UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS
GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Luiz Felipe Pereira

C.E.R.B.E.R.U.S.:

Robô Bombeiro

São Carlos
2016

Luiz Felipe Pereira

C.E.R.B.E.R.U.S.:

Robô Bombeiro

Trabalho de Conclusão de Curso apresentado à Escola de Engenharia de São Carlos, da Universidade de São Paulo.

Curso de Engenharia Elétrica

ORIENTADOR: Dr. Ivan Nunes da Silva

São Carlos

2016

Pereira, Luiz Felipe
P436c C.E.R.B.E.R.U.S.: Robô Bombeiro / Luiz Felipe Pereira; orientador Ivan Nunes da Silva. São Carlos, 2016.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2016.

1. robô. 2. bombeiro. 3. embarcado. I. Título.

FOLHA DE APROVAÇÃO

Nome: Luiz Felipe Pereira

Título: "C.E.R.B.E.R.U.S.: Robô Bombeiro"

Trabalho de Conclusão de Curso defendido e aprovado em 13/06/2016,

com NOTA 10,0 (DEZ, ZERO), **pela Comissão Julgadora:**

Prof. Associado Ivan Nunes da Silva - (Orientador - SEL/EESC/USP)

Prof. Dr. Danilo Hernane Spatti - (Universidade Tecnológica Federal do Paraná)

Mestre Guilherme Henrique Favaro Fuzato - (Doutorando - SEL/EESC/USP)

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Dr. José Carlos de Melo Vieira Júnior

Resumo

PEREIRA, L.F. **C.E.R.B.E.R.U.S.: Robô Bombeiro**. 2016. Trabalho de Conclusão de Curso – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos. 2016.

Todos os anos, bilhões são perdidos em bens materiais devido ao estrago causado por incêndios. Por mais ágil que seja a equipe de bombeiros, não há como evitar o tempo gasto com transporte desde a estação do Corpo de Bombeiros até o local do incidente. Sistemas de combate alternativos como os chamados chuveiros automáticos (*sprinklers*) nem sempre são adequados, uma vez que ocasionam perdas materiais quando os objetos atingidos são sensíveis à água. Em um *data center* de uma empresa, por exemplo, toda estrutura de redes e tecnologia da informação (computadores, servidores, *firewalls* etc.) pode ser sacrificada na ocasião de um pequeno incêndio no local. O projeto apresentado aqui ilustra um robô bombeiro, capaz de responder de forma imediata a um sinal de alarme de incêndio, navegar pelo ambiente e atingir a zona onde o fogo está localizado. Achada a fonte de radiação, o mesmo é programado para operar um motor que pressiona o gatilho do extintor de incêndio acoplado ao seu chassi na tentativa de suprimir o fogo, até que o agente extintor seja totalmente expelido. Após executada esta função, seu papel secundário é agir como um alarme audiovisual por meio do acionamento de uma sirene piezoelétrica e de uma luz estroboscópica embarcadas, a fim de auxiliar os bombeiros na localização da zona de incêndio. Em razão de algumas restrições aplicáveis ao robô, como incapacidade de abrir portas e descer ou subir escadas, sua aplicação é mais eficiente em ambientes abertos, escritórios, armazéns, supermercados e algumas residências.

Palavras-chave: **fogo, incêndio, robô, bombeiro, sistema embarcado.**

Abstract

PEREIRA, L.F. **C.E.R.B.E.R.U.S.: Firefighting Robot**. 2016. Graduation Thesis – Engineering School of São Carlos, University of São Paulo, São Carlos. 2016.

Every year, billions worth of material assets are lost due to the damage caused by fires. No matter how agile the firefighting team is, it is inevitable that a certain amount of time will be spent on transportation from the fire station to the place of the incident. Alternative fire-extinguishing systems such as sprinklers may not always be adequate since they cause material loss when the objects hit are sensitive to water. In a company's data center, for instance, all the network and information technology infrastructure (computers, servers, firewalls etc.) may be sacrificed in the occurrence of a small fire. The project presented here illustrates a firefighting robot, capable of responding immediately to a fire signal, navigating through the environment and reaching the zone where the fire is located. Upon finding the source of radiation, the robot is programmed to operate a motor that squeezes the lever of the fire extinguisher attached to its chassis in the attempt to suppress the fire until the extinguishing agent is completely expelled. After execution of this function, its secondary duty is to act as an audio-visual alarm device by activating an embedded piezo buzzer and strobe light in order to assist the firefighters in locating the fire zone. In reason of restrictions applicable to the robot such as inability to open doors and travel along stairways, its use is more efficient inside open areas, offices, warehouses, grocery stores and some residences.

Keywords: **fire, robot, firefighting, embedded system.**

Sumário

1. Introdução

Bombeiros podem ser considerados verdadeiros heróis da era moderna, arriscando suas vidas para proteger pessoas e seus pertences. No entanto, fatos simples podem tornar difícil o trabalho desses profissionais. Um pequeno incêndio dobra em tamanho no intervalo de 1 a 2 minutos, enquanto bombeiros podem demorar até 9 minutos para chegar ao local do incidente, mesmo em áreas urbanas. Após a chegada, eles ainda necessitam, muitas vezes em ambiente nebuloso, buscar informações que os levem até o foco do incêndio. Sendo assim, o fogo que era pequeno quando inicialmente detectado pode estar proporcionalmente 30 vezes maior no momento em que é encontrado pela equipe de bombeiros, ocasionando danos irreversíveis. Por consequência, esses desastres acabam causando bilhões de dólares em perdas e milhares de mortes a cada ano apenas nos Estados Unidos (HOWARD et al., 2015).

No Brasil, a ausência de um instituto nacional que direcione seus estudos para a área em questão, torna a obtenção de dados oficiais sobre incêndios extremamente difíceis. De acordo com informações publicadas pelo Instituto Sprinkler Brasil (ISB) em seu site oficial, resultantes de um cruzamento de dados do Sistema Único de Saúde (SUS) com os de uma pesquisa realizada pela *Geneva Association*, o Brasil se posiciona como terceiro país com maior número de mortes por incêndio no mundo. Ainda segundo o ISB, foram contabilizadas 1349 ocorrências de incêndio no ano de 2015, em edifícios como depósitos, hospitais, hotéis, escolas, prédios públicos etc. (ISB, 2015).

Atualmente, o sistema alternativo de combate mais comumente empregado é o de chuveiros automáticos (*sprinklers*), que são ativados automaticamente a partir do momento em que a temperatura no ambiente ultrapassa certo limiar, liberando água para contenção do fogo até a chegada dos bombeiros. Geralmente instalado no teto, tal dispositivo possui a vantagem de não requerer intervenção humana para seu acionamento. Entretanto, duas grandes desvantagens em relação a esse método se destacam: o alto custo de instalação das tubulações para cobrir inteiramente o local e a deterioração de dispositivos eletrônicos e documentos atingidos pela água. Além disso, o acúmulo de água pode levar ao crescimento de fungos em locais indevidos.

Com o intuito de abordar a questão, foi proposto a um grupo de cinco estudantes de engenharia nas áreas de elétrica, mecânica e computação que desenvolvessem um projeto multidisciplinar, apresentando uma solução para o problema apontado no início desta seção. O trabalho foi realizado durante três trimestres letivos dentro do programa da disciplina de *Senior Design*, na *Seattle Pacific University* nos Estados Unidos.

Inicialmente, foram discutidas entre os integrantes do grupo possíveis soluções e cenários com o objetivo de se criar um conjunto de opções a partir do qual a mais viável fosse escolhida. Ao final, chegou-se a conclusão de que seriam elas:

1. Sistema de extinção manual, em que extintores seriam espalhados por vários pontos do local e, na ocasião de incêndio, as próprias pessoas presentes seriam encarregadas de agir em resposta ao alarme de incêndio, utilizando os extintores para combater as chamas. O método seria barato, mas extremamente perigoso visto que colocaria a vida de muitos em risco, além do fato de que o fogo poderia facilmente crescer e atingir determinado ponto em que extintores de incêndio comerciais não seriam capazes de contê-lo.

2. Sistema de supressão fixo, composto por robôs diretamente fixados em trilhos que percorreriam todo o ambiente. Neste método, os robôs funcionariam como trens, capazes de navegar à procura do fogo. O uso deste implicaria em resposta rápida e baixo risco de estragos, mas também em altos custos de instalação e restrições como a área de abrangência (o robô só conseguiria se movimentar por onde houvesse trilhos instalados).

3. Sistema de extinção aéreo, composto basicamente por veículos aéreos semelhantes a *drones* munidos de reservatório de substância extintora de fogo. Ao disparo do sinal de alarme de incêndio, os mesmos entrariam em cena sobrevoando o local até localizar o incêndio, acionando o extintor embarcado. A velocidade de ataque seria excelente, porém os requisitos técnicos para construir uma máquina voadora veloz, resistente ao fluxo termodinâmico e capaz de carregar uma carga enorme sem que ocorram colisões, seriam bastante complexos.

4. Sistema terrestre de combate a incêndios, representado por um robô a ser estacionado em local adequado, à espera de sua ativação ao indício de um incêndio. Este possuiria a habilidade de transitar pelo local de forma rápida, verificando através de sensores a presença de algum sinal que indique a localização exata do fogo. Ao encontrá-lo, o mesmo acionaria seu sistema de extinção, contendo total ou parcialmente o avanço das chamas. Tal sistema poderia ser construído facilmente com motores e materiais disponíveis no mercado a um custo mais baixo do que os dois mencionados logo acima, sendo, além disso, mais seguro do que o descrito na primeira opção.

Para se determinar qual desses seria mais conveniente, foram estabelecidos alguns critérios considerados importantes como velocidade, confiabilidade, simplicidade, custo e segurança e notas foram atribuídas a cada sistema. A tabela seguinte ilustra a pontuação atribuída às possíveis soluções:

Tabela 1.1 - Pontuação atribuída às soluções de combate a incêndios.

Critério	Pontuação máxima	Sistema Manual	Sistema Fixo	Sistema Aéreo	Sistema Terrestre
Velocidade	40	25	35	34	32
Confiabilidade	20	12	13	5	13
Simplicidade	30	27	25	5	22
Custo	30	30	10	15	25
Segurança	40	5	35	25	30
Total	160	99	118	84	122

Com base nos dados mostrados na tabela, o sistema tido como mais viável foi o sistema terrestre. Foi estabelecido que a solução para o problema se desse por meio de um robô capaz de responder de forma imediata e autônoma a um alarme de incêndio e de suprimir por completo ou parcialmente o fogo sem nenhuma forma de intervenção humana. A partir daí, iniciaram-se as atividades que deram origem ao projeto CERBERUS Robô Bombeiro, descrito nesta monografia.

2. Teoria

Robô, de acordo com a definição do dicionário online Michaelis, é um aparelho automático, com aspecto de boneco, capaz de executar diferentes tarefas, inclusive algumas geralmente feitas pelo homem (MICHAELIS, 2009). No contexto desta monografia, o robô é projetado justamente para realizar uma tarefa que geralmente é atribuída ao homem (bombeiro): extinguir um incêndio. Para que isso seja possível, deve haver um conjunto de sistemas elétricos, eletrônicos e mecânicos que permitam que o dispositivo se movimente e execute determinadas funções específicas da aplicação para o qual foi desenhado.

Nesta seção, serão esclarecidos os aspectos teóricos em diversas áreas que estão direta ou indiretamente relacionadas ao projeto. Primeiramente, serão abordados as características do fogo, tipos de agentes extintores e alarmes de incêndio. Em seguida, os sistemas citados acima, integrantes do robô bombeiro, serão detalhados do ponto de vista conceitual, bem como seus respectivos componentes.

2.1 Características do Fogo

Fogo pode ser descrito de várias maneiras, entre elas através do chamado “Tetraedro do Fogo”, uma representação geométrica do que é necessário para que ele exista: combustível, elemento que pode estar em estado sólido, líquido ou gasoso e cuja queima alimenta a combustão; comburente, representado pelo oxigênio presente no ar; calor, energia requerida para que o processo se inicie e se mantenha e reação em cadeia, que torna o ciclo autossustentável. Quando o fogo foge ao controle do homem e se alastra, tem-se um incêndio (DIETRICH, 2015).

Para um ser humano, os efeitos da exposição às chamas dependem basicamente da temperatura, como esquematizado na Tabela 2.1 (NIST, 2013):

Tabela 2.1 - Relação entre temperatura e consequências causadas pelo fogo.

Temperatura (°C)	Efeito
37	Temperatura normal do corpo humano
44	Princípio de dor na pele
48	Queimadura de primeiro grau
55	Queimadura de segundo grau
62	Efeito de dormência na pele queimada
72	Pele é destruída instantaneamente

Além dos danos causados à pele, qualquer pessoa que aspire a fumaça tóxica que permeia um ambiente de incêndio pode sofrer de um processo inflamatório das vias aéreas e lesão nos pulmões. A Sociedade Brasileira de Pneumologia estima que, durante um incêndio, 77% das vítimas vão a óbito por inalação de fumaça e não por queimaduras (SAÚTIL).

2.1.1 Métodos de Extinção

Há essencialmente quatro maneiras de se extinguir o fogo: através da retirada do material combustível, resfriamento, abafamento ou utilizando-se substâncias químicas. O primeiro refere-se ao simples ato de retirar o material combustível ainda não atingido da área de propagação do fogo. O segundo consiste em diminuir a temperatura do material em combustão, reduzindo a liberação de substâncias tóxicas. O terceiro método envolve a diminuição drástica da quantidade de comburente (oxigênio) presente nas proximidades do corpo combustível, freando a reação. Pode ser implementado com o auxílio de cobertores, tampas ou dióxido de carbono. O último dá-se pela adição de agentes extintores que, quando em contato com o fogo, têm suas moléculas desassociadas, formando uma mistura não inflamável (BOMBEIROS).

Os extintores de incêndio comerciais produzidos em larga escala e amplamente utilizados são caracterizados pela classe ou classes de incêndio a que são destinados. Estas se dividem em quatro, de acordo com o combustível que deu origem ao fogo. Classe A diz respeito aos combustíveis que queimam em superfície e profundidade, deixando resíduos. Os exemplos usuais são madeira, papel e tecido. Classe B envolve líquidos e gases inflamáveis como álcool, óleo e GLP, que queimam em superfície apenas, sem formar resíduos. Classe C engloba equipamentos elétricos energizados e classe D, metais pirofóricos tais como magnésio, alumínio em pó e sódio (DIETRICH, 2015).

2.1.2 Sistemas de Detecção e Alarme de Incêndio

Em um sistema convencional de alarme de incêndio, a detecção do fogo se dá por meio da instalação de sensores em locais estratégicos do local a ser protegido. Também chamados de detectores, estes são encarregados de monitorar a variação da temperatura no ambiente ou, em alguns casos, a emissão de radiação infravermelha e ultravioleta. As informações coletadas são então direcionadas a uma central de processamento, onde os dados são verificados e, caso haja necessidade, são acionados alarmes sonoros e visuais para sinalizar aos ocupantes que a área deve ser evacuada.

Além do convencional, em que os detectores são agrupados em zonas, existem também os sistemas de alarme analógico e endereçável. Este caracteriza-se por apresentar detectores endereçáveis individualmente, indicando o local exato do foco de incêndio.

Aquele é utilizado normalmente em grandes instalações, possuindo uma central de detecção analógica que permite a configuração da sensibilidade dos detectores e informa sobre o acúmulo de poeira nos mesmos. São providos também de saídas de comunicação RS485 e protocolos abertos, podendo ser integrados com softwares gráficos (GLOBAL SYSTEM).

2.2 Sistema Motor

Constituído predominantemente pelos motores, o sistema motor é responsável por conferir mobilidade ao robô, permitindo seu deslocamento no espaço físico. Em robótica, na grande maioria dos casos, são empregados dois tipos de motores para essa finalidade: os de corrente contínua com escovas (*brushed*) e os sem escovas (*brushless*). Estes são de maneira geral mais caros e de controle mais complexo que aqueles, produzindo em contrapartida menos ruído. São também mais eficientes e duráveis e consomem menos energia, o que justifica o maior preço. A preferência por um ou outro depende de alguns pontos específicos como a verba disponível para o projeto, restrições de tensão e corrente, torque requerido e o tipo de controle a ser implementado.

Os motores, no entanto, devem ser regidos por controladores que limitem a tensão durante a partida e que controlem de forma precisa a velocidade, a posição e o torque. Além disso, atuam na proteção contra aquecimento excessivo e redução da corrente inicial (*stall current*). Podem ser projetados sob demanda ou adquiridos comercialmente, na forma de um produto completo provido de circuito eletrônico de controle, dissipador de calor e conectores de entrada e saída, com as opções de controle por modulação de largura de pulso (PWM) e via serial.

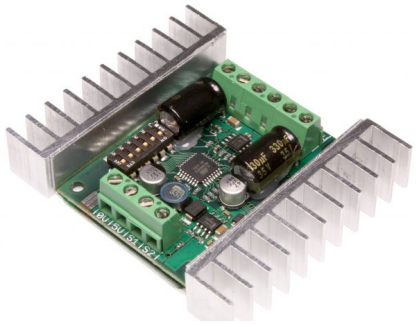

Figura 2.1: Controlador de motor.

2.3 Sensores

Os sensores integrados ao robô compõem uma de suas mais importantes estruturas, cuja finalidade é tornar possível o reconhecimento do ambiente à sua volta e, a partir daí, a tomada de decisões como desviar de um obstáculo, seguir em uma determinada direção ou acionar algum mecanismo específico. Existem diversas categorias de sensores comerciais que têm como objetivo apurar a distância em relação a objetos, detectar a presença de certo

tipo de radiação ou distinguir entre diferentes cores, cada uma associada a uma ação. Usualmente, o que se tem é uma combinação desses dispositivos, que são estrategicamente posicionados de forma a garantir ao robô um controle absoluto sobre o meio em que ele se encontra.

2.3.1 Sensor Infravermelho

Fundamentalmente, um sensor infravermelho, conforme ilustrado na Figura 2.2, tem a função de medir a distância entre ele e um determinado objeto refletor. É geralmente equipado de detector sensível à posição (PSD), diodo emissor de luz infravermelha (IRED) e um circuito de processamento de sinais, que possibilita, através da reflexão da luz e do método de triangulação, a mensuração da distância, que é traduzida em um valor analógico de tensão. Este pode ser medido no terminal de saída e convertido novamente, por meio de uma fórmula matemática, a um valor que representa a distância calculada. O alcance típico deste tipo de sensor é aproximadamente entre 10 cm e 1 metro.

Figura 2.2: Sensor de distância infravermelho.

2.3.2 Sensor Ultrassônico

Assim como o sensor infravermelho, o sensor ultrassônico (Figura 2.3) serve como aparato eletrônico de medida de distância. O mecanismo utilizado, embora muito similar ao do primeiro, difere em relação ao tipo de radiação utilizada; neste caso, são emitidas ondas de som de 40 kHz que se propagam e refletem no objeto cuja separação em relação ao sensor deseja-se estimar. A onda refletida é então captada pelo seu microfone ultrassônico após um intervalo de tempo. No momento da emissão, a saída é colocada em nível digital alto e quando a onda refletida é detectada ocorre transição para nível baixo. Assim, forma-se um pulso cuja duração é armazenada em uma variável. Partindo-se desse valor, que indica o tempo necessário para o som se propagar até o objeto e voltar, pode-se calcular a distância com base na velocidade de propagação do som no ar. Esses sensores são capazes de detectar distâncias ligeiramente superiores a 3 metros.

Figura 2.3: Sensor de distância ultrassônico.

2.3.3 Sensor de Chama

Um sensor de chama, conforme ilustrado na Figura 2.4, é constituído por detectores de radiação infravermelha na faixa de comprimento de onda entre 760 e 1100 nanômetros, denominada infravermelho próximo. Matrizes de sensores que monitoram radiação nessa região do espectro são possivelmente a melhor tecnologia disponível para detecção de fogo. O ângulo de alcance é de aproximadamente 60 graus e a sensitividade pode ser ajustada por meio de um potenciômetro. O sinal de saída é digital, sendo nível alto associado à presença de fogo e nível baixo, à sua ausência.

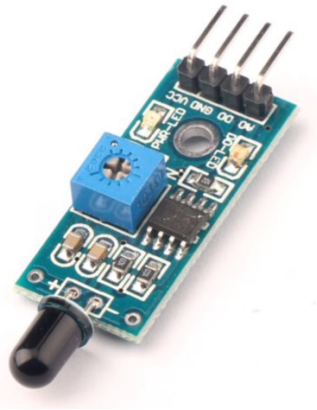

Figura 2.4: Sensor de detecção de chama.

2.3.4 Câmera

Em determinadas aplicações, é estritamente necessário que um robô reconheça objetos, cores ou elementos que o orientarão em suas escolhas de ação e, para suprir essa necessidade, câmeras são anexadas a ele, como aquela mostrada na Figura 2.5. Dentre elas, destacam-se as câmeras convencionais e aquelas que apenas reconhecem e diferenciam entre diferentes cores. Módulos podem ser acrescentados a elas e lentes podem ser substituídas com a finalidade de isolar tipos de radiação indesejadas, agindo como filtros. Para o intercâmbio de dados entra ela e o microcontrolador, são usualmente

empregados protocolos de comunicação como SPI (*Serial Peripheral Interface*) e I2C (*Inter-Integrated Circuit*).

Figura 2.5: Ilustração de câmera robótica.

2.4 Sistema de Extinção

Formado por componentes mecânicos e elétricos, o sistema de extinção tem a função única de suprimir chamas. É composto por um extintor de incêndio fixado no centro do chassi, sensores de chama e dois motores de corrente contínua com escovas. Um destes é atrelado a um cabo de aço cuja distensão comprime o gatilho do extintor. O outro motor destina-se a, com o auxílio de uma haste metálica, ajustar a posição da mangueira do extintor, direcionando o jato.

2.5 Sistema de Processamento

O sistema de processamento do robô se assemelha ao cérebro humano, sendo o local onde todas as informações provenientes dos outros sistemas são armazenadas e confrontadas. É representado por um microcontrolador dotado de um conjunto de portas de entrada e saída digitais e analógicas, um microprocessador e memórias voláteis e não voláteis. Nele são processados os sinais advindos dos sensores e, a partir de um programa escrito em sua memória ROM, as decisões de ação são tomadas. Isso envolve a programação de portas de saída, sinalizando o ligamento ou desligamento de certo sistema ou mecanismo. Interligado ao sistema de processamento, há uma placa de circuito impresso, que serve como ponte entre o microcontrolador e os sistemas de extinção, de distribuição de energia, motor e os sensores.

2.6 Sistema de Distribuição de Energia

Composto por uma bateria polímero de lítio (Li-Po) e blocos de metal com entrada única e múltiplas saídas, o sistema de distribuição de energia tem como objetivo principal alimentar todos os sistemas elétricos e eletrônicos presentes no robô. O fato de uma única fonte (a bateria) prover energia para diferentes sistemas, – sensores, controlador de motores e microcontrolador – justifica a utilização de um bloco de distribuição que divida uma única

entrada entre vários canais. Este sistema também inclui uma chave de fácil acesso para interromper a ligação entre a bateria e os elementos por ela alimentados.

2.7 Sistemas mecânicos

Somam-se aos sistemas mecânicos citados anteriormente as peças de alumínio que dão estrutura ao chassi, cortadas sob medida a partir de chapas de alumínio retangulares por meio de jato de água sob pressão, as rodas laterais e traseiras e as placas de desativação. Posicionadas perifericamente, estas têm o objetivo de parar o robô em caso de emergência. Podem ser pressionadas manualmente ou, de preferência, com os pés, através de chute, fechando uma chave fim de curso que envia um sinal de nível alto ao microprocessador indicando que os motores sejam desativados.

3. Modelagem do Projeto

Depois de apresentadas algumas características gerais de cada sistema e aspectos teóricos a eles relacionados, são descritas primeiramente neste tópico as diretrizes que orientaram o desenvolvimento do projeto e a definição dos requisitos técnicos, riscos e planos de mitigação, bem como os métodos e materiais utilizados e as razões pelas quais foram escolhidos. Em um segundo momento, são abordados a elaboração da placa de circuito impresso integrada ao microcontrolador, a documentação referente ao software base e os testes que serviram de referência para a análise dos resultados obtidos e do cumprimento dos requisitos.

3.1 Contexto e Diretrizes Gerais

O processo de modelagem e concepção do robô bombeiro iniciou-se com as atividades previstas no programa da disciplina *Senior Design* da *Seattle Pacific University* no trimestre de outono de 2014. No primeiro mês, houve reuniões frequentes entre os membros do grupo para discussão de ideias e atribuição de tarefas, que envolviam basicamente a pesquisa sobre um assunto específico ligado ao projeto. Em seguida, uma visita foi feita à estação mais próxima do Corpo de Bombeiros a fim de se obter informações privilegiadas sobre incêndios assim como conselhos e sugestões dos profissionais ali presentes.

Com todos esses dados colhidos após os estudos feitos pelo grupo e a conversa com os bombeiros, foi possível identificar os requisitos técnicos do dispositivo e visualizar os sistemas que o formariam. Nos meses posteriores, foram escritos os documentos técnicos e uma lista de materiais foi compilada. Em janeiro de 2015, teve início a montagem do robô, seguida de testes preliminares e correção de erros, modificações e programação para execução de funções simples. Depois de completamente construído o robô, foram identificados e parcialmente corrigidos problemas elétricos, permitindo a realização de testes mais complexos e refinamento do software. Por fim, o cumprimento dos requisitos foi analisado e validado e uma demonstração final foi feita.

3.2 Características Gerais

CERBERUS é um robô bombeiro com as características enumeradas abaixo. O prédio Otto Miller Hall (OMH) da Seattle Pacific University é tomado como referência para as especificações dadas, incluindo a área de andar/sala.

i) O dispositivo age como um alarme audiovisual após a tentativa de extinguir o fogo.

ii) O dispositivo não exige intervenção humana frequente para funcionar.

iii) O dispositivo possui duas chaves de desativação separadas, uma delas que notifica o microprocessador e a outra que desconecta a alimentação dos motores.

iv) O dispositivo deve poder ser guardado por um ano sem perda de funcionalidade.

v) O dispositivo funciona em apenas um andar de um prédio.

vi) O dispositivo lança mão de objetos pré-colocados que auxiliam na sua navegação.

vii) O dispositivo permanece próximo ao fogo após a liberação do agente extintor.

3.3 Definição de Requisitos

A seguir são apresentados os requisitos técnico-funcionais, definidos na etapa inicial do projeto, anteriormente à construção do primeiro protótipo.

3.3.1 Confiabilidade (R)

Dado que o fogo se expande exponencialmente quando foge ao controle humano, é crítico que o dispositivo seja confiável em todos os aspectos, sendo especialmente configurado para detectar e eliminar potenciais incêndios. A norma NFPA 1720 foi usada como padrão para o tempo de resposta desde a chamada até a chegada dos bombeiros, especificamente em áreas urbanas e suburbanas, de forma a assegurar que o robô esteja operando quando da chegada da equipe de incêndio. Os requisitos de confiabilidade são os seguintes:

R001: O dispositivo deve percorrer o prédio a uma taxa de 10000 pés quadrados (aproximadamente 929,03 metros quadrados) em 10 minutos. O dispositivo deveria realizar a busca em um andar de 50000 pés quadrados (4645,15 metros quadrados) em 10 minutos. Este requisito será validado percorrendo-se um andar aleatório do prédio em busca do fogo, tomando-se a média de cinco tentativas, em pelos menos duas zonas de incêndio diferentes. A área do andar será medida levando-se em consideração a área que o dispositivo pode acessar sem abrir nenhuma porta ou transpor qualquer rampa com grau de inclinação maior que 15 graus.

R002: O dispositivo deve identificar a presença de um incêndio que se encontre na mesma sala, com menos de 1500 pés quadrados (139,35 metros quadrados) de área, dentro de 30 segundos. O dispositivo deveria identificar a presença de um incêndio que se encontre na mesma sala, com menos de 1500 pés quadrados (139,35 metros quadrados) de área, dentro de 15 segundos. Este requisito será validado posicionando-se o robô na entrada de uma sala, colocando-se uma fonte de radiação infravermelha de teste em um local aleatório dentro da mesma sala e tomando-se a média de tempo percorrido até que o fogo seja encontrado, ao longo de cinco tentativas, em pelo menos três salas diferentes. Isso assegura que o dispositivo tenha tempo suficiente para navegar em salas dentro do tempo alocado de acordo com R001.

R003: O dispositivo deve identificar a presença de um incêndio que esteja em sua linha de visão dentro de 10 segundos. O dispositivo deveria identificar a presença de um incêndio que esteja em sua linha de visão dentro de 2 segundos. Este requisito deverá ser validado colocando-se uma fonte de radiação infravermelha de teste a uma distância aleatória do robô que seja maior que 2 pés (60,96 cm), mas menor que 30 pés (9,14 metros), a uma altura a partir do chão menor que 6 pés (1,83 metros) e medindo-se o tempo que o dispositivo leva para reconhecer a fonte de radiação infravermelha. Incêndios podem ocorrer em locais fora dessa região de detecção, sendo provável, porém, que o robô não seria capaz de contê-los.

R004: O dispositivo deve ter uma chave de desligamento que seja facilmente acessível. Este requisito é validade colocando-se o robô a um ângulo aleatório do indivíduo de teste e tomando-se o tempo necessário para que ele acesse a chave e desligue o dispositivo. O tempo deve ser inferior a 5 segundos.

R005: O dispositivo deve ser capaz de funcionar por 10 minutos após a ativação, sem intervenção humana. O dispositivo deveria ser capaz de funcionar por 15 minutos após a ativação, sem intervenção humana. Este requisito será validado por pelo menos cinco testes dentro de um prédio da *Seattle Pacific University*, com o tempo sendo transcorrido desde o momento em que o robô é ativado até quando o mesmo para de se mover completamente, ou pelo menos 10/15 minutos.

R006: O dispositivo deve receber um sinal do sistema centralizado de alarme de incêndio. Este sinal deverá ter a forma de uma sequência digital padrão TTL (0 – 5 V).

R007: O dispositivo deve estar pronto para operar durante seis meses, sem nenhuma intervenção humana. O dispositivo deveria estar pronto para operar durante 12 meses, sem nenhuma intervenção humana.

3.3.2 Custo (C)

Os requisitos de custo são os seguintes:

C001: O preço de mercadoria vendida (custo de produção, incluindo materiais, mão de obra, empacotamento e entrega ao vendedor) para o robô bombeiro como produto final, excluindo-se modificações no painel de controle de incêndio e sistemas de comunicação, não deve exceder $1500. O preço de mercadoria vendida atribuído ao produto final não deveria exceder $1200 a fim de torná-lo apelativo como um produto comercial. Este requisito deve ser validado considerando-se o preço total na lista de materiais, levando-se em conta preços para produção de 1000 unidades e estimando-se a taxa de salário vigente para o centro de manufatura bem como os custos de empacotamento e entrega.

3.3.3 Interface de Usuário e Operação (U)

Os requisitos de interface são os seguintes:

U001: O produto final não requererá nenhum aparato externo para programação após a configuração inicial.

U002: Qualquer chave de desativação deve estar claramente visível a 25 pés de distância (7,62 metros) caso o campo de visão esteja desobstruído.

U003: O dispositivo deve ser facilmente detectado visualmente a 150 pés de distância (45,72 metros, aproximadamente o comprimento de um andar do OMH) sem grandes obstruções. O dispositivo deveria ser facilmente detectado visualmente a 300 pés de distância (91,44 metros) sem grandes obstruções.

U004: O dispositivo deve ser facilmente detectado visualmente a 50 pés de distância (15,24 metros) através de fumaça moderada. O dispositivo deveria ser facilmente detectado visualmente a 100 pés de distância (30,48 metros) através de fumaça moderada. Este requisito garante que os bombeiros terão tempo para reagir à presença do dispositivo.

U005: O dispositivo deve ser facilmente detectado auditivamente a 150 pés de distância (45,72 metros) se nenhum outro som acima de 50 dB estiver presente. O dispositivo deveria ser facilmente detectado auditivamente a 300 pés de distância (91,44 metros) se nenhum outro som acima de 50 dB estiver presente.

3.3.4 Requisitos Mecânicos (M)

Os requisitos mecânicos são os seguintes:

M001: O dispositivo deve ter as dimensões máximas de 24” x 24” x 36” (60,96 cm x 60,96 cm x 91,44 cm). O dispositivo deveria ter as dimensões máximas de 18” x 18” x 28” (45,72 cm x 45,72 cm x 71,12 cm). Isso permite que ele seja instalado facilmente dentro de prédios sem que haja necessidade de se alterar sua estrutura.

M002: O robô deve ser capaz de suportar qualquer extintor de pó químico seco ABC com peso total menor que 10 libras (4,54 kg), raio de menos de 4.5” (11,43 cm) e altura menor que 20” (50,8 cm) porém maior que 8” (20,32 cm). O robô deveria ser capaz de suportar qualquer extintor de incêndio com peso total menor que 15 libras (6,80 kg), raio de menos de 6” (15,24 cm) e altura menor que 24” (60,96 cm) porém maior que 6” (15,24 cm).

3.3.5 Requisitos Elétricos (E)

Os requisitos elétricos são os seguintes:

E001: O dispositivo terá uma bateria interna para uso fora do local onde ele é estacionado.

E002: O dispositivo deveria ter um conector localizado na sua parte inferior que serviria como interface entre a bateria e o carregador, permitindo a recarga da bateria enquanto ele estiver estacionado.

3.3.6 Requisitos de Programação (P)

Os requisitos de programação são os seguintes:

P001: A programação do robô será feita utilizando-se as práticas mais adequadas e uma linguagem de programação amplamente reconhecida.

3.4 Riscos e Plano de Mitigação

Praticamente, qualquer dispositivo eletromecânico pode causar riscos - em maior ou menor escala - à integridade física de quem o opera. No contexto de um projeto, além do mencionado, a palavra risco adquire outros sentidos como possibilidade de falha de operação ou mesmo de que os custos ultrapassem os valores pré-fixados. Sendo assim, é crucial que durante o desenvolvimento do projeto esses riscos sejam avaliados e de alguma forma medidos e que haja planos para sua eliminação ou pelo menos atenuação. As tabelas seguintes exibem os riscos considerados relevantes e alternativas para mitigá-los.

Tabela 3.1 - Análise de riscos.

Risco	Probabilidade	Severidade	Código de Análise de Risco (CAR)
Atingir/impedir civis em fuga	4	4	16
Eletrocutar civis	1	4	4
Superaquecimento da bateria	1	5	5
Falha de energia	1	3	3
Falha de ativação	1	3	3
Adentrar a área de fogo	3	2	6
Perder-se durante navegação	3	2	6
Falha nos sensores	3	2	6
Extinção inadequada	3	1	3
Falha em cumprir prazos	2	2	4
Custo do projeto exceder o orçamento	3	1	3
Produto sem apelo comercial	2	2	4
Obstrução por obstáculos	2	1	2

*Na escala de probabilidade, o valor 1 refere-se a um risco de 1:20000; o valor 2, 1:2000; o valor 3, 1:200; o valor 4, 1:20 e o valor 5, 1:2. Na escala de severidade, 1 corresponde a um atraso de menos de 10 segundos; 2, a um atraso de menos de 1 minuto; 3, a robô inoperante; 4, a humanos feridos e 5, a humanos mortos ou fogo criado/ampliado.

Tabela 3.2 - Plano de mitigação de riscos.

Risco	Mitigação	Probabilidade Pós-Mitigação	Severidade Pós-Mitigação	CAR Pós-Mitigação
Atingir/impedir civis em fuga	Instalação de luzes/sirenes	2	3	6
Eletrocutar civis	-	1	4	4
Superaquecimento da bateria	-	1	5	5
Falha de energia	-	1	3	3
Falha de ativação	-	1	3	3
Adentrar a área de fogo	Teste intensivo, ajustes	1	2	2
Perder-se durante navegação	Teste intensivo, ajustes	1	2	2
Falha nos sensores	Redundância	1	2	2
Extinção inadequada	-	3	1	3
Falha em cumprir prazos	-	2	2	4
Custo do projeto exceder o orçamento	-	3	1	3
Produto sem apelo comercial	-	2	2	4
Obstrução por obstáculos	-	2	1	2

3.5 Materiais e Métodos

Antes que uma lista completa de materiais pudesse ser compilada, deveriam ser definidos o microcontrolador a ser utilizado, as funções atribuídas à placa de circuito impresso a ele conectada, o método de navegação a ser empregado, o formato do chassi e o tipo de sistema de direção, a maneira de detectar o fogo e a distribuição dos sensores ao longo do corpo do robô. Dos acima citados, apenas o tipo de navegação foi definido e permaneceu inalterado. Os demais sofreram modificações ao longo do processo de desenvolvimento e as escolhas serão examinadas ao longo desta seção. Uma lista completa dos materiais adquiridos para a construção do robô é apresentada no Apêndice B.

O principal método utilizado pelo grupo durante a etapa de desenvolvimento foi o de intensa discussão e planejamento de um protótipo sobre o qual poderiam ser feitos testes que levariam ao refinamento do mesmo e orientariam modificações e correções do projeto inicial. Assim, inúmeras versões de protótipo tomaram forma ao longo de várias iterações. A fim de se ter uma noção concreta da aparência real do robô, foram criados modelos com o programa CAD *SolidWorks* desenvolvido pela *Dassault Systèmes SolidWorks Corporation* até se obter uma versão final e definitiva a ser construída.

A primeira versão foi elaborada exclusivamente por um membro do grupo de forma prévia, na tentativa de estabelecer um ponto de partida para a primeira reunião sobre o projeto. O modelo apresenta uma base motora omnidirecional com três pares de rodas, separados por 120 graus, conforme mostra a Figura 3.1. O chassi, cujo formato é hexagonal, é formado por duas placas paralelas, com três sensores infravermelhos montados sobre a placa superior e três pares de sensores ultrassônicos espalhados pelas laterais, posicionados entre as duas placas. Como não houve uma pesquisa extensa previamente à sua concepção e uma participação de todos os membros do grupo, esta versão foi rapidamente descartada, dando lugar a outros designs mais aprimorados.

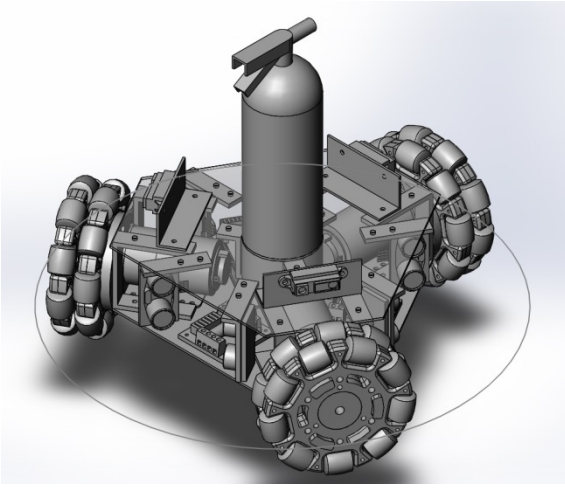

Figura 3.1: Robô CERBERUS (versão 1).

Na versão 2, conforme ilustra Figura 3.2, houve alteração no formato da placa superior de hexagonal para eneagonal, com o objetivo de aumentar a distância de separação entre os sensores ultrassônicos, diminuindo a interferência entre seus cones de detecção. As caixas brancas representam o espaço reservado para componentes eletrônicos como microcontrolador e controlador de motores. A estrutura é composta inteiramente de alumínio, para resistir ao derretimento pelo calor excessivo próximo às chamas. Em relação às rodas, vários materiais resistentes ao fogo foram investigados e

chegou-se à conclusão de que seus coeficientes de atrito e custos inviabilizariam seu uso, sendo preferidas rodas de borracha. Caso estas sofressem de desintegração quando próximas ao incêndio, ou estaria-se diante de um cenário em que o robô encontra-se bem perto do local do fogo, permitindo a execução de sua função e, portanto, não significando um grande impasse, ou de um cenário em que o fogo atingiu uma dimensão tão grande que não poderia mais ser extinto pelo dispositivo.

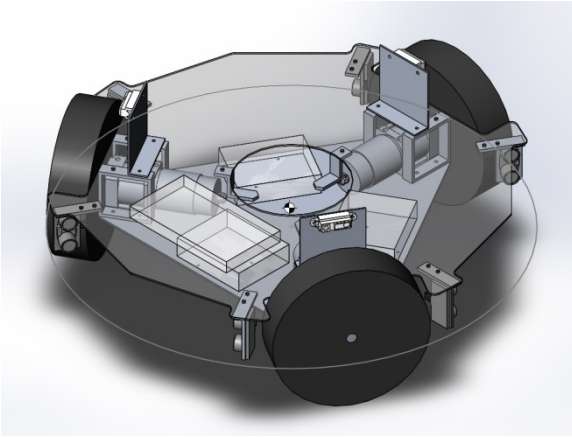

Figura 3.2: Robô CERBERUS (versão 2).

Em sua terceira versão, ilustrada na Figura 3.3, a mudança mais dramática foi a substituição do sistema motor omnidirecional de três eixos pelo coaxial com duas rodas. A alteração foi motivada principalmente pela redução de custos e complexidade. Aquele proposto inicialmente demandaria maiores esforços computacionais, aumentando o grau de dificuldade na produção do software, enquanto este, mais simples, é de fácil controle, diminuindo o tempo gasto na confecção do código de programa. Neste modelo, foram integrados alguns componentes secundários como a luz estroboscópica e a sirene piezoelétrica, além de itens importantes como a bateria, alocada no centro da placa inferior, e a câmera frontal.

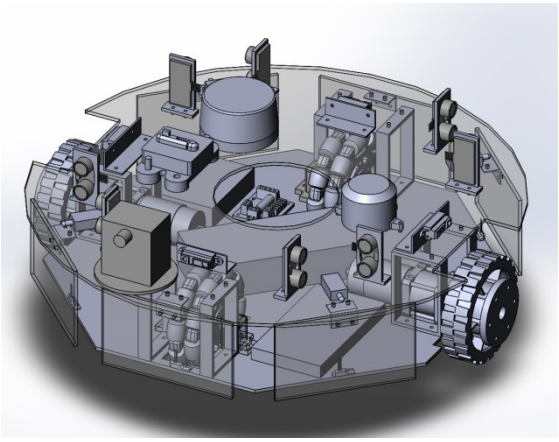

Figura 3.3: Robô CERBERUS (versão 3).

O sistema de navegação escolhido foi o de orientação por faixas-guias coloridas fixadas ao chão, sendo cada cor associada a uma zona de incêndio. Para isso, seria ideal que a câmera fosse simples, sendo capaz apenas de distinguir e identificar cores, e barata. Preenchendo ambos os requisitos, a Pixy CMUcam5, em conjunto com um sistema de ajuste de posição, foi selecionada. Dentre as opções rejeitadas e os motivos pelos quais foram deixadas de lado, destacam-se: etiquetas RFID fixadas nas paredes, por limitação do alcance de comunicação a poucos centímetros; fios subterrâneos guias, pela inconveniência causada pela instalação dos mesmos, requerendo remodelação do piso no local de uso do robô; teleoperação (controle remoto), por não conferir autonomia, necessitando intervenção humana por um operador; mapas programados em memória ROM, por configurar um método extremamente complicado.

Adicionalmente, foram feitas escolhas preliminares em relação a componentes eletrônicos e motores a serem utilizados, permitindo uma análise do consumo de bateria e tempo de autonomia. Foi decidido a favor de motores CC com escovas, uma vez que requerem um sistema de controle mais simplificado e possuem custo mais compatível com o orçamento da equipe, sendo possível adquirir motores bem mais robustos em termos de torque e velocidade máximos. Nesta versão, foram incluídas também a chave de desligamento, duas rodas livres traseiras para aumento da estabilidade e as placas de desativação, que ao serem chutadas, paralisam o robô imediatamente. A ideia desse sistema surgiu depois da constatação pelos bombeiros de que seria custoso para eles, vestindo roupas pesadas e pouco flexíveis, terem que se abaixar para desligar o robô através da chave de desligamento localizada na placa superior. A disposição dos sensores for rearranjada de forma a garantir não só redundância como também cobertura em todos os quatro lados, simétricos rotacionalmente. Cinco sensores de chama, cujos cones de detecção estão representados em rosa na Figura 3.4, foram adicionados como forma primária de reconhecimento do fogo. Na placa inferior do chassi, nota-se a organização de componentes eletrônicos e da bateria, com alguns espaços ainda vazios.

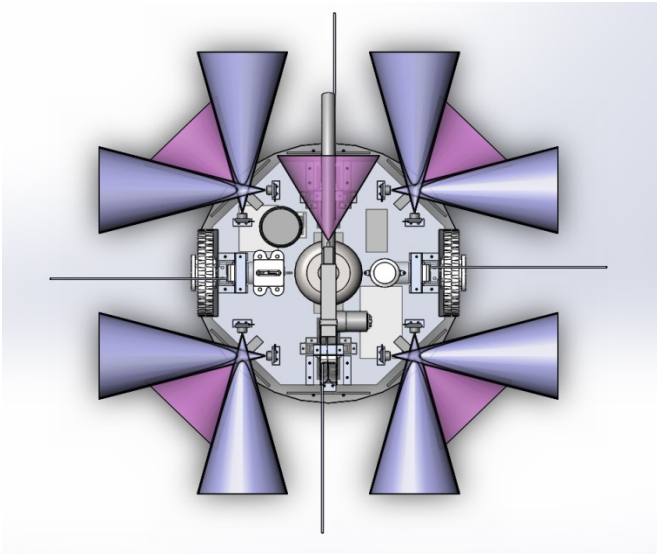

Figura 3.4: Robô CERBERUS (versão 3) com cones de detecção.

Na versão 4, ilustrada na Figura 3.5, o sistema de extinção de fogo foi aprimorado, com a inserção de moldes - duas peças em U conectadas por parafuso e rosca - feitos em impressora 3D para colocação do sensor de chamas próximo à extremidade da mangueira do extintor de incêndio e de uma barra de metal interligando a mesma ao motor de atuação. Foi fabricado o suporte para o extintor de incêndio, composto por duas barras laterais de alumínio presas a duas peças canal L através das quais passam abraçadeiras de cano do tipo cinta perfurada. No que tange à ativação do extintor, uma extensão foi adicionada ao seu gatilho com o intuito de reduzir o torque necessário para pressioná-lo. Um cabo de aço galvanizado conecta a extensão no gatilho ao motor atuador localizado na placa superior do chassi. Outras transformações notáveis foram o aumento no tamanho das rodas, de quatro para seis polegadas, inclusão de suporte para a câmera de navegação para que ela não obstruísse o campo de visão de um dos sensores infravermelho, anexação da primeira versão da placa de circuito impresso, posicionada no quadrante traseiro-esquerdo sobre a placa inferior do chassi e do painel de controle de incêndio. Este tem como papel simular a comunicação do robô com o sistema de alarme central do prédio, já que seria difícil uma intervenção para adaptação do mesmo às necessidades de teste do projeto CERBERUS. É composto simplesmente por quatro botões, cada um representando uma zona de incêndio fictícia.

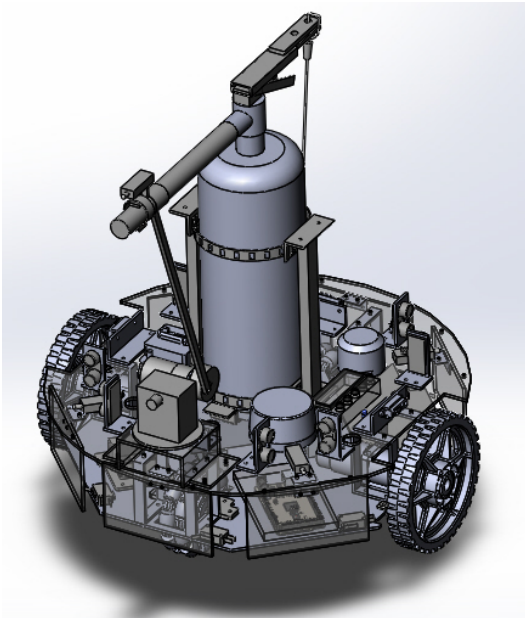

Figura 3.5: Robô CERBERUS (versão 4).

Comparada à versão anterior, a quinta versão apresenta poucas diferenças conforme ilustra a Figura 3.6. A maior delas é na disposição dos sensores ultrassônicos, que passou de vertical para horizontal. Inicialmente, foi feita uma interpretação, a partir da leitura da documentação técnica, de que os cones de detecção desses sensores tinham formato circular. Mais tarde, foi verificado que os cones, na realidade, tinham formato elíptico. Por isso, foi necessária a mudança a fim de evitar interferências entre os sinais de sensores adjacentes. Secundariamente, outra modificação foi a incorporação de furos na placa superior para passagem dos fios que conectam os dispositivos no topo desta ao microcontrolador e à placa de circuito impresso que se encontram entre as placas do chassi.

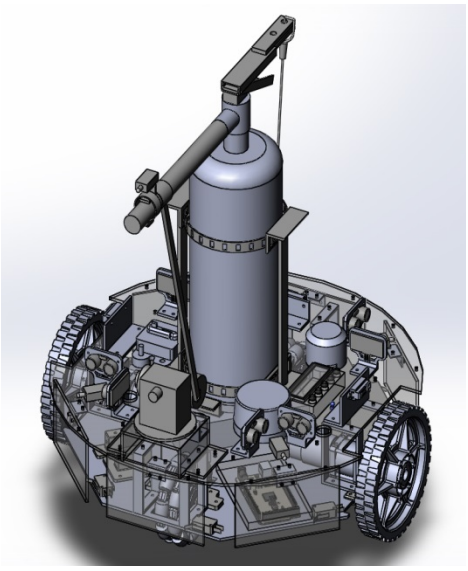

Figura 3.6: Robô CERBERUS (versão 5).

Na versão 6, conforme ilustra a Figura 3.7, foram redesenhados alguns componentes do sistema de extinção, a plataforma de apoio da câmera de navegação e a placa de circuito impresso, que aumentou significativamente de tamanho, obrigando uma reorganização dos elementos internos que culminou na realocação da mesma, movendo-se do quadrante traseiro-esquerdo para o dianteiro-esquerdo. Originalmente, seriam utilizados cinco sensores de chama para detecção do fogo. No entanto, percebeu-se que seu alcance e sensitividade seriam fatores limitantes, o que suscitou a busca por um sensor alternativo. Durante uma pesquisa online, foi encontrado um kit de conversão para a câmera Pixy CMUcam5, já utilizada para navegação, permitindo a troca de sua lente original por outra com filtro infravermelho, tornando-a capaz de identificar chamas. Uma segunda Pixy CMUcam5 portanto foi adquirida, tomando o lugar do sensor de chamas na extremidade da mangueira do extintor. Com isso, a comunicação com o microcontrolador, que antes era feita por protocolo SPI através de sua porta ICSP, passou a ser realizada por meio do protocolo I2C, ambas as câmeras compartilhando o mesmo barramento.

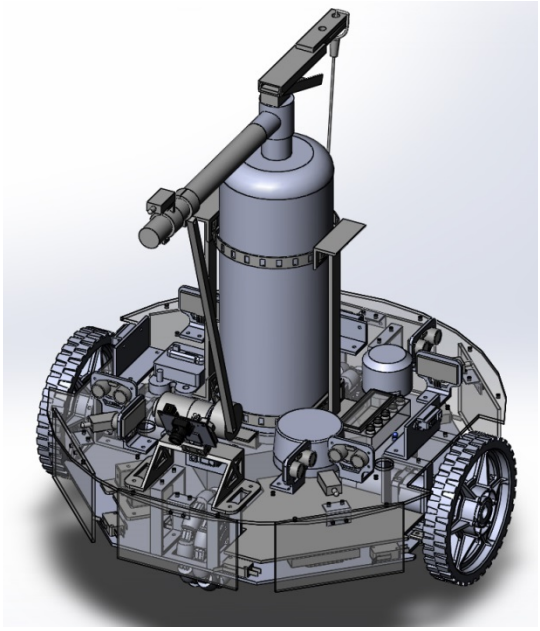

Figura 3.7: Robô CERBERUS (versão 6).

Na sétima e última versão, conforme ilustra a Figura 3.8, o sistema de extinção sofreu algumas alterações para maior eficiência e robustez: as peças canal L foram trocadas por canal U, sendo rebitadas às barras metálicas laterais, que inicialmente foram mantidas presas às peças por meio de cola epóxi (JB *Weld*). Além disso, os moldes foram refabricados e uma espécie de armação foi colocada na mangueira conferindo maior poder de sustentação e rigidez, tornando o mecanismo de atuação dos motores mais robusto.

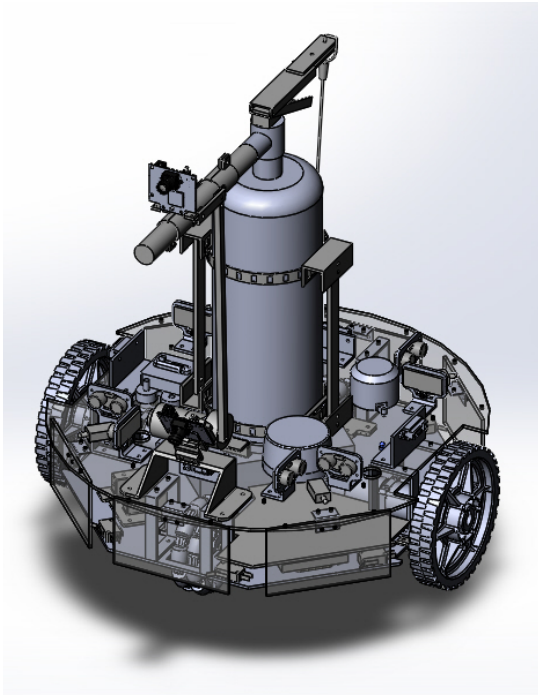

Figura 3.8: Robô CERBERUS (versão 7).

A figura 3.9 ilustra a versão final de protótipo, destacando-se os componentes eletrônicos dos principais sistemas do robô CERBERUS.

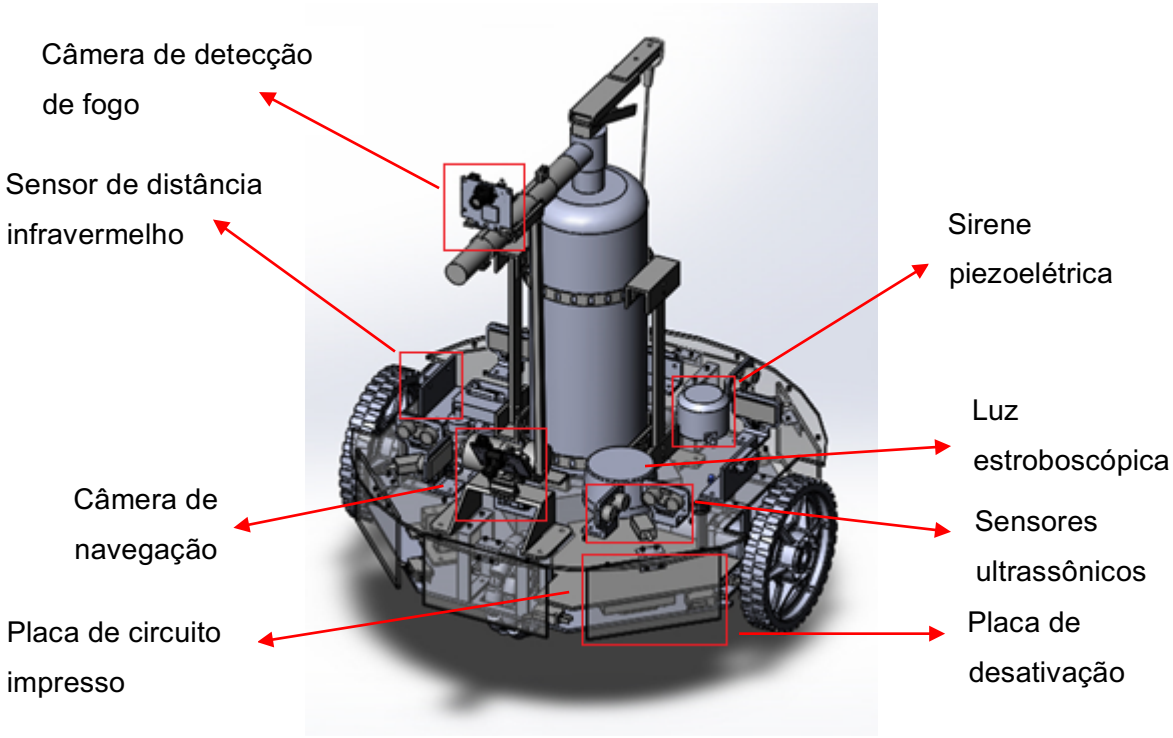

Figura 3.9: Principais componentes do robô CERBERUS.

3.6 Software

O robô bombeiro autônomo CERBERUS consiste em um projeto altamente eletromecânico, porém todas as partes mecânicas e elétricas devem ser controladas via software. Sendo assim, é fundamental a presença de uma unidade de processamento, representada por um microcontrolador, que possa controlar basicamente todas as funções e sistemas do robô, como o motor, os sensores e o sistema de extinção de fogo. O programa não só o auxilia a encontrar sua rota através de fitas coloridas no chão, mas também processa todas as informações sobre o ambiente que respaldarão as tomadas de decisão, determinando onde está o fogo e acionando o sistema de extinção no momento certo.

O planejamento e a implementação do código de programação tiveram como base o documento de arquitetura de software, preparado durante a fase inicial do projeto, logo após a escrita dos documentos de especificações técnicas. Em linhas gerais, este documento previa a divisão do software em diversos procedimentos específicos, elaborados separadamente e posteriormente integrados em um único arquivo principal. A única ferramenta aplicada para o desenvolvimento e gravação do código foi a plataforma de desenvolvimento integrado Arduino IDE.

3.6.1 Visão Geral da Estrutura do Software

Com o propósito de poupar energia a ter uma resposta rápida, o sistema funciona a partir de interrupções. Um laço de repetição é executado deixando o programa em modo ocioso até que um botão é pressionado no painel de controle de incêndio. Como mencionado anteriormente, serão disponibilizados quatro botões no painel, cada um simulando uma área (ou zona) do prédio. Assim que o botão é acionado, um sinal é enviado ao microcontrolador que prontamente inicia o procedimento de seguir uma das linhas coloridas no chão, de acordo com a zona. Em uma futura aplicação no mundo real, o sinal deverá ser recebido através de comunicação sem fio entre o robô e o sistema central de alarme de incêndio. Ao chegar ao final da linha, é executado o algoritmo de busca pelo fogo, que ao ser encontrado, faz com que o programa inicie a rotina de extinção de incêndio.

3.6.2 Principais Rotinas do Programa

A rotina principal do software é aguardar por um sinal de interrupção, que quando recebido, desencadeia a execução das demais rotinas descritas a seguir:

3.6.2.1 lineFollowing (int color)

- Propósito: seguir a linha de determinada cor, que guiará o robô até a área desejada.
- Entradas: Pixy CMUcam5 (câmera), sensores ultrassônicos e infravermelhos
- Saída: controlador de motores.

- Algoritmo: o algoritmo recebe dados dos sensores de distância e os processa a fim de controlar os motores, direcionando o robô até o seu destino, acompanhando a linha correta e desviando de possíveis obstáculos.
- Restrições: devido à quantidade significativa de sensores sendo utilizados, o fluxo de informações que chegam ao processador pode ser grande, aumentando o tempo de processamento. A câmera pode demorar para processar a imagem da linha, fazendo com que fique difícil para o robô manter a trajetória ao fazer curvas.

3.6.2.2 fireSearch ( )

- Propósito: encontrar o local do fogo e guiar o robô até ele.
- Entradas: Pixy CMUcam5 (câmera), sensores ultrassônicos e infravermelhos.
- Saídas: controlador de motores e sistema de ajuste de posição da câmera.
- Algoritmo: inicialmente, o algoritmo recebe dados da câmera com filtro infravermelho e os processa. O sistema de ajuste de posição a coloca em posição reta verticalmente (inclinação igual a 0 graus) e a gira 180 graus lateralmente, varrendo toda a área. Se há reconhecimento de fogo no local, o robô entra na sala e inicia a navegação dentro dela, utilizando o algoritmo de desvio de obstáculos e os dados da câmera para aproximação.
- Restrições: a luz ambiente pode interferir na identificação do fogo através da câmera.

3.6.2.3 fireExtinguisher ( )

- Propósito: ativar o sistema de extinção de fogo.
- Entrada: câmera de detecção de fogo.
- Saída: controlador de motores.
- Algoritmo: através dos dados recebidos da câmera com filtro infravermelho, tem-se dimensão da distância em relação ao fogo. Quando é determinado pelo processador que a distância é adequada, um sinal de PWM é enviado ao controlador de motores, acionando os motores atuadores, um deles responsável por ajustar a mangueira e o outro por pressionar o gatilho no extintor de incêndio.
- Restrições: dificuldade de programação do procedimento de ajuste da posição da mangueira a partir dos dados do encoder do motor atuador.

3.6.3 Drivers de dispositivo

O objetivo desta seção é descrever a interface entre o principal sistema de computação do robô e os componentes do hardware como sensores, câmera, luz, sirene e controladores de motores. O propósito principal da utilização de drivers é permitir à unidade de processamento (microcontrolador) comunicar-se com os dispositivos periféricos através de rotinas de software.

3.6.3.1 Luz e Sirene

1. Visão geral

   A luz e a sirene requerem um fio único para comunicação com o microcontrolador. Para ligá-las ou desligá-las, ambas estão conectadas a um circuito de chaveamento controlado pelo microcontrolador.
2. Interface de hardware
   a. Pino A12: saída digital
3. Rotinas do driver
   a. Buzzer_Lights_On
      i. Protótipo: digitalWrite(buzzerLightsPort, HIGH);
      ii. Descrição: ativa a luz e a sirene
   b. Buzzer_Lights_Off
      i. Protótipo: digitalWrite(buzzerLightsPort, LOW);
      ii. Descrição: desativa a luz e a sirene

3.6.3.2 Controladores de motores

1. Visão geral

   Os controladores de motores requerem um sinal de largura de pulso modulada e de referência provenientes do microcontrolador.
2. Interface de hardware
   a. Pinos 10 a 13: saída de PWM
   b. GND: referência de tensão (terra)
3. Rotinas do driver

   A biblioteca “Servo.h” é utilizada para programação dos controladores de motores.
   a. Motor_Define
      i. Protótipo: servo.attach(pin, min, max);
      ii. Descrição: atrela a variável Servo a um pino do microcontrolador. – min: é a largura de pulso, em microssegundos, correspondente ao ângulo mínimo no motor;
      - max: é a largura de pulso, em microssegundos, correspondente ao ângulo máximo no motor;
   b. Motor_Write
      i. Protótipo: servo.write(angle);
      ii. Descrição: controla o eixo do motor de acordo com o valor de “angle”. Em um servo-motor, seleciona o ângulo do eixo em graus. Em um motor de rotação contínua, seleciona a velocidade do motor, sendo 0

associado a velocidade máxima em uma direção, 180, a velocidade máxima na direção contrário e 90, a ausência de movimento.

4. Estruturas de dados e variáveis
    a. Servo
        i. leftMotor
        ii. rightMotor
        iii. panTilt
        iv. extinguisherMotor

3.6.3.3 Pixy CMUcam5

1. Visão Geral

   A Pixy CMUcam5 é uma câmera que pode se comunicar com o microcontrolador através dos protocolos SPI e I2C ou de um sinal analógico.
2. Interface de hardware
    a. SDA: sinal digital de dados, transmitido de forma serial
    b. SCL: *clock* serial
    c. 5V: tensão de alimentação do circuito
    d. GND: referência de tensão (terra)
3. Rotinas do driver

   A câmera Pixy CMUcam5 é controlada por intermédio da biblioteca “Pixy.h”.
    a. Pixy_getBlocks
        i. Protótipo: pixy.getBlocks();
        ii. Descrição: retorna o número de objetos detectados pela câmera
    b. Pixy_Signature
        i. Protótipo: pixy.blocks[i].signature;
        ii. Descrição: número de assinatura do objeto detectado (1-7)
    c. Pixy_X
        i. Protótipo: pixy.blocks[i].x;
        ii. Descrição: posição horizontal do centro do objeto detectado (0 a 319)
    d. Pixy_Y
        i. Protótipo: pixy.blocks[i].y;
        ii. Descrição: posição vertical do centro do objeto detectado (0 a 199)
    e. Pixy_Width
        i. Protótipo: pixy.blocks[i].width;
        ii. Descrição: largura do objeto detectado (1 a 320)
    f. Pixy_Height
        i. Protótipo: pixy.blocks[i].height;

ii. Descrição: altura do objeto detectado (1 a 200)

4. Estruturas de dados e variáveis
   a. Pixy
      i. pixy

3.6.3.4 Sensores ultrassônicos

1. Visão geral

   Emitem uma onda de ultrassom quando ativados através de seu pino *trigger* e recebem a onde refletida, transmitindo um sinal ao microcontrolador por meio de seu pino denominado *echo*.

2. Interface de hardware
   a. Pinos 22-37: entrada/saída digital (pinos pares correspondem aos *triggers* e pinos ímpares, aos *echoes*)
3. Rotinas do driver

   Para controlar os sensores ultrassônicos, é utilizada a biblioteca “NewPing.h”.

   a. Constructor
      i. Protótipo: NewPing sonar(trigger_pin, echo_pin, max_cm_distance);
      ii. Descrição: inicializa a biblioteca NewPing, associando a variável “trigger_pin” ao pino *trigger* do sensor, a variável “echo_pin” ao pino *echo* do sensor e definindo opcionalmente a distância máxima em centímetros através da variável max_cm_distance (o valor padrão, caso ela não seja definida, é de 500 cm)
   b. Get_Distance
      i. Protótipo: sonar.ping_cm();
      ii. Descrição: retorna a distância, em centímetros, do sensor em relação ao objeto identificado. Se não houver detecção da onda refletida, a função retorna o valor 0.
4. Estruturas de dados e variáveis
   a. NewPing:
      i. US1-US8

3.6.3.5 Sensores infravermelhos

1. Visão geral

   O sensor de distância infravermelho emite um feixe de luz infravermelha e recebe a onda refletida, transmitindo ao microcontrolador um sinal analógico que é proporcional à distância entre o sensor e o objeto refletor.

O sensor infravermelho de chama é sensibilizado por radiação cujo comprimento de onda encontra-se na faixa de 760 a 1100 nanômetros e envia um sinal analógico ao microcontrolador que corresponde à intensidade da chama.

2. Interface de hardware
    a. Pinos A0-A4: entradas analógicas (sensores de chama)
    b. Pinos A5-A8: entradas analógicas (sensores de distância)
3. Rotinas do driver
    a. Read_Distance
        i. Protótipo: readDistance(IRdistanceSensor);
        ii. Descrição: lê os valores de saída dos sensores infravermelhos e os converte no valor da distância correspondente em centímetros.
    b. Read_Flame
        i. Protótipo: readFlame(IRflameSensor);
        ii. Descrição: lê os valores de saída dos sensores de chama e os converte em um valor relativo à intensidade do fogo.
4. Estruturas de dados e variáveis
    a. Int
        i. flame1-flame5
        ii. IRdist1-IRdist4

3.6.3.6 Chaves fim de curso

1. Visão geral

   Quando pressionadas, as chaves de fim de curso enviam um sinal de 5 volts ao microcontrolador, ativando uma interrupção responsável por desativar o sistema motor do robô.
2. Interface de hardware
    a. Pino 2: porta de interrupção
3. Rotinas do driver
    a. Interruption
        i. Protótipo: attachInterrupt(interrupt, ISR, mode);
        ii. Descrição: aguarda até que um sinal seja recebido em um pino específico. Quando isso acontece, executa uma rotina e retorna à função principal.

            - interrupt: o número da interrupção (variável int)

            - ISR: rotina a ser chamada quando a interrupção ocorre. Esta função não deve aceitar parâmetros nem retornar nenhum valor

- mode: define o momento em que a interrupção deve ser engatilhada. Pode assumir os valores LOW (interrupção em nível baixo), CHANGE (interrupção em transição de estado), RISING (interrupção em borda de subida) e FALLING (interrupção em borda de descida).

4. Estruturas de dados e variáveis
   a. int:
      i. kickPlates

3.6.4 Controlador PID

A primeira rotina descrita, caracterizada por controlar os motores de forma a fazer com que o robô siga a trajetória descrita pelas faixas coloridas no solo, envolve a utilização de um método de controle. Neste projeto, foi lançado mão do algoritmo de um controlador proporcional integral derivativo (PID) otimizado, para Arduino, proposto por Brett Beauregard em seu blog (BEAUREGARD, 2011). Parte-se da seguinte equação básica de controle PID:

$$Saída = K_P e(t) + K_I \int e(t)dt + K_D \frac{d}{dt} e(t) \qquad (1)$$

onde o erro $e = Ponto\ de\ ajuste - Entrada$

Para o cálculo da fórmula no microcontrolador, são feitas algumas aproximações. O diferencial de tempo “dt” é substituído pelo intervalo de tempo decorrido entre duas chamadas consecutivas da função *Compute* (valor atribuído à variável *timeChange*). Assume-se que esse tempo seja infinitamente pequeno devido à velocidade extremamente alta do processador e no contexto em que *Compute* esteja dentro de um laço de repetição. Sendo assim, a integral do erro transforma-se na soma acumulada do resultado da multiplicação do erro atual por *timeChange*, e a derivada, na divisão entre a diferença dos erros calculados nas duas últimas chamadas da função *Compute* e *timeChange*. Tem-se então o equivalente da equação matemática (1) em formato de código de programação:

Tabela 3.3: Código de programa referente ao controlador PID básico.

```
1   /*working variables*/
2   unsigned long lastTime;
3   double Input, Output, Setpoint;
4   double errSum, lastErr;
5   double kp, ki, kd;
6   void Compute()
7   {
8     /*How long since we last calculated*/
9     unsigned long now = millis();
10    double timeChange = (double)(now - lastTime);
11
12    /*Compute all the working error variables*/
```

```
13    double error = Setpoint - Input;
14    errSum += (error * timeChange);
15    double dErr = (error - lastErr) / timeChange;
16
17    /*Compute PID Output*/
18    Output = kp * error + ki * errSum + kd * dErr;
19
20    /*Remember some variables for next time*/
21    lastErr = error;
22    lastTime = now;
23  }
24
25  void SetTunings(double Kp, double Ki, double Kd)
26  {
27    kp = Kp;
28    ki = Ki;
29    kd = Kd;
30  }
```

A partir daí, são recomendadas uma série de modificações para aperfeiçoamento da função *Compute*, corrigindo imperfeições ou falhas advindas do modelo anterior. Entre elas, destacam-se o cálculo das variáveis - chamada da função *Compute* - em intervalos de amostra fixos e eliminação do salto derivativo (*derivative kick*), que ocorre quando a derivada do erro (e) é infinita. O código final após as alterações é mostrado abaixo.

Tabela 3.4: Código de programa referente ao controlador PID após as modificações.

```
1   /*working variables*/
2   unsigned long lastTime;
3   double Input, Output, Setpoint;
4   double ITerm, lastInput;
5   double kp, ki, kd;
6   int SampleTime = 1000; //1 sec
7   double outMin, outMax;
8
9   void Compute()
10  {
11    unsigned long now = millis();
12    int timeChange = (now - lastTime);
13    if(timeChange>=SampleTime)
14    {
15      /*Compute all the working error variables*/
16      double error = Setpoint - Input;
17      ITerm+= (ki * error);
18      if(ITerm > outMax) ITerm= outMax;
19      else if(ITerm < outMin) ITerm= outMin;
20      double dInput = (Input - lastInput);
21
22      /*Compute PID Output*/
23      Output = kp * error + ITerm- kd * dInput;
24      if(Output > outMax) Output = outMax;
25      else if(Output < outMin) Output = outMin;
```

```
26
27          /*Remember some variables for next time*/
28          lastInput = Input;
29          lastTime = now;
30        }
31      }
```

O algoritmo acima é adaptado para controlar os motores do robô fazendo com que o mesmo siga a linha colorida no chão. Para isso é necessário manter o objeto (linha) detectado pela câmera sempre na posição central. Como mencionado anteriormente, o método pixy.blocks[i].x representa a posição horizontal do objeto, com os limites inferior e superior iguais a 0 e 319, respectivamente. Configura-se então o ponto de ajuste como 160, exatamente na metade da faixa estabelecida pelos valores acima. A entrada refere-se à posição instantânea do objeto e é dada por pixy.blocks[i].x, enquanto o valor de saída (*Output*) calculado pela função Compute() é adicionado ao valor do parâmetro de entrada da função de configuração do motor da roda direita e subtraído do valor do parâmetro de entrada da função de configuração do motor da roda esquerda. Com isso, os erros ora positivos (quando a linha se encontra à esquerda do centro) ora negativos (linha à direita do centro), imprimem uma velocidade maior ora na roda direita ora na roda esquerda, permitindo ao robô acompanhar a rota descrita pela faixa colorida no solo. A implementação completa do controlador PID no código final do robô, com os valores escolhidos para as constantes K_P, K_I e K_D, é exibida no código presente no Apêndice A.

3.7 Hardware

Esta subseção trata primordialmente do microcontrolador, da placa de circuito impresso e do conjunto de fios que promovem a comunicação entre eles.

Em um primeiro momento, durante os testes inicias das primeiras versões de protótipo, foi empregado o microcontrolador Arduino Mega 2560 (Figura 3.9) devido à disponibilidade de uma grande quantidade de pinos de entrada e saída neste modelo, compatível com o número elevado de sensores de distância presentes no robô, doze no total.

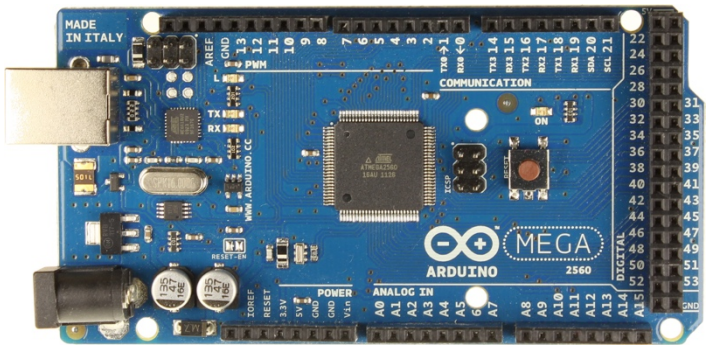

Figura 3.10: Microcontrolador do robô CERBERUS

Os sensores ultrassônicos tiveram seus pinos de *echo* e *trigger* diretamente ligados aos pinos de entrada e saída de propósito geral (GPIOs) do microcontrolador, assim como as chaves de fim de curso. Os nove sensores infravermelhos (quatro de distância e cinco de chama), por sua vez, tiveram seus pinos de saída conectados às entradas analógicas de A0 a A8. A câmera de navegação foi diretamente ligada ao conector ICSP do Arduino, estabelecendo-se uma conexão com base no protocolo SPI.

Para a alimentação de todos os componentes do robô, foi projetada uma placa de circuito impresso que funcionasse, de maneira geral, como um centro de distribuição de energia. A placa tinha como entrada os 14,8 volts da bateria, que eram regulados para 5 e 12 volts, com o objetivo de fornecer a tensão de entrada para os sensores, sirene, luz e controladores de motores, além de possuir circuitos de chave eletrônica com transistor BJT para acionamento da luz e da sirene através de portas de saída digital e um relê de controle que atua como uma chave geral, desligando a alimentação de todos os sensores quando necessário.

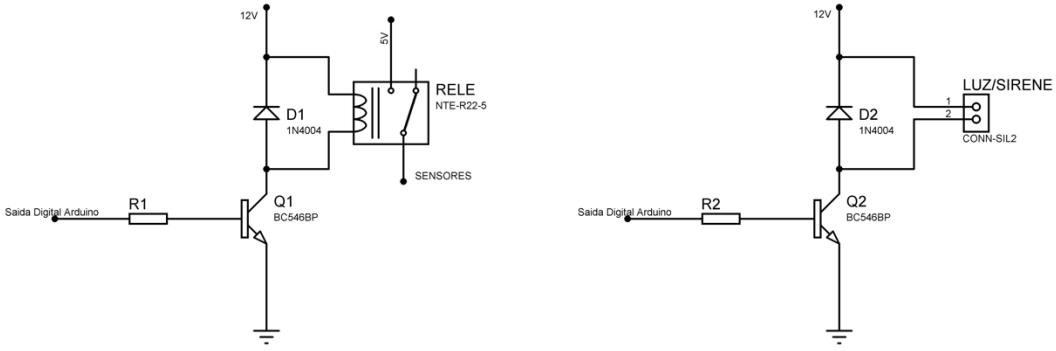

Figura 3.11: Circuito de alimentação do robô CERBERUS

A Figura 3.11 a seguir mostra o formato da placa de circuito impresso, desenhado para preencher exatamente a área de um dos quadrantes da placa inferior da estrutura do robô, e a disposição dos componentes sobre ela.

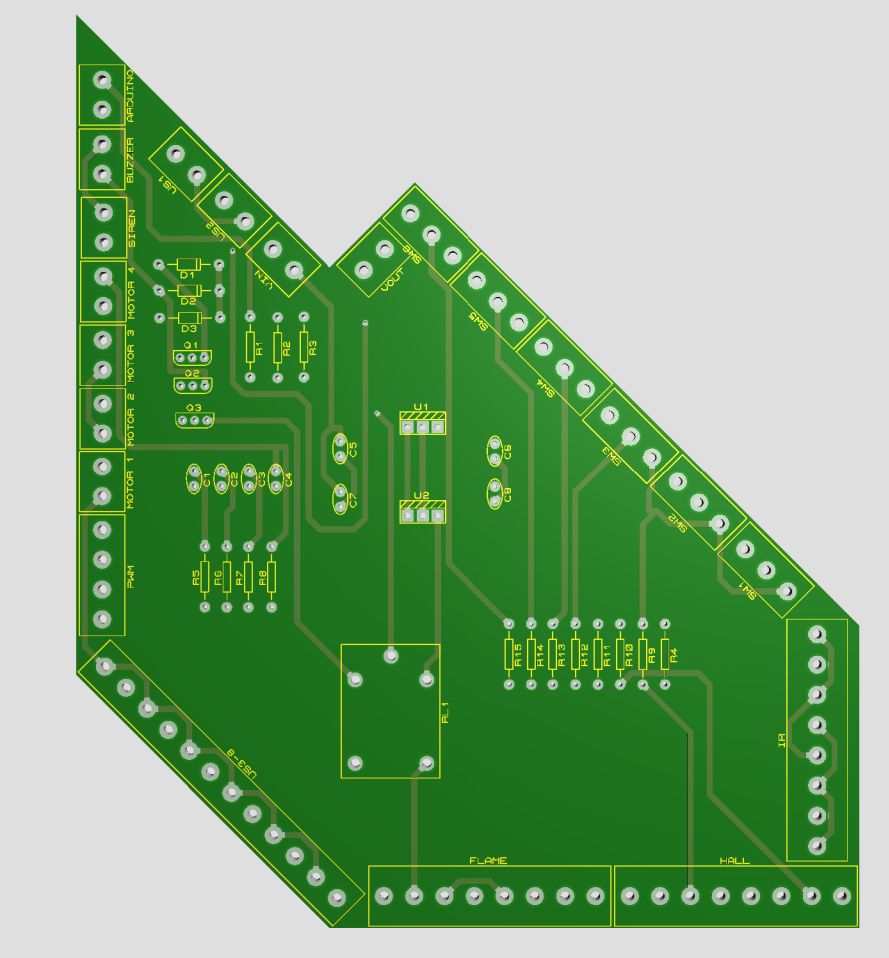

Figura 3.12: Formato da placa de circuito impresso do robô CERBERUS

Os conectores possuem espaçamento padrão entre os pinos de 2,54 milímetros. Os que se referem aos motores (“MOTOR 1”, “MOTOR 2”, ...) representam os sinais de largura de pulso modulada ligados às entradas de PWM nos controladores de motores, ao passo que o de nome “PWM” é conectado às saídas de PWM no microcontrolador. São empregados dois controladores de motores, ambos de canal duplo, um para controle dos dois motores das rodas e o outro, dos dois motores do sistema de extinção de fogo. O primeiro suporta corrente contínua de até 12 A e correntes de pico de até 25 A por poucos segundos; o segundo, corrente contínua de até 5 A, com picos de até 10 A.

Em um segundo momento, em razão da inscrição do projeto CERBERUS na edição de 2015 da competição de sistemas embarcados Intel-Cornell Cup, houve a substituição do microcontrolador Arduino pelo Intel Edison, atendendo a um dos requisitos para participação no evento. Para facilitar o acesso aos pinos de entrada e saída, foi utilizado o Arduino Breakout Kit, exibido na Figura 3.12.

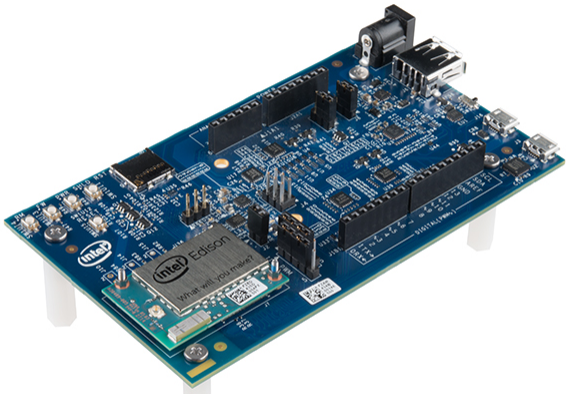

Figura 3.13: Arduino Breakout Kit para o Intel Edison.

Com uma CPU Intel Atom de dois núcleos a 500 MHz e um microcontrolador Quark de 32 bits a 100 MHz, a migração para a plataforma Intel Edison significou um aumento excepcional de poder de processamento frente ao Arduino Mega 2560, que funciona com relógio de 16 MHz. O número de portas de entrada e saída acessíveis no kit, contudo, diminuiu drasticamente em relação ao Arduino Mega, o que resultou na reavaliação do sistema eletrônico a fim de diminuir o número de pinos utilizados pelos sensores.

Devido à mudança de microcontrolador, à adição de componentes e para se adequar aos problemas encontrados durante os testes com a placa citada anteriormente, foi projetada uma nova versão da placa de circuito impresso (Figura 3.13), 60% maior e mais sofisticada. Ao contrário da primeira versão, que funcionava apenas como distribuidora de energia, esta foi desenhada para se encaixar diretamente sobre o kit do microcontrolador, intermediando as ligações entre ele e os periféricos. Consequentemente, o número de fios foi reduzido significativamente, pois passaram a se concentrar exclusivamente entre o conjunto placa-microcontrolador e os periféricos, ocasionando uma melhora na organização do sistema de fiação interno.

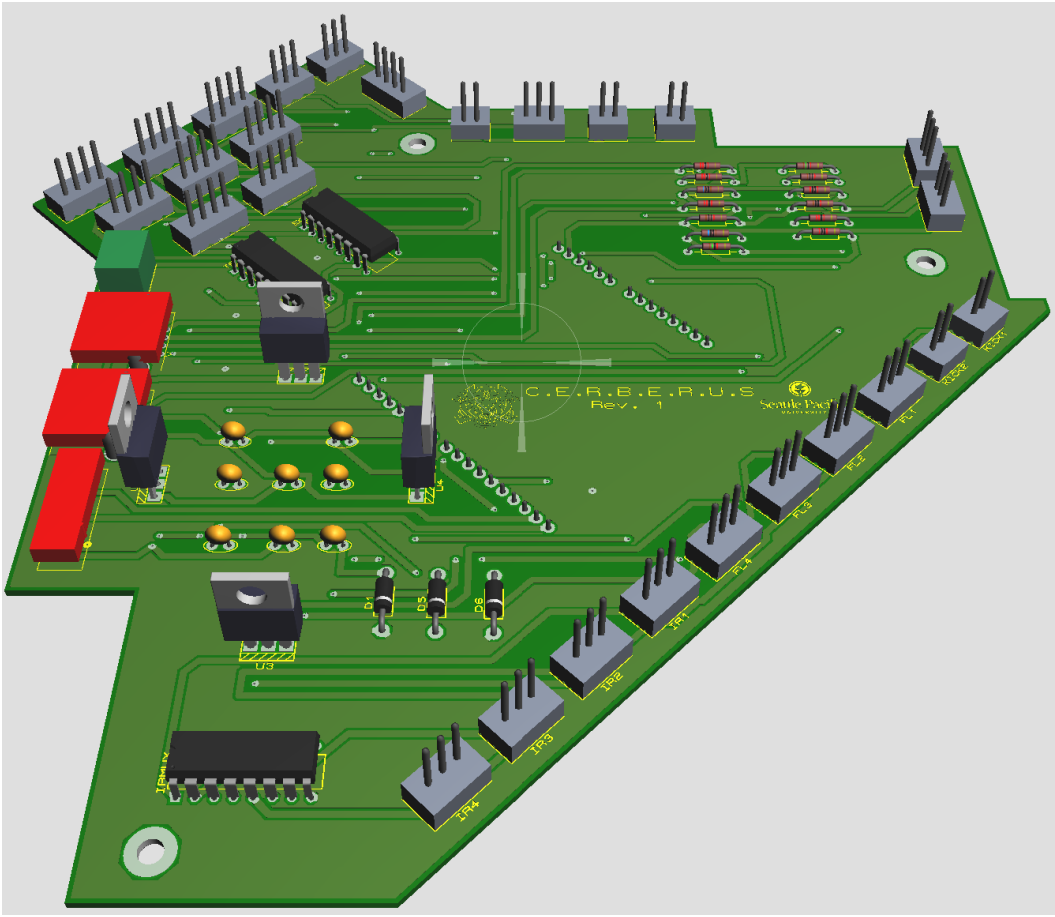

Figura 3.14: Nova versão da placa de circuito impresso usada no robô CERBERUS

Ao longo do período de testes do protótipo com a primeira versão da placa de circuito impresso, foi verificado que o consumo de corrente da câmera com o sistema de inclinação vertical e lateral era superior ao máximo suportado pelo Arduino, não sendo possível a alimentação via porta ICSP. Aliado a isso, foi adicionada uma segunda câmera para detecção de fogo, inviabilizando a comunicação através do protocolo SPI. Optou-se pelo protocolo I2C, que permite a comunicação, utilizando apenas dois fios, entre o microcontrolador e vários dispositivos escravos, compartilhando o mesmo barramento. Cada escravo possui um endereço único pelo qual é identificado durante a troca de dados com o mestre. As duas câmeras foram conectadas à placa por meio de um receptáculo IDC e cabo *flat*, sendo alimentadas através de sua porta mini-USB (Figura 3.14). Na imagem da placa acima, os dois retângulos vermelhos idênticos representam os conectores USB, logo acima do receptáculo IDC. Dois reguladores de tensão de 5 V com capacidade máxima de 2 A de corrente foram empregados para o circuito de alimentação das câmeras.

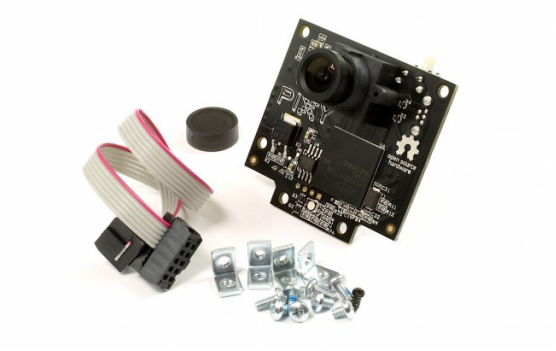

Figura 3.15: Componentes da câmera usada no robô CERBERUS

Para resolver o impasse do número elevado de pinos do microcontrolador sendo usados, foram agregados três multiplexadores, sendo um multiplexador para o sinal de *trigger* dos sensores ultrassônicos, um demultiplexador para o sinal de *echo* e um multiplexador para os sensores analógicos infravermelhos. Isso acabou por solucionar outro problema encontrado na primeira versão: a inconsistência dos dados obtidos a partir dos sensores ultrassônicos por consequência de interferência entre os sinais de sensores adjacentes. Com os multiplexadores, apenas um ultrassônico é acionado por vez.

Nesta nova versão da placa de circuito impresso, não há um relê para corte da alimentação dos sensores, visto que o consumo de corrente é extremamente baixo, não havendo necessidade de racionamento de energia. As chaves eletrônicas de acionamento da luz e da sirene foram removidas, dando lugar a um interruptor para ligamento/desligamento de ambas simultaneamente. Essa mudança foi feita para facilitar os testes e demonstrações, não havendo necessidade de modificação e reprogramação do código para acioná-las toda vez que solicitado. Adicionalmente, foi previsto o uso de um conector para o painel de controle de incêndio, em que cada botão corresponde a um valor de sinal analógico na faixa de 0 a 5 volts. O circuito das chaves de fim de curso foi corrigido em relação à primeira versão, que previa incorretamente o uso de conectores de três pinos, quando apenas dois são suficientes. Alguns componentes da placa acabaram não sendo aproveitados, como é o caso dos sensores de efeito Hall que seriam colocados nas rodas para aferição de velocidade e posição. No centro da placa, nota-se a presença de duas barras de pino paralelas, usadas para o encaixe da mesma sobre o kit do microcontrolador. O conector utilizado, do tipo barra de pino empilhável, é mostrado na Figura 3.15.

Figura 3.16: Conector para o painel de controle do robô CERBERUS

A montagem do esquemático, as simulações, o design e o layout de ambas as placas de circuito impresso foram feitos no software Proteus 8.0 Professional da *Labcenter Electronics*. A primeira versão possuía apenas uma camada de cobre, enquanto a segunda, pelo fato de ser bem mais densa, foi impressa em duas camadas. A fabricação foi encomendada junto à empresa *Advanced Circuits*, com as especificações padrão: material FR-4, espessura da camada de cobre de 1 onça (35 micrômetros) e largura de traço mínimo de 5 mils (5 milésimos de polegada). Todos os componentes são do tipo PTH (*pin through hole*) e foram soldados manualmente.

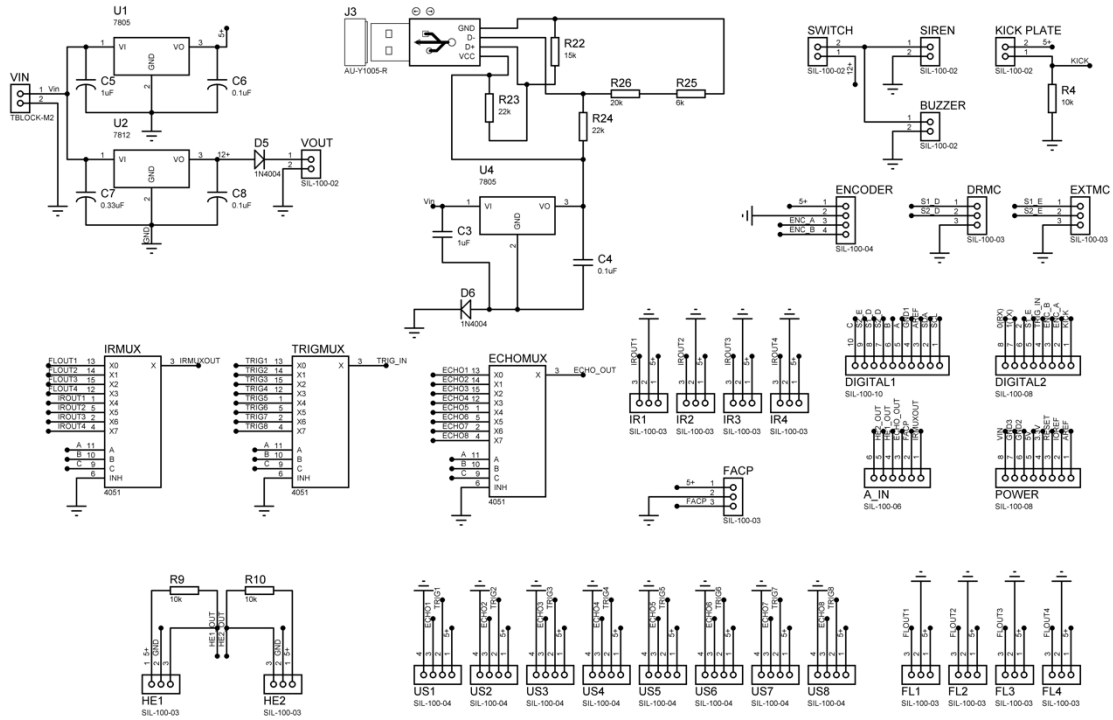

Figura 3.17: Esquemático dos circuitos usados no robô CERBERUS

4. Resultados

Na etapa inicial do projeto, conforme demonstrado na seção 3.3, foram definidos os requisitos funcionais do robô, que refletem não só os limiares mínimos de operação como também as metas a serem possivelmente alcançadas no futuro, após sucessivas iterações e aperfeiçoamentos. Por se tratar de um dispositivo complexo, composto por sistemas mecânicos e eletroeletrônicos elaborados, foi planejado um conjunto de testes que pudessem verificar o funcionamento e desempenho de subsistemas e funções separadamente. Por fim, foi realizada uma demonstração completa, envolvendo navegação, busca pelo fogo, desvio de obstáculos e acionamento do extintor de incêndio. Nas subseções seguintes, são explicados os procedimentos de teste e apresentados os resultados e estatísticas pertinentes. Ao final, é feita uma descrição do que foi observado durante a demonstração completa.

4.1 Resultados de Testes

A seguir são apresentados os resultados de teste realizados para investigação de algumas funcionalidades relevantes para que o robô execute sua função essencial de buscar, identificar e apagar incêndios com segurança.

4.1.1 Taxa de Busca

A tabela a seguir mostra os resultados de cinco repetições do teste de busca pelo fogo em um andar de prédio.

Tabela 5.1 - Resultados do teste de busca pelo fogo

Tentativa	Configuração do espaço	Tempo decorrido (s)
1	Corredor 1	62
2	Sala Pequena 1	56
3	Sala Grande 1	59
4	Sala Grande 2	38
5	Corredor 2	33
	Média	50
	Desvio Padrão	13

Considerando que durante uma busca completa pode ser necessário que o robô trafegue por um ou dois corredores e em média por três ou quatro salas no máximo, visto que será guiado diretamente para a zona onde o fogo se encontra, em média, o tempo necessário não deve ultrapassar 6 minutos, menos que o limite inferior de 10 minutos estabelecido nos requisitos.

4.1.2 Reconhecimento de Fogo

Os resultados a seguir dizem respeito ao tempo transcorrido até que a fonte de radiação infravermelha, presente em um espaço como um corredor ou uma sala, seja identificada pelo robô e o mesmo acione o sistema de extinção. Inicialmente, o dispositivo é posicionado na entrada da sala ou início do corredor. Liga-se o robô e dispara-se o cronômetro.

Tabela 4.2 - Resultados do teste de reconhecimento de fogo.

Tentativa	Configuração do espaço	Tempo decorrido (s)
1	Corredor	28,2
2	Sala Grande 1	19,4
3	Sala Pequena 1	9,4
4	Sala Pequena 2	8,5
5	Sala Grande 2	19,6
	Média	17,02
	Desvio Padrão	8,185

A média de 17,02 segundos é menor que o limite inferior estabelecido no requisito (30 segundos) e ligeiramente acima do limite superior de 15 segundos definido no requisito R002.

4.1.3 Rapidez de Detecção de Fogo

Os resultados mostrados abaixo se referem à rapidez de detecção de fogo quando ele se encontra na linha de visão do robô.

Tabela 4.3 - Resultado do teste de rapidez de detecção de fogo.

x (pés)	y (pés)	Tempo (s)
7	3	0,12
17	4	0,16
12	2	0,15
4	3	0,06
30	0	0,41
3	1	0,10
5	0	0,12
27	1	0,28
2	0	0,07
30	6	0,32
	Média	0,18
	Desvio Padrão	0,12

O valor de x representa a distância horizontal da fonte em relação ao robô enquanto y representa a altura dela em relação ao solo. Para este teste, foi utilizada uma lâmpada incandescente de 184 W. A média das tomadas de tempo mostra que o robô consegue rapidamente, em bem menos tempo do que havia sido previsto, identificar uma chama que se encontre em sua linha de visão.

4.1.4 Acessibilidade da Chave de Desligamento

Para que o robô seja seguro de uma maneira geral, ele deve apresentar uma chave de desligamento que permita às pessoas desativá-lo quando preciso. Para aferir a facilidade com que pode ser acessada, posiciona-se o robô a um ângulo aleatório do sujeito de teste e mede-se o tempo necessário para que ele desligue o dispositivo.

Tabela 4.4 - Resultado do teste de acessibilidade da chave de desligamento.

Ângulo (graus)	**Tempo (s)**
34	1,024
17	0,952
160	1,144
295	1,400
333	1,377
39	1,136
360	1,424
309	1,644
224	1,976
104	1,344
359	1,633
32	1,344
133	1,208
261	1,312
196	1,809
95	1,344
151	1,352
65	1,771
358	1,472
125	1,632
Média	1,415
Desvio Padrão	0,2656

Em média, após várias repetições do teste, o tempo decorrido desde a identificação da presença do robô até o reconhecimento e acionamento da chave de desligamento mostra que a mesma é de fato facilmente acessível.

4.1.5 Duração da Bateria

A fim de verificar a autonomia da bateria, foi feito um teste em que o robô, incialmente com sua bateria completamente carregada, segue uma trajetória circular, delineada por uma faixa colorida no solo, com sua luz e sirene acionadas, até que um dos eventos a seguir aconteça: a luz pare de piscar, o barulho da sirene cesse ou ele não seja capaz de continuar o trajeto. Com isso, tem-se uma noção da duração da bateria com o robô operando com consumo próximo do máximo. Nesse cenário, estão em funcionamento os sistemas que mais consomem energia como a câmera, a luz, a sirene e principalmente ambos os motores das rodas, na máxima velocidade possível levando-se em conta as restrições do controlador PID. O teste foi realizado cinco vezes, sendo que em todas elas, depois de transcorridos 20 minutos, o robô ainda era capaz de percorrer o caminho com a luz piscando e a sirene ativada. O tempo é considerado satisfatório, uma vez que a equipe de bombeiros, na grande maioria dos casos, já estará presente no local de incêndio após esses 20 minutos.

4.2 Demonstração Final

De forma a avaliar o desempenho do robô de maneira mais abrangente, foi montada uma demonstração envolvendo o disparo do dispositivo, navegação desde um ponto distante até uma sala, desvio de obstáculos, reconhecimento de radiação infravermelha e ativação do sistema de extinção.

A princípio, o robô foi posicionado em um ponto aleatório de um dos corredores do primeiro andar do prédio *Otto Miller Hall* da *Seattle Pacific University*. Por meio de uma fita vermelha fixada ao solo, foi traçado um caminho até dentro de uma sala que se encontra aproximadamente 25 metros distante do ponto inicial. Dentro da sala, foram colocados alguns obstáculos no chão como caixas, mesas e cadeiras e uma lâmpada incandescente de 184 W para simulação da radiação infravermelha emitida pelo fogo.

O robô foi ligado e o cronômetro acionado. O caminho descrito pela fita foi seguido até um ponto dentro da sala. A partir do momento em que a câmera não identificava mais a fita de cor vermelha, iniciou-se a execução dos algoritmos de busca pelo fogo e desvio de obstáculos. O robô moveu-se em direção a alguns obstáculos, desviou-se deles e, após alguns segundos, identificou a fonte de radiação e navegou em direção a ela. O tempo total decorrido desde o acionamento do robô até o reconhecimento da fonte de radiação e acionamento do sistema de extinção foi de aproximadamente 1 minuto. Apesar de limitações como a ausência de comunicação entre o robô e o sistema central de combate a incêndio do prédio e a impossibilidade de execução de testes com fogo, a demonstração foi suficiente para revelar a eficácia do sistema de navegação por faixas no solo, direcionando o robô até

a zona de incêndio fictícia e sua capacidade de desviar de obstáculos, detectar fontes de radiação infravermelha e acionar um sistema de extinção embarcado no momento correto.

Foram encontrados problemas e dificuldades tanto com os sistemas mecânicos, quanto com os sistemas eletrônicos e a programação do código base. O ambiente de desenvolvimento e a linguagem utilizada foram propícios para a implementação das funções separadamente, mas inconvenientes quando foi necessária a integração das mesmas em um único programa denso, capaz de controlar todas as funcionalidades do robô. O software poderia ter sido otimizado caso fosse utilizado um sistema operacional de tempo real e as rotinas substituídas por tarefas de sistema. Essas tarefas teriam prioridades diferentes e o sistema operacional seria responsável pelo chaveamento entre elas de forma organizada, levando a uma maior fluidez durante a execução. O algoritmo de desvio de obstáculos foi projetado inicialmente para ser implementado utilizando-se lógica *fuzzy*. Após fracassadas tentativas de fazê-lo funcionar, foi utilizado um algoritmo mais simples, que apenas manipula aritmeticamente os valores de distância provenientes dos sensores ultrassônicos. Este revelou-se extremamente ineficiente, principalmente diante de obstáculos como pés de mesas e cadeiras. Em relação à placa de circuito impresso, não houve isolação entre os canais de terra dos motores e dos sensores. Por consequência, o ruído proveniente dos motores acabou interferindo nos sinais recebidos pela câmera e pelos sensores de distância, causando inconsistência dos dados recebidos pelo microcontrolador e dificultando os testes. Tanto a câmera de navegação quanto a de detecção de fogo mostraram-se extremamente sensíveis à luminosidade do local. Sendo assim, em diversas oportunidades o robô subitamente parava sobre a linha, pois a mudança de luminosidade prejudicava seu reconhecimento. Além disso, a luz do sol que entra pelas janelas das salas muitas vezes fez com que o robô movesse em direção a elas de maneira equivocada, acionando o sistema de extinção. Uma correção poderia ser feita utilizando-se câmeras melhores ou estudando-se uma forma mais eficaz de distinção entre fogo e a luz solar, ambos emissores de radiação infravermelha. No que se refere aos sistemas mecânicos, o maior impasse encontrado foi com o mecanismo de ajuste da mangueira. Ficaram evidentes algumas falhas de projeto que fizeram com que o ajuste de ângulo não funcionasse como esperado.

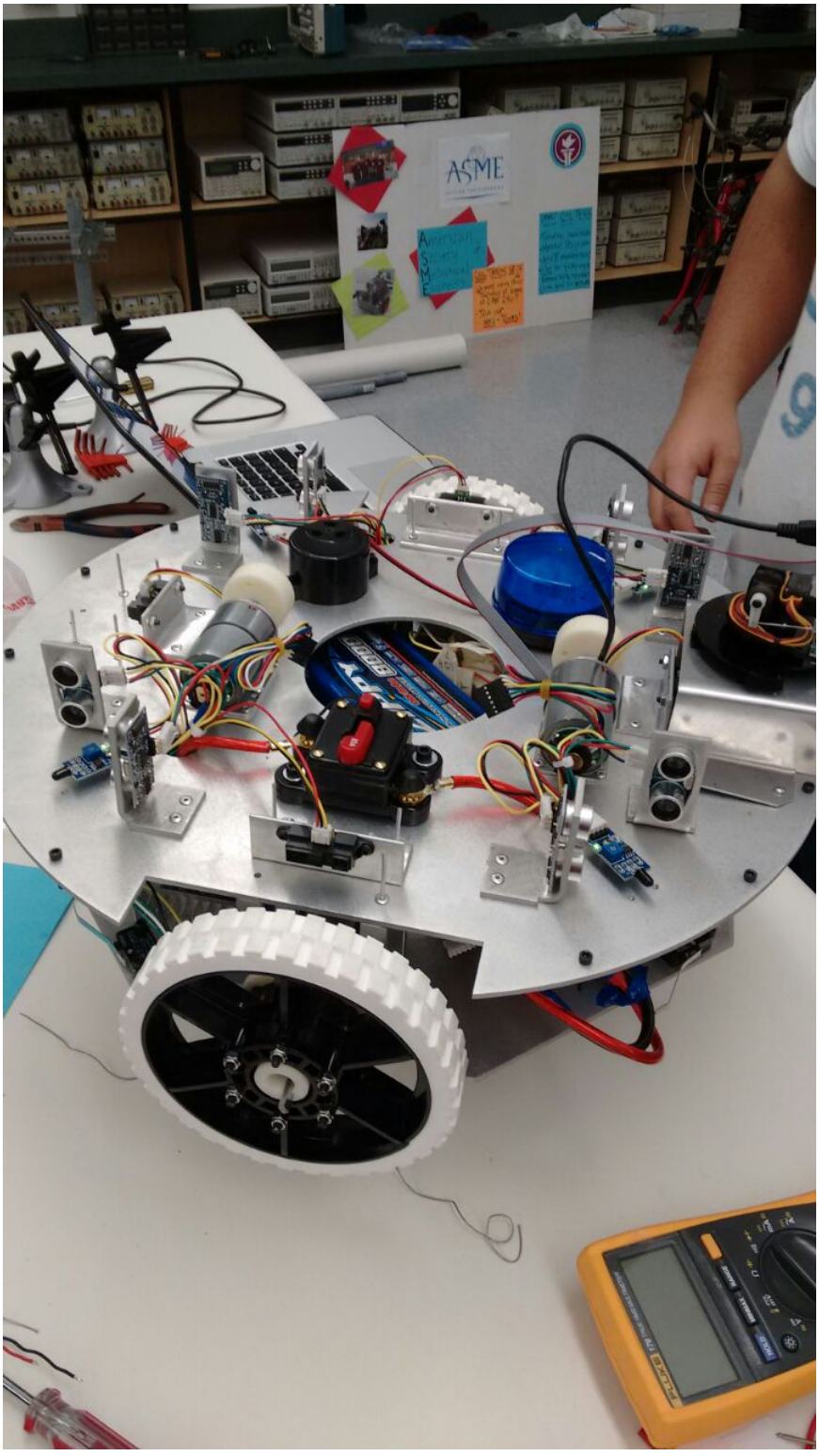

Figura 4.1: Foto do primeiro protótipo construído

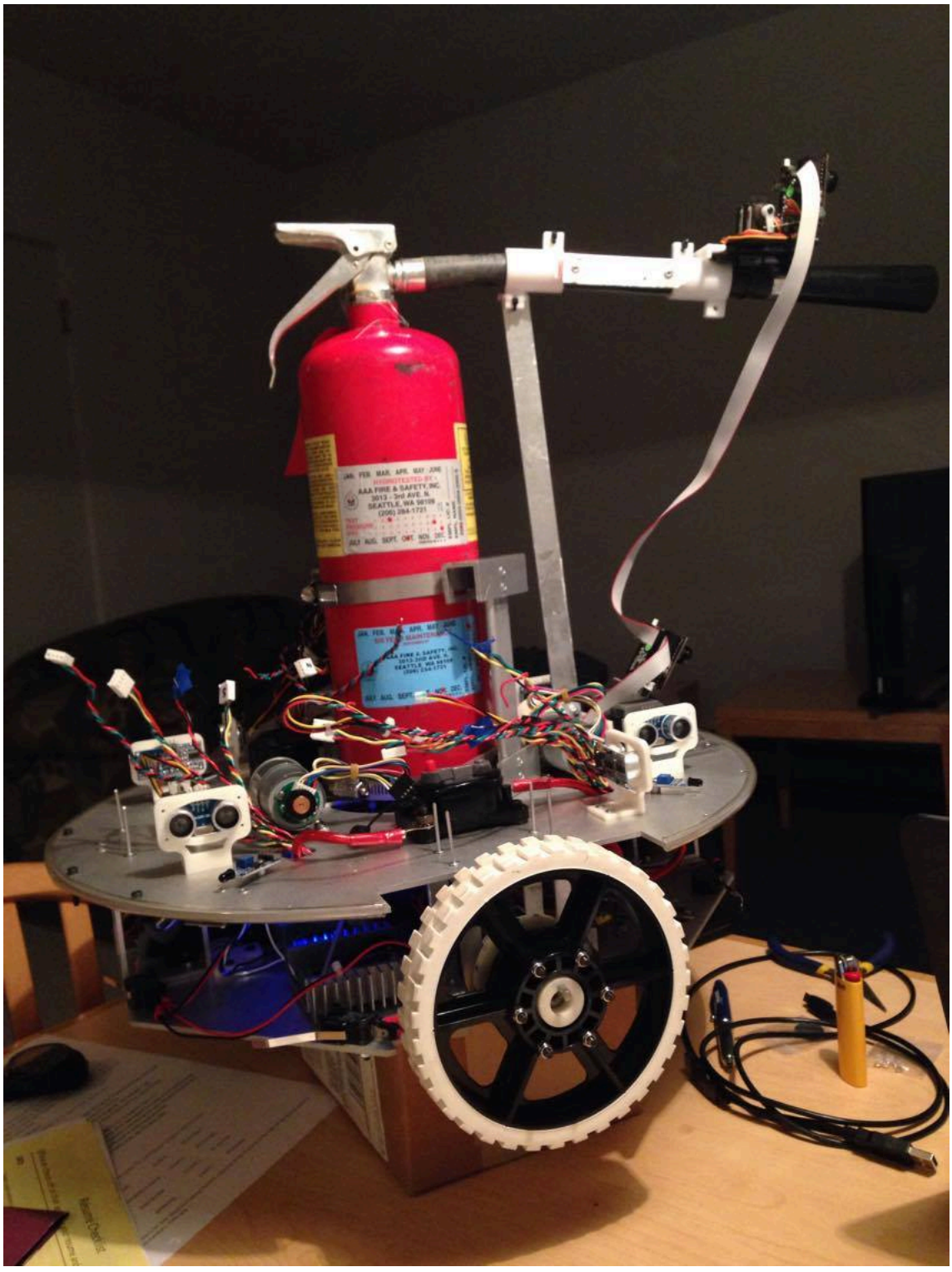

Figura 4.2: Foto da versão final de protótipo com extintor acoplado

5. Conclusão

O projeto do robô CERBERUS surgiu como uma proposta de alternativa para um problema grave e constantemente atacado por fabricantes de produtos e engenheiros de diversas áreas: a destruição causada por incêndios. Os estragos e perdas de vidas e bens materiais que sucedem esses desastres motivam a busca por um método de combate não só mais eficiente como também menos invasivo. Quando o tempo de resposta da equipe de bombeiros não é satisfatório e os sistemas de chuveiros automáticos tornam-se inconvenientes, abre-se espaço para uma terceira solução, descrita nesta monografia.

Os desafios, em contrapartida, são enormes em todas as áreas de engenharia envolvidas em um projeto multidisciplinar de tamanha complexidade como este. Para os engenheiros mecânicos e de materiais, encontrar materiais altamente resistentes ao fogo, com propriedades eletromagnéticas convenientes e ao mesmo tempo adequados para servir de estrutura e suporte aos componentes eletrônicos; para os engenheiros eletricistas, gerenciar o consumo de energia de forma a maximizar o tempo de duração da bateria e a durabilidade do robô, que em determinados casos pode permanecer intocado por longos períodos; para os engenheiros de computação, desenvolver um código de programação coeso, modular e confiável, potencializando sua performance e reduzindo ao máximo o risco de acidentes e o tempo total necessário para se chegar precisamente ao local de incêndio.

Com este trabalho, verificou-se que a viabilidade de um robô bombeiro autônomo capaz de responder de forma imediata a um sinal de alarme de incêndio, navegar em busca do fogo e suprimir completamente as chamas antes que estas atinjam proporções maiores, é real e considerável. Os resultados das simulações mostraram que o tempo decorrido desde o acionamento do robô até que ele efetivamente encontrasse o local do incidente foi satisfatório em diferentes cenários. Os tempos médios de busca e reconhecimento do fogo atestam a rapidez do dispositivo em percorrer o ambiente e detectar a presença do incêndio. Além disso, os testes de autonomia de bateria mostraram que o robô pode funcionar por mais de 20 minutos desde que bateria esteja completamente carregada quando o mesmo entra em ação, confirmando a capacidade do dispositivo de exercer sua função enquanto a equipe de bombeiros não chega ao local.

Apesar das dificuldades, tanto em se encontrar métodos de navegação que não exijam nenhuma modificação na estrutura do ambiente de aplicação do dispositivo quanto de se estabelecer uma comunicação confiável entre ele e o sistema central de incêndio, há maneiras eficazes de contorná-las, possibilitando a execução de testes que podem dar indícios de que esta solução pode vir a tornar-se até mesmo um produto comercializável.

Referências Bibliográficas

HOWARD, C. et al. Project C.E.R.B.E.R.U.S.: The Community Extinguishing Robot for the Brisk and Expedited Restoration of Unexcited Space. Cornell Cup, Seattle, 2015.

INSTITUTO SPRINKLER BRASIL. Estatísticas 2015. 2015. Disponível em: <http://www.sprinklerbrasil.org.br/instituto-sprinkler-brasil/estatisticas/estatisticas-2015-anual/>. Acesso em: 6 mar. 2016.

INSTITUTO SPRINKLER BRASIL. Brasil é o 3° país com o maior número de mortes por incêndio. 2015. Disponível em: <http://www.sprinklerbrasil.org.br/imprensa/brasil-e-o-3o-pais-com-o-maior-numero-de-mortes-por-incendio-newsletter-no-5/>. Acesso em: 6 mar. 2016.

MICHAELIS, DICIONÁRIO ONLINE. Disponível em: <http://michaelis.uol.com.br/moderno/portugues/index.php?lingua=portugues-portugues&palavra=rob%F4>. Acesso em: 10 mar. 2016.

DIETRICH, EIKE. Prevenção e Combate a Incêndio. 2015. Disponível em: <http://www.iqm.unicamp.br/sites/default/files/08.04%20e%2015.04_Treinamento%20contra%20inc%2B¬ndio%20Teoria%20e%20Pr%2B%C3%ADtica_Eike%20Dietrich_0.pdf>. Acesso em: 10 mar. 2016.

THE NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. Fire Dynamics. 2013. Disponível em: <http://www.nist.gov/fire/fire_behavior.cfm>. Acesso em: 13 mar. 2016.

SAÚTIL. Inalação de Fumaça. 2016. Disponível em: <http://www.sautil.com.br/saude-para-voce/saude-respiratoria/conteudo/inalacao-de-fumaca>. Acesso em: 13 mar. 2016.

BOMBEIROS. Combustão é fogo. Disponível em: <http://www.bombeiros.com.br/br/utpub/combustao_fogo.php>. Acesso em: 17 mar. 2016.

GLOBALSYST. Sistema de Detecção e Alarme de Incêndio. Disponível em: <http://www.globalsyst.com.br/sist_detec_alarme_incendio.php>. Acesso em: 21 mar. 2016.

BEAUREGARD, B. Improving the Beginner's PID. 2011. Disponível em: <http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/>. Acesso em: 24 mar. 2016.

Apêndice A – Código-fonte do robô CERBERUS

```
#include <Sabertooth.h>
#include <Wire.h>
#include <PixyI2C.h>
#include <Servo.h>
#include "text_reader.c"

#define X_CENTER    160L
#define Y_CENTER    100L
#define RCS_MIN_POS     0L
#define RCS_MAX_POS     1000L
#define RCS_MIN_POS_TILT     200L
#define RCS_MAX_POS_TILT     700L
#define RCS_CENTER_POS          ((RCS_MAX_POS-RCS_MIN_POS)/2)
#define maxSpeed 50
#define minSpeed 20
#define leftMotor 2
#define rightMotor 1

#define s0 13
#define s1 12
#define s2 8

//---------------------------------------
// Servo Loop Class
// A Proportional/Derivative feedback
// loop for pan/tilt servo tracking of
// blocks.
// (Based on Pixy CMUcam5 example code)
//---------------------------------------
class ServoLoop
{
  public:
    ServoLoop(int32_t proportionalGain, int32_t derivativeGain);

    void update(int32_t error);

    int32_t m_pos;
    int32_t m_prevError;
    int32_t m_proportionalGain;
    int32_t m_derivativeGain;
};

// ServoLoop Constructor
ServoLoop::ServoLoop(int32_t proportionalGain, int32_t derivativeGain)
{
  m_pos = RCS_CENTER_POS;
  m_proportionalGain = proportionalGain;
  m_derivativeGain = derivativeGain;
  m_prevError = 0x80000000L;
}
```

```
// ServoLoop Update
// Calculates new output based on the measured
// error and the current state.
void ServoLoop::update(int32_t error)
{
  long int velocity;
  char buf[32];
  if (m_prevError != 0x80000000)
  {
    velocity = (error * m_proportionalGain + (error - m_prevError) * m_derivativeGain) >> 10;

    m_pos += velocity;
    if (m_pos > RCS_MAX_POS)
    {
      m_pos = RCS_MAX_POS;
    }
    else if (m_pos < RCS_MIN_POS)
    {
      m_pos = RCS_MIN_POS;
    }
  }
  m_prevError = error;
}
// End Servo Loop Class
//---------------------------------------

// Multiplexers select pins value;
int r0 = 0;
int r1 = 0;
int r2 = 0;

// Ultrasonic pins
const int outPin = 4; // Using FAST_IO
const int inPin = A2; // Using FAST_IO

//Web Controller
const char * delimiter = "\n";

char * str;
char * pch;

String function = 0;
bool ledStatus = false;
bool stringIsOk = false;
double ITerm, lastInput;
String remoteIp;
int new_function = 0;


// Pixy I2C address
PixyI2C pixy(0x57); // You can set the I2C address through PixyI2C object
PixyI2C pixyIR(0x56); // You can set the I2C address through PixyI2C object

// Pixy Servos PID tune
ServoLoop panLoop(200, 200);  // Servo loop for pan
```

```
ServoLoop tiltLoop(150, 200); // Servo loop for tilt

// Declare ExtinguisherMotor
Servo extinguisherMotor;
Servo tiltMotor;

// Motor Controller address
Sabertooth ST(128);

// PID algorithm constants
float Kp = 0.35; // 0.35 @ 50 -- 0.2 @ 80
float Ki = 0.05; // 0.05 @ 50 -- 0.5 @ 80
float Kd = 1.30; // 1.30 @ 50 -- 1.00 @ 80
double outMin;
double outMax;
double Input, Output, Setpoint;

const int sampleRate = 1; // Variable that determines how fast our PID loop runs

// Communication setup
const long serialPing = 500; //This determines how often we ping our loop
// Serial pingback interval in milliseconds

unsigned long now = 0; //This variable is used to keep track of time
// placehodler for current timestamp

unsigned long lastMessage = 0; //This keeps track of when our loop last spoke to serial
// last message timestamp.

// VARIABLES - INPUT
int sonarLeftVAL = 0;
int sonarRightVAL = 0;
int sonarFrontLeftVAL = 0;
int sonarFrontRightVAL = 0;
int sonarBackVAL = 0;

// VARIABLES - OUTPUT
int motorSPD_L = 0;
int motorSPD_R = 0;

// VARIABLES - SYSTEM
int newMotorSPD_L = 0;
int newMotorSPD_R = 0;
int maxPing = 100; // Limit sensors to reading 100cm
int basicVelocity = 0;  // How much the CI feels like it should be moving forward
int urgTurn_L = 0;   // urge to turn to the Lefteft
int urgTurn_R = 0;   // urge to turn to the Right
int urgMotor_L = 0;   // urge to move Left motor forward
int urgMotor_R = 0;   // urge to move Right motor forward
int urgFatigue = 0; // Determines motor acceleration rate

void setup()
{
  // Initialize Serial output
  Serial.begin(9600);
```

```
  // Initialize the motors
  SabertoothTXPinSerial.begin(9600);
  ST.setBaudRate(38400);
  SabertoothTXPinSerial.end();
  SabertoothTXPinSerial.begin(38400);

  // initialize the Ultrasonic Sensor and Mutiplexers for input.
  pinMode(outPin, OUTPUT_FAST);
  pinMode(inPin, INPUT_FAST);
  pinMode(s0, OUTPUT);    // s0
  pinMode(s1, OUTPUT);    // s1
  pinMode(s2, OUTPUT);    // s2

  // Set the PWM ports outputs
  setPwmSwizzler(3, 9, 10, 11);
//  tiltMotor.attach(3, 900, 1920);
  extinguisherMotor.attach(3, 900, 1920);
  extinguisherMotor.write(90);

  // Set the setpoint for the PID
  Setpoint = X_CENTER;

  // Initialize pixy
  pixy.init();
  pixyIR.init();

  // Kick Plates Interruption
  attachInterrupt(2, kickPlate, CHANGE);

  // Starting
  lastMessage = millis();
  delay(2000);
  //  Serial.println("Starting...");
}

uint32_t lastBlockTimeIR = 0;
uint32_t lastBlockTime = 0;
volatile int color = 1;
volatile int go = LOW;

void loop()
{
  if (go == HIGH) {
    ST.motor(leftMotor, 0);
    ST.motor(rightMotor, 0);
    delay(3000); //Waits 5 secs until the robot starts running again
    go = LOW;
  } else {
      webController();
      Serial.println(new_function);
    if (new_function == 1) {
      lineFollowing();
    } else if (new_function == 2) {
      ReadSensors();
```

```
      AvoidWalls();
      SetMotors();
    } else if (new_function == 3) {
      fireTracking();
    } else if (new_function == 4) {
      extinguisherSystem();
      new_function = 0;
    }
  }
}

void lineFollowing() {
  int j = 0;
  Setpoint = 160;
  static int i = 0;
  uint16_t blocks;

  blocks = pixy.getBlocks();
  if (blocks) {
    for (j = 0; j < blocks; j++)
    {
      if (pixy.blocks[j].signature == color) {
        Input = pixy.blocks[j].x;
        Compute3();
        motors(Output);
        now = millis();
        lastBlockTime = millis();
      }
      else if (millis() - lastBlockTime > 100)
      {
        ST.motor(leftMotor, 0);
        ST.motor(rightMotor, 0);
      }
    }
  }
  else if (millis() - lastBlockTime > 100)
  {
    ST.motor(leftMotor, 0);
    ST.motor(rightMotor, 0);
  }
}

void motors(double Output) {
  double leftSpeed = constrain((maxSpeed - Output), minSpeed, maxSpeed);
  double rightSpeed = constrain((maxSpeed + Output), minSpeed, maxSpeed);
  Serial.print(leftSpeed);
  Serial.print("\t");
  Serial.print(rightSpeed);
  Serial.print("\n");

  ST.motor(leftMotor, leftSpeed);
  ST.motor(rightMotor, rightSpeed);
}
```

```
unsigned long lastTime;
double errSum, lastErr;

int SampleTime = 1; //1 sec
bool inAuto = false;

void Compute3()
{
  unsigned long now = millis();
  int timeChange = (now - lastTime);
  if (timeChange >= SampleTime)
  {
    /*Compute all the working error variables*/
    double error = Setpoint - Input;
    ITerm = ITerm + (Ki * error);

    if (ITerm > outMax) ITerm = outMax;
    else if (ITerm < outMin) ITerm = outMin;
    double dInput = (Input - lastInput);

    /*Compute PID Output*/
    Output = Kp * error + ITerm - Kd * dInput;

    /*Remember some variables for next time*/
    lastInput = Input;
    lastTime = now;
  }
}

void kickPlate()
{
  static unsigned long last_interrupt_time = 0;
  unsigned long interrupt_time = millis();
  // If interrupts come faster than 200ms, assume it's a bounce and ignore
  if (interrupt_time - last_interrupt_time > 200)
  {
    go = HIGH;
  }
  last_interrupt_time = interrupt_time;
}

void webController() {

  str = readFile();
  pch = strtok (str, delimiter);
  if (pch != NULL) {
    ledStatus =  (String(pch) == "true");
    pch = strtok (NULL, delimiter);
    function = String (pch);
    pch = strtok (NULL, delimiter);
    remoteIp = String (pch);
    pch = strtok (NULL, delimiter);
    stringIsOk  = String (pch) == "OK";

    //read the rest of the string, you can omit this
```

```
    while ((pch != NULL))
    {
      pch = strtok (NULL, delimiter);
    }
  }

  if (stringIsOk) {

    char buf[function.length()];
    function.toCharArray(buf, function.length() + 1);
    new_function = atof(buf);

    //    Serial.println(function);

    while (ledStatus == LOW) {
      Serial.println("PARADO");
      ST.motor(leftMotor, 0);
      ST.motor(rightMotor, 0);
      str = readFile();
      pch = strtok (str, delimiter);
      if (pch != NULL) {
        ledStatus =  (String(pch) == "true");
        pch = strtok (NULL, delimiter);
        function = String (pch);
        pch = strtok (NULL, delimiter);
        remoteIp = String (pch);
        pch = strtok (NULL, delimiter);
        stringIsOk  = String (pch) == "OK";

        //read the rest of the string, you can omit this
        while ((pch != NULL))
        {
          pch = strtok (NULL, delimiter);
        }
      }
    }
  }
}

// ------------- READ SENSOR VALUES AND STORE -------------
void ReadSensors() {

  // ----- SONAR -----
  delay(5); // delay to avoid echos between sensors
  sonarLeftVAL = readUltrasonic(1);  // Get distance in cm from sonar device
  delay(5); // delay to avoid echos between sensors
  sonarFrontRightVAL = readUltrasonic(3);
  delay(5); // delay to avoid echos between sensors
  sonarRightVAL = readUltrasonic(2);
  delay(5); // delay to avoid echos between sensors
  sonarFrontLeftVAL = readUltrasonic(0);
}


// ------------- PROCESS AND INTERPRET SENSOR DATA -------------
```

```
void AvoidWalls() {

  // GET/SET URGE VALUES
  basicVelocity = 25; // Set above 0 to make it continuously move forward
  urgMotor_L = 0;
  urgMotor_R = 0;
  urgFatigue = 0;

  // AVOID WALLS AT SIDE
  urgTurn_L += maxPing * maxPing - ((maxPing - sonarRightVAL) * (maxPing - sonarRightVAL)); //inverse proportional to square of rightval
  urgTurn_R += maxPing * maxPing - ((maxPing - sonarLeftVAL) * (maxPing - sonarLeftVAL));
  urgMotor_L -= 0.1 * (maxPing * maxPing - ((maxPing - sonarRightVAL) * (maxPing - sonarRightVAL)));
  urgMotor_R -= 0.1 * (maxPing * maxPing - ((maxPing - sonarLeftVAL) * (maxPing - sonarLeftVAL)));

  // AVOID OBJECTS IN FRONT
  urgMotor_L += maxPing * maxPing - 0.5 * ((maxPing - sonarFrontLeftVAL) * (maxPing - sonarFrontLeftVAL)) - ((maxPing - sonarFrontRightVAL) * (maxPing - sonarFrontRightVAL));
  urgMotor_R += maxPing * maxPing - 0.5 * ((maxPing - sonarFrontRightVAL) * (maxPing - sonarFrontRightVAL)) - ((maxPing - sonarFrontLeftVAL) * (maxPing - sonarFrontLeftVAL));

  // SCALE URGES TO PWM output values (255)
  urgTurn_L = basicVelocity - map(urgTurn_L, 0, 1.8 * maxPing * maxPing, -basicVelocity, basicVelocity); // Scale to within PWM output limits
  urgTurn_R = basicVelocity - map(urgTurn_R, 0, 1.8 * maxPing * maxPing, -basicVelocity, basicVelocity);
  urgMotor_L = map(urgMotor_L, 0, 1.8 * maxPing * maxPing, -basicVelocity, basicVelocity); // Scale to within PWM output limits
  urgMotor_R = map(urgMotor_R, 0, 1.8 * maxPing * maxPing, -basicVelocity, basicVelocity);

  // SET MOTOR SPEED
  newMotorSPD_L = basicVelocity + (urgMotor_L * 2) + (urgTurn_R / 2) - (urgTurn_L / 1) + 9;
  newMotorSPD_R = basicVelocity + (urgMotor_R * 2) + (urgTurn_L / 2) - (urgTurn_R / 1) + 9;

  // Clip to 255/-255 (negative value means reverse direction)
  if (newMotorSPD_L > basicVelocity) newMotorSPD_L = basicVelocity;
  if (newMotorSPD_L < -basicVelocity) newMotorSPD_L = -basicVelocity;
  if (newMotorSPD_R > basicVelocity) newMotorSPD_R = basicVelocity;
  if (newMotorSPD_R < -basicVelocity) newMotorSPD_R = -basicVelocity;
  if ((newMotorSPD_R <= 0) && (newMotorSPD_R > -10)) newMotorSPD_R = -10;
  if ((newMotorSPD_R > 0) && (newMotorSPD_R <= 10)) newMotorSPD_R = 10;
  if ((newMotorSPD_L <= 0) && (newMotorSPD_L > -10)) newMotorSPD_L = -10;
  if ((newMotorSPD_L > 0) && (newMotorSPD_L <= 10)) newMotorSPD_L = 10;

}


// ------------- SET MOTOR DIRECTION AND PWM OUTPUTS -------------
void SetMotors() { // Convert calculated speed changes to actuator control data
  int lrgSpdDelta = 0; //Used to store largest change in speed
```

```
  if (urgFatigue > 0) { // If using acceleration
    // CALCULATE SPEED CHANGES
    int dltaSpd_L = newMotorSPD_L - motorSPD_L;
    int dltaSpd_R = newMotorSPD_R - motorSPD_R;
    // Find largest speed difference
    if (abs(dltaSpd_L) >= abs(dltaSpd_L)) {
      lrgSpdDelta = abs(dltaSpd_L);
    } else {
      lrgSpdDelta = abs(dltaSpd_R);
    }

    // ACCELERATE MOTORS
    for (int i = 0; i < lrgSpdDelta; i++) {
      if (newMotorSPD_L < motorSPD_L) motorSPD_L--;
      if (newMotorSPD_L > motorSPD_L) motorSPD_L++;
      if (newMotorSPD_R < motorSPD_R) motorSPD_R--;
      if (newMotorSPD_R > motorSPD_R) motorSPD_R++;
      ST.motor(leftMotor, motorSPD_L);
      ST.motor(rightMotor, motorSPD_R);
      delay(urgFatigue);// Determines Acceleration
    }
  } else {
    motorSPD_L = newMotorSPD_L;
    motorSPD_R = newMotorSPD_R;
    ST.motor(leftMotor, motorSPD_L);
    ST.motor(rightMotor, motorSPD_R);
  }

}// END SetMotors


float readUltrasonic(int count)
{
  r0 = bitRead(count, 0);
  r1 = bitRead(count, 1);
  r2 = bitRead(count, 2);

  digitalWrite(s0, r0);
  digitalWrite(s1, r1);
  digitalWrite(s2, r2);

  float duration, cm;

  fastDigitalWrite(outPin, LOW);
  waitMicros(2);
  fastDigitalWrite(outPin, HIGH);
  waitMicros(10);
  fastDigitalWrite(outPin, LOW);

  duration = pulseIn(inPin, HIGH); // calls fastGpioPciDigitalRead


  // convert the time into a distance
  cm = microsecondsToCentimeters(duration);
  (cm > 101) ? (cm = maxPing) : (cm = cm);
```

```
  delay(50);

  return cm;
}

void waitMicros(int val)
{
  unsigned long a = micros();
  unsigned long b = micros();
  while ((b - a) < val)
  {
    b = micros();
    if (a > b)
    {
      break;
    }
  }
}

float microsecondsToCentimeters(float microseconds)
{
  // The speed of sound is 340 m/s or 29 microseconds per centimeter.
  // The ping travels out and back, so to find the distance of the
  // object we take half of the distance travelled.
  return microseconds / 29 / 2;
}

// ------------- SEND DEBUG INFO ON SERIAL PORT -------------
void SerialDebug() {

  Serial.print("Sonar LEFT = ");
  Serial.println(sonarLeftVAL);
  Serial.print("Sonar FRONT LEFT = ");
  Serial.println(sonarFrontLeftVAL);
  Serial.print("Sonar FRONT RIGHT = ");
  Serial.println(sonarFrontRightVAL);
  Serial.print("Sonar RIGHT = ");
  Serial.println(sonarRightVAL);

  Serial.println("LEFT\t\tRIGHT");
  Serial.print(motorSPD_L);
  Serial.print("\t\t");
  Serial.println(motorSPD_R);
}

void fireTracking()
{
  uint16_t blocksIR;
  blocksIR = pixyIR.getBlocks();

  // If we have blocksIR in sight, track and follow them
  if (blocksIR)
  {
    int trackedBlock = TrackBlock(blocksIR);
    FollowBlock(trackedBlock);
```

```
    lastBlockTimeIR = millis();
  }
  else if (millis() - lastBlockTimeIR > 100)
  {
    ST.motor(leftMotor, 0);
    ST.motor(rightMotor, 0);
    ScanForBlocks();
  }
}

int oldX, oldY, oldSignature;

//---------------------------------------
// Track blocksIR via the Pixy pan/tilt mech
// (based in part on Pixy CMUcam5 pantilt example)
//---------------------------------------
int TrackBlock(int blockCount)
{
  int trackedBlock = 0;
  long maxSize = 0;

  //       Serial.print("blocksIR =");
  //       Serial.println(blockCount);

  for (int i = 0; i < blockCount; i++)
  {
    if ((oldSignature == 0) || (pixyIR.blocks[i].signature == oldSignature))
    {
      long newSize = pixyIR.blocks[i].height * pixyIR.blocks[i].width;
      if (newSize > maxSize)
      {
        trackedBlock = i;
        maxSize = newSize;
      }
    }
  }

  int32_t panError = X_CENTER - pixyIR.blocks[trackedBlock].x;
  int32_t tiltError = pixyIR.blocks[trackedBlock].y - Y_CENTER;

  panLoop.update(panError);
  tiltLoop.update(tiltError);

  pixyIR.setServos(panLoop.m_pos, tiltLoop.m_pos);

  oldX = pixyIR.blocks[trackedBlock].x;
  oldY = pixyIR.blocks[trackedBlock].y;
  oldSignature = pixyIR.blocks[trackedBlock].signature;
  return trackedBlock;
}

//---------------------------------------
// Follow blocksIR via the Zumo robot drive
//
// This code makes the robot base turn
```

```
// and move to follow the pan/tilt tracking
// of the head.
//---------------------------------------
int32_t size = 70;
void FollowBlock(int trackedBlock)
{
  int32_t followError = RCS_CENTER_POS - panLoop.m_pos;  // How far off-center are we
looking now?

  // Size is the area of the object.
  // We keep a running average of the last 8.
  size += pixyIR.blocks[trackedBlock].width * pixyIR.blocks[trackedBlock].height;
  size -= size >> 3;

  // Forward speed decreases as we approach the object (size is larger)
  int forwardSpeed = constrain (30 - (size / 100), -30, 30);

  // Steering differential is proportional to the error times the forward speed
  int32_t differential = (followError + (followError * forwardSpeed)) / 900; //AJUSTAR!!!!!! NAO
SEI COMO!!!???

  // Adjust the left and right speeds by the steering differential.
  int leftSpeed = constrain(forwardSpeed + differential, -30, 30);
  int rightSpeed = constrain(forwardSpeed - differential, -30, 30);

  // And set the motor speeds
  ST.motor(leftMotor, leftSpeed);
  ST.motor(rightMotor, rightSpeed);
}

//---------------------------------------
// Random search for blocksIR
//
// This code pans back and forth at random
// until a block is detected
//---------------------------------------
int scanIncrement = (RCS_MAX_POS - RCS_MIN_POS) / 150;
uint32_t lastMove = 0;

void ScanForBlocks()
{
  if (millis() - lastMove > 20)
  {
    lastMove = millis();
    panLoop.m_pos += scanIncrement;
    if ((panLoop.m_pos >= RCS_MAX_POS) || (panLoop.m_pos <= RCS_MIN_POS))
    {
      tiltLoop.m_pos = random(RCS_MAX_POS_TILT * 0.6, RCS_MAX_POS_TILT);
      scanIncrement = -scanIncrement;
      if (scanIncrement < 0)
      {
        ST.motor(leftMotor, map(-250, -400, 400, -70, 70));
        ST.motor(rightMotor, map(250, -400, 400, -70, 70));
      }
      else
```

```
      {
        ST.motor(leftMotor, map(+180, -400, 400, -70, 70));
        ST.motor(rightMotor, map(-180, -400, 400, -70, 70));
      }
      delay(random(250, 500));
    }

    pixyIR.setServos(panLoop.m_pos, tiltLoop.m_pos);
  }
}

void extinguisherSystem() {
  extinguisherMotor.write(180);
  delay(100);
  extinguisherMotor.write(90);
  delay(1000);
  extinguisherMotor.write(0);
  delay(100);
  extinguisherMotor.write(90);
  delay(100);
  delay(5000);
}
```

Apêndice B – Lista de materiais utilizados

Description	Manufacturer and complete part number	Supplier
ELECTRICAL	---	---
Ultrasonic Sensor (2 - 450cm)	MULTICOMP HC-SR04	Ebay
IR Distance Sensor (15 - 150cm)	Sharp GP2Y0A21YK0F	Ebay
Flame Sensor (1 - 80cm)	Shenzhen Ezoneda Technology LM393 IR Flame Temp Sensor Board	Ebay
Hall Effect Sensor	Melexys US1881	Ebay
Piezzo Buzzer (108dB @ 30cm)	108DB PIEZO BUZZER	Radio Shack
Strobe Signal	Commodity	Ebay
8 AH, 30C Lithium Polymer Battery	Zippy FlightMax Z80004S-30	Hobby King
Wires	Commodity	Commodity
Circuit Breaker	80 AMP 12V DC CIRCUIT BREAKER REPLACE FUSE 80A	Uneek Supply
Power Plug (Arduino)	2.1MM COAX POWER PLUG W/ PIGTAIL LEADS	All Electronics
Fuse	Crimpable Inline ATC/ATO Blade Fuse Holder	Power Werx
Battery Charger	Thunder AC6 Smart LiPo Balance Charger	Hobby Partz
CMUcam5 Pixy	Pixy (CMUcam5)	Amazon
CMUcam5 Pan/Tilt	Pan/Tilt Mechanism Kit for Pixy (CMUcam5)	Amazon
Bullet Connectors	Polymax 5.5mm Gold Connectors 10 pairs (20pc)	Hobby King
XT60 Male Connectors	Commodity	Hobby King
Power Distribution Block	4-Way Car Audio Stereo Amp Power/Ground Cable Splitter Distribution Block 4ga	Ebay
LCD Display	16x2 Character LCD LCM Display Module HD44780 Controller Blue Backlight	Ebay
Ceramic Capacitor 0.1 uF	Commodity	Commodity
Ceramic Capacitor 0.33 uF	Commodity	Commodity
Ceramic Capacitor 1 uF	Commodity	Commodity
Resistor 6 kΩ	Commodity	Commodity
Resistor 10 kΩ	Commodity	Commodity
Resistor 15 kΩ	Commodity	Commodity
Resistor 20 kΩ	Commodity	Commodity
Resistor 22 kΩ	Commodity	Commodity
Voltage Regulator 5V	Commodity	Commodity
Voltage Regulator 12V	Commodity	Commodity
Transistor NPN (BC546)	Commodity	Commodity
Diodes 1N4004	Commodity	Commodity
Terminal Block Connector 2 pins	Commodity	Commodity
Relay 12V	Commodity	Commodity
PCB	Custom	Custom
PCB Revision 1	Custom	Custom
Micro Limit Switch	Omron V-10G3-1C24-K	DigiKey
Micro Limit Switch	Omron V-153-1C25	Ebay
Switch Housing	TE Receptacle Faston 3CIR .187	Digikey
Switch Pin	TE QC RCPT 20-24AWG 0.187	Digikey
Push Buttons	Tactile Push Button Switch Tact Switch 6X6X5mm 4-pin DIP	Ebay
Header IDC	FCI Connector Header 10 Position Dual Vertical PCB	Digikey
USB Type A Female Connector	Assmann Connector USB Rectangular Female Type A PCB	Digikey
IDC Socket	On Shore Technology Connector Socket IDC 10 Position	Digikey
Crimping Tool	Molex 14-24 AWG Universal Crimp Tool	Digikey
Connector Pin	Molex Connector Terminal Female 22-30 AWG Tin	Digikey
2-Position Header	Molex Connector Header 2 Position .100 Vertical Tin	Digikey
3-Position Header	Molex Connector Header 3 Position .100 Vertical Tin	Digikey
4-Position Header	Molex Connector Header 4 Position .100 Vertical Tin	Digikey
2-Position Housing	Molex Connector Housing 2 Position .100	Digikey
3-Position Housing	Molex Connector Housing 3 Position .101	Digikey
4-Position Housing	Molex Connector Housing 4 Position .102	Digikey
IR Lock Kit	IR-LOCK Filter Kit For Pixy	IR Lock
MECAHCNICAL	---	---
Aluminum 6061-T6 1/8" Sheet	Commodity	OnlineMetals.com
Aluminum 6061-T6 1" x 1" x 1/8" L-channel	Commodity	OnlineMetals.com
Aluminum 6061-T6 2" x 3" x 1/8" Rectangle Tube	Commodity	OnlineMetals.com
Drive Motor	Mabuchi RS-555 DC Motor	Banebots.com
16:1 Gearbox	Banebots P60 Gearbox: Stock, Standard Shaft, RS-540/550 Mount, 16:1	Banebots.com
3D printing	Stratasys Uprint Material	Through SPU
1/8" x 1/16" magnets	Commodity	Amazon.com
Motor Controller -2x12A	Dimension Engineering Sabertooth dual 12A motor driver	DimensionEngineering.com
Motor Controller -2x5A	Dimension Engineering Sabertooth dual 5A motor driver	DimensionEngineering.com
4" Omni Wheel	Vex Robotics 217-2584	VexRobotics.com
6" Normal Wheel	Andymark AM-0940a	Andymark.com
Small Motor w/ Gearbox/Encoder	Pololu 30:1 Metal Gearmotor 37Dx52L mm with 64 CPR Encoder	Pololu.com
0.375" ID x 1.125" OD x 0.375in Bearings	Commodity	Andymark.com
Spring Loaded Hinge	Sugatsune HG-SH25C	Sugatsune HG-SH25C
1/8" Rivets	Commodity	Commodity
#6-32 Bolts (3/8" long)	Commodity	Commodity
#6-32 Bolts (1/2" long)	Commodity	Commodity
#10-32 Bolts (0.25" long)	Commodity	Commodity
#10-32 Bolts (3/8" long)	Commodity	Commodity
3/8" Aluminum rod	Commodity	Commodity
#6-32 Nuts	Commodity	Commodity
4mm x 0.7mm Bolts (50mm long)	Commodity	Commodity
4mm x 0.7 mm Nylock Nuts	Commodity	Commodity
White Lithium Grease	Commodity	Commodity
1/8" Steel Cable	Commodity	Commodity
Badger 5 lb Fire Extinguisher	Badger 21007866	Kats Enterprises 21007866
J-B Weld puddy	Epoxy Adhesive, Wet/Dry, 2 oz, Stick	zoro.com
Pipe Clamp	Commodity	ebay.com
Smaller Pipe Clamp	Commodity	clipsandfasteners.com
Motor Mount	Pololu Stamped Aluminum L-Bracket Pair for 37D mm Metal Gearmotors	Pololu.com
Compression sleeve	Commodity	Home Depot
Aluminum Channel	Commodity	Onlinemetals.com
PROGRAMMING	---	---
Arduino Mega	Arduino Mega 2560 Rev3	Ebay
Intel Edison	Intel® Edison and Arduino Breakout Kit	SparkFun