Software Serial

1.0

Gerado por Doxygen 1.10.0

Sumário

Capítulo 1

Namespaces

1.1 Lista de Namespaces

Esta é a lista de todos os Namespaces com suas respectivas descrições:

Capítulo 2

Índice Hierárquico

2.1 Hierarquia de Classes

Esta lista de hierarquias está parcialmente ordenada (ordem alfabética):

Capítulo 3

Índice dos Componentes

3.1 Lista de Classes

Aqui estão as classes, estruturas, uniões e interfaces e suas respectivas descrições:

Capítulo 4

Índice dos Arquivos

4.1 Lista de Arquivos

Esta é a lista de todos os arquivos e suas respectivas descrições:

Capítulo 5

Namespace

5.1 Refência do Namespace envia

Componentes

- class XBeeInterface

Variáveis

- root = tk.Tk()
- app = XBeeInterface(root)

5.1.1 Descrição detalhada

```
@file XBeeInterface.py
@brief Módulo para interface de envio de dados XBee.

Este módulo fornece uma classe XBeeInterface para interface de envio de dados XBee,
incluindo a criação dinâmica de botões para envio de pacotes hexadecimais com intervalos configuráveis.

@author Francisco Gilson Pereira Almeida Filho
@date 06 de Fevereiro de 2024
```

5.1.2 Variáveis

5.1.2.1 app

```
envia.app = XBeeInterface(root)
```

5.1.2.2 root

```
envia.root = tk.Tk()
```

5.2 Refência do Namespace main

Variáveis

- app = XBeeDataViewer()

5.2.1 Descrição detalhada

```
@file main.py
Módulo principal para iniciar a aplicação XBeeDataViewer.

Este módulo importa a classe XBeeDataViewer do arquivo XBeeDataViewer.py e
inicia a aplicação XBeeDataViewer.

@author Francisco Gilson Pereira Almeida Filho
@date 06 de Fevereiro de 2024
```

5.2.2 Variáveis

5.2.2.1 app

```
main.app = XBeeDataViewer()
```

5.3 Refência do Namespace RealXBeeData

Componentes

- class RealXBeeData

5.3.1 Descrição detalhada

```
@file RealXBeeData.py
@brief Módulo para comunicação com dispositivo XBee em tempo real.

Este módulo fornece uma classe RealXBeeData para comunicação com um dispositivo
XBee em tempo real, incluindo métodos para iniciar e interromper a comunicação,
receber dados do dispositivo, e baixar os dados recebidos para um arquivo de texto.

@author [Francisco Gilson Pereira Almeida Filho]
@date [06 de Fevereiro de 2024]
```

5.4 Refência do Namespace XBeeDataViewer

Componentes

- class XBeeDataViewer

5.4.1 Descrição detalhada

```
@file XBeeDataViewer.py
@brief Módulo para visualização de dados do XBee.

Este módulo fornece uma classe XBeeDataViewer para visualização de dados do XBee,
incluindo a interface gráfica e a comunicação com o dispositivo XBee.

@author Francisco Gilson Pereira Almeida Filho
@date 06 de Fevereiro de 2024
```

Capítulo 6

Classes

6.1 Referência da Classe RealXBeeData.RealXBeeData

Membros Públicos

- __init__ (self, app, port, baudrate)
- start_real_communication (self)
- stop_real_communication (self)
- join_threads (self)
- receive_data (self)
- download_data (self)

Atributos Públicos

- app
- data_counter
- running
- serial_port
- stop_flag
- port
- baudrate
- buffer
- received_data
- receive_thread
- buffer_lock

6.1.1 Descrição detalhada

```
Classe para comunicação com um dispositivo XBee real.
```

6.1.2 Construtores e Destrutores

6.1.2.1 __init__()

```
RealXBeeData.RealXBeeData.__init__ (
              self,
              app,
              port,
              baudrate )
```

Inicializa uma nova instância de RealXBeeData.

Args:
    app: A aplicação que utiliza a classe RealXBeeData.
    port (str): A porta serial à qual o dispositivo XBee está conectado.
    baudrate (int): A taxa de baud do dispositivo XBee.

```
00023     def __init__(self, app, port, baudrate):
00024         """Inicializa uma nova instância de RealXBeeData.
00025 
00026         Args:
00027             app: A aplicação que utiliza a classe RealXBeeData.
00028             port (str): A porta serial à qual o dispositivo XBee está conectado.
00029             baudrate (int): A taxa de baud do dispositivo XBee.
00030         """
00031         self.app = app
00032         self.data_counter = 0
00033         self.running = False
00034         self.serial_port = None
00035         self.stop_flag = threading.Event()
00036         self.port = port
00037         self.baudrate = baudrate
00038         self.buffer = bytearray()
00039         self.received_data = []
00040         self.receive_thread = None
00041         self.buffer_lock = threading.Lock()
00042 
```

6.1.3 Documentação das funções

6.1.3.1 download_data()

```
RealXBeeData.RealXBeeData.download_data (
              self )
```

Baixa os dados recebidos do dispositivo XBee e os salva em um arquivo de texto.

```
00086     def download_data(self):
00087         """Baixa os dados recebidos do dispositivo XBee e os salva em um arquivo de texto."""
00088         filename_base = 'data_log'
00089         extension = 'txt'
00090 
00091         i = 1
00092         while True:
00093             filename = f"{filename_base}{i}.{extension}"
00094             if not os.path.exists(filename):
00095                 break
00096             i += 1
00097 
00098         # Inicializa contadores
00099         quantidade_silabas_e1 = 0
00100         numero_linhas = 1
00101 
00102         with open(filename, 'w+') as file:
00103             # Preenche o arquivo e conta as sílabas "E1" e o número de linhas
00104             for timestamp, data_hex in self.received_data:
00105                 file.write(f"{timestamp} - XBEE3: {data_hex}\n")
00106                 quantidade_silabas_e1 += data_hex.upper().count("E1")
00107                 numero_linhas += 1
00108 
```

```
00109             # Obtém o conteúdo atual do arquivo
00110             file.seek(0)
00111             existing_content = file.read()
00112 
00113             # Reinicia o cursor para o início e escreve o relatório
00114             file.seek(0)
00115             file.write(f"{'-'*50}\n")
00116             file.write(f"{'-'*50}\n")
00117             file.write(f"Quantidade de Erros de Pacote: {quantidade_silabas_e1}\n")
00118             file.write(f"Quantidade de Linhas: {numero_linhas}\n")
00119             file.write(f"Porcentagem de erro: {(quantidade_silabas_e1/numero_linhas)*100}%\n")
00120             file.write(f"{'-'*50}\n")
00121             file.write(f"{'-'*50}\n")
00122 
00123             # Adiciona de volta o conteúdo anterior
00124             file.write(existing_content)
00125 
00126         print(f"Dados baixados e salvos em: {filename}
```

6.1.3.2 join_threads()

```
RealXBeeData.RealXBeeData.join_threads (
              self )
```

Aguarda a finalização das threads em execução.

```
00053     def join_threads(self):
00054         """Aguarda a finalização das threads em execução."""
00055         if self.receive_thread:
00056             self.receive_thread.join()
00057 
```

6.1.3.3 receive_data()

```
RealXBeeData.RealXBeeData.receive_data (
              self )
```

Recebe os dados do dispositivo XBee e os processa.

```
00058     def receive_data(self):
00059         """Recebe os dados do dispositivo XBee e os processa."""
00060         try:
00061             self.running = True
00062             self.serial_port = serial.Serial(self.port, self.baudrate, timeout=0.1)
00063             while not self.stop_flag.is_set():
00064                 byte = self.serial_port.read(1)
00065                 if byte:
00066                     with self.buffer_lock:
00067                         self.buffer.append(byte[0])
00068                 else:
00069                     with self.buffer_lock:
00070                         if len(self.buffer) > 0:
00071                             timestamp = time.strftime('%Y-%m-%d %H:%M:%S.') + str(int(time.time() *
      1000) % 1000).zfill(3)
00072                             data_hex = ' '.join(f'{b:02X}' for b in self.buffer)
00073                             self.app.update_serial_monitor(f"{timestamp}, Data: {data_hex}\n")
00074                             real_data = {'source': 'XBEE3', 'data': data_hex, 'timestamp': timestamp}
00075                             self.app.update_data_tree(real_data)
00076                             self.received_data.append((timestamp, data_hex))
00077                             self.buffer = bytearray()
00078                             self.data_counter += 1  # Incrementa o contador de dados
00079         except Exception as e:
00080             print(f"Error in receive_data: {e}")
00081         finally:
00082             self.running = False
00083             if self.serial_port and self.serial_port.is_open:
00084                 self.serial_port.close()
00085 
```

6.1.3.4 start_real_communication()

```
RealXBeeData.RealXBeeData.start_real_communication (
              self )
```

Inicia a comunicação com o dispositivo XBee.

```
00043     def start_real_communication(self):
00044         """Inicia a comunicação com o dispositivo XBee."""
00045         self.stop_flag.clear()
00046         self.receive_thread = threading.Thread(target=self.receive_data)
00047         self.receive_thread.start()
00048
```

6.1.3.5 stop_real_communication()

```
RealXBeeData.RealXBeeData.stop_real_communication (
              self )
```

Interrompe a comunicação com o dispositivo XBee.

```
00049     def stop_real_communication(self):
00050         """Interrompe a comunicação com o dispositivo XBee."""
00051         self.stop_flag.set()
00052
```

6.1.4 Atributos

6.1.4.1 app

```
RealXBeeData.RealXBeeData.app
```

6.1.4.2 baudrate

```
RealXBeeData.RealXBeeData.baudrate
```

6.1.4.3 buffer

```
RealXBeeData.RealXBeeData.buffer
```

6.1.4.4 buffer_lock

```
RealXBeeData.RealXBeeData.buffer_lock
```

6.1.4.5 data_counter

```
RealXBeeData.RealXBeeData.data_counter
```

6.1.4.6 port

`RealXBeeData.RealXBeeData.port`

6.1.4.7 receive_thread

`RealXBeeData.RealXBeeData.receive_thread`

6.1.4.8 received_data

`RealXBeeData.RealXBeeData.received_data`

6.1.4.9 running

`RealXBeeData.RealXBeeData.running`

6.1.4.10 serial_port

`RealXBeeData.RealXBeeData.serial_port`

6.1.4.11 stop_flag

`RealXBeeData.RealXBeeData.stop_flag`

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Users/Gilson/Desktop/Xbee_Serial/RealXBeeData.py

6.2 Referência da Classe XBeeDataViewer.XBeeDataViewer

Diagrama de hierarquia da classe XBeeDataViewer.XBeeDataViewer:

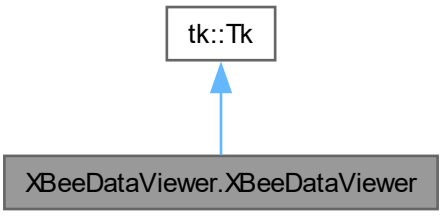
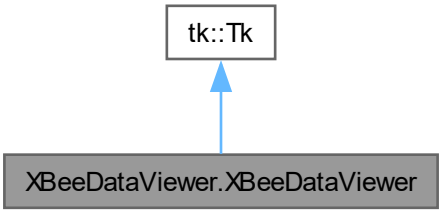


Diagrama de colaboração para XBeeDataViewer.XBeeDataViewer:



Membros Públicos

- __init__ (self, *args, **kwargs)
- create_widgets (self)
- start_real_communication (self)
- stop_real_communication (self)
- download_data (self)
- update_serial_monitor (self, text)
- update_data_tree (self, data)
- clear_monitor (self)
- clear_data (self)
- exit_application (self)

Atributos Públicos

- real_xbee
- exit_application
- paned_window
- serial_monitor_label
- serial_monitor
- data_analysis_label
- data_tree
- button_frame
- start_button
- stop_button
- download_button
- clear_monitor_button
- clear_data_button
- exit_button

6.2.1 Descrição detalhada

```
Classe para visualização de dados do XBee.
```

6.2.2 Construtores e Destrutores

6.2.2.1 __init__()

```
XBeeDataViewer.XBeeDataViewer.__init__ (
              self,
             * args,
             ** kwargs )
```

Inicializa a aplicação XBeeDataViewer.

Configura a interface gráfica e a comunicação com o dispositivo XBee.

Args:
    args: Argumentos posicionais.
    kwargs: Argumentos de palavra-chave.

```
00023     def __init__(self, *args, **kwargs):
00024         """Inicializa a aplicação XBeeDataViewer.
00025 
00026         Configura a interface gráfica e a comunicação com o dispositivo XBee.
00027 
00028         Args:
00029             args: Argumentos posicionais.
00030             kwargs: Argumentos de palavra-chave.
00031         """
00032         tk.Tk.__init__(self, *args, **kwargs)
00033 
00034         self.title("XBee Data Viewer")
00035         self.geometry("900x700")
00036 
00037         # Utilizar RealXBeeData para comunicação real
00038         self.real_xbee = RealXBeeData(self, port="COM4", baudrate=9600)
00039 
00040         # Configurar a chamada de download_data ao fechar a aplicação
00041         self.protocol("WM_DELETE_WINDOW", self.exit_application)
00042 
00043         # Criar e configurar widgets
00044         self.create_widgets()
00045 
```

6.2.3 Documentação das funções

6.2.3.1 clear_data()

```
XBeeDataViewer.XBeeDataViewer.clear_data (
              self )
```

Limpa a árvore de dados.

```
00116     def clear_data(self):
00117         """Limpa a árvore de dados."""
00118         self.data_tree.delete(*self.data_tree.get_children())
00119 
```

6.2.3.2 clear_monitor()

```
XBeeDataViewer.XBeeDataViewer.clear_monitor (
              self )
```

Limpa o monitor serial.

```
00112     def clear_monitor(self):
00113         """Limpa o monitor serial."""
00114         self.serial_monitor.delete(1.0, tk.END)
00115 
```

6.2.3.3 create_widgets()

```
XBeeDataViewer.XBeeDataViewer.create_widgets (
              self )
```

Cria e configura os widgets da interface gráfica.

```
00046     def create_widgets(self):
00047         """Cria e configura os widgets da interface gráfica."""
00048         self.paned_window = ttk.PanedWindow(self, orient=tk.VERTICAL)
00049         self.paned_window.pack(expand=True, fill='both')
00050 
00051         self.serial_monitor_label = tk.Label(self.paned_window, text="Monitor Serial:")
00052         self.serial_monitor_label.pack()
00053 
00054         self.serial_monitor = tk.Text(self.paned_window, height=10, width=70)
00055         self.serial_monitor.pack(expand=True, fill='both')
00056 
00057         self.paned_window.add(self.serial_monitor_label)
00058         self.paned_window.add(self.serial_monitor)
00059 
00060         self.data_analysis_label = tk.Label(self, text="Análise de Dados:")
00061         self.data_analysis_label.pack()
00062 
00063         self.data_tree = ttk.Treeview(self, columns=('Source', 'Data', 'Timestamp'), show='headings',
      height=10)
00064         self.data_tree.heading('Source', text='Source')
00065         self.data_tree.heading('Data', text='Data')
00066         self.data_tree.heading('Timestamp', text='Timestamp')
00067         self.data_tree.pack(expand=True, fill='both')
00068 
00069         self.button_frame = tk.Frame(self)
00070         self.button_frame.pack()
00071 
00072         self.start_button = tk.Button(self.button_frame, text="Iniciar Leitura",
      command=self.start_real_communication)
00073         self.start_button.grid(row=0, column=0, padx=10, pady=10)
00074 
00075         self.stop_button = tk.Button(self.button_frame, text="Parar Leitura",
      command=self.stop_real_communication)
00076         self.stop_button.grid(row=0, column=1, padx=10, pady=10)
00077 
00078         self.download_button = tk.Button(self.button_frame, text="Baixar Dados",
      command=self.download_data)
00079         self.download_button.grid(row=0, column=2, padx=10, pady=10)
00080 
00081         self.clear_monitor_button = tk.Button(self.button_frame, text="Limpar Monitor",
      command=self.clear_monitor)
00082         self.clear_monitor_button.grid(row=0, column=3, padx=10, pady=10)
00083 
00084         self.clear_data_button = tk.Button(self.button_frame, text="Limpar Dados",
      command=self.clear_data)
00085         self.clear_data_button.grid(row=0, column=4, padx=10, pady=10)
00086 
00087         self.exit_button = tk.Button(self.button_frame, text="Sair", command=self.exit_application)
00088         self.exit_button.grid(row=0, column=5, padx=10, pady=10)
00089 
```

6.2.3.4 download_data()

```
XBeeDataViewer.XBeeDataViewer.download_data (
              self )
```

Baixa os dados do dispositivo XBee.

```
00098     def download_data(self):
00099         """Baixa os dados do dispositivo XBee."""
00100         self.real_xbee.download_data()
00101 
```

6.2.3.5 exit_application()

```
XBeeDataViewer.XBeeDataViewer.exit_application (
              self )
```

Fecha a aplicação, interrompendo a comunicação e baixando os dados do dispositivo XBee.

```
00120     def exit_application(self):
00121         """Fecha a aplicação, interrompendo a comunicação e baixando os dados do dispositivo XBee."""
00122         self.real_xbee.download_data()  # Chama a função download_data ao fechar a aplicação
00123         self.stop_real_communication()
00124         self.destroy()
```

6.2.3.6 start_real_communication()

```
XBeeDataViewer.XBeeDataViewer.start_real_communication (
              self )
```

Inicia a comunicação com o dispositivo XBee.

```
00090     def start_real_communication(self):
00091         """Inicia a comunicação com o dispositivo XBee."""
00092         self.real_xbee.start_real_communication()
00093
```

6.2.3.7 stop_real_communication()

```
XBeeDataViewer.XBeeDataViewer.stop_real_communication (
              self )
```

Interrompe a comunicação com o dispositivo XBee.

```
00094     def stop_real_communication(self):
00095         """Interrompe a comunicação com o dispositivo XBee."""
00096         self.real_xbee.stop_real_communication()
00097
```

6.2.3.8 update_data_tree()

```
XBeeDataViewer.XBeeDataViewer.update_data_tree (
              self,
              data )
```

Atualiza a árvore de dados com os dados especificados.

```
00107     def update_data_tree(self, data):
00108         """Atualiza a árvore de dados com os dados especificados."""
00109         values = (data['source'], data['data'], data['timestamp'])
00110         self.data_tree.insert('', 0, values=values)
00111
```

6.2.3.9 update_serial_monitor()

```
XBeeDataViewer.XBeeDataViewer.update_serial_monitor (
              self,
              text )
```

Atualiza o monitor serial com o texto especificado.

```
00102     def update_serial_monitor(self, text):
00103         """Atualiza o monitor serial com o texto especificado."""
00104         self.serial_monitor.insert(tk.END, text)
00105         self.serial_monitor.see(tk.END)
00106
```

6.2.4 Atributos

6.2.4.1 button_frame

```
XBeeDataViewer.XBeeDataViewer.button_frame
```

6.2.4.2 clear_data_button

```
XBeeDataViewer.XBeeDataViewer.clear_data_button
```

6.2.4.3 clear_monitor_button

```
XBeeDataViewer.XBeeDataViewer.clear_monitor_button
```

6.2.4.4 data_analysis_label

```
XBeeDataViewer.XBeeDataViewer.data_analysis_label
```

6.2.4.5 data_tree

```
XBeeDataViewer.XBeeDataViewer.data_tree
```

6.2.4.6 download_button

```
XBeeDataViewer.XBeeDataViewer.download_button
```

6.2.4.7 exit_application

```
XBeeDataViewer.XBeeDataViewer.exit_application
```

6.2.4.8 exit_button

XBeeDataViewer.XBeeDataViewer.exit_button

6.2.4.9 paned_window

XBeeDataViewer.XBeeDataViewer.paned_window

6.2.4.10 real_xbee

XBeeDataViewer.XBeeDataViewer.real_xbee

6.2.4.11 serial_monitor

XBeeDataViewer.XBeeDataViewer.serial_monitor

6.2.4.12 serial_monitor_label

XBeeDataViewer.XBeeDataViewer.serial_monitor_label

6.2.4.13 start_button

XBeeDataViewer.XBeeDataViewer.start_button

6.2.4.14 stop_button

XBeeDataViewer.XBeeDataViewer.stop_button

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Users/Gilson/Desktop/Xbee_Serial/XBeeDataViewer.py

6.3 Referência da Classe envia.XBeeInterface

Membros Públicos

- __init__ (self, root)
- add_button (self)
- toggle_transmission (self, button, title, packet, interval)
- stop_transmission (self)
- transmit_data_loop (self, title, packet, interval)

Atributos Públicos

- root
- buttons_frame
- buttons
- is_transmitting
- stop_button
- result_label

6.3.1 Construtores e Destrutores

6.3.1.1 __init__()

```
envia.XBeeInterface.__init__ (
              self,
              root )
```

Inicializa a aplicação XBeeInterface.

Configura a interface gráfica e as variáveis de controle.

:param root: O widget raiz da aplicação.

```
00020     def __init__(self, root):
00021         """
00022         Inicializa a aplicação XBeeInterface.
00023 
00024         Configura a interface gráfica e as variáveis de controle.
00025 
00026         :param root: O widget raiz da aplicação.
00027         """
00028         self.root = root
00029         self.root.title("XBee Interface")
00030 
00031         self.buttons_frame = tk.Frame(self.root)
00032         self.buttons_frame.pack()
00033 
00034         self.buttons = []  # Lista para armazenar os botões dinamicamente criados
00035         self.is_transmitting = False  # Variável para rastrear se a transmissão está acontecendo
00036 
00037         # Botão fixo para parar
00038         self.stop_button = tk.Button(self.buttons_frame, text="Parar", command=self.stop_transmission,
      state=tk.DISABLED)
00039         self.stop_button.pack(side=tk.LEFT)
00040 
00041         # Botão para adicionar novo botão
00042         tk.Button(self.buttons_frame, text="Adicionar Botão",
      command=self.add_button).pack(side=tk.LEFT)
00043 
00044         # Área para exibir feedback
00045         self.result_label = tk.Label(self.root, text="")
00046         self.result_label.pack()
00047 
```

6.3.2 Documentação das funções

6.3.2.1 add_button()

```
envia.XBeeInterface.add_button (
              self )
```

Adiciona um novo botão para iniciar/parar a transmissão de dados.

```
00048     def add_button(self):
00049         """
00050         Adiciona um novo botão para iniciar/parar a transmissão de dados.
00051         """
00052         # Frame para agrupar elementos relacionados a um botão
00053         button_frame = tk.Frame(self.buttons_frame)
00054         button_frame.pack(pady=10)
00055 
00056         # Variáveis para armazenar as configurações do botão
00057         title_var = tk.StringVar()
00058         packet_var = tk.StringVar()
00059         interval_var = tk.DoubleVar(value=1.0)
00060 
00061         # Campos de entrada para configurar o botão
00062         tk.Label(button_frame, text="Título:").pack()
00063         tk.Entry(button_frame, textvariable=title_var).pack()
00064 
00065         tk.Label(button_frame, text="Pacote Hexadecimal:").pack()
00066         tk.Entry(button_frame, textvariable=packet_var).pack()
00067 
00068         tk.Label(button_frame, text="Intervalo de Envio (segundos):").pack()
00069         tk.Entry(button_frame, textvariable=interval_var).pack()
00070 
00071         # Botão para iniciar/parar a transmissão deste botão específico
00072         new_button = tk.Button(button_frame, text="Iniciar/Parar", command=lambda:
      self.toggle_transmission(new_button, title_var.get(), packet_var.get(), interval_var.get()))
00073         new_button.pack()
00074 
00075         # Adicionar o novo botão à lista
00076         self.buttons.append(new_button)
00077 
```

6.3.2.2 stop_transmission()

```
envia.XBeeInterface.stop_transmission (
              self )
```

Interrompe a transmissão de dados.

```
00102     def stop_transmission(self):
00103         """
00104         Interrompe a transmissão de dados.
00105         """
00106         # Função para parar a transmissão quando o botão de parar é pressionado
00107         self.is_transmitting = False
00108         for button in self.buttons:
00109             button['state'] = 'normal'  # Reativar todos os botões de adicionar
00110         self.result_label.config(text="Transmissão interrompida.")
00111 
```

6.3.2.3 toggle_transmission()

```
envia.XBeeInterface.toggle_transmission (
              self,
              button,
              title,
              packet,
              interval )
```

Inicia ou interrompe a transmissão de dados.

:param button: O botão que disparou a ação.
:param title: O título do botão.
:param packet: O pacote hexadecimal a ser transmitido.
:param interval: O intervalo de envio em segundos.

```
00078     def toggle_transmission(self, button, title, packet, interval):
00079         """
00080         Inicia ou interrompe a transmissão de dados.
00081 
00082         :param button: O botão que disparou a ação.
00083         :param title: O título do botão.
00084         :param packet: O pacote hexadecimal a ser transmitido.
00085         :param interval: O intervalo de envio em segundos.
00086         """
00087         if not self.is_transmitting:
00088             # Iniciar a transmissão
00089             self.is_transmitting = True
00090             for b in self.buttons:
00091                 b['state'] = 'disabled'  # Desativar outros botões durante a transmissão
00092             self.stop_button['state'] = 'normal'  # Ativar o botão de parar durante a transmissão
00093             threading.Thread(target=self.transmit_data_loop, args=(title, packet, interval)).start()
00094         else:
00095             # Parar a transmissão
00096             self.is_transmitting = False
00097             for b in self.buttons:
00098                 b['state'] = 'normal'  # Reativar outros botões
00099             self.stop_button['state'] = 'disabled'  # Desativar o botão de parar
00100             self.result_label.config(text="Transmissão interrompida.")
00101 
```

6.3.2.4 transmit_data_loop()

envia.XBeeInterface.transmit_data_loop (
self,
title,
packet,
interval)

Loop de transmissão de dados.

Este método é executado em uma thread separada para transmitir os dados
com o intervalo especificado até que a transmissão seja interrompida.

:param title: O título do botão.
:param packet: O pacote hexadecimal a ser transmitido.
:param interval: O intervalo de envio em segundos.

```
00112     def transmit_data_loop(self, title, packet, interval):
00113         """
00114         Loop de transmissão de dados.
00115 
00116         Este método é executado em uma thread separada para transmitir os dados
00117         com o intervalo especificado até que a transmissão seja interrompida.
00118 
00119         :param title: O título do botão.
00120         :param packet: O pacote hexadecimal a ser transmitido.
00121         :param interval: O intervalo de envio em segundos.
00122         """
00123         try:
00124             with Serial('COM4', 9600, timeout=1) as ser:  # Substitua 'COM1' pela porta correta
00125                 while self.is_transmitting:
00126                     ser.write(bytes.fromhex(packet))
00127                     time.sleep(interval)
00128         except Exception as e:
00129             self.result_label.config(text=f"Erro na transmissão ({title}): {str(e)}")
00130 
```

6.3.3 Atributos

6.3.3.1 buttons

envia.XBeeInterface.buttons

6.3.3.2 buttons_frame

```
envia.XBeeInterface.buttons_frame
```

6.3.3.3 is_transmitting

```
envia.XBeeInterface.is_transmitting
```

6.3.3.4 result_label

```
envia.XBeeInterface.result_label
```

6.3.3.5 root

```
envia.XBeeInterface.root
```

6.3.3.6 stop_button

```
envia.XBeeInterface.stop_button
```

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Users/Gilson/Desktop/Xbee_Serial/envia.py

Capítulo 7

Arquivos

7.1 Referência do Arquivo C:/Users/Gilson/Desktop/Xbee_Serial/envia.py

Componentes

- class envia.XBeeInterface

Namespaces

- namespace envia

Variáveis

- envia.root = tk.Tk()
- envia.app = XBeeInterface(root)

7.2 Referência do Arquivo C:/Users/Gilson/Desktop/Xbee_Serial/main.py

Namespaces

- namespace main

Variáveis

- main.app = XBeeDataViewer()

7.3 Referência do Arquivo C:/Users/Gilson/Desktop/Xbee_Serial/RealXBeeData.py

Componentes

- class RealXBeeData.RealXBeeData

Namespaces

- namespace RealXBeeData

7.4 Referência do Arquivo C:/Users/Gilson/Desktop/Xbee_Serial/XBeeDataViewer.py

Componentes

- class XBeeDataViewer.XBeeDataViewer

Namespaces

- namespace XBeeDataViewer

Índice Remissivo