APPLICATION NOTES

4D SYSTEMS

TURNING TECHNOLOGY INTO ART

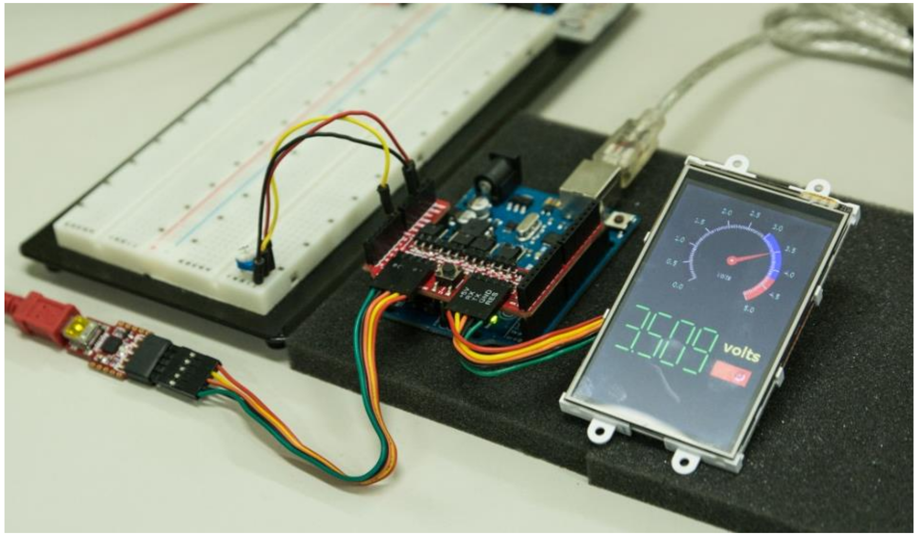
ViSi-Genie A Simple Digital Voltmeter Application using an Arduino Host

DOCUMENT DATE:	29th April 2019
DOCUMENT REVISION:	1.1

WWW.4DSYSTEMS.COM.AU

Description

This application note explains how to use a 4D display module in displaying voltage readings received from a host controller. The host is an AVR-ATmega328-microcontroller-based Arduino Uno board. The host can also be an Arduino Mega 2560. Ideally, the application described in this document should work with any Arduino board that tolerates an analog input voltage value from 0 to 5 volts and with at least one UART serial port. See specifications of Aduino boards here.



NOTE: Some Arduino boards such as the Due operate at 3.3 volts and tolerate analog input voltage levels from ground up to 3.3 V only. Applying more than 3.3 V may damage the chip. If attempting to use 3.3V Arduino boards for this application, make sure to limit the analog input voltage to 3.3 volts only and please read carefully your board's datasheet.

This application note requires:

- Any of the following 4D Picaso display modules:

  uLCD-24PTU
  uLCD-28PTU
  uLCD-32PTU
  uLCD-32WPTU
  uLCD-43(P/PT/PCT)
  uVGA-III

  and other superseded modules which support the ViSi Genie environment
- The target module can also be a Diablo16 display

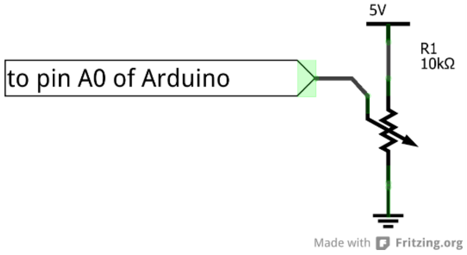
  uLCD-35DT
  uLCD-70DT

  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 or µUSB-PA5
- micro-SD (µSD) memory card
- Workshop 4 IDE (installed according to the installation document)
- Any Arduino board with a UART serial port
- 4D Arduino Adaptor Shield (optional) or connecting wires
- Arduino IDE
- 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

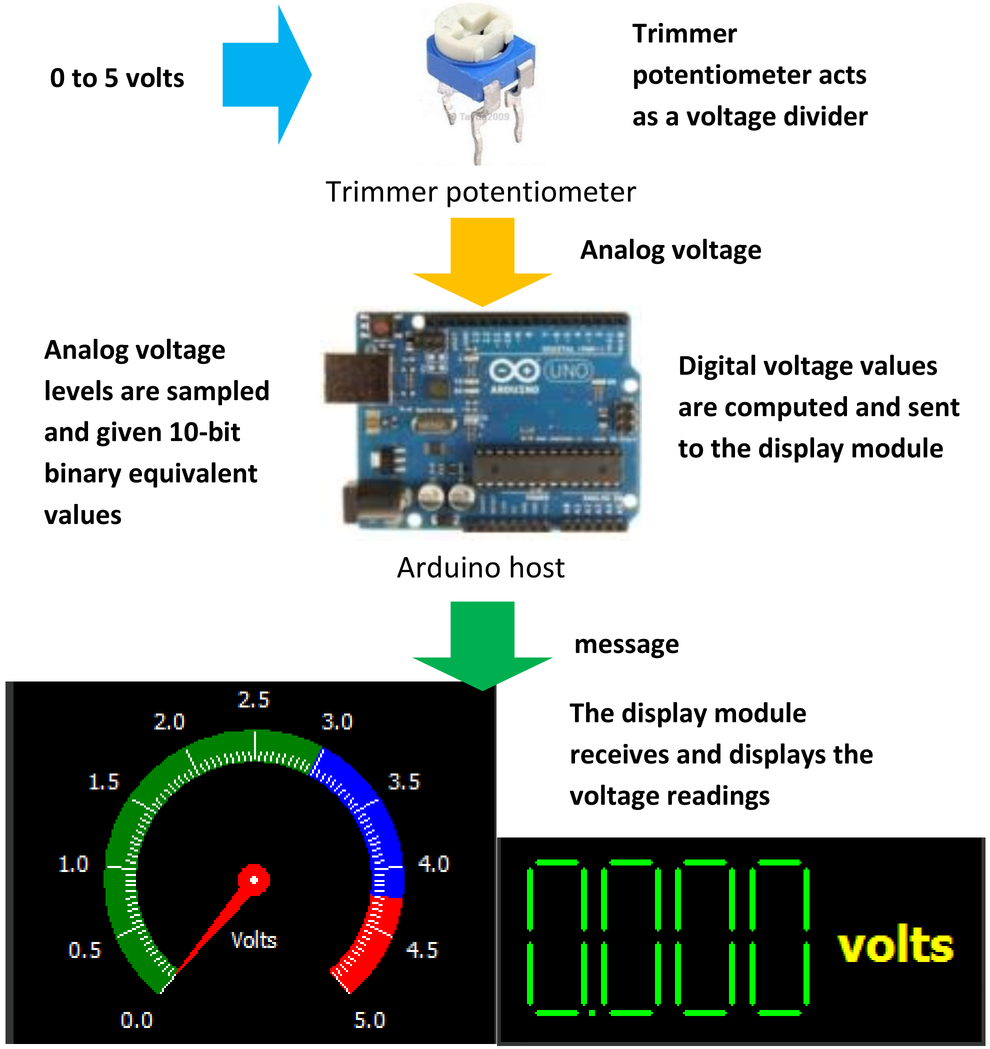
The application developed in this document works in a manner illustrated in the diagram shown on the next page. First, the wiper of a trimmer potentiometer, acting as a voltage divider, is connected to pin A0 of the Arduino Uno.



Thus, the voltage level at pin A0 can be of any value between 0 and 5 volts. Thru ADC, an analog voltage level sampled by pin A0 is given a 10-bit binary equivalent value. Given the reference voltage, which is 5 volts by default, the digital voltage value can now be computed. The digital voltage reading is then sent to the display module. **Note that the default range of allowable analog input voltage for the Uno and Mega 2560 is 0 to 5 volts only.**

The Arduino host is programmed in the Arduino IDE to perform ADC and send the voltage readings to the display module. The display module, on the other hand, is programmed in Workshop (ViSi Genie environment) to display the voltage readings.

This application note comes with a ViSi Genie program and an Arduino sketch. The process of creating the ViSi Genie program is first shown. Then the flow of the Arduino sketch is discussed. The sketch can be used to develop more complex applications related to ADC. The last section shows how the display module, Arduino host, and trimmer potentiometer are connected.



0 to 5 volts

Trimmer potentiometer acts as a voltage divider

Trimmer potentiometer

Analog voltage

Analog voltage levels are sampled and given 10-bit binary equivalent values

Digital voltage values are computed and sent to the display module

Arduino host

message

The display module receives and displays the voltage readings

Genie objects

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

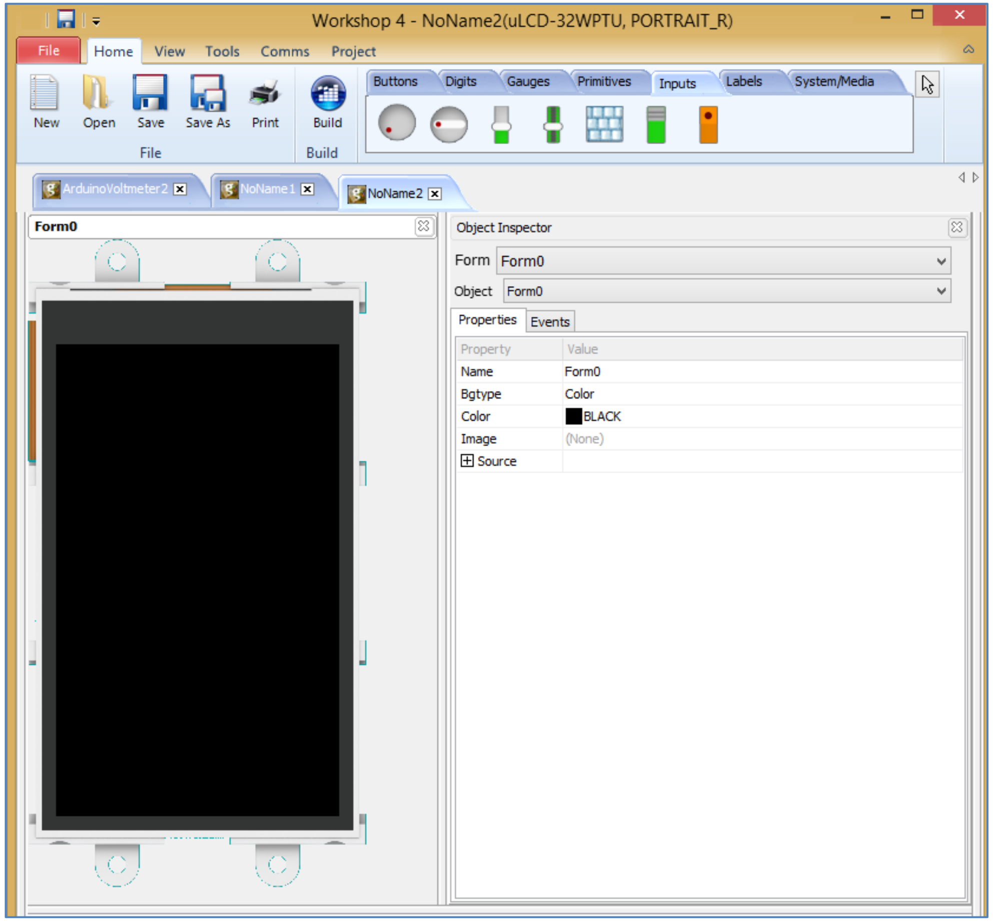
Create a New Project

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 Project

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. The form can contain **widgets** or **objects**, like sliders, displays or keyboards. Below is an empty form.



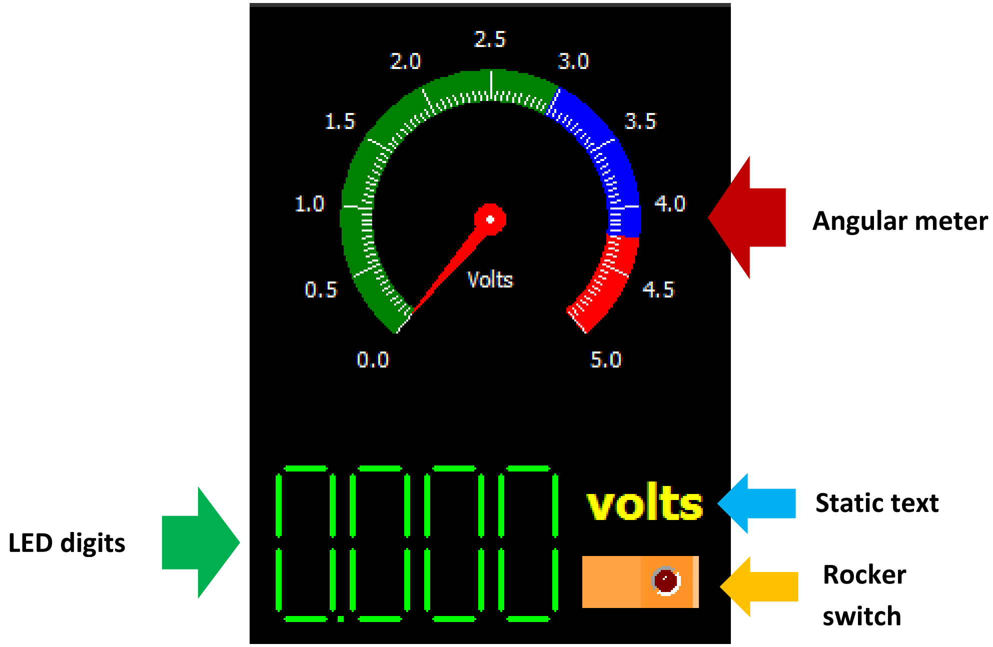
Form: Form0

Object: Form0

Properties | Events

Property	Value
Name	Form0
Bgtype	Color
Color	■ BLACK
Image	(None)
⊞ Source	

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



Angular meter

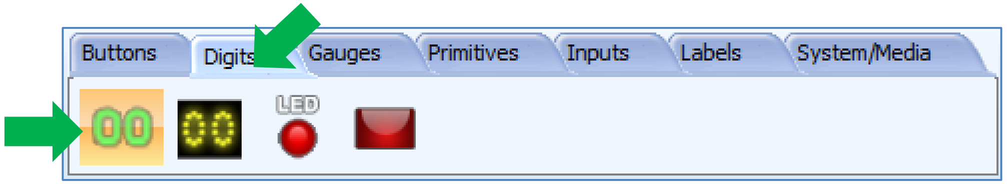
LED digits

Static text

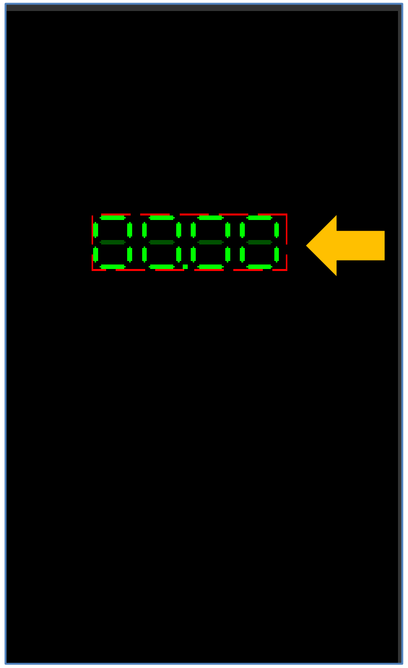
Rocker switch

Create a LED Digits Object

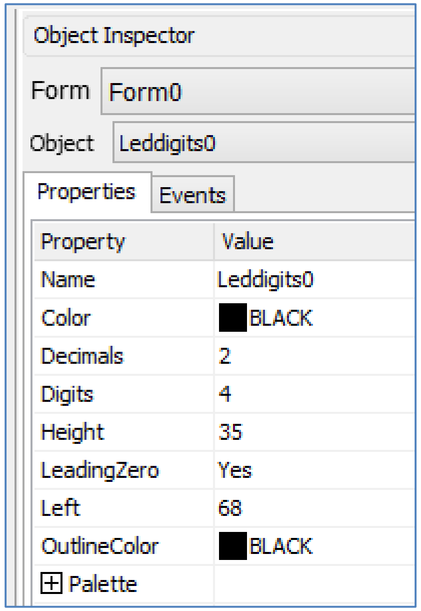
The **LED digits** object will display values received from the Arduino host. To add a LED digits object, go to the **Digits** pane and select the first icon.



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**.



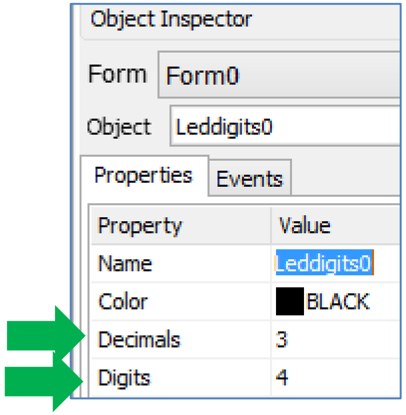
Object Inspector

Form Form0

Object Leddigits0

Property	Value
Name	Leddigits0
Color	■ BLACK
Decimals	2
Digits	4
Height	35
LeadingZero	Yes
Left	68
OutlineColor	■ BLACK
⊞ Palette	

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

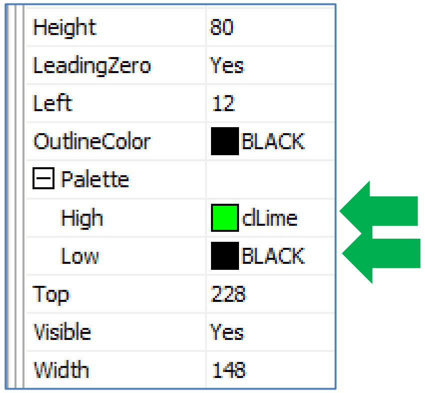


Object Inspector

Form Form0

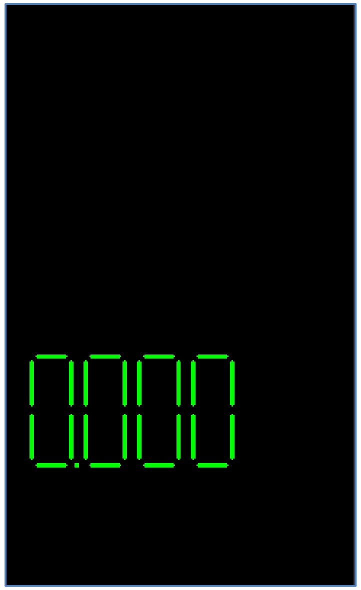
Object Leddigits0

Property	Value
Name	Leddigits0
Color	■ BLACK
Decimals	3
Digits	4



Height	80
LeadingZero	Yes
Left	12
OutlineColor	■ BLACK
⊟ Palette	
High	🟩 clLime
Low	■ BLACK
Top	228
Visible	Yes
Width	148

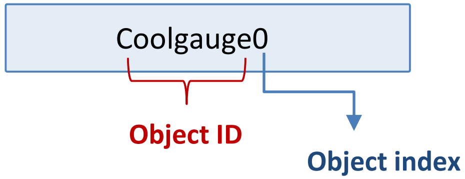
Apply the properties. The WYSIWYG screen is updated.



The object has four digits, three of which are to the left of the decimal point. The object is positioned near the bottom of the screen.

Naming of Objects

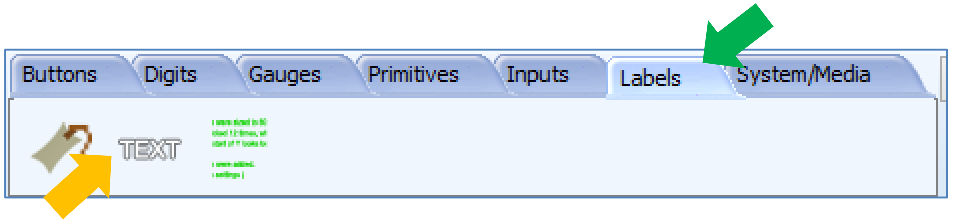
Naming is important to differentiate between objects of the same kind. For instance, suppose the user adds another LED digits object to the WYSIWYG screen. This object will be given the name Leddigits1 – it being the second LED digits in the program. The third LED digits object will be given the name 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.



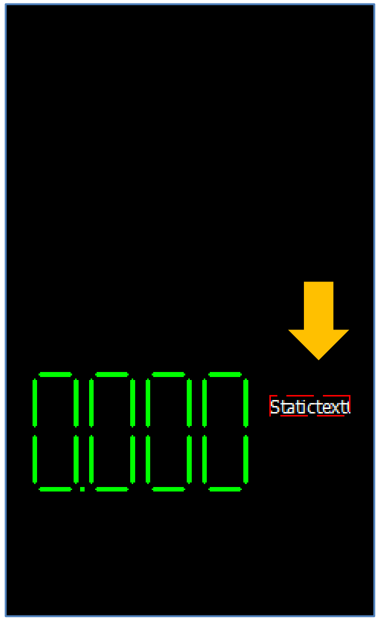
It is important to take note of an object's ID and index. When programming in the Arduino IDE, an object's status can be polled or changed if its ID and index are known. The process of doing this will be shown later.

Add a Static Text

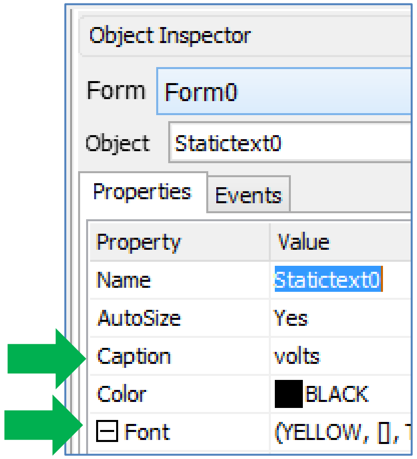
Static text objects are useful for labelling purposes. As the name implies, the status of these objects cannot be changed when the program runs. To add a static text, go to the Labels pane and click on the static text icon.



Click on the WYSIWYG screen to place the object.



In the Object Inspector, change the caption to volts, and increase the font size. The static text object used in this example has the following properties.



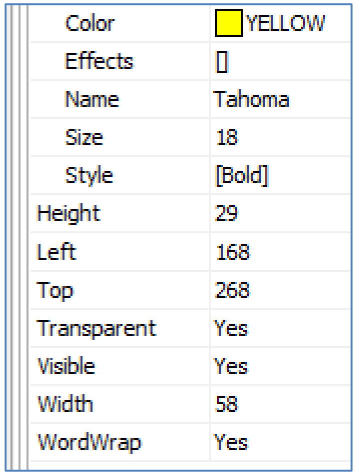
Object Inspector

Form Form0

Object Statictext0

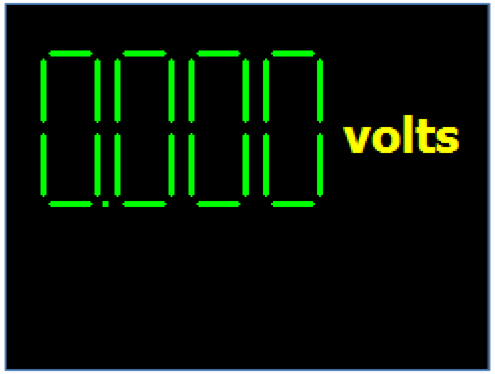
Properties Events

Property	Value
Name	Statictext0
AutoSize	Yes
Caption	volts
Color	⬛ BLACK
⊟ Font	(YELLOW, [], T



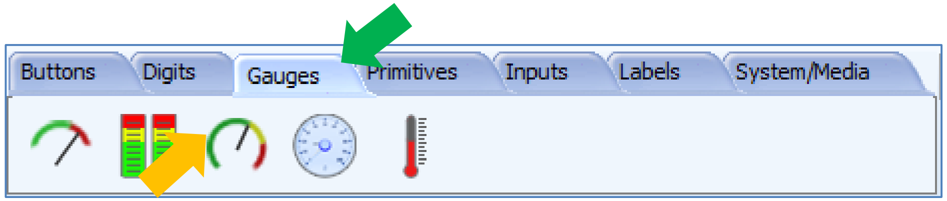
Color	🟨 YELLOW
Effects	[]
Name	Tahoma
Size	18
Style	[Bold]
Height	29
Left	168
Top	268
Transparent	Yes
Visible	Yes
Width	58
WordWrap	Yes

When done, the WYSIWYG screen should look similar to the one shown below.

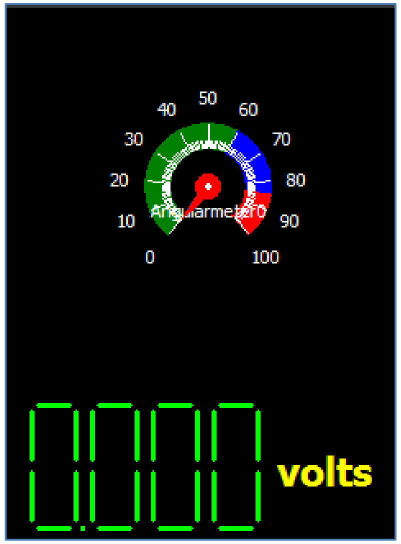


Add an Angular Meter

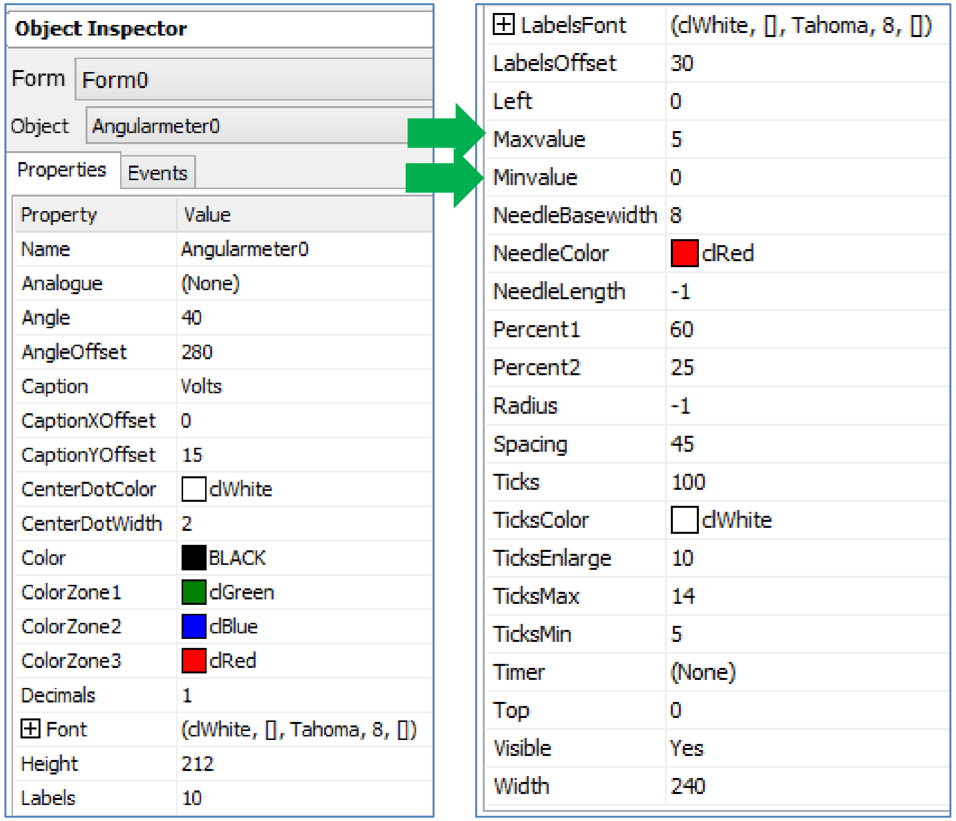
To demonstrate the use of meters and to give the application an “analog touch”, an angular meter will be added to the screen. Go to the gauges pane and click on the angular meter icon.



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 angular meter used in this example has the following properties.



Object Inspector

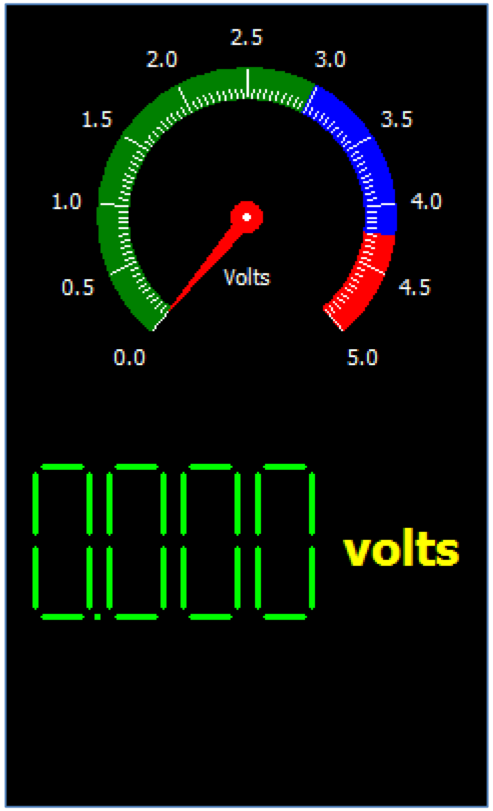
Form: Form0

Object: Angularmeter0

Properties | Events

Property	Value
Name	Angularmeter0
Analogue	(None)
Angle	40
AngleOffset	280
Caption	Volts
CaptionXOffset	0
CaptionYOffset	15
CenterDotColor	□ dWhite
CenterDotWidth	2
Color	■ BLACK
ColorZone1	🟩 dGreen
ColorZone2	🟦 dBlue
ColorZone3	🟥 dRed
Decimals	1
⊞ Font	(dWhite, [], Tahoma, 8, [])
Height	212
Labels	10
⊞ LabelsFont	(dWhite, [], Tahoma, 8, [])
LabelsOffset	30
Left	0
Maxvalue	5
Minvalue	0
NeedleBasewidth	8
NeedleColor	🟥 dRed
NeedleLength	-1
Percent1	60
Percent2	25
Radius	-1
Spacing	45
Ticks	100
TicksColor	□ dWhite
TicksEnlarge	10
TicksMax	14
TicksMin	5
Timer	(None)
Top	0
Visible	Yes
Width	240

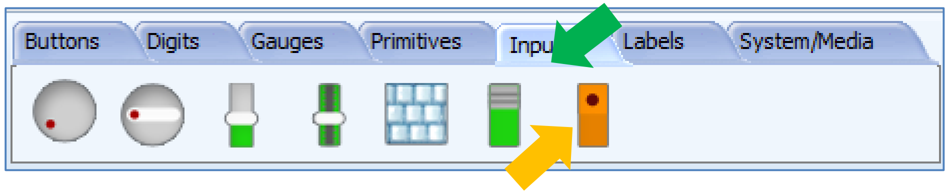
Take note of the decimals, maximum, and minimum values. When done, the form will look like the image shown to the right.



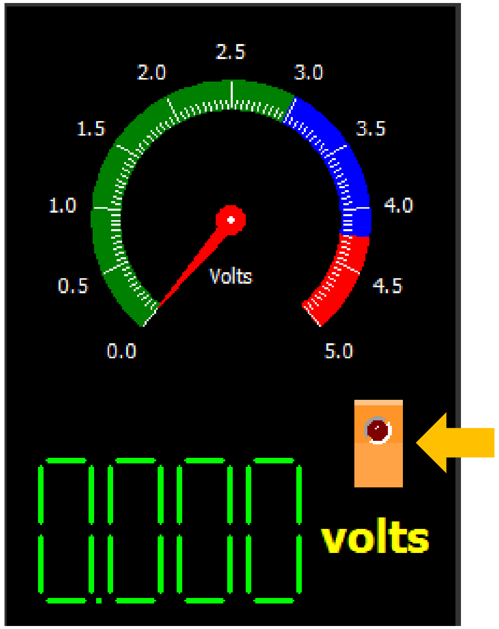
To know more about meters and gauges, read ViSi-Genie Gauges.

Add a Rocker Switch

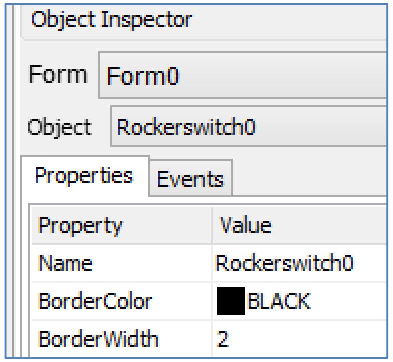
To demonstrate the use of an input widget and to make the application more interactive, a rocker switch will be added as a means to turn on and off the voltmeter. When the rocker switch is off, the angular meter and LED digits objects will display zero volts. This makes the application appear to be turned off. To add a rocker switch, go to the inputs pane and click on the rocker switch icon.



Click on the WYSIWYG screen to place the object.



The rocker switch used in this example has the following properties.



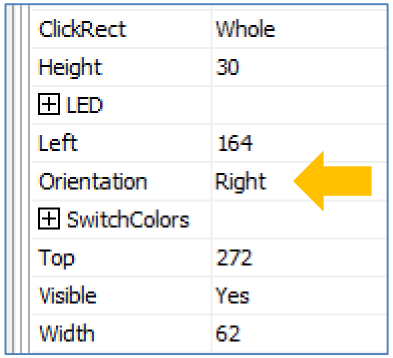
Object Inspector

Form: Form0

Object: Rockerswitch0

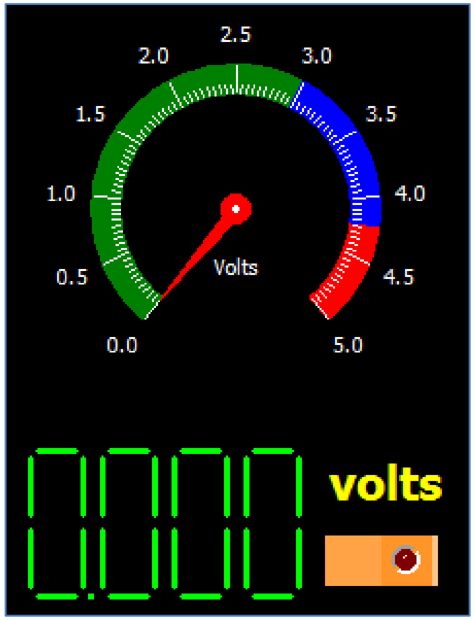
Properties | Events

Property	Value
Name	Rockerswitch0
BorderColor	■ BLACK
BorderWidth	2



ClickRect	Whole
Height	30
⊞ LED	
Left	164
Orientation	Right
⊞ SwitchColors	
Top	272
Visible	Yes
Width	62

The user may need to move the static text object. The final form is shown below.



To know more about inputs and switches, read ViSi-Genie Inputs.

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**.

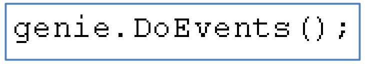
- How to download and install the ViSi-Genie-Arduino library
- How to open a serial port for communicating with the display and how to set the baud rate
- The genieAttachEventHandler() function
- How to reset the host and the display
- How to set the screen contrast
- How to send a text string
- The 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.

The Main Loop - Writing Data to the Display

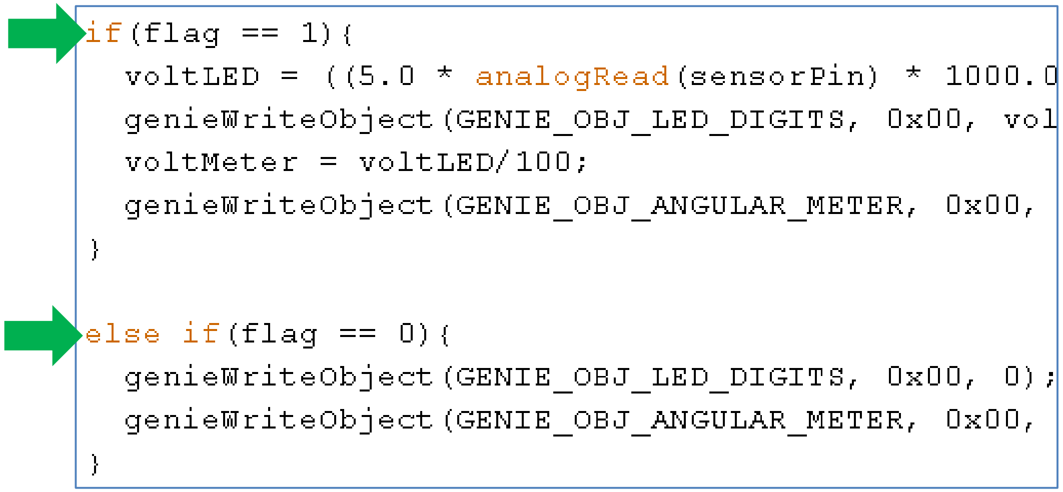
This and the following sections discuss how the main loop and the user's event handler work. It is strongly recommended that the user opens and analyses the accompanying Arduino sketch while reading this text.

In the main loop, the function below receives and queues the data received from the display.



```
genie.DoEvents();
```

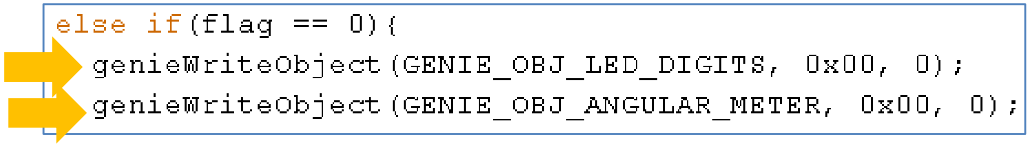
Initially, there is no data received from the display since none of the objects were configured to report an event. The program now evaluates if the voltmeter is turned on or off.



```
if(flag == 1){
  voltLED = ((5.0 * analogRead(sensorPin) * 1000.0
  genieWriteObject(GENIE_OBJ_LED_DIGITS, 0x00, vol
  voltMeter = voltLED/100;
  genieWriteObject(GENIE_OBJ_ANGULAR_METER, 0x00,
}

else if(flag == 0){
  genieWriteObject(GENIE_OBJ_LED_DIGITS, 0x00, 0);
  genieWriteObject(GENIE_OBJ_ANGULAR_METER, 0x00,
}
```

Since **`flag`** is initially set to **0**, the following statements are executed.

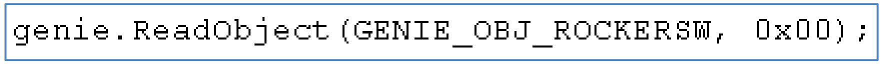


```
else if(flag == 0){
  genieWriteObject(GENIE_OBJ_LED_DIGITS, 0x00, 0);
  genieWriteObject(GENIE_OBJ_ANGULAR_METER, 0x00, 0);
```

These lines send or write the value **0** to Leddigits0 and Angularmeter0. The voltmeter application is "off" in this case.

The Main Loop – Interrogating the Display

After writing the value zero to Leddigits0 and Angularmeter0, the program executes the command

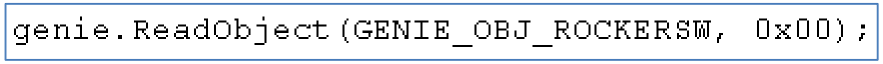


```
genie.ReadObject(GENIE_OBJ_ROCKERSW, 0x00);
```

This command sends a message to the display inquiring for the status of Rockerswitch0. The display will respond with the appropriate message, which will be received at the next iteration of the loop. The program now goes back to the start of the loop.

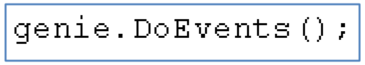
The Main Loop – Receiving Data from the Display

In the second execution of the main loop, data is received from the display as a result of the statement



```
genie.ReadObject(GENIE_OBJ_ROCKERSW, 0x00);
```

The data or message is received and queued by

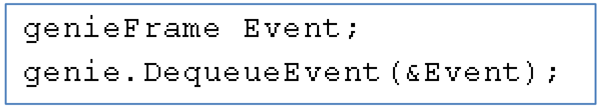


```
genie.DoEvents();
```

Another function (to be written by the user) is needed to process the received data. This function is the user's event handler, which was arbitrarily given the name **myGenieEventHandler()**. This function is called from inside the function **genie.DoEvents()**.

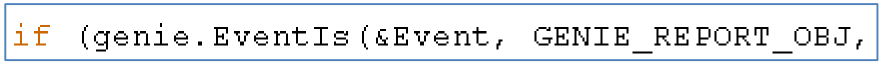
The User's Event Handler

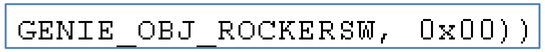
The user's event handler now takes out a message from the events queue and evaluates it. Hence, the following lines:



```
genieFrame Event;
genie.DequeueEvent(&Event);
```

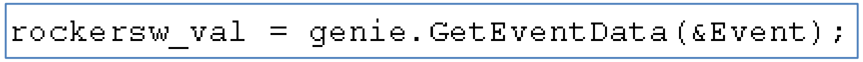
The message is then tested if it is a REPORT_OBJ event from Rockerswitch0. This is the expected response of the display to the interrogation message sent earlier.





```
if (genie.EventIs(&Event, GENIE_REPORT_OBJ,
                          GENIE_OBJ_ROCKERSW, 0x00))
```

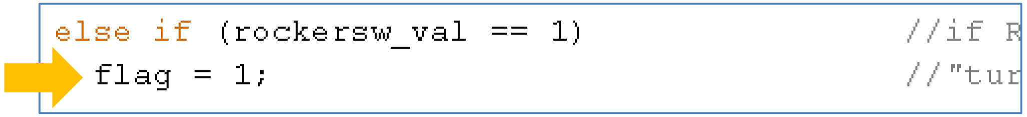
If the above condition is true, the value of the event is stored in **rockersw_val**.



```
rockersw_val = genie.GetEventData(&Event);
```

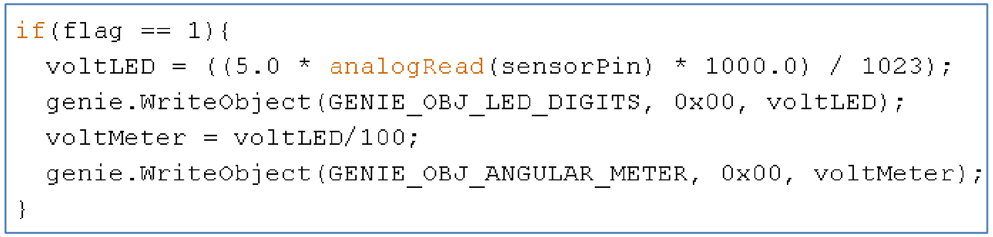
The value of an object event or message represents that object's status. For instance, the message for Rockerswitch0 will have a value of 0 when it is off, a value of 1 when it is on. This concept is applicable to widgets or objects having only two states such as the DIP switch (default mode), and the Winbutton (toggle mode).

Suppose the user has turned on Rockerswitch0 by touch, the statement below is executed.



```
else if (rockersw_val == 1)                    //if R
  flag = 1;                                    //"tur
```

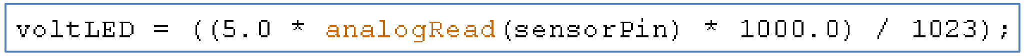
The program exits the user's event handler and goes back to the main loop. There it processes the block



```
if(flag == 1){
  voltLED = ((5.0 * analogRead(sensorPin) * 1000.0) / 1023);
  genie.WriteObject(GENIE_OBJ_LED_DIGITS, 0x00, voltLED);
  voltMeter = voltLED/100;
  genie.WriteObject(GENIE_OBJ_ANGULAR_METER, 0x00, voltMeter);
}
```

since **flag** has just been set to **1**.

The line



```
voltLED = ((5.0 * analogRead(sensorPin) * 1000.0) / 1023);
```

performs the conversion below.

$$voltLED = sensorPin\ levels\ x\ \frac{5\ volts}{1023\ levels}\ x\ 1000$$

ADC resolution of the Arduino Uno

The variable **voltLED** will hold a value between 0 and 5000, which will be sent to Leddigits0.

```
genie.WriteObject(GENIE_OBJ_LED_DIGITS, 0x00, voltLED);
```

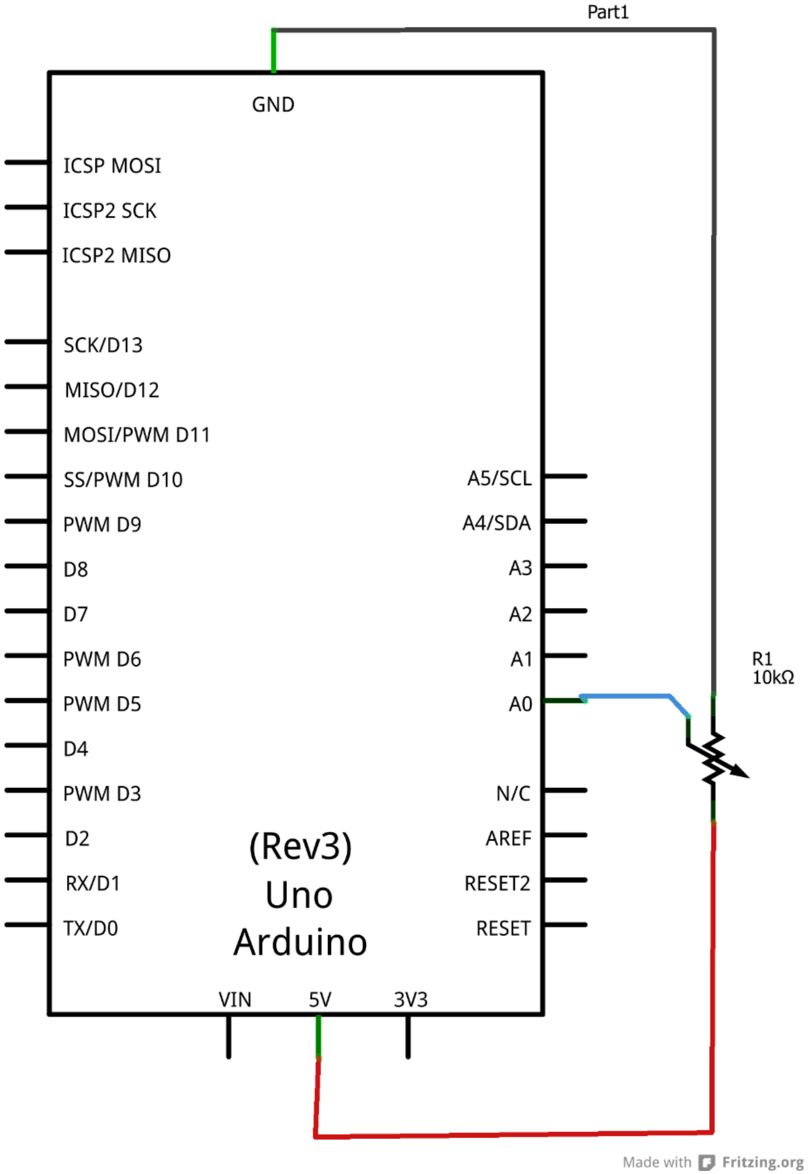
Remember that Leddigits0 has four digits so it can display 0000 to 5000. Now, since three of the four digits are to the right of the decimal point, it will display 0.000 to 5.000.
Angularmeter0, on the other hand, is configured to display 0.0 to 5.0. For more information on writing to Genie objects, refer to ViSi-Genie Writing to Genie Objects Using an Arduino Host.

Set Up the Project

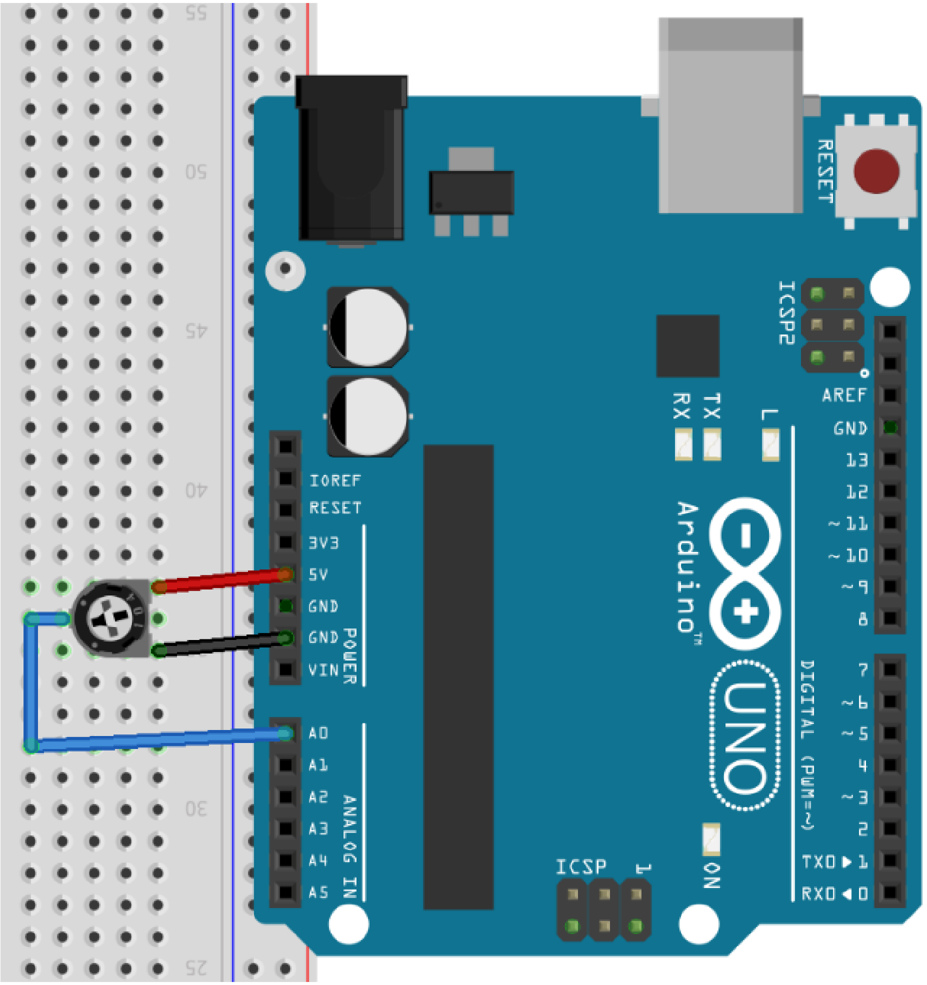
Refer to the section "**Connect the Display Module to the Arduino Host"** of the application note "**ViSi-Genie Connecting a 4D Display to an Arduino Host**" for the following topics:

- 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

Connect the Trimmer Potentiometer to the Arduino

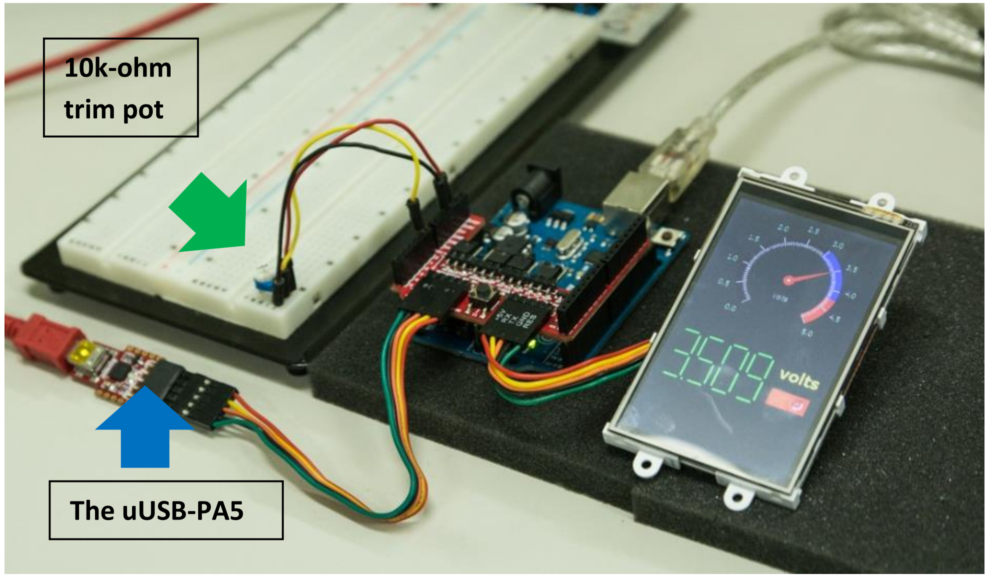


Schematic for trimmer potentiometer-to-Arduino-Uno connection

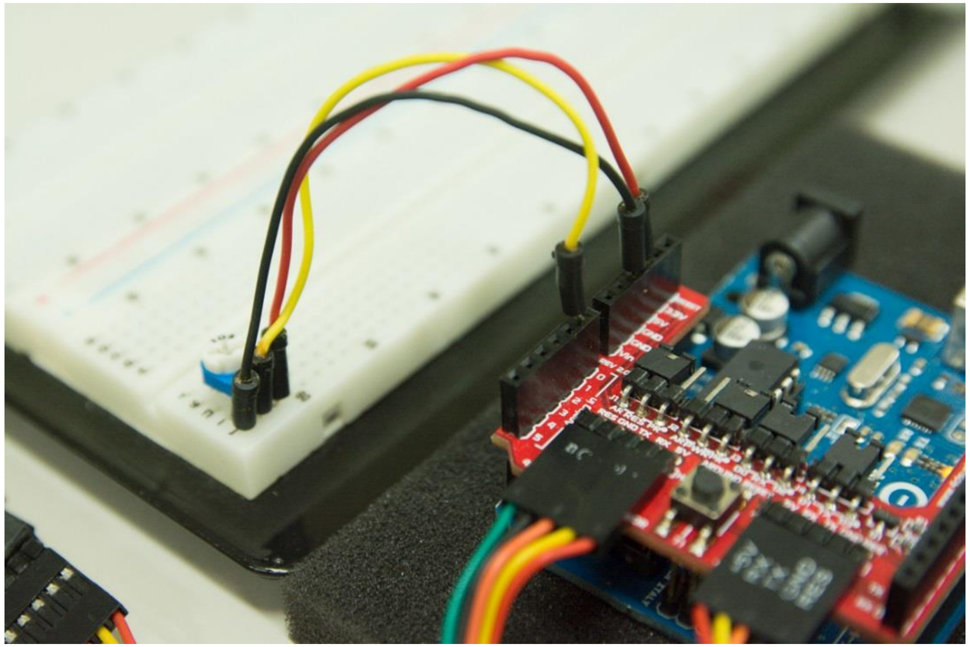


Breadboard layout for trimpot- to-Arduino-Uno connection (made with Fritzing)

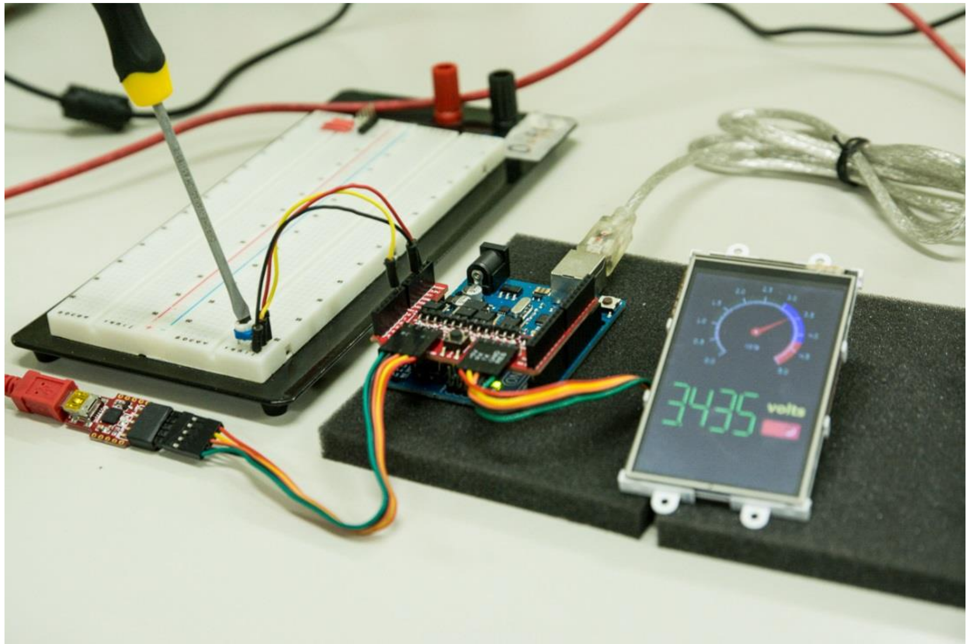
The Complete Project

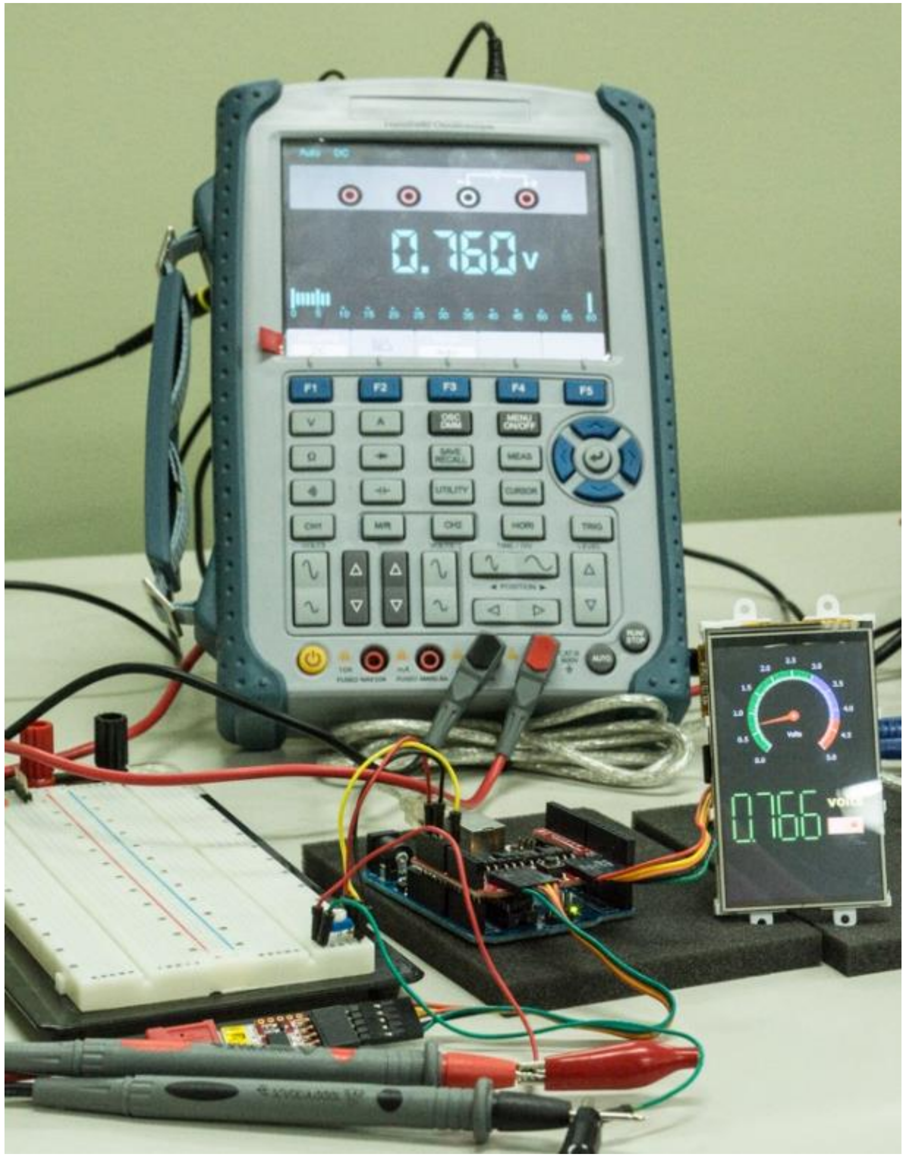


The uUSB-PA5 programming adaptor is used here only to power the display.



Adjusting the trimmer potentiometer.





Proprietary Information

The information contained in this document is the property of 4D Systems Pty. Ltd. and may be the subject of patents pending or granted, and must not be copied or disclosed without prior written permission.

4D Systems endeavours to ensure that the information in this document is correct and fairly stated but does not accept liability for any error or omission. The development of 4D Systems products and services is continuous and published information may not be up to date. It is important to check the current position with 4D Systems.

All trademarks belong to their respective owners and are recognised and acknowledged.

Disclaimer of Warranties & Limitation of Liability

4D Systems makes no warranty, either expresses or implied with respect to any product, and specifically disclaims all other warranties, including, without limitation, warranties for merchantability, non-infringement and fitness for any particular purpose.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications.

In no event shall 4D Systems be liable to the buyer or to any third party for any indirect, incidental, special, consequential, punitive or exemplary damages (including without limitation lost profits, lost savings, or loss of business opportunity) arising out of or relating to any product or service provided or to be provided by 4D Systems, or the use or inability to use the same, even if 4D Systems has been advised of the possibility of such damages.

4D Systems products are not fault tolerant nor designed, manufactured or intended for use or resale as on line control equipment in hazardous environments requiring fail – safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines or weapons systems in which the failure of the product could lead directly to death, personal injury or severe physical or environmental damage (‘High Risk Activities’). 4D Systems and its suppliers specifically disclaim any expressed or implied warranty of fitness for High Risk Activities.

Use of 4D Systems’ products and devices in 'High Risk Activities' and in any other application is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless 4D Systems from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any 4D Systems intellectual property rights.