UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA

ANA CAROLINA DOS SANTOS PINTO
LEONARDO PACHECO DE AGUIAR

ARMADILHA FOTOGRÁFICA PARA VIDA SILVESTRE

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA

2020

ANA CAROLINA DOS SANTOS PINTO
LEONARDO PACHECO DE AGUIAR

ARMADILHA FOTOGRÁFICA PARA VIDA SILVESTRE

Trabalho de Conclusão de Curso apresentado ao Departamento Acadêmico de Eletrônica da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do grau de "Bacharel em Engenharia Eletrônica" – Área de Concentração: Engenharia Eletrônica.

Orientador: Professor Dr. Rafael Eleodoro de Góes

CURITIBA

2020

ANA CAROLINA DOS SANTOS PINTO

LEONARDO PACHECO DE AGUIAR

ARMADILHA FOTOGRÁFICA PARA VIDA SILVESTRE

Este Trabalho de Conclusão de Curso de Graduação foi apresentado como requisito parcial para obtenção do título de Engenheiro Eletrônico, do curso de Engenharia Eletrônica do Departamento Acadêmico de Eletrônica (DAELN) outorgado pela Universidade Tecnológica Federal do Paraná (UTFPR). Os alunos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Curitiba, 09 de junho de 2020.

_____

Prof. Dr. Robinson Vida Noronha
Coordenador de Curso
Engenharia Eletrônica

_____

Profª. Drª. Carmen Caroline Rasera
Responsável pelos Trabalhos de Conclusão de Curso
de Engenharia Eletrônica do DAELN

BANCA EXAMINADORA

_____

Prof. Doutor Rafael Eleodoro de Góes
Universidade Tecnológica Federal do Paraná
Orientador

_____

Prof. Doutora Fabiana Pottker
Universidade Tecnológica Federal do Paraná

_____

Prof. Doutor Guilherme de Santi Peron
Universidade Tecnológica Federal do Paraná

A folha de aprovação assinada encontra-se na Coordenação do Curso de Engenharia Eletrônica.

AGRADECIMENTOS

Concluir uma graduação requer muita persistência, dedicação e apoio das pessoas ao nosso redor. Por isso agradeço. Aos meus pais, por serem pacientes e pelo apoio e suporte que sempre me deram. Aos meus amigos, pelas conversas e acreditarem em mim em vezes que eu não acreditei. Ao meu namorado, Tomás, por estar disposto a aprender coisas novas e assim também me incentivar a aprender mais. Aos colegas de graduação, pelos estudos em conjunto e apoio principalmente nos fins de semestre. Ao Leonardo, amigo de graduação e dupla de TCC, pela motivação, já que, em equipe é sempre possível que quando um desanime o outro possa apoiar. Ao nosso orientador Prof. Dr. Rafael Eleodoro de Góes, por sempre nos atender e nos guiar com entusiasmo rumo a um melhor desenvolvimento do trabalho. Quero agradecer também à Universidade Tecnológica Federal do Paraná e todos os professores que contribuíram com meu aprendizado. A todos que me apoiaram, meu mais sincero obrigada!

Ana Carolina dos Santos Pinto

A conclusão de um curso de Engenharia tem inúmeros percalços, noites sem dormir; o ”frio na barriga”em uma apresentação ou prova; ou momentos de grandes dificuldades. Porém, há pessoas que me ajudaram imensamente neste processo, com conselhos, apoio psicológico e amor. Por isso sou muito grato à elas. Aos meus amigos de graduação que foram capazes de animar o ambiente e ao mesmo tempo ter a seriedade nos estudos, me ajudando a crescer como engenheiro e ser humano. À Universidade Tecnológica Federal do Paraná e aos meus professores que, de formas singulares, me ajudaram a construir uma base sólida de conceitos técnicos, éticos e críticos. Aos meus pais e irmão por terem me dado todo suporte necessário ao longo dessa caminhada, pelas broncas nos momentos certos e principalmente por todo amor. Eu amo vocês! À minha namorada, Larissa, à quem tenho imensa admiração por toda sua força e trajetória. Ao longo do curso, ela me ajudou com muito amor, carinho, conselhos e incentivos. À Ana, parceira e amiga de graduação, que apesar de nossas divergências de ideias, foi quem aceitou entrar de alma e coração nesse grande desafio final. Ao meu orientador Prof. Dr. Rafael Eleodoro Goés que, mesmo sem nos conhecer a priori, aceitou o convite para nos orientar. Durante o percurso, o Prof. Rafael sempre foi capaz de impulsionar o nosso projeto de forma animada e motivadora ao passo que ponderava os ricos e desafios do mesmo.

Leonardo Pacheco de Aguiar.

RESUMO

DOS SANTOS PINTO, Ana Carolina; PACHECO DE AGUIAR, Leonardo. ARMADILHA FOTOGRÁFICA PARA VIDA SILVESTRE. 65 f. Trabalho de Conclusão de Curso – Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná. Curitiba, 2020.

O projeto teve como intuito a elaboração de um dispositivo de baixo consumo de energia com a capacidade de documentar, por meio de fotos, a vida selvagem animal e vegetal. Como o equipamento ficará em locais distantes por longos períodos, foi necessária a utilização de uma conexão remota entre ele e seu usuário para possibilitar que o equipamento seja capaz de enviar seus dados a um portal, no qual as imagens estarão disponíveis para acesso do usuário. Portanto, o dispositivo tem conexão com a internet para a transferência de arquivos, caracterizando-o como um equipamento IoT(*Internet of Things*). Para construir tal equipamento foi utilizado um modem GSM(*Global System for Mobile communications*) o qual possibilitou a conexão com a rede de telefonia móvel, uma câmera a qual efetuou a captura de imagens e um microcontrolador que integrou e gerenciou o sistema como um todo, seja na captação dos dados, seja na transmissão dos mesmos.

Palavras-chave: Dispositivo IoT, modem GSM, vida selvagem, registro fotográfico.

ABSTRACT

DOS SANTOS PINTO, Ana Carolina; PACHECO DE AGUIAR, Leonardo. WILDLIFE CAMERA TRAP. 65 f. Trabalho de Conclusão de Curso – Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná. Curitiba, 2020.

The goal of this project was to build a low power consumption device that could document, by photographs, wildlife species and habitats. As the camera trap was designed to remain in remote locations for long periods of time, a connection to a remote network was essential to enable the device to send all the collected data to a remote server. In order to build this equipment a GSM (Global System for Mobile communications) modem was used along with a camera, providing connection to a mobile network and a manner of taking photographs. Besides those resources, a microcontroller was also used to integrate the parts of the system. The pictures and data collected were sent to a website where the user could see and manage the received data. Thus, as the camera trap has an internet connection it can be defined as an IoT (Internet of Things) device.

Keywords: IoT device, GSM modem, wildlife, photograph.

LISTA DE FIGURAS

LISTA DE SIGLAS

GSM	Global System for Mobile (Sistema Global Móvel)
GPRS	General Packet Radio Service (Serviço de Rádio de Pacote Geral)
IoT	Internet of Things (Internet das Coisas)
I2C	Inter-Integrated Circuit(Circuito Inter-Integrado)
SCCB	Serial Camera Control Bus (Barramento Serial de Controle da Câmera)
SPI	Serial Peripheral Interface
GPIO	General Purpose Input Output
CSI	Camera Serial Interface (Interface Serial da Câmera)
SIM	Subscriber Identification Module
PIR	Passive Infrared Sensor (Sensor Passivo Infravermelho)
LDR	Light Dependent Resistor (Fotoresistor)
SO	Sistema Operacional
NOOBS	New Out Of the Box Software
API	Application Program Interface (Interface de Programação de Aplicações)
EXIF	Exchangeable Image File Format
DMS	Degrees, Minutes and Seconds (Graus, Minutos e Segundos)
PPP	Point-to-Point Protocol (Protocolo Ponto a Ponto)
APN	Access Point Name (Ponto de Acesso)
HDMI	High-Definition Multimedia Interface
MIME	Multipurpose Internet Mail Extensions (Extensões Multi função para Mensagens de Internet)
LED	Light Emitting Diode (Diodo Emissor de Luz)

SUMÁRIO

1 INTRODUÇÃO

O estudo do comportamento dos animais em seus habitats naturais muitas vezes exige grandes esforços para deslocamentos de pesquisadores e equipamentos. Estes habitats podem ser das mais diversas formas e cada um apresenta suas peculiaridades. Portanto cada local apresenta suas dificuldades referentes a deslocamentos, necessidade de acampamentos, entre outros (MARQUES, 2004).

Após enfrentado o percurso anteriormente mencionado, é chegada a hora de montar os equipamentos de monitoramento. Uma vez instalados, os pesquisadores tendem a se afastar dos locais onde se encontram esses equipamentos a fim de observar o comportamento natural dos animais da região. Ainda, segundo Marques (2004), a utilização de armadilhas fotográficas é um método muito interessante, pois, trata-se de uma metodologia não invasiva de captura informações, assim, funcionando como se os animais estivessem sendo analisados pelos olhos dos pesquisadores. Ou seja, não é necessário capturar um dado animal silvestre e o levar para algum cativeiro para, então, estudá-lo. Fato este, segundo Marques(2004), que inclusive torna a utilização de armadilhas fotográficas recomendada para o estudo de espécies ameaçadas de extinção, pois se evita interações desnecessárias e estressantes para a espécie em questão.

Tendo em vista esses aspectos citados, o método de estudo utilizando armadilhas fotográficas é benéfico para pesquisadores e animais. No entanto, ao longo do processo, ainda existe uma etapa que pode gerar uma interação desnecessária entre os animais e os cientistas, a qual pode ocorrer durante a coleta dos dados pelo pesquisador, pois é necessário que o pesquisador vá até a armadilha fotográfica para coletar estes dados de forma presencial. Analisando esse problema enfrentado pelos estudiosos da vida silvestre propõe-se construir um dispositivo IoT capaz de fazer a aquisição desses dados e enviá-los de forma remota aos pesquisadores. Em outras palavras, será construída uma armadilha fotográfica capaz capturar esses dados da vida silvestre e que seja capaz de enviar os mesmos para um servidor ou serviço na Internet. De tal forma, só se fará necessário retornar ao local onde o equipamento foi instalado quando o mesmo estiver sem energia, for desejada uma mudança de local ou na desinstalação do mesmo.

1.1 MOTIVAÇÃO

Considerando que podem ser barreiras para os pesquisadores o deslocamento regular aos locais onde os animais se encontram, bem como uma possível interação humana danosa para algumas espécies, como por exemplo espécies ameaçadas de extinção. Pretende-se diminuir esse fator desmotivacional e as vezes até proibitivo no estudo da vida silvestre por intermédio de um dispositivo autônomo que seja capaz de facilitar este processo. Não obstante, atualmente o tráfego de informação é muito dinâmico por causa da Internet, portanto, é interessante aplicá-lo para auxiliar e facilitar os estudos já existentes sobre o monitoramento da vida silvestre.

1.2 OBJETIVOS

1.2.1 OBJETIVO GERAL

O objetivo geral do projeto é desenvolver um dispositivo IoT capaz de adquirir fotografias da vida silvestre sem interagir com a mesma. Além disso, este dispositivo deve enviar os dados coletados de forma remota aos pesquisadores por intermédio da Internet, para que os pesquisadores não interajam danosamente com as espécies em estudo.

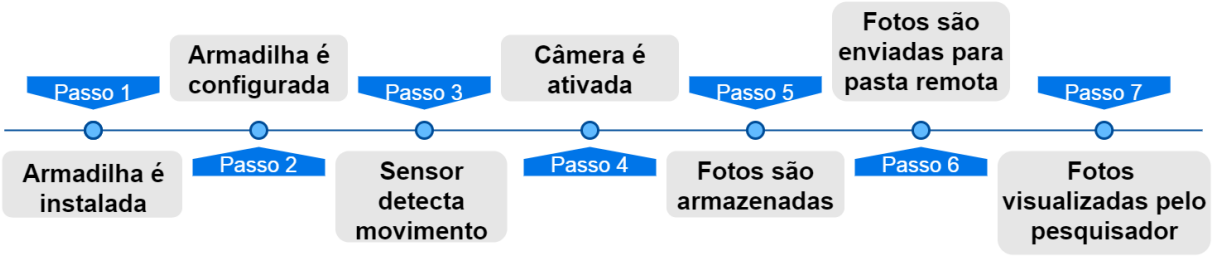
1.2.2 OBJETIVOS ESPECÍFICOS

- Implementar um método de aquisição de imagens para que se possa obter registros da vida silvestre;
- Estabelecer uma forma de armazenar as imagens em memórias não voláteis, mesmo em regimes de alimentação intermitente;
- Permitir que o usuário possa configurar o dispositivo de forma remota;
- Enviar as imagens da vida silvestre via módulo GSM/ GPRS;
- Estabelecer um método de detecção de presença da vida silvestre para o momento correto da aquisição de imagens;
- Utilizar um sistema de alimentação independente de rede elétrica capaz de fornecer energia para o sistema;
- Entregar as imagens ao usuário final com informações a respeito de qual dispositivo a adquiriu, bem como hora, data e localização.

2 CONCEPÇÃO DO PROJETO

Nesta seção são abordados os conceitos do projeto de um aspecto global. Desta forma, tem-se como intuito o desenvolvimento de um dispositivo IoT capaz de capturar imagens da vida silvestre em seus ambientes naturais e enviá-las de forma remota aos pesquisadores. Como afirma Marques (2004), dispositivos que capturaram imagens de animais são chamados de armadilhas fotográficas, então o dispositivo a ser desenvolvido pode ser nomeado de armadilha fotográfica IoT, pois, além de capturar os dados, é capaz de transferi-los para a Internet. A conectividade com a Internet sendo um diferencial quando comparada com outros dispositivos com mesmo propósito disponíveis no mercado, pois a maioria das armadilhas fotográficas encontradas durante a fase de pesquisa não apresentavam tal ferramenta (GAMECAMERAWORLD, 2019). A seguir, na Figura 1 é possível verificar o ciclo de uso do dispositivo, o qual consiste desde sua instalação em um local determinado até o recebimento dos dados desejados pelos pesquisadores no fim do ciclo.

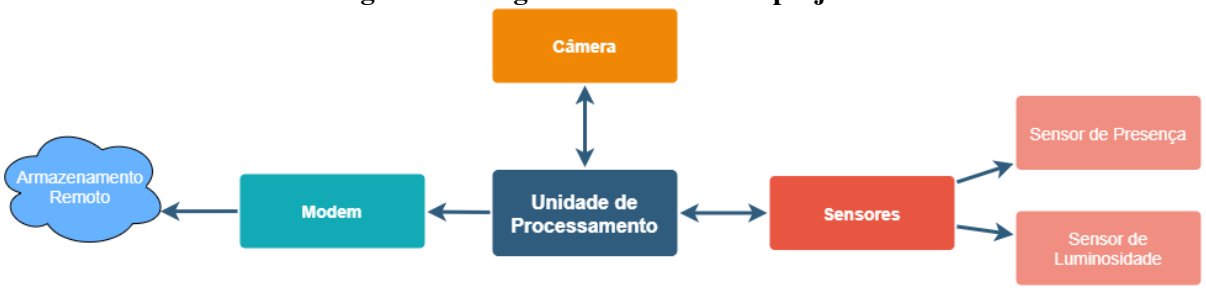
Figura 1 – Sequência de passos funcionais do projeto.



Fonte: Própria (2020)

Isto posto, a Figura 2, a seguir, ilustra quais são os módulos necessários para execução do objetivo global desse dispositivo.

Figura 2 – Diagrama conceitual do projeto



Fonte: Própria (2020)

Assim, sendo esses os módulos necessários para confecção do projeto, são detalhadas nas próximas seções as características necessárias do funcionamento de cada um para o projeto.

2.1 CÂMERA

Este módulo do dispositivo tem como intuito a aquisição de imagens de animais silvestres, para garantir a relevância destas imagens elas precisam ter um certo grau de qualidade. Para atingir tal grau de qualidade objetiva-se ter uma boa visualização do animal silvestre para que seja possível sua identificação dentro das necessidades do usuário e das limitações do sistema. A utilização da câmera é para que haja a miníma interferência possível com vida silvestre durante a coleta de dados, além disso, é efetuada a marcação temporal e espacial durante este processo.

2.2 SENSOR DE PRESENÇA

Para a captura das imagens desejadas se faz necessário dispor de uma ferramenta para a detecção desses alvos. No entanto, vale salientar novamente que se fazem necessárias algumas medidas que consistam na não interferência dos dispositivos de monitoramento no cotidiano dos animais. Assim, as fotos podem retratar um contexto real de interação da vida silvestre com a natureza. Portanto, o detector de presença a ser escolhido escolhido deve gerar pouca interação com o ambiente, a fim de evitar que os animais sejam afugentados pelo mesmo.

2.3 SENSOR DE LUMINOSIDADE

Um aspecto importante para que uma foto tenha uma boa qualidade de imagem é a luz, portanto, é relevante obter informações sobre luminosidade no ambiente no momento da captura. Para cumprir tal objetivo, se faz necessário utilizar um sensor que consiga informar

qual é quantidade de luz no ambiente. A partir dessa informação é possível tomar algumas decisões quanto à utilização de funções da câmera, seja no uso do modo noturno, ajuste de ganho ou até mesmo o não registro fotográfico por falta de condições adequadas.

2.4 MODEM

Como foi salientado nas seções anteriores em alguns cenários o deslocamento para recuperar as imagens na armadilha fotográfica é custoso. Buscando a diminuição do impacto deste inconveniente para o proprietário do dispositivo, se faz necessário o transporte dessas informações de forma remota. Para exercer tal função de comunicação remota se faz necessária a utilização um modem que seja capaz de transferir os dados coletados para alguma espécie de armazenamento externo. Este módulo do projeto é um dos pilares que sustentam a nomenclatura de dispositivo IoT, pois ele faz a ponte entre as informações contidas no equipamento e a rede mundial de computadores.

2.5 ARMAZENAMENTO REMOTO

Reforçando a linha de ser um equipamento IoT, além do envio, é efetuado o armazenamento das imagens de forma remota. Pois, desta forma além de poder acessar os dados da armadilha fotográfica sem se deslocar até a mesma, estes dados podem ser acessados por diversos dispositivos que sejam capazes de se conectar à rede de computadores. Assim sendo, nessa etapa do projeto é necessário estudar alguma plataforma capaz de realizar esse armazenamento em um servidor remoto, bem como garantir que estes dados tenham uma camada de segurança de forma que apenas os usuários detentores do dispositivo tenham acesso.

2.6 UNIDADE DE PROCESSAMENTO

Esta parte do dispositivo é de fundamental importância pois, como ilustrado na Figura 2, é o responsável por toda a parte de processamento de dados, ao mesmo tempo que funciona como nó central de controle e comunicação entre as partes da armadilha fotográfica. Este nó central tem como uma de suas tarefas a recepção de dados do ambiente, como a presença ou não de animais, bem como o nível de luminosidade, a partir dos quais é capaz de tomar decisões de grande influência em outras partes do projeto. Por exemplo, em um cenário sem luminosidade a captura da imagem estaria comprometida e o acionamento da câmera seria apenas um gasto de energia desnecessário. Como objetiva-se a não interferência humana no dispositivo, quanto

menor o gasto de energia quando não houver necessidade, maior a autonomia do dispositivo.

Para que o nó de processamento seja capaz de fazer a intercomunicação entre os módulos se faz necessária a capacidade de comunicação com cada módulo individualmente. Em outras palavras, é preciso utilizar alguns protocolos de comunicação para cada parte do dispositivo. Por exemplo, caso um módulo necessite de um método de compressão de imagem, o nó central deve ser capaz de realizar esta tarefa. Além dessa tarefa exemplificada, existem momentos em que o nó central tem mais de uma tarefa para realizar, portanto deve-se levar essa questão em conta na escolha da unidade de processamento.

3 DESENVOLVIMENTO DA ARMADILHA FOTOGRÁFICA

Inicialmente, durante a definição do projeto a placa de desenvolvimento escolhida para a execução do sistema da armadilha fotográfica havia sido a **Tiva C (TM4C1294)** da Texas Instruments em conjunto com o ambiente de desenvolvimento **IAR**. Esta placa e ambiente de desenvolvimento foram selecionados pela familiaridade de utilização dos mesmos, já que haviam sido utilizados na disciplina de Sistemas Embarcados durante a graduação.

Durante este mesmo período, o módulo OV7670 havia sido selecionado com a finalidade de capturar imagens. Para efetuar a comunicação deste módulo com a **Tiva C** seria necessário o desenvolvimento de um driver, o qual efetuaria o interfaceamento entre o protocolo de circuito inter-integrado (I2C), da placa de desenvolvimento, e o barramento serial de controle da câmera (SCCB). Este driver serviria, ainda, para efetuar a correta leitura dos dados oriundos da câmera, além disso, seria necessária a implementação de um algorítimo que convertesse os bits em um formato de imagem padrão.

Ao longo do desenvolvimento do projeto com estes itens, foram encontrados empecilhos que favoreceram a escolha de outra placa de desenvolvimento e câmera. Um ponto de grande impacto foi a questão do armazenamento das fotos na armadilha fotográfica. É importante salientar que, a formatação utilizada nas imagens coletadas pelo dispositivo foi JPEG, o que produziu fotos com um tamanho médio de 450 KiB. Vale ressaltar que ao longo do texto foi utilizado a notação binária, mais assertiva para se referir ao tamanho de dados digitais, como o KiB e o MiB, que são respectivamente dois elevado a décima potência e dois elevado a vigésima potência. Ao invés da base decimal que era tradicionalmente utilizada, como por exemplo KB e MB que são, respectivamente dez elevado a terceira potência e dez elevado a sexta potência (IBM, 2019). Como já era sabido que a memória da **Tiva C** apresenta uma capacidade pequena para fins de armazenamento de imagens deste porte, pois, ela possui apenas 1 MiB de memória *Flash*, deste modo, seria necessária a implementação de um sistema de memória adicional (TEXAS INSTRUMENTS, 2014). No intuito de solucionar tal problema foi decidido usar um cartão de memória extra para efetuar este armazenamento, o módulo de cartão de memória pesquisado utilizava o protocolo de comunicação de Interface Periférica

Serial (SPI). Porém, para realizar o armazenamento das imagens em um formato de arquivo seria necessário a implementação de um sistema de arquivos, o qual precisaria ser desenvolvido, fato que atribuiria mais uma etapa ao projeto.

Assim, considerando os impactos negativos que o tempo extra de desenvolvimento pode ter sobre um projeto, como, por exemplo, elevando seus custos, e levando em conta que novas tecnologias poderiam ser utilizadas para um desenvolvimento mais assertivo, ficou decidido que era o momento de buscar uma nova plataforma para o desenvolvimento do projeto.

Durante esta busca foram encontrados alguns microcontroladores que atendiam a certas necessidades do projeto, entre eles se destacaram a **ESP32** e a **Raspberry Pi**. Inicialmente o projeto seria desenvolvido utilizando a linguagem de programação C, pois essa linguagem já era conhecida, o que facilitaria o desenvolvimento do projeto. Então, para manter este aspecto do projeto presente, mesmo não sendo um fator essencial, foi escolhida a utilização da **ESP32-CAM** que já possuía uma câmera integrada e um ***slot*** para cartão de memória. Estes dois componentes foram ferramentas relevantes para a escolha deste microcontrolador, sendo algumas das partes de maior importância para o projeto.

Primeiramente foram feitos testes programando a **ESP32-CAM** através da IDE do **Arduino**, estes primeiros testes tiveram bons resultados e portanto foi iniciado o processo de estudo para o desenvolvimento com a **ESP32-CAM** utilizando a **IDE Eclipse** por sugestão da empresa desenvolvedora da placa. Nesta etapa foi estudada a documentação para utilização da placa com o Eclipse, porém foi verificada uma escassez de detalhes na documentação quando se tratava da utilização da placa com o ambiente de desenvolvimento em questão.

Visto isso, deu-se início às pesquisas sobre a outra opção de placa de desenvolvimento, a **Raspberry Pi**, para verificar a viabilidade de uma última mudança na unidade de processamento do projeto. Nesta etapa, foi verificado que esta troca seria favorável ao desenvolvimento do projeto, pois traria benefícios como: facilidade de interfaceamento com sensores, possibilidade de programação em uma linguagem de alto nível e uma documentação mais segura e abrangente. Embora ainda fosse possível a utilização da linguagem C para programação da placa, a utilização do **Python**, linguagem selecionada para o projeto, possibilitou a utilização de diversas bibliotecas já existentes. Essa escolha como plataforma base para do projeto, permitiu, assim, o redirecionamento de esforços para desenvolvimento das funcionalidades alvo do dispositivo.

Levando em conta o consumo energético, foi selecionada a **Raspberry Pi Zero W**, pois ela possui o segundo menor consumo quando comparada a outras placas da família **Raspberry Pi**, como pode ser visto na Tabela 1, a seguir. A placa selecionada perde o

posto de menor consumo para a sua versão mais simplificada que é a **Raspberry Pi Zero**, a qual não possui módulos ***Bluetooth*** nem ***Wi-Fi***, este último foi um recurso muito utilizado no decorrer do desenvolvimento do projeto. Apesar de ambos módulos impactarem no consumo, ao serem desligados o consumo dos dois modelos **Pi Zero** se equiparam. Ainda, a **Raspberry Pi** selecionada atende a todas as necessidades do projeto: interfaceamento com periféricos de aquisição de imagem, pinos digitais de entrada e saída, interrupções configuráveis, interface de comunicação serial, *softwares* e bibliotecas de apoio. Desta forma, havendo a possibilidade de utilização da **Raspberry Pi** em conjunto com um sistema operacional, foi decidida a instalação do ***Raspbian*** e para tal foi utilizado um cartão de memória de 16 GiB. Desta memória disponível metade fica para o sistema e os 8 GiB restantes para o desenvolvimento da armadilha, garantindo, assim, espaço suficiente para o armazenamento de até dezoito mil fotos capturadas.

Tabela 1 – Consumo típico dos modelos disponíveis de Raspberry Pi

Modelo	Consumo típico
Raspberry Pi Modelo A	200 mA
Raspberry Pi Modelo B	500 mA
Raspberry Pi Modelo A+	180 mA
Raspberry Pi Modelo B+	330 mA
Raspberry Pi 2 Modelo B	350 mA
Raspberry Pi 3 Modelo B	400 mA
Raspberry Pi 3 Modelo A+	350 mA
Raspberry Pi 3 Modelo B+	500 mA
Raspberry Pi 4 Modelo B	600 mA
Raspberry Pi W Zero	150 mA
Raspberry Pi Zero	100 mA

Fonte: (FOUNDATION, 2017)

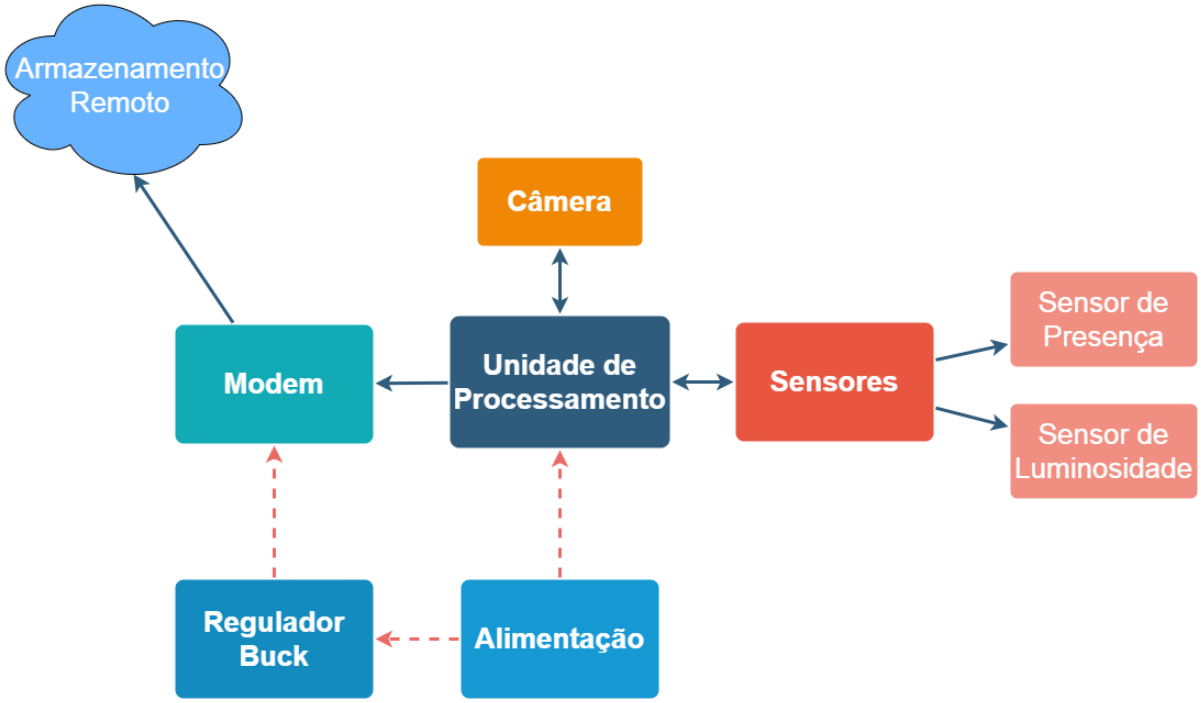
3.1 INTERCONEXÃO DOS MÓDULOS

Conforme o diagrama apresentado na Figura 3, a unidade de processamento é alimentada diretamente por um *power bank* fornecendo 5V. O Modem GSM, devido às suas especificações de tensão e corrente, é alimentado pela mesma bateria, porém utilizando um regulador de tensão do tipo *Buck* com tensão de saída fixada em 4,2V. Ainda de acordo com a Figura 3 as linhas contínuas em azul representam linhas de dados e as linhas tracejadas e em vermelho conexões elétricas.

Para a alimentação do sistema, foi escolhida uma bateria recarregável ligada diretamente a **Raspberry Pi Zero W** e ligada por meio de um regulador de tensão ao modem. Os sensores são alimentados diretamente pela **Raspberry Pi**, sendo ligados aos seus pinos

GPIO os sensores de presença e luminosidade e a uma entrada CSI a câmera. Na Figura 3 é possível observar um diagrama dos módulos escolhidos para o projeto.

Figura 3 – Diagrama dos módulos escolhidos



Fonte: Própria (2020)

Sendo esses os módulos selecionados para realização do projeto, são apresentadas nas próximas seções características do funcionamento de cada um dos dispositivos e o que levou a essas escolhas.

3.2 RASPBERRY PI ZERO W

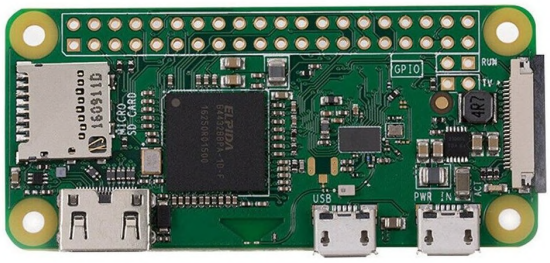
Para o nó de processamento foi escolhido a **Raspberry Pi Zero W**. Esta funciona como um computador, podendo ser utilizada em conjunto com o ***Rapbian***, um sistema operacional baseado no ***Debian***, o qual é uma distribuição ***Linux*** (FOUNDATION, 2017). Assim, permitindo uma programação em alto nível e a utilização de todo o suporte de bibliotecas e pacotes disponibilizados para a **Raspberry Pi**.

Foi escolhida a **Raspberry Pi Zero W** por seu menor tamanho e também por seu reduzido consumo energético, quando comparada com os outros modelos de **Raspberry Pi**.

Este modelo, representado na Figura 4, conta com as seguintes características:

- 512 MB RAM
- Processador BCM2835 1 GHz ARM 11 de núcleo único
- Conector CSI (Camera Serial Interface)
- Slot para memória externa não volátil (Cartão MicroSD)
- 40 pinos de uso geral (GPIO)

Figura 4 – Raspberry Pi W Zero.



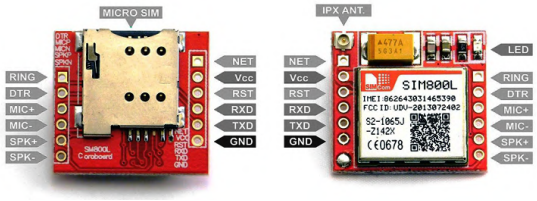
Fonte: (FOUNDATION, 2017)

3.3 MODEM GSM SIM800L

O SIM800L é um módulo GSM/GPRS que pode ser utilizado em vários países diferentes, pois permite efetuar conexão à rede de telefonia móvel em quatro diferentes frequências se adaptando a região, sendo elas: 850 MHz, 900 MHz, 1800 MHz ou 1900 MHz. A taxa de transmissão de dados características da rede GPRS, també conhecida como rede 2G, opera em torno de 26 a 40 KiB por segundo (SICHONANY et al., 2012). Ele também possui uma entrada para cartão SIM, o qual é necessário para se comunicar com a rede de telefonia móvel, semelhante a um celular. O módulo possui também pinos para interfaceamento com outros dispositivos por meio de comunicação serial e uma entrada para antena externa, no projeto foi utilizada uma antena helicoidal. Como sua tensão ideal de alimentação é entre 3,7V e 4,2V foi necessária a utilização de um conversor *Buck* para efetuar sua ligação a alimentação do sistema.

Ainda em relação a serial do modem, ela possui dois modos de operação: um no qual é feito o tráfego de comandos AT e no outro o canal fica disponível para troca de dados. Os comandos ATs podem ser usados para várias tarefas do modem, que podem verificar, por exemplo: o nível de sinal da rede, verificar as informações sobre o cartão SIM, em qual rede o SIM tá conectado, entre outros. A seguir, na Figura 5, é apresentado o modem a ser utilizado.

Figura 5 – Modem SIM800L

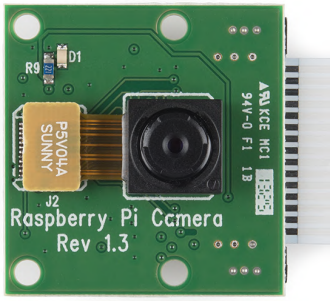


Fonte: (SCHEMATICS, 2019)

3.4 CÂMERA

Para captura de imagens foi selecionado o sensor CMOS OV5647, câmera oficial da ***Raspberry Foundation***. Esta câmera possui resolução máxima de 5 Mega *pixels*, a qual fornece imagens de 2592 *pixels* por 1944 *pixels*, um total de 5.038.848 *pixels* (5 M*pixels*). Considerando o formato RGB cada pixel é composto por 3 bytes, a quantidade total de bytes de uma imagem em máxima resolução é de 14,41 MiB, utilizando o algoritmo de compressão JPG foi possível comprimir a imagem em até 2 MiB. No entanto, considerando algumas limitações do modem selecionado, como a taxa de transmissão de dados, foi necessário reduzir a resolução da câmera para 1280 por 720 *pixels*, gerando um total de 921.600 *pixels* (0,9 Mega *pixels*), desta forma o tamanho máximo da imagem seria de 2,7 MiB, no entanto, com a formatação JPG o tamanho médio das imagens ficou em 450 KiB, tendo uma compressão de cerca de seis vezes. Ainda, a câmera é conectada a placa por meio de seu conector de Interface Serial da Câmera (CSI). Sua utilização em conjunto com o ***Raspbian***, quando comparado com os outros modelos de câmeras, é facilitada por já ser configurada nativamente no sistema sendo necessária apenas sua habilitação. A seguir, na Figura 6, exemplifica-se a câmera em questão.

Figura 6 – Camera OV5647



Fonte: (OMNIVISION, 2018)

3.5 SENSOR DE PRESENÇA (PIR)

Para efetuar a detecção de movimentação em frente a armadilha o sensor selecionado foi o HC-SR501, um tipo de sensor PIR, ou sensor passivo infravermelho.

Este sensor foi selecionado para que a captura das imagens ocorresse de forma a causar a menor interferência possível na fauna local. Como é um sensor passivo ele efetua uma leitura da intensidade de luz infravermelha incidente sobre ele sem a necessidade de efetuar um tipo de emissão luminosa ou sonora. Uma grande vantagem quando comparado com um sensor ultrassom, por exemplo, que para efetuar uma detecção precisa efetuar o envio de ondas ultrassônicas, podendo afastar algumas espécies que conseguem detectar este tipo de onda.

O sensor passivo infravermelho é um sensor piroelétrico, o qual efetua detecção de calor, uma vez que, este é emitido na forma de ondas infravermelhas. Todos os objetos emitem radiação infravermelha, dependendo de sua temperatura, portanto, quando um animal ou uma pessoa passa em frente ao sensor, esta radiação infravermelha é detectada e o sensor, então, ativado.

Há a possibilidade de selecionar entre dois modos distintos de operação, normalmente estes modos podem ser selecionados através de pinos que podem ser conectados. Estes modos são o H *repeatable mode* e o L *non-repeatable mode*. Nos dois casos quando um objeto emissor de infravermelho (animais e seres humanos), entra dentro do campo de captura do sensor este envia um sinal lógico alto (3,3 V) para o microcontrolador. A diferença entre os modos de operação está no sinal enviado em caso de permanência do objeto em frente ao sensor.

No modo H, o sinal na saída do sensor permanece em nível lógico alto por um tempo de atraso que pode ser selecionado manualmente ao movimentar o potenciômetro presente na

parte posterior do sensor, em seguida, ele volta a enviar um sinal lógico baixo e caso continue a haver ativação do sensor ele volta a enviar um sinal lógico alto. No modo L, o sinal na saída do sensor permanece em nível lógico alto enquanto o objeto emissor de infravermelho permanecer no campo de captura do sensor, passando para o nível lógico baixo apenas após o objeto sair do campo de captura mais o tempo selecionado manualmente através do potenciômetro.

O modo de operação escolhido para utilização do sensor PIR foi o H, ou *repeatable mode*. Este modo de operação foi utilizado, pois, assim é possível capturar fotos enquanto o objeto estiver no alcance do sensor, permitindo múltiplas fotos de um mesmo animal.

Dados obtidos da folha de dados:

- Distância de identificação de presença: até 7 metros
- Ângulo de identificação de presença: até 100 graus

Figura 7 – Sensor PIR

Fonte: (ROBOTICA, 2019)

3.6 SENSOR DE LUMINOSIDADE

O LDR é um resistor foto-sensível, assim, sua resistência varia dependendo da intensidade luminosa incidente sobre o sensor.

Este sensor é utilizado para determinar a intensidade luminosa incidente sobre a armadilha. Para isso foram determinados dois valores de limite para a tensão medida nos pinos ligados ao sensor de luminosidade, um dos valores determina se uma foto deve ser tirada no modo normal de funcionamento da câmera ou no modo noturno, o outro determina se uma foto deve ser tirada no modo noturno ou se não deve ser capturada.

Figura 8 – Fotoresistor

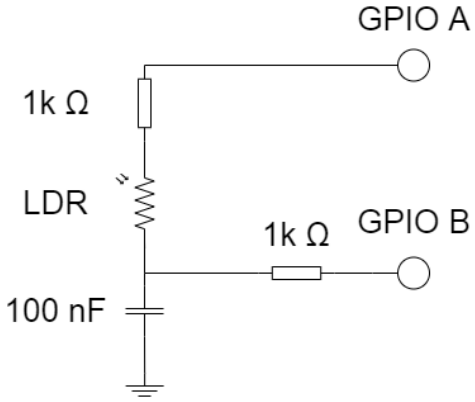
Fonte: (FILIPEFLOP, 2020)

3.6.1 AQUISIÇÃO DE DADOS ANALÓGICOS

A **Raspberry Pi Zero W** possui 40 pinos GPIO, porém não apresenta pinos que efetuem leitura de entrada analógica, assim, para efetuar a coleta de dados analógicos do LDR, foi feita a utilização de um circuito com capacitores, que pode ser visto na Figura 9, em conjunto com um código em **python** (WILCHER, 2016) para efetuar a leitura dos pinos digitais ligados ao circuito. Desta forma, efetuando a carga e descarga do capacitor e contando seu tempo de carregamento é possível estimar o valor analógico da carga.

Neste código é utilizada a técnica de resposta ao degrau, portanto, não é necessária a utilização de um conversor analógico Digital (ADC) para efetuar a leitura dos dados analógicos pela **Raspberry Pi** (WILCHER, 2016). Na imagem a seguir, é apresentado o circuito utilizado para aquisição dos dados do LDR, baseado no circuito originalmente disponibilizado.

Figura 9 – Diagrama do circuito de aquisição de dados analógicos



Fonte: Própria (2020)

3.7 *POWER BANK*

Para alimentação do sistema foi utilizado um *power bank* com as seguintes características:

- 10000 mAh
- Entrada 5,0V e 2,4A
- Saída 5,0V e 2,4A

Foi selecionada a utilização de um *power bank* como forma de energização do sistema por conta de sua facilidade de recarga, grande disponibilidade no mercado e capacidade de fornecimento de energia por consideráveis períodos de tempo.

3.7.1 ALIMENTAÇÃO DO MODEM GSM: LM2596

Como para a alimentação do sistema como um todo foi utilizado um *power bank* fornecendo 5V de tensão e até 2,4A de corrente, foi necessária a utilização de um conversor de tensão para tornar a tensão de alimentação apropriada para o funcionamento do módulo GSM SIM800L, que opera entre 3,7V e 4,2V.

Para isso foi utilizado o conversor de tensão LM2596, um regulador abaixador de tensão ou regulador *Buck*. Utilizando este conversor a tensão de alimentação do módulo GSM foi regulada para 4,2V. Além disso, o regulador é capaz de suportar os 2A de pico de corrente requisitados pelo SIM800L em momentos de transmissão de dados e conexão com a rede de telefonia móvel, tal consumo de corrente é descrito na folha de dados do SIM800L, na Figura 10 é possível observar o regulador em questão.

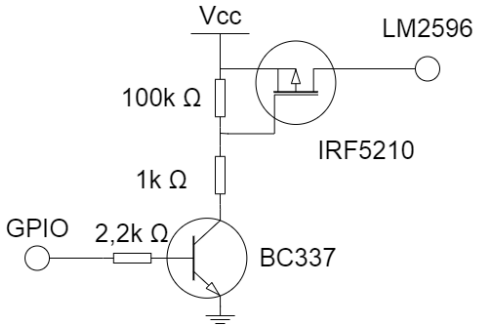
Figura 10 – Regulador de tensão LM2596

Fonte: (ELETRONICA, 2019)

3.7.2 CHAVE DE ALIMENTAÇÃO

Para efetuar a ativação e desativação do regulador de tensão e também do modem GSM/GPRS foi utilizado um circuito chaveador controlado por um I/O da **Raspberry Pi**, o qual utiliza transistores para o chaveamento: um transistor bipolar BC-337 e um MOSFET IRF5210. Assim, é possível garantir um consumo praticamente nulo de energia por esta parte do sistema nos momentos em que não houver necessidade de transmissão de dados entre a armadilha fotográfica e o servidor remoto. A simples conexão do modem com a rede de telefonia móvel em um momento em que não houvesse envio de arquivos já resultaria em um consumo desnecessário de energia. Outra possibilidade para minimizar o consumo seria utilizar o ***sleep mode*** do modem, no entanto, apesar de diminuir o consumo, este não seria nulo, sendo cerca de 700 μA segundo sua folha de dados. Em contrapartida, o consumo do circuito utilizado é praticamente desprezível, pois quando o transistor está na região de corte, seu consumo de corrente reversa é inferior a 1 μA conforme sua folha de dados. Na Figura 11 é possível ver o circuito chaveador utilizado.

Figura 11 – Circuito chaveador Modem



Fonte: Própria (2020)

3.8 RASPBIAN

Neste projeto um dos principais aspectos para a escolha da **Raspberry Pi** foi a possibilidade de utilização de um sistema operacional, assim, permitindo a utilização de suas ferramentas de configuração e também de seus processos já configurados, viabilizando a programação com um nível maior de abstração.

O sistema operacional selecionado foi o ***Raspbian*** por ser um sistema próprio para uso com a placa selecionada e também o SO oficial para utilização com a mesma. Este é um sistema operacional baseado no ***Linux*** distribuição ***Debian***, gratuito e que pode ser baixado

do site oficial da ***Raspberry Pi Foundation*** de duas maneiras, baixando o sistema em si ou baixando o NOOBS, um instalador para o sistema em questão. Neste projeto o ***Raspbian*** foi instalado utilizando o NOOBS.

Para possibilitar uma instalação ágil e simples da armadilha em campo é necessário que a mesma inicie seu procedimento de configuração e posteriormente de captura, armazenamento e envio de arquivos assim que energizada. Para isso era necessário que o software da armadilha fotográfica inicializasse sua operação assim que possível, para tal foi utilizado o arquivo ***Crontab***.

Crontab é um arquivo de configuração onde podem ser escritos os programas a serem executados pelo sistema operacional, permitindo que os programas escolhidos sejam iniciados ao energizar a placa ou em um horário específico. Para tal é necessário inserir no ***Crontab*** os comandos ***bash*** a serem executados, por exemplo para um programa ser inicializado assim que reiniciada a placa utiliza-se o comando ***@reboot*** e o *path* para o programa que deve ser inicializado.

Desta maneira, ao ligar a placa a aplicação de configuração da armadilha fotográfica é iniciada, configurando a aplicação que faz a leitura dos sensores e captura das imagens da vida silvestre. Quanto à aplicação que realiza o envio dos arquivos, ela foi configurada para acontecer uma vez ao dia, no entanto, caso o pesquisador queira enviar as fotos logo após serem capturadas, eles podem alterar o modo de operação no arquivo de configuração.

3.9 ARMAZENAMENTO REMOTO

3.9.1 ENVIO DE ARQUIVOS

O armazenamento das fotos deve ser feito em um servidor remoto a fim de facilitar o acesso aos dados capturados pela armadilha fotográfica. Assim, o serviço selecionado para o armazenamento foi o **Google Drive**. O **Google**, entre seus diversos projetos, possui uma plataforma chamada *Google Developers* onde são disponibilizados tutoriais e exemplos de como utilizar funcionalidades mais específicas de seus produtos.

Para a utilização do serviço de armazenamento do **Google Drive** foi utilizada sua Interface de Programação de Aplicações (API), a qual permite fazer *download* e *upload* de arquivos para contas do **Google Drive**. Utilizando esta API é possível realizar de forma ordenada o *upload* de mais de uma armadilha fotográfica para mesma conta, ou seja, é possível por intermédio deste sistema de arquivos salvar as fotos de uma armadilha fotográfica em uma pasta específica para ela.

Uma API é um *software* que habilita e facilita a comunicação entre dois *softwares* diferentes, assim, esta API possibilita a configuração do envio e recebimento de arquivos para uma pasta do **Google Drive** de forma automatizada.

Para utilizar esta funcionalidade e automatizar o envio de arquivos é necessário primeiramente habilitar a utilização da API (Google Drive API) pela conta do **Google Drive** onde os arquivos devem ser armazenados, no caso a conta própria para armazenamento de arquivos das armadilhas instaladas, ao fazer isso é gerada uma credencial, que deve ser salva para os passos futuros de configuração e acesso.

Em seguida é preciso utilizar o código em **pyhton** disponibilizado pelo **Google Developers**, chamado ***quickstart.py***, para efetuar a configuração e habilitação do acesso a conta pelo novo dispositivo permitindo acesso aos dados armazenados na conta, neste caso a **Raspberry Pi Zero W**. Estes passos são para efetuar a autenticação do acesso da placa à conta do **Google** especificada e também salvar o acesso, assim, esses passos precisam ser seguidos apenas na primeira conexão.

Como a configuração anterior permite a leitura de arquivos na pasta, é também necessário efetuar uma configuração para que seja possível efetuar a escrita. Assim, alterando o arquivo ***quickstart.py*** foi possível configurar o envio de arquivos da **Raspberry Pi** para a conta do **Google Drive**.

3.9.1.1 METADADOS

Os metadados usados nas imagens JPEG da armadilha fotográfica estão no formato EXIF, o qual é um padrão utilizado na industria de câmeras digitais a fim de fornecer ao usuário da câmera fotos com etiquetas de dados. Desta maneira o usuário não precisa se preocupar em salvar informações como hora e local da realização da foto, pois estes estão anexados as fotos. Assim, por exemplo, caso o usuário da câmera precise pesquisar uma foto específica em um grande acervo de fotos, ele conseguirá filtrar suas buscas com o auxilio destes dados embutidos nas fotos (SIMIONATO, 2012).

As fotos armazenadas podem ser visualizadas no **Google Photos**, serviço no qual a localização das fotos pode ser observada em um mapa. Isso é possível pois essas imagens são armazenadas com metadados, nos quais estão presentes informações como suas coordenadas, modelo da câmera, horário que a foto foi criada, entre outras informações.

3.9.2 DONWLOAD DE ARQUIVOS

O *download* de arquivos é utilizado para atualizar o documento de configuração do sistema e acontece em um horário pré-determinado, no caso, todos os dias as 6 horas da manhã. Este arquivo pode ser alterado pelo usuário e está presente, no **Google Drive**, em uma pasta chamada ***configurations***, dentro da pasta da armadilha fotográfica. Desta maneira, a partir da configuração dos dados do arquivo no **Google Drive** o pesquisador pode configurar as coordenadas do local onde a armadilha foi posicionada, alterar o modo de operação e horário de envio das imagens.

Na mesma pasta onde está o arquivo de configurações, também há instruções sobre como preenchê-los para que seja efetuada a correta leitura do arquivo quando baixado pela armadilha. Estes dados estão em um arquivo PDF chamado ***configuration details***. Devem ser preenchidas as três primeiras linhas do arquivo de configuração com as seguintes informações:

- 1ª linha : coordenadas (DD MM SSSS, DD MM SSSS)
- 2ª linha : horário de envio dos arquivos (HH:MM)
- 3ª linha : modo de operação do envio (1 ou 2)

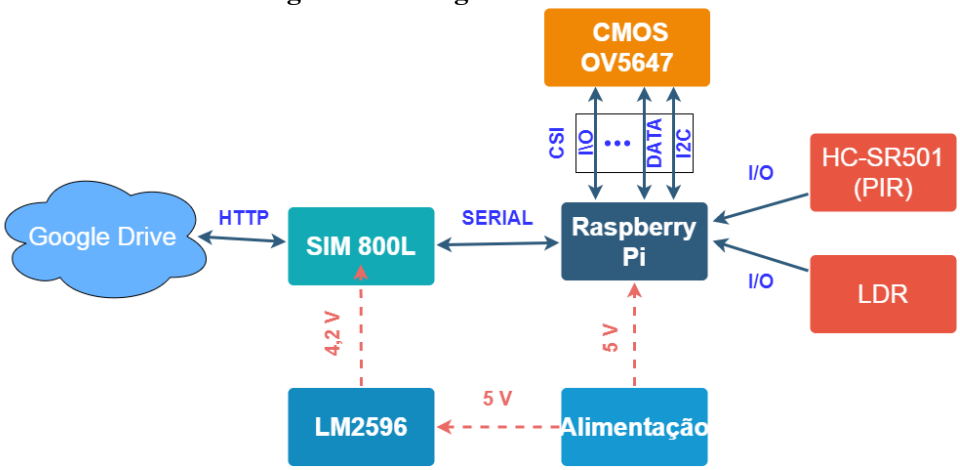
Os dados entre parênteses indicam o formato em que cada item deve ser preenchido. Na primeira linha, os dados de coordenadas devem ser preenchidos no formato DMS. Na segunda linha, os dados de horário de envio dos registros fotográficos devem ser preenchidos indicando a hora e os minutos em que o arquivo deve ser enviado diariamente para a pasta do **Google Drive**. Na terceira linha, o valor 1 indica o modo de envio diário, em que as fotos são enviadas no horário definido na linha anterior; já o valor 2 indica o modo de operação de envio contínuo, assim que cada nova imagem for registrada.

Para efetuar o *download*, como para efetuar o *upload*, foi utilizada a API do **Google Drive**, utilizando o ***quickstart.py*** com pequenas modificações também disponibilizadas no **Google Developers**.

3.10 INTEGRAÇÃO DO SISTEMA

A Figura 12 ilustra as conexões entre os módulos do sistema bem como os protocolos e padrões necessários em cada parte do projeto.

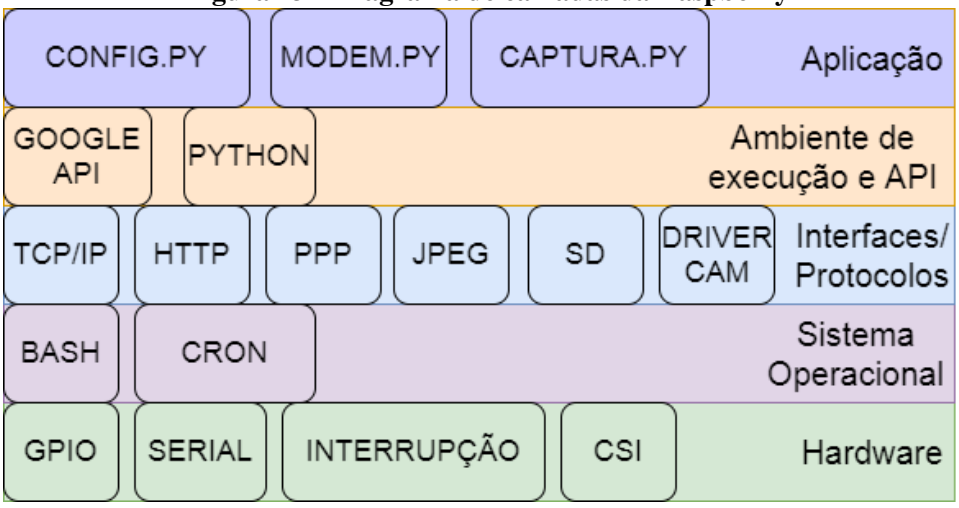
Figura 12 – Diagrama de conexões



Fonte: Própria (2020)

O diagrama de conexões apresentado na Figura 12 demonstra a efetividade da escolha da **Raspberry Pi**, pois a grande maioria de interfaces, protocolos ou mesmo bibliotecas necessários para interfaceamento entre os módulos possuem suporte para a **Raspberry Pi**. O diagrama de camadas da Figura 13 mostra quais partes da mesma foram aproveitadas no projeto. Como é possível ver na Figura 13 o diagrama foi dividido em cinco partes: *hardware*; sistema operacional; interfaces e protocolos; ambiente de execução e API; aplicações.

Figura 13 – Diagrama de camadas da Raspberry



Fonte: Própria (2020)

Na camada do *hardware* as principais funções fornecidas foram a utilização de pinos de interrupção para ser capaz de ativar o dispositivo quando necessário; os pinos seriais, para a comunicação entre **Raspberry Pi** e o modem GSM/GPRS; e também os pinos GPIO, que servem para a comunicação com os sensores, além de permitir diminuição do consumo da armadilha, ligando ou desligando o modem, permitindo assim alongar a vida da bateria.

Em relação aos *softwares* legados, como protocolos, interfaces, *drivers* e programas, foram herdados alguns protocolos de muita utilidade no projeto: o *driver* da câmera facilitou a utilização da mesma, não havendo necessidade de criar todo arcabouço de ferramentas para lidar com a mesma. A formatação em JPEG é umas das mais difundidas no processamento de imagens, esta é capaz de armazenar os metadados no formato EXIF, portanto, a gravação de fotos neste padrão facilita significativamente a utilização, a transferência e a confiabilidade desses dados anexados a imagem; o protocolo HTTP serve para prover a camada de conexão para uso da API, efetuando a comunicação entre a armadilha fotográfica e a Internet.

3.11 APLICAÇÃO

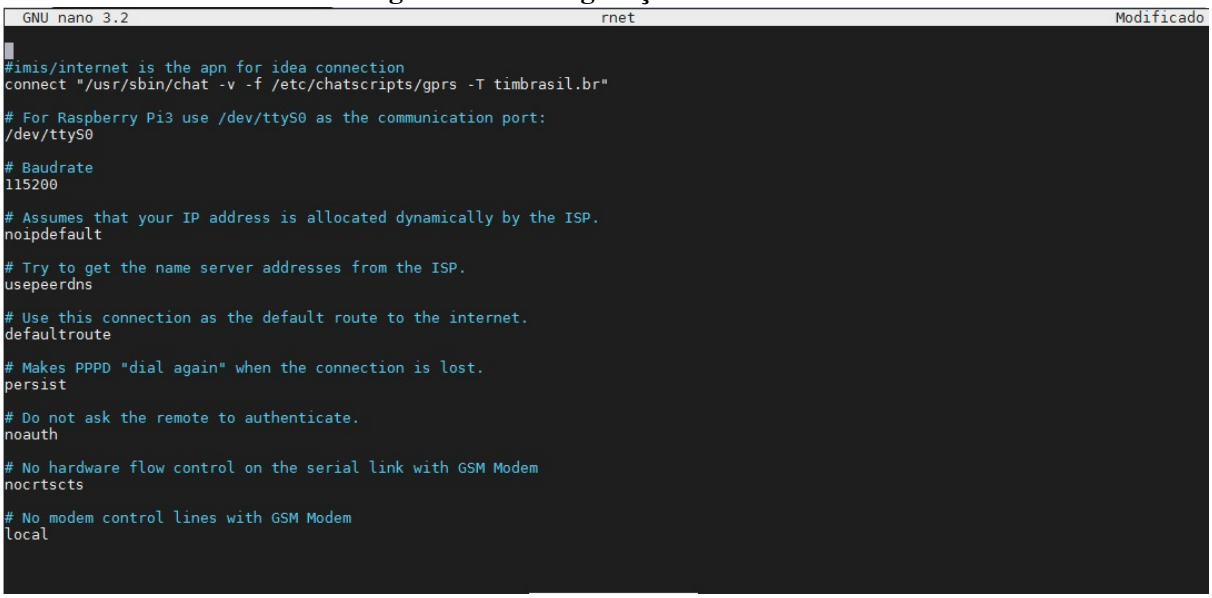
3.11.1 CONEXÃO MODEM E INTERNET

Como relatado anteriormente, a conexão entre a armadilha fotográfica e a rede mundial de computadores é feita por intermédio do modem SIM800L utilizando o Protocolo Ponto a Ponto (PPP). Para realizar tal tarefa foi utilizado o software ***PPP link*** que pode ser baixado diretamente pela **Raspberry Pi** utilizando o comando **bash** ***sudo apt-get install ppp screen elinks*** na linha de comandos (RHUDOL BZ,, 2016). A palavra ***sudo***, que pode ser encontrada no comando, faz com que o sistema entre em modo de super usuário, o qual tem permissão para utilizar os comandos ***apt-get install***, os quais buscam pacotes de um *software* desejado, efetuam o *download* e a instalação do mesmo. O ***ppp***, sendo o *software* que se deseja instalar, o ***screen*** permite a continuação do *download* em caso de interrupção do mesmo, enquanto, o comando ***elinks*** permite *downloads* em segundo plano.

Após a instalação foi necessário configurar o *software* em questão, com dados como taxa de transmissão de dados, portas seriais seriam utilizadas, *gateway* para conexão. A Figura 14 exibe as configurações necessárias para o *software* em nossa aplicação. Para que haja comunicação entre a rede GSM/GPRS e a Internet é necessário fornecer um *gateway* para o modem, nesse caso o *gateway* utilizado foi da operadora de celular Tim: ponto de acesso (APN) ***www.timbrasil.br***. Já na **Raspberry Pi** foi necessário desativar as demais funções que estavam ligadas a esta serial que se comunica com o modem para que não houvesse interferências na

comunicação. E por fim, para ativar o novo padrão de conexão foi utilizado o comando ***sudo pon rnet*** na linha de comando da **Raspberry Pi**, onde o comando ***pon*** executa a ativação do ***PPP link*** baseado no arquivo de configuração mostrado na Figura 14, o qual foi nomeado de ***rnet***. Desta forma, o ***PPP link*** é capaz de fazer o interfaceamento entre a rede de telefonia móvel, no caso GSM/GPRS, com a Internet propriamente dita, possibilitando, assim, que todo os recursos disponíveis pela API do **Google Drive** sejam utilizados.

Figura 14 – Configurações do PPP link



Fonte: (RHUDOL BZ„ 2016) (adaptada)

Quando o *software* ***PPP link*** está ativo a conexão com modem fica bloqueada para outros tipos de acesso via serial, como por exemplo, fica impossibilitada a utilização de comandos AT para verificação do nível de sinal da rede. Para desativar o ***PPP link*** deve-se usar o comando ***sudo poff rnet*** na linha de comandos, onde o comando ***poff*** faz a desativação do programa liberando a porta serial que está definida no arquivo ***rnet***. Para resolver o problema de necessidade de ligar e desligar o ponto de acesso com a Internet foram preparados dois arquivos ***shell scripts*** que têm como função ligar e desligar o programa ***PPP link***. Desta forma, quando for necessária a utilização da interface AT do modem é necessário utilizar as funções capazes de chamar os comandos que estão em ***shell script***, os quais podem desligar o ***PPP link*** liberando, assim, a porta serial que ficará disponível para a interface AT.

3.12 INICIALIZAÇÃO DA ARMADILHA FOTOGRÁFICA

Ao inicializar, a **Raspberry Pi Zero W** busca pela rede móvel para poder efetuar a primeira conexão e então, ao se conectar, obtém dados de data e hora, já que quando

desenergizada estes dados são perdidos e são essenciais para o correto armazenamento das fotos, indicando a data, hora, minutos e segundos em que a imagem foi capturada.

Após se conectar a uma rede é feito o *download* e a leitura do arquivo texto oriundo do **Google Drive**, assim, são configuradas as coordenadas do local onde a armadilha foi instalada e também o horário de envio de imagens.

Estes dados obtidos através da leitura do arquivo proveniente da pasta específica da armadilha no **Google Drive** podem ser configurados pelo usuário, isto pode ser feito modificando o arquivo, como apresentado na seção 3.9.2. Para efetuar a correta modificação do arquivo de configuração, na mesma pasta onde está o arquivo, há instruções sobre como o arquivo deve ser preenchido para que sua leitura seja efetuada corretamente pela armadilha fotográfica.

3.13 OPERAÇÃO

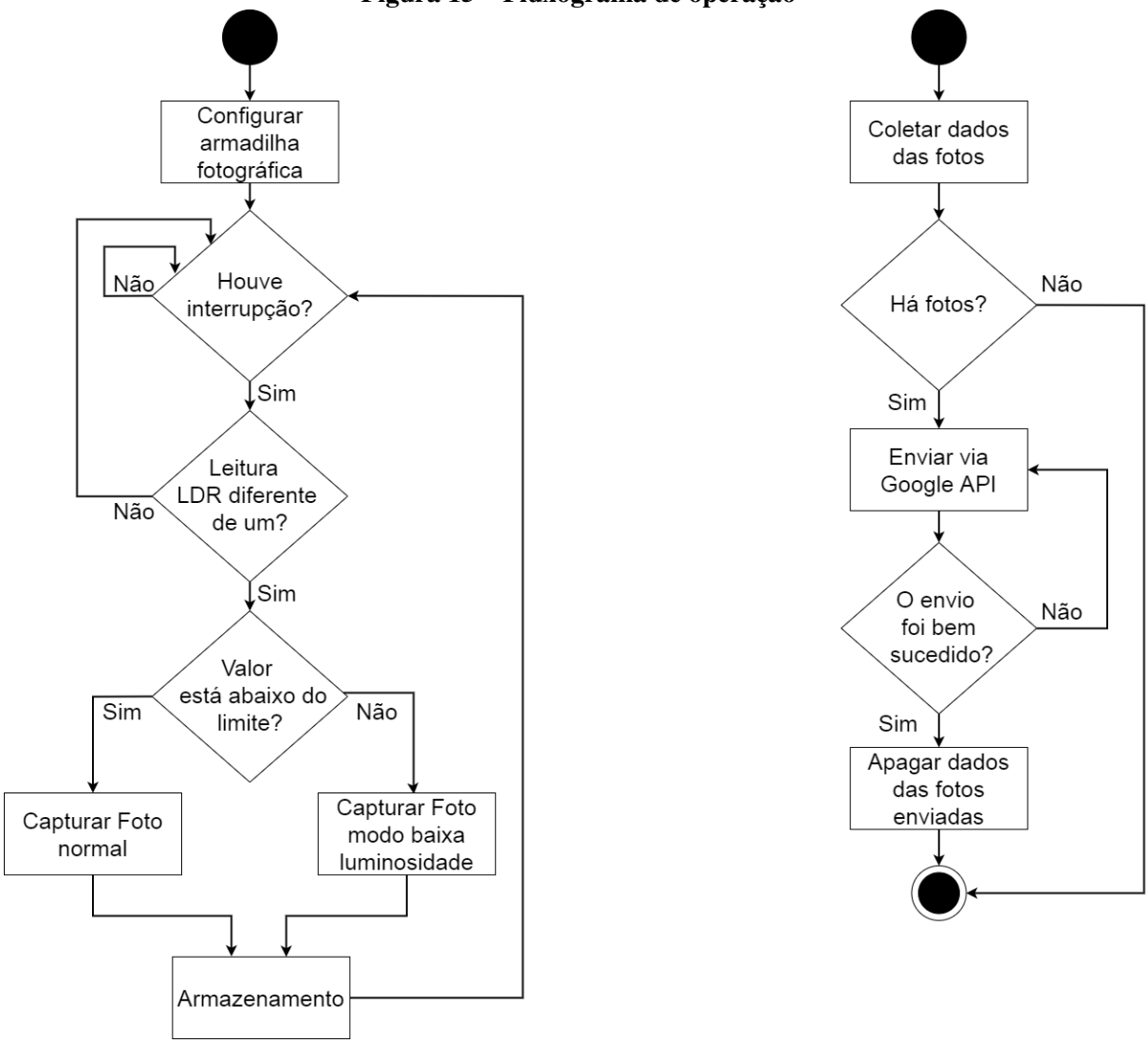
O sistema como um todo possui duas ***threads*** principais de operação, que podem ser observadas na Figura 15, a primeira relacionada à câmera e a segunda ao envio de dados para uma pasta remota.

A ***thread*** relacionada à câmera, também está ligada a todos os componentes necessários para identificação do meio a fim de informar quando é necessário que uma nova foto seja registrada. Assim, estando ligada também ao sensor PIR e ao LDR.

Deste modo em um laço é verificada a intensidade luminosa no local onde a armadilha foi instalada, através do sensor LDR. Quando um animal passar em frente ao sensor PIR uma interrupção irá acontecer no sistema, assim, levando ao registro de uma foto, o modo de registro das imagens depende da intensidade luminosa no ambiente na hora em que a interrupção ocorreu. Com base nos dados de luminosidade, o processo de aquisição fotográfica pode continuar de três maneiras: registro fotográfico sem processamento da imagem, caso a luminosidade seja ideal; registro fotográfico em modo processado ou noturno, em caso de luminosidade abaixo do ideal; ou não registro, caso a luminosidade esteja abaixo do limite estabelecido. Os valores limites de luminosidade foram definidos experimentalmente.

A ***thread*** relacionada ao envio de dados é ativada de acordo com os dados preenchidos no arquivo de configuração, explicado em detalhes na seção 3.9.2, presente na pasta do **Google Drive** onde as fotos são salvas, podendo ser ativada de duas maneiras selecionadas pelo usuário, em um horário específico do dia ou a cada nova foto registrada.

Figura 15 – Fluxograma de operação

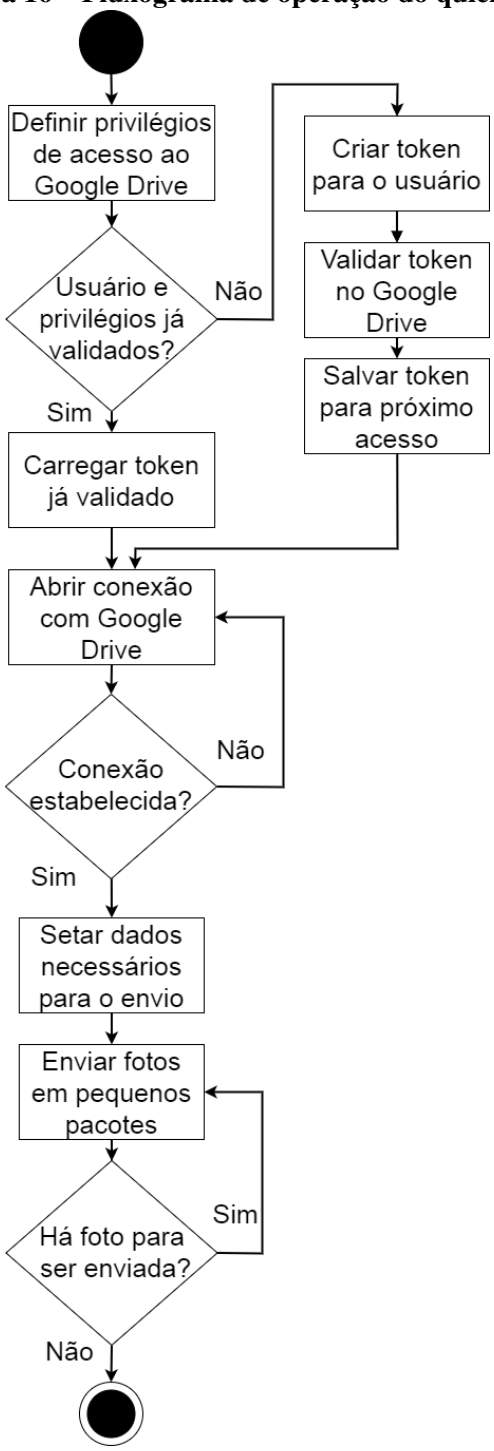


Fonte: Própria (2020)

A Figura 16 demonstra o funcionamento do programa ***quickstart.py***, programa com o qual é possível fazer a autenticação com o ***Google Drive*** de forma automática. Como pode ser visto a seguir, quando o ***quickstart*** é iniciado ele verifica se já há uma credencial válida para acessar a conta desejada, caso não exista ainda é feito a configuração desta credencial que serve como chave de acesso ao **Google Drive**. Após essa validação de credencial o programa começa a estabelecer uma conexão de serviços para poder fazer leituras ou escritas na pasta do servidor.

A versão utilizada da API do **Google Drive** suporta *upload* de arquivos com até 5 MiB em uma única requisição ***HTTP***. Portanto, ao configurar este valor para o seu máximo permitido cada foto é enviada em uma única requisição ***HTTP***, pois como dito anteriormente o tamanho médio das imagens é de 450 kiB. Fato este que pode gerar alguns problemas, pois em caso de perda de conexão com o servidor do **Google** enquanto está sendo feito o *upload*

Figura 16 – Fluxograma de operação do quickstart



Fonte: Própria (2020)

de uma imagem, este processo de *upload* é reiniciado, fazendo com que seja necessário o reenvio de todos os dados que já haviam sido enviados antes do erro. Desta maneira, utilizou-se um recurso interessante disponibilizado pela API do **Google** no qual é permitido modificar a quantidade de dados enviados em cada requisição ***HTTP***, no caso para imagens escolheu-se

pacotes de 256 KiB, a fim de melhorar a robustez nas transferências dos arquivos para o serviço do **Google**, pois assim, havendo erro na transmissão de um pacote a retransmissão desse pacote representaria um percentual menor do arquivo completo. Além de facilitar o *upload* de pacotes por serem menores, nesse formato também há uma estratégia interessante, na qual quando há perda de conexão com o servidor só é necessário transmitir os pacotes faltantes, referentes à foto que estão sendo salvas na nuvem. Caso fosse usado um formato mais simples de envio, no qual a conexão fosse aberta e a foto enviada em grandes pacotes, a chance de perda de pacotes seria maior, pois exigiria mais tempo de conexão para efetuar o *upload* de cada pacote. Consequentemente, levando em conta as possibilidades de erro de transmissão, um consumo energético e de dados móveis maior.

3.14 CONSUMO

Com objetivo de reduzir o consumo energético do sistema, garantindo sempre o mínimo consumo possível e sabendo que a **Raspberry Pi** não possui um modo de operação nativo para proporcionar baixo consumo, foram tomadas algumas ações com o intuito de reduzir o consumo energético. Para isso foram desativados os periféricos que não são utilizados durante o funcionamento da armadilha fotográfica. Foram desativados: o adaptador HDMI, o LED indicador de funcionamento da placa e o adaptador WI-FI.

Outra medida foi a configuração da ativação do processo de captura fotográfica, a qual foi efetuada por meio da configuração de uma interrupção no pino onde foi conectado o sensor infravermelho. O sensor é apenas ativado quando há identificação de movimentação em frente à armadilha fotográfica. Isso permite que a câmera fique desligada quando não estiver efetuando registros fotográficos e garante uma maior economia de energia para o sistema.

A fim de garantir o mínimo consumo por parte do modem GSM/GPRS, foram utilizados transistores como chave para ativar e desativar o regulador que alimenta o modem, portanto, quando não houver envio de dados a chave é aberta e o consumo por parte do modem e do regulador de tensão é da ordem de micro ampères.

Com estas configurações, o consumo médio esperado do sistema em repouso é de 60mA e quando inicializado o processo de captura de imagens e transmissão de dados de 200mA, como é apresentado na seção 4.3 . O que representa uma mudança significativa quando comparado com o funcionamento da **Raspberry Pi Zero W** com estes periféricos habilitados, como foi mostrado na Tabela 1.

3.14.1 REDUÇÃO DE CONSUMO

Uma maneira que foi considerada para reduzir o consumo do sistema, seria o interfaceamento com outro microcontrolador com menor consumo energético, de maneira que o sensor de presença ficasse ligado a este microcontrolador auxiliar, no qual seria programada a interrupção caso houvesse ativação do sensor de presença. Após o sensor de presença e consequentemente o microcontrolador serem ativados, haveria então a ativação da **Raspberry Pi Zero W**. Desta maneira a placa auxiliar permaneceria em modo de baixo consumo enquanto o sensor de presença não detectasse nada. Portanto, consumiria níveis de corrente na ordem de μA durante seu tempo ocioso e a **Raspberry Pi**, que possui maior consumo e não possui modo de baixo consumo, seria inicializada apenas nos momentos necessários, para registro e envio de dados.

Porém, esta nova maneira de integração do sistema não se aplica a este caso, pois o tempo de inicialização da **Raspberry Pi** é muito longo, quando considerado o tempo que um animal poderia levar para passar em frente a armadilha. Ou seja, se a latência do sistema for muito alta, no momento em que a foto fosse efetivamente registrada o animal já poderia ter se distanciado da armadilha e do campo de captura da câmera. Caso a captura das fotos fosse um evento que não dependesse de um evento externo, a presença de um animal, e fosse regular no tempo, seria possível utilizar esse microcontrolador auxiliar com a estratégia de ***wake up*** no horário necessário. Um exemplo de um possível caso de utilização desse sistema seria o estudo e acompanhamento da recuperação de uma área de floresta desmatada.

4 TESTES

4.1 UPLOAD DE IMAGENS

A fim de testar e aprender como funciona a API do **Google Drive** foi testado o código do ***quickstart.py*** disponibilizado pelo **Google**. Neste teste inicial era possível acessar uma conta do **Google Drive** sem a necessidade de intervenção humana, exceto durante o primeiro acesso, pois este é o acesso responsável por gerar o arquivo de credencial necessário para o acesso automático. Após ter um acesso válido, o código exemplo do ***quickstart.py*** era capaz de ler todos os arquivos presentes na conta do usuário e listar até vinte itens na tela. Esta lista continha duas informações que seriam importantes para o desenvolvimento do projeto, são elas o nome do arquivo ou diretório e também sua chave identificadora.

Foi necessário então entender como mudar o processo do ***quickstart.py*** para que ao invés de ler do **Google** fosse possível escrever no mesmo. O primeiro passo para tal objetivo foi alterar a variável ***SCOPES*** do ***quickstart.py***, pois ela é responsável pelos tipos de permissões que a aplicação pode ter. Na versão exemplo do **Google** a aplicação só teria acesso para leitura de dados, portanto, mudou-se o tipo de permissão para leitura e escrita. Este fato de mudança de permissão tem uma consequência na aplicação, pois é necessário refazer a credencial de acesso automático toda vez que se alterar o tipo de permissão. Outra mudança que pode invalidar o arquivo de credencial é caso haja uma mudança na versão do **python** utilizado, ou seja, se a credencial for gerado pela **python** na sua versão 2.7 esta credencial só é válida para esta versão. Caso se queira utilizar a versão do **python** 3.7 é necessário gerar uma nova credencial para a aplicação.

Para o *ulpload* de uma imagem era necessário fornecer algumas informações para a aplicação, sendo elas: o nome do arquivo, o diretório que o arquivo estava presente, bem como o tipo de mídia de Internet, o MIME *type*. Neste caso os MIME *types* utilizados para imagens foram ***image/jpeg*** e ***image/png***, onde o tipo da imagem e o subtipo era JPEG ou PNG, o qual corresponde para o método de compressão utilizada na imagem. Por fim, para começar o *upload* da imagem foi necessário escolher qual método que seria utilizado, a API fornece três opções

de *upload*: simples, em várias partes e recuperável.

O *upload* simples serve para uma transmissão única, ou seja, requisição única ao servidor, ele suporta arquivos de até 5 MiB, o qual não tem o compromisso de salvar metadados, e em caso de falha de conexão seria necessário reiniciar o processo. O *upload* de várias partes, se assemelha ao método anterior com a diferença de que se é possível salvar os metadados, este método também trabalha com requisição única ao servidor. E, por fim, o *upload* recuperável, este método é o mais completo das três, pois além de contemplar as mesmas características das anteriores nele é possível fazer *upload* de arquivos maiores do que 5 MiB, pois ele faz múltiplas requisições ao servidor, ou seja, ele é capaz de salvar os arquivos grandes em requisições HTTP de 5 MiB por vez e em caso de falha de conexão ele retoma o *upload* do último pacote enviado. Outro fator interessante já mencionado no capitulo passado é que neste método existe a possibilidade de modificar o tamanho da requisição do HTTP para valores menores.

Como eram os primeiros testes da aplicação utilizou-se o método mais simples para *upload*. Nessa altura do projeto escolheu-se imagens arbitrariamente para o teste. Testou-se o *upload* para o *Google Drive* de dez imagens com tamanhos variando de 50 KiB até 150 KiB, obteve-se êxito neste teste utilizando a conexão Wi-Fi da **Raspberry Pi**. Após este feito, a estratégia de *upload* que era capaz de salvar os metadados da imagem no servidor foi testada. Igualmente, este teste também ocorreu sem problemas.

Então, com a API validada foi possível começar os testes com o modem SIM800L em conjunto com o ***PPP link*** como provedor de Internet para a **Raspberry Pi**. Assim, após devidamente configurada para tal, foi testado o *upload* na rede móvel 2G. Neste ponto foi possível observar alguns problemas durante o teste, as imagens com tamanho menores conseguiam ser salvas no **Google Drive**, no entanto, as imagens maiores, dependendo do horário do dia, não eram salvas e geravam exceções que interrompiam a execução do código e paravam a aplicação. Este problema era gerado devido a intermitência do sinal 2G, pois ao tentar enviar a mesma imagem após alguns minutos obteve-se êxito. Em um primeiro momento o código responsável pelo *upload* da imagem foi alterado para que caso uma exceção gerada por perda de conexão com o servidor ocorresse, a aplicação fosse capaz de reiniciar o envio sem interromper a aplicação como ocorria anteriormente.

Até esse ponto do projeto, a estratégia utilizada para o *upload* das imagens era considerada eficaz, no entanto, nas seções seguintes é mostrado como ainda foram necessários mais ajustes para solidificar esta parte do projeto. Para concluir o teste da API, foi testado o *upload* das imagens em pastas especificas do **Google Drive**, para que, em caso de utilização de mais de uma armadilha fotográfica na mesma conta **Google** os arquivos de cada uma delas

tivesse seu devido local de armazenamento. Para realizar tal tarefa, foi necessário ler os arquivos que estavam no **Google Drive** via API para obter a chave identificadora do diretório que receberia as imagens, sem o mesmo os arquivos seriam salvos na raiz do sistema de arquivos do **Google Drive**.

4.2 CAPTURA DE IMAGENS

A fim de efetuar o teste de captura de imagens foi utilizada a armadilha contendo a **Raspberry Pi Zero W**, o *power bank*, os sensores de luminosidade (LDR) e presença (PIR) e a câmera, já dentro de seu gabinete de proteção para tirar fotos de um animal de estimação, uma dessas imagens obtidas é apresentada a seguir, na Figura 17.

Figura 17 – Teste inicial de aquisição de registros fotográficos

Fonte: Própria (2020)

Este ambiente de teste foi selecionado pela sua controlabilidade, diferente do encontrado ao registrar imagens de animais silvestres, que poderiam se assustar com a presença de pessoas, considerando que se tratando de um teste inicial poderia ser necessário a realização de ajustes a armadilha.

Após este teste inicial, foi realizado um teste posicionando a armadilha em um local mais arborizado e com presença anteriormente confirmada de jacus e gambás. Assim, quando selecionado um local para o posicionamento da armadilha fotográfica foi aguardado que seu sensor de presença fosse ativado para registrar três imagens. Algumas das fotos capturadas podem ser vistas nas Figuras 18, 19 e 20.

Figura 18 – Teste aquisição de registros fotográficos 1

Fonte: Própria (2020)

Figura 19 – Teste aquisição de registros fotográficos 2

Fonte: Própria (2020)

Ao utilizar este padrão de captura de três fotos, sempre que o sensor de presença foi ativado, em 7 horas de teste, aproximadamente, foram capturadas 1700 fotos. Ocorreu uma quantidade de fotos muito maior que o esperado para este período de tempo, o que levou a decisão de efetuar apenas uma captura por interrupção gerada pelo PIR. Desta maneira, se um animal passasse em frente a armadilha sua foto seria capturada apenas uma vez e se um animal permanecesse em frete a armadilha sua imagem seria registrada diversas vezes, assim, a quantidade de registros fotográficos seria relativa ao tempo de permanência do animal no local.

Figura 20 – Teste aquisição de registros fotográficos 3

Fonte: Própria (2020)

4.2.1 CAPTURA DE IMAGENS COM BAIXA INCIDÊNCIA LUMINOSA

Além da captura de fotos no modo padrão, também é possível efetuar a captura das imagens com maior tempo de exposição, este permite uma melhor visualização do ambiente quando houver baixa luminosidade, assim, funcionando como um tipo de modo noturno.

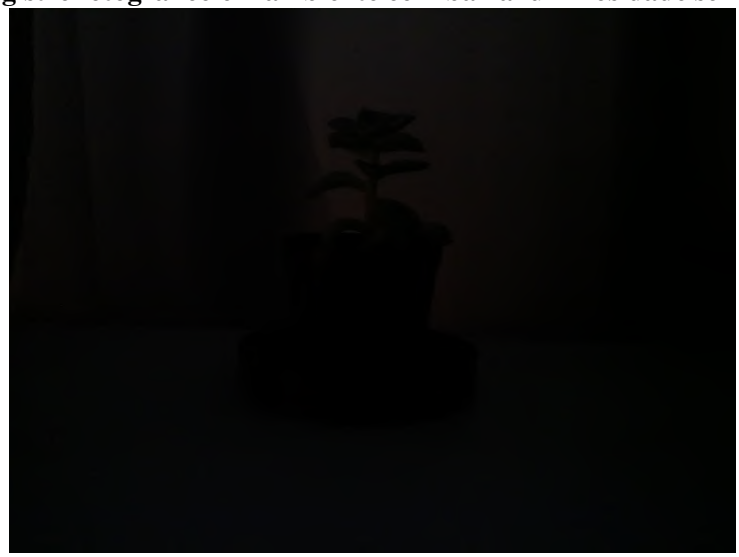
Esse teste foi realizado utilizando apenas a **Raspberry Pi Zero W** e a câmera, a fim de validar a funcionalidade desta opção de registro fotográfico.

A fim de efetuar registros com melhor visibilidade em ambientes com menor luminosidade foi utilizado o comando **bash** ***raspistill -ss 1000000 -ex sports -o imagem.jpg***, onde -ss significa *shutter*, ou obturador, e configura velocidade do obturador, este parâmetro é configurado em micro-segundos; -ex significa *exposure*, ou exposição, que pode ser selecionada entre algumas opções pré existentes de configuração; e -o significa *output*, ou saída, definindo como nome com o qual a imagem é salva o próximo parâmetro do comando.

Assim, com os valores passados para estes parâmetros o obturador fica aberto por 1 segundo para efetuar o registro e o modo *sports* é utilizado pois aumenta o ganho, aumentando a sensibilidade da câmera a luz.

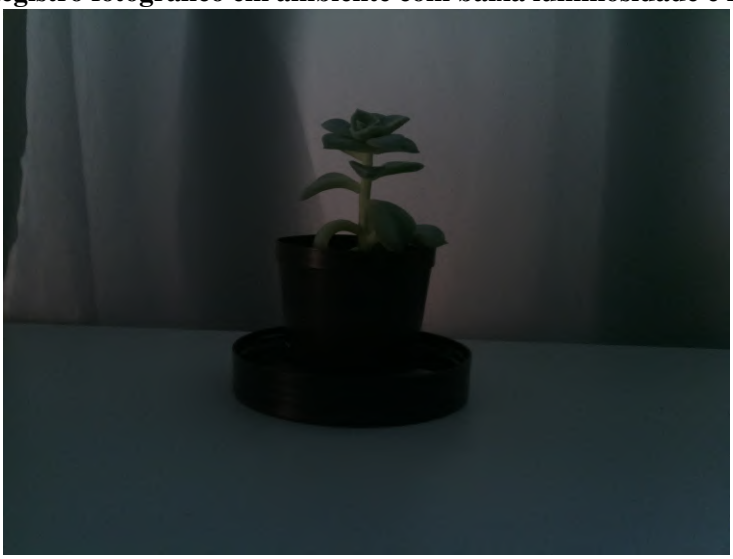
Foram registradas duas imagens, uma com o comando ***raspistill -o imagem.jpg***, sem configurações extra, o resultado pode ser observado na Figura 21, já com o comando configurando a velocidade do obturador e a exposição da imagem, o resultado pode ser observado na Figura 22. Esses comandos para aquisição das imagens de testes foram chamados a partir de um código em **python**.

Figura 21 – Registro fotográfico em ambiente com baixa luminosidade sem modo noturno



Fonte: Própria (2020)

Figura 22 – Registro fotográfico em ambiente com baixa luminosidade e modo noturno



Fonte: Própria (2020)

4.3 ENVIO AO SERVIDOR REMOTO

Após obtidas as imagens dos testes de captura de imagens, foi efetuado o teste de envio ao servidor remoto, para tal, as fotografias obtidas no teste anterior foram enviadas utilizando o modem GPRS para uma pasta do **Google Drive** .

Apesar do tamanho médio das fotos ser três vezes maior que as imagens utilizadas no primeiro teste de *upload*, pois foram capturas reais da armadilha fotográfica enquanto as

eram imagens arbitrárias, acreditava-se que este fato não geraria problemas, pois o tamanho das imagens ainda estava muito abaixo do limite imposto pelo método de *upload* de várias partes. Desta forma, a armadilha foi configurada para fazer o *upload* de 100 fotos registradas no teste anterior, simulando o envio em um horário específico, sendo escolhido às 00:00. Este teste durou cerca de seis horas e não teve êxito para salvar nenhuma foto no **Google Drive** , fato explicado pela intermitência da rede 2G, por ter uma imagem com um tamanho muito acima do da média (a primeira imagem tinha cerca de 900 KiB) e pelo fato da estratégia de *upload* ser de requisição única para cada arquivo.

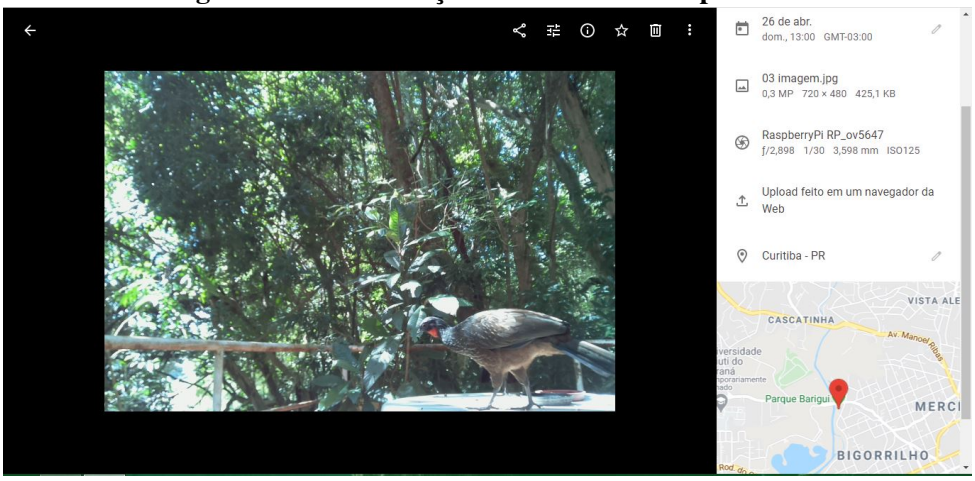
Portanto, foi necessário modificar a estratégia de *upload* para o método recuperável. Além dessa mudança, o recurso disponibilizado pela API de modificar o tamanho da requisição HTTP de 5 MiB para 256 KiB foi utilizado. Apesar de serem necessárias mais conexões que o método anterior, este método é mais eficaz, pois como não é possível prever como se comportará a rede 2G, quanto menor o tamanho do pacote a ser enviado menor o tempo dessa requisição de transferência entre a armadilha fotográfica e o **Google Drive** . Em caso de falha de envio de um pacote, o mesmo é reenviado sem influenciar os pacotes restantes, ou seja, se o último pacote tivesse sido perdido apenas ele seria reenviado, enquanto no método anterior caso ocorresse uma perda de conexão quando o *upload* estivesse quase terminado seria necessário reiniciar o envio.

Neste teste foi encontrado o desafio do tempo de envio de cada imagem ao servidor pela rede GPRS, cada foto levou em torno de 3,6 minutos para ser enviada, assim, para enviar as 1700 fotos, obtidas no teste anterior, seriam necessários 6120 minutos, ou seja, 102 horas. Fator que corroborou com a decisão de diminuir a quantidade de fotos registradas a cada ativação do sensor infravermelho de três para uma. Além disso, foi possível verificar que a modificação da estratégia de *upload* para o método recuperável teve êxito no envio das cem fotos, ou seja, a armadilha a partir deste ponto era capaz de salvar as 1700 fotos capturadas no teste anterior.

Apesar do tempo para tal tarefa ser relativamente alto, nossos esforços foram pautados em garantir a presença de todas as fotos capturadas no servidor remoto. Dadas as escolhas ao longo do projeto a capacidade que a rede GSM/GPRS nos fornece é satisfatória porém ao mesmo tempo um limitante, pois o tráfego de dados da rede 2G de telefonia móvel é bastante limitada como explicado anteriormente. Caso o foco principal do projeto fosse em direção de minimizar o tempo de envio de dados poderiam ser estudadas tecnologias que permitissem uma transmissão de dados com maior taxa de transferência. No entanto, considerando as limitações da tecnologia escolhida trabalhou-se para que o envio das fotos fosse um processo mais robusto, confiável e que tivesse alta repetibilidade. Assim, toda foto capturada será salva no servidor

remoto, desde que sejam respeitadas as condições mínimas de funcionamento da armadilha fotográfica. A Figura 23, a seguir, exemplifica como o usuário pode ver a foto capturada no sua conta do **Google Photos**.

Figura 23 – Visualização da foto no fim do processo



Fonte: Própria (2020)

Apenas um adendo, a fim de resguardar a privacidade das pessoas que cederam o local para a realização do teste de captura das fotos dos jacus, a localização mostrada foi alterada para o Parque Barigui em Curitiba, região na qual é possível encontrar tal animal.

4.4 CONSUMO

Para testar o consumo da armadilha, levando em consideração todos os componentes escolhidos para o projeto, foram selecionados dois casos de teste.

Para o primeiro caso de teste foi registrada apenas uma foto por ativação do sensor de presença, sendo o envio do registro para o servidor remoto efetuado em seguida. Deste modo cada foto registrada, resultava em uma conexão com o servidor do **Google Drive** para efetuar o envio da mesma.

No segundo caso de teste, para simular a configuração na qual as fotos são salvas no servidor remoto apenas uma vez por dia, foram realizados três registros fotográficos por ativação do sensor de presença, assim, cada conexão resultava no envio de três registros fotográficos.

Os testes foram realizados utilizado um canal de um osciloscópio e um resistor ***shunt*** de 1 Ohm, 1 Watt e 1% de tolerância, em série com o dispositivo. Desta maneira, foi efetuada a medição de tensão no resistor a fim de calcular o consumo de corrente da armadilha nos dois casos de teste. Este valor de resistor foi selecionado a fim de influenciar o mínimo possível

sobre o consumo do sistema e também por facilitar a interpretação do consumo de corrente.

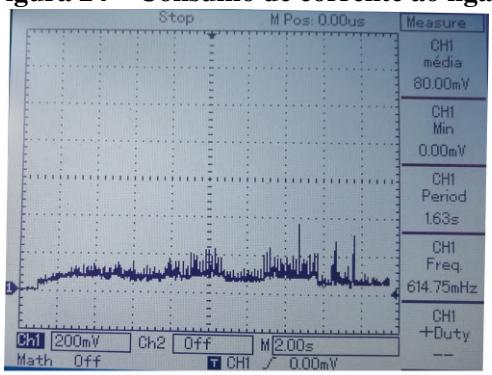
Os testes foram efetuados com o intuito de identificar e mensurar o consumo de corrente da armadilha fotográfica em funcionamento em dois momentos, durante aquisição dos registros fotográficos e durante o envio dos mesmos ao servidor remoto. Desta forma, obtendo dados sobre o consumo médio do sistema e os utilizando em conjunto com dados da bateria e da quantidade média de fotos registradas, por dia, foi possível estipular o tempo médio de funcionamento do sistema.

4.4.1 ENVIO APÓS CADA REGISTRO FOTOGRÁFICO

Este teste foi realizado em duas etapas, primeiramente, a aquisição de um registro fotográfico ao haver ativação do sensor infravermelho, em seguida, o envio deste registro fotográfico ao servidor remoto do **Google Drive** via API. Essas duas etapas foram feitas 3 vezes a fim de obter a média e desvio padrão dos valores de corrente medidos.

A seguir, na Figura 24, é possível observar o perfil de consumo quando o sistema está sendo ligado, como mencionado anteriormente, o consumo médio é de aproximadamente 80 mA.

Figura 24 – Consumo de corrente ao ligar



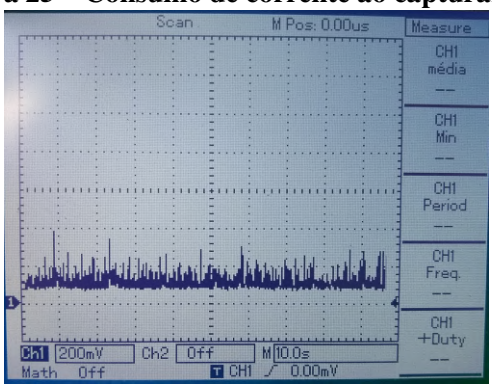
Fonte: Própria (2020)

Ao iniciar o processo de captura de imagem, inicializando a câmera, o consumo médio de corrente se eleva, como podemos ver na Figura 25, ficando próximo de 120 mA com alguns picos que passam 200 mA de corrente.

Quando é iniciada a comunicação com servidor do **Google** para iniciar o *upload* das imagens o consumo eleva-se novamente, como pode ser observado nas Figuras 26 e 27, onde há presença de picos de consumo de corrente superiores a 400 mA.

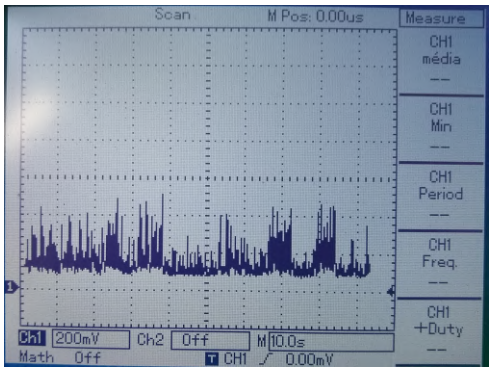
Durante o envio das fotos para o servidor foi possível observar alguns picos de

Figura 25 – Consumo de corrente ao capturar foto



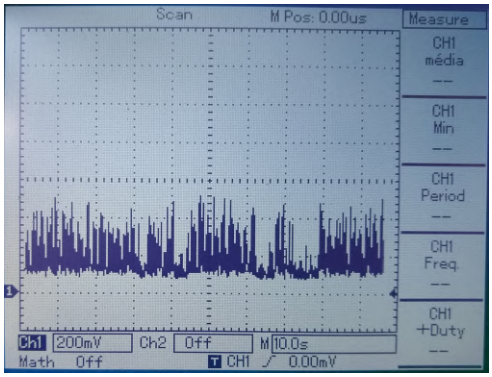
Fonte: Própria (2020)

Figura 26 – Consumo de corrente ao iniciar transmissão



Fonte: Própria (2020)

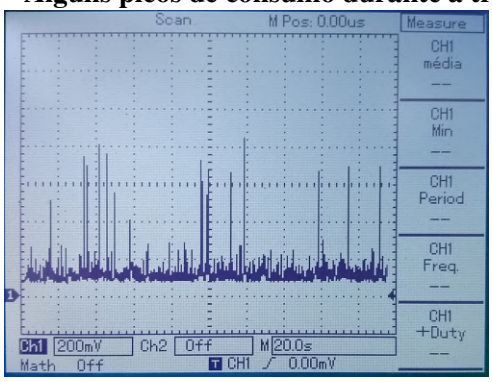
Figura 27 – Consumo de corrente durante a transmissão



Fonte: Própria (2020)

consumo, como o da Figura 28, isto pode acontecer devido a perda de conexão com o servidor do **Google Drive**, o que faria necessária a reinicialização do processo de envio, ou ainda devido a intermitência no sinal da rede GSM/GPRS.

Figura 28 – Alguns picos de consumo durante a transmissão



Fonte: Própria (2020)

4.4.2 DADOS OBTIDOS

A partir dos dados obtidos efetuando três medições de cada uma das fases de funcionamento da armadilha, sendo elas, em ordem cronológica, repouso, detecção, foto, repouso novamente, conexão e envio, foi obtida a Tabela 2.

Tabela 2 – Consumo pelo tempo no envio de três fotos

Fase	Período[s]	I[mA]	I[mA]	I[mA]	Média	Desvio padrão
Ociosa	0 a 50	60	64	58	60.67	3.06
Detecção e registro fotográfico	50 a 56	120	112	120	117.33	4.62
Ociosa	56 a 176	60	64	58	60.67	3.06
Conexão	176 a 376	120	160	145	117.33	4.62
Envio	376 a 576	220	200	225	215	13.23

Fonte: Própria (2020)

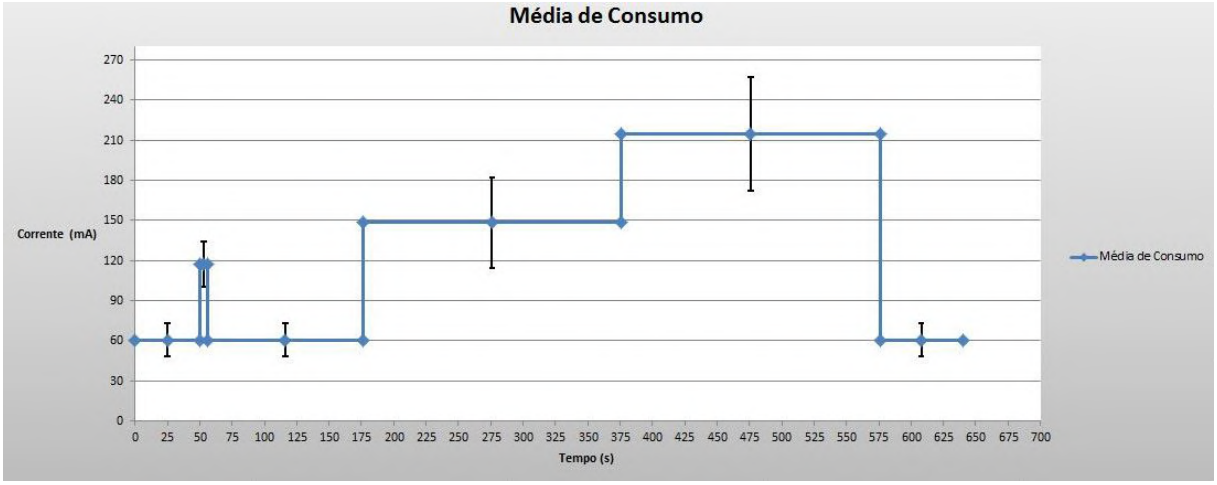
A partir dos dados da Tabela 2, foi gerado um gráfico, que é apresentado na Figura 29.

Este gráfico apresenta as 5 fases do ciclo de operação apresentadas na Tabela 2 em relação ao tempo. Desta maneira, no gráfico é possível observar com maior clareza as variações do consumo total do sistema em cada uma das fases.

4.4.3 ENVIO EM HORÁRIO ESPECÍFICO

Este teste foi realizado em apenas uma etapa, já que o valor utilizado para o consumo médio ao registrar cada imagem foi obtido no caso de teste anterior. Assim, neste contexto foi escolhido um valor arbitrário de registros fotográficos e posteriormente foi mensurado o consumo do sistema ao executar o envio deste conjunto de fotos. Para que fosse possível

Figura 29 – Gráfico de consumo médio, operando em modo de constante

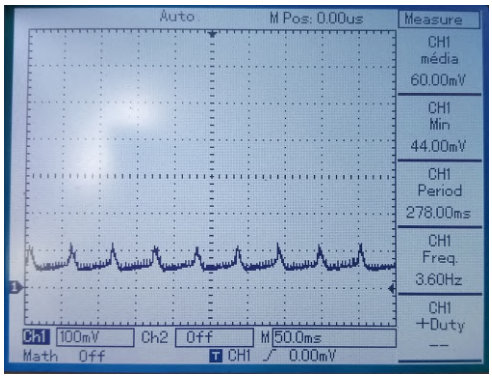


Fonte: Própria (2020)

saber em que estágio os processos da armadilha se encontravam, foi necessário adicionar dois LEDs para *feedback* visual do testador. Quando a armadilha estava no processo de aquisição de imagens um LED verde era acionado e ao iniciar a comunicação, por consequência a transmissão dos dados ao **Google Drive** um LED vermelho era acionado.

Na Figura 30 é apresentado o padrão de consumo médio da armadilha com modem e LEDs desligados, ainda não tendo ocorrido detecção pelo sensor de presença. Desligar o modem em momentos em que não estava sendo efetuado o envio de arquivos resultou em uma diminuição de aproximadamente 20 mA no consumo de corrente.

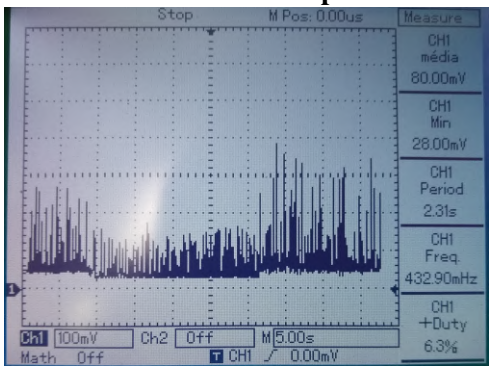
Figura 30 – Padrão de consumo sem realizar tarefas



Fonte: Própria (2020)

Em seguida houve ativação do sensor de presença e, assim, foi dado início ao processo de aquisição de imagens. A alteração no consumo do dispositivo pode ser observada na Figura 31. Vale ressaltar, nesse ponto, que o LED adicionado no teste também impactou no consumo durante esse processo. Seria possível a utilização de outro canal do osciloscópio ou de alimentar o LED utilizando uma fonte externa a fim de evitar este impacto, embora o LED estivesse ligado na mesma fonte, sua estimativa de consumo foi desconsiderada nos cálculos de consumo total da armadilha.

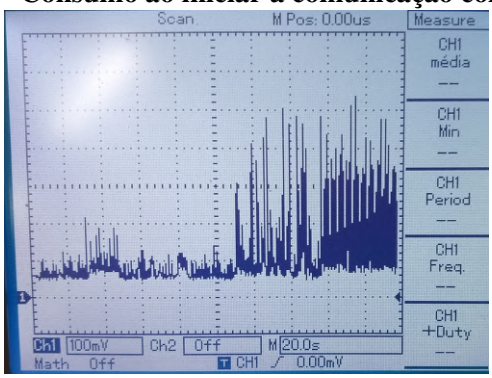
Figura 31 – Consumo durante a captura das três imagens



Fonte: Própria (2020)

Após a aquisição das três fotos o sistema da armadilha, então, desligou a câmera e ligou o modem GSM/GPRS a fim de simular uma transmissão em horário pré definido. Na Figuras 32, 33 e 34 é possível observar o consumo da armadilha durante todo o processo de *upload* das três imagens para o **Google Drive** .

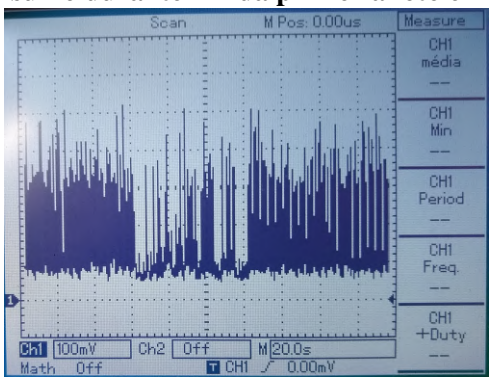
Figura 32 – Consumo ao iniciar a comunicação com servidor



Fonte: Própria (2020)

Na Figura 32, observa-se o inicio da transferência de uma imagem aos 100 segundos, na escala do osciloscópio. É possível observar o fim da transmissão da primeira foto aproximadamente aos 60 segundos na Figura 33.

Figura 33 – Consumo durante fim da primeira foto e inicio da segunda



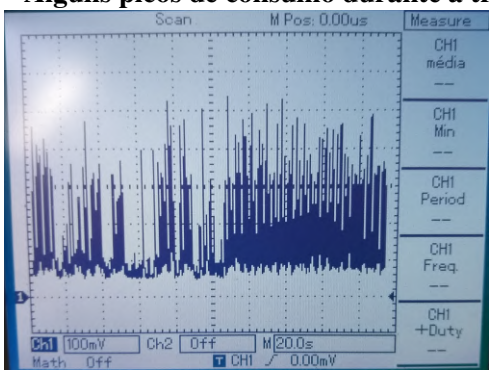
Fonte: Própria (2020)

Ao encerrar o *upload* da primeira foto não há desconexão com o servidor do **Google**, como havia no outro padrão de envio possível da armadilha. Assim, o processo de *upload* da segunda foto se encerra aproximadamente aos 10 segundos da Figura 34. E, em seguida, é iniciado o processo de *upload* da última foto, que se encerra aos 15 segundos da Figura 35.

Ao longo dos dois testes foi observada uma maior demora no envio da primeira foto para o servidor (no primeiro caso, todas as fotos das quais foi efetuado *upload* eram únicas). O tempo médio entre a captura da foto e sua disponibilidade para o usuário no **Google Drive**, foi de 9 minutos. Como a segunda foto a ser enviada ao servidor havia sido capturada anteriormente, não era necessário encerrar a conexão com o **Google**, então, o tempo de espera para a segunda foto estar disponível para o usuário, foi menor.

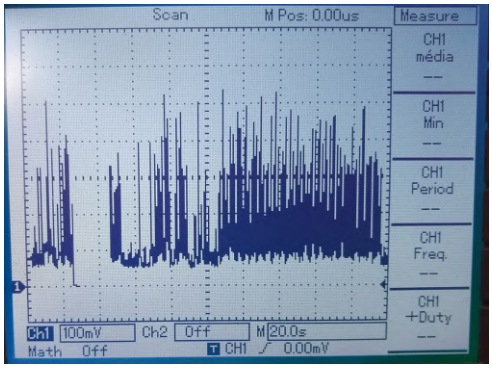
Finalmente, a fim de definir qual o tempo médio de *upload* das imagens para a Internet

Figura 34 – Alguns picos de consumo durante a transmissão



Fonte: Própria (2020)

Figura 35 – Consumo ao fim da terceira foto enviada



Fonte: Própria (2020)

mais um teste foi realizado. Este, por sua vez consistiu em realizar o envio ininterrupto de 100 imagens para o **Google Drive**, tal processo durou 6 horas. Ou seja, o tempo médio para que cada foto ficasse disponível ao usuário foi de aproximadamente 3 minutos e 36 segundos.

4.4.4 DADOS OBTIDOS

A partir dos dados obtidos efetuando três medições de cada uma das fases de funcionamento da armadilha, sendo elas, em ordem cronológica, repouso, detecção e foto, repouso novamente, conexão e envio, foi obtida a Tabela a seguir.

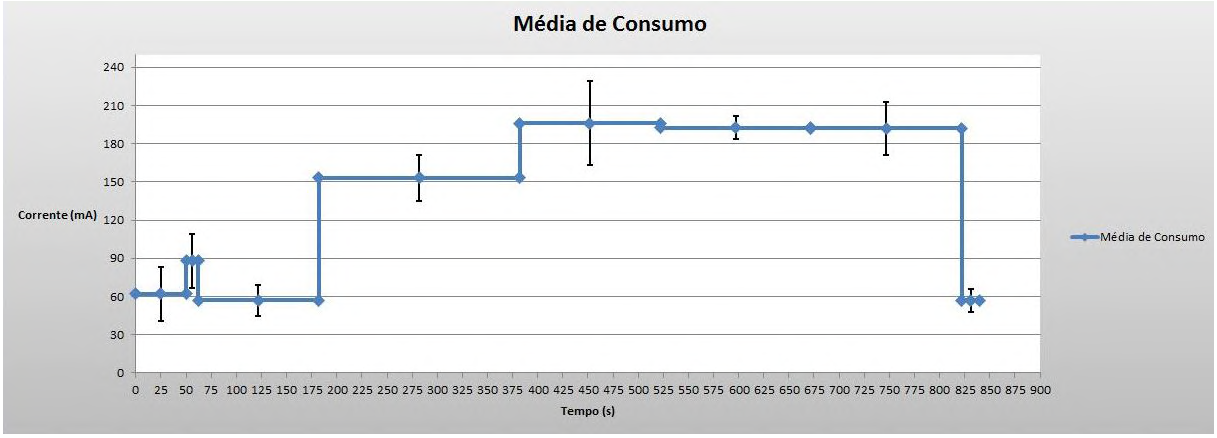
Tabela 3 – Consumo pelo tempo no envio de três fotos

Fase	Período[s]	I[mA]	I[mA]	I[mA]	Média	Desvio padrão
Ociosa	0 a 50	70	56	70	62	7.21
Detecção e registro fotográfico	50 a 62	90	94	80	88	7.21
Ociosa	62 a 182	55	56	60	57	2.65
Conexão	182 a 382	160	152	148	153.33	6.11
Envio foto 1	382 a 522	210	184	200	198	13.12
Envio foto 2	522 a 672	190	192	196	192.67	3.06
Envio foto 3	672 a 822	200	186	190	190	7.21

Fonte: Própria (2020)

A partir dos dados da Tabela 3, foi gerado um gráfico, que é apresentado na Figura 36.

Figura 36 – Consumo pelo tempo no envio de três fotos



Fonte: Própria (2020)

Este gráfico apresenta as 5 fases do ciclo de operação apresentadas na Tabela 3 em relação ao tempo. Desta maneira, no gráfico é possível observar com maior clareza as variações do consumo total do sistema em cada uma das fases.

4.5 ESTIMATIVAS DE CONSUMO

Considerando os valores de consumo obtidos a partir dos testes foi possível estimar o tempo de funcionamento do sistema. Para isso foi considerado o consumo da **Raspberry Pi Zero W** em repouso sendo de 45 mA, e supondo que seriam efetuados 20 registros fotográficos a cada 4 horas, efetuando após a aquisição dos registros seu envio.

A aquisição de vinte registros fotográficos dura em média um minuto, assim, neste intervalo de tempo o consumo seria de aproximadamente 90mA, considerando o tempo médio

de envio de cada foto sendo de 3,6 minutos. Portanto, 20 fotos durariam 72 minutos consumindo 190mA e considerando o tempo de conexão de aproximadamente 3,3 minutos consumindo 130mA.

Assim, por ciclo de funcionamento, para o envio seriam utilizados 228mAh, para a conexão 7,15mAh, para as fotos 1,5mAh e para o tempo ocioso entre as conexões 122,78mAh. Portando a cada ciclo de funcionamento seriam utilizados 359,43mAh e por dia 2156,58mAh. Isto significaria a possibilidade de funcionamento por 4,63 dias ou 111 horas.

Tabela 4 – Estimativa de consumo por ciclo de funcionamento de 4 horas

Fase	Tempo[s]	I[mA]	I[mAh]
Registro fotográfico	60	90	1,5
Conexão	198	130	7,15
Envio	4320	190	228
Ociosa	9822	45	122,78
Total	14400	-	359,43

Fonte: Própria (2020)

4.6 TESTES FUNCIONAIS

4.6.1 FUNCIONAMENTO CONTINUO

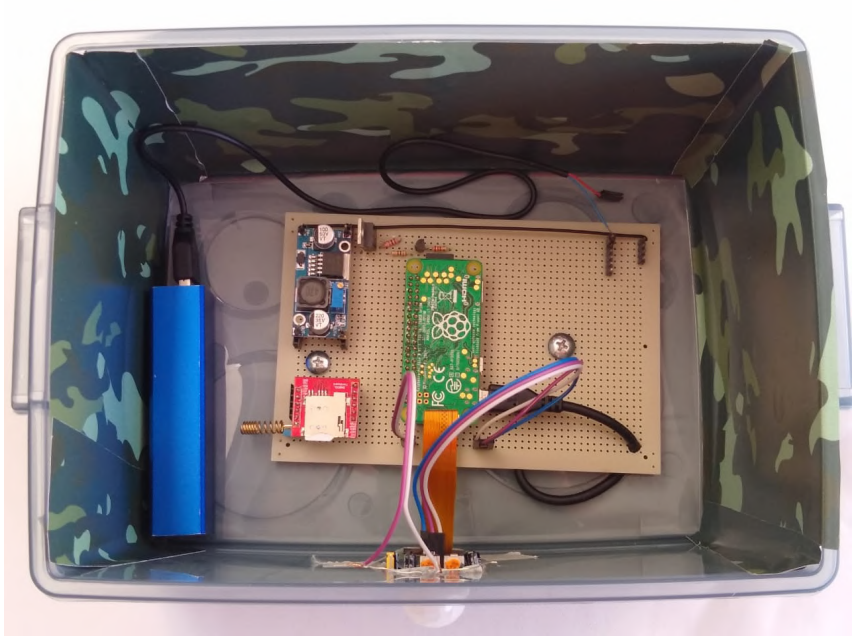
A fim de validar o funcionamento do sistema como um todo, foi realizado um teste de funcionamento contínuo incluindo todas as funcionalidades da armadilha fotográfica. Para verificar o funcionamento de todas as partes, incluindo atualizações realizadas em horários específicos, o teste teve duração de dois dias. As imagens a seguir são de um dos protótipos da armadilha fotográfica.

Neste teste, o sensor da armadilha fotográfica foi ativado propositalmente com o objetivo de gerar o registro fotográfico e, em seguida, foi realizado seu envio para o servidor remoto.

Figura 37 – Armadilha fotográfica

Fonte: Própria (2020)

Figura 38 – Armadilha fotográfica aberta



Fonte: Própria (2020)

4.6.2 BROWN OUT

Este teste foi realizado com o objetivo de verificar o tempo de operação da armadilha fotográfica, utilizando todos os dispositivos selecionados para compô-la, possibilitando a análise do consumo dos dispositivos em conjunto e do fornecimento de energia do sistema.

Para realização deste teste a armadilha foi colocada em um modo de funcionamento ininterrupto, parando apenas quando o *power bank* se descarregasse o suficiente para não mais conseguir suprir energeticamente o sistema.

Como modo de funcionamento, a cada 4 horas foram registradas 20 fotos e em seguida foi efetuado o envio desses registros para uma pasta no **Google Drive** através da rede GPRS. Este teste verificou o tempo de funcionamento do *power bank* utilizado na confecção desta armadilha fotográfica, bem como a alteração de forma, ou aparelho, de fornecimento de energia acarretaria em resultados completamente diferentes dos obtidos.

Ao final do teste haviam 160 fotos na pasta mencionada e o teste teve duração total de aproximadamente 32 horas. Ocorrendo, portanto, oito envios ao servidor remoto durante o período de testes.

Ao verificar os dados das imagens no **Google Drive** é possível verificar o horário em que foi efetuada a criação do arquivo e ao olhar os metadados contidos nas fotos é possível verificar o momento em que a imagem foi registrada. Analisando estes dados foi possível estimar que o tempo médio para o envio de 20 fotos foi de 1 hora e 5 minutos. Sabendo também que o tempo para efetuar o registro de 20 fotos é de aproximadamente 1 minuto e que no restante do tempo de teste a placa estava em repouso, é possível estimar o consumo energético médio de cada uma das etapas de teste, tanto quanto do teste total.

4.7 RESULTADOS OBTIDOS

A partir da comparação dos resultados obtidos ao longo do testes e dos cálculos de estimativa do período de funcionamento do sistema, foi observado que o volume morto deve ser levado em conta ao considerar o tempo total de funcionamento de um dispositivo que funcione a base de uma bateria.

4.7.1 ALTERNATIVAS

Uma maneira de aumentar a autonomia do sistema seria a partir da utilização de painéis solares. Considerando a grande disponibilidade deste tipo de equipamento no mercado,

é necessário um dimensionamento do painel solar para o consumo energético do sistema, assim, tendo em vista que o painel selecionado não fosse excessivamente maior, nem menor, do que o necessário para suprir as necessidades da armadilha fotográfica.

Deste modo, com os dados obtidos do teste de consumo é possível montar um cenário hipotético de utilização de um painel solar na armadilha fotográfica. Supondo que sejam capturados vinte registros da vida silvestre ao longo de um dia, e que se fizesse apenas o *upload* das fotos em um horário específico o consumo diário neste cenário pode ser visto na Tabela 5 a seguir.

Tabela 5 – Estimativa de consumo diário

Fase	Tempo	I[mA]	Consumo diário[mAh]
Registro fotográfico	60 s	90	1,5
Conexão	198 s	130	7,15
Envio	1,2 h	190	228
Ociosa	22,7 h	45	1021,5
Total	24 h	-	1258,15

Fonte: Própria (2020)

Como a alimentação do sistema é de 5V, a energia consumida diariamente é de aproximadamente 6,29 Wh. Portanto, considerando que o período médio de sol pleno no Brasil é de 6 horas e a utilização de um painel monocristalino solar de 10 W, o qual possui uma saída USB de 5V, e com dimensões de 26x14x0,2 cm, como apresentado na Figura 39, este é suficiente para aumentar a autonomia do sistema. Pois, mesmo em cenários em que haja pouca luminosidade, como em dias nublados ou chuvosos, ele é capaz de fornecer energia para o sistema e consequentemente permitindo o funcionamento do mesmo por mais tempo. Vale salientar que em um cenário de pouca geração de energia solar e de muitas capturas fotográficas a armadilha pode ficar sem energia.

Outra alternativa seria a utilização de um outro tipo de dispositivo para efetuar a alimentação do sistema, como uma bateria estacionária chumbo-ácido. Este tipo de bateria tem uma capacidade de armazenamento maior em comparação com a bateria do *power bank* escolhido, mas seu tamanho físico também é maior. Um exemplo de de baterias de chumbo-ácido é mostrado na Figura 40, a seguir, a qual tem uma capacidade de 40 Ah, saída de 12V e as dimensões de 19,6 por 16,1 por 17,1.

Figura 39 – Painel Solar 10 W, com saída em 5V

Fonte: (MERCADO LIVRE, 2020)

Figura 40 – Bateria Chumbo-Ácido

Fonte: (UNIPOWER, 2020)

Todavia, para utilização deste tipo de bateria seriam necessários alguns ajustes na armadilha: como a disposição interna dos componentes, pois a bateria ocuparia um grande volume; adicionar um controlador de carga para fazer o correto carregamento da bateria; e, por fim, um conversor abaixador de tensão na entrada do sistema para garantir os 5V necessários na alimentação dos componentes eletrônicos, visto que a baterias chumbo-ácidas em geral trabalham em tensões acima de 5V.

5 CONCLUSÃO

Para o desenvolvimento de um equipamento capaz de capturar imagens de animais silvestres em seus habitats naturais foram utilizados os conhecimentos adquiridos ao longo do curso de Engenharia Eletrônica, bem como, foi de grande importância a análise de tecnologias atuais a fim de entregar as funcionalidades desejadas. Suas características e como seu funcionamento afeta o desenvolvimento do projeto, também foi preciso levar em conta o comportamento dos animais em relação a alguns tipos de tecnologia.

Sendo a proposta inicial o desenvolvimento de um equipamento com baixo consumo de energia, assim, possibilitando seu funcionamento por longos períodos sem a necessidade de assistência humana, acredita-se o objetivo foi alcançado. Foi notado ao longo de todo o processo que haveriam certas decisões que limitariam o projeto, por exemplo, a utilização da rede 2G limitaria a velocidade de transmissão. No entanto, esta rede foi escolhida por já ser um sistema amplamente difundido e consolidado com o qual existem inúmeros modems que podem ser utilizados. Portanto, ao invés de mudar o *hardware* escolhido trabalhou-se com o intuito de o deixar robusto, ou seja, houve esforços para diminuir a quantidade de dados transmitidos e foram desenvolvidas melhorias na tarefa de envio de dados e na comunicação com o servidor.

É possível que ao longo do projeto os projetistas acreditem que já alcançaram seus objetivos e nesses momentos os testes são de extrema importância, pois nesses momentos podem ocorrer fatos não esperados. Por exemplo, em certa altura do projeto acreditava-se que a comunicação com o servidor estava pronta, no entanto, fatores como tamanho das imagens ou mesmo nível de sinal da rede móvel influenciaram nessa comunicação, tornando clara a importância de testes sistêmicos de integração entre as partes do sistema. Depois de alguns ciclos de testes, falhas, estudos e implementação foi possível chegar à versão final. Ponto este do projeto no qual é possível transmitir cada foto tirada ao servidor mesmo em situações adversas de baixo sinal de rede móvel ou com arquivos maiores que o tamanho médio das fotografias capturadas no projeto.

Levando em conta o intuito inicial do projeto é necessário ressaltar que ao modificar

o dispositivo utilizado para energizar o sistema os resultados também seriam alterados. Sendo assim, acreditamos que em projetos futuros a complementação da armadilha utilizando um sistema de alimentação oriundo de fontes renováveis como painéis solares pode ser feita. Bem como a troca da rede móvel 2G para uma rede 4G, possibilitando maior velocidade de tráfego de dados, ou até mesmo, dependendo do local no qual a armadilha for instalada, a utilização de uma rede *Wi-Fi* para a comunicação com a Internet, por exemplo em um parque urbano, ou conexão satelital em locais mais isolados e sem acesso a rede de telefonia móvel. No caso da utilização do *Wi-Fi* ao invés do 2G, essa mudança seria transparente para a tarefa de comunicação com servidor remoto, pois o sistema operacional utilizado é capaz de tomar conta dessa questão.

Além disso, este projeto teve como ponto importante a convergência de vários conceitos e teorias aprendidas ao longo da graduação. Desde as noções básicas de programação, a utilização de fluxogramas, os conceitos aprendidos em microcontroladores, até o funcionamento de um sistema embarcado. Não obstante, foi possível notar que nem todos os conhecimentos necessários para desenvolver o projeto foram vistos na graduação. Contudo, eis aqui o ponto chave do curso: através de todas as ferramentas adquiridas ao longo dos anos na Universidade, os alunos foram capazes de aplicar estas para a resolução desse problema proposto para a conclusão do curso. Além disso, este projeto aproximou mais os alunos da tecnologia IoT, pois foi posto muito esforço na busca de um sistema capaz de trabalhar de forma remota e, então, caso haja um novo desafio na linha de Internet das Coisas, eles já tem algum conhecimento obtido.

REFERÊNCIAS

ELETRONICA, B. da. **Módulo Regulador de Tensão LM2596**. 2019. Disponível em: <https://www.baudaeletronica.com.br/media/catalog/product/cache/1/image/800x/9df78eab33525d08d6e5fb8d27136e95/l/m/lm2596.jpg>. Acesso em: 8 de janeiro de 2020.

FILIPEFLOP. **Sensor de Luminosidade**. 2020. Disponível em: <https://www.filipeflop.com/produto/sensor-de-luminosidade-ldr-5mm/>. Acesso em: 26 de maio de 2020.

FOUNDATION, R. P. **Raspberry Pi W Zero**. 2017. Disponível em: <https://www.raspberrypi.org/products/raspberry-pi-zero-w/>. Acesso em: 8 de janeiro de 2020.

GAMECAMERAWORLD. **10 Best Solar Panel Trail Camera**. 2019. Disponível em: <https://gamecameraworld.com/best-solar-panel-trail-camera/>. Acesso em: 30 de setembro de 2019.

IBM. **Unidades de medida para dados de armazenamento**. 2019. Disponível em: <https://www.ibm.com/support/knowledgecenter/pt-br/SSQRB8/com.ibm.spectrum.si.doc/fqz0_r_units_measurement_data.html>. Acesso em: 9 de julho de 2020.

MARQUES, R. V. **A UTILIZAÇÃO DE ARMADILHAS FOTOGRÁFICASPARA O ESTUDO DE MAMÍFEROSDE MÉDIO E GRANDE PORTE**. 2004. Disponível em: <https://www.researchgate.net/publication/260244590_A_utilizacao_de_armadilhas_fotografica_para_o_estudo_de_mamiferos_de_medio_e_grande_porte>. Acesso em: 23 de abril de 2019.

MERCADO LIVRE. **Painel Solar 10W, saída de 5V**. 2020. Disponível em: <https://produto.mercadolivre.com.br/MLB-1349341734-painel-solar-do-silicone-monocrystalline-ultra-fino-portatil-_JM?matt_tool=26177295&matt_word&gclid=CjwKCAjwrcH3BRApEiwAxjdPTYXfGy1CpgqbjEicItnWUS2KglVHw5us4eyDX5lesiXSZ6n19RwRoxoCNEgQAvD_&quantity=1>. Acesso em: 20 de junho de 2019.

OMNIVISION. **OmniVision OV5647 Camera Module**. 2018. Disponível em: <https://github.com/techyian/MMALSharp/wiki/OmniVision-OV5647-Camera-Module>. Acesso em: 8 de janeiro de 2020.

RHUDOL BZ,. **Raspberry Pi : How to access the Internet using GSM / GPRS Modem (SIM900/SIM800)**. 2016. Disponível em: <https://www.rhydolabz.com/wiki/?p=16325>. Acesso em: 6 de junho de 2019.

ROBOTICA, C. da. **Sensor de Presença**. 2019. Disponível em: <https://http2.mlstatic.com/sensor-de-presenca-movimento-modulo-hc-sr501-pir-arduino-pic-D_NQ_NP_616905-MLB25106658701_102016-F.jpg>. Acesso em: 8 de janeiro de 2020.

SCHEMATICS, E. **How To Play with SIM800L: Part 1**. 2019. Disponível em: <https://www.electroschematics.com/introducingsim800l/>. Acesso em: 8 de janeiro de 2020.

SICHONANY, O. R. de A. O. et al. **Telemetria na transmissão de dados de desempenho de máquinas agrícolas utilizando tecnologias GSM/GPRS e ZigBee**. 2012. Disponível em: <https://www.scielo.br/scielo.php?pid=S0103-84782012000800016&script=sci_arttext&tlng=pt>. Acesso em: 15 de maio de 2020.

SIMIONATO, A. C. **REPRESENTAÇÃO, ACESSO, USO E REUSO DA IMAGEM DIGITAL**. 2012. Disponível em: <https://www.marilia.unesp.br/Home/Pos-Graduacao/CienciadaInformacao/Dissertacoes/Simionato%20A.C._mestrado_C.I._2012.pdfL>. Acesso em: 6 de junho de 2019.

TEXAS INSTRUMENTS. **Descrição técnica da Placa EK-TM4C1294XL**. 2014. Disponível em: <http://www.ti.com/tool/EK-TM4C1294XL>. Acesso em: 26 de novembro de 2019.

UNIPOWER. **Bateria chumbo-acida estacionaria**. 2020. Disponível em: <https://unipower.com.br/produto/bateria-estacionaria-vrla-12v-40ah-mod-up12400/>. Acesso em: 20 de junho de 2019.

WILCHER, D. **Building Raspberry Pi Controllers Part 5: Reading Analog Data with an RPi**. 2016. Disponível em: <https://www.allaboutcircuits.com/projects/building-raspberry-pi-controllers-part-5-reading-analog-data-with-an-rpi/>. Acesso em: 19 de maio de 2020.

APÊNDICE A – QUICKSTART DO GOOGLE DRIVE

```
from __future__ import print_function
import pickle
import os.path
from googleapiclient.discovery import build
from google_auth_oauthlib.flow import InstalledAppFlow
from google.auth.transport.requests import Request
from apiclient.http import MediaFileUpload

# If modifying these scopes, delete the file token.pickle.
SCOPES = ['https://www.googleapis.com/auth/drive']

def upload_to_drive(name,folder_path):
    """Shows basic usage of the Drive v3 API.
    Prints the names and ids of the first 10 files the user has access to.
    """
    path_token = '/home/pi/Desktop/tccSend/'
    creds = None
    # The file token.pickle stores the user's access and refresh tokens, and is
    # created automatically when the authorization flow completes for the first
    # time.
    if os.path.exists(path_token+'token.pickle'):
        with open(path_token+'token.pickle', 'rb') as token:
            creds = pickle.load(token)
    # If there are no (valid) credentials available, let the user log in.
    if not creds or not creds.valid:
        if creds and creds.expired and creds.refresh_token:
            creds.refresh(Request())
        else:
            flow = InstalledAppFlow.from_client_secrets_file(
                'credentials.json', SCOPES)
            creds = flow.run_local_server(port=0)
        # Save the credentials for the next run
        with open(path_token+'token.pickle', 'wb') as token:
            pickle.dump(creds, token)

    service = build('drive', 'v3', credentials=creds)
    TimeData = tempo(name)
```

Figura 41 – Quickstart.py com alteração para upload

Fonte: Google API Developers (adaptada)

```
    TimeData = tempo(name)
    coordinate = set_gps()
    print(TimeData)
    folder_id = '1oJFgVwC7fuUdCKMVwiwYvscSQD0sRKWP'#Id_pasta_google_drive
    path = (folder_path+name)
    file_metadata = {'name': name,
                     'parents': [folder_id],
                     'description':"Coordenadas:"+ coordinate +
                                    TimeData
    }
    resumable = MediaFileUpload(path,
                        mimetype='image/jpg',
                        chunksize = 256*1024,
                        resumable=True)
    file = service.files().create(body=file_metadata,
                                  media_body=resumable,
                                  fields='id').execute()

    print ("Photo " + name + " upped\n")
    return 0

def tempo(nome):
    ano = nome[1:5]
    mes = nome[5:7]
    dia = nome[7:9]
    hor = nome[11:13]
    minu = nome[13:15]
    seg = nome[15:17]
    dado = "\nData : "+dia+"/"+mes+"/"+ano+"\nHorario: " + hor +":" + minu + ":" +seg+"\n"
    return dado
def set_gps()
    return ("-25.478841, -49.238509")
if __name__ == '__main__':
    upload_to_drive("D20200620-T161455.jpg","/home/pi/Desktop/tccSend/pictures/")
```

Figura 42 – Quickstart.py com alteração para upload (continuação)

Fonte: Google API Developers (adaptada)

```
request = service.files().export_media(fileId=file_id,
                                       mimeType='text/plain')
fh = io.FileIO('/home/pi/Desktop/deviceConfig/configuration', 'wb')
downloader = MediaIoBaseDownload(fh, request)
done = False
while done is False:
    status, done = downloader.next_chunk()
```

Figura 43 – Quickstart.py alteração para download

Fonte: Google API Developers (adaptada)