Sistema de Descoberta Dinâmica e Configuração Remota de Dispositivos Microcontrolados Voltados para IoT

Deivison Morim Pasa[1], Reiner Franthesco Perozzo[1]

[1]Curso de Sistemas de Informação – Centro Universitário Franciscano (UNIFRA)
97010-491 – Santa Maria – RS – Brasil

{deivison,reiner.perozzo}@unifra.br

Abstract. *This paper presents a proposal to develop a system capable of finding, dynamically, new devices in environments of IoT and configuring them remotely, regardless of their role. The system also offers a web interface to centralize the management of new and existing devices, members of the same environment. To do such, five activities, were applied in the FDD methodology to plan and build these features, using communication protocols TCP, markup languages and style HTML5 and CSS3, programming PHP and SQL, and specific domain (DSL) for the Arduino based on C/C++. Results showed success in achieving the objectives of this project.*

Resumo. *Este trabalho apresenta a proposta de desenvolvimento de um sistema capaz de encontrar, dinamicamente, novos dispositivos em ambientes de IoT e configurá-los, remotamente, independente de sua função. Também disponibiliza uma interface web para centralizar a gerência dos novos, e atuais, dispositivos integrantes deste mesmo ambiente. Aplica as cinco atividades presentes na metodologia FDD para planejar e construir estas funcionalidades, utilizando protocolos de comunicação TCP, linguagens de marcação e estilo HTML5 e CSS3, de programação PHP e SQL, e de domínio específico (DSL) para Arduino baseada em C/C++; obtendo êxito ao alcançar os objetivos na conclusão do projeto.*

1. Introdução

A conectividade entre alguns dispositivos é uma funcionalidade necessária, pode-se dizer crítica, para cumprir o propósito de IoT (*Internet of Things* — Internet das Coisas). Devido à crescente popularidade dos dispositivos móveis na última década, estima-se que existam cerca de 1,5 bilhões de computadores pessoais com Internet e mais de 1 bilhão de dispositivos móveis habilitados para Internet hoje; o número de "coisas" conectadas à Internet já excedeu o número de pessoas na Terra em 2008 [Evans 2011].

Ao acompanhar este crescimento, é possível perceber também o desenvolvimento de soluções focadas em melhorar os processos que envolvem IoT. Segundo Perera *et al.* (2013), as soluções voltadas para este ambiente podem ser divididas em dois segmentos: soluções de *hardware* baseadas em sensores e soluções de *software* focadas na nuvem. Este trabalho propõe um sistema para que, dinamicamente, ocorra a descoberta e a configuração remota de dispositivos voltados para IoT.

Esta proposta está ambientada na dinamização dos processos de criação e ampliação de ambientes IoT, centralizando o gerenciamento dos dispositivos presentes no cenário atual. Foram utilizados sistemas embarcados (Arduino e Raspberry Pi) e meios de comunicação Ethernet como apoio na construção do cenário de teste e validação do trabalho.

1.1. Objetivo Geral

Desenvolver um sistema capaz de identificar novos dispositivos IoT microcontrolados conectados em uma rede, a fim de permitir sua configuração remota, dispensando a necessidade de ser implementado um novo sistema *in loco* a cada nova inclusão de dispositivo.

1.2. Objetivos Específicos

Dentre os objetivos específicos do sistema, destacam-se os seguintes:

- Dinamizar a inclusão de novos dispositivos em ambientes IoT;
- Manter o controle e a gerência, de forma centralizada, dos dispositivos que já foram descobertos e adicionados ao cenário;
- Possibilitar flexibilidade na expansão da ferramenta de acordo com novos requisitos;
- Desenvolver uma interface gráfica de gerenciamento (GUI *Manager*) em ambiente *web*.

2. Referencial Teórico

Esta seção apresenta alguns dos principais conceitos e tecnologias utilizados neste trabalho, também estão inseridos os trabalhos relacionados que colaboraram para a construção da proposta.

2.1. IoT — Internet of Things

O conceito de IoT foi mencionado pela primeira vez em 1999, por Kevin Ashton — britânico cofundador do Auto-ID Center no MIT, em uma apresentação para executivos da Procter & Gamble, cuja ideia era etiquetar, eletronicamente, os produtos da empresa para facilitar a logística da cadeia de produção, por meio de identificadores de radiofrequência (RFID) [Rangel 2014].

Para Ashton, duas redes distintas — rede de comunicação humana e rede de comunicação das coisas (dispositivos) — precisam se unir. Segundo o britânico, o ponto de encontro se dará aonde não apenas serão utilizados computadores, mas onde um computador seja usado de forma independente, com a finalidade de tornar a vida humana mais eficiente. Desta maneira as coisas estarão conectadas em rede de modo inteligente, e passarão a sentir e interagir com o ambiente ao seu redor [Rangel 2014].

Em 2003, segundo informações da Forrester Research (empresa americana de pesquisa de mercado e tecnologia), havia aproximadamente 6,3 bilhões de pessoas vivendo no mundo e 500 milhões de dispositivos conectados à Internet, já em 2010 o número de dispositivos passou para 12,5 bilhões, enquanto a população subiu para 6,8 bilhões.

Pesquisadores chineses descobriram, em 2009, que a Internet duplica de tamanho a cada 5,32 anos e estimam, que em 2020, aproximadamente 50 bilhões de dispositivos estarão conectados à Internet no mundo todo [Zyga 2009].

Esses dados, junto as previsões para um total de 7,6 bilhões de pessoas no planeta em 2020 — pelo Censo dos Estados Unidos (U.S. Census Bureau) — eleva o número de 0,08 dispositivos por pessoa em 2003, para 6,57 dispositivos por pessoa em 2020 [Evans 2011].

Isto faz com que surjam várias redes de dispositivos conectados entre si, porém isoladas de outras redes — um carro possui uma rede de sensores e controladores para monitorar e acionar diversas funções do veículo, mas esta rede não está interligada a uma rede residencial que, também, possui dispositivos conectados (sensor de luminosidade ou acionador de portão por exemplo). A medida que a IoT evolui, essas redes e diversas outras, serão conectadas e permitirão um controle/monitoramento de qualquer coisa em qualquer lugar [Evans 2011].

A Figura 1 ilustra a abordagem de uma rede composta por várias outras redes:

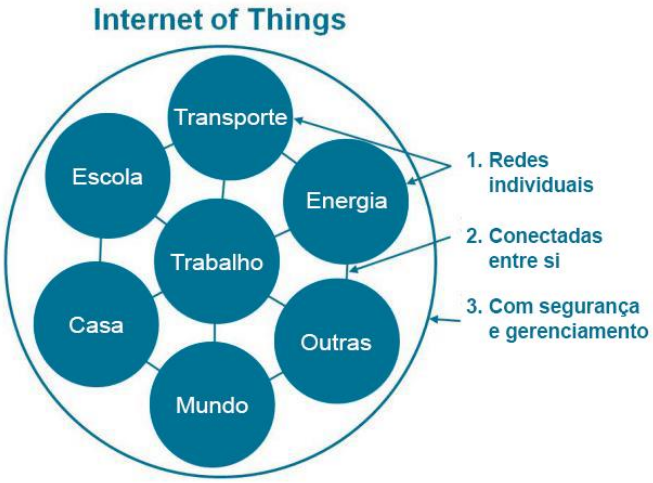

Figura 1. IoT vista como uma rede de redes (adaptado de Evans 2011).

Nesse contexto compreende-se que a Internet das Coisas é a evolução da Internet. Esse salto permite o desenvolvimento de sistemas que podem mudar o cotidiano das pessoas e torná-las mais proativas e menos reativas.

2.2. Redes de Comunicação de Dados

O termo comunicação, em outras palavras, refere-se essencialmente a compartilhar informações entre dois ou mais indivíduos, esta troca de informação pode ocorrer em um ambiente interno ou externo. A palavra dados, refere-se a informações apresentadas em qualquer forma, desde que seja conhecida pelas partes envolvidas, tanto no envio quanto no recebimento. A comunicação de dados é então, a troca de mensagens entre dois dispositivos, ou mais, intermediado de algum meio de transmissão composto da combinação entre *hardware* e *software.* [Forouzan 2009].

Forouzan (2009) define rede como um conjunto de dispositivos conectados (algumas vezes chamados de *nós*) por um *link* de comunicação. Uma rede de comunicação de dados é este conjunto de dispositivos trocando informações entre si (por

exemplo, dispositivos microcontrolados monitorando e controlando sensores em um ambiente IoT).

Essa rede de comunicação deve funcionar, com qualquer conexão que seja capaz de interligar o transmissor aos seus receptores, independente da forma de transmissão e protocolo — Ethernet, Wi-Fi, Bluetooth, ZigBee, entre outras.

O Ethernet (IEEE 802.3), que será utilizado neste trabalho, é um padrão de comunicação pertencente a camada física e a camada de enlace do Modelo *Open System Interconnection* (OSI) — um modelo para protocolos de comunicação da Organização Internacional de Normalização — o endereçamento é feito por meio de uma numeração que é única para cada interface de comunicação. Essa numeração é conhecida como endereço *Media Access Control* (MAC) [Dias *et al.* 2002].

2.3. Sistemas Embarcados

Um sistema embarcado, é classificado desta forma quando este é dedicado a uma única tarefa e interage de forma contínua com o ambiente a sua volta, por meio de sensores e/ou atuadores [Chase 2007].

As principais características de um sistema embarcado são a sua independência de operação e capacidade computacional, pois possuem um processamento de informações originados de um *software* executado internamente na unidade [Chase 2007].

No decorrer deste trabalho será utilizado o Arduino, que é uma placa microcontrolada, não possui um sistema operacional, mas permite uma programação, afim de processar entradas e saídas entre o dispositivo e os componentes externos conectados a ele [McRoberts 2015].

O Raspberry Pi, o qual também será utilizado neste trabalho, possui um sistema operacional que admite, se necessário, realizar as atividades essenciais de um computador convencional, mas ainda assim permite a programação das portas *General Purpose Input/Output* (GPIO).

Embora seja comum associar o Raspberry Pi a uma placa de desenvolvimento como o Arduino, o Raspberry Pi é mais semelhante, por exemplo, aos componentes internos de um *smartphone*. Porém, com a exposição dos pinos de entrada e saída de dados programáveis, em diferentes linguagens de programação [Richardson *et al.* 2013].

2.4. Trabalhos Relacionados

Esta subseção relata os trabalhos relacionados que são inerentes ao tema. São apresentadas algumas abordagens que contribuem para a elaboração da proposta do trabalho.

2.4.1. Exploring IoT Application Using Raspberry Pi

Com a ideia de permitir a troca de informações entre dispositivos de plataformas distintas, [Zhao *et al.* 2015] estabeleceu uma estrutura em que o Raspberry Pi atua como um servidor tolerante a diferentes formas de comunicação. O trabalho explora três cenários diferentes, com comunicação cliente-servidor via Wi-Fi, ZigBee e com enfoque no compartilhamento de arquivos, em que o servidor se mantem disponível para solicitações dos clientes — objetivo principal do projeto.

O primeiro cenário, que consiste na comunicação cliente-servidor por Wi-Fi, é composto de um computador pessoal, um Raspberry Pi B+ e outros dispositivos que são requeridos no contexto como, por exemplo, o roteador de dados.

No segundo cenário, justificando a proposta de diversificar as comunicações entre cliente-servidor, a comunicação acontece por módulos XBee (interface de *hardware* que possibilita a utilização do protocolo de comunicação ZigBee). Nesse ambiente, o trabalho utiliza um módulo XBee para o cliente e outro para o servidor, ambos conectados via *Universal Serial Bus* (USB).

Já no terceiro cenário, há uma preparação do servidor para, além de compartilhar arquivos, responder as solicitações dos clientes. Dessa forma, um servidor *web* — instalado no Raspberry Pi — atua como porta de entrada para as solicitações dos clientes e aguarda uma mensagem para efetuar alguma ação como resposta a solicitação enviada, por exemplo: acionamento de algum dispositivo (atuador).

Em síntese, a principal característica desse projeto é a centralização de diferentes tipos de comunicação no Raspberry Pi. Essa característica permite, por exemplo, que diferentes dispositivos com diferentes interfaces de comunicação de dados (em um mesmo ambiente de IoT) consigam trocar informações entre si. Nesse caso, um dispositivo que utiliza interface Wi-Fi poderia se comunicar com outro dispositivo que usa ZigBee. Isso, em função de que o Raspberry realiza o papel de *gateway*.

2.4.2. Context-aware Dynamic Discovery and Configuration of 'Things' in Smart Environments (CADDOT)

Dentro do contexto de IoT [Perera *et al*. 2013] propõe um modelo de descoberta dinâmica de dispositivos em ambientes inteligentes. O modelo CADDOT, segundo os autores, é composto de 8 fases, são elas: detecção, extração, identificação, localização, recuperação, registro, raciocínio e configuração. Algumas fases são realizadas pela ferramenta desenvolvida (*SmartLink*) e outras são executadas por um *middleware* na nuvem. Ainda assim, há momentos em que fases são executadas coletivamente por ambos, *SmartLink* e *middleware*.

Na primeira fase (detecção) os sensores são configurados para procurar pontos abertos de acesso sem fio — Wi-Fi ou Bluetooth — para que possam se conectar sem a necessidade de autenticação, já que nessa etapa os sensores não possuem configuração personalizada. Um dispositivo com a ferramenta *SmartLink* instalada atua, então, como *hotspot* e permite que um sensor se conecte.

Na fase de extração, o dispositivo com a ferramenta *SmartLink* que, por exemplo, pode ser um *smartphone* ou um Raspberry Pi, recebe a informação enviada pelo sensor e a trata para seguir com a identificação do componente.

A fase de identificação é o momento em que a *SmartLink* envia a informação extraída do sensor recém detectado a um *middleware* IoT na nuvem, que possui uma ontologia com as descrições de inúmeros sensores, visando obter o perfil completo do dispositivo.

A quarta fase, localização, é a fase em que, com base no perfil identificado anteriormente, o *middleware* IoT localiza e retorna ao *SmartLink* um *plug-in*, ou *driver* compatível, capaz de se comunicar com o sensor em questão.

Na fase de recuperação a *SmartLink* já sabe se comunicar com o sensor por meio do *plug-in* enviado pelo *middleware* e então, recupera as informações que o sensor possa

fornecer. A sexta fase, ocorre logo após a coleta desses dados. As informações são então, modeladas, enviadas e registradas na nuvem.

A sétima fase desempenha um papel de grande importância no processo de configuração do sensor. Um plano de programação para o sensor é concebido e o raciocínio ocorre de forma distribuída. O *middleware* IoT da nuvem considera o contexto atual e identifica as disponibilidades e capacidades dos sensores para projetar uma estratégia otimizada.

A última fase contempla, com base no plano de programação, a configuração do sensor. Nesse momento, horário, frequência de comunicação e taxas de amostragem são definidos e enviados ao sensor pela *SmartLink* — via *plug-in*.

Uma visão geral da arquitetura proposta do modelo CADDOT é ilustrada pela Figura 2:

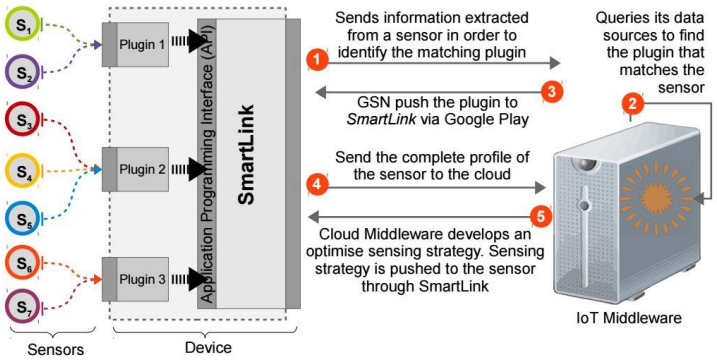

Figura 2. Arquitetura do modelo CADDOT (Perera *et al*. 2013).

2.4.3. Considerações sobre os trabalhos relacionados

A abordagem feita por Zhao *et al.* (2015) é uma alternativa indicada para integração entre diferentes dispositivos, com diferentes tipos de comunicação. Porém, há uma necessidade de configurar, manualmente, cliente e servidor. Esse é o ponto crítico do trabalho, pois as configurações requerem um conhecimento técnico avançado para cada perfil escolhido, tanto do dispositivo quanto do tipo de comunicação pretendido.

O modelo apresentado por Perera *et al.* (2013) não tem foco na comunicação de diferentes dispositivos, mas traz dinamismo na inclusão de novos componentes ao ambiente IoT. A proposta de utilizar uma ferramenta (*SmartLink)* que se conecta na nuvem (*middleware* IoT) em busca de configurações predefinidas, é o que torna a proposta possível. Porém, o modelo é refém de uma plena comunicação com o *middleware*. Nesse caso, uma possível falha no mediador resulta na falha de todo o modelo.

Uma estratégia interessante para IoT seria um terceiro projeto reunir os pontos positivos dos trabalhos abordados, o que resultaria em um sistema capaz de realizar a descoberta dinâmica e configuração remota entre diferentes modelos de dispositivos presentes em um ambiente IoT.

3. Metodologia

De acordo com Fonseca (2002), metodologia é o estudo da organização e dos caminhos a serem percorridos para se realizar uma pesquisa, estudo, ou para se fazer ciência. Em

outras palavras, significa o estudo dos instrumentos utilizados para fazer uma pesquisa científica.

A metodologia a ser implementada para este trabalho é a *Feature Driven Development* (FDD), pelo fato de ser uma metodologia ágil, flexível e de atender as necessidades presentes neste trabalho, que passam desde as etapas de concepção e planejamento e seguem até a construção do projeto.

3.1. FDD — Feature Driven Development

A FDD (Desenvolvimento Guiado por Funcionalidades) é uma metodologia ágil, para gerenciamento e desenvolvimento de *software* que visa um equilíbrio entre as filosofias tradicionais e as mais extremas [Heptagon 2017].

A metodologia FDD consiste na execução de cinco atividades: Desenvolver um Modelo Abrangente, Construir a Lista de Funcionalidades, Planejar por Funcionalidade, Detalhar por Funcionalidade e Construir por Funcionalidade. Cada uma dessas atividades possui tarefas e requisitos a serem executados, porém nenhum de forma obrigatória.

4. Proposta

Conforme relatado nos trabalhos relacionados, a possibilidade de fusão entre as características de dinamismo na descoberta de dispositivos encontradas no trabalho de Perera *et al.* (2013), com as características de diversificação de comunicação presentes no trabalho de Zhao *et al.* (2015), permitirá a criação de uma nova proposta com descoberta dinâmica e configuração remota de dispositivos IoT.

Dessa forma, este trabalho propõe um sistema que simplifique e agilize a conexão entre dispositivos IoT utilizando uma infraestrutura de rede já existente no ambiente em questão — por meio do protocolo de comunicação TCP/IP.

A abordagem consiste de microcontoladores (nós distribuídos) e um agente microprocessado (nó controlador de IoT) que reconheça a sua existência e ofereça os recursos de configuração e comunicação. Uma visão geral da proposta é ilustrada na Figura 3.

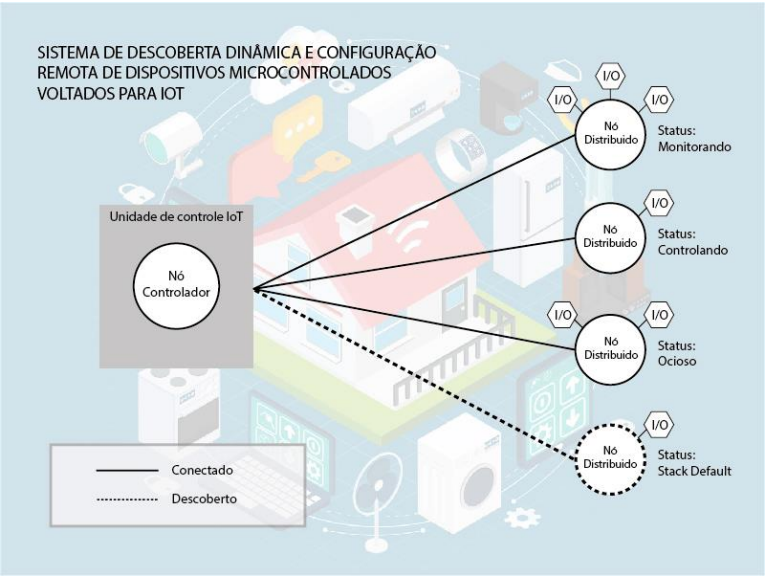

Figura 3. Visão geral da proposta.

O cenário exposto pela Figura 3, ilustra seis dispositivos de entrada e saída (I/O) voltados para IoT, interligados por meio de três nós distribuídos, que por sua vês estão conectados a um nó controlador, integrante da unidade de controle IoT. Na Figura 3 também é ilustrado outro dispositivo de I/O conectado a um quarto nó distribuído, já encontrado, porém não registrado ao nó controlador.

O nó controlador consiste em um agente microprocessado (Raspberry Pi) que se comunica via rede (Ethernet) aos nós distribuídos (Arduino), com a finalidade de configurar e interagir com os dispositivos de I/O conectados.

O nó distribuído recebe, previamente, um sistema padrão — denominado Stack Default — e informações mínimas do ambiente em questão, como o endereço IP do nó controlador. O Stack Default irá permitir que o nó controlador identifique o dispositivo microcontrolado como um integrante do ambiente IoT e então, o reconheça como um novo nó distribuído permitindo a configuração remota e acesso aos dispositivos I/O conectados.

As subseções a seguir apresentam como este trabalho será gerenciado e desenvolvido, de acordo com a metodologia adotada, nas cinco atividades que compõe a metodologia FDD.

4.1. Atividade I — Desenvolver um Modelo Abrangente

Como requisito da primeira etapa da FDD, essa subseção contempla um modelo abrangente da proposta, o qual é ilustrado pela Figura 4, e apresenta o diagrama de processos em um alto nível de abstração.

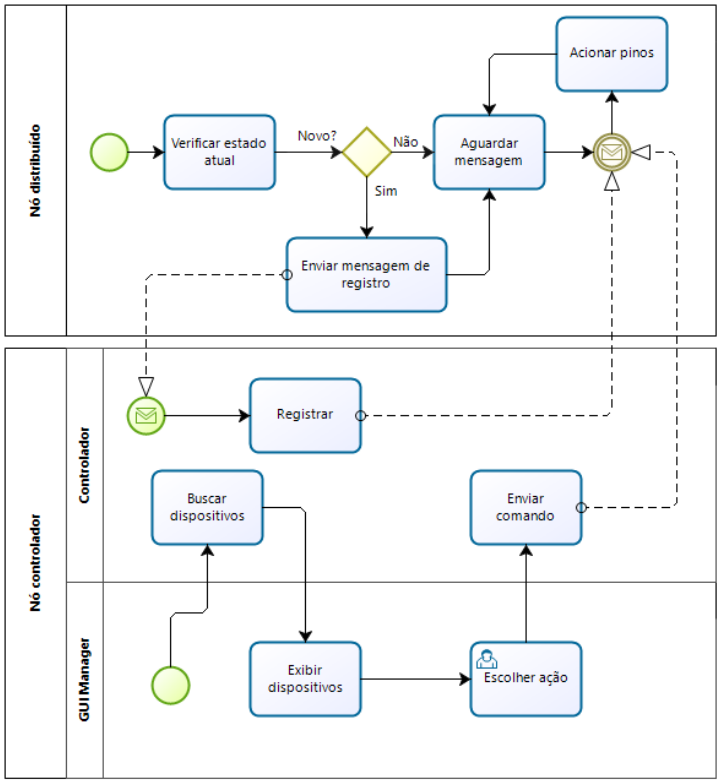

Figura 4. Diagrama de Processos — Modelo Abrangente.

O diagrama ilustrado na Figura 4 possui três raias que representam os principais elementos da proposta e como deverão ser, em um modelo abrangente, seus fluxos dentro do funcionamento do projeto — o nó distribuído com o Stack Default e o nó controlador, como controlador e interface de gerenciamento (GUI *Manager*).

4.2. Atividade II — Construir a Lista de Funcionalidades

Considerando a segunda atividade da metodologia FDD, esta subseção apresenta as funcionalidades disponíveis no sistema do presente trabalho, que são:

- Descobrir dinamicamente novos dispositivos microcontrolados em ambiente IoT;
- Conectar a dispositivos microcontrolados via protocolo TCP/IP;
- Configurar remotamente dispositivos I/O ligados aos nós distribuídos;
- Monitorar remotamente dispositivos I/O ligados aos nós distribuídos;
- Acionar remotamente dispositivo I/O ligados aos nós distribuídos;
- Desconectar dispositivos microcontrolados em desuso no ambiente IoT.

4.3. Atividade III — Planejar por Funcionalidade

Nesta subseção há o planejamento por funcionalidade, sugerido pela metodologia FDD em sua terceira atividade, representado na Figura 5 pelo Gráfico de Gantt.

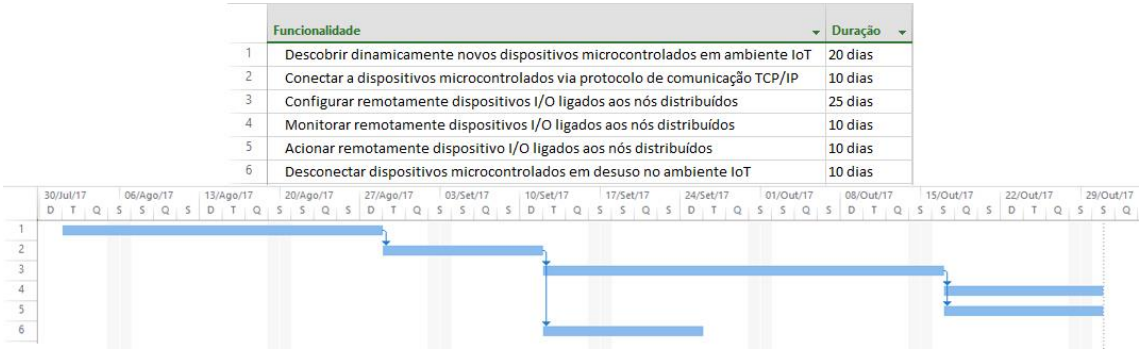

Figura 5. Planejamento por Funcionalidade.

4.4. Atividade IV — Detalhar por Funcionalidade

A união entre a visão geral e o modelo abrangente, desenvolvido na Atividade I do FDD, contribuiu para que se obtivesse um novo diagrama de processos, detalhando as funcionalidades da proposta.

Ainda que em outra atividade, é perceptível a manutenção das três raias contidas no diagrama de processos do modelo abrangente, porém, o detalhamento exigido na Atividade IV permite que sejam utilizados elementos da visão geral voltados diretamente para a implementação do projeto, por conta disso, os termos Nó distribuído e Nó controlador são substituídos por Arduino e Raspberry Pi respectivamente — dispositivos que foram utilizados no cenário de teste e validação.

O diagrama de processos detalhado é apresentado na Figura 6.

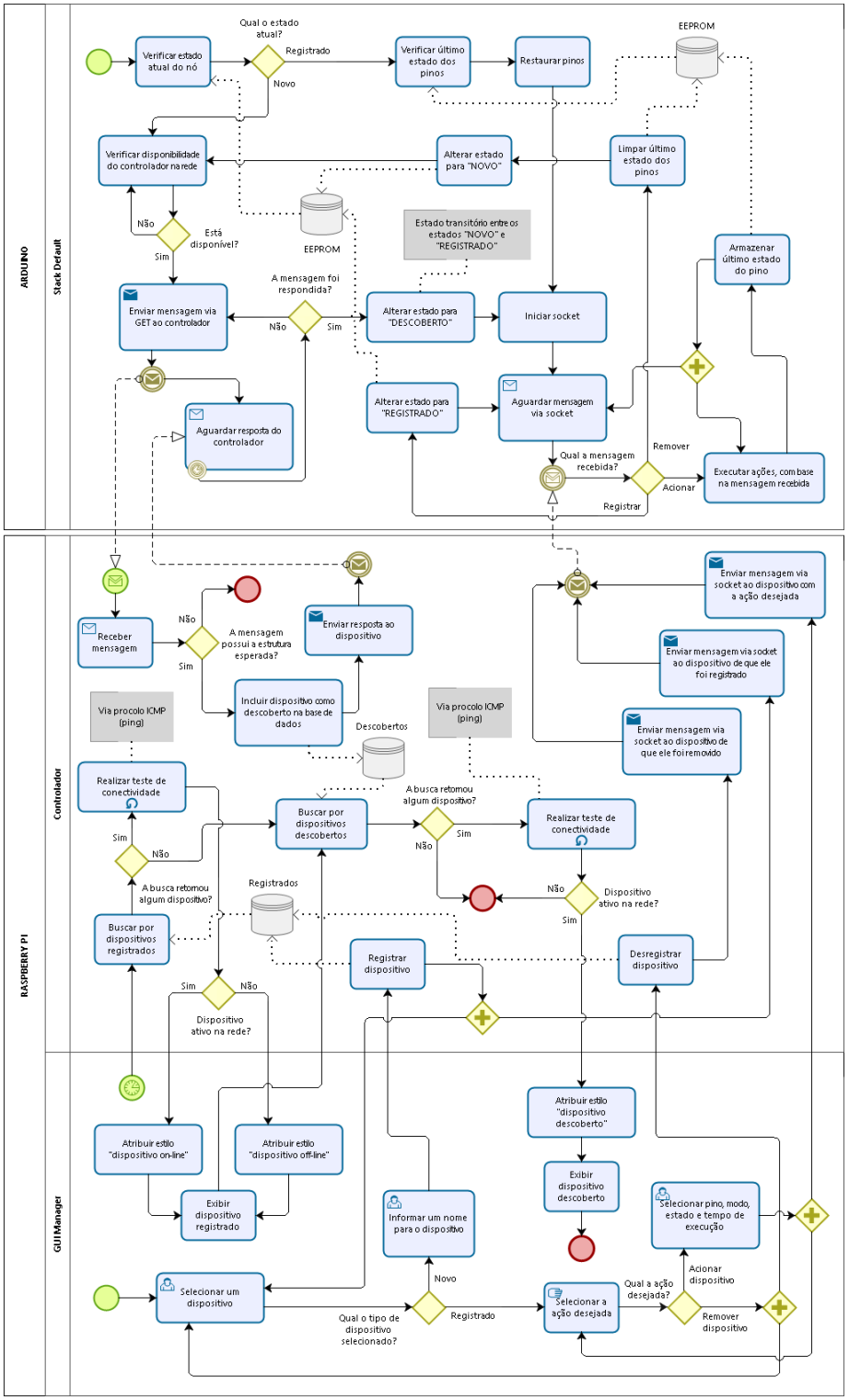

Figura 6. Diagrama de Processos — Detalhado.

4.5. Atividade V — Construir por Funcionalidade (Implementação)

Na quinta e última atividade segundo a metodologia FDD, tem-se a construção das funcionalidades com base nos elementos elencados na lista de funcionalidades (Atividade II) tendo como referência o detalhamento obtido na Atividade IV.

As subseções a seguir descrevem a implementação das três raias que compõem o diagrama de processos apresentado pela Figura 6.

4.5.1. Arduino — Stack Default

Assim que ligado, o Stack Default desenvolvido e instalado no Arduino, realiza a verificação no endereço pré-estabelecido (endereço 191) da memória EEPROM (tipo de memória não-volátil usada em dispositivos eletrônicos, armazena dados que precisam ser salvos quando a energia é removida) em busca do atual estado do nó distribuído, verificando assim se o dispositivo é novo ou já foi registrado por algum controlador (valor 0 para novo e 1 para registrado).

Caso o dispositivo já faça parte de um ambiente IoT, uma nova leitura da memória EEPROM é realizada para verificar o estado em que os pinos estavam quando o Arduino foi desligado pela última vez, desta forma os restaura se for necessário antes de abrir uma conexão *socket* (fluxo de comunicação entre processos através de uma rede) na porta pré-definida (2807).

A Figura 7 ilustra o método *lerEstadoPino()* que restaura os pinos correspondentes após varrer a memória do endereço 100 ao endereço 100 + (D_PIN - 1), aonde D_PIN é uma constante com o número de pinos digitais disponíveis — os endereços de memória foram estabelecidos pensando na usabilidade e no *range* disponível (0 a 1023 no caso do Arduino UNO).

```
void lerEstadoPino(){
  for(int i=0; i<D_PIN; i++){
    pinMode(i,OUTPUT);
    digitalWrite(i,EEPROM.read(100+i));
  }
}
```

Figura 7. Método *lerEstadoPino().*

Se o dispositivo for um novo candidato a ingressar no ambiente IoT, inicia-se uma tentativa de conexão ao controlador com o intuito de enviar um pacote através do protocolo TCP via método GET com uma mensagem contendo a TAG de identificação do dispositivo, seu número de série e endereço IP, separados pelo delimitador pré-definido "@" conforme a Figura 8.

```
bool conn = false;

do{
  if (client.connect(server, 80)) {
    client.println("GET /detecta.php?dispositivo="+tag+"@"+numSerie+"@"+getIp());
    client.println("Connection: close");
    conn = true;
  } else {
    delay(3000);
  }
}while(!conn);
```

Figura 8. Conexão e envio de mensagem ao controlador.

Caso a conexão não seja estabelecida ou o controlador não responder adequadamente, uma nova mensagem é enviada após três segundos até que haja sucesso na solicitação, permitindo que um *socket* seja aberto (porta 2807) para aguardar os pacotes enviados pelo controlador, sejam eles de registro, acionamento ou remoção.

As mensagens contidas nesses pacotes são montadas pelo controlador e variam de acordo com cada ação — mesmo assim possuem uma estrutura definida e utilizam o delimitador "@".

Uma mensagem iniciada por “1@” indica que o controlador está registrando o Arduino no seu ambiente, a mensagem deve estar acompanhada do ID de registro em sua base de dispositivos (“1@ID”), este ID também será armazenado na memória EEPROM do Arduino no momento da alteração de estado (novo para registrado) — o ID é utilizado para conferencia em caso de remoção do dispositivo.

A mensagem de remoção tem a estrutura “0@ID”, aonde o Arduino verifica na memória EEPROM se o ID enviado é o mesmo ID fornecido no momento do registro, garantindo que a ordem de remoção é, de fato, para ele.

Mensagens iniciadas por “2@” são mensagens de acionamento, e possuem uma estrutura definida por “2@PINO@MODO@ESTADO@TEMPO”. O número do pino desejado deve estar entre os primeiros delimitadores, o modo (“OUT” para *output* e “IN” para *input*) entre o segundo e terceiro delimitador, o estado (“H” para *high* “L” para *low*) entre o terceiro e o quarto delimitador, enquanto o último elemento é o tempo de execução da ação, em segundos — ações permanentes são representadas por 0 (zero).

A Figura 9 apresenta o método *comando()*, executado quando a mensagem recebida no pacote via *socket* for de acionamento, neste momento a mensagem já foi tratada e seus fragmentos são passados por parâmetros para compor a execução.

```
void comando(int _pino, String _mode, String _estado, int _tempo){
  pin[_pino] = digitalRead(_pino);

  if(_mode == "OUT") pinMode(_pino,OUTPUT);
  else if (_mode == "IN") pinMode(_pino,INPUT);
  else { return; }

  if(_estado == "H") {
    digitalWrite(_pino,HIGH);
    gravarEstadoPino(_pino, HIGH);
  }
  else if(_estado == "L") {
    digitalWrite(_pino,LOW);
    gravarEstadoPino(_pino, LOW);
  }
  else { return; }

  if(_tempo > 0){
    tPino[_pino].setInterval(_tempo*1000);
    tPino[_pino].enabled = true;
  }
}
```

Figura 9. Método *comando()*.

No momento em que um pino é acionado seu novo estado é gravado na memória EEPROM (método *gravarEstadoPino()*) para tornar o estado legível após o reinício do sistema no Arduino.

É possível perceber pelo código exibido na Figura 9 que se o tempo de execução for informado, uma *thread* virtual é habilitada — um vetor de pinos (pin[]) é associado a um vetor de *threads* (tPino[]) no carregamento do Stack Default — desta forma o Arduino não interrompe sua execução durante o tempo de acionamento e fica disponível para novas tarefas (esperando novos pacotes via *socket*), dando a impressão de paralelismo.

4.5.2. Raspberry Pi — Controlador

Conforme o diagrama de processos da Figura 6 apresenta, o Raspberry Pi é responsável por duas partes da implementação, uma é atuar como controlador e outra é disponibilizar uma interface de gerenciamento ao usuário (GUI *Manager*).

As tarefas como controlador sempre são decorrentes de um acionamento externo, seja via protocolo TCP, disparado pelo nó distribuído, ou pelo usuário na utilização da interface de gerenciamento. Por conta disso o Raspberry Pi com o sistema operacional Raspbian (derivado do Debian), possui instalado o Apache e o MySQL para interpretação da linguagem PHP e acesso ao banco de dados.

No momento em que o Raspberry Pi receber uma mensagem de algum Arduino, via método GET e endereçada ao arquivo *detecta.php*, desde que contenha a estrutura esperada para uma nova descoberta, o controlador armazena esta mensagem na base de dados dos dispositivos descobertos e informa ao Arduino que já o descobriu, isto faz com que o dispositivo pare de enviar novas mensagens até que algum usuário intervenha no ambiente IoT.

Assim que a interface de gerenciamento é carregada pelo usuário, um *script* inicia a listagem e testes de conectividade, esse *script* é executado de forma automática a cada três segundos enquanto a interface estiver aberta ou sempre que alguma ação for tomada pelo usuário.

O teste de conectividade é realizado em dispositivos descobertos e registrados inseridos na base de dados, o intuito é saber sua disponibilidade na rede através do protocolo ICMP, pelo o IP do dispositivo na porta 2807, utilizando o método *fsockopen()* do PHP, o resultado do tipo *bool* é armazenado no atributo "online" como ilustra a Figura 10.

```php
$online = @ fsockopen($descoberto['ip'], 2807, $errno, $errstr, 1);
```

Figura 10. Teste de conectividade.

Assim que o usuário escolher registrar um Arduino descoberto, o controlador deve armazenar a mensagem recebida no ato da descoberta, de forma fragmentada, na base de dados dos dispositivos registrados e informar o dispositivo em questão, via *socket*, que ele agora está registrado — mensagem "1@ID".

A Figura 11 apresenta como a mensagem com o ID de registro é construída e enviada ao dispositivo.

```php
$idRegistro = $mysqli->query("SELECT LAST_INSERT_ID()")->fetch_array(MYSQLI_NUM)[0];
$socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
$porta = 2807;
$mensagem = "1@$idRegistro";
$recebido = null;

socket_connect($socket, $ip, $porta);
socket_write($socket, $mensagem);

while ($out = socket_read($socket, $porta)) {
    $recebido = $recebido.$out;
}
```

Figura 11. Construção e envio de mensagem.

Ainda na Figura 11, tem-se o retorno ao Raspberry Pi do que foi recebido pelo Arduino. Esta validação é utilizada em todos envios de mensagens como forma de garantir a integridade da mensagem em uma comparação estrita, antes que o sistema siga sua execução — utilizando o operador de equivalência (recebido === mensagem).

Caso seja solicitado pelo usuário o acionamento de algum pino do Arduino, a mensagem é construída com o início "2@" e completada com base nas seleções realizadas na interface de gerenciamento conforme a estrutura já mencionada no tópico 4.5.1.

Assim como nas mensagens de acionamento, a remoção do dispositivo quando solicitada pelo usuário (construída com “0@ID”) utilizam o mesmo protocolo e padrão adotado na mensagem de registro, de acordo com o código da Figura 11, tanto para envio quanto para teste de integridade.

4.5.3. Raspberry Pi — GUI Manager

A interface de gerenciamento utiliza componentes *web* e carrega os elementos resultantes dos *scripts* PHP, estruturados pela linguagem marcação HTML5 e estilizada pela linguagem CSS3. Seu comportamento é dinâmico e se modifica automaticamente assim que houver alguma alteração no ambiente IoT.

A indicação “Off-line” e o tema sem cor são aplicados ao dispositivo registrado quando seu teste de conectividade falhar, a indicação “On-line” e um tema colorido são aplicados quando este teste obtiver resposta. Para um dispositivo descoberto, uma imagem em alusão a um dispositivo sem identificação é utilizada, juntamente com o nome temporário de “Novo dispositivo”. A Figura 12 ilustra as três variações possíveis para os dispositivos.

Figura 12. Tela de dispositivos.

A tela de registro para dispositivos descobertos, possui apenas um campo editável, de caráter obrigatório, e refere-se ao nome que este dispositivo é identificado pelo utilizador do sistema.

A tela de controle de dispositivo lista os pinos disponíveis do Arduino para que o usuário escolha qual será acionado, seguido do modo, tipo e tempo de acionamento conforme ilustra a Figura 13. Todos os campos são obrigatórios já que são informações requeridas para a construção da mensagem a ser enviada pelo controlador.

Figura 13. Tela de controle.

Nesta mesma tela é possível remover o dispositivo do ambiente atual a fim de torna-lo um dispositivo disponível para novas detecções.

5. Cenário de Teste e Validação

Diante do objetivo deste trabalho, optou-se por construir um cenário de teste e validação que simulou um ambiente IoT heterogêneo e que possibilitou a inclusão e exclusão de novos microcontroladores.

Dessa forma, foram utilizados dispositivos microcontrolados Arduino (com *shield* de comunicação Ethernet), realizando a função de nós distribuídos, agindo como âncora para elementos ligados a seus pinos digitais — operando como dispositivos I/O. O nó controlador foi atribuído a um dispositivo microprocessado Raspberry Pi, que além de realizar as tarefas atribuídas ao controlador, recebe as solicitações realizadas pelo usuário, via interface gráfica.

O cenário permitiu ao Raspberry Pi descobrir dispositivos Arduino assim que foram conectados à rede e também se comportou como o esperado, tolerando os testes de acionamento remoto dos pinos, independente do dispositivo.

A construção dos elementos, ligados ao Arduino, foi baseada em uma *Protoboard* (matriz de contato com furos e conexões condutoras), a comunicação pela rede foi estabelecida utilizando um *switch* conforme a Figura 14 pode ilustrar.

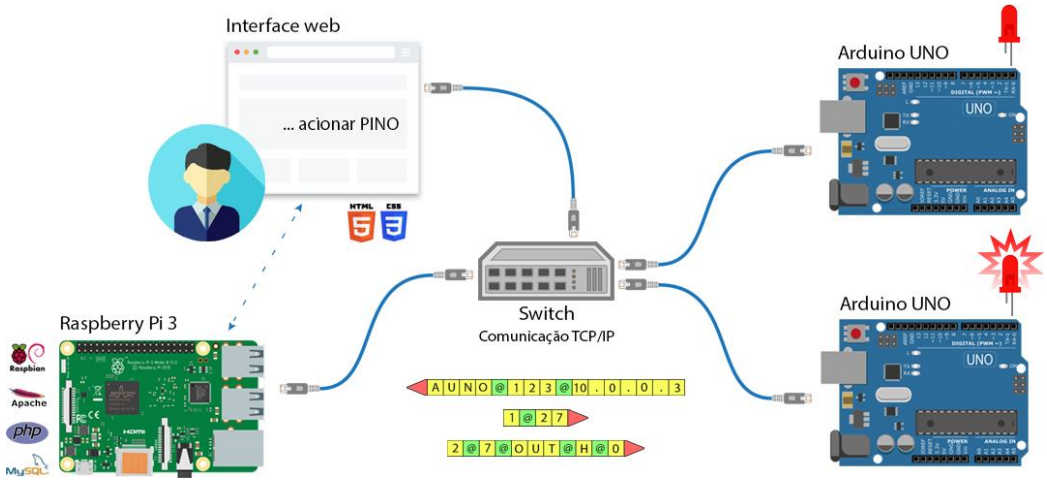

Figura 14. Cenário de teste.

Este ambiente simulado permitiu que fosse avaliado o comportamento do sistema em circunstâncias semelhantes de um ambiente IoT real, mas também, por se tratar de uma escala reduzida, proporcionou flexibilidade para inclusão de novas situações durante o desenvolvimento (exibir informações em um *display* de cristal líquido e *shield* de rede Wi-Fi por exemplo).

6. Conclusão

O tema IoT é recente, por conta disso, abre oportunidade para diversas linhas de pesquisa que devem ser exploradas até que o tema se consolide e torne parte de algo maior, ainda desconhecido. Este trabalho atuou no modo como os novos dispositivos vão se comportar em ambientes IoT já existentes. O desenvolvimento desta proposta trouxe uma dinamicidade necessária à IoT, sobretudo, inteligência na forma em que novos dispositivos devem ser utilizados e configurados por usuários experientes ou não.

A implementação e os testes do projeto ressaltam que mesmo uma estrutura diversificada, composta por troca de informações entre dispositivos distintos e diferentes

linguagens de programação, são capazes de disponibilizar ao usuário um sistema que seja intuitivo e de fácil utilização.

Sendo assim, o trabalho atingiu seu objetivo ao abordar a descoberta dinâmica e configuração remota de dispositivos, com uma interface de gerenciamento centralizada em ambiente *web*, flexibilizando a expansão da ferramenta. Desta forma, abre horizontes para que trabalhos futuros implementem novas funcionalidades que não foram mapeadas nesta abordagem.

Um trabalho futuro poderá conter, por exemplo, nó controlador tolerante a falhas, para que o sistema continue operando caso o controlador principal esteja indisponível. Além disso, mensagens de *input* e ações nos pinos analógicos dos nós distribuídos possuem a base definida na atual versão, porém, ainda são limitações do projeto e não foram implementadas, nas quais também poderão ser exploradas em um trabalho futuro.

Referências

Chase, Otavio (2007) "Sistemas Embarcados", SBAJovem 2010. Disponível em: <http://www.lyfreitas.com.br/ant/pdf/Embarcados.pdf>. Acesso em 24 de abril de 2017.

Dias, Betthovem, Z.; Alves, Nilton (2002) "Evolução do Padrão Ethernet", Centro Brasileiro de Pesquisas Físicas, Rio de Janeiro.

Evans, Dave (2011) "The Internet of Things How the Next Evolution of the Internet Is Changing Everything", Cisco Internet Business Solutions Group (IBSG).

Fonseca, João, J. S. (2002) "Metodologia da pesquisa científica", Universidade Estadual do Ceará, Fortaleza.

Forouzan, Behrouz, A. (2009) "Comunicação de Dados e Redes de Computadores", AMGH Editora.

Heptagon, TI Ltda. (2017) "FDD – Feature Driven Development". Disponível em: <http://www.heptagon.com.br/fdd>. Acesso em: 03 de Maio de 2017.

McRoberts, Michael (2015) "Arduino Básico - 2ª Edição", Novatec Editora, São Paulo.

Perera, Charith; Jayaraman, Prem, P.; Zaslavsky, Arkady; Christen, Peter; Georgakopoulos, Dimitrios (2013) "Context-aware Dynamic Discovery and Configuration of 'Things' in Smart Environments", Disponível em: < https://arxiv.org/abs/1311.2134>. Acesso em: 13 de março de 2017.

Rangel, Rogério (2014) "Internet das Coisas, nova revolução da conectividade", Revista Inovação em Pauta – Finep, Nº 18.

Richardson, Matt; Wallace, Shawn (2013) "Getting started with Raspberry Pi", Make Magazine, O'Reilly Media.

Zhao, Cheah, W.; Jegatheesan, Jayanand; Loon, Son, C. (2015) "Exploring IOT Application Using Raspberry Pi", International Journal of Computer Networks and Applications 2015, Volume 2.

Zyga, Lisa (2009) "Internet Growth Follows Moore's Law Too". Disponível em: <https://phys.org/news/2009-01-internet-growth-law.html>. Acesso em: 02 de abril de 2017.