CENTRO UNIVERSITÁRIO UNICARIOCA

ELISSON COELHO PEREIRA
RODRIGO SILVA VIANNA

A UTILIZAÇÃO DE IOT EM UM SISTEMA REMOTO DE CONTROLE DE USO DE COMPUTADORES

RIO DE JANEIRO
2020

ELISSON COELHO PEREIRA
RODRIGO SILVA VIANNA

A UTILIZAÇÃO DE IOT EM UM SISTEMA REMOTO DE CONTROLE DE USO DE COMPUTADORES

Trabalho de Conclusão de Curso (Graduação em Ciência da computação) – Centro Universitário Unicarioca, Rio de Janeiro, 2020.

Orientador: André Luiz Avelino Sobral

RIO DE JANEIRO
2020

FICHA CATALOGRÁFICA

Pereira, Elisson Coelho.
A utilização de IoT em um sistema remoto de controle de uso de computadores. / Elisson Coelho Pereira, Rodrigo Silva Vianna. – Rio de Janeiro, 2020.

44f.

Orientador: André Luiz Avelino Sobral.

Trabalho de Conclusão de Curso (Graduação Superior em Ciência da Computação) – Centro Universitário Carioca, Rio de Janeiro, 2020.

1. IoT. 2. Sensor. 3. Laboratório I. Vianna, Rodrigo Silva. II. Sobral, André Luiz Avelino, prof. orient. III. Título.

CDD 005

ELISSON COELHO PEREIRA
RODRIGO SILVA VIANNA

A UTILIZAÇÃO DE IOT EM UM SISTEMA REMOTO DE CONTROLE DE USO DE COMPUTADORES

Trabalho de Conclusão de Curso (Graduação em Ciência da computação) – Centro Universitário Unicarioca, Rio de Janeiro, 2020.

Banca Examinadora

_____

Professor André Luiz Avelino Sobral - Orientador

Centro Universitário Unicarioca

_____

Professor Alberto Tavares da Silva

Centro Universitário Unicarioca

_____

Professor Sergio Assunção Monteiro

Centro Universitário Unicarioca

RESUMO

Atualmente, com a possibilidade de retomada às atividades práticas relacionadas aos cursos de graduação da universidade, surge a necessidade de utilização dos laboratórios de informática pelos alunos e, para que sejam atendidos os protocolos de segurança de distância mínima entre os estudantes, será necessário um controle mais rigoroso sobre a ocupação dos equipamentos e das instalações. O objetivo deste trabalho é, através do conceito de IoT na interação entre objetos inteligentes (implementados com sensores e outros sistemas digitais) e pessoas, desenvolver um sistema que possibilite informar quais equipamentos estarão disponíveis para uso em cada um dos laboratórios de informática.

Palavras-chave: IoT, sensor, nuvem, arduino, aplicação, laboratório

ABSTRACT

Currently, with the possibility of resuming the practical activities related to the university's undergraduate courses, there is a need for students to use computer labs and, in order to comply with the minimum distance security protocols between students, a control will be necessary stricter on the occupation of equipment and facilities. The objective of this work is, through the concept of IoT in the interaction between intelligent objects (implemented with sensors and other digital systems) and people, to develop a system that makes it possible to inform which equipment will be available for use in each of the computer labs.

Keywords: IoT, sensor, cloud, arduino, application, laboratory

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

A biossegurança é uma abordagem estratégica e integrada para analisar e gerenciar riscos relevantes para a vida e saúde humana, animal e vegetal e os riscos associados para o meio ambiente (WHO, p.1,2010).

Tal abordagem é necessária para prevenir, amenizar ou eliminar os riscos da pandemia da COVID-19 nas atividades presenciais das Instituições de Ensino.

Dentre vários protocolos de biossegurança criados em diversas áreas, o protocolo feito pelo MEC (2020) procura orientar gestores de Instituições de Ensino quanto à tomada de decisão na volta às aulas presenciais. Onde destaca a importância do uso da máscara por todos, verificação de temperatura na entrada de salas e auditórios, a disponibilização uso do álcool em gel 70%, manter ambientes ventilados (janelas abertas), manter limpa as salas na troca de cada turma em salas e auditórios, manter o distanciamento mínimo de 1,5m, entre outras medidas.

> É importante que, antes do retorno das atividades, a Instituição de Ensino realize capacitações com os docentes, técnico-administrativos, prestadores de serviços e colaboradores que estarão em atendimento aos alunos e ao público em geral (MEC, p.17,2020).

Todas as medidas estabelecidas e outras que podem criadas caso sejam necessárias precisam ser seguidas de forma rígida por cada funcionário e estudante, preservando não apenas a sua saúde, mas também a de todos ao seu redor.

Este trabalho, apresenta como medida de biossegurança, o desenvolvimento de um protótipo de monitoramento remoto, com o auxílio de dispositivos IoT.

O conceito de Internet das coisas (IoT), é o modo de como os objetos físicos estão conectados e se comunicando entre si e com o usuário, através de sensores inteligentes e softwares que transmitem dados para uma rede. É uma espécie de um grande sistema nervoso, que possibilita a troca de informações entre dois ou mais pontos.

O foco da IoT é voltado para os equipamentos do dia a dia de um indivíduo, instituições, empresas e até grandes cidades.

1.1 OBJETIVO GERAL

Este trabalho tem como objetivo geral realizar o desenvolvimento de um sistema capaz de informar equipamentos disponíveis para uso em cada um dos laboratórios de informática.

1.1.1 Objetivo Específico

Este trabalho, apresenta como medida de biossegurança, o desenvolvimento de um protótipo de monitoramento remoto, com o auxílio de dispositivos IoT.

1.2 JUSTIFICATIVA

A razão do monitoramento remoto de equipamentos é uma medida extra de segurança, pois evita a circulação desnecessária de funcionários e estudantes nos laboratórios de informática, que pode resultar na redução de riscos de contágio.

1.3 ORGANIZAÇÃO DO TRABALHO

O trabalho é disposto em capítulos, com o objetivo de esclarecê-lo. O capítulo 1 apresenta a introdução e a divisão do trabalho.

O capítulo 2 será apresentada uma visão geral sobre os tópicos que levaram ao entendimento da solução para este trabalho.

O capítulo 3 será apresentado a Análise e Especificação dos requisitos necessários para a implementação da solução do problema.

O capítulo 4 será apresentada a implementação do protótipo e seus detalhes técnicos.

O capítulo 5 será apresentada a conclusão e os resultados obtidos ao longo do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Jeon et al. (2018, p.2) relatam que com o aparecimento da IoT e o acelerado progresso da tecnologia de sensores, a computação ciente de contexto evoluiu de uma variedade de ambientes de computação ciente de contexto (computação ubíqua e ambientes mobile web) para o ambiente de IoT. Isso significa que os sensores que, além de coletar informações, também podem fornecer uma abordagem de vários pontos de vista para um determinado problema usando os dados coletados do sensor.

Essa abordagem é utilizada em uma proposta de desenvolvimento de um sistema de detecção baseado em IoT em ambientes residenciais internos por Jeon et al. (2018).

Ao longo deste capítulo, foram abordadas algumas tecnologias necessárias para o entendimento do desenvolvimento da solução.

2.1 A INTERNET DAS COISAS

> Se tivéssemos computadores que soubessem de tudo o que há para saber sobre coisas, usando dados que foram colhidos, sem qualquer interação humana, seríamos capazes de monitorar e mensurar tudo, reduzindo o desperdício, as perdas e o custo. Saberíamos quando as coisas precisassem ser substituídas, reparadas ou recuperadas e se estavam novas ou ultrapassadas (ASHTON,2009, p.1).

> A interconexão de dispositivos físicos, edifícios, veículos etc. é chamada de Internet das Coisas (IoT). Visa oferecer conectividade avançada entre dispositivos, sistemas e serviços e incluir vários protocolos, domínios e aplicativos para que esses dispositivos possam comunicar uns com os outros e completar tarefas específicas sem qualquer intervenção do usuário (ASHOK e col.,2017, p.2).

Portanto, a IoT interconecta coisas físicas e virtuais de acordo com as tecnologias de informação e comunicação interoperáveis existentes e que evoluem de forma constante.

De acordo com Saint-Exupery (2009), uma “coisa” pode ser definida como uma entidade real/física ou digital/virtual que existe e se move no espaço e no tempo e é capaz de ser identificada. Também relata que as coisas são comumente identificadas por números de identificação atribuídos, nomes e/ou endereços de localização.

> Uma IoT é uma rede que conecta "Coisas" exclusivamente identificáveis à Internet. As “Coisas” têm recursos de detecção / atuação e potencial programação. Por meio da exploração de identificação e detecção exclusivas, as informações sobre a "Coisa" podem ser coletadas e o estado da "Coisa"

> pode ser alterado de qualquer lugar, a qualquer hora, por qualquer coisa (IEEE, p.74, 2015).

Uma vez conectados, objetos criam dados ao transmitirem informações uns aos outros, e esses mesmos dados podem ser analisados e utilizados para tomada de decisões humanas, novos modelos de negócio etc.

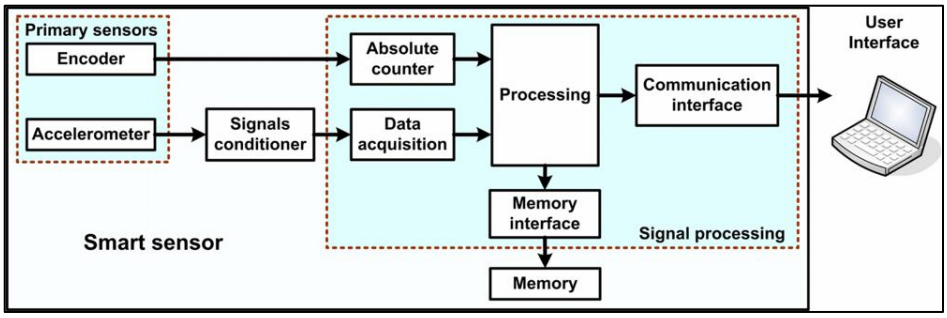
2.2 SENSORES

> As definições comumente usadas para os termos sensor e transdutor devem ser as primeiras na lista de muitos termos que serão definidos. Um transdutor é um dispositivo que converte energia de um domínio para outro, calibrado para minimizar os erros no processo de conversão. Um sensor é um dispositivo que fornece uma saída útil para um mensurando especificado. O sensor é um elemento básico de um transdutor, mas também pode se referir a uma detecção de tensão ou corrente no regime elétrico que não requer conversão (FRANK, 2000, p.2).

A especificação IEEE 1451.2 (1998) define um sensor inteligente como um sensor que fornece funções além das necessárias para gerar uma representação correta de uma quantidade detectada ou controlada.

Um sensor inteligente possui três componentes principais (Figura 01): um sensor que captura dados de um dispositivo ou ambiente, um microprocessador que calcula a saída do sensor via programação e recursos de comunicação que informam à saída do processador ação a ser tomada.

Figura 01: Arquitetura do Sensor Inteligente



Fonte: <https://www.mdpi.com/sensors/sensors-10-04114/article_deploy/html/images/sensors-10-04114f3-1024.png>. Acesso em 01 set. 2020

Outros sensores podem compor um sensor inteligente, como transdutores, transceptores, amplificadores etc.

Algumas características são capazes de definir um sensor inteligente eficaz, como o baixo custo, conexão wireless (já que não é sempre possível ter conexão a fio), pouca ou nenhuma manutenção, realizar pré-processamento de dados com o objetivo de reduzir recursos de nuvem e carga em gateways, entre outras.

A massiva coleta de dados de vários sensores pode ser combinada e correlacionada para gerar soluções de potenciais problemas, como por exemplo, a detecção de falha mecânica através de dados dos sensores de temperatura e vibração. Segundo Accenture (2014), a manutenção preditiva pode economizar até 12% dos reparos, reduzindo a manutenção geral em até 30% e eliminando as perdas em até 70%.

> Indústria 4.0 e o IoT influenciam e demandam um avanço no mercado dos sensores inteligentes de uma forma muito agressiva, onde cada vez mais o mundo necessita de uma otimização de processos, recursos e aumento na rentabilidade 100% nos processos. Os sensores inteligentes são basicamente o coração da indústria 4.0 e IoT; eles têm o papel fundamental na aquisição de dados de forma rápida, eficiente e precisa. (TAKIZAWA, 2019, p. 1)

É ampla a quantidade de diferentes áreas que os sensores inteligentes atendem. São capazes de realizar monitoramento ambiental, monitoramento de inundação e nível de água, controle de tráfego, aplicações nas indústrias, agricultura, transportes e lógística etc.

Há diversos tipos de sensores, como os de movimento, que detectam o movimento em uma determinada área e transforma este mesmo movimento em um sinal elétrico; sensores de temperatura, que permitem detectar a mudança física na temperatura de um fonte específica e converte dados para um dispositivo ou usuário; sensores de nível, que determinam o nível ou quantidade de fluidos, líquidos ou outras substâncias que fluem em um

2.3 COMPUTAÇÃO EM NUVEM

> A computação em nuvem é um modelo para permitir o acesso onipresente, conveniente e sob demanda à rede a um pool compartilhado de recursos de computação configuráveis (por exemplo, redes, servidores, armazenamento, aplicativos e serviços) que podem ser rapidamente provisionados e liberados com esforço mínimo de gerenciamento ou interação do provedor de serviços. Este modelo possui cinco características essenciais, três modelos de serviço e quatro modelos de implantação. (MELL; GRANCE, 2011, pag.2).

2.3.1 Características Essenciais

De acordo com Mell e Grance (2011), um serviço de Computação em nuvem adequado requer estas cinco características denominadas essenciais: serviço self-service, pool de recursos, acesso à rede em banda larga, rápida elasticidade e serviço medido.

2.3.1.1 Serviço Self-Service

Recursos de computação podem ser fornecidos de forma unilateral por um consumidor sem a necessidade interação humana entre usuário e provedor, ou seja, de forma automática.

2.3.1.2 Acesso à rede em banda larga

Todas as funcionalidades estão disponíveis pela rede e são acessíveis por mecanismos padrão, que promovem o uso de plafaformas-cliente heterogêneas (estações de trabalho, celulares, laptops, tablets etc.).

2.3.1.3 Pool de recursos

Em um modelo multi-tenant (multilocatário), vários recursos de computação do provedor são reunidos para atender vários consumidores e de acordo com a demanda de cada um, são concedidos de forma dinâmica os recursos físicos e virtuais. O consumidor não possui controle ou conhecimento sobre a localização exata dos recursos disponibilizados, porém é capaz de determinar a localização em um nível maior de abstração (por exemplo, país, estado ou centro de dados).

2.3.1.4 Rápida elasticidade

Há uma rápida realocação de recursos e, em certos casos, automaticamente, para que as capacidades dos recursos sejam aumentadas ou a liberação deles caso não haja mais necessária a utilização. Os recursos podem ser obtidos em qualquer quantidade e a qualquer momento.

2.3.1.5 Serviço medido

O uso de recursos é controlado e otimizado automaticamente pelos sistemas em nuvem, e efetua a medição de utilização de forma adequada a cada tipo de serviço, como armazenamento utilizado, contas de usuários ativas etc. O monitoramento e controle do uso de recursos deve ser feito de forma transparente, tanto para o fornecedor, quanto para o consumidor dos serviços.

2.3.2 Modelos de Serviço

Os Modelos de Serviço são capazes de admitir a seleção de nível de controle sobre as informações e tipos de serviço que precisam ser fornecidos. De acordo com Mell e Grance (2011), os principais modelos de serviço são: SaaS (Software as a Service), IaaS (Infraestructure as a Service) e PaaS (Platform as a Service).

2.3.2.1 SaaS

> O recurso fornecido ao consumidor é usar os aplicativos do provedor em execução em uma infraestrutura em nuvem. Os aplicativos são acessíveis a partir de vários dispositivos clientes por meio de uma interface de cliente fino, como um navegador da web (por exemplo, e-mail baseado na web) ou uma interface de programa. O consumidor não gerencia ou controla a infraestrutura de nuvem subjacente, incluindo rede, servidores, sistemas operacionais, armazenamento ou mesmo recursos de aplicativos individuais, com a possível exceção de definições de configuração de aplicativos específicas do usuário. (MELL; GRANCE, 2011, pag.2)

O Software é oferecido como um serviço, a configuração e instalação são providas pela nuvem. Logo, o que é comprado pelo cliente é o direito de utilizar o serviço. A utilização é baseada por meio de assinaturas, ou seja, é necessário que o usuário pague um valor significativo para que certos recursos do software sejam liberados a ele dentro de um intervalo de tempo, mensal ou anual.

2.3.2.2 IaaS

> A capacidade fornecida ao consumidor é fornecer processamento, armazenamento, redes e outros recursos de computação fundamentais onde o consumidor é capaz de implantar e executar software arbitrário, que pode incluir sistemas operacionais e aplicativos. O consumidor não gerencia ou controla a infraestrutura de nuvem subjacente, mas tem controle sobre os sistemas operacionais, armazenamento e aplicativos implantados; e possivelmente controle limitado de componentes de rede selecionados (por exemplo, firewalls de host). (MELL; GRANCE, 2011, pag.3).

Aluga recursos de infraestrutura, como servidores, datacenters, hardware, entre outros, para viabilizar a implementação de um serviço em nuvem. A IaaS possibilita escabilidade de recursos. Apenas é pago por aquilo que é utilizado.

2.3.2.3 PaaS

> O recurso fornecido ao consumidor é implantar na infraestrutura de nuvem aplicativos criados ou adquiridos pelo consumidor, criados usando linguagens de programação, bibliotecas, serviços e ferramentas com suporte do provedor. O consumidor não gerencia ou controla a infraestrutura em nuvem subjacente, incluindo rede, servidores, sistemas operacionais ou armazenamento, mas tem controle sobre os aplicativos implantados e possivelmente as definições de configuração do ambiente de hospedagem de aplicativos. (MELL; GRANCE, 2011, pag.2-3)

É um ambiente completo para desenvolvimento e entrega de aplicativos sob demanda. Como a manutenção, atualização e administração da aplicação é feita pelo provedor, a equipe

de desenvolvimento só se preocupa com a codificação.

2.3.3 Modelos de Implantação

Para cada modelo de implantação há diferentes níveis de restrições e acessos. Para que um modelo seja devidamente implantado depende de alguns fatores, como processo de negócio, tipo de informação e nível de visão. De acordo com Mell e Grance (2011) os modelos de implantação são: Nuvem pública, nuvem privada, nuvem híbrida e nuvem comunitária.

2.3.3.1 Nuvem pública

> Nuvem pública refere-se à computação em nuvem no sentido tradicional, em que os recursos são provisionados dinamicamente em uma base de autosserviço de baixa granularidade pela Internet. Esses recursos são provisionados por meio de aplicativos / serviços da web, de um provedor externo que compartilha recursos e cobra do cliente em uma base de computação utilitária de baixa granularidade. (RAJASEKAR; MANIGANDAN, 2015, pag.1241)

A Nuvem pública é composta por um hardware físico e compartilhado, fornecido por um provedor de serviço de nuvem terceirizado disponível por qualquer cliente, pessoa ou empresa, que deseja contratá-lo. É um modelo mais barato, devido aos seus custos operacionais que são diluídos entre diversos clientes atendidos. Todos os recursos estão disponíveis via web e são compartilhados entre todos os usuários. É a mais adequada aos Softwares como Serviços (SaaS). Utilizada em organizações pequenas, médias ou iniciantes.

2.3.3.2 Nuvem privada

> Um termo que é semelhante e derivado do conceito de Rede Privada Virtual (VPN) é aplicado à Computação em Nuvem. A nuvem privada oferece os benefícios da computação em nuvem com a opção de otimizar em segurança de dados, governança corporativa e confiabilidade. (RAJASEKAR; MANIGANDAN, 2015, pag.1241)

A Nuvem privada garante uma segurança robusta. Toda a infraestrutura da nuvem é dedicada ao uso exclusivo de uma empresa, assim os processos "moldados" com precisão à realidade dela. Como consequência, gastos desnecessários de recursos são evitados. É utilizada principalmente em empresas que lidam com dados críticos, confidenciais e estratégicos, como instituições financeiras e governamentais.

2.3.3.3 Nuvem híbrida

> A nuvem híbrida é mais complexa do que os outros modelos de implantação, pois envolve uma composição de duas ou mais nuvens (privada, comunidade ou pública). Cada membro permanece uma entidade única, mas está vinculado aos outros por meio de tecnologia padronizada ou proprietária que permite a portabilidade de aplicativos e dados entre eles. (JANSEN; GRANCE, 2011, pag. 3).

É uma junção de aspectos da nuvem privada, nuvem pública e comunitária, conseguindo trazer um melhor custo-benefício. De acordo com a necessidade do negócio, algumas aplicações podem ser direcionadas à pública, e outras mais críticas à privada. A nuvem híbrida facilita a portabilidade de dados, aplicativos e serviços.

2.3.3.4 Nuvem comunitária

> Uma nuvem comunitária fica entre nuvem pública e privada em relação ao conjunto-alvo de consumidores. É um pouco semelhante a uma nuvem privada, mas a infraestrutura e os recursos computacionais são exclusivos para duas ou mais organizações que têm privacidade, segurança e considerações regulatórias comuns, em vez de uma única organização. (JANSEN; GRANCE, 2011, pag. 3).

Este modelo de implantação é designado a um conjunto limitado de organizações ou funcionários (bancos ou chefes de empresa de comércio). Geralmente os membros da comunidade compartilham recursos semelhantes como segurança, privacidade, desempenho e conformidade. Este modelo pode existir dentro ou fora das instalações e é administrado por alguma empresa da comunidade ou por terceiros.

2.4 COMPUTAÇÃO CIENTE DE CONTEXTO

A Computação ciente de contexto é um paradigma computacional que tem como objetivo apresentar um meio de coletar informações contextuais de dispositivos, entradas capazes de refletir as condições atuais do usuário e do ambiente do qual o usuário e o dispositivo se encontram, levando em consideração as características de hardware, software e comunicação.

De acordo com Dey (2001), contexto é qualquer informação que possa ser utilizada para caracterizar a situação de entidades (pessoa, lugar ou objeto) que sejam consideradas relevantes para interação entre um usuário e uma aplicação (incluindo o usuário e a

aplicação).

2.4.1 Dimensões de Contexto

> O contexto envolve mais do que posição e identidade. A maioria dos sistemas sensíveis ao contexto ainda não incorpora conhecimento sobre o tempo, a história (passado recente ou antigo), outras pessoas além do usuário, bem como muitas outras informações frequentemente disponíveis em nosso ambiente. Embora uma definição completa de contexto seja ilusória, os "cinco Ws" de contexto são um bom conjunto mínimo de contexto necessário. (ABOWD; MYNATT, 2000, pag. 8)

2.4.1.1 Who (Identificação)

Com base na presença de outras pessoas, os seres humanos lembram de eventos passados e adaptam suas atividades. Logo, é necessário que os sistemas forneçam informações contextuais do usuário e de todas as pessoas envolvidas em um ambiente ou atividade.

2.4.1.2 Where (Localização)

É uma dimensão de contexto bastante explorada. Em junção com a dimensão When (Tempo), tem-se o aprendizado da história de movimentos do usuário ao longo do tempo. Adotada, por exemplo, em alguns sistemas de guia turístico.

2.4.1.3 When (Tempo)

As informações contextuais temporais são utilizadas para indexar registros ou indicar por quanto tempo o usuário está em uma determinada localização. A partir dessas informações, é possível interpretar atividades humanas e detectar padrões de comportamento. Por exemplo, monitorar os cômodos de uma casa e verificar em qual deles a pessoa fica por mais tempo, caso ela passe mais tempo na cozinha, pode-se presumir que cozinhar é uma atividade agradável a ela.

2.4.1.4 What (Atividade)

O objetivo é interpretar, com base nas informações, o que o usuário está fazendo. Pode se tornar uma tarefa complexa ao identificar uma atividade específica de um usuário em um determinado momento, em que diversas atividades podem exercidas. Essas informações são obtidas normalmente por sensores.

2.4.1.5 Why (Intenção)

Com base nas informações contextuais, busca entender o motivo da ação do usuário. É desafiador para a Computação ciente de contexto interpretar essas informações que podem caracterizar o estado de uma pessoa.

2.4.2 Tipos de Contexto

> Caracterizamos quatro tipos de informações de contexto imperfeitas. Consideramos a imperfeição em relação a propriedades ou atributos específicos. Esses são aspectos do contexto que podem ser descritos por informações atômicas, como a localização ou atividade de um determinado usuário ou a capacidade de um dispositivo de computação. (HENRICKSEN; INDULSKA, 2004, pag. 1-2)

2.4.2.1 Estático

Tipo de informação contextual fornecida pelo usuário ou administrador, é invariável e altamente precisa.

2.4.2.2 Sentido (Sensed)

Tipo de informação contextual dinâmica gerada por sensores físicos e lógicos, geralmente muda com frequência, e como consequência se torna imprecisa e desatualizada. Como os usuários podem não atualizar com frequência suas informações, geralmente mudam lentamente por necessidade.

2.4.2.3 Perfilado (Profiled)

Tipo de informação contextual dinâmica obtida por usuários na forma de perfis do usuário ou por aplicativos do usuário de forma indireta. Tais informações geralmente são desatualizadas ou incompletas.

2.4.2.4 Derivado (Derived)

Tipo de informação contextual dinâmica que é derivada a partir de outras informações. O processo de derivação pode introduzir erros e imprecisões nas mesmas.

2.5 ARDUINO

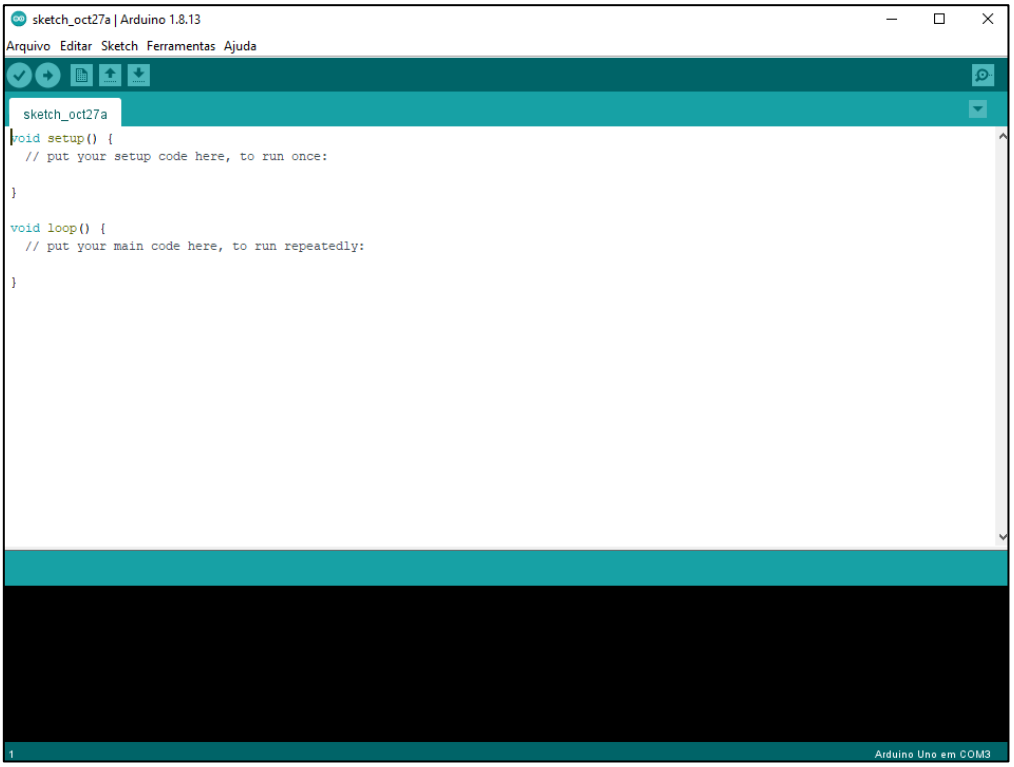
De acordo com Cavalcante et. al (2011, p.2), Arduino é uma plataforma que foi construída para promover a interação física entre o ambiente e o computador utilizando dispositivos eletrônicos de forma simples e baseada em softwares e hardwares livres.

A placa Arduino possibilita a conexão com sensores, módulos e shields. Os módulos são placas menores que podem ter sensores e outros componentes auxiliares, como leds etc. Os shields são placas de circuito pré-construídas que têm a função de estender os recursos da placa Arduino.

> A plataforma utiliza-se de uma camada simples de software implementada na placa, que é um bootloader, e uma interface amigável no computador que utiliza a linguagem Processing, baseada na linguagem C/C++, a qual é também open source. Através do bootloader dispensa-se o uso de programadores para o chip - no caso a família AVR do fabricante ATMEL - facilitando ainda mais o seu uso uma vez que não exige compiladores ou hardware adicional. Neste ambiente de desenvolvimento, são disponibilizadas bibliotecas que permitem o interfaceamento com outros hardwares, permitindo o completo desenvolvimento de aplicações simples ou complexas em qualquer área (SOUZA et al, 2011, pag. 2).

Cada programa é composto por dois blocos, setup() e loop() (Figura 02). O bloco setup() contém as variáveis de inicialização, como a porta utilizada para entrada ou saída, a taxa de transferência em bits por segundo (baud rate) etc. O bloco loop(), como o próprio nome já indica, é responsável por criar um laço de repetição em uma estrutura de comandos, de forma contínua ou até que o Arduino pare de enviar/receber dados.

Figura 02: Arduino IDE



Fonte: Autor, 2020

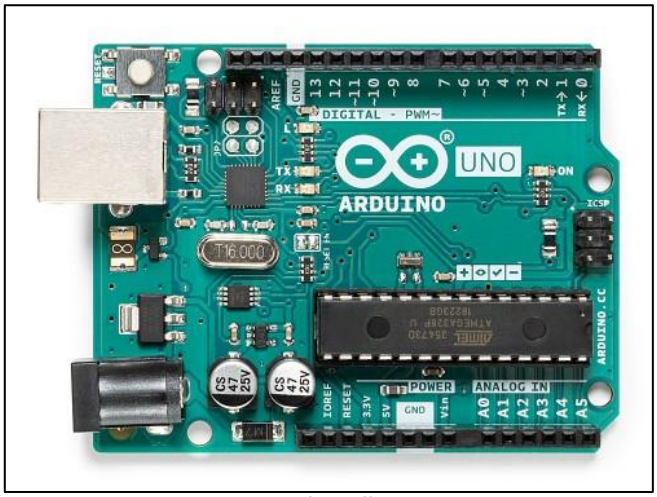
2.5.1 Arduino Uno

A placa Arduino Uno (Figura 03) é baseada no microcontrolador ATmega328, com uma velocidade de clock de 16MHz. A alimentação da placa é por meio da conexão USB ou conector de alimentação externa (recomendável utilizar de 7 a 12 volts). Há um botão de reset, cuja função é reiniciar o Arduino Uno.

Possui 14 portas digitais, das quais 6 podem ser usadas como PWM (3,5,6,9,10,11) através da função analogWrite(). As portas PWM também podem simular portas analógicas. As portas RX (1) e TX (0) são utilizadas para entrada e saída de dados para comunicação serial. Há 6 portas analógicas (A0, A1, A2, A3, A4, A5), são utilizadas para entrada de dados e é comum que se comuniquem com sensores.

A programação da placa Arduino Uno é feita através de comunicação serial, por conta de o microcontrolador vir programado com bootloader. O que é vantajoso para o programador, tornando desnecessária a gravação do binário na placa.

Figura 03: Arduino Uno



Fonte: <https://store-cdn.arduino.cc/usa/catalog/product/cache/1/image/500x375/f8876a31b63532bbba4e781c30024a0a/a/0/a000066_front_8.jpg.>
Acesso em 27 out. 2020

3 ANÁLISE E ESPECIFICAÇÃO

Neste capítulo, serão apresentadas as informações ao respeito da aplicação a partir do levantamento de requisitos e sua análise.

3.1 LEVANTAMENTO DE REQUISITOS

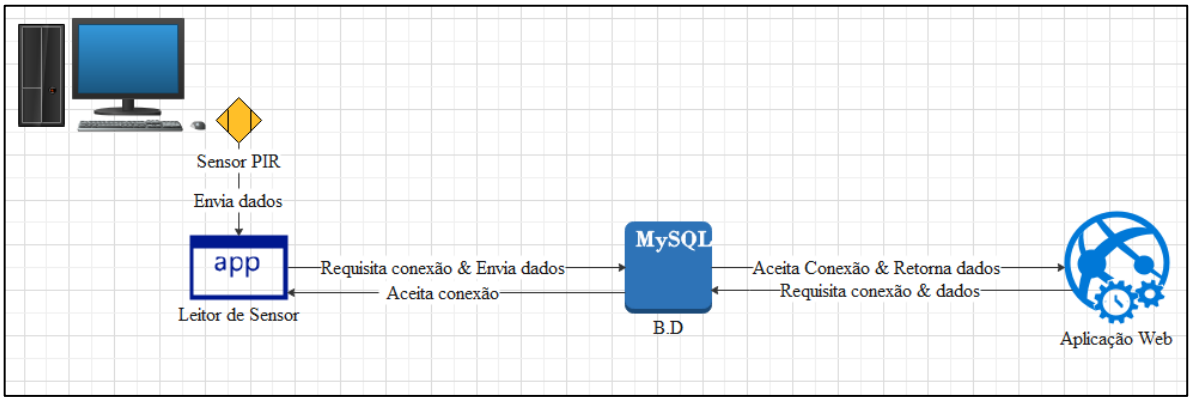
Os requisitos necessários para a aplicação responsável por monitorar os equipamentos estão descritos a seguir.

1. Atualizar o status de disponibilidade (Disponível/Ocupado) do computador de acordo com a mudança de estado do sensor de presença.
2. Para testar o status “Ocupado” do equipamento, predefinir um tempo de espera ao sensor de presença, passado o tempo ele manterá o status “Ocupado” ou caso confirme a ausência do usuário mudará para o status “Disponível.
3. Dado o distanciamento social de 1,5m exigido para cada pessoa, representar em um layout a posição física de cada equipamento no laboratório de informática.

3.1.1 Descrição do Problema

O problema a ser tratado é acentuar a ideia de IoT em um laboratório de informática, onde será verificada a disponibilidade dos equipamentos (computadores) para os usuários (alunos). Esta implementação tem como principal fator o sensor de presença, que ficará próximo ao computador, verificando de forma constante, com o objetivo de gerar informação de disponibilidade. Essa informação estará disponível para acesso à recepção da universidade. A Figura 04 representa a arquitetura proposta da aplicação.

Figura 04: Arquitetura proposta da aplicação



Fonte: Autor, 2020

3.1.2 Requisitos Funcionais

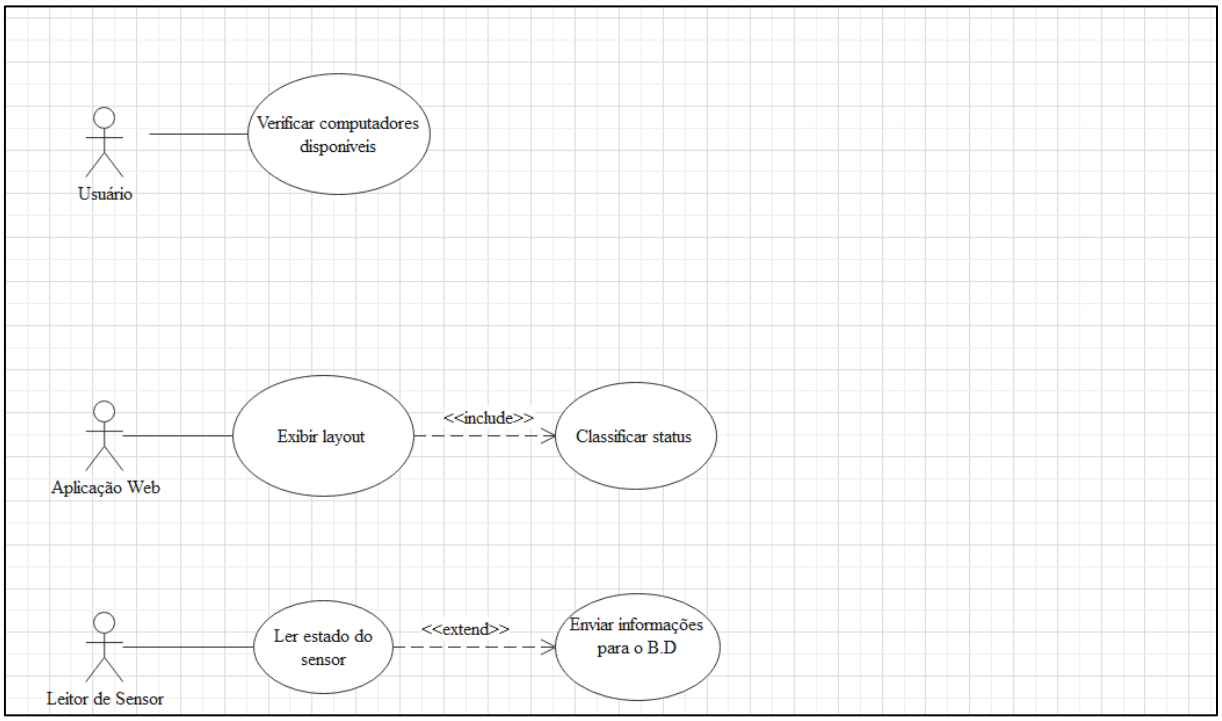
- O sistema deve exibir um layout capaz de representar a posição física de todos os equipamentos (computadores) em cada um dos laboratórios de informática**.**
- O sistema deve classificar o status de disponibilidade através de cores, verde para “Disponível” e vermelho para “Ocupado”.
- O sistema deve ler o estado do sensor de presença.
- O sistema deve atualizar automaticamente o status dos computadores, de acordo com a mudança de estado do sensor de presença.

3.1.3 Requisitos Não Funcionais

- O sistema deve atualizar automaticamente a página web do layout do laboratório de informática a cada 5 segundos.
- O sistema deve iniciar de forma rápida.
- O sensor de presença deve ter um tempo de espera de 1 minuto para manter o estado de “Ocupado” ou mudar para o estado de “Disponível”.
- O sistema deve ser desenvolvido em ASP.NET.

3.2 DIAGRAMA DE CASO DE USO

Figura 05: Diagrama de Casos de Uso



Fonte: Autor, 2020

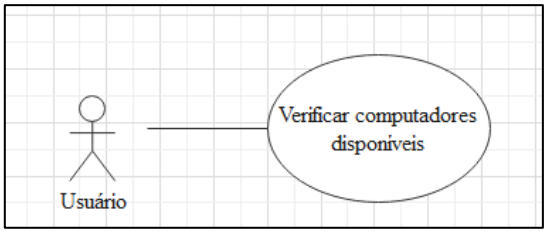
Lista de Casos de Uso

1. Verificar computadores disponíveis
2. Exibir layout
3. Ler estado do sensor

3.3 ESPECIFICAÇÃO DE CASO DE USO

3.3.1 Caso de Uso UC1

Figura 06: Caso UC1



Fonte: Autor, 2020

1. Objetivo

- Visualizar no layout os computadores disponíveis (classificados em cor verde).

2. Atores

- Usuário

3. Pré-condições

- Acessar a página web do layout do laboratório de informática.

4. Evento Inicial

- Iniciar a aplicação web.

4. Fluxo Principal

a) A aplicação web é iniciada.

b) O usuário acessa a página web do layout do laboratório de informática.

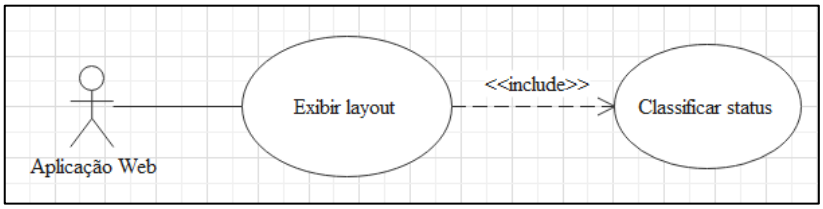
c) O usuário visualiza os computadores disponíveis (cor verde).

5.Casos de Teste

- Cancelar o caso de uso.

3.3.2 Caso de Uso UC2

Figura 07: Caso de Uso UC2



Fonte: Autor, 2020

1. Objetivo

- Exibir a página web que contém o layout do laboratório de informática.

2. Atores

- Aplicação Web

3. Pré-Condições

- Iniciar a aplicação web

4. Fluxo Principal

a) A aplicação web é iniciada.

b) A aplicação web exibe o layout na página web, classifica o status de cada computador (cor verde para “Disponível” e cor vermelha para “Ocupado”).

5. Fluxo de Exceção

E1 – Desconexão com o B.D

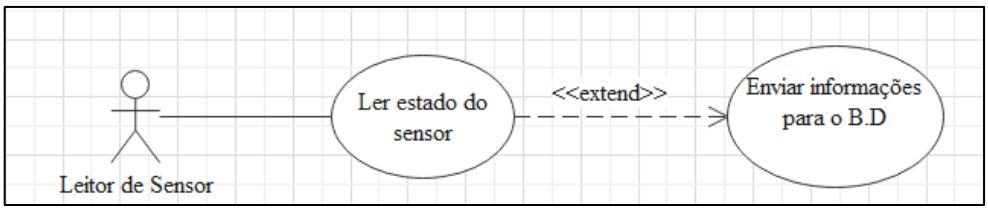
b) A aplicação web não está conectada ao B.D, logo nenhuma informação é retornada.

6. Casos de Teste

- É verificado constantemente a página web para eventuais alterações nos status dos computadores.
- Cancelar caso de uso.

3.3.3 Caso de Uso UC3

Figura 08: Caso de Uso UC3



Fonte: Autor, 2020

1. Objetivo

- Ler o estado do sensor de presença.

2. Atores

- Leitor de Sensor

3. Pré-Condições

- Habilitar o sensor de presença e iniciar a aplicação leitora de sensor.

4. Fluxo Principal

a) O leitor de sensor é iniciado.

b) O leitor de sensor verifica mudança de estado.

c) O leitor detecta mudança de estado.

d) O leitor envia informações de mudança de estado do sensor para o B.D e as informações são atualizadas.

6. Fluxo Alternativo

A1 – Não há mudança de estado.

b) O leitor não detecta mudança de estado.

7. Fluxo de Exceção

E1 – Erro de leitura do sensor

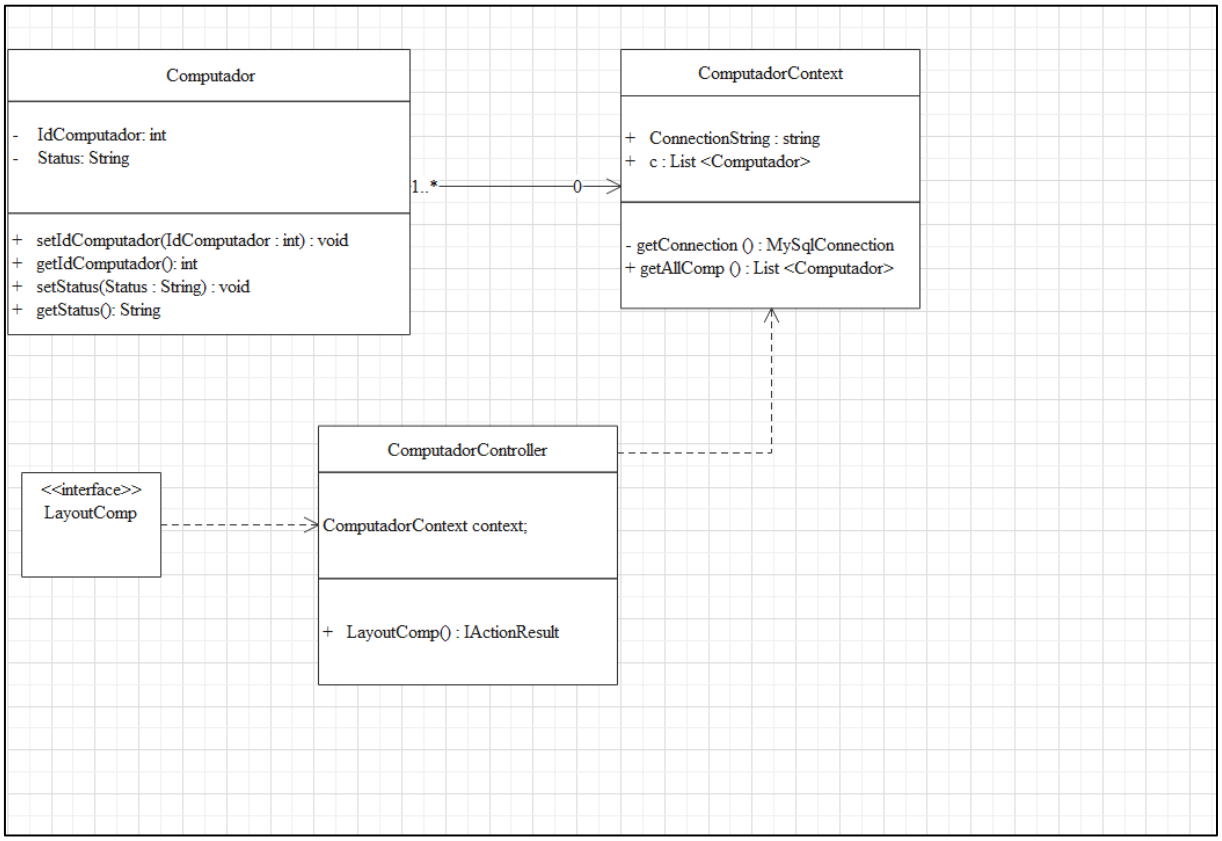
b) O leitor não detecta a saída do sensor de presença.

8. Casos de Teste

- É verificado constantemente se há a mudança de estado de sensor.
- Cancelar caso de uso.

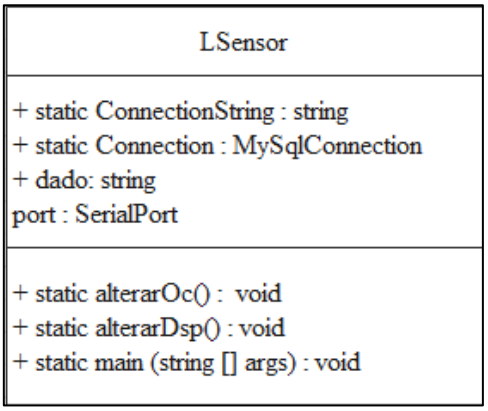
3.4 DIAGRAMA DE CLASSE

Figura 09: Diagrama de Classe da Aplicação de Monitoramento



Fonte: Autor, 2020

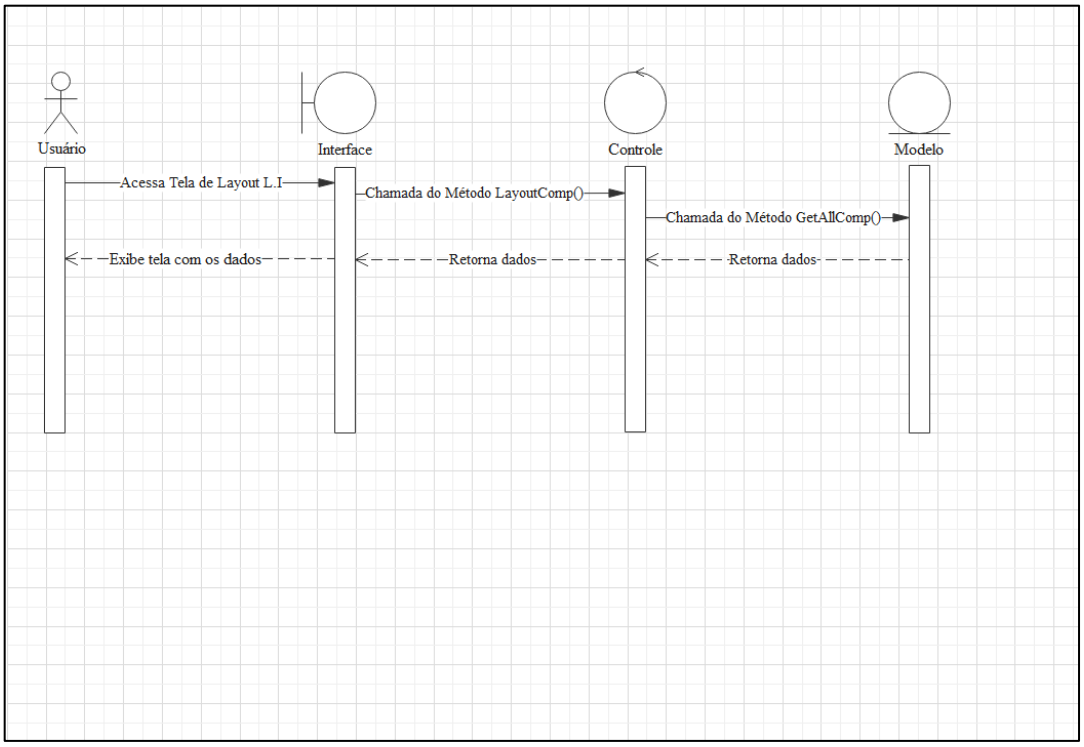
Figura 10: Diagrama de Classe da Aplicação Leitora do Sensor



Fonte: Autor, 2020

3.5 DIAGRAMA DE SEQUÊNCIA

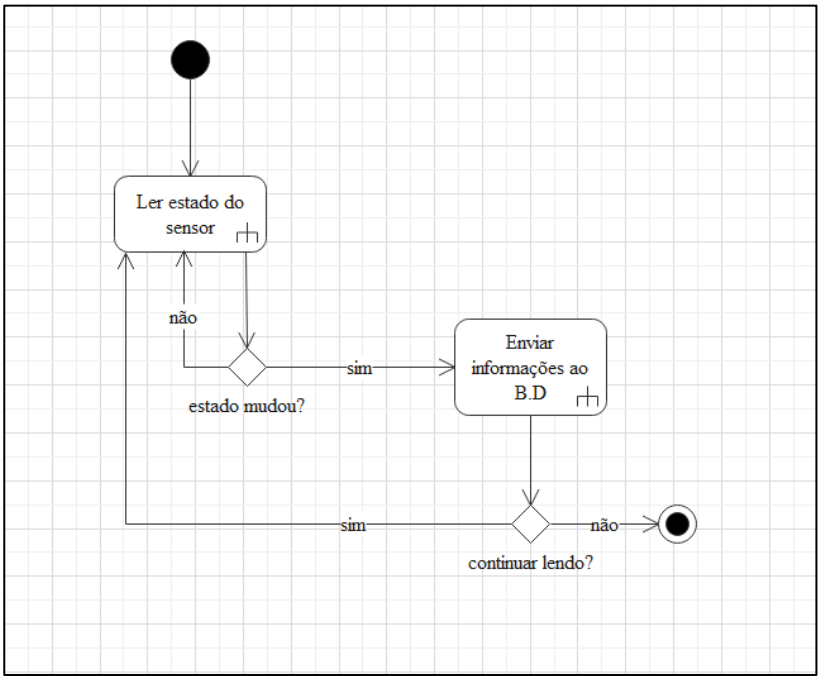
Figura 11: Diagrama de Sequência de Exibição do Layout



Fonte: Autor, 202

3.6 DIAGRAMA DE ATIVIDADE

Figura 12: Diagrama de Atividade do Leitor de Sensor



Fonte: Autor, 2020

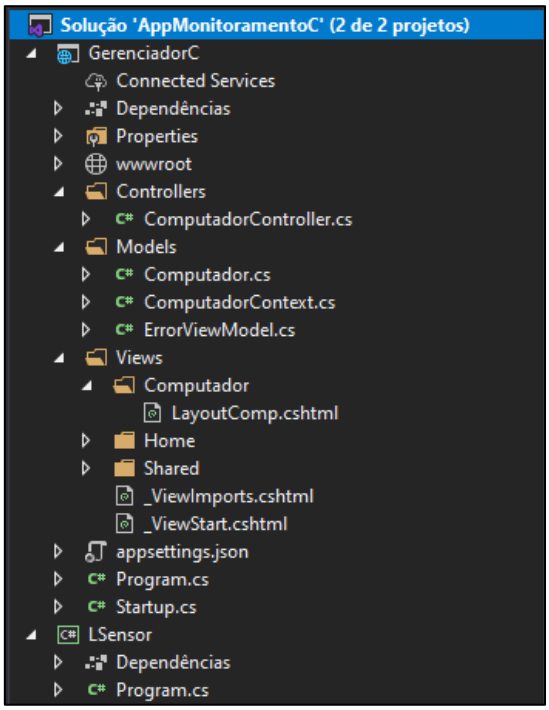
4 IMPLEMENTAÇÃO

Para a implementação das aplicações foi utilizado o ambiente de desenvolvimento Visual Studio. A aplicação **GerenciadorC** é baseada no modelo MVC voltado ao desenvolvimento web, utiliza a plataforma ASP.NET. Tanto a aplicação **GerenciadorC** quanto a aplicação **LSensor** foram desenvolvidas em C#.

4.1 ORGANIZAÇÃO DO PROJETO

O projeto **AppMonitoramentoC** é composto pela aplicação web **GerenciadorC** e a aplicação **LSensor**. A Figura 13 exibe a organização das duas aplicações no projeto.

Figura 13: Organização do Projeto



Fonte: Autor, 2020

4.1.1 GerenciadorC

A aplicação é composta pelas classes de Modelo: ***Computador*** e ***ComputadorContext***, e a de Controle: ***ComputadorController***. Além das classes, há também a página ***LayoutComp***.

Computador – Classe responsável pela criação de objetos.

ComputadorContext – Classe que lista objetos de acordo com os métodos, também realiza a conexão com o Banco de Dados.

ComputadorController – Classe responsável por enviar os dados para a página web.

4.1.1.1 Classes

A Classe ***Computador*** (Figura 14) é responsável pela criação de objetos. Possui os atributos **IdComputador** e **Status**, além de seus métodos *getters* e *setters*.

Figura 14: Classe Computador

```
12 referências
public class Computador
{

    [Required]
    6 referências
    public int IdComputador { get; set; }

    3 referências
    public String Status { get; set; }

}
```

Fonte: Autor, 2020

A classe ***ComputadorContext*** lista os objetos de acordo com os métodos, também realiza a conexão com o Banco de Dados.

O método *GetAllComp*() (Figura 15) é responsável por listar os objetos de acordo com a consulta na Base de Dados.

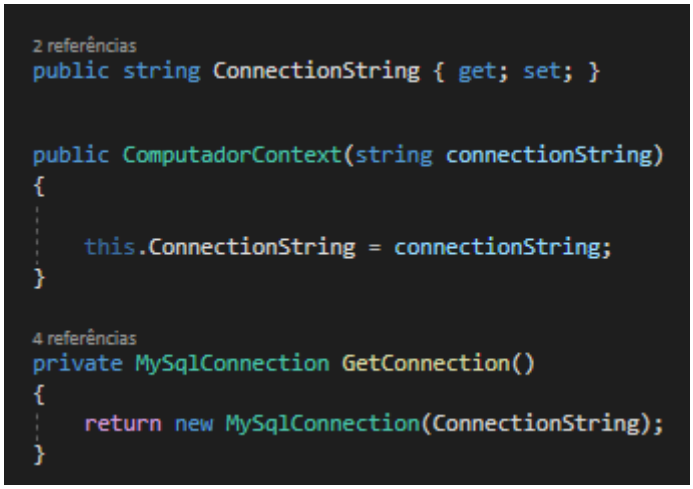
Figura 15: Método GetAllComp()

```
1 referência
public List<Computador> GetAllComp()
{
    List<Computador> c = new List<Computador>();
    using (MySqlConnection conn = GetConnection())
    {
        String sc = "Select * from Computador";
        try {
            conn.Open();
            MySqlCommand cmd = new MySqlCommand(sc, conn);
            using var reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                c.Add(new Computador()
                {
                    IdComputador = Convert.ToInt32(reader["Id"]),
                    Status = reader["Status"].ToString()
                });
            }
        } catch (Exception e) {
            throw e;
        }
        return c;
    }
}
```

Fonte: Autor, 2020

O método *GetConnection*() (Figura 16) é responsável por realizar conexão com o SGBD MySQL, o atributo **ConnectionString** (Figura 16) contém as informações da Base de Dados utilizada, como seu nome, servidor, porta, usuário e senha.

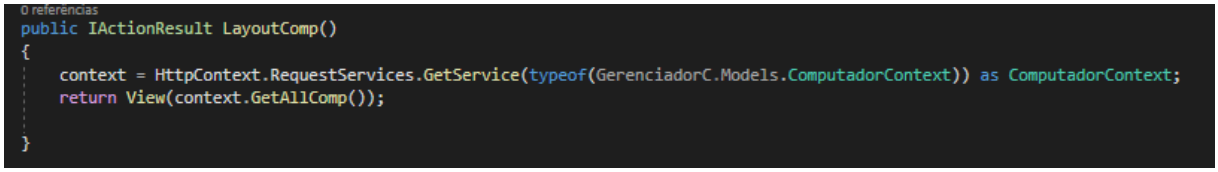
Figura 16: Conexão ao Banco de Dados



Fonte: Autor, 2020

A classe ***ComputadorController*** é responsável por enviar os dados para a página web. O método *LayoutComp()* (Figura 17) realiza a consulta dos dados ao método *GetAllComp()* da classe ***ComputadorContext***. Os dados requisitados são retornados para a página web.

Figura 17: Método LayoutComp()

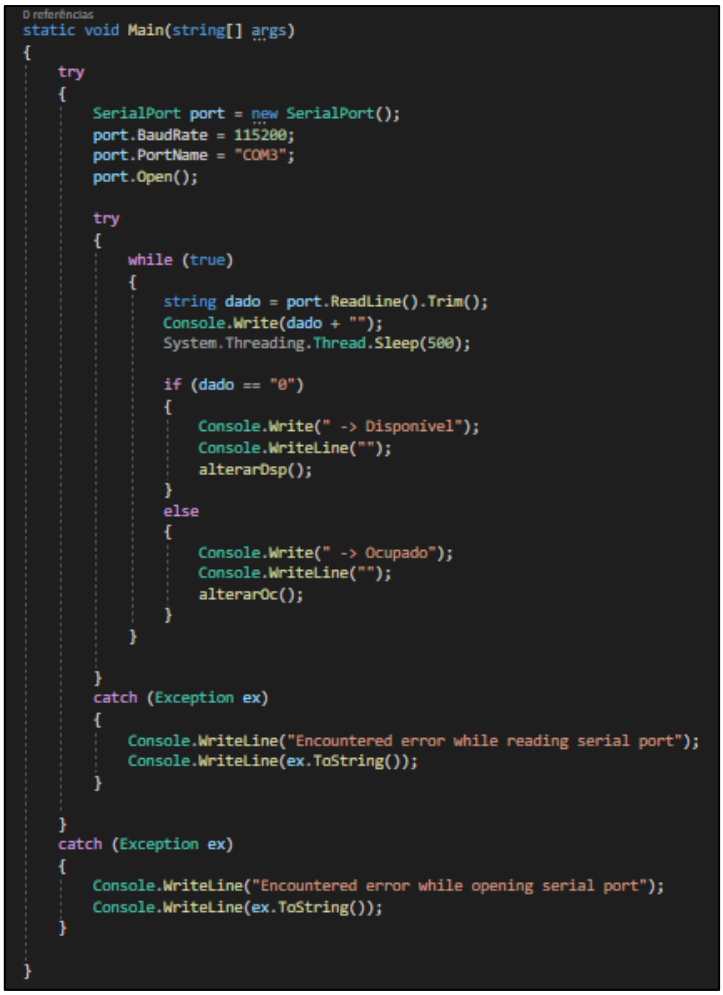


Fonte: Autor, 2020

4.1.2 LSensor

A aplicação é formada apenas pela classe ***Program***. Classe que contém dentro do método *main()* (Figura 18) as informações necessárias para ler os dados de saída do sensor de presença.

Figura 18: Método Main da Classe Program



```
0 referências
static void Main(string[] args)
{
    try
    {
        SerialPort port = new SerialPort();
        port.BaudRate = 115200;
        port.PortName = "COM3";
        port.Open();

        try
        {
            while (true)
            {
                string dado = port.ReadLine().Trim();
                Console.Write(dado + "");
                System.Threading.Thread.Sleep(500);

                if (dado == "0")
                {
                    Console.Write(" -> Disponivel");
                    Console.WriteLine("");
                    alterarDsp();
                }
                else
                {
                    Console.Write(" -> Ocupado");
                    Console.WriteLine("");
                    alterarOc();
                }
            }

        }
        catch (Exception ex)
        {
            Console.WriteLine("Encountered error while reading serial port");
            Console.WriteLine(ex.ToString());
        }

    }
    catch (Exception ex)
    {
        Console.WriteLine("Encountered error while opening serial port");
        Console.WriteLine(ex.ToString());
    }

}
```

Fonte: Autor, 2020

Detectada a mudança de estado, é disparado um dos dois métodos. O *alterarDsp()* (Figura 19) ocorre quando o estado é mudado de 1 para 0, cujo status se resulta em “Disponível”. O método *alterarOc()* (Figura 19) é a operação oposta, de 0 para 1, resultando em “Ocupado”. Essas informações são alteradas e salvas no Banco de Dados.

Figura 19: Métodos Update

```
public static void alterarOc()
{
    try
    {
        Connection = new MySqlConnection(ConnectionString);
        Connection.Open();
        MySqlCommand cmd = new MySqlCommand();
        cmd.Connection = Connection;
        cmd.CommandText = "Update Computador Set Status = 'Ocupado' Where Id = '1';";
        cmd.ExecuteNonQuery();
    }
    finally
    {
        if (Connection != null)
            Connection.Close();
    }
}

1 referência
public static void alterarDsp()
{
    try
    {
        Connection = new MySqlConnection(ConnectionString);
        Connection.Open();
        MySqlCommand cmd = new MySqlCommand();
        cmd.Connection = Connection;
        cmd.CommandText = "Update Computador Set Status = 'Disponivel' Where Id = '1';";
        cmd.ExecuteNonQuery();
    }
    finally
    {
        if (Connection != null)
            Connection.Close();
    }
}
```

Fonte: Autor, 2020

4.1.3 Telas de Interface

4.1.3.1 Layout do Laboratório de Informática

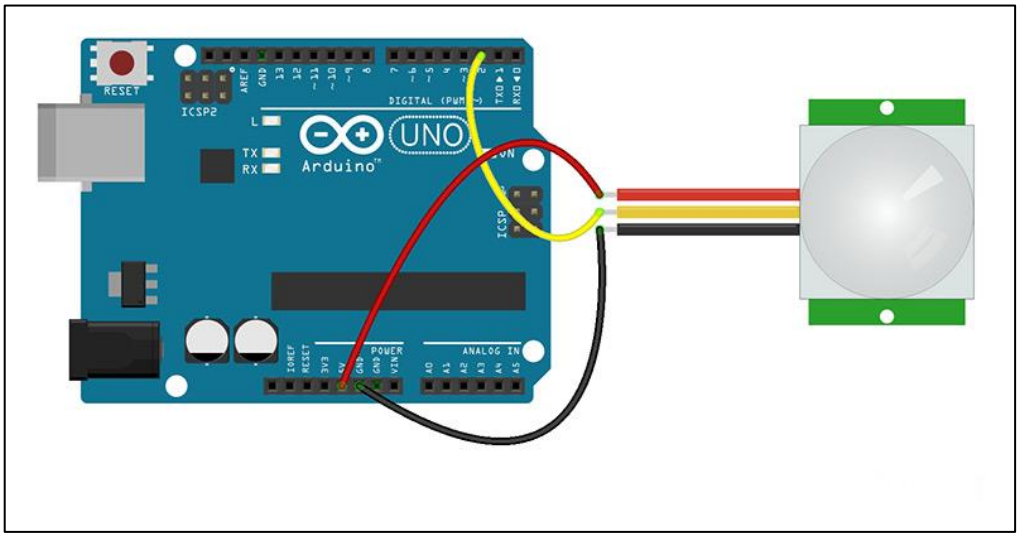
A figura 20 exibe a tela responsável por mostrar o layout dos computadores localizados no laboratório de informática (página LayoutComp). Cada equipamento é identificado por um número (id) e seu status é classificado por uma cor, a cor verde define o equipamento como “Disponível” e a cor vermelha o define como “Ocupado”.

Figura 20: Tela Layout dos Computadores

Fonte: Autor, 2020

4.2 SENSOR PIR E PLACA ARDUINO UNO

Figura 21: Esquema de Implementação do sensor e placa Arduino Uno

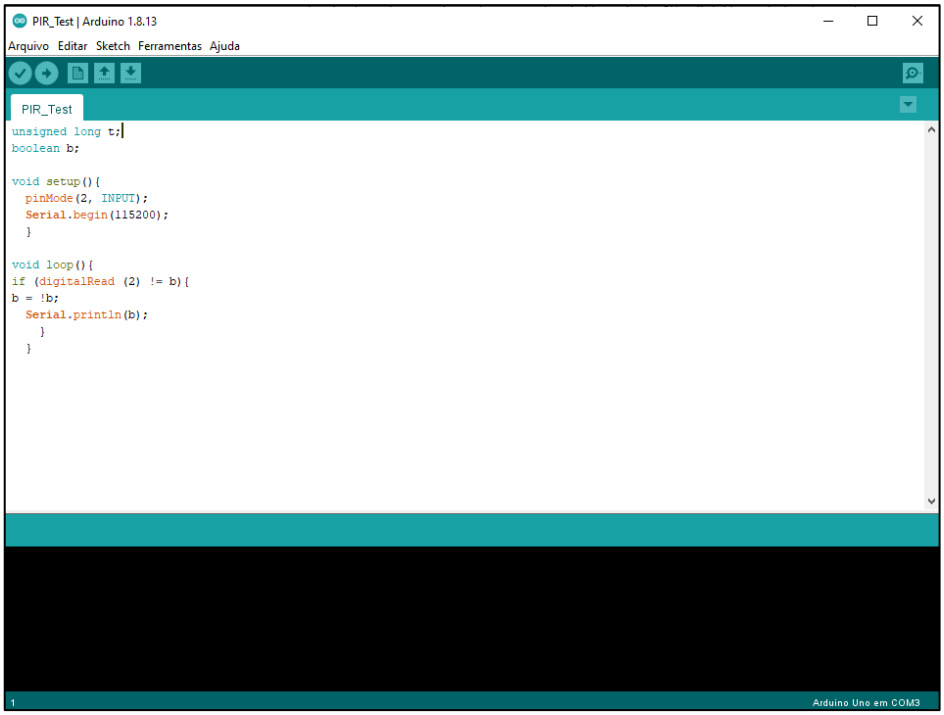


Fonte: <https://electropeak.com/learn/wp-content/uploads/2019/02/pir_fz.jpg>. Acesso em 27 out. 2020.

4.2.1 Programação no Arduino IDE

O programa da figura 22 é responsável por exibir a saída de dados do sensor de presença PIR. O comando *Serial.println(b)* exibe no console o valor booleano 1 para movimento detectado e 0 para ausência de movimento.

Figura 22: Programa PIR_Test



Fonte: Autor, 2020

5 CONCLUSÃO

Para a proposta apresentada neste trabalho, foi observado que, o Arduino apresentou-se como uma ferramenta de fácil implementação em conjunto com o sensor de presença PIR, contribuindo também com um ótimo custo-benefício.

Referindo-se ao objetivo central, que era a realização do monitoramento de equipamentos do laboratório de informática com o auxílio do Arduino e do sensor de presença PIR, foi obtido sucesso ao aliar o baixo custo de investimento do projeto e conceitos de IoT.

Foi observado também que a verificação dos equipamentos com a integração de sensores tornou a atividade automatizada e mais eficiente. Trazendo como vantagem a redução de trabalho do encarregado dessa função, também reduzindo o risco de contágio a ele e aos alunos.

5.1 DIFICULDADES ENCONTRADAS

Para gerenciar e armazenar os dados, foi escolhido um banco de dados de conexão local. A ausência da utilização da Nuvem não invalida a solução implementada neste trabalho, mas impede que mais informações pertinentes sejam geradas a partir da informação de disponibilidade.

Apesar do entendimento do funcionamento e da fácil implementação do Arduino e o sensor de presença PIR, observa-se que o projeto está a nível didático. Visto que a placa Arduino Uno adquirida não tem conexão à internet, foi necessário criar uma outra aplicação (LSensor) para que dados do sensor fossem lidos e fossem repassados ao banco de dados. Uma placa Arduino com conexão à internet reduziria o trabalho de codificação, a atualização de informações ao banco de dados ficaria a cargo apenas do código implementado no IDE do Arduino.

Observa-se também que a arquitetura do projeto está reduzida, todo o processo se resume a um só local. A máquina que roda a aplicação é a mesma que é monitorada.

5.2 TRABALHOS FUTUROS

Para trabalhos futuros, foi pensado em expandir a arquitetura além da que foi apresentada neste trabalho, com a possibilidade de ser executado no próprio laboratório de informática da Unicarioca. Utilizando a Nuvem e placas Arduino que se conectem à internet, tornando o processo mais eficiente.

É pensado também em novas formas de se verificar a presença de um aluno no laboratório de informática, visto que a sua ausência indicada pelo sensor de presença em um intervalo pequeno de tempo não confirma que ele deixou de utilizar o equipamento. É importante identificar se no momento da ausência, o aluno realizou logoff no equipamento, por exemplo.

Seria ideal controlar a entrada e saída do laboratório de informática, com o auxílio de sensores ou até mesmo a utilização de catracas, da qual seria necessário a impressão digital do aluno para registrar sua entrada ou saída, por exemplo.

Há diversos cenários a serem considerados e testados para definir o conceito de presença. E serão esses cenários que irão aperfeiçoar o sistema de forma gradual ao longo do tempo.

REFERÊNCIAS BIBLIOGRÁFICAS

WHO. **Biosecurity: An integrated approach to manage risk to human, animal and plant life and health**. Disponível em: https://www.who.int/foodsafety/fs_management/No_01_Biosecurity_Mar10_en.pdf. Acesso em 10 set. 2020.

MEC. **Protocolo de biossegurança para retorno das atividades nas Instituições Federais de Ensino**. Disponível em https://www.gov.br/mec/pt-br/centrais-de-conteudo/campanhas-1/coronavirus/CARTILHAPROTOCOLODEBIOSSEGURANAR101.pdf/view. Acesso em 10 set. 2020.

JEON et. al. **Iot-based Occupancy Detection System in Indoor Residential Environments**. Disponível em: https://www.researchgate.net/profile/Yunwan_Jeon/publication/322819341_IoT-based_Occupancy_Detection_System_in_Indoor_Residential_Environments/links/5cf7829a92851c4dd02a0109/IoT-based-Occupancy-Detection-System-in-Indoor-Residential Environments.pdf. Acesso em 08 set. 2020.

ASHTON, Kevin. **That 'Internet Things' Thing**. Disponível em: http://www.itrco.jp/libraries/RFIDjournal-That%20Internet%20of%20Things%20Thing.pdf. Acesso em 03 set. 2020.

ASHOK et. al**. IoT based Monitoring and Control System for Home Automation**. Disponível em: http://ijetsr.com/images/short_pdf/1511164902_413-418-site166_ijetsr.pdf. Acesso em 03 set. 2020.

SAINT-EXUPERY, Antoine de. **Internet of Things Strategic Research Roadmap**. Disponível em: http://www.internet-of-things-research.eu/pdf/IoT_Cluster_Strategic_Research_Agenda_2009.pdf. Acesso em 05 dez. 2020.

IEEE. **Towards a definition of the Internet of Things (IoT)**. Disponível em: https://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf. Acesso em 03 set. 2020.

RANDY, Frank. **Understanding Smart Sensors**. Segunda Edição. Norwood, MA: Artech House, 2000.

IEEE. **1451.2-1997 - IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Transducer to Microprocessor Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats**. Piscataway, NJ: IEEE Standards Department, 1998.

Autor Desconhecido. **Smart Sensor Technology for the IoT**. Disponível em: https://www.techbriefs.com/component/content/article/tb/pub/features/articles/33212. Acesso em 01 set. 2020.

PINTO, Gustavo. **Como funcionam os sensores inteligentes em soluções de IoT?**. Disponível em: https://v2com.com/2020/07/02/iot-sensores-inteligentes/. Acesso em 01 set. 2020.

ÖZDOĞAN, Hakan. **Smart sensors and the future of intelligent systems**. Disponível em: https://www.prescouter.com/2019/10/smart-sensors-and-the-future-of-intelligent-systems/. Acesso em 01 set. 2020.

Autor Desconhecido. **Indústria 4.0 e IoT expandem aplicações com sensores inteligentes**. Disponível em: https://revista-automacao.com/market-overview/17169-ind%C3%BAstria-4-0-e-iot-expandem-aplica%C3%A7%C3%B5es-com-sensores-inteligentes. Acesso em 01 set. 2020.

SHARMA, Rita**. Top 15 Sensor Types Being Used Most By IoT Application Development Companies**. Disponível em: https://www.finoit.com/blog/top-15-sensor-types-used-iot/. Acesso em 01 set. 2020.

MELL, Peter; GRANCE, Timothy. **The NIST Definition of Cloud Computing**. Disponível em: http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf. Acesso em 14 set. 2020.

RAJASEKAR, B; MANIGANDAN, S.K. **An Efficient Resource Allocation Strategies in Cloud Computing**. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1078.2472&rep=rep1&type=pdf. Acesso em 05 dez. 2020.

JANSEN, Wayne; GRANCE, Timothy. **Guidelines on Security and Privacy in Public Cloud Computing**. Disponível em: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-144.pdf. Acesso em 05 dez. 2020.

DOS ANJOS, André G.R; FERRAZ, Carlos A.G. **Aplicações móveis de contexto**. Disponível em https://www.cin.ufpe.br/~tg/2006-1/agra.pdf. Acesso em 12 set. 2020.

LOUREIRO, Antonio A.F. **Computação Ubíqua Ciente de Contexto: Desafios e Tendências**. Disponível em: https://www.researchgate.net/publication/239607395_Computacao_Ubiqua_Ciente_de_Contexto_Desafios_e_Tendencias. Acesso em 05 out. 2020.

KUDO, Taciana N; MICHELOTTI, Gislaine; ARAÚJO, Regina B. De. **Software de Suporte à Computação Ciente de Contexto**. Disponível em: http://www2.dc.ufscar.br/~regina/RecursosUCPG/suporte_contexto.pdf. Acesso em 05 out. 2020.

DEY, A. K. **Understanding and Using Context**. Disponível em: https://www.researchgate.net/publication/220141486_Understanding_and_Using_Context. Acesso em 05 out. 2020.

ABOWD, Gregory D.; MYNATT, Elizabeth D. **Charting Past, Present, and Future Research in Ubiquitous Computing**. Disponível em: https://modernmobile.cs.washington.edu/docs/millenium.pdf. Acesso em 05 out. 2020.

HENRICKSEN, Karen; INDULSKA, Jadwiga. **Modelling and Using Imperfect Context Information**. Disponível em: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.59.7327&rep=rep1&type=pdf. Acesso em 05 out. 2020.

SILVA et. al. **Plataforma Arduino integrado ao PLX-DAQ: Análise e aprimoramento de sensores com ênfase no LM35**. Disponível em: https://www.researchgate.net/publication/305771112_Plataforma_Arduino_integrado_ao_PLX-DAQ_Analise_e_aprimoramento_de_sensores_com_enfase_no_LM35. Acesso em 26 out. 2020

CAVALCANTE et al. **Física com Arduino para Iniciantes**. Disponível em: https://www.scielo.br/pdf/rbef/v33n4/18.pdf. Acesso em 27 out. 2020

SOUZA et al. **A placa Arduino: uma opção de baixo custo para experiências de física assistidas pelo PC**. Disponível em: https://www.scielo.br/pdf/rbef/v33n1/26.pdf. Acesso em 27 out. 2020.

THOMSEN, Adilson. **O que é Arduino?**. Disponível em: https://www.filipeflop.com/blog/o-que-e-arduino/. Acesso em 27 out. 2020.

Autor Desconhecido. **Arduino Uno Rev3**. Disponível em: https://store.arduino.cc/usa/arduino-uno-rev3. Acesso em 27 out. 2020.

THOMSEN, Adilson. **Qual Arduino Comprar? Conheça os Tipos de Arduino**. Disponível em: https://www.filipeflop.com/blog/tipos-de-arduino-qual-comprar/. Acesso em 27 out. 2020.

SOUZA, Fabio. **Arduino UNO**. Disponível em: https://www.embarcados.com.br/arduino-uno/. Acesso em 27 out. 2020.

HOSSEINI, Saeed. **PIR Motion Sensor: How to Use PIRs with Arduino & Raspberry Pi**. Disponível em: https://electropeak.com/learn/pir-motion-sensor-how-pir-work-how-use-with-arduino/. Acesso em 27 out. 2020.