APPLICATION NOTES

4D SYSTEMS
TURNING TECHNOLOGY INTO ART

# ViSi-Genie Password Implementation with an Arduino Host

DOCUMENT DATE: **25th APRIL 2019**
DOCUMENT REVISION: **1.1**

## Description

This application note shows how to write a sketch for an Arduino program that prints characters on string object from keyboard object to create a simple Password Implemented Application . 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.

This application note requires:

- Any of the following 4D Picaso and gen4 Picaso display modules:

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

and other superseded modules which support the ViSi Genie environment

- 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 |

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 / uUSB-PA5/uUSB-PA5-II
  for non-gen4 displays(uLCD-xxx)
- 4D Programming Cable & gen4-PA, / gen4-IB / 4D-UPA
  for gen4 displays (gen4-uLCD-xxx)
- 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



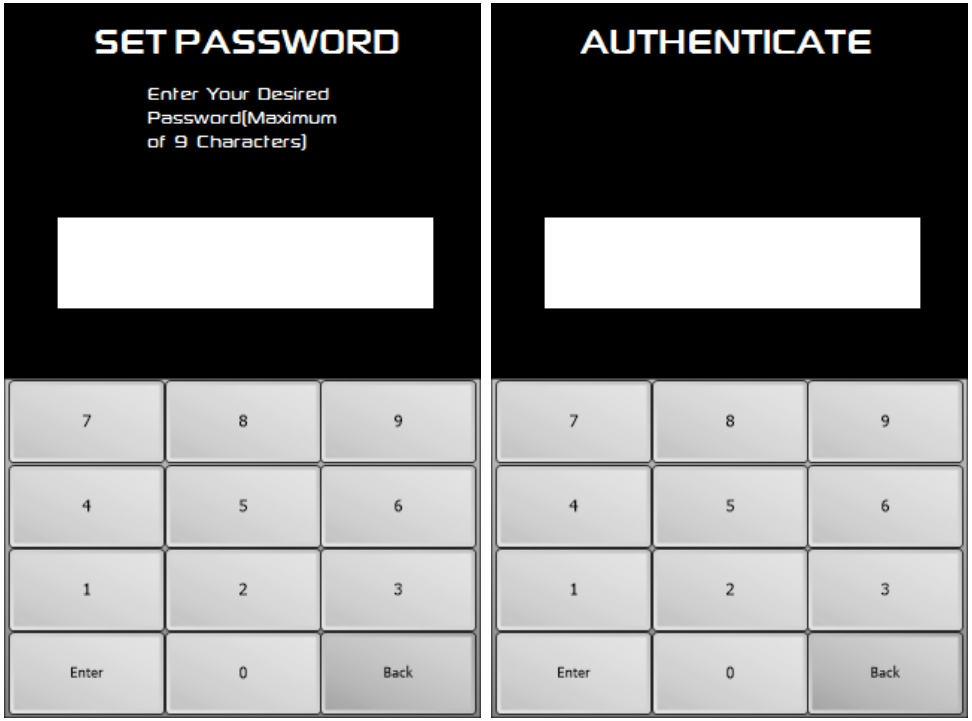
## Application Overview

The application developed in this document works in two forms. First form is to Set/Save a desired password, the second form compares any inputted password to the saved password in the first form.

The Arduino host is programmed in the Arduino IDE to perform gathering of input from Keyboard object and then stores it in array. The input is also written on a string object. The arduino then compares two arrays and if correct it will print "Correct Password" in the string object, and if not it will print "Incorrect Password".

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.

**OUTPUT:**




## 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

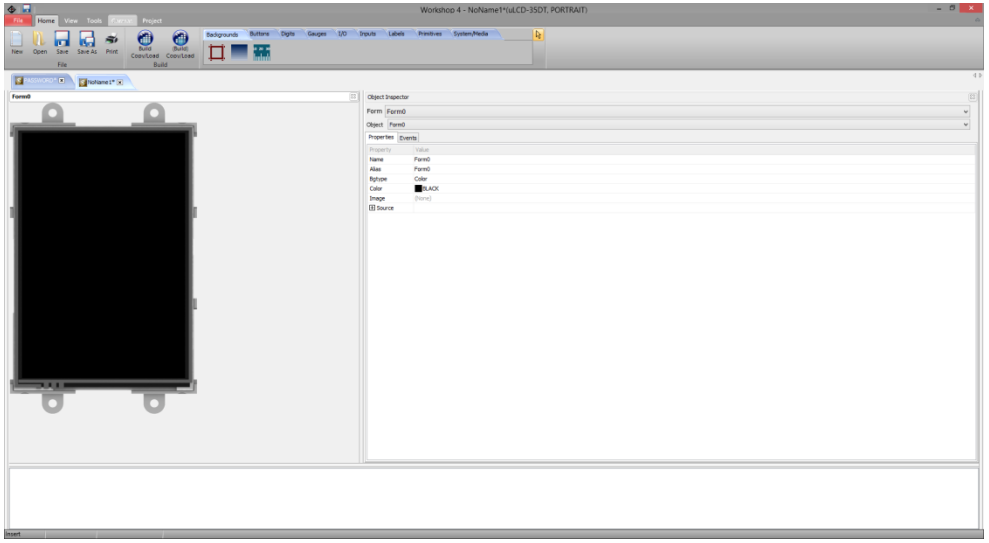
### Create a New Project

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

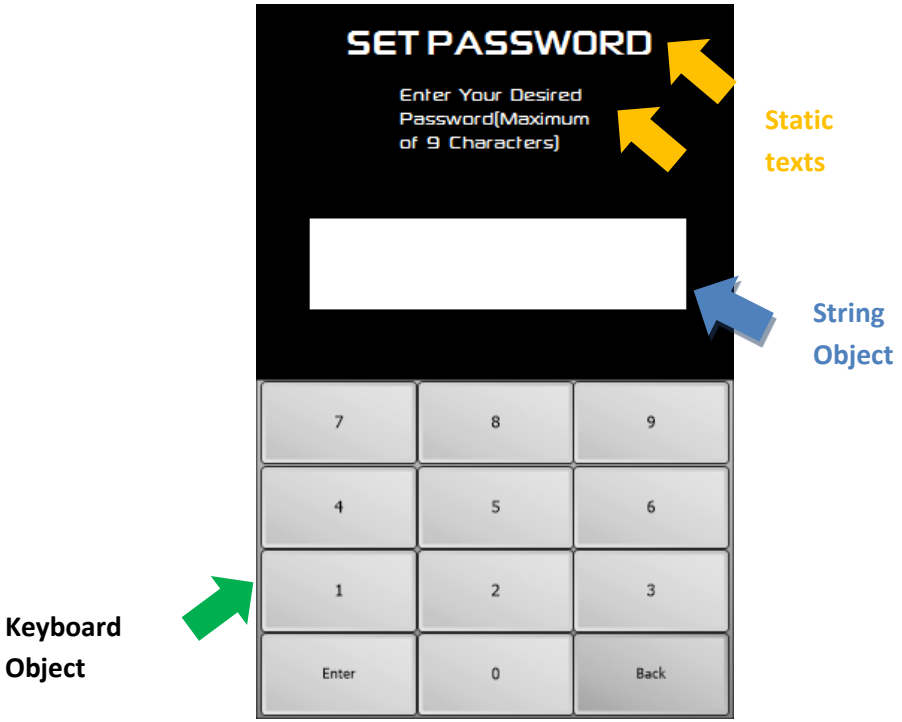
**ViSi Genie Getting Started – First Project for Picaso Displays** (for Picaso)
or
**ViSi Genie Getting Started – First Project for Diablo16 Displays** (for Diablo16).

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

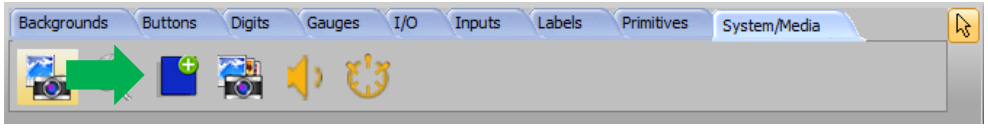


At the end of this section, the user will able to create a form with four objects. The final form will look like as shown below, with the labels excluded.

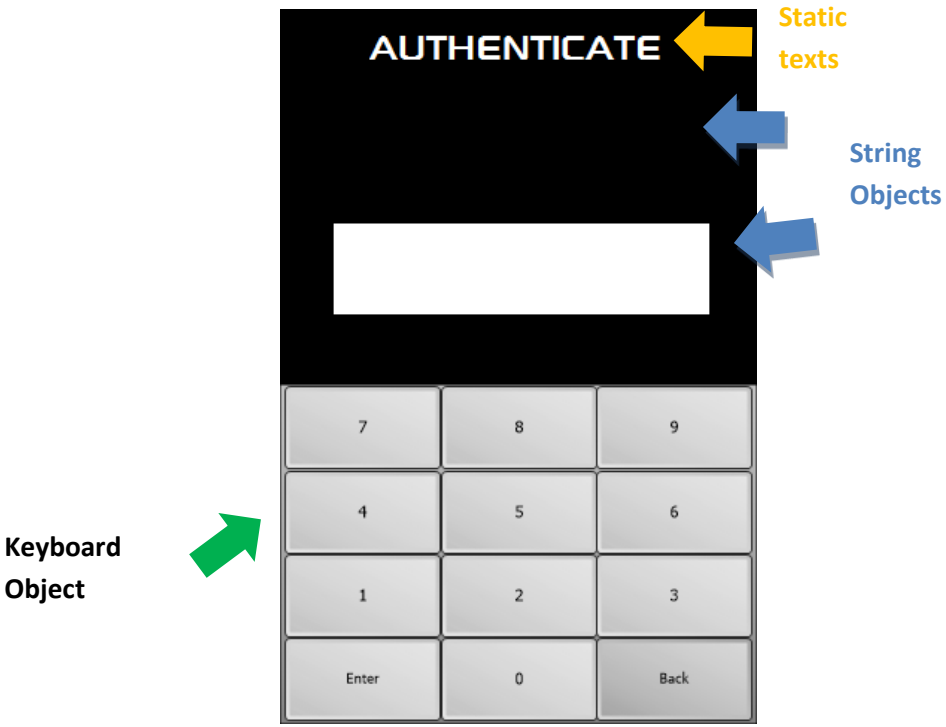



### Create a new Form

The **Form** object will create a new Form. To add a FORM object, go to the **System/Media** pane and select the third icon.



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.




## Object Properties

The images shown below are the properties of the objects that are used in the project included in this application note.

### StaticText0 and Statictext2

Form: Form0
Object: Statictext0

Properties | Events

| Property | Value |
|---|---|
| Name | Statictext0 |
| Alias | Statictext0 |
| AutoSize | Yes |
| Caption | SET PASSWORD |
| Color | BLACK |
| ⊞ Font | (WHITE, [], Sony Sketch EF, 24, [Bold]) |
| Height | 32 |
| Left | 60 |
| Top | 8 |
| Transparent | Yes |
| Width | 209 |
| WordWrap | Yes |

### StaticText1

Form: Form0
Object: Statictext1

Properties | Events

| Property | Value |
|---|---|
| Name | Statictext1 |
| Alias | Statictext1 |
| AutoSize | Yes |
| Caption | Enter Your Desired Password(Maximum of 9 Characters) |
| Color | BLACK |
| ⊞ Font | (WHITE, [], Sony Sketch EF, 12, []) |
| Height | 48 |
| Left | 96 |
| Top | 52 |
| Transparent | Yes |
| Width | 127 |
| WordWrap | Yes |

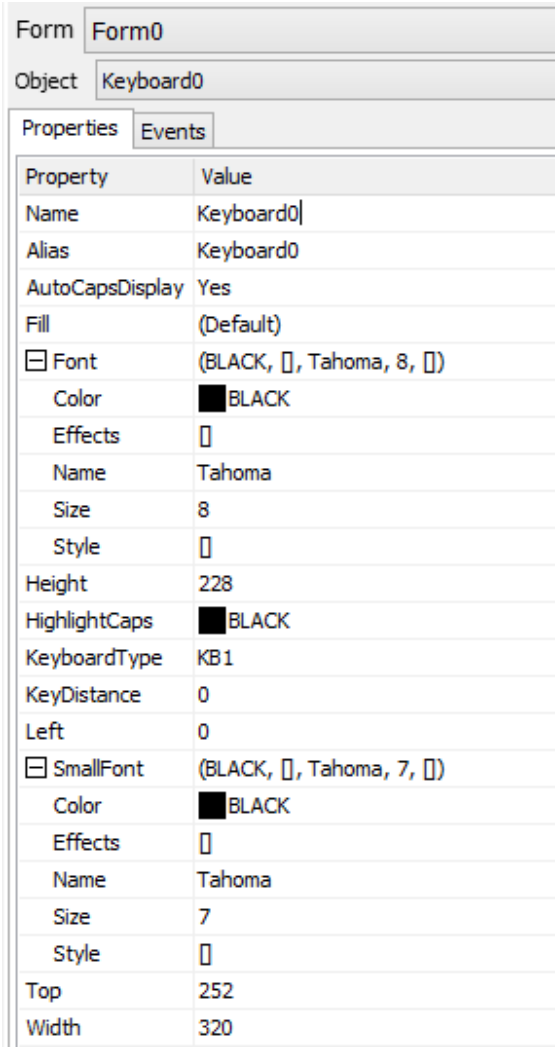
### Strings

Form: Form0
Object: Strings0

Properties | Events

| Property | Value |
|---|---|
| Name | Strings0 |
| Alias | Strings0 |
| Alignment | Left |
| BGcolor | WHITE |
| Height | 61 |
| FGcolor | BLACK |
| ⊟ Font | |
| Bold | No |
| CharSet | ANSI |
| Italic | No |
| Name | Sony Sketch EF |
| Opaque | Yes |
| Size | 34 |
| Strikethrough | No |
| Underline | No |
| Left | 36 |
| Strings | \n |
| StringsStyle | Message |
| Top | 144 |
| Width | 252 |

### Keyboard




For in depth details on customizing a Keyboard object : ViSi-Genie Customised Keyboard

## 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.

# Writing the Host Code

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

The data or message is received and queued by:

```
//Process events
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().**

## Event Handler

```
void myGenieEventHandler(void)
{
  genieFrame Event;
  genie.DequeueEvent(&Event);

  //If the cmd received is from a Reported Event (Events tr
  if (Event.reportObject.cmd == GENIE_REPORT_EVENT)
  {
    if(Event.reportObject.object == GENIE_OBJ_KEYBOARD)
    {
```

This part of the code is the event handler. The image above checks if the report message comes from a keyboard object.

### Keyboard Object 0

```
if (Event.reportObject.index == 0)
{
  temp = genie.GetEventData(&Event);
  if(temp >= 48 && temp <= 57 && counter <=9)
  {
    keyvalue[counter] = temp;        // Receive t
    genie.WriteStr(0,keyvalue);
    counter = counter + 1; //increment array
  }
  else if(temp == 8)
  {
    counter--;
    keyvalue[counter] = 0;
    genie.WriteStr(0,keyvalue);
  }
  else if(temp == 13)
  {
    genie.WriteObject(GENIE_OBJ_FORM,0x01,0);
  }
}
```

If the Key is pressed in keyboard0 object this part of the program will execute. This simply prints to the string objects the key that is pressed from the keyboard while storing it in an array.

If "Back" is pressed then the value that will be received from "genie.GetEventData(&Event)" is 8, decumal value of backspace in ASCII Code. The array Index will be decremented and NULL will be printed.

If "Enter" is pressed then the value that will be received from "genie.GetEventData(&Event)" is 13, decimal value of carriage return in ASCII Code. This will write to FORM1 so the display will display FORM1 where keyboard1 exist and AUTHENTICATION starts.

### Keyboard Object 1

```
if (Event.reportObject.index == 1)
{
  temp2 = genie.GetEventData(&Event);
  if(temp2 >= 48 && temp2 <= 57 && counter2 <=9)
  {
    newkeyvalue[counter2] = temp2;
    asterisk[counter2] = temp2;
    genie.WriteStr(1,asterisk);
    asterisk[counter2] = '*';
    counter2 = counter2 + 1;
    delay(200);
    genie.WriteStr(1,asterisk);
  }
  else if(temp2 == 8 && counter2 >0)
    {
      counter2--;
      newkeyvalue[counter2] = 0;
      asterisk[counter2] = ' ';
      genie.WriteStr(1,asterisk);
    }
```

If the Key is pressed in keyboard1 object this part of the program will execute. Same process with keyboard 1 if a key/number or Back is pressed. Their difference is that when a key is pressed in keyboard2 the number will be printed on the string object and will be replaced by '*' asterisk after.

```
else if(temp2 == 13)
{
  for(int x=0; x<9;x++)
  {
    if(keyvalue[x] == newkeyvalue[x])
    {
      flag = 1;
    }
    else
    {
      flag = 0;
    }
    if(flag == 0)
    {
      break;
    }
  }
  if(flag == 0)
  {
    genie.WriteStr(2,"INCORRECT PASSWORD");
    delay(2000);

  }
  else if(flag == 1)
  {
    genie.WriteStr(2,"CORRECT PASSWORD");
    for(int y=0;y<9;y++)
    {
      newkeyvalue[y] = 0;
      asterisk[y] = 0;
    }
    delay(2000);
    flag = 0;
    counter2 = 0;
    genie.WriteStr(1,asterisk);
    genie.WriteObject(GENIE_OBJ_FORM,0x01,0);
  }
```

The difference in operation between keyboard0 and keyboard1 is when "Enter" key is pressed. The code presented compares the arrays that holds the value inputted in keyboard0 and keyboard1. If the password is correct then it will write on the string object "CORRECT PASSWORD" else if the password inputted is incorrect it will write on the string object "INCORRRECT PASSWORD".

## 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

## Proprietary Information



## Disclaimer of Warranties & Limitation of Liability