UGV Solutions: UGVs e suas aplicações para Cidades Inteligentes

Daniel Rodrigues Ferraz Izario
daniel_izario@hotmail.com
INATEL

Letícia Paiva Magalhães
leticiapaiva@get.inatel.br
INATEL

Carlos Nazareth Motta Marins
carlosn@inatel.br
INATEL

Resumo: Existe uma crescente demanda por sistemas robóticos, entre eles encontram-se os veículos terrestres não-tripulados (VTNT) que são robôs terrestres desenvolvidos para serem utilizados como uma extensão dos recursos humanos. O desenvolvimento de um veículo controlado remotamente envolve, principalmente, o controle de seus mecanismos básicos de condução: aceleração, frenagem e direção. O objetivo é mostrar a implementação dos projetos UGV Solutions, em um sistema de sensoriamento, monitoramento em tempo real e aplicabilidades para uso em cidades inteligentes, utilizando a ideia dos VTNTs. Toda essa ideação é desenvolvida em um diagrama que apresenta como o sistema pode ser aplicado nas cidades inteligentes.

Palavras Chave: UGVs - Cidades Inteligentes - Arduino - Monitoramento - Sensoriamento

1. INTRODUÇÃO

O desenvolvimento de Veículos Terrestres Não-Tripulados (VTNT), em inglês, *Unmanned Ground Vehicles* (*UGV*) é dividido em sistema automático de controle, pelo qual os mecanismos recebem comandos para funcionamento e efetuam medições, sistema mecânico, sistema de comunicação que envolve os sistemas de transmissão e recepção de dados, e o sistema elétrico, que em geral proporciona toda a alimentação dos sistemas anteriores. Na automação são aplicadas técnicas computadorizadas ou mecânicas para diminuir riscos ao ser humano e deixar o projeto viável para sua aplicação.

Baseado na necessidade de monitorar áreas remotas, de alto risco, ou até mesmo auxiliar no monitoramento de condomínios, foram desenvolvidos os projetos apresentados na Figura 1. São 4 projetos: *Bender UGV*, *Military UGV*, *Robotic Arm* e *RF Remote Control*.

Figura 1: Projetos *UGV Solutions*.

Existem diversas aplicações para os projetos, porém será apresentado aplicabilidades para uso em cidades inteligentes, onde a gestão deve estar presente para que toda a sociedade possa estar integrada com a tecnologia.

Esses veículos possuem sensores que podem monitorar e alertar, por exemplo, que determinada região está sobre risco de alagamento ou incêndio. Tudo isso só é possível devido a sensores integrados aos veículos que enviam dados como temperatura, luminosidade, nível de gás carbônico, humidade, em tempo real.

É possível acoplar, caso necessário, um braço robótico ao veículo, observando que nesse trabalho será demonstrado a parte de *software* para essa funcionalidade e um protótipo visual em acrílico.

Todo o controle dos veículos é feito por um controle remoto via rádio frequência (RF). Um suporte de câmera foi instalado na frente do veículo de modo que todo o trajeto do veículo seja acompanhado em tempo real.

O restante desse artigo é organizado como se segue, no tópico 2, foi feita a explicação sobre os 4 projetos desenvolvidos, no 3, um detalhamento sobre a comunicação RF, no 4, o *website* é analisado com suas abas de desenvolvimento; em 5 é apresentado o diagrama de gestão e integração do sistema na sociedade, e pôr fim à conclusão.

2. UGV SOLUTIONS

2.1. BENDER UGV

O primeiro projeto desenvolvido foi o *Bender UGV*, apresentado na Figura 2, tendo como principal desafio criar um veículo capaz de proporcionar uma relação autonomia x peso viável para as aplicações de monitoramento.

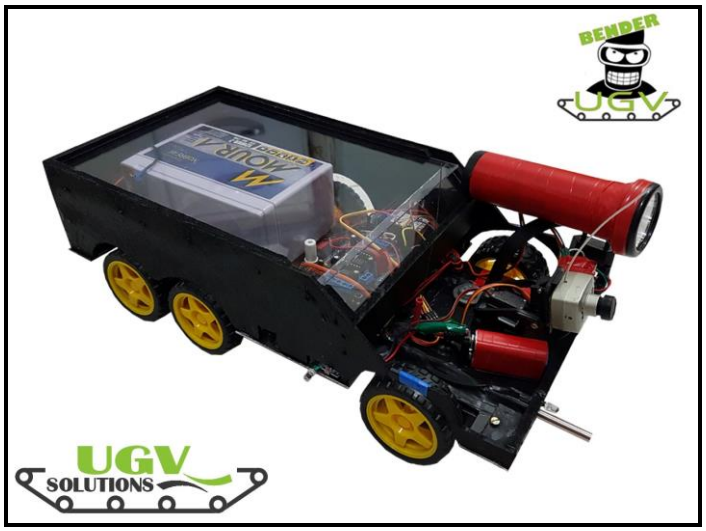

Figura 2: *Bender UGV.*

Deste modo, o chassi foi construído em alumínio (material resistente e ao mesmo tempo leve), com 6 motores *DC*. Cada roda tem seu aro fabricado em plástico resistente e pneus emborrachados com um certo grau de robustez.

O conjunto de rodas/motor foi fixado à estrutura do veículo em uma placa utilizando parafuso e porca, sendo utilizadas tiras de alumínio para proteger a carenagem do motor. A fixação foi feita de forma a permitir um deslocamento do conjunto em 360º. Tal deslocamento permite a rotação do veículo por direções distintas, conforme comando dado pelo controle remoto RF (PAULI, 1996).

Todos os motores foram conectados por meio de uma *Shield L293d - Driver Ponte H*, que faz o controle dos motores do veículo e do suporte da câmera, este circuito/*shield* foi o que apresentou as melhores características de acionamento. Sendo adequado para o controle de velocidade e direção dos motores utilizados, facilitando sua mobilidade durante a realização dos movimentos nas diferentes direções e ângulos: frente, ré, direita e esquerda.

Este veículo possui uma câmera que envia imagens em tempo real a um *software* que auxilia na segurança do ser humano, possibilitando que o risco de morte ao necessitar monitorar uma área de risco seja mitigado, ou seja, o veículo substitui o ser humano e proporciona a ele uma visão mais afastada do local.

Para viabilizar o controle do veículo foi necessário buscar informações para fazer todo o levantamento de dados e assim, desenvolver um sistema de controle capaz de executar o as ações. Tal levantamento serviu como incentivo para a identificação e determinação da melhor forma de controle e para a escolha dos recursos de *hardware* necessários para a sua construção e automação (SANTOS, 2011).

A construção e automação do sistema foi feita em duas etapas, sendo desenvolvidas em paralelo. A construção do *hardware* integrado à plataforma *arduino* e a elaboração de vários algoritmos, baseados na linguagem C, foram mais adequados ao propósito do projeto. Para maior autonomia de energia foi utilizada uma bateria recarregável de 12V / 7A.

Todo esse conceito de *hardware* é apresentado na Figura 3, sendo possível observar todas as interligações existentes entre os componentes e o microcontrolador.

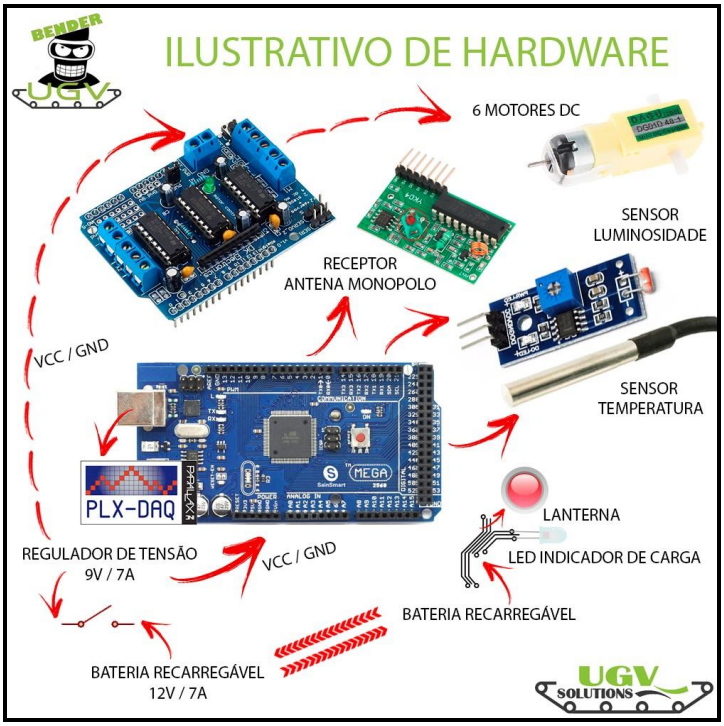

Figura 3: Ilustrativo do *hardware* - *Bender UGV*.

2.2. MILITARY UGV

No desenvolvimento do *Military UGV*, apresentado na Figura 4, foi feito um estudo para diminuir o tamanho e funções do projeto, criando assim, um veículo *UGV* para uma única função, que é fazer a medição de gases tóxicos em um local e trazer um alerta para o usuário por meio de um farol *LED* de emergência acoplado ao veículo.

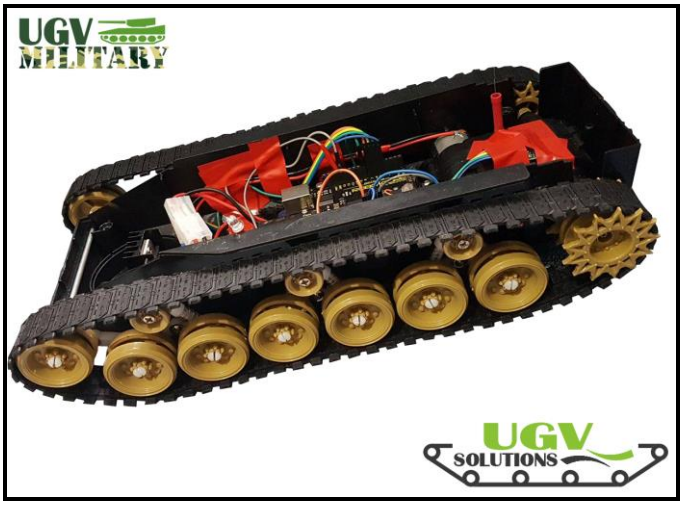

Figura 4: *Military UGV*.

Gases tóxicos e inflamáveis podem gerar grandes explosões, como são comprovadas por experiências já realizadas em laboratórios, transcorridos de vazamentos que geraram nuvens tóxicas ou explosivas no ambiente. Levando essa ideia em consideração, foi desenvolvido esse modelo de *UGV*. Nesse veículo, não foram feitos estudos para acionamento do circuito sem gerar faísca, para implementação deve-se levar em conta esse fato (LOPES, 2012).

Com a diminuição das funcionalidades, foi possível diminuir a bateria para 7,2V / 2500mA. O chassi é de polímero, que é um material resistente, com apenas 2 motores *DC*, utilizando lagartas, também conhecidas como esteiras, que são as rodas do tanque de guerra, foram desenvolvidas para atuarem em vários tipos de terreno, tendo uma melhor aderência ao solo.

A fixação dos motores foi feita na parte de trás do veículo para prender as esteiras permitindo um deslocamento do conjunto em 360º, tudo controlado via comando dado pelo controle remoto RF. Isso só foi possível por meio da utilização da *Shield L298n - Driver Ponte H*, que faz o controle dos motores do veículo, este circuito/*shield* foi o que apresentou as melhores características de acionamento.

Todo o conceito de *hardware* desenvolvido no protótipo é apresentado na Figura 5, nele é possível observar todas as interligações existentes entre os componentes e o microcontrolador.

Figura 5: Ilustrativo do *hardware* - *Military UGV*.

2.3. ROBOTIC ARM

O *Robotic Arm*, apresentado na Figura 6, foi criado com o intuito de poder ser anexado aos veículos *UGVs* ou não, essa possibilidade depende da utilização do mesmo no ambiente. Um *arduino UNO* tem a função de fazer o total controle do braço robótico, recebendo comandos enviados pelo controle remoto RF.

O Braço Robótico faz uso de 4 *Micro-Servos TowerPro MG90s*, que possibilitam uma grande variação nos eixos X, Y e Z. Além disso, permite que o mesmo possa pegar objetos pequenos no ambiente onde se encontra o veículo.

O controle de forma sincronizada, tendo uma resposta adequada, só foi possível devido a *Shield* Controladora Servo 16 canais, nela é possível o envio do sinal apenas uma vez e o canal mantém a posição *PWM* escolhida enquanto desejar, facilitando assim, o *arduino* identificar a qual servo deveria funcionar e em qual ângulo.

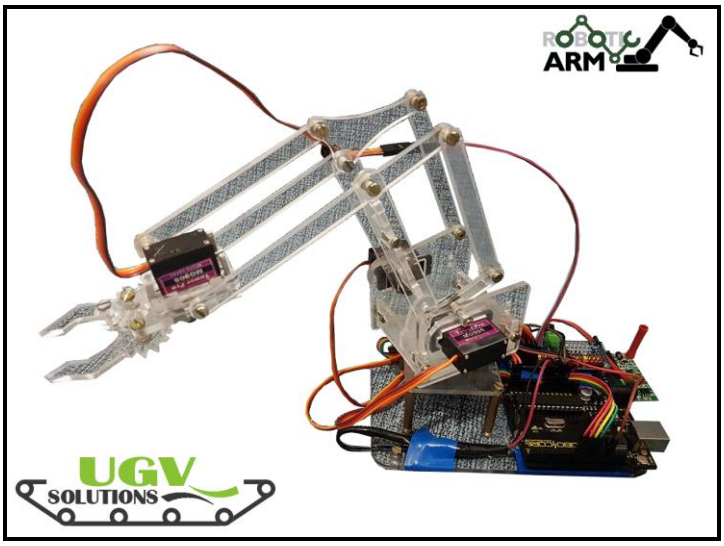

Figura 6: *Robotic Arm.*

Para conseguir que os servos estivessem nas posições solicitadas, foi preciso fazer um estudo, eles podem ser danificados se não for aplicada uma tensão de controle correta ou correr o risco de ter movimentos limitados, com isso, foi necessário desenvolver um algoritmo, capaz de adaptar cada servo a um ângulo inicial, utilizando como base as bibliotecas do *arduino*, "*Wire*" e "*Adafruit_PWMServoDriver*" (MONK, 2014).

Figura 7: Ilustrativo do *hardware* - *Robotic Arm*.

Um servo se move aproximadamente 180 graus, variando de um modelo para o outro. Portanto, para centralizar a posição é possível adotar 90 graus. Os pulsos mínimos (ANGULOMIN) e máximos (ANGULOMAX) são definidos como fixos para adequar e simplificar o código, mas isso, vai depender da necessidade do braço robótico. O código lê uma informação de entrada pela porta serial do *arduino*, que é o canal onde o servo está conectado e a sua posição, avaliada pelo ângulo desejado (de 0 a 180) e comandos para ligar ou encerrar, no caso, "*ON*" e "*OFF*".

Todo esse conceito foi viabilizado por meio de muitos testes para chegar a um código mais resumido e capaz de realizar todas as necessidades levantadas durante o processo de desenvolvimento do *Robotic Arm*, mas de forma geral acabou sendo útil para entender qual é o processo de comunicação de uma *shield* interligada com o *arduino*.

Todo o conceito de *hardware* desenvolvido no protótipo do braço robótico é apresentado na Figura 7, nele é possível observar todas as interligações existentes entre os componentes e o microcontrolador.

2.4. RF REMOTE CONTROL

O *RF Remote Control*, apresentado na Figura 8, foi criado para a comunicação com os *UGVs*, nele é possível dar comandos específicos via um transmissor RF, e os veículos contendo um receptor RF interpretam essa informação e executam.

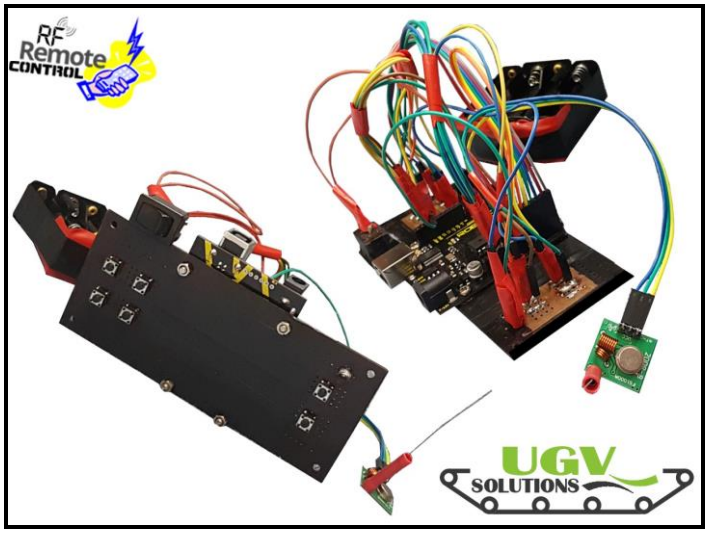

Figura 8: *RF Remote Control.*

Para o *arduino* ser capaz de interpretar as informações dos *inputs* dos botões e depois enviar esse dado via um transmissor RF em forma de uma variável *int* que controla qual o movimento o veículo deverá executar, utilizando como base a biblioteca "*VirtualWire*" (MONK, 2014).

Todo o conceito de *hardware* desenvolvido no protótipo é apresentado na Figura 9, nele é possível observar todas as interligações existentes entre os botões e o microcontrolador, além da conexão com o transmissor de sinal RF.

Figura 9: Ilustrativo do *hardware* - *RF Remote Control*.

3. COMUNICAÇÃO RF

Existem várias maneiras de fazer a comunicação entre sistemas que envolvem *arduino*, as mais comuns são o uso de *shields*: *bluetooth* e de RF (Rádio Frequência). A comunicação via *bluetooth* é amplamente utilizada no *arduino*, ela é uma forma simples e barata de enviar e receber informações remotamente, mas esse envio só funciona para curtas distâncias, o alcance do módulo segue o padrão *bluetooth*, que é de aproximadamente 10 metros (RIBEIRO, 2012).

Devido a essa limitação do *Shield Bluetooth*, os projetos do *UGV Solutions*, utilizam o Módulo RF Transmissor + Receptor 433 MHz, apresentado na Figura 10.

Figura 10: RF Transmissor + Receptor 433 MHz.

Existem vários modelos de antenas, todas elas executam uma tarefa importante nas redes de transmissão/recepção de sinal. É com elas, que acontece a transferência da energia a partir do transmissor para o meio onde se propagará, e daquele meio ao receptor. A eficiência desse sistema depende do desempenho dos fatores irradiantes ou das recepções a ele conectado, por isso se desenvolveram vários modelos de antenas, dentre estas, podem ser citadas as mais comuns: a monopolo e a dipolo (RIBEIRO, 2012).

A antena monopolo é muito utilizada atualmente, o modelo monopolo de quarto de onda foi escolhido para ser colocada nos projetos da *UGV Solutions*, devido ao seu padrão de radiação omnidirecional, com isso, quando há alguma mudança em seu posicionamento, não precisa estar orientada a algum ponto para manter constantes os sinais. A antena monopolo de quarto de onda deve estar em um plano Terra, como apresentado na Figura 11, é deste que deriva a sua polarização. A monopolo deve necessariamente estar polarizada em relação ao seu plano de Terra, verticalmente (RIBEIRO, 2012).

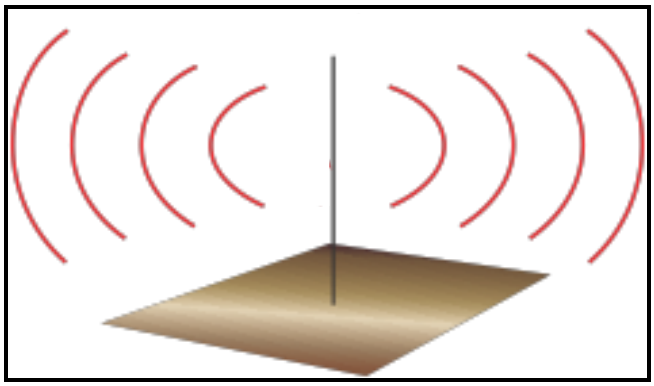

Figura 11: Ilustrativo da Antena.

No uso com os projetos da *UGV Solutions*, foi preciso desenvolver um código capaz de executar mais de um movimento, então, a cada direção será enviado um número representado por um conjunto de *bits*. Cada execução é gerada por meio de um tempo dado em milissegundos e o demodulador interpretará essa duração como sendo um certo espaço de

tempo para a execução, obtidas no osciloscópio *AGILENT 1000MHz DSO3102A*, como apresentado na Figura 12 e Figura 13, nelas são visualizados dois sinais diferentes retirados da saída da porta de dados do *arduino,* a primeira imagem tem uma direção específica e a segunda é a ausência de movimentos.

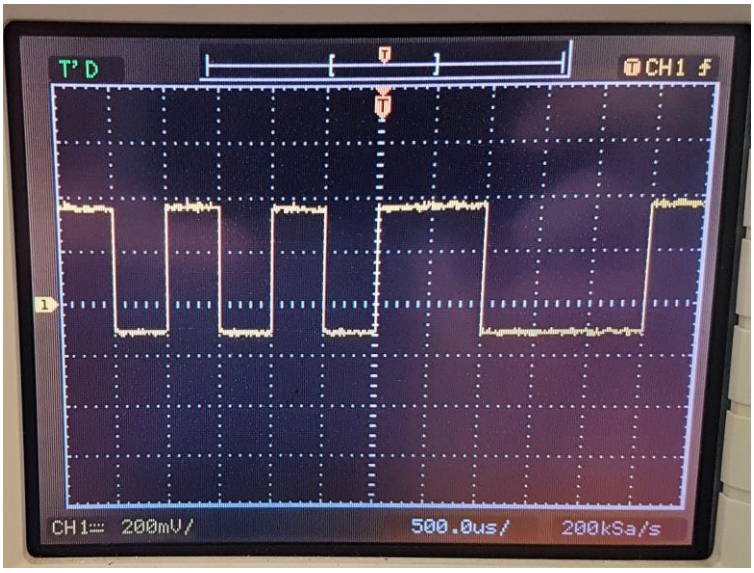

Figura 12: Osciloscópio apresentando um conjunto de *bits*.

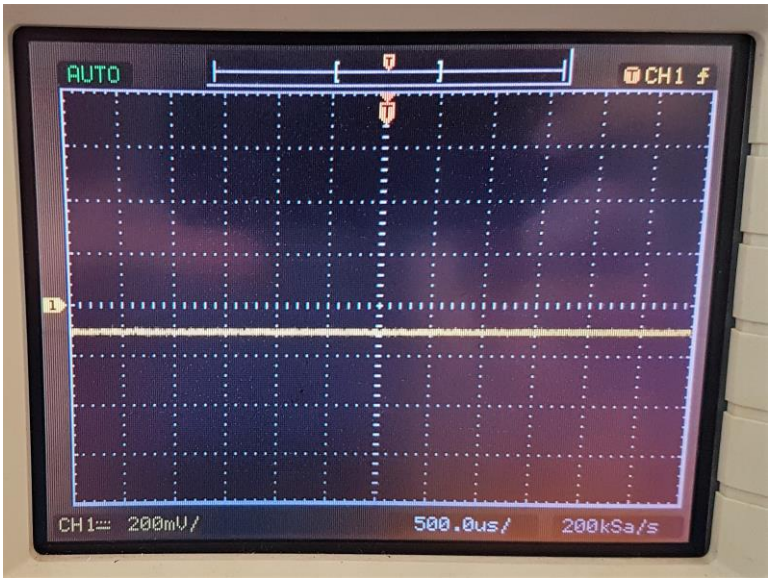

Figura 13: Osciloscópio com ausência de movimento (*bits*).

3.1. CÁLCULO DA ANTENA

Utilizando a formula: λ = C/F, que tem os significados: λ é o comprimento da onda em metros, F é a frequência da onda disposta em Hz e C é a velocidade da luz no vácuo expressa em 300.000.000 m/s (RIBEIRO, 2012).

Desse modo, aplicando a frequência de operação usada de 433MHz e também o valor da velocidade da luz (C) na fórmula, resulta em λ = 0.69m (metros). Como foi utilizado antena monopolo de quarto de onda então: L = λ/4, resultando então em: 0,1732m.

4. WEBSITE

Durante a ideação do projeto, foi pensado em como facilitar o uso de todos os recursos e passar aos usuários a melhor comodidade de acesso, com isso, foi criado um *website*

responsivo, como apresentado na Figura 14. O *website* muda a sua aparência e disposição dos elementos com base no tamanho da tela em que o site é exibido, além disso, é totalmente em inglês para uma melhor compreensão global.

Figura 14: *Website* responsivo - *UGV Solutions*.

Nas linhas de código, apresentado na Figura 15, é demonstrado como foi desenvolvido o *CSS* (*Cascading Style Sheets*) para cada forma responsiva da tela, tendo como base o tamanho que essa tela vai ter em *pixels*. Cada tela tem sua medida especificada pelo comando *width*, que significa largura em português, utilizando os padrões de mercado dos produtos, foi testado para chegar a uma melhor qualidade para o uso dos usuários, utilizando o *website*.

```
@media (min-width: 768px) and (max-width: 979px) {
  .hidden-desktop {
    display: inherit !important;
  }
  .visible-desktop {
    display: none !important ;
  }
  .visible-tablet {
    display: inherit !important;
  }
  .hidden-tablet {
    display: none !important;
  }
}
@media (max-width: 767px) {
  .hidden-desktop {
    display: inherit !important;
  }
  .visible-desktop {
    display: none !important;
  }
  .visible-phone {
    display: inherit !important;
  }
  .hidden-phone {
    display: none !important;
  }
}
```

Figura 15: *CSS* responsivo da página.

Em todo o seu desenvolvimento, foi utilizado a linguagem de marcação *HTML5* (*HyperText Markup Language*), a de estilização *CSS3* e a de programação *JavaScript*. A primeira aba é a "*Home*", nela é possível observar um *SlideShow* com os projetos da *UGV*

Solutions, um resumo geral do projeto e páginas sociais. A próxima aba é a "*About*" nela existe um texto explicando sobre o projeto e sobre todos os membros. Em seguida vem a aba "*Contact*", nela encontram-se as formas de contato com os membros e o mapa de localização da região de desenvolvimento do *UGV Solutions*.

Como apresentado na Figura 16, pode-se observar a aba "*Project*", nela contém as principais ações do projeto, que são: "*Real Time Video*", e "*Remote Data*" utilizando *PLX-DAQ*.

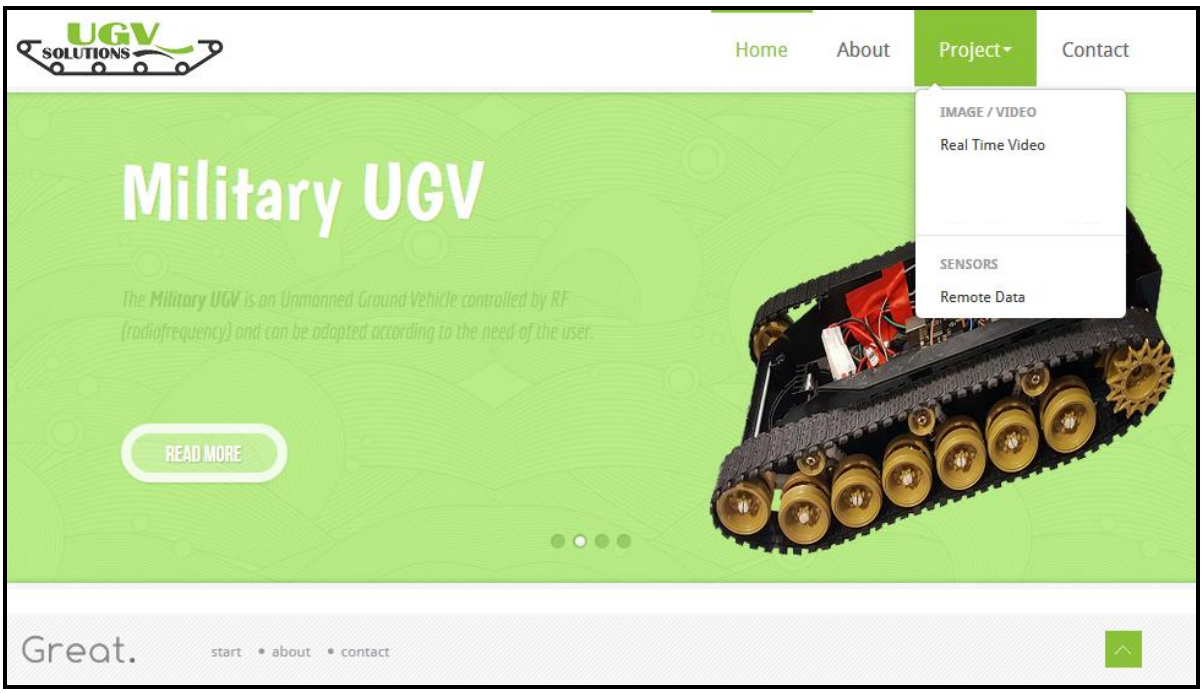

Figura 16: Página *UGV Solutions*.

4.1. REAL TIME VIDEO

A primeira ação da aba "*Project*" é a de "*Real Time Video*". Nessa aba é possível assistir em tempo real o percurso feito pelo *UGV* e quando necessário deslocar a posição da câmera em até 180 graus para observar o ambiente. O sistema visual humano possui uma notável capacidade de reconhecer padrões. Contudo, ele dificilmente é capaz de processar o enorme volume de informação presente numa imagem. Vários tipos de degradações e distorções, inerentes aos processos de aquisição, transmissão e visualização de imagens, contribuem para limitar ainda mais essa capacidade do olho humano. Levando em consideração esse fato, além de poder assistir, é possível salvar todo esse conteúdo do vídeo para uma análise mais profunda (SOLOMON, 2013).

4.2. REMOTE DATA

Ao clicar nessa opção o *website* interpreta a informação e busca um arquivo especifico no diretório do computador do usuário, esse arquivo é um *software* pré-instalado na máquina para poder ser utilizado juntamente com o *arduino* e com as informações obtidas por sensores instalados nos *UGVs*. O *arduino* é carregado com um código que realiza continuamente medidas e as envia por meio de uma saída *USB* (*Universal Serial Bus*) a um computador conectado à placa, que pode capturar essas medidas e mostrar os dados numéricos na interface de *software arduino* ("*Serial Monitor*") (DWORAKOWSKI, 2016).

Ao invés desse, utilizasse o *software PLX-DAQ*, que permite enviar os dados para uma planilha *excel*. Com os dados na planilha, é possível utilizar as facilidades deste *software* e construir, por exemplo, um gráfico com as informações recebidas dos sensores (temperatura,

luminosidade, nível de gás carbônico, humidade, entre outros), por meio do *arduino* e do *software* da *Parallax2*, como apresentado na Figura 17.

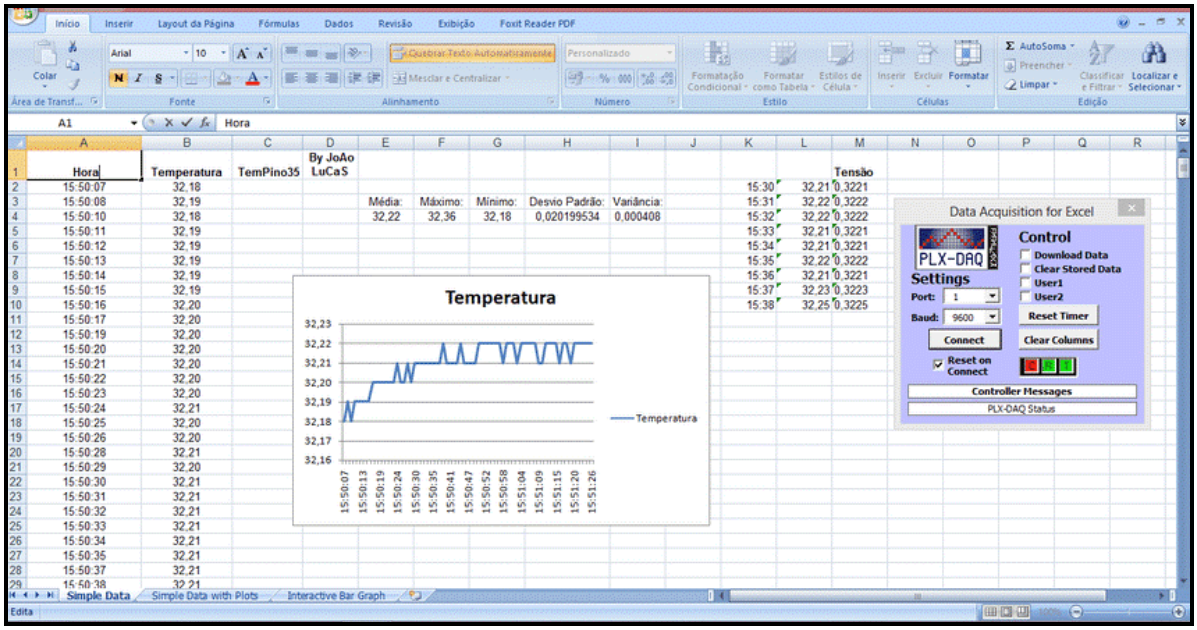

Figura 17: *PLX-DAQ (Parallax Data Acquisition tool).*

4.2.1. OUTRAS OPÇÕES

Existem outras possibilidades para armazenar esses dados e mostrar depois em uma planilha, isso pode ser feito por meio de uma saída *USB* ou envio remotamente por meio do próprio RF.

A primeira opção de armazenagem é a de utilizar a memória *EEPROM* do *arduino*, para isso, é preciso escrever um *byte* de cada vez, a leitura e escrita dessa memória requer uma biblioteca chamada "*EEPROM*" pré-instalada no *IDE* (*Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado). No código, apresentado na Figura 18, é demonstrado um exemplo de leitura e escrita desses dados (MONK, 2014).

```
//Declaração de biblioteca
#include <EEPROM.h>
//Declaração de função
void setup()
{
    //Valor a ser salvo na memória
    byte valueToSave = <ValorSensor>;
    //Valor escrito no endereço 0
    EEPROM.write(0, valueToSave);
}
//Ler a memória
EEPROM.read(0);
```

Figura 18: Exemplo de leitura e escrita na *EEPROM*.

Quando é utilizado o comando *read*, é passado o endereço da *EEPROM*, no caso, no exemplo é utilizado endereço 0. O problema dessa opção, é que a leitura e a escrita são lentas (aproximadamente 3 ms), e também, só existe a garantia de ser confiável os dados de até 100.000 escritas.

Outra memória possível para utilizar no projeto seria a *FLASH*, o *arduino* tem muito mais dessa memória do que qualquer outra, só que, existem alguns problemas, ele só aceita em torno de 10.000 escritas, e nela contém o programa que está sendo executado, com isso, qualquer dado inválido pode acarretar em mal funcionamento do mesmo. Utilizando a

biblioteca "*AVR/PGMSPACE*" foi desenvolvido o código, apresentado na Figura 19, que escreve e faz a leitura da memória (MONK, 2014).

```
//Declaração de biblioteca
#include <avr/pgmspace.h>
//Armazena qualquer estrutura de dado
PROGMEM int value[] = {<ValoresSensor>};
//Declaração de função
void setup()
{
    //Define a taxa de dados em bits por segundo (baud)
    //para a transmissão de dados em série
    Serial.begin(9600);
    //Executa até todos os dados da memória forem lidos
    for (int i = 0; i < (QuantidadeDeValores); i++)
    {
        //Ler a memória
        int x = pgm_read_word(&value[i]);
        //Printar o valor lido da memória
        Serial.println(x);
    }
}
```

Figura 19: Exemplo de leitura e escrita na *FLASH*.

O parâmetro da função "*pgm_read_word*" usa o símbolo *&* na frente do nome do *array*, porque é necessário utilizar o endereço da memória *FLASH*.

A última solução apresentada é a utilização de um cartão *SD*, embora as placas de *arduino* não tenham soquetes para esse modelo de memória, existem várias *shields* capazes de suprir essa necessidade, como por exemplo, a *Shield Ethernet*. Para utilizar o *SD* é preciso adicionar ao código do *arduino* a biblioteca "*SD*", que já vem pré-instalada no *IDE*. Para realizar a escrita e a leitura no cartão é preciso seguir o exemplo, apresentado na Figura 20 (MONK, 2014).

```
//Declaração de biblioteca
#include <SD.h>
//Abra um novo arquivo para escrita no SD
File dataFile = SD.open("datalog.txt", FILE_WRITE);
//Se o arquivo estiver disponível, escreve no cartão
if (dataFile)
{
    //Printa a informação no arquivo
    dataFile.println(dataString);
    //Fecha o arquivo
    dataFile.close();
    //Printa a informação do arquivo
    Serial.println(dataString);
}
```

Figura 20: Exemplo de leitura e escrita no cartão *SD*.

Essas maneiras mostradas de gravações em memória com os dados dos sensores são possíveis caso desejar envia-los depois para uma planilha *excel*.

5. DIAGRAMA DE GESTÃO E INTEGRAÇÃO DO SISTEMA NA SOCIEDADE

Para que as cidades junto com os órgãos públicos possam transformar uma cidade tradicional em uma cidade inteligente, e seja capaz de gerenciar todos os recursos em prol de seus habitantes, o diagrama da Figura 21 foi desenvolvido e é composto de 4 etapas sequenciais.

A primeira etapa, referenciada com (1), é a base onde serão avaliados os recursos disponíveis e necessários para a etapa (2). Na etapa subsequente, a sociedade é ouvida e o *UGV* de melhor aplicação deve ser utilizado para atender a necessidade do público alvo da região. A etapa (3), etapa de integração, garante que os erros serão mitigados se o fluxo de integração for seguido, permitindo que um plano de ação seja estabelecido. E por último, etapa (4), é a implantação do projeto, fase onde o projeto é construído e integrado a rede TICs (tecnologia da informação e comunicação), tendo a sociedade integrada a rede pública de

serviços através da internet e através do *website* acompanhar todos os dados em tempo real do que está sendo capturado e monitorado pelos veículos.

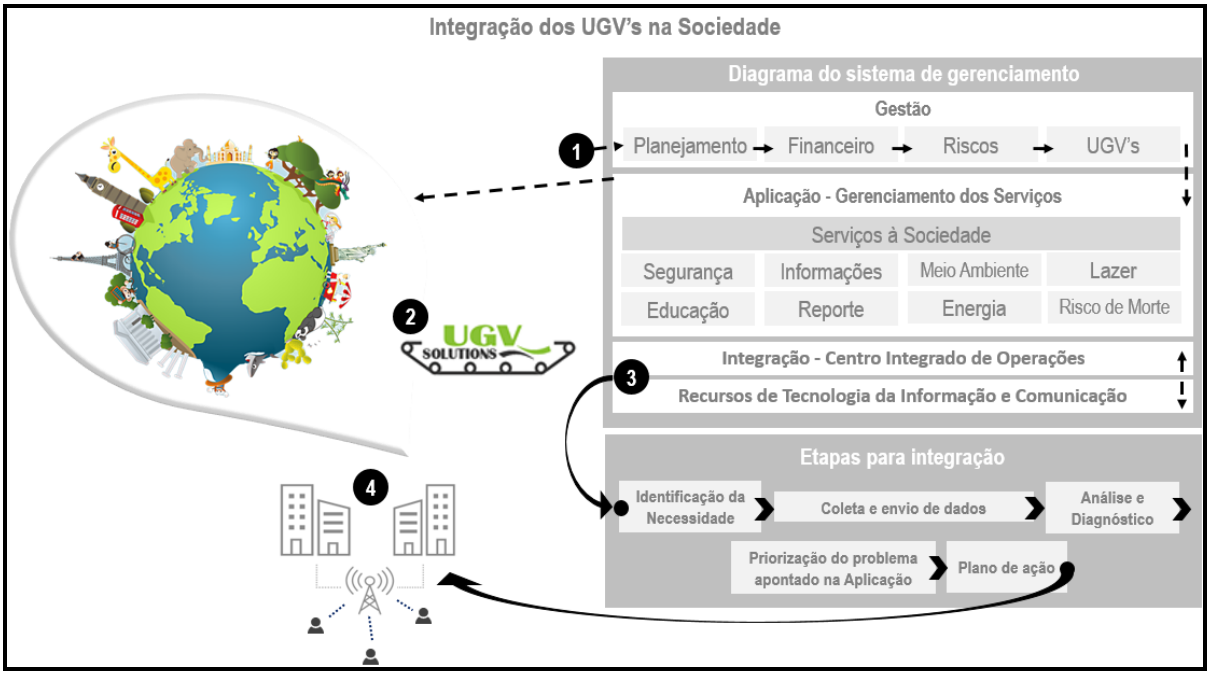

Figura 21: Diagrama de Gestão e Integração.

6. CONCLUSÕES

Durante o desenvolvimento destes veículos foram encontradas algumas dificuldades técnicas, tais como, criação de um *hardware* capaz de economizar energia durante todos os comandos e ações realizadas por ele, com isso, foram levantados todos os consumos das *shields* e como seria construído as ligações de *VCC* e *GND* para não direcionar carga para somente um lado do circuito.

Existe a possibilidade da criação de outros projetos, envolvendo maior qualidade no desenvolvimento de protótipos e robustez para qualquer tipo de terreno, para isso, estudos voltados para a área militar, doméstica, hospitalar e empresarial estão sendo levantados.

A aplicação em cidades inteligentes proporcionou um avanço para os veículos de modo que torne um ambiente mais sustentável, capaz de existir uma gerencia das cidades para atender com ética toda a sociedade. A utilização dos projetos pode ser aplicada em pontos estratégicos que necessitam ser monitorados. Trazer maior segurança para a sociedade sem colocar a vida das mesmas em risco.

Poder gerenciar toda a aplicação remotamente e mostrar em tempo real o que está ocorrendo, traz a sociedade maior satisfação em residir na cidade.

7. REFERÊNCIAS

PAULI, EVANDRO ARMINI DE; ULIANA, FERNANDO SAULO. Senai / Cst (Companhia Siderúrgica de Tubarão). Mecânica: Noções Básicas de Elementos de Máquinas. 1996. Disponível em: <http://www.abraman.org.br/arquivos/72/72.pdf>. Acesso em: fevereiro de 2017.

SANTOS, TIAGO ARGENTINO MATOS. ROVIM - Robô de Vigilância de Instalações Militares - Comunicações e Posto de Controle. 2011. Disponível em: <https://fenix.tecnico.ulisboa.pt/downloadFile/395143514515/dissertação.pdf>. Acesso em: fevereiro de 2017.

LOPES, EDIMILSON DA SILVA; LIMA, ISIS MARCELLY SOUZA; GONÇALVES, TAYNÁ CARDOSO. A Importância de Detecção de Gases para Prevenção de Danos à Segurança, Meio Ambiente e

Saúde. Revista de divulgação do Projeto Universidade Petrobras e IF Fluminense, v. 2, n. 1, p. 301-304, 2012. Disponível em: <http:// essentiaeditora.iff.edu.br/index.php/BolsistaDeValor/article/viewFile/2431/1319>. Acesso em: março de 2017.

RIBEIRO, JOSÉ ANTÔNIO JUSTINO. Engenharia de Antenas: Fundamentos, Projetos e Aplicações. Érica, 2012. 584 p.

SOLOMON, CHRIS. Fundamentos de Processamento Digital de Imagens: Uma Abordagem Prática com Exemplos em MATLAB. LTC, 2013. 281 p.

DWORAKOWSKI, LUIZ ANTONIO; HARTMANN, ÂNGELA MARIA; KAKUNO, EDSON MASSAYUKI; DORNELES, PEDRO FERNANDO TEIXEIRA. Uso da plataforma *arduino* e do *software PLX-DAQ* para construção de gráficos de movimento em tempo real. Revista Brasileira - Ensino Física. vol.38, no.3, São Paulo, 2016. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1806-11172016000300603>. Acesso em: março de 2017.

MONK, SIMON. Programação com *arduino* II: Passos Avançados com Sketches. Bookman, 2014. 247 p.