InoSensor: Ferramenta para Gerenciamento de Sensores para Arduino

Haissam Yebahi[1], Alex Luiz de Sousa[1], Mário Ezequiel Augusto[1], Nilson Ribeiro Modro[1]

[1]Universidade do Estado de Santa Catarina (UDESC)
Centro de Educação do Planalto Norte (CEPLAN)

hyebahi@gmail.com, alex.sousa@udesc.br, mario.augusto@udesc.br, nilson.modro@udesc.br

Resumo. *A ferramenta InoSensor foi desenvolvida com o objetivo de facilitar o uso de sensores e permitir o gerenciamento e monitoramento da plataforma Arduino remotamente e sem a necessidade de programar o hardware quando houver a necessidade de modificar as configurações dos sensores conectados. Entre os diferenciais existentes na ferramenta InoSensor, destaca-se a capacidade de monitoramento remoto de sensores e seu uso por pessoas sem conhecimento em programação. A solução desenvolvida é responsável por apresentar uma alternativa de baixo custo na área de Domótica.*

Abstract. *The InoSensor tool was developed to facilitate the use of sensors and enable easy management and monitoring of the Arduino platform remotely, with no need to program the firmware to modify sensors settings. Among the existing differentials in InoSensor tool, there is the ability of remote monitoring sensors across a local network or the Internet, and the use of the solution by people without programming knowledge because the sensors and actuators' setting are carried out on the Web server. The developed solution is responsible for presenting a low-cost and easy to use alternative in Home Automation area.*

1. Introdução

O uso de dispositivos eletrônicos, seja para a realização de tarefas do cotidiano ou para Automação Industrial, está inserido nos mais variados ambientes. O simples fato de utilizar um sensor infravermelho para detectar a presença de movimento, ou ligar uma lâmpada automaticamente ao escurecer, envolve componentes eletrônicos que reduzem a necessidade de executar tarefas manualmente.

Independente da aplicação, o uso de tecnologia para Automação, seja para residências, empresas ou indústrias, proporciona o gerenciamento de recursos de forma bastante eficiente e sustentável, além de obter diversos benefícios como redução de custos, manutenção e segurança [DIAS; PIZZOLATO, 2004].

A Automação Residencial, também conhecida como Domótica, é formada por um conjunto de dispositivos e equipamentos interconectados através de uma rede para comunicação. Nesse contexto, as informações são coletadas e processadas, conforme as regras são definidas. Determinadas ações podem ser executadas com o objetivo de supervisionar ou gerenciar o ambiente em que estão inseridas e assim satisfazer as necessidades de comunicação, conforto e segurança [DIAS; PIZZOLATO, 2004].

Devido às constantes mudanças e às necessidades individuais de cada cenário, o comportamento de sistemas automatizados deve proporcionar flexibilidade e uma fácil adaptação às condições do ambiente [SGARBI; TONIDANDEL, 2006].

Segundo alguns princípios presentes na Automação e Domótica, é necessário que haja uma forma simples de gerenciar os dispositivos e as ações que devem ser executadas com base nas regras estabelecidas. Levando-se em conta o ambiente que se deseja controlar ou monitorar, é preciso que se considere a possibilidade de realizar um monitoramento remoto de forma centralizada, para facilitar e acelerar o processo de análise de informações.

Para Sousa (2009), a crescente popularização da Internet, possibilitou o desenvolvimento de novos modelos de sistema baseados na execução de ações remotas, para controle e atuação de forma bastante rápida e segura.

Muitas das ações envolvidas ao se criar projetos baseados em sensores incluem padrões e rotinas em código programado bastante semelhantes, mas que necessitam de retrabalho para ajustar as ações executadas. Essas atividades despendem tempo e dificultam a manutenção, quando uma simples parametrização poderia adaptar as regras e comportamentos desejados a um novo contexto.

Arduino é uma das plataformas para prototipação e desenvolvimento baseada em microcontroladores mais populares da atualidade. Em 2011, já haviam sido vendidas mais de trezentas mil unidades da versão original. Isso se deve principalmente à sua característica *open source* o que torna possível o desenvolvimento de diversas soluções para uma mesma arquitetura (TORRONE, 2011).

Ainda que a simplicidade da plataforma Arduino facilite o desenvolvimento das mais variadas soluções, é necessário que os interessados, além de conhecimento em eletrônica básica, também tenham conhecimentos mais avançados de lógica e programação.

A ferramenta InoSensor foi desenvolvida com o propósito de facilitar e simplificar o processo de adição de sensores e permitir o monitoramento e controle de Arduino em uma rede local ou Internet. Sua utilização estende-se como uma ferramenta de apoio para disciplinas como Eletrônica Básica e Redes de Computadores, pois pode facilitar a compreensão de conceitos teóricos aprendidos em cursos de graduação com foco em tecnologia ou para pessoas interessadas em alternativas para Automação de baixo custo.

2. Composição do Sistema

A ferramenta InoSensor foi desenvolvida baseada em uma arquitetura distribuída, onde o servidor é responsável por monitorar as informações, enviar *e-mails*, desligar alarmes remotamente e alterar o comportamento das ações executadas por cada Arduino. Já o Arduino, é responsável por processar as informações geradas pelos sensores e executar as ações com base nas condições configuradas em cada sensor. Para exemplificar o funcionamento do sistema, a Figura 1 ilustra, de forma geral, como é feita a comunicação entre o servidor InoSensor e o Arduino via infraestrutura padrão das redes de computadores.

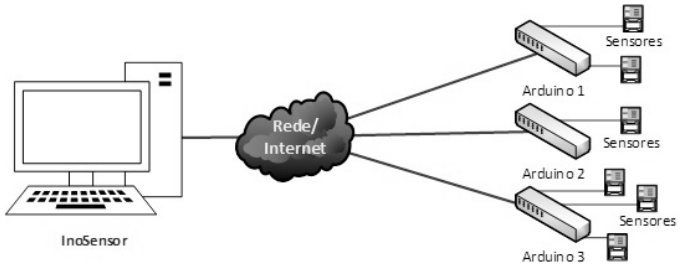

Figura 1. Visão macro da representação do sistema.

2.1. Sensores Utilizados no Projeto

Os sensores são fabricados em circuitos prontos para o uso, sendo apenas necessário conectar as portas do sensor diretamente às portas do Arduino. A alimentação de energia é de 5V para todos os sensores. Abaixo são listados os sensores e principais componentes eletrônicos utilizados no projeto para a comunicação com o Arduino:

- **Sensor de Temperatura e Umidade**;
- **FotoResistor**;
- **Sensor de Fogo**;
- **Sensor de Gás**;
- **Sensor de Vibração**;
- **Sensor Magnético**;
- **Detector de presença InfraVermelho**;
- **Relé**;
- **LED RGB**; e
- ***Buzzer***.

3. Comportamento do Sistema

A ferramenta InoSensor permite cadastrar os sensores e atuadores para executarem as ações de acordo com as necessidades. De forma simplificada, o processo é iniciado com o cadastro dos sensores. Ao cadastrar um sensor, deve-se selecionar as portas disponíveis que serão utilizadas para o módulo do sensor (Figura 2).

Em seguida são cadastradas as ações que devem ser executadas, com base nas condições específicas para cada tipo de sensor (figura 3) e, ao finalizar a configuração, deve ser feita a transmissão, para o Arduino entrar em funcionamento.

O Arduino é responsável pelo gerenciamento dos sensores conectados e pela execução das ações que foram configuradas por meio da ferramenta InoSensor. As configurações, condições e ações são armazenadas em um arquivo texto, salvo na memória externa no cartão SD.

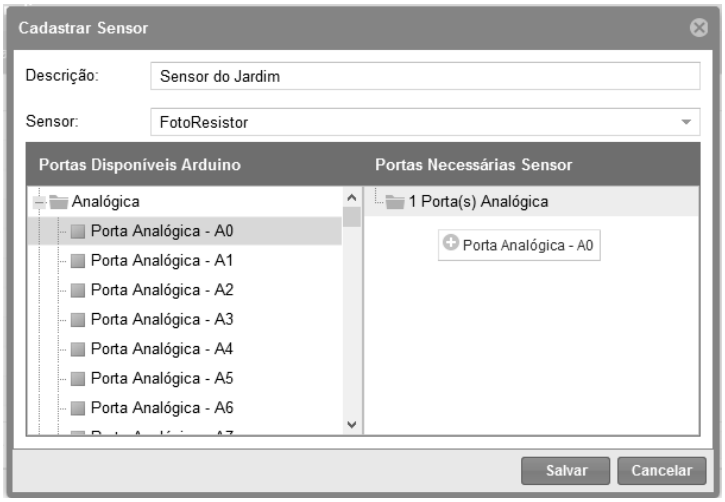

Figura 2. Tela para cadastro de sensor.

Figura 3. Tela para cadastro de ação de um sensor.

Ao iniciar o Arduino, o *firmware* inicializa todas as portas de entrada e saída com base nas informações coletadas do arquivo de configuração e em seguida, a interface *ethernet* também é inicializada para possibilitar a troca de informações com o servidor. A partir desse momento, as informações são lidas dos sensores que estão configurados e caso uma condição seja verdadeira, é iniciada a execução dos atuadores.

3.1. Ações

O sistema foi desenvolvido com a capacidade de responder à ocorrência de eventos externos e possibilitar o controle e gerenciamento dos sensores conectados, sendo assim, o Arduino possui a capacidade de executar três tipos de ações distintas, descritas abaixo:

- **Acionar Atuador**: Possui a função de executar um atuador do sistema: (i) disparar alarmes através de um *buzzer*; (ii) ligar ou desligar algum equipamento elétrico por meio de um relé; e (iii) ligar e desligar LED RGB;
- **Enviar *e-mail***: Para situações de emergência que devem notificar algum responsável, é possível configurar o Arduino para disparar uma solicitação de envio de *e-mail* para o servidor. Isso permite deixar todo o trabalho necessário para envio de *e-mail,* como autenticação, configuração e tratamento de erros, a cargo do servidor e manter o Arduino responsável pelo gerenciamento dos sensores;

- **Notificar servidor**: É possível configurar quando da ocorrência de uma condição de um sensor, o envio de uma notificação para o servidor InoSensor. Essa ação irá aparecer no painel de monitoramento do servidor, informando a data, hora, dispositivo e informação relativa ao sensor que disparou a ação.

3.2. Configurações do Arduino

Uma característica importante, implementada na ferramenta InoSensor, é a possibilidade de ajustar as configurações dos sensores e atuadores sem a necessidade de atualizar o *firmware* do Arduino, facilitando o uso da ferramenta e eliminando a necessidade de conhecimento na linguagem de programação do *firmware*.

Devido à capacidade de ajuste variável das configurações, as informações são armazenadas em arquivo texto, gravadas na memória externa do cartão SD. O uso da memória externa diminui as limitações de memória do Arduino, pois as particularidades de cada dispositivo são ajustadas e gravadas na memória externa, sem comprometer o funcionamento do hardware por estouro de memória. Outra vantagem é tornar o *firmware* do Arduino mais robusto e facilitar a depuração de erros, já que o *firmware* se torna padrão.

3.3. Monitor de notificações

O Arduino pode controlar os sensores de forma isolada, executando as ações com base nas condições configuradas, desde que as configurações estejam devidamente gravadas na memória do cartão SD. Nesse modelo, a operação do Arduino está restrita aos atuadores conectados no próprio dispositivo, como relé, *buzzer* e LED RGB.

Já para possibilitar o monitoramento remoto do dispositivo, é necessário que haja uma conexão de rede ativa. Nesse modo de operação o Arduino pode trocar informações com o servidor InoSensor e notificá-lo da ocorrência de eventos e, se necessário, o servidor pode enviar e-mail para notificar os responsáveis.

A ferramenta InoSensor conta com um painel de notificações atualizado constante e automaticamente, e é responsável por mostrar os últimos eventos ocorridos em cada Arduino configurado no sistema.

4. Arquitetura do Sistema

Esta seção apresenta um resumo das principais características utilizadas no desenvolvimento da solução InoSensor.

4.1. Atualização de Firmware

O firmware é padrão para todos os Arduinos conectados com a ferramenta, porém devido às necessidades de configuração da comunicação (*e.g.,* IP, máscara de rede, *MAC Address*) e segurança (*e.g.,* usuário, senha), foi necessário compilar o código com as particularidades de cada Arduino.

4.2. Protocolo de Configuração do Arduino

Para a comunicação entre o Servidor InoSensor e o Arduino, foi definido um protocolo de comunicação com socket TCP/IP. O Arduino pode atuar de duas formas: como um cliente *socket*, responsável por enviar as notificações da ocorrência de eventos

diretamente para o servidor PHP, ou através do envio de mensagens com o protocolo HTTP 1.1[1].

Já para receber novas configurações do servidor InoSensor, o Arduino cria um servidor *socket*, realiza a autenticação para validar o usuário e senha armazenados no Arduino e em seguida inicia a transmissão das informações, para finalmente gravá-las em um arquivo texto no padrão *Comma Separated Value* (CSV).

A figura 4 mostra a estrutura simplificada do arquivo de configurações. Cada nó possui um identificador iniciado com o prefixo *begin* (*e.g., beginauth*) seguido do nome do bloco e pela palavra *end* (*e.g., endauth*), que informa o fim de um bloco.

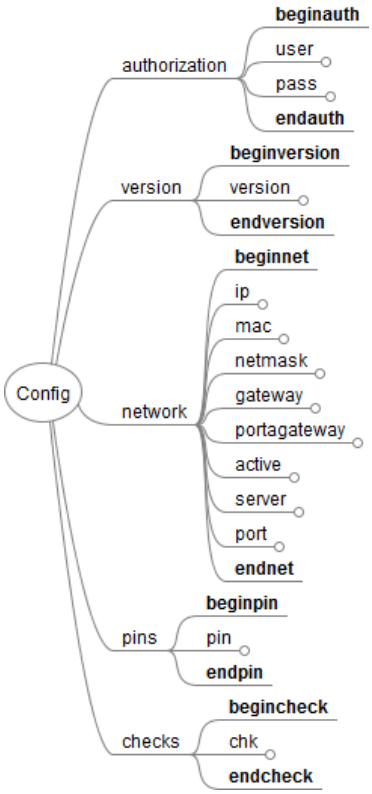

Figura 4. Estrutura simplificada do arquivo CSV.

Os cinco nós principais existentes no arquivo de configuração são descritos abaixo:

- ***Authorization***: É formado pelas informações para autenticação. Caso a autenticação falhe, o *firmware* irá abortar as verificações e ações dos sensores;
- ***Version***: Possui as informações da versão do *firmware*, permitindo ao servidor identificar a versão instalada no Arduino;
- ***Network***: Possui as informações necessárias à configuração e ao acesso à rede, além de conter as informações para comunicação com o servidor InoSensor;
- ***Pins***: Possui as informações relacionadas às configurações das portas do Arduino, identificando-as como portas de entrada ou saída e inicializando seu estado;
- ***Checks***: Esse bloco pode conter uma ou mais verificações, armazenadas, linha à linha. Para cada tipo de sensor existe uma regra padrão para montar o registro.

[1]Disponível em: http://w3.org/Protocols/rfc2616/

Ao final de cada registro de verificação existe a ação que deve ser executada quando a condição do sensor for verdadeira.

5. Apresentação da Ferramenta

Nesta seção é apresentado o módulo Web responsável por interagir com o usuário e realizar a configuração e monitoramento do Arduino e o assistente de atualização de firmware para facilitar o processo de upgrade do programa do Arduino.

5.1. Módulo Web InoSensor

A interface principal do sistema (figura 5) possui uma estrutura padrão de navegação. Em sua parte central, são carregadas as funcionalidades, de acordo com a função que deve ser selecionada no menu, posicionado à esquerda da tela. O painel de notificações fica disponível na parte inferior da aplicação para que, durante o uso da ferramenta, novas notificações possam ser visualizadas rapidamente.

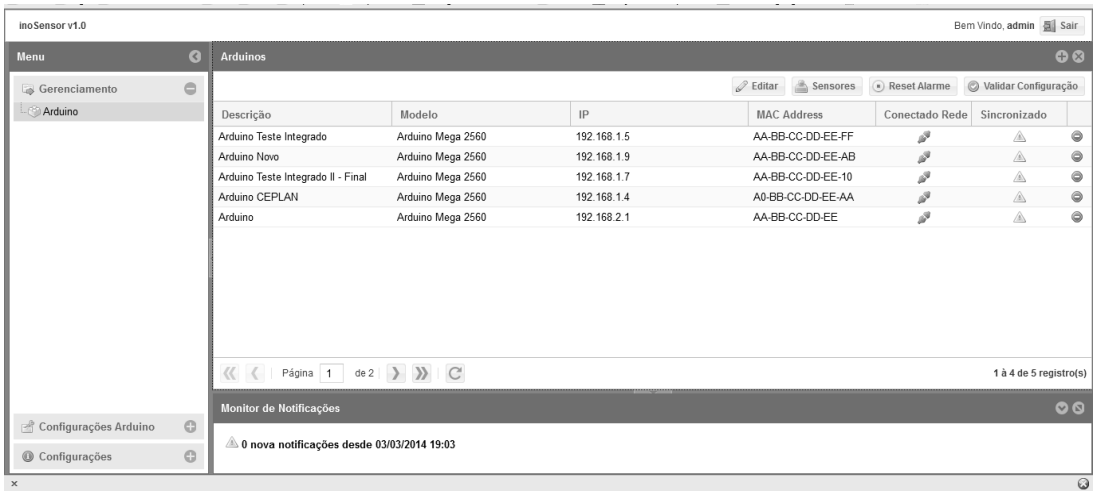

Figura 5. Tela principal da ferramenta InoSensor.

5.2. Módulo Atualizador de Firmware

O utilitário é responsável por atualizar o firmware com a versão mais recente disponível no servidor InoSensor. Para utilizá-lo, é necessário ter um usuário cadastrado no servidor InoSensor. O utilitário é executado localmente em um computador com interface USB, através da qual será conectado ao Arduino para atualização (Figura 6). O processo é composto de três etapas descritas na abaixo.

Na primeira fase, devem ser informados os dados do servidor e usuário para autenticação. No segundo passo, deve-se selecionar o Arduino que se deseja atualizar e, por fim, o Arduino deve ser conectado à interface USB, para que o sistema possa detectá-lo e assim iniciar a gravação do firmware.

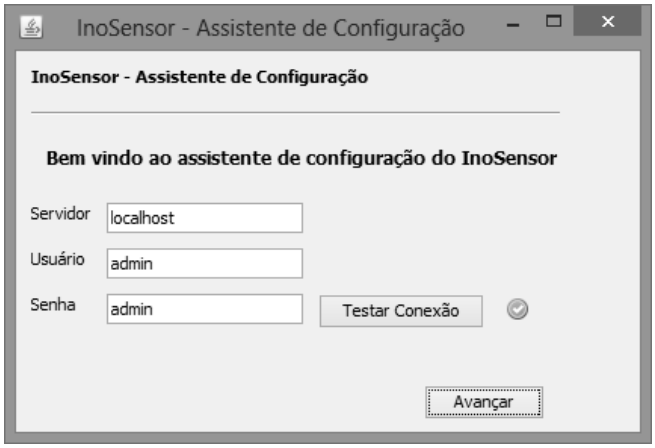

Figura 6. Tela do assistente de configuração

6. Testes e Validação

Durante o desenvolvimento do *firmware* do Arduino, foram realizados testes individuais para avaliar o funcionamento das características específicas de cada sensor e atuador. Foram testados dois modelos de Arduino, o Arduino Uno e Arduino Mega2560.

A tabela 1 apresenta as principais diferenças de processamento, portas e memória entre os modelos Arduino Uno e Arduino Mega [WHEAT, 2011].

Tabela 1. Comparação da especificação dos Arduinos.

Especificação	Arduino Uno	Arduino Mega 1280	Arduino Mega 2560
Modelo Processador	ATmega 328	ATmega 1280	ATmega 2560
Clock Processador	16MHz	16MHz	16MHz
Memória Programa (FLASH)	32KB	128KB	256KB
Memória Dados (SRAM)	2KB	8KB	8KB
EEPROM	1KB	4KB	*4KB*
Pinos Processador	28/32	100	100
Portas Entradas/Saídas	14	54	54
Portas Analógicas	6	16	16
Portas PWM	6	14	14

Durante os testes, optou-se por definir rotinas específicas para cada sensor, com o objetivo de facilitar os testes individuais e uma melhor integração entre todas as partes que compõem o *firmware* do Arduino.

Após os testes dos sensores e atuadores, foi desenvolvido o *firmware* padrão para o Arduino. Para simular o arquivo de configurações que posteriormente seria gravado na memória externa do cartão SD, foi montada uma estrutura de dados para armazenar as informações. A partir desse ponto, foi possível ter uma ideia de como o

sistema se comportaria após as configurações serem transmitidas do servidor InoSensor para o Arduino.

A qualidade dos sensores é um ponto que merece destaque. Devido ao baixo custo dos sensores adquiridos inicialmente, observou-se que a acurácia das informações lidas de alguns sensores acabava sendo afetada, tornando o sistema ineficaz e em alguns casos até gerando informações incorretas. Para resolver esse problema, foram adquiridos novos sensores com um custo um pouco mais alto, porém que mostraram ser mais precisos e tolerantes a falhas.

Os testes dos sensores e atuadores foram realizados com o uso do monitor da interface serial USB, presente no Ambiente de Desenvolvimento Integrado (IDE) do Arduino, para verificar o comportamento do firmware.

6.1. Teste de Integração

Após testar todos os sensores individualmente, foram iniciados os testes de integração entre o servidor InoSensor e o Arduino, para avaliar o comportamento de todo o conjunto.

Para facilitar o estudo das funções de rede disponíveis para o Arduino, inicialmente foram feitos os testes através dos exemplos disponíveis na própria IDE do Arduino. Em seguida, foram criadas as rotinas de conexão via *socket* no servidor PHP, necessárias tanto para o envio de notificações do Arduino para o Servidor, quanto para a transmissão de configurações do Servidor para o Arduino. As bibliotecas disponíveis em ambos os ambientes facilitaram a integração e se mostraram bastante estáveis durante o envio e recebimento de informações.

O mesmo processo foi realizado para testar o armazenamento das configurações no cartão de memória SD, disponível no *shield ethernet*. Foi identificada uma característica interessante, que facilitou bastante o uso de ambas as soluções: a seleção automática do barramento de comunicação serial entre o microcontrolador e os periféricos, já que somente um periférico por vez pode acessar o barramento. Em versões mais antigas das bibliotecas do Arduino, o controle era feito de forma manual e a responsabilidade ficava por conta do desenvolvedor. Nas versões mais recentes, é possível deixar que a própria biblioteca gerencie os periféricos no barramento *Serial Peripheral Interface* (SPI).

6.2. Teste de Escalabilidade

Ao desenvolver soluções de *firmware* para microcomputadores, convencionalmente não é necessário controlar a alocação de memória, já que esses recursos possuem grande capacidade de armazenamento. Normalmente, essa tarefa é controlada pelo sistema operacional, que gerencia a memória. Porém, o desenvolvimento para microcontroladores exige um conhecimento aprofundado da plataforma e deve se levar em consideração as limitações de processamento e memória de cada dispositivo.

O tamanho total da versão do *firmware* ficou em aproximadamente 53KB (53954 *bytes*) e a memória das variáveis em torno de 3KB (3116 *bytes*). Como o Arduino Mega2560 possui uma memória de programa de 256KB, isso representa aproximadamente 20,6% da capacidade total do Arduino, o que permite ampliar as funcionalidades sem comprometer as limitações do *hardware*.

7. Conclusão

Esse projeto teve como objetivo desenvolver uma ferramenta para gerenciamento de sensores na plataforma Arduino e facilitar o processo de atualização de configurações sem a necessidade de reprogramar o *firmware*, além de permitir que interessados possam utilizar a plataforma mesmo sem conhecimentos em programação.

Inicialmente, a proposta era empregar os dois modelos convencionais de Arduino, Uno e Mega. Porém, durante o desenvolvimento, foi identificado que a limitação de 32KB de memória disponível no Arduino Uno, impediu seu uso na versão final da ferramenta. Apesar do custo maior do modelo Mega em relação ao modelo Uno, a diferença é compensada pela maior capacidade de memória e quantidade de portas do modelo Mega, que consequentemente permite adicionar uma maior quantidade de sensores. Em relação à estabilidade do *firmware*, foi identificado um problema na biblioteca de *Strings*, que acabava congelando o processamento do Arduino. Após pesquisas feitas na Internet, foi identificado que o problema havia sido solucionado em uma nova versão da biblioteca do Arduino.

Ao avaliar a versão final do projeto, pode-se concluir que todos os objetivos foram alcançados, além da ferramenta poder contribuir para soluções de monitoramento de sensores na plataforma Arduino, sem a necessidade de conhecimentos em programação. A versão final da solução integra quatro linguagens de p'rogramação: Java, PHP, JavaScript e C. Essa integração demonstra que, apesar das particularidades de cada tecnologia, elas podem se complementar, para que a solução desenvolvida alcance os objetivos desejados.

Referências

DIAS, C, L. de A. and PIZZOLATO N. D. (2004) Domótica: Aplicabilidade e Sistemas de Automação residencial. [Online] Available from: http://www.essentiaeditora.iff.edu.br/index.php/vertices/article/view/98/86. [Accessed: 17th September 2016].

SGARBI, J. A. and TONINANDEL, F. (2006). Domótica Inteligente: Automação residencial baseada em Comportamento, 2006. [Online] Available from: http://fei.edu.br/~flaviot/pub_arquivos/WTDIA06.pdf. [Accessed: 17th September 2016].

SOUSA, A. (2009). Um sistema de apoio à tomada de decisão para o monitoramento remoto de centrais de alarmes patrimoniais. [Online] Available from: http://www.tede.udesc.br/tde_busca/arquivo.php?codArquivo=2943. [Accessed: 17th September 2013].

TORRONE, P. (2011). Why the Arduino won and why it's here to stay. [Online] Available from: http://makezine.com/2011/05/12/why-google-choosing-arduino-matters-and-the-end-of-made-for-ipod-tm. [Accessed: 17 th September 2016].

WHEAT, D. (2011). Arduino Internals (1st. ed). New York: Apress Editor, Advances in Computer Science, pages 555-566. Publishing Press.