Web Smart Camera: Plataforma Web de Ensino com Câmera Integrada a um Sistema Embarcado

Cláudia A. Rodrigues[1], **Vanessa C. R. Vasconcelos**[1], **Ricardo Ferreira**[1]

[1]Departamento de Informática – Universidade Federal Viçosa (UFV)
CEP: 36.570-900 – Viçosa – Minas Gerais – Brazil

claudia_rodrigues11@hotmail.com, vanessa.cr.vasconcelos@gmail.com

ricardo@ufv.br

Abstract. *Internet of Things brings to reality the remote control of daily life activities. Examples are that it is possible to control a house lightning through a smartphone, and it is possible to control heart rate using a smartwatch. Thus, this project aims to bring technology and classrooms closer. Using an embedded system and a camera, the professor will be able to project and/or stream lectures without losses on the interaction with the students. Becoming possible to add a new dynamic to classes besides blackboard and Power Point presentations. The project also aims to keep the recorded lectures to posterior access. Furthermore, the platform is robust and controls user access.*

Resumo. *A Internet das Coisas transforma em realidade o controle remoto de diversas tarefas do cotidiano. Exemplos são a possibilidade de controlar a iluminação de uma casa através de um smartphone, e de controlar a frequência cardíaca através de um smartwatch. Assim, este projeto propõe trazer esta tecnologia para a sala de aula. Utilizando um sistema embarcado integrado à uma câmera, o professor será capaz de projetar e/ou transmitir suas aulas sem perder a interação com os alunos, adicionando assim uma nova dinâmica às aulas além de quadro e slides. O projeto também visa o armazenamento do vídeo após a realização das aulas para acesso posterior. Adicionalmente, a plataforma que engloba a transmissão é robusta e controla o acesso dos usuários.*

1. Introdução

Com o fácil acesso a sistemas embarcados e periféricos, fica cada vez mais palpável desenvolver interfaces que facilitam tarefas cotidianas. Este trabalho em específico, visa projetar e desenvolver uma plataforma web composta de um sistema embarcado e uma câmera, que juntos estarão aptos a projetar e/ou transmitir vídeo e áudio em tempo real. Esta transmissão poderá ser assistida através de um endereço na rede local e o usuário deve ter permissão de acesso para assistir a transmissão em questão.

A plataforma a ser desenvolvida durante este projeto tem como público alvo professores, que uma vez em posse do sistema completo poderão tornar suas aulas mais dinâmicas e motivadoras.

O website desenvolvido em conjunto com a plataforma câmera-embarcado disponibiliza, além da possibilidade de assistir a transmissão ao vivo, a oportunidade de acessar este vídeo posteriormente, tudo de maneira amigável e intuitiva. Existe diferentes

permissões de acesso para os usuários da plataforma de acordo com os seus papéis. As funcionalidades da plataforma são demonstradas na seção 4.1.

Em se tratando das especificações técnicas, este trabalho desenvolveu testes usando o *Raspberry Pi 3 Model B* e uma *Webcam HD 720P C270 Logitech* que é compatível com o Sistema Operacional Linux, ambos são melhor detalhados durante a seção 3 deste artigo.

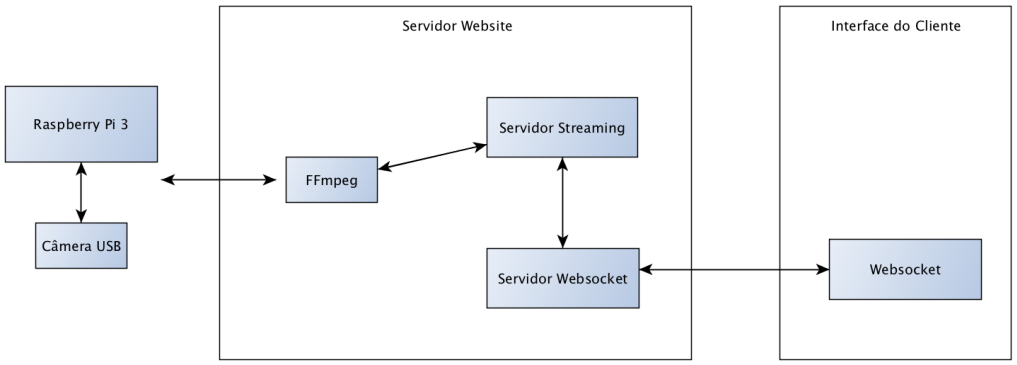

Figura 1. Diagrama de blocos da transmissão

O diagrama de blocos da Figura 1 ilustra como a transmissão de áudio e vídeo é feita. O servidor streaming recebe o vídeo capturado no *Raspberry Pi* via *FFmpeg* e o repassa para a interface do cliente via *webSockets*. O projeto completo pode ser encontrado no GitHub em [Rodrigues and Vasconcelos 2017], em conjunto com um tutorial.

2. Referencial Teórico

O que existe de similar a plataforma proposta neste trabalho são as *Document Cameras*. Este dispositivo funciona com uma câmera de qualidade regular (em torno de 1.3 Mega Pixels), conectada a um suporte flexível e a uma base de controle. A base permite executar operações como zoom in e zoom out, autofoco, brilho e controle de iluminação. Existem também algumas empresas como a *Smart Education* que oferecem serviços de *Smart Document Camera*, com a possibilidade de realizar transmissões similarmente a plataforma proposta, porém, estes serviços são extremamente caros.

Há registros de projetos utilizando o módulo de computação *Intel Edison* para realizar transmissão de vídeo. O *Intel Edison* é um sistema embarcado oferecido pela Intel que contém um processador com dois núcleos Intel Quark x86 que operando a 400MHz e 1Gb de memória, tem conexão Wi-fi e bluetooth. O sistema nativo é chamado de Yocto Linux.

```
from flask import Flask, Response
import cv2
import numpy as np

class Camera(object):
    def __init__(self):
        self.cap = cv2.VideoCapture(0)
```

```
        # Reset camera capture size for faster processing
        self.cap.set(3,480)
        self.cap.set(4,360)

    def get_frame(self):
        ret, frame = self.cap.read()
        # Apply laplacian edge detection to image
        laplacian = cv2.Laplacian(frame,cv2.CV_64F)
        # Write out original and edge detected images at once
        cv2.imwrite('blah.jpg',np.hstack((frame,laplacian)))
        return open('blah.jpg', 'rb').read()


app = Flask(__name__)

def gen(camera):
    while True:
        frame = camera.get_frame()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')

@app.route('/')
def video_feed():
    return Response(gen(Camera()),
                    mimetype='multipart/x-mixed-replace; boundary=frame
    ')

if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True)
```

Listing 1. Código em Python para Transmissão de Vídeo

Durante o desenvolvimento deste projeto, realizamos testes com o *Intel Edison*. A listagem 1 mostra um código escrito em Python, utilizando o framework Flask e o OpenCV, o qual executando neste realiza a transmissão de vídeo [Moyerman 2015]. Entretanto, o OpenCV é pesado e o atraso na imagem foi perceptível durante nossos testes, apesar do código ser simples.

Em [Kim 2016], também é possível encontrar um projeto utilizando *Node.js* e *FFmpeg* que realiza a transmissão de vídeo no *Intel Edison*. A qualidade do vídeo é notável e o atraso não é perceptível, porém, este projeto não suporta a transmissão de áudio. Nossos testes com o *Intel Edison* foram encerrados depois da notícia que a Intel está descontinuando a linha [List 2017].

Tratando apenas da transmissão de vídeo, existem algumas APIs e *plugins* que se propõem a fazê-lo. Um exemplo é o *WebRTC*, API desenvolvida pela *World Wide Web Consortium (W3C)* com o intuito de permitir aos navegadores executar aplicações de chamada telefônica, vídeo chat e compartilhamento par-a-par (P2P). A API está bastante difundida na comunidade de desenvolvedores, mas ela não funciona em todos navegadores.

Outro exemplo é o *ffserver* que é um programa servidor para transmitir áudio e vídeo via protocolo HTTP. Ele faz parte da distruibuição do *FFmpeg* que é utilizada neste projeto. Porém, pela sua difícil manuntenção, ele está sendo removido no próximo *release*

do próprio *FFmpeg*, o que inviabiliza a sua utilização.

Ademais, existe uma grande comunidade desenvolvendo projetos para o *Raspberry Pi*, logo não é difícil encontrar exemplos de código para *streaming* on-line. Realizar transmissão de áudio e vídeo em conjunto e sem atraso por outro lado, é mais complicado. O decoder encontrado em [Szablewski 2017b] serviu como base deste trabalho, por viabilizar a transmissão de áudio e vídeo com o *Raspberry Pi*, mas sem oferecer as demais funcionalidades que este projeto disponibiliza.

3. Material e Métodos

Os principais materiais utilizados durante o projeto são o *Raspberry Pi 3 Model B* e a *Webcam HD 720P C270 Logitech*. Esta seção não apenas os detalha mas também traz uma listagem dos principais métodos e *frameworks* utilizados durante a construção desta plataforma.

3.1. Raspberry Pi 3 Model B

Raspberry é um sistema embarcado criado na Inglaterra pela Fundação *Raspberry Pi* com o objetivo principal de possibilitar o estudo e a prática de informática e programação para crianças, através de um pequeno computador de baixo custo, com o mínimo de periféricos conectados para funcionar e fonte de alimentação com baixa voltagem de 5V. O primeiro modelo apresentado para mercado foi *"Raspberry Pi 1 Model A"* em 2012 por US$25, com uma memória interna de 256 Mb, CPU de 700 Mhz, saída de vídeo HDMI, entrada para cartão SD, e diversas GPIOs para periféricos externos. O *Raspberry* roda principalmente versões de Linux produzidas para ele mesmo.

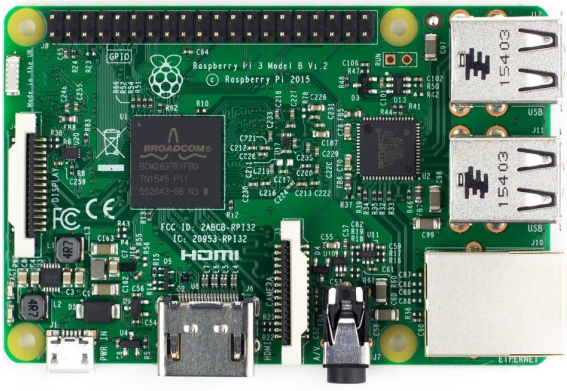

Figura 2. Raspberry Pi 3 Model B

O modelo utilizado neste trabalho foi a versão lançada em 2016, *"Raspberry Pi 3 Model B"*, ver Figura 2, com processador BCM2837 de 64 bits, rodando com 4 núcleos a 1,2 Ghz, 1 Gb de memória RAM, CPU de 900Mhz, Wi-Fi e Bluetooth nativas, 4 portas USB e uma entrada para Micro SD. Houve um grande ganho de desempenho em relação as versões anteriores e o preço de mercado até a data de desenvolvimento deste trabalho era em média R$35. Esta versão foi a escolhida para este projeto, justamente por possuir Wi-Fi nativa.

3.2. Webcam HD 720P C270 Logitech

A *Webcam HD 720P C270 da Logitech*, Figura 3, é uma câmera UVC (*USB video device class* ou apenas *USB video class*), o que significa que não é necessário nenhuma instalação adicional para que esta funcione no *Raspbian Jessie*, sistema operacional do *Raspberry Pi* durante este projeto.

Figura 3. Webcam HD 720P C270 Logitech

Além disso a webcam tem microfone integrado e embutido, logo pode-se capturar áudio e vídeo sem necessitar de um outro periférico. E por último a qualidade da imagem pode chegar a 1280 x 720 pixels, o que é essencial para uma boa transmissão.

3.3. FFmpeg, JSMpeg e WebSockets

De acordo com o site oficial, o *FFmpeg* é um *framework* multimídia capaz de decodificar, codificar, transcodificar, mux, demux, transmitir, filtrar e reproduzir praticamente qualquer coisa. Ele suporta desde os fornatos antigos mais obscuros até os mais atuais e é altamente portátil: funciona no *Linux, Mac OS X, Microsoft Windows, BSDs, Solaris*, entre outros.

O JSMpeg é um decoder de vídeo MPEG1 e áudio MP2 em *JavaScript*. Escrito por Dominic Szablewski, é um vídeo player que consiste em um demuxer MPEG-TS, decodificadores de vídeo MPEG1 e áudio MP2, renders WebGL e Canvas2D e saída de som WebAudio. O JSMpeg pode carregar arquivos estáticos via Ajax e permite transmissão com latência baixa (aprox. 50ms) via *webSockets*.

A especificação de *webSockets* define uma API que permite páginas web se comunicarem bidirecionalmente com um host remoto. Ela apresenta a interface do *webSocket* e define um canal de comunicação full-duplex que opera através de um único soquete na web. Outra vantagem dos *webSockets* é que, com exceção do navegador Android, todos os navegadores mais atuais suportam a sua última especificação.

3.4. Node.js e frameworks

O *Node.js* é uma plataforma multiprotocolo, ou seja, com ele é possível trabalhar de maneira simplificada com o HTTP (*Hypertext Transfer Protocol*), DNS(*Domain Name System*), TCP(*Transmission Control Protocol)*, *WebSockets*, entre outros, e é muito utilizado para aplicações escaláveis no lado do servidor. O *Node.js* foi a plataforma escolhida para implementar este projeto.

Os *frameworks* utilizados durante o desenvolvimento deste projeto são:

- **Express Framework**: Contém um conjunto robusto de recursos para desenvolver aplicações web, como um sistema de views intuitivo (MVC: *Model-View-Controller*), um sistema robusto de roteamento e possui integração com ferramentas de template de views.
- **Express-session**: O *middleware express-session* armazena os dados da sessão no servidor, salvando apenas o id da sessão no cookie, e por padrão, usando armazenamento em memória.
- **Passport**: *Middleware* que implementa a autenticação em um aplicativo. Tem uma arquitetura modular que quebra o mecanismo de autenticação em estratégias que são empacotados como módulos individuais. As aplicações podem escolher qual estratégia empregar sem criar dependências desnecessárias. Também possibilita a autenticação com redes sociais.
- **Bcrypt**: Gera hashes criptograficamente fortes de senhas de usuários e performa a comparação com as senhas enviadas. O Bcrypt suporta métodos síncronos e assíncronos.
- **Connect-flash**: O flash é uma área especial da sessão usada para armazenar mensagens. As mensagens são gravadas no flash e desmarcadas depois de serem exibidas. Ele normalmente é usado em combinação com os redirecionamentos, garantindo que a mensagem está disponível para a próxima página a ser renderizada.
- **EJS - JavaScript Embarcado**: O EJS é uma engine de visualização, com ela é possível transportar dados do back-end para o front-end.
- **Body-parser**: Facilita a troca de dados entre código HTML e *JavaScript*.
- **Express file-upload**: Lida com o carregamento e com o tratamento de arquivos.
- **Socket.io**: É uma biblioteca para aplicações web que permite comunicação em tempo real e bidirecional entre clientes e servidores. Tem duas partes: uma biblioteca do lado do cliente que é executada no navegador e uma biblioteca do lado do servidor para *Node.js*.

A API HTTP nativa foi preterida no projeto para a utilização do framework Express pois ela não é apropriada para o desenvolvimento de aplicações complexas, visto que todo o gerenciamento de rotas e outros recursos desse módulo são tratados de maneira bem básica.

Com relação ao uso do *framework* Session, além deste já possuir suporte para o Express, também tem como objetivo evitar o uso de nomes de cookies genéricos, que podem deixar o aplicativo aberto à ataques de invasores em potencial.

Quanto ao *middleware* para autenticação e segurança de roteamento entre páginas, foi utilizada a função isLoggedIn juntamente com o framework Passport, a qual confere se o usuário está logado para acessar uma determinada rota.

A escolha da engine EJS foi devida a possibilidade de reúso do código no lado do cliente.

3.5. MySQL

MySQL foi o banco de dados escolhido para este projeto. Esta escolha se deu por este ser gratuito, de fácil utilização em conjunto com *JavaScript*, confiável e por possuir um bom desempenho.

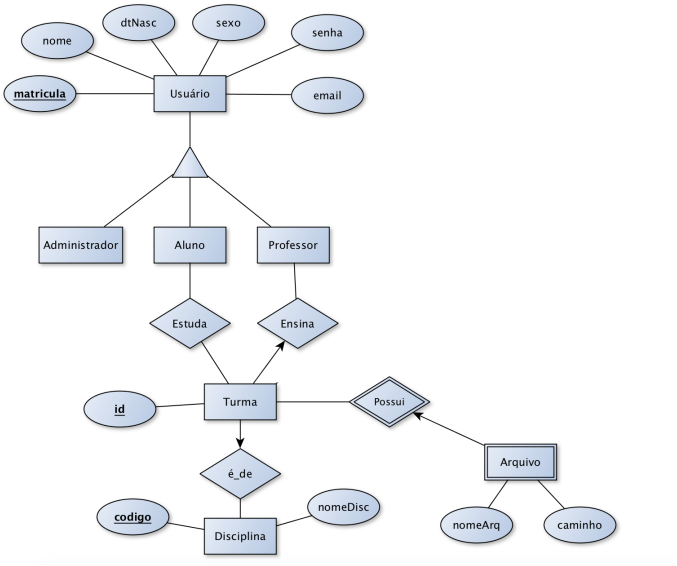

Figura 4. Modelagem Entidade-Relacionamento

A Figura 4 mostra o diagrama entidade-relacionamento que modela o banco de dados utilizado na plataforma proposta. No diagrama é possível ver que as turmas são compostas de um professor e diversos alunos e que apenas o professor e os alunos matriculados em determinada turma tem acesso aos seus arquivos. Arquivos estes que são uma entidade fraca, pois no sistema proposto não há necessidade de um arquivo existir fora do escopo da sua relação com a turma. Primariamente, os dados de usuário, arquivo, turma e disciplina são os dados persistidos no banco.

4. Desenvolvimento e Resultados

4.1. Permissões de usuário

Para melhor organizar as funcionalidades da plataforma, existem três tipos de usuários. Eles são administrador, professor e aluno, que serão melhor explicados a seguir.

4.1.1. Administrador

O administrador é o único que pode cadastrar novos usuários na plataforma. Essa foi uma decisão de projeto, que aumenta a segurança da plataforma, uma vez que não permite qualquer pessoa cadastrar-se, matricular-se e ter acesso aos materiais disponíveis.

Por default, a plataforma vem com um administrador e senha que devem ser modificados no primeiro acesso para aumentar a segurança e viabilizar o explicado acima.

O administrador também pode cadastrar e excluir disciplinas, criar turmas associando um professor a uma disciplina, além de poder matricular estudantes.

4.1.2. Aluno

O aluno tem como página inicial uma lista com todas as turmas nas quais ele está matriculado e os respectivos links para as páginas destas. Existe também uma barra de notificações com um alerta para os últimos 5 arquivos carregados pelo professor.

Uma vez na página da turma estão disponíveis os links para assistir a transmissão ao vivo, todos os arquivos disponibilizados pelo professor durante o semestre e as gravações das aulas anteriores, se o professor optou por disponibilizá-la. Na página da transmissão ao vivo existe também o chat, onde é possível comunicar-se com o professor e com os colegas matriculados na mesma turma.

4.1.3. Professor

A interface do professor não difere muito da interface do aluno. Sua página inicial é uma lista com as turmas que ele leciona no semestre, e existe a notificação dos últimos 5 arquivos enviados por ele, que serve também como confirmação do envio.

A maior diferença entre as duas interfaces é o professor tem as opções adicionais de iniciar o vídeo na página da turma e na página da transmissão o professor também tem a opção de encerrar o *streaming* e de persistir ou não o vídeo para acesso posterior.

4.2. Transmissão de Áudio/Vídeo e Chat

A transmissão de áudio e vídeo foi feita utilizando o decoder de vídeo MPEG1 e áudio MP2 encontrado em [Szablewski 2017b]. Com este decoder é necessário apenas capturar o input da webcam com o *FFmpeg* e transmiti-lo via *webSockets*. O script *FFmpeg* utilizado pode ser visto na listagem 2. Áudio e vídeo são capturados no mesmo script e a flag muxdelay trata da sincronização entre os mesmos, para assim transmiti-los em conjunto para http://localhost:8081. Existe também a possibilidade de capturar áudio e vídeo separadamente em dois processos *FFmpeg* diferentes, dependendo da situação, esta escolha pode ser mais apropriada. Entretanto como explicado anteriormente, neste projeto ambos são capturados no mesmo script.

```
ffmpeg \
  -f v4l2 \
    -framerate 25 -video_size 640x480 -i /dev/video0 \
  -f alsa \
    -ar 44100 -ac 1 -i hw:1,0 \
  -f mpegts \
    -codec:v mpeg1video -s 640x480 -b:v 1000k -bf 0 \
    -codec:a mp2 -b:a 128k \
    -muxdelay 0.001 \
   http://localhost:8081
```

Listing 2. FFmpeg para capturar áudio e vídeo em conjunto

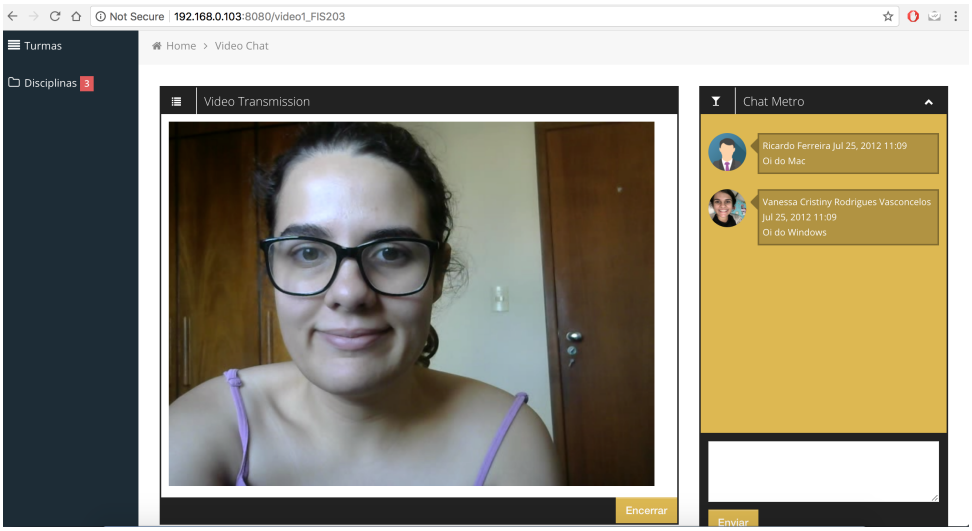

Figura 5. Transmissão de vídeo e chat

Somando-se a isto, no lado do cliente é necessário apenas criar o *JSMpeg Player* na página HTML na qual se deseja exibir o vídeo indicando o caminho do *websocket* responsável pela transmissão, depois de instanciá-lo. Este player também reproduz vídeos do sistema de arquivos, se necessário, porém esta funcionalidade não está sendo utilizada neste projeto.

A Figura 5 mostra a página de transmissão da plataforma, acessada via navegador *Google Chrome* com a permissão de acesso do professor. Na imagem é possível notar além do vídeo, a opção de encerrá-lo, o chat à direita com a identificação dos usuários, e a barra lateral com as turmas nas quais o professor leciona.

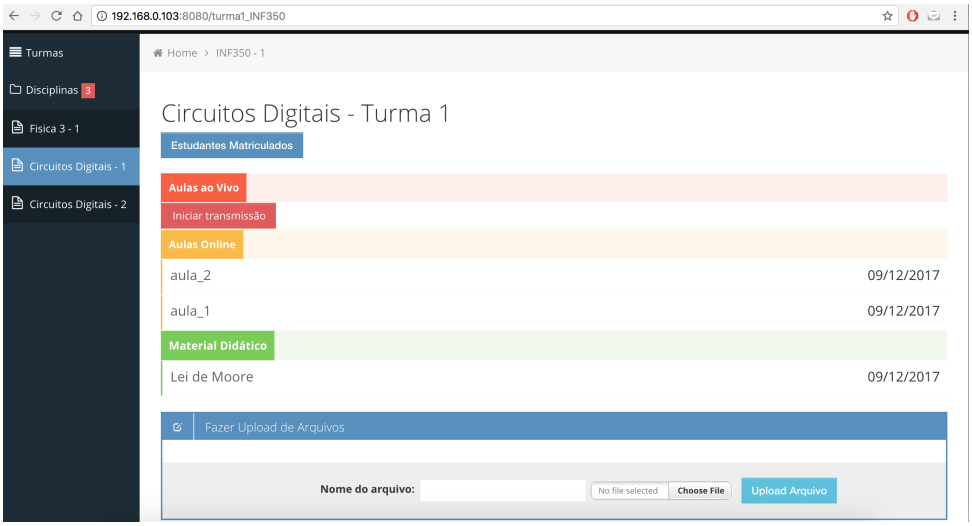

Figura 6. Página da disciplina

Sobre o chat, destaca-se que a imagem do usuário exibida em conjunto com as suas mensagens, pode ser trocada na página de perfil pelo próprio usuário. No momento da imagem, a plataforma estava sendo acessada por dois usuários distintos em duas máquinas diferentes, um deles possuindo permissão de professor e o outro de aluno matriculado na

disciplina Física 3 (FIS203).

Como dito anteriormente, o professor tem a opção de disponibilizar as aulas para serem assistidas posteriormente, assim o link para *download* das mesmas pode ser acessado via a página da turma vista na Figura 6. Nesta figura, também vê-se a interface do professor, logo há a opção de adicionar novos arquivos e iniciar uma transmissão. Também é possível notar a opção que mostra todos os estudantes matriculados na turma, e a lista com os arquivos de vídeo e de texto, ordenados por data.

5. Conclusão e Trabalhos Futuros

De posse dos resultados, conclui-se que o projeto proposto é uma plataforma viável. O conjunto *Raspberry Pi* e câmera USB é bastante acessível, a imagem durante a transmissão é nítida e não existe atraso perceptível entre áudio e vídeo. Ademais, toda a plataforma construída ao redor da transmissão é amigável ao usuário e acrescenta segurança e maiores funcionalidades ao sistema.

Futuramente é possível acrescentar novas funções ao sistema, como por exemplo reconhecimento facial para controle de presença em sala de aula, e um tratamento mais sofisticado da imagem. Existe a possibilidade também de trabalhar com a Pi câmera, ao invés de uma câmera USB, porque esta proporciona uma melhor latência na captura do vídeo.

Referências

FFmpeg (2017). Ffmpeg. https://ffmpeg.org/. Accessed: 2017-12-03.

Guru99 (2017). Node.js express framework tutorial - learn in 10 minutes. https://www.guru99.com/node-js-express.html. Accessed: 2017-12-03.

Kim, E. J. (2016). Edi-cam. https://github.com/drejkim/edi-cam. Accessed: 2017-12-03.

List, J. (2017). Intel discontinues joule, galileo, and edison product lines. https://hackaday.com/2017/06/19/intel-discontinues-joule-galileo-and-edison-product-lines/. Accessed: 2017-06-20.

Moyerman, S. (2015). Edison web video processed. https://github.com/smoyerman/EdisonWebVideoProcessed. Accessed: 2017-05-15.

NodeBR (2016). Primeiros passos com passport e express em node.js. http://nodebr.com/primeiros-passos-com-passport-e-express-em-node-js/. Accessed: 2017-12-03.

Node.js, F. (2017). Melhores práticas em produção: Segurança. http://expressjs.com/pt-br/advanced/best-practice-security.html. Accessed: 2017-12-03.

Planet, C. (2015). Getting started with node.js and bcrypt. https://codeplanet.io/getting-started-with-node-js-and-bcrypt/. Accessed: 2017-12-03.

Rodrigues, C. and Vasconcelos, V. (2017). Web smart camera. https://github.com/CacauAR/WebSmartCamera. Accessed: 2017-12-03.

Sevilleja, C. (2014). Use ejs to template your node application. https://scotch.io/tutorials/use-ejs-to-template-your-node-application. Accessed: 2017-12-03.

Szablewski, D. (2017a). Jsmpeg - decode it like it's 1999. http://jsmpeg.com/. Accessed: 2017-12-03.

Szablewski, D. (2017b). Jsmpeg – mpeg1 video & mp2 audio decoder in javascript. https://github.com/phoboslab/jsmpeg. Accessed: 2017-10-29.

Websockets (2017). Websocket.org. https://www.websocket.org/index.html. Accessed: 2017-12-03.

Zerner, A. (2015). How bodyparser() works. https://medium.com/@adamzerner/how-bodyparser-works-247897a93b90. Accessed: 2017-12-03.