**Protocol Design**

*WarePro*

**Company:** MBT

**Date:** 14.05.2024

**Version:** 1.1

# Introduction

Without an effective protocol, communication might not work as expected and as result the robot might have unexpected behavior. This protocol is designed for communication between our C# control application and the master microcontroller. The commands are designed with the intention to create a robust and responsive protocol that handles different situations and errors.

# Protocol Architecture

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| START | COMMAND | DATA | CHECKSUM | END |
| $ | MOVE | AA | 441 | # |

## Explanation:

Between the command and data, and data and checksum we use a separator “ | “ so an example string will be: “$MOVE|A1|425#”.

To indicate the start of a message, we use the “$”, followed by that we have a command that indicates what the robot should do (all commands with explanations can be found in the Possible commands table). After the command we have a separator as mentioned before and after we have the data that goes with the command. After the data we have a separator again and then the checksum, we use a checksum as error handling.

In our case the checksum calculates the ASCII value of the command and data combined and sends it in the protocol message for the other device to compare and check the integrity of the received data.

We use the “#” symbol to indicate the end of the message.

## Possible Commands

|  |  |  |
| --- | --- | --- |
| Action | Explanation | Command |
| Move to location | Instructs the robot to move to location | MOVE |
| Emergency Stop | Stops the robot immediately | STOP |
| Speed data | Sends current speed information | SPEED |
| Location data | Sends current location information | LOC |

# Error Handling

In our communication protocol, a checksum is utilized to ensure data integrity and to detect any errors that might occur during data transmission. The checksum is calculated by summing the ASCII values of each character in the message, excluding the checksum itself and the end delimiter. This calculation includes the command and its associated data, formatted as "$COMMAND|DATA|". After computing the checksum, it is appended to the message followed by the end delimiter `#`, forming the complete message structure: "$COMMAND|DATA|CHECKSUM#".

For example, if the command is "MOVE" and the associated data is "A1", the message for checksum calculation would be "$MOVE|A1|". The checksum is then calculated as the sum of the ASCII values of all characters within this string. This checksum value is then appended to form the final message transmitted over the network. This method allows both the sender and the receiver to compute and verify the checksum independently. If the checksums match on both ends, it confirms that the message has been transmitted without alteration, thus ensuring the reliability of the communication.