## Device

Το πρώτο πράγμα που πρέπει να ρυθμιστεί πάντα είναι το Device. Ο σκοπός του είναι να μπορεί να είναι μία συσκευή προσομοιώσιμη. Δύο πράγματα πρέπει να ρυθμιστούν:

1. Τα **Signals**, τα οποία τυπικά ορίζονται στον constructor.
2. Διαχείριση του **UpdateDeviceValuesBySerialMessage**. To Device δεν έχει να κάνει με SerialDevice και για αυτό η μέθοδος αυτή είναι public ώστε να κληθεί τυπικά από τον Agent μέσα στη μέθοδο **OnMessageRetrieved** (δες παρακάτω). Το πεδίο message είναι το κείμενο που στέλνει η συσκευή προς τον υπολογιστή, ενώ η παράμετρος messageType είναι η εντολή που έχει προηγηθεί (από τον υπολογιστή προς τη συσκευή). Η μέθοδος τυπικά είναι απαραίτητη όταν η συσκευή έχει στείλει ένα μήνυμα προς τον υπολογιστή χωρίς να έχει προηγηθεί κάποια εντολή προς τη συσκευή.

public class KernScale : Device

{

public KernScale(bool isSimulated = false) : base(isSimulated)

{

signals.Add("Weight", new Signal("Weight"));

Weight.IsSimulated = isSimulated;

}

public Signal Weight { get { return signals["Weight"]; } }

…

. public override void UpdateDeviceValuesBySerialMessage(string message, string messageType = "")

{

message = message.Trim();

if (message.Contains("\_"))

{

Weight.ActualValue = 0.0f;

\_isOverloaded = false;

OnNewWeightReceived();

}

else if (message.Contains("="))

{

Weight.ActualValue = WeightInvalidValue;

\_isOverloaded = true;

OnNewWeightReceived();

…

## Agent (SerialDevice)

Είναι η οντότητα που αναλαμβάνει την επικοινωνία του προγράμματος με τη συσκευή. Είναι πάντα SerialDevice και περιέχει ένα non-simulated Device. Αναγκαίες ρυθμίσεις:

1. O Constructor τυπικά δηλώνει ένα non-simulated Device.
2. Οι εντολές μπορεί να δηλώνονται:
   1. (παλιός τρόπος) ως ασύγχρονες **SendMessage**: σε αυτές απλά στέλνεται η εντολή και επιστρέφεται true όταν υπάρξει οποιοδήποτε πρόβλημα στην αποστολή της εντολής μόνο. Σε αυτήν την περιπτωση θα πρέπει να γίνεται handle το **OnMessageRetrieved** του Agent και τυπικά το **UpdateDeviceValuesBySerialMessage** του Device.
   2. ως σύγχρονες χρησιμοποιώντας αντικείμενα SerialCommand. Σε αυτήν την περίπτωση δεν χρειάζονται τα overrides των OnMessageRetrieved, UpdateDeviceValuesBySerialMessage. Έτσι, μπορούν να ανιχνευτούν με ακρίβεια exceptions κλπ. Η περίπτωση αυτή καλύπτεται αργότερα.

Παράδειγμα 2a

public FlukeAgent(string portName)

: this(new SerialPort(portName, 115200, Parity.None, 8, StopBits.One)) { }

public FlukeAgent(SerialPort port)

: base(port)

{

SendMessageSuffix = "\r";

ReceiveMessageSuffix = "\r";

ReceiveMessage2Suffix = "\r";

Fluke = new Fluke(false);

}

public Fluke Fluke { get; private set; }

public bool Read()

{

return SendMessage("QM");

}

…

protected override void OnMessageReceived()

{

base.OnMessageReceived();

try

{

Fluke.UpdateDeviceValuesBySerialMessage(this.LastSerialMessage.ReceivedFilteredMessage2, this.LastSerialMessage.MessageSent);

}

catch { }

}

## SerialCommand

Κάθε σύγχρονη εντολή έχει νόημα μόνο όταν επιστρέφεται κάποια απόκριση από τη συσκευή. Το update των τιμών του Device γίνεται ακριβώς εδώ εφόσον δεν υπάρχει κάποιο σφάλμα. Μπορεί να είναι ανάμεσα στις παρακάτω περιπτώσεις:

1. **SimpleSerialCommand**: Εντολή που δεν περιμένει κάποιο αποτέλεσμα, εκτός από την επιβεβαίωση ότι για παράδειγμα ότι δεν υπήρξε κάποιο σφάλμα, όπως για παράδειγμα εσφαλμένη εντολή.

public static SimpleSerialCommand AssignPortToMfcCommand(int port, int mfc) =>

new SimpleSerialCommand($"PORT {port} MFC = {mfc}");

…

public async Task<SimpleSerialCommand> AssignPortToMfc(int port, int mfc)

{

//the ReadMfcCount and ReadValidPorts must be called before calling this function

//it is assumed that only a valid port is assigned to the mfc

var response = await AssignPortToMfcCommand(port, mfc).SendAndGetCommand(this);

if (!response.IsError)

GasMixer.MFCs[mfc].CurrentPort = GasMixer.Ports[port];

return response;

}

public async Task<SimpleSerialCommand> Stop()

{

//return await SendMessageAndReadResponse("STOP");

return await GasMixerCommands.Stop().SendAndGetCommand(this);

}

1. **SimpleSerialCommandWithResponse<T>**: Εντολή που επιστρέφει μία ή περισσότερες τιμές. Ανάλογα με τον τύπο των μεταβλητών που επιστρέφονται, χρειάζεται να περνάει το inputPattern (regex) και η συνάρτηση που κάνει parse το αποτέλεσμα. Υπάρχουν έτοιμοι static constructors στο **SerialCommands**, όπως φαίνεται παρακάτω:

public static SimpleSerialCommandWithResponse<int?> ReadMfcCount() =>

SerialCommands.CreateWithIntResponse("NUMBER MFC ?");

//new SimpleSerialCommandWithResponse<int?>("NUMBER MFC ?", intPattern, getInt);

public static SimpleSerialCommandWithResponse<float?> ReadFlowTotalActual() =>

SerialCommands.CreateWithFloatResponse("CONC TOT ACTUAL ?");

//new SimpleSerialCommandWithResponse<float?>("CONC TOT ACTUAL ?", floatPattern, getFloat);

public static SimpleSerialCommandWithResponse<List<float>> ReadConcentrationAllActual() =>

SerialCommands.CreateWithListFloatResponse("CONC ALL ACTUAL ?");

//new SimpleSerialCommandWithResponse<List<float>>("CONC ALL ACTUAL ?", floatListPattern, getFloats);

public static SimpleSerialCommandWithResponse<List<int>> ReadPortAssignments() =>

SerialCommands.CreateWithListIntResponse("PORT ALL MFC ?");

//new SimpleSerialCommandWithResponse<List<int>>("PORT ALL MFC ?", intListPattern, getInts);