Sistema para Controle Distribuído de Veículos Autônomos Terrestres

Charles Steinmetz[1], Reiner Franthesco Perozzo[1]

[1]Curso de Sistemas de Informação – Centro Universitário Franciscano - Caixa Postal 97010-032– Santa Maria – RS – Brasil

charlessteinmetzk@gmail.com, reiner.perozzo@unifra.br

Abstract. *The process automation is increasingly present in the lives of human beings. It can be very useful for repetitive and monotonous tasks as it can offer precision and quality from start to end of tasks. One way to help the man in these activities is the use of autonomous robots. In this sense, this paper proposes the development of a flexible and modular system, based on routing algorithms and applied to land vehicles with multiple functionalities. In addition, project-related work are presented and discussed that serve as a basis for the development of the presented solutions. At the end the work is validated with the implementation of the route algorithm applied to the prototype, also developed within the proposed.*

Resumo. *A automatização de processos está cada vez mais presente na vida do ser humano. Ela pode ser muito útil para tarefas repetitivas e monótonas, pois pode oferecer precisão e qualidade do início ao fim das tarefas. Uma forma de ajudar o homem nessas atividades é a utilização de robôs autônomos. Nesse sentido, este trabalho propõe o desenvolvimento de um sistema flexível e modular, baseado em algoritmos de rotas e aplicado aos veículos terrestres de múltiplas funcionalidades. Além disso, são apresentados e discutidos trabalhos relacionados ao projeto e que servem como base para o desenvolvimento das soluções apresentadas. Ao final, o trabalho é validado com a aplicação do algoritmo de rota em um protótipo desenvolvido também no âmbito da proposta.*

1. Introdução

Para o homem, tarefas repetitivas e monótonas podem causar cansaço e, consequentemente, uma perda de qualidade ou de precisão na execução dessas tarefas. Nesse sentido, a automação busca oferecer sistemas ativos que são projetados numa tentativa de executar tarefas do início ao fim do processo, buscando realizar a atividade, por mais monótona que essa possa ser, da mesma forma de quando iniciada [Moraes e Castrucci 2001], [Silveira e Winderson 2004]. A utilização de robôs autônomos pode oferecer vantagens no desenvolvimento de tarefas consolidadas a medida que priva sua execução de possíveis erros humanos e/ou busca oferecer precisão na atividade, além de poder proporcionar conforto a seu utilizador [Thrun, Wolfram e Fox 2005].

Atualmente, existem projetos de veículos autônomos para as mais variadas áreas como, por exemplo, (i) setor aéreo, através dos Veículos Aéreos não Tripulados (VANTs), (ii) aquático, em que o projeto GT5 – Veículos Aquáticos e Subaquáticos Autônomos do Instituto Nacional de Ciência e Tecnologia em Sistemas Embarcados Críticos (INCT-SEC) busca realizar atividades relacionadas com o monitoramento

ambiental e o controle de fronteiras [INCT 2012] e, também, (iii) os terrestres, em que é possível destacar o carro autônomo da Google [Teck 2010]. Essas são algumas das iniciativas que buscam oferecer mais comodidade ao ser humano na execução de atividades repetitivas ou tediosas [Thrun, Wolfram e Fox 2005], as quais poderiam ser automatizadas por uma máquina, na busca por maior produtividade, eficiência e precisão [Silveira e Winderson 2004].

Assim, este trabalho propõe um sistema flexível e modular para veículos terrestres de múltiplas funcionalidades, que servirá tanto para ambientes *indoor* quanto para ambientes *outdoor* e será controlado por um sistema externo. O trabalho será validado com o robô autônomo aplicado em um cenário composto por uma sala de 9 m² de área, onde o veículo deverá explorar todo o ambiente, buscando visitar todas as áreas. Essa proposta justifica-se pela necessidade de um sistema que seja híbrido, destinado para múltiplas funcionalidades, dentro do contexto atual em que a robótica está cada vez mais presente nas atividades diárias do ser humano.

2. Referencial Teórico

Este Capítulo tem por objetivo apresentar a revisão bibliográfica sobre o tema escolhido para este trabalho, incluindo conceitos, tecnologias e trabalhos relacionados que servem como base para a elaboração da proposta.

2.1 Revisão Bibliográfica

Neste Subcapitulo serão apresentados algumas tecnologias e conceitos necessários para a construção do trabalho. Serão abordados a Robótica, o Arduino, os sensores em geral e o padrão de comunicação *Bluetooth*.

2.1.1 Robótica

A palavra robô, tem origem da palavra tcheca “robota” e significa “servidão ao trabalhador forçado” [Silveira e Winderson 2004]. A Associação Industrial de Robótica - *Robotic Industrial Association* (RIA) - define robô como “um manipulador reprogramável multifuncional projetado para manusear materiais, peças, ferramentas ou dispositivos especiais, através de movimentos programados para a realização de uma variedade de tarefas” [Silveira e Winderson 2004], [Tsai 1999].

Em termos gerais, os robôs, são constituídos por uma unidade de controle (parte que constitui o sistema e que, geralmente, é programada), por atuadores (dispositivos que transformam sinais elétricos em ações a serem executadas), por estrutura mecânica (proporcionando os movimentos e definindo o grau de liberdade do robô) e sensores (que são responsáveis pela precisão dos movimentos além do fornecimento de variáveis ambientais, como pressão, humidade, temperatura) [Silveira e Winderson 2004].

Os problemas na robótica podem ser divididos em três grandes áreas: (i) Mapeamento, (ii) Localização e (iii) Planejamento do movimento. Na intersecção dessas áreas surgem diversos desafios a serem superados, ainda mais complexos que cada segmento em separado [Makarenko et al 2002].

O problema de mapeamento, na robótica, busca adquirir um modelo espacial do ambiente físico em que o robô está inserido. Por isso, pode-se dizer que o mapeamento

é geralmente considerado um dos problemas mais importantes para se construir robôs móveis autônomos [Thrun, 2002].

Outro problema relevante da robótica é a localização do robô. A principal tarefa desempenhada neste segmento é a estimação da posição corrente do robô dentro de um ambiente previamente conhecido por ele [Olson 2000]. Uma linha de técnicas amplamente utilizada para realizar tal atividade é baseada no método de Monte Carlo, onde pode-se destacar o *particle filter* [Fox et al. 1999].

Por fim, o terceiro grande problema da robótica é o planejamento da movimentação ou *path planning*. A preocupação nesta subárea é determinar a melhor trajetória que o robô deve desempenhar a partir de sua posição inicial, ponto A, até uma posição objetivo no espaço, ponto B, dado que sabe precisamente sua posição e estrutura do ambiente no qual está imerso.

Normalmente, os algoritmos de *path planning* são implementados de forma a prever obstáculos e tratar situações imprevistas, afim de evitar colisões. Existe uma vasta gama de algoritmos para navegação que buscam determinar o caminho ótimo de um ponto de origem a um ponto de destino, como por exemplo A* (*AStar*) e o algoritmo de *Djikstra*. O algoritmo A*, normalmente, é usado com mapas de ocupação (*grids*), sendo possível também, usá-lo com mapas métricos em que se tornou muito conhecido devido a sua aplicação em agentes autônomos para jogos [Rabin 2002]. Esse algoritmo é baseado em grafos, árvores ou *grids*, sendo que cada nó é um estado do caminho e cada estado possui transições para outros estados. Segundo Deloura (2000), A* é um dos algoritmos mais populares para o *path-finding*, ele é flexível pois, mudando sua heurística, pode ser usado em outros contextos.

2.1.2 Arduino

Arduino é um uma plataforma de *hardware* livre, para prototipação eletrônica, com suporte de entrada e saída de sinais que permite aos usuários programarem processos entre o dispositivo e os componentes externos que estão conectados a ele como, por exemplo, sensores. Essa plataforma torna a robótica mais acessível para diversos públicos, podendo ser utilizada para desenvolver objetos autônomos, automatizar processos, conectá-los a computadores ou em uma rede [Mcroberts 2010].

O projeto Arduino iniciou na Itália em 2005, com o objetivo de ser um *hardware* flexível quanto ao uso. É possível que seu *hardware* seja montado ou adquirido pré-montado. A adaptação do *hardware* do Arduino é livre, conforme a necessidade de cada projeto [Arduino 2014].

2.1.3 Sensores

Sensores podem ser definidos como dispositivos eletroeletrônicos que convertem grandezas físicas (temperatura, pressão, velocidade, corrente, aceleração, etc.) em um sinal elétrico que pode ser transmitido a um elemento indicador para que seja exibido ou processado. Com base nesse valor uma decisão pode ser tomada [Wendling 2010].

Existem sensores, com suas particularidades, que podem ser implantados na detecção de obstáculos em seu caminho. Esses sensores podem ser indutivos, capacitivos ou ultrasônicos. Eles são úteis na detecção de objetos na faixa entre 2

centímetros até 4 metros, desde que estejam na mesma altura do ponto onde está localizado o sensor e que possam refletir a radiação [Wendling 2010].

O sensor ultrassônico possui o funcionamento semelhante à alguns animais como golfinhos e morcegos (que utilizam a distância para detectar presas): um oscilador emite ondas ultrassônicas que, ao refletirem num objeto, retornam para o sensor. O sinal é captado e assim permite-se deduzir a distância do objeto ao sensor baseado no tempo de trânsito do sinal [Wendling 2010].

2.1.4 *Bluetooth*

Em 1994, a empresa L.M. Ericsson junto com outras quatro empresas (IBM, Intel, Nokia e Toshiba) formou o consórcio SIG (*Special Interest Group*) com o objetivo de desenvolver um padrão sem fio para interconectar dispositivos de computação e comunicação, incluindo acessórios. O projeto foi denominado Bluetooth [Tanenbaum 2003].

Segundo Stallings (2005), o conceito do Bluetooth é fornecer um protocolo padrão de comunicação sem fio, do tipo PAN (*Personal Area Network*), universal e baseado em *microchips* transmissores em cada dispositivo.

2.2 Trabalhos Correlatos

Esta seção descreve trabalhos cujas abordagens estão relacionadas com veículos autônomos e contribuem para o desenvolvimento da proposta.

Dentro desse contexto, existem diversas propostas de trabalhos relacionados, dentre os quais é possível citar o desenvolvimento de um sistema de visão artificial para um robô explorador [Schueroff 2014] e um robô para mapeamento de ambientes estruturados [Paschoal 2009]. Entretanto, em dado momento, eles acabam convergindo e apresentando conceitos e tecnologias com utilização semelhantes. De qualquer modo, foram selecionados e descritos aqueles que mais se aproximam da abordagem proposta no âmbito deste trabalho.

2.2.1 Desenvolvimento De Um Veículo Autônomo Que Detecta Obstáculos

Nesse trabalho [Teixieira 2013] é proposto o desenvolvimento de um algoritmo de detecção de obstáculos, aplicando-o em um veículo para que este seja capaz de identificar e desviar de obstáculos, de forma autônoma.

O trabalho desenvolvido apresenta um sistema embarcado, sendo o mesmo um veículo robotizado capaz de sensoriar o ambiente em busca de obstáculos e desviar dos mesmos. A implementação do protótipo funciona da seguinte forma: enquanto não for identificado nenhum obstáculo, o veículo se deslocava para frente. Assim que identificado um obstáculo, executa-se um processamento para definir a nova rota.

Como resultados, o autor detectou um problema na velocidade de cada motor – em que um gira mais rápido que o outro - que acabou afetando diretamente os resultados. Entretanto, o trabalho contribui com a planta para o cenário de testes e o fator reação ao detectar algum obstáculo e decidir nova rota.

2.2.2 Desenvolvimento De Um Protótipo De Robô Cortador De Grama

Esse trabalho [Devenz 2012] apresenta um protótipo de um robô cortador de grama, destinado às áreas com superfície plana e com limitadores de terreno. Ao ser ligado, o Robô verifica se existe limitações à sua direita ou a à sua esquerda. Essa detecção que definirá o sentido da primeira curva. Se houver uma limitação à esquerda, a primeira curva será à direita, ou se houver limitação à direita, a primeira curva será à esquerda. Após essa definição, o Robô seguirá em frente até encontrar um obstáculo, que o fará efetuar a curva. Esse procedimento será feito até o final do terreno.

Ao final, o autor apresenta resultados que atenderam os objetivos do trabalho em terrenos planos, com limitações e com grama de 30mm de altura. Como sugestões de melhoria, o autor propõe a utilização de *encoders* acoplados aos motores de corrente contínua. Ainda, o autor propõe o uso de GPS (*Global Positioning System),* afim de melhorar o posicionamento e a interação com objetos, permitindo um melhor controle do processo do corte de grama.

2.2.3 Robô Explorador De Labirintos 2D

Esse trabalho [Meneguele 2011] propõe um projeto de construção de um robô autônomo, capaz de explorar e solucionar um labirinto 2D. O robô é composto por um conjunto de sensores que fornecem informações para as tomadas de decisões. A tomada de decisão é baseada no algoritmo Seguidor de Parede (*Wall follower*), também conhecido como Regra da Mão Esquerda/Direita.

O veículo foi implementado de forma que seja capaz de encontrar a saída a partir de qualquer ponto inicial em que ele fosse largado. Assim que a saída for encontrada, o robô, posto no mesmo ponto inicial, é capaz de encontrar a saída percorrendo o menor caminho.

Algumas dificuldades foram apontadas pelo autor, tais como: o alinhamento sobre o caminho (onde o veículo andava em zigue-zague), os *encoders* (em que houve dificuldade na fixação dos componentes de forma que eles ficassem em uma posição desejável).

2.2.4 Arquitetura Compacta Para Projeto De Robôs Móveis Visando Aplicações Multipropósitos

Esse trabalho [Silva 2008] tem como proposta o desenvolvimento de uma arquitetura para robótica móvel, com diversas aplicações e com alta capacidade de processamento. O projeto tem como características mais relevantes o fato de ser modular, expansível, controlável remotamente e permitir o mapeamento de ambientes.

Para trabalhos futuros, o autor sugere a utilização de motores com caixa de redução, instalação de *encoders* no eixo do motor, melhorias na parte de comunicação com equipamentos externos, como o computador.

2.2.5 Considerações sobre os trabalhos relacionados

Os trabalhos relacionados oferecem uma variedade de soluções e ideias que contemplam a proposta apresentada neste trabalho. Em linhas gerais, pode-se perceber uma convergência no sentido da adoção de estratégias de controle para definição de rotas e para solucionar alguns problemas em comum. Por exemplo, no trabalho de

Teixeira (2013) é possível identificar as dificuldades que o autor teve em relação à velocidade dos motores.

Por outro lado, no trabalho de Meneguele (2011) e no de Silva (2008) foi identificada uma possível barreira para o presente trabalho, relacionada com a sensibilidade dos sensores (dispositivos que fornecem dados para a tomada de decisões). Por exemplo, os valores fornecidos pelos sensores podem fazer com que sejam tomadas decisões diferentes em um curto intervalo de tempo, o que pode fazer com que o robô desloque-se em zigue-zague

Ainda, no trabalho de Silva (2008) aceita-se um argumento que tende a convergir com a proposta deste trabalho, o de que a modularidade e a flexibilidade do sistema podem permitir a sua generalidade para múltiplas aplicações.

3. Proposta

Com base na análise do estado da arte, observa-se uma lacuna nos trabalhos já desenvolvidos, que consiste justamente no desenvolvimento de um sistema computacional modular e flexível capaz de ser utilizado para controlar veículos autônomos em diversos ambientes. Buscando tornar o sistema modular e escalável, propõe-se um sistema de controle distribuído, onde o veículo apenas enviará os valores obtidos pelos sensores e receberá comandos de ação. Assim, o sistema de controle não restringe-se à capacidade de processamento da placa Arduino, podendo ser hospedado por plataformas computacionais de múltiplas funcionalidades.

3.1 Projeto

A proposta desse trabalho apresenta um sistema (de controle de veículo autônomo terreste) modular e capaz de ser utilizado para diversos fins. Para intensificar essa modularidade, optou-se por criar um sistema distribuído, em que são implementadas apenas rotinas básicas no veículo (no microcontrolador) e o sistema de controle fica disponível em um outro dispositivo com capacidade variada de processamento (*tablets, smartphones, PCs*, etc). Assim, entre esses dois sistemas, haverá uma comunicação sem fio, em que o veículo enviará os valores obtidos dos sensores para o sistema de controle e o sistema de controle receberá essas informações (processando-as e armazenando-as localmente em um arquivo) e retornará uma ação para o veículo. A Figura 1 apresenta uma visão geral do fluxo de comunicação do sistema modular, em que um veículo autônomo terrestre de múltiplas funcionalidades é comandado, remotamente, por um sistema de controle distribuído.

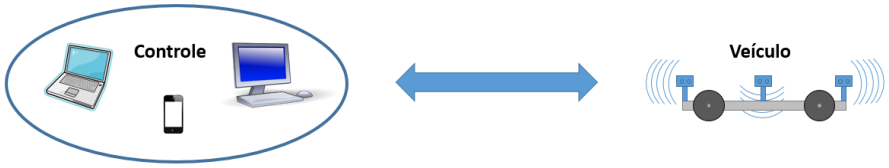


Figura 1 - Demonstração do fluxo de comunicação do sistema

Com a apresentação da visão geral da proposta, cabe-se ressaltar que o desenvolvimento deste projeto está baseado no uso das metodologias ágeis, as quais buscam auxiliar organizações a obterem um dinamismo para acompanhar as mudanças de requisitos em um projeto [Miller 2009]. Essas metodologias se destacam das tradicionais, principalmente, por darem maior prioridade à implementação das

funcionalidades através código ao invés da produção da documentação escrita, oferecendo respostas rápidas às possíveis mudanças [Erdogmus, Morisio, Torchiano 2005].

Diante disso, a metodologia utilizada neste trabalho é a *Feature Driven Development* (FDD) pelo fato dela atender os requisitos do domínio da aplicação e por ela possuir características importantes que permitem reprojetar quando necessário, realizando entregas frequentes, mensuráveis e repetitivas.

A metodologia FDD consiste na execução de cinco processos: desenvolver modelo inicial, criar lista de funcionalidades, planejar por funcionalidade, arquitetar por funcionalidade e construir por funcionalidade. O primeiro processo resulta na criação do modelo, uma arquitetura inicial, chamado de *object model*. Este modelo conceitual é criado após a análise do escopo e objetivos do projeto e pode tomar forma como um diagrama de classes, diagrama de domínio ou outra forma que ilustre uma estrutura física. O modelo inicial é manutenido durante as diversas iterações que o projeto vai executando, mantendo-se atualizado até a conclusão do projeto [Palmer e Felsin 2002].

a) Modelo conceitual de *software* do *smartphone*

O sistema de controle distribuído é projetado para um dispositivo externo, em que também foi modelado um diagrama de classes. Na Figura 2, é apresentado o diagrama de classes do sistema desenvolvido para Android.

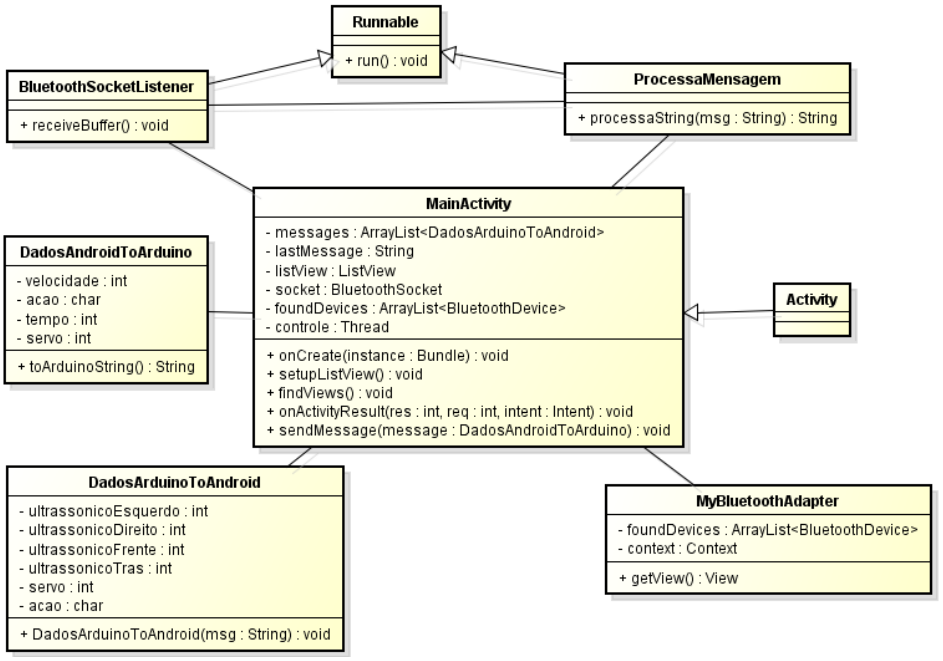


Figura 2 - Diagrama de classes do sistema Android

O diagrama de classes do sistema desenvolvido para Android apresenta oito classes usadas para recebimento das informações, processamento e envio de informações para o veículo. A classe MainActivity extende de Activity, ou seja, ela é uma classe usada para criar telas no Android. Assim, essa classe é a principal, pois é nela que são instanciadas as outras classes e o algoritmo de controle é implementado.

b) Modelo conceitual de *software* do veículo

A Figura 3 ilustra o diagrama de classes do sistema desenvolvido para o microcontrolador, as quais auxiliam na obtenção de valores dos sensores e no gerenciamento dos motores. Ainda, existe uma classe principal chamada de Controle, na qual são pré-processadas as informações enviadas ao sistema de controle externo.

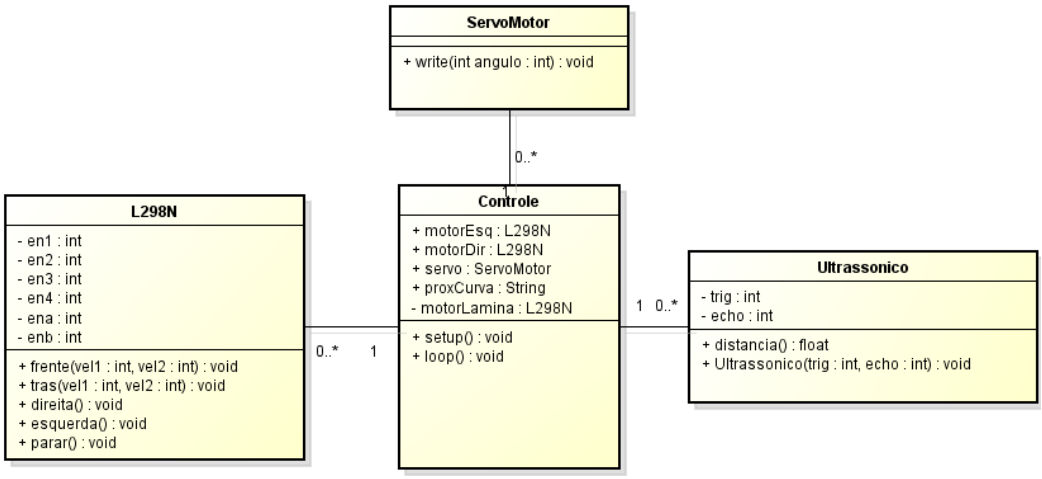


Figura 3 - Diagrama de classes do sistema do veículo

No sistema do veículo existem quatro classes, a L298N destinada para o gerenciamento dos motores, a Ultrassonico que facilita a obtenção dos valores lidos pelos sensores, a classe ServoMotor que possibilita a interação com servo motores e, por fim, tem-se a classe Controle em que existem instâncias das classes anteriores e dois métodos padrão do Arduino, sendo *setup()* o responsável por inicializar as portas e criar os objetos e o método *loop()* que é executado enquanto o Arduino estiver ligado.

c) Modelo conceitual de *hardware* do veículo

Quanto à parte de modelagem de *hardware*, a representação da estrutura do veículo desenvolvido é obtida a partir do *software Fritzing*. A Figura 4 demonstra a modelagem de *hardware*.

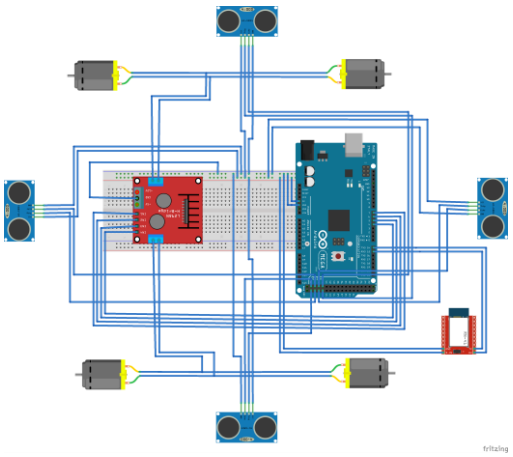


Figura 4 - Esquema de hardware

Na modelagem são apresentados sensores (representados por sensores distância), atuadores (representados por motores) e um controle (que é o Arduino). Ainda, é apresentado um componente para comunicação com o sistema externo.

d) Modelo conceitual da comunicação entre o veículo e o *smartphone*

A comunicação entre os dois sistemas (veículo e sistema de controle) é feita através de comunicação sem fio, afim de evitar que hajam cabos entre os dois dispositivos, possibilitando um distanciamento e uma autonomia maior. A Figura 5 ilustra a comunicação entre os dois sistemas.

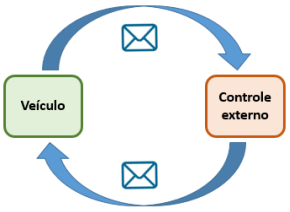


Figura 5 - Modelo de comunicação

3.2 Implementação

A implementação do sistema seguiu a modelagem projetada inicialmente, em que se são considerados quatro segmentos: a) *software* externo de controle para *smartphone;* b) *software* para o veículo; c) *hardware* do veículo e d) sistema de comunicação.

a) Implementação do *software* de controle para *smartphone*

Na aplicação de controle (externo) foi desenvolvida uma aplicação para Android, pois grande parte dos dispositivos de computação móvel possuem interfaces para comunicação sem fio. Para o desenvolvimento da aplicação foi utilizada a linguagem Java que é a linguagem padrão do Android e também é orientada a objetos. Entre as classes implementadas, uma que tem grande importância para a aplicação é a que recebe as informações enviadas pelo veículo, conforme pode ser observada abaixo na Figura 6.

A
```
public class BluetoothSocketListener implements Runnable {
    @Override
B   public void run() {
        int bufferSize = 1024;
        byte[] buffer = new byte[bufferSize];
        try {
            InputStream instream = socket.getInputStream();
            int bytesRead = -1;
            String message = "";
C           while (true) {
                if (instream.available() != 0) {
                    bytesRead = instream.read(buffer);
                    while(bytesRead != -1){
                        message = new String(buffer, 0, bytesRead);
                        handler.post(new ProcessaMensagem(textView, message));
                        bytesRead = instream.read(buffer);
                    }

                } else{
                    message = "";
                }
            }
        } catch (IOException e) {
            Log.d("BLUETOOTH", e.getMessage());
        }
    }
}
```

Figura 6 - Classe BluetoothSocketListener

Em 6A está a declaração da classe que implementa a interface *Runnable*, pois esta classe deverá ter um trecho de código executável para ser executado em uma *thread*. A interface *Runnable* disponibiliza o método *run()* (6B) que é chamado ao executar esta classe. Por fim, o trecho destacado por 6C representa o recebimento e a

leitura dos *bytes* enviados pelo veículo e transformado em *String* para processamento posterior (classe ProcessaMensagem).

Outra parte crucial do sistema é a do controle, em que são processadas as informações recebidas e é retornada uma ação para o veículo. A Figura 7 ilustra um trecho da implementação do algoritmo de controle.

```
A
Runnable r = new Runnable() {
    @Override
    public void run() {
B       while (flagThread) {
            try {
                if (messages != null && messages.size() > 0
                        && last < messages.size()) {
C                   final DadosArduinoToAndroid dados = new DadosArduinoToAndroid(
                            messages.get(messages.size() - 1));
D                   runOnUiThread(new Runnable() {

                        @Override
                        public void run() {
                            text_messages.setText(dados.toString());
                        }
                    });
E                   if (dados.ultrassonicoFrente < 50
                            && dados.acao.equals("F")) {
                        DadosAndroidToArduino d = new DadosAndroidToArduino(
                                0, 'P', 0, 90);
                        sendMessage(socket, d.toArduinoString());
                    }
                }
                SystemClock.sleep(200);
            } catch (Exception e) {
                Log.i("SSS", "EXC:" + e.getMessage());
            }
        }
    }
};
t = new Thread(r);
flagThread = true;
t.start();
F
```

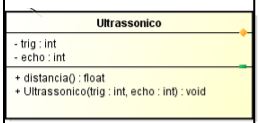
Figura 7 - Trecho do algoritmo de controle implementado para Android

Primeiramente (7A) é criado um objeto do tipo Runnable que conterá códigos que serão executados por uma *thread* (7F). Ao ser instanciada e executada a *thread* (7F) existe uma condição (flagThread) que é habilitada ou desabilitada pelo usuário através da interface gráfica, ou seja, ao ser habilitada a "flagThread" o algoritmo de controle entra em ação e só vai parar quando o usuário desabilitar a "flagThread". Ao iniciar a execução, é buscada a última mensagem recebida pelo dispositivo e convertida para um objeto da classe DadosArduinoToAndroid (7C). Em 7D a mensagem é mostrada na tela para o usuário acompanhar o que está sendo processado. Já em 7E há um exemplo de tomada de decisão, onde é verificado o valor fornecido pelo sensor da frente em que, se for menor que 50 e a ação for "F" (simboliza "andar para frente"), então é mandada uma mensagem para o veículo parar.

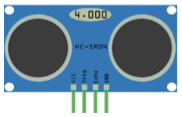
b) Implementação do *software* do veículo

O sistema implementado para o veículo foi desenvolvido na linguagem C++ seguindo o diagrama de classes proposto no modelo conceitual. A Figura 8 demonstra um exemplo de implementação de uma classe modelada no diagrama de classes e implementada via código.

Classe



Sensor



Cabeçalho

```
#include "Arduino.h"
class Ultrassonico
{
    private:
        int trig, echo;
    public:
        Ultrassonico(int trig, int echo);
        int distancia();
};
#endif
```

Implementação

```
#include "ultrassonico.h"
Ultrassonico::Ultrassonico(int _trig,int _echo ){
  trig = _trig;
  echo = _echo;
  pinMode(trig, OUTPUT);
  pinMode(echo,INPUT);}
int Ultrassonico::distancia(){
  digitalWrite(trig, LOW);
  delayMicroseconds(2);
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);
  unsigned long duracao = pulseIn(echo, HIGH);
  int distancia = duracao / 58;

  return distancia;
}
```

Figura 8 - Exemplo de classe implementada

Uma classe é uma abstração de um conjunto de objetos com características parecidas. A Figura 8 apresenta uma classe para o sensor ultrassônico (ilustrado abaixo da classe), ou seja, é um modelo lógico, com atributos e métodos que serve de base para a implementação. Uma das convenções da programação em C++ é a criação de um arquivo de cabeçalho (.h) e um arquivo de implementação (.cpp). No cabeçalho é criada uma representação do que deve ser implementado, ou seja, são declarados os métodos e atributos. Na implementação é desenvolvida a lógica de cada método. Nesse exemplo, existem dois atributos privados "trig" e "echo" que representam os pinos do sensor ultrassônico (representado na imagem acima).

Outra classe importante no sistema do veículo é a de controle, pois é ela que utilizada as outras classes e onde ocorre o fluxo de execução do início ao fim. A Figura 9 demonstra um exemplo de utilização das demais classes criadas.

A Declaração

```
L298N *ponteH;
Ultrassonico *ultrassonico_frente;
Ultrassonico *ultrassonico_tras;
Ultrassonico *ultrassonico_direita;
Ultrassonico *ultrassonico_esquerda;
Servo motor_frente;
```

B Inicialização

```
void setup() {
    ponteH = new L298N(42,43,44,45,4,5);
    ultrassonico_frente = new Ultrassonico(48,49);
    ultrassonico_tras = new Ultrassonico(50,51);
    ultrassonico_esquerda = new Ultrassonico(52,53);
    motor_frente.attach(3);
}
```

C Utilização dos métodos

```
void loop() {
    ponteH->frente(velocidade,velocidade);
    int distanciaF = ultrassonico_frente->distancia();
    int distanciaT = ultrassonico_tras->distancia();
    int distanciaD = ultrassonico_direita->distancia();
    int distanciaE = ultrassonico_esquerda->distancia();
    if(distanciaF<10 && acao == 'F'){
        ponteH->parar();
    }
```

Figura 9 - Exemplo de utilização de outras classes

Em 9A tem-se as declarações dos objetos que serão utilizados. Já em 9B tem-se a inicialização desses objetos, onde são passadas as portas do componente no método construtor. E em 9C tem-se um exemplo de utilização dos objetos dentro do método *loop()*, onde o veículo recebe o comando de andar para frente com uma determinada velocidade, após são lidos os valores dos sensores e é validada a distância da frente: se for menor que dez e o veículo está indo para frente, os motores devem parar, caso contrário o processamento segue.

c) Implementação do *hardware*

Quanto a parte de *hardware*, foi implementado um protótipo de veículo terrestre, em que foram acoplados sensores ultrassônicos, uma Dupla Ponte H (L298N) para controlar os motores, um módulo *bluetooth* para enviar e receber os dados do sistema de controle externo e um Arduino Mega que continha um sistema para manipulação dos componentes. A Figura 10 demonstra o protótipo com seus componentes.

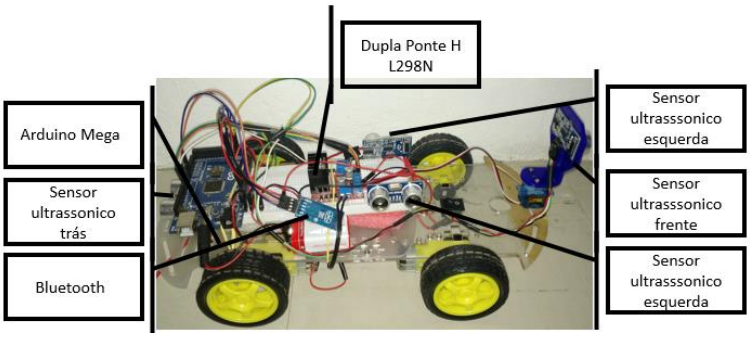


Figura 10 - Protótipo de veículo terrestre com seus principais componentes

d) Implementação do sistema de comunicação

A comunicação entre os dois sistemas é realizada através da tecnologia *bluetooth*, pois é de baixo consumo de energia e possui um custo reduzido para ambos os dispositivos utilizados. O envio das informações é dado por uma sequência de caracteres, conforme pode ser observada abaixo.

$37*UF:62;UT:3;EU:2;UD:300;A:F;S:90;#

A sequência de caracteres contém informações dos sensores, do servo motor e um número que indica a quantidade de caracteres que foram enviados. Uma descrição das informações contidas na *string* exemplificada anteriormente pode ser dada como:

"$" - Indica o início da sequência.
"37" – indica a quantidade de caracteres que foram enviados.
"*" - indica que após esse caractere, serão enviadas informações dos sensores.
"UF:62" – UF indica "Ultrassônico Frente" e 62 indica o valor lido desse sensor.
"UT:3" – UF indica "Ultrassônico Trás" e 3 indica o valor lido desse sensor.
"UE: 2" – UF indica "Ultrassônico Frente" e 2 indica o valor lido desse sensor.
"UD:300" – UF indica "Ultrassônico Frente" e 300 indica o valor lido desse sensor.
"A:F" – indica a "Ação" e F significa que está indo para "Frente".

4 Cenário de testes/Validação

Este trabalho tem como proposta o desenvolvimento de um sistema computacional, em que se tem um veículo autônomo terrestre para múltiplas funcionalidades, controlado por um sistema externo implementado em um *smartphone* com sistema operacional Android.

Dentro desse contexto, o sistema de controle foi desenvolvido de tal forma que o veículo buscasse explorar um ambiente, visitando todos os locais do mesmo. Assim, foram realizados testes a fim de validar o algoritmo de controle desenvolvido.

Os testes foram executados em um ambiente *indoor* com uma área de aproximadamente 9 metros quadrados. O ambiente esteve cercado para auxiliar na detecção dos limites do mesmo por parte do veículo. A Figura 11 demonstra o estudo de caso utilizado para validar a proposta.

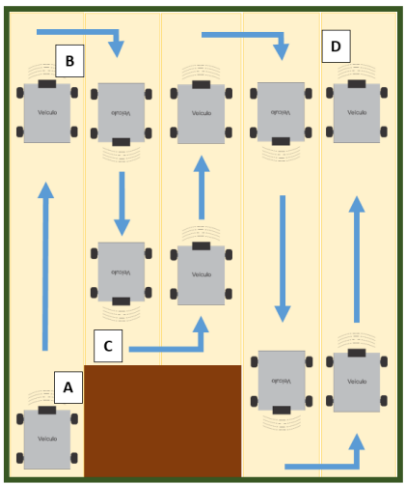


Figura 11 - Cenário de testes

Considerando o cenário de testes, em 11A, o veículo encontra-se em sua posição inicial, em uma extremidade do ambiente. De acordo com a lógica implementada no sistema de controle, o veículo deve se locomover para frente enquanto não encontrar um obstáculo. Ao se deparar com um limite (11B), o veículo deve parar e efetuar uma curva. Em 11B, observa-se que o veículo efetuou a curva para a direita, portanto, sua próxima curva será para a esquerda, afim de não passar por um lugar já visitado. Em 11C observa-se que o veículo encontra um obstáculo e efetua a curva (agora para a esquerda) antes da limitação do ambiente. Como critério de parada, o sistema de controle avalia os dados obtidos do que já foi percorrido e se o veículo não poder efetuar a curva para o sentido determinado na curva anterior (11D), o sistema considera que o ambiente foi todo explorado.

5 Resultados e discussões

O projeto inicial previa a criação de um sistema autônomo para veículos terrestres controlado por Arduino, tanto para ambientes *indoor* como *outdoor*. Em ambientes *indoor* buscava-se controlar o veículo através de sensores de distância. Já em ambientes *outdoor* pensou-se em controlar o posicionamento do veículo através de GPS.

A utilização do GPS trouxe problemas que tornaram inviável a utilização desse componente, pois eram fornecias coordenadas muito imprecisas para o mesmo ponto, ou seja, com o veículo parado o GPS marcava posições diferentes e com erros aleatórios. Assim, optou-se por focar o trabalho apenas na utilização de sensores de distância.

No decorrer do projeto, percebeu-se que distribuir o algoritmo de controle tornaria o sistema mais modular e escalável, ou seja, faria com que se pudesse implementar o controle em equipamentos com maior capacidade de processamento e proporcionaria a adição de novos componentes que demandassem maior processamento. Assim, como estudo de caso, implementou-se o algoritmo de controle em um *smartphone* Android.

Para que os dois sistemas pudessem se comunicar, criou-se um padrão de comunicação, em que as informações eram transmitidas em uma sequência de caracteres separadas por ponto e vírgula (;). Assim, tem-se a possibilidade de criar sistemas de controle em diversos aparelhos, porém utilizando o mesmo padrão de comunicação.

Um problema encontrado nessa comunicação foi a quantidade de informações que eram geradas pelo veículo, ou seja, os valores lidos dos sensores variavam muito, o que gerava uma nova informação a ser enviada ao controle. Dessa forma, eram enviadas várias *strings* em um curto espaço de tempo. Como os valores lidos dos sensores variavam muito pouco, optou-se por calcular uma margem de valor para cada sensor, assim uma informação era considerada nova quando o valor lido ultrapassava a margem determinada. A Figura 12 demonstra a redundância de informações geradas.

```
$37*UF:62;UT:3;UE:2;UD:301;A:F;S:90;#
$37*UF:63;UT:3;UE:2;UD:299;A:F;S:90;#
$37*UF:62;UT:3;UE:3;UD:301;A:F;S:90;#
$37*UF:64;UT:3;UE:3;UD:300;A:F;S:90;#
$37*UF:61;UT:4;UE:2;UD:299;A:F;S:90;#
$37*UF:61;UT:3;UE:2;UD:301;A:F;S:90;#
$37*UF:62;UT:4;UE:3;UD:302;A:F;S:90;#
$37*UF:63;UT:3;UE:2;UD:300;A:F;S:90;#
```

Figura 12 - Valores transmitidos do veículo

Outro desafio encontrado (e um dos mais importantes) foi o tempo de comunicação entre os dispositivos, em que as informações recebidas e enviadas ao sistema de controle não eram instantâneas, o que teve que ser levado em conta. Para contornar esse problema, aumentou-se a "sensibilidade" com os valores recebidos do veículo, ou seja, se a lógica imposta para o veículo parar era de quando o sensor da frente marcar uma distância menor ou igual a 10cm, aumentou-se essa distância (para 20cm, por exemplo) afim de considerar que o veículo estará andando até receber o comando de parar.

6 Conclusões e trabalhos futuros

Este trabalho apresentou a proposta de um sistema flexível e modular para veículos autônomos de múltiplas funcionalidades que abrange o uso de recursos como o Arduino, sensores, comunicação sem fio e dispositivos com sistema operacional Android.

Uma análise do estado da arte dentro do domínio da aplicação forneceu subsídios para o desenvolvimento do trabalho, pois envolveu o uso do Arduino aplicado na automatização de tarefas, bem como na criação de veículos com autonomia parcial ou total.

Dentro do contexto proposto no trabalho, é possível destacar vantagens como a modularidade e a flexibilidade do sistema para o uso de novos componentes, possibilitada pela distribuição do sistema de controle.

Uma das principais contribuições da proposta é o algoritmo capaz de adaptar-se em diversos ambientes, uma vez que ela possui como caraterística a flexibilidade e a modularidade, permitindo a inserção de novos módulos com funcionalidades diversas, o que diferencia este trabalho dos correlatos.

No que tange a metodologia para desenvolvimento deste trabalho, optou-se pela FDD, que é uma metodologia ágil que busca o desenvolvimento por funcionalidades, permitindo fazer entregas frequentes, mensuráveis e repetitivas e que aborda diretamente a produção de sistemas utilizando uma metodologia de desenvolvimento simples de implementar e acompanhar, inclusive reprojetar algo no desenvolvimento.

Sob o ponto de vista crítico, pode-se destacar que este projeto possui características semelhantes aos projetos de Teixeira (2013), Devenz (2012), Meneguele (2011) e de Silva (2008), principalmente no que se refere às tecnologias utilizadas e ao modo em que buscam oferecer equipamentos autônomos. Por outro lado, o projeto busca contribuir, justamente, com um algoritmo híbrido que seja capaz de ser implementado para ambientes *indoor* e *outdoor*, numa tentativa de criar uma abordagem capaz de atender uma demanda pelo gerenciamento de veículos autônomos de uso genérico.

Trabalhos futuros poderão ser desenvolvidos buscando incluir novos sensores e componentes ao veículo, bem como poderão haver alterações no algoritmo de controle, buscando aumentar o campo de aplicação da proposta apresentada. Um campo a ser explorado é a estereoscopia, a qual busca-se obter informações de um espaço tridimensional podendo-se mensurar distâncias das câmeras de vídeo até certo ponto, através de imagens. Ainda, pode-se medir e analisar o tempo de comunicação entre o sistema de controle e o veículo.

Referências

Arduino. (s.d.). Acesso em 27 de Abril de 2014, disponível em www.arduino.cc

Deloura M. (2000) "*Game Programming Gems*" (Vol.I). Charles River Media Ed.

Devenz, G. (2012) "Desenvolvimento De Um Protótipo de Robô Cortador De Grama", Acesso em Março de 2014, disponível em: http://www.professorpetry.com.br/Ensino/Defesas_Pos_Graduacao/Defesa%2021_Gabriel%20Devenz_Desenvolvimento%20de%20um%20Prototipo%20de%20Robo%20Cortador%20de%20Grama.pdf

Erdogmus, H.; Morisio, M.; Torchiano, M. (2005) "*On the effectiveness of the test-first approach to programming.*" IEEE Trans. Softw. Eng., IEEE Press, Piscataway, NJ, USA, v. 31; Acesso em Maio de 2014, Disponível em: http://dx.doi.org/10.1109/TSE.2005.37

Fox, D., Burgard, W., Dellaert, F. e Thrun, S. (1999) "*Monte carlo localization: Efficient position estimation for mobile robots*", *In Proceedings of the National Conference on Artificial Intelligence* (AAAI), Orlando.

INCT (2012) "GT5 - Veículos Aquáticos e Subaquáticos Autônomos", Acesso em Maio de 2014, disponível em: http://www.inct-sec.org/br/grupos-de-trabalho/gt5-veiculos-aquaticos-e-subaquaticos-autononomos

Makarenko, A.A.; Williams, S.B.; Bourgault, F.; Durrant-Whyte, H.F. (2002) "*An Experiment in Integrated Exploration*" Australian Centre for Field Robotics, Sydney Univ., NSW, Austrália; Acesso em Maio de 2014, disponível em: http://ieeexplore.ieee.org/xpl/abstractCitations.jsp?tp=&arnumber=1041445&queryText%3DAn+experiment+in+integrated+exploration

Mcroberts, M. "*Beginning Arduino.*" 1. ed. [S.l.]: Apress, v. I, 2010.

Meneguele, B. E. D. O. (2011) "Robô Explorador de Labirintos 2D", Acesso em Março de 2014, disponível em http://www.pessoal.utfpr.edu.br/msergio/Monog-11-1-Robo-Explorador-Labirinto.pdf

Miller L, Sy D, (2009) "*Agile user experience SIG*", Conference on Human Factors in Computing Systems, Boston.

Moraes, C. C. D.; Castrucci, P. D. L. (2001) "Engenharia de Automação Industrial" Rio de Janeiro: LTC Editora.

Olson, C. F. (2000) "*Probabilistic self-localization for mobile robots. IEEE Transactions On Robotics And Automation*", vol. 16.

Palmer, Stephen R.; Felsin, John. M. (2002) "*A Practical Guide to Feature-Driven Development.*" Prentice Hall.

Paschoal, R. A. (2009) "Robô para Mapeamento de Ambientes Estruturados" Acesso em Março de 2014, disponível em http://www.leandrohsouza.com.br/engcomp/attachments/article/85/ROB%C3%94S%20PARA%20MAPEAMENTO%20DE%20AMBIENTE-%20Roberto%20Paschoal.pdf

Rabin, S. (2002) "*AI Game Programming Wisdom.*" Charles River Media.

Schueroff, J. (2014) "Desenvolvimento de um Sistema de Visão Artificial para um Robô Explorador" Acesso em Março de 2014, disponível em http://siaiweb06.univali.br/seer/index.php/acotb/article/view/5323/2783

Silva, A. (2008) "Arquitetura Compacta Para Projeto De Robôs Móveis Visando Aplicações Multipropósitos", Acesso em Março de 2014, disponível em http://www.teses.usp.br/teses/disponiveis/18/18153/tde-17032009-115712/en.php

Silveira, P. R. Da , Winderson Eugenio dos Santos. (2004) "Automação e controle discreto." Érica, São Paulo.

Stallings, W. (2005) "Redes e sistemas de comunicação de dados", 5ª Ed. ed. Campus.

Tanenbaum, A. S.; Souza, Vandenberg D. de (trad.); Jamhour, Edgard (rev.) (2003). "Redes de computadores." Rio de Janeiro: Elsevier.

Teixieira, J. R. S. (2013), "Desenvolvimento De Um Veículo Autônomo Que Detecta Obstáculos", Trabalho Final de Graduação II – Curso de Ciência da Computação, Centro Universitário Franciscano, Santa Maria-RS.

Teck, (2010), "*Google Automated Self Driving Car Test Drove Successfully!*", Acesso em 28 de março de 2014, disponível em: http://teck.in/google-automated-self-driving-cars-tested-successfully.html#ixzz30Bsc3n3R

Thrun, S.; Wolfram, B.; Fox, D. (2005) "*Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*", *Massachusetts Institute of Technology* (MIT) *Press*.

Thrun, S. (2002) "*Robotic Mapping: A Survey*." Carnegie Mellon University.

Tsai, Lung-Wen. (1999) "*Robot analysis: the mechanics of serial and parallel manipulators.*" John Wiley & Sons.

Wendling, M. (2010) "Sensores" V2.0. Acesso em Maio de 2014, disponível em

http://www2.feg.unesp.br/Home/PaginasPessoais/ProfMarceloWendling/4---sensores-v2.0.pdf