INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGICA DE SANTA CATARINA
CÂMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO SUPERIOR DE TECNOLOGIA EM ELETRÔNICA INDUSTRIAL

JOÃO VICTOR NAGAE MENEGHINI

PROPOSTA DE TAXIMETRO EMBARCADO COM RASTREAMENTO E REGISTRO DE IMAGENS

FLORIANÓPOLIS, FEVEREIRO DE 2017.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGICA DE SANTA CATARINA

CÂMPUS FLORIANÓPOLIS

DEPARTAMENTO ACADÊMICO DE ELETRÔNICA

CURSO SUPERIOR DE TECNOLOGIA EM ELETRÔNICA INDUSTRIAL

JOAO VICTOR NAGAE MENEGHINI

PROPOSTA DE TAXIMETRO EMBARCADO COM RASTREAMENTO E REGISTRO DE IMAGENS

Trabalho de Conclusão de Curso submetido ao Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina como parte dos requisitos para obtenção do título de Tecnólogo em Eletrônica Industrial.

Professor Orientador: Samir Bonho, Mestre em Engenharia Elétrica

FLORIANÓPOLIS, FEVEREIRO DE 2017.

PÁGINA PARA COLOCAÇÃO
DA FICHA DE IDENTIFICAÇÃO DA OBRA

PROPOSTA DE TAXIMETRO EMBARCADO COM RASTREAMENTO E REGISTRO DE IMAGENS

JOÃO VICTOR NAGAE MENEGHINI

Este trabalho foi julgado adequado para obtenção do Título de Tecnólogo em Eletrônica Industrial e aprovado na sua forma final pela banca examinadora do Curso Superior de Tecnologia em Eletrônica Industrial do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis, 13 de Fevereiro, 2017.

Banca Examinadora:

Prof. Samir Bonho, Me.

Prof. Reginaldo Steinbach, Me.

Prof. Pedro Giassi Junior, Dr.

AGRADECIMENTOS

Agradeço a todos que ajudaram na concretização desse projeto, aos Professores que me forneceram o conhecimento, ao Meu Orientador que me guiou durante essa jornada e Minha Família e amigos que me apoiaram o tempo todo.

RESUMO

O serviço de taxi é o transporte publico individual mais usado atualmente. Com a popularização do smartphone, foram criados diversos aplicativos para uso dessa categoria. Este documento apresentará o desenvolvimento de um protótipo de taxímetro com sistema integrado de vigilância e rastreamento. O sistema desenvolvido é composto por um aplicativo para sistema operacional Android, um firmware implementado em Raspberry PI para processamento de sensores e aferição de valores e distancias e o servidor utilizado para recepção e armazenamento de dados. São também apresentados os resultados gerados pelo projeto, dentre os quais as capturas de telas do aplicativo, logs do servidor e simulação do sinal do sensor de efeito hall através de um gerador de funções.

Palavras-chave: Taxímetro. Smartphone. Android. Bluetooth. Raspberry PI.

ABSTRACT

The Taxi service is currently the most used individual public transport. With the popularization of the smartphones, several apps were created to facilitate the use of taxis by the passenger. However, there is still a lack of integration amongst this apps, the user, the driver and the car itself. Based on this issue, this work presents the development of a taximeter prototype with image recording and tracking system. The developed system is composed of an app for Android operational system, a firmware implemented on Raspberry PI for sensor processing (odograph) and distance measurement. Also a webserver was created for data reception and storage. As results this document gives the screenshots of the application, the server logs and simulation of the functionalities using function generator signals as the odometer.

Key-words: Taximeter. Smartphone. Android. Bluetooth. Raspberry PI.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

IFSC – Instituto Federal de Santa Catarina

DAELN – Departamento Acadêmico de Eletrônica

RPI - Raspberry PI

CPU – Unidade de Processamento Central(*Central Processing Unit)*

ARM - Advanced RISC Machine

RAM – *Random Access Memory*

I2C – *Inter-Integrated Circuit*

SPI – *Serial Peripheral Interface*

USB – *Universal Serial Bus*

HDMI - *High-Definition Multimedia Interface*

MAC - *Media Access Control*

OSI - *Open Systems Interconnection*

SUMÁRIO

1 INTRODUÇÃO

O táxi pode ser citado como meio de transporte público individual mais utilizado no mundo. Segundo estatísticas de 2015 da Associação das Empresas de Taxi de Frota do Município de São Paulo, a frota brasileira de taxi é composta por volta de 130 mil taxis, sendo as maiores frotas a do estado de São Paulo e Rio de Janeiro com respectivos 33.992 veículos e 33 mil. A frota de Florianópolis é composta por 471 veículos tendo uma taxa de 997 habitantes por taxi.

Com a popularização dos smartphones, tendo a intenção de manter este serviço de transporte atualizado, aplicativos como “99Taxi” e “EasyTaxi”, são usados como uma alternativa ao usuário, em relação ao uso das centrais de rádio táxi

Estes softwares para smartphone vieram para melhorar a interação entre passageiro e taxista, uma vez que em ambos, o aplicativo disponibiliza variadas informações sobre o taxista que irá atender a chamada. O sistema também possibilita a personalização do serviço, por exemplo, tamanho de porta-malas, atendimento especializado em relação a mobilidade reduzida, etc. Diversas opções de pagamento também são propiciadas pelos aplicativos, como cartões de credito ou debito.

Entretanto, informações que possam garantir a segurança e controle das corridas ainda não estão totalmente integradas aos sistemas de cooperativas de Táxi. Propõe-se nesse trabalho um sistema embarcado tendo como função principal um taxímetro que faça comunicação com um servidor de dados com informações que possam trazer benefícios mútuos à cooperativa, taxista e passageiro.

1.1 JUSTIFICATIVA

Com a popularização dos aplicativos dedicados ao serviço de taxi, teve-se como consequência o aumento do uso de smartphone pelos taxistas, assim, observa-se uma possiblidade de agregar novas tecnologias ao conceito do taxímetro, transportando o mesmo para o smartphone.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Desenvolver um protótipo de taxímetro que envolva um sistema de localização e captura de imagens integrado ao totalizador.

1.2.2 Objetivos Específicos

Foram definidos como objetivos específicos desse trabalho:

1. Desenvolver um sistema de vigilância por imagens

2. Implementar firmware para captura de dados para aferição de distância percorrida, em linguagem Python.

3. Desenvolver um aplicativo para Integração do sistema de captura de imagens, localização e totalização de valores

4. Implementar um servidor central para coleta, processamento e armazenamento de dados

2 REVISÃO BIBLIOGRAFICA

2.1 SERVIÇO DE TAXI

O princípio do serviço de taxi pode ser datado do século 17, na Europa, sendo usado veículos de tração animal para o transporte. Através dos tempos, este serviço sofreu diversas modernizações, como por exemplo, a invenção do taxímetro, por Wilhelm Friedrich Nedler, Ferdinand Dencker e Friedrich Wilhelm Gustav Bruhn em 1890, sendo o mesmo completamente mecânico. Na mesma época houve a substituição da tração animal por veículos motorizados. Em 1940, o sistema passou por uma nova inovação com a adoção do sistema de rádio, possibilitando a comunicação entre taxista e central, e em 1980 a implementação do taxímetro digital.

Os taxímetros atuais, na sua maioria, fazem uso de microcontroladores para o processamento e aferição das corridas realizadas. Uma vez que um microcontrolador foi introduzido, criando a necessidade de se utilizar um método compatível da aquisição da distancia percorrida de uma corrida contratada e como solução foi adotado o uso de um sensor magnético de efeito Hall, nesse caso, tendo como funcionalidade especifica transformar a distancia percorrida em um sinal elétrico que possa ser interpretado.

O sensor utilizado nos taxis, em sua maioria, é montado na transmissão do veiculo, onde é localizado o cabo do velocímetro, sendo esse montado no sensor. Em veículos mais modernos, já é feita a conexão da entrada do taxímetro ao circuito do sensor de velocidade do próprio veiculo, descartando o uso do sensor em questão.

2.1.1 Taxímetro

O taxímetro foi criado em 1891 por Wilhelm Friedrich Nedler, Ferdinand Dencker e Friedrich Wilhelm Gustav Bruhn, sendo esse completamente mecânico e montado fora do veículo. Com o passar do tempo, o mesmo foi transferido para o interior do veículo e por volta de 1980 houve a implementação do taxímetro digital.

A aferição da distância era feita através de um eixo acoplado a roda do veículo, sendo esse sistema baseado na teoria do odômetro já existente nos carros, tendo um mecanismo adicional para totalização do valor da corrida. Para a contagem do tempo parado era feito através de um relógio de corda integrado ao sistema. Na figura 1 é representado um exemplo de um taxímetro mecânico utilizado pelos taxis londrinos na década de 30.

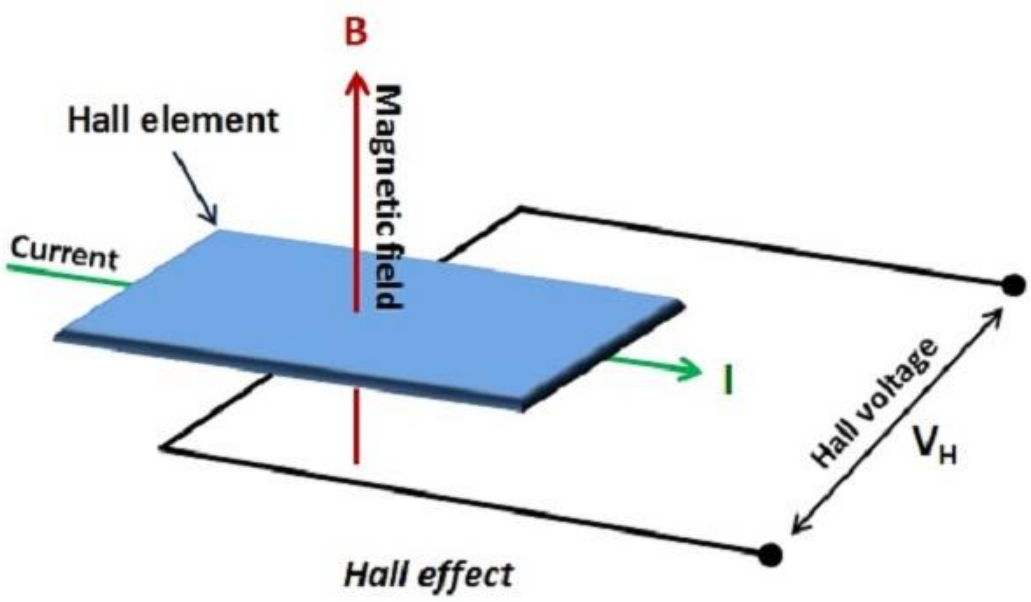
Figura 1 - Taxímetro Argo - 1930

Fonte: http://www.lvta.co.uk/history.htm

2.1.2 Sensor Efeito Hall

O efeito hall pode ser descrito como um diferencial de tensão em um condutor elétrico transversal a corrente e paralelo ao campo magnético aplicado ao mesmo. Descoberto em 1879 por Edwin Hall

Figura 2 - Efeito Hall



Fonte: http://dangerousprototypes.com

O principal uso do sensor de efeito hall possui é monitoramento de posição e movimento, sendo esse podendo ser usado um alcance de medição definido, por exemplo posição de válvulas, ou como uma chave tendo valores definidos em zero ou 1, sendo esse modo utilizado nos taxis para a medição de distância percorrida e consequentemente o valor a ser cobrado.

Uma das vantagens que esse tipo de sensor tem sobre os outros é uma expectativa de vida útil maior, uma vez que este não depende de contato físico entre o sensor e o objeto para o funcionamento do mesmo. Contribui-se para a longa vida útil o fato do sensor poder ser construído de uma forma que o torne imune ao ambiente, como poeira, umidade e entre outros.

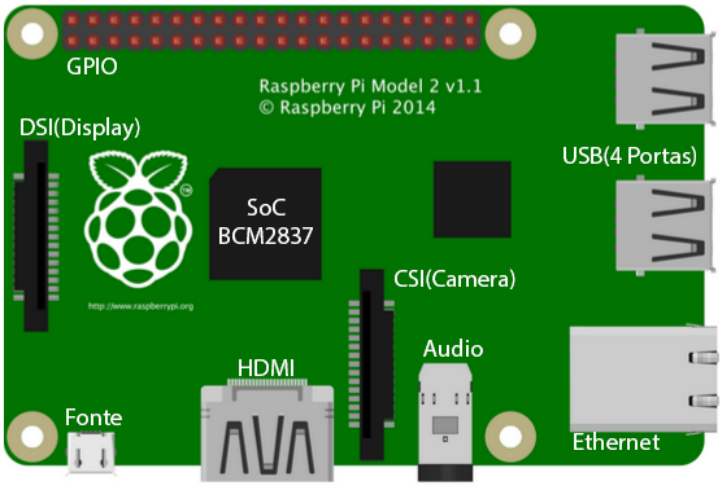
Por causa do seu princípio de funcionamento, o sensor é vulnerável a interferências magnéticas, podendo gerar leituras errôneas.

2.2 HARDWARE

2.2.1 Raspberry PI

O modelo usado neste sistema é o da segunda geração, modelo b plus, sendo esse representado pelo diagrama na figura 2. Os motivos da escolha desse hardware para a aplicação do sistema é sua modularidade, capacidade de processamento, além da imediata disponibilidade para uso

Figura 3 - Raspberry PI



Fonte: Raspberry PI (*www.*raspberrypi*.org)*

Seu núcleo de processamento é constituído de um System on Chip (SoC), que o mesmo compreende uma unidade de processamento central (CPU), unidade de processamento gráfico (GPU), memória RAM e os chipsets north e south bridge.

O SoC utilizado é o Broadccom BCM2836, que tem como sua CPU um ARM Cortex A7, tendo seu clock de processamento em 900Mhz, com 1 GB de memória RAM disponível.

Para a interface de hardware são disponibilizados 40 pinos de I/O de proposito geral, tendo nesses, acesso para interface i2c e SPI, interface para uma câmera e um touchscreen, ambos disponibilizados pela Raspberry PI Foundation.

Para conexão de dispositivos periféricos é fornecido 4 portas do tipo USB 2.0, uma HDMI para a saída de vídeo e uma porta Ethernet para interface de rede e internet.

No que se refere a armazenamento de dados e seu sistema operacional, é necessário o uso de um cartão de memória do tipo miniSD. O uso de sistema de arrefecimento não é necessário, exceto se forem feitas modificações nos parâmetros de funcionamento de seu SoC que gere um aumento em sua dissipação de potencia suficiente para que se faça necessário um sistema de resfriamento dedicado

O System on Chip BCM2836 tem como sua CPU o ARM Cortex A7, um processador de 32 bits, arquitetura armV7, memoria cache L2 variável entre 128KB a 1MB. Esse processador tem sua arquitetura compatível com os modelos A15 e A17, permitindo o uso em conjunto com o mesmo, usando a tecnologia criada pela ARM, chamada de big.LITTLE, possibilitando o processamento em multicluster heterogêneo.

O Sistema Operacional utilizado é o Raspian, que é uma versão do Debian modificado especialmente para o uso na Raspberry PI e é fornecido pela Raspberry PI Fondation.

A Alimentação do sistema é feita através do sistema elétrico presente no veículo fazendo uso de um adaptador de 12 Volts para 5 Volts.

2.2.2 Câmera

CameraPi, demonstrada na Figura 3, é o modulo oficial criado pela Raspberry PI Fondation para a fácil e rápida implementação de uma solução para captura de imagens, seja estática ou dinâmica (foto/vídeo). O sensor utilizado é o OV5647, produzido pela OmniVision, com sua resolução em 2592 pixels por 1944 pixels. No que se refere a performance, ela é capaz de fotos com resolução de 5MP sendo 8 na versão 2, vídeos em 1080 com framerate de 30 frames por segundo, 720p em 60 frames e 480p com um máximo de 90 frames por segundo.

Figura 4 - CameraPi

Fonte: SparkFun Eletronics (www.sparkfun.com)

Seus componentes são alojados numa placa de circuito impressa de formato quadrado e possui uma altura máxima de 1 centímetro. A conexão com a Ri é feita através de um flatcable, em um conector dedicado ao periférico. Por padrão tem 15 centímetros de comprimento, mas é possível a troca por 30 centímetros. O controle da mesma pode ser feito através do prompt de comando do sistema operacional Raspian, possibilitando mudanças de parâmetros em tempo real, ou via software. Isso é possível uma vez que todo o firmware necessário é presente no sistema operacional.

2.3 BLUETOOTH

Bluetooth é uma tecnologia de transmissão de dados fazendo uso de ondas de radio, com uso principal para transmissões em curta distancia. Esse sistema tem diversos usos, sendo as principais a conexão de acessórios e periféricos sem a necessidade de uso de cabos, a implementação de rede de sensores. Criado em 1999, por um consorcio formado pela Ericsson, Nokia, IMB, Intel e Toshiba, que foi chamado de Bluetooth SIG (Special Interest Group). Essa tecnologia pode ser dividida em 4 gerações.

Geração 1 foi o ponto de início, Geração 2 foi a mais difundida, sendo separada em 2.0 e 2.1, conhecida como basic data rate e enhanced data rate respectivamente.

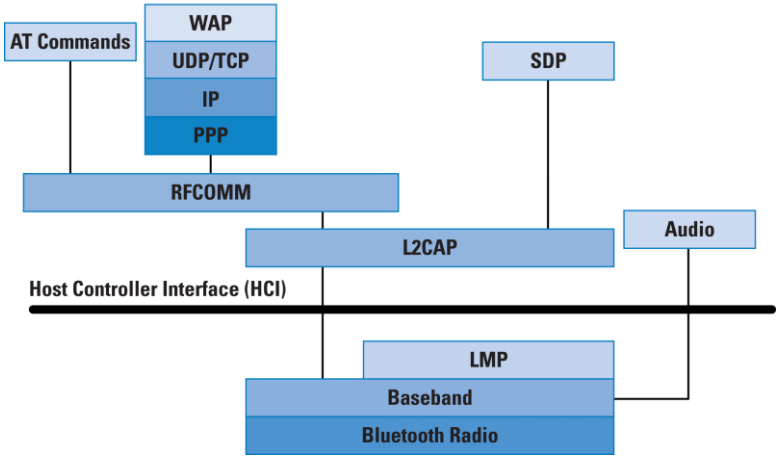
Bluetooth 3.0 foi conhecido como High Speed, sendo capaz de usar o protocolo IEEE 802.11 em conjunto para a transmissão de dados, gerando um aumento em sua velocidade de transmissão.

A geração 4 é conhecida como Bluetooth Low Energy. Essa foi criada tendo como proposito uma solução tecnológica para a comunicação usada no conceito de Internet of Things. Ao contrario da versão anterior, em que é focado em transmissões de longa duração, o sistema BLE foca em curta duração, tendo consequentemente um menor consumo de energia.

Sistema Bluetooth tem como sua arquitetura principal a chamada “piconet”, podendo conter no máximo 8 dispositivos, consistindo um nó “mestre” e tendo até 7 nós “escravos.

A pilha do sistema, representada na Figura 4, é composta dos seguintes protocolos: camada física, “base-band”, “link Manager protocol”, “Host control Interface”, “Logical Link Control Adaptation Protocol” e “Service Discovery Protocol”.

Figura 5 - Pilha do Sistema



Fonte: Cisco (www.cisco.com)

A camada física é composta por um radio operando na banda ISM (Industrial, Scientific and Medical), mas especificamente na faixa de 2.4GHz a 2.48GHz. Pelo fato de inúmeros outros sistemas usarem essa banda, para minimizar possíveis problemas, é aplicado o espectro de espalhamento por salto de frequência, chegando a 1600 saltos por segundo. A mudança é feita em toda a piconet, tendo sua temporização e sequencia de salto comandada pelo dispositivo mestre. Por causa de interferências nas primeiras versões, foi implementada como solução a adaptação da sequencia de saltos, excluindo as frequências que estejam sendo ocupadas por outros sistemas, isso foi denominado salto de frequência adaptativo.

Camada de base-band funciona em conjunto com a camada física, tendo semelhanças com a camada MAC do modelo OSI, sendo responsável pelo controle de sincronismo, endereçamento e empacotamento. Nessa camada que é realizado o emparelhamento entre dispositivos. Nas primeiras versões era realizado através de um *pin code c*onfigurado pelo usuário, definido no dispositivo mestre e era requerido a checagem desse pin code no escravo. Na versão seguinte, uma *passkey* é gerada

pelo sistema no mestre, requerendo ao usuário no dispositivo escravo a entrada dessa *passkey,* tendo assim a vantagem de ser mais seguro em relação ao sistema antigo.

Após o pareamento é realizado, o Link manager protocol estabelece o enlace, tendo dois tipos que podem ser utilizados, sendo eles o SCO (Synchronous Connection Orientated) e o ACL (Asynchronous Connectionless). O primeiro é usado para comunicações que necessitam de uma transmissão de dados em tempo real, já o ACL é usado para transmissão de pacotes de dados.

Host Control Interface é a camada responsável pela comunicação com o hardware bluetooth, utilizado para a configuração dos parâmetros do mesmo.

A camada L2CAP é responsável pelos parâmetros de Quality of service (QoS), entre eles a divisão dos pacotes de dados em quadros para transmissão e vice-versa, determinação de destino em relação a serviço, detecção de erros e sistema de retransmissão. Por ultimo esta o Service Discovery Protocol. Uma vez que o bluetooth tem vários usos e para cada uso existe o que é chamado de perfil e num mesmo ambiente pode haver múltiplos dispositivos, essa camada é responsável por identificar qual o serviço fornecido, sem a necessidade de intervenção do usuário, facilitando o processo. Essa camada utiliza para a identificação o chamado UUID (Universally Unique Identifier) único para cada serviço.

2.4 BANCO DE DADOS

2.4.1 Structure Query Language (SQL)

Structure Query Language é uma linguagem criada para a manipulação de sistemas de banco de dados. Chamada inicialmente de SEQUEL (Structured English Query Language), quando em seu estagio de desenvolvimento na IBM Research na década de 70, tendo como função a interface de um banco de dados experimental chamado Sistema R. No seu desenvolvimento, o nome foi mudado para SQL, se tornando a linguagem padrão para os sistemas de banco de dados comerciais. Em 1986, o primeiro padrão do SQL, o chamado SQL-86 ou SQL1, foi publicado em um esforço conjunto do ANSI (American National Standard Institute) e ISO (International Standards Organization), logo em seguida, em 1992, uma versão revisada e expandida foi lançada, o SQL-92/SQL2. A versão atual utilizada é a SQL3/SQL-99. Uma vez que esta em constante expansão, o SQL-99 foi dividido em duas partes, o core (Núcleo) e seus packages (pacotes), sendo o core obrigatório para todos os sistemas de banco de dados que o usam, e os packages opcionais, que podem ser adicionados seguindo a aplicações especificas.

As informações são armazenadas em um conjunto de tabelas, A Estrutura básica da sintaxe SQL, no caso de fazer uma simples consulta é composta de três clausulas: Select, from e where. Select é utilizado para a seleção de uma coluna, From é para a seleção da tabela a ser consultada e Where define uma ou mais condições especificas na consulta sendo realizada, sendo essa última opcional.

No apêndice C encontra-se o funcionamento básico das funções citadas acima e exemplos de uso das mesmas.

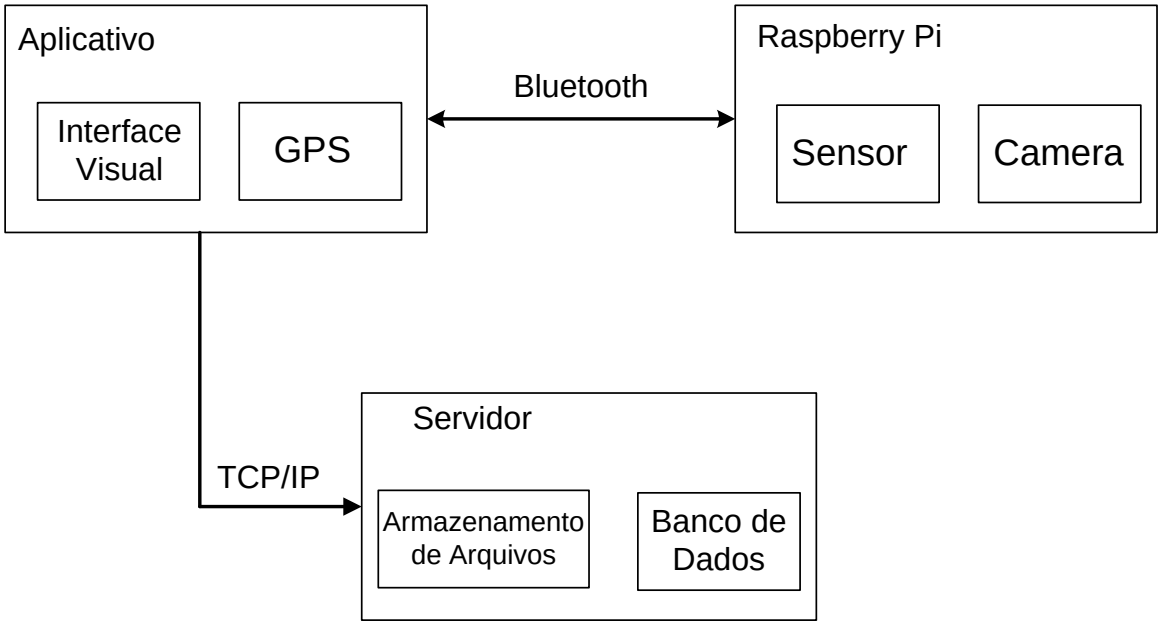
3 DESENVOLVIMENTO

Para o embasamento teórico desse trabalho foi feito uma revisão da tecnologia a ser aplicada no sistema a ser desenvolvido. Em seguida, foi desenvolvido um software protótipo para testar o funcionamento do aplicativo utilizado, assim como o hardware implementado. Após essa etapa, foram realizados testes de funcionalidade dos sistemas de monitoramento, comunicação e aferição dos sensores.

3.1 VISÃO GERAL

O Sistema é composto por três partes, sendo esse representado pelo diagrama de blocos da figura 5: o Aplicativo instalado no smartphone, a Raspberry PI e o servidor.

Figura 6 - Diagrama de Blocos do sistema



Fonte: Criação Própria

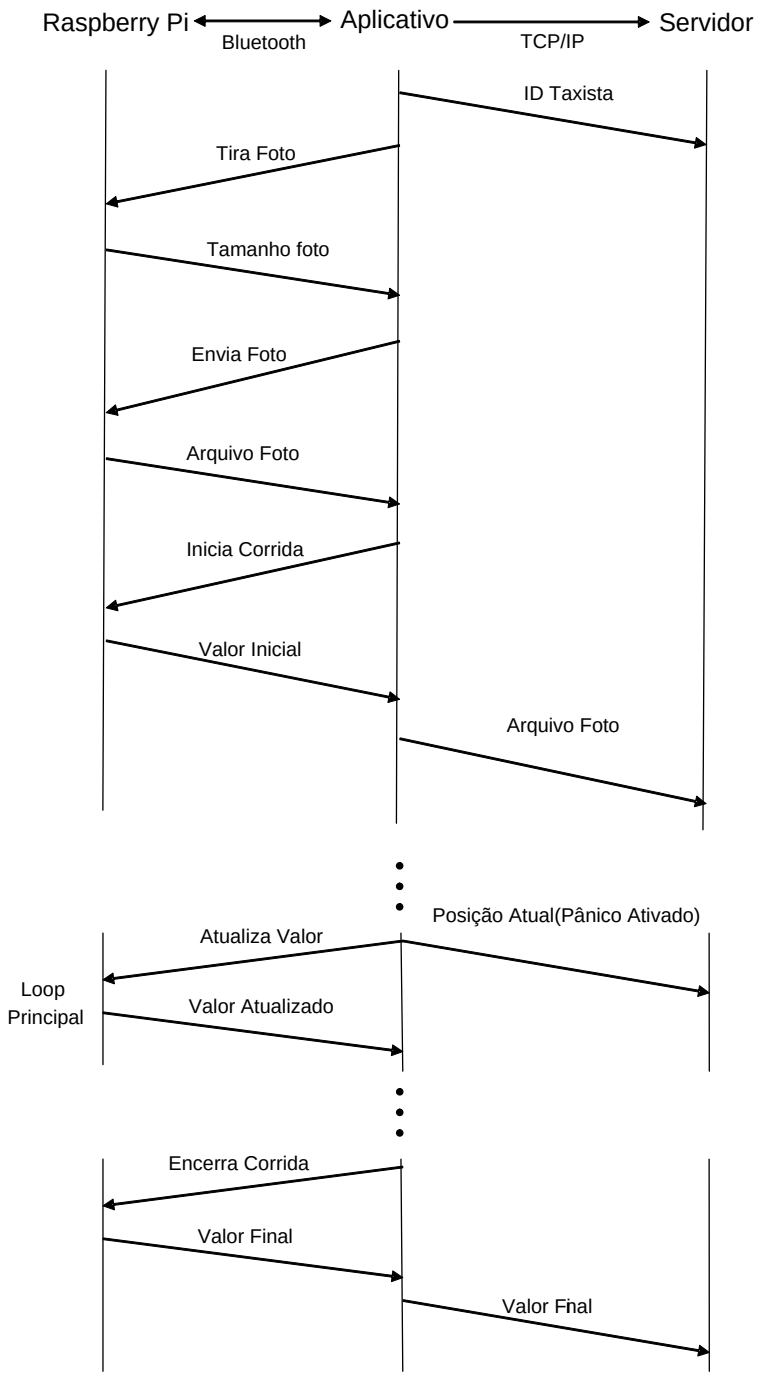
O aplicativo é responsável pela interface, tanto para a operação realizada para o motorista, quanto para a disponibilização de informações para o passageiro.

O mesmo também é responsável pela aferição do sistema de GPS existente no smartphone, usado para a localização atual do veículo, seja para registros ou no caso do sistema de pânico ser acionado, seu posicionamento em tempo real. A Raspberry PI carrega os elementos essenciais para o funcionamento do taxímetro, que são a câmera fotográfica e o sensor de efeito hall, sendo esse ligado à transmissão do veículo, mas para princípio de testes está sendo usado um gerador de função, uma vez que a saída do sensor é um trem de pulsos.

Por final temos o servidor, responsável pelo recebimento e armazenamento de arquivos e o gerenciamento do banco de dados, que é usado para os registros da corridas feitas, contendo informações sobre a mesma como horário de início e fim, valor final e se houve anomalias, além de estabelecer a ligação dos arquivos em relação a corrida em que foram gerados.

O diagrama da Figura 6 mostra a sequência de comunicação do sistema, desde a parte inicial, onde são feitas as configurações necessárias para o processo seguinte. Este é composto pelo loop principal do sistema e o processo de finalização, onde a corrida é totalizada, logs de finalização gerados e enviados ao servidor e por fim, valor final é disponibilizado na interface visual.

Figura 7 - Diagrama Sequencial do sistema



Fonte: Criação Própria

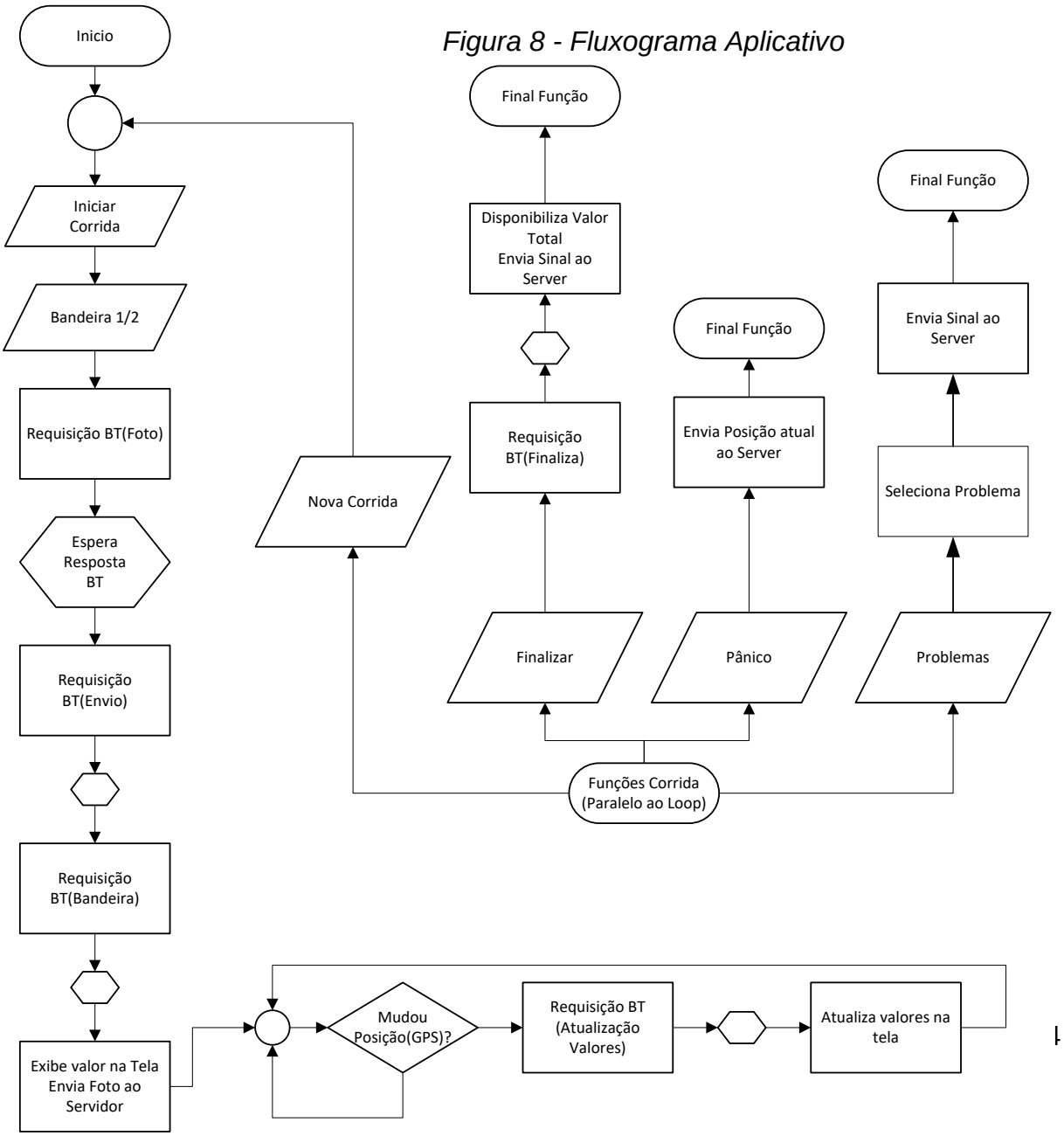
3.2 SOFTWARE

3.2.1 APLICATIVO ANDROID

Para a criação desse aplicativo foi utilizado o Android Studio versão 2.2,3 no sistema operacional Linux.

O Funcionamento padrão do sistema, este representado pelo fluxograma presente na figura 7, consiste em duas partes especificas: setup e loop principal. O setup consiste em obter a foto do passageiro e inicializar a Raspberry PI para o modo corrida enquanto o loop principal é responsável pela aferição de distância e consequentemente a totalização de valores.

Figura 8 - Fluxograma Aplicativo



Fonte: Criação Própria

O Setup começa com a entrada do usuário em relação a opção de qual tarifa vai ser utilizada. Após é feito o processo para obter a foto, usando o protocolo bluetooth. Esse é dividido em duas requisições, a primeira envia a ordem para tirar a foto, tendo como retorno o tamanho do arquivo para o uso na segunda requisição, em que a resposta esperada é o arquivo em si. Após finalizado a obtenção da foto, o aplicativo envia para a Raspberry PI a tarifa selecionada. Uma vez recebida a tarifa, a aferição é iniciada e o valor iniciado é retornado para o smartphone. Finalizando a etapa de setup tem-se o envio da foto para o servidor.

O loop principal tem como base a checagem da mudança de posição através do sistema de posicionamento global (GPS), sendo a mesma feita a cada cinco segundos. No momento que a posição é mudada, é feita uma requisição bluetooth para obter o ultimo subtotal aferido, tendo o mesmo atualizado na tela principal no momento que a requisição é finalizada. Quando em modo corrida, são disponibilizadas três funções para o usuário: Finalizar a corrida, reportar um problema e acionar o pânico.

A função finalizar é usada para sinalizar a chegada do percurso para a Raspberry PI, assim totalizado o valor e enviando a aferição mais atualizada. Neste momento é feito o envio das informações da corrida para fins de registro em banco de dados em relação a valor final e horário de chegada.

Relatório de problemas, representado pela função problemas, é usado para reportar a central situações que não represente ameaça direta ao taxista, como possíveis danos ao veículo que resultem na imobilidade do mesmo, e no caso, a necessidade do envio de uma unidade para substituir ou a possível ocorrência do não pagamento da corrida.

O sistema de pânico, indicado por Pânico no fluxograma, é usado quando o taxista encontra uma ameaça direta a si mesmo, funcionando de forma silenciosa. Quando acionado, a central recebe o alerta de qual veículo foi acionado. Enquanto isso, toda vez que o GPS é atualizado, é enviado a posição para a central, assim, rastreando o veículo, possibilitando uma interceptação e caso não seja possível, existe uma foto ligada a corrida, sendo possível entregar a mesma para as autoridades competentes.

3.2.1.1 ALGORITIMO DE TRANSMISSÃO DE DADOS UTILIZANDO LINK BLUETOOTH

O algoritmo para comunicação bluetooth pode ser considerado parte essencial do sistema, uma vez que o aplicativo tem como função ser a interface homem-máquina, executando o envio das requisições do usuário para a Raspberry PI. A implementação em thread foi usada para que a comunicação bluetooth possa ser feita em paralelo à operação do aplicativo. Para garantir que duas conexões bluetooth não sejam estabelecidas ao mesmo tempo, foi implementado um semáforo em uma classe singleton, sendo já usada para o transporte de variáveis entre activities. Assim, toda vez que uma requisição bluetooth é realizada, uma flag é levantada, mantendo o sistema em um loop, até a requisição tenha finalizado, mas a mesma é feita em uma thread em segundo plano, evitando o congelamento do programa.

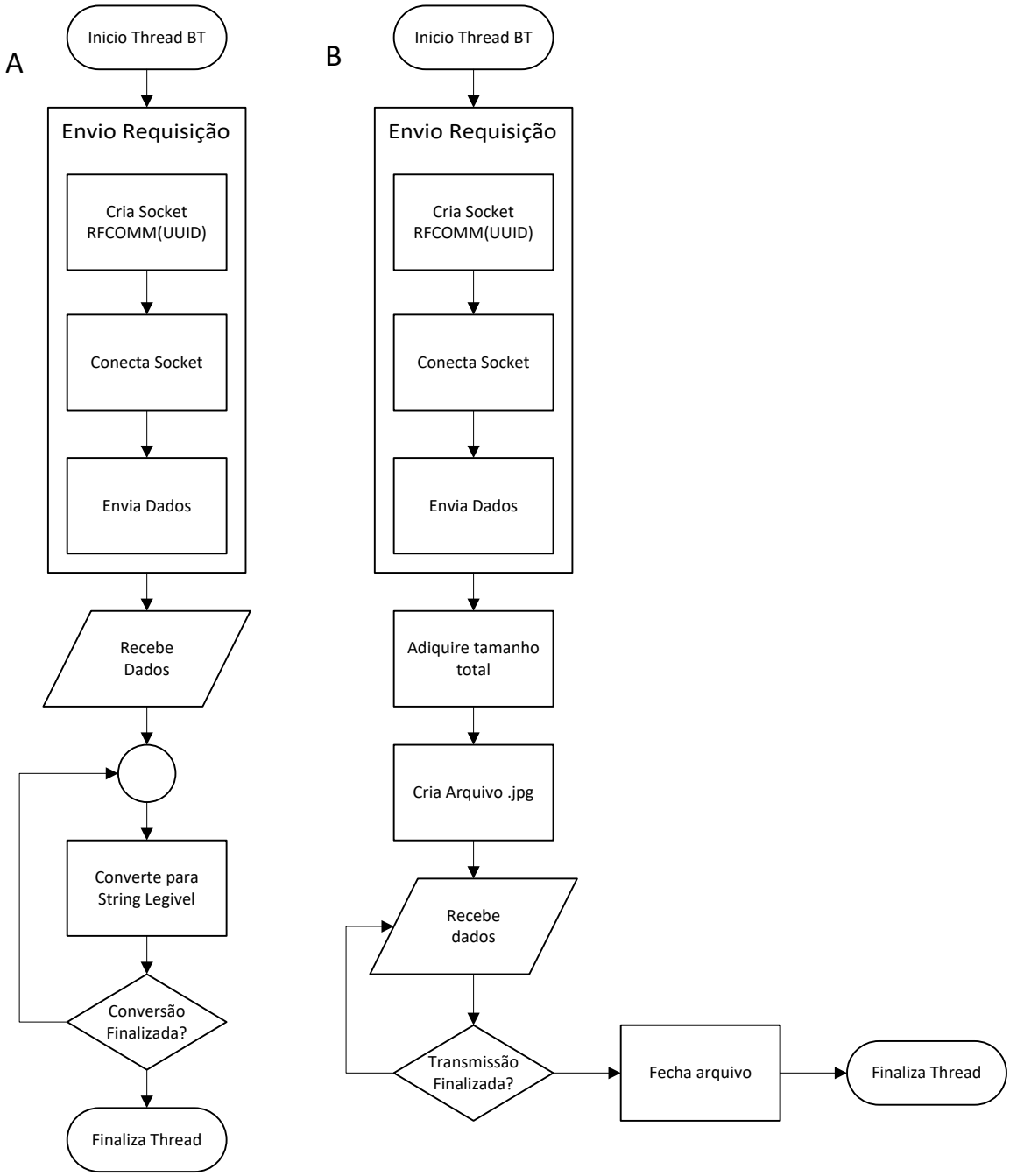
O sistema foi dividido em dois algoritmos, um para a transferência de comandos e o outro para a transferência da foto, representados na figura 8, sendo o “A” para comandos e o “B” para transferência de fotos. Para ambos, o início é semelhante, tendo como diferença, no modo comando, a maneira que o socket bluetooth é criado, uma vez que o mesmo pode ser chamado inúmeras vezes numa corrida, este fato pode gerar erro no sistema de descoberta de serviço, gerando um erro que congela o sistema bluetooth do smartphone. Para resolver isso, ao invés de usar diretamente o método usado para estabelecer um socket, a chamada foi feita através da reflexão de classe, no caso de chamadas repetidas, usadas na atualização de valores.

Após criado o socket e o mesmo conectado a Raspberry PI, é feito o envio da requisição. A partir desse ponto, o prosseguimento da thread se diferencia para comandos e arquivos.

Para comandos, é feito de imediato o recebimento da resposta enviada pela RPI e convertida de bytes para string legível e uma vez a conversão finalizada, a string é disponibilizada na tela ou armazenada para uso posterior, o socket é fechado e a thread é finalizada.

Para o modo de transferência de arquivo, primeiro é adquirido o tamanho total do arquivo a ser recebido, em seguida é criado um arquivo jpeg em branco para receber os dados. Uma vez criado, é feita a transmissão e quando finalizada, o arquivo é fechado e a thread é eliminada. Para garantir que o arquivo é enviado em sua totalidade e que o socket será fechado foram tomadas duas precauções: a primeira é ligar a continuidade do processo de transferência ao tamanho total do arquivo, recebido anteriormente através da resposta da requisição da foto, sendo esse comparado a cada pacote recebido, a segunda é uma flag de final de transmissão enviada pela Raspberry PI, sendo essa responsável pela finalização da thread.

Figura 9 - Fluxograma Protocolo Bluetooth

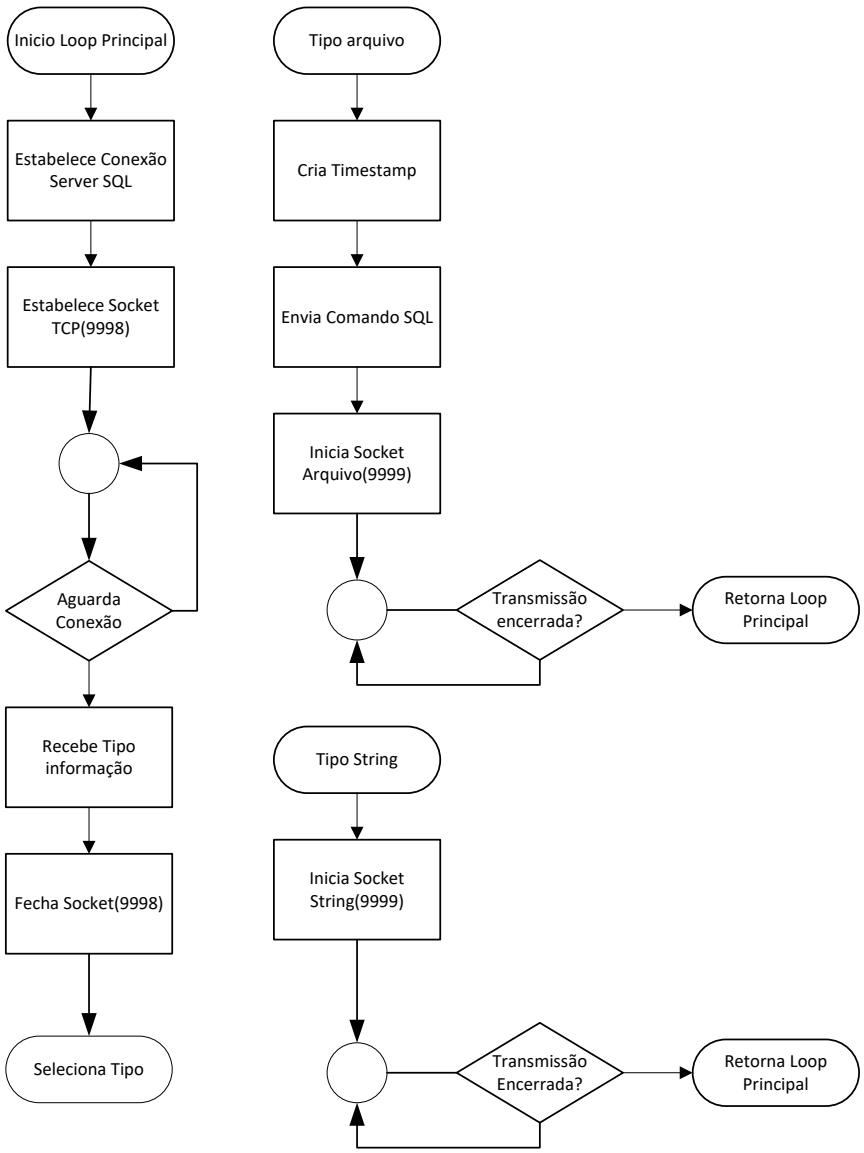


Fonte: Criação Própria

3.2.2 SERVIDOR JAVA

O servidor tem como função principal o armazenamento dos dados enviados pelo veículo, sendo eles: fotos, informações sobre localização, distância e valores. A transferência de dados é feita em duas etapas, a primeira tendo como função preparar o servidor para o recebimento de dados e a segunda sendo a transferência em si. A primeira etapa é um loop responsável por deixar o sistema pronto para receber dados. Antes de entrar no loop, é realizada a conexão com o banco de dados SQL implementado, após isso é estabelecido um socket TCP/IP na porta 9998. A partir desse ponto, o servidor fica aguardando a conexão do veículo. No momento que recebe uma requisição, o sistema interpreta a mesma e fecha o socket 9998.

Figura 10 – Fluxograma Socket Principal

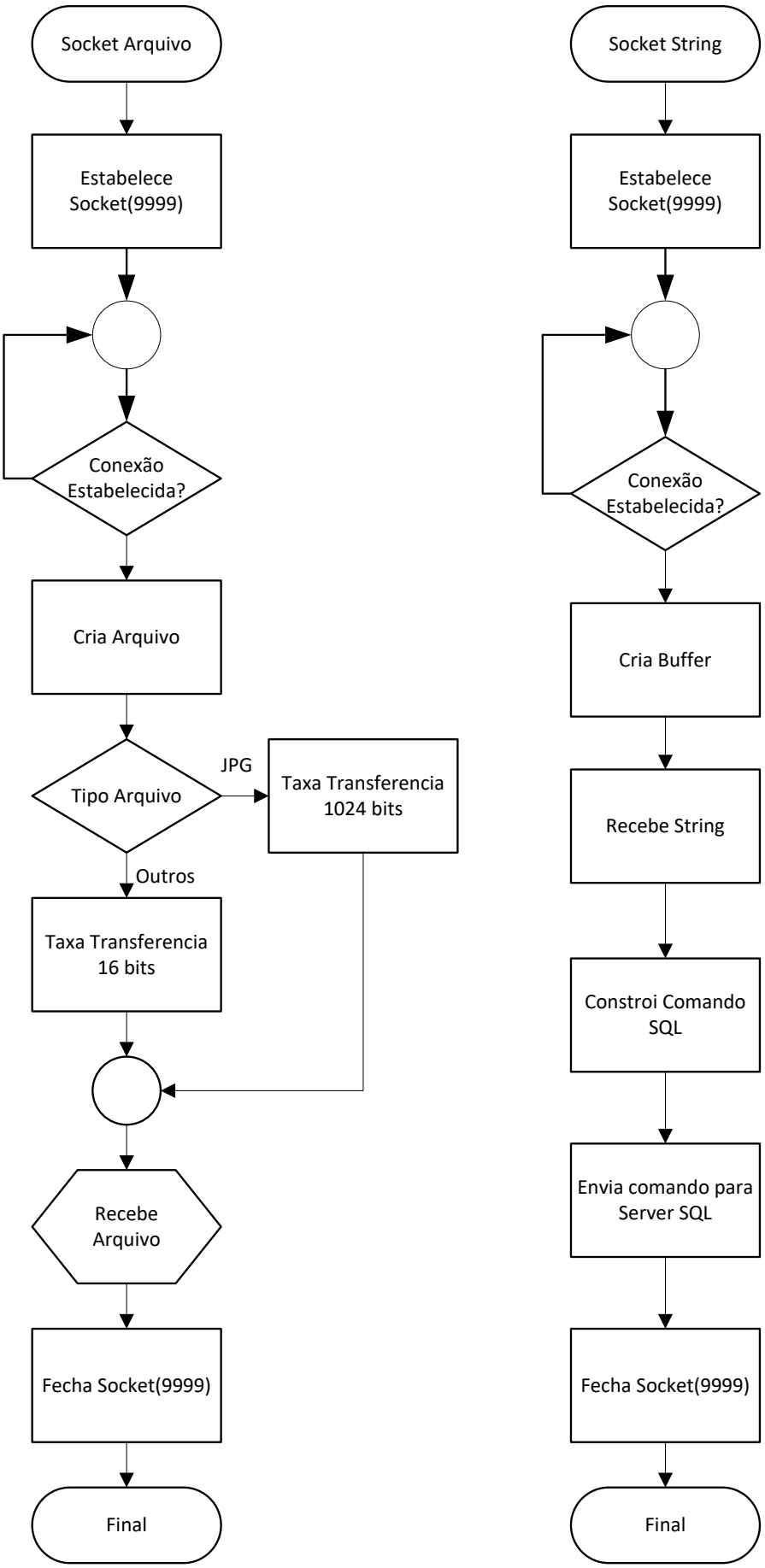


Fonte: Criação Própria

Dois tipos de requisições podem ser feitos: Transferência de arquivos e envio de strings. O algoritmo da transferência de arquivos é usado para três propósitos: envio da foto para a central, envio do sinal de pânico e relatório de problemas. O que diferencia as três funções é o comando SQL enviado para a central, composto do endereço onde o arquivo recebido estará localizado no sistema operacional e o comando especifico para cada arquivo. Para a foto é enviado as informações de início de corrida e para pânico e problemas é enviado a flag referente. Este processo inicial do socket na porta 9998 é representado pela figura 8 na página anterior.

Finalizando a comunicação com o banco de dados, é aberto um socket TCP/IP implementado porta 9999, este representado pela figura 10 na página seguinte. No caso de transferência de arquivos, após aberto, a conexão é estabelecida e o arquivo que receberá os dados é criado. A taxa de transmissão é ajustada de acordo com o tipo de arquivo sendo recebido, sendo ela 1024 bytes para fotos e 16 bytes para outros. Feito isso, a transferência é realizada e quando finalizada, o socket na porta 9999 é fechado e retorna ao loop principal. Para a transferência de string, não é feita nenhuma operação antes do socket ser aberto, como na transferência de arquivo. Uma vez aberto, o socket fica aguardando a conexão e no momento que a mesma é estabelecida, é criado o buffer que receberá a string. Após finalizado, é construído a expressão SQL que será enviada ao banco de dados e uma vez finalizado, o socket é fechado e retorna ao loop principal, como na transferência de arquivos.

Figura 11 - Fluxograma Socket Transmissão



Fonte: Criação Própria

3.3 FIRMWARE

3.3.1 RASPBERRY PI

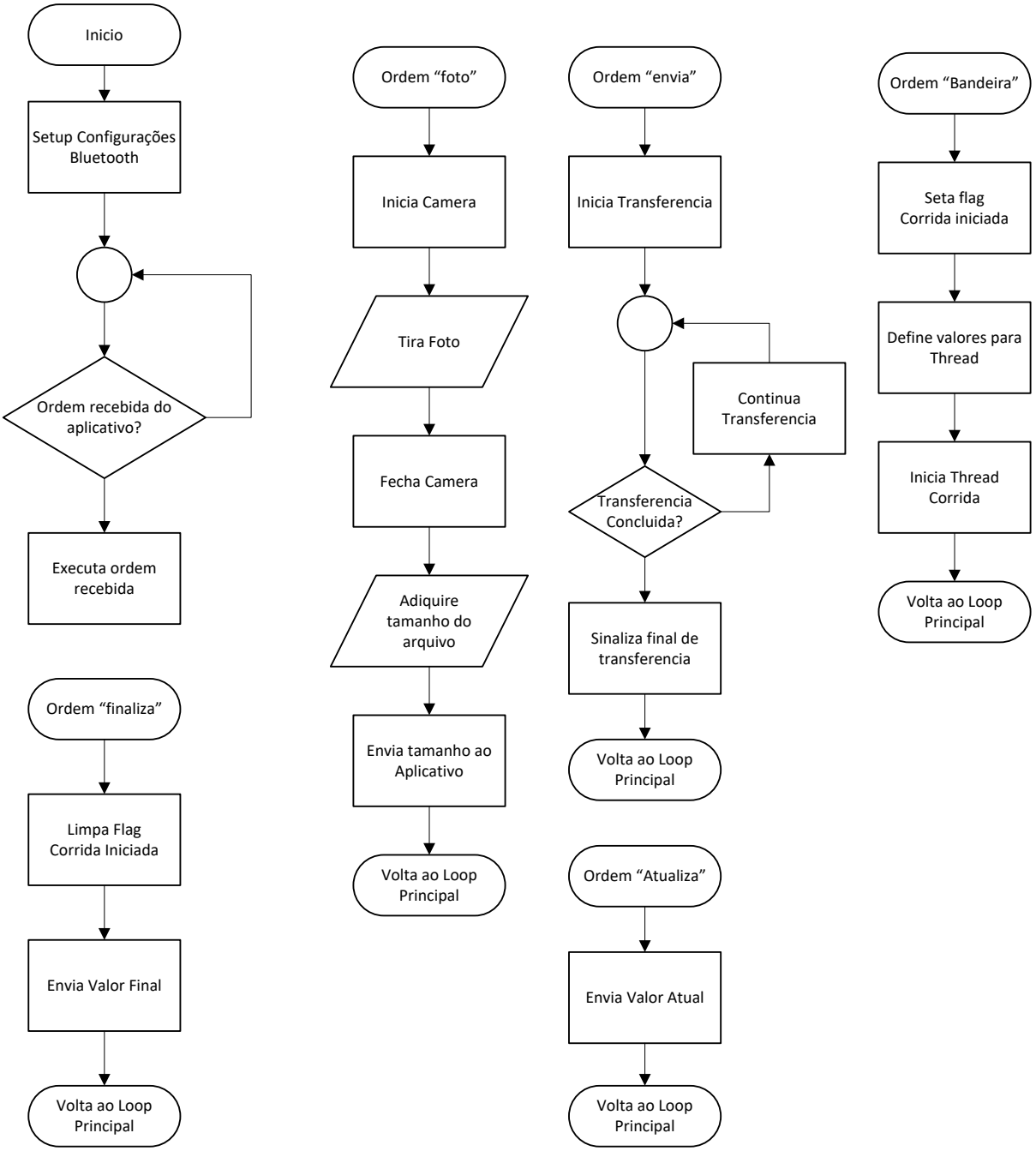
O firmware responsável pela aferição e totalização do taxímetro, representado na figura 11, aplicado na Raspberry PI funciona em um loop básico utilizado para deixar o sistema pronto para qualquer requisição feita pelo aplicativo. O programa inicia dando partida no sistema bluetooth conectado a mesma, inserindo as configurações necessárias e abrindo o socket necessário para comunicações. O sistema é preparado para receber 5 tipos de requisições: finalizar, tirar foto, enviar foto, atualizar, iniciar corrida.

A ordem “foto” é utilizada para a Raspberry PI obter uma fotografia do passageiro usando a câmera instalada no sistema, sendo sua execução composta de iniciar o periférico, tirar a fotografia, e fechar o periférico, garantindo que o mesmo estará livre para uso posterior. Após a criação do arquivo, é feita uma rápida leitura para obter seu tamanho total sendo o mesmo é enviado como resposta da requisição feita. Após a transferência de arquivo, executada pela ordem “envia”, ser finalizada, o sistema envia um sinal indicando o termino.

Para o início de corrida, é usada a ordem “bandeira”, que nesse caso, esta dividida em duas funções, uma para cada tarifa, bandeira 1 e bandeira 2. Quando executada, é definida a flag de corrida iniciada, inserido os valores referentes a tarifa escolhida e por fim, iniciada a thread. O uso de thread foi escolhido pelo fato de que a medição de distância e valores, uma vez iniciada, não pode sofrer interrupções para manter sua precisão. A mesma só pode ser parada quando solicitada e executada em paralelo ao loop principal do firmware.

As ordens “atualizar” e “finalizar” funcionam de forma semelhante, tendo em sua diferença, na “finalizar”, o fato que a mesma limpa a flag que mantem a thread responsável pela aferição e totalização em execução

Figura 12 - Fluxograma Raspberry PI



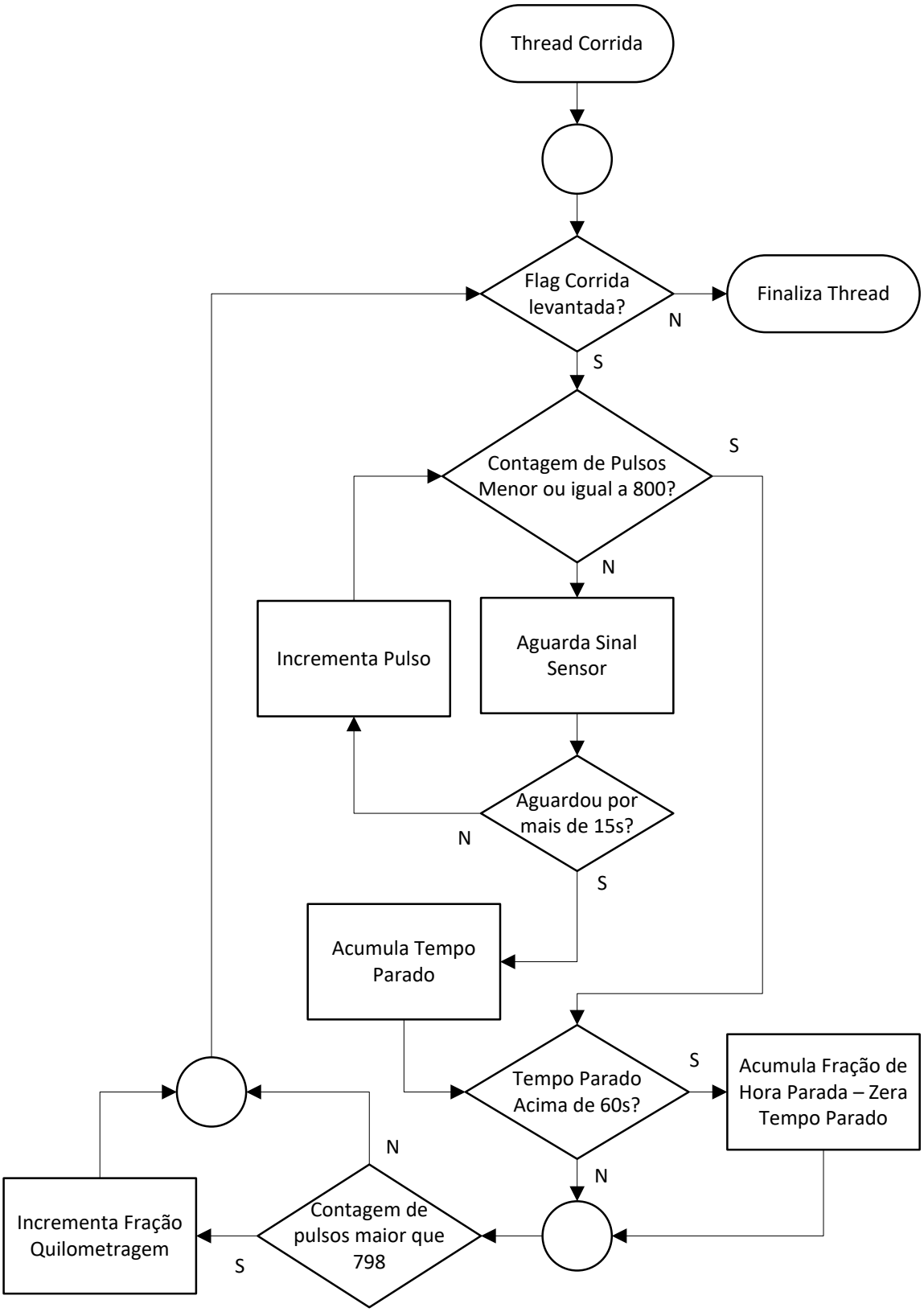
Fonte: Criação Própria

O funcionamento da thread, demonstrado pelo fluxograma na figura 12, tem como base a contagem de pulsos gerados pelo sensor de efeito hall instalado no veículo. No momento que a contagem de pulsos chega ao valor definido, que nesse caso está definido em 800 pulsos, considerando que 1 metro corresponde a 8 pulsos e mantendo medição em frações de 100 metros, a fração do valor do quilometro rodado é adicionada no subtotal da corrida.

Para a validação do algoritmo de aferição de distância foi utilizado um gerador de funções conectado a uma das portas de I/O da Raspberry PI, simulando o sinal gerado pelo sensor de efeito hall, caracterizado por um trem de pulsos tendo sua frequência variada pela velocidade do veículo. Um algoritmo de teste foi elaborado, sendo esse composto por um contador simples com o gatilho para incremento sendo o sinal recebido na porta de I/O e reiniciando a contagem no momento em 800. Esse mesmo algoritmo foi utilizado como base para o sistema principal implementado na Raspberry.

Como a maioria dos percursos realizados por taxistas é realizado em ambiente urbano, há a possibilidade do trajeto possuir engarrafamentos. Uma vez que a cobrança é feita a partir da movimentação do veículo, um modo de cobrança secundário é usado nesses casos, sendo esse chamado de hora parada. Para esse sistema foi aplicado uma checagem de tempo em que o firmware fica parado na checagem do sinal do sensor, e caso o tempo limite seja estourado, definido em um minuto, uma fração do valor da hora parada é acumulado. Esse processo ocorre indefinidamente até a ordem de finalizar é invocada, limpando a flag que mantem a thread ativada, sendo a mesma checada no início do loop

Figura 13 - Fluxograma Thread



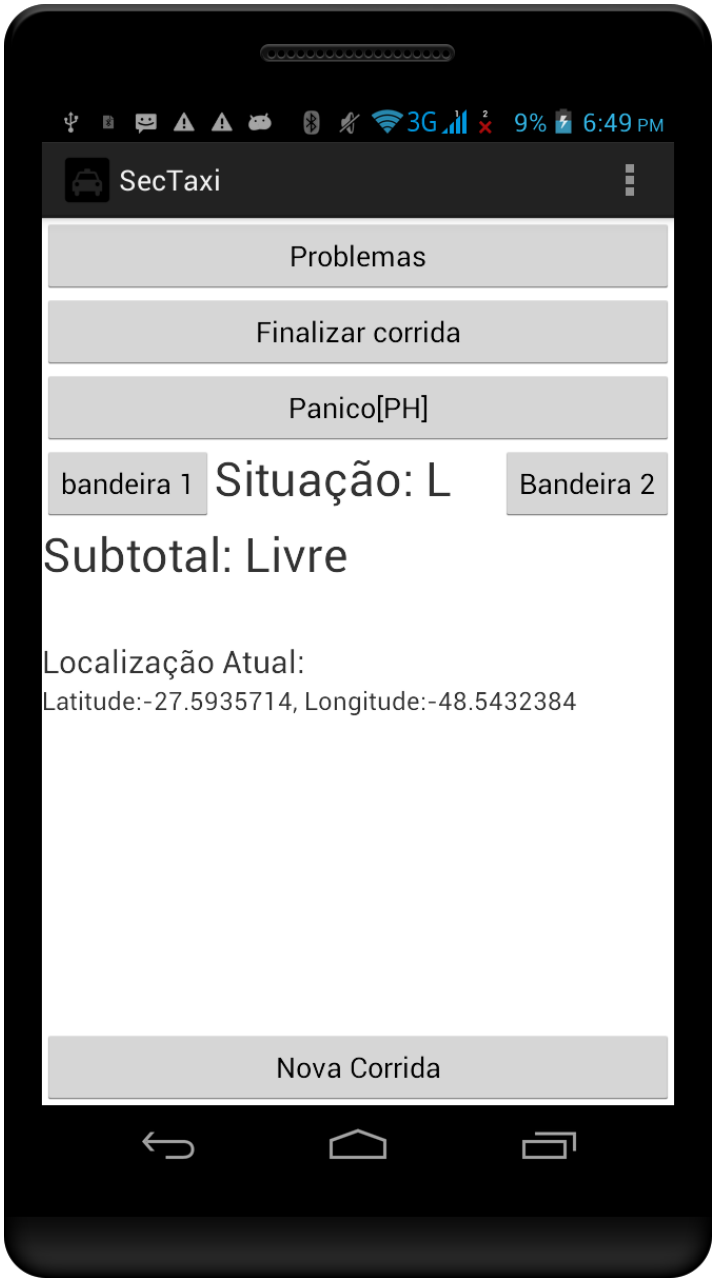
Fonte: Criação Própria

4 RESULTADOS

Nesta parte serão mostradas as telas do aplicativo criado, demonstrando os diferentes modos de operação e as fotos do protótipo montado.

A Figura 13 demonstra a tela o aplicativo no modo “livre”, correspondendo ao momento que uma corrida não está em andamento. Para uma corrida ser iniciada, o motorista pressiona uma das duas opções disponibilizadas.

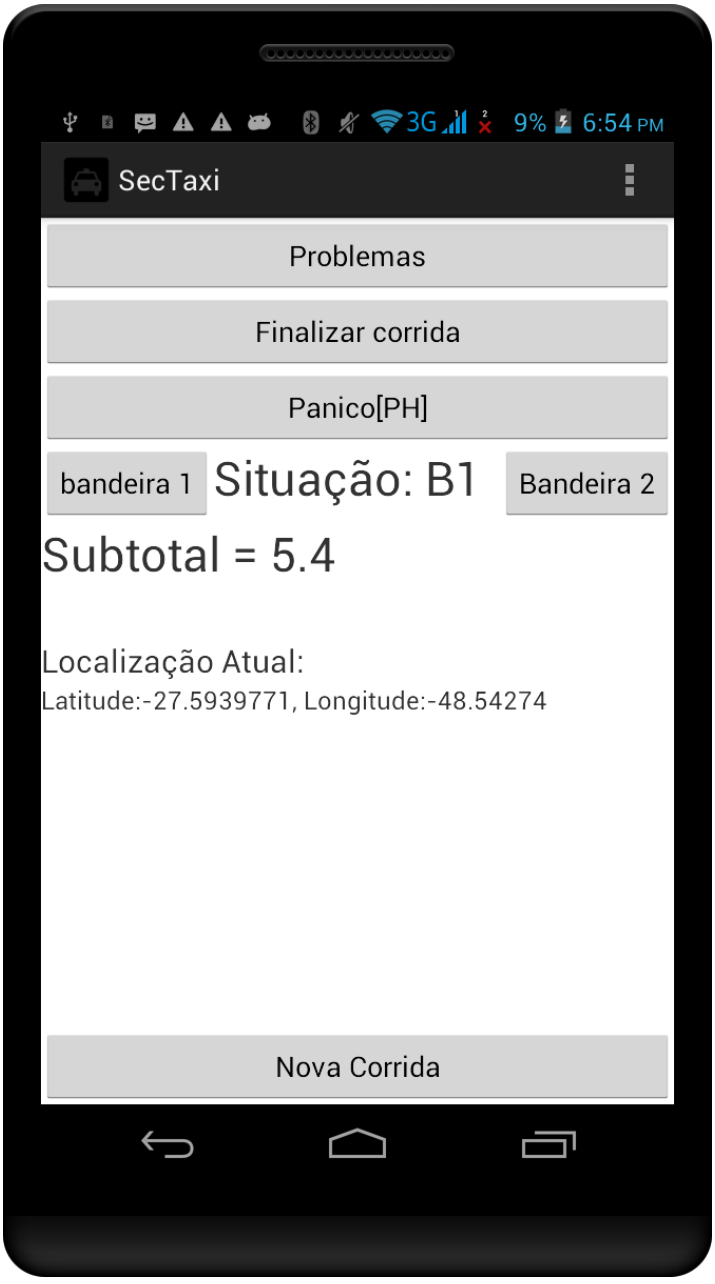
Figura 14 - Motorista Livre



Fonte: Criação Própria

Figura 14 demonstra o estado do aplicativo no momento em que uma corrida esta em andamento, indicado pela Situação “B1”, nesta situação indicando Bandeira 1 e também o Subtotal da corrida atual. Para finalizar o motorista pressiona finalizar corrida.

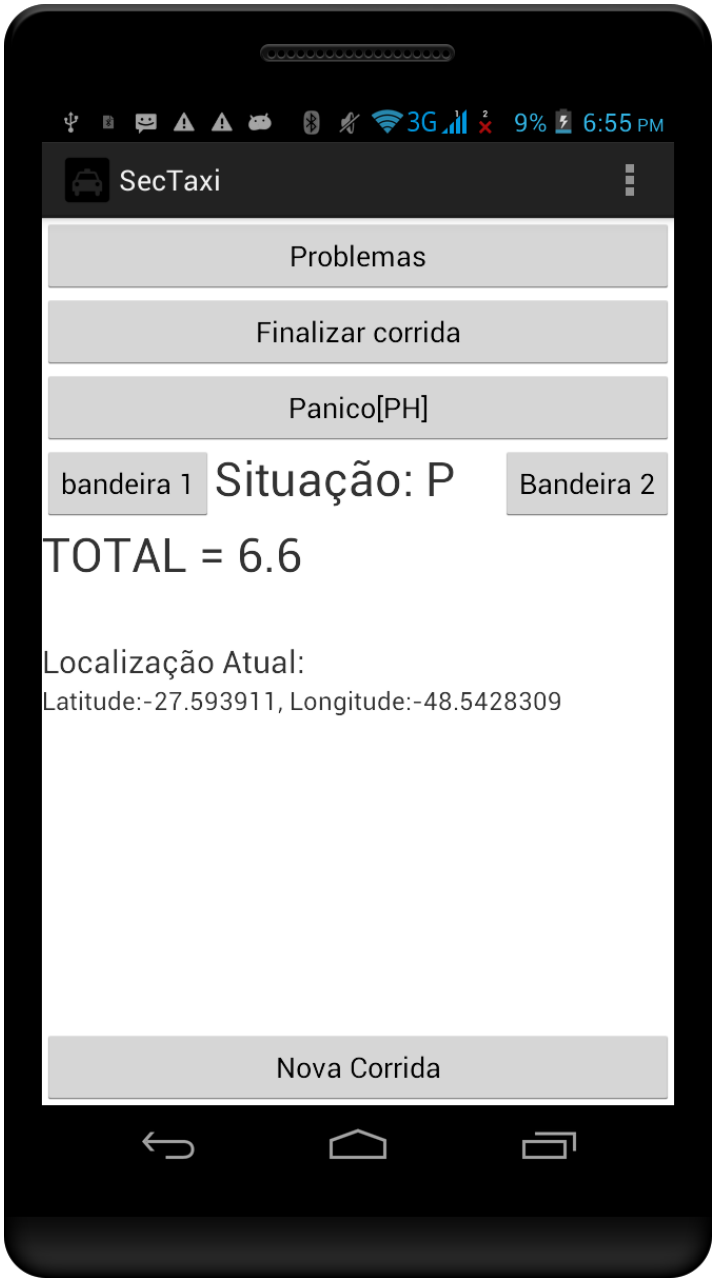
Figura 15 - Corrida em Andamento



Fonte: Criação Própria

Figura 15 demonstra o aplicativo no estado de finalização, indicado pela situação “P”, significando parada ou *park*, e o valor final da corrida de demonstração. Uma vez o valor coletado, para iniciar uma nova corrida, o motorista pressiona o botão “Nova Corrida”

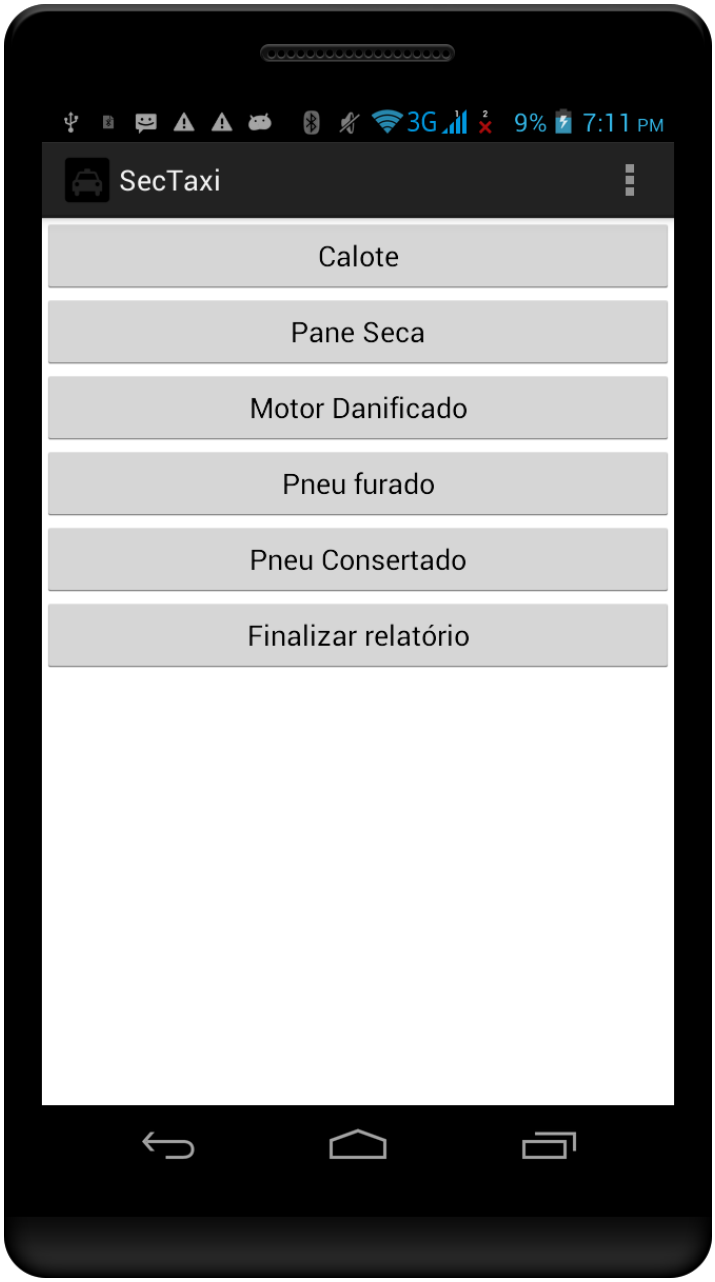
Figura 16 - Corrida Finalizada



Fonte: Criação Própria

Na Figura 16 está a tela responsável pelo relatório de problemas, neste o motorista pressiona o botão correspondente ao problema a ser relatado e a mensagem é enviada ao servidor. O botão “Finalizar Relatório” retorna a tela onde mostra o subtotal e situação.

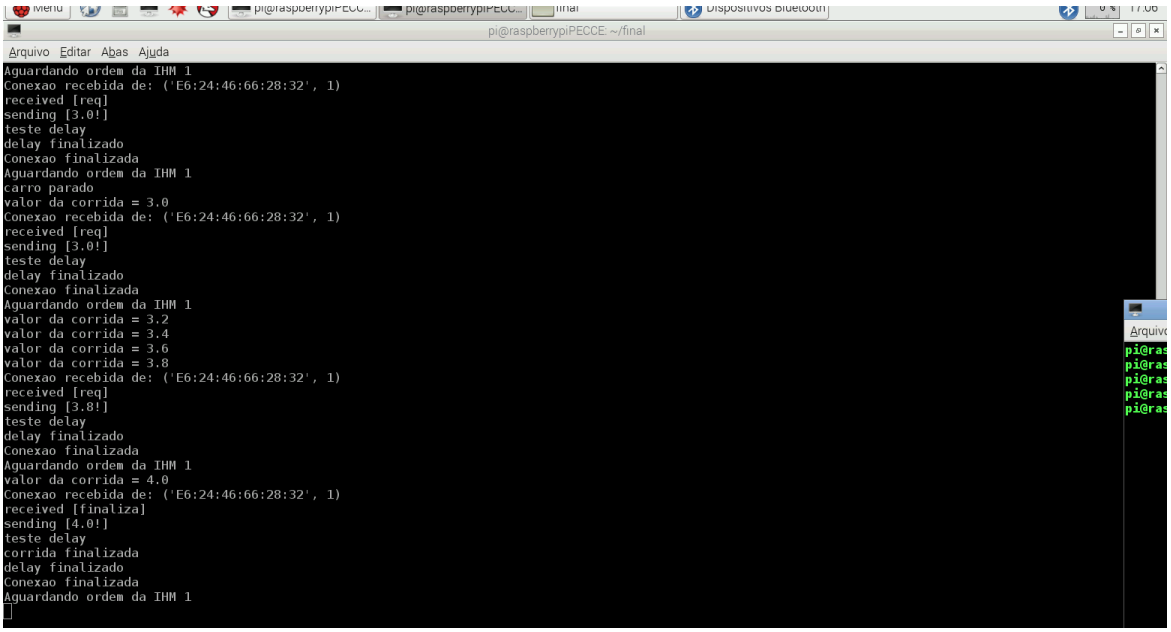
Figura 17 - Tela "Problemas"



Fonte: Criação Própria

Figura 17 Mostra a saída do console que esta executando o código em python, na Raspberry PI, responsável pela aferição do sensor, totalização de valores e obtenção das fotos. O log completo está presente no Apêndice A – Log Raspberry PI. No Apêndice B está a saída do console do servidor responsável por coletar e armazenar os dados.

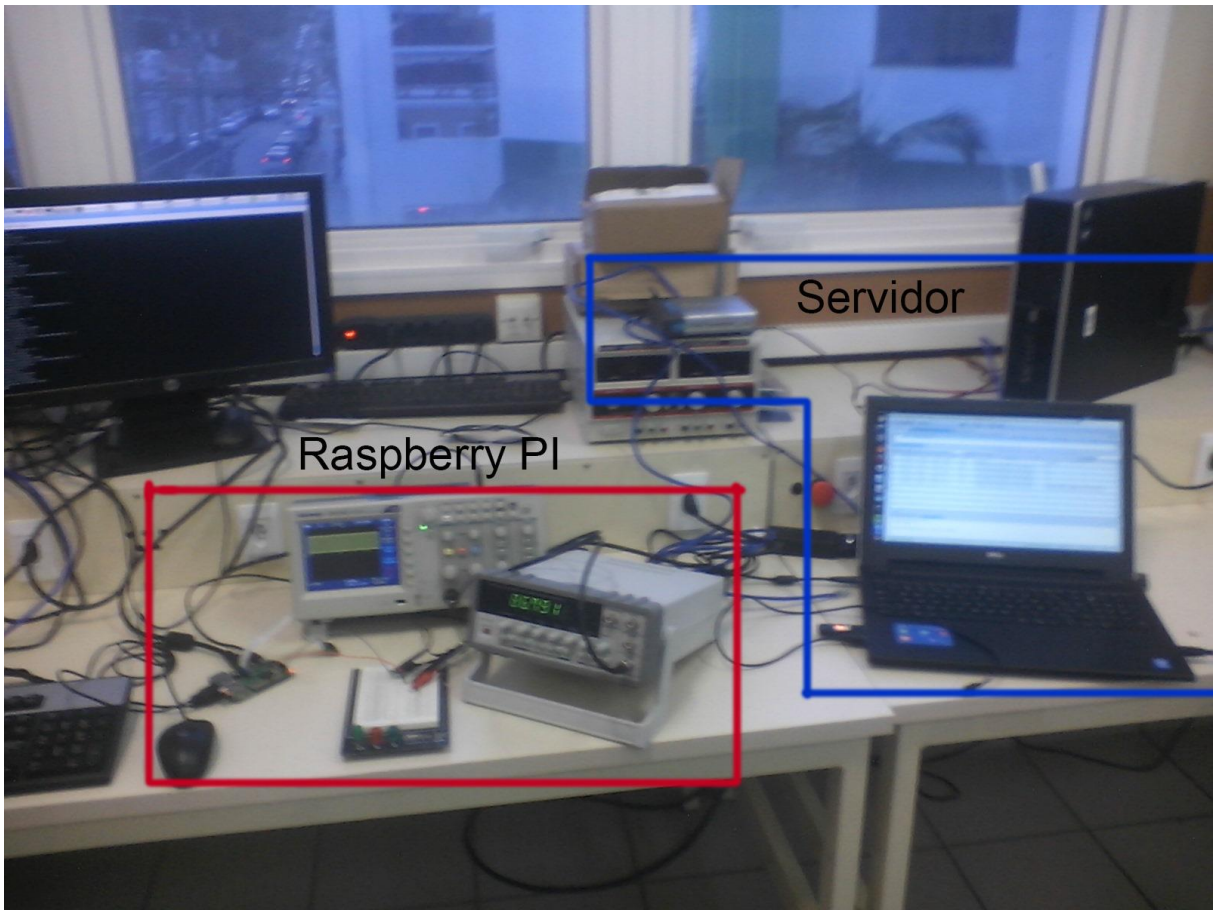
Figura 18 - Console Raspberry PI



Fonte: Criação Própria

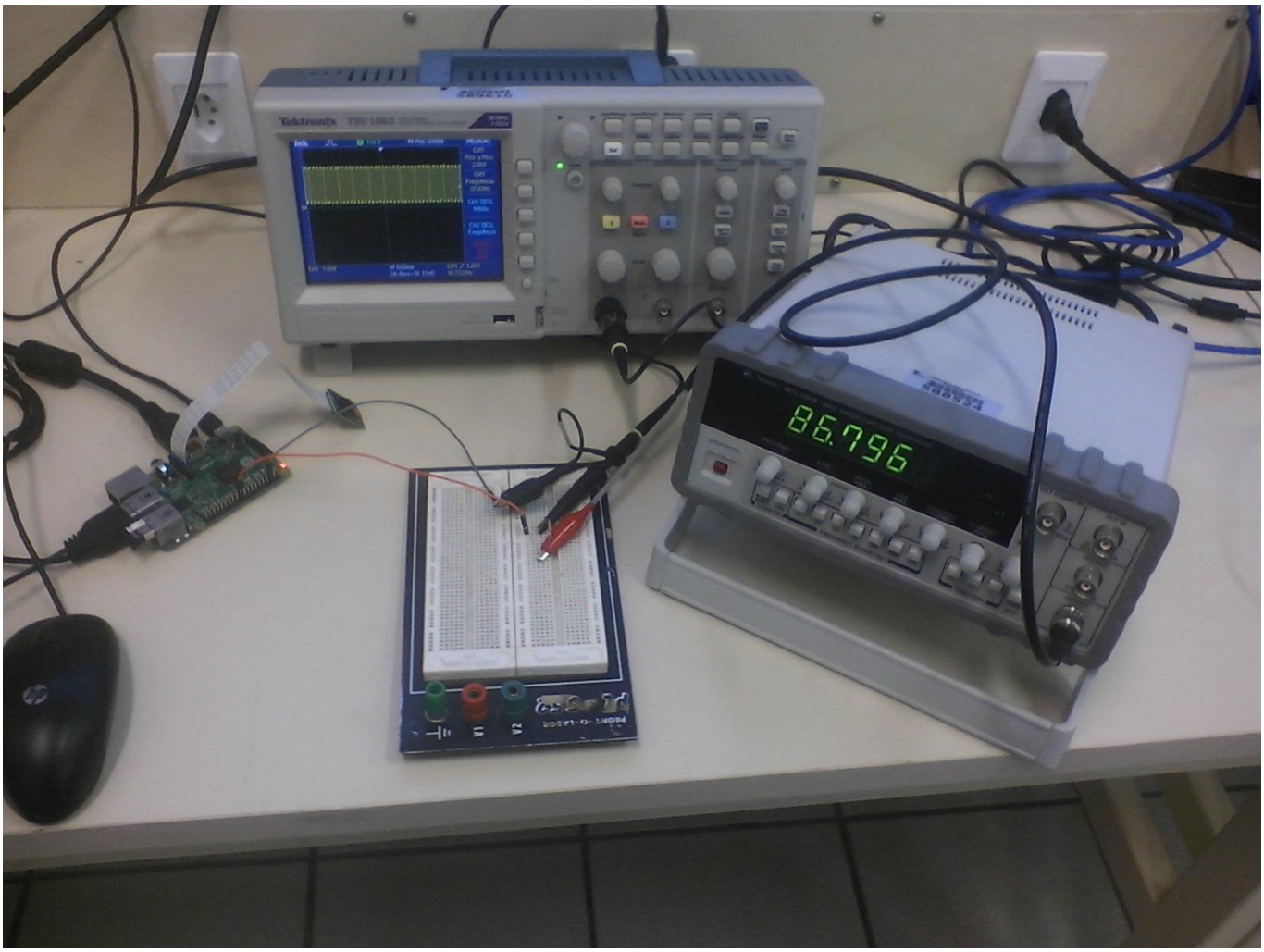
Figuras 18, 19 e 20 demonstram o protótipo montado em laboratório. Na figura 18 estão todos os dispositivos do sistema, incluindo o smartphone, que neste caso, esta sendo usado como câmera fotográfica para registro. A esquerda, sobre a mesa, se encontra a Raspberry PI, demarcado pela área em vermelho, onde a câmera e o gerador de sinais estão conectados à mesma, assim como o monitor. O Gerador foi usado para a simulação do sensor de efeito hall. A direita, marcado pela área azul, está sendo executado no laptop o servidor e em cima da fonte de tensão ao centro esta o roteador wireless, que esta estabelecendo a conexão entre smartphone e servidor. Figura 19 demonstra, em um ângulo aproximado, a Raspberry PI e seus periféricos instalados. Figura 20 é uma foto tirada a partir da câmera instalada na Raspberry PI, com o intuito de demonstrar o funcionamento da câmera e o smartphone utilizado. No Apêndice C se encontra as imagens referentes ao banco de dados utilizado.

Figura 19 - Sistema Completo



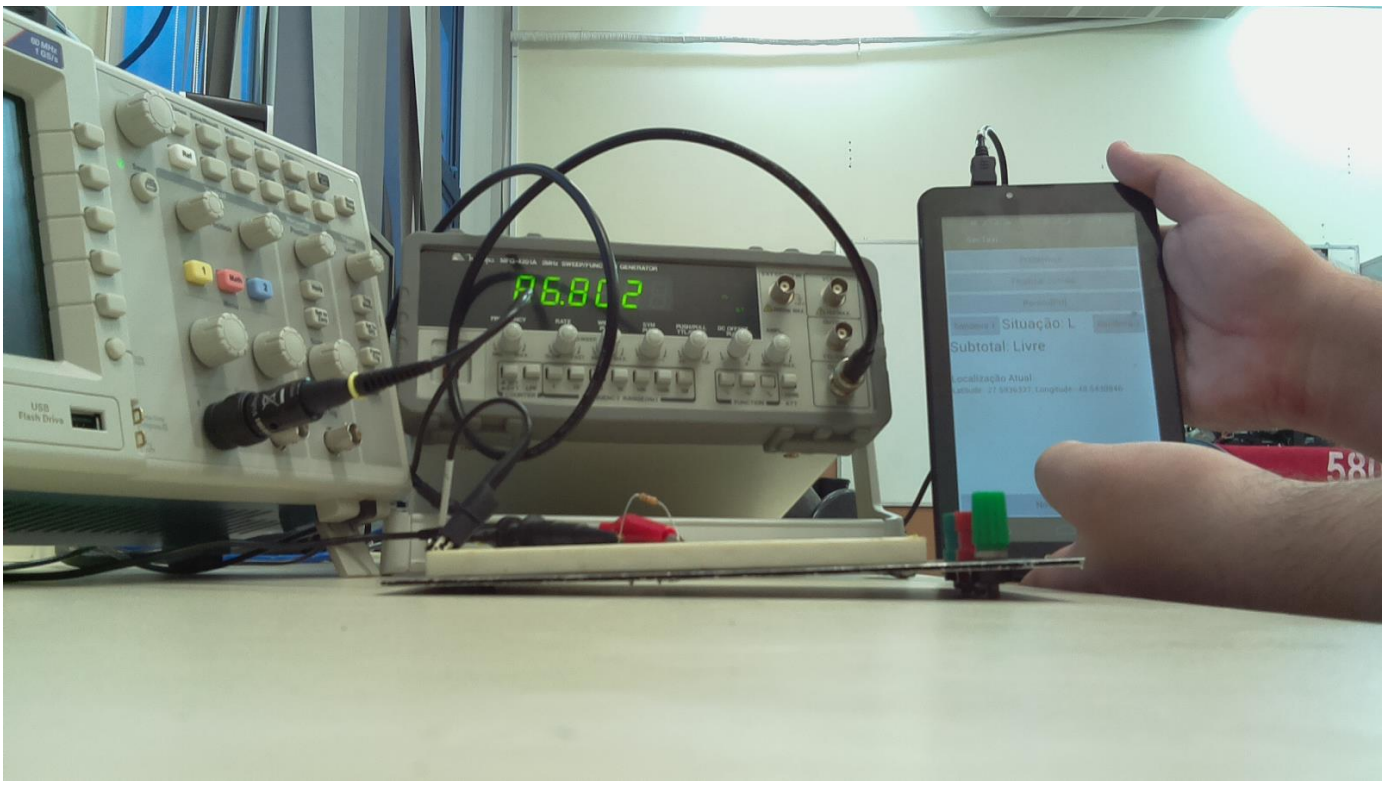
Fonte: Criação Própria

Figura 20 - Close Raspberry PI



Fonte: Criação Própria

Figura 21 - Angulo CameraPi



Fonte: Criação Própria

Houveram problemas no desenvolvimento desse projeto, sendo na maioria deles na parte que se refere a comunicação via Bluetooth. Um deles é uma desativação acidental do sistema que ocorre na primeira tentativa de conexão. No presente momento, a solução temporária para esse problema é preemptivamente desativar e ativar a comunicação bluetooth logo após a inicialização da Raspberry PI. Tendo como base a solução, julga-se que esse problema pode ter como origem um conflito do sistema operacional como o middleware utilizado para estabelecer a conexão.

Também houve problemas no que se refere a velocidade de transmissão, tendo como máximo 1kB por segundo, abaixo do normal esperado aferido através de transmissões de teste feitos com protocolos existentes, como o OBEX, sendo por volta de 100kB por segundo. Esse problema gerou um aumento excessivo no tempo total de execução da rotina de início de corrida, levando um tempo médio de três minutos partindo da escolha da bandeira até a inicialização do sistema na Raspberry PI, levando em conta uma foto com 180kB de tamanho.

Tem-se como possibilidade para este a forma que foi implementado o algoritmo de transmissão de dados via bluetooth, baseando o mesmo em algoritmos comunicação TCP/IP, pela necessidade de que a transferência de arquivo entre smartphone e Raspberry PI fosse realizada de forma completamente autônoma, algo que os protocolos existentes não possibilitam. Uma vez que em seu funcionamento normal, o smartphone executa inúmeras requisições para a atualização de valores, houve momentos em que um dos sistemas do bluetooth, o Serviço de descoberta de protocolo, apresentava falhas e para resolver o mesmo, foi usado o método de reflexão de classe para a atualização de valores.

Ao final deste projeto, não foram implementado na sua totalidade dois sistemas, sendo eles: foi o sensor de efeito hall, que para propósito de testes o mesmo foi simulado através de um gerador de função MFG-4201A da Minipa, tendo como saída um trem de pulsos. O outro sistema foi o botão de pânico, este estando provisoriamente no aplicativo.

5 CONCLUSÃO

O presente trabalho apresentou o desenvolvimento de um protótipo taxímetro com captura de imagem e transmissão de localização via TCP/IP para um servidor web. Para a realização do mesmo, foram desenvolvidos os códigos-fonte para cada parte do sistema, utilizando linguagem Java para o servidor e aplicativo para sistema operacional Android e linguagem python para o firmware implementado na Raspberry PI. Uma vez prontos, os mesmos foram testados em bancada, primeiramente separados, para averiguar se funcionou da forma esperada para cada sistema e finalizando com a integração e testes do sistema no todo. Os objetivos definidos para esse projeto, tanto o geral quanto os específicos foram alcançados, uma vez que o sistema possui as funcionalidades definidas no início do projeto.

É possível uma gama de melhorias que podem ser aplicadas ao mesmo, entre eles estão a implementação do sensor real e do botão físico de pânico, a melhora da interface visual do sistema, a melhoria do sistema de servidor para um sistema multithreading, que neste momento, opera de forma singular. Como pontos principais para um trabalho futuro está a otimização do protocolo de comunicação, uma vez que o taxi é uma alternativa de transporte rápido, o sistema deve acompanhar sua rapidez e uma possível integração desse sistema com os aplicativos já existentes como o 99Taxis e EasyTaxi, unindo a funcionalidade do sistema desenvolvido por este projeto à todas facilidades que estes aplicativos trazem.

REFERENCIAS

ADETAX - ASSOCIAÇÃO DAS EMPRESAS DE TAXI DE FROTA DO MUNICIPIO DE SÃO PAULO. Estatísticas. Disponível em: http://www.adetax.com.br/index.php/informacoes-e-servicos/estatisticas/.Acesso em: 25 Mar. 2016.

BROWN, Peter Jensen. "A History of the Taximeter". Early Sports 'n' Pop-Culture History Blog. Disponível em: http://esnpc.blogspot.com.br/2016/05/taximeter-taximeter-uber-alles-history.html. Acesso em: 11 Fev. 2016.

LONDON VINTAGE TAXI ASSOCIATION. London Taxi History. Disponível em: http://www.lvta.co.uk/history.htm. Acesso em 17 Fev 2017.

TANENBAUM, A. S.; WETHERALL, D. **Redes de Computadores**. 5. ed. São Paulo: Pearson Education, 2011. p. 201 – 204.

SILVERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistema de Banco de Dados**. 3. ed. São Paulo: Makron Books, 1999. p. 109 – 149.

ELMASRI, R.; NAVATHE, S. B. **Sistema de Banco de Dados**. 4. ed. São Paulo: Pearson Education, 2005. p. 148 – 174.

RASPBERRY PI FOUNDATION. Frequently Asked Questions. Disponível em: https://www.raspberrypi.org/help/faqs/#introWhatIs. Acesso em: 20 Fev. 2016.

RASPBERRY PI FOUNDATION. Frequently Asked Questions. Disponível em: https://www.raspberrypi.org/help/faqs/#camera. Acesso em: 20 Fev. 2016.

ARM. Cortex-A7 Processor – ARM. Disponível em: https://www.arm.com/products/processors/cortex-a/cortex-a7.php. Acesso em: 18 Fev. 2016

APENDICE A – LOG RASPBERRY PI

pi@raspberrypiPECCE ~/final $ sudo python TaxiRpiServer.py

Aguardando ordem da IHM 1

Conexao recebida de: ('E6:24:46:66:28:32', 1)

received [foto]

sending [198359!]

teste delay

delay finalizado

Conexao finalizada

Aguardando ordem da IHM 1

Conexao recebida de: ('E6:24:46:66:28:32', 1)

received [envia]

Iniciando Envio...

Envio Finalizado!

sending [done!]

teste delay

delay finalizado

Conexao finalizada

Aguardando ordem da IHM 1

Conexao recebida de: ('E6:24:46:66:28:32', 1)

received [B1]

entrou na thread

sending [iniciada!]

teste delay

delay finalizado

Conexao finalizada

Aguardando ordem da IHM 1

Conexao recebida de: ('E6:24:46:66:28:32', 1)

received [req]

sending [3.0!]

teste delay

delay finalizado

Conexao finalizada

Aguardando ordem da IHM 1

carro parado

valor da corrida = 3.0

Conexao recebida de: ('E6:24:46:66:28:32', 1)

received [req]

sending [3.0!]

teste delay

delay finalizado

Conexao finalizada

Aguardando ordem da IHM 1

carro parado

valor da corrida = 3.0

Conexao recebida de: ('E6:24:46:66:28:32', 1)

received [req]

sending [3.0!]

teste delay

delay finalizado

Conexao finalizada

Aguardando ordem da IHM 1

carro parado

valor da corrida = 3.0

carro parado

valor da corrida = 3.4

Conexao recebida de: ('E6:24:46:66:28:32', 1)

received [req]

sending [3.4!]

teste delay

delay finalizado

Conexao finalizada

Aguardando ordem da IHM 1

carro parado

valor da corrida = 3.4

Conexao recebida de: ('E6:24:46:66:28:32', 1)

received [req]

sending [3.4!]

teste delay

delay finalizado

Conexao finalizada

Aguardando ordem da IHM 1

carro parado

valor da corrida = 3.4

valor da corrida = 3.6

valor da corrida = 3.8

Conexao recebida de: ('E6:24:46:66:28:32', 1)

received [req]

sending [3.8!]

teste delay

delay finalizado

Conexao finalizada

Aguardando ordem da IHM 1

valor da corrida = 4.0

valor da corrida = 4.2

valor da corrida = 4.4

valor da corrida = 4.6

Conexao recebida de: ('E6:24:46:66:28:32', 1)

received [req]

sending [4.6!]

teste delay

delay finalizado

Conexao finalizada

Aguardando ordem da IHM 1

valor da corrida = 4.8

valor da corrida = 5.0

valor da corrida = 5.2

valor da corrida = 5.4

Conexao recebida de: ('E6:24:46:66:28:32', 1)

received [req]

sending [5.4!]

teste delay

delay finalizado

Conexao finalizada

Aguardando ordem da IHM 1

valor da corrida = 5.6

valor da corrida = 5.8

valor da corrida = 6.0

valor da corrida = 6.2

Conexao recebida de: ('E6:24:46:66:28:32', 1)

received [req]

sending [6.2!]

teste delay

delay finalizado

Conexao finalizada

Aguardando ordem da IHM 1

valor da corrida = 6.4

valor da corrida = 6.6

Conexao recebida de: ('E6:24:46:66:28:32', 1)

received [finaliza]

sending [6.6!]

teste delay

corrida finalizada

delay finalizado

Conexao finalizada

Aguardando ordem da IHM 1

APENDICE B – LOG SERVIDOR

Corrida normal:

run:

**********Server Program**********

Server Socket bounded to Port : 9998

Client Socket Connected : /192.168.0.198

TAXISTA02

Server Socket bounded to Port : 9999

Client Socket Connected : /192.168.0.198

NULL

Server Socket bounded to Port : 9998

Client Socket Connected : /192.168.0.198

jpg

testeSQL

testeSQL

INSERT into TAXISTA02 VALUES (6, '16.11.24', '18.51.58', 'ind', 'ind', 'ind','/home/samir/teste_server/2016.11.24.18.51.58_foto.jpg', 'ind')

Server Socket bounded to Port : 9999

Client Socket Connected : /192.168.0.198

jpg

Iniciando Recebimento...

No. of Bytes Received : -1

File Received...

Server Socket bounded to Port : 9998

Client Socket Connected : /192.168.0.198

TAXISTA02

Server Socket bounded to Port : 9999

Client Socket Connected : /192.168.0.198

6.6

testeSQL

UPDATE TAXISTA02 SET hora_final= '18.55.01',valor_final='6.6' WHERE ID=6

Server Socket bounded to Port : 9998

corrida(panico):

**********Server Program**********

Server Socket bounded to Port : 9998

Client Socket Connected : /192.168.0.198

TAXISTA02

Server Socket bounded to Port : 9999

Client Socket Connected : /192.168.0.198

NULL

Server Socket bounded to Port : 9998

Client Socket Connected : /192.168.0.198

jpg

testeSQL

testeSQL

INSERT into TAXISTA02 VALUES (7, '16.11.24', '19.02.41', 'ind', 'ind', 'ind','/home/samir/teste_server/2016.11.24.19.02.41_foto.jpg', 'ind')

Server Socket bounded to Port : 9999

Client Socket Connected : /192.168.0.198

jpg

Iniciando Recebimento...

No. of Bytes Received : -1

File Received...

Server Socket bounded to Port : 9998

Client Socket Connected : /192.168.0.198

txt

testeSQL

UPDATE TAXISTA02 SET problema= 'P', caminho_problemas= '/home/samir/teste_server/2016.11.24.19.03.43_gps.txt' WHERE ID=7

Server Socket bounded to Port : 9999

Client Socket Connected : /192.168.0.198

txt

Iniciando Recebimento...

No. of Bytes Received : -1

File Received...

Server Socket bounded to Port : 9998

Client Socket Connected : /192.168.0.198

txt

testeSQL

UPDATE TAXISTA02 SET problema= 'P', caminho_problemas= '/home/samir/teste_server/2016.11.24.19.04.03_gps.txt' WHERE ID=7

Server Socket bounded to Port : 9999

Client Socket Connected : /192.168.0.198

txt

Iniciando Recebimento...

No. of Bytes Received : -1

File Received...

Server Socket bounded to Port : 9998

Client Socket Connected : /192.168.0.198

txt

testeSQL

UPDATE TAXISTA02 SET problema= 'P', caminho_problemas= '/home/samir/teste_server/2016.11.24.19.04.23_gps.txt' WHERE ID=7

Server Socket bounded to Port : 9999

Client Socket Connected : /192.168.0.198

txt

Iniciando Recebimento...

No. of Bytes Received : -1

File Received...

Server Socket bounded to Port : 9998

Client Socket Connected : /192.168.0.198

txt

testeSQL

UPDATE TAXISTA02 SET problema= 'P', caminho_problemas= '/home/samir/teste_server/2016.11.24.19.04.43_gps.txt' WHERE ID=7

Server Socket bounded to Port : 9999

Client Socket Connected : /192.168.0.198

txt

Iniciando Recebimento...

No. of Bytes Received : -1

File Received...

Server Socket bounded to Port : 9998

Client Socket Connected : /192.168.0.198

TAXISTA02

Server Socket bounded to Port : 9999

Client Socket Connected : /192.168.0.198

6.6

testeSQL

UPDATE TAXISTA02 SET hora_final= '19.04.50',valor_final='6.6' WHERE ID=7

Server Socket bounded to Port : 9998

Client Socket Connected : /192.168.0.198

txt

testeSQL

UPDATE TAXISTA02 SET problema= 'P', caminho_problemas= '/home/samir/teste_server/2016.11.24.19.05.03_gps.txt' WHERE ID=7

Server Socket bounded to Port : 9999

Client Socket Connected : /192.168.0.198

txt

Iniciando Recebimento...

No. of Bytes Received : -1

File Received...

Server Socket bounded to Port : 9998

Client Socket Connected : /192.168.0.198

txt

testeSQL

UPDATE TAXISTA02 SET problema= 'P', caminho_problemas= '/home/samir/teste_server/2016.11.24.19.05.23_gps.txt' WHERE ID=7

Server Socket bounded to Port : 9999

Client Socket Connected : /192.168.0.198

txt

Iniciando Recebimento...

No. of Bytes Received : -1

File Received...

Server Socket bounded to Port : 9998

corrida problemas:

run:

**********Server Program**********

Server Socket bounded to Port : 9998

Client Socket Connected : /192.168.0.198

TAXISTA02

Server Socket bounded to Port : 9999

Client Socket Connected : /192.168.0.198

NULL

Server Socket bounded to Port : 9998

Client Socket Connected : /192.168.0.198

jpg

testeSQL

testeSQL

INSERT into TAXISTA02 VALUES (8, '16.11.24', '19.10.15', 'ind', 'ind', 'ind','/home/samir/teste_server/2016.11.24.19.10.15_foto.jpg', 'ind')

Server Socket bounded to Port : 9999

Client Socket Connected : /192.168.0.198

jpg

Iniciando Recebimento...

No. of Bytes Received : -1

File Received...

Server Socket bounded to Port : 9998

Client Socket Connected : /192.168.0.198

tro

testeSQL

UPDATE TAXISTA02 SET problema= 'T', caminho_problemas= '/home/samir/teste_server/2016.11.24.19.10.44_tro.txt' WHERE ID=8

Server Socket bounded to Port : 9999

Client Socket Connected : /192.168.0.198

tro

Iniciando Recebimento...

No. of Bytes Received : -1

File Received...

Server Socket bounded to Port : 9998

Client Socket Connected : /192.168.0.198

TAXISTA02

Server Socket bounded to Port : 9999

Client Socket Connected : /192.168.0.198

8.4

testeSQL

UPDATE TAXISTA02 SET hora_final= '19.11.29',valor_final='8.4' WHERE ID=8

Server Socket bounded to Port : 9998

APENDICE C – BANCO DE DADOS SQL

#	ID	data_corrida	hora_inicio	hora_final	valor_final	problema	caminho_foto	caminho_problemas
		YY.MM.DD	HH.MM.SS	HH.MM.SS	RRRR.CC	PTN	CAMINHO_SO	CAMINHO_SO
1	0	16.07.26	17.51.46	17.52.03	0.00	ind	/home/samir/teste_server/2016.07.26.17.51.46_foto.jpg	ind
2	1	16.11.24	17.06.08	17.13.38	110.6	ind	/home/samir/teste_server/2016.11.24.17.06.08_foto.jpg	ind
3	2	16.11.24	17.17.09	17.20.34	0	ind	/home/samir/teste_server/2016.11.24.17.17.09_foto.jpg	ind
4	3	16.11.24	17.23.12	17.27.13	18.4	ind	/home/samir/teste_server/2016.11.24.17.23.12_foto.jpg	ind
5	4	16.11.24	18.43.53	18.47.28	6.6	ind	/home/samir/teste_server/2016.11.24.18.43.53_foto.jpg	ind
6	5	16.11.24	18.51.58	18.55.01	6.6	ind	/home/samir/teste_server/2016.11.24.18.51.58_foto.jpg	ind
7	6	16.11.24	19.02.41	19.04.50	6.6	P	/home/samir/teste_server/2016.11.24.19.02.41_foto.jpg	/home/samir/teste_server/2016.11.24.19.05.23_gps.txt

#	ID	data_corrida	hora_inicio	hora_final	valor_final	problema	caminho_foto	caminho_problemas
		YY.MM.DD	HH.MM.SS	HH.MM.SS	RRRR.CC	PTN	CAMINHO_SO	CAMINHO_SO
1	0	16.07.26	17.51.46	17.52.03	0.00	ind	/home/samir/teste_server/2016.07.26.17.51.46_foto.jpg	ind
2	1	16.11.24	17.06.08	17.13.38	110.6	ind	/home/samir/teste_server/2016.11.24.17.06.08_foto.jpg	ind
3	2	16.11.24	17.17.09	17.20.34	0	ind	/home/samir/teste_server/2016.11.24.17.17.09_foto.jpg	ind
4	3	16.11.24	17.23.12	17.27.13	18.4	ind	/home/samir/teste_server/2016.11.24.17.23.12_foto.jpg	ind
5	4	16.11.24	18.43.53	18.47.28	6.6	ind	/home/samir/teste_server/2016.11.24.18.43.53_foto.jpg	ind
6	5	16.11.24	18.51.58	18.55.01	6.6	ind	/home/samir/teste_server/2016.11.24.18.51.58_foto.jpg	ind
7	6	16.11.24	19.02.41	19.04.50	6.6	P	/home/samir/teste_server/2016.11.24.19.02.41_foto.jpg	/home/samir/teste_server/2016.11.24.19.05.23_gps.txt
8	7	16.11.24	19.04.50	19.11.29	8.4	T	/home/samir/teste_server/2016.11.24.19.10.15_foto.jpg	/home/samir/teste_server/2016.11.24.19.10.44_tro.txt

#	ID	data_corrida	hora_inicio	hora_final	valor_final	problema	caminho_foto	caminho_problemas
		YY.MM.DD	HH.MM.SS	HH.MM.SS	RRRR.CC	PTN	CAMINHO_SO	CAMINHO_SO
1	0	16.07.26	17.51.46	17.52.03	0.00	ind	/home/samir/teste_server/2016.07.26.17.51.46_foto.jpg	ind
2	1	16.11.24	17.06.08	17.13.38	110.6	ind	/home/samir/teste_server/2016.11.24.17.06.08_foto.jpg	ind
3	2	16.11.24	17.17.09	17.20.34	0	ind	/home/samir/teste_server/2016.11.24.17.17.09_foto.jpg	ind
4	3	16.11.24	17.23.12	17.27.13	18.4	ind	/home/samir/teste_server/2016.11.24.17.23.12_foto.jpg	ind
5	4	16.11.24	18.43.53	18.47.28	6.6	ind	/home/samir/teste_server/2016.11.24.18.43.53_foto.jpg	ind
6	5	16.11.24	18.51.58	18.55.01	6.6	ind	/home/samir/teste_server/2016.11.24.18.51.58_foto.jpg	ind
7	6	16.11.24	19.02.41	19.04.50	6.6	P	/home/samir/teste_server/2016.11.24.19.02.41_foto.jpg	/home/samir/teste_server/2016.11.24.19.05.23_gps.txt
8	7	16.11.24	19.04.50	19.11.29	8.4	T	/home/samir/teste_server/2016.11.24.19.10.15_foto.jpg	/home/samir/teste_server/2016.11.24.19.10.44_tro.txt
9	8	16.11.24	19.10.15	19.11.29	8.4	T	/home/samir/teste_server/2016.11.24.19.10.15_foto.jpg	/home/samir/teste_server/2016.11.24.19.10.44_tro.txt

A esquerda se encontra as imagens do banco de dados utilizado, seguindo a ordem da esquerda para a direita. O primeiro mostra registros de corridas sem ocorrências, o central mostra uma corrida com registro de pânico e o ultimo mostra, além do registro de pânico, o registro de problemas na corrida seguinte.