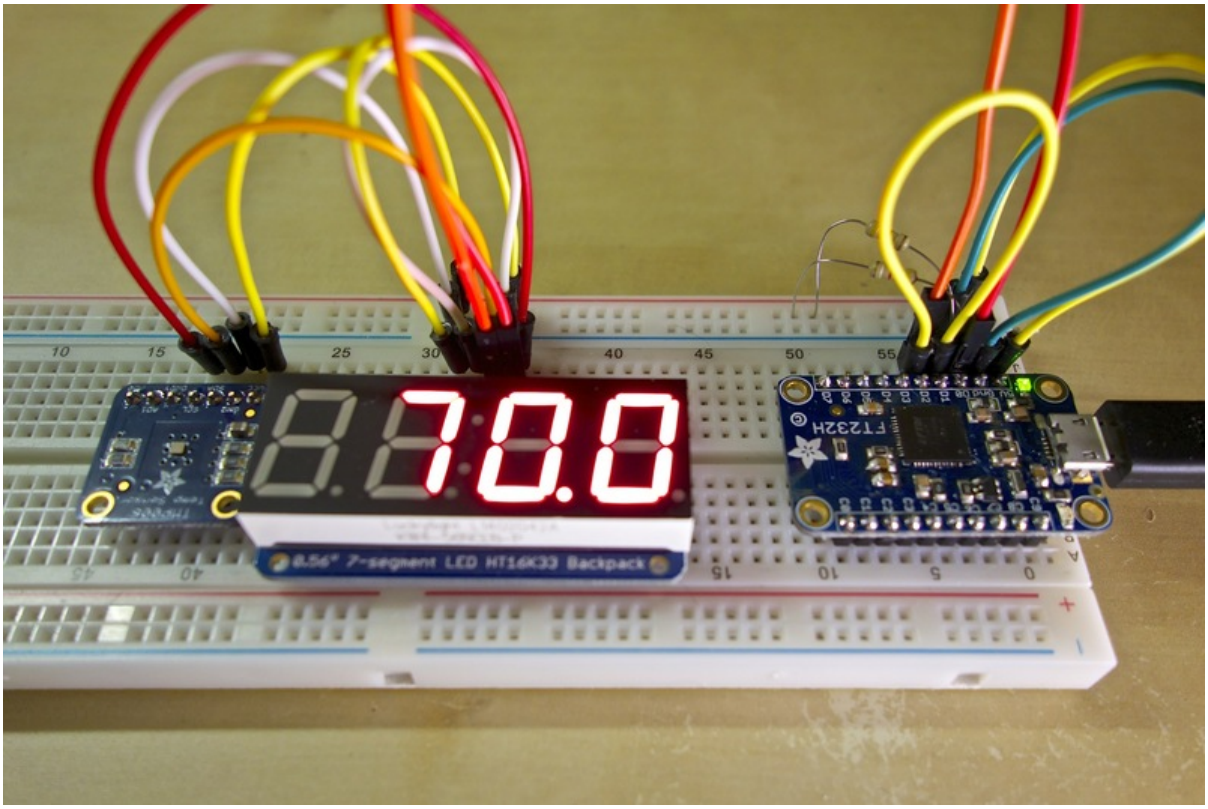
Adafruit FT232H With SPI & I2C Devices

Created by Tony DiCola



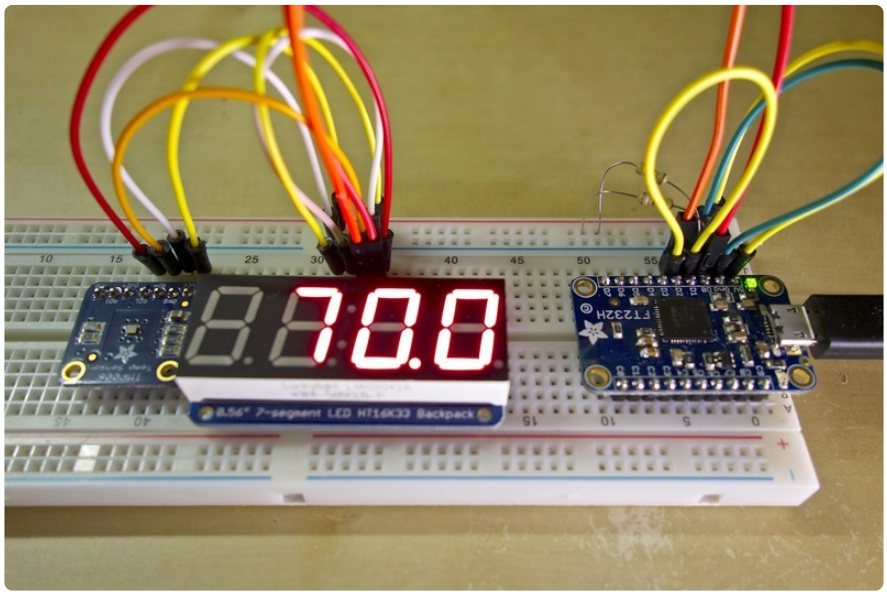
https://learn.adafruit.com/adafruit-ft232h-with-spi-and-i2c-libraries

Last updated on 2024-06-03 01:37:49 PM EDT

Table of Contents

Overview

This library/tutorial has been deprecated! We now have a library that can use all of our CircuitPython drivers and tutorials here https://learn.adafruit.com/circuitpython-on-any-computer-with-ft232h



The Adafruit FT232H breakout (http://adafru.it/2264) is a great way to add GPIOs and an I2C or SPI bus to your computer. There are lots (https://adafru.it/dLP) of (https://adafru.it/dLQ) Adafruit (https://adafru.it/kMD) breakout (https://adafru.it/Ceh) boards (https://adafru.it/Cei) that use I2C or SPI and have been ported to Python for the Raspberry Pi & BeagleBone Black, so can you use those boards with the FT232H breakout too? Yes, you can! With a few simple changes you can make sensors and displays built with the Adafruit Python GPIO library (https://adafru.it/eaG) work with the FT232H breakout too. Follow this guide to learn how to connect SPI / I2C devices and update their code to work the FT232H breakout!

Before you get started you'll want to make sure you're familiar with the FT232H guide (https://adafru.it/Cej) and have followed its steps to install the FT232H driver software.

SPI Devices

This library/tutorial has been deprecated! We now have a library that can use all of our CircuitPython drivers and tutorials here https://learn.adafruit.com/circuitpython-on-any-computer-with-ft232h

Adafruit SPI devices which use the Adafruit Python GPIO library (https://adafru.it/eaG) (i.e. are designed to work with the Raspberry Pi and BeagleBone Black) can easily be configured to work with the FT232H over a SPI connection. Some of these devices include:

- MAX31855 Thermocouple Amplifier Breakout (https://adafru.it/Cek)
- Nokia LCD (https://adafru.it/dLP)
- SSD1306 OLED Displays (https://adafru.it/dLQ)
- ILI9341 LCD (https://adafru.it/Cel)

Wiring

To use one of these devices you'll first want to first read the device's tutorial to get an overview of its connections and library usage. Then connect the device to the FT232H breakout just like you're connecting the device to a Raspberry Pi or BeagleBone Black, but with the following SPI connections:

- **Device SCLK** or clock to **FT232H D0** / serial clock.
- **Device MOSI** or data in to **FT232H D1** / serial output.
- **Device MISO** or data out to **FT232H D2** / serial input.

For other digital pins on the device, such as chip select, reset, data, etc., you should connect those pins to any free GPIO pins on the FT232H, such as **C0** to **C7** (GPIO numbers **8** to **15**).

For device power you can use the 5V pin on the FT232H board to supply up to ~500mA of 5 volt power. Also remember the FT232H board digital outputs operate at 3.3 volts and the digital inputs can safely accept either 3.3 volts or 5 volts.

Software Setup

Follow the device's tutorial and make sure you have installed on your PC any external Python libraries that the device uses For most devices like sensors there are no other Python dependencies to install.

Python Imaging Library For Displays

Many of the display libraries, such as the Nokia LCD, SSD1306, or ILI9341 display, use the Python imaging library (PIL) (https://adafru.it/ed0) which you'll need to install on your PC. On a Windows PC it can be a little tricky to install Python libraries so I recommend installing the PIL setup executable from its website (https://adafru.it/ed0). On a Mac you'll need to make sure Xcode command line tools (https://adafru.it/df2) and PIP (https://adafru.it/deZ) are installed, then run:

```
sudo pip install pil
```

or

```
sudo pip install pillow
```

On a Debian or Ubuntu Linux machine you can install PIL with an apt package by running:

```
sudo apt-get update
sudo apt-get install python-imaging
```

After any dependencies are installed you can install the device's software library just like you were installing it on a Raspberry Pi or BeagleBone Black. You should run the software library's **setup.py** script and pass it the install parameter. For example on a Mac or Linux machine, in a command terminal navigate to the directory with the library's source code and execute:

```
sudo python setup.py install
```

On a Windows machine open a command terminal, navigate to the directory with the library's source code and execute:

```
python setup.py install
```

(make sure Python is added to your path (https://adafru.it/eaM) before running the above!)

Usage

To make the device's example code work with the FT232H you'll need to make a few small changes. First you'll need to include the FT232H module, enable the FT232H, and create an FT232H device by adding to the start of the code:

```
import Adafruit_FT232H as FT232H

# Temporarily disable FTDI serial drivers to use the FT232H device.
FT232H.use_FT232H()

# Create an FT232H device instance.
ft232h = FT232H.FT232H()
```

Next create an FT232H SPI object using the FT232H device. Note that you can optionally set the chip select / secondary select pin by specifying the FT232H GPIO number that's connected to the device's chip select pin.

For example if the device's chip select pin is connected to the FT232H **GPIO 8** / **C0** pin the SPI device would look like:

```
# Create an FT232H SPI object with the specified chip select pin.
ft232h_spi = FT232H.SPI(ft232h, cs=8)
```

Finally create the device object and pass in the FT232H SPI device that was created above as the **spi** parameter of the initializer. If the device uses other GPIO pins, such as a reset or data pin, you'll also need to pass the FT232H device as the **gpio** parameter value.

For example to configure a Nokia LCD with the **data** / **DC pin** connected to FT232H **GPIO 9** / **C1** and the **reset pin** connected to FT232H **GPIO 10** / **C2**, the code would look like:

```
display = LCD.PCD8544(9, 10, spi=ft232h_spi, gpio=ft232h)
```

Putting everything together, here's the full Nokia LCD shapes.py example (https://adafru.it/ed1) converted over to use the FT232H with the **chip select** / **CS** pin connected to **GPIO 8** / **C0**, **data** / **DC pin** connected to **GPIO 9** / **C1**, the **reset** / **RST pin** connected to **GPIO10** / **C2:**

```
    import time

import Adafruit_Nokia_LCD as LCD
import Adafruit_FT232H as FT232H

from PIL import Image
```

```
from PIL import ImageDraw
from PIL import ImageFont

# Temporarily disable FTDI serial drivers to use the FT232H device.
FT232H.use_FT232H()

# Create an FT232H device instance.
ft232h = FT232H.FT232H()

# Create an FT232H SPI object with the specified chip select pin.
ft232h_spi = FT232H.SPI(ft232h, cs=8)

# Create the Nokia display object.
disp = LCD.PCD8544(9, 10, spi=ft232h_spi, gpio=ft232h)

# The code below is exactly the same as the Nokia LCD library's shapes.py example:

# Initialize library.
disp.begin(contrast=60)

# Clear display.
disp.clear()
disp.display()

# Create blank image for drawing.
# Make sure to create image with mode '1' for 1-bit color.
image = Image.new('1', (LCD.LCDWIDTH, LCD.LCDHEIGHT))

# Get drawing object to draw on image.
draw = ImageDraw.Draw(image)

# Draw a white filled box to clear the image.
draw.rectangle((0,0,LCD.LCDWIDTH,LCD.LCDHEIGHT), outline=255, fill=255)

# Draw some shapes.
draw.ellipse((2,2,22,22), outline=0, fill=255)
draw.rectangle((24,2,44,22), outline=0, fill=255)
draw.polygon([(46,22), (56,2), (66,22)], outline=0, fill=255)
draw.line((68,22,81,2), fill=0)
draw.line((68,2,81,22), fill=0)

# Load default font.
font = ImageFont.load_default()

# Alternatively load a TTF font.
# Some nice fonts to try: http://www.dafont.com/bitmap.php
# font = ImageFont.truetype('Minecraftia.ttf', 8)

# Write some text.
draw.text((8,30), 'Hello World!', font=font)

# Display image.
disp.image(image)
disp.display()

print 'Press Ctrl-C to quit.'
while True:
    time.sleep(1.0)
```

I2C Devices

This library/tutorial has been deprecated! We now have a library that can use all of our CircuitPython drivers and tutorials here https://learn.adafruit.com/circuitpython-on-any-computer-with-ft232h

Adafruit I2C devices which work with the Raspberry Pi & BeagleBone Black also use the Adafruit Python GPIO library (https://adafru.it/eaG) and can easily be configured to work with the FT232H. Some of these devices include:

- TMP006 Non-Contact Temperature Sensor (https://adafru.it/Cei)
- MCP9808 Precision Temperature Sensor (https://adafru.it/mEm)
- BMP085/180 Pressure Sensor (https://adafru.it/Ceh)
- Adafruit LED Backpacks (https://adafru.it/kMD)
- SSD1306 OLED (https://adafru.it/dLQ)

To use the FT232H with I2C devices you must add pull-up resistors to the circuit. Read the Wiring section below to understand exactly what's necessary!

Wiring

Like the FT232H guide mentions (https://adafru.it/Cem), when using I2C you'll need to setup your circuit with external pull-up resistors connected to the I2C clock and data lines. This is necessary because the FT232H is a general purpose chip which doesn't include built-in pull-up resistors.

As a quick summary of the I2C wiring, make the following connections:

- Connect FT232H **D1** and **D2** together with a jumper wire. This combined connection is the I2C **SDA** data line.
- Add a 4.7 kilo-ohm resistor from the I2C **SDA** data line (pins **D1** and **D2** above) up to FT232H 5V.
- Add a 4.7 kilo-ohm resistor from FT232H **D0** up to FT232H 5V. This pin **D0** is the I2C **SCL** clock line.

Next wire your I2C device to the FT232H **SDA** and **SCL** lines just as you would for a Raspberry Pi or BeagleBone Black. For other digital pins on the device, such as chip

select, reset, data, etc., you should connect those pins to any free GPIO pins on the FT232H, such as **C0** to **C7** (GPIO numbers **8** to **15**).

For device power you can use the 5V pin on the FT232H board to supply up to ~500mA of 5 volt power. Also remember the FT232H board digital outputs operate at 3.3 volts and the digital inputs can safely accept either 3.3 volts or 5 volts.

Note with the setup above the I2C pins SDA and SCL will be pulled up to 5 volts when idle! Make sure your I2C device can handle this voltage (Adafruit breakout boards, unless noted otherwise, are made to handle 5 volts). If you are using a 3.3V device, simply connect the pullups to 3.3V instead of 5V

Software Setup

Follow the device's tutorial and make sure you have installed on your PC any external Python libraries that the device uses For most devices like sensors there are no other Python dependencies to install.

If you're using a display like the SSD1306 OLED you'll likely need to install the Python Imaging Library (PIL). Check the instructions on the SPI setup page for more details on installing this library (https://adafru.it/Cen).

After any dependencies are installed you can install the device's software library just like you were installing it on a Raspberry Pi or BeagleBone Black. You should run the software library's **setup.py** script and pass it the install parameter. For example on a Mac or Linux machine, in a command terminal navigate to the directory with the library's source code and execute:

```
sudo python setup.py install
```

On a Windows machine open a command terminal, navigate to the directory with the library's source code and execute:

```
python setup.py install
```

(make sure Python is added to your path (https://adafru.it/eaM) before running the above!)

Usage

To make the device's code work with the FT232H you need to make a few small changes to the code. First you'll need to include the FT232H module, enable the FT232H, and create an FT232H device by adding to the start of the code:

```
import Adafruit_GPIO.FT232H as FT232H

# Temporarily disable FTDI serial drivers to use the FT232H device.
FT232H.use_FT232H()

# Create an FT232H device instance.
ft232h = FT232H.FT232H()
```

Then in the device's initializer pass in the FT232H device as the value of the optional **i2c** parameter. For example with the BMP085/180 breakout the code would look like:

```
sensor = BMP085.BMP085(i2c=ft232h)
```

Note that if the device uses other digital pins, such a reset pin, you should also pass the ft232h object as an optional **gpio** parameter. See the SPI page for an example of passing this gpio parameter (https://adafru.it/Cen).

That's all there is to configuring a device to use the FT232H with I2C! You can now use the device just like the library example code shows.

Here's the full simpletest.py example for the BMP085/180 (https://adafru.it/ed5) ported over to use the FT232H:

```
import Adafruit_BMP.BMP085 as BMP085

import Adafruit_GPIO.FT232H as FT232H

# Temporarily disable FTDI serial drivers to use the FT232H device.
FT232H.use_FT232H()

# Create an FT232H device instance.
ft232h = FT232H.FT232H()

# Create BMP085 device with FT232H as I2C provider.
sensor = BMP085.BMP085(i2c=ft232h)

# You can also optionally change the BMP085 mode to one of BMP085_ULTRALOWPOWER,
# BMP085_STANDARD, BMP085_HIGHRES, or BMP085_ULTRAHIGHRES.  See the BMP085
# datasheet for more details on the meanings of each mode (accuracy and power
# consumption are primarily the differences).  The default mode is STANDARD.
#sensor = BMP085.BMP085(mode=BMP085.BMP085_ULTRAHIGHRES, i2c=ft232h)

print 'Temp = {0:0.2f} *C'.format(sensor.read_temperature())
print 'Pressure = {0:0.2f} Pa'.format(sensor.read_pressure())
print 'Altitude = {0:0.2f} m'.format(sensor.read_altitude())
print 'Sealevel Pressure = {0:0.2f} Pa'.format(sensor.read_sealevel_pressure())
```