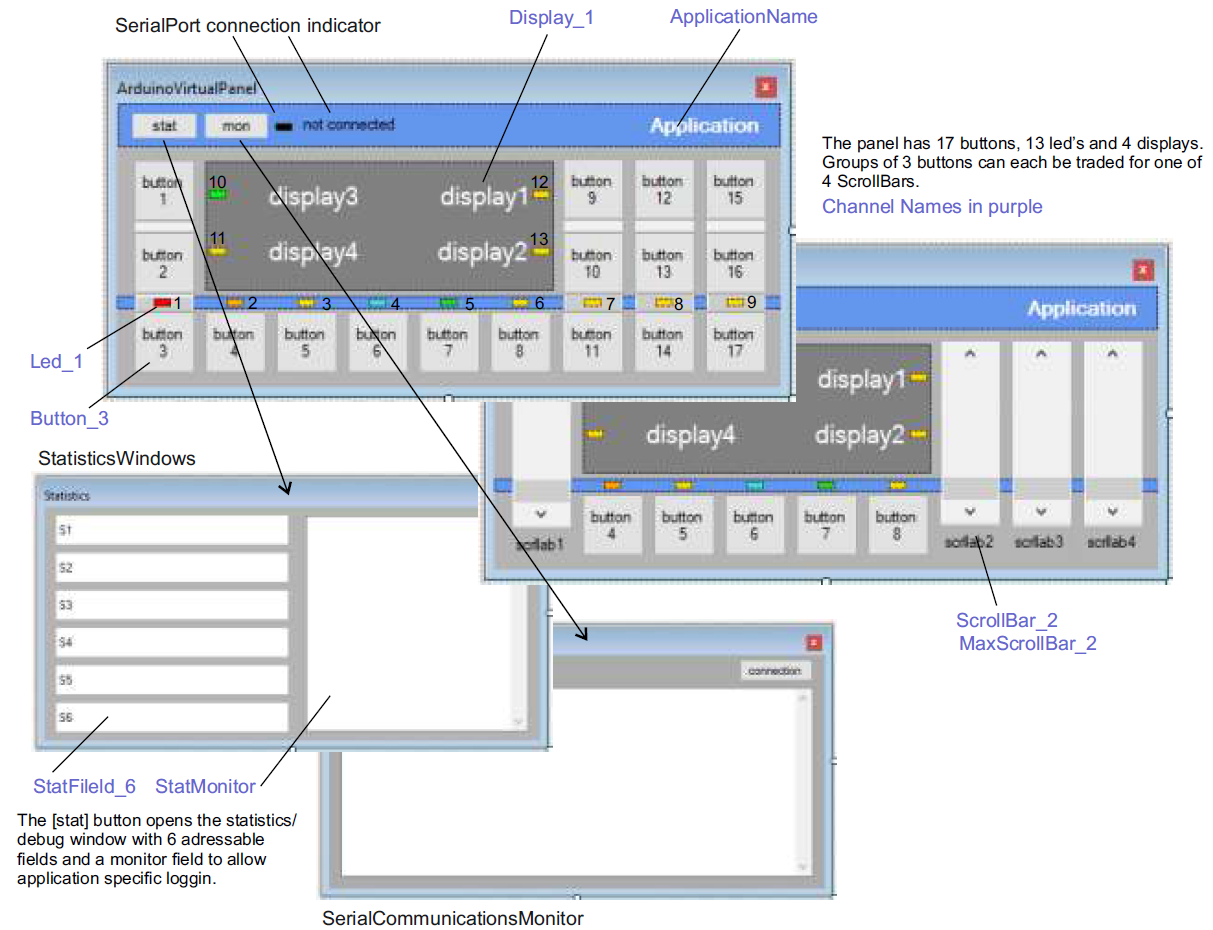
**Panel One Quick Reference**

Arduino Experiment Control Panel

Jaap Daniëlse V0.1 18-7-2018

**Application name**

Panel.Send( ApplicationName, “My Application Name” );

|  |  |
| --- | --- |
| string | Display text as application name |

**Display**

Panel.Send( Display\_x, Value );

Panel.Send( Display\_x, false);

|  |  |
| --- | --- |
| Boolean False | Display invisible |
| Boolean True | Display visible |
| string | Display text |
| byte | Display as text |
| integer | Display as text |
| long | Display as text |
| $NORMAL | Normal font |
| $BOLD | Bold font |
| $BIG | Large font |

**Panel.Sendf**

Panel.Sendf( StatField\_x, “Rec. %d units”, Value );

Using printf formatting.   
See Arduino documentation.

**Led**

Panel.Send( Led\_x, “$RED”);

Panel.Send( Led\_x, false);

|  |  |
| --- | --- |
| Boolean False | Led Invisible |
| $OFF | ▄ |
| $RED | ▄ |
| $GREEN | ▄ |
| $YELLOW | ▄ |
| $ORANGE | ▄ |

**Scrollbar**

Panel.Send( ScrollBar\_x, “text”);

Panel.Send( ScrollBar\_x, IntegerValue );

|  |  |
| --- | --- |
| Boolean False | ScrollBar invisible |
| Boolean True | ScrollBar visible |
| string | ScrollBar label text |
| Integer value | Set scrollbar position |

Receive

|  |  |
| --- | --- |
| Integer | Scrollbar value in panel public var vpr\_int |

**MaxScrollBar**

Panel.Send( MaxScrollBar\_x, IntegerValue);

|  |  |
| --- | --- |
| Integer value | Set scrollbar maximum value |

Default MaxScrollBar = 100

Only positive values. Scale within sketch

**Button**

Panel.Send( Button\_x, “text”);

Panel.Send( Button\_x, “$ONOFF”);

|  |  |
| --- | --- |
| Boolean False | Button Invisible |
| Boolean True | Button Visible |
| Text | Button text |

Send special text

|  |  |
| --- | --- |
| $ONOFF | 🞅● |
| $LEFT | ◀ |
| $RIGHT | ▶ |
| $UP | ▲ |
| $DOWN | ▼ |
| $DOT | ● |
| $LTURN | ⭯ |
| $RTURN | ⭮ |
| $RUN | ▶ |
| $PAUSE | ▌▌ |
| $STOP | ■ |
| $SET | 🞿 |

Receive

|  |  |
| --- | --- |
| void | Button pressed |

**Statistic Field**

Panel.Sendf( StatField\_x, “Rec. %d units”, Value );

|  |  |
| --- | --- |
| string | Display text |
| byte | Display as text |
| integer | Display as text |
| long | Display as text |

**Statistic Monitor**

Panel.Send( StatMonitor, “debug message” );

|  |  |
| --- | --- |
| string | Log message |

**PanelConnected**

Receive only

|  |  |
| --- | --- |
| void | Panel has connected |

*! On receive send panel layout*

**DynamicDisplay**

Panel.Send( DynamicDisplay, true);

|  |  |
| --- | --- |
| Boolean True | Set panel to send req.  every 500 ms. |

**StaticDisplay**

Panel.Send( StaticDisplay, true);

|  |  |
| --- | --- |
| Boolean True | Set panel to send req. when button clicked. |

**MinimalPanel Sketch**

One button, one led and one (auto updated) display

// Minimal Panel

#include "VirtualPanel.h" // Panel Library

Include library and chanel declaration (in .h file as part of your sketch) and create the panel object

#include "PanelOneV01.h" // Panel One Channel declaration

VirtualPanel MyPanel(PanelID, PanelCallback, Serial, 115200);

boolean Power = false; //application power variable

int Value; //application value variable

void setup()

Include Mypanel.Init() in setup()  
to initialize the serial port

{ MyPanel.Init(); // VirtualPanel.Initialize -init panel protocol and serial port

}

void loop()

Include panel.receive in loop() to handle

incoming commands and data

{ MyPanel.Receive(); // VirtualPanel.Receive - handle incoming panel data

if (Power) Value = random(1,1000); else Value=0; //application logic :)

}

void PanelCallback(int channel, int type)

Create the PanelCallback routine to handle commands coming from the panel exe. Callback declaration in PanelOneV01.h

{ // called through Panel.Receive when receiving incoming data

switch (channel)

{

case PanelConnected:

{ // panel layout request

Use PanelConnected to initialize the panel layout; Name, buttons, leds etc.

MyPanel.Send(ApplicationName, "MinimalPanel"); // set the application name

MyPanel.Send(DynamicDisplay, true); // set panel to requests dynamic values

MyPanel.Send(Button\_3, "$ONOFF"); // make button\_3 a power button (power symbol)

MyPanel.Send(Led\_1, "$OFF"); // activate led\_1 (above the power button) and set it to off

break;

}

Buttons and ScrollBars each have their own case. Used to invoke the required actions.

case Button\_3:

{ // power button pressed

Power = !Power; // toggle power var

if (Power) MyPanel.Send(Led\_1, "$RED"); else MyPanel.Send(Led\_1, "$OFF"); // led on/off

break;

}

case DynamicDisplay:

If enabled DynamicDisplay is sent from the panel every 500 ms. to request a dynamic display (measured values) update.

{ // dynamic display request

MyPanel.Sendf(Display\_1,"%03d", Value); // Display value

}

}

}

**Handling scrollbar**

Set initial value: Mypanel.Send( ScrollBar\_1, value); // value between 0 – max. (def. 100)

Set maximum value (in PanelConnected) MyPanel.Send( MaxScrollBar\_1, value); // positive only (scale in sketch)

case ScrollBar\_1:

{ //scrollbar change

MyScrollValue = MyPanel.vpr\_int ; // copy panel value to sketch variable (can only be int)

MyPanel.Sendf(Display\_2,"%03d ", MyScrollValue ); // Display value

Scrollbar does not invoke StaticDisplay to avoid serial congestion: Send static data from case. Value is transferred via panel public var.

}