SIMPLY COMMAND V1.0.1

This is the documentation file for the library SimplyCommand developed by Paramesh Sriram P S.

This protocol aims to be platform independent and transmission method independent as well. It uses Strings to command Servo motors, and simplifies the process of controlling Servo objects using PWM values.

Download Link

STRUCTURE:

SimplyCommand uses messages (of String type) to communicate information. Given below is the general structure of a message

SET **S:senderName R:receiverName P:PASSWORD** E1:34 E3:45
(1) **(2)** (3)

(1)-Instruction Type
(2)-Authentication
(3)-Instruction

Details:

- **SET:** This informs SimplyCommand that the message is of type SET. Currently, only SET is supported but it will soon be followed by **GET** and **INFORM**.
- **S:senderName:** We define the name of the controller, that is the name of the system sending the commands.
- **R:receiverName:** We define the name of the receiver, that is the name of the microcontroller accepting the commands.
- **P:PASSWORD:** We define the password for the communication.
- **E1:34 E2:45:** We define the values that we want to modify. The name of servo is separated from the value of the motor by a ‘:’ (colons), and the servo names are separated using ‘ ’ (spaces).

ALERT: The messages are space sensitive. Please ensure that any unwanted spacing is removed. If that is not possible, then please use alterParameters() (Defined in the next page) to alter the delimiters.

LINK(GitHub): theBoss12kS/SimplyCommand: An Arduino library designed to control Servos easily (github.com)

The functions, their arguments and their description are given below.

1) void **defineControlScheme**(int numMotors, String motorListName[], int pinList[], String sender, String receiver, String password): This function is used to initialise all the motors.

Example:
String motorList[]={“E1”, “E2”, “E3”}; //Names to be assigned
int pinList[]={1,2,3} //Pins to attach
defineControlScheme(3,motorList,pintList, “senderName”, “receiverName”. “PASSWORD”);

2) int **executeInstruction**(String instruction): This function is used to execute an instruction directly, with authenticity verification. Returns 1 if successfully executed.

Example:
int ret=executeInstruction(“SET S:senderName R:receiverName P:PASSWORD E1:34 E2:45”);

3) void **alterParameter**(char instParam, char motorValSplit): This function is used to change the breakers used to split instructions. It is set by default to ‘ ‘ and ‘:’.

Example:
alterParameter(‘,’ , ‘S’); //Where , will replace ‘ ‘ and S will replace ‘:’.

4) void **setAction**(String actionName, String instruction): This function is used to set an action. An action is a set of instructions that can be given a name. The authenticity is NOT verified since it is called from within the code itself.

Example:
setAction(“MOVE UP”, “E1:45 E2:56”): //MOVE UP is the action name and the other argument is the instruction.

5) void **callAction**(String actionName): This function is used to call the action that was previously set.

Example:
callAction(“MOVE UP”); //Where MOVE UP is the name of the action.

6) void printAllActions(): This function is used to print all the available actions and their corresponding instructions.

Example:
printAllActions(); //This will print all actions onto the Serial monitor.

7) void doAction(String action): This function is used to perform an action directly, without authenticity verification.

Example:
doAction(“E1:34 E2:45”); //Just enter the instruction directly

ALERT: NEVER make this function parameter dynamic, as in received from an external controller. Always use executeInstruction() or callAction() when using external commands.