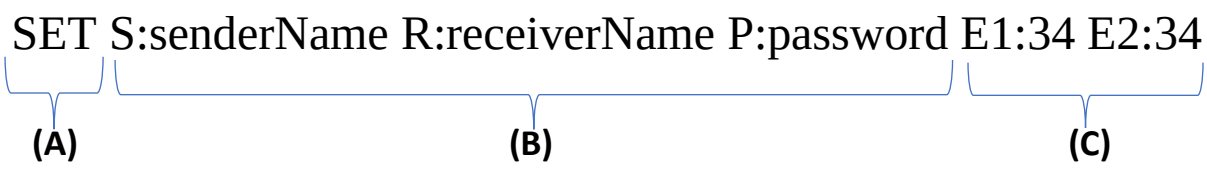
SimplyCommand

DESCRIPTION: This documentation is for a platform and language independent inter-device communication standard known as SimplyCommand. Developed by Paramesh Sriram P S

If you have any questions, comments or feedbacks, please email me at parameshsriram2020@gmail.com

1) MESSAGE: Defined as the set of characters that will carry the instructions. Can be in the form of Strings or character arrays, depending on language.

2) STRUCTURE OF MESSAGE: SimplyCommand uses spaces to separate each parameter and uses semi-colons to separate the parameter from their values.



(A) *TYPE*: This is used to define the type of message. Possible types include SET, GET and INFORM.

(B) *CREDENTIALS*: This piece is used to verify the authenticity of the message. It must be the same for both sender and receiver side.

NOTE: While not necessary, it is recommended that the sender's name and the receiver's name be included in each message. This will help prevent cross-talk when using multiple devices over the same network.

(C) *CONTENT:* This piece is the actual message, containing the desired parameters and their values.

3) TYPES OF MESSAGES: There are primarily three classes of messages, which are SET, GET and INFORM.

NOTE: SimplyCommand is generally asynchronous. However, using INFORM enables us to get a confirmation. The code must be designed to handle this, and the choice between asynchronous and synchronous commands are left to the developer.

a) SET: This is a command that is used to change or update the values of any parameter
   Example: SET S:senderName R:UAV P:pass E1:34 E3:45
b) GET: This is a command that is used to obtain values of a parameter. A GET message MUST have a returning INFORM message.
   Example: GET S:senderName R:UAV P:pass E1 E2
c) INFORM: This is a command that is used to transmit the values of a parameter. It can be upon request using GET, or just transmitted without any request. This is asynchronous by nature.
   Example: INFORM S:UAV R:senderName P:pass E1:34 E2:45

4) HOW TO USE: There are two ways to use SimplyCommand, that is using any of the ready made implementations like those found in the repository, or making a custom implementation.

a) Ready-made implementation: This involves using libraries like the SimplyCommand library to command and control servos using PWM values.

b) Custom implementation: This involves using a set of while loops and 'contains' functions to split the message into components. The general overview of the code is given below, although it will vary between languages.

GENERAL IMPLEMENTATION:

String messageReceived= "SET S:devStation R:Drone P:pass E1:34 E5:56 ";

//First, we split the message into it's components.

String splitMessage[]=messageReceived.split(" ");

int numberOfElements = splitMessage.getCountOfElements();

counter=0;

while(counter<numberOfElements)

{

//Split the component into element and value.
String splitMessageAgain[]=splitMessage[counter].split(":");

//This first split is used to identify the split message type. The sender, password and receiver ID need to be verified before interpreting the command.

```
if(splitMessageAgain[0]== “S”){
        If(senderName=splitMessage[1]){ sender=1;}
}
else if(splitMessageAgain[0]== “R”){
        If(receiverName=splitMessage[1]){ recv=1;}
}
else if(splitMessageAgain[0]== “P”){
        If(password=splitMessage[1]){ pass=1;}
}
//Once the authenticity is verified, the users can proceed with interpreting the message.
If(pass==1 && recv==1 && sender==1){

If(splitMessageAgain[0]== “E1”){
int value= splitMessageAgain[1];
//Do something with the value, such as command a motor.
}
}
counter++;
}
```

NOTE: This code is not written in any particular language; it is purely meant to explain the flow of the program.

Released on 12/01/23

Website: [paramsThoughts – Just the opinions of a tech enthusiast (wordpress.com)]

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