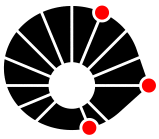

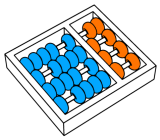

Dispositivo IoT para Coleta Inteligente de Pilhas e Baterias

Tiago Eidi Hatta *Juliana Freitag Borin*

Relatório Técnico - IC-PFG-18-12
Projeto Final de Graduação
2018 - Julho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Dispositivo IoT para Coleta Inteligente de Pilhas e Baterias

Tiago Eidi Hatta Juliana Freitag Borin*

Resumo

A Prefeitura Universitária da Unicamp, juntamente de docentes e discentes, tem desenvolvido a iniciativa *Smart Campus*, que tem como objetivo utilizar o conceito de Internet das Coisas no contexto da universidade. Assim, em busca de um campus inteligente, diferentes projetos têm sido trabalhados atualmente. Um deles é denominado Coleta Diferenciada e busca prover o recolhimento eficiente de pilhas e baterias pelo campus de Barão Geraldo. A ideia deste projeto foi, portanto, desenvolver um protótipo para otimizar esse processo, de forma que os coletores tenham um sistema robusto de sensoriamento e que possam enviar dados para a internet. Através disso, as informações podem ser processadas para que se obtenha uma rota de coleta eficaz.

1 Introdução

Internet of Things (IoT), ou em português Internet das Coisas, é um conceito que se refere a interconexão de objetos presentes em nosso dia a dia, mas que apresentam um certo tipo de inteligência. Em termos técnicos, tais objetos são sistemas embarcados conectados a rede, que recebem e enviam dados do mundo real, e que a partir deles tomam decisões para determinadas necessidades. O termo *Internet of Things* foi introduzido por Kevin Aston em 1999 no contexto de gestão de *Supply Chain* e desenvolvimento da tecnologia *RFID*[1]. Aliás, a popularização de aparelhos aptos a se comunicarem via rede sem fio em diferentes tipos de tecnologias como *Bluetooth, NFC, WiFi* e serviços de dados (3G, 4G e, mais recentemente, 5G) tem sido determinante para o aumento no interesse do desenvolvimento da Internet das Coisas, inclusive com a criação de *Low-Power Wide-Area Network* (LPWAN), redes sem fio com maior alcance e menor consumo de energia. Em um sistema IoT tudo se comunica, tudo é identificado e tudo interage [2]. Cada dispositivo deve ter a capacidade de se conectar com os outros, transferindo informações para determinadas aplicações. A partir disso, aspectos relevantes para IoT estão no custo de produção, no sensoriamento correto do ambiente, na eficiência energética, na conexão segura e constante para envio de dados, e no manuseio da informação para tomada de decisão. Esses pontos são essenciais para que se possa fornecer produtos e serviços acessíveis e confiáveis para a sociedade.

Um outro conceito que tem sido foco de pesquisas e discussões é denominado *Smart Cities*. Ele aborda soluções para os diferentes tipos de problemas gerados nas cidades. Seja

*Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP

na dificuldade para gestão de resíduos, limitação de recursos, poluição, questões de saúde pública, trânsito, entre outros. A Internet das Coisas tem papel essencial nas cidades inteligentes, ao prover dados através de sensoriamento do ambiente e distribuir essas informações através de uma arquitetura de rede urbana [3].

Uma implementação de cidade inteligente é a *Padova Smart City* [4], na Itália. Nela, existe uma rede com mais de 300 nós, entre diferentes tipos de dispositivos, para prover dados à administração da Universidade de Padova, que pode então atuar para monitorar poluição e iluminação de pontos estratégicos do campus. Atualmente, a Prefeitura Universitária da Unicamp tem uma iniciativa similar. É o programa *Smart Campus*[1], que busca implementar soluções IoT em favor do bem estar da comunidade. Entre os projetos desenvolvidos estão o fornecimento a localização em tempo real do ônibus circular e o gerenciamento e o controle da iluminação de postes de iluminação pública da Unicamp.

Outro projeto no contexto do *Smart Campus* é a coleta de pilhas e baterias de forma eficiente. As pilhas e baterias são componentes amplamente consumidos no dia a dia. Segundo o Relatório de Pesquisa Industrial Mensal do IBGE (Abril/2018)[2], nos últimos 12 meses, considerando a data de publicação do documento, a fabricação de pilhas, baterias e acumuladores elétricos teve crescimento de 18.4%. Esses dispositivos podem conter em sua composição metais pesados, como mercúrio, cádmio e chumbo, extremamente danosos ao meio ambiente. O Conselho Nacional do Meio Ambiente (CONAMA) estabeleceu a resolução 401/2008[3] definindo limites máximos de tais materiais na produção de pilhas e baterias comercializadas no território nacional, além de critérios e padrões para o seu gerenciamento ambientalmente adequado. Já a Instrução Normativa Ibama n° 8 (30 de setembro de 2012)[4] institui, para fabricantes nacionais e importadores, os procedimentos relativos ao controle do recebimento e da destinação dos componentes elétricos ou de produtos que os incorporem. Assim sendo, a Unicamp mantém vários coletores (Fig. 1) espalhados pelo campus de Barão Geraldo, para que possa destinar os materiais corretamente aos centros de reciclagem. Entretanto, atualmente, a cada coleta, os funcionários precisam passar pela universidade inteira, em um trajeto que despende tempo e recursos físicos desnecessários, pois muitos coletores podem estar vazios ou com pouca capacidade preenchida durante o trajeto.

Desse modo, esse projeto tem como objetivo a criação de um sistema IoT que auxilie no processo de coleta de pilhas e baterias no campus da Unicamp. Para tanto, foram estabelecidos os seguintes objetivos específicos:

- Pesquisa e escolha de componentes que viabilizassem a criação de um protótipo IoT, considerando custo, eficiência energética e facilidade de implementação.

[1] *Smart Campus* Unicamp: http://smartcampus.prefeitura.unicamp.br/

[2] Relatório de Pesquisa Industrial Mensal do IBGE (Abril/2018): https://biblioteca.ibge.gov.br/visualizacao/periodicos/228/pim_pfbr_2018_abr.pdf

[3] CONAMA Resolução 401/2008: http://www.mma.gov.br/port/conama/legiabre.cfm?codlegi=589

[4] Instrução Normativa Ibama n° 8: http://www.ibama.gov.br/sophia/cnia/legislacao/IBAMA/IN0008-030912.PDF

- Programação do protótipo para coleta e envio de dados a um sistema na nuvem que monitore a capacidade dos coletores.
- Criação de um sistema amigável que apresente rotas eficientes de coleta.
- Realização de testes e melhorias no protótipo.
- Disponibilização do protótipo e do sistema desenvolvido para a equipe do projeto *Smart Campus*.

Figura 1: Coletor Instalado no Campus da Unicamp

2 Metodologia

2.1 Pesquisa de Componentes

Inicialmente, foi realizado um levantamento de requisitos necessários para o projeto:

- Sensor para medição da capacidade do coletor;
- Dispositivo para requisitar as leituras e receber os dados
- Dispositivo para enviar informações para a nuvem, que tenha habilitada uma conexão sem fio
- Fontes de alimentação para outros componentes

Através disso, foram levantados diferentes opções atendendo a cada requisito, com o recolhimento de informações como preço, disponibilidade no mercado, eficiência energética, portabilidade, facilidade de instalação e acessibilidade de informações.

Sensor	Tipo	Tensão Necessária	Intervalo de Valores	Preço[6]
HC-SR04[7]	Ultrassônico	5V	2 cm - 4 m	R$15,00
GP2Y0A21YK0F[8]	Infravermelho	4.5 - 5.5V	10 cm - 80 cm	R$49,00
Célula de Peso 50kg	Peso[9]	5 - 10V	0 Kg - 50 Kg	R$15,00

Tabela 1: Tabela 1 - Opções de Sensores para o Projeto

2.2 Gerenciamento de Dados

Existem serviços web focados em gerenciar dados de dispositivos IoT. Em consenso com a Prefeitura Universitária foi definido qual seria utilizado. Além disso, determinou-se qual o protocolo de comunicação em que as mensagens seriam enviadas. Isso foi feito baseado nas alternativas suportadas pela plataforma web.

2.3 Criação de um Sistema Web

Para desenvolvimento web foram pesquisados *frameworks* para desenvolvimento ágil de sites e que fossem integrados com a plataforma de dados IoT e a API do *Google Maps*[5], além de se conectar a um banco de dados.

2.4 Testes e Melhorias

O protótipo foi testado inicialmente fora do coletor, para verificar a exatidão das distâncias calculadas pelo sensor. Em seguida foram observados o recebimento dos dados pela plataforma web. Após a fase inicial, um protótipo foi instalado em um coletor localizado no Instituto de Computação (IC-3) na Unicamp. Foram encontrados problemas e desafios referentes à medição de distâncias, envio de dados para a nuvem e eficiência energética. A partir disso, adaptações foram feitas para resolver essas questões.

3 Resultados e Discussão

3.1 Componentes do Protótipo

Foram encontrados três possíveis sensores para medição da capacidade dos coletores. A tabela 1 mostra informações sobre o sensor ultrassônico HC-SR04, o sensor infravermelho GP2Y0A21YK0F e a célula de peso 50Kg.

A primeira opção descartada foi a célula de peso. A principal razão foi a incerteza de como se daria o processo de coleta das pilhas e baterias, se a tampa superior ou fundo seria removido, ou então se haveria a abertura frontal do coletor (Fig. 1). Enfim, devido a essa questão estrutural, instalar um sensor de peso na parte inferior do coletor era muito

[5]API Google Maps: https://developers.google.com/maps/?hl=pt-br
[6]Baseados no site https://www.robocore.net em Junho/2018
[7]Cytron Techonologies - Product User's Manual – HCSR04 Ultrasonic Sensor
[8]Datasheet do Sensor Sharp GP2Y0A21YK0F
[9]Datasheet da Célula de Peso 50Kg https://www.sparkfun.com/datasheets/Sensors/loadsensor.pdf

arriscado. Além disso, a célula retorna valores analógicos de baixo sinal e que requerem um módulo amplificador separado, como o Conversor HX711[10]. Dentre os outros dois sensores, o infravermelho retorna valores de no mínimo 10 cm. Observa-se na Figura 2, uma ilustração de acordo com medições realizadas em um dos coletores (Fig 1). A distância entre o protótipo (em laranja) e a altura de maior capacidade do balde é de 9 cm. Nessa faixa de valor, o GP2Y0A21YK0F é instável. Além disso, o sensor ultrassônico (Fig. 3) é mais barato. Devido a essas razões, o HC-SR04 foi escolhido para o projeto.

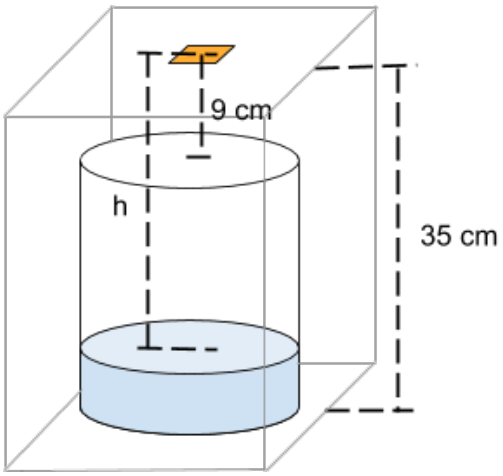

Figura 2: Desenho esquemático de dentro do coletor (Fig. 1). Em laranja está o protótipo e em azul a capacidade hipotética preenchida. O sensor retorna a distância h.

Figura 3: Sensor Ultrassônico HC-SR04

[10] *24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales Datasheet*: https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf

Figura 4: Plataforma de desenvolvimento NodeMcu

Para conseguir controlar e enviar dados para a nuvem, foi escolhido a NodeMCU[11] (Fig. 4). Essa plataforma *open source* consiste em um kit de desenvolvimento composto pelo microcontrolador ESP8266, o módulo WiFi ESP-12F[12], pinos digitais e interface USB para sua programação.

Um outro ponto positivo da NodeMCU é sua boa popularidade entre o movimento maker (*Do-It-Yourself*) que compartilha diversas informações de diferentes projetos na internet o que torna seu uso acessível. Para a programação, foi utilizada a IDE da plataforma Arduino[13], que tem uma linguagem baseada em C++ e várias bibliotecas já implementadas.

Para alimentar esses componentes, foram levantadas as seguintes opções:

- Bateria Alcalina 9V
- Fonte Bivolt Saída 5V com cabo micro usb
- Pilhas Alcalinas AA 1.5V
- Pilhas de Lítio 18650 3.7V
- *Power* Bank 5V e interface micro usb (Fig. 5)

A ESP8266 tem tensão de funcionamento entre 2.5V e 3.6V[14]. Já o sensor HC-SR04 requer cerca de 5V. Portanto, são necessárias duas fontes distintas de alimentação ou apenas uma que acompanhe um regulador de tensão. A NodeMCU já conta com um regulador de tensão em sua porta Vin e micro usb que aceita até 12V (recomendado 9V). Por isso, optou-se por utilizar um conjunto de 3 pilhas alcalinas AA ou um *power bank*. Essa última opção apesar de ter menor capacidade em mAh, permite que possa ser recarregado. A bateria alcalina teria um desperdício maior de tensão e a fonte, apesar de funcionar, limita o protótipo a ambientes que tenham tomada elétrica.

[11]Site do NodeMcu: http://nodemcu.com/index_en.html

[12]Descrição da ESP-12F: http://www.electrodragon.com/w/ESP-12F_ESP8266_Wifi_Board

[13]Site Oficial Arduino: https://www.arduino.cc/

[14]Espressif ESP8266EX Datasheet: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf

Figura 5: Power Bank 5V

Pino NodeMCU	**Pino HC-SR04**
D2	Trigger
D5	Echo

Tabela 2: Tabela 2 - Ligação de Pinos entre NodeMCU e HC-SR04

O circuito de funcionamento (Fig. 6) conta com uma ligação entre pinos do NodeMCU com o sensor (Tabela 2) e a fonte de energia. Para o cálculo da distância é necessário transmitir um pulso de cerca de 10 us através do trigger. Com isso, oito pulsos ultrassônicos de 40 kHz são emitidos pelo sensor ultrassônico, que recebe um sinal de resposta (alto) redirecionado para o pino echo. Através do tempo do sinal em *HIGH* é possível calcular a distância pela fórmula:

$$distância = t_{alto} \cdot v_{som}{}^{15} \tag{1}$$

Para receber o dado do sensor em echo foi implementado um divisor de tensão com 1k e 2k ohms, de forma que a porta D5 do NodeMCU não seja danificada pela tensão de 5V. No caso das pilhas, a conexão é feita no pino Vin, mesmo utilizado para alimentar o sensor ultrassônico.

Para economizar energia, é programado o modo de economia de energia *deep sleep*. Nele Rádio *WiFi/Wireless*, CPU e CLOCK ficam desligados, Apenas o RTC (*Real Clock Time*) continua funcionando. O RTC tem no entanto uma limitação de até cerca de 70 minutos. Para contornar essa situação, o programa contém um contador de horas salvo na memória persistente da plataforma.

O NodeMCU tem um espaço de memória alocado para guardar arquivos, que mesmo com a placa desligada não são apagados. Para gerenciar isso, existe o sistema de arquivo

[15]Velocidade do som igual a aproximadamente 340 m/s.

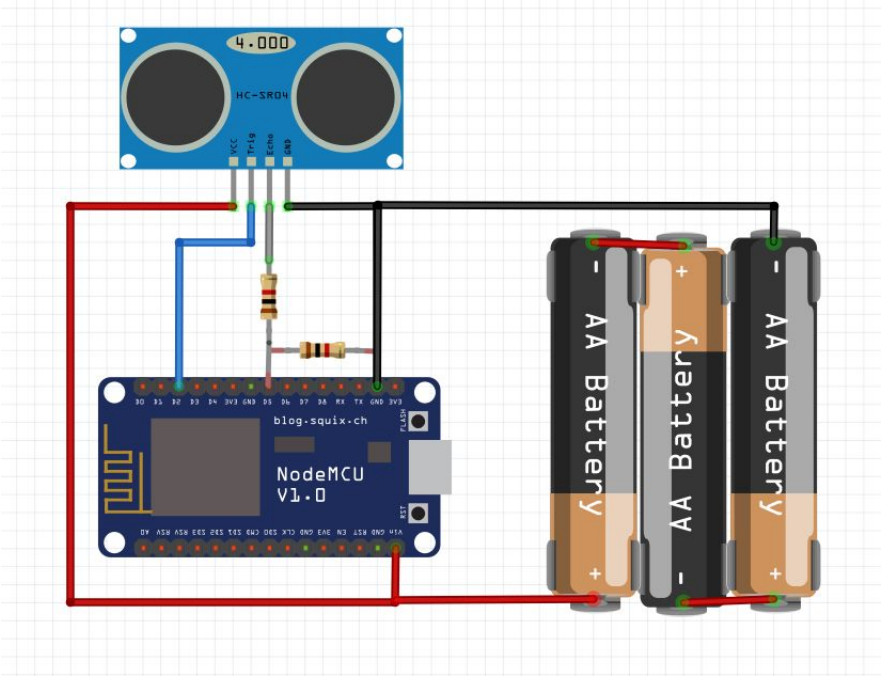

Figura 6: Circuito do Protótipo

SPIFFS (*SPI Flash File System*)[16]. Ele se baseia em controlar dados em memórias SPI-NOR flash, ou seja, fazer a interface do usuário com as células de armazenamento baseadas em portas lógicas NOR e interface SPI (*Serial Peripheral Interface*).

O sistema é programado para acordar a cada 60 minutos e verificar o valor do contador. Caso seja igual a 12, então a variável é zerada e passa a ser realizada a medição de 100 valores da distância. Caso um valor absurdo seja medido, como maior da altura máxima do coletor, ou menor do que 0, o dado é descartado. A média desses valores é então determinada como a distância considerada. O processador é acordado por interrupção pelo RTC através de um sinal do pino D0 para RST (*Reset*).

Código Principal

```
void setup() {
  // Contador de horas
  int sleepCounter = 0;

  // SPIFSS File System
  SPIFFS.begin();

  // Inicialização da Serial
```

[16] *ESP8266 Arduino Core - File System*: http://esp8266.github.io/Arduino/versions/2.0.0/doc/filesystem.html

```
  Serial.begin(115200); /
  Serial.setTimeout(2000);
  while (!Serial) {}
  Serial.println("");
  Serial.println("");

  // Recupera o contador da memória persistente
  if (memManager.recoverIntVariable(&sleepCounter, "/store/counter") < 0 ) {
    Serial.println("Error! File did not exist!");
  }
  else {
    Serial.print("File was opened and counter variable was recovered. Its value
        is: ");
    Serial.println(sleepCounter);
  }

  // Só executa as medições se for o horário apropriado
  if (sleepCounter >= SLEEP_TIME) {
    sleepCounter = 0;
    // Conecta ao WiFi
    connectWifi();
    // Pinos do sensor ultrassônico
    pinMode(trigPin, OUTPUT); // Seta trigPin como Output
    pinMode(echoPin, INPUT); // Seta echoPin como Input

    client.setServer(mqtt_server, mqtt_port);

    // Mede as distâncias
    measureDistance();
    sendMessageToBroker();

  }
  else {
    // se não foi tempo suficiente dormindo, incrementa contador
    sleepCounter++;
  }

  Serial.println("Done, let's sleep...");
  // Dorme por 1h
  ESP.deepSleep(36e8, WAKE_RF_DEFAULT); // 36e8 us = 1 hour

}
```

3.1.1 Gerenciamento de Dados

Para escolher a plataforma de recebimento dos dados primeiro considerou-se usar a ThingSpeak[17], bastante popular em projetos de Internet das Coisas. No entanto, optou-se pela Konker[18], empresa que fornece uma solução eficiente e que tem sido parceira da Prefeitura Universitária em outros projetos do *Smart Campus*.

A plataforma suporta o uso de dois protocolos: MQTT[19] e HTTP. O MQTT (*Message Queuing Telemetry Transport*) é amplamente utilizado para aplicações com Internet das Coisas e funciona sobre TCP/IP (portas 8883 e 1883). Suas principais características são simplicidade, flexibilidade e leveza, que propiciam seu uso em ambientes que requerem pouca banda, como no caso de transmissão de dados medidos por sensores, por exemplo. Isso acontece através de um baixo *overhead*, além de ser agnóstico com relação aos dados transferidos. Os seus pacotes são constituídos por header fixo (contém dados de controle, como identificador de operações), header variável (opcional para algumas operações) e payload (também opcional).

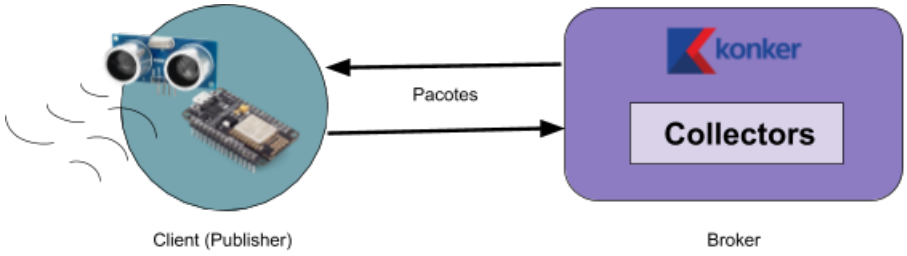

Figura 7: Esquema Publisher - Broker

O protocolo funciona com um *broker*, responsável por ser intermediário entre clientes *subscribers* e/ou *publishers* para determinados tópicos. No caso do projeto, o protótipo é apenas *publisher* (Fig. 7), ou seja, somente envia informações para o broker da Konker (Fig. 8). As informações são enviadas em pacote usando a formatação Json (Fig. 9), incluindo campos para a distância lida pelo sensor e também um identificador do coletor para um tópico (ou canal) chamado "collectors". Antes disso, o cliente se conecta através de uma autenticação fornecida pelo broker.

[17]Site do ThingSpeak https://thingspeak.com/
[18]Site da Konker: https://www.konkerlabs.com/index-en.html
[19]Site Oficial do MQTT: http://mqtt.org/

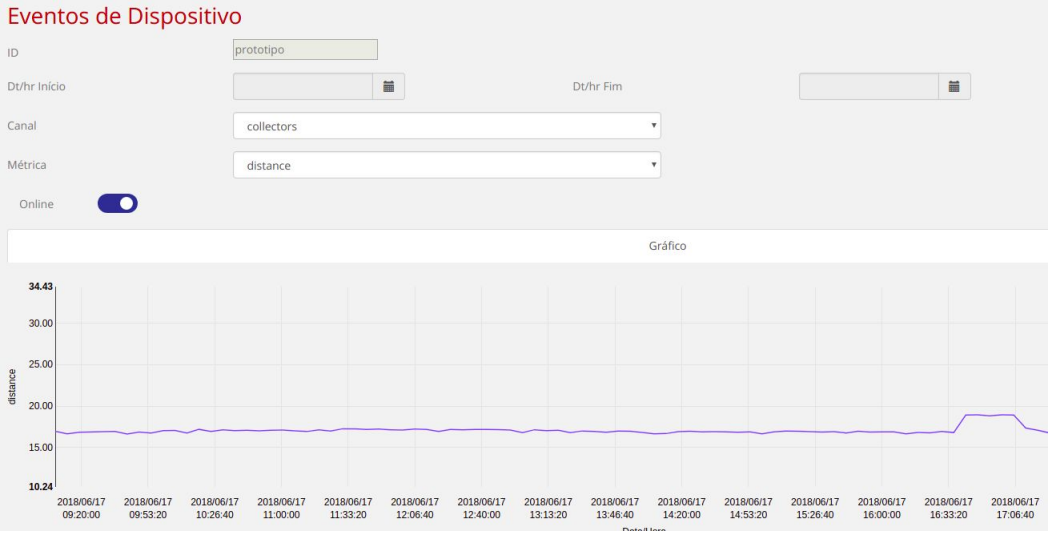

Figura 8: Plataforma da Konker

Data/Hora	Canal	Mensagem
2018/06/17 18:55:32.546 BRT	collectors	{ deviceId : "Iot Battery Collector", distance : 10.53235 }

Figura 9: Mensagem Recebida na Plataforma em formato Json

3.1.2 Criação de um Sistema Web

Com os dados na rede, é necessário manuseá-los para conseguir a melhor rota de coleta. Assim, foi implementado um website com uso de HTML, CSS, Javascript e o framework Django[20][21], baseado em *Python*. Esse foi o *framework* escolhido pelo aluno já ter conhecimento maior sobre *Python*, além permitir uma arquitetura MVC (*Model-View-Controller* sólida e suporte ao uso de banco de dados. Para a parte de mapas, foi usada a API do *Google Maps* (Javascript) e para guardar dados relevantes do sistema foi usado o banco de dados Sqlite.

Buscou-se construir uma interface amigável e de fácil uso. A cada acesso ao site o sistema requisita dados dos coletores da plataforma da Konker via HTTP (Requisição *GET*).

[20]Django Overview: https://www.djangoproject.com/start/overview/
[21]Documentação Django https://docs.djangoproject.com/en/2.0/

Código de Manipulação de Dados provenientes da Nuvem

```
def getAndUpdateData():
    Objeto com dados do sistema
   mSystemData = SystemData.objects.last()
   // Faz requisição GET para o servidor da Konker
   r = requests.get("http://data.demo.konkerlabs.net/sub/usuario/out?offset=" +
       str(mSystemData.lastTimeStamp), auth=('usuario', 'senha'))
   // Recebe Json
   expressions = r.json()

   // Manipula json
   if(len(expressions) > 0):
          for i in range(len(expressions)):
              mCollectorName = expressions[i-1]["data"]["deviceId"]
              // Atualiza valores de distância no Banco de Dados
              try:
                  mCollectorData =
                      CollectorData.objects.get(collectorName=mCollectorName)
                  mCollectorData.distanceRead =
                      expressions[i-1]["data"]["height[cm]"]
                  mCollectorData.save()
              except ObjectDoesNotExist:
                  print("ATENCAO! Existe um coletor nao cadastrado no BD!")
```

A ideia é que os funcionários responsáveis pela coleta das pilhas e baterias possam definir pontos de partida e chegada do caminhão de coleta (Fig. 10), nível em porcentagem de capacidade do coletor que devem ser recolhidos (Fig. 11) e sua rota otimizada (Fig. 12).

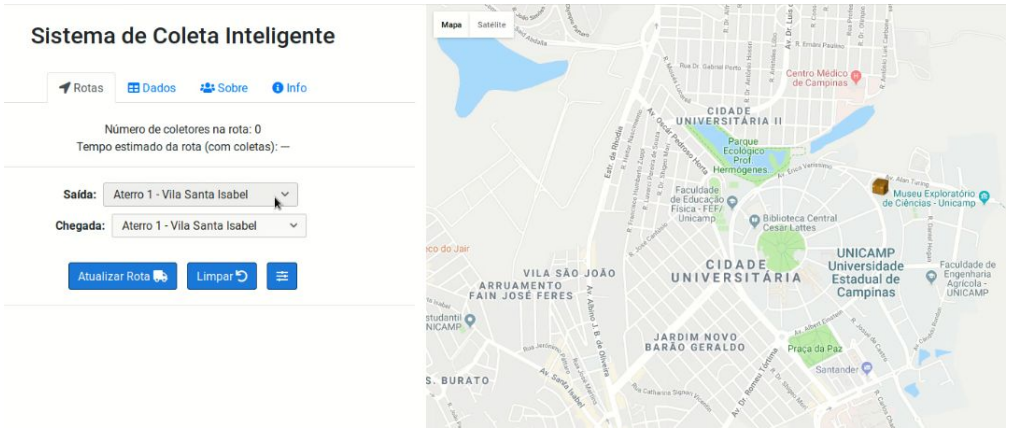

Figura 10: Página Inicial do Sistema de Roteamento

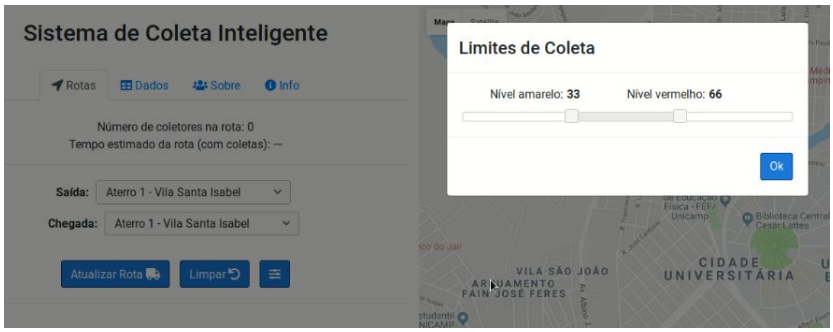

Figura 11: Sistema permite configurar limites da capacidade(%) preenchida para coleta

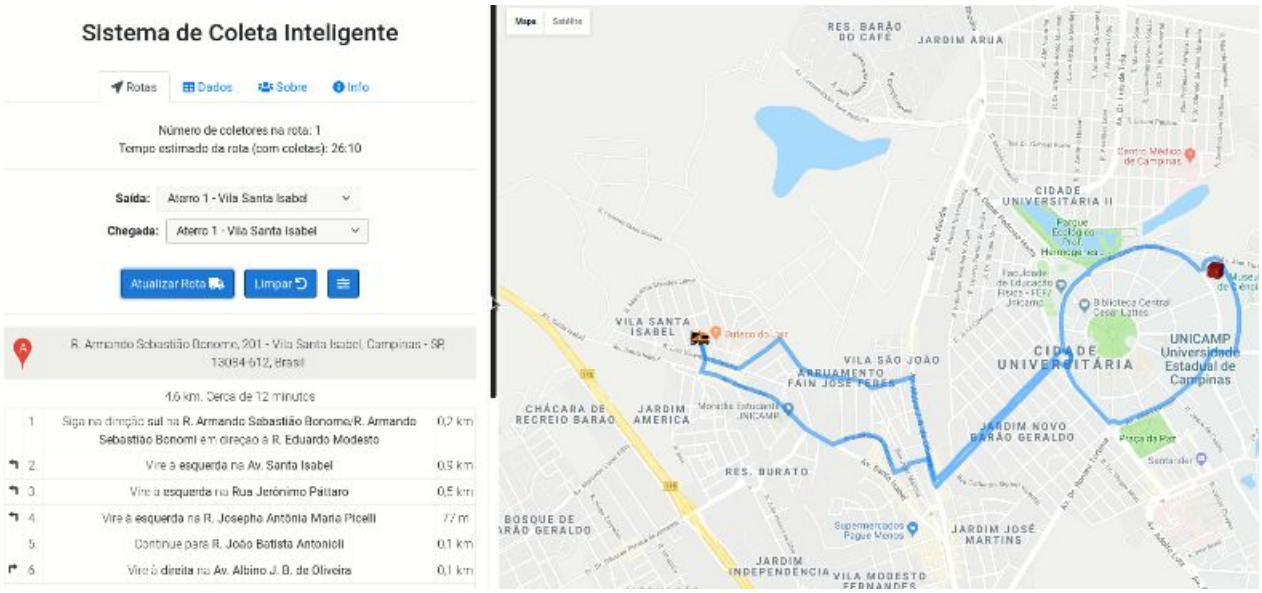

Figura 12: Sistema mostrando rota eficiente

Figura 13: Sistema conta com abas de fácil acesso

O banco de dados deve ser preenchido com informações básicas dos coletores, como coordenadas de localização e suas dimensões (Fig. 14) e só pode ser acessado por pessoas autorizadas.

Figura 14: Exemplo do formulário a ser preenchido sobre um coletor no banco de dados do sistema

Além disso, o usuário também poderá ser ver detalhes sobre os coletores e filtrá-los na aba "Dados" (Fig. 13). As outras abas "Sobre" (Fig. 15) e "Info" (Fig. 16) são utilizadas para dar informações e suporte sobre o projeto.

Figura 15: Aba com informações do projeto

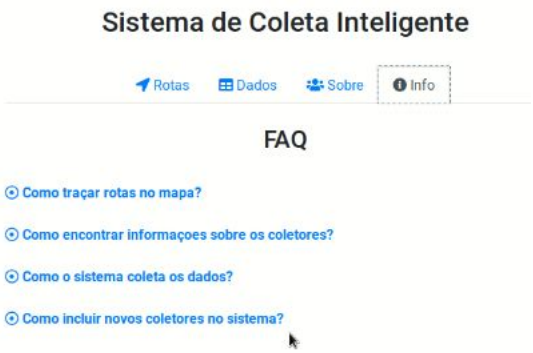

Figura 16: Aba com FAQ para auxiliar os usuários

O desenvolvimento do website baseou-se em uma metodologia ágil. Primeiro foram implementadas as funções mais básicas como uso da API do *Google Maps* para mostrar os coletores e definir as rotas. Depois foi adicionada a funcionalidade de pesquisa, listagem e filtro dos coletores. Em seguida, houve a reestilização do sistema com bootstrap e a criação de abas para separar os conteúdos. Por fim, foram adicionadas as informações do projeto e o FAQ.

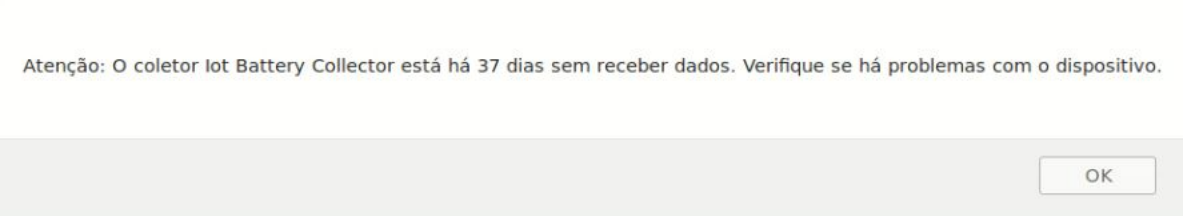

Figura 17: Janela informando problema com coletor não responsivo

Uma última modificação foi resultado de testes na plataforma. Até então, não se sabia quando um coletor parava de receber dados pelo sistema. Assim, foi implementado um aviso caso um determinado dispositivo não esteja enviando dados a um certo tempo, o website notifica o usuário (Fig.17).

3.1.3 Testes e Melhorias

Inicialmente o sensor ultrassônico foi testado fora do coletor. Foram feitos testes com o balde utilizado dentro dos coletores (Fig. 18). O gráfico da Figura 19 mostra os resultados encontrados para uma distância de 38 cm, tomando a média de 100 medições. Pode-se ver que os valores medidos não tem uma variação tão relevante para o projeto. Um detalhe percebido durante os testes é que o sensor é bastante sensível a choques, retornando valores absurdos quando não deixado bem fixado em uma superfície sólida. Como melhoria a isso,

foi realizada a mudança no código para que valores absurdos (muito acima da capacidade do balde) fossem ignorados. O gráfico da Figura 20 é resultado do mesmo teste com as alterações citadas.

Figura 18: Balde utilizando dentro do coletor

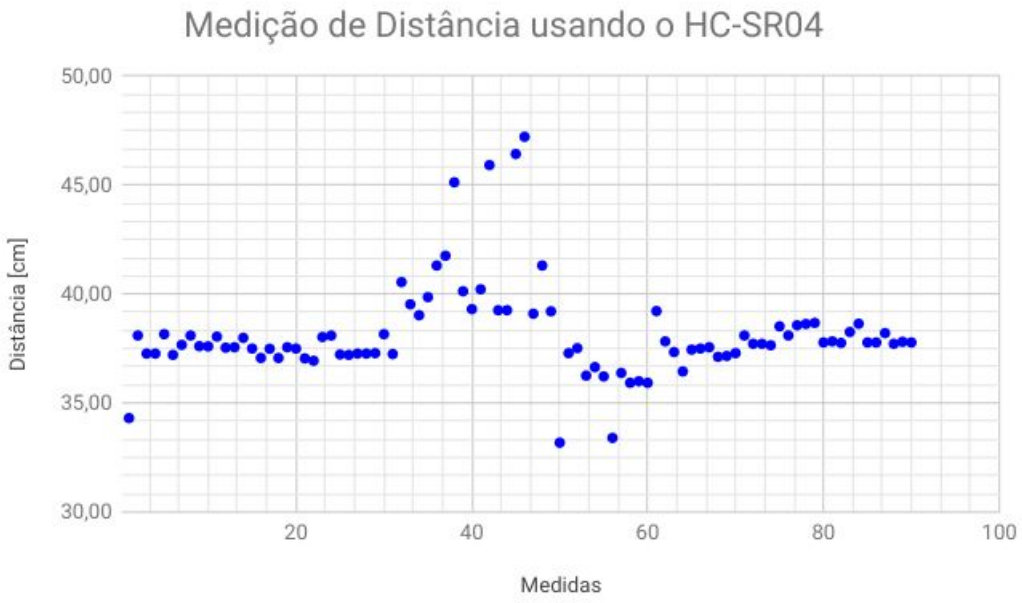

Figura 19: Gráfico da Medição de distância usando o HC-SR04

Uma diferença de 2 a 4 cm não é considerada determinante para a coleta, pois a capacidade (em volume) não sofre alterações significativas.

Dessa forma, instalou-se um protótipo no Instituto de Computação da Unicamp para monitorar as medições do coletor. O primeiro problema foi colocar o sensor em uma posição reta. A tampa da caixa (Fig. 1) não faz um ângulo próximo de 90º com a parede, fazendo com que o sensor não meça distâncias referentes ao fundo do balde e prejudicando os resul-

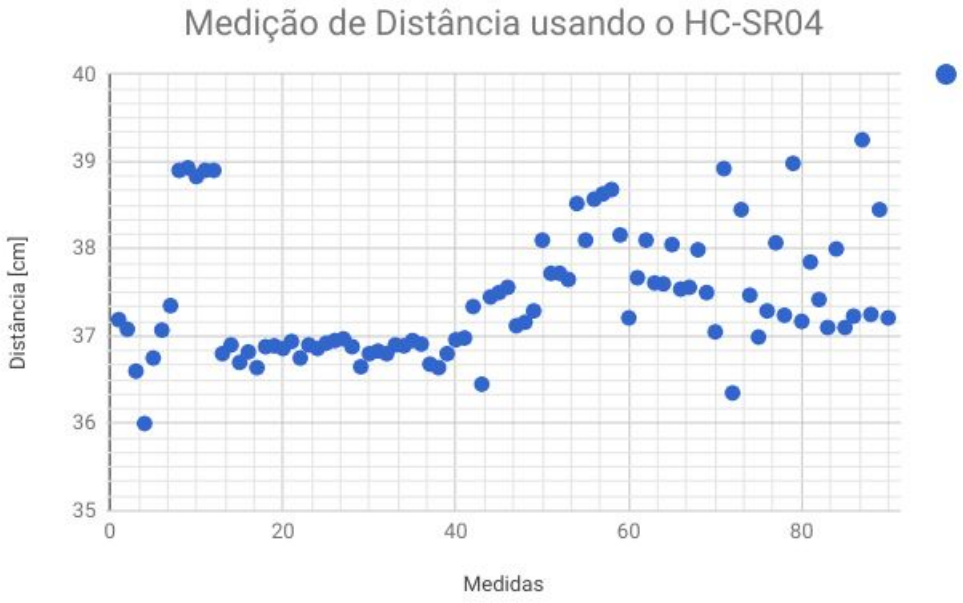

Figura 20: Gráfico da Medição de distância usando o HC-SR04 ignorando valores absurdos

tados. Foi utilizado um calço para conseguir medidas mais confiáveis.

Outra tentativa durante os testes foi enviar para a plataforma da Konker dados já calculados no dispositivo, como volume e capacidade (%) preenchida. Entretanto, o sistema apresentou problemas ao receber os dados com pacotes json maiores. Isso também limitaria o protótipo a somente ser instalado em um modelo de coletor. Decidiu-se então mandar dados apenas da distância lida e no sistema web fazer os cálculos necessários.

Durante algumas semanas dados foram recolhidos do coletor. No entanto, não houve grande variação do nível retornado pelo protótipo. Por um lado positivo, isso mostrou que o dispositivo estava funcionando corretamente e de forma regular (as distâncias estavam condizentes) (gráfico 3 da Fig. 21). Por outro, o fato de poucas pessoas depositarem pilhas e baterias no coletor limitou os resultados e a confiabilidade do projeto a longo prazo. Além disso, o *power bank* utilizado para alimentar o protótipo teve de ser carregado a cada 10 dias.

Um novo protótipo foi instalado na entrada da Prefeitura Universitária e tem recebido dados corretamente[22] (Fig. 22 e Fig. 23).

[22]Unicamp Inovação: *Smart Campus* passa a contar com plataforma para Internet das Coisas: http://www.unicamp.br/unicamp/noticias/2018/07/03/smart-campus-passa-contar-com-plataforma-para-internet-das-coisas

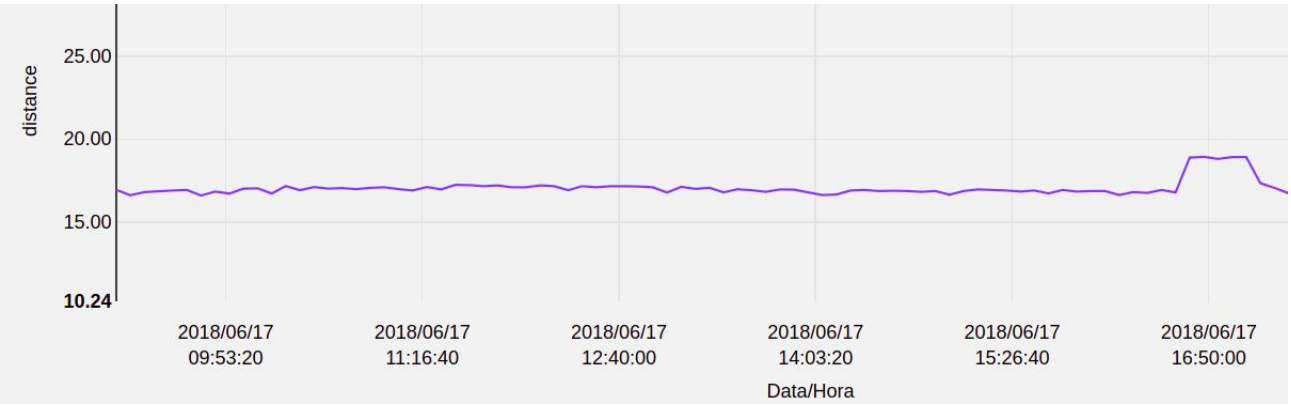

Figura 21: Parte do resultado dos testes do protótipo

Figura 22: Protótipo instalado na Prefeitura Universitária

4 Conclusão

O protótipo se mostrou eficiente, sem variações que comprometam o serviço de coleta. Ademais, os pacotes de dados que transitam entre dispositivo e nuvem são leves e fluem sem problemas. O sistema de roteamento consegue receber as informações da Konker e mostrar as rotas de maneira fácil e intuitiva para o usuário. Entretanto, é preciso padronizar a instalação do protótipo no coletor evitando problemas de leitura incorretas. Além disso, também são necessários mais testes de longa duração, para aumentar a confiabilidade do dispositivo. Entre os desafios encontrados no projeto, o maior é a eficiência energética. Mesmo com o modo *deep sleep*, a corrente média consumida é de cerca de 18mA. O responsável por isso é o regulador de tensão de 5V para 3.3V. Várias estratégias foram pensadas, como uso de transistores para controlar a corrente para o sensor ultrassônico ou uso de outros micro-

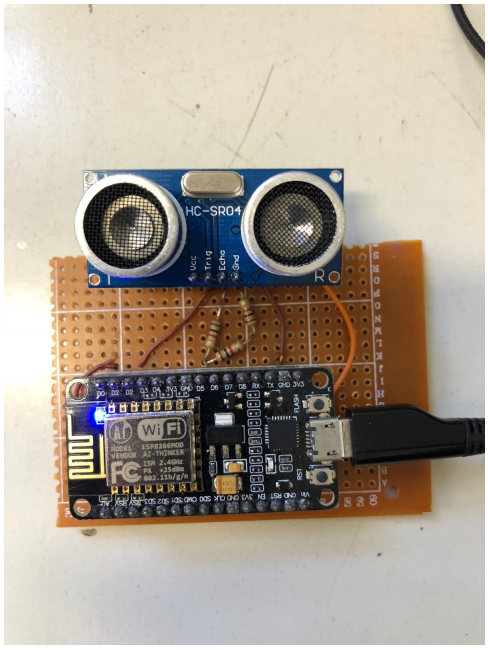

Figura 23: Coletor com protótipo instalado na Prefeitura Universitária da Unicamp

controladores como o Arduino. No entanto, por não ter no mercado um sensor ultrassônico de 3.3V de baixo custo ou um módulo WiFi robusto como o ESP8266 que trabalhe com 5V, o regulador é sempre necessário e aumenta significamente o uso de energia. Além disso, o envio de dados via WiFi não é o ideal, pois como foi mostrado, os pacotes são pequenos, não dependendo de muita banda. Conexões WiFi ainda tem o limitante de alcance, obrigando o bom sinal de internet em diversos pontos da universidade. Uma alternativa a um médio prazo é a tecnologia *LoRaWan*[23], que permite comunicação a distâncias maiores (cerca de 3-4 Km) com baixo nível de energia. Ainda não existem, entretanto, implementações comerciais da rede no Brasil.

Enfim, tendo as limitações e desafios em mente, espera-se que o protótipo seja usado como base tanto para implementação de coletores inteligentes por todo o campus da Unicamp, bem como para um sistema que beneficie o trabalho dos funcionários da Universidade.

Referências

[1] K. Ashton, *That "Internet of Things"thing*, RFiD Journal, 2009.

[2] D. Miorandi, *Ad Hoc Networks 10*, p. 1497–1516, 2012.

[3] A. Zanella, N. Bui, A. Castellani, L. Vangelista e M. Zorzi, *Internet of Things for Smart Cities*, IEEE Internet of Things Journal, v. 1, n. 1, 2014.

[23] LoRaWan https://lora-alliance.org/about-lorawan

[4] A. Cenedese, A. Zanella, L. Vangelista and M. Zorzi, *Padova Smart City: An urban Internet of Things experimentation*, Mobile and Multimedia Networks 2014, p. 1-6, 2014.