



**AN\_318**

**ARDUINO LIBRARY FOR FT800**

**SERIES**

**Version 1.1**

**Issue Date: 2014-07-14**

Use of FTDI devices in life support and/or safety applications is entirely at the user's risk, and the user agrees to defend, indemnify and hold FTDI harmless from any and all damages, claims, suits or expense resulting from such use.

**Future Technology Devices International Limited (FTDI)**

Unit 1, 2 Seaward Place, Glasgow G41 1HH, United Kingdom

Tel.: +44 (0) 141 429 2777 Fax: + 44 (0) 141 429 2758

Web Site: <http://ftdichip.com>

Copyright © 2014 Future Technology Devices International Limited

## **Table of Contents**

1	Introduction .....	4
1.1	Scope .....	4
2	Library Installation .....	5
2.1	Prerequisite .....	5
2.1.1	Hardware: .....	5
2.1.2	Software: .....	5
2.2	Install the Library .....	6
2.2.1	Automatic Installation .....	6
2.2.2	Manual Installation .....	7
3	Library Folder Structure .....	8
3.1	FT800Impl class .....	10
3.2	FT801Impl class .....	11
3.3	FT_GC class .....	12
3.4	FT_RTC class .....	20
3.5	FT_SD class .....	22
3.5.1	FT_SDFile class .....	22
3.5.2	FT_SD .....	24
4	Supported Platforms .....	26
5	Library Usage .....	28
5.1	Platform Setup .....	30
5.1.1	Library Includes .....	30
5.1.2	Global Object Declaration .....	31
5.1.3	Start-up Configuration .....	31
5.1.4	Start-up Configuration for FT801 chip .....	32
5.2	FT_App_Imageviewer Setup .....	33
5.3	FT_App_Imageviewer SD Card Usage .....	34
5.4	FT_App_Imageviewer Functionality .....	34
5.5	FT_App_Imageviewer Bitmap Handling .....	35
5.6	FT_App_Imageviewer Image Reflection .....	35
6	Hardware Setup .....	37
6.1	VM800P35 .....	37
6.2	VM800P43_50 .....	37
6.3	VM801P43_50 .....	37
6.4	ADAM_4DLCD_FT843 .....	37
6.5	FT_Breakout_4DLCD_FT843 .....	39
6.6	VM800B43_50 .....	41
6.7	VM800B35 .....	44

7	Contact Information .....	45
	Appendix A- References .....	46
	Document References .....	46
	Acronyms and Abbreviations .....	46
	Appendix B – List of Tables & Figures .....	47
	List of Figures .....	47
	Appendix C- Revision History .....	48

## 1 Introduction

What is EVE? EVE, or the Embded Video Engine, is a family of ICs designed to control TFT displays. FT800 and FT801 are the first two devices in this series which in addition to controlling the display also includes embedded support for touch control and audio output. The devices are controlled over a low bandwidth SPI or I<sup>2</sup>C interface allowing for interfacing to nearly any microcontroller with a SPI or I<sup>2</sup>C master port. Simple and low-pin-count microcontrollers can now offer a high-end graphical user interface by using the FT800 series graphics controller with EVE technology.

### 1.1 Scope

This document can be used as a guide by programmers to develop GUI applications using the FT800 series graphics controller with Arduino development boards via SPI. Users not familiar with the FTDI graphics controller should refer to the following documents:

- [FT800 datasheet](#)
- [FT801 datasheet](#)
- [Programming Guide covering EVE command language](#)
- [AN\\_240 FT800 From the Ground Up](#)
- FTDI [VM800B](#) "basic" development modules - comes with a display and a component board in a plastic enclosure. These modules are designed to control 3.5", 4.3" or 5" TFT displays and they are using the FT800 graphics controller.
- FTDI [VM800P](#) "plus" development modules - In addition to the features supplied with the basic modules, the plus boards are also equipped with an MCU, the ATmega328p, RTC with battery backup, an SD card port, and expansion IO pins. These modules are using the FT800/FT801 graphics controller.
- Capacitive touch controllers supported by the FT801 include: [azoteg](#) and [focaltech](#)

## 2 Library Installation

### 2.1 Prerequisite

#### 2.1.1 Hardware:

A supported Arduino board is required for use with the FTDI Chip libraries. Please consult the "Supported Platforms" section of this document for the targeted platforms, as well as a listing of those utilized during library verification. Note that additional functionality and platforms may be added in the future.

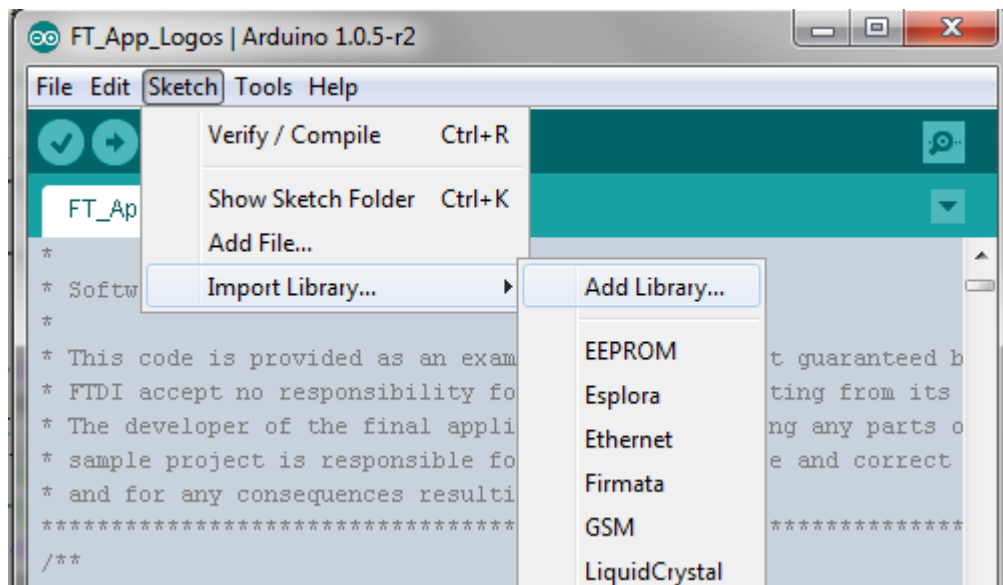
#### 2.1.2 Software:

- A copy of the library is available at:  
[http://www.ftdichip.com/Support/SoftwareExamples/FT800\\_Projects.htm](http://www.ftdichip.com/Support/SoftwareExamples/FT800_Projects.htm)
- The correct version of Arduino IDE for the development board.
  - Arduino IDE version 1.5.6-r2 BETA is needed for Arduino Yun and Due.  
<http://arduino.cc/en/main/software#toc3>
  - Intel Galileo Arduino SW is need for the Arduino Galileo.  
[https://downloadcenter.intel.com/Detail\\_Desc.aspx?DwnldID=23171](https://downloadcenter.intel.com/Detail_Desc.aspx?DwnldID=23171)
  - Arduino IDE version 1.0.4 or 1.0.5-r2 is recommended for all other boards.  
<http://arduino.cc/en/main/software#toc3>

## 2.2 Install the Library

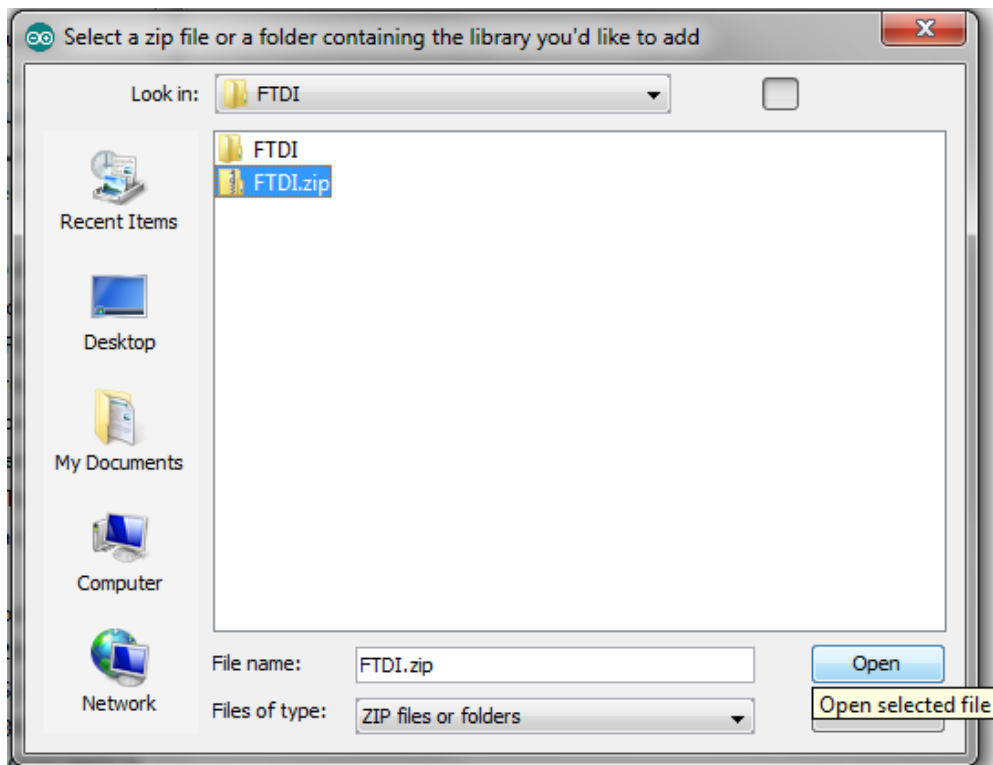
### 2.2.1 Automatic Installation

The automatic method will install the library by the IDE and the library itself can be a ZIP file or unzipped folder. This method will install the library to the location specified in the IDE's Sketchbook location. The default directory, on OSX, would be at "`~/Documents/Arduino/`". On Windows, it would be at "`My Documents\Arduino\`". For more information on library installation, please refer to <http://arduino.cc/en/Guide/Libraries>.



**Figure 2-1 : Arduino IDE - add library option**

In the Arduino IDE, click "Sketch", hover over the "Import Library" option in the drop down menu, and then click the "Add Library..." item.

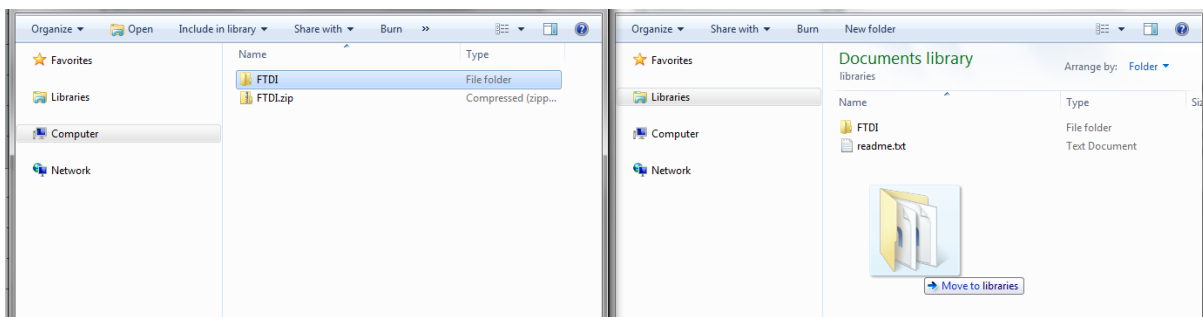


**Figure 2-2 : Arduino IDE - library selection browser**

Navigate to the downloaded FTDI library file or unzipped folder and open it. The IDE will automatically install the files to the library folder of the Sketchbook location.

### 2.2.2 Manual Installation

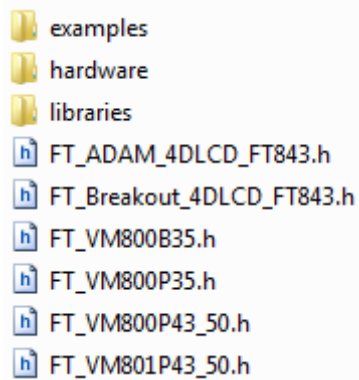
To manually install the library, simply unzip the library and put the FTDI folder in the library folder of the IDE's Sketchbook location. The Sketchbook location is in the preferences window accessed by choosing *File->Preferences*. Users will need to manually create a folder called "libraries" in the Sketchbook location if it doesn't contain one.



**Figure 2-3 : Library Installation - Manual Installation**

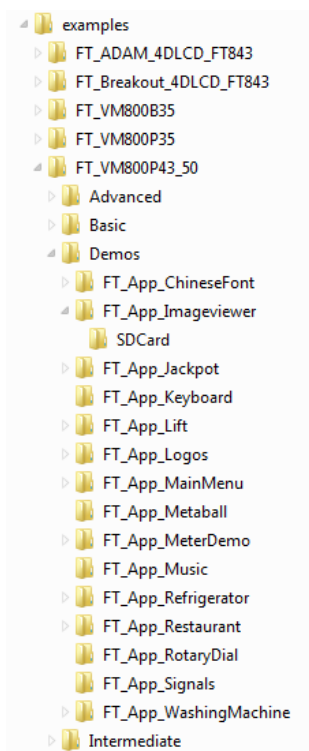
### 3 Library Folder Structure

The library files are shown below:



**Figure 3-1 : FTDI Library folder contents**

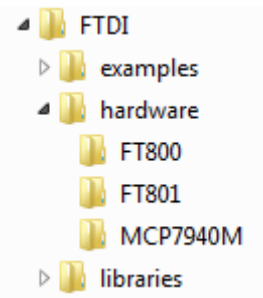
The contents of the FTDI library folder are as shown above.



**Figure 3-2 : FTDI Library - examples folder**

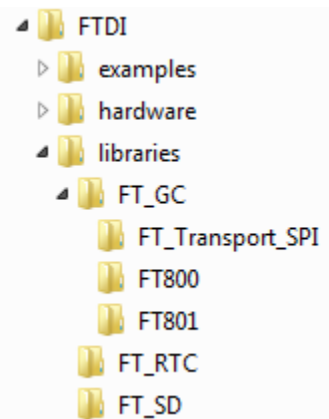
The examples folder contains example sketches in each of the sub folders of the respective platform. Sketches are organized by increasing difficulty. If there is an *SDCard* folder within a sketch folder then the contents of the folder need to be copied to the root level of the SD card before running the sketch.





**Figure 3-3 : FTDI Library – hardware folder**

The hardware folder contains hardware specific macros for the FT800 series graphics controller.



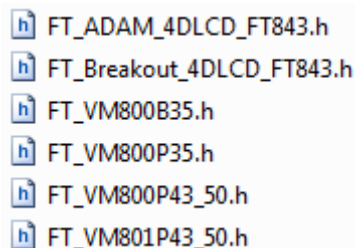
**Figure 3-4 : FTDI Library - libraries folder**

The libraries folder contains APIs to access hardware functionalities.

"FT\_GC" - This library contains the APIs to access all the features of the FTDI EVE graphics controller.

"FT\_RTC" - This library contains the APIs to access the RTC on the FTDI Plus boards.

"FT\_SD" - This library contains the APIs to access the SD card on FTDI Plus boards and the ADAM\_4DLCD\_FT843.



**Figure 3-5 : FTDI Library - platform headers**

Each of these platform headers includes hardware specific macros and functionalities available only to the platform. Include only one of these headers in the sketch.

### 3.1 FT800Impl class

The FT800Impl, in the FT\_GC/FT800Impl folder, is a Subclass of the FT\_GC class which implements the functionalities that are specific to the FT800 graphics controller with resistive touch capability. Refer to Hardware Setup chapter for more information on hardware details such as CS pin number, PDN pin number, Display pin etc.

#### Available Objects

None.

#### APIs

Prototype	Description
FT800Impl(void)	API to use default pin for cs, pdn. No interrupt pin assignment and channel initialization.
FT800Impl(uint8_t csPin,uint8_t pdnPin)	API to use specified cs and pdn pin.
FT800Impl(uint8_t csPin,uint8_t pdnPin,uint8_t IntPin)	API to use specified cs, pdn, and interrupt pin. Interrupt channel will also be initialized.
FT_Status Init(uint8_t ResType)	Initialization API with the specified display resolution.
FT_Status Init(uint16_t hperiod,uint16_t vperiod,uint16_t hfrontporch,uint16_t hbackporch,uint16_t hpulsewidth, uint16_t vfrontporch,uint16_t vbackporch,uint16_t vpulsewidth, uint8_t polarity,uint8_t swizzle,uint8_t fps)	API to program the output display wrt display parameters.
FT_Status Exit()	End SPI connection
void GetVersion(uint8_t &Major, uint8_t &Minor, uint8_t &Build)	API to get version of the library
uint8_t SetIntMask(uint8_t IntMask)	APIs related to graphics processor
uint8_t IsPendown()	This function reports whether the touch screen is currently being touched or not. 1 for touch detected and 0 for no touch detected.
void SetTouch(uint8_t AdcMode,uint16_t Charge,uint8_t Settle,uint16_t Oversample, uint16_t RZTreshold)	set touch parameters

## 3.2 FT801Impl class

The FT801Impl, in the FT\_GC/FT801Impl folder, is a Subclass of the FT\_GC class which implements the functionalities that are specific to the FT801 graphics controller with capacitive touch capability. Refer to Hardware Setup chapter for more information on hardware details such as CS pin number, PDN pin number, Display pin etc.

### Available Objects

None.

### APIs

Prototype	Description
FT801Impl(void)	API to use default pin for cs, pdn. No pin will be assigned for the interrupt pin.
FT801Impl(uint8_t csPin,uint8_t pdnPin)	API to use specified cs and pdn pin.
FT800Impl(uint8_t csPin,uint8_t pdnPin,uint8_t IntPin)	API to use specified cs, pdn, and interrupt pin.
FT_Status Init(uint8_t ResType ,uint16_t options1)	Initialization API with the specified display resolution and clock source.
FT_Status Init(uint16_t hperiod,uint16_t vperiod,uint16_t hfrontporch,uint16_t hbackporch,uint16_t hpulsewidth, uint16_t vfrontporch,uint16_t vbackporch,uint16_t vpulsewidth, uint8_t polarity,uint8_t swizzle,uint8_t fps , uint16_t options1)	API to program the output display wrt display parameters.
FT_Status Exit()	End SPI connection
void GetVersion(uint8_t &Major, uint8_t &Minor, uint8_t &Build)	API to get version of the library
uint8_t SetIntMask(uint8_t IntMask)	APIs related to graphics processor
uint8_t IsPendown()	This function reports whether the touch screen is currently being touched or not. 1 for touch detected and 0 for no touch detected.
void SetCTouchMode(uint8_t	Set capacitive touch mode. 1 for compatibility mode and 0

CTPMode)	for extended mode.
void GetCTouchXY(sCTouchXY &sCTouchxy)	API to read the touch registers.
FT_GEStatus Cmd_CSketch(int16_t x, int16_t y, uint16_t w, uint16_t h, uint32_t Ptr, uint16_t Format, uint16_t Freq)	Sketch API for capacitive touch controller. x,y,w,h parameters determine the location and area of sketch. Ptr is output bitmap location in graphics ram. Format to specify L8 or L1 output format. Freq for interpolation of bitmap points between touch points and range is from 0 to 1500.

### 3.3 FT\_GC class

The FT\_GC class, in the FT\_GC folder, implements the graphical functionalities of the FT800 series graphics controller.

#### Available Objects

- sTagXY - The members of this structure contains the touched screen coordinates and the tag value of the tagged object at that location, if any. Passing an instance of this structure to the *FT\_GC::GetTagXY* function will update all the members.
- sTrackTag - The members of this structure contain the tag and calculated tracking value of the tracked object, if any. Passing an instance of this structure to the *FT\_GC::GetTrackTag* function will update all the members.
- FT\_Fonts\_t - This font table structure is for both ROM and RAM fonts. There are 16 continuous ROM font tables starting at FT\_FONT\_TABLE\_POINTER location; each of the ROM font table occupies 148 bytes. More information on ROM and RAM font can be found in the FT800 Series Programmers Guide.
- FT\_AEStatus - Audio coprocessor related enums.
- FT\_Status - FT\_GC status enums, used for API return type, error type etc.
- FT\_GEStatus - Status enums for graphics engine.
- FT\_TESStatus - Touch coprocessor related enums.
- FT\_DISPLAY\_QVGA\_320x240 - This macro can be used to compare against the FT\_DISPLAY\_RESOLUTION macro in the platform header to determine if the current screen size is QVGA.
- FT\_DISPLAY\_WQVGA\_480x272 - This macro can be used to compare against the FT\_DISPLAY\_RESOLUTION macro in the platform header to determine if the current screen size is WQVGA.

**APIs**

Prototype	Description
/* Initialization and setup API */	
FT_GC()	Default constructor
~FT_GC();	Default destructor
FT_Status Init(uint8_t ResType,uint16_t options1)	API to set the resolution of output display and clock selection.
FT_Status Init(uint16_t hperiod,uint16_t vperiod,uint16_t hfrontporch,uint16_t hbackporch,uint16_t hpulsewidth,uint16_t vfrontporch,uint16_t vbackporch,uint16_t vpulsewidth, uint8_t polarity,uint8_t swizzle,uint8_t fps, uint16_t options1)	API to program the output display wrt display parameters and clock selection.
FT_Status Exit()	Exit state of the graphics controller.
void GetVersion(uint8_t &Major, uint8_t &Minor, uint8_t &Build)	API to get version of the library
void SetDisplayEnablePin(uint8_t GpioBit)	FT_GC GPIO bit for display enable/disable
void SetAudioEnablePin(uint8_t GpioBit)	FT_GC GPIO bit for audio enable/disable
void DisplayOn(void)	Enable backlight
void DisplayOff(void)	Disable backlight
void AudioOn(void)	Enable audio
void AudioOff(void)	Disable audio
void SetInterruptPin(uint16_t Intpin)	Set the interrupt pin
void ResetCopro(void)	Reset only coprocessor
void Reset(void)	Reset the graphics controller via pdn - if pdn is not assigned then reset will not be successful
/* APIs related to power up/power down functionality */	
void DisplayConfigExternalClock(uint8_t ResType);	API to configure display and switch to external clock.
void ActiveInternalClock(void)	API to configure internal clock.

Prototype	Description
void PDN_Cycle(void)	API to power cycle the graphics controller.
/* APIs related to graphics processor */	
void EnableInterrupts(uint8_t GEnable, uint8_t Mask)	Enable or disable interrupts with interrupt source mask.
uint8_t ReadIntReg(void)	Read the interrupt flag register - the register is clear on read
/* APIs related to graphics engine */	
FT_GEStatus AlphaFunc(uint8_t Func, uint8_t Ref)	Refer to the corresponding API in the FT800 Series Programmer's guide.
FT_GEStatus Begin(uint8_t Prim)	""
FT_GEStatus BitmapHandle(uint8_t Handle)	""
FT_GEStatus BitmapLayout(uint8_t Format, uint16_t Linestride, uint16_t Height);	""
FT_GEStatus BitmapSize(uint8_t Filter, uint8_t wrapx, uint8_t wrapy, uint16_t width, uint16_t height)	""
FT_GEStatus BitmapSource(uint32_t Addr)	""
FT_GEStatus BitmapTransformA(int32_t A)	""
FT_GEStatus BitmapTransformB(int32_t B)	""
FT_GEStatus BitmapTransformC(int32_t C)	""
FT_GEStatus BitmapTransformD(int32_t D)	""
FT_GEStatus BitmapTransformE(int32_t E)	""
FT_GEStatus BitmapTransformF(int32_t F)	""
FT_GEStatus BlendFunc(uint8_t Src, uint8_t Dst)	""
FT_GEStatus Call(uint16_t Dest)	""
FT_GEStatus Cell(uint8_t Cell)	""
FT_GEStatus ClearColorA(uint8_t Alpha)	""
FT_GEStatus ClearColorRGB(uint8_t red, uint8_t green, uint8_t blue)	""

Prototype	Description
FT_GEStatus ClearColorRGB(uint32_t rgb)	""
FT_GEStatus Clear(uint8_t c, uint8_t s, uint8_t t)	""
FT_GEStatus Clear(void)	""
FT_GEStatus ClearStencil(uint8_t s)	""
FT_GEStatus ClearTag(uint8_t s)	""
FT_GEStatus ColorA(uint8_t Alpha)	""
FT_GEStatus ColorMask(uint8_t r, uint8_t g, uint8_t b, uint8_t a)	""
FT_GEStatus ColorRGB(uint8_t red, uint8_t green, uint8_t blue)	""
FT_GEStatus Display(void)	""
FT_GEStatus End(void)	""
FT_GEStatus Jump(uint16_t Dest)	""
FT_GEStatus LineWidth(uint16_t Width)	""
FT_GEStatus Macro(uint8_t m)	""
FT_GEStatus PointSize(uint16_t Size)	""
FT_GEStatus RestoreContext(void)	""
FT_GEStatus Return(void)	""
FT_GEStatus SaveContext(void)	""
FT_GEStatus ScissorSize(uint16_t width, uint16_t height)	""
FT_GEStatus ScissorXY(uint16_t x, uint16_t y)	""
FT_GEStatus StencilFunc(uint8_t Func, uint8_t Ref, uint8_t Mask)	""
FT_GEStatus StencilMask(uint8_t Mask)	""
FT_GEStatus StencilOp(uint8_t Sfail, uint8_t Spass)	""
FT_GEStatus TagMask(uint8_t Mask)	""
FT_GEStatus Tag(uint8_t s)	""

Prototype	Description
FT_GEStatus Vertex2f(int16_t x, int16_t y)	""
FT_GEStatus Vertex2ii(uint16_t x, uint16_t y, uint8_t Handle = 0, uint8_t Cell = 0)	""
	""
/* graphics helper APIs */	
FT_GEStatus ColorRGB(uint32_t rgb);	""
FT_GEStatus ColorARGB(uint32_t argb);	""
/* APIs related to coprocessor commands, widgets etc */	
FT_GEStatus Cmd_Logo(void)	""
FT_GEStatus Cmd_Append(uint32_t Ptr, uint32_t Num)	""
FT_GEStatus Cmd_BGColor(uint32_t c)	""
FT_GEStatus Cmd_Button(int16_t x, int16_t y, uint16_t w, uint16_t h, uint8_t Font, uint16_t Options, const char *s)	""
FT_GEStatus Cmd_Calibrate(uint32_t Result)	""
FT_GEStatus Cmd_Clock(int16_t x, int16_t y, int16_t r, uint16_t Options, uint16_t h, uint16_t m, uint16_t s, uint16_t ms)	""
FT_GEStatus Cmd_ColdStart(void)	""
FT_GEStatus Cmd_Dial(int16_t x, int16_t y, int16_t r, uint16_t Options, uint16_t Val)	""
FT_GEStatus Cmd_DLStart(void)	""
FT_GEStatus Cmd_FGColor(uint32_t c)	""
FT_GEStatus Cmd_Gauge(int16_t x, int16_t y, int16_t r, uint16_t Options, uint16_t Major, uint16_t Minor, uint16_t Val, uint16_t Range)	""
FT_GEStatus Cmd_GetMatrix(void)	""
FT_GEStatus Cmd_GetProps(uint32_t &Ptr, uint32_t &w, uint32_t &h)	""
FT_GEStatus Cmd_GetPtr(uint32_t Result)	""



Prototype	Description
FT_GEStatus Cmd_GradColor(uint32_t c)	""
FT_GEStatus Cmd_Gradient(int16_t x0, int16_t y0, uint32_t rgb0, int16_t x1, int16_t y1, uint32_t rgb1)	""
FT_GEStatus Cmd_Inflate(uint32_t Ptr)	""
FT_GEStatus Cmd_Interrupt(uint32_t ms)	""
FT_GEStatus Cmd_Keys(int16_t x, int16_t y, int16_t w, int16_t h, uint8_t Font, uint16_t Options, const char *s)	""
FT_GEStatus Cmd_LoadIdentity(void)	""
FT_GEStatus Cmd_LoadImage(uint32_t Ptr, int32_t Options)	""
FT_GEStatus Cmd_Memcpy(uint32_t Dest, uint32_t Src, uint32_t Num)	""
FT_GEStatus Cmd_Memset(uint32_t Ptr, uint8_t Value, uint32_t Num)	""
FT_GEStatus Cmd_Memcrc(uint32_t Ptr, uint32_t Num, uint32_t &Result)	""
FT_GEStatus Cmd_Memwrite(uint32_t Ptr, uint32_t Num)	""
FT_GEStatus Cmd_Memzero(uint32_t Ptr, uint32_t Num)	""
FT_GEStatus Cmd_Number(int16_t x, int16_t y, uint8_t Font, uint16_t Options, uint32_t n)	""
FT_GEStatus Cmd_Progress(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t Options, uint16_t Val, uint16_t Range)	""
FT_GEStatus Cmd_RegRead(uint32_t Ptr, uint32_t Result)	""
FT_GEStatus Cmd_Rotate(int32_t a)	""
FT_GEStatus Cmd_Scale(int32_t sx, int32_t sy)	""
FT_GEStatus Cmd_ScreenSaver(void)	""
FT_GEStatus Cmd_Scrollbar(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t Options, uint16_t Val, uint16_t Size, uint16_t Range)	""
FT_GEStatus Cmd_SetFont(uint8_t Font,	""

Prototype	Description
uint32_t Ptr)	
FT_GEStatus Cmd_SetMatrix(void)	""
FT_GEStatus Cmd_Sketch(int16_t x, int16_t y, uint16_t w, uint16_t h, uint32_t Ptr, uint16_t Format)	""
FT_GEStatus Cmd_Slider(int16_t x, int16_t y, uint16_t w, uint16_t h, uint16_t Options, uint16_t val, uint16_t Range)	""
FT_GEStatus Cmd_Snapshot(uint32_t OutputAddr)	""
FT_GEStatus Cmd_Spinner(int16_t x, int16_t y, uint8_t Style, uint8_t Scale)	""
FT_GEStatus Cmd_Stop(void)	""
FT_GEStatus Cmd_Swap(void)	""
FT_GEStatus Cmd_Text(int16_t x, int16_t y, uint8_t font, uint16_t Options, const char *s)	""
FT_GEStatus Cmd_Toggle(int16_t x, int16_t y, int16_t w, uint8_t font, uint16_t Options, uint16_t State, const char *s)	""
FT_GEStatus Cmd_Track(int16_t x, int16_t y, uint16_t w, uint16_t h, uint8_t Tag)	""
FT_GEStatus Cmd_Translate(int32_t tx, int32_t ty)	""
/* Helper APIs related to audio engine */	
FT_AEStatus PlaySound(uint8_t Volume, uint16_t SoundNote)	Play specific sound note and the volume specified.
FT_AEStatus PlaySound(uint16_t SoundNote)	Stop output sound and volume will not be modified.
void SetSoundVolume(uint8_t Volume)	Set sound volume.
uint8_t GetSoundVolume(void)	Get the current sound volume.
FT_AEStatus PlayAudio(uint8_t Volume, uint8_t Format, uint16_t SamplingFreq, uint32_t BufferAddr, uint32_t BufferSize, uint8_t Loop)	This API play RAM audio for one time or continuously. Sampling frequency is from 8k to 48k
void SetAudioVolume(uint8_t Volume)	Set audio volume.

Prototype	Description
FT_AEStatus GetAudioStats(uint32_t &CurrPlayAddr)	This API updates the playback audio pointer.
uint8_t GetAudioVolume(void)	Get the audio volume.
void StopAudio(void)	Stop the audio stream and volume will not be modified
<i>/* APIs related to touch engine */</i>	
void SetTouchMode(uint8_t TMode)	One of 0ff/oneshot/frame/continuous. Default being continuous
void SetHostTagXY(uint16_t xoffset,uint16_t yoffset)	API to set coordinates for host specific tag query
uint8_t GetHostTagXY(void)	API to get TAG from FT_GC for coordinates set by SetHostTagXY() API - host needs to wait for at least 1 frame to get these query values
void GetTagXY(sTagXY &sTagxy)	Get the touched object tag and respective coordinates
void GetTrackTag(sTrackTag &sTracktag)	Get the tracking and tag value
<i>/* APIS related to power modes */</i>	
void HostCommand(uint32_t HostCommand)	Send a 32bit host command.
<i>/* Special APIs - for ease of usage in FT_GC */</i>	
void DLStart(void)	Inserts cmd_dlstart() followed by clear(1,1,1) graphics command.
void DLEnd(void)	Inserts display() followed cmd_swap() command.
FT_GEStatus CheckLogo(void)	Special API to check logo completion.
<i>/*APIs to render all the commands to hardware*/</i>	
FT_GEStatus Flush(void)	API to flush out all the commands to FT_GC, does not wait for the completion of the rendering
FT_GEStatus Finish(void)	API to flush out all the commands to FT_GC

Prototype	Description
	and waits for the completion of execution.
FT_GEStatus CheckFinish(void)	Checks FIFO and return if the graphics controller has consumed all the commands.
uint32_t GetError(void)	Check to see if there are any errors from the graphics controller library.
/* API related to coprocessor fifo buffer management - all the below APIs are transfer commands */	
FT_GEStatus WriteCmd(uint32_t Cmd)	Send a single CMD command
FT_GEStatus WriteCmd(uint8_t *Src,uint32_t NBytes)	API to send N command bytes
FT_GEStatus WriteCmdfromflash(prog_uchar *Src,uint32_t NBytes)	Write commands in flash memory to the coprocessor of specified source and size.
FT_GEStatus TransferCmd(uint32_t Cmd)	Transfer a command in continuous mode.
FT_GEStatus TransferCmd(uint8_t *Src,uint32_t NBytes)	Transfer a set of commands in continuous mode.
FT_GEStatus TransferCmdfromflash(prog_uchar *Src,uint32_t NBytes)	Transfer a set of commands located in flash.
void EndTransferCmd()	End transfer in continuous mode.
FT_GEStatus Cmd_GetResult(uint32_t &Result)	Get the execution result of the previous commands such as cmd_memcrc, cmd_calibration, cmd_regread which has return values. Busy status means command is still in progress.
FT_GEStatus Cmd_GetResults(int8_t *pA,uint16_t NBytes)	reads N bytes of result bytes from current write pointer
FT_GEStatus UpdateFreeSpace()	Update the coprocessor free space.
FT_GEStatus ChkGetFreeSpace(uint16_t NBytes)	Check the coprocessor to see if it has specified amount of free space.
FT_GEStatus StartTransferCmd()	Start transfer in continuous mode. Use macro commands.

### 3.4 FT\_RTC class

This library contains functionalities to access the RTC on the FTDI Plus boards (VM8xxP).

#### Available Objects

FT\_RTCStatus - RTC status enums.

DateTime - This object holds the time parameters. Passing an instance of this object to the *GetDateTime* function will update all the fields to the latest time. Similarly, passing an instance of this object to the *SetDateTime* function will set the time in the RTC with parameters in the object.

## APIs

PROTOTYPE	DESCRIPTION
FT_RTC(void);	Constructor to use the default I <sup>2</sup> C address ID.
FT_RTC(uint8_t i2caddr);	Constructor to use specified address of I <sup>2</sup> C RTC device
~FT_RTC(void);	Destructor
void GetVersion(uint8_t &Major,uint8_t &Minor,uint8_t &Build);	Get the RTC version number.
FT_RTCStatus Init(void);	Open I <sup>2</sup> C driver
FT_RTCStatus IsRunning();	Check whether the RTC is currently running
FT_RTCStatus SetFormat(uint8_t HFormat);	Set the date and time formats wrt RTC IC
FT_RTCStatus GetDateTime(DateTime &DT);	Get the date & time from RTC
FT_RTCStatus SetDateTime(const DateTime &DT);	Set the initial date & time and starts the RTC
FT_RTCStatus GetTime(uint8_t &HFormat,uint8_t &Hour,uint8_t &Minutes,uint8_t &Seconds);	get the time from RTC
/*read, write memory*/	
FT_RTCStatus Read(uint32_t Addr,uint8_t &Value);	read one byte from Address
FT_RTCStatus Read(uint32_t Addr, uint8_t *Dst,uint8_t NBytes);	read n bytes
FT_RTCStatus Write(uint32_t Addr,uint8_t Value);	write 1 byte of data
FT_RTCStatus Write(uint32_t Addr,uint8_t *Src,uint8_t NBytes);	write n bytes of data - 1 to 255 bytes
FT_RTCStatus Exit(void);	close I <sup>2</sup> C driver

PROTOTYPE	DESCRIPTION
<i>/*utility APIs*/</i>	
uint8_t DecToBcd(uint8_t Value);	decimal to binary-coded decimal
uint8_t BcdToDec(uint8_t Value);	binary-coded decimal to decimal

### 3.5 FT\_SD class

The FT\_SDFile class is for file operations and only 1 file can be opened at a time. In order to access the SD card files, an instance of FT\_SDFile class is needed in conjunction with the FT\_SD library. FT\_SD and FT\_SDFile usage are demonstrated in the "Library Usage" section of this document.

#### Available Objects

FT\_SDStatus - FT\_SD library status enums.

FT\_DirEnt - Directory entry structure.

#### APIs

##### 3.5.1 FT\_SDFile class

Prototype	Description
FT_SDFile()	Constructor, currently does nothing
~FT_SDFile(){};	Destructor, currently does nothing
<i>/* File system related APIs */</i>	
void FileStart(FT_DirEnt &de);	Search for the file name
FT_SDStatus Init(FT_DirEnt &de);	Initialize SDFile object
void ReadSector(uint8_t *dst);	Read 512 bytes of data into destination buffer.
void ReadSector();	Skip 512 bytes from the current read pointer.
void SeekSector(uint32_t Off);	Seek file to particular offset, offset should be multiple of 512 bytes
void SkipSector();	Skip a sector

<code>void SkipCluster();</code>	Skip the current cluster
<code>void NextCluster();</code>	Goto the next cluster

### 3.5.2 FT\_SD

Prototype	Description
/* sdcard implementation related APIs - internal APIs wrt sd spec */	
FT_SD(void);	Constructor to use default CS pin
FT_SD(uint8_t Pin);	Constructor to use specified CS pin
~FT_SD(void);	Close the SPI channel
FT_SDStatus Init();	Initialize SD card usage
/* Read APIs */	
uint8_t Read(uint32_t Off);	Read 8 bits starting at the Off address
uint16_t Read16(uint32_t Off);	Read 16 bits starting at the Off address
uint32_t Read32(uint32_t Off);	Read 32 bits starting at the Off address
void ReadN(uint8_t *dst, uint32_t Off, uint16_t NBytes);	Read N bytes to buffer
/* SD spec related APIs */	
void Sel(void);	SPI CS enable
void DeSel(void);	SPI CS disable
void SDDelay(uint8_t N);	delay of N*8 clocks
void Cmd(uint8_t Cmd, uint32_t Lba, uint8_t Crc);	SD card specific command
FT_SDStatus R1(uint8_t &r);	Response 1 from sdcard controller
FT_SDStatus SDR7(uint8_t &r);	Response 7 from sdcard controller
FT_SDStatus SDR3(uint32_t &OCR, uint8_t &r);	Response 3 from sdcard controller
void Cmd17(uint32_t Off);	Command 17 to sdcard controller
void AppCmd(uint8_t CC, uint32_t LBA);	SD spec appcmd
/* File operations */	



<b>Prototype</b>	<b>Description</b>
FT_SDStatus OpenFile(FT_SDFile &SdFile,const char *FileName);	Attempt to open a file and update SDFile handle

## 4 Supported Platforms

The current supported platforms are:

- **VM800P35<sup>1</sup>** - 3.5" FTDI Plus board with on-board MCP7940M RTC and SD card Reader.
- **VM800P43 and VM800P50<sup>1</sup>** - 4.3" and 5.0" FTDI Plus boards with on-board MCP7940M RTC and SD card Reader.
- **VM801P43<sup>2</sup> and VM801P50<sup>2</sup>** - 4.3" and 5.0" FTDI Plus boards with on-board MCP7940M RTC, SD card Reader and capacitive touch controller.
- **ADAM\_4DLCD\_FT843<sup>1</sup>** - Can be used on most Arduino boards such as UNO, Arduino Pro, RedBoard, Leonardo, Due, Mega 2560.
- **FT\_Breakout\_4DLCD\_FT843<sup>1</sup>** - Can be used on most Arduino boards such as UNO, Arduino Pro, RedBoard, Leonardo, Due, Mega 2560. The current library does not support SD card for this platform.
- **VM800B43<sup>1</sup> and VM800B50<sup>1</sup>** - 4.3" and 5.0" FTDI Basic boards pairing with most Arduino boards such as UNO, Arduino Pro, RedBoard, Leonardo, Due, Mega 2560. The current library does not support SD card for this platform.
- **VM800C43<sup>1</sup> and VM800C50<sup>1</sup>** - 4.3" and 5.0" FTDI Credit card boards pairing with most Arduino boards such as UNO, Arduino Pro, RedBoard, Leonardo, Due, Mega 2560. The current library does not support SD card for this platform.
- **VM800B35<sup>1</sup>** - 3.5" FTDI Basic board pairing with most Arduino boards such as UNO, Arduino Pro, RedBoard, Leonardo, Due, Mega 2560. The current library does not support SD card for this platform.

NOTE:

<sup>1</sup> - Platform that uses FT800 graphics controller.

<sup>2</sup> - Platform that uses FT801 graphics controller.

The following list of boards was used for the development and verification of this library:

- FTDI Plus boards in all resolutions
- FTDI Basic boards in all resolutions
- Arduino Uno
- Arduino Yun
- Arduino Pro ATmega328 (3.3 V, 8MHz)
- Arduino Pro ATmega328 (5 V, 16 MHz)
- Arduino Mega 2560
- Arduino Due
- Arduino Nano
- Arduino Leonardo
- Redboard ([sparkfun](#))
- ADAM\_4DLCD\_FT843 ([4D Systems](#))

NOTE: The ADAM\_4DLCD\_FT843 has the option to run on 3.3V or 5V, so choosing the correct voltage is imperative. For example, the Arduino Pro ATmega328 (3.3V, 8MHz) can only supply 3.3V to the IO shield so the jumper on the ADAM\_4DLCD\_FT843 must be set to 3.3V in order to prevent damage to the Arduino. Please refer to the development board's specification sheet for more information regarding the issue.

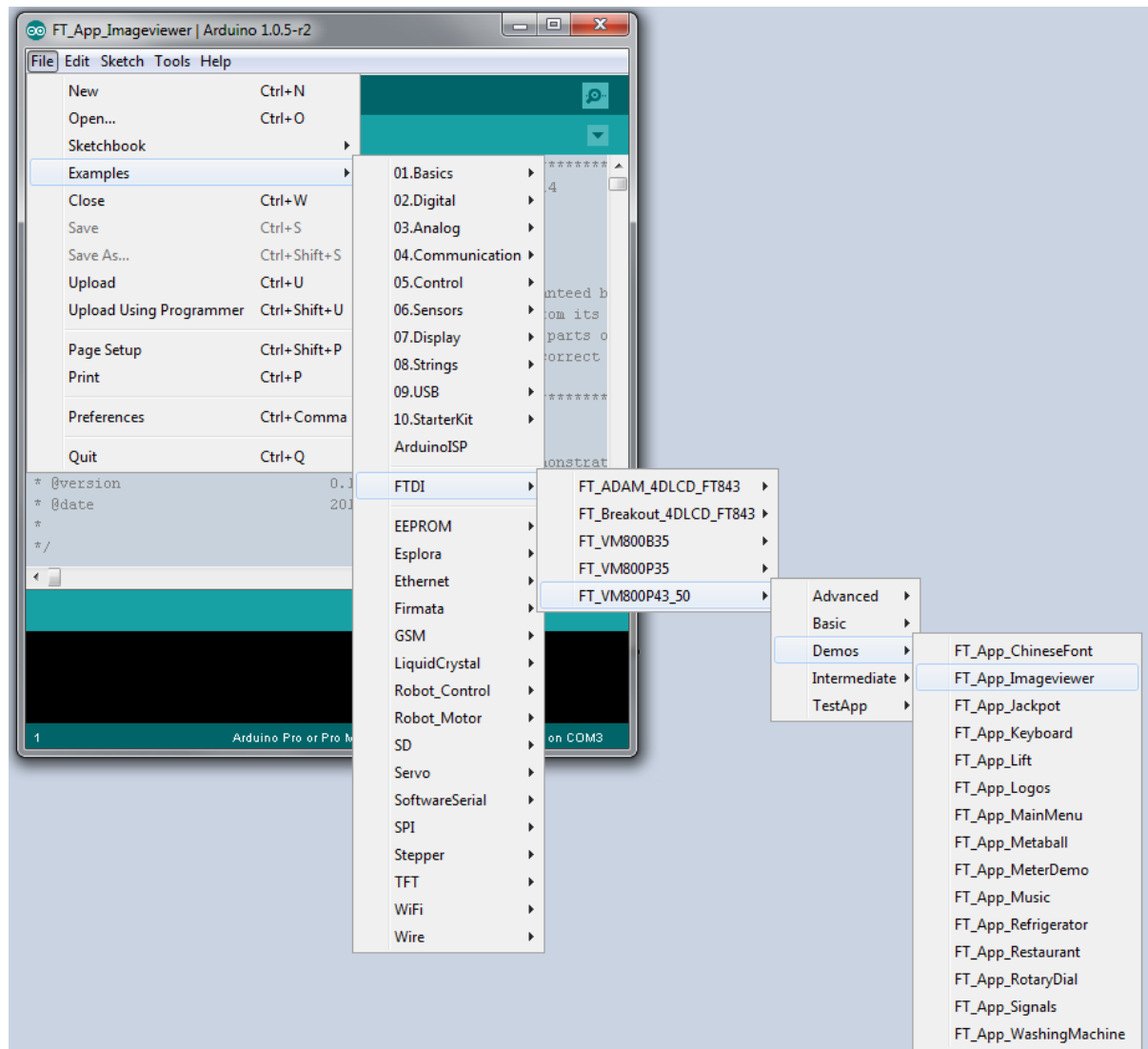
NOTE: Example sketches cannot run on all Arduino boards due to hardware and/or software constraints. For the hardware side, since many Arduino boards do not use the same microcontroller or microcontroller feature set, the actual available SRAM and storage for the sketch may vary. For example, the Arduino Yun, Uno, and Pro with the ATmega328 contains 32 KB of Flash Memory but the boot loader occupies a different amount of space for each of the three

boards: Yun - 4 KB, Pro - 2 KB, Uno - 0.5 KB. Moreover, the Pro and Uno have 2 KB of SRAM while the Yun has 2.5 KB. For the software side, some platforms offer more functionalities than others so their compiled binary size will vary even though the same hardware board is targeted due to the functionality and compilations variations of the Arduino IDE, which is likely to result in slight increase or decrease in binary size. During the development of this Arduino library, the compiled binary size for almost all of the example sketches using versions 1.0.4 and 1.0.5-r2 are the same while version 1.5.6-r2 consistently reported a slightly bigger or smaller size. The best way to determine if a board supports the platform, is to compile and run it.

## 5 Library Usage

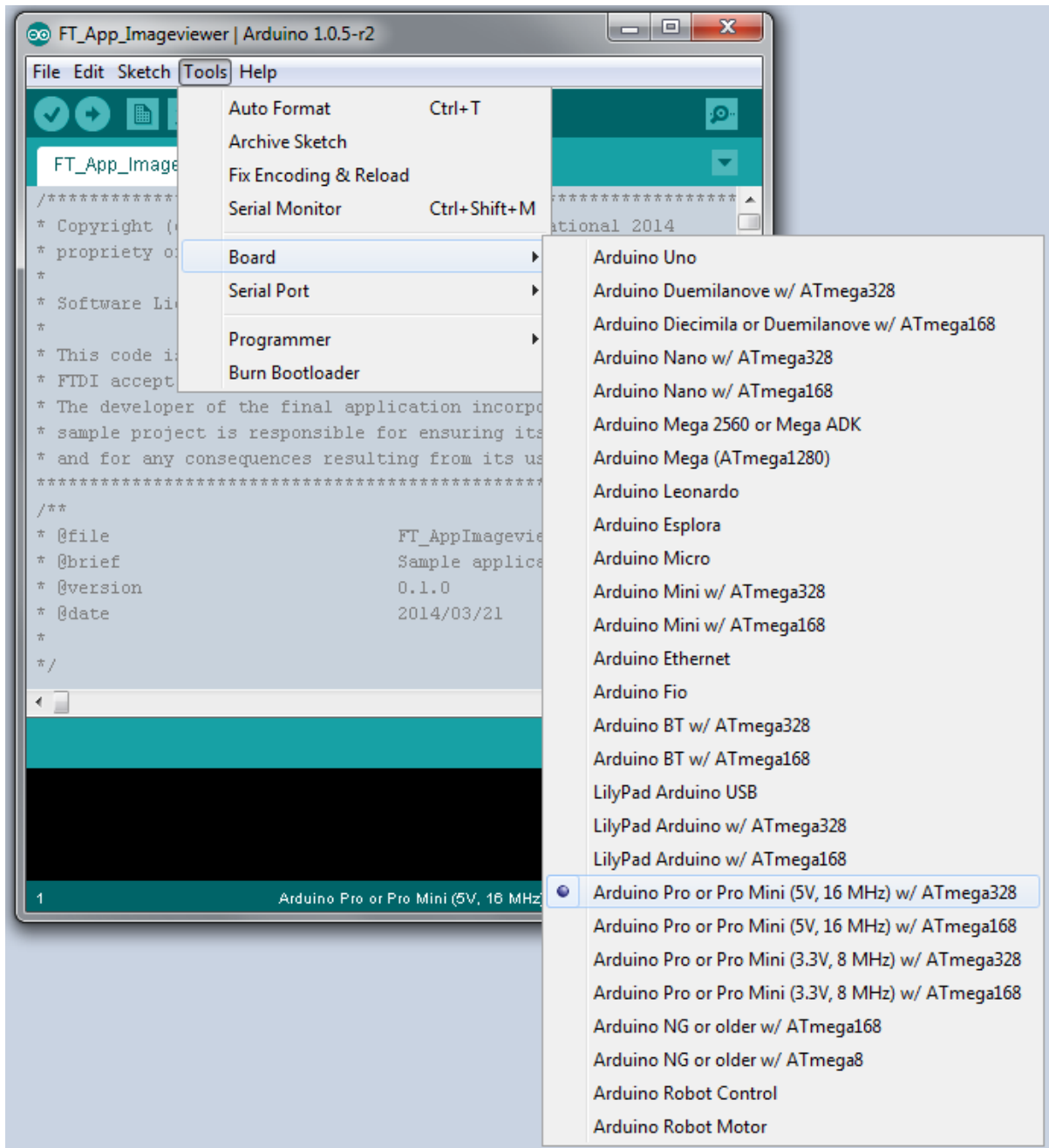
The section demonstrates the usage of the Arduino library. The *FT\_App\_Imageviewer* demo application on the FT\_VM800P43\_50 platform was chosen for this demonstration; this demo application is also available for the FT\_ADAM\_4DLCD\_FT843 and FT\_VM800P35 platforms.

A successfully installed library will contain "FTDI" as one of the items in the *Examples* list menu and hovering over it will reveal all the available sketches as shown below.

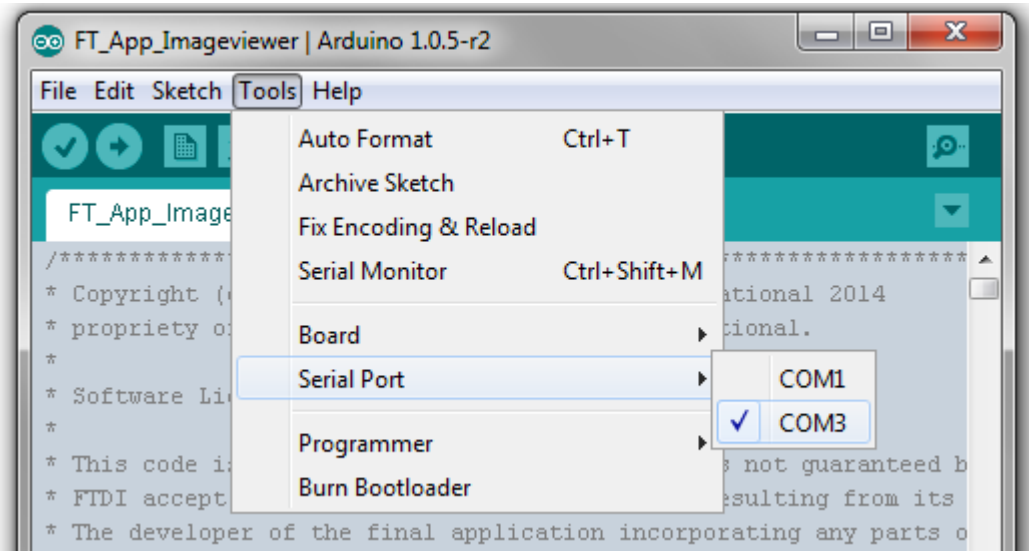


**Figure 5-1 : Arduino IDE - example sketches**

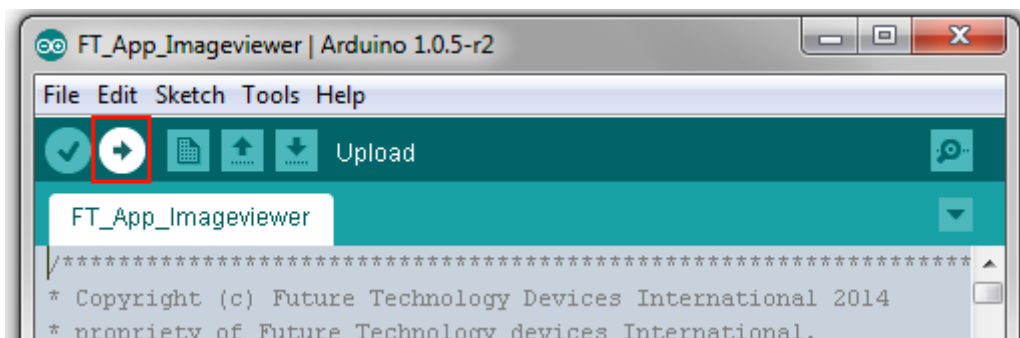
Clicking any one of the example sketches will open it up with a new sketch window. It can be examined and/or upload it to the Arduino board by selecting the correct board and Serial Port first then choosing Upload. For this example, the VM800P43 must use the "Arduino Pro or Pro Mini (5V, 16MHz) w/ Atmega328" board selection.



**Figure 5-2 : Arduino IDE - Board selection**



**Figure 5-3 : Arduino IDE - Serial Port selection**



**Figure 5-4 : Arduino IDE - Upload button**

## 5.1 Platform Setup

The following setup is almost the same for all platforms. FT800Impl shall be used for the FT800 chip and FT801Impl shall be used for the FT801 chip.

### 5.1.1 Library Includes

```

/* The SPI.h is a must for all platform */
#include "SPI.h"

/* Wire.h is required if RTC will be used. Both the Wire.h and the RTC.h in the platform's
header file can be removed to regain some flash memory space if the sketch does not use
RTC.*/
#include "Wire.h"

/* include platform header */
#include "FT_VM800P43_50.h"

```

### 5.1.2 Global Object Declaration

```

/* SD card object*/
FT_SD FtSd(FT_SD_CSPIN);

/* sd_present holds error values during initialization. 0 means no error and all other errors are
non-zero value */
FT_SDStatus sd_present;

/*SD file object for file accessing*/
FT_SDFile Imagefile;

/* Global object for transport */
FT800IMPL_SPI FTImpl(FT_CS_PIN,FT_PDN_PIN,FT_INT_PIN);

```

The FT\_SDStatus and FT\_SDFile objects can be non-global but the FT\_SD and FT800Impl/FT801Impl objects have to be global.

NOTE: When using the ADAM\_4DLCD\_FT843 platform, the SD card object has to be declared before the FT800Impl object in order to function correctly. Also, if the sketch uses the SD card then it has to be in the SD card slot before running the sketch and it can't be in the slot if the sketch doesn't use it. This is a known issue with the ADAM\_4DLCD\_FT843, as it will interfere with the SPI.

### 5.1.3 Start-up Configuration

```

int16_t BootupConfigure(){
    uint32_t chipid = 0;
    FTImpl.Init(FT_DISPLAY_RESOLUTION);//configure the display to the WQVGA

    delay(20);//for safer side
    chipid = FTImpl.Read32(FT_ROM_CHIPID);

    /* Identify the chip */
    if(FT800_CHIPID != chipid)
    {
        Serial.print("Error in chip id read ");
        Serial.println(chipid,HEX);
        return 1;
    }

    /*Platform pressure sensitivity adjustment*/
    FTImpl.Write16(REG_TOUCH_RZTHRESH,1200);

    /* Set the Display & audio pins */
    FTImpl.SetDisplayEnablePin(FT_DISPENABLE_PIN);
    FTImpl.SetAudioEnablePin(FT_AUDIOENABLE_PIN);
    FTImpl.DisplayOn();
    FTImpl.AudioOn();
    return 0;
}

```

The above boot up configuration function is the same for all platforms that use the FT800 graphics controller.

### 5.1.4 Start-up Configuration for FT801 chip

```
int16_t BootupConfigure(){
    uint32_t chipid = 0;
    FTImpl.Init(FT_DISPLAY_RESOLUTION);//configure the display to the WQVGA

    delay(20);//for safer side
    chipid = FTImpl.Read32(FT_ROM_CHIPID);

    /* Identify the chip */
    if(FT801_CHIPID != chipid)
    {
        Serial.print("Error in chip id read ");
        Serial.println(chipid,HEX);
        return 1;
    }

    /* Set the Display & audio pins */
    FTImpl.SetDisplayEnablePin(FT_DISPENABLE_PIN);
    FTImpl.SetAudioEnablePin(FT_AUDIOENABLE_PIN);
    FTImpl.DisplayOn();
    FTImpl.AudioOn();
    return 0;
}
```

The above boot up configuration function is the same for all platforms that use the FT801 graphics controller.

```
/* Display an on-screen message if the storage device is not found */
void CheckStorageDevicePreceance(){
    while(sd_present){
        FTImpl.DLStart();
        FTImpl.Clear(1,1,1);
        FTImpl.ColorRGB(255,255,255);

        FTImpl.Cmd_Text(FT_DISPLAYWIDTH>>1,FT_DISPLAYHEIGHT>>1,29,FT_OPT_
CENTER,"STORAGE DEVICE NOT FOUND");
        FTImpl.DLEnd();
        FTImpl.Finish();
        delay(5000);
    }
}
```

This function is an optional boot up function to check whether the SD card is in the card slot and display an on-screen message to alert the user.



```
void setup()
{
    .
    .
    /*Initialize the SD object, initialization is required before usage. Screen error message
    can only be displayed when the FT800 is ready*/
    sd_present = FtSd.Init();

    /* Set the Display Enable pin*/
    if(BootupConfigure())
    {
        //error case - do not do any thing
    }
    else
    {
        CheckStorageDevicePrencence();
        .
        .
        /* sketch functionalies */
        .
    }
}

void loop()
{
}
}
```

Arduino's setup and loop function.

## 5.2 FT\_App\_Imageviewer Setup

Before entering into the application, one image is loaded to GRAM with handle "r"

```
/* In the function*/
Loadimage2ram(r); // r is the bitmap handle
```

Set the Bitmap properties For the Image

```
/* In the Function*/
FTImpl.BitmapHandle(r);
FTImpl.BitmapSource(r ? 131072L : 100);
FTImpl.BitmapLayout(FT_RGB565,320L*2,194);
FTImpl.BitmapSize(FT_NEAREST,FT_BORDER,FT_BORDER,320,194));
```

Note: All images in this application have a fixed Size of 320x194

## 5.3 FT\_App\_Imageviewer SD Card Usage

```

/* Apis used to upload the image to GRAM from SD card*/
void Load_Jpeg(FT_SDFile &r)
{
  uint8_t imbuff[512];
  uint16_t blocklen;
  while (r.Offset < r.Size) /* Offset - current read offset, Size - file size */
  {
    uint16_t n = min(512, r.Size - r.Offset);
    n = (n + 3) & ~3; // force 32-bit alignment
    r.ReadSector(imbuff); /* Read and fill buffer */
    FTImpl.WriteCmd(imbuff,n); /* copy data continuously into command memory */
  }
}

void Loadimage2ram(uint8_t bmphandle)
{
  static byte image_index = 0,max_files=6;
  if(sd_present)
  {
    max_files = 2;
  }
  if(FtSd.OpenFile(Imagefile,imagenamename[image_index])){
    FTImpl.WriteCmd(CMD_MEMSET);
    FTImpl.WriteCmd((bmphandle ? 131072L : 100));
    FTImpl.WriteCmd((bmphandle ? 150 : 100));
    FTImpl.WriteCmd(320L*2*194);
  }else
  {
    FTImpl.WriteCmd(CMD_LOADIMAGE);
    FTImpl.WriteCmd((bmphandle ? 131072L : 100));
    FTImpl.WriteCmd(0);
    Load_Jpeg(Imagefile);
  }
  image_index++;
  if(image_index > (max_files-1)) image_index = 0;
  return;
}

```

/\* Open a file and associate it with the imagefile FT\_SDFfile object. The FtSd SD object and Imagefile SD file object were declared in the previous section. The OpenFile function returns a zero if the operation is successful and a non-zero value means error.\*/

## 5.4 FT\_App\_Imageviewer Functionality

The FT\_App\_Imageviewer application demonstrates the usage of the FT800 series in-built jpeg decode function. The application constantly monitors the user's input and changes the image accordingly. The application maintains two bitmaps in the RAM (ping pong style of implementation), one for the currently displayed image and another for the next image to be displayed. The application also demonstrates a reflection effect where the same image is flipped and displayed at the bottom with a transparent effect.



Figure 5-5 : FT\_App\_Imageviewer - screen shot 1

## 5.5 FT\_App\_Imageviewer Bitmap Handling

Draw a clear Button by using the Co-processor Command CMD\_BUTTON with TAG 'C' and set the Foreground and Background Colour of the button. The Tag can be read by using the REG\_TOUCH\_TAG register. The register REG\_TOUCH\_TAG returns Zero when touch is not detected. The clear button resets the GRAM by using CMD\_MEMZERO.

```

/* In the function*/
if(px == temp_x && loaded==0)
{
  FTImpl.BitmapHandle(r^1);
  FTImpl.BitmapSource((r^1) ? 131072L : 00);
  FTImpl.BitmapLayout(FT_RGB565,320L*2,194);
  FTImpl.BitmapSize(FT_NEAREST,FT_BORDER,FT_BORDER,320,194);
  Loadimage2ram(r ^ 1);
  loaded = 1;
}

```

## 5.6 FT\_App\_Imageviewer Image Reflection

The image is reflected on the floor, by flipping the image using TRANSLATE and SCALE commands. The Blended effect is created with the blend function.

Reflection

```

/*In the Function*/
FTImpl.Cmd_LoadIdentity();
FTImpl.Cmd_Translate((temp_x)*65536L, 65536L*96.5);
FTImpl.Cmd_Scale(1*65536, 65536*-1);
FTImpl.Cmd_Translate(-(temp_x)*65536L, 65536L*-96.5);
FTImpl.Cmd_SetMatrix();

```

## Blend Function

```
FTImpl.Vertex2ii(x, (10+aspect_ratio),r, 0);  
FTImpl.SaveContext();  
FTImpl.ColorMask(0,0,0,1);  
FTImpl.BlendFunc(FT_ONE,FT_ZERO);  
FTImpl.Vertex2ii(0, 212, 2, 0);  
FTImpl.ColorMask(1,1,1,1);  
FTImpl.BlendFunc(FT_DST_ALPHA, FT_ONE_MINUS_DST_ALPHA);
```

## 6 Hardware Setup

This section shows how to setup each of the supported platforms. For the platforms that require users to connect the SPI connection pins from the system host MCU, please refer to the actual CS, PDN, INT pin number in the respective platform's header file.

More information can be found on the specification sheet of the Arduino board regarding its on-board connectors.

### 6.1 VM800P35

All FTDI Plus modules come with a display, an in-built MCU, the ATmega328P (Arduino PRO 5V, 16MHz), an SD card connector, RTC with battery backup, and the ability to expand the IO capability through a daughter board. Simply plug a mini USB to the programming port at the back and the board is ready to program and run sketches.

### 6.2 VM800P43\_50

Refer to the VM800P35 platform.

### 6.3 VM801P43\_50

Refer to the VM800P35 platform for information other than capacitive touch controller.

### 6.4 ADAM\_4DLCD\_FT843

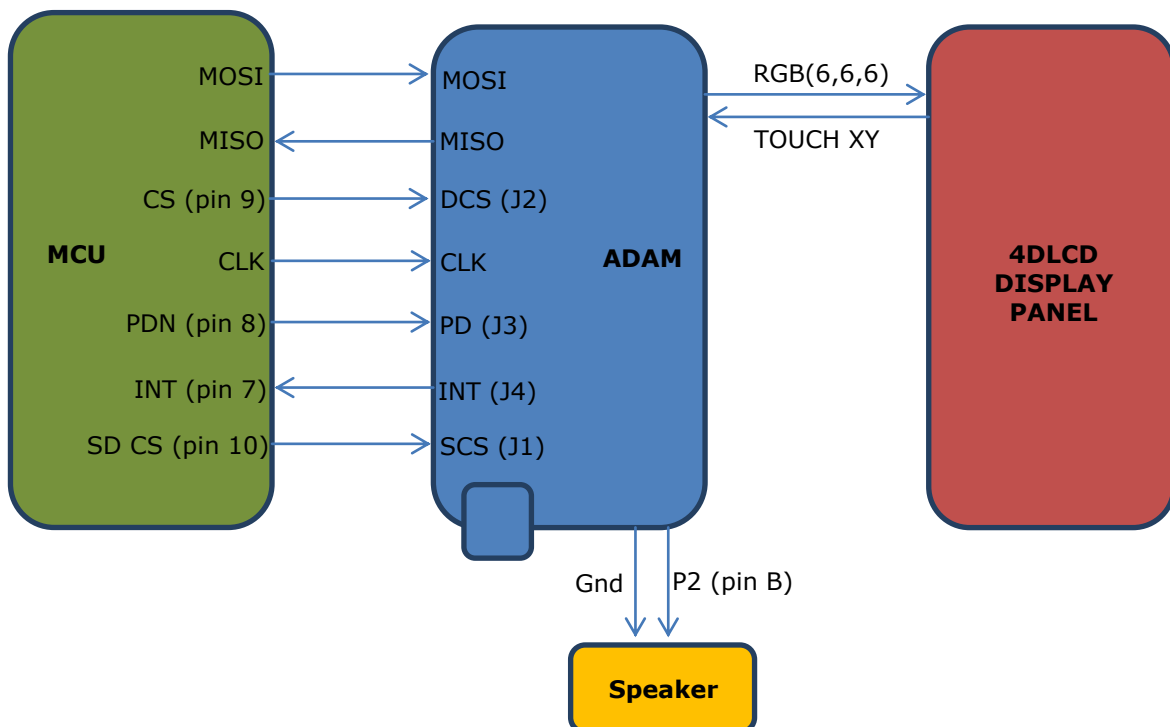
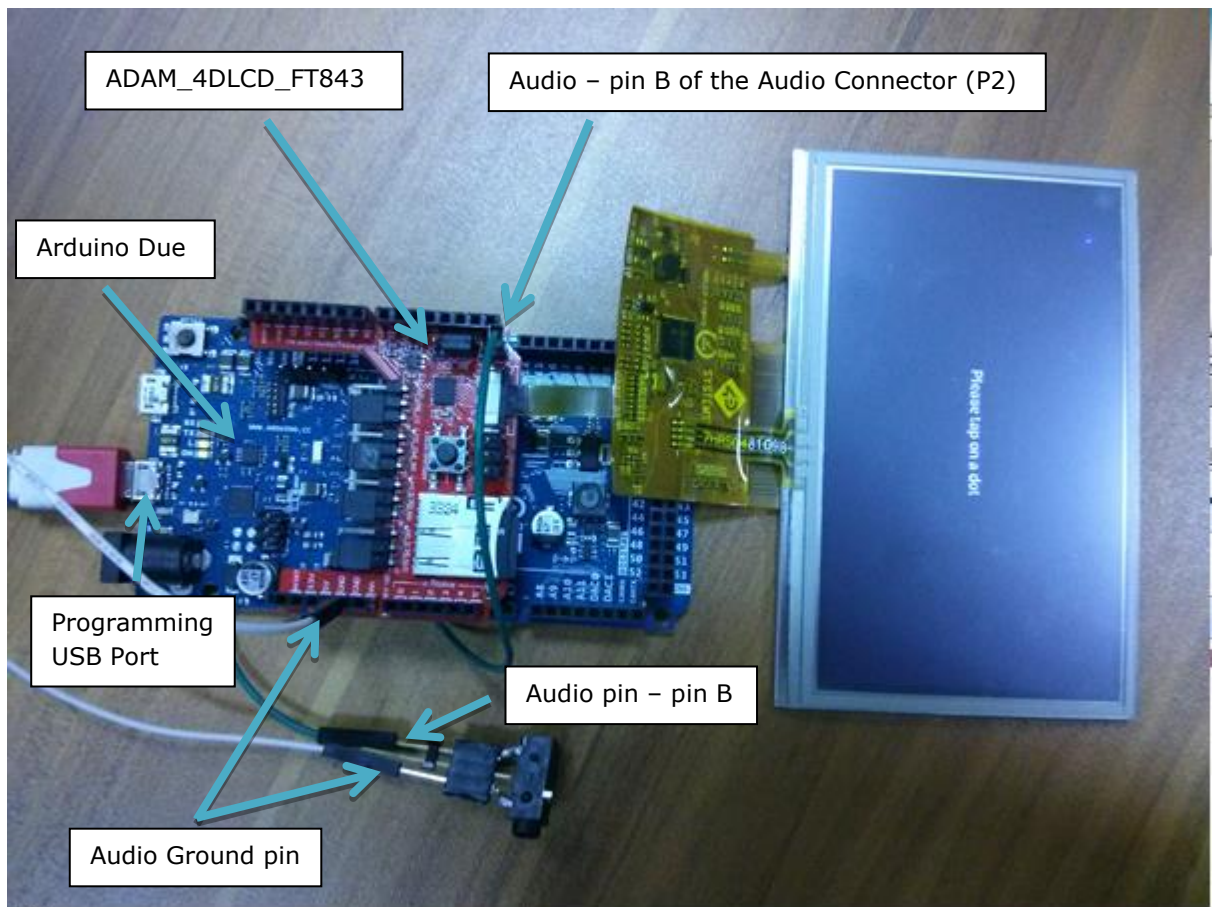


Figure 6-1 : ADAM\_4DLCD\_FT843 platform block diagram



**Figure 6-2 : Hardware setup - ADAM\_4DLCD\_FT843 platform with Arduino Due**

The whole setup consists of aligning the ICSP port at the bottom of the ADAM\_4DLCD\_FT843 module with the ICSP port on the MCU board and connect the two wires for the speaker.

## 6.5 FT\_Breakout\_4DLCD\_FT843

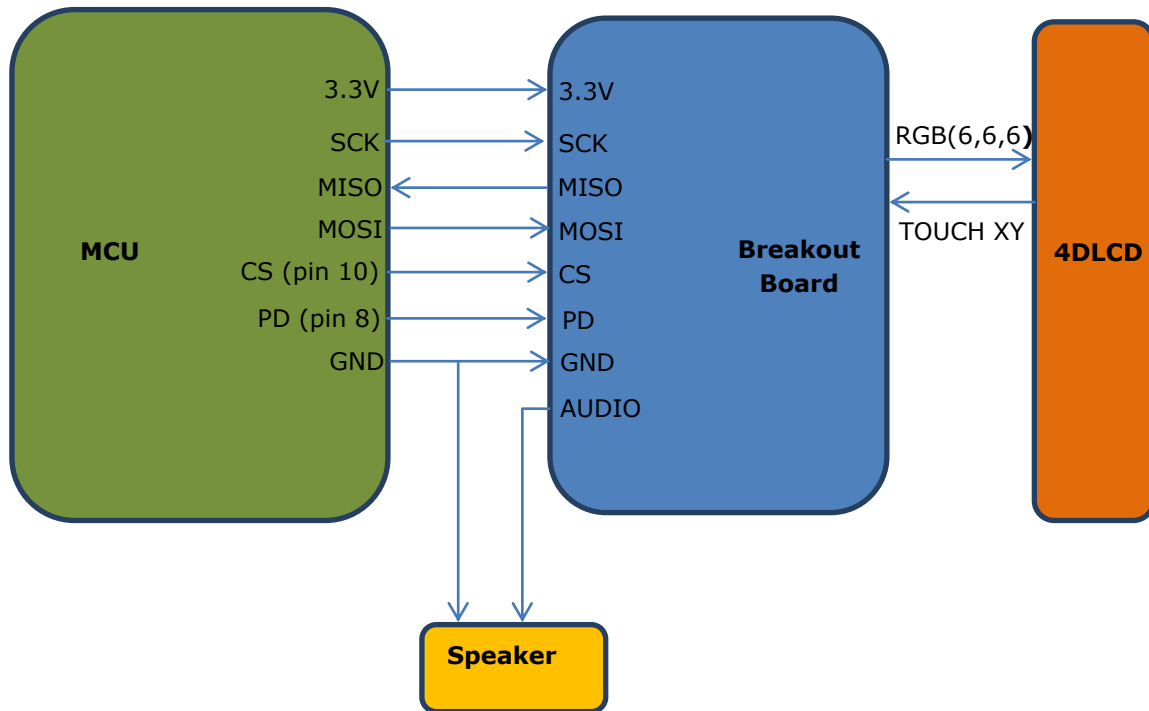
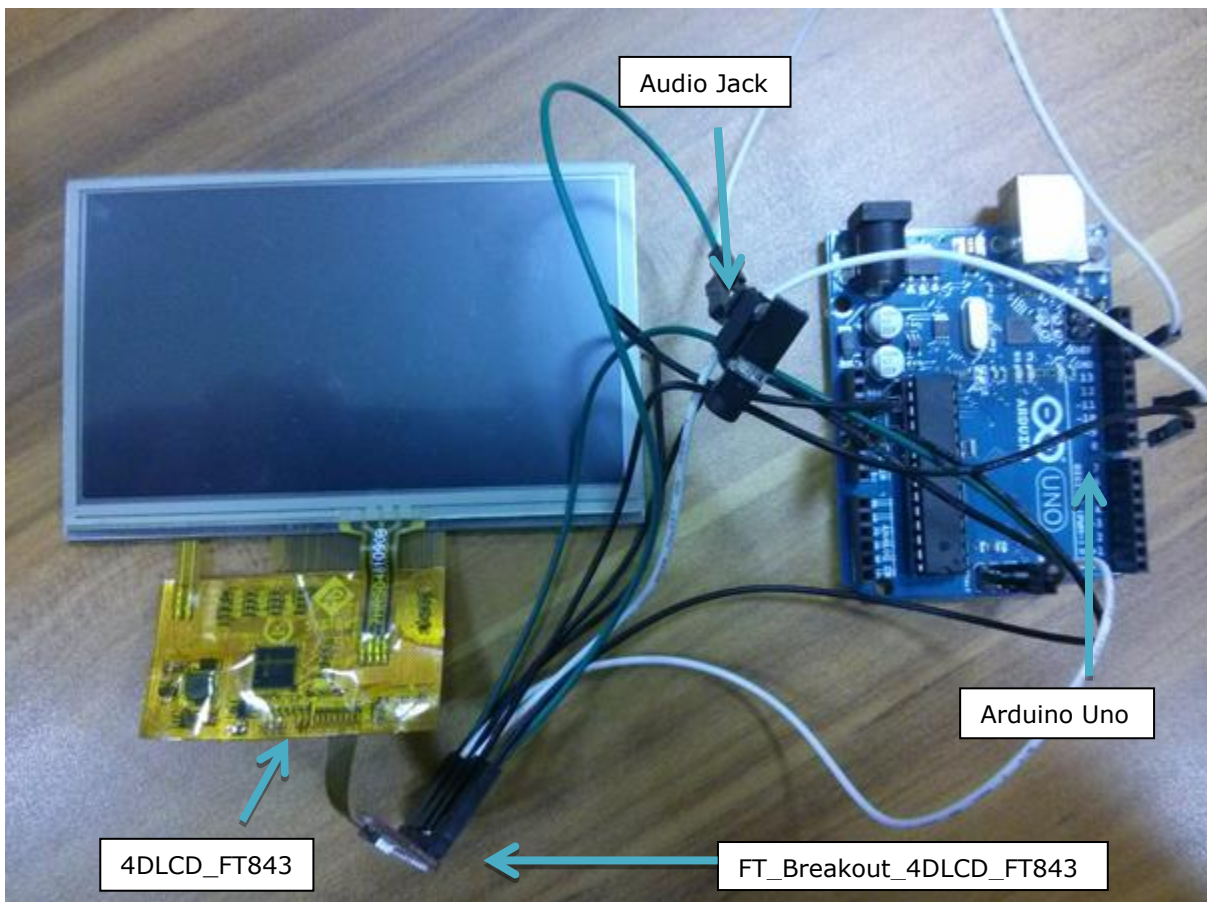
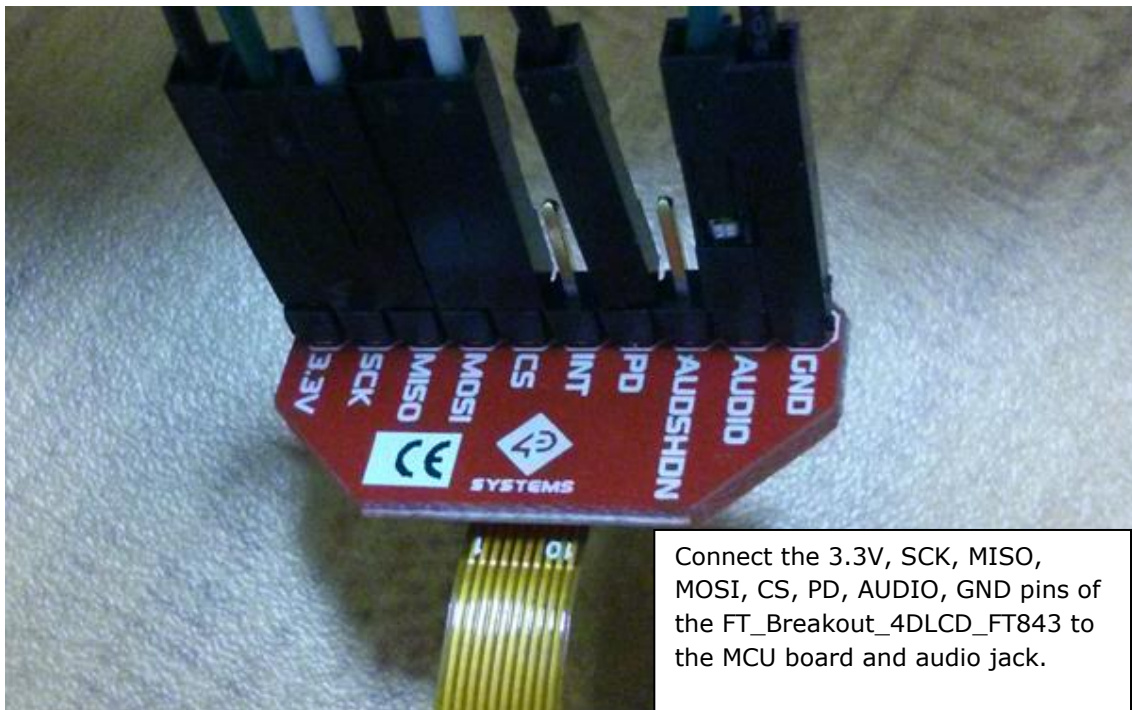


Figure 6-3 : FT\_Breakout\_4DLCD\_FT843 platform block diagram

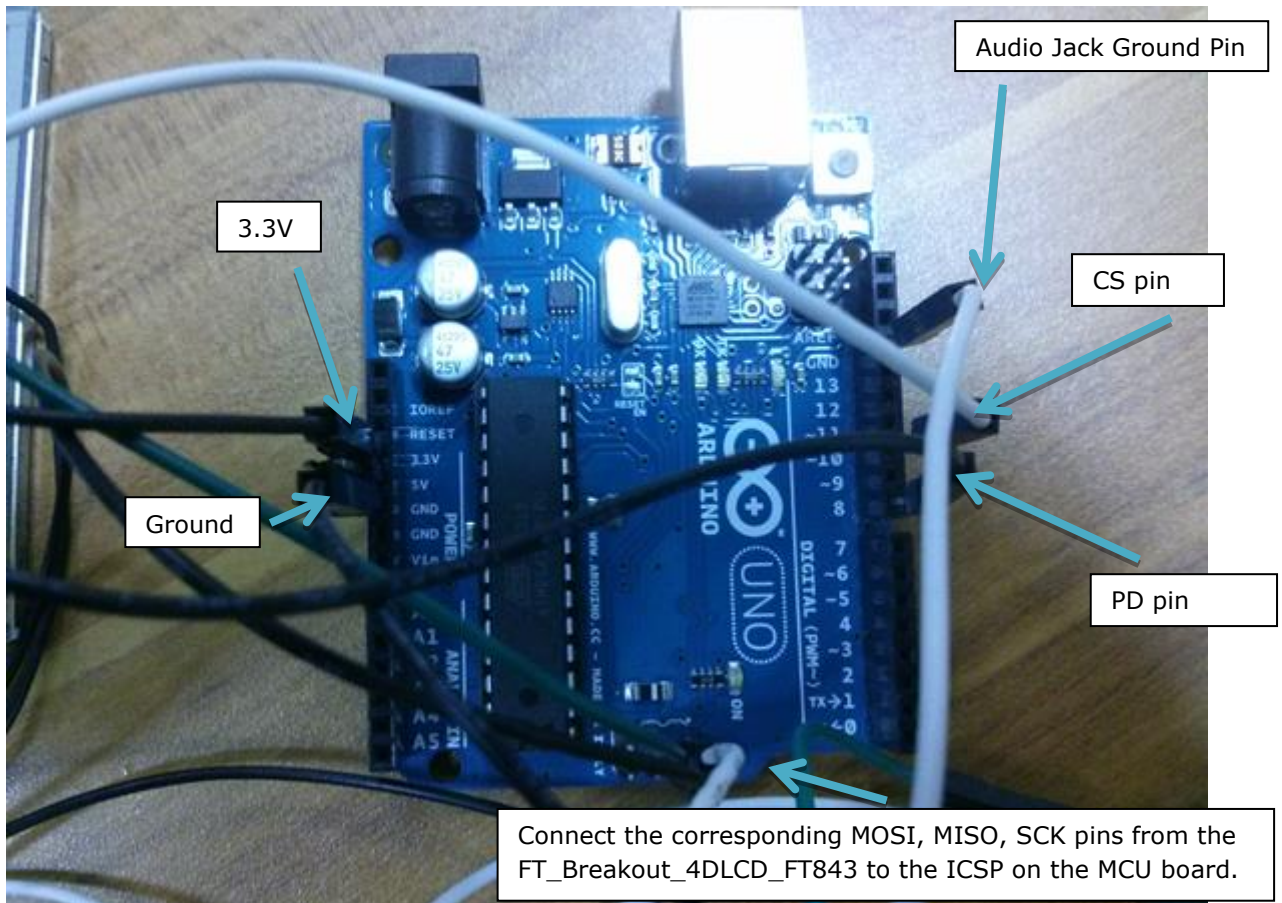


**Figure 6-4 : Hardware setup - FT\_Breakout\_4DLCD\_FT843 platform with Arduino Uno**

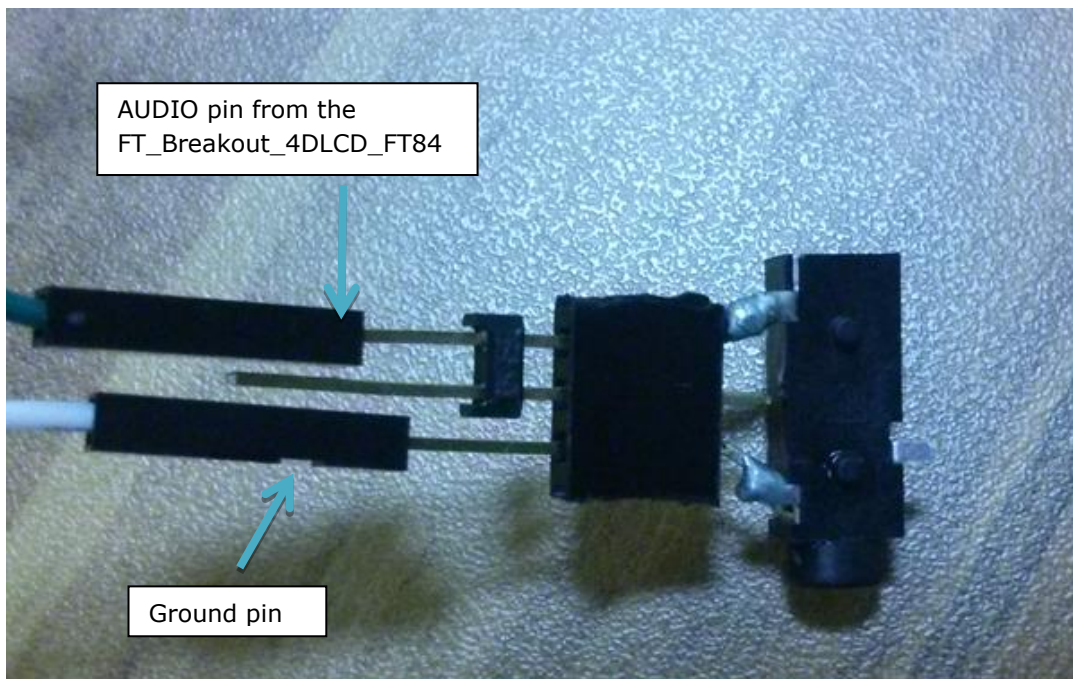


**Figure 6-5 : Populated pins on the FT\_Breakout\_4DLCD\_FT843 breakout board**





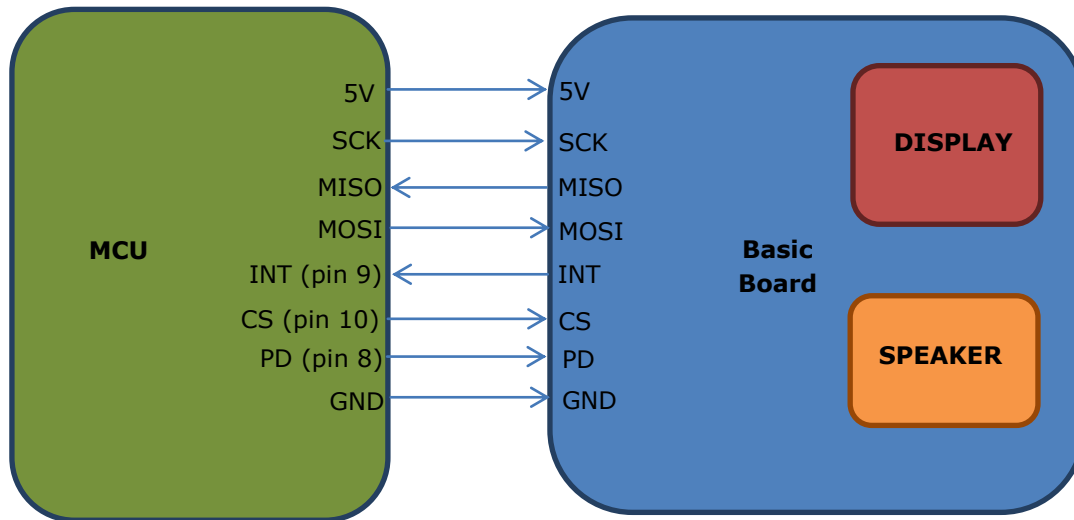
**Figure 6-6 : Wired Arduino Uno for the FT\_Breakout\_4DLCD\_FT843 platform**



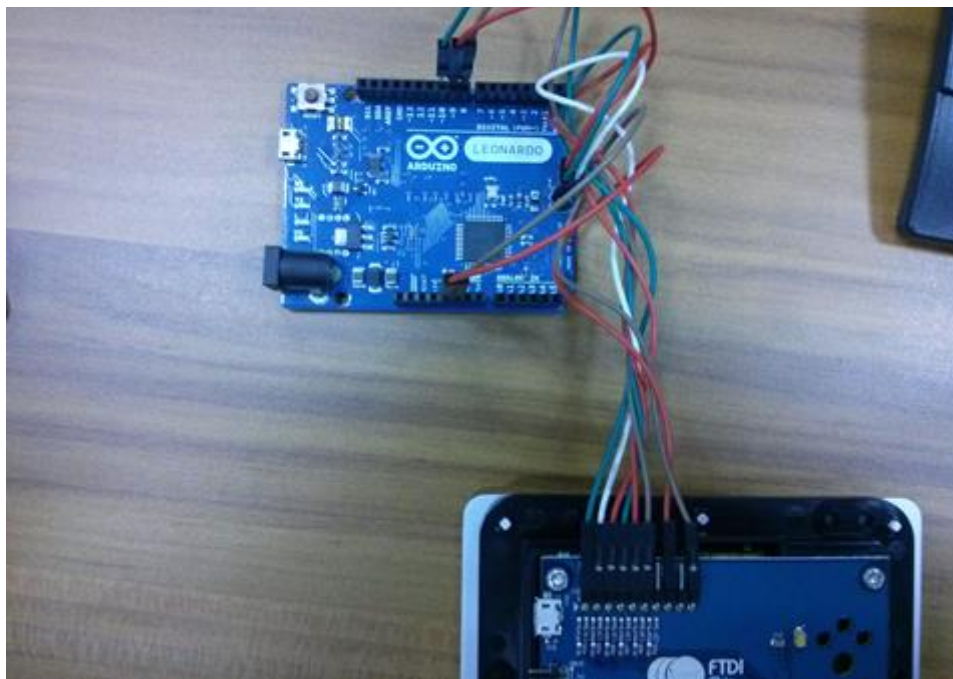
**Figure 6-7 : Audio jack for the FT\_Breakout\_4DLCD\_FT843 platform.**

## 6.6 VM800B43\_50

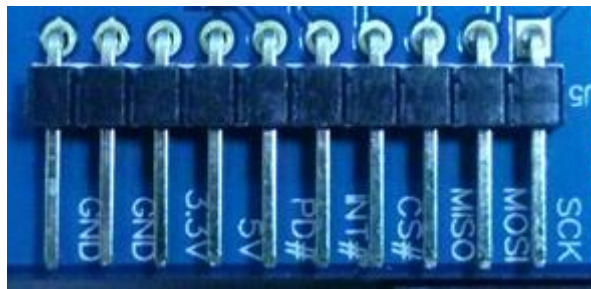
The FTDI Basic board development modules have a display and component board in a plastic enclosure. The modules include a speaker and the back of the FTDI Basic boards have a row of connector sockets to the SPI interface of the system host MCU.



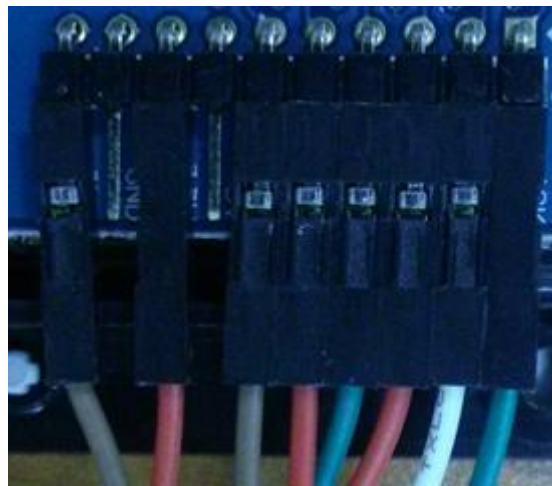
**Figure 6-8 : VM800B43\_50 platform block diagram**



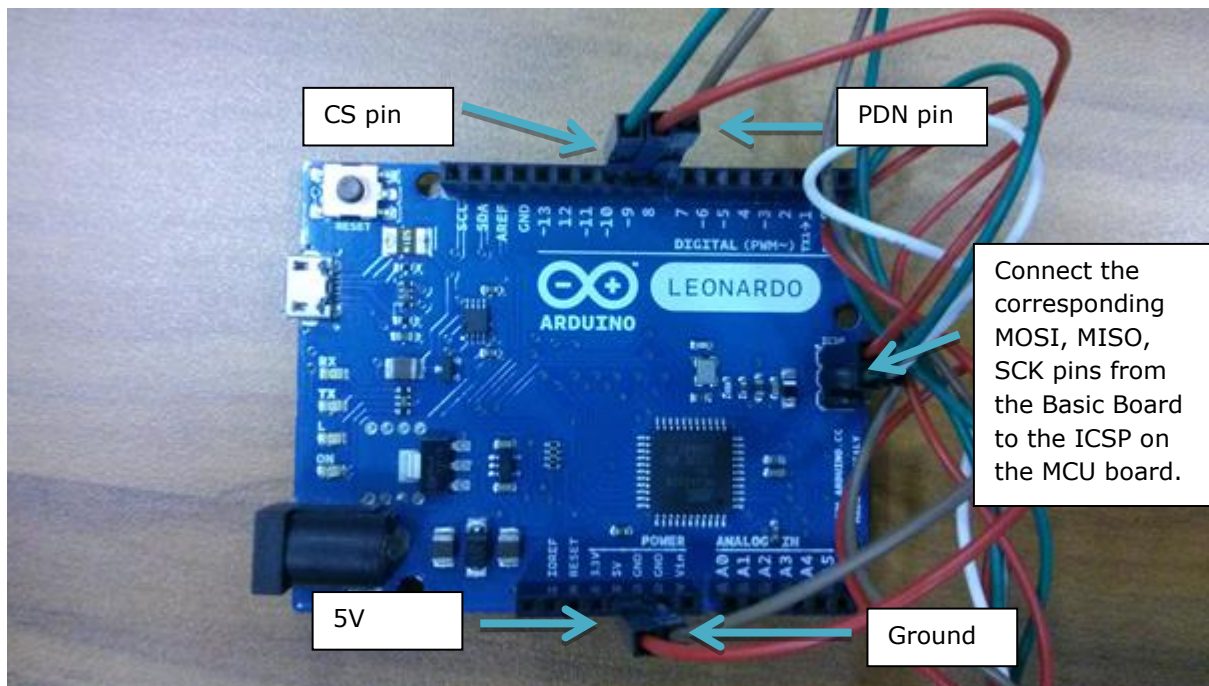
**Figure 6-9 : Basic Board with Arduino Leonardo.**



**Figure 6-10 : Unpopulated pins on FTDI Basic board.**



**Figure 6-11 : Populated pins on FTDI Basic board.**



**Figure 6-12 : Wired Arduino Leonardo for the FTDI Basic board.**

## 6.7 VM800B35

Refer to the VM800B43\_50 platform.

## 7 Contact Information

### Head Office – Glasgow, UK

Future Technology Devices International Limited  
Unit 1, 2 Seaward Place, Centurion Business Park  
Glasgow G41 1HH  
United Kingdom  
Tel: +44 (0) 141 429 2777  
Fax: +44 (0) 141 429 2758

E-mail (Sales) [sales1@ftdichip.com](mailto:sales1@ftdichip.com)  
E-mail (Support) [support1@ftdichip.com](mailto:support1@ftdichip.com)  
E-mail (General Enquiries) [admin1@ftdichip.com](mailto:admin1@ftdichip.com)

### Branch Office – Taipei, Taiwan

Future Technology Devices International Limited  
(Taiwan)  
2F, No. 516, Sec. 1, NeiHu Road  
Taipei 114  
Taiwan, R.O.C.  
Tel: +886 (0) 2 8791 3570  
Fax: +886 (0) 2 8791 3576

E-mail (Sales) [tw.sales1@ftdichip.com](mailto:tw.sales1@ftdichip.com)  
E-mail (Support) [tw.support1@ftdichip.com](mailto:tw.support1@ftdichip.com)  
E-mail (General Enquiries) [tw.admin1@ftdichip.com](mailto:tw.admin1@ftdichip.com)

### Branch Office – Tigard, Oregon, USA

Future Technology Devices International Limited  
(USA)  
7130 SW Fir Loop  
Tigard, OR 97223-8160  
USA  
Tel: +1 (503) 547 0988  
Fax: +1 (503) 547 0987

E-Mail (Sales) [us.sales@ftdichip.com](mailto:us.sales@ftdichip.com)  
E-Mail (Support) [us.support@ftdichip.com](mailto:us.support@ftdichip.com)  
E-Mail (General Enquiries) [us.admin@ftdichip.com](mailto:us.admin@ftdichip.com)

### Branch Office – Shanghai, China

Future Technology Devices International Limited  
(China)  
Room 1103, No. 666 West Huaihai Road,  
Shanghai, 200052  
China  
Tel: +86 21 62351596  
Fax: +86 21 62351595

E-mail (Sales) [cn.sales@ftdichip.com](mailto:cn.sales@ftdichip.com)  
E-mail (Support) [cn.support@ftdichip.com](mailto:cn.support@ftdichip.com)  
E-mail (General Enquiries) [cn.admin@ftdichip.com](mailto:cn.admin@ftdichip.com)

### Web Site

<http://ftdichip.com>

## Distributor and Sales Representatives

Please visit the Sales Network page of the [FTDI Web site](#) for the contact details of our distributor(s) and sales representative(s) in your country.

System and equipment manufacturers and designers are responsible to ensure that their systems, and any Future Technology Devices International Ltd (FTDI) devices incorporated in their systems, meet all applicable safety, regulatory and system-level performance requirements. All application-related information in this document (including application descriptions, suggested FTDI devices and other materials) is provided for reference only. While FTDI has taken care to assure it is accurate, this information is subject to customer confirmation, and FTDI disclaims all liability for system designs and for any applications assistance provided by FTDI. Use of FTDI devices in life support and/or safety applications is entirely at the user's risk, and the user agrees to defend, indemnify and hold harmless FTDI from any and all damages, claims, suits or expense resulting from such use. This document is subject to change without notice. No freedom to use patents or other intellectual property rights is implied by the publication of this document. Neither the whole nor any part of the information contained in, or the product described in this document, may be adapted or reproduced in any material or electronic form without the prior written consent of the copyright holder. Future Technology Devices International Ltd, Unit 1, 2 Seaward Place, Centurion Business Park, Glasgow G41 1HH, United Kingdom. Scotland Registered Company Number: SC136640

## Appendix A– References

### Document References

1. [AN\\_240\\_FT800\\_From\\_the\\_Ground\\_Up](#)
2. [FT800\\_series\\_programmer\\_guide](#)
3. [FT800\\_Embedded\\_Video\\_Engine\\_Datasheet](#)

### Acronyms and Abbreviations

Terms	Description
API	Application Programming Interface
EVE	Embedded Video Engine
IO	Input Output
RTC	Real Time Clock
SD Card	Secure Digital Card
SPI	Serial Peripheral Interface
UI	User Interface
USB	Universal Serial Bus
CTPM	Capacitive touch panel module

## Appendix B – List of Tables & Figures

### List of Figures

<b>Figure 2-1 : Arduino IDE - add library option .....</b>	<b>6</b>
<b>Figure 2-2 : Arduino IDE - library selection browser .....</b>	<b>7</b>
<b>Figure 2-3 : Library Installation - Manual Installation.....</b>	<b>7</b>
<b>Figure 3-1 : FTDI Library folder contents .....</b>	<b>8</b>
<b>Figure 3-2 : FTDI Library - examples folder.....</b>	<b>8</b>
<b>Figure 3-3 : FTDI Library – hardware folder.....</b>	<b>9</b>
<b>Figure 3-4 : FTDI Library - libraries folder.....</b>	<b>9</b>
<b>Figure 3-5 : FTDI Library - platform headers.....</b>	<b>9</b>
<b>Figure 5-1 : Arduino IDE - example sketches .....</b>	<b>28</b>
<b>Figure 5-2 : Arduino IDE - Board selection .....</b>	<b>29</b>
<b>Figure 5-3 : Arduino IDE - Serial Port selection.....</b>	<b>30</b>
<b>Figure 5-4 : Arduino IDE - Upload button .....</b>	<b>30</b>
<b>Figure 5-5 : FT_App_Imageviewer - screen shot 1.....</b>	<b>35</b>
<b>Figure 6-1 : ADAM_4DLCD_FT843 platform block diagram .....</b>	<b>37</b>
<b>Figure 6-2 : Hardware setup - ADAM_4DLCD_FT843 platform with Arduino Due .....</b>	<b>38</b>
<b>Figure 6-3 : FT_Breakout_4DLCD_FT843 platform block diagram .....</b>	<b>39</b>
<b>Figure 6-4 : Hardware setup - FT_Breakout_4DLCD_FT843 platform with Arduino Uno .....</b>	<b>40</b>
<b>Figure 6-5 : Populated pins on the FT_Breakout_4DLCD_FT843 breakout board.....</b>	<b>40</b>
<b>Figure 6-6 : Wired Arduino Uno for the FT_Breakout_4DLCD_FT843 platform .....</b>	<b>41</b>
<b>Figure 6-7 : Audio jack for the FT_Breakout_4DLCD_FT843 platform. ....</b>	<b>41</b>
<b>Figure 6-8 : VM800B43_50 platform block diagram .....</b>	<b>42</b>
<b>Figure 6-9 : Basic Board with Arduino Leonardo. ....</b>	<b>42</b>
<b>Figure 6-10 : Unpopulated pins on FTDI Basic board.....</b>	<b>43</b>
<b>Figure 6-11 : Populated pins on FTDI Basic board.....</b>	<b>43</b>
<b>Figure 6-12 : Wired Arduino Leonardo for the FTDI Basic board. ....</b>	<b>43</b>

## Appendix C– Revision History

Document Title: AN\_318 Arduino Library for FT800 Series

Document Reference No.: FT\_001030

Clearance No.: FTDI# 386

Product Page: <http://www.ftdichip.com/FTProducts.htm>

Document Feedback: [Send Feedback](#)

Revision	Changes	Date
1.0	First release	2014-04-24
1.1	Updated with FT801 details	2014-07-15