados.

O Quadro 5.6 apresenta os resultados obtidos com os testes.

Quadro 5.6 – Teste de horário.

Usuário	Horário permitido	Horário testado	Resposta
84011602	00:00 às 23:59	22:40	SIM
81632275	12:00 às 16:00	13:35	SIM
81632275	12:00 às 16:00	22:45	NÃO
81841434	22:00 às 02:00	13:40	NÃO
81841434	22:00 às 02:00	22:50	SIM
81841434	22:00 às 02:00	00:05	SIM

Esse teste acabou trazendo à tona um problema na lógica do sistema, relacionado a usuários que possuíam um cadastro em que há mudança de dia. Anteriormente a esse teste, o sistema não se preocupava em analisar se o horário de início da permissão era realmente menor que o horário de término da permissão.

O problema foi solucionado analisando-se o intervalo de permissão do usuário e, em caso de o horário de início ser maior que o horário de término, o sistema inverte a verificação e verifica se o usuário está fora do seu horário permitido.

5.3.3. Tolerância do sistema

O sistema foi projetado para aceitar uma tolerância de 5 minutos, tanto para adiantamentos como para atrasos. Os testes consistiram em testar tanto a parte da

tolerância anterior como a parte posterior. Os registros foram alterados de forma a se testar dentro desses 5 minutos.

Os testes foram todos satisfatórios, tanto para usuários adiantados, até 5 minutos antes do seu horário de inicio de permissão, como para os usuários atrasados, até 5 minutos depois do seu horário de termino de permissão.

A lógica usada para tratar dessa questão foi simplesmente subtrair cinco minutos do horário de inicio de permissão e somar cinco minutos ao horário de término de permissão.

5.3.4. Reconhecimento de administrador

Esses testes foram realizados para verificar se o sistema reconhece quem é o seu administrador. Portanto, foram usados números de usuários comuns realizando os comandos permitidos apenas para o administrador.

Dentro do programa existe uma função chamada "armazenar_numero_adm()". Essa função é chamada logo no início do programa e é a responsável por consultar o banco de dados e armazenar o número do administrador.

Conforme dito anteriormente neste trabalho, esse sistema foi desenvolvido para apenas um único administrador. Caso seja necessário mudar o administrador do sistema, este deve ser desligado, a mudança do cadastro deverá ser realizada no banco de dados e, por fim, o sistema deverá ser iniciado novamente.

5.3.5. Solicitações negadas repetidas

Visando reduzir o custo operacional do sistema, ao se enviar várias mensagens SMS para usuários que não são cadastrados, esses testes verificaram se o sistema ignora um número não cadastrado no sistema ao tentar realizar mais de uma solicitação negada em um espaço inferior a uma hora. Desta forma, o sistema também passa a filtrar a quantidade de informações geradas para o administrador.

Os testes realizados foram satisfatórios: o sistema gerou apenas uma mensagem por hora para cada número não cadastrado. O sistema considera sempre o horário da ultima ligação. Portanto, se o solicitante ligar de 15 em 15 minutos, por exemplo, o sistema só gera mensagens na primeira tentativa. Para este mesmo

solicitante receber uma nova mensagem é necessário que ele fique por uma hora ou mais sem ligar.

No caso de um solicitante ser cadastrado no sistema após ter obtido uma solicitação negada por não estar cadastrado, ele não precisa esperar uma hora para fazer a nova ligação. Para tanto, basta apenas que o seu cadastro seja efetuado pelo administrador.

5.3.6. Chamadas simultâneas

Foi o teste mais complexo de ser realizado. Primeiramente, porque é impossível reproduzir dois eventos distintos em um exato momento do tempo. Pode-se realizar no máximo aproximações, porém elas nunca serão iguais. Além disso, o tempo de envio de ligações entre um aparelho e outro, bem como a rede das operadoras de telefonia móvel, atrapalham mais ainda essa aproximação.

No entanto foram pensadas algumas situações que pudessem ocorrer com o sistema, utilizando-se aproximação de tempo:

- Dois usuários com permissão para entrar no momento;
- Um usuário com e o outro sem permissão para entrar no momento;
- Dois usuários sem permissão para entrar no momento;
- Um usuário no limite de sua tolerância e outro sem permissão;
- Um usuário no limite de sua tolerância e outro com permissão;
- Dois usuários no limite de sua tolerância.

Em todos os casos testados, o sistema se portou da mesma forma: a chamada que ele recebeu primeiro foi tratada e a outra foi ignorada. Não foi apresentado nenhum tipo de congestionamento de informação. Contudo, vale ressaltar que esse sistema foi desenvolvido para um número pequeno de usuários, 10, não sendo previsto como ele poderia se comportar com uma grande quantidade de usuários.

5.3.7. Manter porta aberta forçadamente

O teste consistiu em colocar um objeto obstruindo o fechamento da porta. Esse teste teve como finalidade verificar se o sistema identifica uma pequena abertura da porta. Para descobrir isso, foi adicionada uma chave óptica na tranca da porta e esta informa se a tranca está fechada ou não. Caso o sensor óptico identifique que a tranca esteja aberta enquanto deveria estar fechada, o sistema passa a informar o administrador com uma ligação de 30 em 30 segundos.

Foram feitos testes para testar se o sistema se comportaria da forma esperada. Os testes consistiram em deixar a tranca fora da chave óptica, o que indica que a porta está aberta, e verificar se o administrador recebeu as ligações de 30 em 30 segundos. Os resultados destes testes são demonstrados no Quadro 5.7.

Quadro 5.7 – Teste de porta aberta.

Porta	Tempo	Ligação efetuada
Aberta	30s	Sim
Aberta	60s	Sim
Aberta	90s	Sim
Aberta	120s	Sim
Fechada	150s	Não
Fechada	180s	Não

5.4. Resultados Obtidos

Os resultados obtidos foram considerados satisfatórios. O sistema se portou, de uma forma geral, conforme o esperado. Não foram encontradas grandes limitações, com exceção do fato de o sistema não se comunicar com a operadora NEXTEL.

O tempo de resposta obtido foi melhor do que o esperado, pois foi estimado que o administrador recebesse em até 1 minuto as mensagens e o sistema respondeu sempre em até 45 segundos. Era também esperado que o sistema levasse até 10 segundos após a primeira chamada para abrir a porta, e ele levou em média 8 segundos, conforme visto. O sistema manteve-se funcionando corretamente, de forma

ininterrupta, por mais de 80 solicitações, superando com folga a especificação mínima de 30 ligações sem erro.

A integração entre os componentes e o programa desenvolvido não apresentou problemas sérios. Alguns problemas encontrados fogem do escopo deste trabalho, como, por exemplo, o caso da comunicação com a operadora NEXTEL ou ainda problemas por conexão em determinadas redes. Por isso eles não foram explorados. A Seção 5.6 apresenta os problemas encontrados ao longo do desenvolvimento deste projeto.

5.5. Problemas Encontrados

5.5.1. Projeto antigo

A ideia inicial de todo o trabalho era criar um sistema que identificasse problemas na alimentação de energia de uma determinada rede. Esse sistema, assim como o atual, iria informar a um usuário quando essa rede ficasse sem energia por um determinado tempo via mensagem SMS.

Foi adquirido um módulo Benq M22 para esse antigo projeto por ter um menor custo. Contudo, esse módulo veio com problemas de comunicação e não reconhecia nenhum comando que lhe era enviado. Ele foi adquirido através de terceiros e já era usado. Também não possuía assistência técnica para verificar os defeitos. Desta forma, não foi possível aproveitar o módulo adquirido para o antigo projeto.

Realizando-se uma nova pesquisa, foi encontrado o Kit de Comunicação GSM/GPRS da empresa Cerne Tecnologia. Ele veio com o módulo SIM340. Diferentemente do módulo adquirido anteriormente, o kit de comunicação possuía assistência técnica. Foi verificado então que este kit exigia uma alimentação de 12 V com corrente mínima de 1 A, o que acabou resultando na modificação deste trabalho, uma vez que não seria viável manter um aparelho que exige uma corrente tão alta apenas com baterias.

5.5.2. Problemas com o kit de comunicação

Após obter o novo módulo GSM e modificar o trabalho, foram realizados os primeiros testes com o novo aparelho. À primeira vista, o kit não apresentava nenhum problema e até reconhecia os comandos AT que lhe eram enviados. Contudo, ele não conseguia se comunicar com o cartão SIM que estava inserido nele e não realizava e nem recebia ligações ou mensagens SMS.

Entrando em contato com a empresa, que tem sede no estado do Rio de Janeiro, ela solicitou o envio do aparelho para verificá-lo.

5.5.3. Atraso devido à empresa

O kit logo foi enviado à empresa para que ela pudesse realizar os testes no aparelho. Constatado o defeito, foi informado que um novo kit seria enviado em poucos dias. Quase um mês após a constatação do defeito do kit, o aparelho novo ainda não havia chegado. Mesmo pressionada, a empresa não informava com precisão sobre a situação do novo kit. Só após um mês e quatro dias foi que o novo kit chegou, o que gerou um grande atraso na implementação do projeto.

5.5.4. Comunicação com a operadora

Nos primeiros testes, foi verificado que o kit de comunicação GSM/GPRS não conseguia se comunicar com cartões SIM pertencentes à operadora NEXTEL. Quando se ligava de um número pertencente a essa operadora para o sistema, a linha telefônica sempre dava sinal de ocupada. Coisa semelhante ocorria mesmo ao ligar do sistema para o celular.

Era possível enviar mensagens SMS de um para o outro, porém com grande atraso no recebimento. Nenhuma outra operadora testada apresentou este problema. Foram realizadas pesquisas e descoberto que a operadora NEXTEL trabalha com dois tipos de tecnologia: via rádio e TDMA. No entanto, não está no escopo deste projeto tratar de problemas relacionados a outras tecnologias que não a GSM.

5.5.5. Comunicação com a porta serial

O primeiro problema encontrado durante a construção do programa foi realizar a comunicação com a porta serial do computador. Especificamente, não se conseguia ler o que estava chegando nessa porta. O caminho contrário de envio de informações era realizado sem problemas.

Foram realizadas várias pesquisas e vários testes até se chegar à função “SerialGetc” da biblioteca “Windows.h”. Esta função lê um caractere por vez que está na entrada da porta serial do computador.

5.5.6. Atraso no recebimento de mensagens

Algumas vezes o sistema demorou a reconhecer o recebimento de uma mensagem. Em alguns casos isolados, ele não reconheceu a chamada. Existem algumas causas prováveis para isso ocorrer.

Pode ocorrer erro na transmissão de algum caractere pela porta serial do computador. Basta que apenas um caractere esteja errado que o sistema não identifica o comando. Tentando contornar essa possibilidade foi feita uma rotina de espera maior no recebimento de dados. Porém, mesmo assim, é possível que aconteçam erros na transmissão.

Outro fator a ser considerado é a falta de memória do cartão SIM. Esta possibilidade foi contornada fazendo-se uma limpeza na memória toda vez que o sistema é iniciado.

Por fim, existe a possibilidade de a operadora estar com muito tráfego de informações e atrasar o envio da mensagem. Conforme dito, não está no escopo deste projeto tratar de problemas referentes às operadoras de telefonia móvel.

5.5.7. Horários de acesso

O último problema encontrado foi com relação ao horário de acesso dos usuários. Aparentemente não haveria problemas com relação ao horário de acesso dos usuários, pois existia o banco de dados contendo as informações desses usuários. Contudo, desenvolvendo-se a lógica apareceu a possibilidade de um usuário possuir um cadastro que ocupasse horas de dois dias. Por exemplo, um usuário possuir um cadastro que lhe permitisse acesso das 22h às 04h. Pela lógica implementada no código, isso se tornou um problema porque o programa primeiramente verificava se o horário da ligação era maior que o início da sua permissão e depois verificava se era menor que o término.

Foi então preciso fazer uma verificação anterior: se o início da permissão era menor do que o término da permissão. Se maior, a solução encontrada foi inverter os horários de início e de fim. A lógica é verificar se o usuário está dentro do período das 02 às 22h. Se estiver, sua permissão é negada.

CAPÍTULO 6 – CONCLUSÃO

Neste capítulo é apresentada a conclusão deste trabalho. É feita uma comparação entre os resultados esperados e os resultados obtidos e também uma comparação entre o sistema aqui apresentado e um modelo semelhante recentemente proposto. Por fim, são apresentadas as propostas para melhoria do projeto futuramente.

6.1. Resultados

Conforme visto no Capítulo 5, o sistema se portou da forma que era esperada, ou seja, atendeu aos requisitos mínimos exigidos para o seu correto funcionamento: abrir a tranca quando solicitado por um usuário com cadastro e em horário permitido; impedir que os usuários que não se encontravam nessa situação abrissem a porta; armazenar todas as informações pertinentes quanto às solicitações e horários de abertura da tranca; enviar ao administrador do sistema as informações sobre as solicitações de abertura da tranca em até um minuto; informar ao usuário solicitante se sua solicitação foi atendida ou não; permanecer funcionando sem que ocorram erros por no mínimo 30 solicitações.

O sistema não conseguiu se comunicar com celulares da operadora NEXTEL. Essa operadora não consegue identificar o módulo no modo de dados e sempre a ligação fica como ocupada. No caminho inverso da mesma forma quando se é realizada uma ligação a linha fica como ocupada, porém quando se envia uma mensagem SMS, esta chega com um certo atraso ao aparelho da referida operadora. As outras operadoras apresentaram os resultados que eram esperados e não se teve nenhum problema de comunicação com o módulo GSM.

De uma forma geral, o tempo de resposta do sistema ficou menor do que os 10 segundos estipulados na especificação do projeto: na média, demorou 8 segundos entre a ligação ser encerrada e a mensagem ser recebida. Apenas em uma situação foi que o envio da mensagem informando a solicitação ao administrador demorou mais do que um minuto, podendo ser considerado um caso isolado, uma vez que em todas as outras solicitações a mensagem SMS chegou ao administrador em menos de 45 segundos.

Conclui-se que o sistema se portou de forma eficiente e rápida superando as expectativas iniciais. Analisando as informações armazenadas no banco de dados, foi

concluído que é possível se obter um controle maior sobre os acessos ocorridos no local em que o sistema está empregado. Um sistema como o mostrado neste trabalho poderia ter evitado alguns casos de acesso não autorizado a locais importantes, como no caso do ENEM de 2009, mencionado na introdução deste trabalho.

6.2. Proposta Similar

No dia 03/11/2010 foi vista uma matéria sobre um projeto similar ao apresentado neste trabalho. A ideia central do projeto é abrir uma porta usando um aparelho celular.

Este projeto está sendo testado em um hotel na Suécia. Sua principal diferença com relação ao projeto apresentado nesta monografia é o tipo de acesso que ele possibilita. Não é preciso realizar ligações e nem enviar mensagens SMS para poder abrir a porta; basta que o usuário aproxime o seu aparelho da porta que ela se abre. Contudo, ele só funciona com *Smartphones* de última geração que possuem um mecanismo que emite ondas em uma determinada frequência de rádio (G1, 2010).

Isso mostra uma desvantagem deste projeto do hotel em relação ao projeto apresentado neste trabalho. A maioria dos *Smartphones* não possui tal mecanismo, o que restringe bastante o uso dessa tecnologia desenvolvida, ao passo que o uso de chamadas via rede GSM não limita muito o tipo de usuário para o sistema desse trabalho.

6.3. Propostas de Projetos Futuros

Muitas melhorias podem ser realizadas neste projeto, principalmente no que se refere ao aumento de segurança do sistema.

Por motivos financeiros e por se tratar de um protótipo, não foi adaptada nenhuma mola para fechar porta, item que seria importante para um maior controle da segurança do local.

Pode-se integrar ao sistema um dispositivo para fazer medições biométricas, tais como digital, reconhecimento por íris ou, ainda, reconhecimento por voz. Sugere-se que, para um reconhecimento por voz, após realizar a ligação, o sistema,

ao invés de recusar a ligação, a atenda e exija que o usuário fale uma frase. Desta forma, o sistema teria maior confiabilidade.

Há a possibilidade de se integrar um sistema de câmeras de vídeo no qual, com o uso de programas desenvolvidos, fossem analisadas quantas pessoas desejam abrir a tranca. O sistema, por exemplo, poderia negar acesso caso detecte a presença de mais de uma pessoa no local.

Aconselha-se também a implementação de um gerador próprio de energia para esse sistema, de modo que ele não fique dependente do sistema de rede elétrica.

Não foi desenvolvida nenhuma interface para o banco de dados, ficando este com o simples objetivo de armazenar as informações. Sugere-se que se construa uma plataforma para internet, ou intranet, que permita outras pessoas terem acesso às informações geradas pelo sistema.

O banco de dados pode ser melhorado de forma a permitir uma maior quantidade de detalhes a serem armazenados. Isto não foi implementado neste projeto pelo fato de não ser necessário armazenar muitas informações, mas, dependendo da situação em que o sistema seja implementado, pode ser interessante armazenar uma quantidade maior de informação. Pode ser feita uma normalização melhor no banco, criando tabelas exclusivas para telefones, por exemplo, para o usuário caso se deseje permitir mais de um número por usuário. Essa normalização também permite o cadastro de intervalos de horários de permissão.

Além da normalização do banco, pode-se criar uma interface para se trabalhar com o mesmo. Pode-se igualmente, criar uma aplicação para a internet que permita acesso as informações contidas no banco.

O sistema prevê apenas um administrador para reduzir os custos de envio de mensagens SMS e restringir o acesso de outras pessoas a ele. Porém, pode-se ampliar a quantidade de administradores, tendo-se o cuidado de realizar as devidas modificações em todo o sistema.

6.4. Aplicação em Outros Campos

Uma das grandes vantagens deste projeto é a grande flexibilidade que ele possui, sendo facilmente empregado em outras aplicações.

Como este projeto se preocupa com a abertura de uma tranca de uma porta, e não com a porta propriamente dita, fica viável usá-lo de formas diferentes. Ele pode ser utilizado em cofres, em armazéns, como forma de acesso a uma Central de Processamento de Dados (CPD), ou ainda ser usado como um dos critérios para acesso a um lugar.

Com algumas modificações, de forma a reduzir a quantidade de informação gerada para o administrador, e integrado a outros sistemas, ele pode servir como parte de um sistema de ponto de funcionários de empresas.

REFERÊNCIAS BIBLIOGRÁFICAS

BRAGA, Newton. *Acopladores e Chaves ópticas*. Em: <http://www.newtoncbraga.com.br/index.php/como-funciona/872-acopladores-e-chaves-opticas-art120.html>. Acesso em: 10/11/2010.

CERNE. *Apostila de GSM.* Cerne Tecnologia. 2010

FRANCISCO, Antônio. *Motores Eléctricos*. Portugal. Editora ETEP - Edições Técnicas e Profissionais. 2008.

Hotel na Suécia substituirá chaves por telefone celular. **G1.** Disponível em: <http://g1.globo.com/tecnologia/noticia/2010/11/hotel-na-suecia-substituira-chaves-por-telefone-celular.html>. Acesso em 04/11/2010.

Indiciado por roubo de provas do Enem diz que gráfica não tinha segurança para evitar vazamento. **O GLOBO.** Disponível em: <http://oglobo.globo.com/educacao/vestibular/mat/2009/10/10/indiciado-por-roubo-de-provas-do-enem-diz-que-grafica-nao-tinha-seguranca-para-evitar-vazamento-768005894.asp>. Acesso em 29/09/2010.

ISAAC, Cristiano Rabelo. Trabalho de conclusão do Curso de Engenharia de Computação do Centro Universitário de Brasília. *Comunicação Entre Um Veículo e o Usuário Através de um Sistema GSM*. Brasília, 2010. Orientador Prof. Ms. José Julima.

MELO, Luciana Ferreira. Trabalho de conclusão do Curso de Engenharia de Computação do Centro Universitário de Brasília. *RFID Em Sistemas de Segurança em Prédios*. Brasília, 2010. Orientador Prof. Ms. José Julima.

MILANI, Andre. MySQL Guia do programador .
Novatec editora ltda - São Paulo 2008.
Páginas: 397.

MYSQL, *Why MySQL?* (Por que o MySQL?). Disponível em: <http://www.mysql.com/why-mysql/>. Acesso em 18/10/2010

NICOLOSI, Denys E. C. *Microcontrolador 8051 Detalhado*. São Paulo: Ed. Érica, 2004.

ROGERCOM, Porta Paralela. Disponível em:
<http://www.rogercom.com/pparalela/introducao.html>. Acesso em 20/10/2010.

SERVODATABASE, *Hextronik HXT900 - 9g Micro Servo*. Disponível em:
< http://www.servodatabase.com/servo/hextronik/hxt900>. Acesso em 05/11/2010.

SIMCOM. *SIM340 Hardware Interface Description.* 2006

WIKIPEDIA, Interface serial. Disponível em:
<http://pt.wikipedia.org/wiki/Porta_serial>. Acesso em 20/10/2010.

APÊNDICE A – CÓDIGO FONTE DO PROGRAMA

Este apêndice apresenta o código fonte do programa desenvolvido para esse trabalho.

```
#include <stdio.h>
#include <conio.h>
#include <windows.h>
#include <time.h>
#include <string.h>
#include <mysql/mysql.h>
#define HOST "localhost"
#define USER "root"
#define PASS "shionmam"
#define DB "sistema_tranca"
#include "E:\projeto_novo\drive_motor.c"

MYSQL conexao;
MYSQL_RES *resp;
MYSQL_ROW resultado;
MYSQL_FIELD *campos;

typedef short _stdcall (*PtrInp)(short EndPorta);
typedef void _stdcall (*PtrOut)(short EndPorta, short valor);

void limpar_rxstring();
void separa_hora();
void registrar_logadm();
void comandos_adm();
void verifica_ligacao();
char id_chamada[3], mensagem[10], reconhecido[10];
int sol_expr, ult_expr, pri_expr;
int sol_ano, sol_mes, sol_dia, sol_hora, sol_min, sol_seg;
int ult_ano, ult_mes, ult_dia, ult_hora, ult_min, ult_seg;
char data_temp[5];
int AbrePorta();
char SerialString();
void FechaPorta(void);
HANDLE hCom;
char comando_at[150];
char rxstring[256];
void SerialPutc(HANDLE hCom, char txchar);
char SerialGetc(HANDLE hCom);
char c, query[1000], horario_solicitacao[20];
```

```
HANDLE teste;
int pos = 0;
void chamada_usuario();
void enviar_comando(char *comando_at);
void limpar_comando(char *comando_at);
void limpar_numero(char *numero_chamando);
char numero_chamando[13], numero_adm[13];
int k;
void ligacao_comum();
int status=0;
void armazenar_numero_adm();
void limpa_query();
char id_usuario[3], hora_ini[10], hora_fim[10], nome_usuario[50];
void espera_caractere();
void limpa_memoria();

int main()
{
//Programacao porta paralela - carrega inpout32.dll

  HINSTANCE hLib; //Instância para a DLL inpout32.dll.
  PtrInp inportb;     //Instância para a função Imp32().
  PtrOut outportb;  //Instância para a função Out32().

  //Carrega a DLL na memória.
  hLib = LoadLibrary("inpout32.dll");

  if(hLib == NULL) //Verifica se houve erro.
  {
    printf("Erro. O arquivo inpout32.dll não foi encontrado.\n");
    getch();
    return -1;
  }

  //Obtém o endereço da função Inp32 contida na DLL.
  inportb = (PtrInp) GetProcAddress(hLib, "Inp32");

  if(inportb == NULL) //Verifica se houve erro.
  {
    printf("Erro. A função Inp32 não foi encontrada.\n");
    getch();
    return -1;
  }

  //Obtém o endereço da função Out32 contida na DLL.
```

```
   outportb = (PtrOut) GetProcAddress(hLib, "Out32");

   if(outportb == NULL) //Verifica se houve erro.
   {
      printf("Erro. A função Out32 não foi encontrada.\n");
      getch();
      return -1;
   }

// Fim porta paralela
   int teste;
         teste = inportb(0x379);

    if(teste != 95) {
          CtrlMotor(470);
    }
    fflush(stdin);
    if(AbrePorta() == 0)
    {
       printf("Modulo SIM340 Configurado\n" );
    }
    else
    {
       printf("Erro ao Configurar Modulo");
    }
    sprintf(comando_at,"AT"); // inicializa a conversa com o modulo
    enviar_comando(comando_at);
    limpar_comando(comando_at);
    sprintf(comando_at,"ATE0"); // desliga o eco
    enviar_comando(comando_at);
    limpar_comando(comando_at);
    strcpy(comando_at, "AT+CMGF=1"); // configura a mensagem para o padrão texto
    enviar_comando(comando_at);
    limpar_comando(comando_at);
    fflush(stdin);
    armazenar_numero_adm();
    limpa_memoria();
    printf("Aguardando Comandos\n");
    while(1)
    {
       limpar_comando(comando_at);
       limpar_rxstring();
       espera_caractere();
       if(strncmp(rxstring,"RING",4)==0)
       {
```

```
            chamada_usuario();
        }
        else if(strncmp(rxstring,"+CMTI",5)==0)
        {
            comandos_adm();
        }
    limpar_rxstring();
    }
    Sleep(500);
    FechaPorta();
    FreeLibrary(hLib);

    return 0;

}

int AbrePorta()
{
    DCB dcb;
    BOOL fSuccess;
    char *pcCommPort = "COM5";
    int baudRate = 115200;
    hCom = CreateFile( pcCommPort,
                    GENERIC_READ | GENERIC_WRITE,
                    0,    // a porta deve ser aberta com acesso exclusivo
                    NULL, // sem atributos de segurança
                    OPEN_EXISTING,
                    0,
                    NULL);
    if (hCom == INVALID_HANDLE_VALUE)
    {
        // Gravar o erro.
        printf ("Nao foi possivel conectar ao Modulo GSM\nPorta: %s\nVelocidade:
%dbps",pcCommPort,baudRate);
        return(1);
    }
    //Timeouts
    COMMTIMEOUTS  timeOuts;
    timeOuts.ReadIntervalTimeout         = 0xFFFFFFFF;
    timeOuts.ReadTotalTimeoutConstant    = 0;
    timeOuts.ReadTotalTimeoutMultiplier  = 0;
    timeOuts.WriteTotalTimeoutConstant   = 5000;
    timeOuts.WriteTotalTimeoutMultiplier  = 0;
    SetCommTimeouts(hCom, &timeOuts);
    fSuccess = GetCommState(hCom, &dcb);
```

```
   if (!fSuccess)
   {
      // Gravar erro.
      printf ("GetCommState falhou com o seguinte erro: %d.\n", GetLastError());
      return(2);
   }
   // configurar porta como: 115200bps, 8 data bits, sem paridade e 1 stop bit.
   dcb.DCBlength = sizeof(DCB);
   GetCommState(hCom, &dcb);
   dcb.BaudRate  = baudRate;
   dcb.ByteSize  = 8;
   dcb.fRtsControl = RTS_CONTROL_DISABLE;
   fSuccess = SetCommState(hCom, &dcb);
   if (!fSuccess)
   {
      printf ("GetCommState falhou com o seguinte erro: %d.\n", GetLastError());
      return(3);
   }
   printf ("Modem GSM configurado na porta %s com sucesso.\n", pcCommPort);
   return (0);
}

void FechaPorta(void)
{
   CloseHandle(hCom);
}

void SerialPutc(HANDLE hCom, char txchar)
{
   BOOL bWriteRC;
   static DWORD iBytesWritten;
   bWriteRC = WriteFile(hCom, &txchar, 1, &iBytesWritten,NULL);
   return;
}

char SerialGetc(HANDLE hCom)
{
   char rxchar;
   BOOL    bReadRC;
         static  DWORD   iBytesRead;
   bReadRC = ReadFile(hCom, &rxchar, 1, &iBytesRead, NULL);
   return rxchar;
}

void chamada_usuario()
```

```
{
   int j, i=0;
   Sleep(500);
   limpar_numero(numero_chamando);
   strcpy(comando_at, "AT+CLCC"); // Verifica qual o número está chamando
   enviar_comando(comando_at);
   Sleep(500);
   c = SerialGetc(hCom);
   espera_caractere();
   if(strncmp(rxstring,"+CLCC",5)==0)
   {
      for(j=16;j<24;j++)
      {
         numero_chamando[i]=rxstring[j];
         i++;
      }
      printf("%s\n",numero_chamando);
      limpar_comando(comando_at);
      strcpy(comando_at, "ATH"); //encerra a ligação recebida
      enviar_comando(comando_at);
      limpar_comando(comando_at);
      ligacao_comum(); //conecta com o banco de dados
      switch (status)
      {
         case 1:
         sprintf(comando_at,"AT+CMGS=\"%s\"",numero_chamando); // comando para
enviar mensagem SMS
         enviar_comando(comando_at);
         limpar_comando(comando_at);
         sprintf(comando_at, "%s, a tranca foi aberta e fechara em 15 segundos\x1A",
nome_usuario); // Envia a mensagem
         enviar_comando(comando_at);
         limpar_comando(comando_at);
         sprintf(comando_at,"AT+CMGS=\"%s\"",numero_adm);; // comando para
enviar mensagem SMS
         enviar_comando(comando_at);
         limpar_comando(comando_at);
         sprintf(comando_at, "O usuario %s, solicitou abertura da tranca as %s e esta foi
aberta\x1A", nome_usuario, horario_solicitacao); // Envia a mensagem
         enviar_comando(comando_at);
         limpar_comando(comando_at);
         CtrlMotor(240);//Abre a tranca
         Sleep(15000);
         CtrlMotor(470);//Fecha a tranca
         break;
```

```
        case 2:
        sprintf(comando_at,"AT+CMGS=\"%s\"",numero_chamando); // comando para
enviar mensagem SMS
        enviar_comando(comando_at);
        limpar_comando(comando_at);
        sprintf(comando_at, "%s, voce realizou uma solicitacao de abertura fora do seu
horario permitido\x1A", nome_usuario); // Envia a mensagem
        enviar_comando(comando_at);
        limpar_comando(comando_at);
        sprintf(comando_at,"AT+CMGS=\"%s\"",numero_adm);; // comando para
enviar mensagem SMS
        enviar_comando(comando_at);
        limpar_comando(comando_at);
        sprintf(comando_at, "O usuario %s, solicitou abertura da tranca as %s e esta foi
negada por ser fora do horario permitido\x1A", nome_usuario, horario_solicitacao); //
Envia a mensagem
        enviar_comando(comando_at);
        limpar_comando(comando_at);
        break;
        case 3:
        sprintf(comando_at,"AT+CMGS=\"%s\"",numero_chamando); // comando para
enviar mensagem SMS
        enviar_comando(comando_at);
        limpar_comando(comando_at);
        sprintf(comando_at, "Voce nao possui permissao para a abertura da tranca. Entre
em contato com o administrador no numero %s, caso deseje se cadastrar no
sistema\x1A", numero_adm); // Envia a mensagem
        enviar_comando(comando_at);
        limpar_comando(comando_at);
        sprintf(comando_at,"AT+CMGS=\"%s\"",numero_adm);; // comando para
enviar mensagem SMS
        enviar_comando(comando_at);
        limpar_comando(comando_at);
        sprintf(comando_at, "O numero %s, que nao possui cadastro, solicitou abertura
da tranca\x1A", numero_chamando); // Envia a mensagem
        enviar_comando(comando_at);
        limpar_comando(comando_at);
        break;
        case 4:
        break;
        case 5:
        sprintf(comando_at,"AT+CMGS=\"%s\"",numero_chamando); // comando para
enviar mensagem SMS
        enviar_comando(comando_at);
        limpar_comando(comando_at);
```

```
            sprintf(comando_at, "%s, a tranca foi aberta e fechara em 15 segundos\x1A",
nome_usuario); // Envia a mensagem
            enviar_comando(comando_at);
            limpar_comando(comando_at);
            CtrlMotor(240);
            Sleep(15000);
            CtrlMotor(470);
            break;
         }
      }
      Sleep(500);
      limpar_rxstring();
      return;
}

//envia os comandos AT para o kit GSM
void enviar_comando(char *comando_at)
{
   int j;
   for(j=0;j<=strlen(comando_at);j++)
   {
      SerialPutc(hCom,comando_at[j]);
   }
   SerialPutc(hCom,'\r');
   printf("%s\n", comando_at);
   Sleep(10);
   //espera_caractere();
   return;
}

//limpa a string onde sao armazenados os comandos AT
void limpar_comando(char *comando_at)
{
   int j;
   for(j=0;j<=strlen(comando_at);j++)
   {
      comando_at[j]='\0';
   }
   return;
}

//limpa a string que contem o numero solicitante
void limpar_numero(char *numero_chamando)
{
   int j;
```

```
   for(j=0;j<=strlen(numero_chamando);j++)
   {
      numero_chamando[j]='\0';
   }
   return;
}

//classifica que tipo de ligacao foi recebida
void ligacao_comum()
{
   int i;
   char temp_hora[10];
   mysql_init(&conexao);
   if(mysql_real_connect(&conexao,HOST,USER,PASS,DB,0,NULL,0))
   {
      sprintf(query,"SELECT id_tb_usuarios FROM tb_usuarios where numero_tel =
\"%s\";", numero_chamando); //verifica se o numero solicitante possui cadastro no
sistema
      mysql_query(&conexao,query);
      resp=mysql_store_result(&conexao);
      limpa_query();
      if ((resultado=mysql_fetch_row(resp)) != NULL)
      {
      // Armazena o horario de inicio da permissao do usuario
         strcpy(id_usuario, resultado[0]);
         sprintf(query,"select horario_inicio from tb_usuarios where id_tb_usuarios =
'%s';", id_usuario);
         mysql_query(&conexao,query);
         limpa_query();
         resp=mysql_store_result(&conexao);
         resultado=mysql_fetch_row(resp);
         strcpy(hora_ini, resultado[0]);
         // Armazena o horario de termino da permissao do usuario
         sprintf(query,"select horario_fim from tb_usuarios where id_tb_usuarios =
'%s';", id_usuario);
         mysql_query(&conexao,query);
         limpa_query();
         resp=mysql_store_result(&conexao);
         resultado=mysql_fetch_row(resp);
         strcpy(hora_fim, resultado[0]);
         // Armazena o nome do usuario
         sprintf(query,"select nome_usuario from tb_usuarios where id_tb_usuarios =
'%s';", id_usuario);
         mysql_query(&conexao,query);
         limpa_query();
```

```
        resp=mysql_store_result(&conexao);
        resultado=mysql_fetch_row(resp);
        strcpy(nome_usuario, resultado[0]);
        // Registra a solicitacao do usuario
        sprintf(query,"insert into tb_solicitacoes (tb_usuarios_id_tb_usuarios) values
('%s');", id_usuario);
        mysql_query(&conexao,query);
        limpa_query();
        // Verifica o horario da solicitacao
        sprintf(query,"select data_solicitacao from tb_solicitacoes where
resposta_solicitacao is NULL;");
        mysql_query(&conexao,query);
        resp=mysql_store_result(&conexao);
        resultado=mysql_fetch_row(resp);
        strcpy(horario_solicitacao, resultado[0]);
        for (i=0;i<20;i++)
        {
           horario_solicitacao[i]=horario_solicitacao[i+11];
        }
        //Compara os horarios e verifica se o usuario possui a permissao
        if(strcmp(numero_chamando,numero_adm)==0)
        {
           status=5;
           sprintf(query,"update tb_solicitacoes set resposta_solicitacao = 'sim' where
resposta_solicitacao is NULL;");  //atualiza a resposta do sistema a solicitacao para sim
           mysql_query(&conexao,query);
           resp=mysql_store_result(&conexao);
        }
        else
        {
           separa_hora();
           if(pri_expr<ult_expr)
           {
              if(ult_expr>sol_expr)
              {
                 if(pri_expr<sol_expr)
                 {
                    status=1;
                    sprintf(query,"update tb_solicitacoes set resposta_solicitacao = 'sim'
where resposta_solicitacao is NULL;"); //atualiza a resposta do sistema a solicitacao
para sim
                    mysql_query(&conexao,query);
                    resp=mysql_store_result(&conexao);
                 }
                 else
```

```
                {
                    status=2;
                    sprintf(query,"update tb_solicitacoes set resposta_solicitacao = 'nao'
where resposta_solicitacao is NULL;"); //atualiza a resposta do sistema a solicitacao
para nao
                    mysql_query(&conexao,query);
                    resp=mysql_store_result(&conexao);
                }
            }
            else
            {
                status=2;
                sprintf(query,"update tb_solicitacoes set resposta_solicitacao = 'nao'
where resposta_solicitacao is NULL;"); //atualiza a resposta do sistema a solicitacao
para sim
                mysql_query(&conexao,query);
                resp=mysql_store_result(&conexao);
            }
        }
        else
        {
            if(ult_expr<sol_expr)
            {
                if(pri_expr>sol_expr)
                {
                    status=2;
                    sprintf(query,"update tb_solicitacoes set resposta_solicitacao = 'nao'
where resposta_solicitacao is NULL;");
                    mysql_query(&conexao,query);
                    resp=mysql_store_result(&conexao);
                }
                else
                {
                    status=1;
                    sprintf(query,"update tb_solicitacoes set resposta_solicitacao = 'sim'
where resposta_solicitacao is NULL;");
                    mysql_query(&conexao,query);
                    resp=mysql_store_result(&conexao);
                }
            }
            else
            {
                status=1;
                sprintf(query,"update tb_solicitacoes set resposta_solicitacao = 'sim'
where resposta_solicitacao is NULL;");
```

```
                mysql_query(&conexao,query);
                resp=mysql_store_result(&conexao);
            }
        }
    }
    else
    {
        sprintf(query,"insert into tb_solicitacoes_invalidas (numero_solicitante) values
('%s');", numero_chamando); //registra a solicitacao na tabela de solicitacoes sem
cadastro
        mysql_query(&conexao,query);
        limpa_query();
        sprintf(query,"select data_solicitacao from tb_solicitacoes_invalidas order by
id_tb_solicitacoes_invalidas desc limit 1 ;"); // verifica o horario da solicitacao
        mysql_query(&conexao,query);
        resp=mysql_store_result(&conexao);
        resultado=mysql_fetch_row(resp);
        strcpy(horario_solicitacao, resultado[0]);
        //Armazena nas variaveis temporarias os valores do horario da solicitacao
        strncpy(data_temp, horario_solicitacao,4);
        data_temp[4]='\0';
        sol_ano = atoi(data_temp); //armazena o ano da solicitacao
        for (i=0;i<20;i++)
        {
            horario_solicitacao[i]=horario_solicitacao[i+5];
        }
        strncpy(data_temp,horario_solicitacao,2);
        data_temp[2]='\0';
        sol_mes = atoi(data_temp);//armazena o mes da solicitacao
        for (i=0;i<20;i++)
        {
            horario_solicitacao[i]=horario_solicitacao[i+3];
        }
        strncpy(data_temp,horario_solicitacao,2);
        data_temp[2]='\0';
        sol_dia = atoi(data_temp);//armazena o dia da solicitacao
        for (i=0;i<20;i++)
        {
            horario_solicitacao[i]=horario_solicitacao[i+3];
        }
        strncpy(data_temp,horario_solicitacao,2);
        data_temp[2]='\0';
        sol_hora = atoi(data_temp);//armazena a hora da solicitacao
        for (i=0;i<20;i++)
```

```
        {
           horario_solicitacao[i]=horario_solicitacao[i+3];
        }
        strncpy(data_temp,horario_solicitacao,2);
        data_temp[2]='\0';
        sol_min = atoi(data_temp);//armazena os minutos da solicitacao
        limpa_query();
        //verifica se o numero já fez uma chamada anterior
        sprintf(query,"select id_tb_solicitacoes_invalidas from tb_solicitacoes_invalidas
where numero_solicitante = '%s' order by id_tb_solicitacoes_invalidas desc limit 1 ;",
numero_chamando);
        mysql_query(&conexao,query);
        resp=mysql_store_result(&conexao);
        resultado=mysql_fetch_row(resp);
        strcpy(id_chamada, resultado[0]);
        //Verifica ultima ligacao do numero
        limpa_query();
        sprintf(query,"select data_solicitacao from tb_solicitacoes_invalidas where
numero_solicitante = '%s' and id_tb_solicitacoes_invalidas < '%s' order by
id_tb_solicitacoes_invalidas desc limit 1 ;", numero_chamando, id_chamada);
        mysql_query(&conexao,query);
        resp=mysql_store_result(&conexao);
        if ((resultado=mysql_fetch_row(resp)) != NULL)
        {
           strcpy(horario_solicitacao, resultado[0]);
           //Armazena nas variaveis temporarias os valores do horario da ultima ligacao
do numero
           strncpy(data_temp, horario_solicitacao,4);
           data_temp[4]='\0';
           ult_ano = atoi(data_temp); //armazena o ano da solicitacao
           for (i=0;i<20;i++)
           {
              horario_solicitacao[i]=horario_solicitacao[i+5];
           }
           strncpy(data_temp,horario_solicitacao,2);
           data_temp[2]='\0';
           ult_mes = atoi(data_temp);//armazena o mes da solicitacao
           for (i=0;i<20;i++)
           {
              horario_solicitacao[i]=horario_solicitacao[i+3];
           }
           strncpy(data_temp,horario_solicitacao,2);
           data_temp[2]='\0';
           ult_dia = atoi(data_temp);//armazena o dia da solicitacao
           for (i=0;i<20;i++)
```

```
            {
               horario_solicitacao[i]=horario_solicitacao[i+3];
            }
            strncpy(data_temp,horario_solicitacao,2);
            data_temp[2]='\0';
            ult_hora = atoi(data_temp);//armazena a hora da solicitacao
            for (i=0;i<20;i++)
            {
               horario_solicitacao[i]=horario_solicitacao[i+3];
            }
            strncpy(data_temp,horario_solicitacao,2);
            data_temp[2]='\0';
            ult_min = atoi(data_temp);//armazena os minutos da solicitacao
            verifica_ligacao();
         }
         else
         {
            status=3;
         }
         mysql_free_result(resp);
      }
   }
   else
   {
      printf("\nFalha na conexão com o banco de dados");
      if(mysql_errno(&conexao))
      {
         printf("\nErro %d. %s",mysql_errno(&conexao), mysql_error(&conexao));
      }
   }

   mysql_close(&conexao);
   return;
}

//Armazena o numero do telefone do administrador
void armazenar_numero_adm()
{
   mysql_init(&conexao);
   if(mysql_real_connect(&conexao,HOST,USER,PASS,DB,0,NULL,0))
   {
      sprintf(query,"SELECT numero_tel FROM tb_usuarios where tipo_usuario =
'adm';");
      mysql_query(&conexao,query);
      resp=mysql_store_result(&conexao);
```

```
        resultado=mysql_fetch_row(resp);
        strcpy(numero_adm,resultado[0]);
        limpa_query();
        numero_adm[8]='\0';
    }
    else
    {
        printf("\nFalha na conexão com o banco de dados");
        if(mysql_errno(&conexao))
        {
            printf("\nErro %d. %s",mysql_errno(&conexao), mysql_error(&conexao));
        }
    }
    mysql_close(&conexao);
    printf("%s\n",numero_adm);
    return;
}

//Limpa os comandos SQL enviados ao banco de dados
void limpa_query()
{
    int j;
    for(j=0;j<=strlen(query);j++)
    {
        query[j]='\0';
    }
    return;
}

//Funcao que faz verificaçao das ligacoes recebidas
void verifica_ligacao()
{
    sol_expr = (sol_hora*60+sol_min);
    ult_expr = (ult_hora*60+ult_min);
    if(sol_ano - ult_ano <=1)
    {
        if(sol_mes - ult_mes <=1)
        {
            if(sol_dia - ult_dia <=1)
            {
                if(sol_expr - ult_expr<=60)
                {
                    status = 4;
                }
                else
```

```
            {
               status = 3;
            }
         }
         else
         {
            status=3;
         }
      }
      else
      {
         status = 3;
      }
   }
   else
   {
      status = 3;
   }
   return;
}

//Identificacao dos comandos do administrador
void comandos_adm()
{
   int j,k=0;
   char id_msg[3];;
   limpar_numero(numero_chamando);
   fflush(stdin);
   for(j=9;j<=strlen(rxstring);j++)
   {
      id_msg[k]=rxstring[j];
      k++;
   }
   fflush(stdin);
   limpar_comando(comando_at);
   sprintf(comando_at,"AT+CMGR=%s",id_msg); // Le a mensagem
   enviar_comando(comando_at);
   limpar_comando(comando_at);
   limpar_rxstring();
   espera_caractere();
   strcpy(mensagem,rxstring);
   printf("%s\n",mensagem);
   Sleep(1500);
   mensagem[8]='\0';
   if(strcmp(numero_chamando,numero_adm)==0)
```

```
    {
        if(strncmp(mensagem,"abrir",5)==0)
        {
            printf("FAZER CHAMADA NO DRIVE DO MOTOR PARA ABRIR\n");
            CtrlMotor(240);//Abre a tranca
            registrar_logadm();
        }
        else
        {
            if(strncmp(mensagem,"fechar",6)==0)
            {
                printf("FAZER CHAMADA NO DRIVE DO MOTOR PARA fechar\n");
                CtrlMotor(470);//Fecha a tranca
                registrar_logadm();
            }
            else
            {
                if(strncmp(mensagem,"posicao",7)==0)
                {
                    printf("FAZER CHAMADA NO DRIVE DO MOTOR PARA verificar
posicao\n");
                    registrar_logadm();
                }
            }
        }
    }
    sprintf(comando_at,"AT+CMGD=%s",id_msg); // apaga a mensagem
    enviar_comando(comando_at);
    limpar_comando(comando_at);

    return;
}

//rotina de espera por um caractere valido na porta serial
void espera_caractere()
{
    limpar_rxstring();
    int i=0, k=0, j=0, msg=0;
    char d;
    fflush(stdin);
    while(i<2)
    {
        c = SerialGetc(hCom);
        Sleep(5);
        if(c>'\"')
```

```
        {
           rxstring[k]=c;
           k++;
        }
        else if(c=='\n')
        {
           if(strncmp(rxstring,"+CMGR",5)==0)
           {
               for(j=0;j<=strlen(rxstring);j++)
               {
                   rxstring[j]=rxstring[j+19];
               }
               strncpy(numero_chamando,rxstring,8);
               limpar_rxstring();
               k=0;
               while(i<4)
               {
                  c = SerialGetc(hCom);
                  Sleep(5);
                  if(c>='a')
                  {
                     rxstring[k]=c;
                     k++;
                  }
                  else if(c=='\n')
                  {
                     +i++;
                  }
               }
           }
           else
           {
               i++;
           }
        }
    }
    printf("%s\n",rxstring);
    return ;
}

//registra os comandos do administrador SMS
void registrar_logadm()
{
    mysql_init(&conexao);
    if(mysql_real_connect(&conexao,HOST,USER,PASS,DB,0,NULL,0))
```

```
    {
        sprintf(query,"insert into tb_log_adm (operacao_realizada) values ('%s');",
mensagem);
        mysql_query(&conexao,query);
        resp=mysql_store_result(&conexao);
        limpa_query();
    }
    else
    {
        printf("\nFalha na conexão com o banco de dados");
        if(mysql_errno(&conexao))
        {
            printf("\nErro %d. %s",mysql_errno(&conexao), mysql_error(&conexao));
        }
    }
    mysql_close(&conexao);
    return;
}

//separa as horas de permissao do usuario para calcular o acesso
void separa_hora()
{
    char temp_hora[9];
    int i;
    strcpy(temp_hora,hora_ini);
    strncpy(data_temp,temp_hora,2);
    ult_hora=atoi(data_temp);
    for(i=0;i<=strlen(temp_hora);i++)
    {
        temp_hora[i]=temp_hora[i+3];
    }
    strncpy(data_temp,temp_hora,2);
    ult_min=atoi(data_temp);
    for(i=0;i<=strlen(temp_hora);i++)
    {
        temp_hora[i]=temp_hora[i+3];
    }
    strncpy(data_temp,temp_hora,2);
    ult_seg=atoi(data_temp);
    pri_expr=ult_hora*60*60+ult_min*60+ult_seg-300;
    if(sol_expr<0)
    {
        pri_expr=86400+sol_expr;
    }
    printf("%i\n",pri_expr);
```

```
   strcpy(temp_hora,hora_fim);
   strncpy(data_temp,temp_hora,2);
   ult_hora=atoi(data_temp);
   for(i=0;i<=strlen(temp_hora);i++)
   {
      temp_hora[i]=temp_hora[i+3];
   }
   strncpy(data_temp,temp_hora,2);
   ult_min=atoi(data_temp);
   for(i=0;i<=strlen(temp_hora);i++)
   {
      temp_hora[i]=temp_hora[i+3];
   }
   strncpy(data_temp,temp_hora,2);
   ult_seg=atoi(data_temp);
   ult_expr=ult_hora*60*60+ult_min*60+ult_seg+300;
   if(ult_expr>86400)
   {
      ult_expr=ult_expr-86400;
   }
   printf("%i\n",ult_expr);
   strcpy(temp_hora,horario_solicitacao);
   strncpy(data_temp,temp_hora,2);
   ult_hora=atoi(data_temp);
   for(i=0;i<=strlen(temp_hora);i++)
   {
      temp_hora[i]=temp_hora[i+3];
   }
   strncpy(data_temp,temp_hora,2);
   ult_min=atoi(data_temp);
   for(i=0;i<=strlen(temp_hora);i++)
   {
      temp_hora[i]=temp_hora[i+3];
   }
   strncpy(data_temp,temp_hora,2);
   ult_seg=atoi(data_temp);
   sol_expr=ult_hora*60*60+ult_min*60+ult_seg;
   printf("%i\n",sol_expr);
   return;
}

//Apaga mensagens na memoria do cartao sim
void limpa_memoria()
{
   int i;
```

```
    for(i=50;i>0;i--)
    {
        sprintf(comando_at,"AT+CMGD=%i",i);
        enviar_comando(comando_at);
        limpar_comando(comando_at);
        Sleep(150);
    }
    return;
}

void limpar_rxstring()
{
    int i;
    for(i=0;i<257;i++)
    {
        rxstring[i]='\0';
    }
    rxstring[0]='\0';
    return;
}
```

APÊNDICE B – DRIVE DO MOTOR

O apêndice B apresenta o drive do motor que foi desenvolvido para o controle do posicionamento do motor.

```
typedef short _stdcall (*PtrInp)(short EndPorta);
typedef void _stdcall (*PtrOut)(short EndPorta, short valor);



void CtrlMotor(int var);

void CtrlMotor(int var) {
  HINSTANCE hLib; //Instância para a DLL inpout32.dll.
  PtrInp inportb;     //Instância para a função Imp32().
  PtrOut outportb;  //Instância para a função Out32().

  //Carrega a DLL na memória.
  hLib = LoadLibrary("inpout32.dll");

  if(hLib == NULL) //Verifica se houve erro.
  {
    printf("Erro. O arquivo inpout32.dll não foi encontrado.\n");
    getch();
    exit(1);
  }

  //Obtém o endereço da função Inp32 contida na DLL.
  inportb = (PtrInp) GetProcAddress(hLib, "Inp32");

  if(inportb == NULL) //Verifica se houve erro.
  {
    printf("Erro. A função Inp32 não foi encontrada.\n");
    getch();
    exit(1);
  }

  //Obtém o endereço da função Out32 contida na DLL.
  outportb = (PtrOut) GetProcAddress(hLib, "Out32");

  if(outportb == NULL) //Verifica se houve erro.
  {
     printf("Erro. A função Out32 não foi encontrada.\n");
     getch();
```

```
      exit(1);
   }

    int i,j;
    for (i=0; i<5;i++) {
            outportb(0x378,0x01);
        for (j=0;j<var;j++) {
           outportb(0x80,0x00);
        }
        outportb(0x378,0x0);
        Sleep(60);
   }

   if(inportb(0x379) == 255) {
                printf("Aberto");
   }
   else if (inportb(0x379) == 95) {
       printf("Fechado");
   }
   else {
       printf("Entre aberto");
   }
}
```

APÊNDICE C – CONSTRUÇÃO DO BANCO DE DADOS

É apresentada neste apêndice a sequência de comandos SQL necessária para a construção do banco de dados.

```
CREATE TABLE tb_solicitacoes_invalidas (
  id_tb_solicitacoes_invalidas INTEGER(5) UNSIGNED NOT NULL
AUTO_INCREMENT,
  numero_solicitante INTEGER(10) UNSIGNED NOT NULL,
  data_solicitacao TIMESTAMP NOT NULL,
  PRIMARY KEY(id_tb_solicitacoes_invalidas)
)
TYPE=InnoDB;

CREATE TABLE tb_usuarios (
  id_tb_usuarios INTEGER(3) UNSIGNED NOT NULL AUTO_INCREMENT,
  numero_tel INT(10) NOT NULL,
  nome_usuario VARCHAR(50) NOT NULL,
  tipo_usuario CHAR(3) NOT NULL,
  horario_inicio TIME NULL,
  horario_fim TIME NULL,
  PRIMARY KEY(id_tb_usuarios)
)
TYPE=InnoDB;

CREATE TABLE tb_log_adm (
  id_tb_log_adm INTEGER(4) UNSIGNED NOT NULL AUTO_INCREMENT,
  data_log TIMESTAMP NOT NULL,
  operacao_realizada CHAR(8),
  PRIMARY KEY(id_tb_log_adm)
)
TYPE=InnoDB;

CREATE TABLE tb_solicitacoes (
  id_tb_solicitacoes INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  tb_usuarios_id_tb_usuarios INTEGER(3) UNSIGNED NOT NULL,
  data_solicitacao TIMESTAMP NOT NULL,
  resposta_solicitacao CHAR(3),
  PRIMARY KEY(id_tb_solicitacoes),
  INDEX tb_solicitacoes_FKIndex1(tb_usuarios_id_tb_usuarios),
  FOREIGN KEY(tb_usuarios_id_tb_usuarios)
    REFERENCES tb_usuarios(id_tb_usuarios)
```

```
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
)
TYPE=InnoDB;
```

ANEXO A - SIM340 HARDWARE INTERFACE DESCRIPTION

Este anexo apresenta as especificações do módulo SIM340. O conteúdo deste se encontra no CD entregue juntamente com a monografia