Carlos Eduardo Bartz

DESENVOLVIMENTO DE UMA INTERFACE MÓVEL DE CONTROLE E MONITORAMENTO DE REAÇÕES QUÍMICAS BASEADAS EM TITULAÇÃO POR TERMOMETRIA

Trabalho de conclusão apresentado ao Curso de Ciência da Computação da Universidade de Santa Cruz do Sul – UNISC para a obtenção do título de bacharel em Ciência da Computação.

Orientador: Prof. Msc. Gilson Augusto Helfer

Santa Cruz do Sul

2017

Carlos Eduardo Bartz

DESENVOLVIMENTO DE UMA INTERFACE MÓVEL DE CONTROLE E MONITORAMENTO DE REAÇÕES QUÍMICAS BASEADAS EM TITULAÇÃO POR TERMOMETRIA

Este trabalho de conclusão foi submetido ao Curso de Ciência da Computação da Universidade de Santa Cruz do Sul – UNISC, como requisito parcial para a obtenção do título de bacharel em Ciência da Computação.

Prof. Msc. Gilson Augusto Helfer
Orientador

Prof. Dr. Adilson Ben da Costa
Avaliador

Profa. Msc. Daniela Saccol Peranconi
Avaliadora

Santa Cruz do Sul
2017

RESUMO

Este trabalho tem o propósito de desenvolver uma interface gráfica de controle e monitoramento para um sistema de titulação termométrica utilizando *smartphone*. Para tanto, foi utilizado um microcomputador *Raspberry Pi* para realizar a leitura de um pirômetro infravermelho e a transmissão, através de uma conexão *Bluetooth*, dos dados de temperatura obtidos para um dispositivo *Android*. Neste dispositivo, o aplicativo desenvolvido foi utilizado para receber os valores de temperatura em tempo real e exibi-los em um gráfico de linha. Conforme são recebidos, os valores são também armazenados em um banco de dados para que possam ser visualizados posteriormente e mesmo compartilhados. Ao fim do desenvolvimento foi possível afirmar que a utilização de um equipamento remoto para monitoramento da variação da temperatura ao longo do tempo é viável e pode ser obtida a um custo baixo se comparado à utilização de equipamentos como câmeras térmicas de vídeo.

Palavras-chave: Entalpimetria, Infravermelho, *Android*.

ABSTRACT

This paper has the purpose of developing a control and monitoring graphical interface for a thermometric titration system by use of a smartphone. For such, a Raspberry Pi microcomputer was utilized to perform the reading of an infrared pyrometer and the transmission, through a Bluetooth connection, of the obtained temperature data to an Android device. In this device, the developed application was utilized to receive the temperature values in real time and to show them on a line chart. As they are received, the values are also registered in a database so they can be later visualized and even shared. At the end of the development it was possible to assert that the use of a remote equipment to monitor temperature variation over time is viable and can be obtained at a low cost if compared to utilizing equipment such as infrared video cameras.

Keywords: Enthalpymetry, Infrared, Android.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

No campo da química analítica, a titulação é uma técnica largamente empregada em laboratórios de controle de qualidade, bem como em aulas experimentais de química (COSTA *et al.*, 2015). A titulação corresponde à adição de um reagente com valor de concentração conhecido em uma substância com valor de concentração desconhecido até que se julgue completa a reação, e é realizada para medir os volumes das soluções envolvidas na reação química (DIAS *et al.*, 2016). Mais especificamente, a titulação termométrica é um método volumétrico no qual o efeito de calor da reação de titulação é usado para medir a concentração de uma amostra (BARTHEL e WACHTER, 1975).

A mudança na entalpia da reação química em curso causa uma alteração na temperatura que, quando plotada contra o volume do reagente, pode ser usada para determinar o ponto final da reação (STAHL, 1994).

A utilização de câmeras infravermelhas como sensores de temperatura para realização de procedimentos de titulação termométrica já foi sugerida, citando vantagens como a eliminação do contato físico entre o sensor e a amostra e a utilização de amostras de volume reduzido (COSTA *et al.*, 2015). Além disso, o reduzido tempo de análise permite o processamento de um elevado número de amostras em comparação a métodos tradicionais (BARIN *et al.*, 2015).

Apesar das vantagens, a utilização de câmeras infravermelhas pode ser desencorajada em função do alto custo do equipamento. Com isso em vista, o presente trabalho propõe uma solução mais acessível para a realização de análises de titulação termométrica, através da implementação de um sistema que combina um computador portátil *Raspberry Pi*, um sensor de temperatura por infravermelho e uma aplicação para dispositivos móveis Android, que permite o controle e monitoramento da medição da temperatura de reações químicas em tempo real.

1.1 Motivação

Procedimentos de titulação potenciométricos são invasivos, pois necessitam de um sensor em contato com a amostra sendo avaliada. Em função disso, a relação entre o volume da amostra e o tamanho do sensor tem influência sobre a qualidade dos resultados. Além disso, a necessidade de limpeza do equipamento torna o processo moroso. Com isso em vista, o monitoramento da temperatura através da medição da emissão de radiação infravermelha pode ser uma alternativa importante para o desenvolvimento de um procedimento não invasivo que possa ser utilizado em amostras de volume reduzido (COSTA *et al.*, 2015). Além de ser um

método sem contato, a medição de temperatura por infravermelho é um método não destrutivo, que pode ser realizado à temperatura ambiente e sem fontes de iluminação adicionais (BARIN *et al.*, 2015).

Outro fator de motivação para a presente proposta, é o alto custo de câmeras infravermelhas quando comparadas ao custo dos equipamentos que são utilizados no sistema proposto neste trabalho. Por exemplo, a câmera da marca FLIR, modelo A35 utilizada no trabalho de (COSTA *et al.*, 2015) pode ser encontrada no *website* do representante a um valor de US$ 4.995,00 (OEM Cameras, 2016). Já o modelo E60, do mesmo fabricante, usado na pesquisa de (BARIN *et al.*, 2015) pode ser encontrado na loja do fabricante por US$ 4.999,00 (Flir Store, 2013). Enquanto isso, o microcomputador *Raspberry Pi* 3 Modelo B SBC que foi utilizado nesta pesquisa é encontrado na página do representante pelo valor de £ 32.99 (RS Components, 2016) e o sensor infravermelho da marca *Melexis* modelo MLX90614 pelo valor de US$ 14,08 (Digi-Key Electronics, 2016). Isto combinado com a utilização de um aplicativo de *smartphone* para controle e monitoramento da temperatura da reação química implica em um sistema bastante acessível financeiramente.

1.2 Objetivos

O objetivo principal deste trabalho foi desenvolver uma interface gráfica de controle e monitoramento para um sistema de titulação termométrica utilizando *smartphone*. O sistema móvel de análises térmicas consistiu no uso de sensores de temperatura por infravermelho conectados a um computador portátil (*Raspberry Pi*).

Sendo assim, atingiu-se os seguintes objetivos específicos:

• Estudar diferentes tipos de comunicação M2M (*Machine-To-Machine*) mais apropriados para conexão entre dispositivos;

• Implementar métodos para controlar remotamente o processo de leitura dos dados;

• Implementar gráficos em *real-time* para monitoramento dos dados obtidos pelo sensor via *smartphone*;

• Desenvolver meios para exportação de dados.

1.3 Organização do trabalho

O presente trabalho está organizado em cinco seções, sendo que a primeira fez uma introdução ao projeto, apresentando as motivações e objetivos que se desejava alcançar. Em seguida foram abordados os fundamentos e conceitos relativos ao trabalho desenvolvido, com

uma breve apresentação dos tópicos titulação e titulação termométrica, tecnologias sem fio para conexão dos dispositivos utilizados e desenvolvimento de aplicativos para dispositivos *Android*. Após isso, foram abordados alguns trabalhos relacionados ao presente projeto. A seguir a solução desenvolvida foi exposta, com a descrição dos dispositivos utilizados bem como dos *scripts* e do aplicativo utilizados para a realização dos objetivos. Por fim foram apresentadas as considerações finais relativas ao desenvolvimento do trabalho e possibilidades de trabalhos futuros.

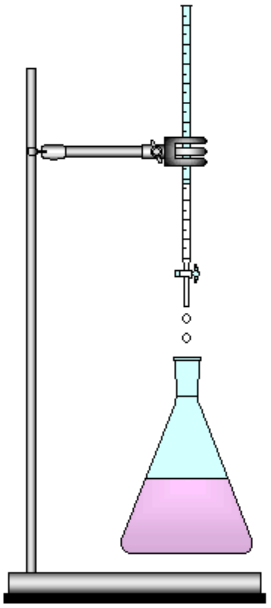
2 REVISÃO BIBLIOGRÁFICA

Este capítulo apresenta fundamentos e conceitos relacionados ao trabalho desenvolvido. Primeiramente, é feita uma descrição superficial dos tópicos titulação e titulação termométrica para contextualizar o assunto com o presente trabalho. Depois são abordados os tipos de conexões sem-fio possíveis para os dispositivos utilizados neste projeto. Em seguida, são descritos conceitos relacionados ao desenvolvimento de aplicativos para a plataforma *Android*.

2.1 Titulação

Titulação ou análise volumétrica consiste em adicionar um reagente de concentração conhecida em uma amostra de concentração desconhecida e, após o término da reação, medir o volume final da solução para então calcular a concentração que era desconhecida. Algumas substâncias químicas mudam de cor quando reagem com outras, fornecendo um sinal claro do ponto final da reação, outras no entanto não apresentam mudança perceptível. Nestes casos é necessária a adição de um indicador, que nada mais é do que um reagente auxiliar que muda de cor quando a reação termina (CONSTANTINO, DA SILVA e DONATE, 2004). A Figura 1 ilustra a configuração de um experimento de titulação.

Figura 1 – Configuração comum do processo de titulação

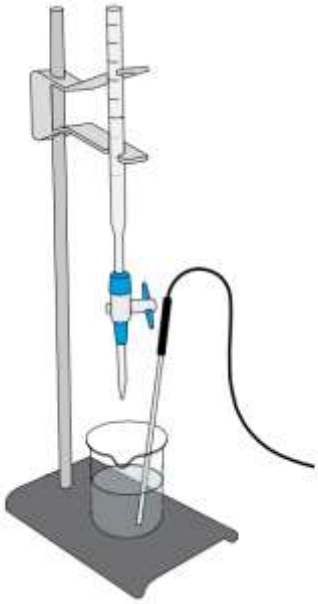


Fonte: Disponível em < http://www2.fc.unesp.br/lvq/exp1202.gif >, acesso em 18/06/2017.

Reações químicas são acompanhadas de mudanças de energia, que podem se manifestar como calor de transformação. Esta variação de energia é chamada de entalpia da transformação. Uma reação é chamada exotérmica quando a temperatura da mistura aumenta durante o

processo, e endotérmica quando a temperatura diminui durante a reação (CONSTANTINO, DA SILVA e DONATE, 2004). Esta variação na temperatura, quando plotada contra o volume do reagente pode ser utilizada para encontrar o ponto final da titulação (IUPAC, 1997). A Figura 2 apresenta a configuração de um experimento de titulação termométrica.

Figura 2 – Configuração do equipamento de titulação termométrica



Fonte: <http://www.data-harvest.co.uk/worksheets/student/40_Thermometric_Titration.pdf>

2.2 Conexões machine to machine

Conexões *machine to machine* (M2M) geralmente referem-se a tecnologias de informação e comunicação capazes de medir, entregar, digerir, e reagir à informação de forma autônoma, ou seja, sem ou com mínima interação humana durante as fases de implantação, configuração, operação e manutenção (ANTON-HARO e DOHLER, 2014). Outra definição afirma que M2M se refere àquelas soluções que permitem comunicação entre dispositivos do mesmo tipo e uma aplicação específica, tudo através de redes cabeadas ou sem-fio (HOLLER *et al.*, 2014). Afirma-se ainda que M2M usa um dispositivo (tal como um sensor ou medidor) para capturar um evento (tal como temperatura e nível de estoque), que é transmitido por uma rede (sem-fio, cabeada ou híbrida) para uma aplicação (*software*) que traduz o evento capturado em informação significativa (DOHLER, WATTEYNE e ALONZO-ZARATE, 2010).

Os dispositivos utilizados para a realização deste projeto são descritos mais detalhadamente no capítulo 4, mas para efeito de fundamentação teórica, são descritas a seguir as tecnologias de comunicação sem-fio por eles disponibilizadas.

2.2.1 Wi-fi

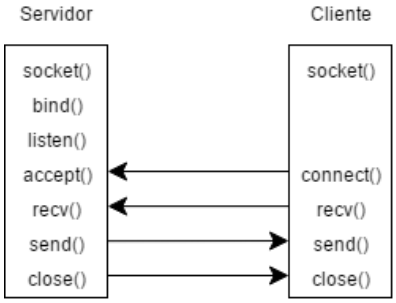
O termo *Wi-fi* descreve redes locais sem fio (WLAN). Este tipo de conexão permite a comunicação de dados a velocidades de banda-larga entre dispositivos conectados por um ponto de acesso ou em modo *ad hoc* (LEE, SU e SHEN, 2007).

Dispositivos *Wi-fi* se conectam utilizando ondas de rádio e o ponto de acesso é o dispositivo que cria a rede sem fio e dá acesso aos demais dispositivos (UNDERDAHL, 2011). Já no modo *ad hoc*, a rede é formada pela cooperação de nós independentes, sem que existam papéis definidos para cada dispositivo. Cada um dos nós da rede toma suas decisões baseado na situação atual da rede (BASAGNI *et al.*, 2004).

2.2.2 Bluetooth

De forma simples, Huang e Larry (2007) definem *Bluetooth* como uma forma para dispositivos se comunicarem de modo sem-fio através de curtas distâncias. O protocolo RFCOMM é responsável por emular uma porta serial RS-232 para transferência de sinais de controle e dados (PRABHU e REDDI, 2004). A Figura 3 apresenta os passos do processo de comunicação através de *sockets Bluetooth*.

Figura 3 – Comunicação com sockets Bluetooth



Fonte: Adaptado de <https://jan.newmarch.name/ClientServer/socket/udp_seq.gif>

2.2.3 Comparação

A Tabela 1 compara algumas das características básicas apresentadas por ambas as tecnologias de comunicação sem fio.

Tabela 1 – Comparação tecnologias wireless

Padrão	*Bluetooth*	*Wi-fi*
Especificação IEEE	802.15.1	802.11a/b/g
Faixa de frequência	2.4 GHz	2.4 GHz; 5 GHz
Taxa máxima de sinal	1 Mb/s	54 Mb/s
Alcance nominal	10 m	100 m

Fonte: Adaptado de (LEE, SU e SHEN, 2007)

2.3 Android

O sistema *Android* fornece uma estrutura que permite criar aplicativos para dispositivos móveis em um ambiente de linguagem *Java*. Aplicativos para *Android* são criados como uma combinação de componentes distintos que podem ser invocados individualmente, ou ainda, iniciar um componente em outro aplicativo. Esse modelo fornece vários pontos de entrada para um único aplicativo e permite que qualquer aplicativo se comporte como o "padrão" de um usuário para uma ação que outros aplicativos podem invocar. Esta estrutura permite ainda fornecer recursos exclusivos para diferentes configurações de dispositivos. É possível criar diferentes arquivos de *layout* para diversos tamanhos de tela e o sistema determina qual *layout* deverá aplicar com base no tamanho da tela do dispositivo atual (GOOGLE E OPEN HANDSET ALLIANCE, 2015a). A seguir serão descritos brevemente os principais conceitos relacionados ao desenvolvimento de uma aplicação para o sistema *Android* pertinentes à realização deste trabalho.

2.3.1 Arquivo de manifesto

O arquivo de manifesto é um item obrigatório em um aplicativo *Android*. Este arquivo deve estar no diretório raiz do aplicativo e contém informações essenciais ao sistema *Android* a respeito do aplicativo. Neste arquivo devem ser descritos, por exemplo, os componentes do aplicativo, filtros de ações (*intents)*, permissões que o aplicativo necessita, entre outros (GOOGLE E OPEN HANDSET ALLIANCE, 2009).

2.3.2 Intents

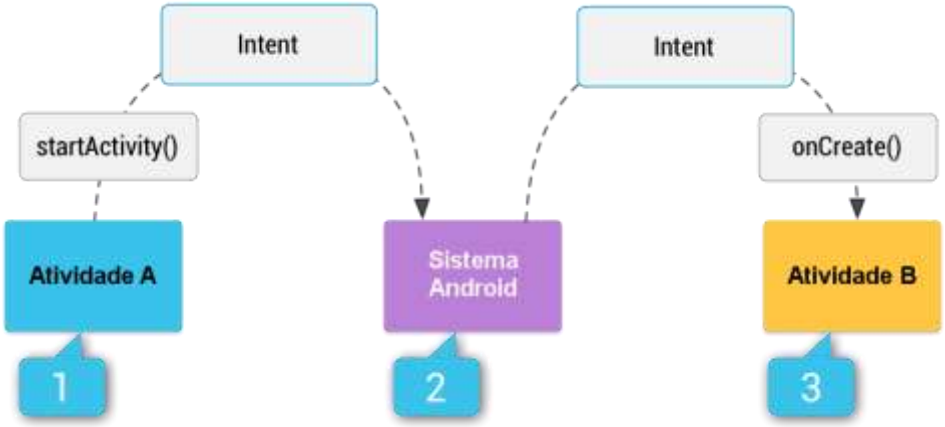
Intents são objetos de mensagem que podem ser usados para solicitar uma ação de outro componente do aplicativo ou até de um componente em outra aplicação. Existem dois tipos de *intents*: explícitos e implícitos.

Os *intents* explícitos especificam o componente a iniciar pelo nome de classe. Um *intent* explícito é geralmente usado para iniciar um componente no próprio aplicativo, pois o nome de classe da atividade ou serviço que se deseja iniciar é conhecido.

Os *intents* implícitos declaram uma ação geral a realizar em vez de nomear um componente específico, o que permite que um componente de outro aplicativo processe essa ação. A criação de uma atividade com a utilização de um *intent* implícito é demonstrada na Figura 4. No diagrama, a atividade "A" solicita a criação de uma atividade, sem determinar a classe específica, mas uma descrição da atividade que deseja iniciar. A seguir, o sistema

Android busca nos aplicativos instalados um filtro de *intent* que corresponda à ação desejada, e, caso encontre, inicia a atividade repassando o *intent*.

Figura 4 – Criação de intent implícito



Fonte: Adaptado de <https://developer.android.com/images/components/intent-filters@2x.png>

Para informar ao sistema *Android* qual o tipo de ação que seu aplicativo pode realizar, o desenvolvedor precisa associar filtros de *intents* ao mesmo. Tais filtros são expressões no arquivo de manifesto, associadas à descrição do componente, que determinam o tipo de *intent* que o componente pode receber. Estes filtros permitem que o componente em questão seja criado mesmo a partir de outra aplicação, sem que se saiba o nome específico da sua classe (GOOGLE E OPEN HANDSET ALLIANCE, 2013).

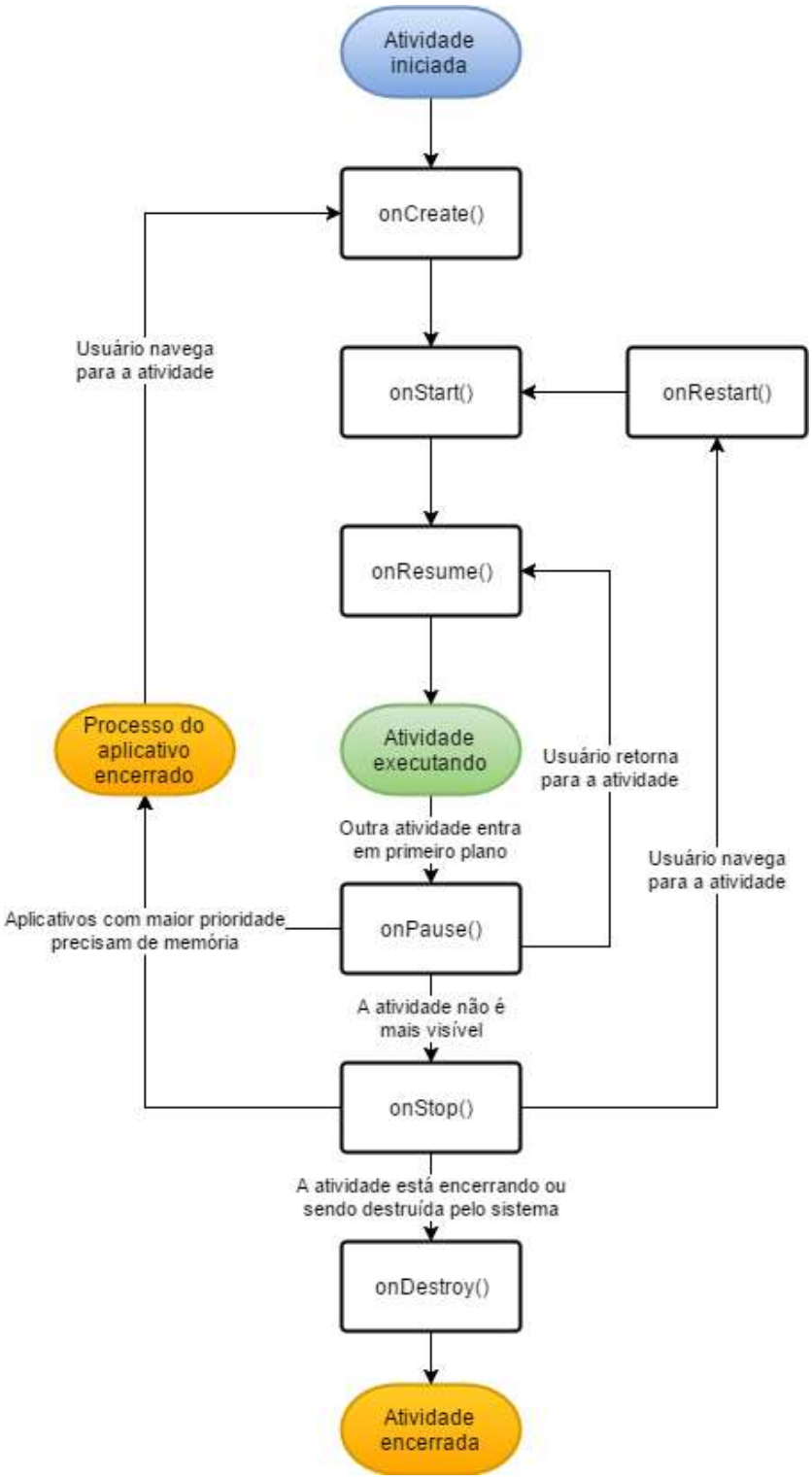
2.3.3 Componentes de aplicativo

Um aplicativo *Android* pode ser construído com quatro tipos de componentes: atividades, serviços, provedores de conteúdo e receptores de transmissão. Cada um deles tem uma finalidade distinta e um ciclo de vida específico que define como o componente é criado e destruído. A criação destes componentes ocorre indiretamente, com uma solicitação ao sistema *Android* através de um *Intent* (GOOGLE E OPEN HANDSET ALLIANCE, 2015b).

Atividades são componentes de aplicativo que fornecem uma tela com a qual os usuários podem interagir para fazer algo. Cada atividade recebe uma janela que mostra a interface do usuário. Geralmente, a janela preenche a tela, mas pode ser menor que a tela e flutuar sobre outras janelas (GOOGLE E OPEN HANDSET ALLIANCE, 2015c). O ciclo de vida de uma atividade é gerenciado pelo sistema *Android*, conforme a navegação do usuário pelo aplicativo. Este ciclo está descrito na Figura 5, onde se pode observar de forma simplificada os métodos que são invocados pelo sistema *Android* em resposta à mudança de estado da atividade. Estes métodos podem ser implementados para determinar o comportamento do aplicativo em cada

mudança de estado. Por exemplo, é possível salvar o estado de uma atividade antes de encerrá-la para que, quando o usuário reabra o aplicativo, seja possível recuperar este estado e permitir ao usuário que continue utilizando o aplicativo do mesmo ponto em que parou (GOOGLE E OPEN HANDSET ALLIANCE, 2016a).

Figura 5 – Ciclo de vida de uma atividade



Fonte: Adaptado de <https://developer.android.com/images/activity_lifecycle.png?hl=pt-BR>

Um serviço é um componente de aplicativo que é executado em segundo plano, sem fornecer uma interface para o usuário. Geralmente é utilizado para executar operações longas como por exemplo transações de rede (GOOGLE E OPEN HANDSET ALLIANCE, 2011).

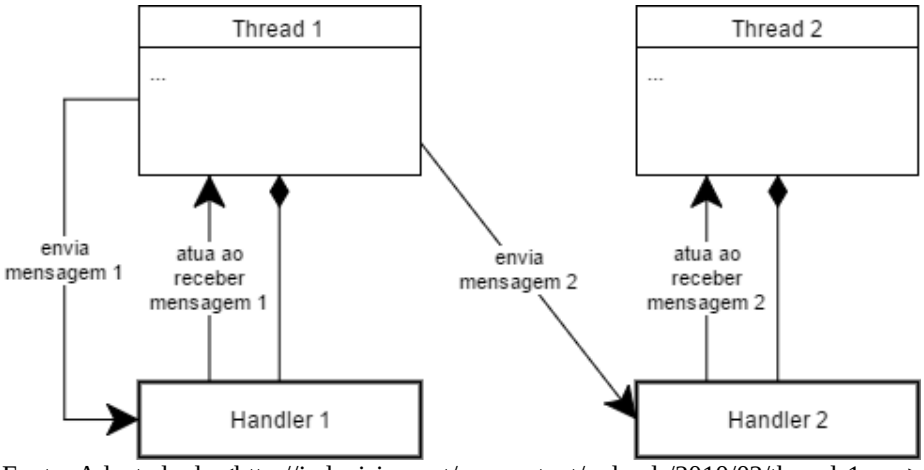
Os provedores de conteúdo são componentes que fazem o gerenciamento do acesso a um conjunto estruturado de dados, encapsulando estes dados e fornecendo mecanismos para definir a sua segurança. Provedores de conteúdo são a interface padrão que conecta dados em um processo com código em execução em outro processo (GOOGLE E OPEN HANDSET ALLIANCE, 2016b). Com a utilização de provedores de conteúdo é possível oferecer dados ou arquivos complexos a outros aplicativos, permitir a cópia de dados complexos do seu aplicativo para outros aplicativos e até fornecer sugestões de pesquisa personalizada usando a estrutura de pesquisa (GOOGLE E OPEN HANDSET ALLIANCE, 2012a).

Os componentes chamados de receptores de transmissão permitem que um aplicativo receba notificações do sistema ou de outro aplicativo quando da ocorrência de determinado evento. Por exemplo, um receptor de transmissão pode ser registrado para ser avisado de que uma tela foi desligada, a bateria está baixa ou uma tela foi capturada. Podem ser registrados estaticamente no arquivo de manifesto do aplicativo bem como dinamicamente em uma atividade (GOOGLE E OPEN HANDSET ALLIANCE, 2015b).

2.3.4 Handler

Handlers são parte da estrutura de gerenciamento de *threads* do sistema *Android*. Conforme ilustra a Figura 6, um objeto *handler* é associado à uma *thread* e é capaz de receber mensagens e executar código para manipular estas mensagens (GOOGLE E OPEN HANDSET ALLIANCE, 2012b).

Figura 6 – Funcionamento de handlers



Fonte: Adaptado de <http://indyvision.net/wp-content/uploads/2010/02/threads1.png>

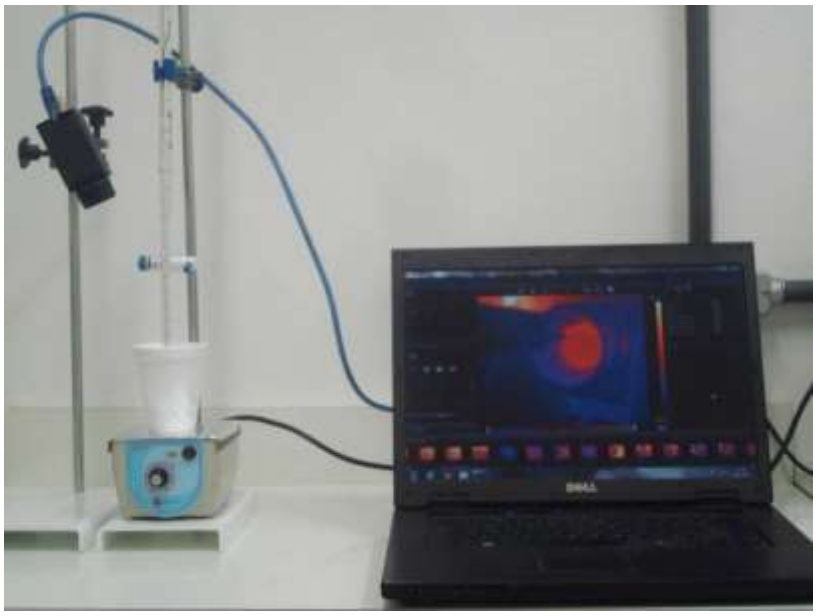
3 TRABALHOS RELACIONADOS

Este capítulo descreve trabalhos que serviram de inspiração para o desenvolvimento deste projeto. Dois deles utilizam câmeras infravermelhas para monitoramento de titulação termométrica, um utiliza um sistema de baixo custo para gerar imagens térmicas e o outro utiliza um sensor infravermelho para realizar análises entalpimétricas.

3.1 Aplicação da termografia por infravermelho para titulações termométricas

Neste trabalho, é demonstrado o potencial do emprego de termografia por infravermelho para o monitoramento de reações químicas. Para isso, foi utilizada uma câmera térmica para registrar em vídeo a alteração de temperatura causada pela adição de um reagente a uma amostra contida em um frasco posicionado sobre um agitador magnético, conforme ilustra a Figura 7. Os autores afirmam que os resultados de temperatura apresentados com a utilização desta metodologia foram semelhantes ao procedimento de referência e àqueles previstos na literatura, e que além disso o método utilizado produz um registro em vídeo do procedimento que pode ser utilizado para revisão posterior (COSTA *et al.*, 2015).

Figura 7 – Utilização de câmera térmica para monitoramento de reação química



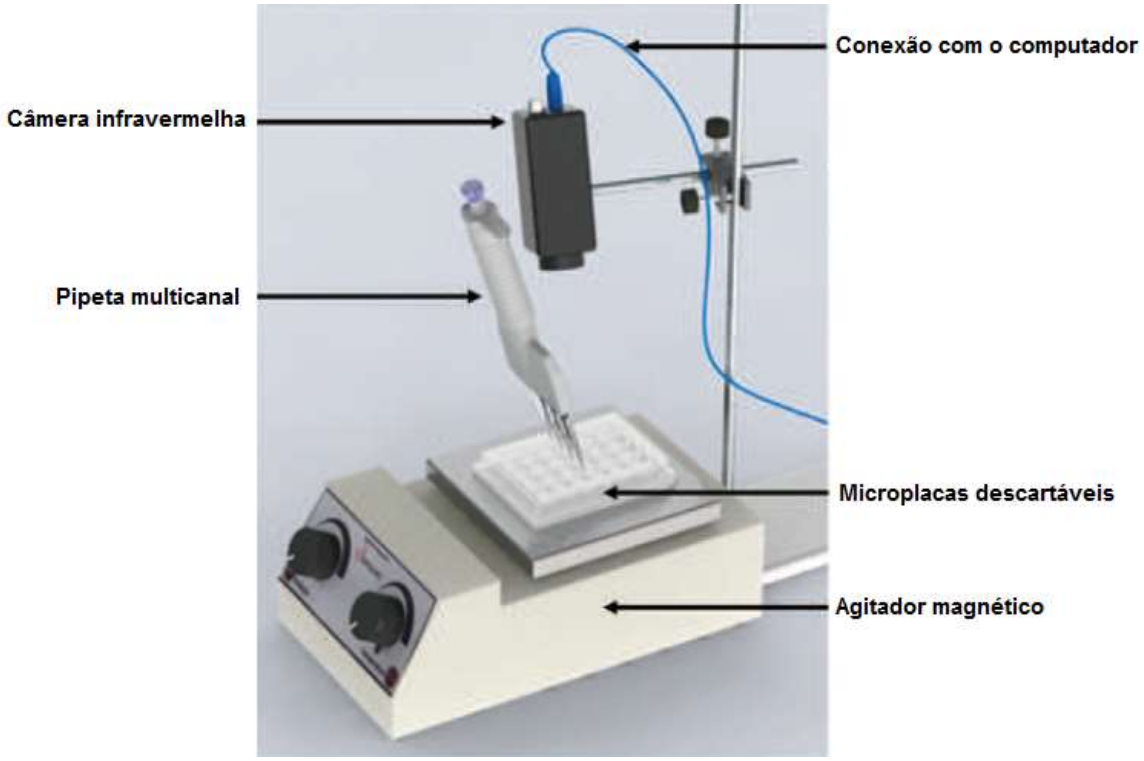
Fonte: (COSTA *et al.*, 2015).

3.2 Infrared Thermal Imaging: A Tool for Simple, Simultaneous, and High-Throughput Enthalpimetric Analysis

Este projeto demonstra a viabilidade da utilização de imagens térmicas por infravermelho para mostrar seu potencial para utilização na química analítica. Para demonstrar a

confiabilidade deste sistema, o trabalho utilizou um sistema de captura de imagens térmicas para realizar análises entalpimétricas de múltiplas reações de forma simultânea. A configuração do equipamento utilizado é demonstrada na Figura 8. Os resultados obtidos através dos experimentos realizados foram comparados com métodos tradicionais e, segundo os autores, obteve concordância entre 96% e 101%, com uma taxa de análises podendo chegar a milhares de amostras analisadas em 1 hora (BARIN *et al.*, 2015).

Figura 8 – Uso de imagem térmica para monitoramento simultâneo de reações

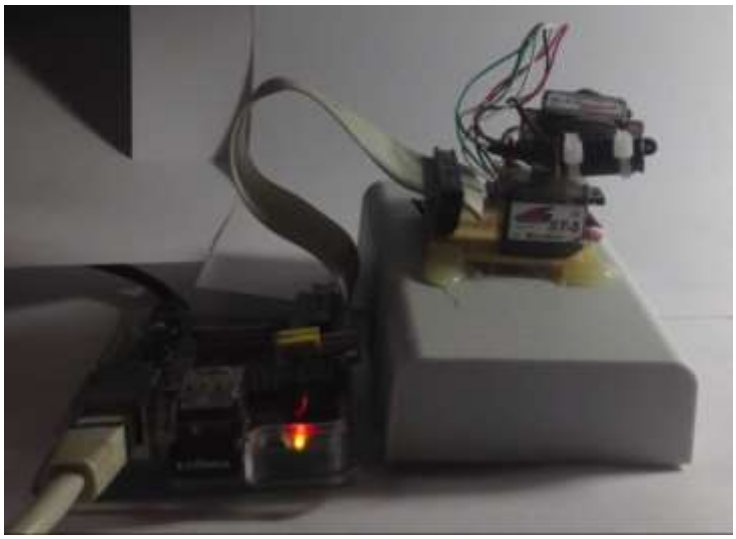


Fonte: Adaptado de (BARIN *et al.*, 2015).

3.3 Thermocam-raspi

O presente trabalho foi desenvolvido com base no projeto de uma câmera termográfica desenvolvida por Hermann Kurz (KURZ, 2013), no qual é utilizado um microcomputador *Raspberry Pi*, acoplado a um sensor infravermelho montado sobre servomotores, para gerar uma imagem térmica. A Figura 9 mostra o dispositivo montado.

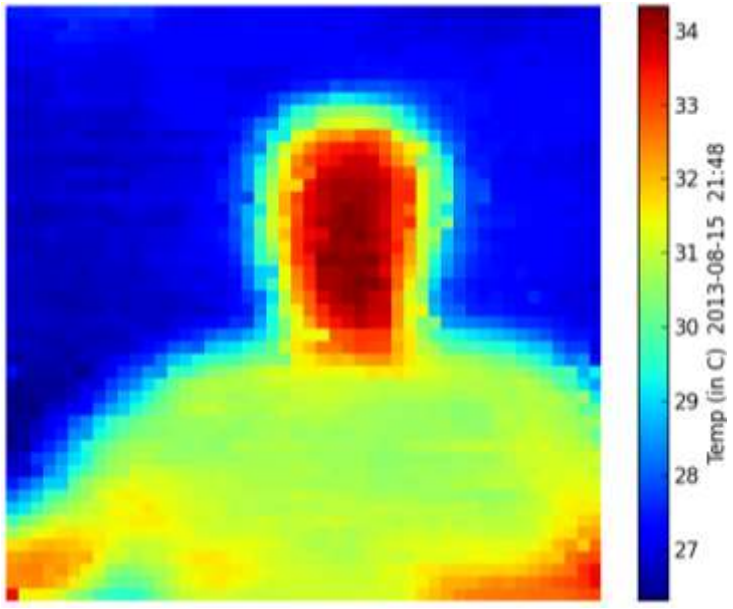
Figura 9 – Câmera térmica com Raspberry Pi



Fonte: Captura do vídeo demonstrativo disponível em <https://www.youtube.com/watch?v=lU73iyA51m8&feature=youtu.be&t=44s> acesso em 08/06/2017.

A leitura do sensor é feita em um *script Perl* executado no *Raspberry Pi*, e o valor lido pode ser consultado através de uma requisição *HTTP*. O programa principal é um servidor *web* escrito em *Python* que controla o movimento dos servomotores para realizar a leitura da temperatura e montar a imagem um *pixel* de cada vez, para então exibi-la em um navegador. O resultado da captura de uma imagem térmica com o sistema desenvolvido pode ser observado na Figura 10.

Figura 10 – Imagem térmica capturada

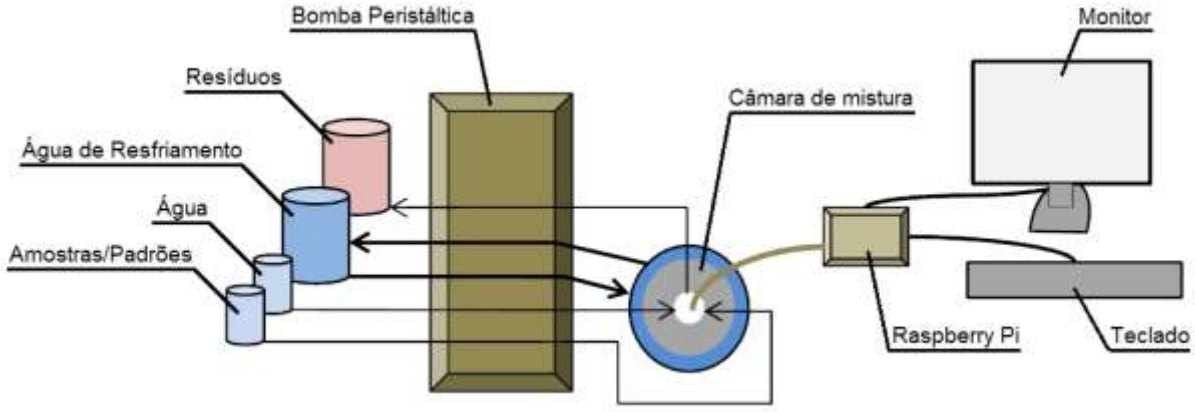


Fonte: Disponível em <http://1.bp.blogspot.com/-F1AtXyK4bsg/Un1iOoHYJmI/AAAAAAAAAVQ/Ja1NVYCymVY/s200/heatmap.png>, acesso em 08/06/2017.

3.4 Sistema para determinação, em fluxo, de etanol em bebidas destiladas por entalpimetria no infravermelho

O trabalho apresentado neste artigo demonstrou a viabilidade da combinação das técnicas de de análise em fluxo e entalpimetria no infravermelho para a determinação da quantidade de etanol em bebidas alcoólicas destiladas. Para tanto, os autores desenvolveram uma câmara de mistura, que foi posicionada sobre um agitador magnético, e na qual as amostras e reagentes foram injetados com a utilização de uma bomba peristáltica. Em um encaixe sobre esta câmara de mistura foi acondicionado um sensor infravermelho para monitorar a variação da temperatura da reação química.

Figura 11 – Sistema integrado TIE/FIA



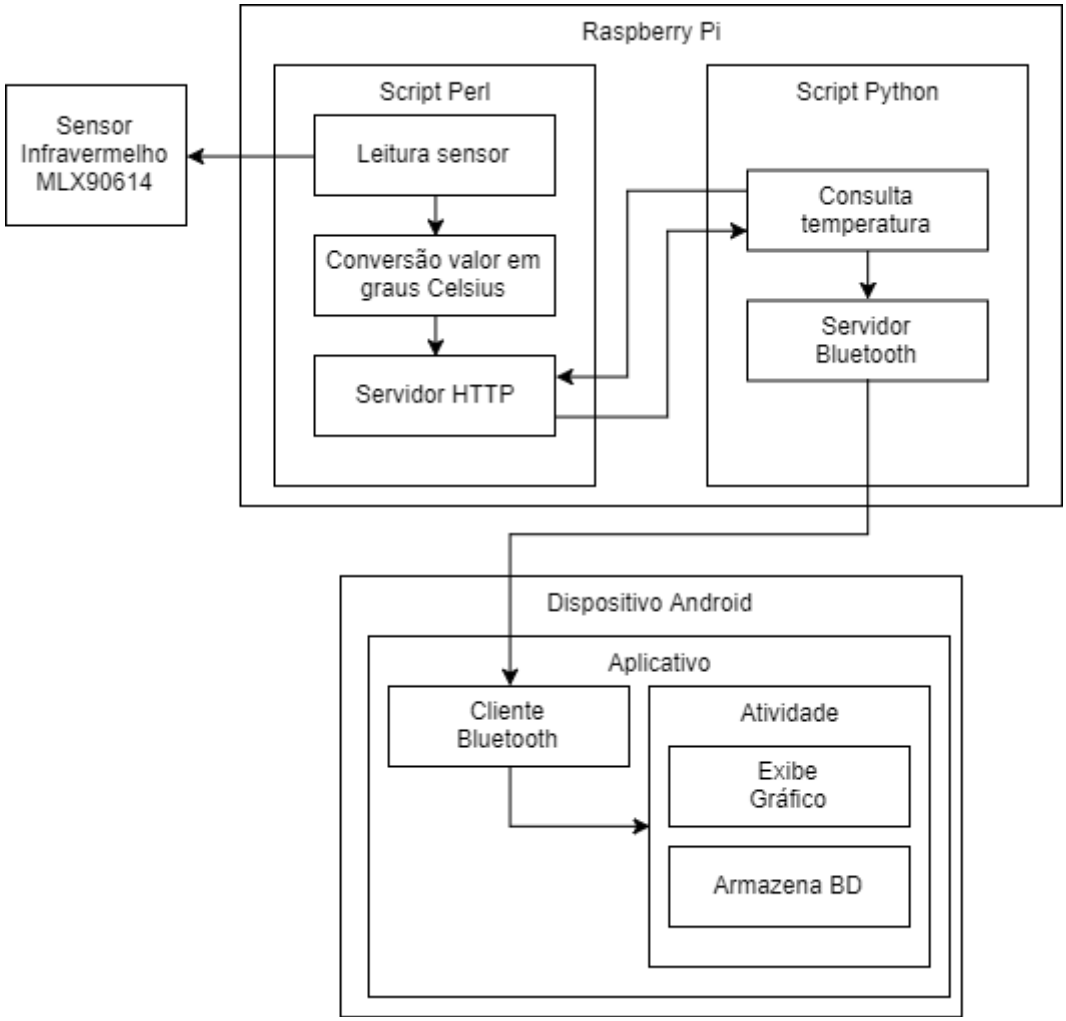
Fonte: Adaptado de (SANTOS, 2017).

Os autores afirmam que a metodologia utilizada eliminou a necessidade de tratamento prévio das amostras analisadas, o que diminuiu o tempo de realização da análise de aproximadamente 9 horas para 30 minutos. Adicionalmente, também foi reduzido em cinquenta vezes o volume das amostras. Conforme os autores, a viabilidade da metodologia proposta foi comprovada, apresentado resultados com concordância média de 98,99% em relação ao método tradicional e de 97,73% em relação aos rótulos dos produtos testados (SANTOS, 2017).

4 SOLUÇÃO IMPLEMENTADA

O presente capítulo descreve os detalhes tecnológicos da implementação do sistema. O mesmo consiste de duas partes: um microcomputador *Raspberry Pi*, que realiza a leitura do sensor infravermelho e a transmissão dos valores lidos por *Bluetooth*, e um aplicativo *Android*, que recebe os valores através da conexão *Bluetooth*, exibe-os em um gráfico ao mesmo tempo em que os armazena em um banco de dados e permite o compartilhamento dos dados por e-mail. O diagrama de blocos da Figura 12 demonstra o fluxo das funcionalidades básicas do sistema desenvolvido.

Figura 12 – Diagrama de blocos do sistema



Fonte: O autor.

4.1 Equipamento utilizado

Para a realização deste trabalho foram utilizados um microcomputador *Raspberry Pi 3 Model B*, um módulo de sensor de temperatura MLX90614 modelo GY-906 e um dispositivo *Android Alcatel Pixi 4*. Estes dispositivos são brevemente descritos a seguir.

4.1.1 Raspberry Pi 3 Model B

Conforme as informações do site do fabricante (RASPBERRY PI FOUNDATION, 2016a), este modelo conta com um processador 64-*bits* ARMv8 *quad-core* de 1,2GHz, 1GB de memória RAM, interfaces *Wifi* 802.11n e *Bluetooth*, entre outras características. A Figura 13 apresenta o aspecto do microcomputador *Raspberry Pi*.

Figura 13 – Microcomputador Raspberry Pi

Fonte: Disponível em <http://media.rs-online.com/t_large/F8968660-01.jpg>, acesso em 05/03/2017.

4.1.2 Sensor infravermelho MLX90614

De acordo com a documentação do fabricante (MELEXIS NV, 2015), o sensor utilizado tem tamanho reduzido e baixo custo, tendo capacidade para ler temperaturas de -70 a 380 graus Celsius. O sensor é ilustrado na Figura 14.

Figura 14 – Sensor Infravermelho MLX90614

Fonte: Disponível em <https://media.digikey.com/photos/Melexis%20Photos/MLX90614(E,K)SF-xxA.JPG>, acesso em 05/03/2017.

4.1.3 Android Alcatel Pixi 4

O dispositivo *Android* utilizado para a instalação do aplicativo conta com sistema operacional *Android* 6.0 *Marshmallow*, processador *quad-core* 1.3 GHz, 1GB de memória *RAM* e conectividade *Wifi* 802.11 b/g/n e *Bluetooth.* A Figura 15 apresenta o dispositivo utilizado.

Figura 15 – Dispositivo Android Alcatel Pixi 4

Fonte: Disponível em <http://www.alcatel-mobile.com/jo/upload/PIXI-4-(4)-Amber-Orange-Front.jpg>, acesso em 17/06/2017.

4.2 Raspberry Pi

Nesta seção serão explicitados os procedimentos que foram necessários para configurar o microcomputador *Raspberry Pi* para que fosse possível obter o valor da temperatura a partir do sensor infravermelho e posteriormente transmiti-la ao dispositivo *Android*.

4.2.1 Leitura dos dados

Para realizar a leitura da temperatura, primeiramente foi necessário que o sensor fosse fisicamente conectado ao *Raspberry*. Especificamente, o sensor utilizado neste projeto é identificado pelo número de modelo GY-906, um módulo que consiste de um pirômetro infravermelho MLX90614 montado em um circuito que oferece um regulador interno para alimentação de 3 a 5 Volts e resistores para os conectores de comunicação de dados (ARDUINO

HOUSE CWB, 2016). O aspecto do módulo do sensor de temperatura pode ser observado na Figura 16.

Figura 16 – Placa do sensor GY-906

Fonte: Disponível em <https://www.wiltronics.com.au/wp-content/uploads/images/make-and-create/arduino-non-contact-infra-red-temperature-sensor-module-top.jpg>, acesso em 15/06/2017

Para o propósito de conectar dispositivos desta natureza, o microcomputador *Raspberry* oferece um banco de conectores denominado *GPIO – General Purpose Input/Output* (RASPBERRY PI FOUNDATION, 2014), ou entrada/saída de propósito geral em tradução livre. De acordo com a documentação, estes pinos são a interface física entre o *Pi* e o mundo externo.

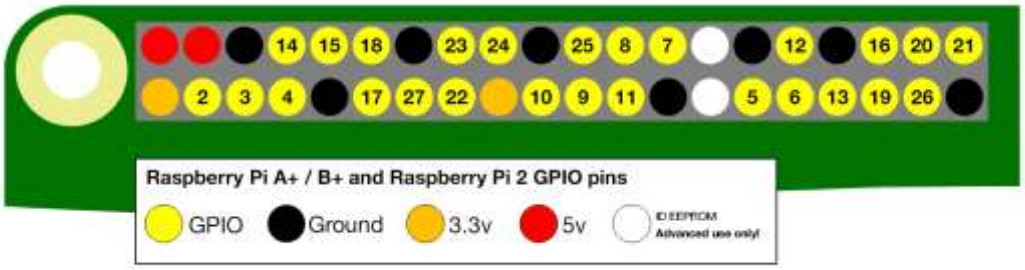
A disposição dos pinos pode ser observada na Figura 17. São 40 pinos no total, sendo que 26 destes são pinos para entrada e saída, 2 são para saída de tensão em 3,3 Volts, 2 para saída de tensão em 5 Volts, 8 para aterramento e 2 reservados para comunicação com dispositivos específicos chamados *HAT – Hardware Attached on Top* (RASPBERRY PI FOUNDATION, 2013), que pode ser traduzido como equipamento acoplado. Tais equipamentos devem atender a uma série de especificações para funcionarem acoplados ao *Raspberry*, no entanto não serão utilizados no escopo deste trabalho, tornando desnecessária uma descrição aprofundada.

Figura 17 – GPIO – Pinos de entrada/saída de propósito geral

Fonte: Disponível em <https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/images/gpio-pins-pi2.jpg>, acesso em 15/06/2017.

O diagrama na Figura 18 demonstra a disposição dos pinos de entrada e saída com as funções específicas dos pinos descritas na legenda.

Figura 18 – Funções dos pinos de entrada/saída



Fonte: Disponível em <https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/images/gpio-numbers-pi2.png>, acesso em 15/06/2017.

As conexões entre o sensor e o *Raspberry* são detalhadas na Tabela 2, gerada a partir de informações contidas em (MELEXIS NV, 2015) e (RASPBERRY PI FOUNDATION, 2016b). O pino *VIN* do sensor de temperatura serve para sua alimentação, o pino *GND* é para aterramento, o pino *SCL* recebe um sinal de *clock* enquanto o pino *SDA* disponibiliza o valor de temperatura lido pelo sensor.

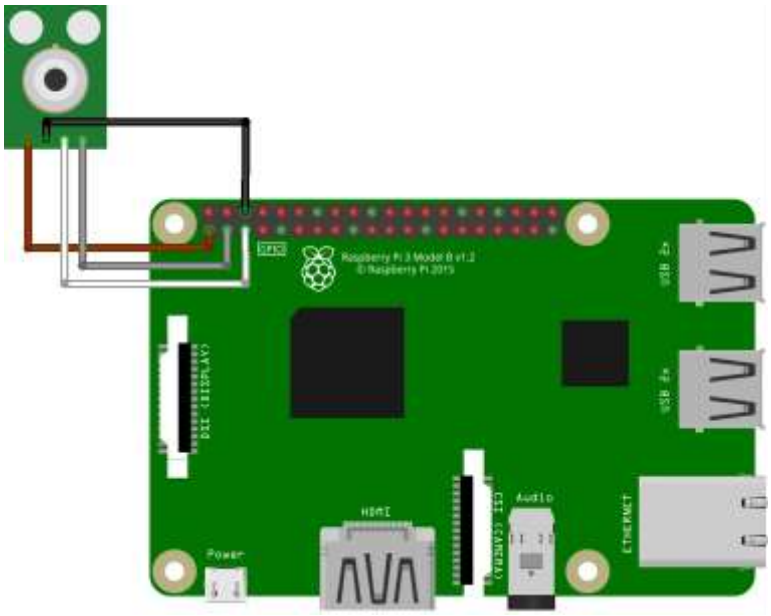
Tabela 2 – Pinagem para conexão do sensor ao Raspberry Pi

GY-906	***Raspberry Pi***	**Função**
VIN	3,3 Volts	Alimentação
GND	GND	Aterramento
SCL	SCL	Sinal de *clock*
SDA	SDA	Sinal de dados

Fonte: O autor

O diagrama demonstrando a ligação física entre o sensor e o *Pi* é mostrado na Figura 19, feito com a utilização do *software Fritzing*, uma iniciativa de *hardware open-source* para tornar a eletrônica acessível a todos como material criativo (FRIENDS OF FRITZING FOUNDATION, 2016).

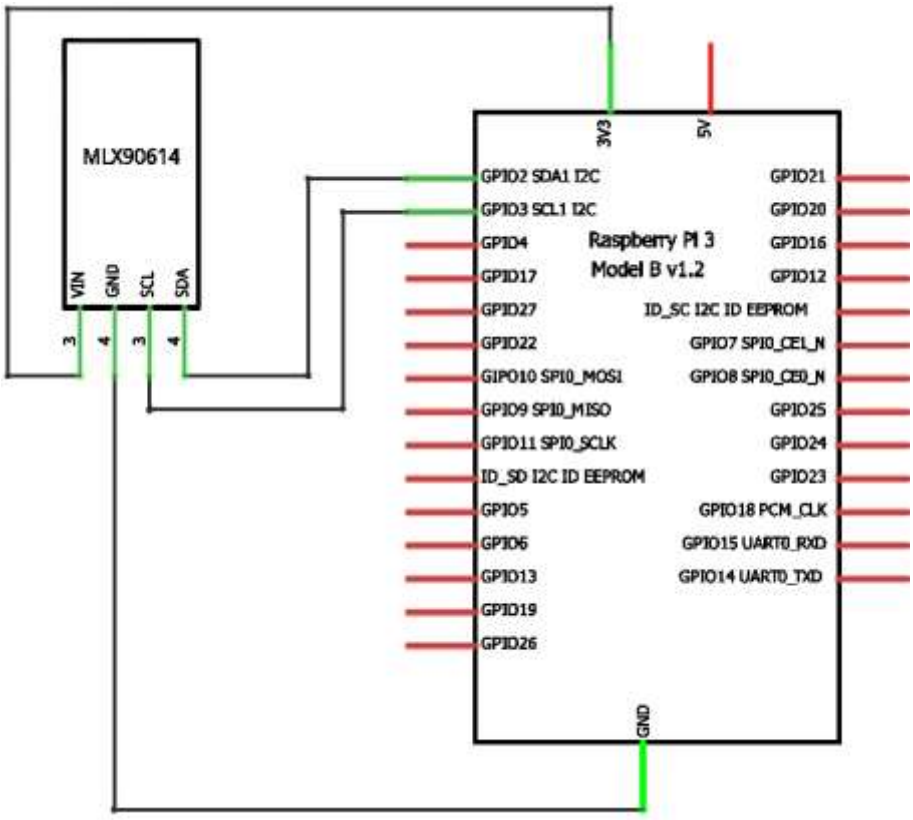
Figura 19 – Diagrama de ligação do sensor MLX90614 ao Raspberry



Fonte: O autor.

Na Figura 20 é possível verificar o esquema de ligação do sensor, com o detalhamento dos pinos utilizados em cada dispositivo.

Figura 20 – Esquema de ligação do sensor MLX90614 ao Raspberry



Fonte: O autor.

Para a leitura do sensor infravermelho em *software* e posterior disponibilização, foram seguidos os passos descritos por Hermann Kurz (KURZ, 2013), com a utilização do *script* em *Perl*.

A instalação do módulo *HiPi::Perl* para possibilitar a leitura do sensor através da interface I2C é realizada conforme a documentação em (DOOTSON, 2013), através da execução dos comandos exibidos no ANEXO A em um terminal no *Raspberry*. O comando *wget* faz o *download* dos arquivos indicados pela *URL* indicada como argumento (FREE SOFTWARE FOUNDATION, 2007), que neste caso é um *script* na linguagem *Perl* que tem como objetivo fazer a instalação dos módulos. Já o comando seguinte executa o *script* que foi baixado no interpretador *Perl*, efetivando a instalação.

Para possibilitar a consulta do valor lido pelo sensor o *script* utilizado inicia um servidor *HTTP*. Para fazer isso, foi necessária a instalação do módulo *Perl HTTP::Server::Simple,* com utilização do comando demonstrado no ANEXO B. A instrução apresentada utiliza o módulo *CPAN* para automatizar ou ao menos facilitar a intalação de módulos e extensões *Perl* buscando os arquivos em um repositório, e desempacotando-os em um diretório dedicado (PERL 5 PORTERS, 2009).

Uma vez instalados os módulos *Perl* citados, ao executar o *script* de leitura do sensor, será iniciado um servidor *web* executando na porta 8080 do *Raspberry Pi*, que ao receber uma requisição *HTTP* responderá conforme demonstrado na Tabela 3.

Tabela 3 – Requisições e respostas do servidor HTTP

Requisição	Resposta
http://endereço_Raspberry_Pi/c	Temperatura em graus Celsius.
http://endereço_Raspberry_Pi/f	Temperatura em graus Fahrenheit.
http://endereço_Raspberry_Pi/raw	Valor “cru” lido no sensor.
http://endereço_Raspberry_Pi/json	Todos os valores acima formatados em um objeto JSON.

Fonte: O autor.

A leitura do valor no sensor é realizada pela instrução `@reg_val = $dev->i2c_read_register_rs($register, 0x02);` que realiza a leitura no registro cujo endereço é informado no parâmetro `$register`, previamente definido no código com o endereço 0x07 na memória *RAM* do módulo do sensor infravermelho, onde é armazenada a temperatura do objeto medido pelo sensor (MELEXIS NV, 2015). O segundo argumento da instrução informa a quantidade de *bytes* a serem lidos, no caso 2, e o retorno deste comando é um vetor contendo os *bytes* lidos. O comando é realizado dentro do bloco da função `eval` para capturar erros na

leitura e tratá-los em tempo de execução. Em seguida, a instrução `$raw = @reg_val[1] * 256 + @reg_val[0];` regenera os *bytes* lidos em um valor único para posterior conversão para a unidade de medida desejada. A leitura é realizada dentro de um laço e, para que seja mais confiável, ela é repetida até que seja obtido um valor válido ou uma sequência de dois valores com uma diferença menor do que 1 grau Celsius.

A solução desenvolvida neste trabalho acabou utilizando apenas o valor de temperatura em graus Celsius, convertendo o valor "cru" lido no sensor para esta unidade de medida. A instrução `my ($temp_c) = ($raw / 50) - 273.15;` realiza a conversão do valor recuperado da memória do sensor. Segundo a documentação do sensor (MELEXIS NV, 2015), o valor lido deve ser dividido por 50 para obter a temperatura em graus Kelvin, de onde deve-se subtrair 273,15 para obter o valor em graus Celsius.

4.2.2 Conexão Bluetooth

O próximo passo para a realização dos objetivos deste trabalho foi possibilitar a comunicação dos valores obtidos do sensor de temperatura utilizando tecnologia *wireless*. Em função da opção de conexão por *Wi-fi* necessitar de *software* adicional e da configuração de um IP estático para o *Raspberry* ser configurado como um ponto de acesso (RASPBERRY PI FOUNDATION, 2017), e também em função da disponibilidade de exemplos detalhados na documentação da linguagem (KARULIS, 2015), optou-se por utilizar tecnologia *Bluetooth* através da linguagem *Python* para realizar a transmissão de dados entre os dispositivos.

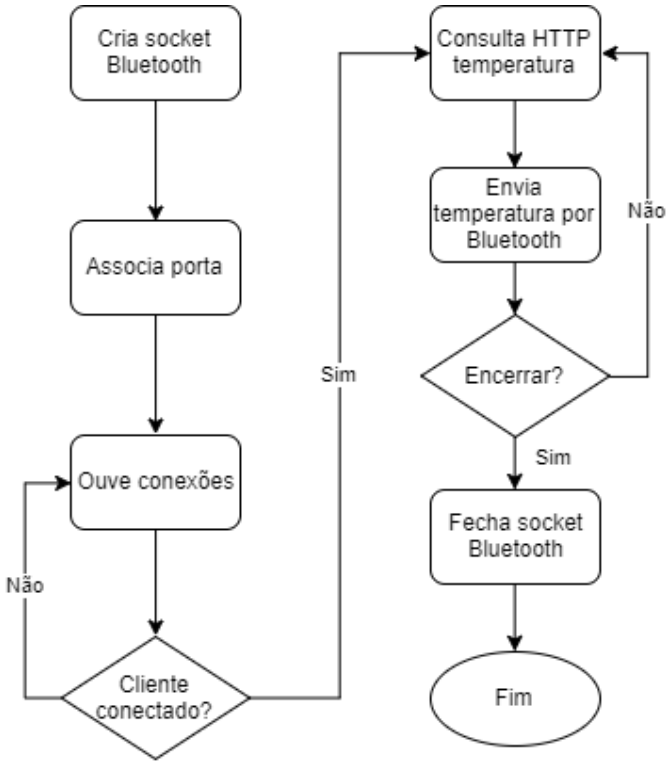
De modo a possibilitar a utilização dos recursos *Bluetooth* do *Raspberry Pi* em um *script Python*, foi necessária a instalação do módulo de extensão *PyBluez*. A instalação do módulo é feita com a utilização da ferramenta pip, cujo objetivo é realizar a instalação de pacotes *Python* (PYPA, 2014). O ANEXO C demonstra o comando para a instalação do módulo *PyBluez*.

Uma vez instalado o módulo *PyBluez*, foi possível utilizar os recursos *Bluetooth* no *script Python*, bastando para isso importar o módulo no início do *script*. A Figura 21 apresenta o fluxograma de transmissão dos dados por *Bluetooth*.

Posteriormente, é criado o *socket* servidor indicando-se o protocolo de transporte utilizado, no caso *RFCOMM*. Em seguida, outro comando indica que o *socket* pode ser atrelado a qualquer porta. O comando a seguir habilita o *socket* a receber conexões, sendo que o argumento na instrução *listen* indica a quantidade máxima de conexões não aceitas o sistema permite antes de recusar novas conexões (KARULIS, 2014a). A última instrução deste trecho anuncia o serviço com o servidor de protocolo de descoberta de serviços local, para possibilitar

a conexão por outros dispositivos que estejam buscando por este tipo de serviço (KARULIS, 2014b).

Figura 21 – Fluxograma de comunicação Bluetooth



Fonte: O autor.

Em seguida, um *socket* cliente é criado quando o *socket* servidor aceitar uma conexão, e então o *socket* cliente é configurado para não bloquear a execução do programa em operações que aguardariam a chegada de dados (HUANG e LARRY, 2007), como por exemplo o comando `select.select([client_sock], [], [], 1)`.

A leitura do valor da temperatura no sensor é realizada em um laço. A entrada neste laço é determinada pela recepção de um comando vindo do dispositivo remoto. A instrução `response=urllib2.urlopen("http://localhost:8080/c")` faz uma consulta *HTTP* ao servidor criado anteriormente para leitura do valor da temperatura. Em seguida este valor é atribuído à variável `temp`, para em seguida concatenar-se a mensagem de saída com um caracter delimitador, antes de transmití-la através do *socket* com o comando `send(msgOut)`.

Depois, a instrução `select.select([client_sock], [], [], 1)` verifica se foi recebida alguma mensagem do *socket* cliente, para determinar se ele deseja encerrar o processo de leitura do sensor. Nesta situação, este comando aguarda por 1 segundo antes de continuar a execução, em função do seu último argumento que indica o tempo de espera em segundos.

Por fim, caso ocorra algum erro ou o programa seja encerrado, são fechados os *sockets Bluetooth* cliente e servidor.

Para possibilitar o funcionamento da comunicação por *Bluetooth* foi necessário o registro do perfil de porta serial (*SPP*) que foi utilizado no *script*, isso é feito com a ferramenta *sdptool*, utilizada para controlar e interrogar servidores de descoberta de serviço (DUMBILL , 2015), conforme demonstrado no ANEXO D.

Após estes passos, o *script* pode ser executado e aguarda a conexão de algum dispositivo, quando isto acontecer é iniciado o laço de leitura e transmissão dos valores de temperatura através do *socket Bluetooth* para o dispositivo remoto.

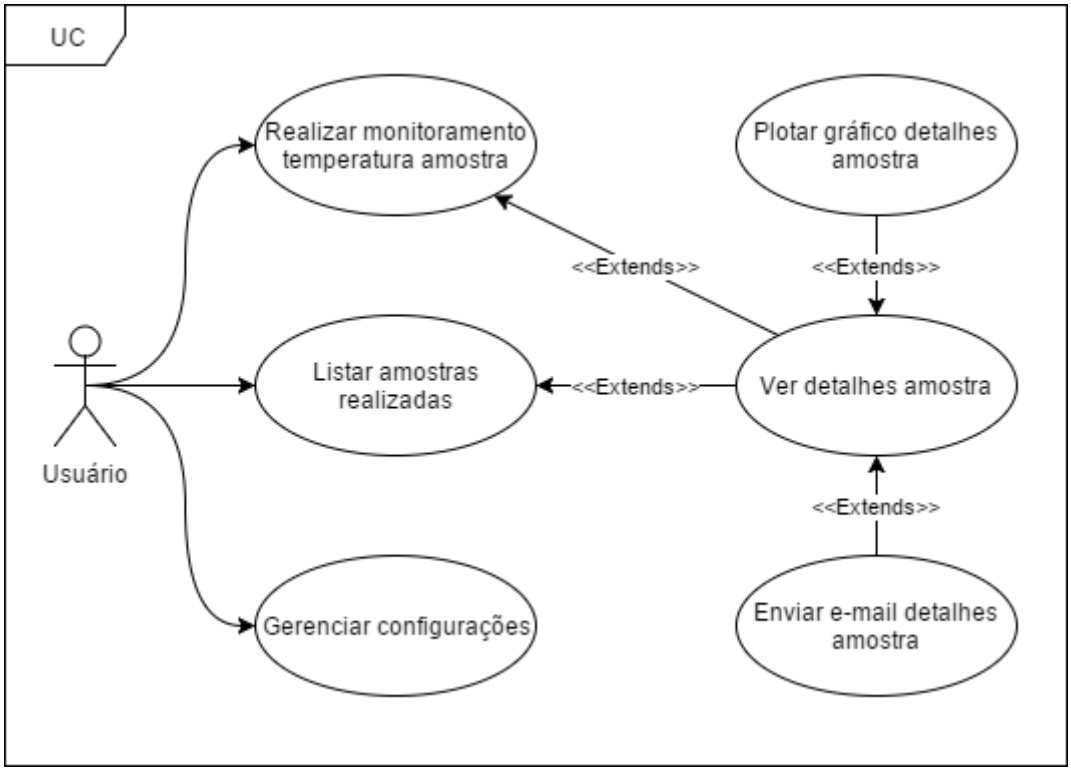
4.3 Aplicativo Android

Para o desenvolvimento do aplicativo para dispositivos *Android*, foi utilizada a *IDE Android Studio*. A seguir são apresentados os aspectos referentes à aplicação desenvolvida.

4.3.1 Casos de uso

Os casos de uso que foram definidos para implementação no aplicativo estão ilustrados na Figura 22.

Figura 22 – Diagrama de casos de uso do aplicativo



Fonte: O autor.

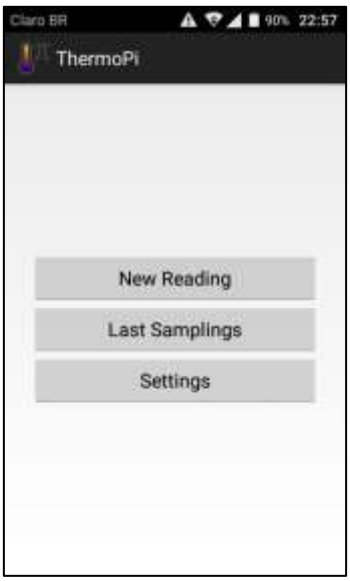
4.3.2 Atividades do aplicativo

A seguir são descritas as principais atividades desenvolvidas para o aplicativo, cada uma destas atividades representa uma tela com a qual o usuário pode interagir.

4.3.2.1 StartScreenActivity.java

Esta atividade é a tela inicial do aplicativo e, como mostra a Figura 23, é composta por três botões: "*New Reading*" que inicia o encadeamento de atividades para a realização de uma nova leitura de temperatura, "*Last Samplings*" que apresenta uma lista com as amostras armazenadas e "*Settings*" que permite a alteração de algumas configurações.

Figura 23 – Tela inicial do aplicativo



Fonte: O autor.

4.3.2.2 SampleActivity.java

Esta atividade é o primeiro passo para o processo de leitura de uma sequência de valores de temperatura de uma amostra. É neste ponto que são informados os dados de identificação da amostra que se deseja testar. Ao clicar no botão, os dados de identificação da amostra são armazenados no banco de dados e a chave de identificação é devolvida para a atividade. Este código então é passado como parâmetro para a atividade de leitura em tempo real para que os valores de temperatura possam ser armazenados com o identificador da amostra a que pertencem. A Figura 24 apresenta a tela de inserção de dados da amostra.

Figura 24 – Tela de dados da amostra

Fonte: O autor.

4.3.2.3 RTTemperatureChartActivity.java

Esta atividade recebe os dados da *thread* que está realizando a comunicação por *Bluetooth*, e através de um *handler* adiciona os valores recebidos no gráfico e no banco de dados conforme os vai recebendo, gerando um gráfico em tempo real. A tela desta atividade é representada na Figura 25.

Figura 25 – Tela do gráfico em tempo real

Fonte: O autor.

4.3.2.4 Bluetooth.java

Esta atividade é iniciada quando o usuário clicar no botão para conectar o dispositivo na atividade do gráfico em tempo real, mostrado na Figura 25, e realiza as funções relativas à conexão por *Bluetooth*, incluindo verificar se o dispositivo é compatível com esta tecnologia, se o adaptador *Bluetooth* está ligado e solicitar ao sistema *Android* que o ligue caso o usuário permita. Caso o adaptador *Bluetooth* esteja ligado, esta atividade lista os dispositivos *Bluetooth*

que estão em modo visível indicando os que já foram pareados, como mostra a Figura 26. É necessário para o funcionamento da comunicação que os dispositivos já tenham sido pareados anteriormente. Quando o usuário selecionar o dispositivo desejado na lista, é realizada a conexão entre o dispositivo *Android* e o *Raspberry Pi* em uma nova *thread*, e é enviado o comando para que o *script* sendo executado no *Raspberry* inicie a consulta e envio dos valores lidos no sensor. Após a conexão ser efetivada, esta atividade é encerrada e o aplicativo retorna para o gráfico em tempo real.

Figura 26 – Tela de seleção do dispositivo Bluetooth

Fonte: O autor.

4.3.2.5 DataActivity.java

Esta atividade apresenta ao usuário uma lista contendo as amostras realizadas que estão armazenadas no banco de dados, conforme a Figura 27. Ao clicar sobre um item da lista, o usuário é levado a uma outra tela que apresenta os detalhes da amostra.

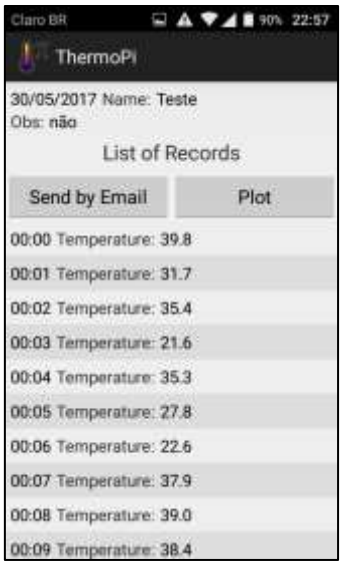
Figura 27 – Tela da lista de últimas amostras

Fonte: O autor.

4.3.2.6 DetailActivity.java

Aqui são exibidos os detalhes da amostra, apresentando uma lista com os valores de temperatura armazenados. Também são disponibilizados botões para que o usuário possa enviar os detalhes por *e-mail* ou exibi-los em um gráfico. A Figura 28 apresenta a tela desta atividade.

Figura 28 – Tela de detalhes da amostra

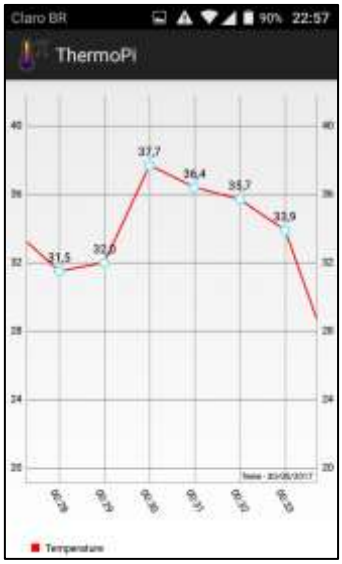


Fonte: O autor.

4.3.2.7 DBChartActivity.java

Nesta tela do aplicativo os detalhes da amostra armazenada são exibidos através de um gráfico de linha. Os valores de temperatura da amostra são recuperados do banco de dados e plotados em um gráfico, como mostra a Figura 29.

Figura 29 – Tela do gráfico de amostra



Fonte: O autor.

4.3.2.8 SettingsActivity.java

Ilustrada na Figura 30, esta atividade permite ao usuário definir algumas configurações do aplicativo de forma permanente. Nesta tela é definida a faixa de temperaturas que é exibida no gráfico em tempo real, além disso também pode-se definir endereços de *e-mail* padrão dos destinatários para os quais serão enviados os *e-mails* contendo os dados de temperatura. Os dados são armazenados utilizando a classe "*SharedPreferences*" que disponibiliza uma forma simples de armazenar dados associados a identificadores para que possam ser recuperados posteriormente.

Figura 30 – Tela de configurações

Fonte: O autor.

4.3.3 Conexão Bluetooth

Para realizar a comunicação por *Bluetooth* entre os dispositivos foi utilizada a *API Bluetooth* para *Android* (GOOGLE E OPEN HANDSET ALLIANCE, 2016). Conforme a documentação, esta interface permite a procura por outros dispositivos *Bluetooth*, a consulta ao adaptador *Bluetooth* local para verificar a existência de dispositivos *Bluetooth* pareados, o estabelecimento de canais *RFCOMM*, a conexão a outros dispositivos por meio da descoberta de serviços, a transferência de dados de e para outros dispositivos e o gerenciamento de várias conexões.

4.3.4 Persistência dos dados

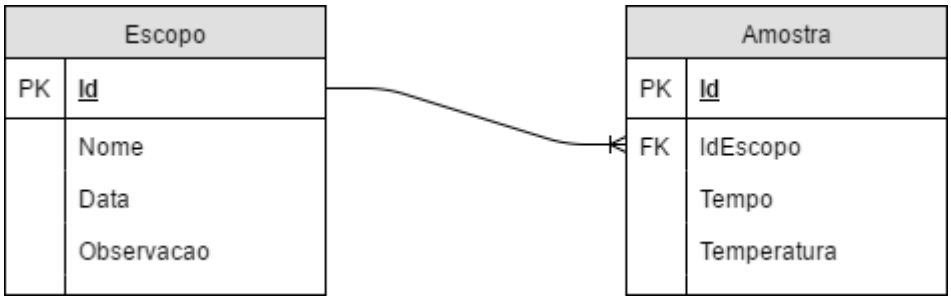
O armazenamento dos valores foi implementado com a criação de um banco de dados *SQLite* (SQLITE CONSORTIUM, 2016). O banco de dados consiste de duas tabelas: “Escopo” e “Amostra”. A Figura 31 descreve em um diagrama entidade-relacionamento as tabelas utilizadas para o armazenamento dos dados.

Pode ser observado que a tabela “Escopo” armazena informações de identificação de cada amostra realizada, contendo os campos “Nome”, destinado ao nome da amostra, “Data”, destinado ao armazenamento da data de realização da leitura, e “Observacao”, para que o usuário possa incluir alguma informação que julgue relevante.

Já a tabela “Amostra” contém os campos “idEscopo”, para relacionar os dados de tempo e temperatura com os dados da tabela “Escopo”, “Tempo”, para armazenar o tempo decorrido

em relação ao início da leitura, e “Temperatura”, para o armazenamento da temperatura lida no sensor naquele instante.

Figura 31 – Diagrama entidade-relacionamento do banco de dados



Fonte: O autor.

4.3.5 Apresentação dos dados

Para possibilitar a apresentação dos dados de temperatura ao usuário do aplicativo, foi utilizada a biblioteca *MpAndroidChart* (JAHODA, 2016), que disponibiliza 8 tipos diferentes de gráficos com recursos de interatividade como *zoom* e arraste da tela para melhor visualização das informações.

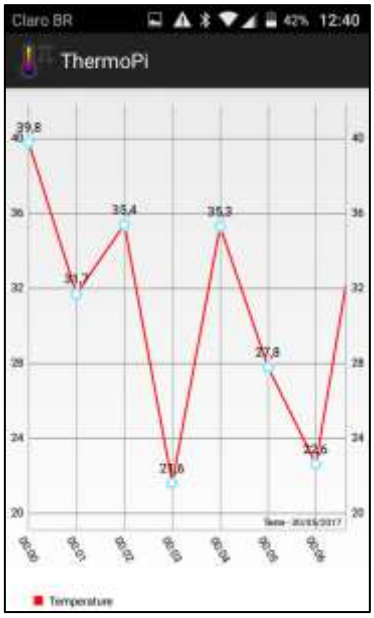
O primeiro passo para a utilização da biblioteca, conforme a documentação contida no repositório do projeto, é adicionar referências a ela nos arquivos *build.gradle* do projeto do *Android Studio*. No arquivo *build.gradle* do nível do projeto são adicionadas as instruções mostradas no ANEXO E, contendo o endereço do repositório da biblioteca, já no arquivo *build.gradle* do nível do aplicativo, é necessário que se adicione o comando apresentado no ANEXO F.

Depois desta etapa, já é possível utilizar a biblioteca para criar gráficos no projeto. Para inserir um gráfico em uma atividade do aplicativo, pode-se adicioná-lo no arquivo “*xml*” de *layout* e em seguida recuperá-lo na atividade correspondente. O trecho de código apresentado no ANEXO G exemplifica este método de criação do objeto gráfico.

Depois de criado o objeto do gráfico é necessário que se adicione dados a ele. Isto é feito com a criação de um objeto do tipo “*List”* contendo objetos “*Entry*”, que são os objetos que representam cada ponto no gráfico com um valor para o eixo X e um valor para o eixo Y. Conforme demonstrado no exemplo contido no ANEXO H, após a criação da lista de entradas, é necessário popular esta lista com novos objetos “*Entry”*, com os valores X e Y. Depois disto, é criado um objeto “*LineDataSet*”, que representa uma linha do gráfico. Um gráfico pode ter várias linhas que podem ser configuradas com estilos diferentes, possibilitando a definição de propriedades como a cor da linha, por exemplo. Em seguida é criado um objeto “*LineData*” que

armazena todas as linhas do gráfico, e por fim adiciona-se este objeto ao gráfico. A Figura 32 ilustra a aparência do gráfico.

Figura 32 – Aparência do gráfico utilizado no aplicativo

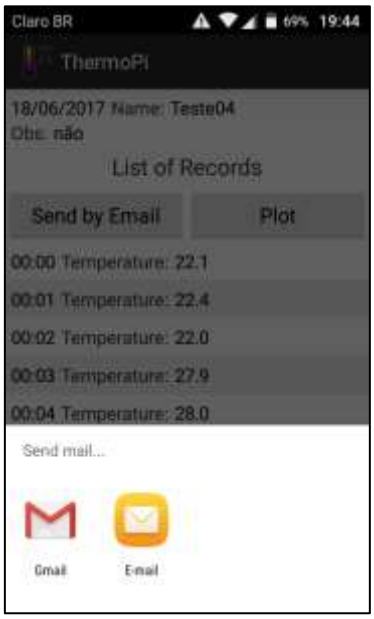


Fonte: O autor.

4.3.6 Compartilhamento dos dados

A função de compartilhamento dos dados é obtida através da criação de um "*Intent*" no aplicativo solicitando ao sistema *Android* o envio de uma mensagem de *e-mail*. A atividade recupera no banco de dados a lista de valores de temperatura armazenados e tabula-os em um objeto "*String*" que é passado como parâmetro ao "*Intent*" para iniciar a atividade do aplicativo de *e-mail*. O sistema *Android* então, apresenta ao usuário as opções de aplicativos de *e-mail* disponíveis conforme demonstra a Figura 33. Os dados são repassados ao aplicativo de *e-mail* que é encerrado após o envio da mensagem, levando o usuário de volta à tela de detalhes da mensagem.

Figura 33 – Envio dos dados por e-mail



Fonte: O autor.

5 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

O objetivo principal deste trabalho foi desenvolver uma interface gráfica de controle e monitoramento para um sistema de titulação termométrica utilizando *smartphone*. Para tanto foram implementados em um microcomputador *Raspberry Pi scripts* que permitem a realização da leitura da temperatura em um sensor infravermelho e posterior transmissão por *Bluetooth* para um aplicativo *Android*. No aplicativo *Android* foram criadas atividades que permitem o acompanhamento da temperatura através de um gráfico plotado em tempo real, bem como o armazenamento destas informações para posterior compartilhamento.

Após a implementação, obteve-se um sistema que é capaz de monitorar a temperatura de uma reação química à distância, limitado apenas pelo alcance do transmissor *Bluetooth*, bem como armazenar estes dados para análise posterior e compartilhar estes dados por *e-mail* em forma de texto tabulado, permitindo por exemplo que seja copiado e colado para utilização em planilhas eletrônicas.

Após o desenvolvimento deste projeto, pode-se afirmar que é possível a implementação de um sistema remoto para monitoramento da temperatura de reações químicas e, como se pode observar na Tabela 4, a um custo relativamente acessível se comparado a sistemas que utilizem câmeras infravermelhas.

Tabela 4 – Comparativo de custo

Equipamento	Custo
Câmera FLIR A35	US$ 4.995,00 (OEM Cameras, 2016)
Câmera FLIR E60	US$ 4.999,00 (Flir Store, 2013)
Raspberry Pi + Sensor Melexis	£ 32.99 (RS Components, 2016) + US$ 14,08 (Digi-Key Electronics, 2016)

Fonte: Elaborado pelo autor.

Com relação a trabalhos futuros, há diversas possibilidades de aperfeiçoamento para este sistema. Como sugestões, pode-se procurar aumentar a autonomia do *Raspberry*, automatizando o pareamento dos dispositivos e a inicialização dos *scripts*, bem como melhorar o processo de conexão pelo aplicativo e adicionar função de criação de curvas de calibração para predição de dados.

REFERÊNCIAS

ANTON-HARO, C.; DOHLER, M. (.). **Machine-to-machine (M2M) communications:** architecture, performance and applications. [S.l.]: Elsevier, 2014.

ARDUINO HOUSE CWB. Sensor de temperatura Infravermelho GY-906. **Arduino House**, 2016. Disponivel em: <http://arduinohouse.com.br/ahcwb/gy-906-sensor-de-temperatura-infravermelho>. Acesso em: 15 jun. 2017.

BARIN, J. S. et al. Infrared Thermal Imaging: A Tool for Simple, Simultaneous, and High-Throughput Enthalpimetric Analysis. **Analytical chemistry**, v. 87, n. 24, p. 12065-12070, 2015.

BARTHEL, J.; WACHTER, R. **Thermometric titrations**. [S.l.]: John Wiley & Sons, v. 45, 1975. 209 p.

BASAGNI, S. et al. (Eds.). **Mobile ad hoc networking**. [S.l.]: John Wiley & Sons, 2004.

CONSTANTINO, M. G.; DA SILVA, G. V. J.; DONATE, P. M. **Fundamentos de Química Experimental Vol. 53**. [S.l.]: Edusp, 2004.

COSTA, A. B. et al. Aplicação da termografia por infravermelho para titulações termométricas. **Orbital - The Electronic Journal of Chemistry**, v. 7, n. 2, p. 196-201, 2015.

DIAS, S. L. P. et al. **Química Analítica:** Teoria e Prática Essenciais. [S.l.]: Bookman Editora, 2016.

DIGI-KEY Electronics, 2016. Disponivel em: <https://www.digikey.com/product-detail/en/melexis-technologies-nv/MLX90614ESF-BAA-000-TU/MLX90614ESF-BAA-000-TU-ND/1647941>. Acesso em: 04 Março 2017.

DOHLER, M.; WATTEYNE, T.; ALONZO-ZARATE, J. Machine-to-machine: An emerging communication paradigm. **Wireless World Research Forum (WWRF)**, 2010.

DOOTSON, M. HiPi Perl Modules for the Raspberry Pi. **Perl and Raspberry Pi**, 2013. Disponivel em: <http://raspberry.znix.com/p/install.html>. Acesso em: 15 jun. 2017.

DUMBILL , E. General Commands Manual - sdptool. **Debian Manpages**, 2015. Disponivel em: <https://manpages.debian.org/jessie/bluez/sdptool.1.en.html>. Acesso em: 16 jun. 2017.

FLIR Store, 2013. Disponivel em: <http://store.flir.com/product/e60-ir-camera/e-series-infrared-cameras>. Acesso em: 04 Março 2017.

FREE SOFTWARE FOUNDATION. GNU Operating System. **GNU Wget**, 2007. Disponivel em: <https://www.gnu.org/software/wget/>. Acesso em: 15 jun. 2017.

FRIENDS OF FRITZING FOUNDATION. Fritzing. **Fritzing - electronics made easy**, 2016. Disponivel em: <http://fritzing.org/home/>. Acesso em: 15 jun. 2017.

GOOGLE E OPEN HANDSET ALLIANCE. Manifesto do aplicativo. **Android API Guides**, 2009. Disponivel em: <https://developer.android.com/guide/topics/manifest/manifest-intro.html>. Acesso em: 10 jun. 2017.

GOOGLE E OPEN HANDSET ALLIANCE. Android API Guide. **Serviços**, 2011. Disponivel em: <https://developer.android.com/guide/components/services.html>. Acesso em: 10 jun. 2017.

GOOGLE E OPEN HANDSET ALLIANCE. Android API Guide. **Criação de um provedor de conteúdo**, 2012a. Disponivel em: <https://developer.android.com/guide/topics/providers/content-provider-creating.html>. Acesso em: 10 jun. 2017.

GOOGLE E OPEN HANDSET ALLIANCE. Android Training. **Communicating with the UI thread**, 2012b. Disponivel em: <https://developer.android.com/training/multiple-threads/communicate-ui.html>. Acesso em: 10 jun. 2017.

GOOGLE E OPEN HANDSET ALLIANCE. Intents and Intent Filters. **Android API Guides**, 2013. Disponivel em: <https://developer.android.com/guide/components/intents-filters.html>. Acesso em: 05 jun. 2017.

GOOGLE E OPEN HANDSET ALLIANCE. Introdução ao Android. **Android API Guides**, 2015a. Disponivel em: <https://developer.android.com/guide/index.html?hl=pt-BR>. Acesso em: 09 jun. 2017.

GOOGLE E OPEN HANDSET ALLIANCE. Fundamentos de aplicativos. **Android API Guides**, 2015b. Disponivel em:

<https://developer.android.com/guide/components/fundamentals.html?hl=pt-BR>. Acesso em: 10 jun. 2017.

GOOGLE E OPEN HANDSET ALLIANCE. Android API Guide. **Atividades**, 2015c. Disponivel em: <https://developer.android.com/guide/components/activities.html?hl=pt-BR>. Acesso em: 08 jun. 2017.

GOOGLE E OPEN HANDSET ALLIANCE. Bluetooth. **Android API Guides**, 2016. Disponivel em: <https://developer.android.com/guide/topics/connectivity/bluetooth.html>. Acesso em: 18 jun. 2017.

GOOGLE E OPEN HANDSET ALLIANCE. Android API Guide. **The activity lifecycle**, 2016a. Disponivel em: <https://developer.android.com/guide/components/activities/activity-lifecycle.html>. Acesso em: 10 jun. 2017.

GOOGLE E OPEN HANDSET ALLIANCE. Android API Guide. **Provedores de conteúdo**, 2016b. Disponivel em: <https://developer.android.com/guide/topics/providers/content-providers.html>. Acesso em: 10 jun. 2017.

HOLLER, J. et al. **From Machine-to-machine to the Internet of Things:** Introduction to a New Age of Intelligence. [S.l.]: Academic Press, 2014.

HUANG, A. S.; LARRY, R. **Bluetooth essentials for programmers**. [S.l.]: Cambridge University Press, 2007.

IUPAC. **Compendium of Chemical Terminology, 2nd ed. (the "Gold Book")**. [S.l.]: Blackwell Scientific Publications, Oxford, 1997. ISBN ISBN 0-9678550-9-8. XML on-line corrected version: http://goldbook.iupac.org (2006-) created by M. Nic, J. Jir.

JAHODA, P. MPAndroidChart. **Repositório GitHub**, 2016. Disponivel em: <https://github.com/PhilJay/MPAndroidChart>. Acesso em: 17 jun. 2017.

KARULIS, P. Class BluetoothSocket - Listen. **PyBluez Documentation**, 2014a. Disponivel em: <https://htmlpreview.github.io/?https://raw.githubusercontent.com/karulis/pybluez/master/docs/public/bluetooth.BluetoothSocket-class.html#listen>. Acesso em: 16 jun. 2017.

KARULIS, P. Module Bluetooth - Advertise Service. **PyBluez Documentation**, 2014b. Disponivel em: <https://htmlpreview.github.io/?https://raw.githubusercontent.com/karulis/pybluez/master/docs/public/bluetooth-module.html#advertise_service>. Acesso em: 17 jun. 2017.

KARULIS, P. PyBluez. **Repositório GitHub**, 2015. Disponivel em: <https://github.com/karulis/pybluez>. Acesso em: 16 jun. 2017.

KURZ, H. Thermocam-raspi. **Repositório GitHub**, 2013. Disponivel em: <https://github.com/hermann-kurz/thermography-raspberrypi>.

LEE, J.-S.; SU, Y.-W.; SHEN, C.-C. A comparative study of wireless protocols: Bluetooth, UWB, ZigBee, and Wi-Fi. **Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE**, 2007. 46-51.

MELEXIS NV. Digital Plug & Play Infrared Thermometer in a TO-Can. **Melexis NV**, 2015. Disponivel em: <https://www.melexis.com/-/media/files/documents/datasheets/mlx90614-datasheet-melexis.pdf>. Acesso em: 15 jun. 2017.

OEM Cameras, 2016. Disponivel em: <http://www.oemcameras.com/flir-a35-19mm.htm>. Acesso em: 04 Março 2017.

PERL 5 PORTERS. Perl Programming Documentation. **CPAN**, 2009. Disponivel em: <http://perldoc.perl.org/CPAN.html>. Acesso em: 16 jun. 2017.

PRABHU, C. S. R.; REDDI, A. P. **Bluetooth Technology:** And Its Applications With Java And J2Me. [S.l.]: PHI Learning Pvt. Ltd., 2004.

PYPA. pip Documentation. **pip stable**, 2014. Disponivel em: <https://pip.pypa.io/en/stable/>. Acesso em: 16 jun. 2017.

RASPBERRY PI FOUNDATION. B+ ADD-ON BOARDS AND HATs. **Repositório GitHub**, 2013. Disponivel em: <https://github.com/raspberrypi/hats>. Acesso em: 15 jun. 2017.

RASPBERRY PI FOUNDATION. GPIO: MODELS A+, B+, RASPBERRY PI 2 B AND RASPBERRY PI 3 B. **Raspberry Pi Documentation**, 2014. Disponivel em: <https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/README.md>. Acesso em: 15 jun. 2017.

RASPBERRY PI FOUNDATION. Raspberry Pi 3 Model B. **Raspberry Pi**, 2016a. Disponivel em: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>. Acesso em: 18 jun. 2017.

RASPBERRY PI FOUNDATION. Raspberry Pi Documentation. **Raspberry Pi 3 Model B (Reduced Schematics)**, 2016b. Disponivel em: <https://www.raspberrypi.org/documentation/hardware/raspberrypi/schematics/Raspberry-Pi-3B-V1.2-Schematics.pdf>. Acesso em: 15 jun. 2017.

RASPBERRY PI FOUNDATION. Setting up a Raspberry Pi as an access point is a standalone network. **Raspberry Pi Documentation**, 2017. Disponivel em: <https://www.raspberrypi.org/documentation/configuration/wireless/access-point.md>. Acesso em: 19 jun. 2017.

RS Components, 2016. Disponivel em: <http://uk.rs-online.com/web/p/processor-microcontroller-development-kits/8968660/>. Acesso em: 04 Março 2017.

SANTOS, R. B. **Sistema para determinação, em fluxo, de etanol em bebidas destiladas por entalpimetria no infravermelho**, Monografia de conclusão do curso de Química Industrial. Universidade de Santa Cruz do Sul, RS, Brasil, 2017.

SQLITE CONSORTIUM. SQLite Home. **SQLite**, 2016. Disponivel em: <http://sqlite.org/>. Acesso em: 21 jun. 2017.

STAHL, J. W. Nomenclature of thermometric and enthalpimetric methods in chemical analysis (IUPAC Recommendations 1994). **Pure and Applied Chemistry**, v. 66, n. 12, p. 2487-2492, 1994.

UNDERDAHL, K. **Wi-Fi Home Networking Just the Steps For Dummies**. [S.l.]: John Wiley & Sons, 2011.

ANEXO A – Instalação dos módulos HiPi::Perl

```
wget http://raspberry.znix.com/hipifiles/hipi-install
perl hipi-install
```

Fonte: Disponível em <http://raspberry.znix.com/p/install.html>, acesso em 15/06/2017.

ANEXO B – Instalação do módulo HTTP::Server::Simple

```
sudo perl -MCPAN -e 'install HTTP::Server::Simple'
```

Fonte: Disponível em <http://www.xn--c-lmb.net/2013/08/waermebildkamera-mit-dem-raspberry-pi-ansteuerung-des-sensors-mlx90614-mit-perl.html>, acesso em 15/06/2017.

ANEXO C – Instalação do módulo de extensão PyBluez

```
sudo apt-get install python-dev
sudo apt-get install libbluetooth-dev
sudo pip install pybluez
```

Fonte: Disponível em < https://github.com/karulis/pybluez>, acesso em 16/06/2017.

ANEXO D – Registro do perfil de porta serial

```
sudo sdptool add SP
```

Fonte: Disponível em < https://manpages.debian.org/testing/obdgpslogger/obdsim.1.en.html>, acesso em 16/06/2017.

ANEXO E – Adição da biblioteca no arquivo build.gradle do nível do projeto

```
allprojects {
       repositories {
              maven { url "https://jitpack.io" }
       }
}
```

Fonte: Disponível em <https://github.com/PhilJay/MPAndroidChart#usage>, acesso em 17/06/2017.

ANEXO F – Adição da biblioteca no arquivo build.gradle do nível do aplicativo

```
dependencies {
        compile 'com.github.PhilJay:MPAndroidChart:v3.0.2'
}
```

Fonte: Disponível em <https://github.com/PhilJay/MPAndroidChart#usage>, acesso em 17/06/2017.

ANEXO G – Criação do gráfico no arquivo xml de layout

Adição no arquivo "*xml"*:

```
<com.github.mikephil.charting.charts.LineChart
        android:id="@+id/chart"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
```

Recuperação do gráfico no código da classe correspondente ao *layout*:

```
LineChart chart = (LineChart) findViewById(R.id.chart);
```

Fonte: Disponível em <https://github.com/PhilJay/MPAndroidChart/wiki/Getting-Started>, acesso em 17/06/2017.

ANEXO H – Adição de valores no gráfico

```
YourData[] dataObjects = ...;

List<Entry> entries = new ArrayList<Entry>();

for (YourData data : dataObjects) {
    entries.add(new Entry(data.getValueX(), data.getValueY()));
}

LineDataSet dataSet = new LineDataSet(entries, "Label"); // add entries to dataset

LineData lineData = new LineData(dataSet);
chart.setData(lineData);
chart.invalidate(); // refresh
```

Fonte: Disponível em <https://github.com/PhilJay/MPAndroidChart/wiki/Getting-Started>, acesso em 17/06/2017.