Project Reflection - Touch Screen Tester

Comark LLC

Om Patel

January 6, 2025

Due to an NDA I cannot go into full detail about certain parts of this project. The information here covers only what I am able to share.

For this project I took an old 3D printer and turned it into a touchscreen testing device that could do the same tests my company used to do by hand. I started by using my knowledge of G-code to make the 3D printer head move to any location I told it to. At first I would write my own G-code file and then put it on an SD card and load it into the printer. When I saw that the printer was moving exactly how I wanted I started to think about how I could make this faster. I realized that Python could take coordinates that I gave it and turn them into G-code commands. This meant I could have the code automatically create the printer movements instead of typing each one. I first made a small Python script that would take size descriptions and make G-code that moved the head from one side of the printer to the other. Once that worked I added to it so the printer could trace around the edges of the printer bed to test its accuracy.

With the basics figured out I went to my boss to find out what tests they normally did on the screens. They told me that they would run the tester from the top to the bottom of the screen, from left to right, and also diagonally. They would also try to touch every single part of the screen at least once, sometimes more than once, to check for what we call double clicks where one touch is read as two touches. With those ideas in mind I updated my Python code so that it would ask for the dimensions of the screen and use that as the maximum and minimum movement range for the G-code.

The first test I made would move the printer head from the far left side of the screen to the far right side. After each pass it would move up by one pixel and then repeat until the top of the screen was reached. This was easy to do with a for loop and a counter that kept track of how many passes were done. After making this test I created another version that moved up and down instead of left to right. All I had to do was swap the X and Y values in the G-code.

Next I made a test that would touch every part of the screen just like a person would if they were checking every pixel by hand. This meant the printer head had to move off the screen and then back on for every single touch. To make that happen I added four extra lines of code that told the head to move away from the screen and then back to the test point before moving to the next point. I also added an input so the user could set how much space they wanted between each touch.

Once the main tests were working I wanted a better way to organize the process so that anyone could use it without editing the code. I decided to use a Python library called Flask which lets you create websites. I made a web page that asked for the width and height of the screen, the spacing between test points, and checkboxes for which tests to run. The page also had a submit button that would run the Python code and generate both a G-code file and a CSV file. The G-code file could be saved to an SD card and loaded into the printer. The CSV file would contain a list of all the points the printer was supposed to touch so it could be compared later with the real results.

To check the actual touch data the company uses a Beagle USB analyzer. This tool connects the touchscreen to a laptop and uses special software to treat the touchscreen like a mouse. It records the exact X and Y coordinates for every touch and saves them into a CSV file. I realized that my CSV output used the same kind of coordinates. That meant I could compare the CSV I made with the one from the analyzer and see if there were any missed touches, double clicks, or touches in the wrong place. I wrote a comparison program that lines up each predicted point with the actual results and then reports the accuracy. It also makes a new CSV that highlights any problems so they can be seen quickly.

I updated the Flask web page so that the submit button also creates this predictive CSV along with the G-code file. I also made it so that after a test is done the user can upload the actual results CSV into the page and see a report on the screen accuracy along with the highlighted CSV file of anomalies.

Before I made this device the company would have one person test each screen for hours to make sure it was accurate enough to be used in harsh military environments. Now the process is automated and can run without anyone having to sit there and tap the screen all day. It saves a lot of time, reduces human error, and makes the tests more consistent. The accuracy report can also be sent to the client as proof that the screen has been tested and meets the standards. It is easy to change the test settings for different screens and new test patterns can be added to the code quickly.

This project was useful because it took a slow manual process and turned it into something faster, more reliable, and repeatable. It taught me how to use hardware like a 3D printer in new ways, how to connect it with software tools like Python and Flask, and how to handle data so that it can be compared and analyzed automatically. It also showed me how to take feedback from a supervisor and turn it into working features that make the process better for everyone.

Approved by Jarrett Schroeder | Electrical Engineer | Comark LLC | 1/7/25