Melhoria e Implementação de Comunicação Broadcast em Robôs Very Small Size

Nicholas Scharan Cysne *

Instituto Tecnológico de Aeronáutica,
São José dos Campos, São Paulo, Brasil

4 de março de 2019

*Prof Carlos César Aparecido Eguti e Prof José Roberto Colombo Júnior

Sumário

1 Plano Inicial

1º Bimestre (ago/set)

- Analisar protocolos de comunicação disponíveis para microprocessadores.

2º Bimestre (out/nov)

- Levantamento de requisitos de comunicação para robôs no padrão VSS.
- Analisar relatórios de outras equipes sobre comunicação entre robôs VSS.

3º Bimestre (dez/jan)

- Projeto de circuito de teste de comunicação entre microprocessadores.
- Realização de testes de velocidade de envio de mensagem e coleta de dados.
- Análise de viabilidade dos módulos para o projeto do VSS.
- Confecção do relatório parcial.

2 Atividades Realizadas

- Analisar protocolos de comunicação disponíveis para microprocessadores.
- Levantamento de requisitos de comunicação para robôs no padrão VSS.
- Projeto de circuito de teste de comunicação entre microprocessadores.
- Desenvolvimento e implementação de comunicação unidirecional via nRF24L01, Bluetooth, WiFi e XBee.
- Coleta de dados do tempo de envio de mensagem de cada método em relação ao payload da mensagem enviada.
- Comparação de eficiência na comunicação dos protocolos estudados e análise de viabilidade para o projeto VSS.

3 Descrição do Problema

A área de Robótica Móvel tem destacado-se como campo de estudo pela grande abrangência e complexidade, que acaba por atrair novas pesquisas. Neste contexto, competições de robótica foram criadas a fim de desenvolver e estimular novos projetos nesta área, entre elas a *Latin America Robot Competition* (LARC), competição latina americana de robótica móvel, sendo a principal competição que a categoria *Very Small Size Soccer* (VSSS) [1] participa.

A categoria VSSS tem por objetivo montar um time de três robôs diferenciais que jogue uma partida de futebol supervisionada, sem interferência humana. Nesta partida há uma câmera superior ao campo que mapeia a posição de cada robô e envia ao computador, onde é executada a lógica de estratégia e controle a ser transformada em comandos, e por fim serem enviados aos robôs por meio de comunicação sem fio.

Os comandos que são transmitidos do computador (host) aos robôs são enviados pelo método de escolha do time, juntamente com o protocolo de comunicação a ser utilizado. Dado que o processamento de imagens e lógica de controle consomem um tempo considerável até o comando ser enviado ao robô, a comunicação deve ser a mais eficiente possível, em termos de quantidade de dados enviados por tempo dispendido, para que não haja um atraso da informação grande o suficiente que atrapalhe na execução da partida.

Neste contexto, o objetivo se torna primeiramente estudar os seguintes módulos de comunicação sem fio: Wifi ESP8266 [7], Bluetooth HC-05 [6], XBee PRO Series 2 [2] e chip Nordic nRF24L01 [5]. E, posteriormente analisar a viabilidade individual de cada módulo e implementar um protocolo que, para o envio de pacote de dados padrão do VSS, apresentar a melhor performance em questão de tempo gasto.

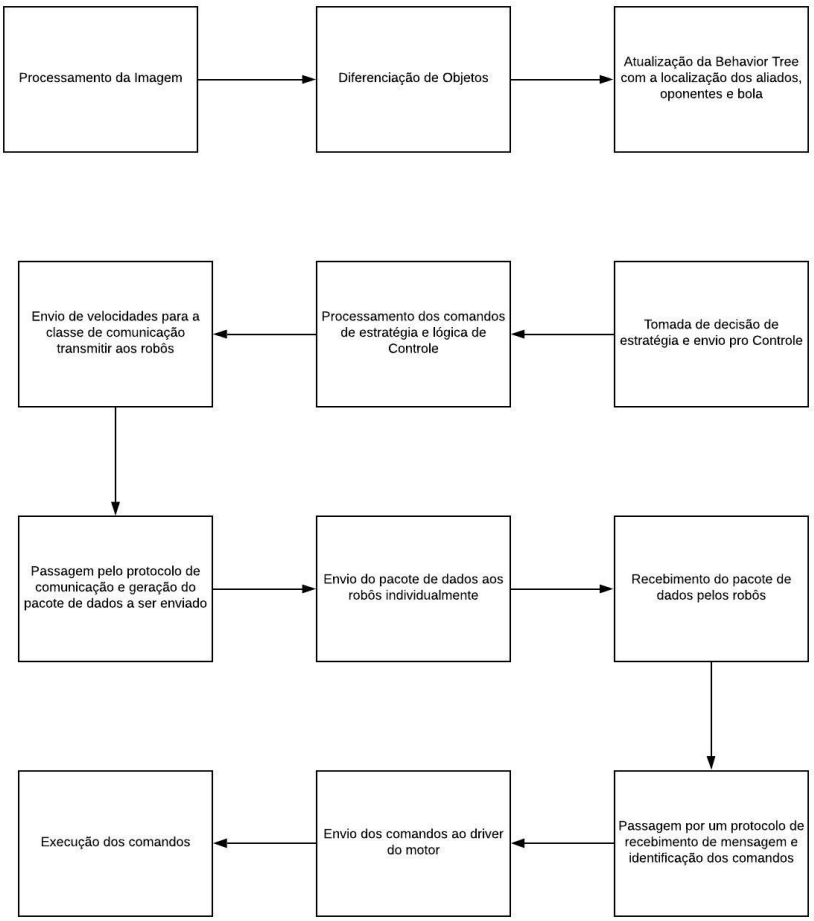


Figure 1: Fluxograma da lógica básica do processamento de comandos de um VSS.

Na Figura 1 apresenta-se um fluxograma que exemplifica a sequência lógica básica desde o processamento de imagem e geração do comando a ser eviado até este chegar no robô e ser executado.

Atualmente utiliza-se um protocolo broadcast serial de método *simplex* [3] para transmitir dados ao robô, os dados são unidirecionais, sendo outra questão a ser considerada pois é de interesse da equipe monitorar o estado dos seus robôs a todo momento. Logo, levanta-se o questionamento se um método *half-duplex* prejudicaria consideravelmente a performance da comu-

nicação e se o *trade-off* para se ter informações do robô, quanto à bateria, temperatura dos motores, etc, seria válido.

Para otimizar o protocolo, a implementação de uma comunicação *broadcast* mostra-se como uma alternativa viável, pois reduze o tempo de envio de mensagens, antes sendo enviadas individualmente aos três robôs [4]. Sendo implementado juntamente ao meio de comunicação que se apresentar mais eficiente entre os estudados. É necessário, entretanto, ter certeza que esta implementação não conflitará com o método *half-duplex* também em estudo.

Um protocolo que viabilize os dois, comunicação *broadcast* e método *half-duplex*, seria o objetivo final de estudo, estabelecendo uma comunicação rápida, de mão dupla e eficiente para a categoria.

4 Resultados

A eletrônica de um robô VSS utiliza o microprocessador da famiília do STM32, e para este tipo de microprocessador é comum se utilizar protocolos de comunicação serial UART. Toda a comunicação do projeto, portanto, se baseia neste protocolo.

4.1 Projeto de circuito de teste

A fim de testar a velocidade de cada módulo de comunicação foi necessário projetar um circuito de teste, cuja lógica de execução está representada num fluxograma, podendo ser visualizado na Figura 2. Para cada módulo um circuito diferente foi projetado devido suas limitações e necesidades individuais dos componentes, entretanto todos seguiam esta mesma lógica: medição do tempo que o circuito demorava desde o envio da mensagem, passando pelo recebimento e interpretação de comandos.

O circuito segue cinco passos básicos, sendo primeiramente o início da medição de tempo, seguido do envio da mensagem por um módulo TX A, recebimento da mesma por um RX B, interpretação da mensagem e, por fim, término da medição. Sabendo que dependendo do payload do pacote de dados o tempo dispendido varia para cada teste, foi determinado que cinco payloads diferentes seriam enviados, de 1 *byte*, 5 *bytes*, 10 *bytes*, 15 *bytes* e 20 *bytes*.

Essa faixa de valores foi escolhida pensando na comunicação broadcast implementada. O payload a ser enviado na forma broadcast necessita 22 *bytes*, posteriormente explicado detalhadamente. Assim, a faixa de valores contemplando até 20 *bytes* nos dá uma boa verificação de como cada meio de comunicação responderá quando integrado ao código do projeto.

A fim de uma melhor verificação da eficiência de cada método, para

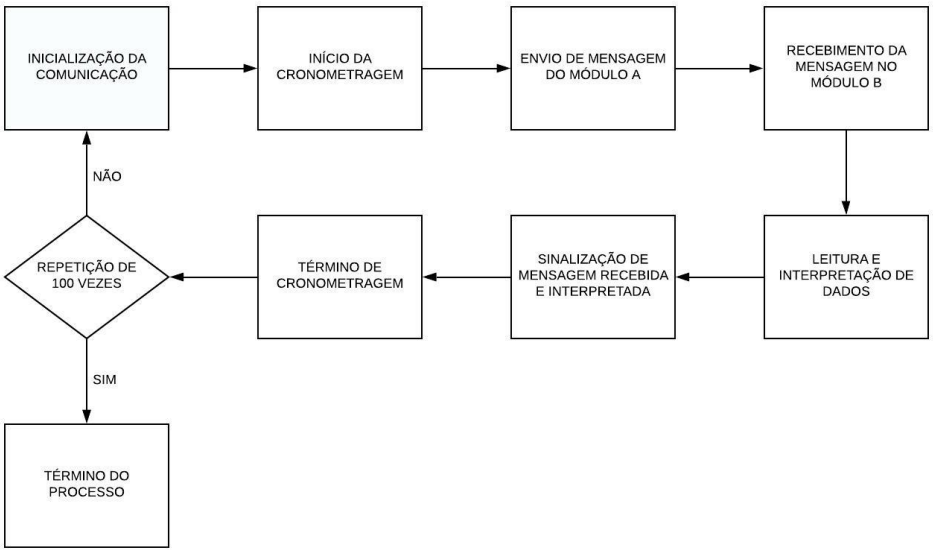


Figure 2: Fluxograma de execução do circuito para teste de velocidade.

cada tamanho de payload foram enviado cem mensagens, assim um total de quinhentas mensagens foram enviadas entre os módulos. Este número de medições nos dá uma base de dados suficiente para a análise.

4.2 Desenvolvimento e implementação de comunicação unidirecional

A seguir, descreve-se o comportamento de cada módulo durante os testes e os respectivos resultados.

4.2.1 Chip nRF24L01

O primeiro módulo a ser testado foi o chip nRF24L01, rádio compatível tanto com o microprocessador STM32 que utiliza-se na ITAndroids, quanto em um Arduino UNO, plataforma de prototipagem mais básica, que por sua

simplicidade foi escolhido para realizar os testes.

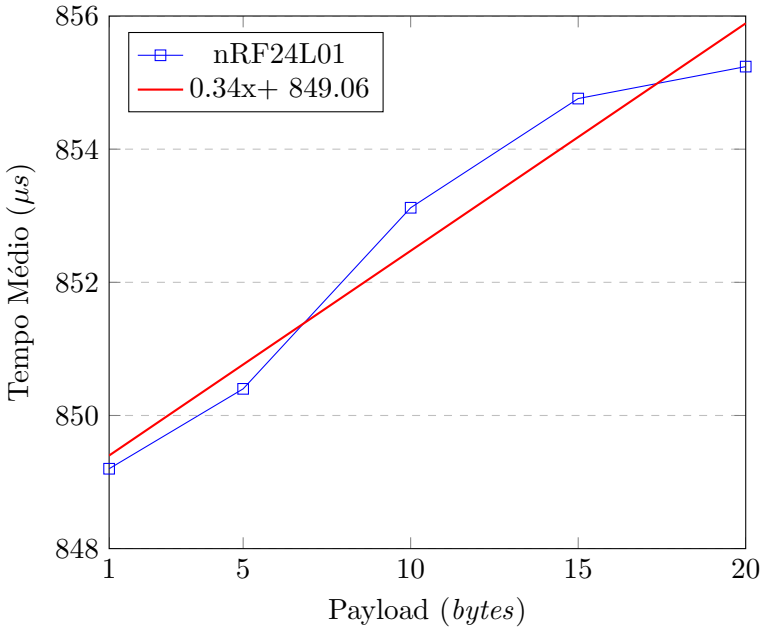
Foram transmitidas cinco séries de cem mensagens de um Arduino UNO a outro semelhante, os dois conectados a rádios nRF24L01. A Tabela 1 retrata os dados de tempo médio gasto para o envio de cada pacotes de dados e o desvio padrão médio da medição.

Payload	1 *byte*	5 *bytes*	10 *bytes*	15 *bytes*	20 *bytes*
Tempo Médio (μs)	849.20	850.40	853.12	854.76	855.24
Desvio Padrão (μs)	11.002	11.526	11.859	11.989	11.840

Tabela 1: Resultados de medição de tempo para o chip nRF24L01.

Com a ajuda do Gráfico 1, construído a partir dos dados obtidos nos testes, pode-se notar uma curva crescente, que representa o tempo médio de envio por tamanho de payload, e a função *y(x)* que melhor aproxima os pontos dados, encontrada a partir de regressão linear. A equação de melhor aproximação nos dá conhecimento que o nRF24L01 possui um tempo fixo de inicialização e envio da mensagem localizado perto de 849.06μs, e varia o tempo médio de envio numa taxa de 0.34 μs/*byte*. Assim, se mostra um meio de transmissão muito estável e rápido, tendo tempo médio de envio de 855.24 μs e um desvio padrão de 11.84 μs no payload de 20 *bytes*, objetivo de maior foco neste estudo.

Gráfico 1: Tempo de envio por payload para nRF24L01



4.2.2 Módulo Bluetooth HC-05

O módulo HC-05 também é compatível com o Arduino UNO utilizado previamente, então este foi reutilizado para o teste de velocidade.

Diferente do chip nRF24L01, o módulo HC-05 necessita ser configurado previamente antes de ser utilizado para comunicação. Um módulo foi configurado como Master, e o outro como Slave para a conexão, e um *baud rate* de 115200 *bits/s* foi definido, buscando o limite máximo de transferência de dados do Bluetooth, para assim comparar sua eficiência máxima.

Os dados resultantes dos testes de tempo médio com este módulo estão representados na Tabela 2.

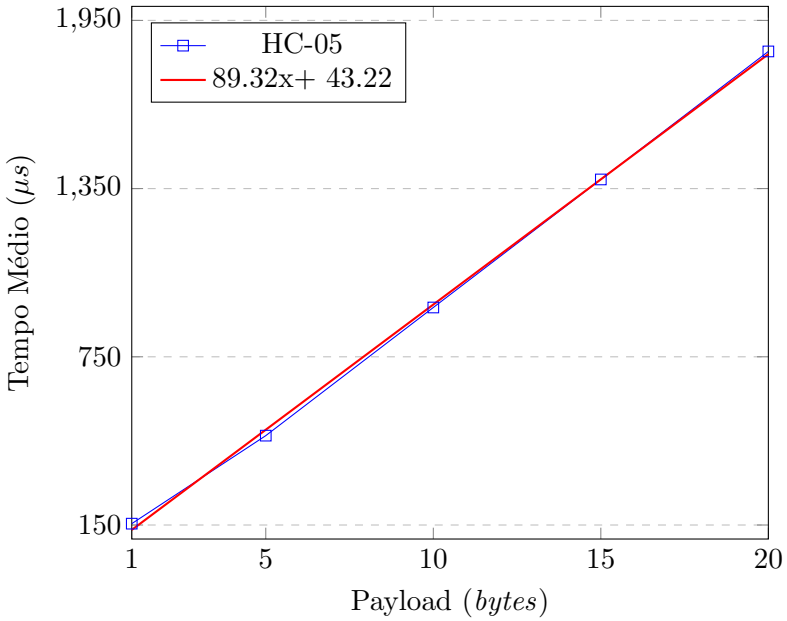
Através do Gráfico 2, construído a partir dos dados obtidos nos testes realizados, pode-se notar uma reta crescente, representando o tempo médio de envio por tamanho de payload, e a função *y(x)* que melhor aproxima os pontos dados. Nota-se uma maior linearidade para o HC-05 no envio de

Payload	1 *byte*	5 *bytes*	10 *bytes*	15 *bytes*	20 *bytes*
Tempo Médio (μs)	154.80	468.76	925.88	1382.60	1839.28
Desvio Padrão (μs)	4.566	3.627	2.633	3.284	2.864

Tabela 2: Resultados de medição de tempo para o módulo HC-05.

mensagens. Este possui um tempo fixo de inicialização e envio da mensagem localizado perto de 43.22 μs, e varia o tempo médio de envio numa taxa de 89.32 $\mu s/byte$. Esses resultados mostram numa análise preliminar que o HC-05 é mais ineficiente em comparação com o chip nRF24L01, quando o módulo envia valores de payload perto de 20 *bytes*, chegando a um tempo de envio de mensagem pertor de 1900μs, apesar de se mostrar uma fonte de transmissão muita rápida para pacotes pequenos, abaixo de 10 *bytes*.

Gráfico 2: Tempo de envio por payload para HC-05



4.2.3 Módulo WiFi ESP8266

O módulo Wifi ESP8266 pode funcionar como próprio microprocessador, sem a necessidade de utilizar uma interface como o Arduino.

Para a utilização do módulo configuramos primeiramente um destes para atuar como *server* local, aparelho a que outros tentarão se concetar, e o outro como *client*, um aparelho externo que deseja transmitir dados. Padronizamos a *baud rate* em 115200 *bits/s*, para buscar a maior velocidade possível disponível para este módulo, e por fim definimos a porta de acesso que cada um deve utilizar. O protocolo de WiFi necessita destes parâmetros por ser mais estruturado e possibilitar mais de um *client* conectado ao *server*.

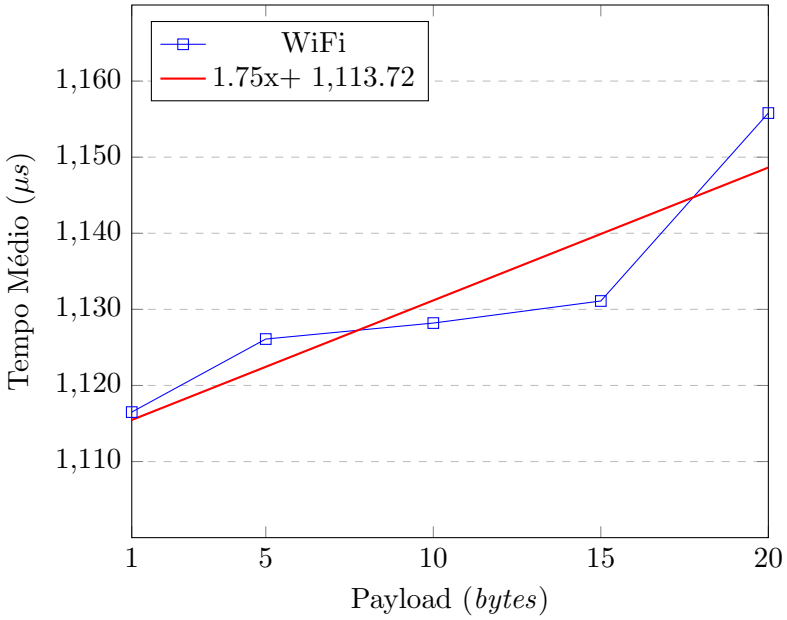
Após configurados, executamos os testes de velocidade de envio de mensagem, cujos resultados estão apresentados na Tabela 3.

Payload	1 *byte*	5 *bytes*	10 *bytes*	15 *bytes*	20 *bytes*
Tempo Médio (μs)	1116.5	1126.1	1128.2	1131.1	1155.8
Desvio Padrão (μs)	11.002	11.002	26.478	25.862	41.884

Tabela 3: Resultados de medição de tempo para o módulo ESP8266.

Assim como o nRF24L01, o ESP8266 possui um tempo de envio de mensagens estável, não variando bruscamente conforme o payload aumenta. Podemos montar o Gráfico 3 para melhor visualização dos dados. No gráfico temos a função *y(x)* que aproxima os pontos dados por regressão linear, esta nos mostra que o tempo de inicialização de mensagem e envio começa em média de 1113.72 μs e aumenta numa taxa de 1.75 μs/*byte*. Chegando aos 20 *bytes* em 1155.8 μs com desvio padrão de 41.884 μs.

Gráfico 3: Tempo de envio por payload para ESP8266



4.2.4 XBee PRO Series 2

O XBee pode ser utilizado com ou sem uma interface como o Arduino, entretanto a medição de tempo só pode ser feita quando este está presente por sua capacidade de programação interna.

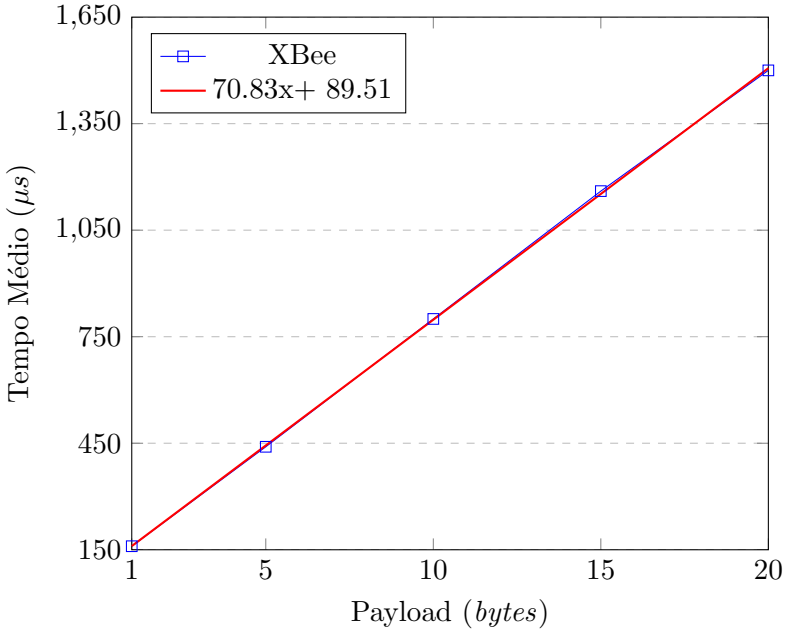
A configuração do XBee deve ser feita através do *software* XCTU, determinando um *baud rate* de 9600 *bit/s*, após configurado programamos em um Arduino sua lógica de execução. As séries de testes foram realizadas e os dados obtidos estão mostrados na Tabela 4.

Payload	1 *byte*	5 *bytes*	10 *bytes*	15 *bytes*	20 *bytes*
Tempo Médio (μs)	160	440	800	1160	1500
Desvio Padrão (μs)	18.593	20.593	19.711	28.014	28.721

Tabela 4: Resultados de medição de tempo para o XBee PRO S2.

O tempo médio de inicialização do XBee é relativamente baixo em relação ao nRF24L01 e ao ESP8266, mas sua taxa de crescimento quanto ao aumento de payload é notavelmente maior. O Gráfico 4 apresenta melhor o comportamento do XBee, construído com os dados obtidos.

Gráfico 4: Tempo de envio por payload para XBee PRO S2



A partir da função *y(x)*, reta que melhor aproxima os pontos dados, podemos estimar que o XBee leva um tempo de 89.51 μs para sua inicialização e aumento o tempo médio de envio de mensagem numa taxa de 70.83 $\mu s/byte$. Quando o payload é de 20 *bytes* este chega a um tempo de envio de 1500 μs e desvio padrão de 28.721 μs.

4.3 Análise de viabilidade para o projeto VSSS

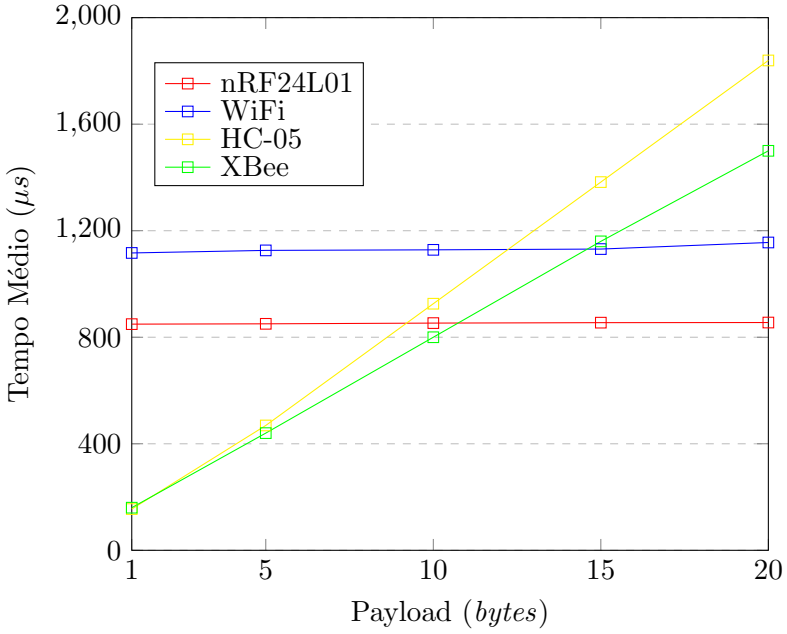
Feitos os testes e obtidos os resultados de cada módulo, podemos determinar para uma comunicação unidirecional qual módulo possui maior eficiênica para determinada taxa de payload.

A partir do Gráfico 5 podemos analisar a eficiência de cada módulo

respectivo a um determinado payload. Para pacotes de até 10 *bytes* os módulos XBee Pro S2 e o Bluetooth HC-05 se mostram melhores que o restante pois possuem uma inicialização do rádio pequena, entretando sua alta taxa de crrescimento logo os deixa ineficientes para pacotes maiores, de 10 *bytes* em diante. O ESP8266 e o nRF24L01 são mais estáveis, mantendo quase o mesmo valor independente do payload a ser enviado.

Como o objetivo é buscar aquele que seja mais eficiente quando transmitindo pacotes de 22 *bytes*, o nRF24L01 e o ESP8266 se mostram possíveis alternativas, pois uma futurra expansão no pacote não mostrará grande atraso na comunicação, entretanto o rádio ainda se mostra o mais viável para o projeto atualmente, sendo escolhido a ser implementado.

Gráfico 5: Comparação de tempo de envio dos módulos



5 Conclusão

Após a análise dos resultados obtidos em cada módulo temos que, para o projeto em questão que necessita a transmissão de 22 *bytes* por mensagem para a funcionalidade broadcast, o chip nRF24L01 possui as melhores características para se integrar ao projeto. Sua possibilidade que ouvir vários canais torna possível a implementação futura de uma comunicação *half-duplex*.

O tempo de envio de mensagens do nRF24L01, quando implementado de forma unidirecional, mostrou-se menor que os demais, determinando sua maior eficiência, que era o buscado neste estudo preliminar.

6 Plano de Trabalho e Cronograma Futuro

4º Bimestre (Fev/Mar)

- Análise da viabilidade de implementar uma comunicação bidirecional.
- Estudo de dados do robô que necessitam serem avaliados continuamente e enviados ao host.

5º Bimestre (Abr/Mai)

- Desenvolvimento de um código base para a implementação de uma comunicação bidirecional.
- Resolver possíveis conflitos da comunicação bidirecional com o protocolo broadcast.

6º Bimestre (Jun/Jul)

- Integração com o resto do código da equipe e análise final do impacto do projeto no dsempenho da equipe.
- Confecção do relatório final.

7 Agradecimentos

Agradeço ao CNPq por prover ajuda financeira para a realização desse projeto. Agradeço também à ITAndroids, por compartilhar conhecimento e ferramentas que tornam esse projeto possível. Em especial a todos os membros da categoria ITAndroids Very Small Size Soccer, que compartilham comigo a dedicação por este projeto.

Agradeço ao professor orientador Carlos César Eguti pelo apoio e pela oportunidade de realização do projeto. Também ao professor coorientador José Roberto Colombo pelo apoio para a execução da pesquisa, sempre se mostrando disposto a ajudar com ideias essenciais para a concretização do produto final.

Finalmente, agradeço aos colegas e amigos de ITA que provisionaram o suporte necessário, não somente técnico mas também pessoal, para este estudo. Em especial Téssio Perotti Arruda e Lara Campos Ibiapina, grandes amigos que trouxeram grandes ideias.

Bibliografia

[1] Latin American Robotics Competition. *IEEE Very Small Size Soccer (VSSS)*. URL: `http://www.cbrobotica.org/?page_id=81`. Acesso: 01.03.2019.

[2] DiGi. *Zigbee RF Modules*. 2014. URL: `https://www.digi.com/resources/documentation/digidocs/PDFs/90000976.pdf`. Acesso: 03.03.2019.

[3] Fred Halsall. *Data Communications, Computer Networks, and Open Systems*. Boston, Massachusetts: Addison-Wesley, 1992.

[4] Alberto Leon-Garcia and Indra Widjaja. *Communication Networks*. New York, NY, USA: McGraw-Hill, Inc., 2004.

[5] Nordic Semiconductor. *nRF24L01 Single Chip 2.4GHz Transceiver Product Specification*. 2. 2007.

[6] Itead Studio. *HC-05 - Bluetooth to Serial Port Module*. 2010.

[7] Espressif Systems. *ESP8266EX Datasheet*. 6. 2018.