**MIKH 2.0 - Dashboard**

*Doc. Code. Dashboard - Com. Track/Pod(MH20)*

**Field Notes**

[PC] → *USB* → [Track] → *RF* → [Pod]

**Référence Trame Port Serie montant**

[PC] → *USB* → [Track]

**List Values**

[Pod]

| **Nom** | **Code** | **Valeur** | **Type** | **Trame** | **Default** |
| --- | --- | --- | --- | --- | --- |
| Battery Level | batteryLevel | 0-99 | int | BB | 00 |
| Battery Level is Critical | CriticalBatteryLevel | 0-1 | bool | C | 0 |
| Motor Speed | motorSpeed | 0-99 | int | MM | 00 |
| Overspeed Motor Mode | overLoadMode | 0-1 | bool | O | 0 |
| Pod Start Lev. Motor | motorActive | 0-1 | bool | L | 0 |
| System Failure | criticalState | 0-1 | bool | F | 0 |
| Abort | abortState | 0-1 | bool | A | 0 |
| Connection to H7 Available | podStatus | 0-1 | bool | U | 0 |
| Connection to H7 Status | podState | 0-1 | bool | P | 0 |

[Track]

| **Nom** | **Code** | **Valeur** | **Type** | **Trame** | **Default** |
| --- | --- | --- | --- | --- | --- |
| Stud deploy State | studState | 0-1 | bool | S | 0 |
| Translation Direction | vfdMode | 0-1 | bool |  | 1 |
| Propulsion Power | vfdActive | 0-1 | bool |  | 0 |
|  |  |  |  |  |  |
| Connection BLE Available | bleStatus | 0-1 | bool | T | 0 |
| Connection BLE Status | bleState | 0-1 | bool |  | 0 |
| Connection to VFD Status | vfdState | 0-1 | bool | V | 0 |
| Connection to Feather Available | trackStatus | 0-1 | bool |  | 0 |
| Connection to Feather Status | trackState | 0-1 | bool | T | 0 |

**Trame**

Serial.Println →

| ***BBCMMOLFAUP*** |
| --- |

**Code C# DataReceived**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO.Ports;

namespace Rx3

{

public partial class Form1 : Form

{

private SerialPort myport;

private DateTime datetime;

private string in\_data;

public Form1()

{

InitializeComponent();

}

private void start\_btn\_Click(object sender, EventArgs e)

{

myport = new SerialPort();

myport.BaudRate = 9600;

myport.PortName = port\_name\_tb.Text;

myport.Parity = Parity.None;

myport.DataBits = 8;

myport.StopBits = StopBits.One;

myport.DataReceived += myport\_DataReceived;

try{

myport.Open();

data\_tb.Text = "";

}

catch (Exception ex) {

MessageBox.Show(ex.Message,"Error");

}

}

void myport\_DataReceived(object sender, SerialDataReceivedEventArgs e)

{

in\_data = myport.ReadLine();

this.Invoke(new EventHandler(displaydata\_event));

}

private void displaydata\_event(object sender, EventArgs e)

{

datetime = DateTime.Now;

string time = datetime.Hour + ":" + datetime.Minute + ":" + datetime.Second;

data\_tb.Text = time + "\t\t\t\t\t" + in\_data;

}

}

}