UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

José Adeilmo Nunes Barbosa Junior

Trabalho de Conclusão de Curso

Implementação de uma interface de comunicação entre sistema embarcado e minicomputador com restrição crítica de tempo

Campina Grande - PB

José Adeilmo Nunes Barbosa Júnior

Implementação de uma interface de comunicação entre sistema embarcado e minicomputador com restrição crítica de tempo

Trabalho de Conclusão de Curso submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, Campus Campina Grande, como parte dos requisitos necessários para obtenção do título de Graduado em Engenharia Elétrica.

Universidade Federal de Campina Grande - UFCG

Centro de Engenharia Elétrica e Informática - CEEI

Departamento de Engenharia Elétrica - DEE

Orientador: Rafael Bezerra Correia Lima, D.Sc.

Campina Grande, Brasil

9 de dezembro de 2019

José Adeilmo Nunes Barbosa Júnior

Implementação de uma interface de comunicação entre sistema embarcado e minicomputador com restrição crítica de tempo

Trabalho de Conclusão de Curso submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, Campus Campina Grande, como parte dos requisitos necessários para obtenção do título de Graduado em Engenharia Elétrica.

Trabalho aprovado em: / /

Rafael Bezerra Correia Lima, D.Sc.
Orientador

George Acioli Júnior, D.Sc.
Convidado

Campina Grande, Brasil
9 de dezembro de 2019

Dedico este trabalho a Edna Gomes de Medeiros, amada mãe e a José Adeilmo Nunes Barbosa, amado pai

Agradecimentos

A minha mãe Edna por estar do meu lado em todos os momentos, meu porto seguro, acho que nunca conseguirei demonstrar o quando sua presença é crucial em minha vida. A meu pai Adeilmo por seu apoio durante toda universidade e me permitir conhecê-lo melhor. A meus irmãos, Ananda Sophia e Erick Lohan por serem tão especias em minha vida.

Aos meus familiares que sempre me apoiaram, principalmente meus avós maternos Everaldo e Marlene e minha avó paterna Irami, por seu carinho especial e me fazer sentir a pessoa mais sortuda do mundo. A Ellen minha querida e amada namorada por ter me suportado durante todo o tempo da universidade, mesmo nos famosos finais de período, sempre me ajudando a resolver problemas externos e internos. A Ravi, meu irmão/primo/amigo/filho, agradeço por todos conhecimentos compartilhados úteis e principalmente os inúteis. A Karol, amiga de longas datas por seus conselhos e companheirismo.

A membros da família Ferreira que me aceitaram em sua família de braços abertos em especial a dona Eunice Serafim e o senhor José Ferreira.

Aos meus amigos do grupo Armário que me adotaram e fizeram todos os dias na universidade serem toleráveis. Agradeço ao restaurante universitário (RU) por sua enorme fila onde os momentos com meus amigos foram prolongados, as noites de pizza, *just dance*, aos piquenique, sou grato a todos os momentos que passei com vocês e espero que continuemos nos encontrando.

Ao Laboratório de Instrumentação Eletrônica e Controle (LIEC) ou simplesmente casa, sou grato ao professor Péricles Barros por ter me convidado para fazer parte desta enorme família e por sempre ter acreditado no meu potencial, ao professor Rafael Bezerra, meu orientador e exemplo, ao Professor George Acióli pelos seus conselhos. Aos meus companheiros de laboratório, Breno Sant'Anna, Egydio, Lucas Porto e Paulo Roberto pelos momentos de procrastinação e diversão no laboratório. Em especial ao meu grande amigo Luquinhas que se tornou um dos amigos mais queridos, mesmo quando me expulsou de sua mesa, obrigado por tudo.

Por fim agradeço a eu mesmo, por ter seguido em frente e dado o melhor de si independente das circunstancias, sempre me dando a certeza de que fiz o melhor que pude em todos os momentos.

"It's times like these you learn to live again
It's times like these you give and give again
It's times like these you learn to love again
It's times like these time and time again."
(Foo Fighters - *Times Like These*)

Resumo

Nesse trabalho de conclusão de curso (TCC) foi desenvolvido uma interface de comunicação USB para um veículo autoguiado (AGV) desenvolvido no LIEC com um microcomputador, com a finalidade de permitir a realização do controle de trajetória do mesmo com o auxílio de ferramentas como o Simulink do MATLAB. Com a disponibilidade de maior capacidade de processamento existente no microcomputador será possível utilizar técnicas avançadas de controle na plataforma. O trabalho foi realizado respeitando as restrições temporais do sistema já existente, sendo realizado ao final uma análise temporal ao final para garantir as propriedades originais do sistema.

Palavras-chaves: Veículo autoguiado, Interface USB, Simulink; Sistemas em Tempo Real.

Abstract

In this work a USB communication interface was developed for a auto-guided vehicle (AGV) developed in LIEC with a microcomputer, with the purpose of allowing its path control with the aid of tools such as MATLAB Simulink. With the availability of higher processing capacity existing in the microcomputer will be possible to use advanced control techniques on the platform. The work was carried out respecting the temporal constraints of the existing system, being performed at the end a temporal analysis at the end to guarantee the original properties of the system.

Key-words: Self-driving vehicle, USB interface, Simulink; Real Time Systems.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AGV	Auto Guided Vehicle
LIEC	Laboratório de Instrumentação Eletrônica e Controle,
STR	Sistema em Tempo Real

Sumário

1 Introdução

O veículo auto-guiado, ou AGV (*Auto Guided Vehicle*) são disposivos comumente utilizados no ambiente industrial, sendo aplicados no gerenciamento de produtos, sejam estes matéria-prima, produtos em processo de transformação ou produtos acabados (Tamara et al., 2018). Contudo, sua utilização requer a aplicação de técnicas avançadas de controle, que necessitam de uma elevada capacidade de processamento do controlador para cumprimento das restrições de tempo real.

Buscando aproximar o aluno do ambiente industrial foi desenvolvido um veículo auto-guiado no Laboratório de Instrumentação Eletrônica e Controle (LIEC) para ser ferramenta de estudo. Porém a implementação de controladores avançados em seu microcontrolador é inviável, pois o há necessidade de uma capacidade computacional não disponível no sistema embarcado.

Além da capacidade computacional restringida, a linguagem de programação utilizada no sistema embarcado é C, o desenvolvimento de qualquer método de controle avançado demandaria mais tempo em sua implementação do que nos testes desejados. Ao utilizar ferramentas de alto nível como o MATLAB, é possível abstrair a implementação do código em C e trabalhar a nível de controle, tornando testes e validações mais simples para os alunos.

Neste trabalho tem-se o objetivo de desenvolver uma interface de comunicação USB entre um sistema embarcado de um veículo autoguiado e um microcomputador, para tornar possível a utilização da ferramentas como MATLAB. Tornando possível a utilização de controladores avançados em sua malha de trajetória e facilitar o acesso aos dados dos experimentos para realização de identificação do sistema.

Inicialmente, fez-se uma revisão das fundamentações teóricas necessárias ao desenvolvimento desta interface, consistindo na busca das formas utilizadas para realizar a integração de veículos auto-guiados e microcomputadores.

Na etapa seguinte foi implementada uma uma comunicação USB em um microcontrolador da STMicroeletronics e posteriormente foi realizada a formatação de dados para o formato JSON.

No MATLAB foi implementado um programa em Simulink para a realização da comunicação USB via *Virtual COM Port* com o sistema embarcado, validação da arquitetura simulando uma dinâmica no micro-controlador. Por fim foi realizada uma análise para verificação de comportamentos emergentes ao realizar a integração dos sistemas.

2 Sistemas em Tempo Real

Sistemas em tempo real (STR) podem ser definidos como qualquer sistema onde a resposta de um computador a eventos externos é vital, respeitando restrições de tempo.

Uma condição necessária para existência de um STR são os prazos, um sistema só pode ser considerado um sistema em tempo real caso as atividades realizadas por este possuam limite temporal para sua execução.

A definição de tarefa segundo (LIU, 2000) é um segmento de código que possui um atributo ou restrição temporal própria, tal como um período ou um *deadline*. Onde o período é o tempo entre as execuções da mesma tarefa e o *deadline* é o momento que a tarefa deve terminar.

Um sistema em tempo real possui uma ou mais tarefas que devem ser executadas em *deadlines* definidos e alguns realizam as atividades de formas periódicas. Este tipo de sistema é classificado em duas categorias:

- *Hard Real Time*: São sistemas em tempo real que o atraso na execução das tarefas ocasionam risco a alguém, por exemplo os sistemas de freio e aceleração elétrico dos carros modernos.
- *Soft Real Time*: São sistemas em tempo real que o atraso na execução das tarefas não colocam ninguém em risco, por exemplo fones de ouvido sem fio onde o atraso ocasiona apenas o desconforto ao usuário.

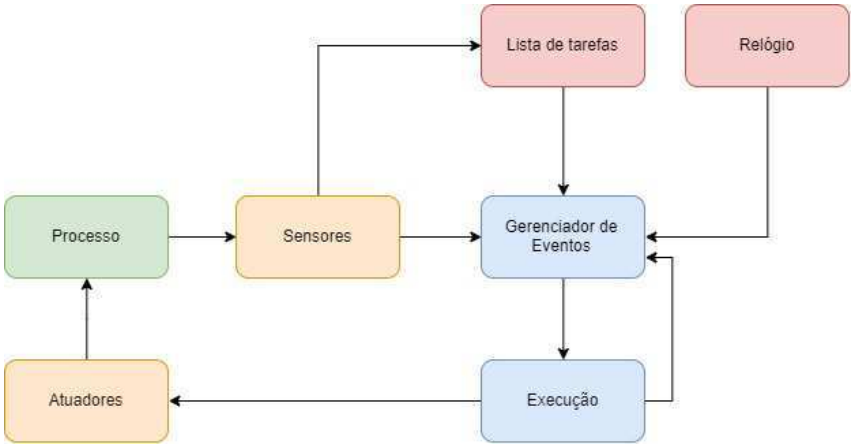
Pode-se representar um sistema em tempo real pelo diagrama mostrada na figura 1. Onde possuímos o processo real que é observado pelo sistema com o auxílio de sensores, dados os valores dos sensores existem tarefas a serem executadas, a execução destas podem alterar o sistema por meio de atuadores, nesta caso está sendo desprezado a existência de um operador(LIU, 2000).

As partes principais de um sistema em tempo real é a existência de um relógio e as listas de tarefas, sem estes não existe um STR.

Outro importante mecanismo em sistemas em tempo real são as interrupções é um meio que sistema tem para avisar ao processador que algo aconteceu, para que este realize a tarefa referente aquela interrupção, existem dois tipos de interrupções:

- Externas: Quando a interrupção é ocasionada por algo externo ao sistema, por exemplo pressionar um botão em uma interface homem-máquina.

Figura 1 – Modelo de um sistema em tempo real.



Fonte: adaptado de (LIU, 2000)

- Internas: São as interrupções que são geradas pelo próprio sistema, são muito utilizados em sistemas periódicos, onde são configurados contadores para contar o tempo e informar quando a tarefa deve ser executada novamente.

Algo importante no desenvolvimento de sistemas em tempo real é manter as propriedades existentes ao adicionar novas funcionalidades, pois como os recursos são limitados (processadores, memória, periféricos) a adição de novas tarefas podem representar o atraso de tarefas já existentes, por isso, em desenvolvimento deste tipo de sistema é necessário realizar um estudo dos impactos das novas funcionalidades.

Ao realizar a integração entre sistemas ocorre o surgimento de comportamentos emergentes (LEE; SESHIA, 2016), ou seja, comportamentos que o sistema não possuía anteriormente, mas adquiriu na composição, deve-se tomar cuidado para que o todo não perca sua característica de tempo real e as propriedades existentes anteriormente não sejam perdidas.

3 Veículos Autoguiados

3.1 Introdução

Os veículos autoguiados ou AGV (*Auto Guided Vehicle*) têm o potencial de mudar o sistema de transporte, possibilitando uma redução de consumo de energia, poluição e acidentes (MOHSENI; VORONOV; FRISK, 2018).

Para que estes objetivos sejam alcançados e o AGV consiga percorrer seu caminho é necessário instrumentar o veículo para que este seja capaz de monitorar o ambiente. Atualmente esta navegação pode ser alcançada com a utilização de: fitas magnéticas; visão computacional; navegação ótica; sonares; LiDAR; entre outros (LEE; BAGHERI; KAO, 2015).

3.2 Aplicações

As aplicações que utilizam veículos autoguiados têm como um de seus objetivos a navegação no melhor caminho em menor tempo, sem que cause qualquer acidente. Aplicações urbanas demandam grandes desafios devido à natureza dinâmica e estocástica do ambiente, uma vez que fatores externos podem interferir, de forma aleatória, na navegação. É possível encontrar modelos de veículos urbanos autoguiados, como da Uber visto na figura 2, porém ainda não são amplamente utilizados.

Figura 2 – Veiculo autoguiado da UBER.

Fonte: UBER Blog.

Ambientes industriais são tipicamente mais controlados. Existem normas e regras de segurança na definição da trajetória dos veículos autoguiados, minimizando o risco a

funcionários (Lynch et al., 2018), além de evitar colisões com outros veículos ou equipamentos (Digani et al., 2014). De forma geral, o ambiente industrial permite um maior controle sobre o ambiente. Dentre as principais aplicações no ambiente industrial destaca-se sua utilização em armazéns, como visto na figura 3, onde com a sua utilização temos uma maior segurança, eficiência e um melhor gerenciamento de estoque (Tamara et al., 2018).

Figura 3 – Exemplos de veiculos autoguiados industriais.

(a) AGV utilizado em armazéns tradicionais.

Fonte: jungheinrich.

(b) AGV para transporte de pequena carga.

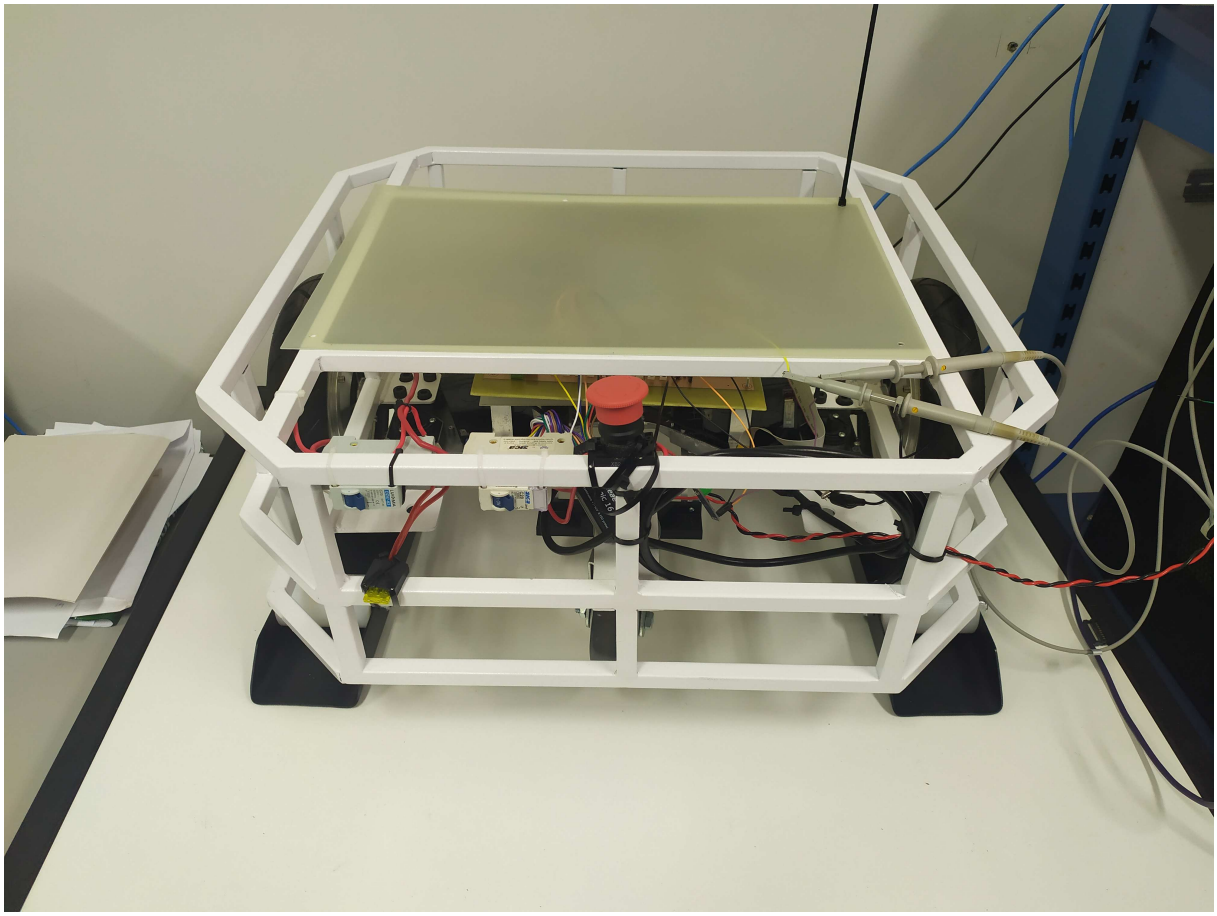
Fonte:ISA.org.

3.3 Veículo autoguiado do LIEC

Tendo em vista a importância dessas ferramentas no cenário industrial atual, foi desenvolvido um AGV no LIEC (Laboratório de Instrumentação Eletrônica e Controle), que pode ser visto na figura 4 que permitisse a simulação de um ambiente industrial real. O modelo do LIEC utiliza motores DC brushless (sem escovas), são motores que necessitam de uma menor manutenção pela ausência das escovas de grafite existentes em motores DC usuais. Outra característica destes motores é a forma de seu acionamento permitindo, por exemplo, o seu funcionamento a torque constante, porém, estes motores possuem uma forte restrição temporal em seu controle sua não execução de forma periódica pode causar problemas em seu sistema de potência. Por este motivo o sistema embarcado encarregado de seu controle possui uma janela de tempo disponível fixa para a realização de qualquer outra operação (DIAS; LIMA; BARROS, 2019).

Dentre as diversas ferramentas que permitem que o AGV consiga observar o ambiente, foram utilizadas as trilhas magnéticas para guiá-lo, pois, possuem um baixo custo de instalação e complexidade quando comparadas com os demais métodos, embora sua flexibilidade ser limitada devido aos percursos serem predefinidos são aplamente utilizadas na indústria (Lynch et al., 2018). Mesmo com sua simplicidade as trilhas magnéticas

Figura 4 – AGV desenvolvido pelo LIEC.



Fonte: autoria própria.

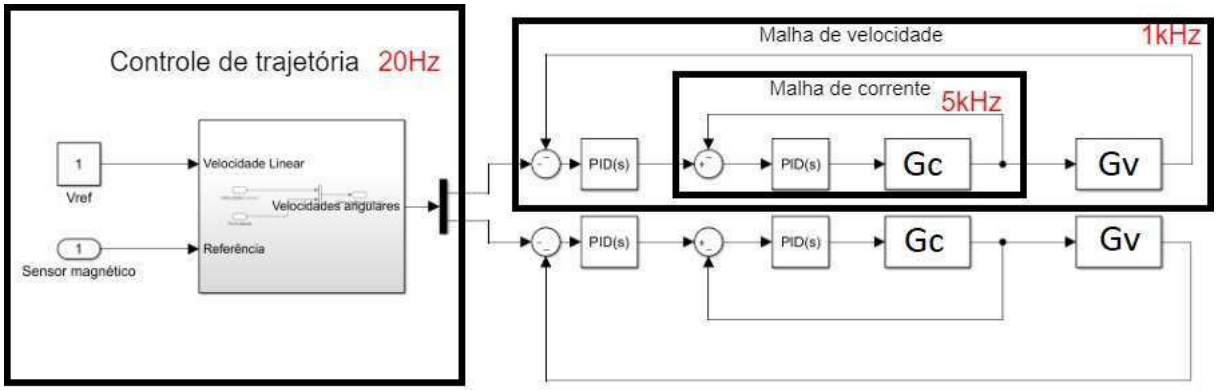
possuem características importantes para aplicações industriais. A movimentação de veículos autoguiados ocorre em uma área delimitada garantindo uma maior segurança em sua operação e o fato de que seu funcionamento não é afetado por sujeira ou mudança de iluminação torna o sistema robusto.

A falta de flexibilidade em determinadas aplicações industriais pode não ser considerada um problema dado que as ações a serem executadas são repetitivas e limitadas (LEE; YANG, 2012).

O AGV desenvolvido tem como objetivo ser plataforma de aprendizado e testes de diversos controladores e diversas estratégias de controle, para isto, que isto seja possível é necessário que haja uma interface de comunicação entre o AGV e um microcomputador, pois algoritmos mais robustos de controle necessitam de maior capacidade de processamento do que o disponível em seu microcontrolador, outra questão importante é o nível de abstração necessário para criação dos algoritmos, com a utilização de um microcomputador é possível utilizar ferramentas como MATLAB, que com o seu maior nível de abstração facilita o desenvolvimento e validação das regras de controle.

O veículo autoguiado do LIEC utiliza 2 malhas para controlar cada um de seus motores e uma malha para controle de trajetória visto na figura 5. A primeira malha é a de corrente que opera a uma taxa de $5kHz$ e sua segunda malha é a de velocidade que opera a uma taxa de $1kHz$ e para o controle de trajetória é utilizado um controlador a uma taxa de $20Hz$.

Figura 5 – Exemplo do sistema de controle utilizado no AGV.



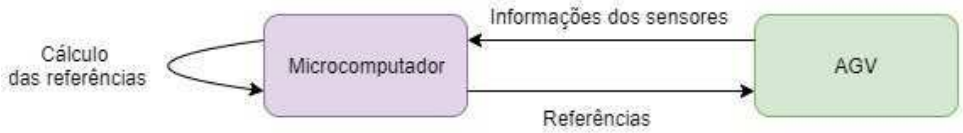
Fonte: autoria própria.

3.4 Mecanismos de comunicação

Para a realização da interface de comunicação entre o AGV e o microcomputador são utilizados alguns mecanismos sem fio, porém são voltados para o gerenciamento dos veículos autoguiados, como por exemplo: ZigBee (Kobayashi; Yamamoto; Yamazaki, 2016), wireless (Anggraeni et al., 2018) e WEB (ZHANG et al., 2017).

Para a aplicação desejada na figura 6 a comunicação com fio melhor se adéqua, pois quando levado em consideração a restrição de tempo que o AGV do LIEC onde o tempo para o transmissão dos dados dos sensores, cálculo das novas referências e escrita das novas referências no AGV para a malha de controle mais lenta do sistema não pode ultrapassar $50ms$.

Figura 6 – Fluxo das informações do sistema desejado.



Fonte: autoria própria.

Para aplicações em sistemas com restrições temporais o USB se destacou dado sua alta taxa de transmissão, possuindo diversas especificações como visto na tabela 1. Em

(KORVER, 2003) é possível observar as taxas de transmissão do USB em seus modos de operação e especificações.

Tabela 1 – Taxa de transmissão de algumas especificações USB.

Especificação	Taxa de transmissão máxima
Low Speed	1.5 Mb/s
Full Speed	12 Mb/s
High Speed	480 Mb/s

Fonte: Adaptado de Sony

O formato para transmissão dos dados escolhido foi JavaScript *Object Notation* (JSON), por tornar as informações transmitidas legíveis para o usuário. As informações são transmitidas como *string* onde temos o nome da propriedade (TAG) e o seu valor separados pelo caractere ':', mais de uma propríedade utiliza-se a vírgula para separá-las e quando se deseja transmitir um objeto ou *struct* utiliza-se o caractere '{' para início e '}'para o final. Na figura 7 tem-se um exemplo de um JSON que representa uma pessoa.

Figura 7 – Codificação das informações de uma pessoa em JSON.

```
{
  "primeiroNome": "José Adeilmo",
  "ultimoNome": "Barbosa",
  "estaVivo": true,
  "age": 27,
  "endereço": {
    "rua": "Rua Alatória",
    "cidade": "Bonito",
    "estado": "PE",
    "CEP": "55400-000"
  },
  "numeroTelefones": [
    {
      "tipo": "casa",
      "numero": "88 3333-3322"
    },
    {
      "tipo": "móvel",
      "numero": "85 98877-6655"
    }
  ]
}
```

Fonte: autoria própria.

4 STMicroeletronics

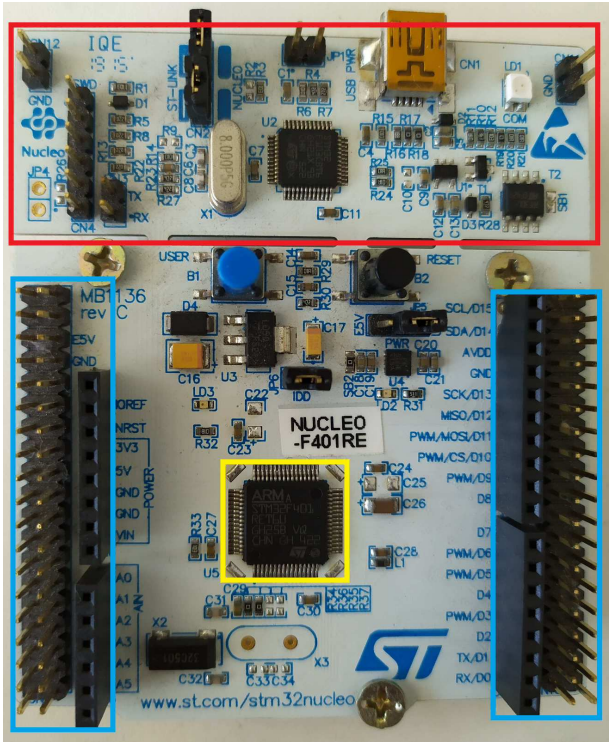
4.1 STM32

A STMicroeletronics é uma multi-nacional Francesa e Italiana fabricante de componentes semi-condutores, fundada em 1987 na união das empresas SGS Microelettronica (italiana)e e Thomson Semiconducteurs (francesa) (STMICROELETRONICS,).

Uma das famílias de microcontroladores produzidos pela STMicroeletronics é a STM32. Nela estão contidos os microcontroladores de 32 bits da fabricante, todos baseados na arquitetura ARM da Cortex-M. Os diversos modelos disponível podem ser utilizados desde aplicações que necessitem de um baixo consumo até aplicações de alta desempenho.

Além da produção dos microcontroladores a STMicroeletronics desenvolvem placas de desenvolvimento que auxiliam na prototipação dos produtos, chamadas de Nucleo, na figura 8 podemos ver a nucleo da família Cortex-M4. Destacado em vermelho é a parte da placa onde é realizada a programação do microcontrolador destacado em amarelo e permite a depuração do código em tempo de execução destacado em azul nas laterais são todos os pinos existentes no microcontrolador.

Figura 8 – Nucleo-f401RE da STMicroeletronics.

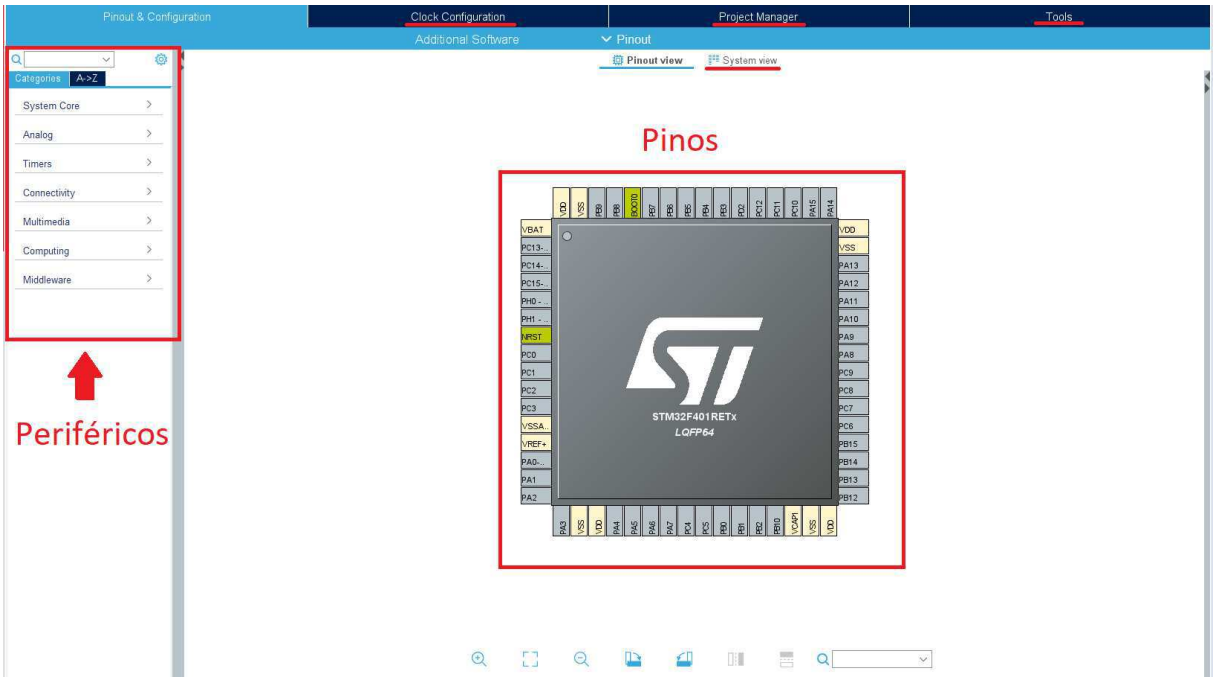


Fonte: autoria própria.

Para o desenvolvimento de soluções para os microcontroladores da STMicroeletronics é disponibilizado pela própria empresa softwares que permitem a configuração do microcontrolador e a geração automática do código de configuração deste, auxiliando o desenvolvimento evitando que o usuário necessite realizar esta configuração em código. Atualmente existem 2 softwares disponíveis o CubeMX que permite apenas a configuração e ao gerar o código é necessário um software de terceiros para dar continuidade ao desenvolvimento e em 2019 a STM lançou o STM32CubeIDE que é o seu ambiente de desenvolvimento integrado, é possível realizar a configuração e programação em um mesmo programa.

Ambos programas ao criar um projeto é necessário selecionar a placa de desenvolvimento ou microcontrolador desejado, após selecionado tem-se a tela vista na figura 9, a esquerda há o acesso aos periféricos e funcionalidades disponíveis no microcontrolador escolhido, ao centro é representado o microcontrolador escolhido e uma representação de seus pinos. Na barra superior é possível ter acesso as configurações de clock, configurações do próprio projeto e por fim uma ferramenta para simular o consumo de bateria dado uma sequência de tarefas a serem executadas.

Figura 9 – Visão geral de um projeto criado no CubeMX.



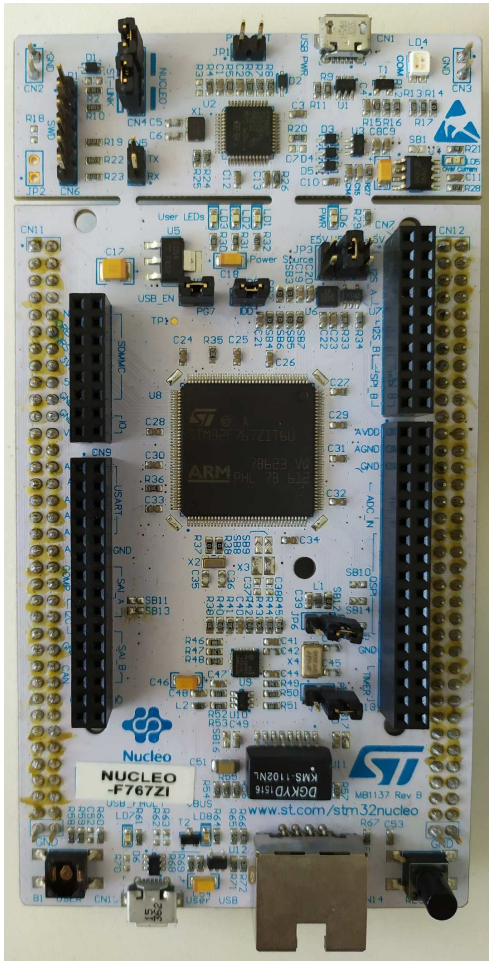
Fonte: autoria própria.

4.2 Nucleo-f767ZI

No AGV desenvolvido pelo laboratório fora escolhido a placa de desenvolvimento Nucleo-f767ZI da STM, ela faz parte da família de placas de desenvolvimento que pos-

suem 144 pinos disponíveis, possuí em sua placa conectores USB além do conector do programador para ser utilizado na aplicação e também há disponível um conector RJ45, como visto na figura 10.Seu processador é um ARM Cortex-M da família M7 que são os processadores com maior performance dentre os Cortex-M.

Figura 10 – Placa de desenvolvimento Nucleo-f767ZI.



Fonte: autoria própria.

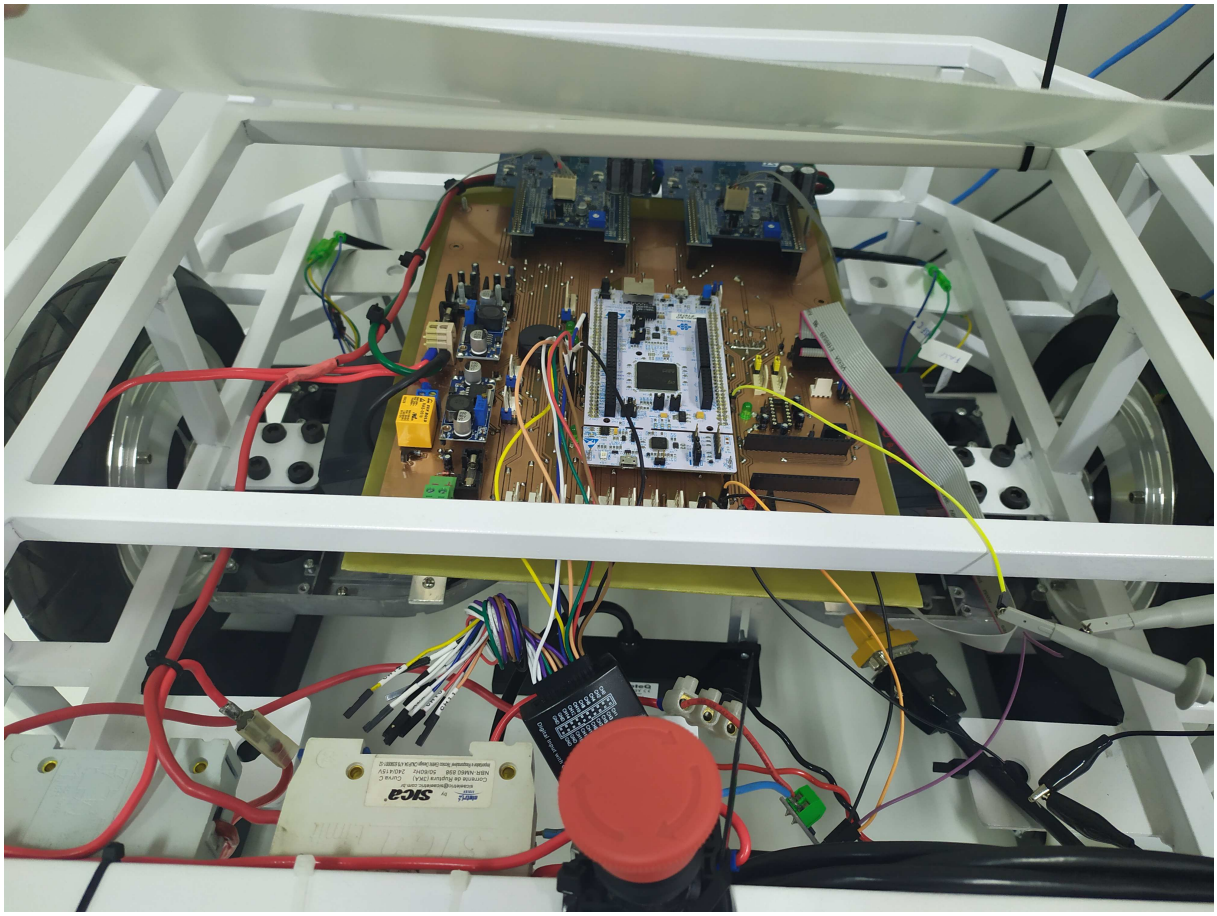
Dentre as principais especificações da placa escolhida tempo:

- ARM®32-bit Cortex®-M7;
- Frequência máxima da CPU: 216 MHz;
- 2 MB Flash;
- 512 KB SRAM;
- 114 GPIOs com capacidade de geração de interrupção externa;
- 3 ADCs de 12-bit com 24 canais;

- Suporte para até 8 barramentos USART/UART;
- Suporte para até 4 barramentos I2C;
- Suporte para até 6 barramentos SPI (6);
- Suporte para até 3 barramentos CAN 2.0B ativo;
- USB 2.0 OTG HS/FS;
- Ethernet.

Esta placa foi escolhida pela sua capacidade de processamento e disponibilidade de barramentos de comunicação, permitindo a ampliação do sistema de instrumentação sendo possível até utilização de outras placas de desenvolvimento mais simples e comunicando-se com o protocolo CAN, tornando a plataforma de aprendizagem mais fiel com as aplicações industriais (KALIAPPAN et al., 2018). Na figura 11 pode ser visto a placa de desenvolvimento fixada a placa de circuito impresso desenvolvida no laboratório.

Figura 11 – Visão da parte interna do AGV do LIEC.



Fonte: autoria própria.

5 Resultados Experimentais

5.1 Configuração da comunicação USB na placa de desenvolvimento

A primeira parte da atividade consistiu na configuração da comunicação USB na placa de desenvolvimento NUCLEO-F767ZI, neste modelo é possível a operação em duas das especificações do padrão USB, modo *Full-Speed* e no modo *High-Speed*. Tendo em vista as portas do microntrolador que foram utilizadas para as demais funções do AGV, era possível operar apenas no modo *Full-Speed*.

O dispositivo foi configurado para comunicação do tipo *virtual COM*, pois desta forma é possível sua utilização no MATLAB como uma entrada e saída serial. Quando conectado a um microcomputador que utiliza como sistema operacional o Windows, este envia um byte para o dispositivo e o dispositivo deve reenviar o mesmo byte para que o controlador USB do sistema operacional saiba qual especificação USB está sendo utilizada e qual a taxa de transmissão.

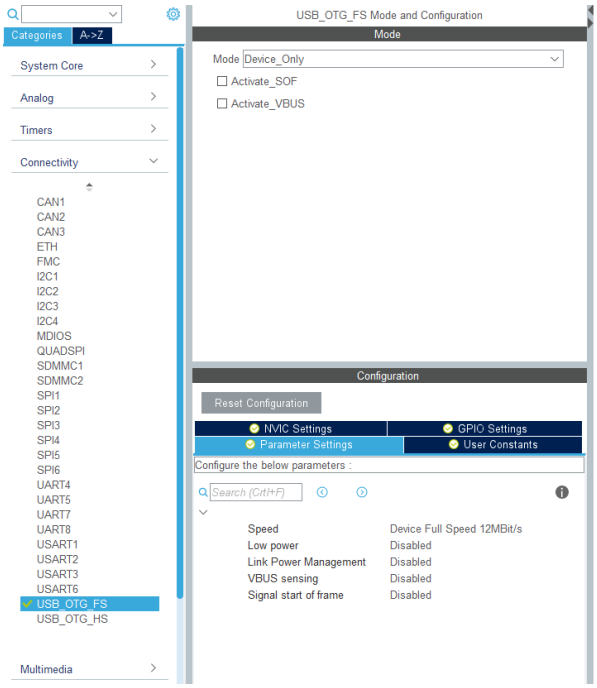
No ambiente do cubeMX é necessário realizar algumas etapas para a correta criação do projeto para a comunicação USB. Primeiramente deve-se habilitar na aba de conectividade o a especificação desejada do USB, neste caso será utilizado o **USB_OTG_FS** que representa a especificação *full-Speed* de $12Mbit/s$, como mostrado na figura 12a.

Para utilizar o USB também é necessário configurar uma camada de *middleware*, dado a complexidade da comunicação USB, não é possível utiliza-lo em o auxílio deste. Ao configura-lo existem diversos tipos de dispositivos USB, será utilizado o modo *Virtual Port Com*. A configuração utilizada pode ver observada na figura 12.

É necessário utilizar uma frequência de 48MHz para o USB como visto na figura 13, as configurações restantes de clock irão depender da aplicação.

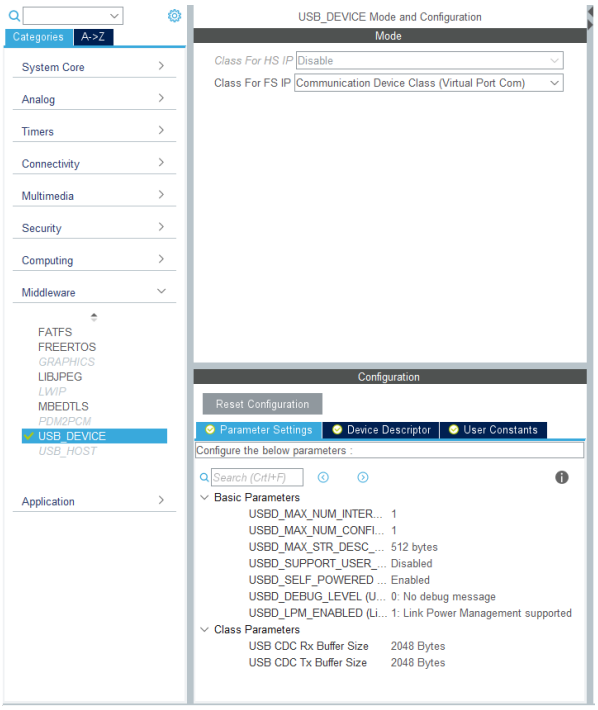
Figura 12 – Configurações para *USB Full-Speed* no STM.

(a) Configuração do **USB_OTG_FS**.



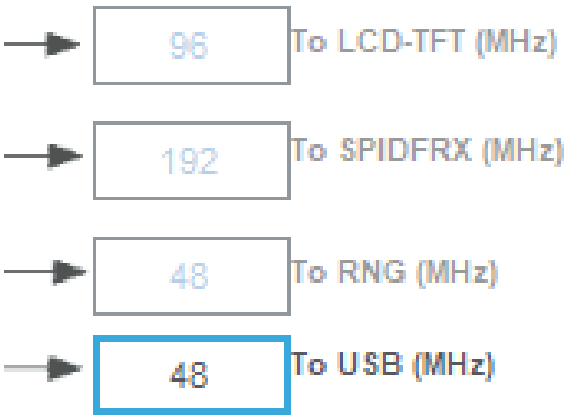
Fonte: autoria própria.

(b) Configuração do **USB_DEVICE**.



Fonte: autoria própria.

Figura 13 – Configuração do clock para o USB.



Fonte: autoria própria.

Por ultimo é necessário aumentar o tamanho mínimo da memória alocada, para alterar é necessário ir nas configurações do projeto, os valores utilizados podem ser encontrados na figura 14.

Ao gerar o código do projeto no cubeMX, a configuração inicial para reenviar o byte de configuração para o controlador USB do microcomputador deve ser realizado no arquivo **usbd_cdc_if.c** na função **CDC_Control_FS**. Primeiramente deve-se criar

Figura 14 – Configuração da memória alocada.

Fonte: autoria própria.

uma variável do tipo **USBD_CDC_LineCodingTypeDef** para armazenar a informação enviada pelo microcomputador, o nome dado a variável foi **LineCoding**. No *switch* da função é necessário adicionar o código mostrado na figura 15, inicialmente todos os *cases* estão vazios. O pbuf é um dos parâmetros de entrada do método **CDC_Control_FS**.

Figura 15 – Código responsável por reenviar o *byte* de configuração.

```
case CDC_SET_LINE_CODING:
    LineCoding.bitrate    = (uint32_t)(pbuf[0] | pbuf[1]<<8 | pbuf[2] << 16 | pbuf[3] <<24);
    LineCoding.format     = pbuf[4];
    LineCoding.paritytype = pbuf[5];
    LineCoding.datatype   = pbuf[6];
   break;

   case CDC_GET_LINE_CODING:
    pbuf[0]= (uint8_t)(LineCoding.bitrate);
    pbuf[1]= (uint8_t)(LineCoding.bitrate >> 8);
    pbuf[2]= (uint8_t)(LineCoding.bitrate >> 16);
    pbuf[3]= (uint8_t)(LineCoding.bitrate >> 24);
    pbuf[4]= LineCoding.format;
    pbuf[5]= LineCoding.paritytype;
    pbuf[6]= LineCoding.datatype;
   break;
```

Fonte: autoria própria.

As funções responsáveis pelo recebimento e transmissão das mensagens também estão localizados no arquivo **usbd_cdc_if.c**, são elas: **CDC_Receive_FS** chamada quando uma mensagem é recebida possui como parâmetros de entrada o apontador para o início da mensagem recebida e o tamanho da mensagem e **CDC_Transmit_FS** função utilizada para realizar a transmissão do dados, possui como argumentos de entrada o apontador do início da mensagem e seu comprimento.

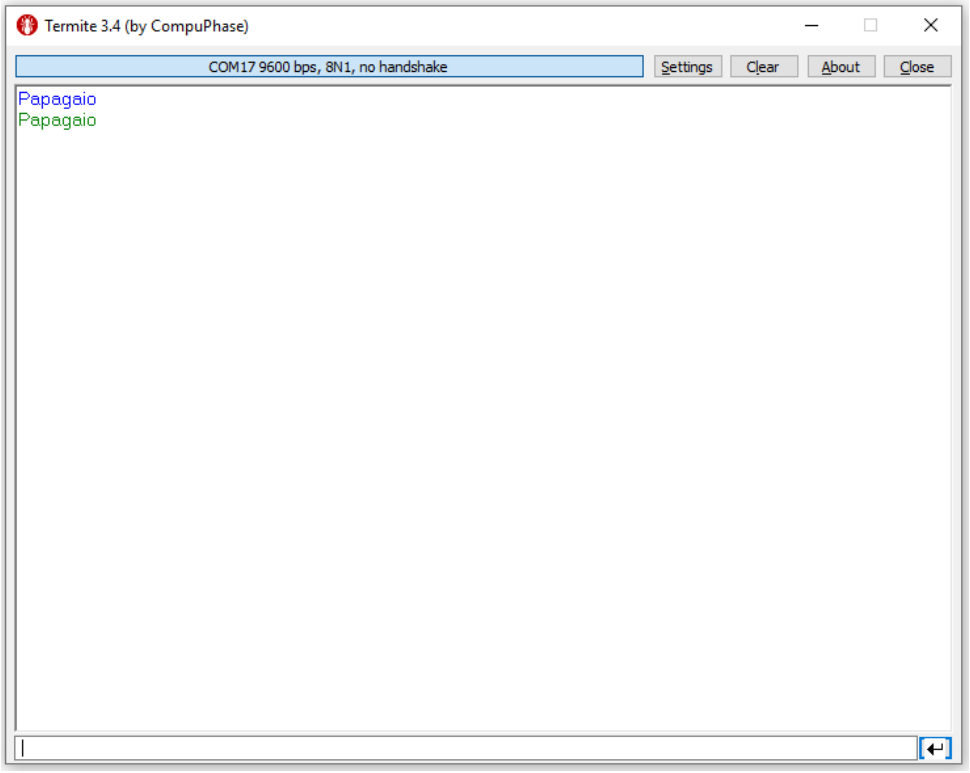
Para testar a comunicação foi criado uma aplicação para repetir a mensagem recebida, como podemos observar na figura 16, apenas a linha 307 foi adicionada, todas as demais são geradas pelo CubeMX. Com o auxílio de um terminal serial, neste teste foi utilizado o Termite 3.4 selecionou-se a porta serial associada a placa de desenvolvimento e toda mensagem enviada para esta, tem-se a sua repetição, na figura 17 é possível em azul a mensagem enviada e em verde a resposta da placa.

Figura 16 – Código para repetir o que for recebido.

```
288⊕  * @brief  Data received over USB OUT endpoint are sent over CDC interface
301⊖ static int8_t CDC_Receive_FS(uint8_t* Buf, uint32_t *Len)
302 {
303   /* USER CODE BEGIN 6 */
304   USBD_CDC_SetRxBuffer(&hUsbDeviceFS, &Buf[0]);
305   USBD_CDC_ReceivePacket(&hUsbDeviceFS);
306
307   CDC_Transmit_FS(Buf, *Len); // Única linha adicionada
308
309   return (USBD_OK);
310   /* USER CODE END 6 */
311 }
312
314⊕  * @brief  CDC_Transmit_FS
324⊖ uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len)
325 {
326   uint8_t result = USBD_OK;
327   /* USER CODE BEGIN 7 */
328   USBD_CDC_HandleTypeDef *hcdc = (USBD_CDC_HandleTypeDef*)hUsbDeviceFS.pClassData;
329   if (hcdc->TxState != 0){
330     return USBD_BUSY;
331   }
332   USBD_CDC_SetTxBuffer(&hUsbDeviceFS, Buf, Len);
333   result = USBD_CDC_TransmitPacket(&hUsbDeviceFS);
334   /* USER CODE END 7 */
335   return result;
336 }
```

Fonte: autoria própria.

Figura 17 – Mensagem enviada e recebida no terminal serial.



Fonte: autoria própria.

5.2 Implementação do codificador JSON no STM32

No AGV do LIEC existem *structs* que representam os motores e o sensor magnético, para a realização do controle de trajetória se torna necessário o envio apenas das informações existentes na *struct* do sensor. Como foi visto o JSON pode ser simplificado a um *array* de caracteres, logo para a codificação das informações do estrutura do sensor utilizou-se a função **sprintf** da biblioteca **stdio.h** que faz parte do C. Nela é possível informar qual a formatação da *string* desejada e os argumentos semelhante ao **printf**, porém a *sring* é salva em um array. Informações adicionais sobre a função pode ser encontrada em (CPLUSPLUS,).

Após a formatação da estrutura de dados do sensor magnético segundo o padrão JSON utilizou-se uma interrupção temporal a uma frequência de $20Hz$ para simular o sistema real, a cada interrupção era enviado os dados, como podemos observar na figura 18. Como a função de transmissão **CDC_Transmit_FS** transmite números inteiros de 8 bits, a informação enviada foi convertida seguindo a tabela ASCII.

Figura 18 – Código para codificar o JSON e transmitir.

```
280 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
281     if(htim->Instance == TIM10)
282     {
283         sprintf(jsonOutput, "{"
284                   "\"Magnetic\": {"
285                    " \"fork_left\": \"%.2f\","
286                    " \"fork_right\": \"%.2f\","
287                    " \"left_marker\": \"%d\","
288                    " \"right_marker\": \"%d\","
289                    " \"track_present\": \"%d\","
290                    " \"track_pwm_duty_cycle\": \"%06.2f\""
291                   "}"
292                 "}", magSensor.fork_left, magSensor.fork_right, magSensor.left_marker,
293                 magSensor.right_marker, magSensor.track_present, magSensor.track_pwm_duty_cycle);
294
295         CDC_Transmit_FS(jsonOutput, LENGTH);
296     }
297 }
```

Fonte: autoria própria.

5.3 Comunicação Serial no Simulink

Ao trabalhar com o simulink é necessário realizar uma série de normalizações relacionadas ao tamanho das mensagens recebidas e o tipo das variáveis, a falta da correta declaração das variáveis ocasiona erros ao tentar testar a aplicação.

Para utilização da comunicação serial, o simulink possui um kit de ferramentas chamado de *Instrument Control Toolbox*. Dentre os blocos disponíveis são necessários apenas 3 para realizar a comunicação serial, são eles:

- *Serial Configuration*: Bloco de configuração onde é configurado taxa de transferência, bit de paridade, entre outros;

- *Serial Receive*: Responsável por receber os dados transmitidos pelo dispositivo, nele deve-se configurar o tamanho máximo da mensagem a ser recebida, qual o bit que representa o final da mensagem e a taxa de amostragem;
- *Serial Send*: Bloco que transmite a mensagem para o dispositivo, pode ser especificado qual o carácter vai representar o término da mensagem. Para uma correta transmissão é necessário converter as informações para o tipo inteiro de 8 bits.

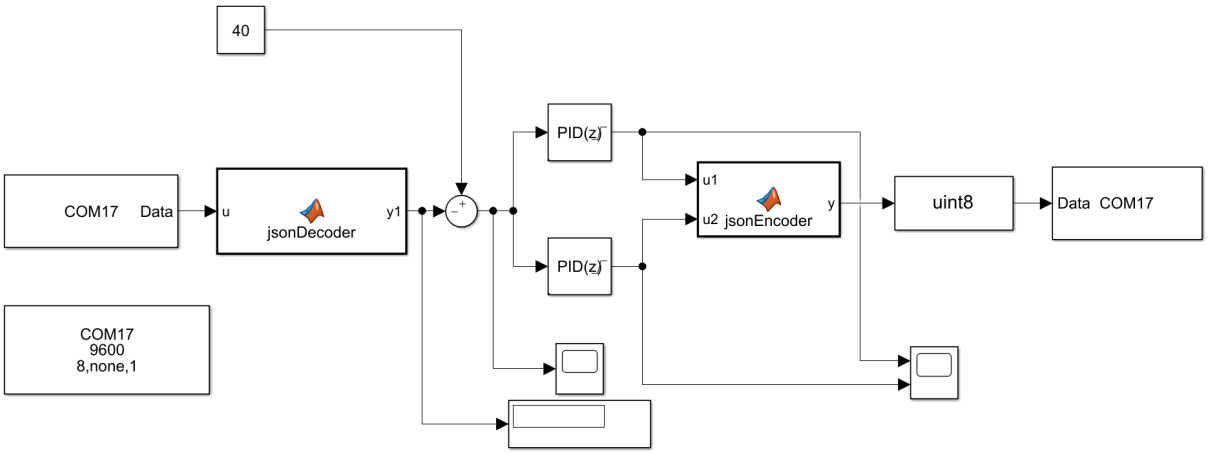
Foi estabelecido um tamanho máximo das mensagens enviadas pelo sistema embarcado e a mesma dimensão foi configurada no ambiente do Simulink.

Foi desenvolvido uma função para realizar a decodificação da mensagem JSON recebida.

Após a correta decodificação da informação recebida foi utilizado dois controladores discretos para simular uma malha de controle e a saída dos controladores representariam as novas referências para os controladores de velocidade dos motores.

Um bloco para codificar as informações e enviar para o sistema embarcado foi implementado e foi criada uma nova *struct* que possuía as informações sobre as referências de cada motor. Todo o sistema pode ser visto na figura 19.

Figura 19 – Projeto criado no Simulink para a comunicação.



Fonte: autoria própria.

5.4 Decodificador JSON STM32

Com o programa em Simulink finalizado, foi necessário implementar um decodificador JSON no sistema embarcado. Como é conhecido as informações que serão mandadas pelo microcomputador, o decodificador necessitava apenas reconhecer o padrão enviado e escrever nas variáveis de referência dos motores.

Foi optado pela implementação dos métodos de codificação e decodificação sem o auxílio de bibliotecas como a cJSON (DAVEGAMBLE, 2019), pois as estruturas são bem definidas. A utilização de bibliotecas de terceiros aumentaria a complexidade do código criado.

Para a decodificação foi feito uma busca no *array* de caracteres recebido e as aspas foram utilizadas como sinalizadores e o caractere que identifica o final da mensagem foi o caractere vazio "\0", caso a mensagem recebida não possua a mesma estrutura da esperada, a alteração não é realizada.

5.5 Simulação de um sistema

A arquitetura desejada do sistema é mostrada na figura 20, onde a interface é composta pelas funções de codificação e decodificação JSON e nas funções de transmissão e recebimento das mensagens. A função dos programas feitos no simulink é para implementar técnicas de controle, já o sistema embarcado tem o papel de receber as novas referências e aplica-las em seu processo.

Para validação da arquitetura foi simulado um sistema mostrado na equação 5.1, para a sua implementação no sistema embarcado utilizou-se a função **c2d** para transformar em um modelo discreto no tempo com uma taxa de amostragem de $0.05s$ obtendo a equação 5.2, que foi implementada via código. Foram utilizados controladores quaisquer apenas para representar uma dinâmica, os controladores utilizados possuíam os ganhos mostrados na tabela 2.

$$G_s = \frac{1}{s+1} \tag{5.1}$$

$$y(n) = 0.9512 * y(n-1) + 0.04877 * u(n-1) \tag{5.2}$$

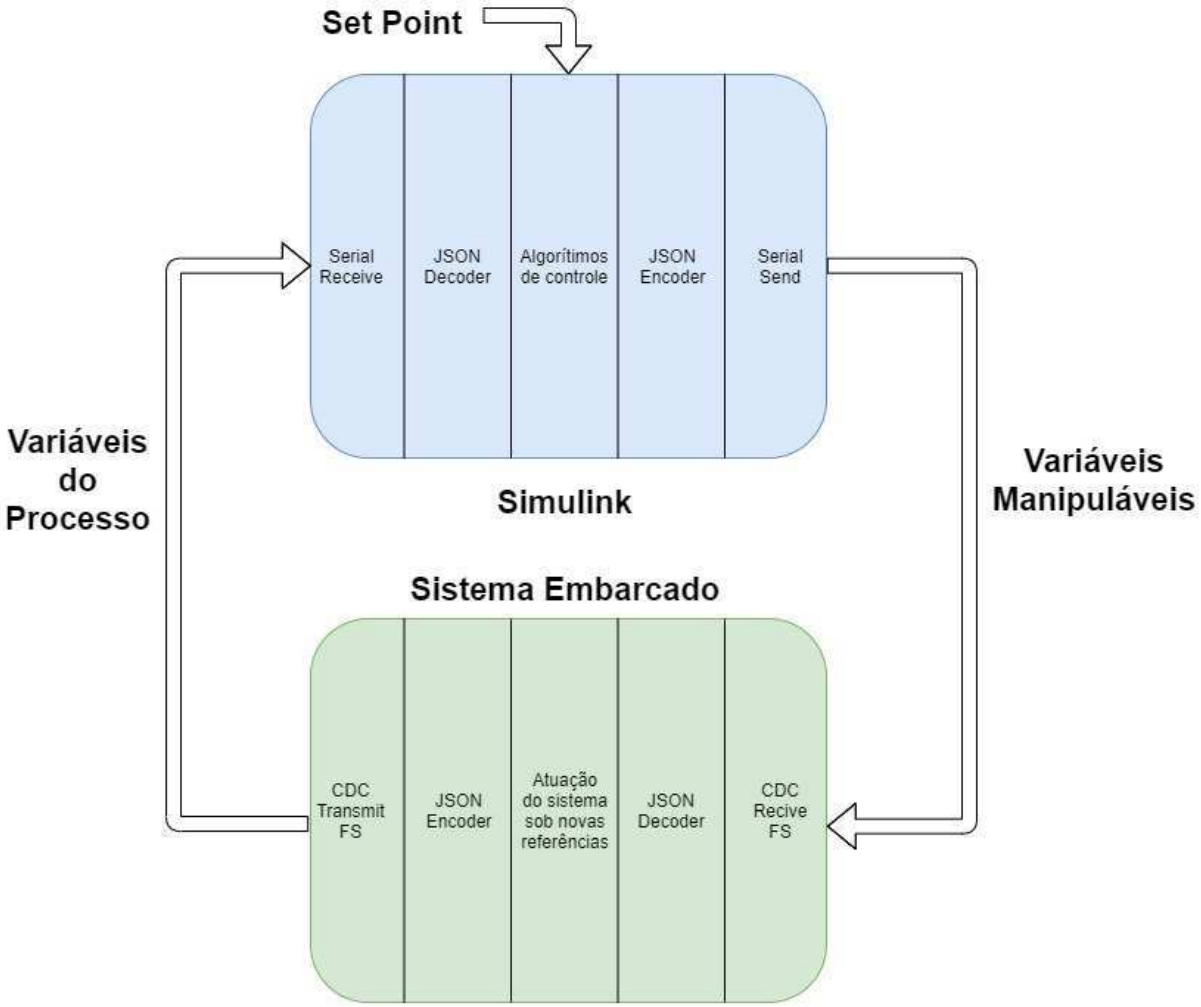
Tabela 2 – Ganhos utilizados nos controladores.

Controladores	Kp	Ki
PI	1	0.5

Fonte: autoria própria.

As saídas adquiridas foram as mostradas nas figuras 21 e 22, durante o experimento a referência foi alterada diversas vezes para observar se a interface USB estava de fato funcionando.

Figura 20 – Arquitetura do sistema.



Fonte: autoria própria.

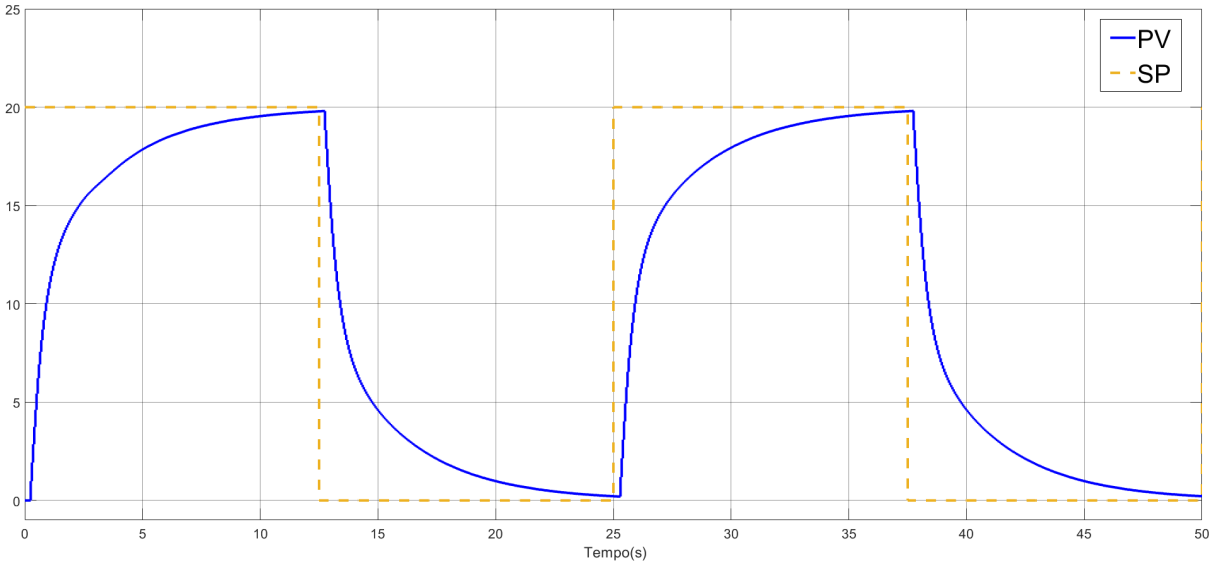
5.6 Verificação de Comportamentos Emergentes

Com a interface USB implementada foi realizado a integração da interface ao código do AGV do LIEC e verificar se a interface desenvolvida não ocasionaria alteração no funcionamento do sistema.

O principal comportamento indesejado era que ao receber as mensagens do micro-computador o sistema embarcado interrompesse o cálculo dos controladores de corrente dos motores, o que ocasionaria problemas no acionamento dos motores que por sua vez danificaria os componentes do sistema de potencia.

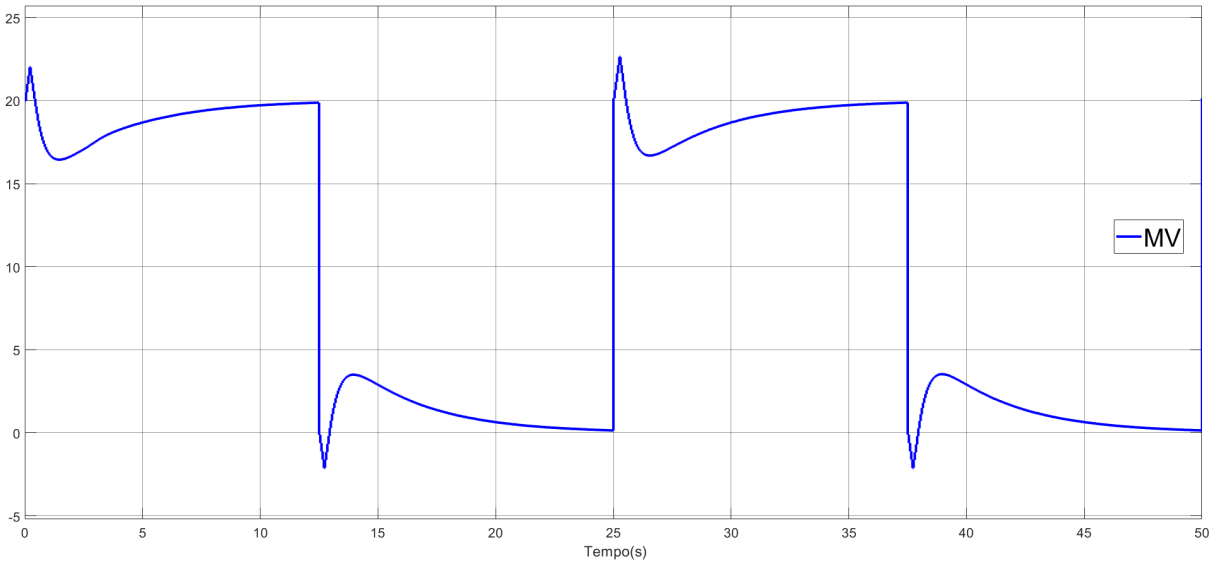
Para a verificação dessa condição a prioridade da interrupção gerada ao receber uma mensagem no sistema embarcado foi diminuída para que a mesma fosse menor do que a prioridade do cálculo dos controladores. Após esta alteração na geração do código, foi realizado a migração da interface para o código do AGV e testada em uma placa de testes.

Figura 21 – Referência e saída do sistema simulado.



Fonte: autoria própria.

Figura 22 – Variável manipulável do experimento simulado.



Fonte: autoria própria.

Ao final da migração foi o sistema foi testado com o mesmo programa do simulink, para verificar o funcionamento da interface USB funcionando no código do próprio AGV.

Para verificar a execução das interrupções no código usou-se um analisador lógico. Para que isso fosse possível utilizou-se 3 pinos da GPIO da placa para serem alterados durante a execução do código, ao inicia-se uma interrupção o pino associado tinha o seu valor colocado em alto e ao fim da interrupção o seu valor era colocado em baixo, gerando uma onda quadrada, onde o tempo em que se estava em alto era o tempo para execução, como pode ser visto na figura 23. Na tabela 3 tem-se a relação dos pinos e seus respectivos

significados.

Tabela 3 – Relação de pino e significado.

Interrupções	Timers	GPIO
Controle Corrente (5kHz)	TIM6	PE3
Leitura Sensor Magnético (20Hz)	TIM14	PE5
Recebimento de referências (USB)	-	PE6

Fonte: autoria própria.

Figura 23 – Lógica utilizada para identificar as interrupções.

```
649     if(htim->Instance == TIM6)
650     {
651         HAL_GPIO_WritePin(Out5K_GPIO_Port, Out5K_Pin, SET);
652         TIM6_UP_ISR();
653         HAL_GPIO_WritePin(Out5K_GPIO_Port, Out5K_Pin, RESET);
654     }
655
656     if(htim->Instance == TIM14)
657     {
658         HAL_GPIO_WritePin(OutUSB_GPIO_Port, OutUSB_Pin, SET);
659         TIM14_UP_ISR();
660         HAL_GPIO_WritePin(OutUSB_GPIO_Port, OutUSB_Pin, RESET);
661     }
```

Fonte: autoria própria.

Utilizou-se o mesmo programa do simulink do teste de interface USB e com o auxílio do analisador lógico foi possível verificar como as interrupções estavam sendo realizadas. Na tabela 4 temos o significado dos canais utilizados do analisador. Na figura 24 pode ser visto como os testes foram realizados na placa de desenvolvimento.

Tabela 4 – Relação de canal do analisador e pinos do sistema embarcado.

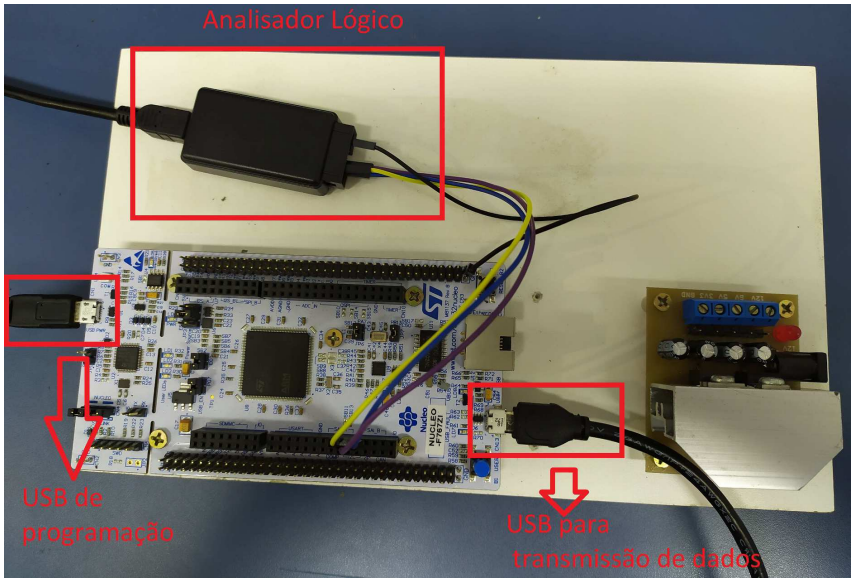
Canal analizador	GPIO
Canal 0: Controle Corrente (5kHz)	PE3
Canal 1: Leitura Sensor Magnético (20Hz)	PE5
Canal 2: Recebimento de referências	PE6

Fonte: autoria própria.

Nas condições normais o comportamento do sistema pode ser observado na figura 25, nesta figura está representado o primeiro segundo, porém o comportamento se repete durante o experimento. Na figura 25 é possível ver a cada envio do sistema embarcado o microcomputador responde antes de uma próximo envio.

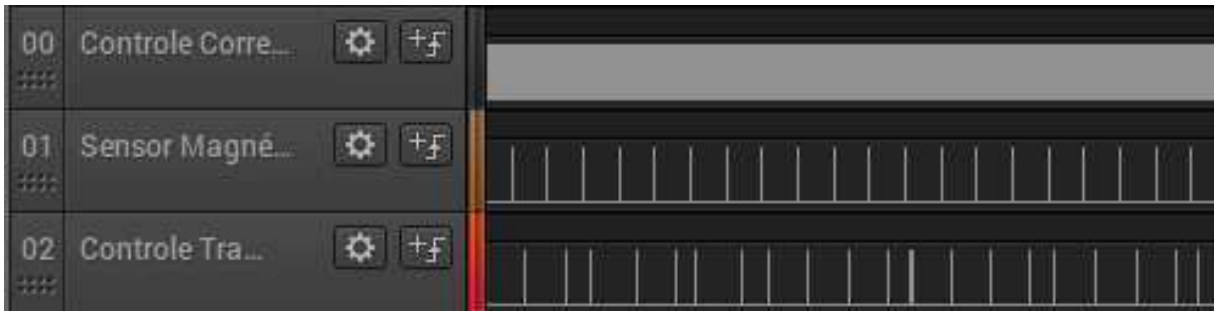
Para observar se a interrupção da comunicação USB influencia no cálculo da malha de corrente é necessário ampliar as informações do analisador lógico, como pode ver visto na figura 26. Em todo o experimento esse cenário foi respeitado, a interrupção do USB ocorreu no momento ocioso da malha de corrente.

Figura 24 – Bancada de teste utilizada.



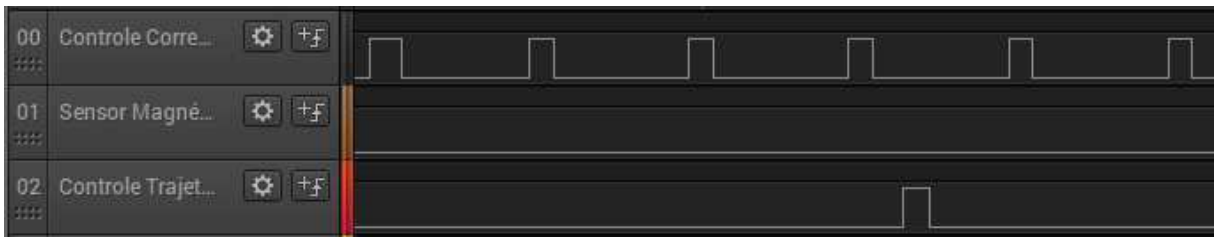
Fonte: autoria própria.

Figura 25 – Parte do experimento a $20Hz$.



Fonte: autoria própria.

Figura 26 – Experimento a $20Hz$ ampliado.



Fonte: autoria própria.

Não foi necessário observar o enviou dos dados pela interrupção do sensor magnético, pois este já foi validado pelo laboratório.

5.7 Limite da arquitetura

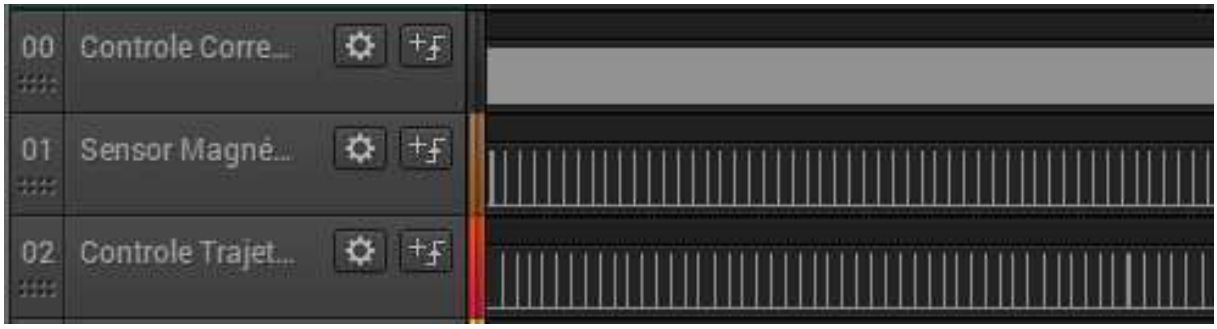
Foram realizados testes mantendo a frequência da malha de corrente em 5kHz e aumentando a frequência do controle de trajetória para verificar as limitações existentes na arquitetura.

Ao aumentar a frequência do controle de trajetória, houve a necessidade da alteração do tempo de amostragem do bloco *Serial Receive* no simulink para que este esteja em condizente com a frequência desejada.

Os experimentos se deram aumentando a frequência em $5Hz$ por experimento e observando o resultado no analisador lógico, os resultados foram bem semelhantes, então abordaremos algumas frequências especificas.

Na figura 27 podemos observar o experimento na frequência de 50 Hz, começou-se a observar uma que as respostas começaram a se aproximar da próxima requisição em alguns instantes, porém não ultrapassou. A variação no tempo de resposta se dá pelo fato de que o Simulink não ter prioridade a utilização da CPU, pois no Windows as tarefas são executadas em paralelo, ou seja, o desempenho é influenciado pelos programas utilizados em paralelo.

Figura 27 – Experimento a $50Hz$.



Fonte: autoria própria.

Ao utilizar uma frequência de 80 Hz, observou-se o atraso no cálculo das referências e transmissão para o sistema embarcado, como visto na figura 28. Porém ao diminuir o tempo de amostragem do bloco *Serial Receive* o sistema voltou a cumprir seus prazos.

No experimento realizado a uma frequência de $100Hz$ e utilizando uma taxa de amostragem de $0,002s$, o sistema ficou mais sensível aos processos em paralelo, o cumprimento dos prazos ocorreram, porém em alguns momentos quando outra tarefa demandava mais recursos do sistema operacional ocorria atrasos, como observado na figura 29.

Também nos experimentos realizados na frequência de $100Hz$ foi a maior prioridade da malha de corrente. Na figura 30, pode ser observado que durante o recebimento das novas referências via a interface USB a malha de corrente interrompeu o recebimento,

Figura 28 – Experimento a $80Hz$.

Fonte: autoria própria.

Figura 29 – Experimento a $100Hz$.

Fonte: autoria própria.

realizou seu cálculo e apenas após ser finalizada deu-se continuidade a comunicação USB.

Figura 30 – Experimento a $100Hz$ prioridade da malha de corrente.

Fonte: autoria própria.

6 Considerações Finais

Mediante aos testes realizados a utilização da arquitetura sugerida é possível e a frequência máxima da malha desejada dependerá da capacidade do micro-computador utilizado e dos programas que estão sendo executados em paralelo. Não é possível manter uma periodicidade nas transmissões do micro-computador, porém é realizável a computação e transmissão antes de uma nova requisição.

Não foi possível a realização de testes no AGV do laboratório, porém os testes realizados utilizando o mesmo código comprovam que o sistema é viável e passível de ser implementado. O trabalho estende-se a quaisquer aplicação em tempo real onde a frequência da tarefa a ser executada é de algumas dezenas de Hz, modificando-se a mensagem transmitida.

Por fim, após a adição deste trabalho na plataforma de aprendizagem do AGV possibilitará o avanço de pesquisa de técnicas avançadas para o seu controle de trajetória, utilizando a fita magnética ou pela existência de um mini-computador no AGV tem-se a possibilidade da utilização de visão computacional e aprendizado de máquina com a mesma finalidade.

Referências

Anggraeni, P. et al. Development of a wireless communication platform for multiple-mobile robots using ros. In: *2018 6th International Conference on Control Engineering Information Technology (CEIT)*. [S.l.: s.n.], 2018. p. 1–6. ISSN null. Citado na página 7.

CPLUSPLUS. Disponível em: <http://www.cplusplus.com/>. Citado na página 17.

DAVEGAMBLE. *DaveGamble/cJSON*. 2019. Disponível em: <https://github.com/DaveGamble/cJSON>. Citado na página 19.

DIAS, C. C.; LIMA, R. B. C.; BARROS, P. R. Projeto baseado em modelo no desenvolvimento de um controlador de velocidade para motor. *SBAI*, 2019. XIV Simpósio Brasileiro de Automação Inteligente. Citado na página 5.

Digani, V. et al. Hierarchical traffic control for partially decentralized coordination of multi agv systems in industrial environments. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.: s.n.], 2014. p. 6144–6149. Citado na página 5.

KALIAPPAN, S. et al. Mechanical design and analysis of agv for cost reduction of material handling in automobile industries. *International Research Journal of Automotive Technology*, v. 1, n. 1, p. 1–7, Jan. 2018. Disponível em: <https://mapletreejournals.com/index.php/IRJAT/article/view/1>. Citado na página 12.

Kobayashi, T.; Yamamoto, H.; Yamazaki, K. Zigbee network system for observing operating activities of work vehicles. In: *2016 International Conference on Computing, Networking and Communications (ICNC)*. [S.l.: s.n.], 2016. p. 1–5. ISSN null. Citado na página 7.

KORVER, N. Adequacy of the universal serial bus for real-time systems. *University of Twente, Tech. Rep. 009CE2003*, 2003. Citado na página 8.

LEE, E.; SESHIA, S. *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*. MIT Press, 2016. (The MIT Press). ISBN 9780262533812. Disponível em: <https://books.google.com.br/books?id=chPiDQAAQBAJ>. Citado na página 3.

LEE, J.; BAGHERI, B.; KAO, H.-A. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters*, v. 3, p. 18 – 23, 2015. ISSN 2213-8463. Disponível em: <http://www.sciencedirect.com/science/article/pii/S221384631400025X>. Citado na página 4.

LEE, S.-Y.; YANG, H.-W. Navigation of automated guided vehicles using magnet spot guidance method. *Robotics and Computer-Integrated Manufacturing*, v. 28, n. 3, p. 425 – 436, 2012. ISSN 0736-5845. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0736584511001372>. Citado na página 6.

LIU, J. *Real-Time Systems*. Prentice Hall, 2000. ISBN 9780130996510. Disponível em: <https://books.google.com.br/books?id=855QAAAAMAAJ>. Citado 2 vezes nas páginas 2 e 3.

Lynch, L. et al. Automated ground vehicle (agv) and sensor technologies- a review. In: *2018 12th International Conference on Sensing Technology (ICST)*. [S.l.: s.n.], 2018. p. 347–352. Citado na página 5.

MOHSENI, F.; VORONOV, S.; FRISK, E. Deep learning model predictive control for autonomous driving in unknown environments. *IFAC-PapersOnLine*, v. 51, n. 22, p. 447 – 452, 2018. ISSN 2405-8963. 12th IFAC Symposium on Robot Control SYROCO 2018. Disponível em: <http://www.sciencedirect.com/science/article/pii/S2405896318333007>. Citado na página 4.

STMICROELETRONICS. Disponível em: <https://www.st.com/content/st_com/en.html>. Citado na página 9.

Tamara, M. N. et al. Electronics system design for low cost agv type forklift. In: *2018 International Conference on Applied Science and Technology (iCAST)*. [S.l.: s.n.], 2018. p. 464–469. ISSN null. Citado 2 vezes nas páginas 1 e 5.

ZHANG, D. et al. The communication system between web application host computers and embedded systems based on node. js. In: IEEE. *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. [S.l.], 2017. p. 1–5. Citado na página 7.