

**REFERÊNCIA PARA REALIZAR A COMUNICAÇÃO ENTRE
CLIENTE E SERVIDOR PARA CONTROLAR O ROBÔ**

Sobre esse documento

Esse documento foi desenvolvido com o intuito de apresentar os aspectos principais sobre o código desenvolvido para realizar a interação entre um dispositivo e o robô que pode ser montado utilizando os componentes listados no manual de montagem. Utilize esse arquivo como uma referência para compreender detalhes gerais sobre os códigos, e como operar corretamente a troca de mensagens entre cliente (dispositivo que envia comandos ao robô) e servidor (Arduíno Uno que controla as funcionalidades do robô).

Sumário

Índice

Comunicação.....	4
Sobre.....	4
Meio de Comunicação.....	4
<i>Padrão de envio de Dados</i>	4
Detalhes relevantes sobre a comunicação.....	4
Detalhes do Código Cliente Disponibilizado.....	6
Sobre.....	6
Requisitos para execução do código Cliente.....	6
Detalhes do Código Servidor Disponibilizado.....	8
Variáveis de Pinagem.....	10
Comandos.....	13
Erros.....	30
Sobre.....	30
Listagem de Erros.....	30

Comunicação

Sobre

O sistema para operar o robô pode ser observado como uma troca de mensagens entre um cliente e um servidor. O cliente refere-se basicamente ao dispositivo encarregado de enviar os comandos ao robô e receber os retornos. Já o servidor refere-se a parte responsável por controlar propriamente o robô, recebendo e interpretando os comandos para desencadear ações.

O papel de cliente pode ser realizado por qualquer dispositivo com capacidade de fazer a transmissão de dados via Bluetooth. Já o papel de servidor deve ser executado por um Arduino Uno que possua o código do servidor (disponibilizado junto a essa documentação) carregado. Esse Arduino deve ser usado para operar os circuitos que constituem o robô.

Meio de Comunicação

O meio de comunicação definido como padrão e implementado nos códigos do cliente e do servidor é via Bluetooth. Optou-se por esse meio devido a facilidade de implementação e o desempenho aceitável obtido através do uso do adaptador HC-05 em conjunto ao Arduino Uno.

Padrão de envio de Dados

Os dados enviados pelo cliente devem ser estritamente os comandos listados em seção posterior, uma vez que o código do servidor está preparado apenas para os aceitar como envios válidos. Já os dados enviados pelo servidor seguem um padrão que deve ser compreendido e descrito.

Por padrão optou-se que todos os dados retornados pelo servidor são binários formados por 4 bytes e que são manipulados internamente pelo sistema como dados float. Sendo assim todos os dados no servidor são passíveis de representação em formato float e ao serem enviados como retorno para o cliente, eles são enviados como uma sequência de 32 bits. Dessa forma, cabe ao cliente que recebe os dados interpretar e converter a informação binária para um formato numérico adequado.

Detalhes relevantes sobre a comunicação

Abaixo são listados alguns detalhes sobre a comunicação entre cliente e servidor que devem ser consideradas durante a execução dos códigos disponibilizados.

- A comunicação é síncrona;
- Após um comando enviado pelo cliente, o servidor sempre irá retornar algum dado. Para comandos que não esperam um dado específico, como comandos para configurar os componentes eletrônicos, o valor 1 é utilizado como um *flag* de retorno padrão;

- Por padrão os pinos 10 e 11 do Arduíno Uno utilizado para executar o servidor são reservados para a comunicação Bluetooth. Observe o documento de referência para montagem dos circuitos do robô para mais detalhes.

Detalhes do Código Cliente Disponibilizado

Sobre

Conforme abordado previamente, a comunicação com o robô funciona através de um modelo cliente e servidor. Observado isso, existe a necessidade de implementar um código para efetuar cada um desses papéis. A seção atual da documentação visa abordar e apresentar o código desenvolvido com a funcionalidade de cliente, sendo ele disponibilizado e recomendado para uso como padrão.

O código foi desenvolvido utilizando-se do paradigma orientado a objetos na linguagem python utilizando bibliotecas para comunicação Bluetooth. Todo o código para a comunicação está organizado em uma classe denominada Cliente, tendo como funcionalidades realizar a comunicação, troca de mensagens, recebimento e interpretação do retorno obtido do servidor.

Requisitos para execução do código Cliente

Como requisitos para a execução adequada do código recomenda-se que sejam utilizados os seguintes recursos com suas respectivas versões. Como título de esclarecimento, os requisitos apresentados em sequência referem-se aos que foram utilizados durante o desenvolvimento do projeto, caso deseje utilizar versões mais recentes é recomendado que testes sejam realizados previamente para garantir que as funcionalidades utilizadas ainda são suportadas.

- Python – Versão 3.8.10
- Biblioteca Python PyBluez – Versão 0.23

A classe Cliente

Conforme comentado, a classe Cliente é o componente principal disponibilizado como padrão para realizar a comunicação com o servidor responsável por gerenciar o robô. Em seguida a sua estrutura básica é apresentada para possibilitar que ela seja utilizada de forma adequada.

Atributos

- addr: Uma String que representa o endereço do módulo Bluetooth HC-05 utilizado no robô para efetuar a comunicação.
- Porta: Número inteiro que identifica a porta do computador que será utilizada para realizar a comunicação com o servidor.
- socket: Objeto do tipo BluetoothSocket (disponibilizado através da biblioteca PyBluez) configurado para usar o protocolo de comunicação Bluetooth RFCOMM. Esse objeto é utilizado para realizar chamadas aos métodos responsáveis pela troca de mensagem

propriamente dita entre os dispositivos Bluetooth do dispositivo cliente e do dispositivo servidor.

Métodos

- `__init__(self, addr, porta)`: Inicializa um objeto da Classe cliente configurado os atributos `addr` e `porta` conforme os valores passados por parâmetro, e crie uma instância do objeto `BluetoothSocket` para utilizar durante a comunicação.
- `conectar(self)`: Tenta conectar o socket Bluetooth ao dispositivo cujo endereço corresponde ao armazenado no atributo `addr`. Em caso de falha, um erro indicando que não foi possível estabelecer a comunicação é disparado.
- `desconectar(self)`: Envia o comando específico para encerrar a comunicação entre cliente e servidor, fechando o socket utilizado para o envio de mensagens após conclusão.
- `enviar_mensagem(self, mensagem)`: Envia uma String recebida pelo parâmetro `mensagem` para o dispositivo conectado através do método `conectar`. Atenção, é necessário que exista uma conexão ativa para que esse método funcione. Utilize esse método para enviar os comandos para o servidor definidos nessa documentação em seção posterior.
- `get_resposta(self)`: Retorna a saída recebida do servidor após a execução de um comando. Caso nenhum comando tenha sido enviado previamente, o cliente ficará aguardando uma resposta, uma vez que a comunicação com o servidor é configurada como síncrona.

Atenção, esse método já interpreta o retorno dado pelo servidor, observado o fato de que os dados transmitidos por ele são enviados em formato binário. O retorno sempre será um dado float.

Atenção, caso o servidor retorne algum valor indicando um possível erro pré-definido pelo projeto, o método atual irá interpretar e disparar um erro interrompendo a comunicação com o servidor.

- `testar_tempo_conexao(self, mensagem)`: Envia uma String para o servidor e calcula o tempo aproximado entre o envio e o recebimento do retorno. Utilize esse método para verificar o tempo de resposta aproximando de um comando enviado ao servidor. Atenção, esse método requer que já exista uma conexão com o servidor.

Classes de Erro

As classes de erros são complementos para a classe `Cliente`, sendo utilizadas apenas para disparar chamadas de erros da linguagem Python quando o servidor retornar valores que notificam que o comando enviado pelo cliente não foi executado conforme o planejado, ou que não foi possível estabelecer uma conexão entre os dispositivos de comunicação.

Essas classes são disponibilizadas em um arquivo próprio que é importado na definição da classe `Cliente`. Cada classe conta apenas com um método de inicialização e um método `__str__()` que é utilizado para exibir a mensagem de erro quando a classe for disparada. Caso deseje saber sobre os possíveis erros e as mensagens exibidas por eles, verifique a seção posterior da documentação que trata sobre o tópico.

Detalhes do Código Servidor Disponibilizado

Sobre

O código do servidor foi desenvolvido para ser executando em um Arduino Uno e controlar o robô através do recebimento de comandos a partir de um cliente. O código foi desenvolvido utilizando a própria linguagem de programação do Arduino Uno e organizado de maneira estruturada, contando com diversas funções que são executadas para controlar o robô.

Requisitos para execução do código Servidor

Como requisitos para a execução adequada do código recomenda-se que sejam utilizados os seguintes recursos com suas respectivas versões. Como título de esclarecimento, os requisitos apresentados em sequência referem-se aos que foram utilizados durante o desenvolvimento do projeto, caso deseje utilizar versões mais recentes é recomendado que testes sejam realizados previamente para garantir que as funcionalidades utilizadas ainda são suportadas.

- Arduino IDE – Versão 1.8.19
- Biblioteca SoftwareSerial
- Biblioteca Wire

Configure primeiro para executar ações depois

O servidor é dividido em dois ciclos de execução, podendo eles serem classificados como ciclo de configuração e ciclo de ação. Cada ciclo permite a execução de comandos específicos (conforme definição e listagem apresentada em seção própria). Toda vez que o servidor for iniciado, ele executará apenas comandos de configuração. Após a configuração, o servidor permite a execução dos comandos de ação que em sua maioria interagem propriamente com os componentes eletrônicos do robô.

Estruturação em Funções

Cada comando aceito pelo servidor dispara a execução de uma função. Cada função desenvolvida é independente uma da outra, mesmo que certos elementos de código poderiam ser reaproveitados. Isso ocorre devido ao fato que cada função deve ser simples e protegida, não permitindo que mudanças em outras funções interfiram em sua operacionalidade. Uma observação importante para citar é que cada função é responsável por realizar a leitura de eventuais parâmetros que são enviados junto com um comando, bem como pela validação dos dados recebidos caso necessário. Esse aspecto é relevante, pois em caso alguma função seja alterada ou adicionada, deve-se considerar esse padrão.

Interpretação de Comandos

A interpretação dos comandos advindos do cliente é simples e direta. O servidor realiza a leitura de uma quantidade específica de bytes em cada ciclo que corresponde ao comando enviado pelo cliente. Durante a etapa de configuração espera-se a leitura de um único byte, sendo ele algum comando de configuração conforme a listagem presente na documentação. Durante a etapa de ação, espera-se a leitura de dois bytes para identificar um comando de ação.

Conforme será observado futuramente, cada comando definido possui uma representação em forma de String, sendo ela usada pelo cliente para enviar o comando ao servidor. Todavia, internamente durante a execução do servidor todo comando é dividido em bytes que são tratados através da sua representação decimal. Por exemplo, considere que o cliente enviou ao servidor o comando “S”. Internamente esse comando é tratado pela sua representação decimal conforme tabela ASCII, isto é, através do número 83.

Obs. Comandos que são compostos por dois bytes são interpretados em duas etapas, sendo cada byte interpretado por vez. Sendo assim é possível observar grandes estruturas condicionais que utilizam os comandos switch/case para analisar cada byte. Visando desempenho, comandos que compartilham o mesmo byte inicial são agrupados em uma única estrutura condicional para interpretação do próximo byte. Dessa forma, em caso de alteração no código do servidor para adição de comandos seguindo o padrão constituído, observe essa organização.

Variáveis de Pinagem

Sobre

Variáveis de pinagens são variáveis declaradas dentro do servidor para gerenciar e armazenar a informação dos pinos usados para cada tipo de componente em específico suportado pelo robô. Sendo assim, cada variável refere-se a um pino físico do Arduino quando configurado e sempre será usada para executar comandos que necessitem dos pinos.

Dessa forma, por exemplo, ao configurar a variável PIN_PBT com o valor 5 através dos comandos de configuração, ao executar o comando para leitura do botão esse pino será verificado. Portanto ressalta-se que o relacionamento entre os pinos e as variáveis de pinagem deve seguir a organização do circuito físico.

Valor Padrão

Todas variáveis de pinagem são iniciadas com o valor padrão de 255. Esse valor é utilizado dentro do sistema para realizar verificações se os devidos pinos necessários para a execução de uma ação foram configurados previamente. Caso a configuração não tenha sido feita, o servidor aborta a execução e retorna um erro conforme discutido posteriormente.

Configuração e atribuição dos valores às variáveis de pinagem

As variáveis de pinagem somente podem ser configuradas na etapa de configuração inicial, sendo impossível reconfigurar os valores após essa etapa sem que o servidor seja reiniciado. Ressalta-se que durante a configuração a atribuição dos valores é livre e pode ser feita conforme necessidade do usuário e conforme o circuito físico que foi construído.

Lista de Variáveis de Pinagem

Abaixo segue uma listagem apresentado o nome da variável de pinagem, bem como o seu propósito. Essas informações são úteis para compreender a descrição dos comandos aceitos pelo servidor, bem como para eventualmente mudanças no código do servidor.

Nome Variável	Armazena
PIN_M1	Pino do Arduino Uno usado para conectar e controlar a entrada IN1 da Ponte H. Serve para controlar o Motor A
PIN_M2	Pino do Arduino Uno usado para conectar e controlar a entrada IN2 da Ponte H. Serve para controlar o Motor A
PIN_M3	Pino do Arduino Uno usado para conectar e

	controlar a entrada IN3 da Ponte H. Serve para controlar o Motor B
PIN_M4	Pino do Arduino Uno usado para conectar e controlar a entrada IN4 da Ponte H. Serve para controlar o Motor B
PIN_ECHO	Pino do Arduino Uno usado para conectar e controlar o pino Echo do sensor ultrassônico HC-SR04.
PIN_TRIG	Pino do Arduino Uno usado para conectar e controlar o pino Trig do sensor ultrassônico HC-SR04.
PIN_LED1	Pino do Arduino Uno usado para conectar e controlar o Led simples identificado numericamente pelo número 1.
PIN_LED2	Pino do Arduino Uno usado para conectar e controlar o Led simples identificado numericamente pelo número 2.
PIN_LEDR	Pino do Arduino Uno usado para conectar e controlar o pino responsável pela taxa de vermelho no Led RGB.
PIN_LEDG	Pino do Arduino Uno usado para conectar e controlar o pino responsável pela taxa de verde no Led RGB.
PIN_LEDB	Pino do Arduino Uno usado para conectar e controlar o pino responsável pela taxa de azul no Led RGB.
PIN_TONE	Pino do Arduino Uno usado para conectar e controlar o pino em que o Buzzer está conectado.
PIN_FR1	Pino do Arduino Uno usado para conectar e controlar o sensor fototresistor identificado numericamente pelo número 1.
PIN_FR2	Pino do Arduino Uno usado para conectar e controlar o sensor fototresistor identificado numericamente pelo número 2.
PIN_OR1	Pino do Arduino Uno usado para conectar e controlar o sensor óptico reflexivo identificado numericamente pelo número 1.
PIN_OR2	Pino do Arduino Uno usado para conectar e controlar o sensor óptico reflexivo identificado numericamente pelo número 2.
PIN_OR3	Pino do Arduino Uno usado para conectar e controlar o sensor óptico reflexivo identificado numericamente pelo número 3.

PIN_PTC	Pino do Arduino Uno usado para conectar e controlar o pino em que o Potenciometro está conectado.
PIN_PBT	Pino do Arduino Uno usado para conectar e controlar o pino em que o Push Buton está conectado.
PIN_MPU1	Pino do Arduino Uno usado para conectar e controlar o pino SCL do sensor MPU-6050 conectado.
PIN_MPU2	Pino do Arduino Uno usado para conectar e controlar o pino SDA do sensor MPU-6050 conectado.

Mapeamento de Pinos do Arduino para valores de configuração das variáveis de pinagem

Conforme fora citado, as variáveis de pinagem armazenam os pinos utilizados pelos componentes eletrônicos acoplados ao robô. Por padrão utiliza-se o Arduino Uno como placa base para operar esses componentes, sendo assim tem-se disponível 14 pinos digitais e 6 pinos analógicos para uso.

Para identificar um pino físico durante as etapas de configuração do servidor, basta seguir a numeração própria do Arduino Uno. Os valores entre 0 e 13 são utilizados para configurar os pinos digitais. Já os pinos analógicos podem ser identificados com valores de 14 a 19, sendo o número catorze equivalente ao pino A0 e o número dezanove ao pino A6.

Dessa forma, por exemplo, caso queira configurar o pino 5 para usar juntamente a um Led simples, basta usar o comando de configuração próprio e passar o valor 5. Caso queira configurar o potenciômetro para usar o pino analógico A1, basta utilizar o comando de configuração própria e informar o valor 15.

Nota de observação: Conforme foi comentando na seção que trata sobre a comunicação entre os dispositivos que executam o papel de cliente e servidor, os pinos 10 e 11 são reservados para serem usados para operar o módulo Bluetooth e não devem ser usados para configurar nenhuma outra variável de pinagem.

Comandos

Sobre

Os comandos são utilizados para desencadear ações no servidor, sendo que cada comando possui uma funcionalidade e características próprias. Todavia, é possível agrupar todos os comandos em dois grandes grupos: Os comandos de configuração, e os comandos de ação. O primeiro refere-se àqueles que podem ser executados durante a etapa de configuração inicial do servidor. Já o segundo refere-se àqueles que são executados após a etapa de configuração e desencadeiam ações propriamente ditas, como o acionamento de componentes eletrônicos, e o envio de dados advindos de sensores ou variáveis do servidor.

Lista de Comandos

Em sequência são apresentados os comandos aceitos pelo servidor. Para facilitar a organização da listagem os comandos foram separados conforme classificação apresentada anteriormente. Além disso as informações estão dispostas em uma tabela padronizada conforme modelo abaixo.

Identificação	É a representação em ASCII do comando. Refere-se ao que deve ser enviado ao servidor para que a ação seja executada.
Nome Função	Maioria dos comandos quando interpretados pelo servidor realizam uma chamada a uma função específica. Esse campo da tabela serve para referenciar essa função.
Descrição	Breve resumo sobre a finalidade do comando.
Parâmetros	Lista os parâmetros que devem ser informados ao enviar o comando para o servidor. Além de apresentar para o que cada parâmetro serve, informações do tipo de dado, bem como os intervalos de dados aceitos são informados.
Retorno	Lista os possíveis retornos do comando. Além de apresentar uma breve explicação do significado do retorno, informações do tipo de dado, bem como os intervalos em que os dados estão são informados.
Erros Disparados	Lista os erros que podem ser enviados pelo servidor caso algum problema ocorra durante a execução do comando. Uma listagem descrevendo cada erro pode ser encontrada em uma seção específica.

Obs. O símbolo “---” é utilizado na listagem de comandos para informar que tal comando em específico não possui determinada característica. Por exemplo, quando um comando não possuir parâmetros o símbolo é apresentado no local determinado para essa informação na tabela.

Comandos de Configuração

Identificação	*
Nome Função	--- Comando não está vinculado a execução de uma função no servidor.
Descrição	<p>Comando utilizado para finalizar a etapa de configuração inicial. Ao ser enviado ao servidor, faz com que o programa avance para a etapa de ação, aceitando os comandos de ação.</p> <p>Caso alguma configuração foi efetuada incorretamente, e esse comando já foi executado, a única maneira de retornar a fase de configuração inicial é resetando o servidor.</p>
Parâmetros	---
Retorno	1 – Configuração inicial concluída
Erros Disparados	---

Identificação	R
Nome Função	resetarPinos
Descrição	Configura todas as variáveis de pinagem do servidor para o valor padrão de 255. Utilize esse comando quando deseja iniciar a configuração inicial a partir do zero novamente.
Parâmetros	---
Retorno	1 – Reset de pinos concluído
Erros Disparados	---

Identificação	V
Nome Função	verificarPinoConfigurado
Descrição	<p>Dado uma variável de pinagem, retorna qual o pino no Arduino que está configurado para essa variável.</p> <p>Por exemplo, caso queira saber qual pino do Arduino está sendo utilizado pela variável de pinagem PIN_M1, envie o comando V/0.</p>
Parâmetros	<p>N – Número inteiro que deve estar no intervalo fechado [0, 20]. Cada número desse intervalo identifica uma variável de pinagem. A numeração segue a definição:</p> <p>0 - PIN_M1 1 - PIN_M2 2 - PIN_M3 3 - PIN_M4 4 - PIN_ECHO 5 - PIN_TRIG 6 - PIN_LED1 7 - PIN_LED2</p>

	8 - PIN_LEDR 9 - PIN_LEDG 10 - PIN_LEDB 11 - PIN_TONE 12 - PIN_FR1 13 - PIN_FR2 14 - PIN_OR1 15 - PIN_OR2 16 - PIN_OR3 17 - PIN_PTC 18 - PIN_PBT 19 - PIN_MPU1 20 - PIN_MPU2
Retorno	[0,19] – Pino do Arduíno no qual a variável de pinagem está configurada. 255 – Variável de pinagem não configurada.
Erros Disparados	ERRO 1

Identificação	M
Nome Função	configurarMotores
Descrição	Comando utilizado para definir os pinos que devem ser utilizado para operar os motores. Caso um erro seja disparado, os valores dos pinos relacionados aos motores são reconfigurados para o valor padrão de 255.
Parâmetros	N1 - Número inteiro que deve estar no intervalo fechado [0, 19]. Identifica o pino que a variável de pinagem PIN_M1 será configurada. N2 - Número inteiro que deve estar no intervalo fechado [0, 19]. Identifica o pino que a variável de pinagem PIN_M2 será configurada. N3 - Número inteiro que deve estar no intervalo fechado [0, 19]. Identifica o pino que a variável de pinagem PIN_M3 será configurada. N4 - Número inteiro que deve estar no intervalo fechado [0, 19]. Identifica o pino que a variável de pinagem PIN_M4 será configurada.
Retorno	1 - Configuração finalizada
Erros Disparados	ERRO 1

Identificação	U
----------------------	---

Nome Função	configurarSensorUltrassonico
Descrição	<p>Comando utilizado para definir os pinos que devem ser utilizado para operar o sensor ultrassônico.</p> <p>Caso um erro seja disparado, os valores dos pinos relacionados ao sensor ultrassônico são reconfigurados para o valor padrão de 255.</p>
Parâmetros	<p>N1 - Número inteiro que deve estar no intervalo fechado [0, 19]. Identifica o pino que a variável de pinagem PIN_ECHO será configurada.</p> <p>N2 - Número inteiro que deve estar no intervalo fechado [0, 19]. Identifica o pino que a variável de pinagem PIN_TRIG será configurada.</p>
Retorno	1 - Configuração finalizada
Erros Disparados	ERRO 1

Identificação	S
Nome Função	configurarLedSimples
Descrição	<p>Comando utilizado para definir um led simples, e o pinos que devem ser utilizado para operar ele.</p> <p>Caso um erro seja disparado, o valor do pino relacionado ao led simples permanece configurado para o valor padrão de 255.</p>
Parâmetros	<p>N1 - Número inteiro que deve estar no intervalo fechado [1, 2]. Identifica qual a variável de pinagem para led simples será configurada. O valor N=1 identifica PIN_LED1, e o valor N=2 identifica PIN_LED2.</p> <p>N2 - Número inteiro que deve estar no intervalo fechado [0, 19]. Identifica o pino que a variável de pinagem do led indicado pelo parâmetro N será configurada.</p>
Retorno	1 - Configuração finalizada
Erros Disparados	ERRO 1

Identificação	L
Nome Função	configurarLedRGB
Descrição	<p>Comando utilizado para definir os pinos que devem ser utilizado para operar o Led RGB.</p> <p>Caso um erro seja disparado, os valores dos pinos relacionados ao Led RGB são reconfigurados para o valor padrão de 255.</p>
Parâmetros	N1 - Número inteiro que deve estar no intervalo fechado [0, 19]. Identifica o pino que a variável de pinagem PIN_LED1 será configurada.

	<p>N2 - Número inteiro que deve estar no intervalo fechado [0, 19]. Identifica o pino que a variável de pinagem PIN_LEDG será configurada.</p> <p>N3 - Número inteiro que deve estar no intervalo fechado [0, 19]. Identifica o pino que a variável de pinagem PIN_LEDB será configurada.</p>
Retorno	1 - Configuração finalizada
Erros Disparados	ERRO 1

Identificação	Z
Nome Função	configurarBuzzer
Descrição	<p>Comando utilizado para definir o pino que devem ser utilizado para operar o Buzzer.</p> <p>Caso um erro seja disparado, o valor do pino relacionados ao Buzzer é reconfigurado para o valor padrão de 255.</p>
Parâmetros	N - Número inteiro que deve estar no intervalo fechado [0, 19]. Identifica o pino que a variável de pinagem PIN_TONE será configurada.
Retorno	1 - Configuração finalizada
Erros Disparados	ERRO 1

Identificação	F
Nome Função	configurarSensorFotoresistor
Descrição	<p>Comando utilizado para definir um sensor fotoresistor, e o pino que deve ser utilizado para operar ele.</p> <p>Caso um erro seja disparado, o valor do pino relacionado ao sensor fotoresistor permanece configurado para o valor padrão de 255.</p>
Parâmetros	<p>N1 - Número inteiro que deve estar no intervalo fechado [1, 2]. Identifica qual a variável de pinagem para sensor fotoresistor será configurada. O valor N=1 identifica PIN_FR1, e o valor N=2 identifica PIN_FR2.</p> <p>N2 - Número inteiro que deve estar no intervalo fechado [0, 19]. Identifica o pino que a variável de pinagem do sensor fotoresistor indicado pelo parâmetro N será configurada.</p>
Retorno	1 - Configuração finalizada
Erros Disparados	ERRO 1

Identificação	O
----------------------	---

Nome Função	configurarSensorOpticoReflexivo
Descrição	<p>Comando utilizado para definir um sensor óptico reflexivo, e o pino que deve ser utilizado para operar ele.</p> <p>Caso um erro seja disparado, o valor do pino relacionado ao sensor sensor óptico reflexivo permanece configurado para o valor padrão de 255.</p>
Parâmetros	<p>N1 - Número inteiro que deve estar no intervalo fechado [1, 3]. Identifica qual a variável de pinagem para sensor sensor óptico reflexivo será configurada. O valor N=1 identifica PIN_OR1, o valor N=2 identifica PIN_OR2, e o valor N=3 identifica PIN_OR3.</p> <p>N2 - Número inteiro que deve estar no intervalo fechado [0, 19]. Identifica o pino que a variável de pinagem do sensor sensor óptico reflexivo indicado pelo parâmetro N será configurada.</p>
Retorno	1 - Configuração finalizada
Erros Disparados	ERRO 1

Identificação	P
Nome Função	configurarPotenciometro
Descrição	<p>Comando utilizado para definir o pino que devem ser utilizado para operar o Potenciômetro.</p> <p>Caso um erro seja disparado, o valor do pino relacionados ao Potenciômetro é reconfigurado para o valor padrão de 255.</p>
Parâmetros	N - Número inteiro que deve estar no intervalo fechado [0, 19]. Identifica o pino que a variável de pinagem PIN_PTC será configurada.
Retorno	1 - Configuração finalizada
Erros Disparados	ERRO 1

Identificação	B
Nome Função	configurarPushButton
Descrição	<p>Comando utilizado para definir o pino que devem ser utilizado para operar o Push Buton.</p> <p>Caso um erro seja disparado, o valor do pino relacionados ao Push Buton é reconfigurado para o valor padrão de 255.</p>
Parâmetros	N - Número inteiro que deve estar no intervalo fechado [0, 19]. Identifica o pino que a variável de pinagem PIN_PBT será configurada.
Retorno	1 - Configuração finalizada

Erros Disparados	ERRO 1
-------------------------	--------

Identificação	X
Nome Função	configurarMPU
Descrição	Configura as variáveis de pinagem utilizadas para operar o sensor MPU-6050. As variáveis são configuradas da seguinte forma: PIN_MPU1 = 18, e PIN_MPU2 = 19. Por definição, as configurações do sensor são mantidas com valores padrão. Em caso de dúvidas, consulte o manual de montagem.
Parâmetros	---
Retorno	1 – Configuração finalizada
Erros Disparados	---

Comandos de Ação

Identificação	VG
Nome Função	getVelocidade
Descrição	Comando utilizado para recuperar o valor da velocidade configurada no servidor. A velocidade é um valor utilizado em comandos que controlam os motores. Por padrão, caso a velocidade não tenha sido configurada pelo cliente, o seu valor é igual a 0.
Parâmetros	---
Retorno	N - Número inteiro que pertencente ao intervalo fechado [0, 255]. Identifica a velocidade utilizada por padrão em comandos que acionam os motores.
Erros Disparados	---

Identificação	VS
Nome Função	setVelocidade
Descrição	Comando utilizado para configurar uma nova velocidade passada por parâmetro ao servidor, conforme especificado abaixo.
Parâmetros	N - Número inteiro que deve estar no intervalo fechado [0, 255]. Identifica a nova velocidade que será configurada.
Retorno	1 – Configuração finalizada
Erros Disparados	ERRO 1

Identificação	MF
Nome Função	andarParaFrente
Descrição	Aciona os dois motores descritos no manual de montagem utilizando a velocidade configurada no servidor previamente, fazendo com que o robô avance.
Parâmetros	---
Retorno	1 – Função acionada
Erros Disparados	ERRO 0

Identificação	MT
Nome Função	andarParaTras
Descrição	Aciona os dois motores descritos no manual de montagem utilizando a velocidade configurada no servidor previamente, fazendo com que o robô retroceda.
Parâmetros	---
Retorno	1 – Função acionada
Erros Disparados	ERRO 0

Identificação	MD
Nome Função	rotacionarDireita
Descrição	Aciona o motor da direita utilizando a velocidade configurada no servidor previamente, fazendo com que o robô gire em sentido anti-horário.
Parâmetros	---
Retorno	1 – Função acionada
Erros Disparados	ERRO 0

Identificação	ME
Nome Função	rotacionarEsquerda
Descrição	Aciona o motor da esquerda utilizando a velocidade configurada no servidor previamente, fazendo com que o robô gire em sentido horário.
Parâmetros	---
Retorno	1 – Função acionada
Erros Disparados	ERRO 0

Identificação	MC
----------------------	----

Nome Função	fazerCurva
Descrição	Aciona os dois motores, porém com velocidades diferentes (conforme esperado) definidas via parâmetro. Para fazer uma curva para a esquerda, basta que velocidade de PIN_M3 seja menor que PIN_M1, e vice-versa.
Parâmetros	N1 - Número inteiro que pertencente ao intervalo fechado [0, 255]. Identifica a velocidade utilizada para acionar o pino configurado em PIN_M1 N2 - Número inteiro que pertencente ao intervalo fechado [0, 255]. Identifica a velocidade utilizada para acionar o pino configurado em PIN_M3
Retorno	1 – Função acionada
Erros Disparados	ERRO 0, ERRO 1

Identificação	MP
Nome Função	pararMotores
Descrição	Usado para desativar todos os motores, fazendo com que o robô pare de se locomover.
Parâmetros	---
Retorno	1 – Função acionada
Erros Disparados	ERRO 0

Identificação	SA
Nome Função	ativarLed
Descrição	Dado um identificador de led, aciona o led especificado. Obs. Um identificador de led é apenas um número inteiro usado para representar diferentes leds aceitos pelo sistema. Cabe ao cliente definir via comando de configuração qual led está em qual pino do Arduino.
Parâmetros	N - Número inteiro que pertencente ao intervalo fechado [1, 2]. Identifica qual a variável de pinagem para led será considerada para realizar a ação. O valor N=1 identifica PIN_LED1, o valor N=2 identifica PIN_LED2.
Retorno	1 – Função acionada
Erros Disparados	ERRO 0, ERRO 1

Identificação	ST
Nome Função	ativarLedDelay

Descrição	Dado um identificador de led, aciona o led especificado, sendo que ele fica ativado por um tempo determinado via parâmetro. Após o tempo de delay, o led em questão é desativado.
Parâmetros	N1 - Número inteiro que pertencente ao intervalo fechado [1, 2]. Identifica qual a variável de pinagem para led será considerada para realizar a ação. O valor N=1 identifica PIN_LED1, o valor N=2 identifica PIN_LED2. N2 – Número inteiro positivo que representa o tempo do delay em milissegundos.
Retorno	1 – Função acionada
Erros Disparados	ERRO 0, ERRO 1

Identificação	SI
Nome Função	inverterLed
Descrição	Dado um identificador de led, altera o seu estado, isto é, se está ativado passa a estar desativado e vice-versa.
Parâmetros	N - Número inteiro que pertencente ao intervalo fechado [1, 2]. Identifica qual a variável de pinagem para led será considerada para realizar a ação. O valor N=1 identifica PIN_LED1, o valor N=2 identifica PIN_LED2.
Retorno	1 – Função acionada
Erros Disparados	ERRO 0, ERRO 1

Identificação	SD
Nome Função	desativarLed
Descrição	Dado um identificador de led, ele é desativado. Obs. Não há problema em executar esse comando caso o led já esteja desativado.
Parâmetros	N - Número inteiro que pertencente ao intervalo fechado [1, 2]. Identifica qual a variável de pinagem para led será considerada para realizar a ação. O valor N=1 identifica PIN_LED1, o valor N=2 identifica PIN_LED2.
Retorno	1 – Função acionada
Erros Disparados	ERRO 0, ERRO 1

Identificação	LA
Nome Função	ativarLedRGB
Descrição	Ativa o led RGB utilizado os parâmetros informados via comando

	para formular a sua cor.
Parâmetros	<p>N1 - Número inteiro que pertencente ao intervalo fechado [0, 255]. Identifica o valor que será usado para a taxa de vermelho do led RGB.</p> <p>N2 - Número inteiro que pertencente ao intervalo fechado [0, 255]. Identifica o valor que será usado para a taxa de verde do led RGB.</p> <p>N3 - Número inteiro que pertencente ao intervalo fechado [0, 255]. Identifica o valor que será usado para a taxa de azul do led RGB.</p>
Retorno	1 – Função acionada
Erros Disparados	ERRO 0, ERRO 1

Identificação	LT
Nome Função	ativarLedRGBDelay
Descrição	Ativa o led RGB utilizado os parâmetros informados via comando para formular a sua cor. Diferencia-se do comando anterior, pois obriga a informação de um tempo de delay, sendo que após esse tempo o led RGB é desativado.
Parâmetros	<p>N1 - Número inteiro que pertencente ao intervalo fechado [0, 255]. Identifica o valor que será usado para a taxa de vermelho do led RGB.</p> <p>N2 - Número inteiro que pertencente ao intervalo fechado [0, 255]. Identifica o valor que será usado para a taxa de verde do led RGB.</p> <p>N3 - Número inteiro que pertencente ao intervalo fechado [0, 255]. Identifica o valor que será usado para a taxa de azul do led RGB.</p> <p>N4 - Número inteiro positivo que representa o tempo do delay em milissegundos.</p>
Retorno	1 – Função acionada
Erros Disparados	ERRO 0, ERRO 1

Identificação	LD
Nome Função	desativarLedRGB
Descrição	Desativa o led RGB configurando todos os seus pinos para o valor 0.
Parâmetros	---
Retorno	1- Função acionada
Erros Disparados	ERRO 0

Identificação	ZA
Nome Função	ativarBuzzer
Descrição	Aciona o buzzer na frequência passada via parâmetro.
Parâmetros	N – Número inteiro positivo que representa a frequência em hz que o buzzer deve ser acionado.
Retorno	1 – Função acionada
Erros Disparados	ERRO 0

Identificação	ZT
Nome Função	ativarBuzzerDelay
Descrição	Aciona o buzzer na frequência passada via parâmetro. O buzzer fica ativo conforme tempo informado via parâmetro, sendo o componente desligado ao término da contagem.
Parâmetros	N1 – Número inteiro positivo que representa a frequência em hz que o buzzer deve ser acionado. N2 - Número inteiro positivo que representa o tempo do delay em milissegundos.
Retorno	1 – Função acionada
Erros Disparados	ERRO 0

Identificação	ZD
Nome Função	desativarBuzzer
Descrição	Desativa o buzzer, fazendo com que esse pare de emitir ondas sonoras.
Parâmetros	---
Retorno	1 – Função acionada
Erros Disparados	ERRO 0

Identificação	RU
Nome Função	lerSensorUltrassonico
Descrição	Permite que seja medido a distância do sensor até algum objeto que tenha refletido a onda ultrassônica emitida e capturada por ele.
Parâmetros	---
Retorno	F – Número float não negativo que identifica a distância em centímetros medida pelo sensor ultrassônico.
Erros Disparados	ERRO 0

Identificação	RF
Nome Função	lerSensorFotoreistor
Descrição	Permite detectar a presença e intensidade de luminosidade sobre o sensor.
Parâmetros	N - Número inteiro que pertencente ao intervalo fechado [1, 2]. Identifica qual a variável de pinagem para sensor fotoreistor será considerada para realizar a ação. O valor N=1 identifica PIN_FR1, o valor N=2 identifica PIN_FR2.
Retorno	N – Número inteiro que pertence ao intervalo fechado [0, 1023]. Identifica a intensidade da luz detectada pelo sensor. Quanto maior for N, menos luz está sendo detectada.
Erros Disparados	ERRO 0, ERRO 1

Identificação	RO
Nome Função	lerSensorOpticoReflexivo
Descrição	Permite detectar se algum objeto refletiu o feixe infravermelho emitido pelo sensor.
Parâmetros	N - Número inteiro que pertencente ao intervalo fechado [1, 3]. Identifica qual a variável de pinagem para sensor óptico reflexivo será considerada para realizar a ação. O valor N=1 identifica PIN_OR1, o valor N=2 identifica PIN_OR2, e o valor N=3 identifica PIN_OR3.
Retorno	N – Número inteiro que pertence ao intervalo fechado [0,1]. Identifica a detecção de uma reflexão do feixe infravermelho emitido pelo componente. O valor N=0 identifica uma reflexão, já o valor N=1 identifica que não houve detecção de reflexão.
Erros Disparados	ERRO 0, ERRO 1

Identificação	RP
Nome Função	lerPotenciometro
Descrição	Permite detectar o quanto o potenciômetro foi acionado através da mudança de tensão.
Parâmetros	---
Retorno	N – Número inteiro que pertence ao intervalo fechado [0, 1023]. Identifica a variação da resistência e da tensão no circuito em que o componente está conectado. Quanto maior for N, maior é a tensão medida.
Erros Disparados	ERRO 0

Identificação	RB
Nome Função	lerBotao
Descrição	Permite detectar se um botão está sendo pressionado ou não no momento de medição.
Parâmetros	---
Retorno	<p>N – Número inteiro que pertence ao intervalo fechado [0, 1]. Identifica se o botão está ou não pressionado. Os valores podem variar a depender da forma em que o circuito foi montado conforme esquema abaixo.</p> <p>Modo Pull-Up: N=0 implica que o botão está pressionado, já N=1 identifica que o botão não está pressionado.</p> <p>Modo Pull-Down: N=1 implica que o botão está pressionado, já N=0 identifica que o botão não está pressionado.</p>
Erros Disparados	ERRO 0

Identificação	RT
Nome Função	lerTemperatura
Descrição	Utiliza o sensor MPU-6050 para retornar a temperatura detectada por esse.
Parâmetros	---
Retorno	F – Número float que pertence ao intervalo fechado [-40, 85]. Identifica a temperatura em graus Celsius medida pelo sensor MPU6050.
Erros Disparados	ERRO 0

Identificação	RA
Nome Função	lerAcelerometroX
Descrição	<p>Utiliza o sensor MPU-6050 para retornar o valor do eixo X detectado pelo acelerômetro do sensor.</p> <p>Obs. Os dados não são retornados “bruto”, isto é, já é realizado a transformação dos dados em medidas mais tangíveis considerando a configuração padrão do sensor.</p>
Parâmetros	---
Retorno	F – Um número float que pertence ao intervalo fechado [-20, 20]. Identifica a aceleração medida em m/s ² no eixo X pelo acelerômetro.
Erros Disparados	ERRO 0

Identificação	RC
Nome Função	lerAcelerometroY
Descrição	Utiliza o sensor MPU-6050 para retornar o valor do eixo Y detectado pelo acelerômetro do sensor. Obs. Os dados não são retornados “bruto”, isto é, já é realizado a transformação dos dados em medidas mais tangíveis considerando a configuração padrão do sensor.
Parâmetros	---
Retorno	F – Um número float que pertence ao intervalo fechado [-20, 20]. Identifica a aceleração medida em m/s ² no eixo Y pelo acelerômetro.
Erros Disparados	ERRO 0

Identificação	RE
Nome Função	lerAcelerometroZ
Descrição	Utiliza o sensor MPU-6050 para retornar o valor do eixo Z detectado pelo acelerômetro do sensor. Obs. Os dados não são retornados “bruto”, isto é, já é realizado a transformação dos dados em medidas mais tangíveis considerando a configuração padrão do sensor.
Parâmetros	---
Retorno	F – Um número float que pertence ao intervalo fechado [-20, 20]. Identifica a aceleração medida em m/s ² no eixo Z pelo acelerômetro.
Erros Disparados	ERRO 0

Identificação	RG
Nome Função	lerGiroscopioX
Descrição	Utiliza o sensor MPU-6050 para retornar o valor do eixo X detectado pelo giroscópio do sensor. Obs. Os dados não são retornados “bruto”, isto é, já é realizado a transformação dos dados em medidas mais tangíveis considerando a configuração padrão do sensor.
Parâmetros	---
Retorno	F – Um número float que pertence ao intervalo fechado [-5, 5]. Identifica a velocidade rotacional medida em rad/s no eixo X pelo giroscópio.
Erros Disparados	ERRO 0

Identificação	RI
Nome Função	lerGiroscopioY
Descrição	Utiliza o sensor MPU-6050 para retornar o valor do eixo Y detectado pelo giroscópio do sensor. Obs. Os dados não são retornados “bruto”, isto é, já é realizado a transformação dos dados em medidas mais tangíveis considerando a configuração padrão do sensor.
Parâmetros	---
Retorno	F – Um número float que pertence ao intervalo fechado [-5, 5]. Identifica a velocidade rotacional medida em rad/s no eixo Y pelo giroscópio.
Erros Disparados	ERRO 0

Identificação	RK
Nome Função	lerGiroscopioZ
Descrição	Utiliza o sensor MPU-6050 para retornar o valor do eixo Z detectado pelo giroscópio do sensor. Obs. Os dados não são retornados “bruto”, isto é, já é realizado a transformação dos dados em medidas mais tangíveis considerando a configuração padrão do sensor.
Parâmetros	---
Retorno	F – Um número float que pertence ao intervalo fechado [-5, 5]. Identifica a velocidade rotacional medida em rad/s no eixo Z pelo giroscópio.
Erros Disparados	ERRO 0

Identificação	ER
Nome Função	encerrarComunicacao
Descrição	Reseta o servidor, consequentemente encerrando todas as comunicações ativas e configurações realizadas. Após o reset, o servidor estará em fase de configuração inicial.
Parâmetros	---
Retorno	---
Erros Disparados	---

Como executar um comando com parâmetros

Como pode ser observado na listagem apresentada anteriormente, certos comandos necessitam de parâmetros para que possam ser executados. Para adicionar parâmetros a um comando basta utilizar a sintaxe abaixo:

COMANDO/PARÂMETRO_1/.../PARÂMETRO_N

Exemplos

SV/125

LT/255/125/125/5000

Observe que as diferentes partes de um comando são separadas pelo caractere de barra. Na verdade, qualquer caractere que não seja alfanumérico pode potencialmente ser utilizado como um separador. De toda forma, recomenda-se o uso da barra como um padrão para fácil leitura e compreensão da informação que está sendo comunicada ao servidor.

Quando um comando resulta em um erro definido, o que ocorre com o servidor?

Quando um comando não pode ser executado corretamente, um erro é enviado como dado de retorno pelo servidor. Dessa forma, cabe ao código do cliente interpretar o valor do retorno e disparar uma notificação de erro. Tomado conhecimento dessa fato, ressalta-se que para o servidor um erro é tratado apenas como um tipo especial de retorno, permitindo assim que a sua execução não seja interrompida e novos comandos possam ser enviados pelo cliente.

Erros

Sobre

Conforme citado em seção anterior, certos comandos podem disparar certos erros como um retorno especial quando a sua execução não pode ser efetuada. Os erros listados nessa seção são extremamente simples e básicos, sendo utilizados para garantir uma segurança mínima durante a execução do servidor e evitar procedimentos exaustivos que comprometem o seu tempo de resposta.

Listagem de Erros

Em sequência são apresentados os erros implementados e que podem estar presentes durante a execução dos códigos do cliente e do servidor. Para facilitar as informações estão dispostas em uma tabela padronizada conforme modelo abaixo.

Identificação	Nomenclatura utiliza para identificar e referenciar o erro na documentação e em trechos de código.
Classe do Erro	Classe disponibilizada junto ao código padrão do cliente que implementa o erro.
Representação Numérica	Valor inteiro utilizado dentro do servidor para representar o erro, observado que todos os dados retornados pelo servidor são numéricos. Esse valor é utilizado quando o cliente necessita interpretar o retorno e verificar se ele se refere a um possível erro.
Mensagem Disparada	Mensagem de erro exibida quando o erro é disparado após interpretação do cliente.
Descrição	Breve resumo do motivo pelo qual o erro foi disparado.
Possível Solução	Listagem de soluções simples que podem ser utilizadas para resolver o erro.

Erros implementados e disponibilizados

Identificação	ERRO 0
Classe do Erro	ErroPinoNaoConfigurado
Representação Numérica	10000000
Mensagem Disparada	O pino não foi definido durante o processo de configuração do robô. Não é possível utilizar essa funcionalidade sem a correta configuração de pinos!
Descrição	Para que o servidor execute certos comandos de ação, é necessário

	<p>que o cliente defina previamente os pinos para aquela ocasião na etapa de configuração inicial do servidor. Caso o pino não seja definido e o cliente tente utilizar uma ação que faça o uso desse pino em questão, o ERRO 0 é disparado.</p> <p>Em caso de dúvidas relacionadas ao processo de configuração inicial do servidor, recomenda-se que a seção NOME_SECAO seja visitada.</p>
Possível Solução	Resetar o servidor e realizar novamente a etapa de configuração inicial, tomando cuidado para definir corretamente os pinos que serão utilizados na hora da execução dos comandos de ação.

Identificação	ERRO 1
Classe do Erro	ErroValorParametroInvalido
Representação Numérica	10000001
Mensagem Disparada	Valor informado como parâmetro para o comando não é aceito.
Descrição	<p>Comandos que aceitam parâmetros possuem uma listagem específica de argumentos que podem ser recebidos, bem como os valores que são esperados. Caso um comando receba parâmetros que não condizem com sua definição, o ERRO 1 é disparado.</p> <p>Em caso de dúvidas relacionadas aos parâmetros aceitos por cada comando do servidor, recomenda-se que a seção NOME_SEÇÃO seja visitada.</p>
Possível Solução	Execute o comando novamente, porém utilize os parâmetros conforme definidos na documentação do comando.

Identificação	ERRO 2
Classe do Erro	ErroComandoInvalido
Representação Numérica	10000002
Mensagem Disparada	Comando Inválido - Envie somente os comandos aceitos pelo servidor.
Descrição	<p>O servidor possui uma lista definida de comandos que podem ser aceitos. Caso seja informado um comando que não foi previamente definido, o ERRO 2 é disparado.</p> <p>Atenção, cada etapa do servidor aceita uma lista de comandos diferente, isto é, os comandos utilizados para configuração inicial não são aceitos durante a etapa de execução dos comandos de ação, e vice-versa.</p> <p>Em caso de dúvidas relacionadas aos comandos aceitos pelo servidor, recomenda-se que a seção NOME_SEÇÃO seja visitada.</p>

Possível Solução	Verifique o comando que originou o erro e busque o comando correto na listagem de comandos aceitos.
-------------------------	---

Identificação	ERRO 3
Classe do Erro	ErroConexaoInexistente
Representação Numérica	Nenhuma – Erro disparado pelo cliente não possui representação numérica, pois não é necessário envio a partir do servidor.
Mensagem Disparada	Cliente não conectado ao Servidor, impossível realizar a transmissão/recepção de dados.
Descrição	<p>O cliente foi disponibilizado para comunicar-se com o servidor. Caso não seja possível estabelecer uma comunicação entre os meios, ou o cliente tente comunicar-se com o servidor sem uma conexão previamente existente, o ERRO 3 é disparado.</p> <p>Considerando o meio de comunicação padrão via bluetooth, erros de comunicação podem ocorrer devido à: Distanciamento entre os dispositivos executando o cliente e o servidor; Servidor já conectado a outro dispositivo/cliente; Endereço e/ou porta para comunicação informado ao cliente estão incorretos;</p>
Possível Solução	<p>Tente identificar o possível motivo da falha de comunicação entre os dispositivos, considerando os pontos apresentados na descrição anterior, e tome devidas proporções para resolver o eventual problema.</p> <p>Reinicie ambos os dispositivos que estão sendo utilizados para realizar a comunicação, e tente realizar uma nova conexão entre cliente e servidor.</p>