UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTOS ACADÊMICOS DE ELETRÔNICA E INFORMÁTICA
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

ANDRÉ FERREIRA ELEUTERIO

PLATAFORMA PARA DIVERSIFICAÇÃO DO USO DE CARTÃO DE TRANSPORTE EM CURITIBA

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2017

ANDRÉ FERREIRA ELEUTERIO

PLATAFORMA PARA DIVERSIFICAÇÃO DO USO DE CARTÃO DE TRANSPORTE EM CURITIBA

Trabalho de Conclusão de Curso Trabalho apresentado como requisito parcial para obtenção do título de Bacharel em Engenharia de computação.

Orientador: Keiko Verônica Ono Fonseca

CURITIBA
2017

RESUMO

O intuito do projeto é desenvolver uma prova de conceito de expansão do uso de cartões de transporte público na cidade de Curitiba para possibilitar novas aplicações, bonificações e recompensas em outros serviços públicos. Introduzir gamificação no transporte público busca aumentar o engajamento da população com propostas de recompensa para tornar o sistema de transporte mais atraente. Será proposta uma plataforma focada em segurança, privacidade e escalabilidade, capaz de possibilitar um grande número de novas aplicações. Para exemplificar, o projeto demonstra um caso de uso que envolve bonificar a entrega de lixo reciclável em estações de coleta no cartão de transporte de usuários de Curitiba.

Palavras-chaves: Cidades inteligentes; Sustentabilidade, Gamificação, Internet das Coisas, RFID

ABSTRACT

The purpose of this project is to develop a proof of concept of expanding the use of public transportation cards in the city of Curitiba. The goal is to enable the development of new use cases and reward systems in other public services. Introducing gamification in public transportation increases the population's engagement and makes the system more appealing. A security, privacy and scalability-focused platform will be proposed that will enable a wide array of new applications. To demonstrate the platform concepts, a use case will be developed where a bonus is given to Curitiba public transportation users who take their recyclable trash to a garbage collection station.

Key-words: Smart cities; Sustainability; Gamification; Internet of Things; RFID

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 OBJETIVOS

O objetivo geral do trabalho é desenvolver uma prova de conceito de plataforma que expande o uso de cartões de transporte para novos serviços.

Os objetivos específicos são:

- Especificar, a plataforma para prover funcionalidades de segurança e privacidade.
- Suportar bonificações e ser expansível para outros fins de gamificação.
- Demonstrar através de um caso de uso para a cidade de Curitiba com cartões de transporte da URBS, como os cartões podem ser utilizados para receber bônus em estações de lixo reciclável.

1.2 REQUISITOS

1.2.1 Requisitos Funcionais

RF0 - A Plataforma deve garantir a segurança dos dados em repouso
RF1 - A Plataforma deve garantir a segurança dos dados em trânsito
RF2 - A Plataforma deve ser expansível para aplicações diversas
RF3 - A Plataforma deve ser escalável
RF4 - A Prova de Conceito deve ler um cartão de transporte
RF5 - A Prova de Conceito deve ter mitigação contra falhas
RF6 - A Prova de Conceito deve produzir timestamp para a leitura
RF7 - A Prova de Conceito deve transmitir dados encriptados

1.2.2 Requisitos Não-Funcionais

RNF0 - A Plataforma deve ser acessada via SSH
RNF1 - A Plataforma deve utilizar chaves AES para encriptação simétrica
RNF2 - A Plataforma deve utilizar RSA para encriptação assimétrica
RNF3 - A Plataforma deve utilizar um protocolo de fácil implementação

RNF4 - A Prova de Conceito deve ser portátil

RNF5 - A Prova de Conceito deve armazenar leituras com falhas no envio

RNF6 - A Prova de Conceito deve ter interface intuitiva

RNF7 - A Prova de Conceito deve utilizar apenas bibliotecas open source

1.3 MOTIVAÇÃO

Smart cities preveem a integração de diversos sistemas, como por exemplo sistema de transporte, coleta de lixo, poda de árvores, manutenção de ruas, alocação de equipamentos urbanos públicos, etc. (ZANELLA *et al.*, 2014).Esta integração depende de diversos fatores, incluindo o engajamento da população em usar eficientemente os sistemas disponíveis e as tecnologias que os integram (HARRISON *et al.*, 2010) Este trabalho demonstra como a utilização de cartões de transporte pode ser ampliada, mas também serve de exemplo que cartões RFID podem ser utilizados amplamente na sociedade, como na área de saúde (YAO; CHU; LI, 2012), gestão de ativos (VO *et al.*, 2011) e rastreamento de objetos (NI; ZHANG; SOURYAL, 2011). Também há aplicações para melhorar o transporte público como melhor gerenciamento de vagas.(PALA; INANC, 2007)

Cartões de transporta utilizam a tecnologia RFID para serem lidos e armazenar valores. Em Curitiba, os cartões são usados exclusivamente para pagar passagens de ônibus. Além disso, os cartões do tipo dos usados em Curitiba podem ser *hackeados*, como demonstrado em um estudo de 2008 (GANS; HOEPMAN; GARCIA, 2008), demonstrado como pode ser comprometido em pouco tempo em 2014 (ALMEIDA,). Atualmente há diversos tutoriais ensinando métodos para adulterar cartões Mifare Classic em poucos segundos por diversos aplicativos disponíveis para celular (MEHLMAUER, 2015).

A utilização de cartões de transporte é limitada atualmente, dadas as possibilidades que as tecnologias empregadas proporcionam. A tecnologia atual possibilita a expansão da plataforma utilizada atualmente para que proporcione maior segurança, privacidade e novas aplicações.

Expandir o uso do cartão de transporte e integrá-lo com outros sistemas da cidade é uma opção para modernizar o transporte público e motivar usuários a praticarem hábitos benéficos para a cidade e meio ambiente. A gamificação do sistema visa incentivar usuários a praticarem tarefas tendo como retorno um bônus no seu cartão de transporte associado ou não a um serviço de transporte.

Foi fornecido um hardware, documentado no Apêndice B, para ser utilizado em possíveis aplicações de gamificação. A dificuldade em reutilizar o software criou a necessidade de avaliação de novos equipamentos que atendem melhor a casos de uso como a prova de conceito desenvolvida.

1.4 ESCOPO

Propor uma plataforma capaz de suportar aplicações baseadas no uso do cartão de transporte de Curitiba. Serão especificadas normas de segurança para a plataforma e para os dados consumidos, além de visar proteger a privacidade dos usuários.

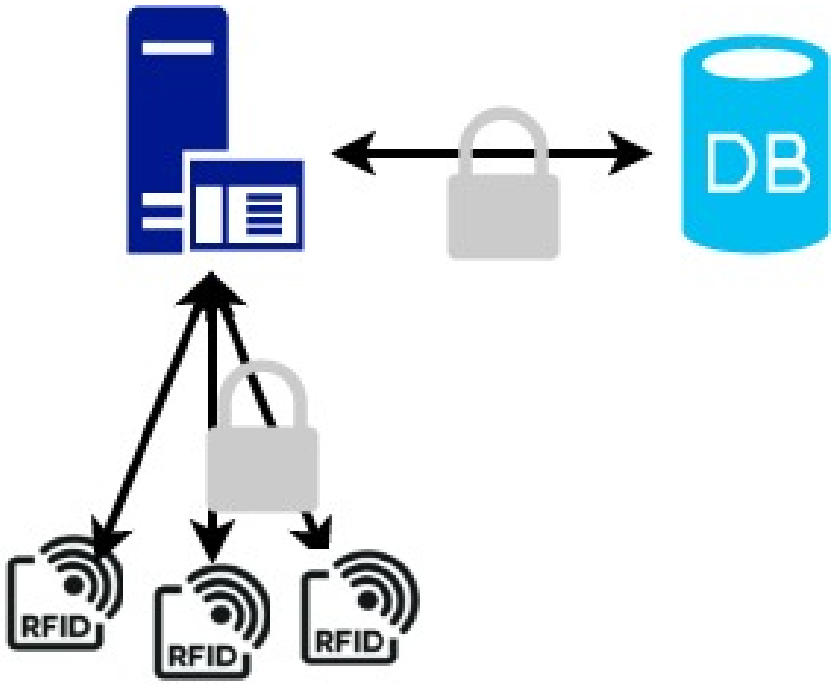

Figura 1 – Diagrama básico da plataforma

A Figura 1 mostra o dispositivo que lê cartões, o servidor de aplicação e o banco de dados. As comunicações são encriptadas. A Figura mostra um servidor de aplicação se comunicando com múltiplos leitores RFID, exemplificando como a plataforma é escalável dado que o o mesmo servidor central pode suportar diversas aplicações com leitores RFID simultaneamente.

Também será desenvolvida uma aplicação para demonstrar um caso de uso, onde um usuário leva seu lixo reciclável para uma estação de coleta e recebe em troca um bônus no seu cartão de transporte. A segurança da plataforma é atingida utilizando conceitos como encriptação e segurança de aplicações, assumindo riscos para garantir a escalabilidade e flexibilidade da plataforma. Tendo em vista que os cartões utilizados são inseguros, o trabalho sugere maneiras de amenizar os riscos associados à utilização de tecnologias legadas e possivelmente comprometidas. A aplicação desenvolvida valida características da plataforma como flexibilidade, simplicidade e segurança. A aplicação irá ler o conteúdo de um cartão de transporte, aceitar *input* do usuário e gerar a mensagem que será enviada ao servidor.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção são descritos alguns conceitos e tecnologias importantes para a compreensão do trabalho

2.1 TECNOLOGIAS

2.1.1 RFID

A tecnologia RFID (do inglês *Radio-Frequency IDentification*) é um método de identificação de objetos através de etiquetas, ou *tags*, através de sinais de rádio. As tags podem, além de servirem como identificadores, armazenar dados. (WANT, 2006)

A grande vantagem de RFID sobre métodos de identificação mais antigos como códigos de barras é a sua capacidade de ler tags à distâncias de mais de 15 metros (FINKENZELLER, 2010), além de conseguir efetuar leituras sem estar alinhado como uma leitura de código de barras requer.

Há inúmeras aplicações para RFID na indústria, como rastreamento de animais (PEREIRA *et al.*, 2008), passaportes (MEINGAST; KING; MULLIGAN, 2007) e levantamento de dados em eventos esportivos.(MOYNIHAN,)

A sua ampla utilização trouxe preocupações nos âmbitos de segurança e privacidade, trazendo a necessidade de padrões para a indústria. ISO/IEC 18000 e ISO/IEC 29167 são padrões para a criptografia, abrangendo autenticação e privacidade. (LOPEZ, 2011). ISO/IEC 20248 define estruturas de dados para garantir a autenticidade das leituras.(PHILLIPS; KARYGIANNIS; KUHN, 2005)

2.1.1.1 Cartões MIFARE

MIFARE é uma marca de chips, propriedade da NXP-Semiconductors, usados amplamente em cartões RFID. De acordo com a MIFARE, mais de 10 bilhões de seus cartões já foram vendidos. (MIFARE,)

Dentre os diversos tipos de cartões MIFARE disponíveis, a URBS utiliza o MIFARE Classic 1k. É o modelo mais antigo da MIFARE, com protocolo de segurança próprio para autenticação e cifras. Este protocolo já foi quebrado e não pode ser considerado seguro.(MEHLMAUER, 2015) O cartão MIFARE Classic 1k atende o protocolo ISO/IEC 14443 Type A, e possui 1kb de armazenamento. (MIFARE,)

2.2 SEGURANÇA E PRIVACIDADE

2.2.1 Encriptação

Para garantir a confidencialidade e integridade das informações transmitidas, é de suma importância que a transmissão de dados seja encriptada. Há duas maneiras de encriptar uma mensagem e enviá-la de maneira que apenas o recipiente possa ver seu conteúdo: encriptação simétrica e encriptação assimétrica. A grande diferença é que encriptação simétrica utiliza uma chave criptográfica compartilhada entre quem envia e quem recebe a mensagem. Encriptação assimétrica assume que os agentes não compartilham uma mesma chave criptográfica. (MAO, 2003)

Outra forma de encriptação é utilizar funções criptográficas de *hash*, utilizadas para verificar a integridade dos dados.

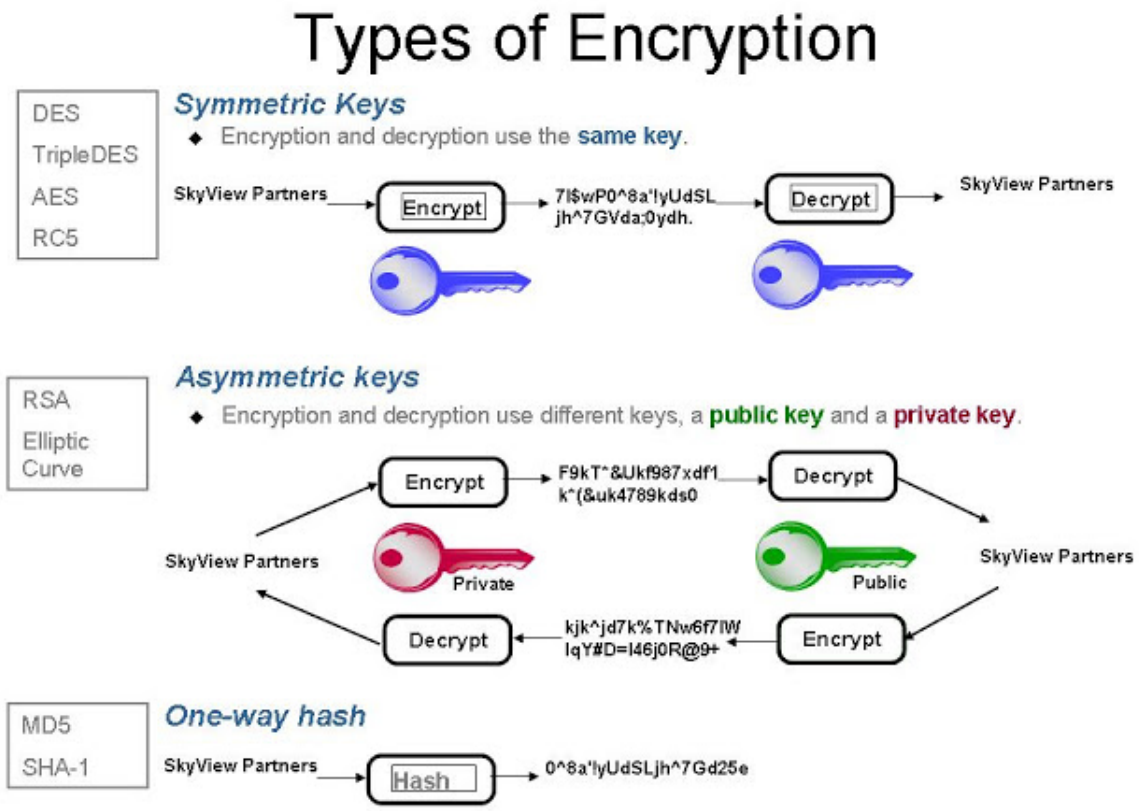

Figura 2 – Formas de encriptação

Fonte: (CHANG,)

A Figura 2 mostra as três formas de encriptação citadas. Para encriptação simétrica são citados os algoritmos DES, TripleDES, AES e RC5, e a figura mostra como a mesma chave (em

azul) é utilizado para encriptar e desencriptar a mensagem.

A Figura apresenta os algoritmos RSA e Curva Elíptica para exemplificar encriptação assimétrica e mostra a chave privada em vermelho e a chave pública em verde. A chave pública deve ser utilizada para encriptar a mensagem e apenas a chave privada pode desencriptar a mensagem.

Por último a figura apresenta dois algoritmos de hash: MD5 e SHA-1 e mostra como essas operações de encriptação não permitem a desencriptação.

2.2.1.1 Simétrica

Encriptação simétrica baseia-se em encriptar e desencriptar uma mensagem com as mesma chaves criptográficas. Ambos os agentes compartilham a chave e a utilizam para encriptar ou desencriptar a mensagem enviada.

Apesar de eficaz e eficiente, há dois grandes riscos de utilizar encriptação simétrica: a força da chave, que deve ser acordada de antemão, e a gerência de chaves. Caso a chave seja roubada ou possa ser comprometida via ataques de força-bruta, toda a comunicação do sistema é comprometida. (MAO, 2003)

A NIST (*National Institute of Standards and Technology*) recomenda a utilização do algoritmos de geração de chaves simétricas AES. (NIST, c)

2.2.1.2 Assimétrica

Encriptação assimétrica se baseia na necessidade de comunicação segura entre dois partidos que não possuem uma chave compartilhada para encriptar seus dados. A solução é utilizar um par de chaves, onde a chave pública de um dispositivo é conhecida por todos os partidos (e potencialmente distribuída por uma Autoridade Certificadora [1]), e a chave privada é mantida apenas pelo dispositivo. Os dados são encriptados com a chave pública e desencriptados com a chave privada.

Por exemplo, se um agente A quer mandar uma informação à um agente B, A encripta os dados utilizando a chave pública de B (conhecida anteriormente ou obtida através de uma AC) e envia à B. B recebe os dados e os desencripta utilizando a sua chave privada, que nenhum outro dispositivo conhece. A Figura 3 demonstra esse exemplo:

[1] uma Autoridade certificadora é uma entidade que emite e distribui certificados digitais, utilizados amplamente para comunicação segura via HTTPS na Internet (DURUMERIC *et al.*, 2013)

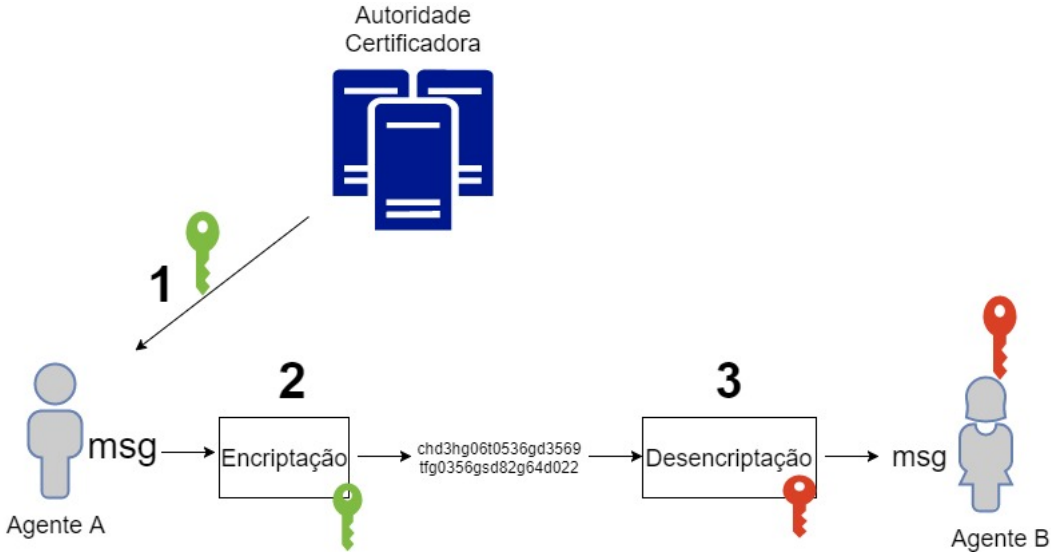

Figura 3 – Encriptação Assimétrica

(MAO, 2003)

1. O Agente A recebe a chave pública (verde) do Agente B, de uma Autoridade Certificadora.
2. O Agente A encripta a sua mensagem utilizando a chave pública.
3. O Agente B desencripta a mensagem utilizando a sua chave privada (vermelha).

A NIST aprova o algoritmo RSA para geração e verificação de assinaturas digitais, e também para estabelecimento de chaves, como utilizado pelo SSL/TLS 1.2. (NIST, c)

2.2.1.3 Hashes

Uma função criptográfica de *hash* é um algoritmo que mapeia dados de qualquer tamanho (a mensagem) em uma *string* de tamanho fixo (o resumo). O resumo não pode ser revertida para o seu conteúdo original de maneira eficiente. A única maneira de reverter o resumo para a mensagem (dado que o algoritmo utilizado seja seguro) é através de força bruta. Uma função de *hash* ideal não permite que isso seja feito de maneira eficiente. Uma função de *hash* apropriada tem 5 características:

1. Determinismo
2. Calcula o resumo rapidamente
3. Ser inviável computacionalmente reverter um resumo à mensagem original
4. Uma pequena mudança na mensagem deve mudar drasticamente o resumo, para que não haja correlações aparentes

5. Ser inviável encontrar duas mensagens que produzam o mesmo resumo

(SILVA, 2003)

2.2.2 Segurança de aplicações

Muitas vulnerabilidades de sistemas não se confinam ao âmbito da rede, segurança física ou sistema operacional (CENTER, 2017) . A maneira como se produz software pode introduzir inúmeras vulnerabilidades de altíssimo risco. Essas vulnerabilidades podem ser evitadas ao aplicar conceitos básicos de segurança de aplicações.

2.2.2.1 Code signing

Ao fazer download de uma aplicação, é imperativo saber se o código que será rodado é exatamente o que se pretende rodar, e que não foi alterado na cadeia logística (MAYNOR *et al.*,). *Code signing* é uma maneira de assinar um código ou executável para assegurar a sua autenticidade e integridade. Outras maneiras de atestar a qualidade do software são auditorias do código, sejam manuais ou automatizadas por ferramentas que fazem análises estáticas e dinâmicas do código. Essas análises encontram bugs e vulnerabilidades no código.(MENTOR,)

Organizações que servem como repositórios de bibliotecas e aplicação como PyPi e Yum garantem a autenticidade do código baixado ao fornecer *hashes* do código original. Com isso o usuário pode comparar uma *hash* calculada do que foi baixado com a original, e garantir que o código é seguro.

A NIST recomenda usar o algoritmo GCM para *Code Signing* utilizando encriptação simétrica, que se aplica a esse trabalho (DWORKIN, 2007). GCM também é utilizado no padrão de segurança Ethernet IEEE 802.1AE (IEEE,), SSH (IGOE; SOLINAS, 2009), TLS 1.2 (SALOWEY; CHOUDHURY; MCGREW, 2008) entre outros.

2.2.2.2 Validação de *input*

Todo dado proveniente do usuário deve ser considerado malicioso. Com isso em mente, a validação de *input* se encarrega de garantir que apenas caracteres relevantes sejam permitidos. É imprescindível que essa filtragem seja feita via *whitelisting* (lista de permissão), ao invés de *blacklisting* (lista de proibição). Validação de *input* é a primeira barreira para evitar vulnerabilidades como *Cross-Site Scripting* e Injeção SQL (KINDY; PATHAN, 2011) (SCHOLTE; BALZAROTTI; KIRDA, 2012). A lista de permissão varia de acordo com a necessidade da apli-

cação, portanto tem que ser avaliada a cada caso. Na prova de conceito desenvolvida, o único input do usuário era através de um teclado numérico virtual, portanto não era possível inserir caracteres que não fossem dígitos.

2.2.2.3 Verificação de tamanho de *buffers*

Toda vez que seja feita escrita em *buffers*, é necessário verificar que os dados sendo escritos não excedam o tamanho do *buffer*. Não tomar essa providência deixa a aplicação vulnerável à *Buffer Overflows*, que podem levar à execução remota de código. Na prova de conceito não havia escrita em buffers.

2.2.2.4 Utilizar bibliotecas seguras

Há diversas bibliotecas antigas que fornecem funções vulneráveis, e é necessário que seja validado se as bibliotecas utilizadas são seguras ou não. Por exemplo, a função *strcpy* da biblioteca *string.c* é vulnerável à *Buffer Overflow*s, e a função *strncpy* da mesma biblioteca não é.

2.2.2.5 Parametrização de *queries*

Toda interação com banco de dados deve ser feita através de *queries* parametrizadas (TAJPOUR; MASSRUM; HEYDARI, 2010). Isso evita com que um agente malicioso possa executar ataques de Injeção SQL. A parametrização de *queries* garante que qualquer *input* do usuário e inserido em uma *query* seja interpretado como texto, e não parte da *query*. A prova de conceito não tinha um banco de dados para efetuar queries, portanto este conceito não se aplica.

2.2.3 Segurança de hardware

Qualquer sistema em que o usuário tenha acesso ao hardware deve assegurar que não seja possível interagir com o embarcado a não ser pela interface prevista pelo design da aplicação. Na prova de conceito, todas as interfaces foram desabilitadas, permitindo apenas as interfaces necessárias. O cartão de memória fica recluso na parte inferior do sistema, evitando fácil acesso.

2.2.3.1 Acesso físico

Nenhum usuário pode ter acesso à portas que não sejam necessárias, pois uma porta aberta (seja USB, serial, Ethernet, etc) introduz diversas vulnerabilidades ao sistema. O acesso à memória física também pode comprometer um sistema pois chaves podem ter sido armazenadas de forma insegura, além de acesso à código fonte e arquivos do sistema operacional.

3 PLATAFORMA

Atualmente a plataforma utilizada pela URBS possibilita apenas que usuários utilizem seus cartões de transporte para pagar passagens de ônibus. Como o crédito é armazenado no próprio cartão, ataques de sobre-escrita ou clonagem de cartões comprometem o sistema.(TECNICA. . . ,). O Apêndice B inclui detalhes sobre o hardware utilizado atualmente pela URBS.

A plataforma proposta visa abrir possibilidades para novas aplicações utilizarem o cartão de transporte da URBS como meio de pagamento ou recebimento de crédito. A plataforma foi projetada para ser versátil e poder suportar uma grande variedade de aplicações. Para tal, foi desenvolvido uma arquitetura que exige muito pouco das aplicações, apenas que uma mensagem de um formato seja criada e padrões de segurança e privacidade sejam obedecidos.

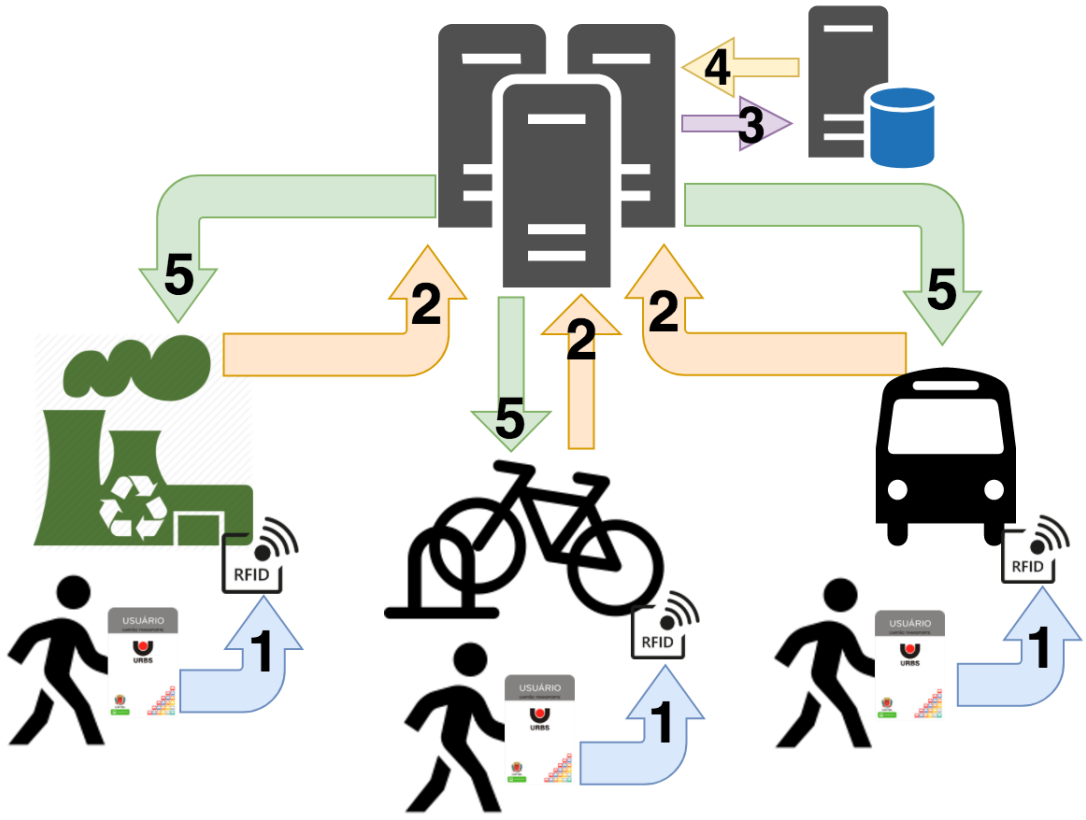

Figura 4 – Diagrama geral da plataforma

A Figura 4 demonstra como diferentes aplicações utilizam a mesma sequência de etapas para realizar suas operações.

1. Os usuários tem seus cartões de transporte escaneados pelos dispositivos de cada serviço.
2. Os dispositivos constroem as suas mensagens e as enviam ao servidor

3. O servidor valida as informações com o banco de dados e processa a operação
4. O banco de dados confirma a validade da operação
5. O servidor envia a resposta para os serviços

3.1 COMPONENTES DA PLATAFORMA

3.1.1 Serviços

Um serviço é uma aplicação desenvolvida sobre a plataforma. Por exemplo a bonificação para coleta de lixo reciclável; aluguel de bicicletas; participação em eventos da cidade. Cada serviço deve

3.1.2 Dispositivos

Cada serviço pode ter diversas instâncias, por exemplo diferentes localidades que recebem lixo reciclável. Cada localidade terá um ou mais dispositivos que dever ser identificados separadamente.

3.1.3 Operações

Cada serviço pode ter mais de uma operação, por exemplo um evento onde as entradas são cobradas a partir do cartão de transporte mas voluntários são recompensados com créditos. As operações de crédito e débito são efetuadas pelo mesmo dispositivo do mesmo serviço e têm que ser diferenciadas.

3.1.4 Mensagem enviada

A mensagem criada pelo dispositivo e enviada à central é composta por:

Tabela 1 – Campos da Mensagem

1	2	3	4	5	6
ID_serviço	ID_dispositivo	ID_operacao	Payload	ID_cartao	Timestamp

3.1.4.1 Campos da mensagem

1. ID_serviço (int): É o ID do serviço, armazenado no banco de dados da central e *hard coded* (ou acessado via API) em cada dispositivo que presta o serviço. Um exemplo seria o ID de um serviço de Aluguel de Bicicletas (serviço 0005), que pode ter várias instâncias que fornecem o mesmo serviço.

2. ID_dispositivo (int): É o ID do dispositivo realizando o serviço. Por exemplo, uma estação de aluguel na Rua 24h tem ID_dispositivo 0001 e outra estação na UTFPR tem ID_dispositivo 0002. Todas tem o mesmo ID_serviço porém ID_dispositivo diferentes.

3. ID_operação (int): Uma aplicação pode ter mais de uma operação. O serviço de aluguel de bicicletas pode debitar do cartão do usuário no momento do aluguel (ID_operação 0001) e creditar um bônus caso o usuário alugue bicicletas por 7 dias seguidos (ID_operação 0002). No banco de dados a operação é uma equação com uma variável. Por exemplo, no banco de dados podemos ter que a Operação 1 do serviço de aluguel de bicicletas é: 5 + 3x, onde x é o número de horas que a pessoa alugou a bicicleta.

4. Payload (int): É a informação utilizada para calcular o débito ou crédito ao final da operação. Pode ter o formato que a aplicação necessitar, e é recomendado que seja enviado em formato JSON ou XML. Por exemplo, se o usuário alugar uma bicicleta por 4 horas, o Payload será {'tempo': 4}.

5. ID_cartão (int): É o ID do cartão RFID escaneado.

6. Timestamp (dd/mm/aaaa hh:mm:ss): É o *timestamp* de quando a operação foi realizada.

De acordo com os exemplos, se um usuário com cartão que tem ID 00101010101, aluga uma bicicleta por 4 horas, na UTFPR, às 14:55 do dia 16/12/2016, a mensagem gerada seria: 0005+0002+0001+4+00101010101+'16/12/2016 14:55:00'

3.1.5 Troca de mensagens

A troca de mensagens entre a aplicação, o servidor de aplicações e o banco de dados segue o diagrama da Figura 5:

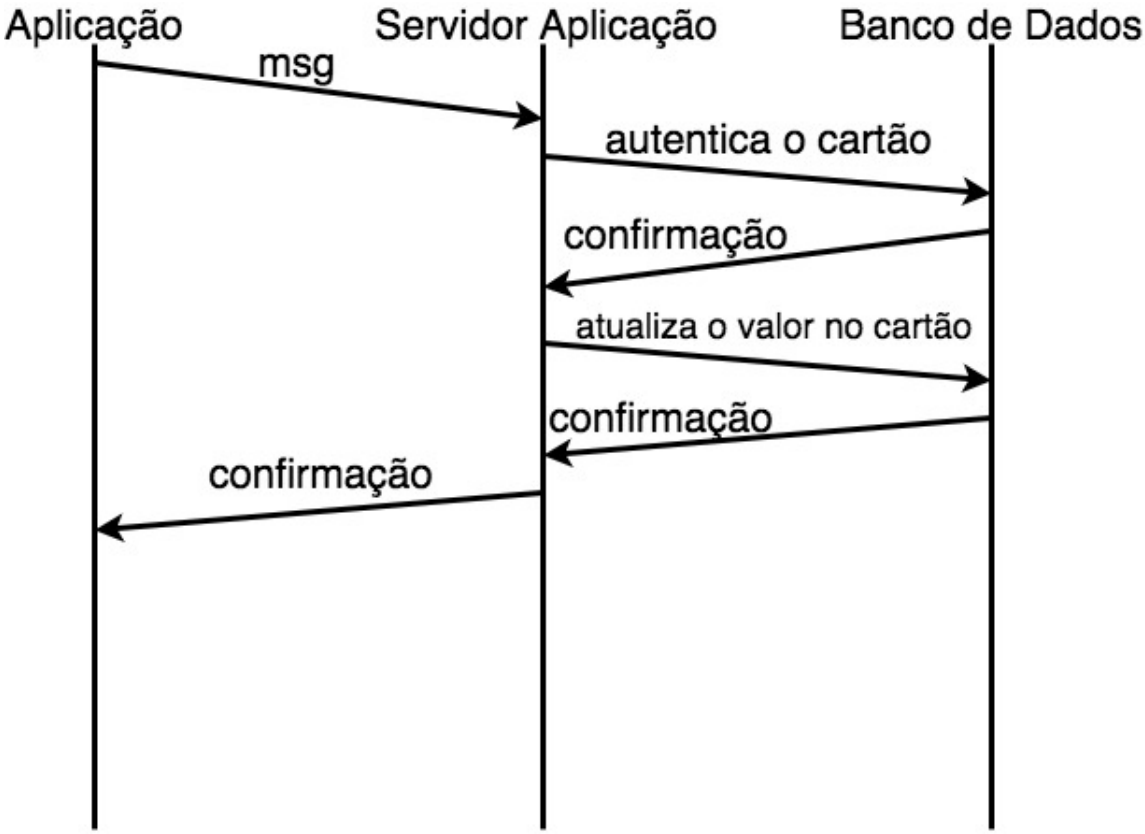

Figura 5 – Diagrama de troca de mensagens

1. A aplicação envia a mensagem para o servidor de aplicação
2. O servidor autentica o cartão no banco de dados
3. O banco de dados confirma a autenticação
4. O servidor envia quanto deve ser acrescentado ou retirado do saldo do cartão
5. o banco de dados confirma a atualização do valor
6. O servidor confirma a operação

3.2 VERSATILIDADE E ESCALABILIDADE

A internet é uma arquitetura cujo sucesso é indiscutível. De acordo com a Internet World Stats, mais de 3.8 bilhões de pessoas têm acesso à internet; mais de metade da população mundial.(STATS,) A sua expansão e ampla adoção se deve ao fato que possibilita inúmeras aplicações em redes que se comunicam por diferentes meios e protocolos. O modelo OSI (STALLINGS, 1987) deixa claro que as suas extremidades (camadas de aplicação e física) possuem muitas possibilidades enquanto as camadas do meio (transporte e rede) não dão aos desenvolvedores opções porém são de fácil implementação.

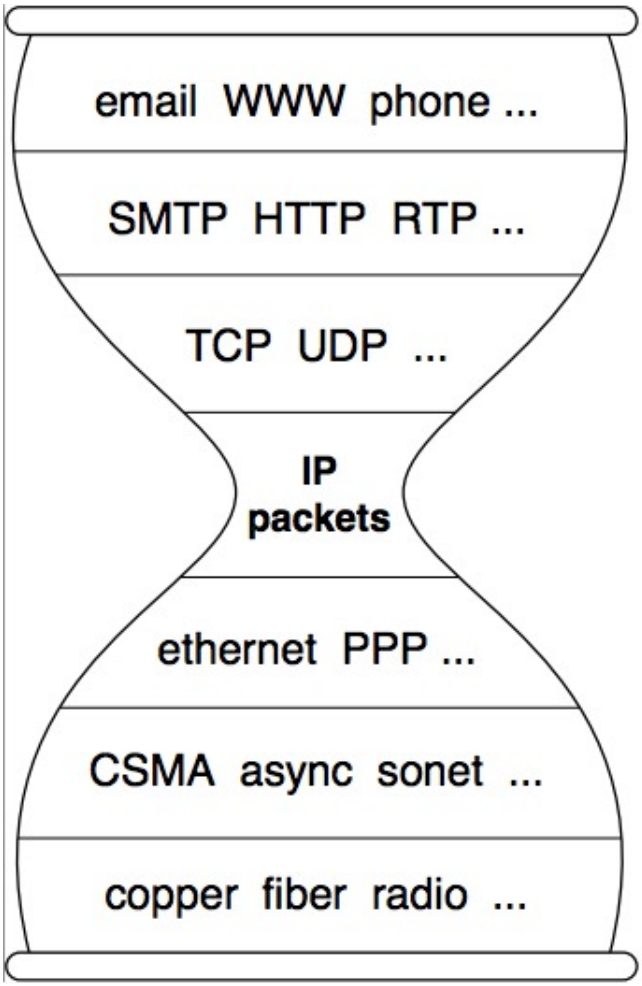

Figura 6 – Arquitetura simplificada da Internet

Fonte: (ZHANG *et al.*, 2014)

A Figura 6 mostra como a Internet é uma arquitetura de cintura fina **(numerar e citar o numero do texto)**. A simplicidade de implementação do protocolo IP possibilita que tantos dispositivos se conectem à Internet, enquanto permite que desenvolvedores possam escolher dentre vários protocolos para suportar as suas aplicações. A restrição de uma arquitetura de cintura fina é a dificuldade de se substituir a camada do meio, fato observado pela enorme dificuldade em substituir o protocolo Ipv4 pelo similar Ipv6 .(CHE; LEWIS, 2010)

A arquitetura proposta pelo trabalho, mostrada na Figura 7 se espelha na arquitetura da Internet ao propor que para fazerem parte da plataforma, dispositivos tem o único requerimento de serem capazes de gerar a mensagem definida. Independentemente do embarcado, sistema operacional ou meio de comunicação, o único requisito funcional é que a mensagem seja criada de acordo com o que é definido por este trabalho. Isto possibilita que a aplicação de coleta de lixo reciclável seja implementado em Raspberry Pi com Debian, enquanto a aplicação de aluguel de bicicletas seja implementado em algum sistema embarcado com RTOS por exemplo. Como não é possível prever as futuras aplicações que uma *smart city* possa ter, é imperativo que a arquitetura seja versátil a ponto de definir o menor número possível de protocolos e requisitos.

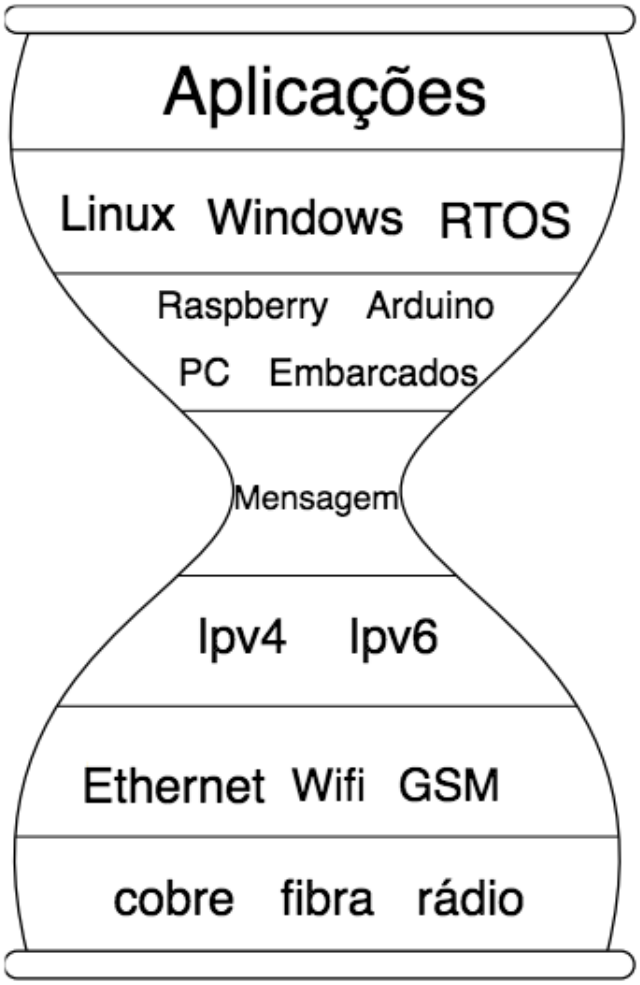

Figura 7 – Arquitetura de cintura fina

3.2.1 Load (carga de processamento) no *backend*

Os cartões Mifare 1k distribuídos pela prefeitura são extremamente vulneráveis e podem ser quebrados em poucos segundos por qualquer celular que possua NFC.(THEEUWES,) Com isso em mente, a plataforma propõe que os cartões sejam utilizados apenas como forma de identificação, podendo ser substituídos ou aliados a formas mais seguras de autenticação como biometria.

Como as aplicações poderão rodar em dispositivos simples, em ambientes não controlados, é importante garantir que todo o processamento e armazenamento de dados seja feito no lado do servidor. Como os cartões são inseguros, será o servidor central que irá armazenar os saldos e validar as operações feitas. Os dispositivos das aplicações serão responsáveis apenas por criar a mensagem e a enviar.

Colocar todo a carga de processamento no *backend* aumenta a capacidade de gerir a segurança da plataforma, além de proporcionar escalabilidade e ter um ponto central de atuali-

zação. A desvantagem dessa abordagem é o risco de prejuízos à disponibilidade da plataforma caso o ponto central seja comprometido. Porém tecnologias modernas de virtualização e gerenciamento de servidores através de *containers* possibilitam que servidores sejam substituídos em pouco tempo.(BANGA; MOGUL, 1999)

4 CASO DE USO: COLETA DE LIXO RECICLÁVEL

Para demonstrar a viabilidade da plataforma, foi desenvolvida uma prova de conceito que aplica os conceitos discutidos. O caso de uso é a coleta de lixo reciclável, onde um usuário do sistema público de transporte de Curitiba pode levar seu lixo reciclável para uma estação de coleta, e em troca receber um bônus no seu cartão de transporte. Isto demonstra como uma aplicação para a plataforma deve ser desenvolvida, além de demonstrar um possível uso para essas tecnologias.

Figura 8 – Diagrama de Caso de Uso

A Figura 8 é um diagrama de caso de uso.

1. O usuário leva seu lixo reciclável à uma estação de coleta, com seu cartão de transporte
2. O usuário entrega seu lixo reciclável e escaneia seu cartão.
3. O usuário recebe um bônus no seu cartão de transporte

4.1 COMPONENTES

4.1.1 Raspberry Pi

O Raspberry Pi é um computador pequeno, de baixo custo e consumo de energia, amplamente utilizado para aplicações embarcadas e de IoT. Foi escolhido o modelo Raspberry Pi 3 Model B, mostrado na Figura 9, pois apesar do seu pequeno tamanho, possui especificações suficientes para inúmeras aplicações, como 2GB de memória RAM, WiFi integrado e diversos mecanismos de I/O.(RASPBERRY,)

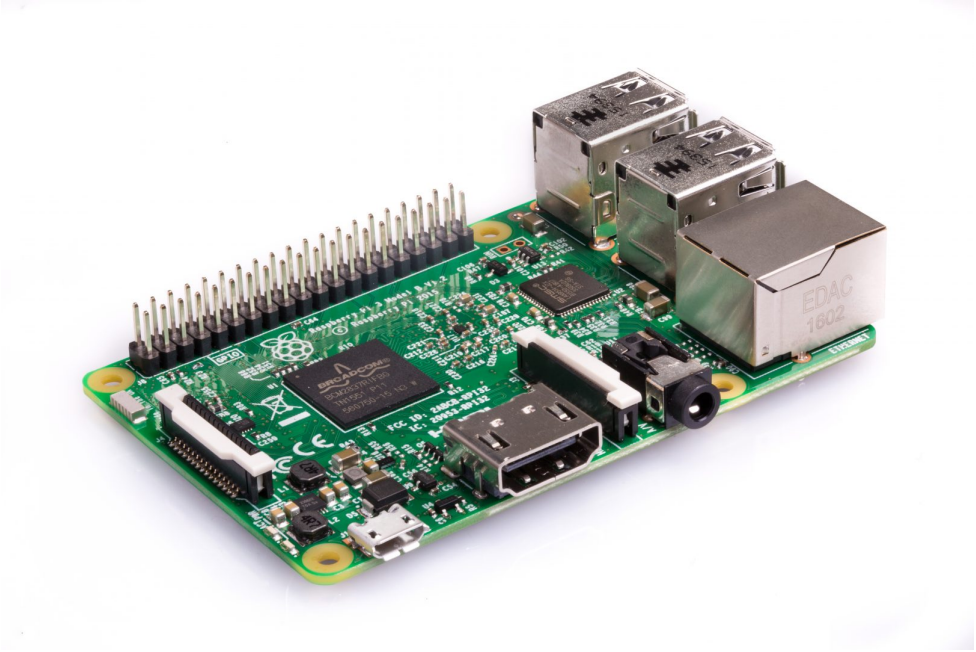

Figura 9 – Raspberry Pi 3 Model B

Fonte: (RASPBERRY,)

4.1.1.1 Sistema operacional

O sistema operacional utilizado foi o Raspbian Jessie. Esse sistema operacional baseado em Debian foi desenvolvido especialmente para o Raspberry Pi e facilita enormemente a interação com dispositivos e gerenciamento de *drivers*. O sistema usa a versão 4.9 do kernel de Linux.(RASPBIAN,). O sistema operacional é fornecido pela Raspberry Pi Foundation e suportado por um time independente de desenvolvedores. (RASPBIAN,) Para a prova de conceito foi carregado o sistema operacional completo, sem nenhuma restrição.

4.1.2 Display

Para interação com o usuário foi escolhido um display touchscreen. Isso facilita o uso do sistema ao não necessitar um mouse e teclado para interagir com a aplicação. O display escolhido foi o Yosoo modelo 101-30-261, com tela de 5 polegadas e resolução de 800x400 pixels. A entrada de vídeo é feita utilizando a saída HDMI do Raspberry Pi, e o controle do touchscreen via I2C. Todos os drivers necessários foram desenvolvidos para Raspbian Jessie. A Figura 10 mostra o Display, e na tela está o Sistema Operacional Raspbian.

Figura 10 – Display touchscreen

Fonte: (HDMI...,)

4.1.3 Leitor RFID

Para fazer a leitura dos cartões RFID, o leitor escolhido foi o PN532, por seu baixo custo e fácil acessibilidade. Ele foi configurado para funcionar em modo Serial. A Figura 11 mostra a parte superior do leitor PN532.

Figura 11 – Leitor RFID

Fonte: (FLOP,)

O leitor opera na frequência de 13.56 MHz, e consegue ler tags com distância de até 10cm. (UM0701-02,)

4.1.4 Adaptar Serial USB

O display touchscreen ocupa as portas de Serial do Raspberry Pi, portanto foi utilizado um adaptador serial-USB PL2303, mostrado na Figura 12, para que a interação com o leitor RFID seja feita por uma das portas USB do computador.

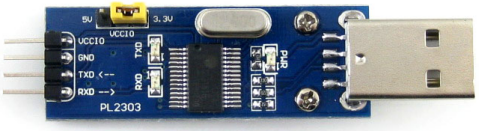

Figura 12 – Adaptador Serial-USB

Fonte: (WAVESHARE,)

4.2 MONTAGEM

O display se conecta à diversas portas do Raspberry Pi: utiliza as portas de GPIO1-GPIO26 para controlar o touchscreen; a porta de HDMI como entrada de vídeo e uma porta USB como fonte de energia.

Como o display ocupa as portas de serial que o leitor RFID necessita, um adaptador serial-USB foi conectado nas portas RX/TX do leitor, e em uma porta USB do Raspberry Pi. O leitor RFID foi acessado através da porta USB0.

A Figura 13 mostra como onde os componentes se encaixam no sistema. A Figura 14 mostra o sistema montado e ligado.

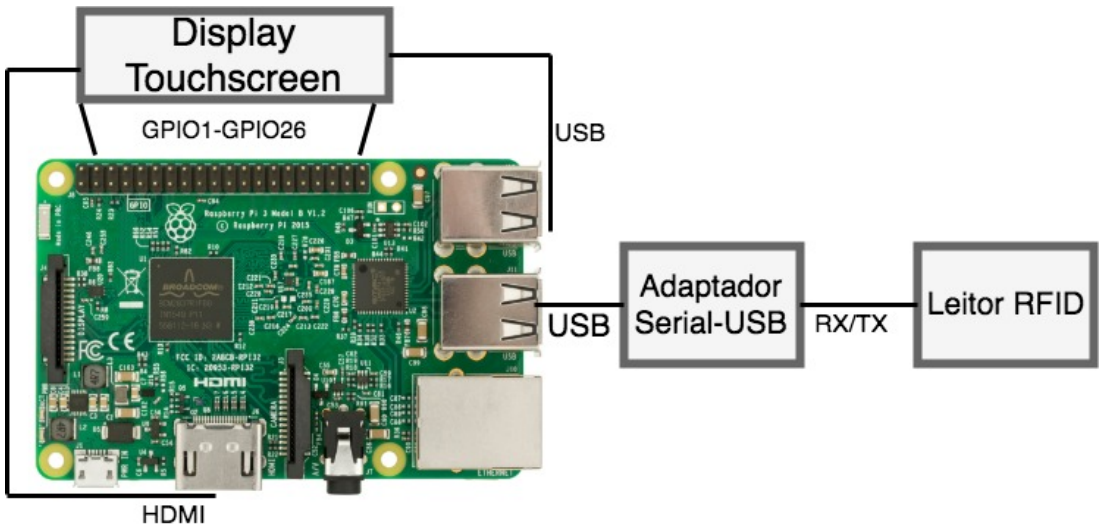

Figura 13 – Componentes do Sistema

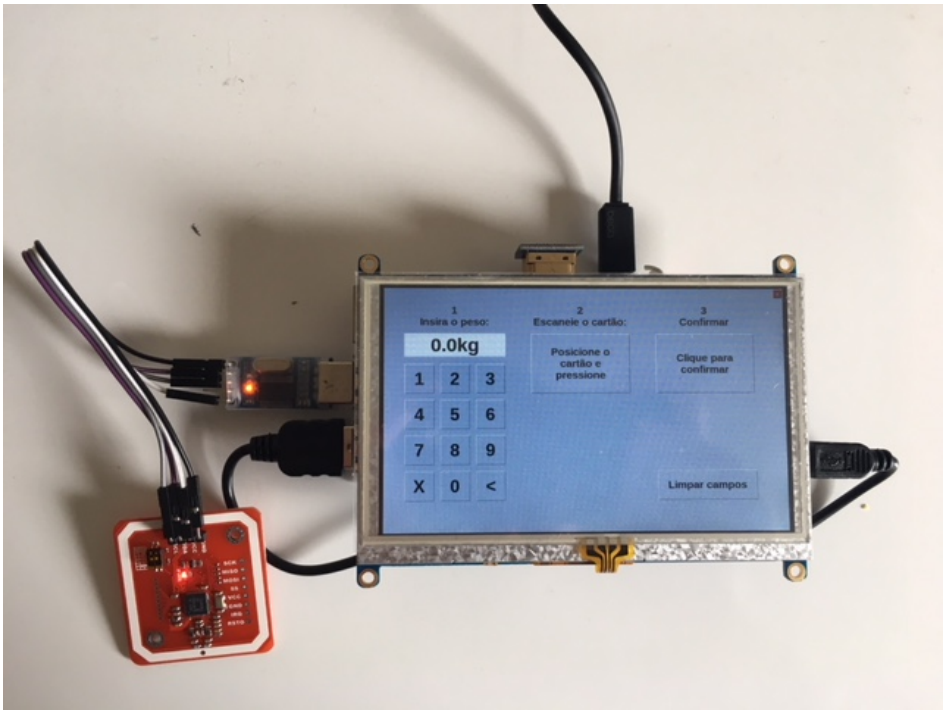

Figura 14 – Sistema Montado

4.3 CONFIGURAÇÃO

A configuração para o Raspberry Pi inclui a instalação do sistema operacional, a habilitação do SSH, e a configuração do display touchscreen. O leitor não precisa de drivers, apenas que a interface USB esteja habilitada, o que é default para o Raspbian Jessie.

Com um MacBook, foi gravado o sistema operacional em um cartão SD, que foi inserido no Rasberry. O Raspberry foi conectado no roteador da mesma rede que o MacBook via Ethernet. Foi configurado um endereço IP estático para o Raspberry, através das configurações do roteador. Também foi alterada a senha padrão para o acesso. O tutorial seguido foi encontrado nos fóruns do raspberrypi.org, escrito pelo usuário fearless. (FEARLESS,)

Para configurar o display touchscreen foi utilizado o tutorial da Circuit Basics, que também fornece links para os drivers necessários, além de recursos para ajustar a precisão do touchsceen. Primeiro deve-se habilitar a interface SPI e então configurar o driver. Após isso, deve-se configurar o módulo de kernel do Raspberry para o display utilizado. As instruções e comandos detalhados podem ser encontrados no tutorial (BASICS,)

4.4 SOFTWARE

O software foi desenvolvido usando a linguagem Python. Foi levado em conta o tamanho da tela do display e a sua precisão ao definir o tamanho das teclas. O Apêndice A inclui referência para o código-fonte comentado.

4.4.1 Interface

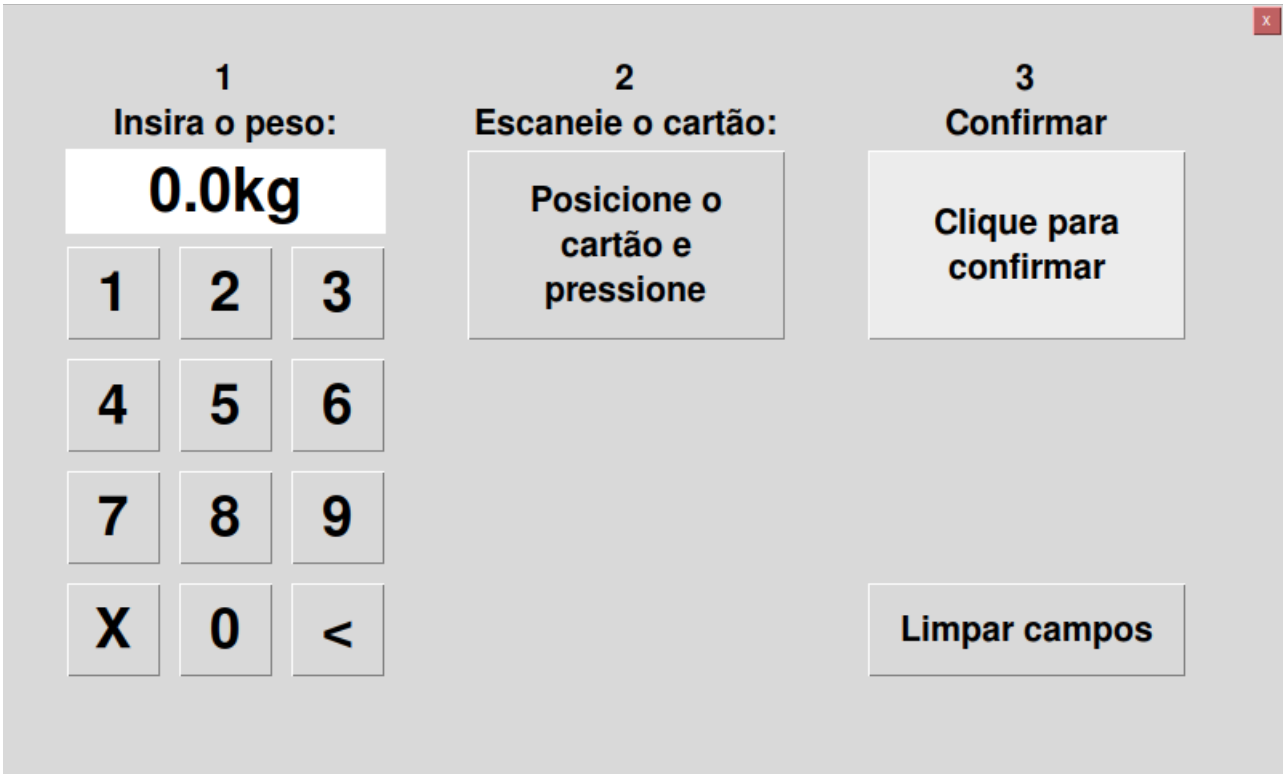

Figura 15 – Interface do software

A interface, mostrada na Figura 15 é dividida em 3 partes:

1. Teclado para inserir o peso do lixo reciclável que o usuário de transporte público levou à estação
2. Botão para ativar o leitor e efetuar a leitura
3. Botão para confirmação e envio. Também há um botão para limpar os campos

4.4.2 Bibliotecas

Tabela 2 – Tabela de bibliotecas utilizadas

Título	Versão	Licença	Função
tkinter	8.6	GPL	Criação da interface gráfica
sys	3.3	PSFL	Função de exit
subprocess	3.6:	PSFL	Funções de threading
datetime	3.2	PSFL	Funções de data e hora
re	2.1	PSFL	Funções de expressões regulares
nfcpy	0.13.3	EUPL	Funções de interação com dispositivos NFC
pycrypto	2.6.1	N/A	Funções de criptografia
socket	2.6	PSFL	Funções de networking
base64	3.4	PSFL	Funções de codificação e decodificação

4.5 DESENVOLVIMENTO

Para desenvolver o software primeiro foi necessário instalar o sistema operacional Raspbian Jessie na partição de boot do computador. A partir disso foi configurado o WiFi e o acesso remoto via SSH. O acesso via SSH foi assegurado ao aletrar a senha do usuário default, não permitindo que agentes sem a senha possam acessar o sistema operacional. O código foi desenvolvido em MacOS, utilizando o editor de texto Visual Studio Code. O código portado para o Raspberry para testes via SCP, e editado com o editor de texto Vim. Como ambos os sistemas são baseados em Unix, não houve problemas de compatibilidade no código nem nas bibliotecas utilizadas.

4.6 RESULTADOS

O software cumpre os requisitos impostos. A interface é amigável e o touchscreen responde satisfatoriamente à interação do usuário. A leitura do cartão é precisa e após 100 testes não foi registrada nenhuma falha. A mensagem é encriptada com chave AES-256 e tem vetor de inicialização randômico, de acordo com normas da NIST. (NIST, c)

É importante citar que o gerenciamento de chaves é de extrema importância ao utilizar encriptação simétrica. A criação, manutenção e rotação de chaves tem que ser feitas de maneira segura para evitar possíveis falhas de segurança. (MAO, 2003) A NIST tem a publicação 800-130, que define práticas para o gerenciamento de chaves (NIST, a)

Não ter o backend implementado impossibilita testes completos do envio das mensagens, porém para validar a prova de conceito e envio de mensagens foi desenvolvido um backend simples. Este servidor para validação recebe mensagens e as imprime como um servidor de eco,

porém não tem capacidade de retornar códigos. Foi possível validar que as mensagens entregues são encriptadas e que agentes maliciosos capturando o tráfego da rede não poderiam interpretar o conteúdos dos pacotes enviados.

O servidor de aplicações deveria ser especificado de acordo com as normas expostas na Seção 2.2, e ser capaz de suportar atualizações. As normas descritas na Seção 2.2 são complementadas pelo framework da NIST para melhorar a cybersegurança de infraestruturas críticas. (NIST, b)

A validação da segurança da plataforma pode ser feita através de um teste de penetração, onde profissionais tentam encontrar e explorar vulnerabilidades em uma aplicação e reportar como podem ser corrigidas. Em uma plataforma onde há diversos componentes de hardware e comunicação com um servidor central, os principais focos de um teste de penetração seriam a segurança de hardware e a encriptação da comunicação com o servidor.(SIBONI *et al.*,) Um possível teste seria interceptar o tráfego e testar a encriptação dos pacotes tentando causar colisão de chaves ou vetores de inicialização inseguros. Como a chave AES criada é de 256 bits, um ataque de força bruta é inviável pois teria que testar um número de chaves igual a 2^{256}, que é da magnitude de $10\hat{}77$. Ataques bem sucedidos de side-channel em chaves AES poderiam ser tentados (OSVIK; SHAMIR; TROMER, 2005) porém a chave foi criada com vetor de inicialização randômico e portanto é segura.

5 CONSIDERAÇÕES FINAIS

A plataforma especificada cumpre requisitos de segurança especificados por agências rígidas como a NIST. Os conceitos utilizados são os mais modernos que a tecnologia atual fornece.

A grande limitação da plataforma é o uso dos cartões MIFARE Classic 1k. Estes cartões não são seguros e portanto foi necessário que o cartão fosse utilizado somente como identificador, e não como armazenamento de fundos. Caso os cartões utilizados fossem uma alternativa segura, como o MIFARE Plus EV1.

Para trabalhos futuros, uma sugestão é o desenvolvimento de uma blockchain para validar as transações. O ledger local pode armazenar apenas as últimas transações e uma central mantém o histórico completo das transações.

Atualmente há inúmeros micro computadores com capacidade para utilizar um sistema operacional como Linux, que pode utilizar SSL/TLS 1.2 para fazer encriptação assimétrica. Essa é uma boa solução para casos onde a gerência de chaves possa ser problemática e onde performance não seja um critério de suma importância.

A aplicação piloto para demonstrar o caso de uso do lixo reciclável na cidade de Curitiba cumpre os objetivos propostos. A aplicação lê cartões e cria mensagens que uma central poderia processar.

Projetos futuros podem explorar a implementação do servidor central da plataforma, e providenciar APIs para que novos serviços possam ser desenvolvidos. É importante fornecer flexibilidade para que a plataforma suporte aplicações diversas.

O trabalho demonstra a factibilidade de uma plataforma que utiliza cartões de transporte para fins maiores do que apenas o pagamento de passagens de ônibus. A modernização de cidades e integração de serviços depende de iniciativas como essa, que permitem que a criatividade dos desenvolvedores não seja limitada pela falta de uma plataforma que suporte serviços diversos.

A CÓDIGO-FONTE

O Código-fonte possui mais de 300 linhas, portanto foi mais apropriado compartilhá-lo via repositório GitHub. O código pode ser encontrado em: <https://github.com/aeleuterio/tcc/blob/master/tcc.py>

B HARDWARE FORNECIDO

A URBS utiliza os equipamentos descritos nesse Apêndice para suas aplicações.

B.1 LEITOR RFID

O leitor RFID é o modelo ER301 da marca Ehuoyan. As suas principais características técnicas são:

- Suporta cartões Mifare/ISO 144443A
- Frequência RF: 13.56MHz
- Velocidade RF: 212kbps
- Tempo de comando: <35ms
- Distância de operação: até 10cm
- Interface: USB para COM virtual
- Baud rate: 115200 bps

A Figura 16 mostra o leitor RFID fornecido.

Figura 16 – Leitor RFID fornecido

B.2 PC INDUSTRIAL

Informações do PC:

- Processador: ATOM N270 CPU 1.6GHz
- Interfaces: VGA/PCI/PCIe/LVDS/Audio/COM/LPT/USB2.0/LAN
- Memória: Suporta até 2GB de memória SDRAM DDR2
- Chipset: Intel 945GSE / ICH7M

B.2.1 I/O

- SIM card socket
- 3x USB 2.0
- 3x COM ports

- LVDS port
- VGA
- Realtek Ethernet
- 2x Mic-in
- 2x Line-out
- HDD LED
- GPIO LED
- Power LED

A Figura 17 mostra a frente do PC e a Figura 18 mostra a parte traseira do PC.

Figura 17 – Parte frontal do PC

Figura 18 – Parte traseira do PC

B.3 LIMITAÇÕES

O Hardware fornecido tem diversas limitações que tornaram necessário o estudo e aquisição de equipamentos mais apropriados para os fins desse trabalho.

O leitor fornecido só possu drivers para versões de Windows anteriores ao Windows 7, e o driver fornecido para Linux não funciona corretamente em versões de Ubuntu mais recentes que 12, até onde foi testado (CO,). Utilizar versões antigas de sistemas operacionais resulta em grande exposição à vulnerabilidades, especialmente quando estes passam a não ser mais suportado e receber atualizações e patches. Este problema ficou evidenciado com o ransomware WannaCry, que prejudicou especialmente versões antigas de Windows (**??**).

Além dos problemas de segurança em utilizar componentes legados, o Software Development Kit (SDK) fornecido pela fabricante do leitor não funciona, e todos os exemplos de código foram testados sem sucesso.

O Hardware também necessitaria de um teclado e monitor, diminuindo a sua capacidade de ser transportado e utilizado em unidades móveis de coleta de lixo reciclável. Além disso, o harware fornecido precisa estar ligado na tomada, sendo que soluções mais apropriadas podem funcionar com baterias portáteis.

REFERÊNCIAS

ALMEIDA, M. **Hacking Mifare Classic Cards**. Disponível em: <https://www.blackhat.com/docs/sp-14/materials/arsenal/sp-14-Almeida-Hacking-MIFARE-Classic-Cards-Slides.pdf>.

BANGA, G.; MOGUL, P. D. andJeffrey C. Resource containers: A new facility for resource management in server systems. 1999.

BASICS, C. **HOW TO SETUP AN LCD TOUCHSCREEN ON THE RASPBERRY PI**. Disponível em: <http://www.circuitbasics.com/setup-lcd-touchscreen-raspberry-pi/>.

CENTER, W. S. T. R. Application security statistics report: The case for devsecops. 2017.

CHANG, P. **3 Main Types of Encryption: Hash, Symmetric, Asymmetric**. Disponível em: <https://medium.com/@peterchang_82818/review-3-main-types-of-encryption-hash-symmetric-asymmetric-tutorial-example-understand-5e57c2903

CHE, X.; LEWIS, D. Ipv6: Current deployment and migration status. **International Journal of Research and Reviews in Computer Science (IJRRCS)**, v. 1, Jun 2010.

CO, E. T. **ER301 SDK**. Disponível em: <http://www.ehuoyan.com/english/SERVICE_display.asp?pid=25&id=228>.

DURUMERIC, Z. *et al.* Analysis of the https certificate ecosystem. 2013.

DWORKIN, M. Computer security. **NIST Special Publication 800-38D**, Nov 2007.

FEARLESS. **Headless setup: no keyboard, display or frustration**. Disponível em: <https://www.raspberrypi.org/forums/viewtopic.php?t=74176>.

FINKENZELLER, K. **RFID Handbook**: Fundamentals and applications in contactless smart cards. [S.l.]: Wiley, 2010.

FLOP, F. **Kit Modulo Leitor RFID NFC PN532**. Disponível em: <https://www.filipeflop.com/produto/kit-modulo-leitor-rfid-nfc-pn532/>.

GANS, G. de K.; HOEPMAN, J.-H.; GARCIA, F. D. A practical attack on the mifare classic. 2008.

HARRISON, C. *et al.* Foundations for smarter cities. **IBM Journal of Research and Development**, v. 54, 2010.

HDMI GPIO 5-Inch. <https://www.amazon.com/gp/product/B00YE1E1UQ/ref=oh_aui_detailpage_o00_s00?ie=UTF8&psc=1>.

IEEE. **802.1AE - Media Access Control (MAC) Security**. Disponível em: <http://www.ieee802.org/1/pages/802.1ae.html>.

IGOE, K.; SOLINAS, J. Aes galois counter mode for the secure shell transport layer protocol. **National Security Agency**, Aug 2009.

KINDY, D. A.; PATHAN, A.-S. K. A survey on sql injection: Vulnerabilities, attacks, and prevention techniques. **Consumer Electronics (ISCE), 2011 IEEE 15th International Symposium on**, Aug 2011.

LOPEZ, T. S. Rfid and sensor integration standards: State and future prospects. **Computer Standards and Interfaces**, v. 33, Mar 2011.

MAO, W. **Modern Cryptography**: Theory and practice. [S.l.]: Prentice Hall Professional Technical Reference, 2003.

MAYNOR, D. *et al.* **The MeDoc Connection**. Disponível em: <http://blog.talosintelligence.com/2017/07/the-medoc-connection.html>.

MEHLMAUER, C. **How to Crack Mifare Classic Cards**. 2015. <https://firefart.at/post/how-to-crack-mifare-classic-cards/>.

MEINGAST, M.; KING, J.; MULLIGAN, D. K. Embedded rfid and everyday things: A case study of the security and privacy risks of the u.s. e-passport. **RFID, 2007. IEEE International Conference on**, Apr 2007.

MENTOR, C. **Performing a Security Audit for your Code: The Basics**. Disponível em: <https://www.codementor.io/learn-programming/performing-security-audit-for-your-code-the-basics>.

MIFARE. **The leading brand of contactless IC products**. Disponível em: <https://www.mifare.net/en/about-mifare/>.

MOYNIHAN, T. **All NFL Players Are Getting RFID Chips This Season**. Disponível em: <https://www.wired.com/2015/08/nfl-players-getting-rfid-chips-season/>.

NI, L. M.; ZHANG, D.; SOURYAL, M. R. Rfid-based localization and tracking technologies. **IEEE Wireless Communications**, v. 18, Apr 2011.

NIST. **A Framework for Designing Cryptographic Key Management Systems**. Disponível em: <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-130.pdf>.

____. **Framework for Improving Critical Infrastructure Cybersecurity**. Disponível em: <https://www.nist.gov/sites/default/files/documents/cyberframework/cybersecurity-framework-021214.pdf>.

____. **Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms**. Disponível em: <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-175B.pdf>.

OSVIK, D. A.; SHAMIR, A.; TROMER, E. Cache attacks and countermeasures the case of aes. 2005.

PALA, Z.; INANC, N. Smart parking applications using rfid technology. **RFID Eurasia, 2007 1st Annual**, oct 2007.

PEREIRA, D. P. *et al.* Model to integration of rfid into wireless sensor network for tracking and monitoring animals. **Computational Science and Engineering, 2008. CSE '08. 11th IEEE International Conference on**, Jul 2008.

PHILLIPS, T.; KARYGIANNIS, T.; KUHN, R. Security standards for the rfid market. **IEEE Security and Privacy**, v. 5, Dec 2005.

RASPBERRY. **Using a mobile phone to clone a MIFARE card**. <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>.

RASPBIAN. **Welcome to Raspbian**. Disponível em: <https://www.raspbian.org/>.

SALOWEY, J.; CHOUDHURY, A.; MCGREW, D. Aes galois counter mode (gcm) cipher suites for tls. **Cisco Systems, Inc.**, Aug 2008.

SCHOLTE, T.; BALZAROTTI, D.; KIRDA, E. Have things changed now? an empirical study on input validation vulnerabilities in web applications. **Computers and Security**, v. 31, May 2012.

SIBONI, S. *et al.* Advanced security testbed framework for wearable iot devices, year=2016, month=Dec, volume=16, keywords=Security and Privacy, Systems Security, Vulnerability Management,. **ACM Transactions on Internet Technology (TOIT) - Special Issue on Internet of Things (IoT): Smart and Secure Service Delivery**.

SILVA, J. E. An overview of cryptographic hash functions and their uses. 2003.

STALLINGS, W. **Handbook of computer-communications standards**: Vol. 1: the open systems interconnection (osi) model and osi-related standards. [S.l.]: Macmillan Publishing Co, 1987.

STATS, I. W. **INTERNET USAGE STATISTICS The Internet Big Picture**. Disponível em: <http://www.internetworldstats.com/stats.htm>.

TAJPOUR, A.; MASSRUM, M.; HEYDARI, M. Z. Comparison of sql injection detection and prevention techniques. **Education Technology and Computer (ICETC), 2010 2nd International Conference on**, Jul 2010.

TECNICA para roubar creditos de passes eletronicos usados em onibus e metros. Disponível em: <https://www.tecmundo.com.br/curiosidade/2525-conheca-mais-nova-tecnica-para-roubar-creditos-de-passes-eletronicos-usados-em-onibus-e-metros-htm>.

THEEUWES, T. **Using a mobile phone to clone a MIFARE card**. Disponível em: <https://timdows.com/projects/using-a-mobile-phone-to-clone-a-mifare-card/>.

UM0701-02.

VO, C. C. *et al.* Radio-mama: An rfid based business process framework for asset management. **Journal of Network and Computer Applications**, v. 34, p. 990–997, May 2011.

WANT, R. An introduction to rfid technology. **IEEE Pervasive Computing**, v. 5, Feb 2006.

WAVESHARE. **PL2303 USB UART Board**. Disponível em: <https://www.waveshare.com/pl2303-usb-uart-board-type-a.htm>.

YAO, W.; CHU, C.-H.; LI, Z. The adoption and implementation of rfid technologies in healthcare: A literature review. **Journal of Medical Systems**, v. 36, p. 3507–3525, Dec 2012.

ZANELLA, A. *et al.* Internet of things for smart cities. **IEEE Internet of Things Journal**, v. 1, Feb 2014.

ZHANG, L. *et al.* Named data networking. **ACM SIGCOMM Computer Communication Review**, v. 44, Jul 2014.