Version 0.3

Welcome to EzScrn

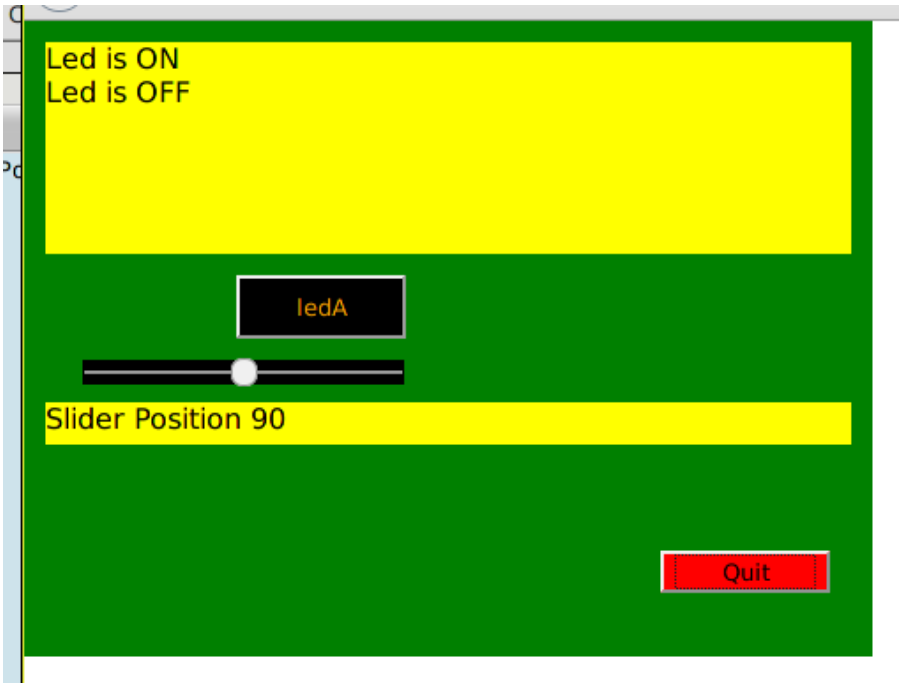
EzScrn is a Python program that works with your Arduino program to provide a simple way to create a GUI screen on a PC (or phone or tablet). The Python program is a small webserver so the GUI screen is accessed through your Browser.

The idea is that all of the design of the GUI is generated with a few lines of Arduino code and all you need to know about Python is how to run the program and access the GUI with your browser.

For example, these few lines of Arduino code create this GUI screen.

```
Serial.println("<+newScrn, size=40x30, tleft=0x0, bg=green, fg=black>");
Serial.println("<+tOut, name=outA, size=38x10, tleft=1x1, bg=yellow, fg=black>");
Serial.println("<+btn, name=ledA, size=8x3, tleft=10x12, bg=black, fg=orange>");
Serial.println("<+slid, name=slidA, size=15x1, tleft=2x16, range=30x150x90x10>");
Serial.println("<+tOut, name=outB, size=38x2, tleft=1x18, bg=yellow, fg=black>");
Serial.println("<+quit, name=Quit, size=8x2, tleft=30x25, bg=red, fg=black>");
Serial.println("<+endScrn>");
```

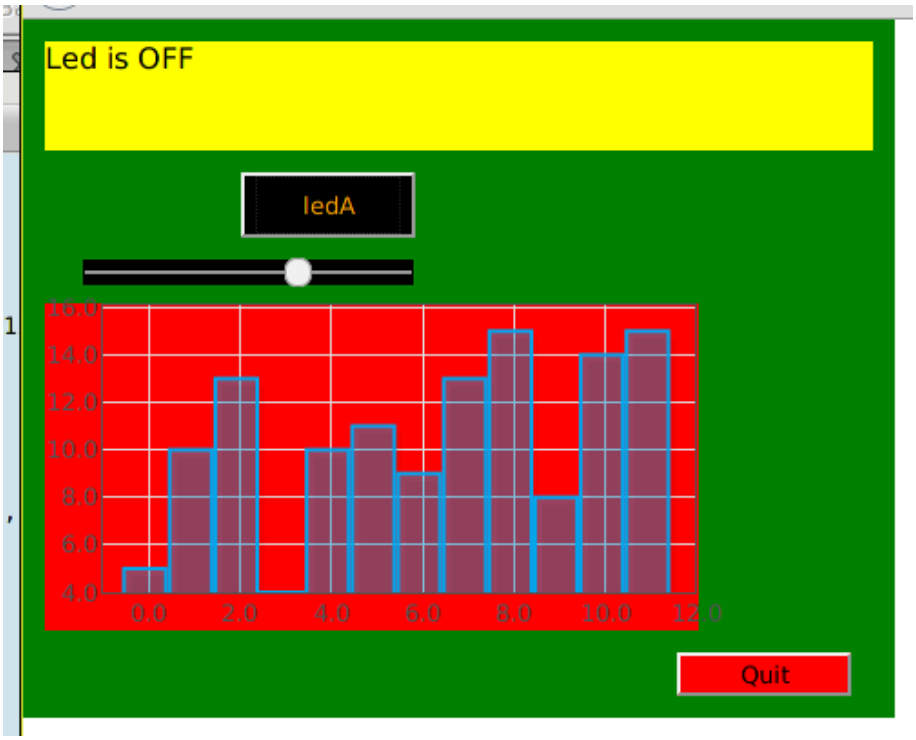
create this GUI screen.



And this code

```
Serial.println("<+newScrn, size=40x32, tleft=0x0, bg=green, fg=black>");
Serial.println("<+tOut, name=outA, size=38x5, tleft=1x1, bg=yellow, fg=black>");
Serial.println("<+btn, name=ledA, size=8x3, tleft=10x7, bg=black, fg=orange>");
Serial.println("<+slid, name=slidA, size=15x1, tleft=2x11, range=4x25x15x1>");
Serial.println("<+chrt, name=chrtA, size=30x15, tleft=1x13, bg=red, fg=black>");
Serial.println("<+quit, name=Quit, size=8x2, tleft=30x29, bg=red, fg=black>");
Serial.println("<+endScrn>");
```

creates this screen with a chart that is updated from values received from the Arduino. In the demo the slider controls the maximum value of the chart data.



Quick Intro ---

To get **EzScrn** to work you need to copy the Python code to a directory on your PC. By default **EzScrn** uses the IP address 127.0.0.1:8085 which will allow the GUI to be viewed in the Browser on the same PC as the Python code. These values are in modules/globalData if you need to change them

One of things you need to think about is stopping the server because that will be be necessary to free up the Serial Port if you want to upload a revised program to your Arduino. The demo Arduino program includes a QUIT button for this purpose and it is advisable to include it in your own GUI designs. If you start the Python program from a terminal window you can stop the server by typing CTRL-C in that window.

Assuming you have the Python code copied to your PC and have assigned an appropriate IP address getting the system working requires the following simple steps. (I am assuming you are using the default IP address of 127.0.0.1:8085).

Upload the demo program to your Arduino in the normal way.

Open a terminal and switch to the directory where the Python program is located.

In the terminal window type python **EzScrn**.py

It should then say

```
Bottle v0.12.7 server starting up (using CherryPyServer())...
Listening on http://127.0.0.1:8085/
Hit Ctrl-C to quit.
```

Open your browser and enter the address 127.0.0.1:8085 and press Enter

You should then see the screen that allows you to select the serial port for your Arduino and when you select it the GUI should appear in the browser.

The LED button should toggle the onboard LED on pin 13 of the Arduino and the slider should adjust the position of a servo connected to pin 8. Some messages will appear in the text box.

Arduino Yun

EzScrn works on a Yun (with only minimal changes) which means that you can have a complete Arduino project controllable from a browser on a PC, Tablet or smartphone in a very small package.

Creating a screen design

Screen Elements

Elements for a screen design are defined with

+newScrn	tells the **EzScrn** to start a new screen design
+endScrn	marks the end of the screen design
+tOut	creates an area to displaytext sent by the Arduino
+tInW	tInW and tInI create an area for inputting data to be sent to the Arduino
+tinI	An input created with tInW will not send the data until a button is pressed
	An input created with tInI will send the data immediately
+btn	creates a button
+slid	creates a slider
+chrt	creates an area to display a chart
+quit	creates a QUIT button to stop the server

Element Attributes

Each element **MUST** be accompanied with

name=NNN	any name of your choice can be used. Short names are recommended DO NOT use the name xx for a button as that is used to indicate that no button was pressed

Each element **MAY** be accomanied by

size=HxV	measurements are in characters HorizontalxVertical separated by an 'x' eg 10x3
tleft=HxV	the top left position of the element
bg=CCC	the background colour (see below for colour details)
fg=CCC	foreground ie text colour

The +slid element **should** also include

range=BxTxVxS

B represents the Bottom or minimum value

T represents the Top or max value

V represents the start position within the range

S represents the step size

for example range=30x150x90x10 results in values from 30 to 150, a starting position of 90 and a step size of 10.

Example Element Definition

The format of a complete element definition is as follows

```
Serial.println("<+btn, name=ledA, size=8x3, tleft=10x12, bg=black, fg=orange>");
```

and, focusing on the content between quotes

```
<+btn, name=ledA, size=8x3, tleft=10x12, bg=black, fg=orange>
```

The < and > are start and end markers that enable **EzScrn** to reliably detect a message

Each of the items is separated by a comma

+btn	indicates a requirement for a button element
name=ledA	defines the name of the element
size=8x3	defines a size 8 chars wide by 3 chars high
tleft=10x12	positions the top left corner at the point 10 chars from the left edge and 12 chars from the top edge
bg=black	defines a black background
fg=orange	defines orange text

Order of items

The element type (e.g. +btn) must come first but the other items can be in any order and may be omitted

Overall Screen Size

The size= values in the +newScrn command define the overall size of the screen. The top left corner will always be 0x0. **EzScrn** will adjust subsequent size and position values to ensure an element stays within the area defined by +newScrn.

Default Values

Default values are defined in modules/globalData

Measurements are in characters

The default value for a screen size is 40x30 (width x height)

The default value for an element size is 10x4   (width x height)

The default position for elements ensures that they align one beneath the other with a blank line between them

The default bg colour is **green**

The default fg colour is **black**

The default range for a slider is 0,10,5,1 (min, max, position, step)

Inherited values

If an element does not include one of size=, bg=, fg= the most recent value will be used

(If you want three buttons the same size you only need to define the size for the first one)

Incorrect or invalid values

Anything that **EzScrn** does not recognize will be ignored

Colours

Where a colour value is required it can be any of the color names recognized by CSS or an RGB number in the style #rrggbb where rr gg and bb are the HEX values for the Red, Green and Blue proportions. For example #FF0000 is red, #FFFFFF is white and #000000 is black

Interacting with a screen

As soon as **EzScrn** detects the +endScrn indicating the end of the screen design it will display the screen

EzScrn allows the Arduino two types of interaction with the screen. It can send text to be displayed, and it can change the background colour of a button.

Sending data FROM the screen TO the Arduino

When the user enters data in an input box, presses a button or moves a slider data will be sent directly to the Arduino.

In the example there is a button “ledA”, and a slider. That means the Arduino will receive two items – the value of the slider position and the name of the button. For example 90,ledA. Or, if no button was pressed the Arduino would receive 90,xx

If there are several input elements – for example input boxes and sliders the data from them will be passed to the Arduino in the order that the inputs were created in the screen design.

The name of the button that was pressed, or xx if none was pressed, will always be the final piece of data.

The example Arduino code illustrates how the received data can be parsed and saved to variables.

Sending text To the screen From the Arduino

Text for a specific output element

The Arduino program can send text to any of the output elements on the screen by including the name of the output element. For example

```
Serial.print(“<+txtA,This is a Test>”
```

will ADD the words “This is a Test” to whatever may be in the box named txtA. And

```
Serial.print(“<-txtA,This is a Test>”
```

will REPLACE the text in the box txtA with the words This is a Test. Note the **-txtA** in this case

Text without an element name

If text is sent to the screen without a valid output box name, or with no name, it will be displayed in the last output box that created in the screen design

Line breaks

If you wish to have text print on a different line you must include a linefeed character within you text. Note that

```
Serial.println("<+txtA,This is a Test>"
```

will NOT produce a new line because the linefeed character is not between the < > characters. To include a linefeed you can either do

```
Serial.print("<+txtA,This is a Test\n>"
```

OR

```
Serial.print("<+txtA,This is a Test
Serial.println()
Serial.print(">")
```

Updating the background colour of a Button

The background colour of a button may be updated by sending the button name and the required background colour. For example

```
Serial.print("<+btnA,bg=red>"
```

Sending data for lines (or bars) on a chart

The Javascript library Flotr2 is used to create the charts. The format of the data for a line on a graph is closely related to the format used by Flotr2.

There may be several lines (or series of bars) on a single chart. All of the data must be sent as a single instruction for **EzScrn**. For example

```
<+line, name=chrtA, data=bars:[2x3]x[4x18.5]x[6x5]x[7x13]~lines:
[2x3]x[4x18.5]x[6x5]x[7x13]>
```

will create a series of bars and a line on chrtA

Keen readers who study the Flotr2 documentation will note that all the Xs will be coverted to commas and that Flotr2 uses “bars:” and “lines:” (rather than “bar:” and “line:”) to define the type of display.

The values to be displayed are simply a series of XY pairs. There can be as many as you wish and both lines do not have to have the same number of pairs.

Note the use of the tilde ~ to allow **EzScrn** to split the data for the different lines.

EzScrn can only display one chart.

Using EzScrn on a Yun

EzScrn works on a Yun with very few changes.

- The most complex change is the need to disable the automatic startup of the Arduino Bridge when the Yun boots. This is because Python needs to use the same serial connection (/dev/ATH0) for communication with the 32u4 Arduino microprocessor.

  Two small Python programs are provided to facilitate this - DisableYunbridge.py and EnableYunbridge.py. Comments within those programs explain what they do.

- The Arduino code for use on the 32u4 must refer to Serial1 rather than Serial. On the Yun Serial is used (as on the Uno and Leonardo) for communication with a PC through the USB connection that is used to upload programs. Serial1 is used to communicate with the Linux side of the Yun.
- The Python file module/globalData.py must be modified to identify the fact that you are using a Yun by setting arduinoBoard = 'Yun'

The following steps are required to get EzScrn working on a Yun. I am assuming some basic familiarity with the Yun.

- Upload the Python code (the directory EzScrn02 and its contents) to the Linux side of the Yun.
- Upload the Arduino code (the file DemoYun.ino) to the 32u4 Arduino microprocessor
- Use SSH from a PC to open a terminal on the Yun. (You can't use the Arduino YunSerialTerminal program because DemoYun is loaded on the Arduino side).
- Change to the EzScrn02 directory.
- If you have not already done so run the program DisableYunBridge.py with the command

  ```
  python DisableYunBridge.py
  ```

- If you need to re-enable the bridge to use with another Yun/Arduino program use the command

  ```
  python EnableYunBridge.py
  ```

- Restart the Yun so that the new bridge setting can take effect.
- When the Yun has restarted with the bridge disabled
- Use SSH to open a terminal
- Change to the EzScrn02 directory and start **EzScrn** with the command (just as if you were running the program on a regular PC)

  ```
  python EzScrn02.py
  ```

- If you leave the terminal open you will be able to see the same messages that appear when running **EzScrn** on your PC.

For Windows users

I believe you need to use the puTTY terminal program to connect to the Yun using SSH

I have used a program called scp to copy files to the Linux side of my Yun from my Linux laptop. I believe there is a program called WinSCP that does the same thing from Windows.

(scp = secure copy)

Operating EzScrn on multiple screens

The IP address used by EzScrn is set in the file modules/globalData. The default value of 127.0.0.1:8085 allows the output of the server to be displayed in a browser on the same PC as the Python code.

Note that 8085 is the port number. If that conflicts with something else on your PC you can use another number.

If you set the IP address to that of an existing network other devices on the network, such as smartphone or tablet may also connect to **EScrn**. They will display the same screen design and allow the same interaction.

Note, however that the option to select the serial port will only be presented to the first browser that accesses **EzScrn** after the Python program is started.

For example, my laptop is connected to a WiFi hotspot and if I use the IP address assigned by the hotspot (something like 192.168.nnn.nnn) I can use the browser on any PC (or phone or tablet) connected to that network.

I have tested **EzScrn** on my laptop and phone at the same time. I assume it will work with a few more connections but it is NOT intended for dozens or hundreds of connections. After all, there is only one Arduino to control.

Internet Security

EzScrn is not intended to be a secure internet product. NO attempt has been made to provide security of any kind. **YOU HAVE BEEN WARNED**.

PC and Software requirements

EzScrn is written in Python 2.7 and uses the Bottle web framework and the server from the CherryPy web framework to operate the screen. PySerial is used to interact with the Arduin.

To the best of my knowledge **EzScrn** needs nothing on your PC apart from the Python 2.7 interpreter. The code for the Bottle web framework, the server part of the CherryPy web framework and the code for PySerial are all included.

I have developed **EzScrn** on Linux and tested it on Windows 7 with a newly installed version of

Firefox. It did not work properly with Internet Explorer 8 and I was not able to upgrade to IE 11.

Note that I have slightly modified the CherryPy server code to ensure the server shuts down cleanly. For those who are interested the changes are all marked with #rmk.

Note also that the simple server included with Bottle would not provide the interaction that I needed – hence the use of server from CherryPy.

Licence and Warranty

The code which I have created may be copied and used as you wish provided that you do nothing to restrict what other people may do with it.

The code for Bottle, CherryPy, PySerial and Flotr2 is subject to their own licence requirements.

This is demonstration code. It comes **without any warranty** of fitness for any purpose.

...END