UNIVERSIDADE ESTADUAL DE PONTA GROSSA
SETOR DE CIÊNCIAS AGRÁRIAS E DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

João Gabriel Corrêa Krüger

SMILE
SISTEMA DE CONTROLE DE ENTRADA E SAÍDA

Ponta Grossa
2018

UNIVERSIDADE ESTADUAL DE PONTA GROSSA
SETOR DE CIÊNCIAS AGRÁRIAS E DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

João Gabriel Corrêa Krüger

SMILE
SISTEMA DE CONTROLE DE ENTRADA E SAÍDA

Anteprojeto apresentado para fins de avaliação da disciplina de Projeto, sob orientação dos professores Ariangelo Hauer Dias, Idomar Augusto Cerutti, Rosane Falate e Zito Palhano da Fonseca

Ponta Grossa
2018

SUMÁRIO

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

1. DIAGNÓSTICO ATUAL

Atualmente vivemos em uma época em que a automação, a computação e a eletrônica são aliadas à resolução de diferentes tipos de problemas. Existe uma enorme diversidade de situações que requerem trabalhos manuais e precisão que podem ser automatizadas para melhora na qualidade e velocidade da tarefa.

O Núcleo de Tecnologia e Educação Aberta e a Distância, ou NUTEAD, da Universidade Estadual de Ponta Grossa foi fundado no ano de 2002 e tem como finalidade possibilitar o acesso à educação e incentivar a comunidade acadêmica a criar projetos semipresenciais ou a distância.

O NUTEAD tem como encargos coordenar as atividades relativas à educação a distância, seja capacitando profissionais ou emitindo pareceres técnicos. O núcleo conta com equipes pedagógicas, técnicas e administrativas que planejam, desenvolvem, acompanham e avaliam as iniciativas de Ensino a Distância, ou EAD, executadas pela Universidade Estadual de Ponta Grossa.

Os funcionários do NUTEAD possuem um expediente de 08:00 até 12:00 durante o período matutino. Durante o período vespertino o NUTEAD possui um expediente de 13:00 até 17:00 horas. Além disso, o núcleo não possui disso atividades noturnas, até o presente momento. O NUTEAD prevê, entretanto, que com novas contratações o período noturno seja explorado para o cumprimento da carga horária dos funcionários novos.

Um funcionário pleno possui uma carga horária de 40 horas semanais enquanto um estagiário possui uma carga horária de 20 horas semanais e estas horas podem ser cumpridas dentro do expediente do núcleo. Devido à essa ampla faixa de possíveis horários de trabalho, se faz necessário um controle mais estrito das horas trabalhadas pelos funcionários todo mês, sabendo se foram cumpridas ou não.

As horas trabalhadas pelos funcionários, atualmente, são feitas por meio de ponto escrito, com auxílio de um sistema onde deve ser feita a entrada manual para gerar a folha de ponto do mês. Isso acarreta em um controle limitado dos horários, além de possibilitar evasões durante o expediente do funcionário, visto que a equipe é grande o suficiente para inviabilizar a fiscalização manual dos horários dos funcionários. Considerando esse cenário, pode-se perceber uma situação onde existe

a possibilidade de registrar o tempo de permanência no NUTEAD, ganhando em eficiência e precisão no ponto e maior produtividade no trabalho.

Soluções atuais para o problema já presentes no mercado incluem limitações como número máximo de funcionários e alto custo de mensalidade e manutenção, além de serem soluções que exigem contato frequente com a empresa para manutenção e adições de funcionalidades, visto que são soluções de código fechado.

2. OBJETIVO DO PROJETO

Produzir um sistema de controle de entrada e saída por meio de uso de componentes eletrônicos de identificação por radiofrequência, ou RFID, e sensores de presença. Além dos componentes da identificação eletrônica, o sistema deverá fornecer um registro de horários em que o usuário com cartão identificador passar pela porta e fotografar entradas e saídas quando não estiver sendo utilizado o cartão identificador na passagem, que salvará os dados em uma mídia de armazenamento.

Esse sistema possuirá também conexão com um banco de dados que terá dados sobre os identificadores como usuário que possui o identificador, horários limites em que deveria passar o identificador no sistema e vezes que não passou o identificador no horário especificado.

3. ESCOPO

O sistema desenvolvido consiste em um método de controle de entrada e saída dos funcionários do Núcleo de Tecnologia e Educação Aberta e a Distância da UEPG, ou NUTEAD. Serão controladas as portas onde existe a entrada e saída de funcionários do NUTEAD, registrando que funcionário passou e que horas passou.

Para fazer o controle, será utilizado um microcontrolador, que realizará as operações lógicas e controles necessários para o desenvolvimento do projeto proposto como o registro de passagem e acionamento da câmera para a captura de fotos. Parâmetros de entrada, como leitura do sensor de presença, recepção de sinal de transponder de radiofrequência são enviados e processados pelo microcontrolador. Dispositivos de saída, como por exemplo a câmera, são acionados pelo sinal do microcontrolador.

É objetivo deste projeto prover dados para uma melhor fiscalização dos funcionários públicos do setor contratante, possibilitando assim um melhor cumprimento dos horários estabelecidos em contrato.

Não é parte do escopo do trabalho controlar, fiscalizar e reforçar o uso dos cartões identificadores pelos funcionários. Essa parte é encarregada ao corpo de funcionários e o chefe do Núcleo, que cuidará do uso próprio dos cartões.

Não é enfoque desse sistema detectar obstruções nas câmeras e sensores. É responsabilidade do cliente garantir que a captação das fotos seja ideal, tendo em mente que as lentes da câmera são ajustáveis. As atividades dantes mencionadas são de responsabilidade do cliente e dos responsáveis por ele encarregados e treinados. As informações para a calibragem do sistema estarão contidas no manual que será desenvolvido e entregue ao cliente ao final do projeto.

O ambiente em que esse projeto será implantado pode ser visualizado na Figura 1.

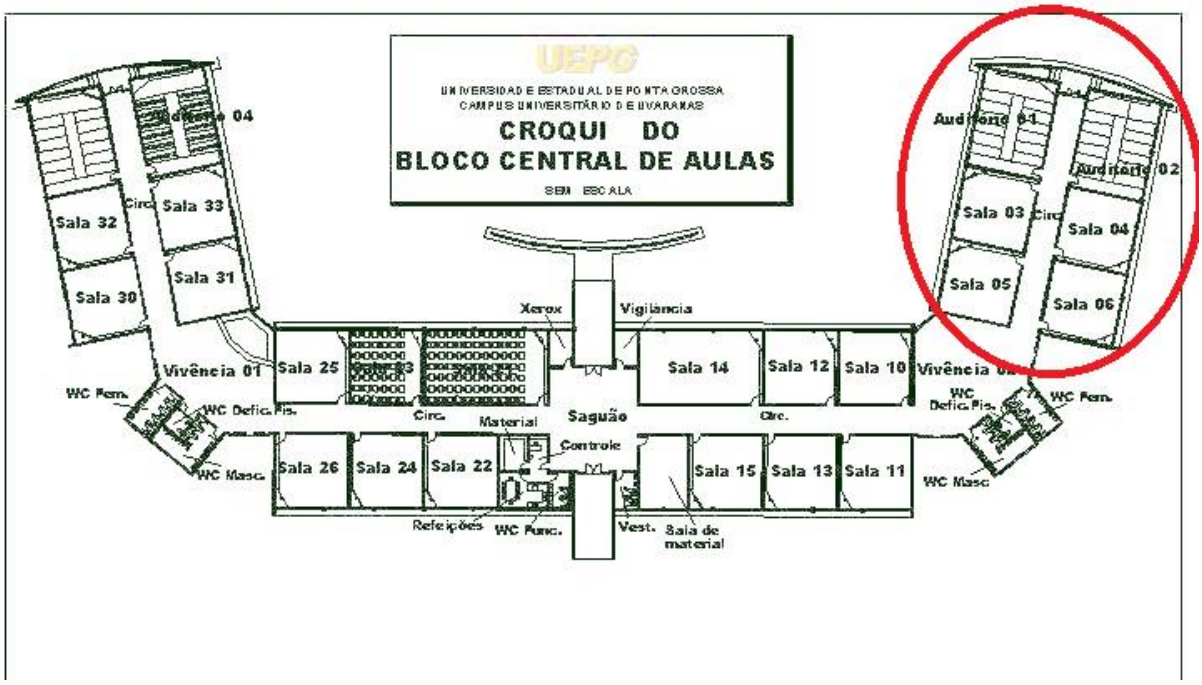

Figura 1 - Croqui do bloco Central de Salas da UEPG

O corredor onde está localizado o NUTEAD conta com seis salas, onde duas destas são auditórios onde são realizadas reuniões, palestras e aulas. Além disso o NUTEAD conta com quatro salas onde os funcionários desenvolvem atividades

relacionadas ao ensino à distância e dão manutenção ao sistema MOODLE da Universidade Estadual de Ponta Grossa.

Para o acesso ao Núcleo, conta-se com duas portas, localizadas nas extremidades do corredor onde está situado, conforme a Figura 1 demonstra.

4. PROPOSTA DO PROJETO

Para o sistema apresentado neste projeto, foi escolhido em primeiro momento o micro controlador presente na placa NodeMCU, o ESP8266. Cogitou-se também utilizar o micro controlador presente no Arduino UNO, o Atmega328p, porém, observou-se que o primeiro apresenta algumas vantagens sobre o microcontrolador presente no UNO.

A principal vantagem apresentada pelo ESP8266 é a possibilidade de conexão à rede sem fio, já o Atmega328p não possui essa função. Para realizar a comunicação sem fio usando o microcontrolador Atmega328p seria necessária a aquisição de componentes eletrônicos que agregariam pontos de possíveis falhas no circuito. Outra vantagem do ESP8266 com relação ao Atmega328p é o custo, visto que a aquisição de módulos para a conexão sem fio torna o custo agregado do Atmega328p maior, sem apresentar vantagens a seu favor. Assim, concluiu-se que a melhor opção para este projeto seria o uso do microcontrolador ESP8266.

Para a construção da fonte deve-se tomar em consideração a tensão de alimentação do microcontrolador e a do módulo RFID. O microcontrolador ESP8266 requer tensão de alimentação na faixa de 5V até 9V. O módulo RFID MFRC522, por sua vez, requer tensão de 3,3V. Conclui-se então que o circuito da fonte irá precisar de dois limitadores, um de 5V e outro de 3.3V, para a adequada alimentação do circuito.

O sistema terá uma câmera para garantir a entrada de funcionários e pessoas que entrem no NUTEAD, isso garantirá que funcionários que eventualmente esqueçam o identificador RFID tenham como provar que estavam no ambiente de trabalho mesmo não tendo passado o identificador RFID no sistema, este sendo uma atividade reservada ao supervisor do núcleo.

A linguagem que será utilizada na programação do micro controlador será C++, cujo desenvolvimento será feito na interface de desenvolvimento do Arduino, visto que é um software livre e compatível com o microcontrolador ESP8266.

4.1 BENEFÍCIOS

Os benefícios trazidos com a implantação do sistema de controle de entrada e saída concernem a segurança, ao controle de evasão de funcionários e a garantia de que empresa e funcionários terão registro do acesso e saída do NUTEAD.

Aprofundando, em que diz respeito à segurança, mesmo não tendo essa função como objetivo, o operador do sistema terá acesso as imagens que a câmera irá obter, tendo, juntamente com a imagem, data e hora em que todas as pessoas entraram e saíram do núcleo.

Em relação ao controle de evasão e a garantia dos registros, o sistema de controle de presença de funcionários é atualmente feito por ponto manual escrito com auxílio de um sistema básico recentemente implementado, onde os funcionários registram todo dia a hora de entrada e saída. o que dá brechas aos funcionários que podem registrar o ponto no início da carga horária, sair durante todo o período de trabalho e voltar a qualquer momento. O sistema não restringirá os horários em que os funcionários sairão do núcleo mas irá registrar as respectivas horas de entrada e saída de cada funcionário.

Além de garantir, com foto e registro, os horários em que os funcionários entraram no núcleo. Exemplificando, caso algum funcionário esqueça seu identificador RFID, ou simplesmente esqueça de registrar sua entrada, ainda haverá uma foto do horário em que chegou para provar que estava presente no horário que não conseguiu passar o identificador RFID, isso traz benefícios tanto para a empresa, quanto para os funcionários. Esse registro será feito manualmente pelo cliente.

4.2 CUSTOS

O custo da mão de obra é dado pelo piso salarial do engenheiro estabelecido na Lei Federal nº 4950-A/66 e da Resolução do Senado Federal n° 12/71. Tendo em mente que o engenheiro envolvido não tem formação completa, será adotado um valor

de 1,5 salário mínimo, valor comum para engenheiros júnior e estagiários em engenharia. O cálculo realizado para o valor da hora trabalhada pode ser observado na tabela 1.

Cálculo do preço da hora de trabalho	
Salário Mínimo Nacional	R$ 954,00
Número de salários mínimos (jornada de 8 horas)	1,5
Número de horas trabalhadas por mês	200
Preço da hora de trabalho	R$ 7,16

Tabela 1 - Cálculo do preço da hora de trabalho

O número de horas trabalhadas para o desenvolvimento do projeto leva em conta 8 horas semanais por parte de ambos os integrantes da equipe. O cálculo é representado na tabela 2.

Horas de Trabalho	
Horas Semanais	30
Número de meses de trabalho	10
Tamanho da Equipe	1
Total de horas	1500

Tabela 2 - Horas de Trabalho

A tabela 3 representa os custos levantados para o desenvolvimento do projeto, incluindo componentes utilizados e mão de obra. O valor usado para o cálculo do custo da hora foi o de metade do valor da hora de um engenheiro pleno, tendo em mente que ambos os membros da equipe são acadêmicos de engenharia, e não engenheiros com a formação completa.

Custo do Projeto				
Quantidade	Nome	Loja	Preço	Total
2	Antena/receptor RFID - MFRC522	Eletrogate	R$ 22,90	R$ 45,80
30	Cartões padrão S50 Mifare1	Eletrogate	R$ 2,84	R$ 85,20
2	NodeMCU ESP8266 - ESP-12e	Conceito Tech	R$ 25,50	R$ 51,00
2	Sensor de Presença Infravermelho	FilipeFlop	R$ 9,90	R$ 19,80

3	Transformador Trafo 220/110 - 12V + 12V 1A	Filipeflop	R$ 28,90	R$ 86,70
3	Regulador de tensão L7805	Baú da Eletrônica	R$ 1,52	R$ 4,56
3	Regulador de Tensão LD1117V33	Robocore	R$ 3,30	R$ 9,90
3	Capacitor eletrolítico 1000uF 25V	Eletrônica Paraná	R$ 1,00	R$ 3,00
8	Capacitor cerâmico 100nF 25V	Eletrônica Paraná	R$ 0,50	R$ 4,00
12	Conectores	Eletrônica Paraná	R$ 2,00	R$ 24,00
12	Diodo 1N4007	Eletrônica Paraná	R$ 0,50	R$ 6,00
2	Fusível 1.2-1.5A	Eletrônica Paraná	R$ 0,50	R$ 1,00
1	Multímetro digital Multilaser	Eletrônica Paraná	R$ 20,00	R$ 20,00
1	Fio de estanho para solda Best	Eletrônica Paraná	R$ 8,50	R$ 8,50
1	Metro de fio de cobre paralelo e revestido	Eletrônica Paraná	R$ 3,00	R$ 3,00
1	Protoboard 870 pontos	Eletrogate	R$ 17,00	R$ 17,00
1	Kit de 40 jumpers macho-macho	Robocore	R$ 40,00	R$ 40,00
1	Ferro de solda 25W 110V	Eletrônica Paraná	R$ 20,00	R$ 20,00
2	Display LCD I2C	Eletrogate	R$ 23,56	47,12
2	Plugue para tomada	Eletrônica Paraná	R$ 2,00	R$ 4,00
4	Câmera Wireless IP HD720p	Americanas	R$ 150,00	R$ 600,00
10	Placa de Circuito impresso	JLC PCB	R$ 12,47	R$ 124,70
1500	Hora de trabalho		R$ 7,16	R$ 10.797,20
Total dos materiais				R$ 1.160,58
Total				R$ 11.957,78

Tabela 3 - Tabela de Custos

A tabela 3 apresenta os custos estimados para o desenvolvimento do projeto proposto. A tabela acima conta com os custos até agora levantados e não conta com gastos imprevistos. Em caso da necessidade de novas aquisições de equipamentos e componentes o custo total será reajustado de acordo.

4.3 RISCOS

Como riscos inerentes ao projeto desenvolvido, há de se levar em conta diferentes situações onde as atitudes dos funcionários, sejam de má fé ou não, possam acarretar no mal funcionamento do sistema. Situações onde membros quadro de funcionários esquecem os seus identificadores acarretam na não identificação automática deles, o que pode levar em problemas caso isso se torne um costume.

O sistema depende da não obstrução da câmera para que a captura das fotos das visitas e das entradas e saídas sem o uso do identificador RFID seja feita. A obstrução da câmera pode acarretar em fotos borradas, pretas e de pouco uso para o reconhecimento de quem passou pelo ambiente.

Os identificadores RFID em formato de chaveiro são de pequeno tamanho podem ser facilmente perdidos. É necessário explicitar aos funcionários e os portadores de identificadores que o cuidado com o pequeno dispositivo é fundamental. Em casos onde a perda é recorrente o cliente deve adotar identificadores RFID em formato de cartão, onde o tamanho é maior e a probabilidade de perda menor.

Os identificadores e antena receptora trabalham na frequência de até 125 MHz, com os usados nesse projeto trabalhando na frequência de 13,56MHz. Segundo a *Federal Communications Commission*, órgão federal estadunidense que regula os padrões e informações dos diferentes meios de comunicação a radiofrequência é inofensiva para humanos, animais e plantas visto que a energia das ondas de radiofrequência não é alta o suficiente para causar danos.

O sensor de movimento que identifica a passagem de pessoas não deve ser obstruído. Em casos de passagem e o buzzer soará para alertar o cliente. Em casos de sons prologados do buzzer o cliente deverá resolver a possível obstrução do sensor.

O sistema pode servir como medida paliativa para possivelmente identificar estranhos e atividade suspeita, visto que fotos serão tiradas quando pessoas não identificadas passam, entretanto, o sistema não possui nenhuma medida que tome uma atitude quanto a isto, visto que esta não é sua proposta.

4.4 DESCRIÇÃO DA MODELAGEM PRELIMINAR

O sistema desenvolvido nesse projeto terá forma geral conforme mostrado nas Figuras 2 a 4.

O diagrama de casos de uso do projeto, visível na figura 2, descreve as funcionalidades básicas e interações com agentes externos.

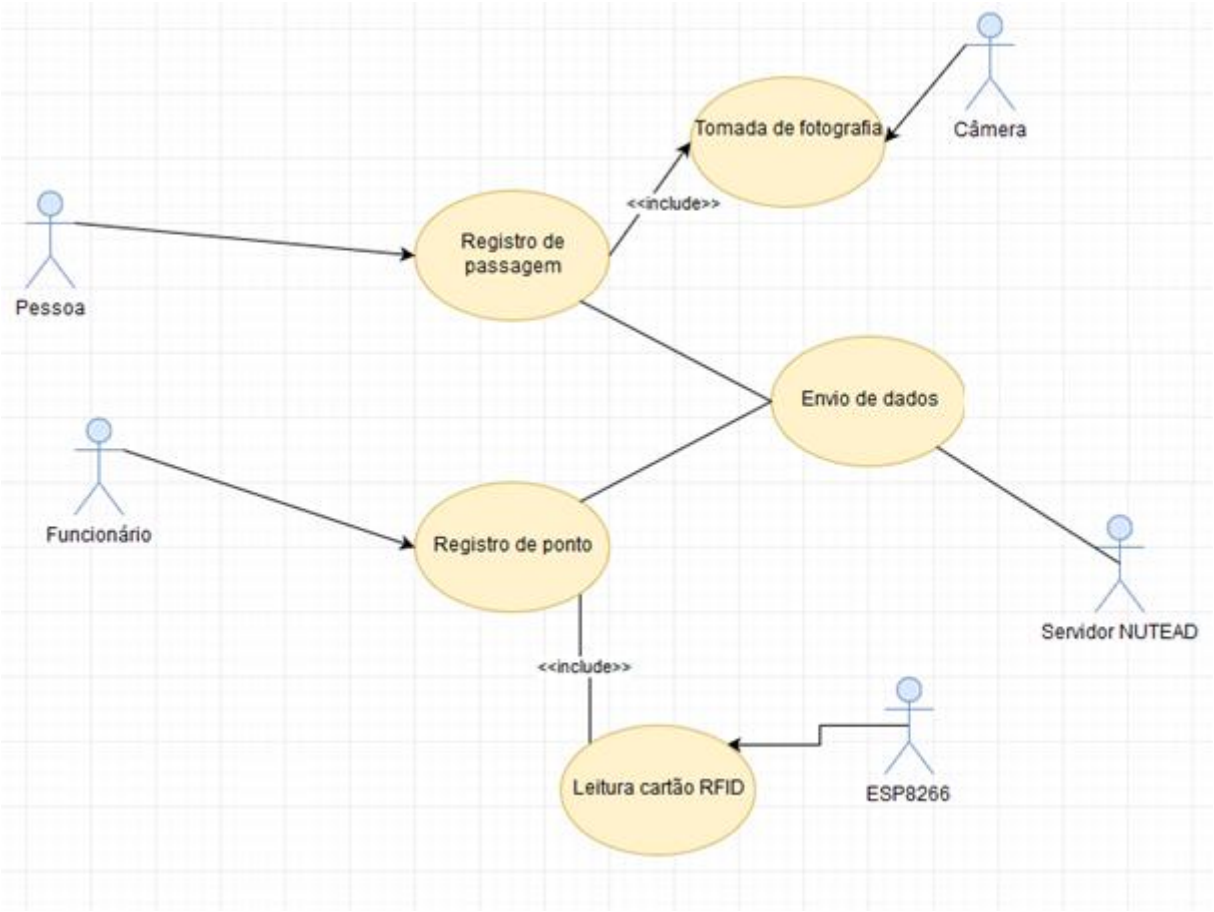

Figura 2 - Diagrama de Casos de Uso

Conforme demonstrado na Figura 2, pode-se verificar que, como agentes, o sistema terá como agentes externos a câmera IP e o servidor do NUTEAD, além das pessoas e funcionários. As funcionalidades oferecidas pelo sistema são a de registrar a saída e entrada(passagem) e a captura das fotos.

Como fluxo de dados e informação, o sistema conta com a Figura 3 para melhor visualização do processo.

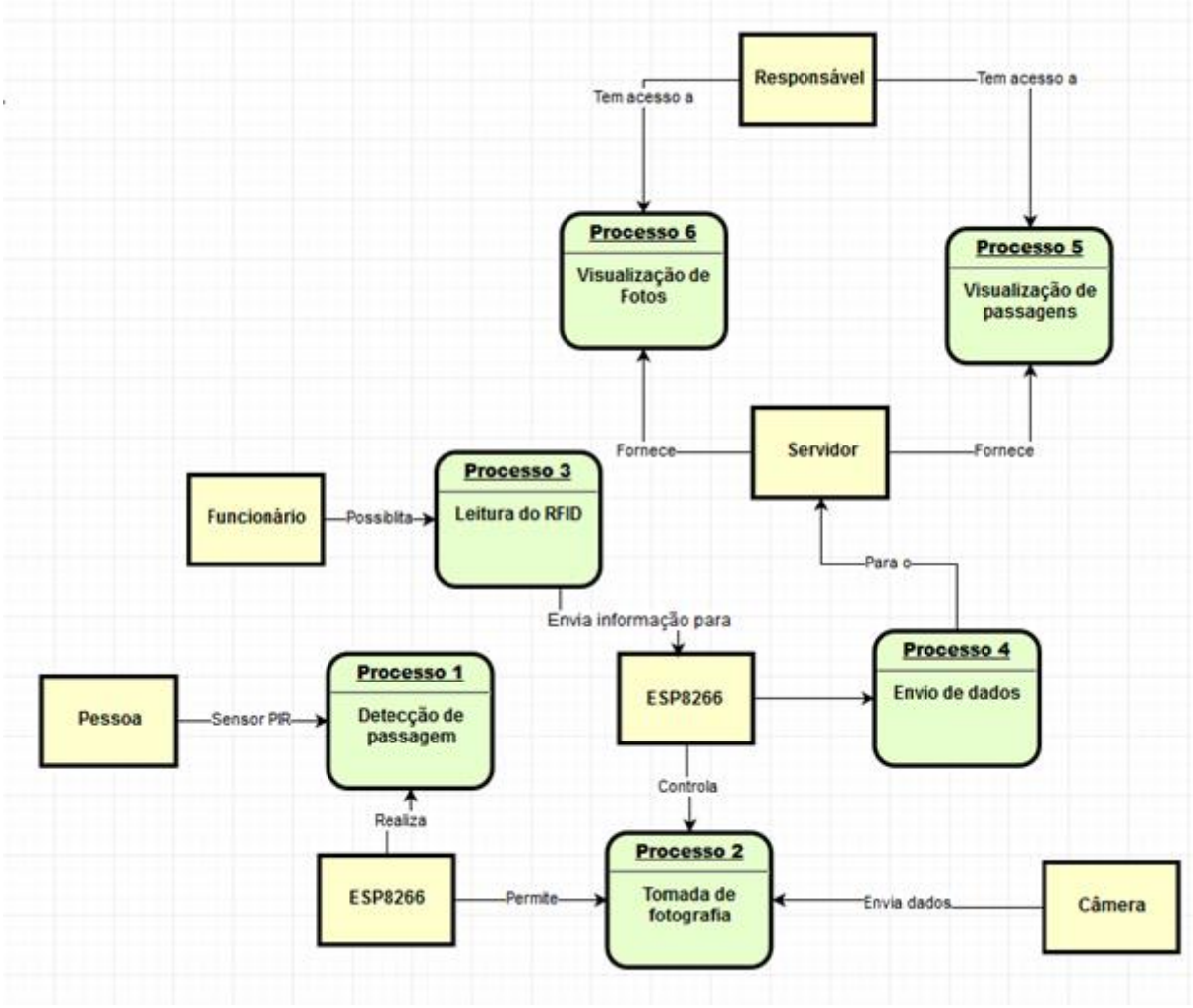

Figura 3- Diagrama de Fluxo de Dados do projeto

O diagrama conta com a definição dos processos e as interações entre os diferentes atores do sistema. O diagrama acima descreve os seguintes passos:

- Pessoa é detectada pelo sensor ligado ao ESP8266
- ESP8266 aciona a câmera IP responsável pela entrada em que está localizado. Esse acionamento é feito usando requisições HTTP/GET usando a rede Wi-Fi.
- ESP8266 realiza a leitura do identificador RFID e envia os dados para o servidor usando a conexão Wi-Fi
- Servidor hospeda a aplicação WEB que possibilita a visualização de passagens e fotos

Para a ilustração da modelagem inicial do banco de dados, conta-se com a Figura 4.

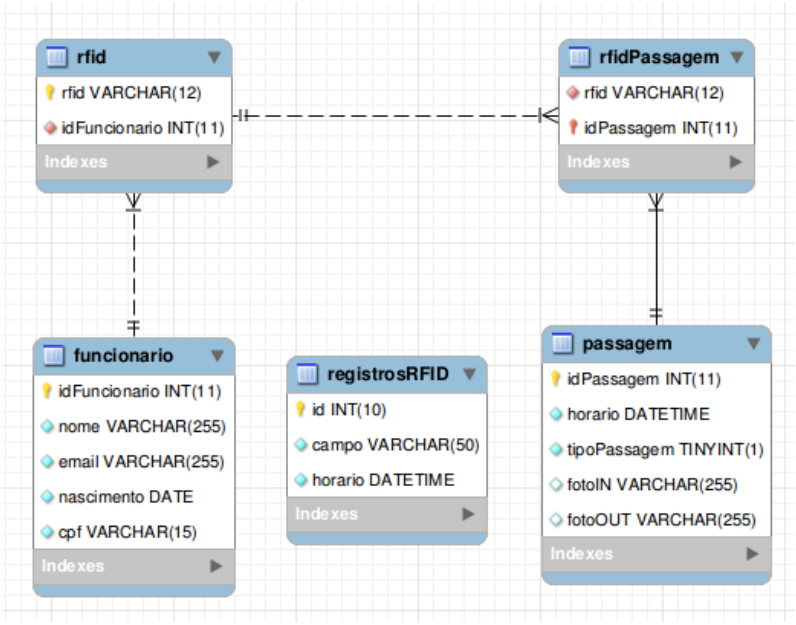

Figura 4 - Diagrama de Martin para o banco de dados

Na tabela passagem é registrada o sentido da passagem, o caminho da foto no sistema de arquivos do servidor e o horário em que a passagem aconteceu. Relacionada a uma passagem pode existir um registro de RFID, que identifica que a passagem foi de um funcionário. A tabela RFID relaciona os identificadores RFID dos funcionários ao seus donos. Na tabela fucionário são armazenados o email, nome, data de nascimento e cpf do funcionário, dados que foram requisitados pelo cliente para a posterior integração do sistema com o sistema central do NUTEAD. Essa integração será feita futuramente pelo cliente.

É importante ressaltar que sistema irá armazenar as imagens tiradas durante a passagem de pessoas no sistema de armazenamento do servidor, guardando apenas o caminho da imagem para posterior visualização. Isso acarreta em maior velocidade para o acesso dos dados.

4.5 PARA DESENVOLVIMENTO DE HARDWARE

4.5.1 DEFINIÇÃO DOS REQUISITOS

Para o circuito funcionar corretamente e sem problemas, se faz necessário uma a construção de uma fonte com tensão de saída 5V e corrente de 1A.

O valor para tensão é dado pelos componentes usados, já o valor de corrente é adotado devido ao consumo de corrente dos componentes, exposto na seção de cálculos deste anteprojeto. Além disso, se faz necessário também alimentação a 3.3V para os módulos de radiofrequência e para os sensores de presença visto que a tensão fora de sua faixa de trabalho ocasiona em mal funcionamento dos componentes eletrônicos.

O circuito integrado (CI) ESP8266 realizará a leitura dos sensores e módulo de radiofrequência, realizando o envio das informações do funcionário para o servidor por meio de conexão Wi-Fi. Além dessa função ele será responsável pelo sinal que deverá acionar a câmera.

Os módulos de câmera disponíveis aos microcontroladores possuem taxa de transferência e de tomada de foto limitadas pela velocidade de transmissão e processamento do microcontrolador, tal qual acarreta em atrasos de processamento da informação que não são compatíveis com o pedido do sistema por parte do cliente.

Para superar essa limitação é necessário usar um meio de transporte e transmissão que não sejam limitados por esses fatores. Será usado, então, câmeras IP ligadas na rede Wi-Fi já existente no núcleo. As câmeras IP possuem protocolos de transferência de dados com maiores velocidades e garantia de recebimento de dados, como o TCP e o HTTP.

Para o registro da entrada e saída dos funcionários será necessária uma maneira de identificar os cartões unicamente. Para isso, poderá ser usado os cartões Mifare1 S50, que possuem o tamanho padrão de cartões de crédito ou tamanho pequeno de chaveiros, ideal para carregar no dia a dia de trabalho.

A leitura dos cartões pode ser realizada por meio do módulo MFRC522, o módulo possui uma biblioteca que possibilita a leitura dos cartões de padrão Mifare1. O módulo requer alimentação de 3,3V e utiliza o padrão de comunicação *Serial Peripheral Interface* (SPI).

Para a verificação da presença de um funcionário saindo do ambiente, é necessário o uso de um par de sensores de obstáculos. O sensor de obstáculos conta com um par de receptor/emissor infravermelho e um circuito integrado LM393 que verifica pela detecção do feixe de luz infravermelha emitida pelo sensor, sendo ideal

para o caso, já que resulta em falsos positivos se estiver bem posicionado. O uso do par de sensores garante a detecção do sentido da entrada pela ordem da ativação dos mesmos.

4.5.2 REVISÃO DOS REQUISITOS

Para o total funcionamento da placa com o microcontrolador ESP8266, é requerida uma tensão na faixa de 4,5V à 9V. A corrente necessária para o funcionamento é de 212mA conforme demonstrado nos cálculos do projeto e no *datasheet* do componente.

Para a fonte de alimentação será usado um transformador 220/110V para 12V. O transformador deverá ter capacidade para a passagem 1A de corrente elétrica, corrente o suficiente para a alimentação do circuito, conforme demonstrado na seção de cálculos.

O circuito integrado limitador de tensão que será usado, o LM7805, pode fornecer uma tensão de 5V, tendo como entrada uma tensão na faixa de 7 até 25V. Sua corrente máxima é de 1A, o que está dentro dos requerimentos dos componentes.

Além disso, componentes como o sensor de presença DYP-ME003 e o módulo leitor de RFID MFRC522 requerem alimentação na faixa de 3,3V. Para tal tensão, será usado o circuito limitador de tensão LD1117V33, circuito que recebe de 4,3V até 15V para fornecer uma tensão de 3,3V com até 800mA, ideal para o funcionamento dos módulos e sensores citados.

Para a leitura da passagem de funcionários, será usado a tecnologia de RFID. O RFID, ou *Radio Frequency Identification*, tem como vantagem o baixo custo, a necessidade de alimentação apenas no leitor e o pequeno tamanho. Os cartões, tags e leitores RFID operam na frequência de 13,56MHz. O módulo RFID MFRC522 lê cartões no padrão Mifare1, um cartão amplamente adotado no mercado pelo seu baixo custo e seu espaço de 1KB de armazenamento.

O módulo MFRC5202 trabalha usando o protocolo de comunicação SPI. O protocolo SPI possibilita a ligação de múltiplos dispositivos nos mesmos pinos do microcontrolador, fazendo o controle, em software, de qual está ativo no momento. A multiplexação dos dispositivos ligados é responsabilidade do programador na hora do desenvolvimento. O uso do protocolo SPI possibilita a ligação de múltiplos módulos

RFID ou ainda outros dispositivos SPI, acarretando assim em um melhor aproveitamento do microcontrolador Esp8266.

Para a captura das fotos, câmeras IP (*Internet Protocol*) são uma escolha sensata. As câmeras IP possuem comunicação usando interface Ethernet (possibilitando grandes distâncias) ou conexão Wi-Fi (deixando o ambiente livre de cabos). O modelo de câmera é independente, contanto que suporte o padrão de streaming por RTSP.

Para a verificação da presença de um funcionário saindo do ambiente, é necessário o uso de um sensor de presença. O sensor de obstáculos emite luz infravermelha e verifica se ela retorna. Caso detectar luz infravermelha significa que há reflexão por parte de algum objeto, assim detectando a presença dele.

4.5.3 DEFINIÇÃO DO DIAGRAMA DE BLOCOS

O diagrama de blocos do sistema está representado na figura 5.

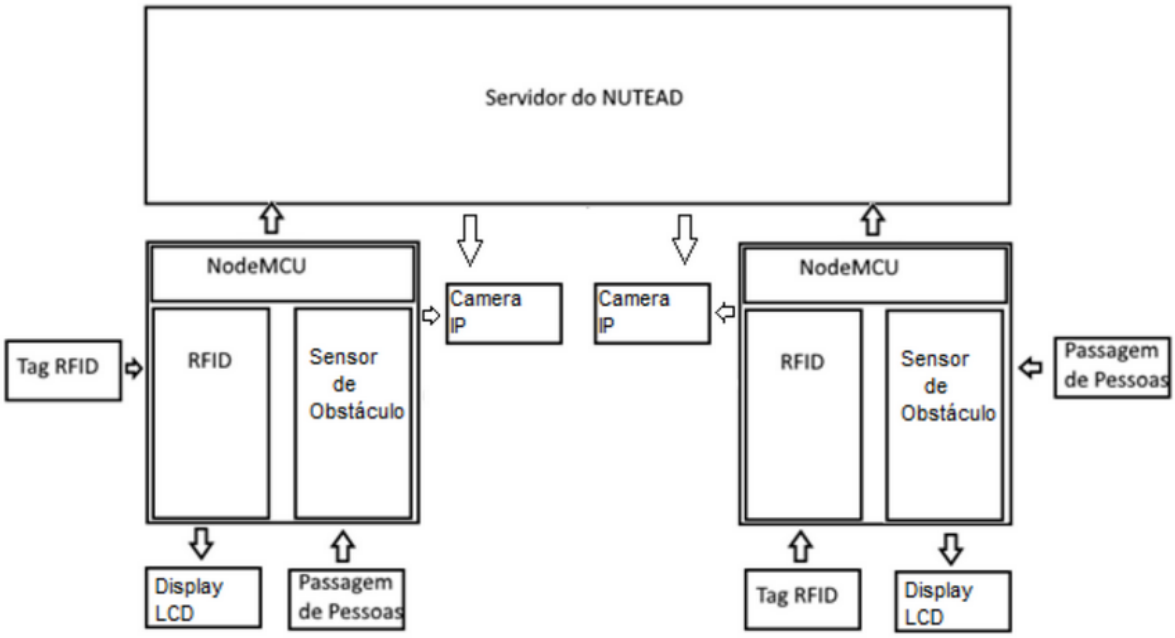

Figura 5 - Diagrama de blocos do projeto

Os componentes da parte de detecção consistem de uma placa NodeMCU, que possui um microcontrolador ESP8266, um módulo RFID MFRC522 para a detecção e leitura de cartões RFID dos funcionários do NUTEAD. Além disso o circuito de detecção também possui com um sensor de obstáculo, que detecta a passagem de pessoas.

Na detecção de uma pessoa o microcontrolador ESP8266 mandará um sinal para uma aplicação executando no servidor do NUTEAD. O servidor por sua vez, realizará a captura de uma foto por meio de uma câmera IP, controlada pela rede Wi-Fi. A câmera, após a captura da foto, a enviará foto para o servidor do NUTEAD, que salvará as imagens em disco rígido. A conexão entre ESP8266, servidor e câmera será feita usando o protocolo TCP/IP, garantindo assim a chegada dos pacotes em segurança.

4.5.4 PROJETO E ELABORAÇÃO DO CIRCUITO ELETROELETRÔNICO

Para atingir os objetivos propostos por este projeto foi desenvolvido um circuito para a obtenção dos pontos dos funcionários, usando a tecnologia de RFID, e o acionamento da câmera por meio do movimento. A figura 6 representa o circuito em seu estado atual.

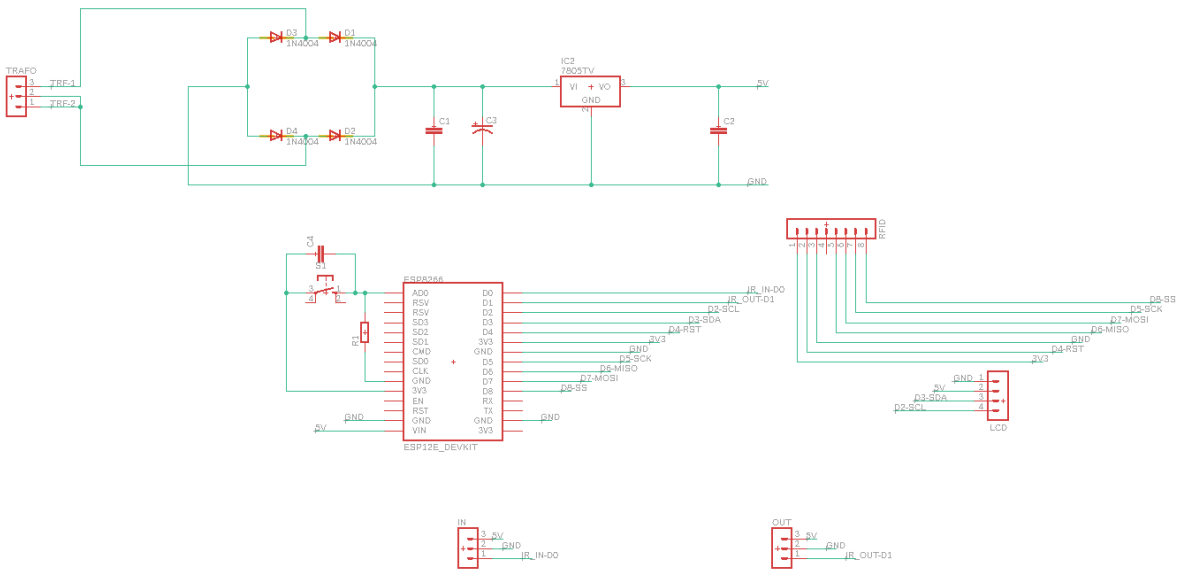

Figura 6 - Circuito do projeto

Irá ser confeccionada uma fonte, já apresentada no circuito demonstrado na figura 6, que fornecerá a tensão de 5V e 3,3V. As tensões fornecidas pela fonte são as tensões previamente estabelecidas para o funcionamento dos diferentes módulos.

A figura 7 pode ser dividida em circuito da fonte de alimentação e circuito do microcontrolador.

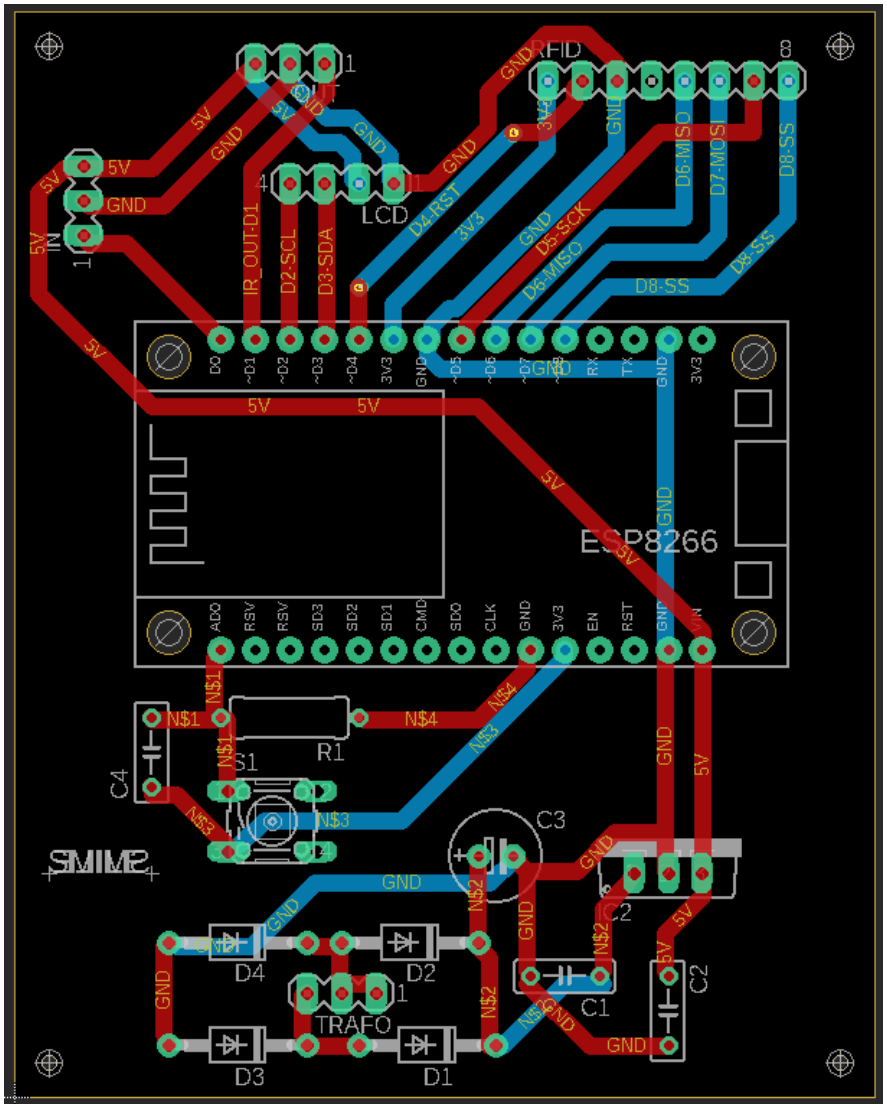

Figura 7 - Circuito impresso

O circuito da fonte conta com alimentação da rede de energia elétrica de 220V a 60Hz. Existe um fusível de proteção antes da energia passar pelo transformador, onde a tensão será reduzida para 12V AC. Após isso, a energia passará por um circuito retificador completo, com 4 diodos. Após a retificação existem dois capacitores, C1 e C2, que são para filtragem e geração do efeito de Ripple, onde a onda é deixada mais próxima de uma reta. Como estágio final da fonte a corrente passa por um

circuito limitador de tensão de 5V e por um circuito limitador de tensão de 3,3V que alimentarão o resto do circuito.

O circuito do microcontrolador conta com um módulo RFID RC522 para a leitura dos identificadores RFID. O módulo é ligado nos pinos de interface SPI da placa NodeMCU. Um *display* LCD nos pinos D3 e D2 do microcontrolador, que serão definidos no código como pinos do protocolo de comunicação serial I2C. Por final, os sensores para detecção da passagem serão ligados nos pinos D1 e D0.

O funcionamento do programa do microcontrolador realizará as seguintes operações repetidamente:

- Detecção da passagem
- Leitura do identificador RFID

Em caso de detecção, o *display* LCD será acionado e indicará a mensagem de sucesso ou falha da operação.

Na figura 7 é possível observar o circuito esquematizado para a corrosão e solda em placa de fibra de vidro revestida de cobre.

4.5.5 DESCRIÇÃO E FORMALIZAÇÃO DO PROJETO EM ENGENHARIA

Será utilizada inicialmente a linguagem C++ para a programação do microcontrolador, utilizando a interface de desenvolvimento da família Arduino de microcontroladores, já que a mesma é compatível com a linguagem e com o controlador escolhido no projeto, o ESP8266.

Para a programação do servidor WEB será utilizada a linguagem de programação PHP

A linguagem utilizada no servidor local para o acionamento das câmeras será Python, por ser linguagem amplamente adotada, sendo uma das mais utilizadas atualmente, e por possuir um conjunto de instruções simples e completas. A comunicação entre o microcontrolador e o servidor local será feita com base no protocolo TCP/IP, tendo em vista que este protocolo apresentado tem a capacidade de retornar as informações enviadas, caso não cheguem ao destino.

A linguagem a ser utilizada no banco de dados será MySQL, já que o projeto não necessitará de um banco muito robusto.

Foram escolhidas câmeras IP para a obtenção das fotos após a passagem no sensor de presença. Já que as duas câmeras ficarão posicionadas relativamente distantes a conexão por cabos é um empecilho no desenvolvimento do projeto.

Os dispositivos de entrada do sistema possuem as seguintes especificações:

- Módulo RFID MFRC522 – Padrão SPI para comunicação, com alimentação de 3,3V
- Sensor de obstáculos – Comunicação simples de dois estados, alimentação de 5V

Foram escolhidas câmeras IP para a obtenção das fotos após a passagem no sensor de presença. Já que as duas câmeras ficarão posicionadas relativamente distantes, é um empecilho usar de conexão cabeada entre câmera e roteador.

Os dispositivos na saída do sistema possuem as especificações a seguir:

- Display LCD – Padrão I2C para comunicação, alimentação de 5V
- Câmera IP – Stream de dados via protocolo RTSP (*Real Time Streaming Protocol*)

Funcionalidades usadas oferecidas pelo microcontrolador ESP8266 que serão utilizadas durante o desenvolvimento do sistema são:

- Comunicação via rede Wi-Fi usando o protocolo TCP
- Uso da interface SPI conforme especificada originalmente pelo fabricante
- Uso da interface I2C, com alteração nos pinos originais para melhor usufruto das portas do microcontrolador

.

4.5.6 ESPECIFICAÇÕES

O ESP8266 é um circuito integrado que possui 32 pinos, sendo 11 deles saídas/entradas digitais e 1 pino para entradas analógicas conforme demonstra a

figura 7. Esse CI funciona a uma tensão de 3,3V. Na placa NodeMCU, existem outros dois CI's, um para conexão serial do ESP para carregamento e substituição dos programas presentes no mesmo, o CP2102, e um regulador de tensão de 5 para 3,3 volts , o AM1117, que serve como auxiliar do CP2102 já que a placa possui conexão micro-USB, tanto para alimentação quanto para funcionamento. A função de alimentação do conector micro-Usb não será utilizada.

A figura 8 ilustra o *pinout* da placa NodeMCU. A placa conta com 16 GPIO, ou entradas digitais, para o uso na ligação de componentes e programação da placa. A placa conta com pinos já especificados para a interface SPI, interface a qual é utilizada pelo módulo RFID MFRC522 adotado no projeto.

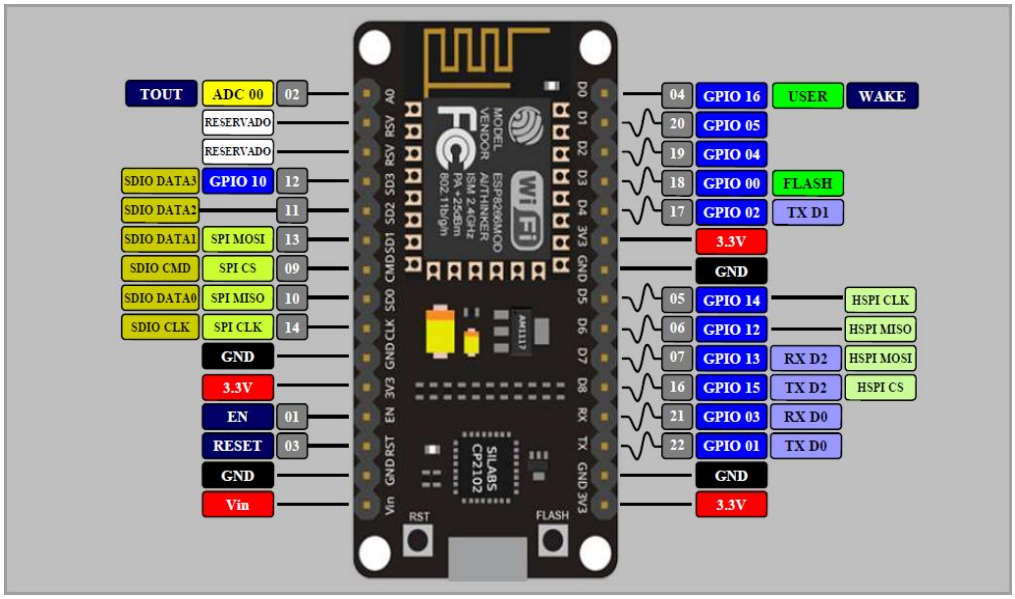

Figura 8 - Pinout da placa NodeMCU e disposição dos componentes

Fonte: http://www.ifuturetech.org/ifuture/uploads/2017/07/AMICA-NODEMCU-ESP8266-LUA-CP2102-WIFI-DEVELOPMENT-MODULE-IOT-gujarat.png

O sistema utilizará câmeras IP com suporte ao protocolo RTSP. Além disso as mesmas possuirão conexão via rede sem fio, evitando conexão por cabos.

O sensor de obstáculos escolhido foi o sensor com emissor/receptor infravermelho com base no LM393, que processa a saída analógica do sensor e transforma em um sinal digital. A sensibilidade desse sensor pode ser ajustada por

um potenciômetro presente no módulo. Esse módulo é equipado. A distância máxima de detecção pode ser ajustada de 2 a 30 centímetros.

O módulo de gravação e leitura de radiofrequência escolhido foi o MFRC522, é um módulo que trabalha a uma frequência de 13,56 MHz. Utiliza tecnologia MIFARE Classic que consegue fazer ambas gravação e leitura de dados a mais de 848kbit/s.

Para a produção do servidor local, será usada a infraestrutura local do NUTEAD, que conta com um servidor Apache2 que interpreta e administra as requisições HTTP. O servidor conta com a linguagem PHP para o desenvolvimento do *back-end*.

4.5.7 FUNÇÕES DO MICROCONTROLADOR E MICROPROCESSADOR

O microcontrolador fará o papel de receber os sinais dos sensores via portas digitais, RFID e de presença, e passar os sinais para o servidor, que fará a comunicação com as câmeras, que obterá as fotos e salvará as entradas no banco de dados do NUTEAD. Além de conexão entre sensores e microprocessador o ESP8266 possui a funcionalidade de fazer operações matemáticas e lógicas.

A comunicação será feita via rede sem fio visto que ambos, microcontrolador e servidor possuem essa possibilidade.

4.5.8 OPERAÇÃO

Depois de ligado à rede de energia elétrica o sistema estará ligado, só será necessário fazer a ligação quando o sistema for desligado manualmente.

Existem várias formas de operar o sistema, cada uma tem um objetivo diferente. Os funcionários irão utilizar o sistema sendo detectados automaticamente pelo sensor de presença, em seguida, devem passar sua identificador RFID no leitor que estará nas proximidades da porta do NUTEAD. Isso valerá tanto para entrada quanto para saída. O administrador do sistema terá acesso aos dados coletados. Esses dados, que incluem horários, quantidade de entradas e saídas, fotos obtidas pela câmera, poderão ser observados e transformados em relatórios para conferências e análises. O administrador poderá adicionar manualmente as entradas em casos que julgar justos e necessários.

4.5.9 RESTRIÇÕES

A seguir estão apresentadas as restrições feitas ou encontradas no projeto, até o presente momento. A restrição que mais se destaca é a de tempo, que tem de ser muito bem administrado pelo engenheiro responsável que realizará o projeto, cumprindo o cronograma e, se possível, adiantando atividades subsequentes.

Existem restrições quanto a utilização do sistema pelos funcionários do núcleo. A mais importante é a de utilização dos identificadores RFID corretamente, no que diz respeito ao oblívio (esquecimento). É estritamente necessária a utilização das mesmas para que funcione como sistema de presença, já que o sistema de ponto adotado pelo núcleo atualmente será substituido pelo apresentado neste projeto.

Outra restrição importante é a que diz respeito a obstrução das lentes das câmeras, que deverá ser de responsabilidade do administrador do sistema ou alguém designado pelo mesmo. Ainda sobre a câmera, existe a restrição de ângulo e perspectiva da mesma, que fará com que a posição do leitor de identificadores RFID tenha que ser bem planejada para melhor identificação de pessoas que passem pelo sistema.

Uma restrição vital para o ideal funcionamento do sistema é a correta configuração do servidor onde será hospedada a aplicação que receberá e mostrará os dados. Em caso de horários errados, a aplicação não funcionará de maneira esperada e apresentará erros nos registros das entradas e saídas no bloco.

É importante ressaltar que o sistema dependerá da rede sem fio disponibilizada pelo núcleo, já que existirá conexão entre partes do projeto realizada pela rede.

5. CÁLCULOS

Como cálculos preliminares, foi realizado o cálculo de corrente usada pela placa NodeMCU para o dimensionamento da fonte que alimentará o dispositivo. A tabela 4 apresenta os valores de consumo de corrente dos respectivos componentes.

Cálculo da corrente no microcontrolador		
Máximo por pino	0,012	A
Número de pinos	16	un.
Consumo passivo	0,02	A

Consumo sensor infravermelho	0,00015	A
Módulo MFRC522	0,026	A
Consumo de corrente (soma dos consumos)	0,23815	A
Consumo de corrente	238,15	mA

Tabela 4 - Cálculo do consumo de corrente

Os valores de corrente representados na tabela 4 foram retirados dos datasheets do componentes apresentados. O cálculo do consumo total foi dado pela soma do consumo de todos os componentes ligados à unidade microcontroladora.

Para o cálculo do valor do resistor de *Ripple* (MILLMAN, 2010), na fonte de alimentação, é adotada fórmula:

$$V_{dc} = V_m - \frac{I_{dc}}{4fC}$$

Onde:

- V_{dc} – Tensão nominal de saída do transformador (12 V)
- V_m – Tensão desejada para o funcionamento do circuito (9 V)
- I_{dc} – Consumo de corrente do circuito (238 mA)
- f – Frequência da rede elétrica (60 Hz)
- C – Valor do capacitor de Ripple

Para a obtenção de uma tensão maior ou igual à 9V, o valor de C calculado é de 330 µF. Sendo assim, o valor do capacitor deve ser maior ou igual a 330µF para correção do efeito Ripple.

Segundo Millman (2010), o valor do capacitor de filtro é calculado usando a fórmula de filtro passa-baixa, dada por

$$f_c = \frac{1}{2\pi RC}$$

Onde:

- R – Resistência no circuito de filtro
- Vm – Tensão desejada para o funcionamento do circuito (9 V)
- Idc – Consumo de corrente do circuito (238 mA)
- f – Frequência da rede elétrica (60 Hz)
- C – Valor do capacitor de filtro

Para o corte de frequências altas (acima de 10 Hz), adotando um baixo valor de resistência(20 Ω), têm-se que para cortar frequências altas o valor do capacitor deve ser ao menos 70 µF.

6. CRONOGRAMA

O cronograma do desenvolvimento do projeto pode ser observado na tabela abaixo

Cronograma											
Atividades	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
Reunião inicial com o cliente	X										
Escrita da Definição do Projeto		X									
Levantamento dos requisitos		X									
Pesquisa por componentes		X									
Escrita da documentação		X									
Protótipo do circuito			X	X							
Testes do protótipo			X	X							
Correção do protótipo			X	X							
Programação do programa das câmeras			X	X							
Programação da interface microcontrolador e Raspberry			X	X	X	X	X	X			
Documentação dos códigos			X	X	X	X	X	X			
Elaboração da placa de circuito impresso			X	X	X	X	X	X			
Testes do circuito impresso						X	X	X	X		
Correção do circuito impresso									X	X	
Criação do vídeo										X	X
Correções do projeto final										X	X

Tabela 5 - Cronograma do projeto

O desenvolvimento do sistema se dará durante o período do ano letivo de 2018.

7. CONCLUSÃO

Pode-se concluir com este anteprojeto que a proposta do cliente para projeto é um problema válido a ser resolvido. O problema do controle de entrada e saída é um excelente caso para o estudo e solução na disciplina de Projeto de Sistemas de Informação.

Além disso, mostrou-se uma proposta para solução do problema do cliente, esta estando sujeita à avaliação da banca da disciplina.

Espera-se durante o ano o bom desenvolvimento da proposta de projeto e a solução do problema apresentado pelo cliente.

8. REFERÊNCIAS BIBLIOGRÁFICAS

Datasheet DYP-ME003. **Open Impulse**. Disponível em: https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf. Acesso em 01 abr 2018.

Datasheet MFRC522. **NXP Semiconductors**. Disponível em: https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf. Acesso em 01 abr 2018.

Documentação do NodeMCU. **Esp8266 Community**. Disponível em: https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf. Acesso em 30 mar 2018.

Documentação do Raspberry Pi. **Raspberry Pi Foundation**. Disponível em: https://www.raspberrypi.org/documentation/. Acesso em 03 abr 2018.

MILLMAN, Jacob. Integrated electronics. 2010.

RF Safety FAQ. **Federal Communications Comission**. Disponível em: https://www.fcc.gov/engineering-technology/electromagnetic-compatibility-division/radio-frequency-safety/faq/rf-safety#Q2. Acesso em 28 mar 2018.

9. RESPONSABILIDADES

Data de assinatura: ___/___/_____

_____

João Gabriel Corrêa Krüger (engenheiro responsável)

_____

Albino Szesz Junior (Gestor de TI – NUTEAD/UEPG)

ANEXO A – Código C++ do microcontrolador Esp8266

```
// Armazenamento
#include <ArduinoJson.h>        // JSON
#include <FS.h>                 // FS - Sistema de arquivos SPIFFS

// Wi-Fi Manager
#include <ESP8266WiFi.h>        // ESP Wi-Fi
#include <DNSServer.h>          // DNS - redireciona as requisições
#include <ESP8266WebServer.h>   // WebServer - Possibilita usar o ESP de
servidor facilmente
#include <WiFiManager.h>        // Portal Captive

// Funcionamento do projeto como um todo
#include <SPI.h>                // SPI - protocolo do MFRC522
#include "MFRC522.h"            // MFRC522 - RFID da Mifare
#include <Wire.h>               // I2C - protocolo do display LCD
#include <LiquidCrystal_I2C.h> // LCD I2C - display para feedback das ações



// Variáveis do dispositivo - preparadas para serem configuradas no portal
Captive
char camera_entrada[128];
char camera_saida[128];
char identificador[64];
char ip_servidor[128];
char porta_servidor[6];
char camera_entrada2[128];
char camera_saida2[128];
char qualidadeStr[2];
int qualidade;
WiFiManagerParameter custom_camera_entrada("ipentrada", "IP da camera de
entrada - HQ", camera_entrada, 128);
WiFiManagerParameter custom_camera_saida("ipsaida", "IP da camera de saida
- HQ", camera_saida, 128);
WiFiManagerParameter custom_camera_entrada2("ipentrada2", "IP da camera de
entrada - LQ", camera_entrada2, 128);
WiFiManagerParameter custom_camera_saida2("ipsaida2", "IP da camera de
saida - LQ", camera_saida2, 128);
WiFiManagerParameter custom_identificador("identificador", "Identificador
do dispositivo", identificador, 64);
WiFiManagerParameter custom_ip_servidor("ip_servidor", "IP do servidor",
ip_servidor, 64);
WiFiManagerParameter custom_porta_servidor("porta_servidor", "Porta do
servidor", porta_servidor, 6);
WiFiManagerParameter custom_qualidade("qualidadeStr", "Qualidade (1 para HQ
ou 0 para LQ)", qualidadeStr, 1);

// Dados do Servidor
char* host; // IP do Servidor
int   port;            // Porta do Servidor


// Flag para salvar dados no FS
bool shouldSaveConfig = false;

// Flag para resetar o ESP para as configurações originais - DEVE SER USADO
PARA TESTES APENAS
```

```
bool deveResetar = false;


// Pinos do NodeMCU: https://jgamblog.files.wordpress.com/2018/02/esp8266-nodemcu-pinout.png

/* Ligando o MFRC522 ao NodeMCU (ESP-12F)
 *  RST     = D4
 *  SDA(SS) = D8
 *  MOSI    = D7
 *  MISO    = D6
 *  SCK     = D5
 *  GND     = GND
 *  3.3V    = 3.3V
 *  RQ      = Não ligado
*/

#define RST_PIN  D4
#define SS_PIN  D8
MFRC522 mfrc522(SS_PIN, RST_PIN); // Instância do MFRC522



/* Ligando o LCD I2C ao NodeMCU
 *  SCL = D2
 *  SDA = D3
 *  VCC = 5.0V
 *  GND = GND
 *  Endereço de memória - 0x27
 */

#define SCL_PIN D2
#define SDA_PIN D3
LiquidCrystal_I2C lcd(0x27, 16, 2); //FUNÇÃO DO TIPO "LiquidCrystal_I2C"


// Pinos para detecção de sentido
#define inPin D0
#define outPin D1



// Prevenção de excesso de registros
const int watchdog = 5000;
unsigned long previousMillis = 0;
/*
void conectaWifi(){
  //Serial.print("Conectando a ");
  //Serial.println(ssid);
  escreveMensagemLCD("Conectando a", ssid);


  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    //Serial.print(".");
  }

  //Serial.println(F("\nConectado ao Wi-Fi\n"));
```

```
  escreveMensagemLCD("Conectado!", "");
  //Serial.print("Endereço IP: ");
  //Serial.println(WiFi.localIP());
  //Serial.println(F("\nPronto para realizar o envio!"));
  //Serial.println(F("=====================================================
=\n\n"));
}*/

void iniciaMFRC(){
  SPI.begin();           // Iniciando barramento SPI
  mfrc522.PCD_Init();    // Iniciando MFRC522
  escreveMensagemLCD("RFID conectado!", "");
}


String recuperaRFID(byte *buffer, byte bufferSize){
  String content = "";
  for (byte i = 0; i < bufferSize; i++) {
    content.concat(String(buffer[i] < 0x10 ? " 0" : " "));
    content.concat(String(buffer[i], HEX));
  }
  content.toUpperCase();
  return content;
}

void leCartao(){
  escreveMensagemLCD("Escaneando RFID", "");
  // Procura novos cartões a serem lidos
  if ( ! mfrc522.PICC_IsNewCardPresent()) {
    delay(50);
    return;
  }
  // Escolhe um dos cartões
  if ( ! mfrc522.PICC_ReadCardSerial()) {
    delay(50);
    return;
  }
  // Mostra detalhes do PICC (tag/cartão)
  //Serial.print(F("UID do Cartão:"));
  String rfid = recuperaRFID(mfrc522.uid.uidByte, mfrc522.uid.size);
  //Serial.print(rfid);
  //Serial.println();
  escreveMensagemLCD("UID do cartao", rfid);

  // Substitui espaços por + para envio de parâmetro via HTTP/GET
  //Serial.print(F("UID do Cartão sem espaços:"));
  String rfidGET = rfid;
  rfidGET.replace(" ","+");
  //Serial.print(rfidGET);
  //Serial.println();

  // Envio para o servidor
  postRFID(rfidGET);
}

void mostraMensagemInicial(){
  //Serial.println(F("Escaneando por cartões e escrevendo a UID:"));
  //Serial.println("");
  escreveMensagemLCD("Escaneando RFID", "");
}
```

```
void postRFID(String rfid){
  unsigned long currentMillis = millis();

  if ( currentMillis> watchdog ) {
    previousMillis = currentMillis;
    WiFiClient client;

    if (!client.connect(host, port)) {
      //Serial.println("connection failed");
      return;
    }

    String url = "/smile/class/teste/postRFID.php?";
    url += "rfid=" + rfid;
    if(qualidade == 1) { // HQ
      url += "&camera_entrada=" + String(camera_entrada);
      url += "&camera_saida="   + String(camera_saida);
    }
    else{
      url += "&camera_entrada=" + String(camera_entrada2);
      url += "&camera_saida="   + String(camera_saida2);
    }
    url += "&identificador="  + String(identificador);
    //url += "&ip=";
    //url += WiFi.localIP().toString();
    //Serial.println(String(host)+url);
    // This will send the request to the server
    client.print(String("GET ") + url + " HTTP/1.1\r\n" +
                 "Host: " + host + "\r\n" +
                 "Connection: close\r\n\r\n");
    unsigned long timeout = millis();
    while (client.available() == 0) {
      if (millis() - timeout > 1500) {
        //Serial.println(">>> Client Timeout !");
        client.stop();
        return;
      }
    }

    // Read all the lines of the reply from server and print them to Serial
    while(client.available()){
      String line = client.readStringUntil('\r');
      //Serial.print(line);
    }
  }
}

void postPASS(int tipo){
  unsigned long currentMillis = millis();

  if ( currentMillis> watchdog ) {
    previousMillis = currentMillis;
    WiFiClient client;

    if (!client.connect(host, port)) {
      //Serial.println("connection failed");
      return;
    }

    String url = "/smile/class/teste/postPASS.php?";
    url += "tipo=" + String(tipo);
```

```
    if(qualidade == 1) { // HQ
      url += "&camera_entrada=" + String(camera_entrada);
      url += "&camera_saida="   + String(camera_saida);
    }
    else{
      url += "&camera_entrada=" + String(camera_entrada2);
      url += "&camera_saida="   + String(camera_saida2);
    }
    //url += "&ip=";
    //url += WiFi.localIP().toString();
    //Serial.println(String(host)+url);
    // This will send the request to the server
    client.print(String("GET ") + url + " HTTP/1.1\r\n" +
                 "Host: " + host + "\r\n" +
                 "Connection: close\r\n\r\n");
    unsigned long timeout = millis();
    while (client.available() == 0) {
      if (millis() - timeout > 5000) {
        //Serial.println(">>> Client Timeout !");
        client.stop();
        return;
      }
    }

    // Read all the lines of the reply from server and print them to Serial
    while(client.available()){
      String line = client.readStringUntil('\r');
      //Serial.print(line);
    }
  }
}

void iniciaLCD(){
  Wire.begin(SDA_PIN, SCL_PIN); // Muda pinos padrão do I2C, no NodeMCU os
pinos são D2 (SDA) e D1 (SCL)
  lcd.init();   // Inicializa o display LCD
  lcd.backlight(); // Habilita luz de fundo do display LCD
  escreveMensagemLCD("Iniciando", "");
}

void escreveMensagemLCD(String linha_1, String linha_2){
  lcd.setCursor (linha_1.length(), 0); // Posiciona o cursor na posição 0
da linha 1
  for (int i = linha_1.length(); i < 16; ++i)
    lcd.write(' '); // Limpa linha 1
  lcd.setCursor (linha_2.length(), 1); // Posiciona o cursor na posição 0
da linha 2
  for (int i = linha_2.length(); i < 16; ++i)
    lcd.write(' '); // Limpa linha 2



  lcd.setCursor(0, 0); // Posiciona o cursor na posição 0 da linha 1
  lcd.print(linha_1); // Escreve mensagem na linha 1 do LCD
  lcd.setCursor(0, 1); // Posiciona o cursor na posição 0 da linha 2
  lcd.print(linha_2); // Escreve mensagem na linha 2 do LCD
}

// Callback que avisa a necessidade de salvar as configurações
void saveConfigCallback () {
  //Serial.println("Deve salvar as configurações");
```

```
  shouldSaveConfig = true;
}

// Recupera informações já salvas no sistema de arquivos
void leFS(){
  escreveMensagemLCD("Iniciando", "Lendo FS");
  // Formata FS, para testes
  if(deveResetar)
    SPIFFS.format();

  // Lê configuração do JSON do FS
  //Serial.println("Montando FS");

  if (SPIFFS.begin()) {
    //Serial.println("Sistema de arquivos montado");
    if (SPIFFS.exists("/config.json")) {
      // Arquivo existe, lendo e carregando
      //Serial.println("Lendo arquivo de config");
      File configFile = SPIFFS.open("/config.json", "r");
      if (configFile) {
        //Serial.println("Abriu arquivo de configuração");
        size_t size = configFile.size();
        // Aloca buffer para abrigar conteúdo do arquivo
        std::unique_ptr<char[]> buf(new char[size]);

        configFile.readBytes(buf.get(), size);
        DynamicJsonBuffer jsonBuffer;
        JsonObject& json = jsonBuffer.parseObject(buf.get());
        json.printTo(Serial);
        if (json.success()) {
          escreveMensagemLCD("Iniciando", "Leu FS!");
          //Serial.println("\nJSON parseado");
          strcpy(camera_entrada, json["camera_entrada"]);
          strcpy(camera_saida, json["camera_saida"]);
          strcpy(camera_entrada2, json["camera_entrada2"]);
          strcpy(camera_saida2, json["camera_saida2"]);
          strcpy(identificador, json["identificador"]);
          strcpy(ip_servidor, json["ip_servidor"]);
          strcpy(porta_servidor, json["porta_servidor"]);
          strcpy(qualidadeStr, json["qualidadeStr"]);
        } else {
          //Serial.println("Falhou ao ler JSON de configuração");
          escreveMensagemLCD("Iniciando", "Falha lendo FS");
        }
        configFile.close();
      }
    }
  } else {
    //Serial.println("Falhou ao montar o FS");
    escreveMensagemLCD("Iniciando", "Falha no FS");
  }
  // Termina leitura
}

void printaVariaveisConfiguradas(){
  //Serial.println("IP do servidor:                 "+
String(ip_servidor));
  //Serial.println("Porta do servidor:              "+
String(porta_servidor));
  //Serial.println("ID do dispositivo:              "+
String(identificador));
```

```
  //Serial.println("IP da camera de entrada - HQ:    "+
String(camera_entrada));
  //Serial.println("IP da camera de saida - HQ:      "+
String(camera_saida));
  //Serial.println("IP da camera de entrada - LQ:    "+
String(camera_entrada2));
  //Serial.println("IP da camera de saida - LQ:      "+
String(camera_saida2));
  //Serial.println("Qualidade (1 para HQ 0 para LQ): "+
String(qualidadeStr));
}

void preparaWifiEFS(){
  // Inicialização do WifiManager é local, só é preciso dele uma vez.
  leFS();
  WiFiManager wifiManager;

  // Configuração da notificação de callback de salvamento
  wifiManager.setSaveConfigCallback(saveConfigCallback);

  // Adicionados os parâmetros
  wifiManager.addParameter(&custom_ip_servidor);
  wifiManager.addParameter(&custom_porta_servidor);
  wifiManager.addParameter(&custom_camera_entrada);
  wifiManager.addParameter(&custom_camera_saida);
  wifiManager.addParameter(&custom_camera_entrada2);
  wifiManager.addParameter(&custom_camera_saida2);
  wifiManager.addParameter(&custom_qualidade);
  wifiManager.addParameter(&custom_identificador);

  //reset settings - for testing
  if(deveResetar)
    wifiManager.resetSettings();

  // recupera SSID e senha salvos e tenta conectar
  // Se não conecta então cria um AP com o nome
  // Fica num loop bloqueante até ser conectado

  escreveMensagemLCD("Tentando conectar","");

  if (!wifiManager.autoConnectCustom("SMILE", "projeto2018"))
    escreveMensagemLCD("Falha de conexao","Configure via AP");
  if(!wifiManager.autoConnect("SMILE", "projeto2018")){
    //Serial.println("Falhou em conectar - timeout");
    escreveMensagemLCD("Timeout","Tente novamente");
    delay(3000);
    //reset e tenta de novo, talvez por em sono profundo
    ESP.reset();
    delay(5000);
  }

  //Serial.println("Conectado ao Wi-Fi, salvando parâmetros");
  escreveMensagemLCD("Conectado ao Wi-Fi","Salvando dados");
  // Lê parâmetros atualizados
  strcpy(camera_entrada, custom_camera_entrada.getValue());
  strcpy(camera_saida, custom_camera_saida.getValue());
  strcpy(camera_entrada2, custom_camera_entrada2.getValue());
  strcpy(camera_saida2, custom_camera_saida2.getValue());
  strcpy(identificador, custom_identificador.getValue());
  strcpy(ip_servidor, custom_ip_servidor.getValue());
  strcpy(porta_servidor, custom_porta_servidor.getValue());
```

```
  strcpy(qualidadeStr, custom_qualidade.getValue());

  // Salva os parâmetros novos no FS
  if (shouldSaveConfig) {
    //Serial.println("saving config");
    DynamicJsonBuffer jsonBuffer;
    JsonObject& json = jsonBuffer.createObject();
    json["camera_entrada"] = camera_entrada;
    json["camera_saida"] = camera_saida;
    json["camera_entrada2"] = camera_entrada2;
    json["camera_saida2"] = camera_saida2;
    json["identificador"] = identificador;
    json["ip_servidor"] = ip_servidor;
    json["porta_servidor"] = porta_servidor;
    json["qualidadeStr"] = qualidadeStr;

    File configFile = SPIFFS.open("/config.json", "w");
    if (!configFile) {
      //Serial.println("Falha ao abrir ou escrever arquivo de
configuracao");
    }

    json.printTo(Serial);
    json.printTo(configFile);
    configFile.close();
    // Fim do save
  }

  //Serial.println("IP local");
  //Serial.println(WiFi.localIP());

  // Carrega variáveis para memória
  leFS();
  host = ip_servidor;
  port = String(porta_servidor).toInt();
  qualidade = String(qualidadeStr).toInt();
}

void IN(){
    //Serial.println("Entrada");
    escreveMensagemLCD("Entrada","");
    postPASS(2);
    delay(1000);
}

void OUT(){
    //Serial.println("Saída");
    escreveMensagemLCD("Saida","");
    postPASS(3);
    delay(1000);
}

void detectaEntrada(){
  int in = !digitalRead(inPin);
  int out = !digitalRead(outPin);
  if(in){
    IN();
  }
  else if(out){
    OUT();
  }
```

```
}

bool checaReset(){
  if(analogRead(A0) > 800){
    escreveMensagemLCD("Reset","do dispositivo");
    delay(3000);
    return true;
  }
  else
    return false;
}

void checaBotaoQualidade(){
  if(analogRead(A0) > 800) // deve alterar qualidade da imagem
  {
      qualidade = (qualidade == 1 )? (0) : (1);
      escreveMensagemLCD("Qualidade", (qualidade == 1 )? ("HQ") : ("LQ"));
      delay(2000);
  }
  else
    return;

}

void setup() {
  ////Serial.begin(115200);
  iniciaLCD();
  iniciaMFRC();
  deveResetar = checaReset();
  preparaWifiEFS();
//  conectaWifi();
  iniciaMFRC();
  mostraMensagemInicial();
  printaVariaveisConfiguradas();
  pinMode(inPin, INPUT);
  pinMode(outPin, INPUT);
}

void loop() {
  leCartao();
  detectaEntrada();
  checaBotaoQualidade();
  //delay(1000);
}
```

ANEXO B – Código Python do servidor para acionamento das câmeras

```
#!/usr/bin python3
import cv2
import os
import sys

ip_camera = sys.argv[1]
nome = sys.argv[2]
#print(ip_camera)
os.environ["OPENCV_FFMPEG_CAPTURE_OPTIONS"] = "rtsp_transport;0"

#cap = cv2.VideoCapture(r"rtsp://192.168.0.103:554/onvif1")
#print(ip_camera)
cap = cv2.VideoCapture(ip_camera)

for x in range(1,10):
    ret,img=cap.read()
print(ret)
cv2.imwrite("/var/www/html/smile/pics/"+nome, img)

```

ANEXO C – Código PHP do servidor para atender requisições GET de passagens

```php
<?php
ini_set('display_errors', 1);
ini_set('display_startup_errors', 1);
error_reporting(E_ALL);
//var_dump($_GET);
$tipo = $_GET['tipo'];


// echo($teste);

require_once('../../db/DBClass.php');
$banco = new DBClass();

$t = time();
$ts = date('Y-m-d-H-i-s', $t);
$in = $_GET['camera_entrada'];
$out = $_GET['camera_saida'];
$STRcmdIN = 'python /var/www/html/smile/pics/camera.py '.$in.' '.$ts.'-in.jpg';
$STRcmdOUT = 'python /var/www/html/smile/pics/camera.py '.$out.' '.$ts.'-out.jpg';
$cmdIN = escapeshellcmd($STRcmdIN);
$cmdOUT = escapeshellcmd($STRcmdOUT);
$output1 = shell_exec($cmdIN);
$output2 = shell_exec($cmdOUT);

$queryNova1= "INSERT INTO `passagem`(`tipoPassagem`,`fotoIN`, `fotoOUT`)
VALUES (" . $tipo . ",'/smile/pics/".$ts."-in.jpg','/smile/pics/".$ts."-out.jpg')";

//echo($queryNova1);

$resultado = $banco->query($queryNova1);

$banco->close();
?>
```

ANEXO D – Código PHP do servidor para atender requisições GET de RFID

```php
<?php
ini_set('display_errors', 1);
ini_set('display_startup_errors', 1);
error_reporting(E_ALL);
//var_dump($_GET);
$rfid = $_GET['rfid'];
$tipo = 0;
//$tipo = $_GET['tipo'];


// echo($teste);

require_once('../../db/DBClass.php');
$banco = new DBClass();
$query = "INSERT INTO `registrosRFID`(`campo`) VALUES ('" . $rfid. "')";
//echo($query);

$resultado = $banco->query($query);


$rfid = substr($rfid, 1);
// $queryNova = "START TRANSACTION;
// INSERT INTO `passagem`(`tipoPassagem`) VALUES (" . $tipo . ");
// SET @passagem_id := LAST_INSERT_ID();
// INSERT INTO `rfidPassagem` (`rfid`, `idPassagem`) VALUES ('". $rfid ."', @passagem_id);
// COMMIT;";
$t = time();
$ts = date('Y-m-d-H-i-s', $t);
$in = $_GET['camera_entrada'];
$out = $_GET['camera_saida'];
$STRcmdIN = 'python /var/www/html/smile/pics/camera.py '.$in.' '.$ts.'-in.jpg';
$STRcmdOUT = 'python /var/www/html/smile/pics/camera.py '.$out.' '.$ts.'-out.jpg';
$cmdIN = escapeshellcmd($STRcmdIN);
$cmdOUT = escapeshellcmd($STRcmdOUT);
$output1 = shell_exec($cmdIN);
$output2 = shell_exec($cmdOUT);

$queryNova1= "INSERT INTO `passagem`(`tipoPassagem`,`fotoIN`, `fotoOUT`) VALUES (" . $tipo . ",'/smile/pics/".$ts."-in.jpg','/smile/pics/".$ts."-out.jpg')";

//echo($queryNova1);

$resultado = $banco->query($queryNova1);

$queryNova2 = "INSERT INTO `rfidPassagem` (`rfid`, `idPassagem`) VALUES ('". $rfid ."', ". $banco->lastInsertedID() .");";

```

```
46 //echo($queryNova2);
47
48 $resultado = $banco->query($queryNova2);
49
50 $banco->close();
51 ?>
```

ANEXO E – Script de criação do banco de dados

```
1  -- MySQL Script generated by MySQL Workbench
2  -- Seg 19 Nov 2018 23:24:13 BRST
3  -- Model: New Model    Version: 1.0
4  -- MySQL Workbench Forward Engineering
5
6  SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
7  SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
   FOREIGN_KEY_CHECKS=0;
8  SET @OLD_SQL_MODE=@@SQL_MODE,
   SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';
9
10 -- -----------------------------------------------------
11 -- Schema mydb
12 -- -----------------------------------------------------
13 -- -----------------------------------------------------
14 -- Schema projeto_joao
15 -- -----------------------------------------------------
16
17 -- -----------------------------------------------------
18 -- Schema projeto_joao
19 -- -----------------------------------------------------
20 CREATE SCHEMA IF NOT EXISTS `projeto_joao` DEFAULT CHARACTER SET
   latin1 ;
21 USE `projeto_joao` ;
22
23 -- -----------------------------------------------------
24 -- Table `projeto_joao`.`administrador`
25 -- -----------------------------------------------------
26 CREATE TABLE IF NOT EXISTS `projeto_joao`.`administrador` (
27   `idAdmin` INT(11) NOT NULL AUTO_INCREMENT,
28   `username` VARCHAR(255) NOT NULL,
29   `password` VARCHAR(255) NOT NULL,
30   `email` VARCHAR(255) NOT NULL,
31   PRIMARY KEY (`idAdmin`),
32   UNIQUE INDEX `idAdmin` (`idAdmin` ASC))
33 ENGINE = InnoDB
34 AUTO_INCREMENT = 4
35 DEFAULT CHARACTER SET = latin1;
36
37
38 -- -----------------------------------------------------
39 -- Table `projeto_joao`.`funcionario`
40 -- -----------------------------------------------------
41 CREATE TABLE IF NOT EXISTS `projeto_joao`.`funcionario` (
42   `idFuncionario` INT(11) NOT NULL AUTO_INCREMENT,
43   `nome` VARCHAR(255) NOT NULL,
44   `email` VARCHAR(255) NOT NULL,
45   `nascimento` DATE NOT NULL,
```

```
  `cpf` VARCHAR(15) NOT NULL,
  PRIMARY KEY (`idFuncionario`))
ENGINE = InnoDB
AUTO_INCREMENT = 19
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `projeto_joao`.`passagem`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `projeto_joao`.`passagem` (
  `idPassagem` INT(11) NOT NULL AUTO_INCREMENT,
  `horario` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `tipoPassagem` TINYINT(1) NOT NULL,
  `fotoIN` VARCHAR(255) CHARACTER SET 'latin1' NULL DEFAULT NULL,
  `fotoOUT` VARCHAR(255) CHARACTER SET 'latin1' NULL DEFAULT NULL,
  PRIMARY KEY (`idPassagem`))
ENGINE = InnoDB
AUTO_INCREMENT = 508
DEFAULT CHARACTER SET = latin1
COLLATE = latin1_general_ci;


-- -----------------------------------------------------
-- Table `projeto_joao`.`registrosRFID`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `projeto_joao`.`registrosRFID` (
  `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  `campo` VARCHAR(50) NOT NULL,
  `horario` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 668
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `projeto_joao`.`rfid`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `projeto_joao`.`rfid` (
  `rfid` VARCHAR(12) CHARACTER SET 'latin1' NOT NULL,
  `idFuncionario` INT(11) NOT NULL,
  PRIMARY KEY (`rfid`),
  INDEX `fk_rfid_funcionario_idx` (`idFuncionario` ASC),
  CONSTRAINT `fk_rfid_funcionario`
    FOREIGN KEY (`idFuncionario`)
    REFERENCES `projeto_joao`.`funcionario` (`idFuncionario`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
```

```
DEFAULT CHARACTER SET = latin1
COLLATE = latin1_general_ci;


-- -----------------------------------------------------
-- Table `projeto_joao`.`rfidPassagem`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `projeto_joao`.`rfidPassagem` (
  `rfid` VARCHAR(12) NOT NULL,
  `idPassagem` INT(11) NOT NULL,
  PRIMARY KEY (`idPassagem`),
  INDEX `fk_rfidPassagem_rfid1_idx` (`rfid` ASC),
  CONSTRAINT `fk_rfidPassagem_passagem1`
    FOREIGN KEY (`idPassagem`)
    REFERENCES `projeto_joao`.`passagem` (`idPassagem`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_rfidPassagem_rfid1`
    FOREIGN KEY (`rfid`)
    REFERENCES `projeto_joao`.`rfid` (`rfid`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```