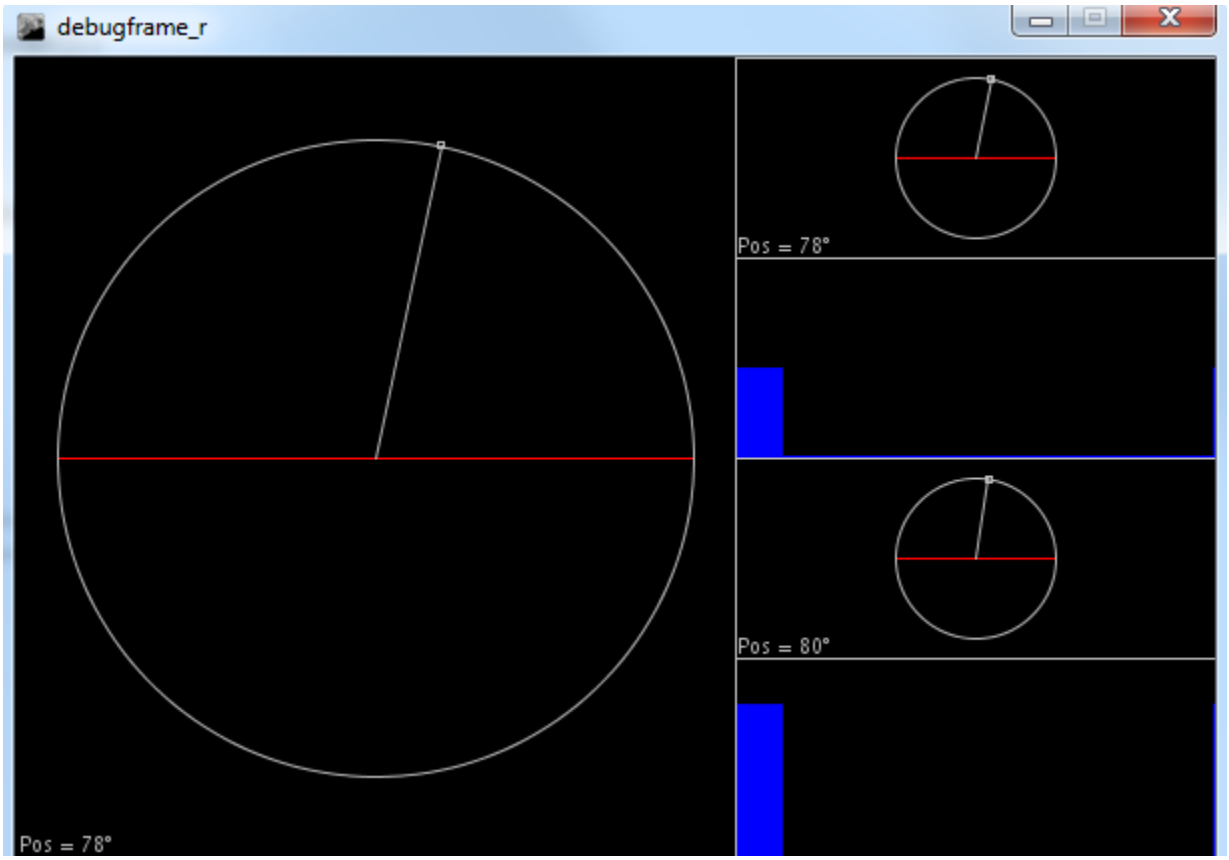
DebugFrame v1.0

DebugFrame is processing sketch that can be used to display serial output from an Arduino sketch and display they values graphically. The current version supports an angle view, useful to display servo positions and a graph view that displays values from a sensor. The graph view is an extension of the graph tutorial for arduino found here: http://arduino.cc/en/Tutorial/Graph. You need to understand that tutorial before proceeding. Feel free to change it, use it. I will be making changes periodically so use

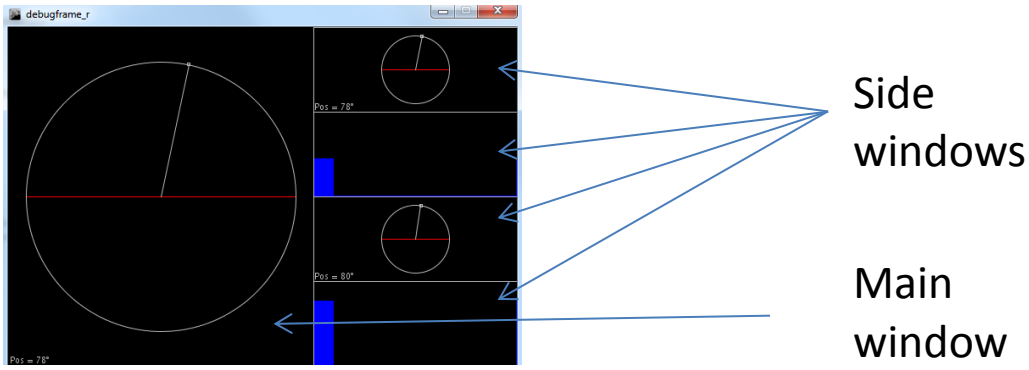


Installation

You need to install processing 32 bit http://www.processing.org/download/. As far as I can tell Serial functions do not work with the 64-bit version.

Description

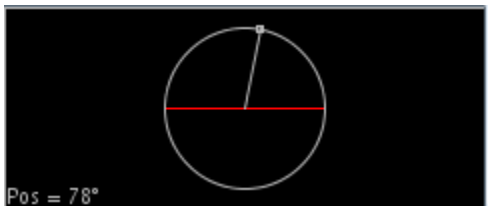
The application consist of a main window and configurable side windows



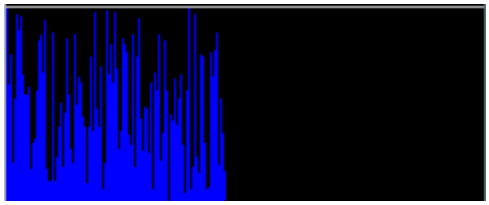
By pressing a numeric key 1..n the side the corresponding side window is displayed in main.

There are two types of side windows

1. Servo type: which can display the input as an angle



2. Barcharts that can display the input as a bar



Configuration

DebugFrame is easy to configure. Most of the information is contained in beginning of the sketch.

```
//=============================================================================================================
===
// Configuration part
//=============================================================================================================
===
// Mainwindow variables
```

```
// fVertical_partition will create a vertical line that splits the processing window into two part. The main partition which serves a zoom window and is configurable side windows.
// the first setting is the precentage of pixels of the processing window that will be dedicated to the zoom window vs the side windows. Using 60% as default the zoom window will be set at width*.60

float fVertical_partition = 0.60;

// iSide_windows will create that many side windows. In this case this will create 4 side windows
int iSide_windows =4 ;

// Below are two constants that tell debugframe the window type. Do not change
int WINDOW_TYPE_SERVO =1;
int WINDOW_TYPE_BARCHART =2;


// This array sets the types of side windows. It must contain at least the number of elements that are specified by iSide_windows. Each element must be a window type listed above.
int [] display_types = {WINDOW_TYPE_SERVO,WINDOW_TYPE_BARCHART,WINDOW_TYPE_SERVO,WINDOW_TYPE_BARCHART};



// This sets the packet characters. A packet is the input that debugframe is reading from the serial port. The corresponidng arduino sketch should serial.print the values in the packet format.
// Each packet starts with a ! (exlamation mark) and then is follows by a set of values separated by a comma. Each value should correspond to a side window in the order set by display_types
// For example !90,450,180,1023 in the default configuration will display 90 in the first servo box, 450 on the chart, 180 on the second servo box and 1023 on the last chart

//You can change the here the characters for packet start and packet separator
// character to denote start of package
char cPacketStart = '!';
//character to denote separator
char cPacketSeparator = ',';


// Parameters for WINDOW_TYPE_SERVO
//The radious of each circle in the servo box, color and fill
int iservocircle_radius_percentage = 80;
int iservocircle_color =187;
int iservocircle_fill =0;

// IMPORTANT: This will draw the limit of each angle in a WINDOW_TYPE_SERVO. The default Max angles of each servo attached in degrees. Array must contain at least as many items as WINDOW_TYPE_SERVOs are in display_types.
// Each element coresponds to each element in display_types in order that it is listed
int [] iServoAngles= {180,180};


// Parameters for WINDOW_TYPE_BARCHART
// the default upper and lower axis limits for each bar chart in pixes. The chart will map each input  and place it between ibar_lower_limits and ibar_upper_limits. Arrays must contain at least as many items as WINDOW_TYPE_BARCHART are in display_types.
// Each element coresponds to each element in display_types in order that it is listed
int [] ibar_lower_limits= {0,0};
int [] ibar_upper_limits= {1023,1023};
int ibarchart_color =#0000ff;
```

```
//This sets a global delay for the draw() in millisecons
int iSychms =15;

// ***** DON'T TOUCH ****
// Datapoint array and packet object.
int [] datapoints = new int [iSide_windows];
cPacket inPacket;
// This sets the parameters for the main frame. creating a MainFrame class oblect. The purpose is to draw the lines of the display
MainFrame mainframe;
// This sets the parameters for the main frame. creating a MainFrame class oblect. The purpose is to contain the different display objects
DisplayWindow main_display;
DisplayWindow[] side_frame_display = new DisplayWindow[iSide_windows];
int iServoTotallength;
// this is a flag indicating that a key was pressed. It will display in the zoom window the corresponding side window depending on the number that was pressed on the keyboard
int ikey_for_side_display=0;
```

Input

In your arduino sketch you need to serial.print a packet that is as follows

!num1,num2,num3,num4…..

Each number corresponds a side window. Here a quick Arduino Sketch that displays two servo angles as modified by a joystick. The sketch is a modification of the servo code http://pastebin.com/gdzGCBX3 and it is used to test debug frame.

```
#include <Servo.h>


const int servo1 = 8;      // first servo

const int servo2 = 9;

const int joyH = 5;       // L/R Parallax Thumbstick

const int joyV = 1;       // U/D Parallax Thumbstick




int servoValH,servoValV;

int tempJH, tempJV;

int old_servoValH=0;

int old_servoValV=0;
```

```
Servo myservo1;  // create servo object to control a servo
Servo myservo2;  // create servo object to control a servo



void setup() {

  // Servo
  myservo1.attach(servo1);  // attaches the servo
  myservo2.attach(servo2);  // attaches the servo

   myservo1.write(85);
   myservo2.write(120);

  // Inizialize Serial
  Serial.begin(9600);
}



void loop(){




   tempJH = analogRead(joyH);
   servoValH = map(tempJH, 0, 1023, 10, 170);

   if (old_servoValH!=servoValH)
      {
        old_servoValH=servoValH;
        myservo1.write(servoValH);
```

```
    }


  tempJV= analogRead(joyV);
  servoValV = map(tempJV, 0, 1023, 70, 170);


  if (old_servoValV!=servoValV)
    {
      old_servoValV=servoValV;
      myservo2.write(servoValV);
    }


  Serial.print("!");
  Serial.print(servoValH);
  Serial.print(",");
  Serial.print(tempJH);
  Serial.print(",");
  Serial.print(servoValV);
  Serial.print(",");
  Serial.print(tempJV);
  Serial.println();



  delay(15);


}
```