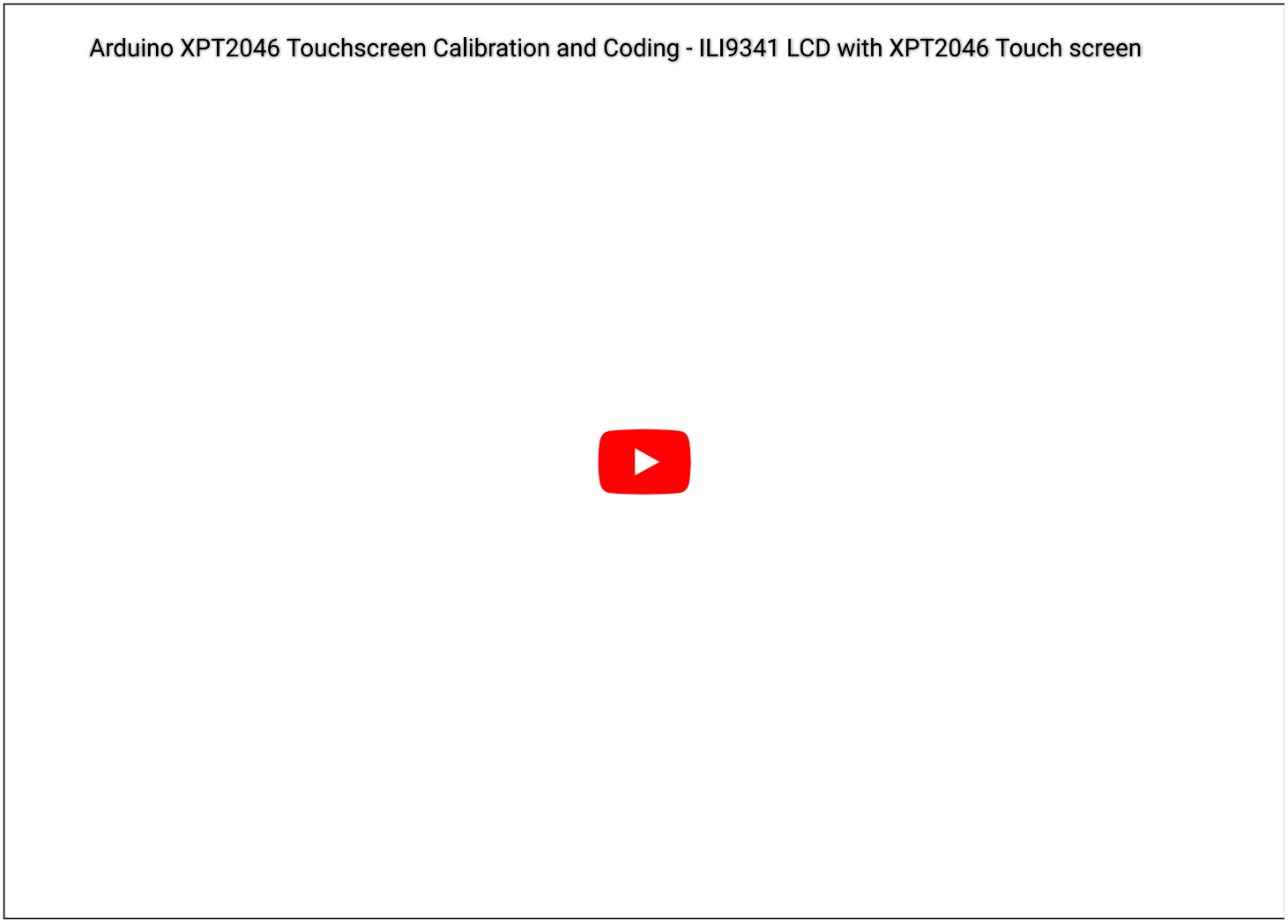
Published by Bob at 2nd December 2020

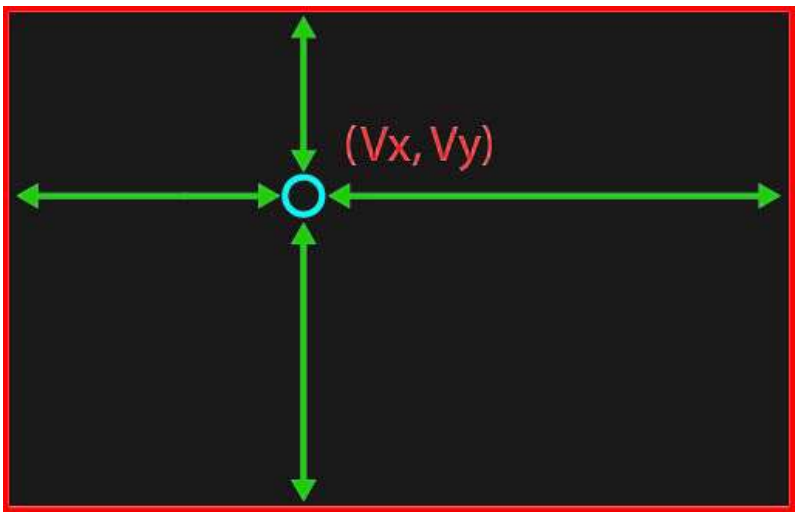
Tags Categories

Calibrating and Coding Your Arduino Touchscreen



How a Resistive Touchscreen Works

In simple terms a resistive touch panel is two sheets of film separated by a non conducting layer, usually a gas or liquid. When you press on the screen you force the two film layers to touch and make an electrical connection. The screen's electronics are able to sense the position of this touch connection using sensors that measure resistance between opposite edges of the panel. Basically the touch point creates a potential divider circuit in both the horizontal and vertical planes of the panel.



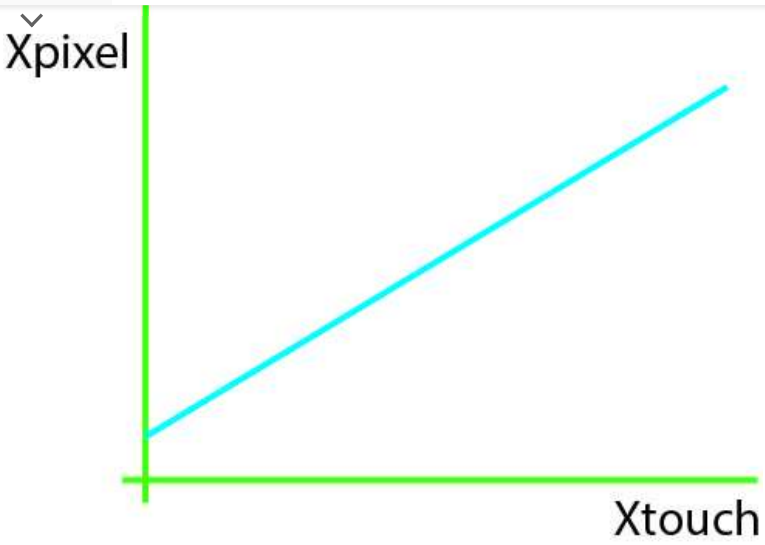
These measurements are then translated into values by a touchscreen driver chip and then made available to the Arduino (or whatever microcontroller you're using).

Calibrating the Touchscreen and LCD Panel

The touch panel and LCD screen are two separate devices. Although they are bonded together there is no connection between the two. The value sent when you touch the resistive panel don't match with the pixel coordinate values on the LCD panel. We need to develop an algorithm that will let us translate touch panel coordinates so they match LCD pixel coordinates. This is a process of calibrating the touchscreen.

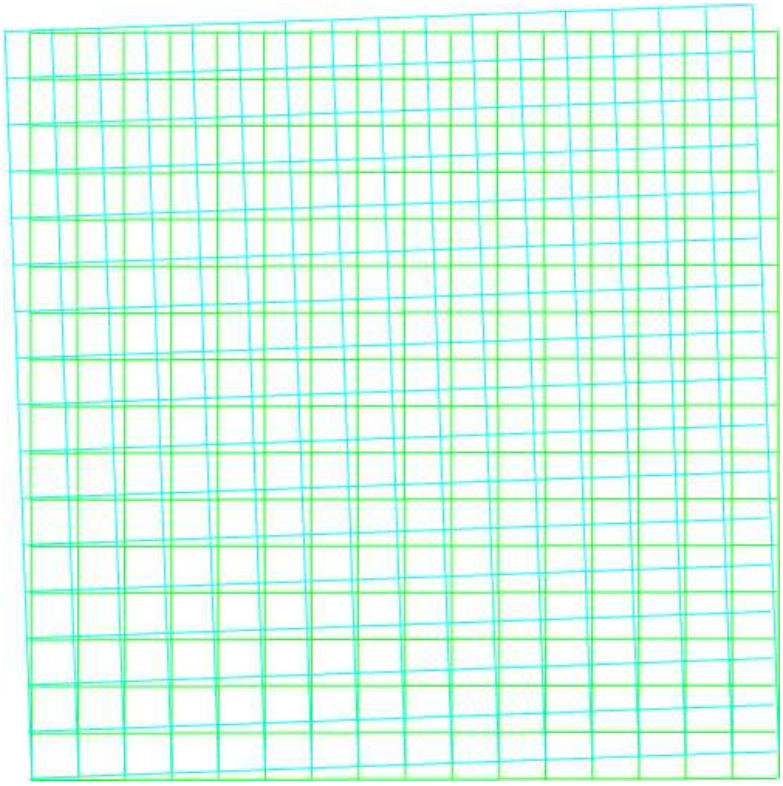
As we've already seen the touch panel driver chip will return X and Y coordinate values based on the resistance measurements horizontally and vertically on the device. These values will vary proportionally across the width and height of the screen. The zero values for the touch panel won't match the zero values for pixels on the LCD screen, but there will be a linear relationship between the two coordinate systems.

This means that we can model the relationship between the X coordinates on the touch and LCD panels as a straight line graph. Similarly the Y



To find the equation of each line we need to take a number of readings so that we can match pixel coordinates with their corresponding touch panel coordinates. The number of readings you need to take depends on how accurately you want to model the connection between the LCD and touch panels. We already looked at the simple horizontal relationships. But this also another manufacturing error that we might want to consider.

When the two panels are joined together the vertical and horizontal grid lines might not be aligned, as shown in the diagram below.



This misalignment can give us an error which will vary across the width and height of the screen.

Having said that, and especially for small screens, this misalignment error is going to be very small. The calibration process involves the user

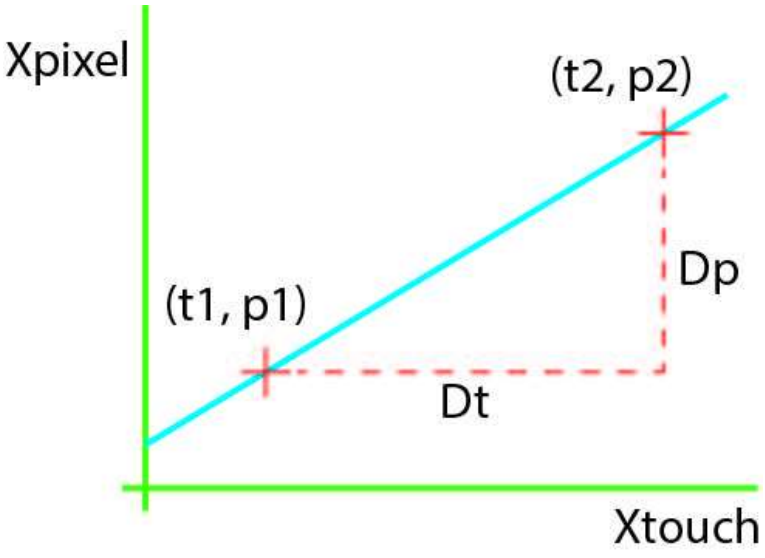
For true touch panel calibration, including the misalignment error, you need to use three point calibration. The mathematics for this are a little bit complicated but again you will be able to find an Arduino package which will have this code pre-written for you.

For this project will be using a simple two point calibration which will let us create the linear relationships between the X and Y coordinates but ignore any misalignment errors.

Calculating the Linear Relationships

To calculate the linear equations we need the user to identify two points on the screen. We'll do this by setting a target in one corner, asking them to touch the target and then reading the touch panel values which can be logged against the known pixel coordinate values of the target. We can then repeat the process in the opposite corner. This gives us two sets of values where we have two sets of X coordinates with pixel values and their matching touch panel values, and two sets of Y coordinate values.

We can then use these to calculate the linear equations for both the X and Y coordinates.



For the X coordinates as shown in the graph above we are trying to create straight-line equation of the form,

Xp = m Xt + c

m = (p2 – p1) / (t2 – t1)

We can then substitute this value into our equation and use one of our measured points to find the value for c.

Xp = m Xt + c

c = Xp – m Xt

c = p2 – m t2

Once we got the values for our X coordinate linear equation we can simply repeat the process for the Y coordinates.

Coding the Conversion Function

We only need to run the calibration code when the Arduino is first turned on. We can then store linear equations within our software and use them in a conversion function which will allow us to passing or touchscreen coordinates and get back a matching set of screen pixel coordinates.

We wanted to we could also store linear equation parameters in the EEPROM of the Arduino. When we then turn on the Arduino we can check the EEPROM to see if there are any valid parameters and then load those instead of running the full calibration routine.

Building a Touchscreen User Interface

Now that we are able to create graphics on the LCD panel, listen for touch events on the touch panel and convert touch coordinates the screen pixel coordinates, we can start to create buttons, sliders and other user interface components for our projects. This opens up a very powerful and user-friendly way of communicating with you Arduino powered devices. The LCD screen gives you an infinite array of options on how your device communicates back to the user and the touch panel can replace a vast array of buttons and dials. It's really down to your imagination to work out what you can do with it.

Code Used in the Video

+ Touchscreen Calibration Code
+ Touchscreen Button Example

Links

LCD module – Amazon

https://amzn.to/35quUJF

LCD eBay

https://rover.ebay.com/rover/1/710-53481-19255-0/1

ILI9341 Library

https://github.com/adafruit/Adafruit_ILI9341

XPT2046 Library

https://github.com/PaulStoffregen/XPT2046_Touchscreen

Adafruit GFX Library

https://github.com/adafruit/Adafruit-GFX-Library

Adafruit LCD tutorial

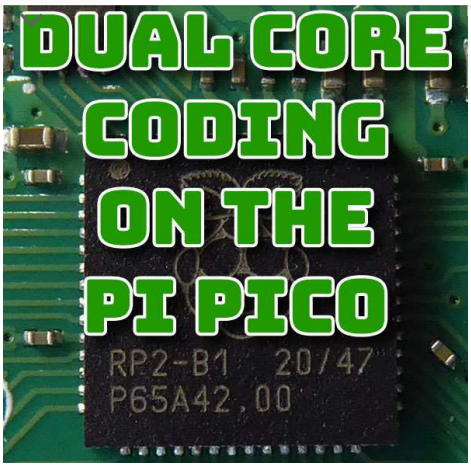
https://learn.adafruit.com/adafruit-2-8-and-3-2-color-tft-touchscreen-breakout-v2/spi-wiring-and-test

Adafruit GFX Library Tutorial

https://learn.adaruit.com/adafruit-gfx-graphics-library/overview

Bob

Related posts



19th April 2022

Multi Thread Coding on the Raspberry Pi Pico in Micropython

Read more

12th April 2022

Pi Pico SPI LCD Driver Using RAM Frame Buffer – ILI9341 and ST7789

Read more

23rd January 2022

Don't Scrap Your Old Laptop. Upcycle It! Use I – Coding Development System

Read more

Comments are closed.

- Privacy Policy
- Contact Me