APPLICATION NOTE

4D SYSTEMS

TURNING TECHNOLOGY INTO ART

Application Note: 4D-AN-P4017

ViSi Genie – Connecting a 4D Display to an Arduino Host

Document Date: 22nd July 2013

Document Revision: 1.0

Description

This Application Note explores the possibilities provided by ViSi-Genie to work with an Arduino host. In this example, the host is an AVR ATmega328 microcontroller-based Arduino Uno board. The host can also be an Arduino Mega 2560 or Due. Ideally, the applications described in this document should work with any Arduino board with at least one UART serial port. See specifications of Aduino boards here.

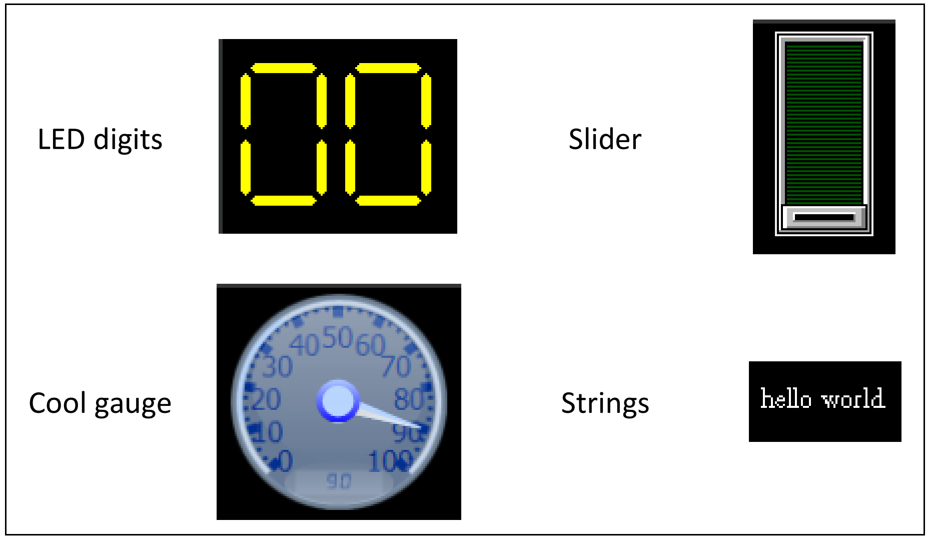
Before getting started, the following are required:

- Any of the following 4D Picaso display modules:
  uLCD-24PTU
  uLCD-28PTU
  uLCD-32PTU
  uLCD-32WPTU
  uLCD-43(P/PT/PCT)
  uVGA-III
  other superseded modules which support the ViSi Genie environment
- 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

Content

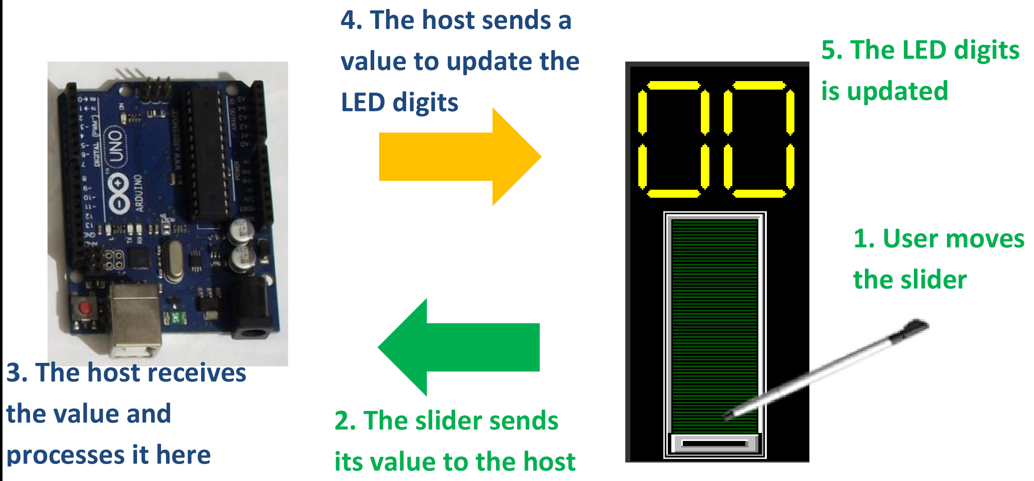
Application Overview

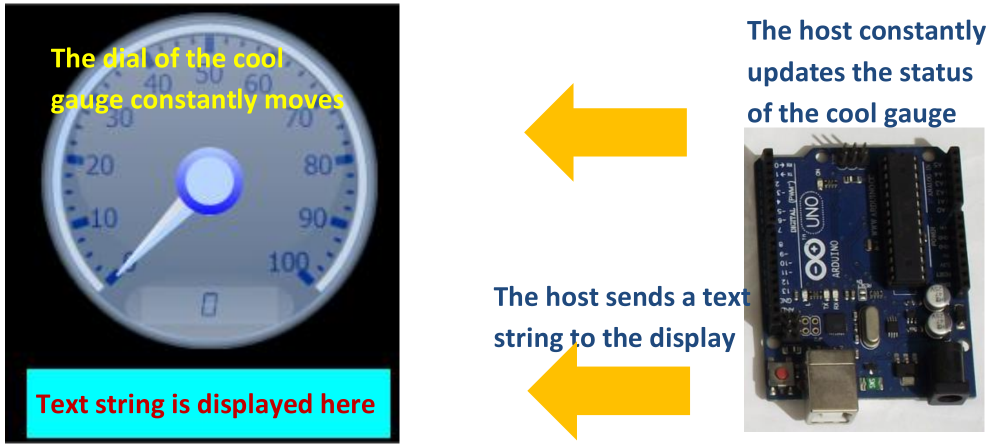
It is often difficult to design a graphical display without being able to see the immediate results of the application code. ViSi-Genie is the perfect software tool that allows users to see the instant results of their desired graphical layout with this large selection of gauges and meters (called widgets) that can simply be dragged and dropped onto the simulated module display. The following are examples of widgets or objects used in this application note.



The purpose of this application note is, besides showing the user how to create a ViSi Genie program, to introduce the use of the new ViSi Genie library for the Arduino IDE. To achieve this objective, a simple project is developed. This consists of a 4D Picaso module displaying four objects – a LED digits, a slider, a cool gauge, and a text string.

These objects interact with an Arduino host in a manner illustrated below.


4. The host sends a value to update the LED digits
5. The LED digits is updated
1. User moves the slider
3. The host receives the value and processes it here
2. The slider sends its value to the host


The dial of the cool gauge constantly moves
The host constantly updates the status of the cool gauge
The host sends a text string to the display
Text string is displayed here

First the user creates a ViSi Genie program in the 4D Workshop IDE and downloads it to a 4D display module. The user can also download and open the sample ViSi Genie program from here. The Arduino host, on the other hand, is programmed using the Arduino IDE. A library is provided by 4D Systems to make programming easier and straight forward.

Another intention of this application note is to guide the user in setting up a working system in the shortest possible time and to cover the important aspects in the simplest possible manner. Hence, for topics that require a lengthy explanation (which compromises brevity and simplicity), references are pointed out to the user for further study.

It is therefore hoped that this application note will serve as a quick yet effective starting guide for developers wanting to make use of 4D displays in developing Arduino-based applications.

Launch Workshop 4

There is a shortcut for Workshop 4 on the desktop.

Launch Workshop 4 by double-clicking on the icon.

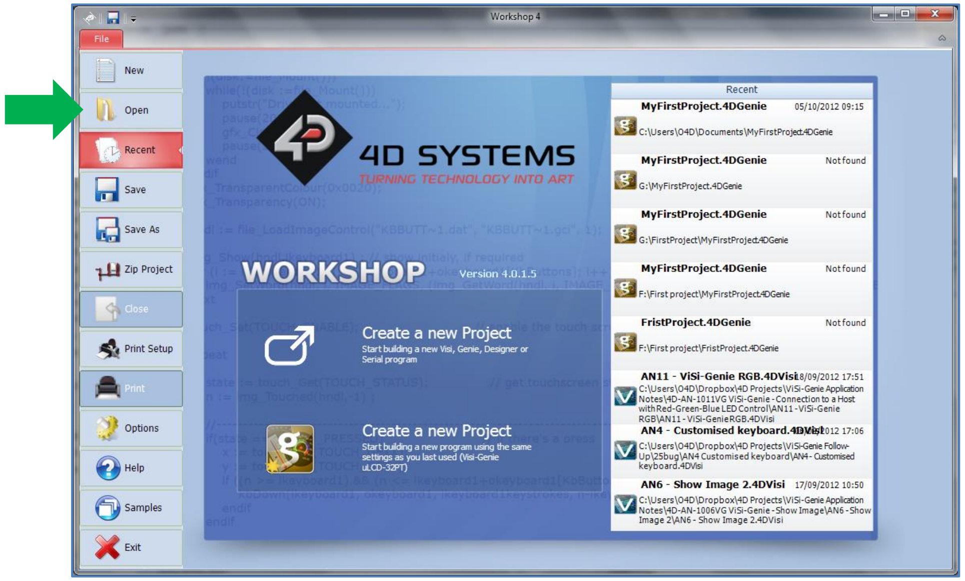
Setup Procedure

Load the Example

The user can download the ViSi-Genie project example from:
https://github.com/4dsystems/ViSi-Genie-Arduino-Library
The ViSi-Genie-Arduino library is also located in the same repository.

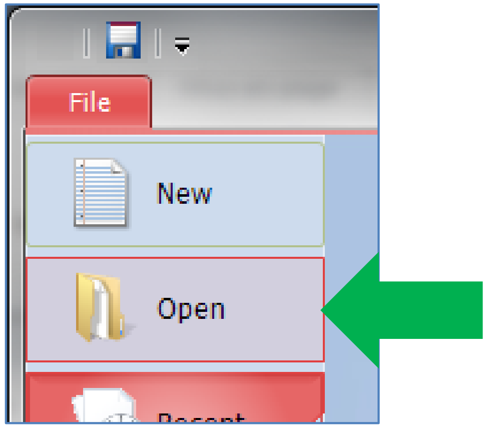
genieArduino-WS4-Demo 7/12/2013 1:52 AM 4D ViSi Genie

For users who want to learn how to create a ViSi Genie application, proceed to the next section.
Workshop 4 opens and displays the **Recent** page.

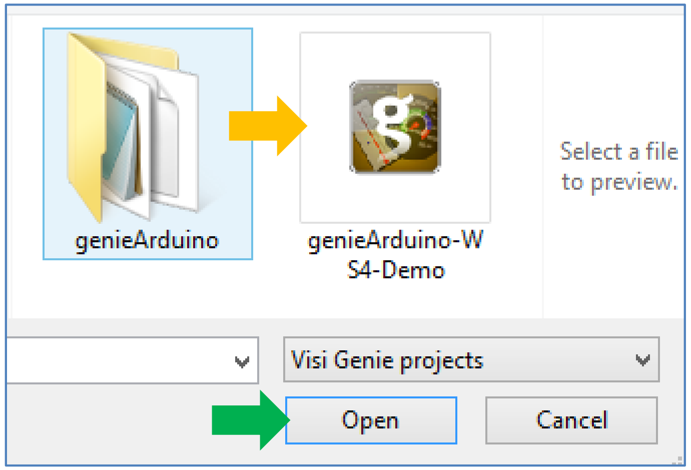

Workshop 4
File
New
Open
Recent
Save
Save As
Zip Project
Close
Print Setup
Print
Options
Help
Samples
Exit
4D SYSTEMS
TURNING TECHNOLOGY INTO ART
WORKSHOP
Version 4.0.1.5
Create a new Project
Start building a new Visi, Genie, Designer or Serial program
Create a new Project
Start building a new program using the same settings as you last used (Visi-Genie uLCD-32PT)
Recent

MyFirstProject.4DGenie 05/10/2012 09:15
C:\Users\O4D\Documents\MyFirstProject.4DGenie
MyFirstProject.4DGenie Not found
G:\MyFirstProject.4DGenie
MyFirstProject.4DGenie Not found
G:\FirstProject\MyFirstProject.4DGenie
MyFirstProject.4DGenie Not found
F:\First project\MyFirstProject.4DGenie
FristProject.4DGenie Not found
F:\First project\FristProject.4DGenie
AN11 - ViSi-Genie RGB.4DVisi 18/09/2012 17:51
C:\Users\O4D\Dropbox\4D Projects\ViSi-Genie Application Notes\4D-AN-1011VG ViSi-Genie - Connection to a Host with Red-Green-Blue LED Control\AN11 - ViSi-Genie RGB\AN11 - ViSi-GenieRGB.4DVisi
AN4 - Customised keyboard.4DVisi 2012 17:06
C:\Users\O4D\Dropbox\4D Projects\ViSi-Genie Follow-Up\25bug\AN4 Customised keyboard\AN4 - Customised keyboard.4DVisi
AN6 - Show Image 2.4DVisi 17/09/2012 10:50
C:\Users\O4D\Dropbox\4D Projects\ViSi-Genie Application Notes\4D-AN-1006VG ViSi-Genie - Show Image\AN6 - Show Image 2\AN6 - Show Image 2.4DVisi

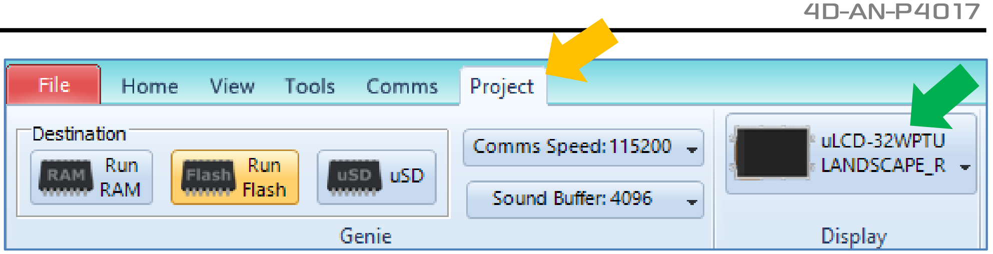
To load the existing project, click on Open.



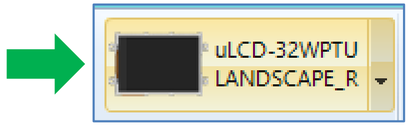
A standard open window asks for a ViSi-Genie project.



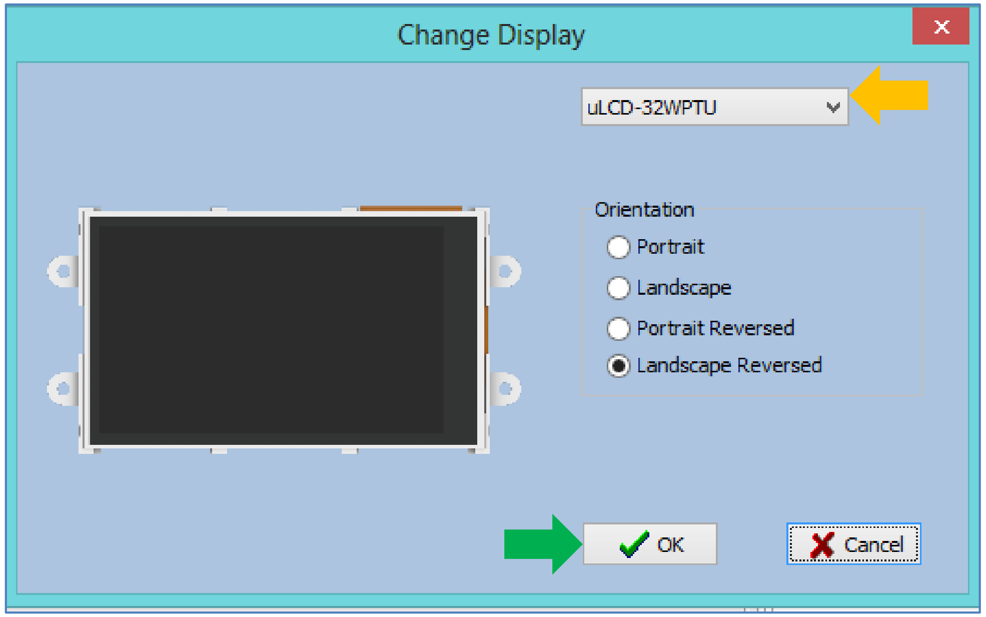
Now, check the type of the screen module by selecting the **Project** menu.



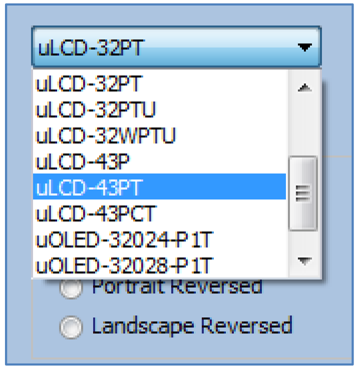
If using a different display module, change the target display module by clicking on the display button.



The Change Display window appears.

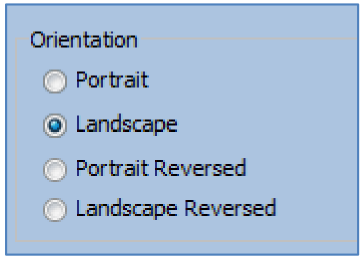


Select the screen on the drop-down list and define the orientation.



uLCD-32PT

- uLCD-32PT
- uLCD-32PTU
- uLCD-32WPTU
- uLCD-43P
- uLCD-43PT
- uLCD-43PCT
- uOLED-32024-P1T
- uOLED-32028-P1T



Orientation

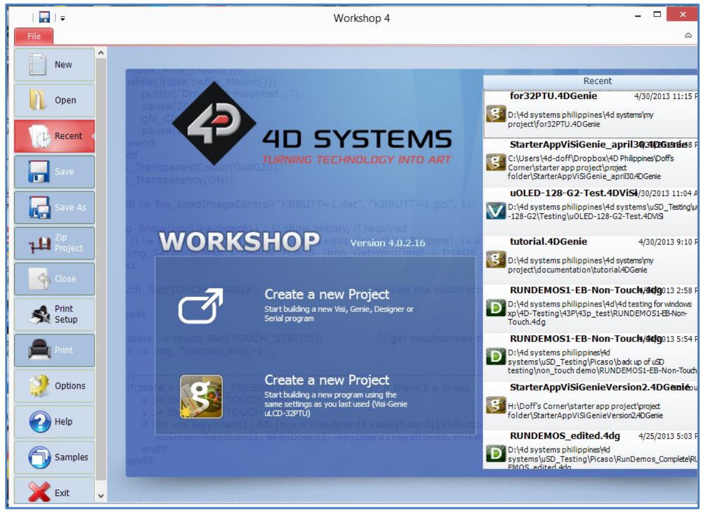
- ○ Portrait
- [✓ Landscape]
- ○ Portrait Reversed
- ○ Landscape Reversed

…and confirm by clicking on .

Create a New Project

Create a New Project

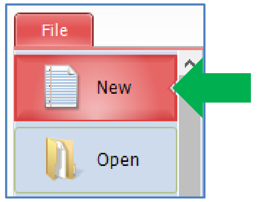
Workshop 4 opens and displays the **Recent** page.



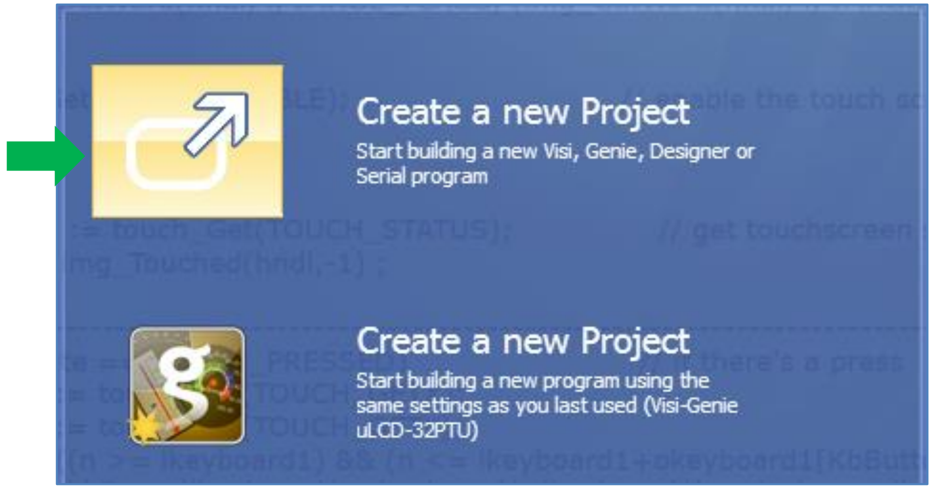
Workshop 4
File
New
Open
Recent
Save
Save As
Zip Project
Close
Print Setup
Print
Options
Help
Samples
Exit
4D SYSTEMS
TURNING TECHNOLOGY INTO ART
WORKSHOP
Version 4.0.2.16
Create a new Project
Start building a new Visi, Genie, Designer or Serial program
Create a new Project
Start building a new program using the same settings as you last used (Visi-Genie uLCD-32PTU)
Recent
for32PTU.4DGenie
StarterAppViSiGenie_april30.4DGenie
uOLED-128-G2-Test.4DViSi
tutorial.4DGenie
RUNDEMOS1-EB-Non-Touch.4dg
RUNDEMOS1-EB-Non-Touch.4dg
StarterAppViSiGenieVersion2.4DGenie
RUNDEMOS_edited.4dg

To create a new project, there are two options.

Click on the top left-most icon, New.

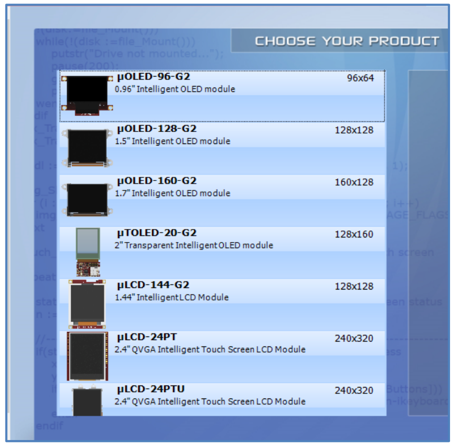


Or Click on the icon beside Create a new Project.


Create a new Project
Start building a new Visi, Genie, Designer or Serial program

Create a new Project
Start building a new program using the same settings as you last used (Visi-Genie uLCD-32PTU)

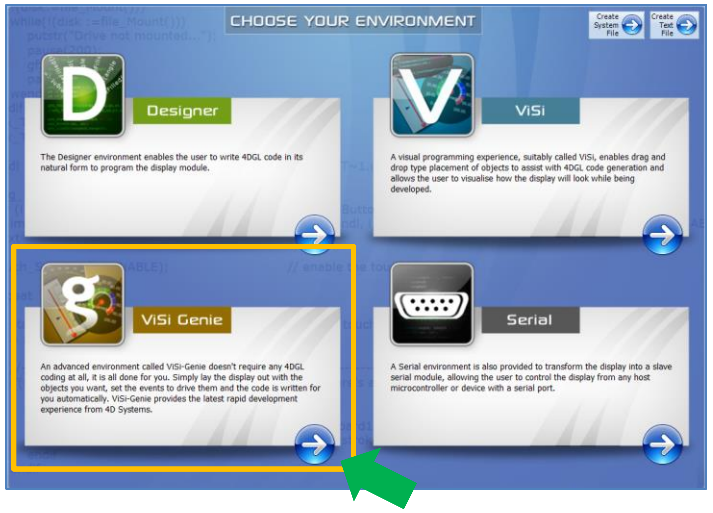
These options update the main window with the selection of the screen.



μOLED-96-G2	0.96" Intelligent OLED module	96x64
μOLED-128-G2	1.5" Intelligent OLED module	128x128
μOLED-160-G2	1.7" Intelligent OLED module	160x128
μTOLED-20-G2	2" Transparent Intelligent OLED module	128x160
μLCD-144-G2	1.44" Intelligent LCD Module	128x128
μLCD-24PT	2.4" QVGA Intelligent Touch Screen LCD Module	240x320
μLCD-24PTU	2.4" QVGA Intelligent Touch Screen LCD Module	240x320

Select the appropriate screen and preferred orientation. The screen used in this example is the **uLCD-32WPTU (landscape reversed orientation)**.

Select ViSi Genie



Designer
The Designer environment enables the user to write 4DGL code in its natural form to program the display module.

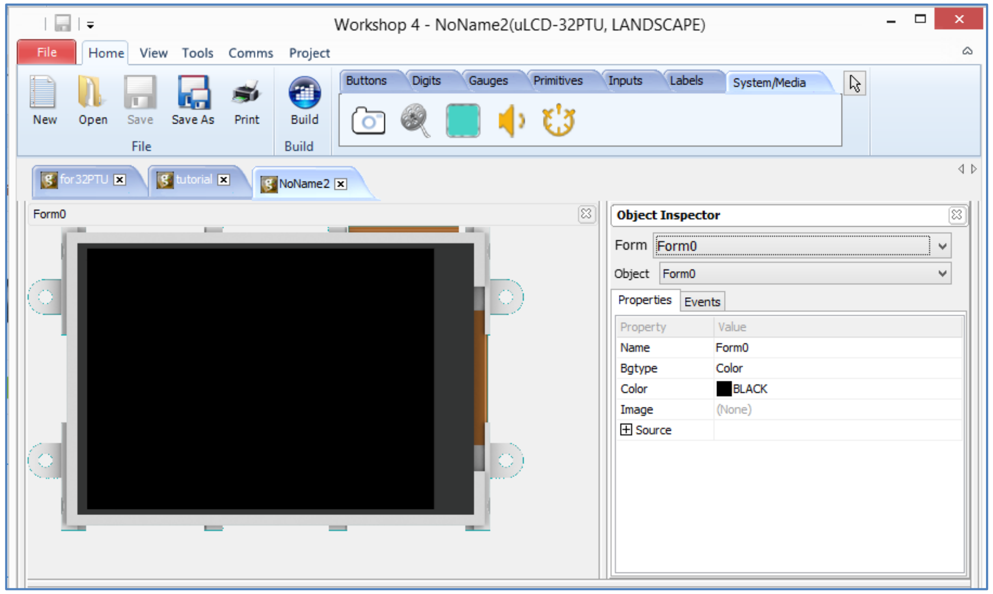
ViSi
A visual programming experience, suitably called ViSi, enables drag and drop type placement of objects to assist with 4DGL code generation and allows the user to visualise how the display will look while being developed.

ViSi Genie
An advanced environment called ViSi-Genie doesn't require any 4DGL coding at all, it is all done for you. Simply lay the display out with the objects you want, set the events to drive them and the code is written for you automatically. ViSi-Genie provides the latest rapid development experience from 4D Systems.

Serial
A Serial environment is also provided to transform the display into a slave serial module, allowing the user to control the display from any host microcontroller or device with a serial port.

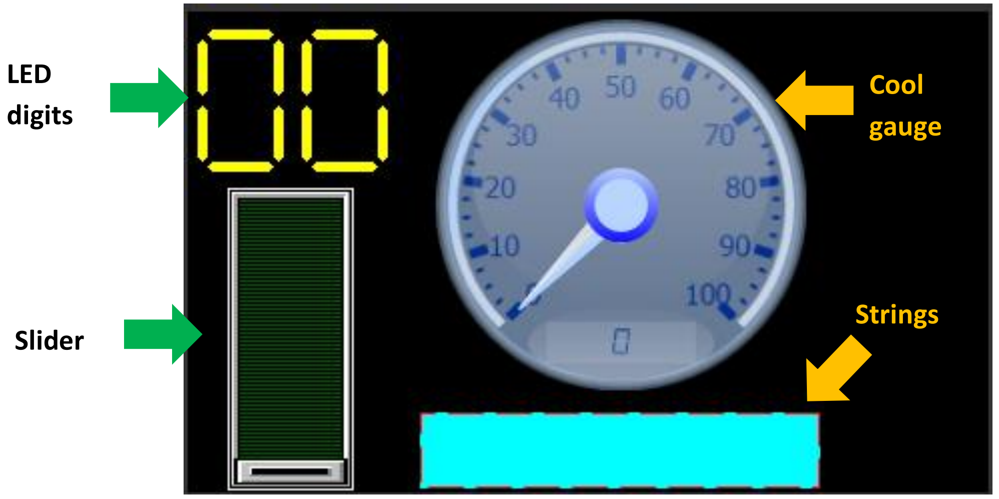
Design the Project

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.



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

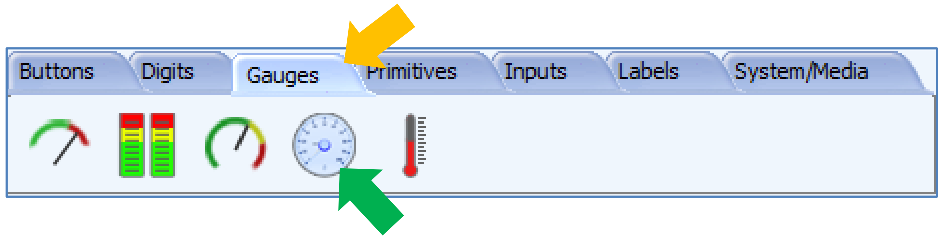
At the end of this section, the user will able to create a form with four objects. The final form will look like as shown below, with the labels excluded.



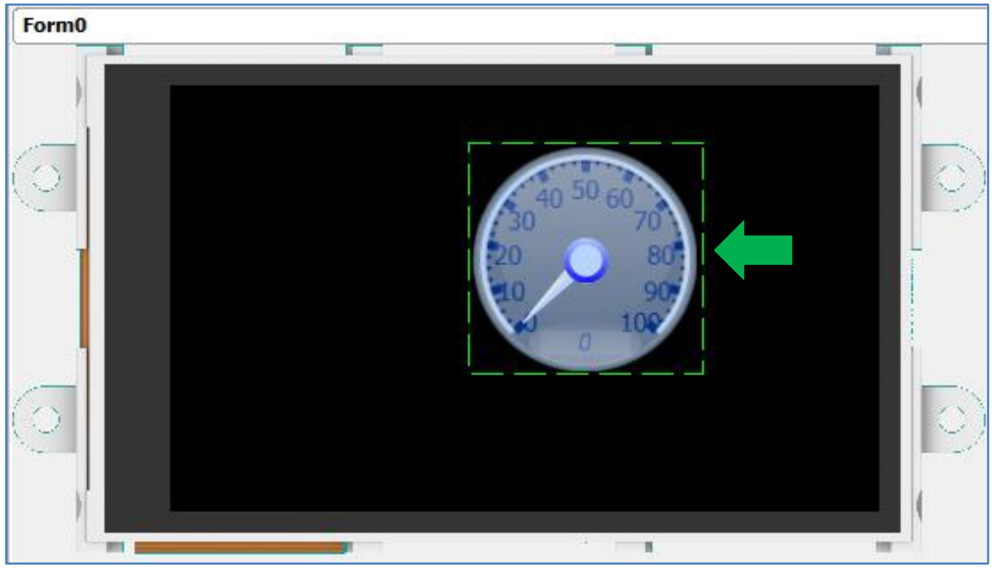
The procedure for adding each of these objects will now be discussed.

Create a Cool Gauge

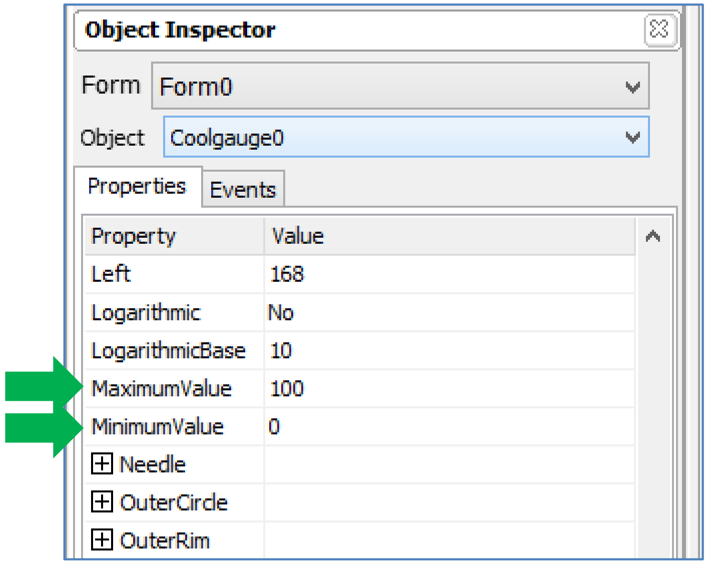
The cool gauge constantly receives values from the Arduino host. The dial of the cool gauge will constantly move to correspond with the received values. To create a cool gauge, go to the **Gauges** pane then click on the **cool gauge** icon.



Click on the **WYSIWYG** (What-You-See-Is-What-You-Get) screen to place the cool gauge. 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 cool gauge object named **Coolgauge0**.



Object Inspector

Form: Form0

Object: Coolgauge0

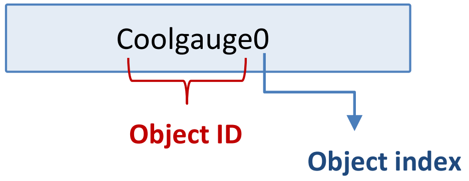
Properties | Events

Property	Value
Left	168
Logarithmic	No
LogarithmicBase	10
MaximumValue	100
MinimumValue	0
⊞ Needle	
⊞ OuterCircle	
⊞ OuterRim	

Feel free to experiment with the different properties. Take note of the maximum and minimum values. These determine the range of values that can be sent from the Arduino host to the cool gauge. To know more about gauges, refer to 4D-AN-P008-ViSi-Genie-Gauges.

Naming of Objects

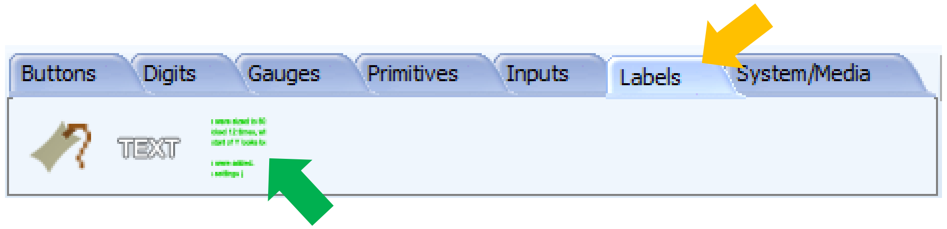
Naming is important to differentiate between objects of the same kind. For instance, suppose the user adds another cool gauge object to the WYSIWYG screen. This object will be given the name Coolgauge1 – it being the second cool gauge in the program. The third cool gauge will be given the name Coolgauge2, and so on. An object’s name therefore identifies its kind and its unique index number. It has an ID (or type) and an index.



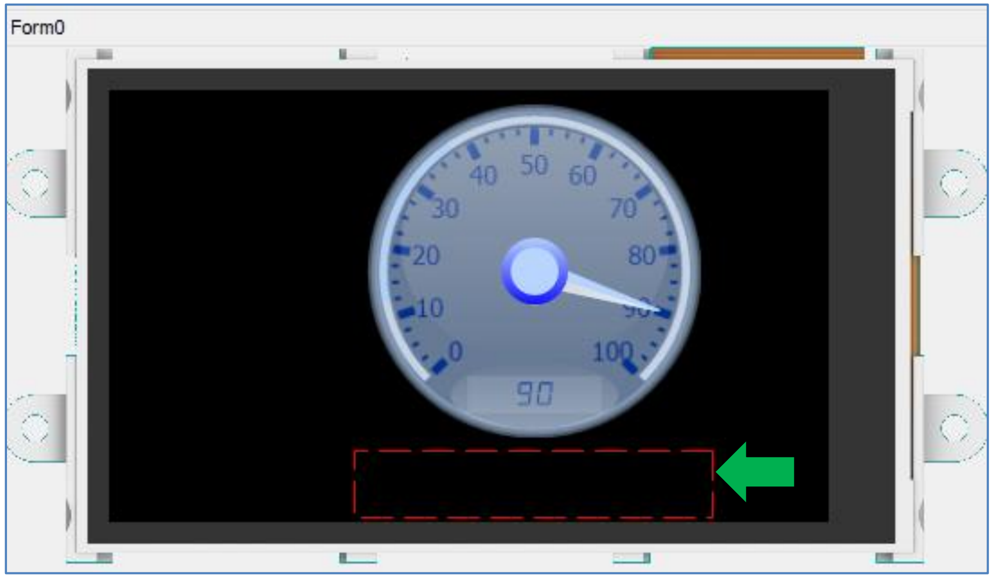
It is important to take note of an object's ID and index. When programming in the Arduino IDE, an object's status can be polled or changed if its ID and index are known. The process of doing this will be shown later.

Create a Text String

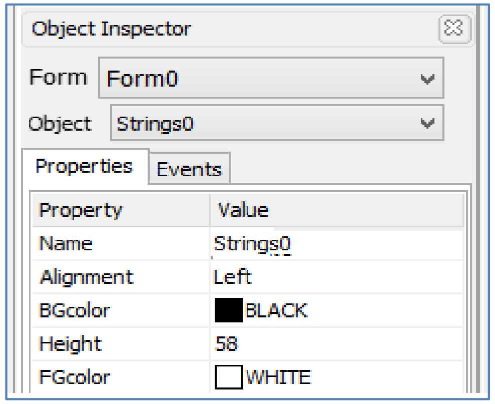
The display module can print text strings received from the Arduino host on the screen. To create a text string object, go to the **Labels** pane then click on the **strings** icon.



Click on the **WYSIWYG** screen to place the string. Again, the WYSIWYG screen simulates the actual appearance of the display module screen.



The space inside the red box will be the space occupied by the text string to be displayed. 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 properties of the newly created strings object named **Strings0**.



Object Inspector

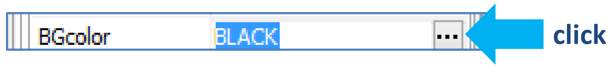
Form: Form0

Object: Strings0

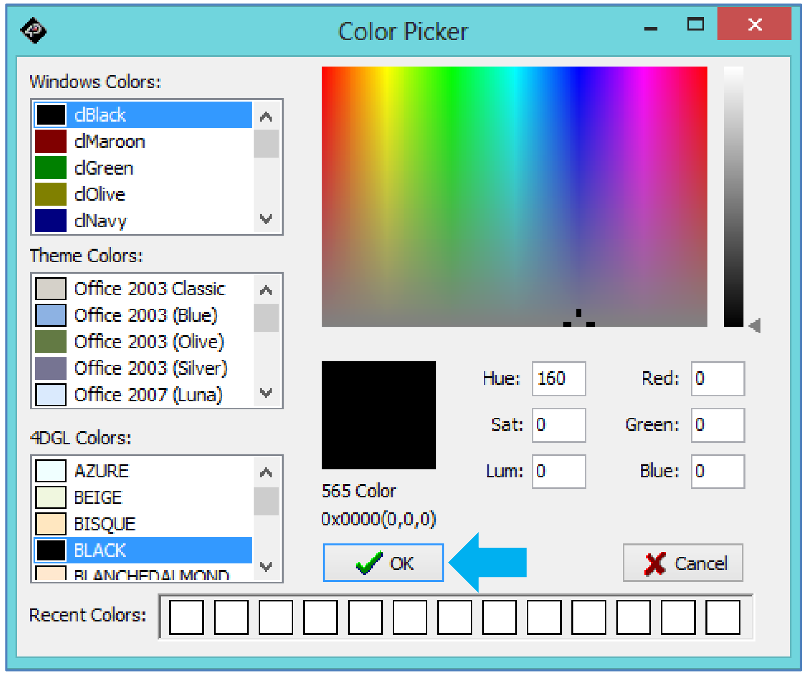
Properties | Events

Property	Value
Name	Strings0
Alignment	Left
BGcolor	■ BLACK
Height	58
FGcolor	□ WHITE

To add background and foreground colours for the text string, edit the properties as shown below.


BGcolor BLACK

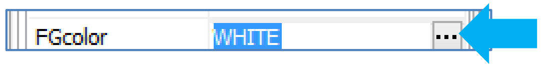
Choose the desired colour and click OK.



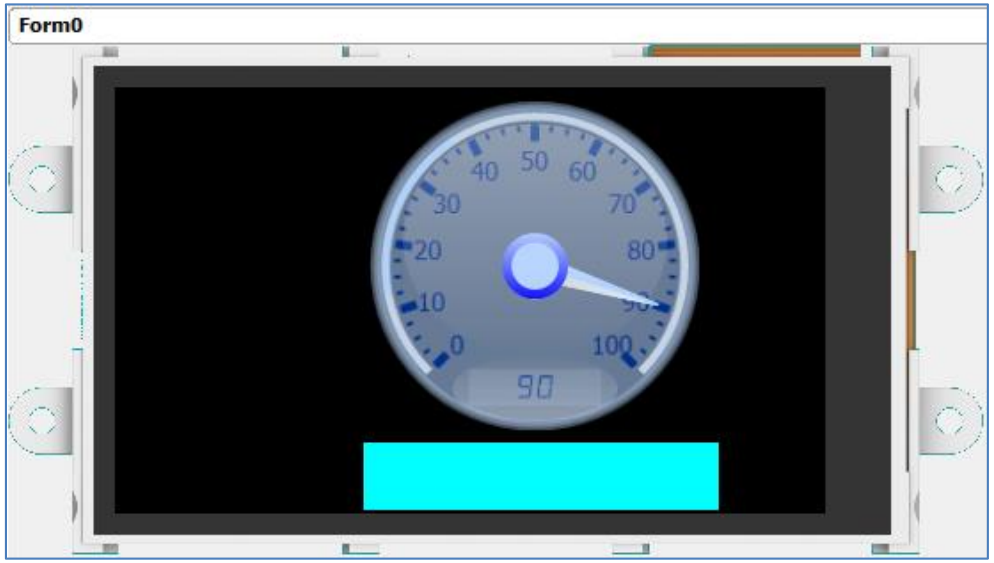
Hue: 160 Sat: 0 Lum: 0 Red: 0 Green: 0 Blue: 0

565 Color 0x0000(0,0,0)

Do the same for the foreground colour.

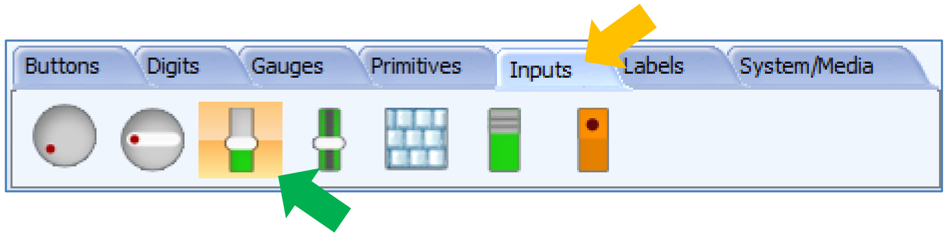

FGcolor WHITE

When done, the form should look similar to that shown below.

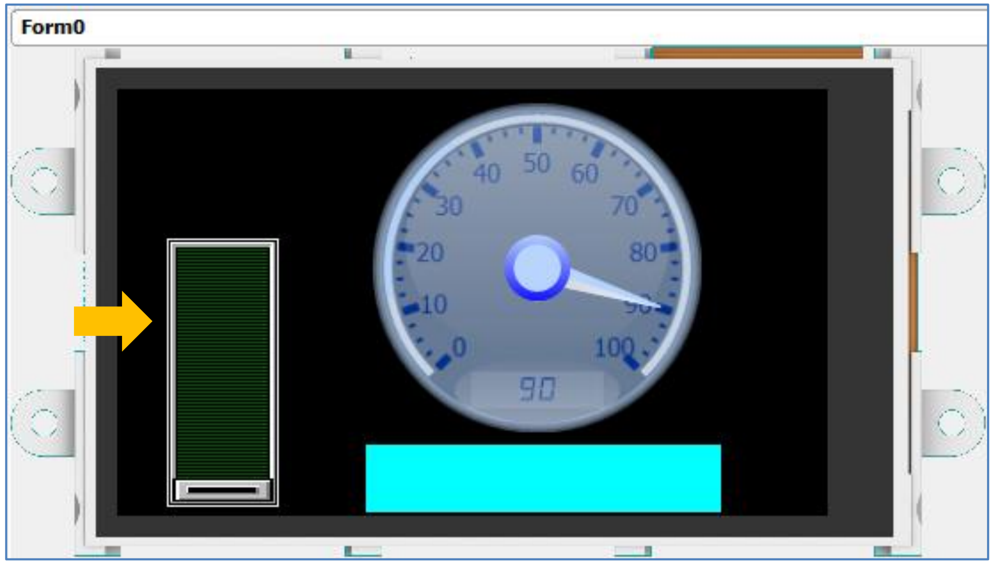


Create a Slider

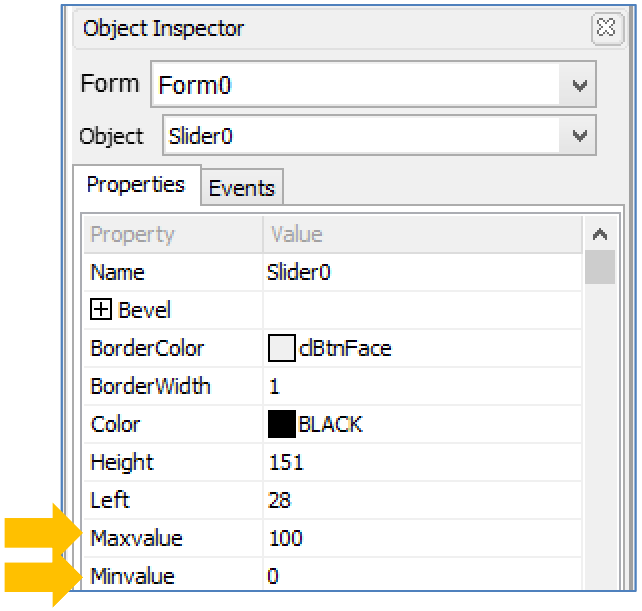
The slider sends a message to the Arduino host when its status has changed. To add a slider, go to the **Inputs** pane and click on the **slider** icon.



Click on the WYSIWYG screen to place a slider object. Drag the object to any desired location.



In the **Object Inspector**, the minimum value is 0 and maximum is 100 by default.



Object Inspector

Form: Form0

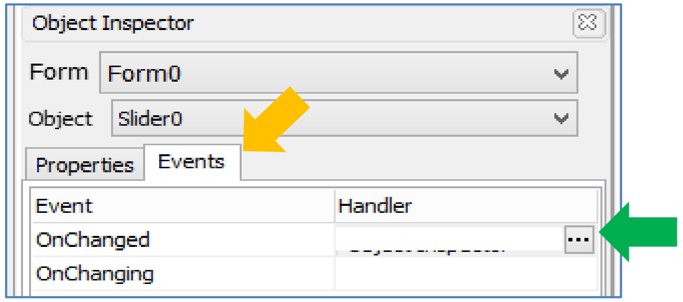
Object: Slider0

Properties | Events

Property	Value
Name	Slider0
⊞ Bevel	
BorderColor	☐ clBtnFace
BorderWidth	1
Color	■ BLACK
Height	151
Left	28
Maxvalue	100
Minvalue	0

Report Event

An object can report its current status independently without being asked by the Arduino host. A slider, for example, can be configured to report its current status to the host each time it is moved. To do this, click on the Events tab in the object inspector and click on the ··· symbol in the **OnChanged** line.



Object Inspector

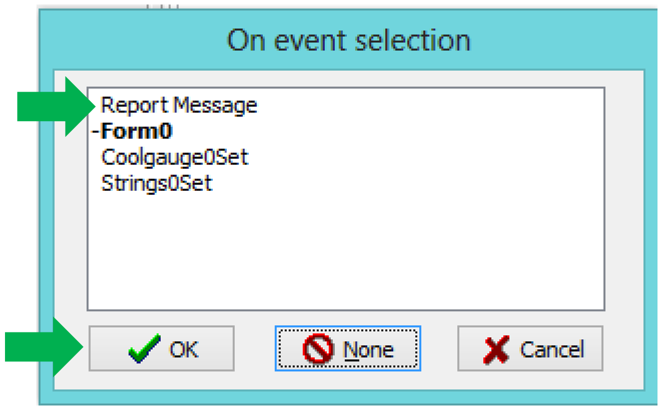
Form: Form0

Object: Slider0

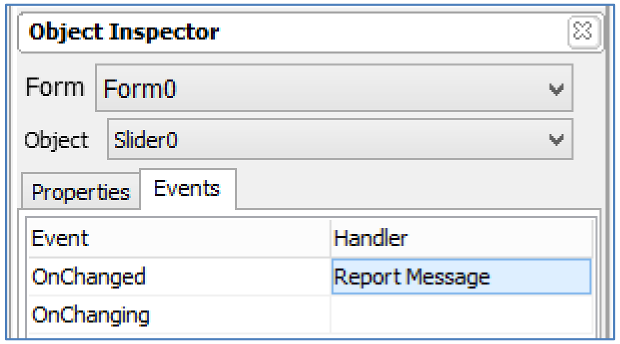
Properties | Events

Event	Handler
OnChanged	
OnChanging	

The **On event selection** window appears. Select **Report Message** and click **OK**.



The Events pane is now updated.



Form	Form0
Object	Slider0

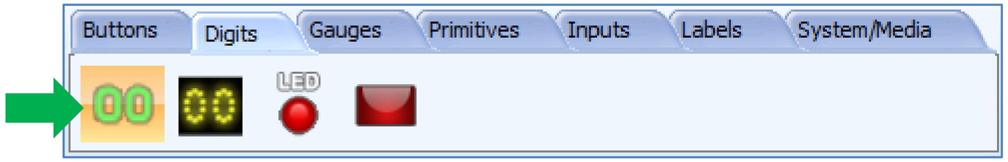
Event	Handler
OnChanged	Report Message
OnChanging	

Now every time the slider is moved or its status has changed, it sends a **message** to the host controller. To be more exact, the slider will send a report message when the stylus or finger moving it is lifted off the screen. Selecting the **OnChanging** event, on the other hand, causes the slider to send messages while it is being moved (the moving finger or stylus is not lifted off yet). To learn more about the onChanged and OnChanging events, read the application note 4D-AN-P4002-ViSi-Genie-onChanging-and-onChanged-Events.

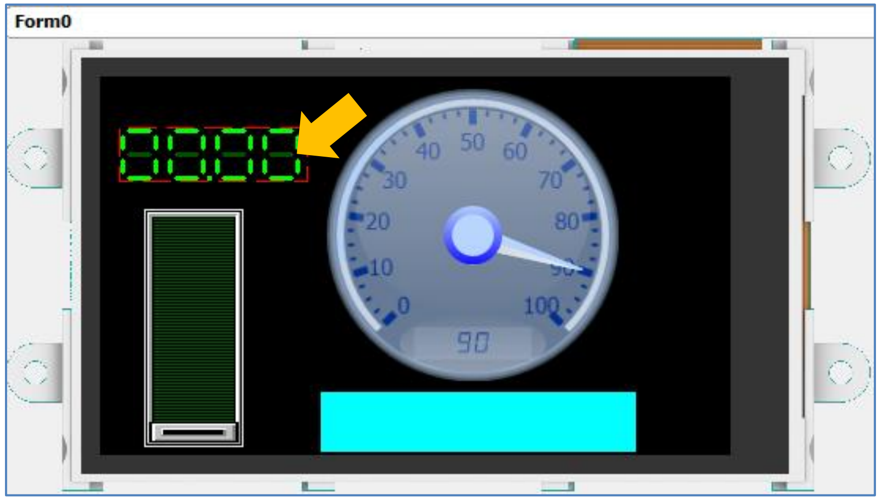
The message or data being sent has a format which the Arduino host must understand. A section of this application note is dedicated to explaining this format (called the ViSi-Genie Communication Protocol) used by the display module. Advanced users may refer to the ViSi Genie Reference Manual.

Create a LED Digits

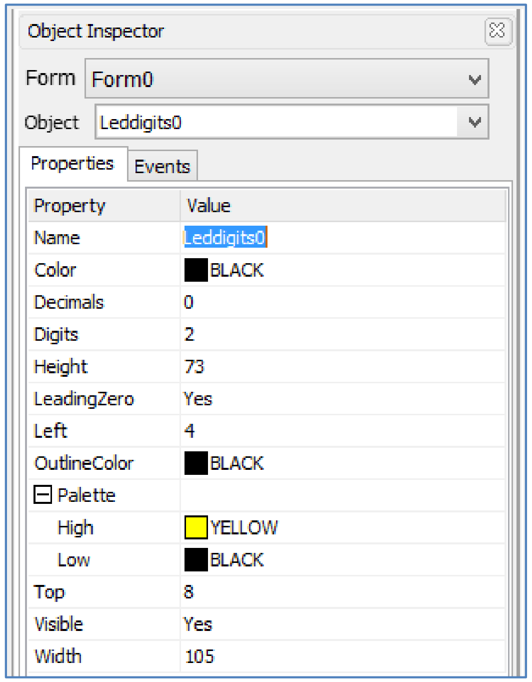
The **LED digits** object will display values received from the Arduino host. To add a LED digits object, go to the **Digits** pane and select the first icon.



Click on the WYSIWYG screen to place it.

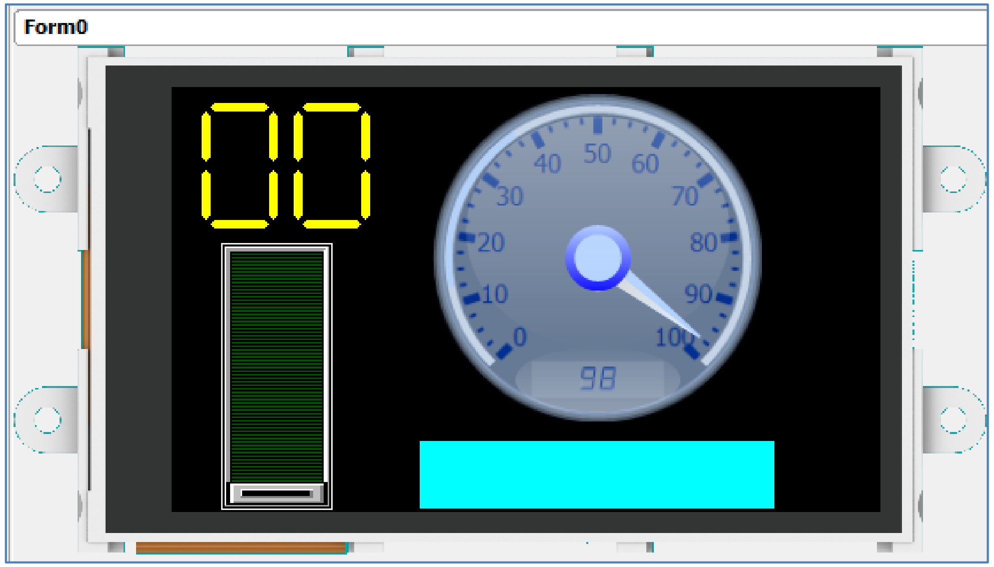


Go to the Object inspector and set the following property values.



Property	Value
Name	Leddigits0
Color	■ BLACK
Decimals	0
Digits	2
Height	73
LeadingZero	Yes
Left	4
OutlineColor	■ BLACK
⊟ Palette	
High	🟨 YELLOW
Low	■ BLACK
Top	8
Visible	Yes
Width	105

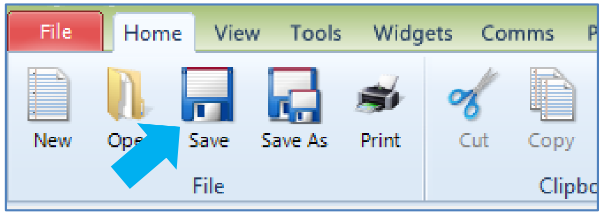
The updated appearance of the LED digits object is shown below.



Build and Upload the Project

How to Save the Program

Save the program with the desired file name first.

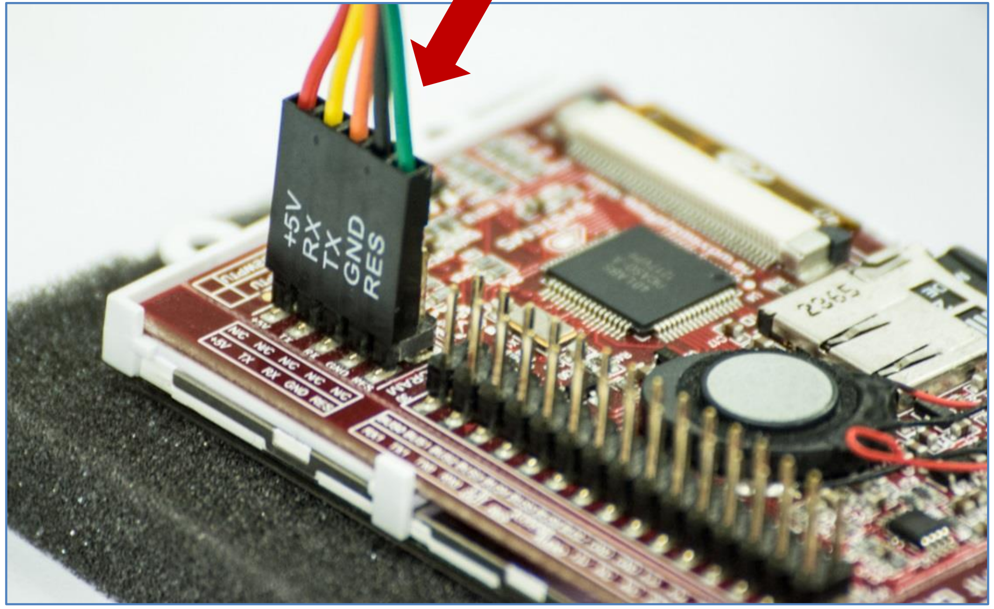


Connect the Display Module

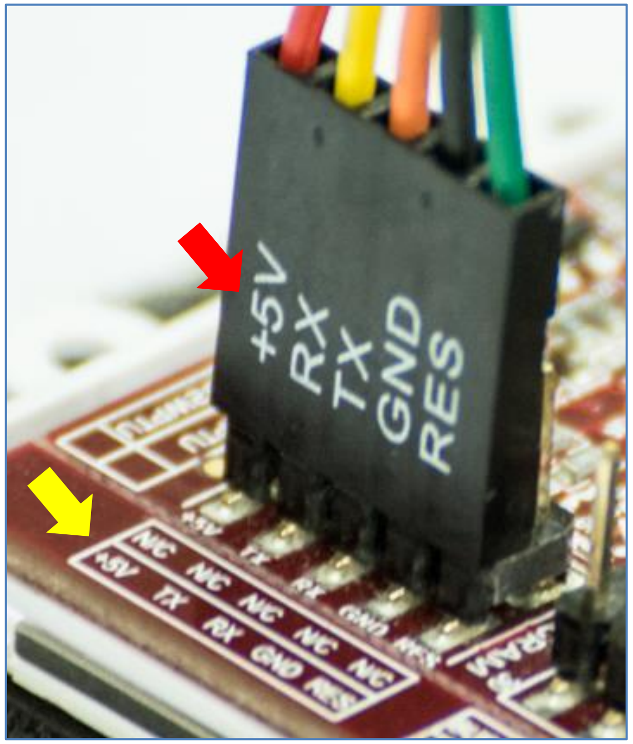
Connect the display module to the PC using a 4D USB Programming Cable or a µUSB-PA5 programming adaptor.

Note: Before using the 4D Programming Cable or the µUSB-PA5 adaptor, the drivers need to be installed first. Click any of the hyperlinks to go to the product page. Follow the instructions on the page for installing the drivers.

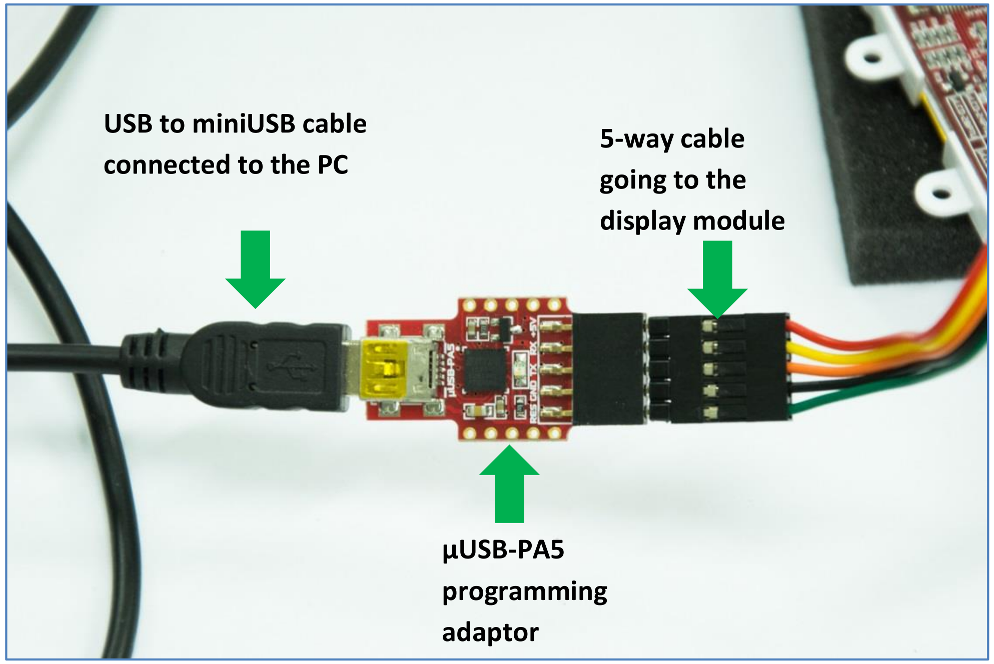
4D USB Programming Cable or a 5-way cable connected to a µUSB-PA5



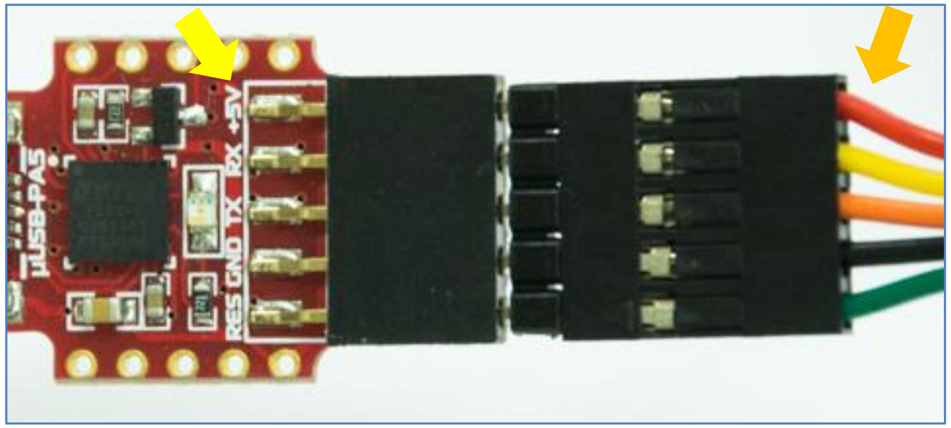
Check the orientation.



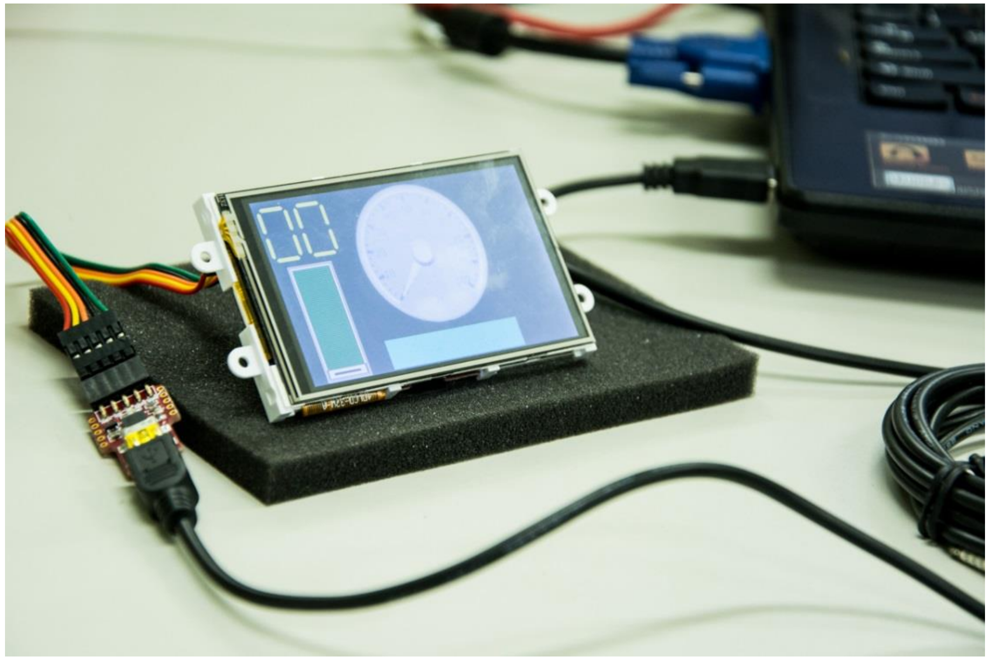
Setup using a µUSB-PA5:


USB to miniUSB cable connected to the PC
5-way cable going to the display module
µUSB-PA5 programming adaptor

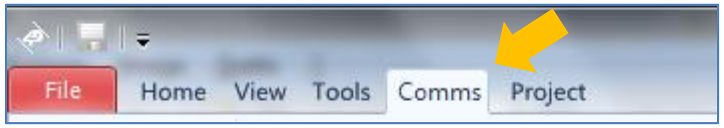
Check the orientation.



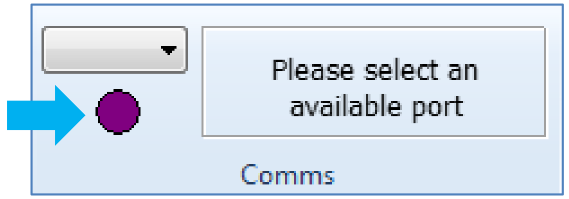
Complete setup:



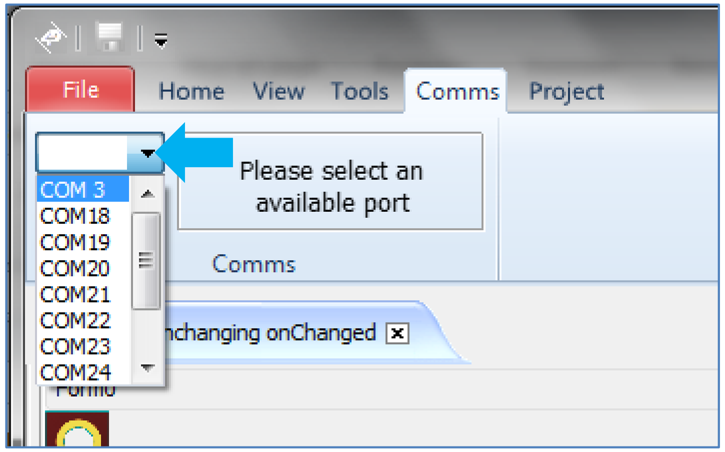
Go to the Comms menu to check if the module is detected.



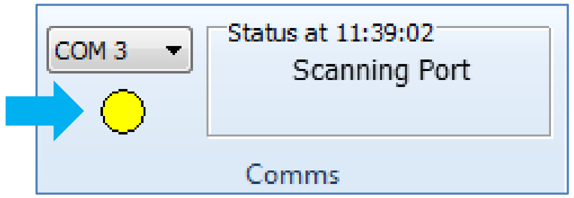
Above the Comms section, the violet light mentions no module is currently connected.



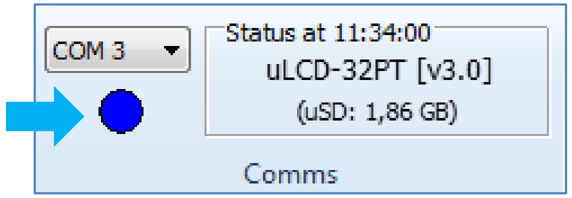
With the display module connected to the 4D USB programming cable (or µUSB-PA5), plug the cable into the USB port. Click on the drop-down list and select the COM port allocated to the cable. The product pages for the programming cable and µUSB-PA5 have instructions on how to determine the allocated COM port.



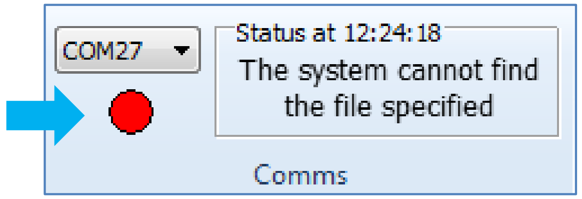
The light turns yellow while the connection is being established:



Finally, the light goes blue when the connection is established.



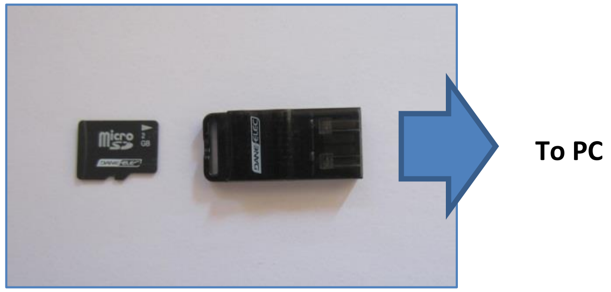
The light turns red when no module is attached to the selected port:



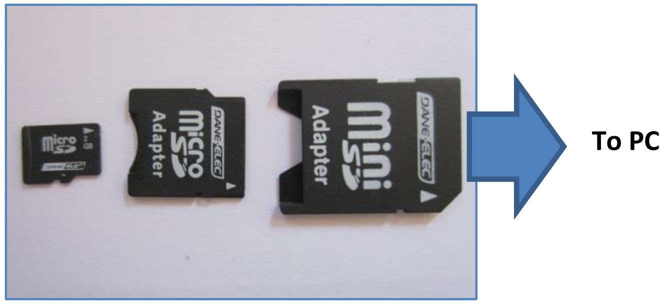
Insert the µSD Card to the PC

For Picaso display modules, a µSD card shall be FAT16-formatted, and partition can't exceed 4 GB.

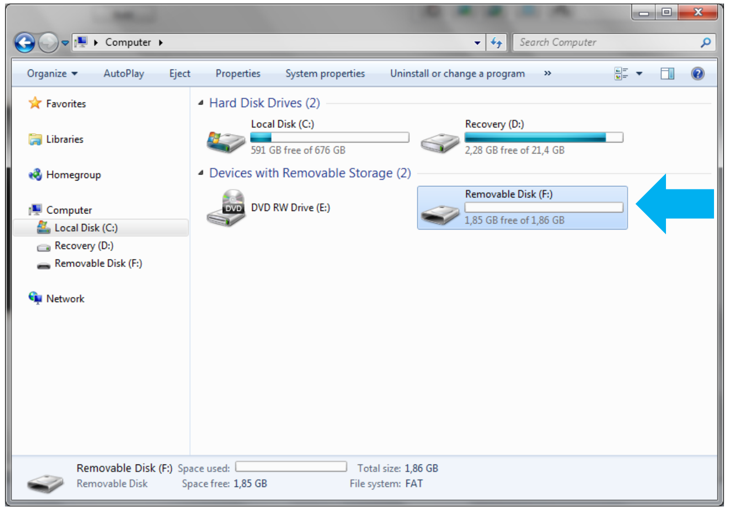
Insert the µSD card into the USB adaptor and plug the USB adaptor into a USB port of the PC.



OR insert the µSD card into a µSD to SD card converter and plug the SD card converter into the SD card slot of the PC.

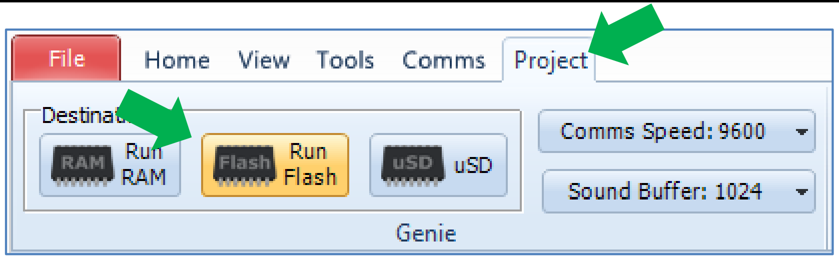


Check if the µSD card is mounted, here it is mounted as drive E:



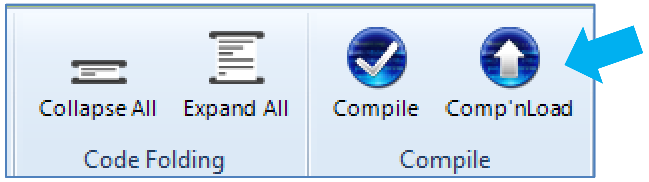
Program Destination

Choose how the project is going to be uploaded to the module. Select the **Project** menu and click on **Flash** as the destination.

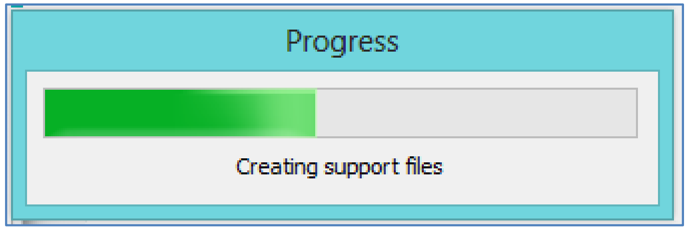


Compile and Download

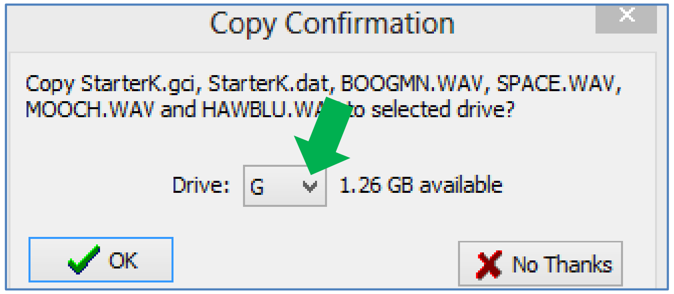
After making sure that the device is detected, go to the **Home** menu and click on the **Comp n'Load** button.



Workshop now builds and downloads the program to the display module.



Workshop will prompt the user for the µSD card. Select the drive on the drop down list then click on OK.

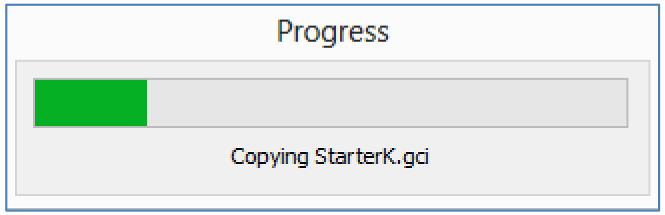


Copy Confirmation

Copy StarterK.gci, StarterK.dat, BOOGMN.WAV, SPACE.WAV, MOOCH.WAV and HAWBLU.WAV to selected drive?

Drive: G 1.26 GB available

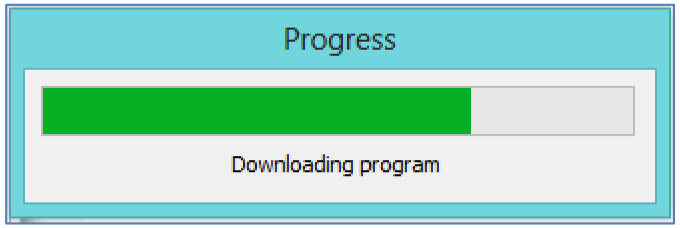
A progress bar is displayed while the necessary files are being copied to the µSD card. Workshop copies two files to the µSD card –the GCI and the DAT files. The GCI file contains the graphics and the DAT file contains a list of the objects inside the GCI file. These files will be accessed by the program when the display module is turned on.



Progress

Copying StarterK.gci

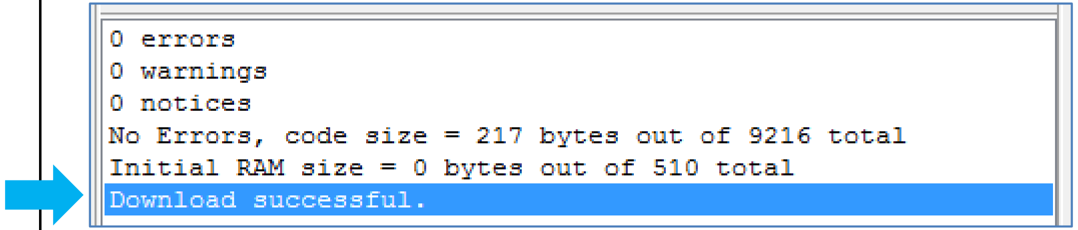
Now Workshop downloads the program to the flash memory of the display module.



Progress

Downloading program

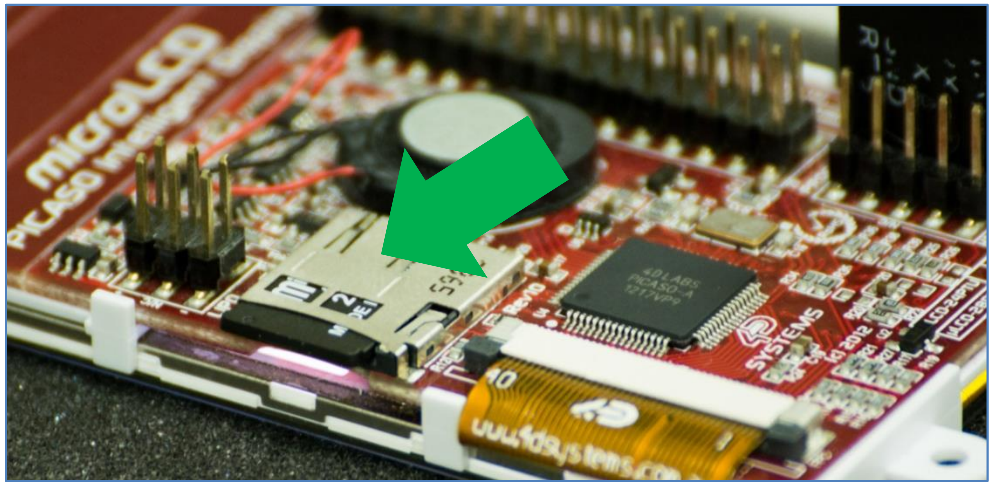
Finally, the message box displays the code size and confirms that the download to the flash memory has been successful.



```
0 errors
0 warnings
0 notices
No Errors, code size = 217 bytes out of 9216 total
Initial RAM size = 0 bytes out of 510 total
Download successful.
```

Insert the µSD Card to the Display Module

Properly disconnect the µSD card from the PC and plug it to the µSD Card slot of the display module. The project now starts and runs on the screen.



Identify the Messages

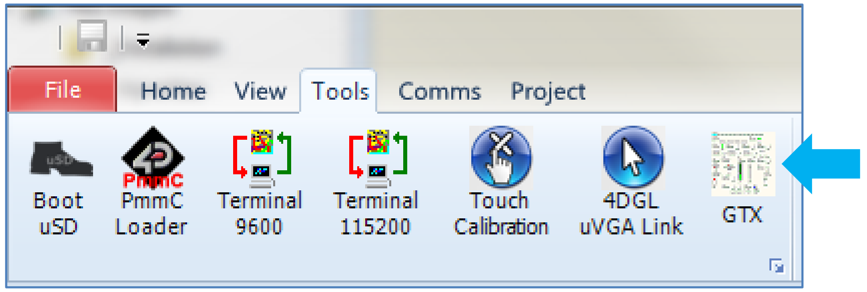
The display module is going to generate and send messages to the host. This section explains to the user how to interpret these messages. An understanding of this section is necessary for the user to be able to properly program the host controller. The ViSi Genie Reference Manual is recommended for advanced users.

Use the GTX Tool to Analyse the Messages

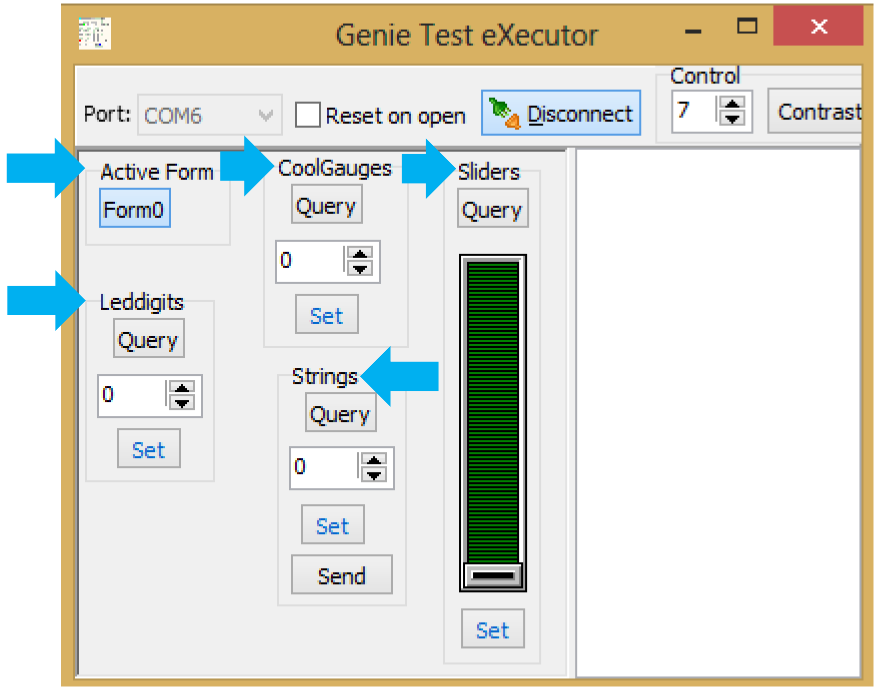
Using the GTX or **Genie Test eXecutor** tool is the first option to get the messages sent by the screen to the host. The GTX tool is a part of the Workshop 4 IDE. It allows the user to receive, observe, and send messages from and to the display module. It is an essential debugging tool.

Launch the GTX Tool

Under tool menu click on the GTX tool button.



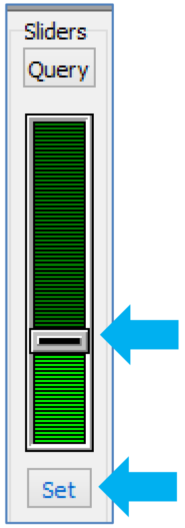
A new window appears, with the form and objects created previously.



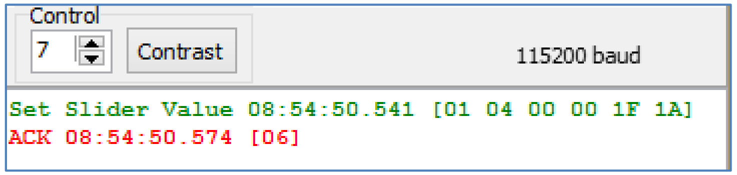
The Slider Object

Change the Status of the Slider

In the GTX tool window, move the slider and press **Set**. On the display module, note that the slider has moved.



Also, messages are sent to and received from the display module.

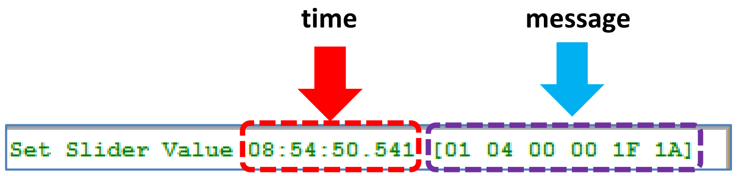


```
Set Slider Value 08:54:50.541 [01 04 00 00 1F 1A]
ACK 08:54:50.574 [06]
```

The white area on the right displays

- in green the messages sent to the display module
- and in red the messages received from the display module

The actual message bytes are those inside the brackets. These values are in hexadecimal. The figure preceding the actual message is the computer time at which the message is sent. A label is also included to tell the observer what the message represents.



```
Set Slider Value 08:54:50.541 [01 04 00 00 1F 1A]
```

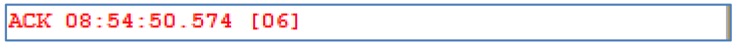
The message sent is formatted according to the following pattern:

Command	Object Type	Object Index	Value MSB	Value LSB	Checksum
01	04	00	00	1F	1A
WRITE_OBJ	Slider	First	0x001F		

The message stands for "Write to the first slider object on the display module the value 0x001F." Converting the hexadecimal value 0x001F to decimal will yield the value 31.

The checksum is a means for the host to verify if the message received is correct. This byte is calculated by XOR'ing all bytes in the message from (and including) the CMD or command byte to the last parameter byte. Then, the result is appended to the end to yield the checksum byte. If the message is correct, XOR'ing all the bytes (including the checksum byte) will give a result of zero. Checking the integrity of a message using the checksum byte is handled automatically by the Arduino host thru the ViSi-Genie-Arduino library.

ACK = 0x06 as shown below

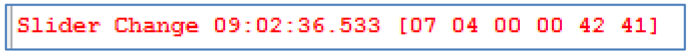


```
ACK 08:54:50.574 [06]
```

is an acknowledgment from the display module which means that it has understood the message.

Message from a Slider

Remember that the slider was configured to **Report a Message** when its status has changed. Now move the slider on the display module with a stylus or a finger. Observe the message sent by the display module to the PC.



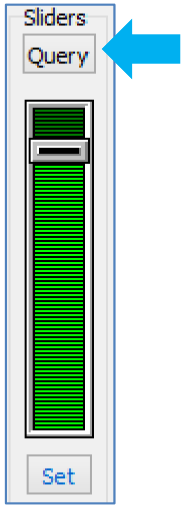
```
Slider Change 09:02:36.533 [07 04 00 00 42 41]
```

The message from slider object is formatted according to the following pattern:

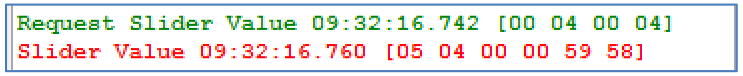
Command	Object Type	Object Index	Value MSB	Value LSB	Checksum
07	04	00	00	42	41
REPORT_EVENT	Slider	First	0x0042		

Interrogate the Display for the Status of the Slider

Suppose the slider object is not configured to report an event when it has moved. The host or the PC can ask the display module for the value of the slider by sending a message. Now on the display module move the slider randomly. In the GTX tool window press Query.



Observe the message area.



```
Request Slider Value 09:32:16.742 [00 04 00 04]
Slider Value 09:32:16.760 [05 04 00 00 59 58]
```

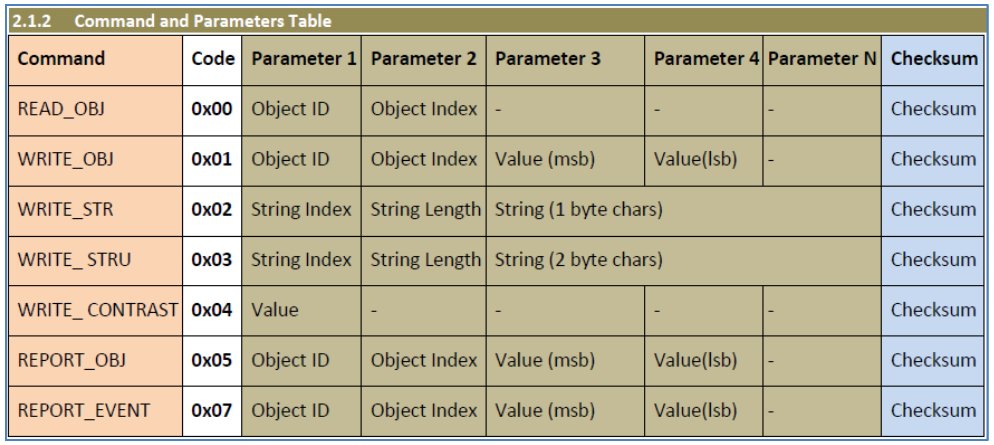
The PC sends a request message. The display module replies with the current value of the slider object. The messages sent and received are formatted according to the following patterns.

Command	Object Type	Object Index	Value MSB	Value LSB	Checksum
00	04	00	-	-	04
READ_OBJ	Slider	First	Not applicable		
05	04	00	00	59	58
REPORT_OBJ	Slider	First	0x0059		

REPORT_EVENT vs. REPORT_OBJ

It is important to take note of the difference between REPORT_EVENT and REPORT_OBJ. **REPORT_EVENT** occurs if the user selects the event of a widget in Workshop to be "Report Message". There is no need for the host to ask the display module for the value of the slider. The slider independently sends its current status since it was configured to do so. Whereas **REPORT_OBJ** is a result of the user doing a read of an object from the host, using the Read Object function.

Experimentation with the different objects using the GTX tool is now left to the user as an exercise. The following tables are shown below for reference. Consult the ViSi Genie Reference Manual for more information.



2.1.2 Command and Parameters Table

Command	Code	Parameter 1	Parameter 2	Parameter 3	Parameter 4	Parameter N	Checksum
READ_OBJ	0x00	Object ID	Object Index	-	-	-	Checksum
WRITE_OBJ	0x01	Object ID	Object Index	Value (msb)	Value(lsb)	-	Checksum
WRITE_STR	0x02	String Index	String Length	String (1 byte chars)			Checksum
WRITE_ STRU	0x03	String Index	String Length	String (2 byte chars)			Checksum
WRITE_ CONTRAST	0x04	Value	-	-	-	-	Checksum
REPORT_OBJ	0x05	Object ID	Object Index	Value (msb)	Value(lsb)	-	Checksum
REPORT_EVENT	0x07	Object ID	Object Index	Value (msb)	Value(lsb)	-	Checksum

This table is found on page 6 of the ViSi Genie Reference Manual .

3.3. Object Summary Table

Object	ID	Input	Output	Notes
Dipswitch	0 (0x00)	✓	✓	
Knob	1 (0x01)	✓	✓	
Rockerswitch	2 (0x02)	✓	✓	
Rotaryswitch	3 (0x03)	✓	✓	
Slider	4 (0x04)	✓	✓	
Trackbar	5 (0x05)	✓	✓	
Winbutton	6 (0x06)	✓	✓	
Angularmeter	7 (0x07)		✓	
Coolgauge	8 (0x08)		✓	
Customdigits	9 (0x09)		✓	
Form	10 (0x0A)		✓	Used to set the current form
Gauge	11 (0x0B)		✓	
Image	12 (0x0C)			Displayed as part of form, no method to alter
Keyboard	13 (0x0D)	✓		Keyboard inputs are always single bytes and are unsolicited
Led	14 (0x0E)		✓	
Leddigits	15 (0x0F)		✓	
Meter	16 (0x10)		✓	
Strings	17 (0x11)		✓	
Thermometer	18 (0x12)		✓	
Userled	19 (0x13)		✓	
Video	20 (0x14)		✓	
Statictext	21 (0x15)			Displayed as part of form, no method to alter
Sound	22 (0x16)		✓	
Timer	23 (0x17)		✓	

This table is found on page 42 of the ViSi Genie Reference Manual .

Program the Arduino Host

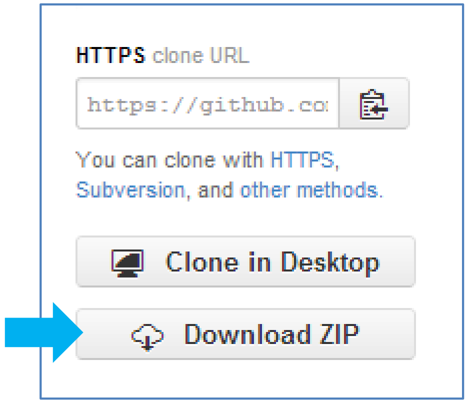
This section discusses how to program the Arduino host to make it work with the display module. It is assumed that the user has a basic understanding of how the Arduino host works and how to program in the Arduino IDE. Inexperienced users may need to frequently refer to the Arduino website for more information.

Download and Install the ViSi-Genie-Arduino Library

The ViSi-Genie-Arduino library files are located here:

https://github.com/4dsystems/ViSi-Genie-Arduino-Library

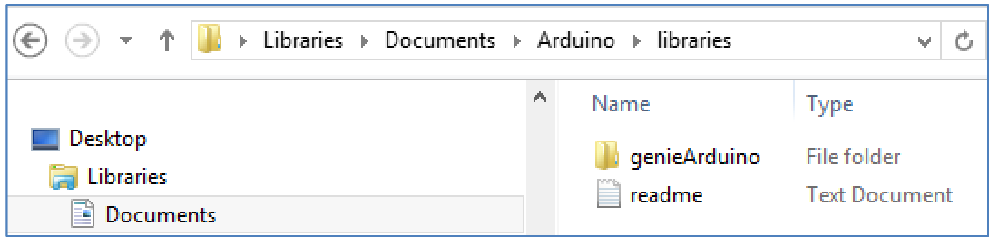
On the right side of the github page, click on the Download ZIP button. Save the zip file and extract its contents to the folder where additional Arduino libraries are saved.



Here is a link to a tutorial on installing additional libraries in the Arduino IDE.

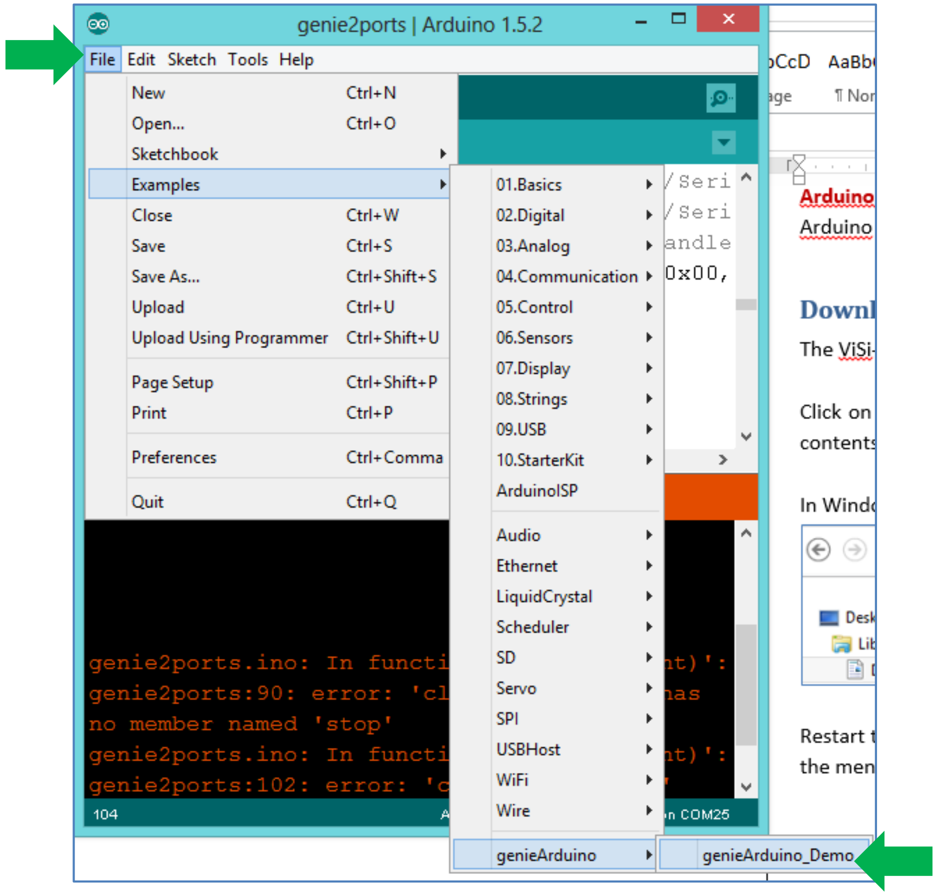
http://arduino.cc/en/Guide/Libraries

In Windows for example, the library files will be saved here:



Name	Type
genieArduino	File folder
readme	Text Document

Remember to restart the Arduino IDE after installing the libraries. The genieArduino demo sketch should be accessible under the File – Examples menu.



```
genie2ports.ino: In functi
genie2ports:90: error: 'cl
no member named 'stop'
genie2ports.ino: In functi
genie2ports:102: error: 'c
```

Understanding the Arduino Sketch Demo

Open the genieArduino_Demo sketch. Note that comments have been added to the code to help the user. Additional explanations are now given below.

Open a Serial Port and Set the Baud Rate

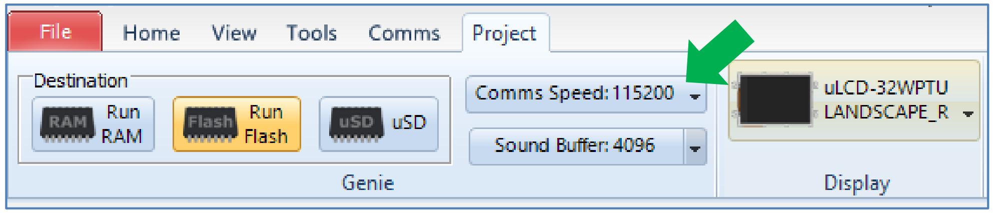
In the **setup ()** function a group of lines allow the user to choose the serial port and the baud rate to be used in talking to the display.

```
void setup()
{
  // a few options to talk to the Display, uncomme
  genieBegin (GENIE_SERIAL, 115200);  //Serial0
  //genieBegin (GENIE_SERIAL_1, 9600);  //Serial1
  //genieBegin (GENIE_SERIAL_2, 9600);  //Serial2
  //genieBegin (GENIE_SERIAL_3, 9600);  //Serial3
```

By default, the first line is uncommented which means that the Arduino host uses the port Serial0 to communicate with the display module. The function

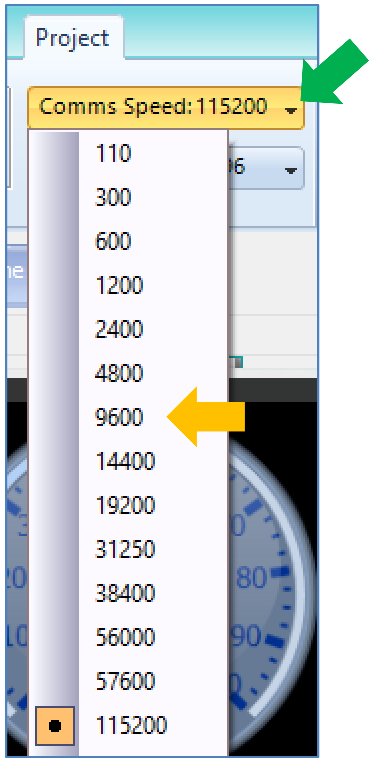
```
  genieBegin (GENIE_SERIAL, 115200);  //Serial0
```

therefore, opens the port Serial0 for communication with the 4D display module at 115200 bps. Logically, the 4D display should also communicate with the Arduino host at the same baud rate. To check the baud rate of the ViSi Genie program go to the project menu in Workshop.



Comms Speed: 115200
Sound Buffer: 4096

To change the baud rate of the ViSi Genie program, simply click on the drop down arrow.



Choose the desired baud rate (9600 bps for instance) and download the program to the display module again. Now change the baud rate of the Arduino host as well.

```
void setup()
{
  // a few options to talk to the Display, uncomme
  genieBegin (GENIE_SERIAL, 9600);  //Serial0
```

For Arduino boards with four hardware serial ports (like the Due and Mega 2560), the user can choose another port to talk to the display by uncommenting the corresponding line. To use Serial2 at 9600 bps for example:

```
void setup()
{
  // a few options to talk to the Display, uncomme
  //genieBegin (GENIE_SERIAL, 9600);  //Serial0
  //genieBegin (GENIE_SERIAL_1, 9600);  //Serial1
  genieBegin (GENIE_SERIAL_2, 9600);  //Serial2
  //genieBegin (GENIE_SERIAL_3, 9600);  //Serial3
```

N.B.: Again, remember that the baud rate of the display module should match that of the Arduino host.

genieAttachEventHandler( )

```
  genieAttachEventHandler(myGenieEventHandler);
```

This function is used to tell the library the name of the function used in the user's code space, so it knows what to call when an event is received and it needs processing. Full explanation is given in the section "Receiving Data from the Display".

Reset the Arduino Host and the Display

It is sometimes necessary to reset the Arduino host and the display module. To make resetting more convenient, the code below resets the display module when the program is restarted.

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

If using the new 4D Arduino Adaptor Shield (Rev 2)

Note that the GPIO pin D4 of the Arduino host is used here for resetting the display. When using the new 4D Arduino Adaptor Shield (Rev 2.00 written on the PCB), simply connect RES to AR in jumper J1. See the section "Connect the 4D Display Module to the Arduino Host". If using the old 4D Arduino Adaptor Shield (Rev 1), simply change the code above. Use pin 2 instead of pin 4.

```
//Reset the Display (change D4 to D2 if you have o
pinMode(2, OUTPUT);  // Set D4 on Arduino to Output
digitalWrite(2, 1);  // Reset the Display via D4
delay(100);
digitalWrite(2, 0);  // unReset the Display via D4
```

If using the old 4D Arduino Adaptor Shield (Rev 1)

If using jumper connecting wires, connect the RESET pin of the display module to the D4 pin of the Arduino with a 1kohm series resistor in between (see the section "Connect the 4D Display Module to the Arduino Host"), and modify the code as shown below.

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

If using jumper wires

Note that the logic state for resetting the display is now 0 instead of 1. This is because the display module's RESET pin is directly connected to D4 via a 1kohm resistor. If using a 4D Arduino Adaptor Shield, the display module's RESET pin is switched by the D4 pin via a transistor.

Set the Screen Contrast and Send a Text String

The 3.5 second – delay below waits for the display module to start up.

```
delay (3500); //let the display start up
```

To make sure that the LCD is turned on, write

```
//Turn the Display on (Contrast) - (Not needed but
genieWriteContrast(1); // 1 = Display ON, 0 = Disp
```

To turn off the display, write

```
genieWriteContrast(0); // 1 = Display ON, 0 = Displ
```

PTU display modules can only have a value of 0 or 1 for contrast. The uLCD-43 modules, however, support contrast values from 0 to 15, which makes power saving possible. This function does not apply to uVGA-II/III modules.

The Main Loop

This is now the main part of the program. Three variables are declared and used here – **`waitPeriod`** for checking how much time has elapsed, **`gaugeAddVal`** holds the increment or decrement value of the cool gauge, and **`gaugeVal`** holds the value to be sent to the cool gauge.

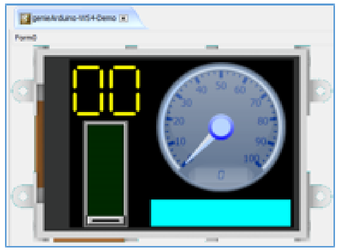
```
void loop()
{
  static long waitPeriod = millis();
  static int gaugeAddVal = 1;
  static int gaugeVal = 50;
```

Receiving Data from the Display

The function below receives and queues the data received from the display.

```
  genieDoEvents();
```

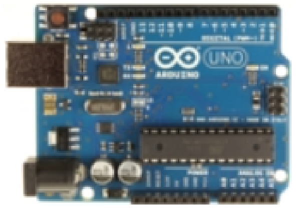
Another function (to be written by the user) is needed to process the received data. A diagram of how data from the display is received and processed is now shown below.



1

genieDoEvents( )
This function receives and queues the data received from the display. The data received can represent an event (a Reported Message sent from the Display due to a user interaction) or can be a result of a Read Object call (a manual poll of information from the user to the display, requesting information from a particular object). genieDoEvents() is defined and located in the ViSi-Genie-Arduino library.

To process the data received, genieDoEvents() calls an external function which is located in the user's code space, not in the library. This external function is called the "Event Handler" – a function written by the user.



2

myGenieEventHandler( )
This is a user-defined function and can be named arbitrarily. This function enables the program to evaluate the data queued by genieDoEvents(). This function takes one set or frame of data (one frame is for one event) from the event queue and evaluates it based on the code defined by the user. The user can break down the event into object type and index number, and extract the data from the event of a particular object when the event is received by the Arduino. At the start of this function, the event is removed from the queue using the **genieDequeueEvent()** function.

genieAttachEventHandler()
This function tells the ViSi-Genie-Arduino library the name of the external function (myGenieEventHandler() above) that needs to be called from the user's code. genieDoEvents() does not know what external function to call for processing the received data since this external function is user-defined. There is therefore a need for the user to tell genieDoEvents() which function to call. Do this by writing genieAttachEventHandler(name-of-your-function-here) at the start of the program. By default, the user's function is called myGenieEventHandler().

To sum it up, the display sends information to the Arduino, whether it is by result of a Reported Message or from a Read Object. The Arduino is executing its code in the main loop. It comes to the **genieDoEvents()** function and calls the genieArduino library. The Arduino 'sees' there is some information available and stores it in a queue. When the Arduino processes the queue, the first event is taken out of the queue and processed based on the commands set out in the user-defined Event Handler function. The Arduino then returns to executing any other functions in its main loop, and then jumps to the start of the loop again.

Change the Cool Gauge's Status

To change the status of the cool gauge, use the function indicated below.

```
// Write to CoolGauge0 with the value in the gaugeVal v
genieWriteObject(GENIE_OBJ_COOL_GAUGE, 0x00, gaugeVal);
gaugeVal += gaugeAddVal;
if (gaugeVal == 99) gaugeAddVal = -1;
if (gaugeVal == 0) gaugeAddVal = 1;
```

GENIE_OBJ_COOL_GAUGE is the object's ID or type, **0x00** is the index, and **gaugeVal** is the value to be written to the object. Note that **gaugeVal** is incremented or decremented by **gaugeAddVal**. Also **gaugeVal** is limited to a value between and including 0 and 99. Remember that the cool gauge in the display module has minimum and maximum values of 0 and 100.

It is possible to change the status of any object as long as the object ID and index are known. The image below lists the object types or IDs already defined in the ViSi-Genie-Arduino library. All of the objects can be written to except the GENIE_OBJ_KEYBOARD and GENIE_OBJ_STATIC_TEXT.

GENIE_OBJ_DIPSW	0		
GENIE_OBJ_KNOB	1	GENIE_OBJ_KEYBOARD	13
GENIE_OBJ_ROCKERSW	2	GENIE_OBJ_LED	14
GENIE_OBJ_ROTARYSW	3	GENIE_OBJ_LED_DIGITS	15
GENIE_OBJ_SLIDER	4	GENIE_OBJ_METER	16
GENIE_OBJ_TRACKBAR	5	GENIE_OBJ_STRINGS	17
GENIE_OBJ_WINBUTTON	6	GENIE_OBJ_THERMOMETER	18
GENIE_OBJ_ANGULAR_METER	7	GENIE_OBJ_USER_LED	19
GENIE_OBJ_COOL_GAUGE	8	GENIE_OBJ_VIDEO	20
GENIE_OBJ_CUSTOM_DIGITS	9	GENIE_OBJ_STATIC_TEXT	21
GENIE_OBJ_FORM	10	GENIE_OBJ_SOUND	22
GENIE_OBJ_GAUGE	11	GENIE_OBJ_TIMER	23
GENIE_OBJ_IMAGE	12		

How to Know the Status of an Object

The Arduino can interrogate the display module for the status of a certain object. For instance, read the current status of the slider using the command indicated below.

```
// The results of this call will be available to myGeni
// Do a manual read from the Slider0 object
genieReadObject(GENIE_OBJ_SLIDER, 0x00);
```

The status of any object can be read as long as the object ID and index are known. The display module will then reply with a message containing the object's current status. This message is received and queued by **genieDoEvents()** and dequeued and evaluated by the user's event handler.

The User's Event Handler

First, the union **Event** of the **genieFrame** type is declared.

```
//LONG HAND VERSION, MAYBE MORE VISIBLE AND MORE LIKE '
void myGenieEventHandler(void)
{
  genieFrame Event;
```

genieFrame is defined in the ViSi-Genie-Arduino library. It contains the structure **genieFrameReportObj**. **genieFrameReportObj** contains five bytes, each representing a byte to be received from the display module. To illustrate:

```
struct genieFrameReportObj {
      uint8_t          cmd;
      uint8_t          object;
      uint8_t          index;
      uint8_t          data_msb;
      uint8_t          data_lsb;
};
```

The next step now is to take an event or a message from the event queue of the **genieDoEvents()** function and place this event to the newly created union **Event**. Observe the correct syntax.

```
  genieDequeueEvent(&Event);
```

Then break down the message into its components. A part of the sketch demonstrates how this is done.

```
//If the cmd received is from a Reported Event
if(Event.reportObject.cmd == GENIE_REPORT_EVENT)
{
  if (Event.reportObject.object == GENIE_OBJ_SLIDER)
  {
    if (Event.reportObject.index == 0)
    {
      slider_val = (Event.reportObject.data_msb << 8) + Event.r
      genieWriteObject(GENIE_OBJ_LED_DIGITS, 0x00, slider_val);
    }
  }
}
```

Here is a list of Genie commands taken from the .h file of the ViSi-Genie-Arduino library.

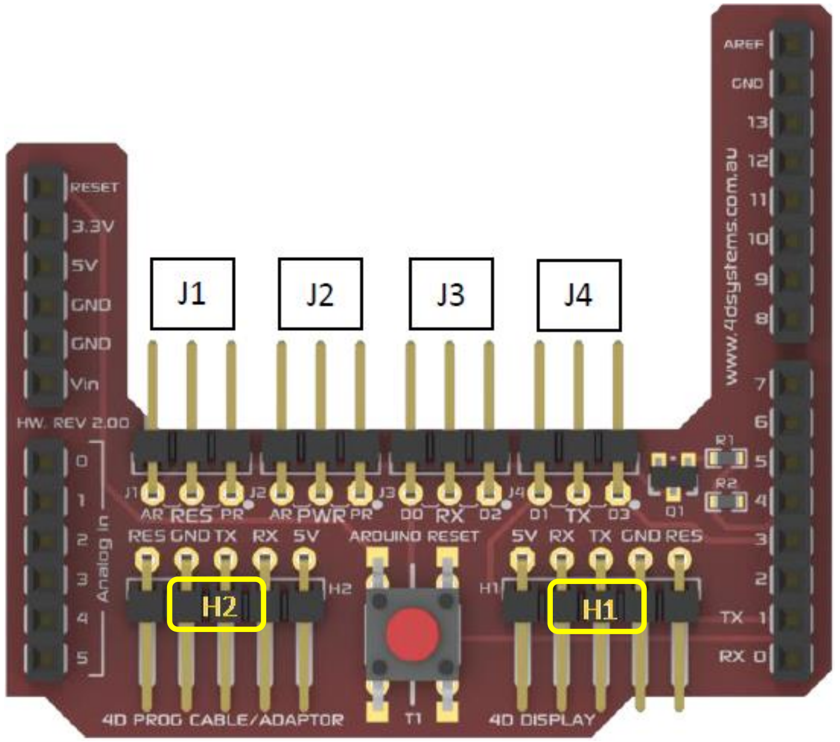
GENIE_READ_OBJ	0
GENIE_WRITE_OBJ	1
GENIE_WRITE_STR	2
GENIE_WRITE_STRU	3
GENIE_WRITE_CONTRAST	4
GENIE_REPORT_OBJ	5
GENIE_REPORT_EVENT	7

In conclusion the program evaluates each byte of a message (command, object, object index, and value), and then makes a decision according to the result of this evaluation.

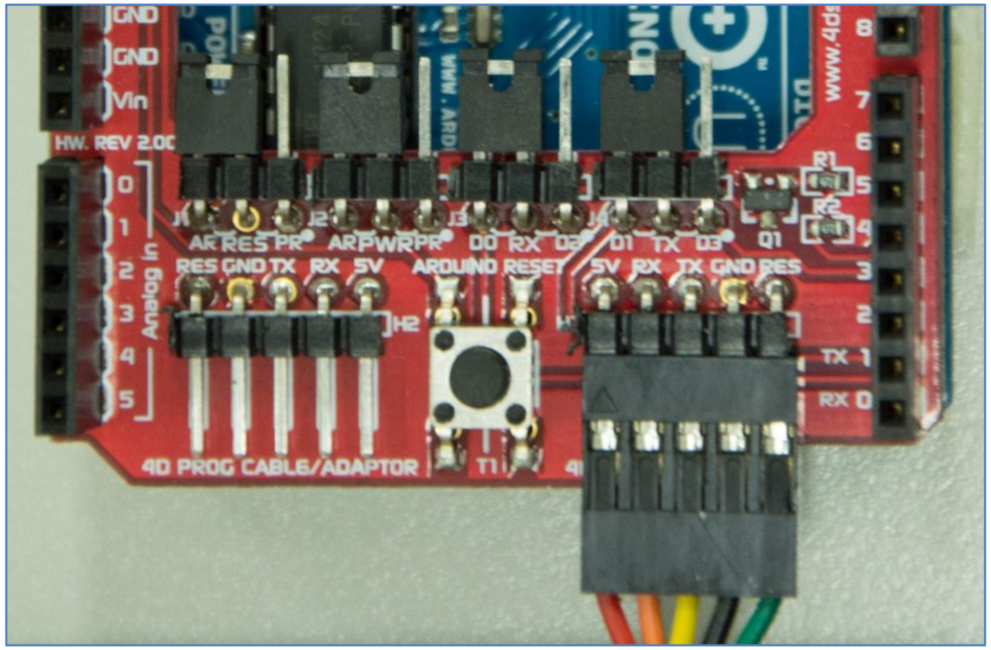
Connect the 4D Display Module to the Arduino Host

This section discusses how to connect the display module to the Arduino host. The user has the option of using a 4D Arduino Adaptor Shield or simply connecting the Tx0, Rx0, RESET, and GND pins of the display module to the corresponding pins of the Arduino host.

Using the New 4D Arduino Adaptor Shield (Rev 2.00)



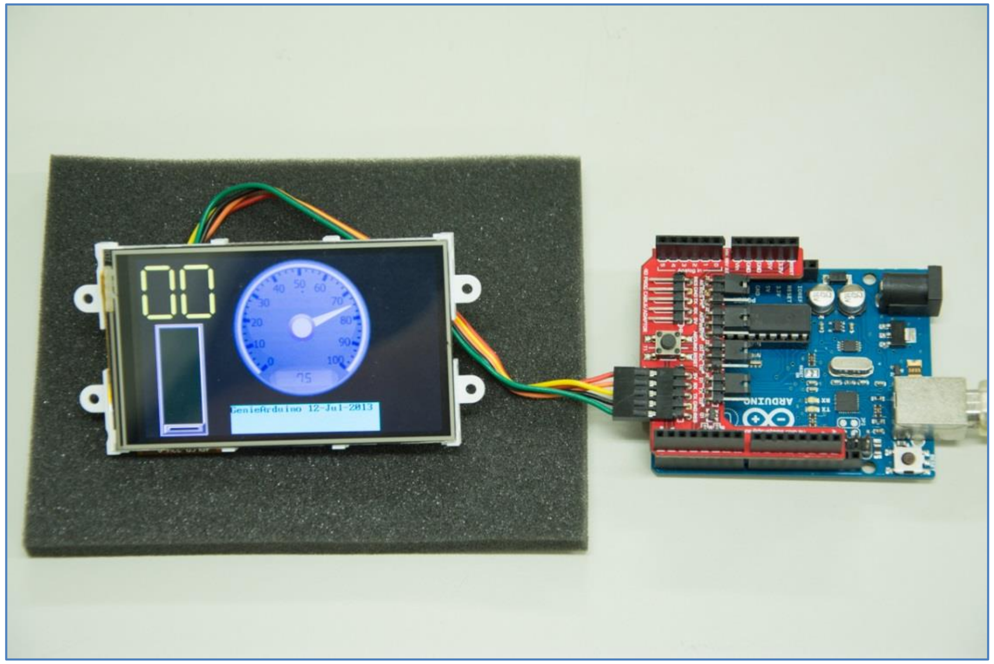
To reset the display using the D4 pin of the Arduino and to power the display from the Arduino host, set jumpers J1 and J2 as shown below.



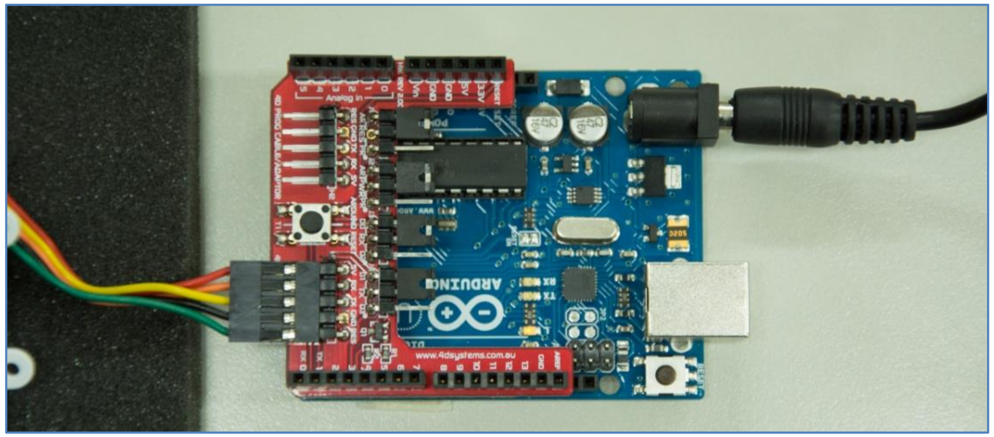
By default, jumpers J3 and J4 are configured to connect RX (RX of display module) to D0 (TX0 of the Arduino) and TX (TX of display module) to D1 (RX0 of the Arduino). To use the other serial ports of the Mega or Due, remove the jumper connectors of J3 and J4 and connect the display TX and RX pins to the desired serial port using jumper wires.

Below are images of a complete setup wherein the display module and the Arduino host have a common power supply. Note that the power supply must be able to supply enough current for both the display module and the Arduino host. Refer to the display module's datasheet for the specified supply current.

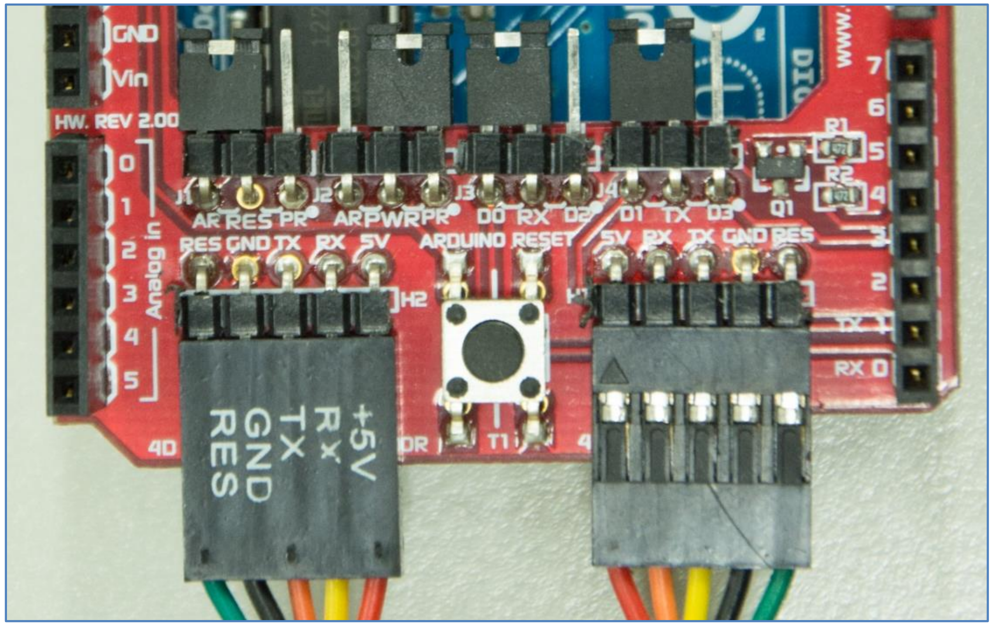
Using the USB cable:



Using the jack:

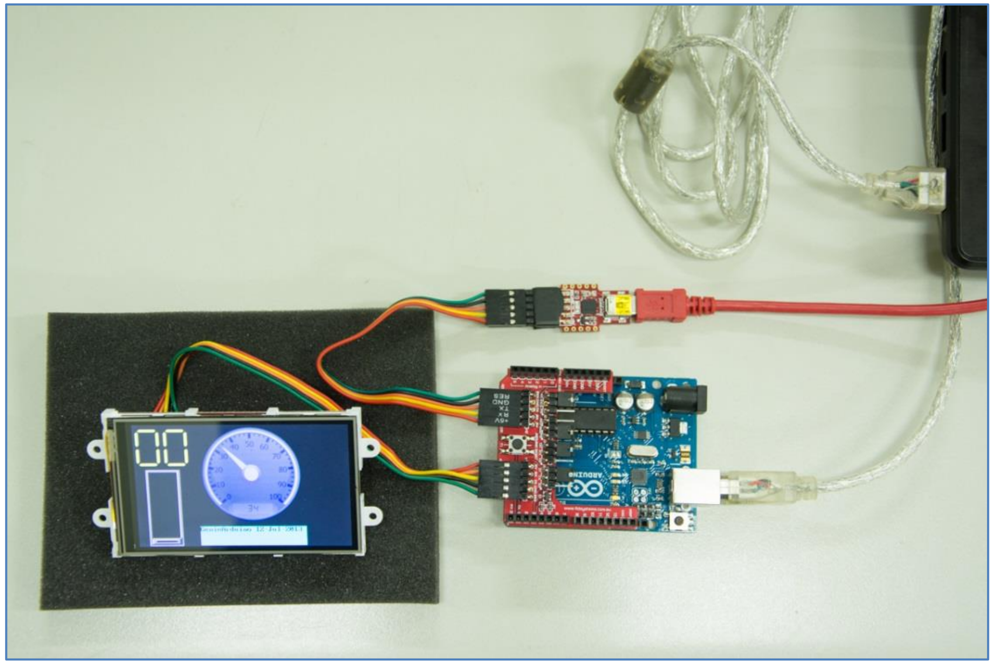


To power the display using a 4D USB Programming Cable or a µUSB-PA5 Programming Adaptor, set J2 as shown below.

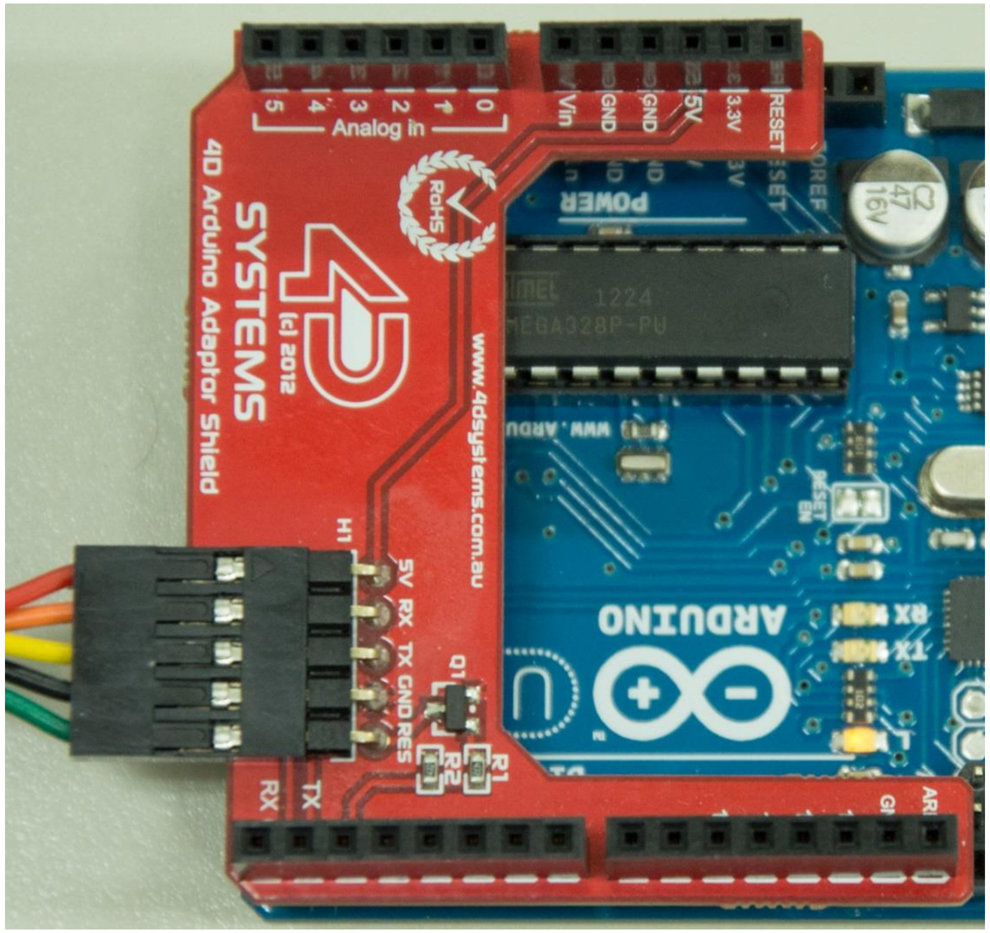


H2 is for the 4D USB Programming Cable or µUSB-PA5 and H1 is for the display module. Note that the display module cannot be programmed in this setup since H2 transfers power only. Always disconnect the display module from the Arduino Adaptor Shield first before programming it. Always check the orientation of the connectors.

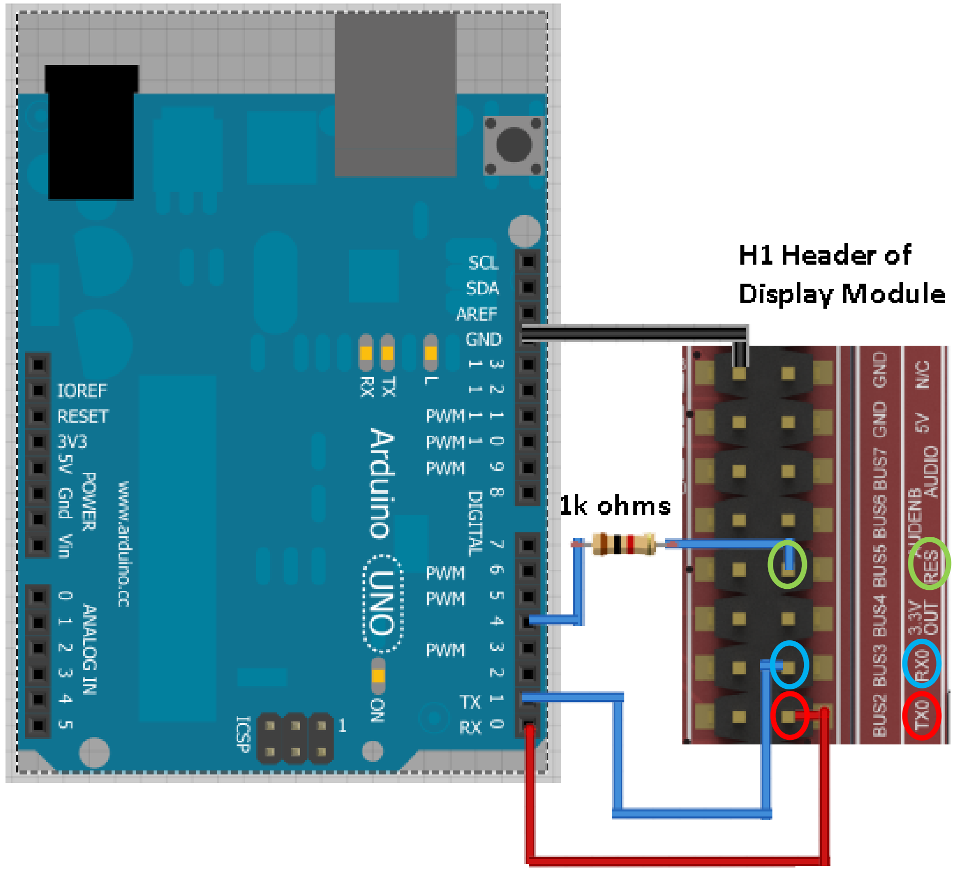
Complete setup:

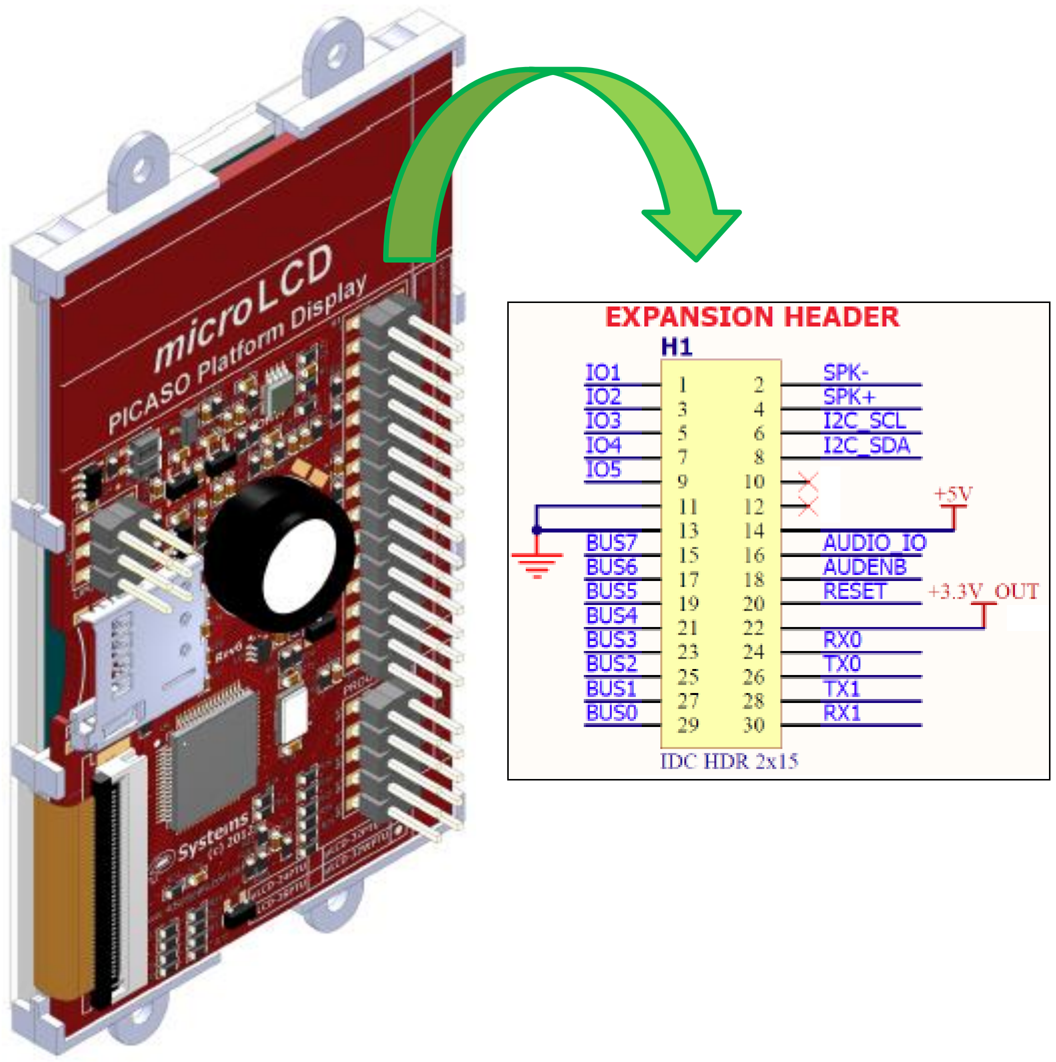


Using the Old 4D Arduino Adaptor Shield (Rev 1)



Connection Using Jumper Wires





Proprietary Information

Disclaimer of Warranties & Limitation of Liability