# Comunicação Serial para um Simulador de Imersão de Carrinho de Rolimã, Comparando JSON, XML e Texto

**Tiago Guerino de Oliveira Bassani[1], Kleber de Oliveira Andrade[1]**

[1]Faculdade de Tecnologia de Americana Ministro Ralph Biasi

Americana - SP - Brasil

***Abstract.*** *In order for the roller car immersion simulator to interact with its controls, the use of serial communication is indispensable. This work aims to address the need to establish a connection through existing message formats. For this it was used an Arduino in communication with a telemetry system developed in Java in order to understand which format is more agile and that has no noticeable delays to users of the simulator.*

***Resumo.*** *Para que o simulador de imersão do carrinho de rolimã possa interagir com seus controles, o uso da comunicação serial é indispensável. Esse trabalho visa atender a necessidade de estabelecer uma conexão através de formatos de mensagens existentes. Para isso foi utilizado de um Arduino em comunicação com um sistema de telemetria desenvolvido em Java a fim de entender qual formato é mais ágil e que não tenha atrasos perceptíveis aos usuários do simulador.*

## 1. Introdução

Um carrinho de rolimã ou carrinho de rolamentos é um carrinho construído, geralmente de madeira e rolamentos de aço para suas rodas, utilizados para se correr em ladeiras abaixo. Seu nome é devido ao fato de suas rodas serem rolamentos [Frederico *et. al.* 2016]. A figura 1 demonstra um exemplo de carrinho de rolimã.

O presente trabalho trata-se de uma pesquisa aplicada. Os resultados obtidos e analisados podem ser empregados em outros simuladores e jogos com controles mais aprimorado. O objetivo do trabalho é analisar diferentes formatos de mensagem enviados através de uma comunicação serial, com fio e sem fio, para controles de direção e freio em um simulador imersivo de carrinho de rolimã.

**Figura 1. Exemplo de carrinho de rolimã**

De acordo com [Banks *et al.* 2009] simulação é uma imitação de uma operação realizada no mundo real, sendo utilizada junto ao computador ou não. Simulações

envolvem a gerações de histórias artificiais e observar como essa história interfere na operação do sistema real.

Para que o simulador de imersão de carrinho de rolimã possa ser controlado de maneira que não ocorra atrasos perceptíveis, uma comunicação serial será estabelecida para otimização do tempo de transferência. O formato de mensagem escolhido para que a comunicação ocorra é de suma relevância para que o tempo de resposta entre os dispositivos seja ágil, integro e uniforme. A comunicação serial é empregada em diferentes meios de comunicação entre dispositivos distintos, por exemplo, o envio de uma foto em celular para um computador ou ainda leitores de códigos de barra em um supermercado.

A importância de um sistema que interprete os dados da comunicação serial é muito alta pois sem essa comunicação o simulador não funcionaria de maneira correta e apresentaria atrasos perceptíveis aos usuários.

## 2. Revisão Bibliográfica

Nesta seção será abordado o conceito de comunicação serial, *bluetooth*, os formatos de mensagem utilizados para as realizações dos testes e as métricas utilizadas.

## 2.1. Comunicação Serial

A comunicação entre dois dispositivos computacionais é possível através de uma interface serial. A comunicação serial é a transferência de dados feita *bit* a *bit*. [Rabello 2009]

Rabello [2009] também diz que cada *byte* enviado ou recebido, na comunicação serial, é feito um *bit* por vez, e que esses podem assumir valores lógicos ligado (1) ou desligado (0). A velocidade com que esses *bits* trafegam é medida em *bits*-por-segundo (bps) ou em em *baud rate*, que correspondem a quantidade de bits que são enviados ou recebidos em um segundo. Nos computadores, a velocidade de tráfego de *bits* pode alcançar 19,2kbps ou mais.

### 2.1.1. Bluetooth

A tecnologia *Bluetooth* foi criada pela L.M.Ericsson em 1994. O nome *Bluetooth* vem do rei da Dinamarca, Harald Blaatand, que traduzido para o inglês seria *Harald Bluetooth,* que unificou os reinos da Dinamarca e Noruega. Em 1998 as maiores empresas de tecnologia e telefones fundaram a *Bluetooth Special Interest Group* (SIG), que hoje conta com mais de 1900 empresas [Zeadally *et al*. 2019].

O *Bluetooth* é um meio de comunicação sem fio (inglês *Wireless*) que opera à curtas distancias na faixa de 2,400GHz a 2,485 GHz, com canais de rádio frequência espaçados a cada 1MHz [Zeadally *et al*. 2019].

Duas topologias de conexão são estabelecidas por Bluetooth, *piconet* e *scatternet*. A primeira consiste em um dispositivo mestre (*master*) e um ou mais dispositivos escravos (*slaves*), com a frequência de operação da *piconet* definida pelo dispositivo *master*. A segunda topologia é uma junção de duas ou mais *piconets* ocupando o mesmo espaço, simultaneamente. Em uma *scatternet* um dispositivo pode operar como *slave* em várias *piconets*, porém apenas em uma como *master* [Lee *et al.* 2007].

### 2.2. JSON

*JSON* (*JavaScript Object Notation*) é um tipo de armazenamento e transmissão de dados no formato de texto, independente de linguagens. É baseado no formato da estrutura de um objeto *Javascript* como descrito [Zunke e D'Souza 2014].

De acordo com o ECMA [2017] pode ser composto por apenas nome e valor ou uma coleção de valores, como um array, lista ou vetor. O nome deve ser uma cadeia de caracteres (*string*) e os valores podem ser booleanos (*true*, *false*), numéricos, nulos (*null*), *strings* ou *arrays*.

A estrutura do *JSON* é composta por um caractere de chave de abertura ("{") e um para fechamento ("}"), cada atributo segue-se o uso de dois pontos (":") e o valor referente a esse atributo. No caso de mais atributos, deve se empregar a virgula (",") para separá-los, conforme apresentado na figura 2. O ECMA [2017] esclarece que, no uso de *arrays* aplica-se o colchete ("[") de abertura e fechamento ("]"), seus valores são separados por vírgula (",").

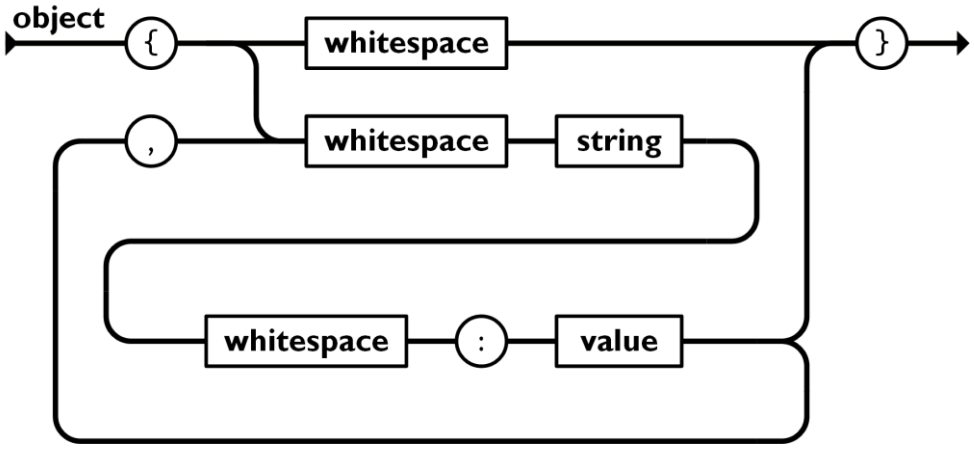


**Figura 2. Estrutura JSON**

## 2.3. XML

*Extensible Markup Language* (*XML*) é um conjunto de regas publicado pelo *Wide Wolrd Web Consortium* (W3C), grupo que regulamenta os padrões utilizados na *internet*. XML foi desenvolvido para facilitar a execução e comunicação com SGML e HTML [Zunke e D'Souza 2014].

O W3C define que documento *XML* contenha um ou mais elementos separados por marcações *(tag)* de início, fim e *tag* vazia. As *tags* de início são compostas por sinal de menor ("<") seguido do nome da *tag*, pode ter ou não atributos, e o sinal de maior (">") indicando o fim da *tag*. As *tags* de fim devem conter sinal de menor seguido de uma barra ("/") esta indica fechamento, repetir o nome da *tag* e o sinal de maior. Já as *tags* vazias são exatamente como as de início, porém antes do sinal de maior, se tem uma barra de fechamento, conforme mostra a figura 3 [Consortium 2008].

## 2.4. Métricas

Para uma análise melhor dos resultados foi feito o uso de média aritmética simples e desvio padrão.

```xml
<?xml version="1.0"?>
<pais cidade="Rio de Janeiro" capital="Brasília">
   Brasil
</pais>
```

**Figura 3. Estrutura XML**

Média aritmética simples é a média mais simples de ser calculada. Conforme a equação 1 a média aritmética simples $\bar{x}$ é dada por uma lista com $n$ elementos, sendo $n > 1$, basta somar seus elementos $x$ e dividir pela quantidade $n$ [Pereira 2014].

$$\bar{x} = \frac{1}{n}\sum_{i-1}^{n} x_i$$

**Equação 1. Média aritmética simples**

Segundo Lunet [2006] desvio padrão representa a variação entre as médias calculadas. A equação 2 representa o cálculo do desvio padrão. O resultado $Dp$ é obtido por meio de uma raiz quadrada da soma de uma lista com $n$ valores, tendo o valor de cada elemento $x$ subtraído da média aritmética $\bar{x}$ elevado ao quadrado e dividido por $n$

$$Dp = \sqrt{\frac{\sum_{i-1}^{n}(x_i - \bar{x})^2}{n}}$$

**Equação 2. Desvio padrão**

## 3. Materiais e métodos

O conteúdo desta seção trata de listar os materiais elétricos, ferramentas utilizadas na codificação e métodos para elaboração do sistema de telemetria

### 3.1. Eletrônica

Durante a elaboração dos controles do simulador utilizou-se os seguintes componentes eletrônicos: um Arduino Uno para a leitura dos valores; três potenciômetros B20K que geram os valores; um cabo USB (*Universal Serial Bus*) tipo AB para a compilação do programa do Arduino e comunicação; e um módulo *bluetooth* HC-05 para envio de dados do *hardware* ao *software*, conforme mostrado no esquema elétrico na figura 4.

Arduino é uma plataforma *open-source* para desenvolvimento e que facilita a comunicação entre *hardware* e *softwares*. Tem uma capacidade de comunicação com vários tipos de componentes como: LEDs, sensores, motores entre outros tipos. A interação entre eles é feita através da linguagem de programação própria que se assemelha as linguagens C/C++ [McRoberts 2011].

Potenciômetro é um componente eletrônico cuja função é variar sua resistência elétrica através de um eixo giratório ou deslizante. O movimento do eixo faz com que um contato deslize sobre uma superfície resistiva como grafite. Potenciômetros são muito utilizados na eletrônica como em controles de volume, sensibilidade entre outros [Braga 2012].

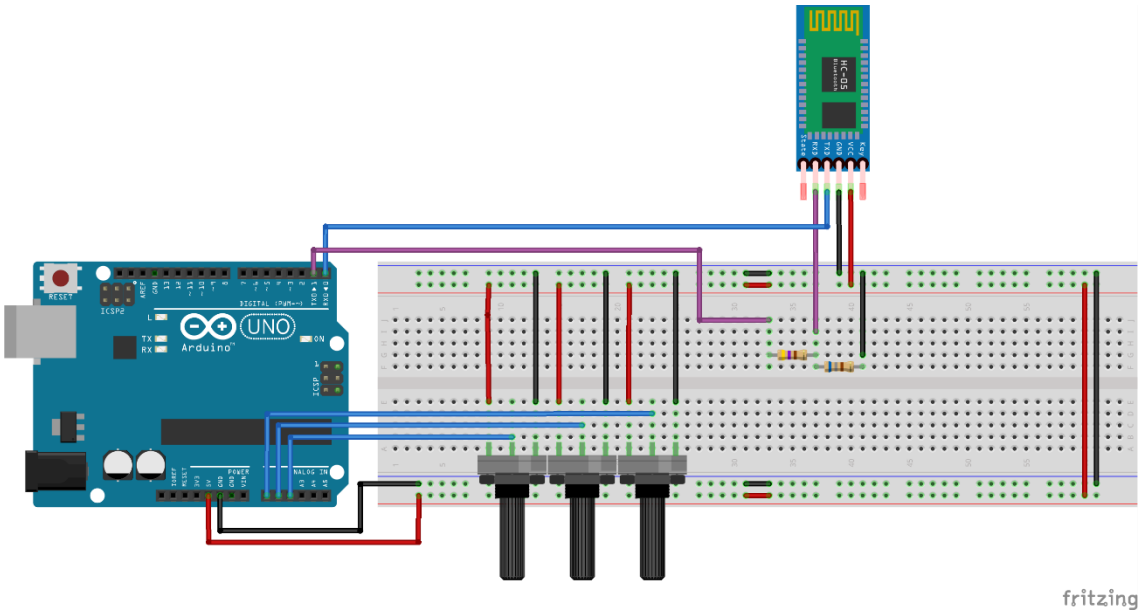


**Figura 4. Esquema elétrico**

### 3.2. Ferramentas

Elaborou-se a programação do Arduino para a leitura dos potenciômetros e envio de mensagens. Esta programação realizou-se no ambiente de desenvolvimento integrado (IDE – *Integrated Development Environment*) Arduino IDE 1.8.10, que está disponível para download no site arduino.cc. Nessa etapa foi utilizado a biblioteca ArduinoJson em sua versão 6.11.0, disponível através no Arduino IDE para *download*, empregada na serialização de *JSON*.

Para que haja a leitura dos valores transportados pela porta serial houve a necessidade de desenvolver um sistema de telemetria em Java. A IDE utilizada para o desenvolvimento foi o *NetBeans* juntamente com o repositório *Maven*, uma plataforma que disponibiliza bibliotecas à serem aplicadas em projetos Java. Todas as bibliotecas utilizadas neste projeto estão disponíveis no *Maven*, exceto *ArduinoJson*, que está disponível na Arduino IDE.

Bibliotecas adicionadas ao projeto são: *JSerialComm*, faz com que os valores indicados no Arduino possam ser interpretados pelo sistema; *JFreeChart* é utilizada para a criação do gráfico telemétrico dos valores encontrados nos potenciômetros; *JSON-java* realiza serialização e desserialização do *JSON* enviado através da porta serial ao sistema Java pelo Arduino e vice-versa; *XStream* serve como serialização e desserialização do *XML* aplicado para comparativo de velocidade entre as mensagens.

### 3.3. Sistema Telemétrico

A codificação do sistema telemétrico ocorreu através da IDE *NetBeans* criando-se uma janela com botões de conexão, caixas de combinação para a seleção da porta serial e *baud rate*, três painéis onde serão exibidos os gráficos e cinco caixas de texto que exibe os valores de cada um dos três potenciômetros: valores mínimo, médio e máximo na cor preto; atual na cor azul; média móvel na cor vermelho. Em seguida foi feita a implementação das classes *MovingAverange* e *StaticsTracker* conforme consta no diagrama 1

Conforme o diagrama 2 foi construído a classe *RealTimeLineChart* para gerar o gráfico de linha em tempo real, a fim de facilitar a visualização da variação dos valores

obtidos pelos potenciômetros. A leitura do gráfico se dá por duas linhas nas cores azul e vermelho que representam os valores atual e média móvel respectivamente.

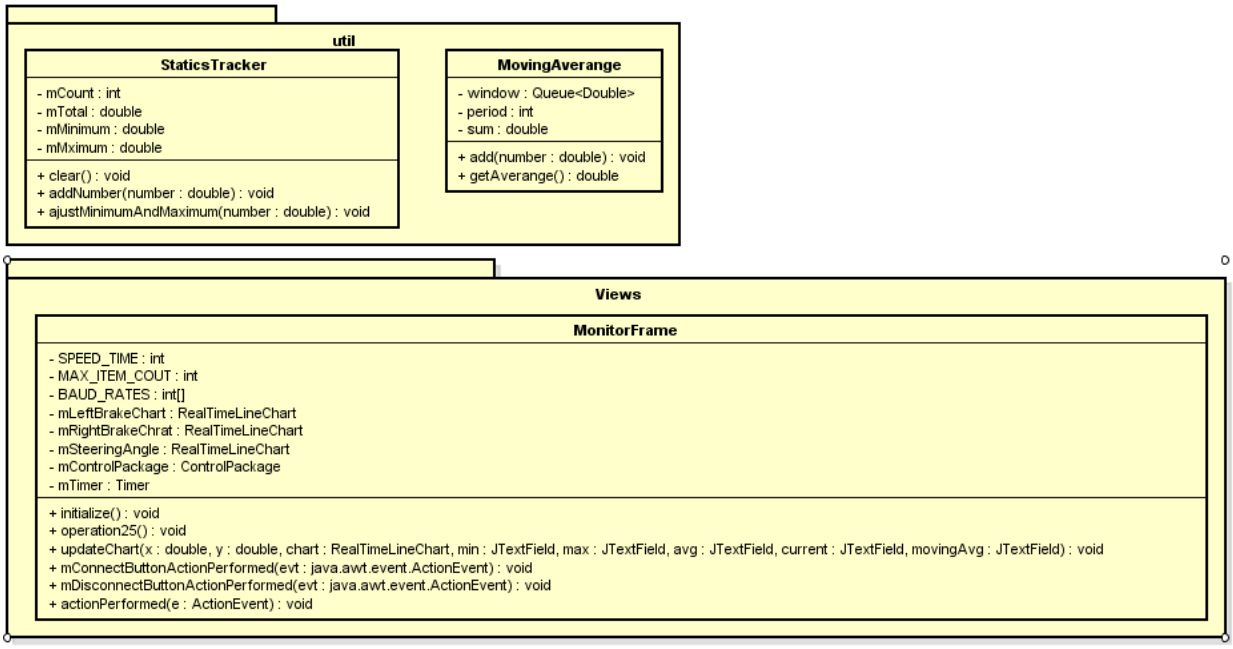


**Diagrama 1. Diagrama de Classes 1**

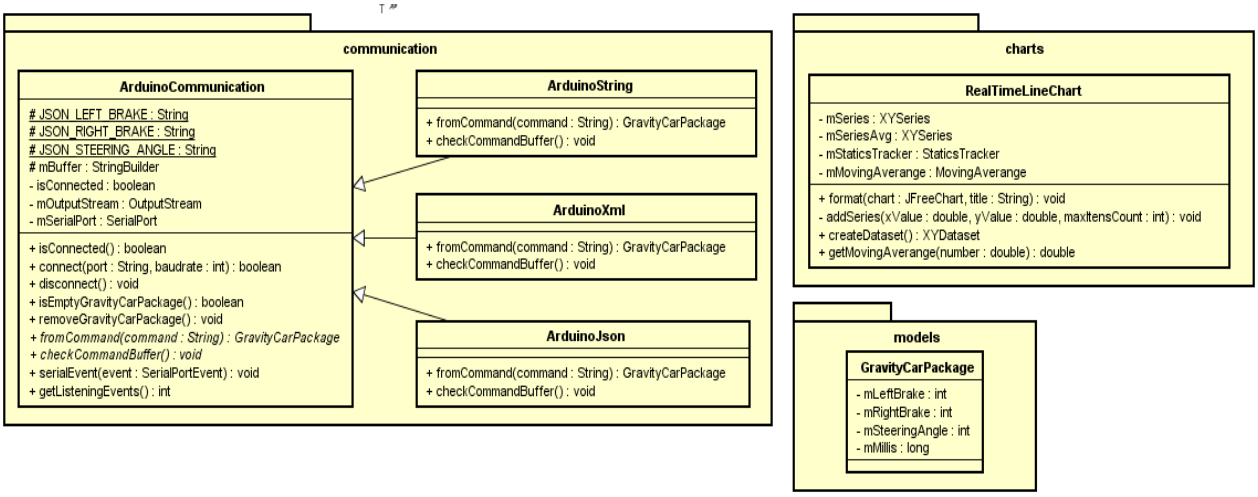


**Diagrama 2. Diagrama de classes 2**

A classe abstrata *ArduinoCommunication*, com a finalidade de transmitir dados através da porta serial. Os métodos *fromCommand* e *checkCommandBuffer* são abstratos para assim serem implementados após a escolha do formato de mensagem. Devido a isso três classes tiveram que implementar os métodos abstratos. Na *ArduinoJson* foi importado a classe *Json-java* para desserialização do *JSON* enviado pelo Arduino. Já na classe *ArduinoXml*, utilizou-se da biblioteca *XStream* para desserializar o *XML* enviado. A comunicação entre o sistema *desktop* e o Arduino se deu por meio da comunicação serial, através do cabo USB e do *Bluetooth*.

## 4. Resultados

A seguir serão mostrados as configurações e resultados obtidos pelo experimento, fazendo o uso das métricas estabelecidas.

### 4.1. Configurações

O experimento de envio de mensagens ao sistema telemétrico foi realizado em um processador *Intel Core i5-7200U*, com memória de 8GB e sistema operacional *Windows 10*.

Ao todo foram mil e vinte (1020) repetições para cada teste de mensagem enviado do Arduino ao sistema de telemetria através da porta serial por meio do *bluetooth* ou USB. A mensuração dos resultados dos tempos de resposta entre cada formato de mensagem realizada, se deu em comparação das equações matemáticas de média e desvio padrão. Assim pode-se analisar qual formato de mensagem mais adequado, pois seu tempo de resposta será o menor.

Nos testes foram utilizados três formatos de mensagem, *JSON*, *XML* e texto. Cada mensagem foi enviada com o conteúdo a palavra “teste1”, que ao ser entregue no sistema de telemetria retornava ao Arduino a mesma mensagem e fazia o cálculo de diferença entre o horário de envio e o horário de recebimento em milissegundos.

### 4.2. Métricas

Nesta etapa foi executado o teste de performance entre *JSON*, *XML* e texto. As mil e vinte mensagens tiveram seu horário de recebimento armazenado em uma lista através da função do Java *Localtime.now()* em milissegundos (ms). O tempo entre as mensagens foi calculado subtraindo o horário da mensagem recebida do horário de recebimento da mensagem anterior. Em seguida é feito o cálculo de média aritmética simples e desvio padrão.

### 4.3. Discussão

O gráfico 1 apresenta os resultados nos testes para os valores dos formatos de mensagem. Como observado na coluna azul os formatos apresentam grande diferença entre si. Pode se observar que o *JSON* apresenta o tempo de envio entre as mensagens menor que *XML* e texto, sendo aproximadamente 31,5x mais rápido que o *XML* e aproximadamente 28,3x mais veloz que o texto. Analisando o desvio padrão, podemos constatar que o *JSON* também possui maior eficácia na variação dos tempos com relação aos demais.

Observado também na coluna laranja que o *JSON* também apresenta valores muito superiores as outras opções de mensagens enviadas. É cerca de 13,1x e 12,4x mais rápida que *XML* e texto respectivamente.

A partir dos resultados obtidos pelas equações matemáticas e em comparação as colunas do gráfico, pode se analisar e optar pelo formato de mensagem e meio de comunicação usada no simulador de imersão. Os valores dos potenciômetros, lidos pelo Arduino, serão enviados através de *JSON* por meio de uma comunicação serial estabelecida por Bluetooth. A escolha da conexão levou em consideração, além da velocidade, a praticidade e conexão com outros tipos de dispositivos, por exemplo, dispositivos móveis. Assim com a definição de ambos já citados concluiu-se o sistema telemétrico conforme mostrado na figura 5.

O Sistema de telemetria apresentado na figura 5 consiste em apresentar em formas de gráficos de linhas os valores obtidos pelos potenciômetros de ângulo de esterço, freio direito e freio esquerdo de maneira independente. As linhas azuis correspondem aos

valores lidos pelo Arduino e as linhas vermelhas os valores depois de aplicado o filtro de média móvel.

## 5. Conclusão

Ao final do comparativo de velocidade apresentado conseguiu-se realizar a análise e consequente escolha do formato de mensagem mais adequado a realidade do simulador de imersão de carrinho de rolimã. O formato te mensagem escolhido para os controles de freio e direção do simulador foi formato *JSON*, que apresenta velocidade superior aos outros formatos analisados, conforme o gráfico 1. O meio de comunicação *bluetooth*, apesar não ser o mais veloz quando envia a mensagem em *JSON*, possibilitará a portabilidade do simulador em diversos dispositivos, como celular e óculos de realidade virtual

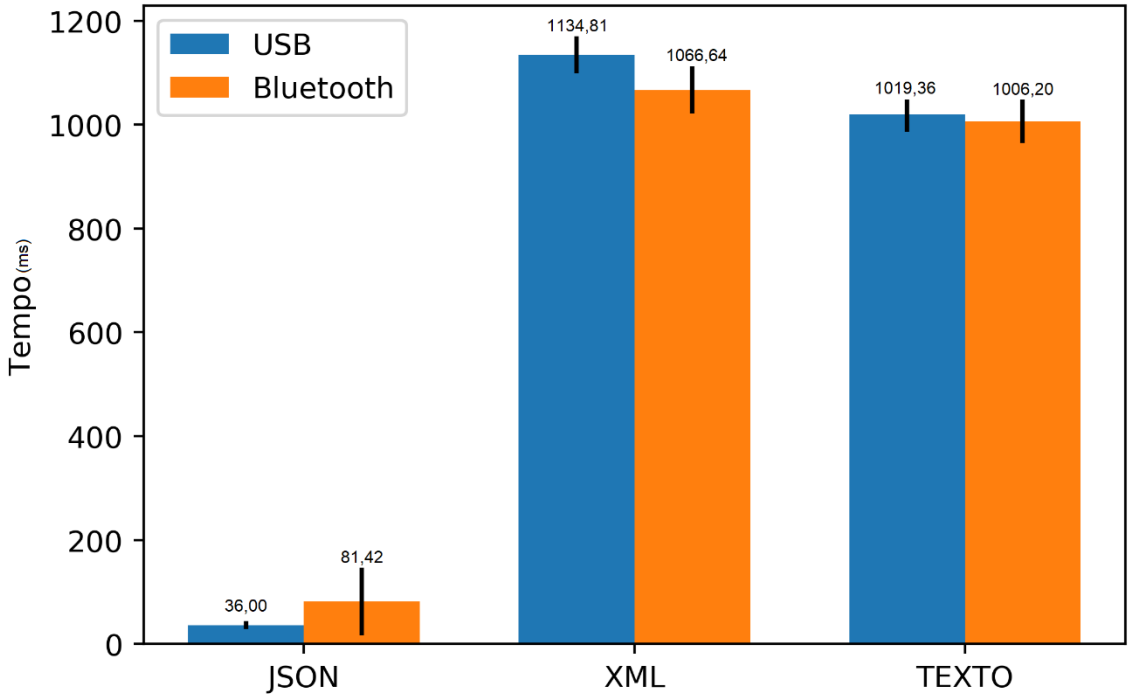


**Gráfico 1. Velocidade de transmissão**

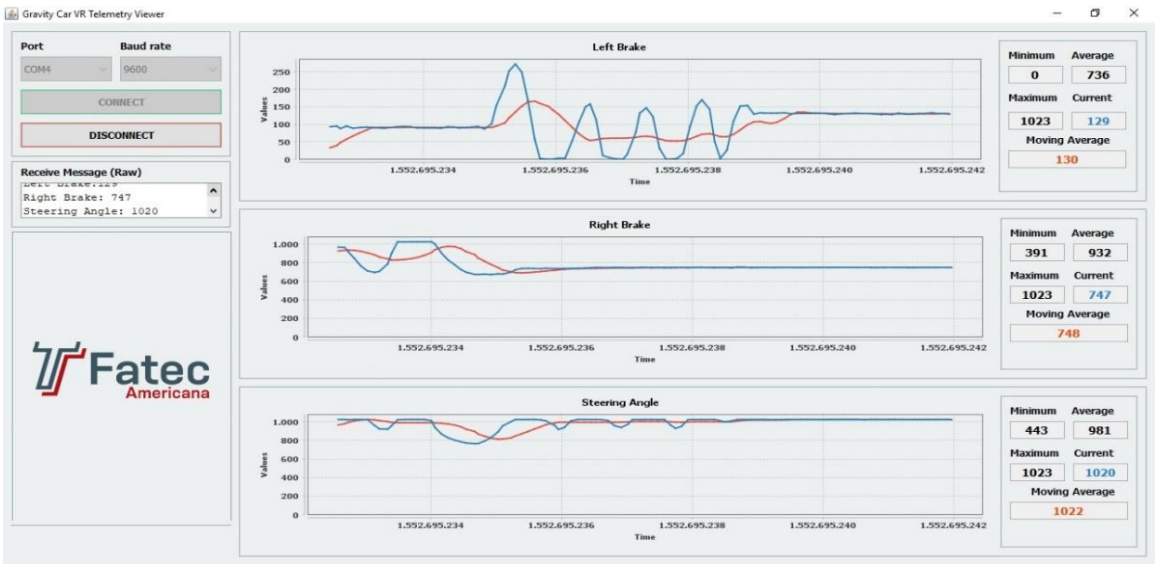


**Figura 5. Sistema de telemetria**

Para trabalhos futuros, deverá ser feito um comparativo utilizando uma rede de internet, por meio *Wi-Fi* e por cabo *Ethernet*, para assim se ter uma melhor avaliação dos dados e *struct* e *Binary JSON* (*BSON*) para formatos para o envio das mensagens. Para o sistema de telemetria controles para motores de vibração afim de proporcional melhor experiencia ao usuário do simulador.

## Referencias


Banks, J., CarsonII, J. S., and Nelson, B. L. (2009). *Discrete-Event System Simulation*. Pearson.

Braga, N. C. (2012). *Conserte Tudo: Guia Prático para Reparador Eletronico*. NBC, São Paulo - Brasil, first edition.

Consotium, W. W. W. (2008). *Extensible Markup Language (XML) 1.0*. Quinta Edição. Disponível em: <https://www.w3.org/> Acesso em: 13 nov. 2019.

ECMA (2017). *Stardard ECMA-404: The JSON Data Interchange Syntax.* Disponível em: <https://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf>. Acesso em 14 out, 2019.

Frederico, A. S., Toledo, G. P., Barros, M. V. de V., Benini, L. (2016). Processo de fabricação de um carrinho de rolimã. *Eu, a Industria e o Mundo*.

Lee, J.-S., Su, Y.-W., and Shen, C.-C. (2007). A comparative study of wireless protocols: Bluetooth,uwb,zigbee,andwi-fi. *The 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON).*

Lunet, N., Severo, M., and Barros, H. (2006). Desvio padrão ou erro padrão. *ArquiMed*.

McRoberts, M. (2011). *Arduino Básico*. Novatec.

Pereira, J. D. C. (2014). Médias: Aritmética, geométrica e harmonica. Tese de mestrado, Universidade Estadual de Campinas.

Rabello, L. M. (2009). Programa em linguagem java para comunicação serial. *Comunicado técnico 109*.

Zeadally, S., Siddiqui, F., and Baig, Z. (2019). 25 years of bluetooth technology. *Future Internet.* DOI:10.3390/fi11090194

Zunke, S. and D'Souza, V. (2014). Json vs xml: A comparative performance analysis of data exchange formats. *IJCSN International Journal of Computer Science and Network*.