APPLICATION NOTES

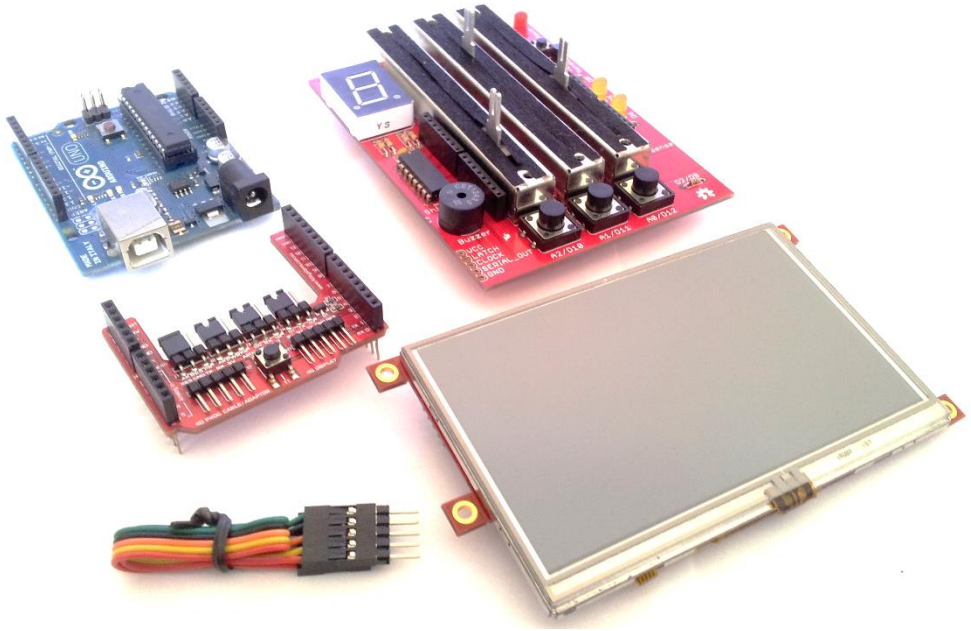
4D SYSTEMS
TURNING TECHNOLOGY INTO ART

ViSi-Genie Arduino Danger Shield

DOCUMENT DATE: **13th April 2019**
DOCUMENT REVISION: **1.1**

Description

This application note explains how to interface a 4D display module, together with the Danger Shield, to an Arduino host. The demo in this application note is based on this video from Sparkfun. The host is an AVR-ATmega328-microcontroller-based Arduino Uno board. Ideally, the application described in this document should work with any Arduino board that is compatible with the Danger Shield and has at least one UART serial port. See specifications of Aduino boards here.



Before getting started, the following are required:

- Any of the following 4D Picaso display modules:

uLCD-24PTU gen4-uLCD-24PT	uLCD-28PTU gen4-uLCD-28PT	uVGA-III gen4-uLCD-32PT

- The target module can also be a Diablo16 display

gen4-uLCD-24D Series	gen4-uLCD-28D Series	gen4-uLCD-32D Series
gen4-uLCD-35D Series	gen4-uLCD-43D Series	gen4-uLCD-50D Series
gen4-uLCD-70D Series		
uLCD-35DT	uLCD-43D Series	uLCD-70DT

See the section "Write to a Pin Output Object" when compiling this project for a Diablo16 display module.

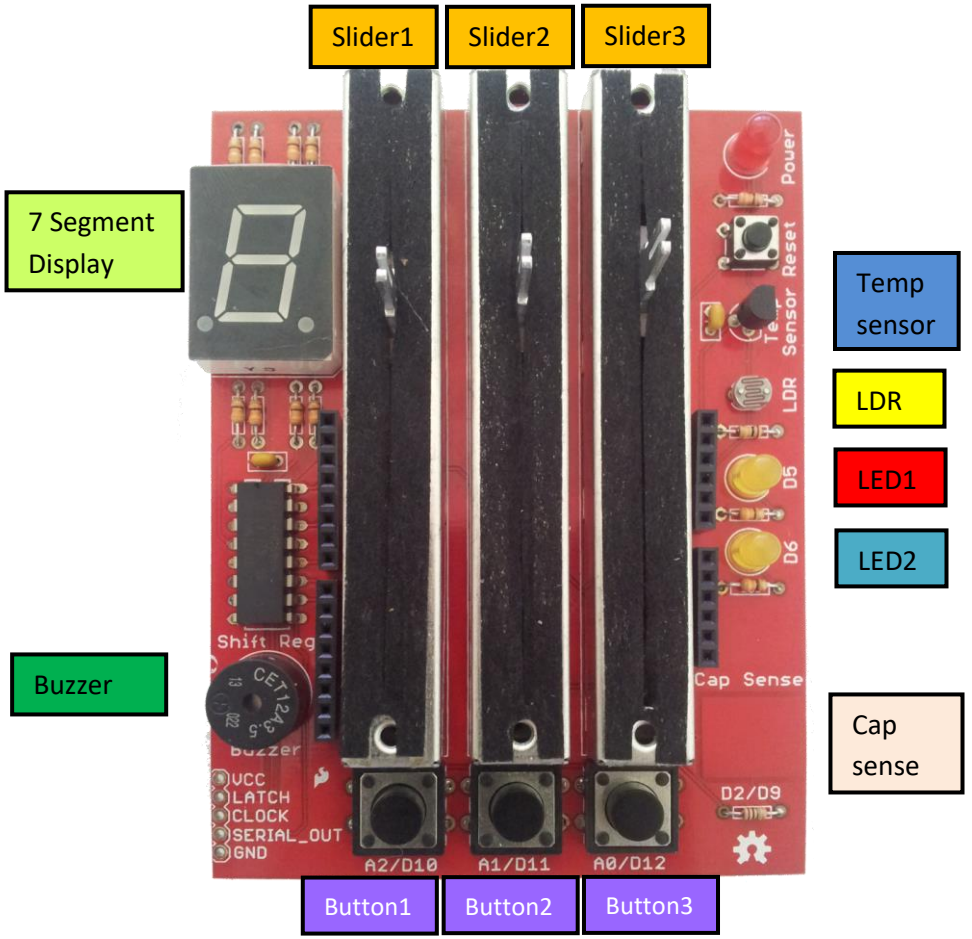
Visit www.4dsystems.com.au/products to see the latest display module products that use the Diablo16 processor.
The display module used in this application note is the uLCD-32PTU, which is a Picaso display. This application note is applicable to Diablo16 display modules as well.

- 4D Programming Cable / µUSB-PA5/µUSB-PA5-II
  for non-gen4 displays (uLCD-xxx)
- 4D Programming Cable & gen4-IB / gen4-Pa / 4D-UPA,
  for gen-4 displays (gen4-uLCD-xxx)
- micro-SD (µSD) memory card
- Workshop 4 IDE (installed according to the installation document)
- When downloading an application note, a list of recommended application notes is shown. It is assumed that the user has read or has a working knowledge of the topics presented in these recommended application notes.

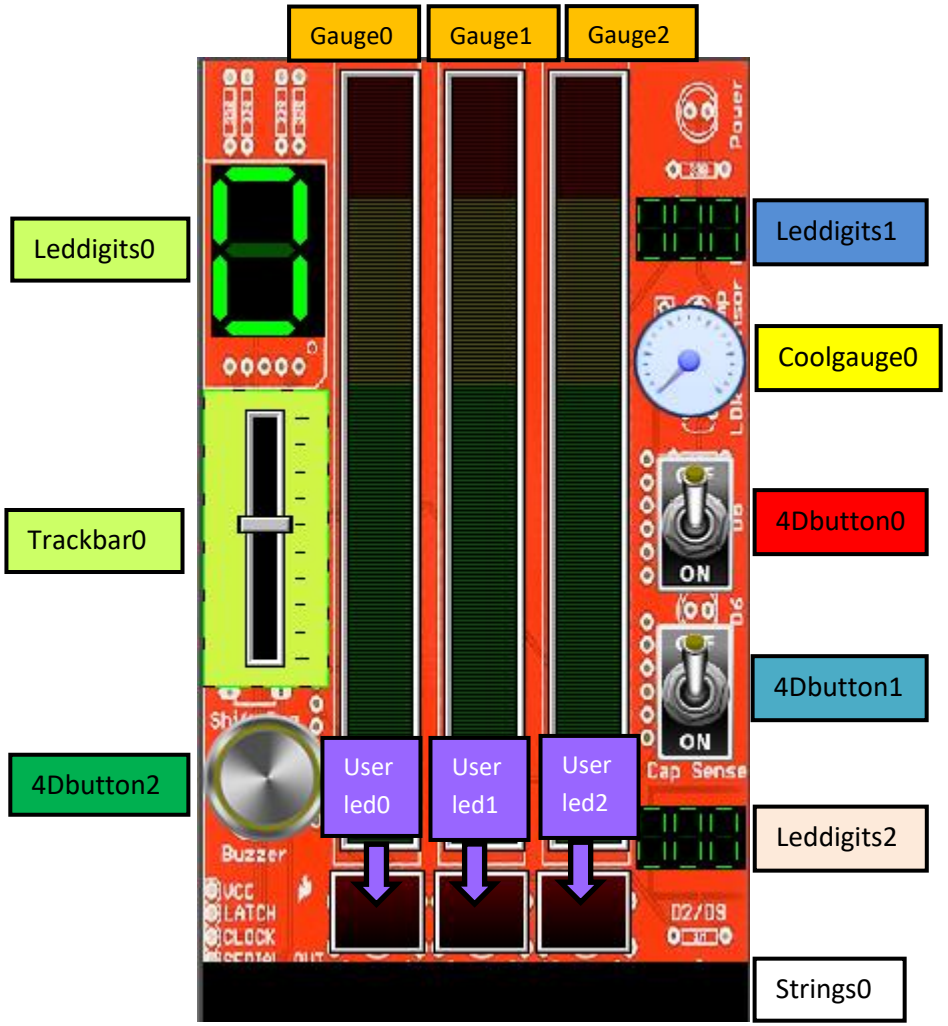
Content

Application Overview

Danger Shield

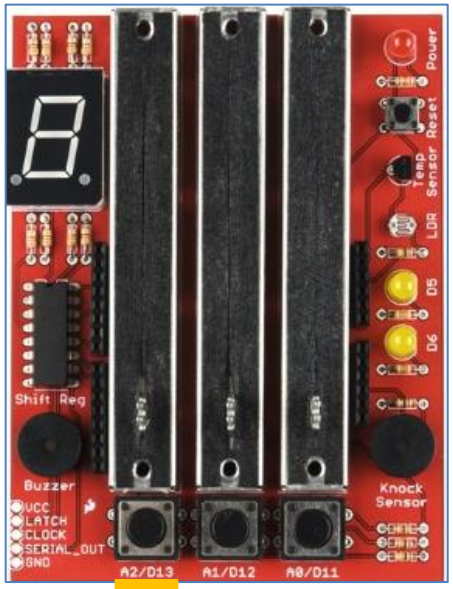


4D uLCD-43PT display



The Danger Shield and the uLCD-43PT display are interfaced to the Arduino host as shown below.

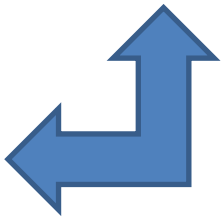
Danger Shield



4D uLCD-43PT display

Connection thru GPIO pins

Arduino Uno



Serial communication

In this application, the objects displayed on the uLCD-43PT will reflect the status of the components on the Danger Shield. Also, since the uLCD-43PT is a touch screen display, it also possible for it to control an output

component of the Danger Shield such as the buzzer and the two LEDs. This two-way control is processed by the Arduino host. Together with the color-coded images on page 4, the table below is presented to help the beginner understand the application and the Arduino sketch. A change initiated by an input triggers a change in an output.

Input – Danger Shield	Output – uLCD-43PT
Slider1	Gauge0
Slider2	Gauge1
Slider3	Gauge2
Button1	Userled0
Button2	Userled1
Button3	Userled2
LDR	Coolgauge0
Temperature Sensor	Leddigits1
Cap sense	Leddigits2

Input – uLCD-43PT	Output – Danger Shield
4Dbutton0	LED1
4Dbutton1	LED2
4Dbutton2	Buzzer
Trackbar0*	7-segment display

***Trackbar0 controls Leddigits0 as well**

The object Strings0 displays strings received from the Arduino host.

Users who want to learn more of how the Danger Shield works with the Arduino host may visit www.sparkfun.com.

Setup Procedure

For instructions on how to launch Workshop 4, how to open a ViSi-Genie project, and how to change the target display, kindly refer to the section "**Setup Procedure**" of the application note:

ViSi Genie Getting Started – First Project for Picaso Displays (for Picaso)
or
ViSi Genie Getting Started – First Project for Diablo16 Displays (for Diablo16).

Create a New Project

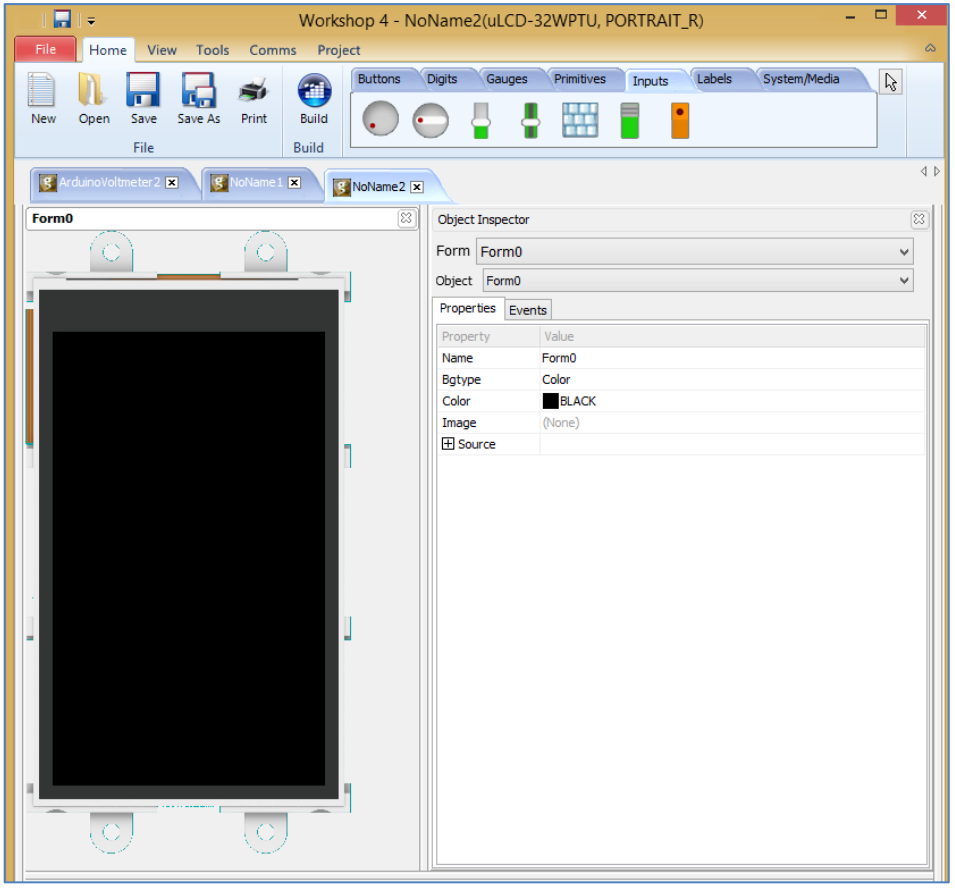
Create a New Project

For instructions on how to create a new ViSi-Genie project, please refer to the section "**Create a New Project**" of the application note

ViSi Genie Getting Started – First Project for Picaso Displays (for Picaso)
or
ViSi Genie Getting Started – First Project for Diablo16 Displays (for Diablo16).

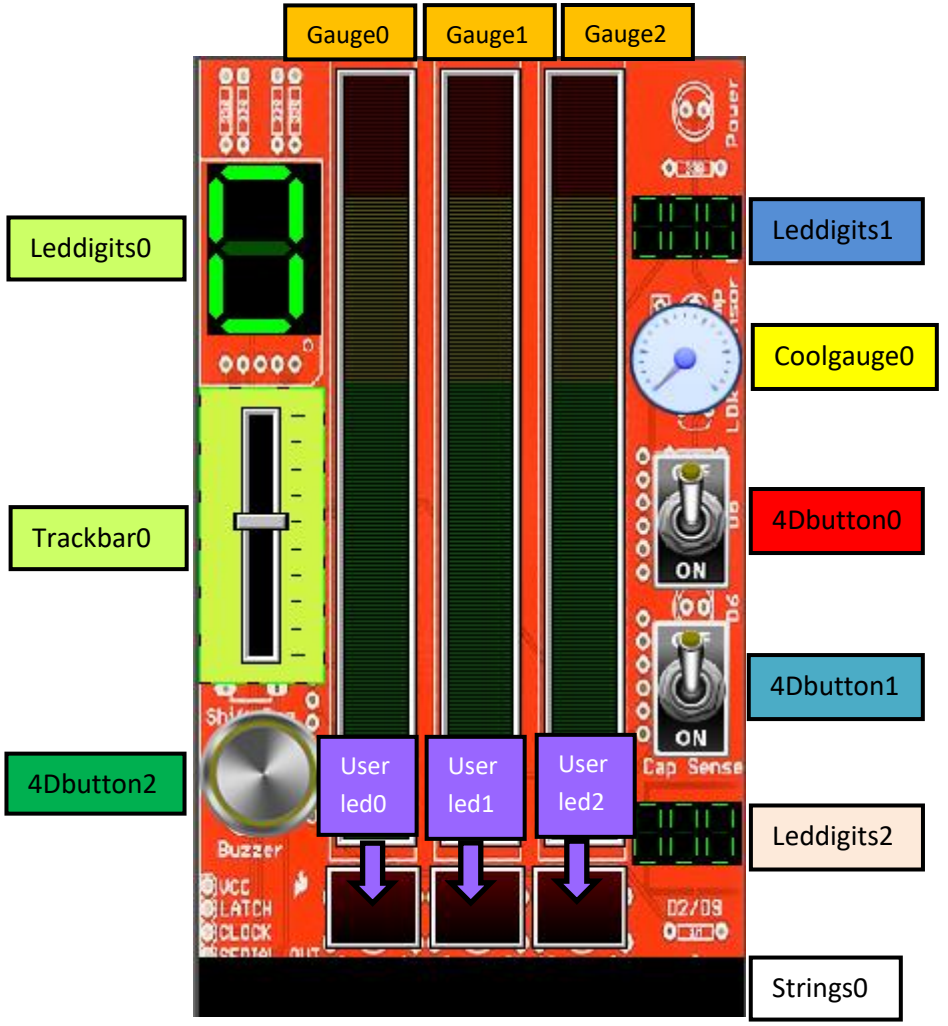
Design the Application

Everything is now ready to start designing the project. **Workshop 4** displays an empty screen, called **Form0**. A **form** is like a page on the screen. It can contain **widgets** or **objects**, like sliders, displays or keyboards. Below is an empty form.



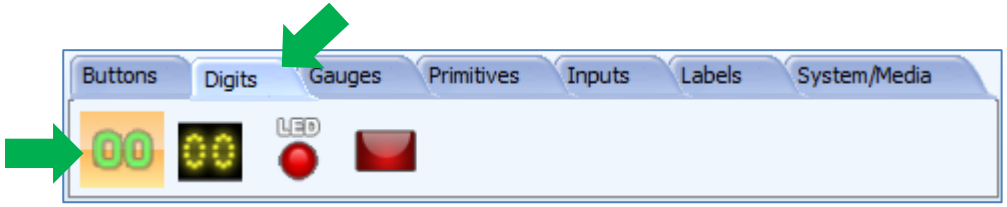
At the end of this section, the user will able to create a form with fifteen objects. The final form will look like as shown below excluding the labels.

4D uLCD-43PT display

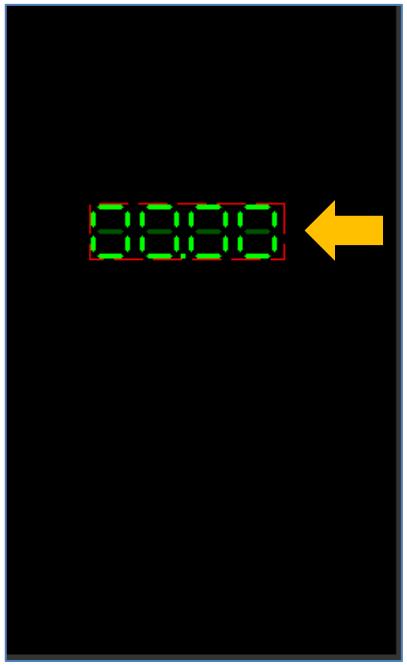


Add a LED Digits Object

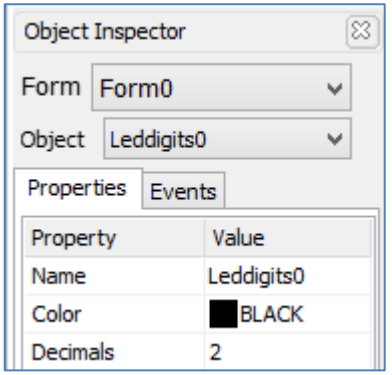
To add a LED digits object, go to the **Digits** pane and select the first icon.

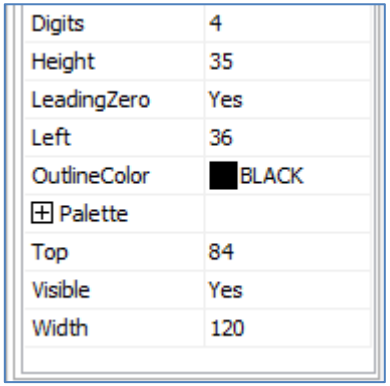


Click on the **WYSIWYG** (What-You-See-Is-What-You-Get) screen to place a LED digits object. The WYSIWYG screen simulates the actual appearance of the display module screen.

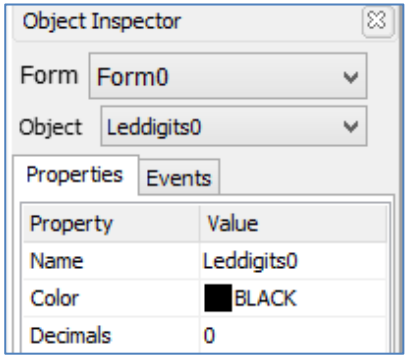


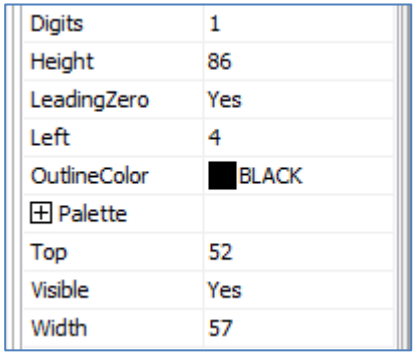
The object can be dragged to any desired location and resized to the desired dimensions. The **Object Inspector** on the right part of the screen displays all the properties of the newly created LED digits object named **Leddigits0**.



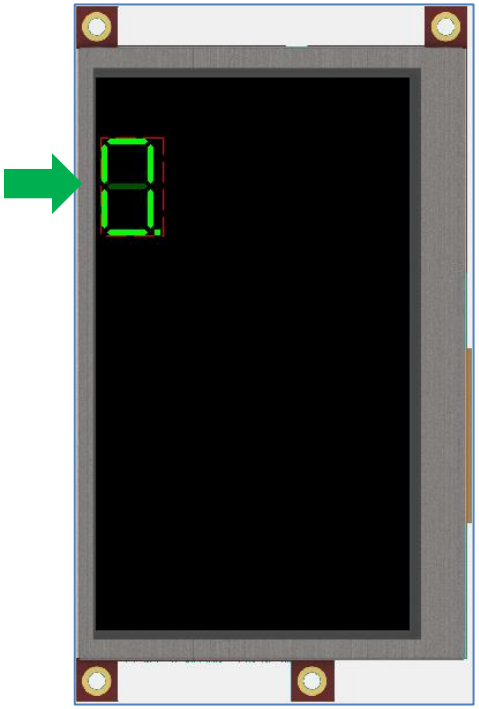


Feel free to experiment with the different properties. To know more about digital display objects, refer to ViSi-Genie Digital-Displays**.** Leddigits0 in this example has the following properties.



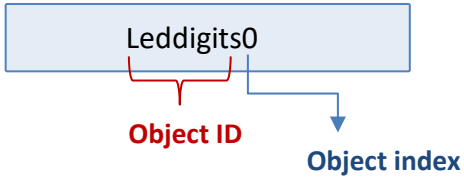


The object has a single digit and is positioned at the left side of the screen.



Naming of Objects

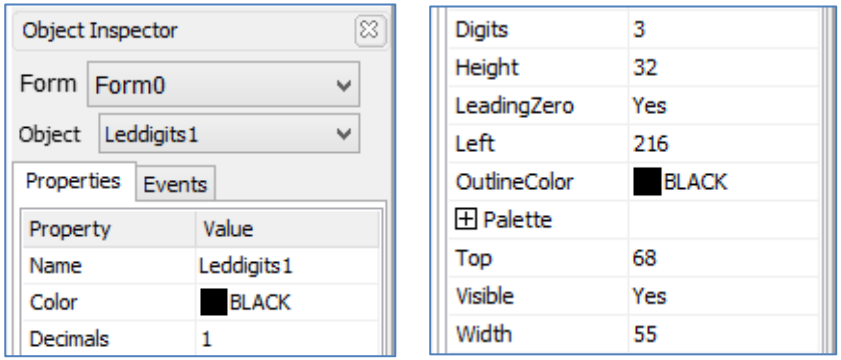
Naming is important to differentiate between objects of the same kind. For instance, suppose the user adds another LED digits object to the WYSIWYG screen. This object will be given the name Leddigits1 – it being the second LED digits object in the project. The third LED digits object will be given the name Leddigits2, and so on. An object's name therefore identifies the object's kind and unique index number. It has an ID (or type) and an index.

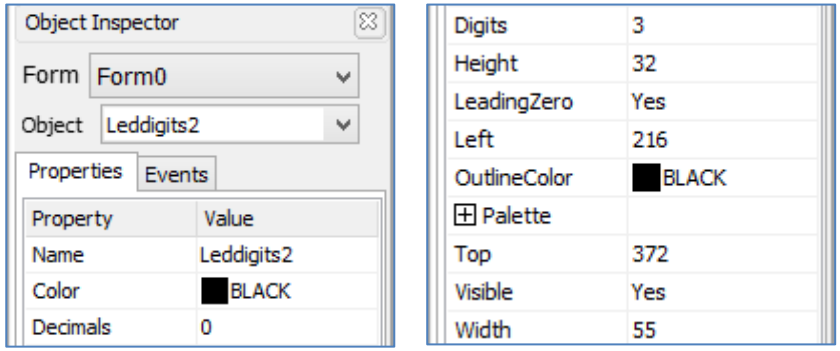


It is important to take note of an object's ID and index. When programming in the Arduino IDE, an object's status can be polled or changed if its ID and index are known.

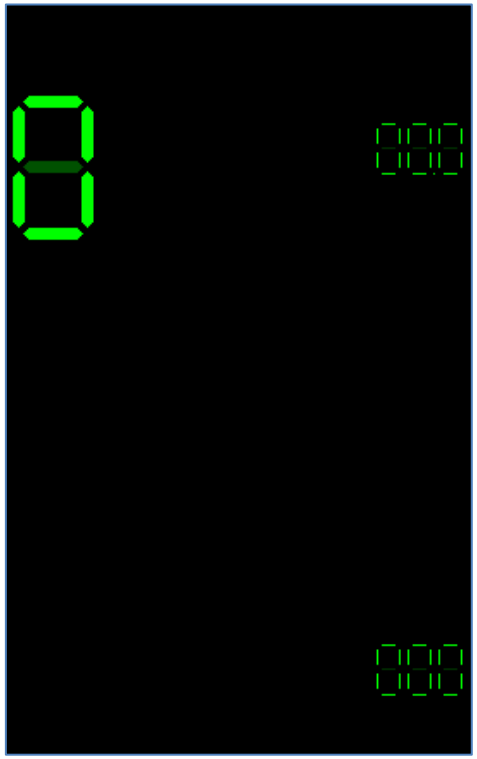
The project has two more LED digits objects which have the following properties.

third parameter is an integer which holds the data to be written to the object. Example:



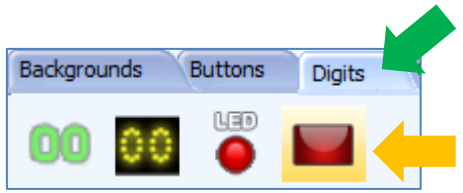


After having added the two LED digits objects, the form will look like as shown below.

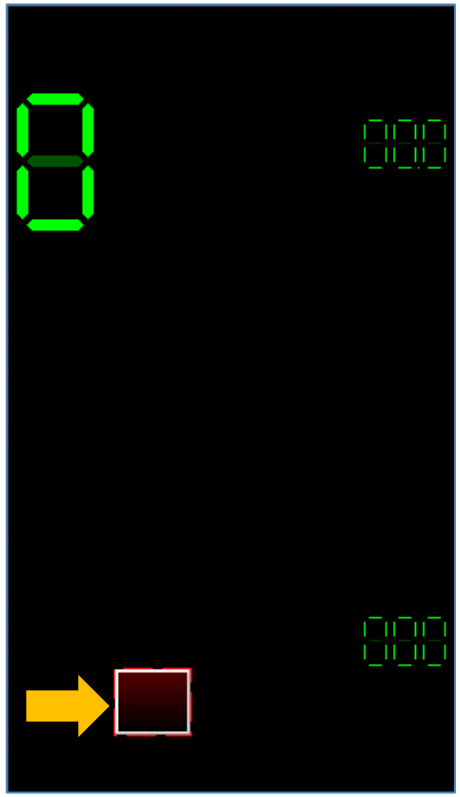


Add a User LED Object

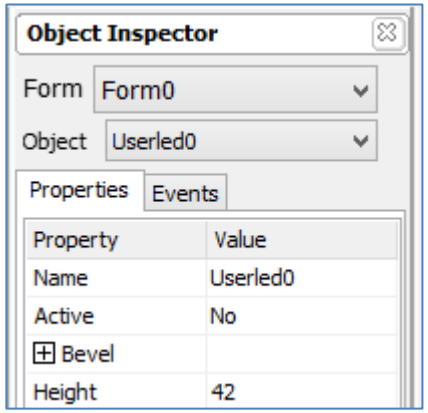
To add a User LED, go to the Digits pane and select the User LED icon

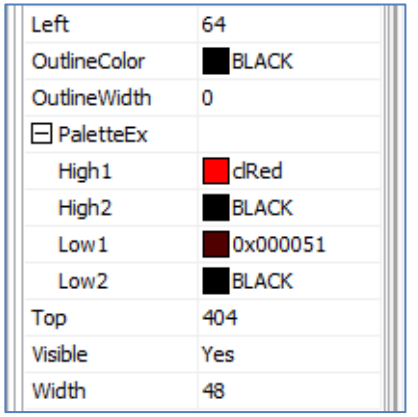


Click on the WYSIWYG screen to place the object.



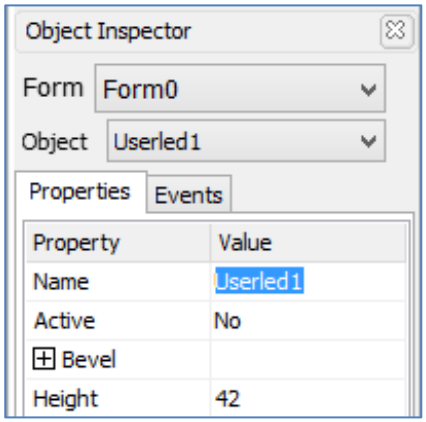
Userled0 of this project has the following properties.

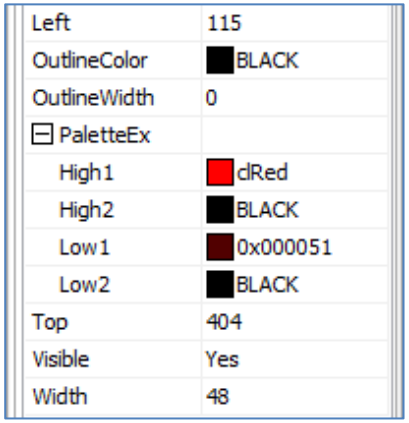


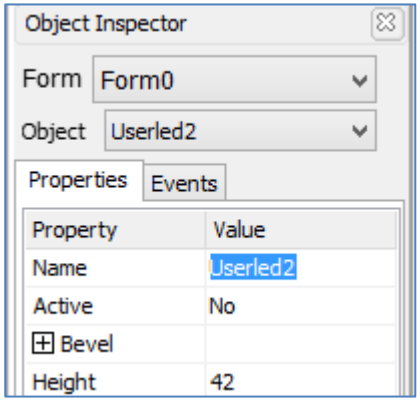


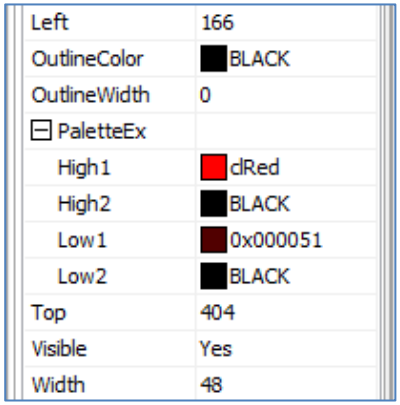
Add two more User LED objects with similar properties to the screen.

Userled1 and Userled2 have the following properties.



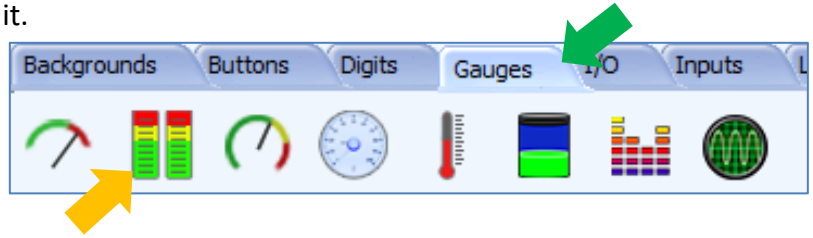




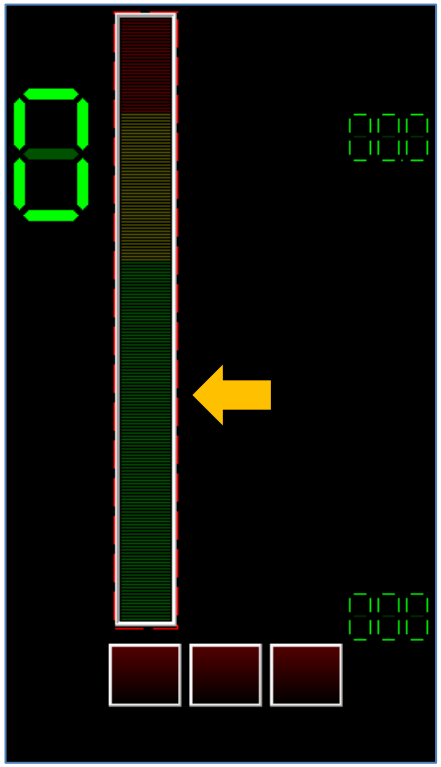


Add a Gauge

To add a gauge object, go to the **Gauges** pane and click on the gauge icon to select it.



Click on the WYSIWYG screen to place the object.



In the Object Inspector, apply the following property values.

Object Inspector

Form: Form0

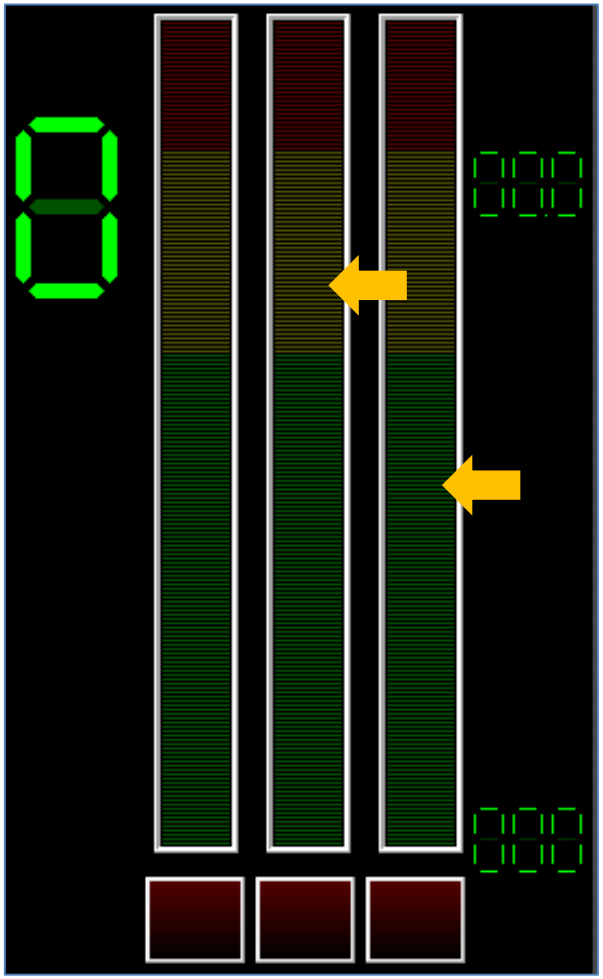
Object: Gauge0

Properties | Events

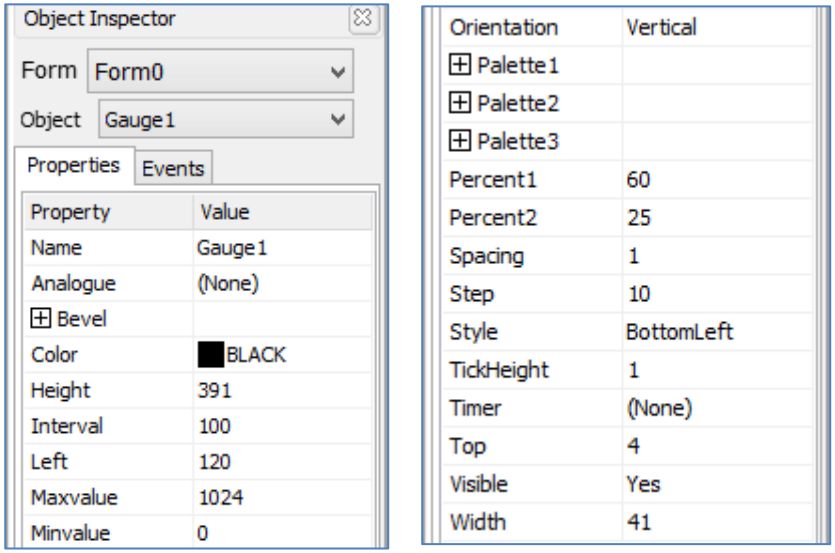
Property	Value
Name	Gauge0
Analogue	(None)
⊞ Bevel	
Color	BLACK
Height	392
Interval	100
Left	68
Maxvalue	1024
Minvalue	0

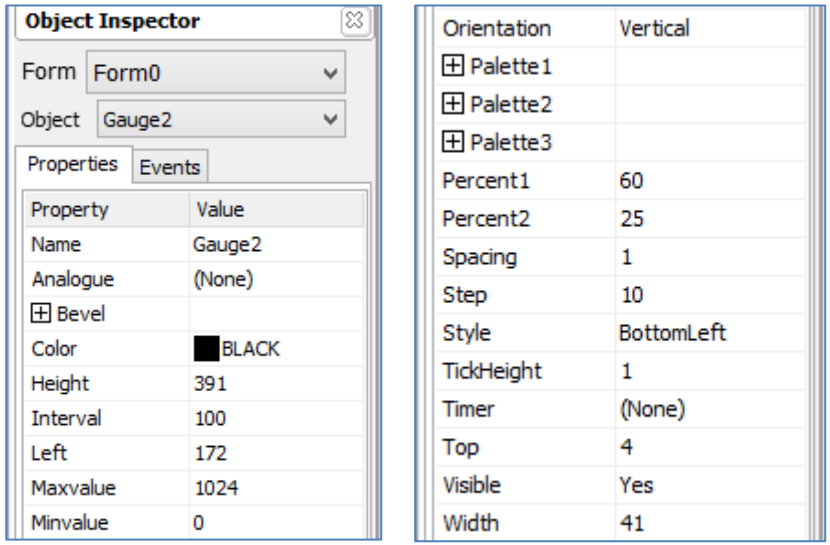
Orientation	Vertical
⊞ Palette1	
⊞ Palette2	
⊞ Palette3	
Percent1	60
Percent2	25
Spacing	1
Step	10
Style	BottomLeft
TickHeight	1
Timer	(None)
Top	4
Visible	Yes
Width	41

Add two more gauge objects to the screen. To know more about meters and gauges, read ViSi-Genie Gauges.



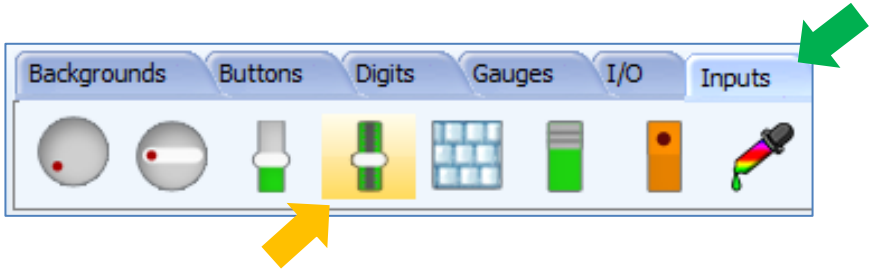
Gauge1 and Gauge2 have the following properties.



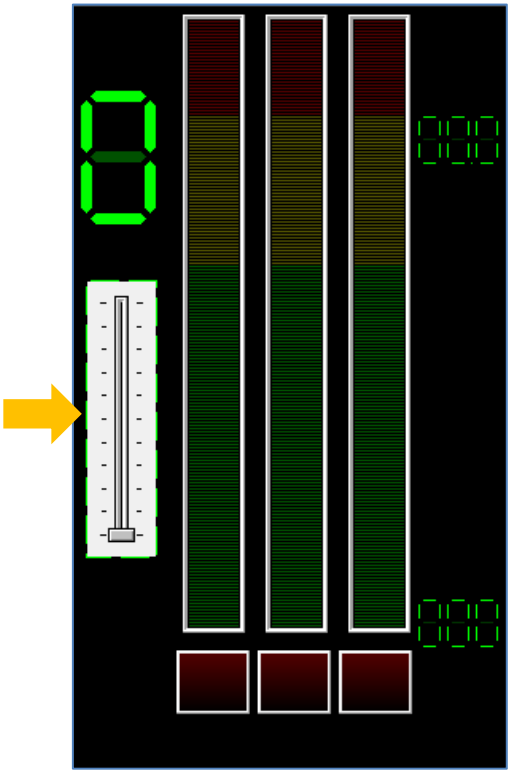


Add a Track Bar

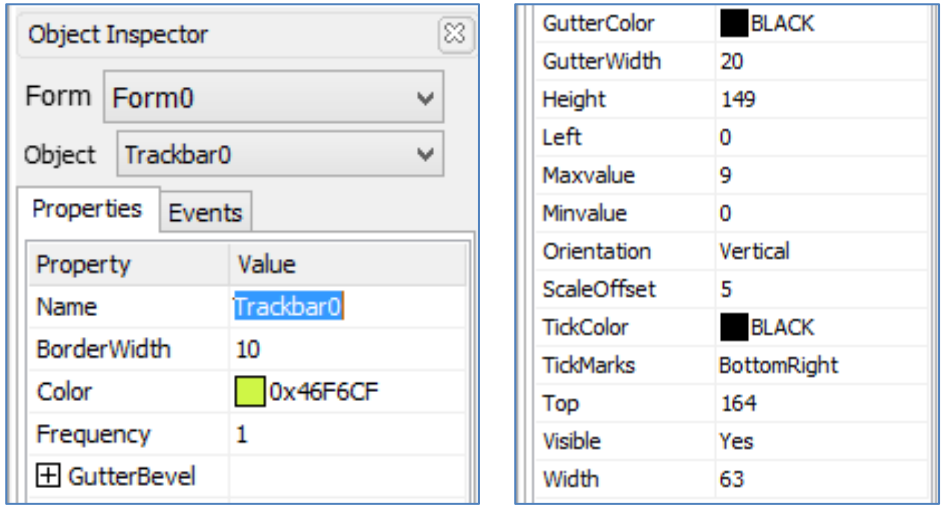
To add a track bar object go to the Inputs pane and click on the track bar icon.



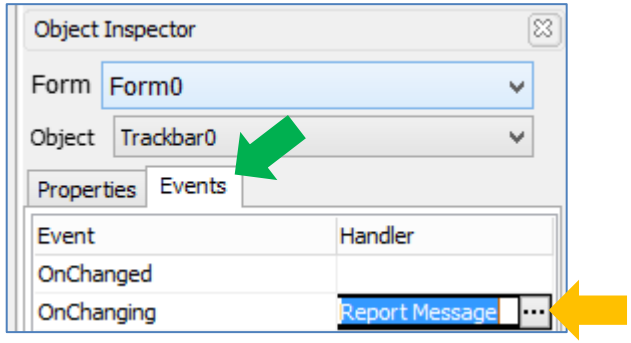
Click on the WYSIWYG screen to add the object.



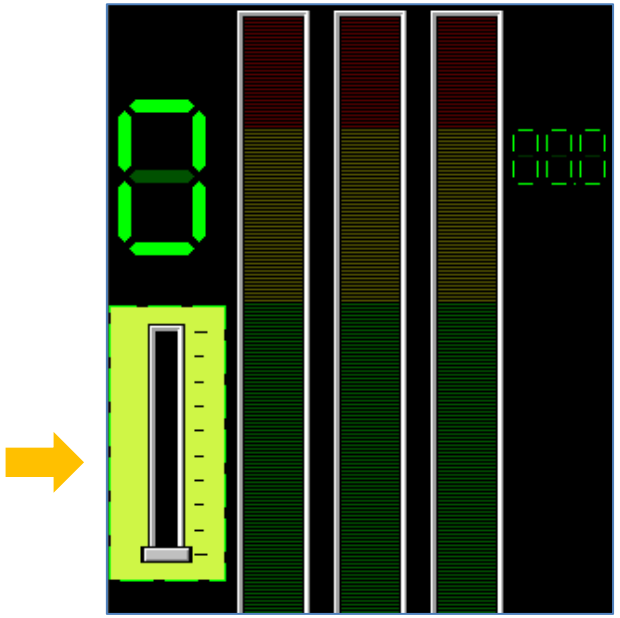
The object can be dragged to any desired location and resized to the desired dimensions. The track bar object used in this example has the following properties.



Take note of the **onChanging** event property – it is set to "**Report Message**". With this configuration, the display will send a message to the Arduino host when Trackbar0 is touched.

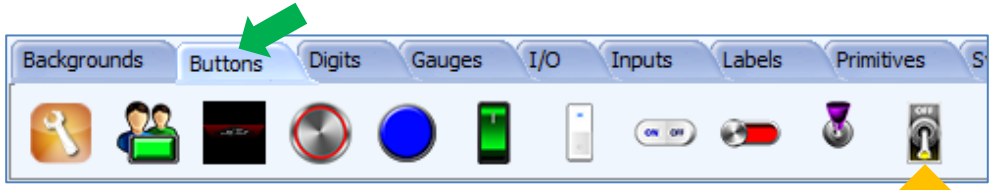


To know more about the OnChanged and OnChanging event properties, read ViSi-Genie onChanging and onChanged Events. When done, the track bar object will look like as shown below.

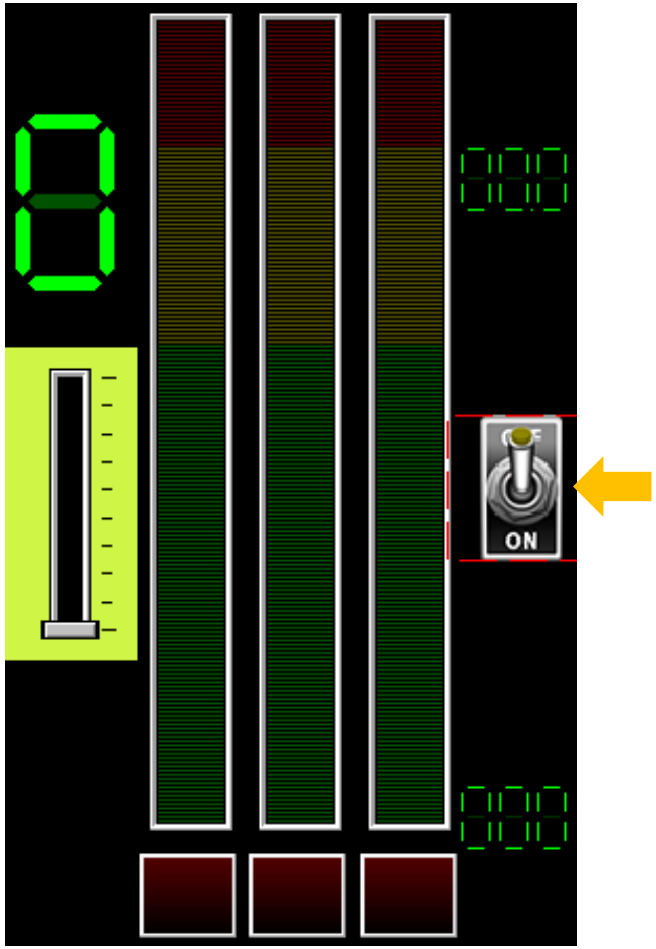


Add a 4D Button

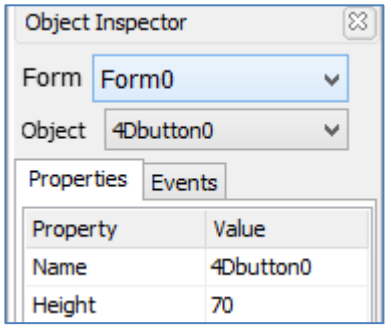
To add a 4D button object, go to the **Buttons** pane and click on one of the 4D button icons. The fourth to eleventh icons are all 4D button objects. Select the Toggle02 type.

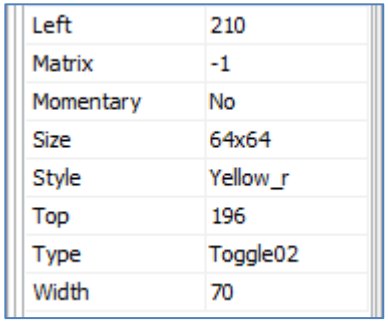


Click on the WYSIWYG screen to place the object.

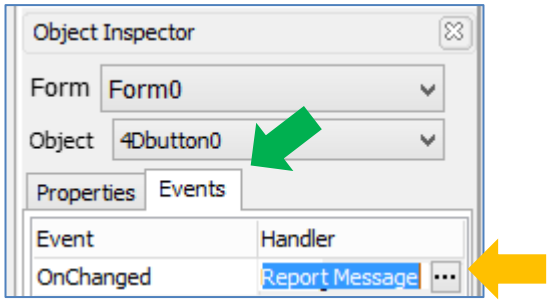


4Dbutton0 of this project has the following properties.

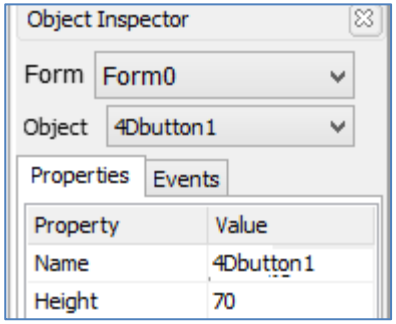


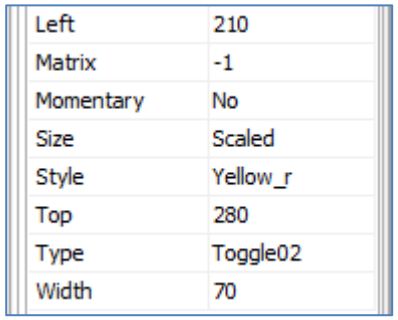


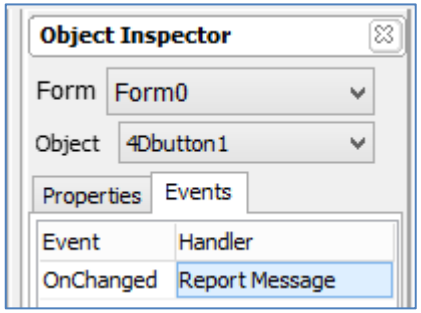
Take note of the **onChanged** event property – it is set to "**Report Message**". With this configuration, the display will send a message to the Arduino host when 4Dbutton0 is touched.



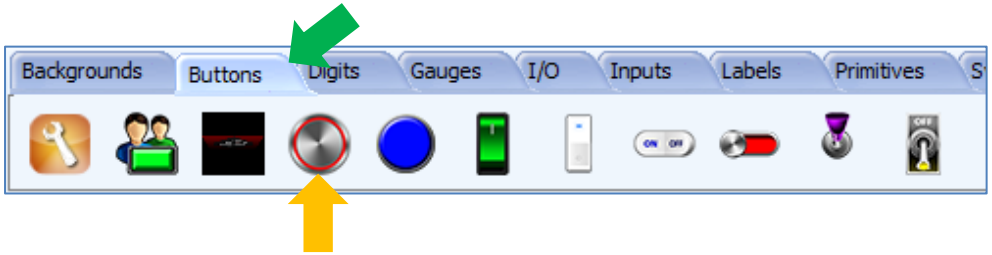
Add two more 4D button object to the screen. **4Dbutton1** of this project has the following properties.



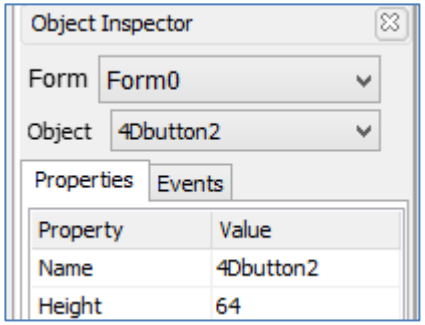


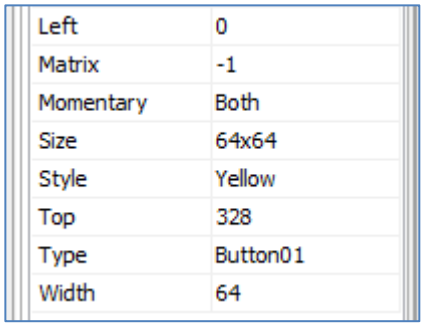


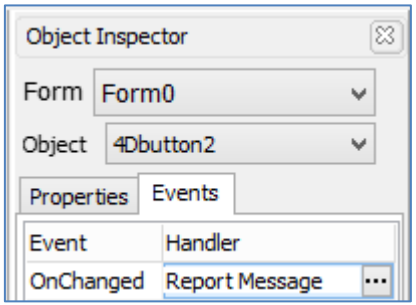
4Dbutton2 has the following properties. Note that 4Dbutton2 is a Button01 type.



Similar to the DIP switch example, the code above displays the rocker switch at state 0 and then at state 1.





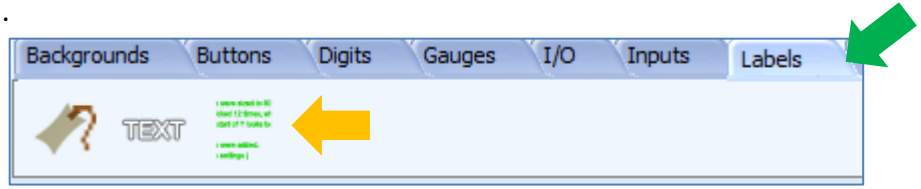


Now there are three 4D button objects, each of which will report a message to the host when touched.

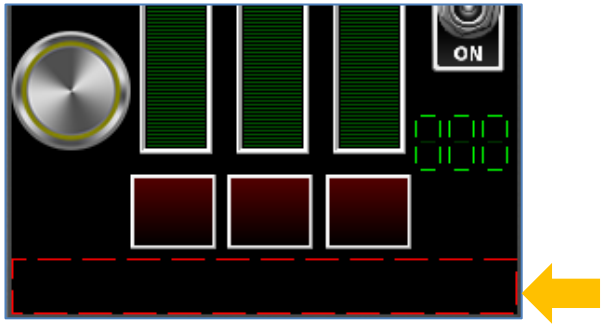
To know more about buttons, read the application notes ViSi-Genie User Button, ViSi-Genie Animated Button, and ViSi-Genie 4D Buttons.

Add a Strings Object

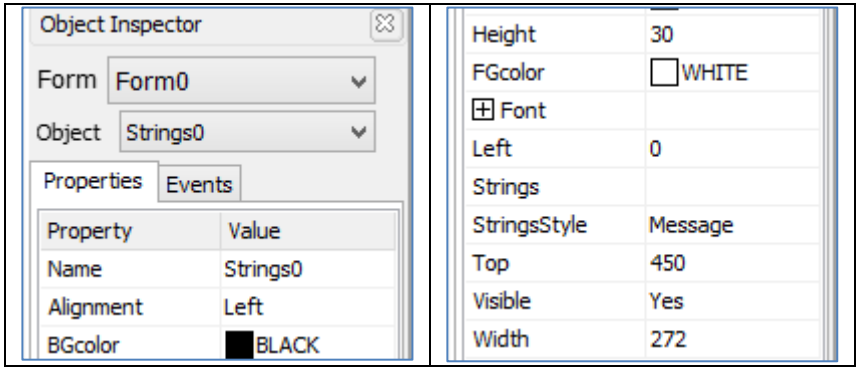
To add a strings object, go to the Labels pane and click on the strings object icon.



Click on the WYSIWYG screen to place the object.

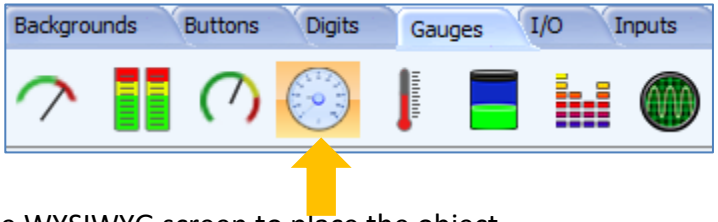


The strings object used in this project has the following properties.

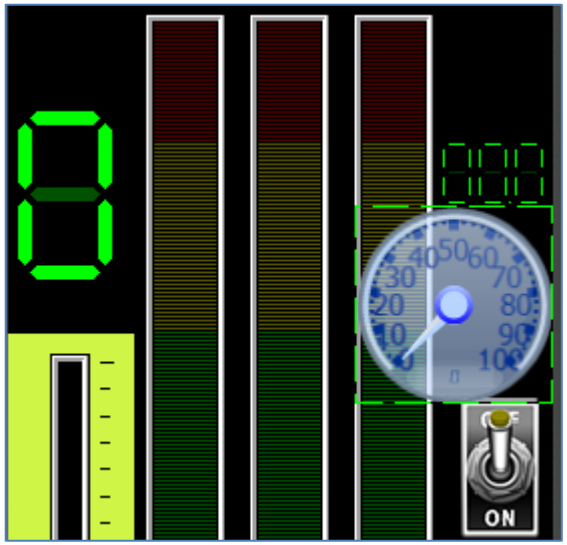


Add a Cool Gauge

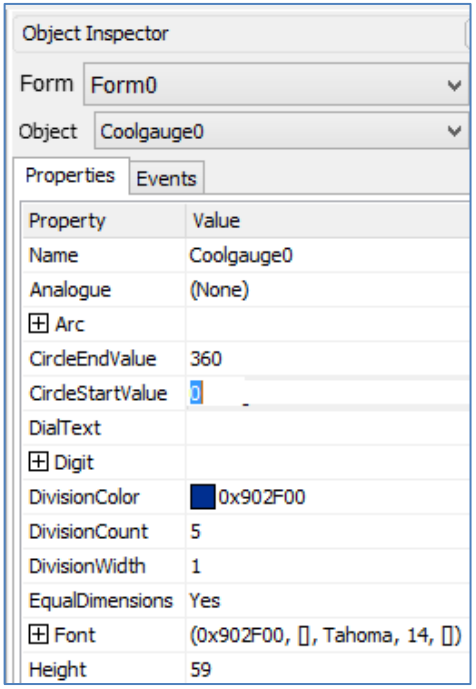
To add a cool gauge object, go to the Gauges pane and click on the cool gauge object icon.

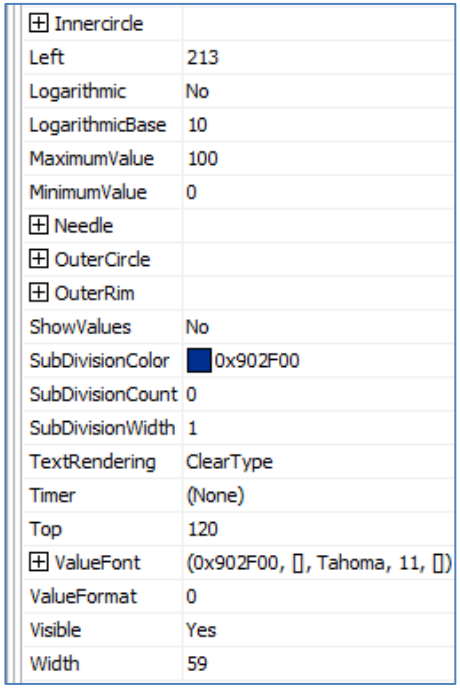


Click on the WYSIWYG screen to place the object.



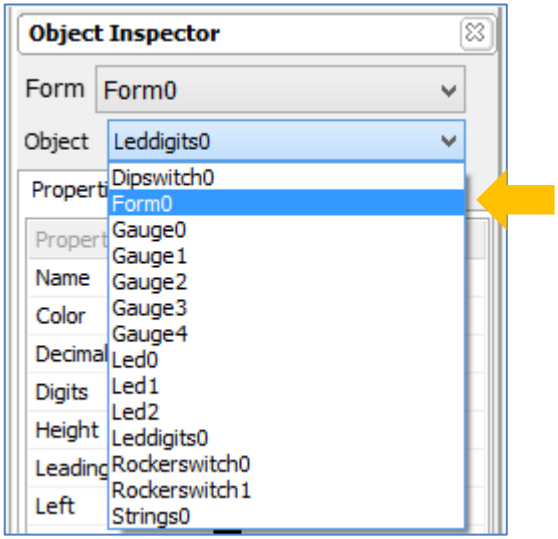
The cool gauge object used in this project has the following properties. Open the attached project to see the full list of property values. It is also possible to copy the attached project's Coolgague0 object by selecting it, pressing **Ctrl + C**, and pressing **Ctrl + V** on the destination form.



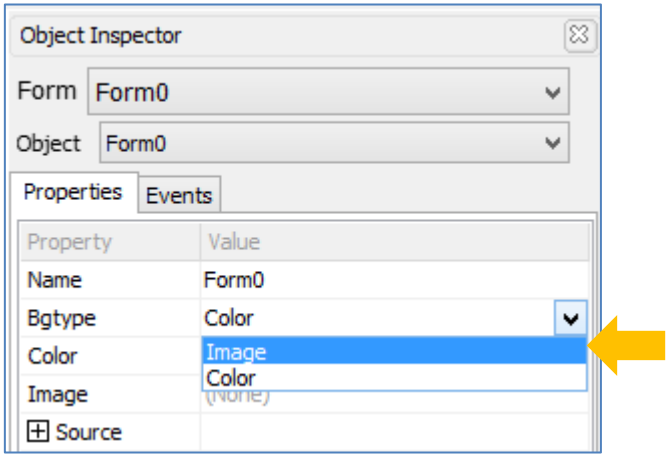


Add a Background Image

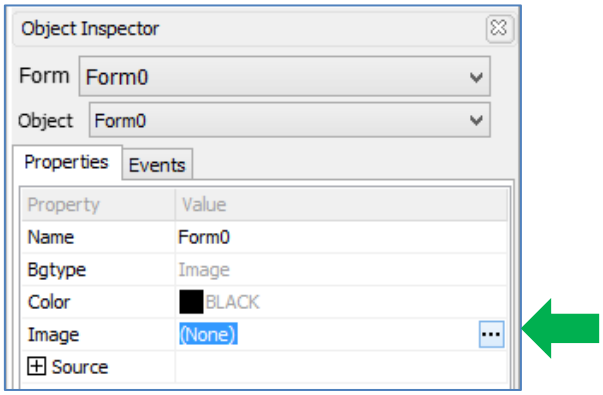
To add a background image, go to the Object inspector and select Form0 on the drop-down list.



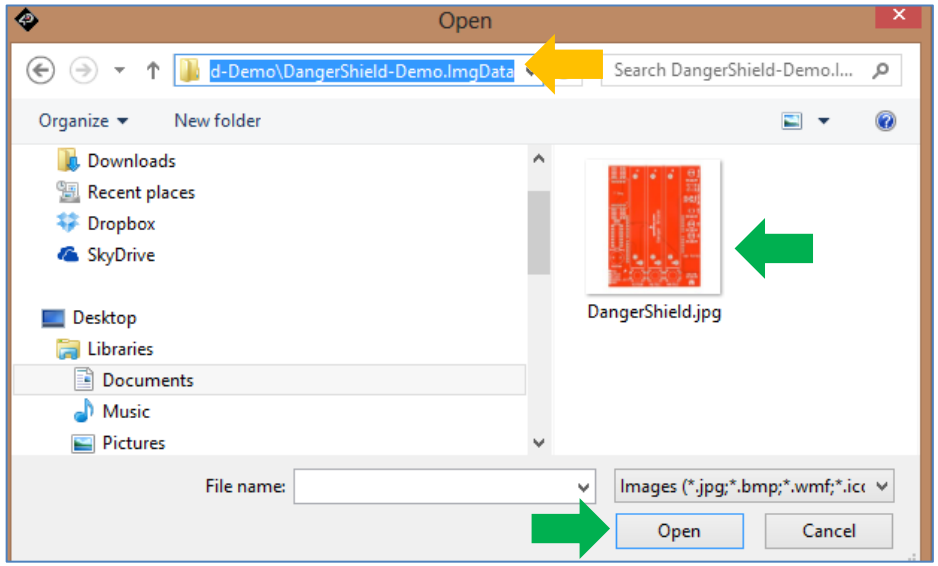
Set the property **Bgtype** to **Image**.



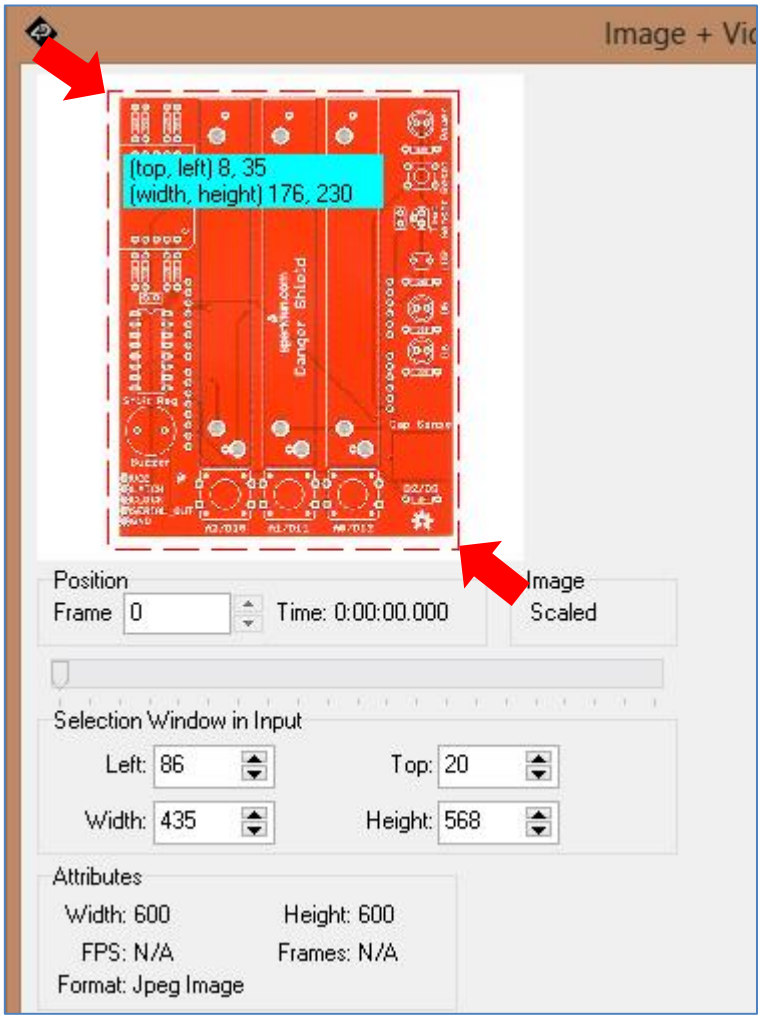
Click on the ellipsis dots of the Image line.



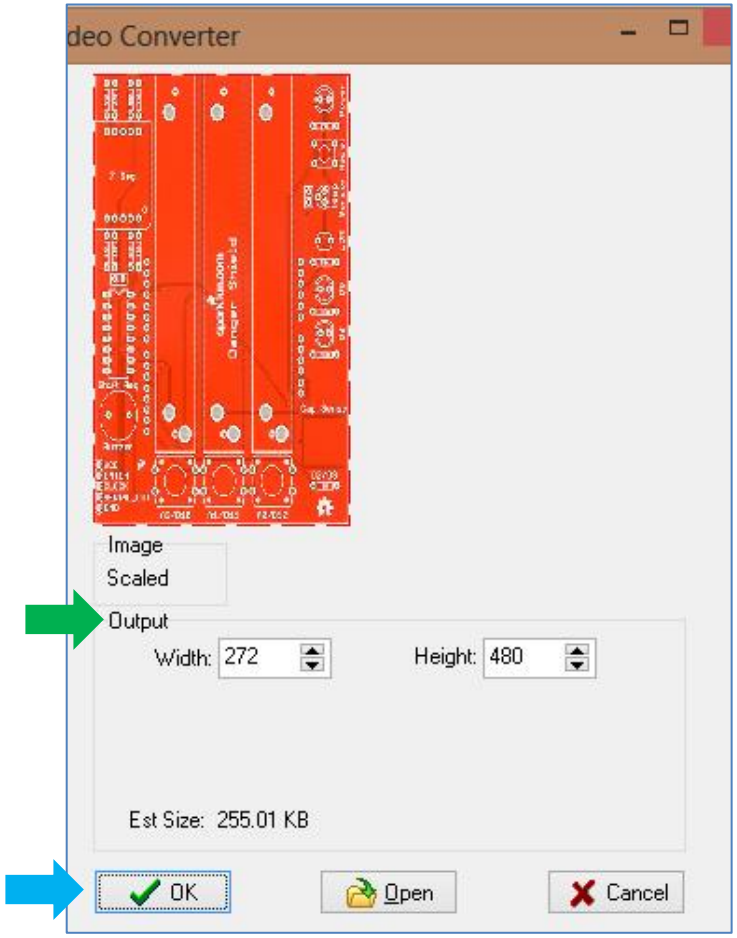
A standard Open window appears. Select the file to be used as a background image. The background image for this demo is found inside the "<demoFileName>.**ImgData**" folder.



The Image + Video Converter window appears. To remove the white background of the original image, crop the image by resizing the red box of the input window.



The output image now looks better. Click OK.



The code above will make Customdigits0 change its state from 0 to 99. To learn how to create a custom digits object, open the ViSi sample program in Workshop under File menu – Samples – Picaso ViSi – CLOCK. The block comment discusses how the bitmap image of the digits was created.

The project is now complete. Reposition the objects if necessary.

Build and Upload the Project

For instructions on how to build and upload a ViSi-Genie project to the target display, please refer to the section "**Build and Upload the Project**" of the application note

ViSi Genie Getting Started – First Project for Picaso Displays (for Picaso)
or
ViSi Genie Getting Started – First Project for Diablo16 Displays (for Diablo16).

The uLCD-32PTU and/or the uLCD-35DT display modules are commonly used as examples, but the procedure is the same for other displays.

Program the Arduino Host

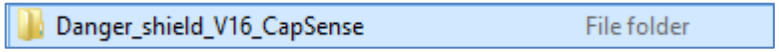
A thorough understanding of the application note **ViSi-Genie Connecting a 4D Display to an Arduino Host** is required before attempting to proceed further beyond this point. **ViSi-Genie Connecting a 4D Display to an Arduino Host** provides all the basic information that a user needs to be able to get started with ViSi-Genie and Arduino. The following is a list of the topics discussed in **ViSi-Genie Connecting a 4D Display to an Arduino Host**.

- How to download and install the ViSi-Genie-Arduino library
- How to open a serial port for communicating with the display and how to set the baud rate
- The genieAttachEventHandler() function
- How to reset the host and the display
- How to set the screen contrast
- How to send a text string
- The main loop
- Receiving data from the display
- The use of a non-blocking delay in the main loop
- How to change the status of an object
- How to know the status of an object
- The user's event handler

Discussion of any of these topics is avoided in other ViSi-Genie-Arduino application notes unless necessary. Users are encouraged to read **ViSi-Genie Connecting a 4D Display to an Arduino Host** first.

For the CapSense library used with the Danger Shield, download the example code from the Sparkfun product page. At the lower part of the page click on the Example Code link.

The zip file contains the header and cpp files for the CapSense library. Install the library.



It is recommended that the user first ensures that the Danger Shield is working before interfacing it with a 4D display. For problems encountered with the Danger Shield, refer to the Sparkfun website.

Understanding the Demo Sketch

Open the **DangerShield_Demo** sketch attached to this document. Note that comments have been added to the code. Additional explanations are now given below.

Reset the Arduino Host and the Display

To ensure that the display has properly started up before the Arduino host starts sending commands, the routine below resets the display and waits for five seconds for the display to boot up.

```
pinMode (A5, OUTPUT);
digitalWrite (A5, LOW); //reset the display
delay(100);
digitalWrite (A5, HIGH);//unreset the display
delay(5000);
```

The program continues after the delay. Note that the reset routine above applies only if the reset pin of the display is connected to pin A5 (or any available GPIO) of the Arduino host using a one-kilo-ohm series resistor. The case is different with the sketch presented in Connecting a 4D Display to an Arduino Host, wherein the host uses either pin D2 or D4 to reset the display. Pin D2 is for the old 4D Arduino Adaptor Shield (Rev 1) and pin D4 is for the new 4D Arduino Adaptor Shield (Rev 2). Shown below is the reset routine used in Connecting a 4D Display to an Arduino Host.

```
//Reset the Display (change D4 to D2 if you have ori
pinMode(4, OUTPUT);  // Set D4 on Arduino to Output
digitalWrite(4, 1);  // Reset the Display via D4
delay(100);
digitalWrite(4, 0);  // unReset the Display via D4
```

The logic states for reset and unreset are reversed since the 4D Arduino Adaptor Shields use switching transistors.

Conflict in the Usage of Pins

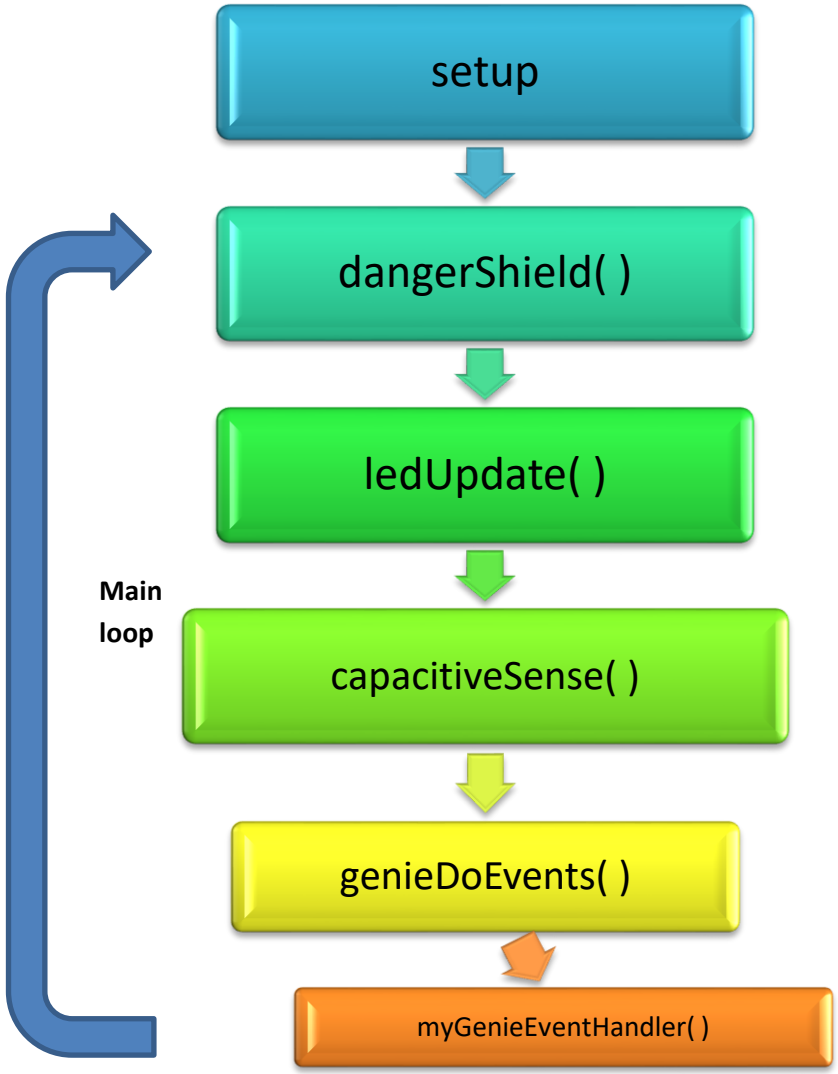
The first version of the 4D Arduino Adaptor Shield (Rev 1) uses pin D2 of the Arduino for resetting the display. However, the Danger Shield also uses pin D2 for the cap sense. As shown in the Sparkfun video, pin D2 of the 4D Arduino Adaptor Shield (Rev 1) was bent down against the board to disconnect it from the host. The host will therefore not reset the display when the program starts. The user has the option of tapping the bent pin of the 4D Arduino Adaptor Shield (Rev 1) to pin A5 of the host to make the reset routine work. Use a 1 kilo-ohm series resistor for this.

The second version of the 4D Arduino Adaptor Shield (Rev 2) uses pin D4 of the Arduino for resetting the display. However, the Danger Shield also uses pin D4 for sending data to the shift register. To prevent conflict, remove the connector of jumper J1 of the 4D Arduino Adaptor Shield (Rev 2) and tap the centre pin to pin A5 of the Arduino host, with a one kilo-ohm series resistor in between. This is shown in the next section “Set up the Project”.

If not using any of the 4D Arduino Adaptor Shields (Rev1 and Rev2) and if using jumper wires instead, connect the reset pin of the display to pin A5 with a one kilo-ohm series resistor and use the same reset routine.

Program Flow

The following diagram illustrates the flow of the program for this demo.



Setup

- Assignment of pins
- Declaration (and initialization) of variables
- Initialization of serial communication
- Reset routine
- Initialization of pin modes

dangerShield( )

- read values from the temperature sensor, LDR, sliders, and buttons of the Danger Shield
- send strings to the display to indicate the status of the Danger Shield buttons
- update the seven segment display of the Danger Shield using the value of the variable **`Digit`**.
- turn on or off the LEDS and the buzzer of the Danger Shield depending on the values of variables **`Led1`**, **`Led2`**, and **`Beep`**.

ledUpdate( )

- update the values of the gauges, LED digit objects, cool gauge, and user LEDs of the display depending on the values read from the corresponding components on the Danger Shield

capacitiveSense( )

- acquire capacitive sensor reading from the Danger Shield and write the value to Leddigits2 of the display

genieDoEvents( )

- buffer messages from the display
- call on myGenieEventHandler( ) if there are messages

myGenieEventHandler( )

- get a message from the buffer and evaluate it
- set the values of the variables **Led1**, **Led2**, **Beep,** and **Digit** and print some strings to indicate the status of 4Dbutton0, 4Dbutton1, 4Dbutton2, and Trackbar0 of the display

What follows is a list of instructions performed in each of these four subroutines - **dangerShield()**, **ledUpdate()**, **capacitiveSense(),** and **myGenieEventHandler()**. Variable names are in **bold font** so as not to be confused with the actual names of components of the Danger Shield and objects on the 4D display.

dangerShield()

- read an analogue value from the temperature sensor. Read value is assigned to the variable **Temp**.
- **Temp** is converted to the equivalent voltage reading **Temp_mV** using the 5V AREF pin of the Arduino.
- **Temp_mV** is converted to the equivalent temperature reading in degrees Celsius, **TempC**.
- **TempC** is converted to an integer, **Temp**.
- read an analogue value from the LDR. Read value is assigned to **Light**.
- read analogue values from Slider1, 2, and 3. Invert the values.
- inverted values are assigned to the variables **Slider1**, **Slider2**, and **Slider3**.
- if there is a change in the value of **Digit**, update the seven segment display.
- read the state of Button1. Assign the value of the state to the variable **Button1**. If there is a change in the value of **Button1**, send a string to **Strings0** indicating the current state of Button1.
- read the state of Button2. Assign the value of the state to the variable **Button2**. If there is a change in the value of **Button2**, send a string to **Strings0** indicating the current state of Button2.
- read the state of Button3. Assign the value of the state to the variable **Button3**. If there is a change in the value of **Button3**, send a string to **Strings0** indicating the current state of Button3.
- turn on or off LED1 (depending on the value of the variable **Led1**)
- turn on or off LED2 (depending on the value of the variable **Led2**)
- turn on or off the buzzer (depending on the value of the variable **Beep**)

ledUpdate()

- write to Gauge0, 1, and 2 the values of **Slider1**, **2**, and **3**, respectively.
- write to Leddigits0 the value of **Digit**
- write to Coolgauge0 the value of **Light**
- write to Leddigits1 the value of **Temp**
- turn on or off Userled0 (depending on the inverted value of **Button1**)
- turn on or off Userled1 (depending on the inverted value of **Button2**)
- turn on or off Userled2 (depending on the inverted value of **Button3**)

capacitiveSense ()

- acquire capacitive sensor reading, constrain it to a certain range, and assign it to the variable **total**.
- write to Leddigits2 the value of **total**.

myGenieEventHandler()

- get a message or an event from the buffer and evaluate it
- if the event is a REPORT EVENT from 4Dbutton0 (note that 4Dbutton0 was configured to report a message when touched)
    - get **data*** of the event and assign this to the variable **Led1**
    - write a message to Strings0 indicating the status of 4Dbutton0
- if the event is a REPORT EVENT from 4Dbutton1 (note that 4Dbutton1was configured to report a message when touched)
    - get **data*** of the event and assign this to the variable **Led2**
    - write a message to Strings0 indicating the status of 4Dbutton1
- if the event is a REPORT EVENT from 4Dbutton2 (note that 4Dbutton2 was configured to report a message when touched)
    - get **data*** of the event and assign this to the variable **Beep**
    - write a message to Strings0 indicating the status of 4Dbutton2
- if the event is a REPORT EVENT from Trackbar0 (note that Trackbar0 was configured to report a message when touched)
    - get **data*** of the event and assign this to the variable **Digit**

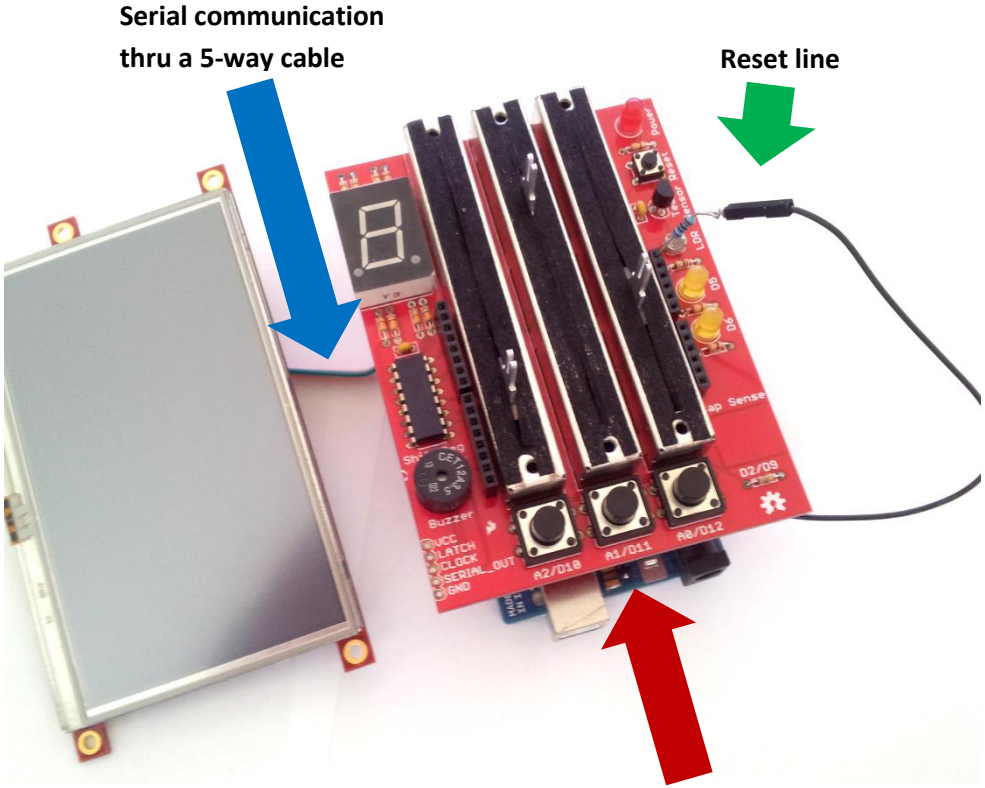
A complete message from the display contains six or more bytes. These bytes represent the type of command or event, type of object, index of object, parameters, and checksum. The "data" of a message is taken from its parameter bytes. For two-state objects such as a DIP switch, the data is '0' if it is off and '1' if it is on. For multi-state objects such as a track bar having 100 states or frames, the data can be an integer of any value from 0 to 99. Refer to the Visi-Genie reference manual for more information.

Set Up the Project

Refer to the section "**Connect the Display Module to the Arduino Host"** of the application note "**ViSi-Genie Connecting a 4D Display to an Arduino Host**" for the following topics:

- Using the New 4D Arduino Adaptor Shield (Rev 2.00)
  - Definition of Jumpers and Headers
  - Default Jumper Settings
  - Change the Arduino Host Serial Port
  - Power the Arduino Host and the Display Separately
- Using the Old 4D Arduino Adaptor Shield (Rev 1)
- Connection Using Jumper Wires
- Changing the Serial port of the Genie Program
- Changing the Maximum String Length

The Complete Project



The 4D Arduino Adaptor shield is under the Danger Shield and on top of the Arduino Uno

Proprietary Information

Disclaimer of Warranties & Limitation of Liability