Available online www.jsaer.com

***Journal of Scientific and Engineering Research*, 2020, 7(7):282-288**

Research Article

ISSN: 2394-2630
CODEN(USA): JSERBR

Low-Cost CNC Plotter for Automated Mobile Testing: Design and Implementation

Sambu Patach Arrojula

email: sambunikhila@gmail.com

Abstract: Manual testing of mobile applications is a tedious and error-prone process, making automation highly beneficial for ensuring consistency and efficiency. But if the test suite has to operate or interact with some external devices regular automation solutions would not be sufficient and automation plotters would be a best fit here. While CNC plotters can be used for this purpose, industrial-grade plotters are often prohibitively expensive. This paper discusses a design and implementation of a cost-effective CNC plotter with moderate precision, tailored for mobile testing automation. The plotter interfaces with test suites running on PCs and achieves sufficient accuracy for mobile testing tasks at a significantly lower cost.

Keywords: CNC plotter, mobile testing automation, low-cost design, moderate precision, Arduino, test suite integration, manual calibration.

Introduction

Automating the mobile application testing process has become increasingly important due to the growing complexity and variety of mobile apps. Manual testing is not only time-consuming and tedious but also prone to human error, which can compromise the quality and reliability of the applications. Automation helps ensure consistency, repeatability, and efficiency in testing, leading to more robust and reliable applications. However, some specific applications have use cases where these applications interact with some external devices, non mobile devices like some kiosk systems or iPhones etc where regular Android application automation tools might not be sufficient. For this purpose we can rely on automation plotters.

CNC plotters, widely used in various industries for precision tasks, offer a promising solution for use cases like laser cutting, manufacturing, model building for various components etc. However, the high cost and advanced features of industrial-grade CNC plotters often make them overfit for the specific needs of mobile testing where their need would be minimal interactions with those external devices.

We here in this paper address this issue by presenting a low-cost CNC plotter specifically designed for mobile testing automation. By focusing on a design that meets the fundamental needs of mobile testing, our plotter offers a cost-effective and efficient solution. It provides the necessary accuracy for executing test cases while maintaining affordability and simplicity.

Use Case

In developing our low-cost CNC plotter, we considered several key use cases to ensure it meets the fundamental requirements for automating mobile testing. The following use cases highlight the specific scenarios for which this plotter is designed:

1. Automation of Mobile Phones with Full Touch Screens

The plotter is intended for automating test cases on mobile phones that feature full touch screens.For now we implemented only taps but not swipes, and multi-touch gestures.

2. Precision Suitable for Mobile Automation

The plotter provides precision that is adequate for mobile testing automation. While it may not match the extreme precision of high-end industrial CNC plotters, it is sufficiently accurate to ensure reliable and consistent test case execution on mobile devices.

3. Applications Interacting with Finger Touch

Our plotter is designed to interact with applications that require finger touch input. It is not intended for applications that require stylus or pencil interactions, which typically demand higher precision and different interaction dynamics.

4. Manual Calibration for Each Mobile Phone

Developers can calibrate the plotter manually for each mobile phone. This calibration involves setting the boundaries on the base platform and determining the appropriate touch pressure levels. Once calibrated, the plotter can accurately execute the necessary touch actions for the specific mobile device being tested.

Design

The primary goal of this CNC plotter is to create a low-cost yet reliable device specifically for mobile testing purposes. To achieve affordability, we chose Medium Density Fiberboard (MDF) as the main construction material instead of metals typically used in industrial CNC plotters. While metals provide superior sturdiness and accuracy, MDF is sufficiently strong for mobile testing automation and significantly cheaper, making it an ideal choice for our application.

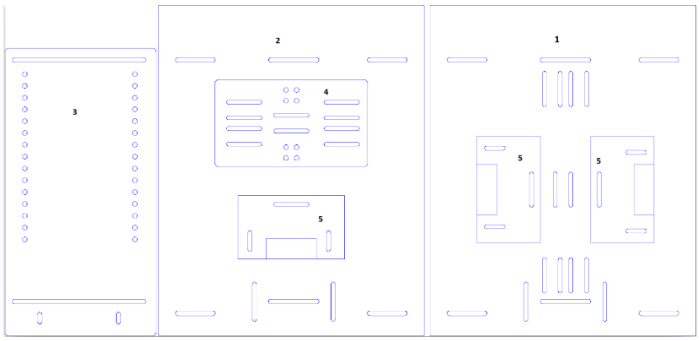
One of the key challenges in using MDF is maintaining the necessary sturdiness and accuracy. Industrial plotters benefit from the rigidity of metal components, but our design compensates by minimizing the moving parts on the Z axis. In many industrial designs, the Z plane moves both vertically and horizontally, but the same would add complexity and potential instability in our case as the MDF is not sturdy enough for such a build. Our plotter, for that reason, isolates the Z axis movement to vertical motion only, while stacking the X and Y planes on the base. This arrangement enhances stability by reducing the load on the Z axis, leveraging the inherent strength of MDF more effectively.

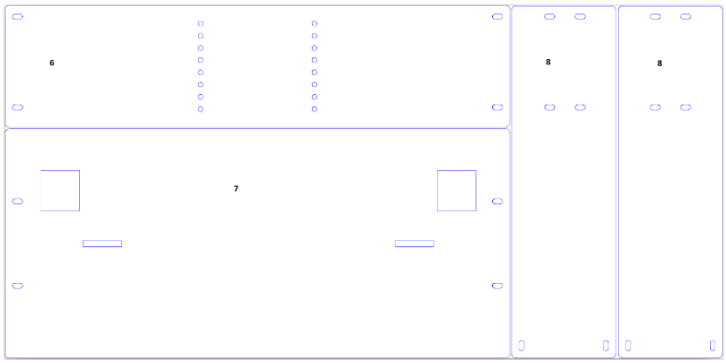
In our design, the X and Y planes are stacked, with the bottom plane dedicated to one directional movement (either X or Y), and the top plane handling the other directional movement along with the initial direction(coz of stacking). This setup does trade off some accuracy due to the stacking of two big planes, but it significantly improves sturdiness. By focusing on the structural integrity provided by this arrangement, we achieve a balance where the reduced accuracy is acceptable within the tolerances required for mobile testing. This design decision emphasizes reliability and cost-efficiency, ensuring that the plotter meets the specific demands of automating mobile test cases without the unnecessary precision and expense of industrial-grade machines.

Components

- MDF sheets 0.25 inch 31X15 - 2
- Step motors Nema 17 Bipolar 40mm 64oz.in(45Ncm) - 1.8 deg. step angle (200 steps/revolution) - 2A - 3
- 300mm & 8mm lead screw rail shaft support pillow block bearings & shaft coupling - 3
- Arduino controller with Expansion Board V3.0 +UNO R3 Board + A4988 Stepper Motor Driver with Heatsink - 1

We used two sheets of 0.251 inch thick MDF each of size 31 X 15 inches and did lazer cut into specific parts as shown below. We had a separate 10 X 10 sheet for the Mobile platform.

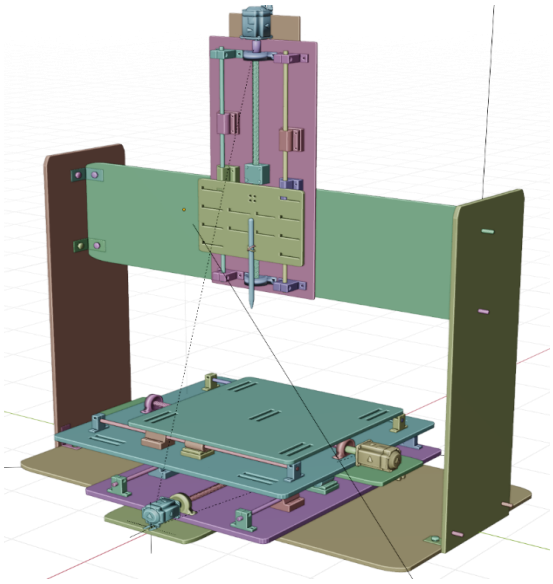




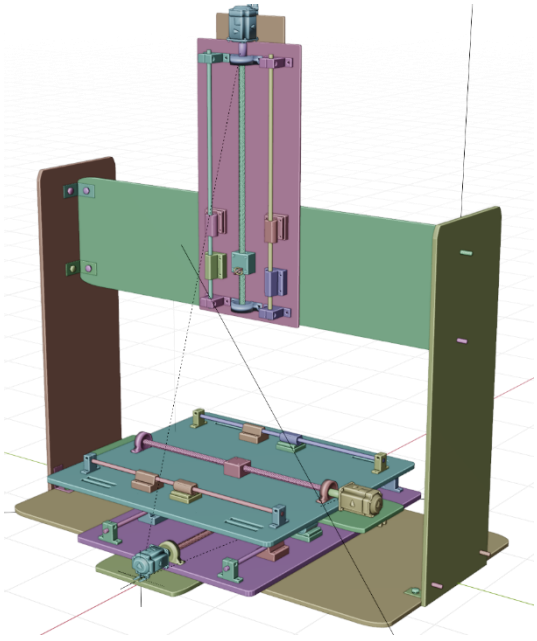
1 - X Plane
2 - Y Plane
3 - Z Plane
4 - Z Plane platform (stylus holder)
5 - Step motor stands (3 of them)
6 - Frame shoulder connecter (holds Zplane)
7 - Frame base
8 - Frame shoulders (2 of them)

First a frame is built with 7,8 and 6 pieces indicated above to host the X and Y planes. then X plane (1 indicated above MDF sheet details) is fixed and then lead screw railings (shown below pic) are fixed on that X plane.

then, step motor fixed at shaft and Y panel (2 indicated in MDF sheet details) is fixed on this railing so that when motor runs this Y panel will move back and forth in (X) railing direction, Where Y panel it self has the same setup as X panel except the Mobile platform (a separate 10X10 MDF) is fixed on its railing. Effectively with both X & Y direction motors we can move the Mobile platform in X, Y plane. Now similar to X & Y planes Z plane (3 indicated in MDF sheet details) would have the railing plus motor with Z platform with stylus fixed on its railing. so that Z direction motor will control the tap operation. Final look of plotter will be something like this.



plotter with all planes installed



plotter railings exposed (platforms removed)

Software

To operate the CNC plotter, we utilize an Arduino controller, which is programmed to interpret and execute commands from a designated test suite. The Arduino controller acts as the intermediary between the software and the hardware components of the plotter. It receives high-level instructions, through serial communication, and translates these instructions into precise step operations that control the targeted motors. These motors, often stepper motors, are responsible for moving the plotter's axes and tool head to the specified coordinates. The Arduino's firmware is configured to handle these conversions accurately, ensuring that each command results in the correct movement and operation of the plotter.

A. Calibration of plotter

Before integrating the plotter-SDK test suite developer will put plotter into calibration mode. in this mode plotter will execute some fixed code which will first place X and Y planes at their beginning positions, indicating them like (0,0) then move X & Y planes positions such that the stylus above draws a fixed size rectangle on the platform. Lets Say

- X plane displacement from (0,0) M
- X plane motors steps for above displacement Ms
- Y plane displacement from (0,0) N
- Y plane motor steps for above displacement Ns

M,N,Ms and Ns are fixed in the plotter and give out the reference rectangle of its play field to the Test suite developer. Now he makes a plot of that MxN rectangle into his provided units which would be granular enough for typical application automation. Say MxN rectangle is made TxU units then

steps required to move X panel to each unit would be

$$S_x = \frac{M_s}{T}$$

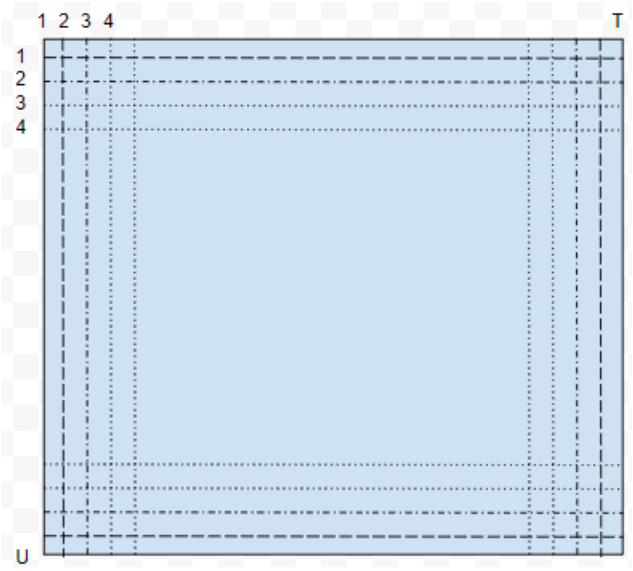
steps required to move Y panel to each unit would be

$$S_y = \frac{N_s}{U}$$

and to move plotter point to (x,y) the steps need to perform corresponding motors would be

$$\left(x \bullet \frac{M_s}{T}, y \bullet \frac{N_s}{U} \right)$$

For now we made these T & U fixed but can be configured in SDK as per Tester needs, say if he needs more granular units for example he can increase T & U values.



Now that Test suite developer knows the plot he places his device in that plot and knows relevant plot positions

B. Arduino driver

Arduino will be programmed to receive commands through serial communication i.e. from the Test suite.

Test suite will be sending string commands through serial port asking like goto (x,y) , tap and driver will convert them to step requirements per motor and executes them.

Pseudo code for the same

```
Function void loop():
1. Initialize Constants:
  - Ms = [value]  // Total steps for X motor
  - T = [value]  // Total distance range for X axis
  - Ns = [value]  // Total steps for Y motor
  - U = [value]  // Total distance range for Y axis
  - tapSteps = [value]  // Number of steps for Z axis tap

2.process command if available:
  a. If data is available on serial port:
    i. Read command from serial port
    ii. If command starts with "GOTO:":
      - Extract x and y values from command
      - Convert x and y values to integers
      - Calculate x_steps as x * (Ms / T)
      - Calculate y_steps as y * (Ns / U)
      - Call moveMotorX(x_steps)
      - Call moveMotorY(y_steps)
    iii. Else if command is "TAP":
      - Call performTap(tapSteps)
  b. Continue loop

Function performTap(tapSteps):
  - Call moveMotorZ(tapSteps)  // Move Z motor clockwise
  - Call moveMotorZ(-tapSteps) // Move Z motor anticlockwise

Function moveMotorX(steps):
    • Send step signals to X plane motor  perform given number of steps

Function moveMotorY(steps):
    • Send step signals to Y plane motor  perform given number of steps

Function moveMotorZ(steps):
    • Send step signals to Z plane motor  perform given number of steps
```

C. Test suite integration

For integration, we have provided a concise Java class that exposes a simple interface for the test suite to include. This allows you to call functions to perform actions such as goto() and tap(). This class relies on the jSerialComm library for serial communication. Testers can first use the plotter to play coordinates and place their Device Under Test (DUT) accordingly. Knowing the specific coordinates needed for each tap function, they can simply use these classes, calling goto with the coordinate values and tap() for the tap operation.

```
<dependency>
  <groupId>com.fazecast</groupId>
  <artifactId>jSerialComm</artifactId>
  <version>1.3.11</version>
</dependency>
```

```
import com.fazecast.jSerialComm.SerialPort;

public class Plotter {
  private SerialPort serialPort;
  private static final String PORT_NAME = "COM3"; // Change to your serial port
name

  public Plotter() {
    serialPort = SerialPort.getCommPort(PORT_NAME);
    serialPort.setComPortParameters(9600, 8, SerialPort.ONE_STOP_BIT,
SerialPort.NO_PARITY);
    serialPort.setComPortTimeouts(SerialPort.TIMEOUT_SCANNER, 0, 0);

    if (serialPort.openPort()) {
      System.out.println("Port opened successfully.");
    } else {
      System.out.println("Unable to open the port.");
      return;
    }
  }

  public void goto(int x, int y) {
    sendCommand("GOTO:" + x + "," + y);
  }

  public void tap() {
    sendCommand("TAP");
  }

  private void sendCommand(String command) {
    try {
      byte[] commandBytes = command.getBytes();
      serialPort.writeBytes(commandBytes, commandBytes.length);
    } catch (Exception e) {
      e.printStackTrace();
    }
  }

  public static void main(String[] args) {
    Plotter plotter = new Plotter();
    plotter.goto(10, 20);
    plotter.tap();
  }
}
```

Conclusion

The design and implementation of a cost-effective CNC plotter tailored for mobile testing automation, as presented in this paper, offer a practical solution to the challenges posed by manual testing, especially when external devices are involved. By leveraging the affordability of Medium Density Fiberboard (MDF) and focusing on moderate precision, this plotter meets the essential needs of mobile application testing without the unnecessary complexity and cost of industrial-grade machines. The plotter's ability to interact with test suites through an Arduino controller provides a seamless integration with existing automation frameworks, ensuring that mobile applications can be tested with consistency and accuracy. While the precision is not on par with high-end CNC plotters, it is sufficient for the specific use cases targeted, making this approach a viable option

for developers and testers seeking to enhance their automation capabilities on a budget. This solution underscores the potential for innovation in mobile testing automation through thoughtful design and resourceful engineering.

Next Steps

1. Y Plane Improvements: Currently, we are using the same motor and railing system for the X, Y, and Z planes. However, the Z plane is primarily used for tapping operations, which are lightweight and do not require the same capacity as the X and Y planes. By reducing the motor capacity and using smaller, lighter railing systems for the Z plane, we can significantly cut costs and enhance the performance of tap operations. This change would involve selecting a motor with lower torque and a more compact rail system that meets the specific demands of Z-axis movements. The reduction in weight and power requirements will not only lower expenses but also enable faster and more precise tapping actions, improving overall operational efficiency.
2. Driver Efficiency: At present, our command processing is sequential, which limits the efficiency of the plotter. The Arduino controller, however, is capable of handling parallel operations. By leveraging this capability, we can allow the X and Y plane motors to operate simultaneously, enabling the plotter to move to a specified coordinate more efficiently. This parallel processing can be implemented by modifying the firmware to execute coordinated movements, thereby reducing the time taken to position the plotter for each operation. This improvement will lead to faster plotting and increased throughput, optimizing the plotter's performance.
3. Enhanced Customization: Our current setup provides fixed plotting coordinates, which may not cater to the varied needs of testers. Enhancing the driver to support more comprehensive calibration will allow testers to define a larger workspace and segment it into finer units according to their specific requirements. This customization could be achieved by integrating a more advanced configuration tool that allows users to map their workspace and calibrate the plotter accordingly. Providing this level of customization will make the plotter more versatile and adaptable to different testing scenarios, thereby increasing its utility and user satisfaction.
4. Expanding Functionality to Include Swipe and Gesture Operations : So far, we have implemented only the tap function. With improvements to the Z plane and more efficient tap operations, it will become feasible to incorporate additional functionalities such as swipe and gesture operations. By optimizing the Z-axis movements for speed and precision, and developing algorithms to handle continuous and dynamic motions, the plotter can be upgraded to perform a wider range of actions. These enhancements will involve updating both the hardware and software components to support complex gestures, thus expanding the plotter's capabilities and making it suitable for more advanced and diverse applications.

References

[1]. https://github.com/mschoeffler/arduino-java-serial-communication/tree/master/example01_print_text_on_LCD_module
[2]. https://fazecast.github.io/jSerialComm/
[3]. https://howtomechatronics.com/tutorials/arduino/stepper-motors-and-arduino-the-ultimate-guide/#Example_code_%E2%80%93_Controlling_multiple_stepper_motors_with_AccelStepper_library
[4]. https://www.amazon.com/gp/product/B00D7CWSCG/ref=ppx_od_dt_b_asin_title_s01?ie=UTF8&psc=1
[5]. https://www.amazon.com/gp/product/B06XJKVLG3/ref=ppx_od_dt_b_asin_title_s00?ie=UTF8&psc=1
[6]. https://www.amazon.com/gp/product/B00PNEQI7W/ref=ppx_yo_dt_b_search_asin_title?ie=UTF8&psc=1
[7]. https://www.amazon.com/gp/product/B071W7XTFN/ref=ppx_yo_dt_b_search_asin_title?ie=UTF8&psc=1
[8]. https://dummyscodes.blogspot.com/2014/08/using-java-rxtx-library-for-serial.html