International Journal of Control Theory and Applications

ISSN : 0974-5572

© **International Science Press**

Volume 10 • Number 31 • 2017

Multi-Touch Testing Robot

Prateek Verma[a] Dushyant Singh Chauhan[a] Rohan Ramaswamy[a] and C. Likith Kumar[b]

[a]*B.Tech/ Electronics and Instrumentation Engineering, SRM University, Kattankulathur, Chennai, Tamil Nadu, India*

E-mail: prateekverma_manish@srmuniv.edu.in, dushyant95chauhan@gmail.com, rohan.1995@gmail.com

[b]*Assistant Professor/ Electronics and Instrumentation Engineering, SRM University Kattankulathur, Tamil Nadu*

E-mail: likithkumar.c@ktr.srmuniv.ac.in

Abstract: Every smart device today consists of touch-based interface for control. Testing of such touch based interfaces has hence become very important to ensure the quality of manufactured devices, as well as to debug hardware and software during their development. The Cartesian robot will help automate physical testing of touch-screen devices. The robot is designed to operate on devices such as touchscreen devices such as mobile phones and tablet devices commonly available in the market running android OS and will operate a touchscreen device using an end effector and will be able to simulate actions such as single taps, multiple taps, swiping, pinching, etc. The system utilizes the android debugging bridge (ADB) to capture touches as well as detect various touch parameters such as touch pressure and duration. Also, it captures various actions such as the element touched on (*e.g*. App icon, button, etc.). The device hence will be able to execute various touch screen test routines and able to test an app with various combinations and permutations of configurations and order of actions. Hence it will be able to save time for testing R&D prototypes and ensure quality control of manufactured devices. It will also allow app development companies to test their software on various devices at once, ensuring compatibility and enabling companies to quickly find and diagnose bugs, if any. It would be a breakthrough for Robotic technology in Test Automation, as the proposed design is highly economical and reliable.

Keywords: *Gantry robot, test automation, touchscreen device*

1. INTRODUCTION

With the increase in touch-screen demand across various devices, it has become essential for touchscreen gadget manufacturers to test touch screen functionality for each and every unit. Automated touchscreen testing helps Original Equipment Manufacturers (OEMs) to test the overall product performance for tablets, mobile phones, etc. and on top of it, product manufacturers can use it for new product development of hardware such as graphics card, processor, etc. In this paper we describe our solution, a gantry robot system which allows small to medium sized manufacturers and software development companies to access to easy to use and scale automated testing, which typically is only affordable for very large companies, which would result shortening the “time to market”; remote testing, diagnosis and capture all test results and diagnostics; create benchmark and gain competitive edge over superior product as well as test new kind of control and activation methods.

2. T-BOT CONCEPT

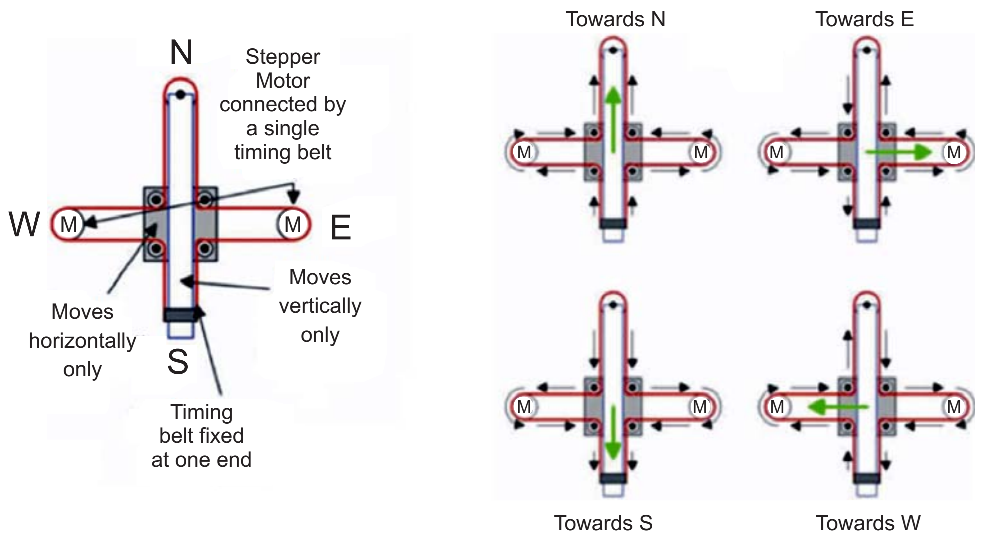
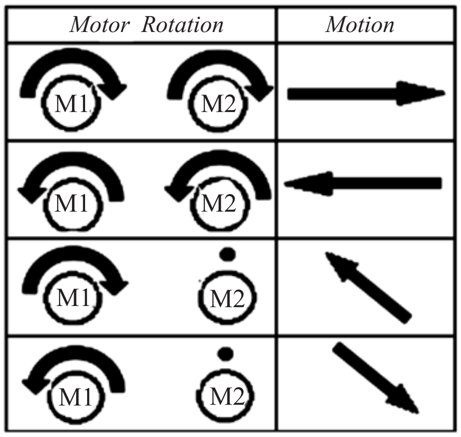


Figure 1: Gantry Motion

The T-bot, so-named because it resembles the letter T is a 2-D robot, a planar mechanism for positioning an object in XY space, such as a plane, finds use in many industrial applications, such as pick-and-place, sorting, gluing, and inspection systems. It is easy to manufacture because it comprises two motors, a timing belt, and two perpendicularly mounted rails.

As in any coordinated-motion system, the computation of the position command to each motor of the T-Bot is just as important as the control scheme you employ to control the robot. The successful combination of these two aspects will lead to accurate positioning, but that concept means different things depending on the application.

Table 1
Stepper Motor Direction



Motor Rotation		*Motion*
↻ M1	↻ M2	→
↺ M1	↺ M2	←
↻ M1	● M2	↖
↺ M1	● M2	↘

3. PROCESS CONTROL SYSTEM

Motion applications typically use a cascade control system, as shown in the figure, that comprises position, velocity, and current loops, all typically proportional integral. The gantry controller has a predefined instruction set for positioning the Touch actuator to the particular coordinate defined by the PC as G-Code over UART. This G-Code is modified according to the feedback from the Touchscreen device through ADB interface.

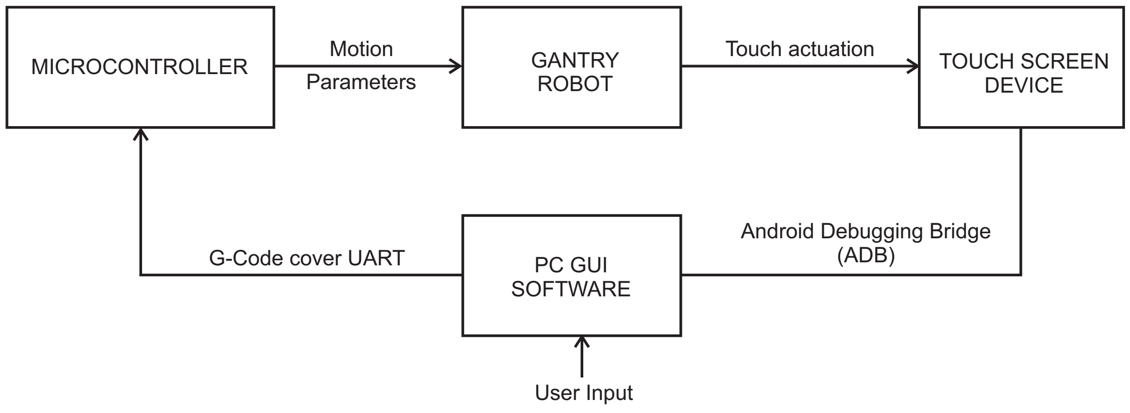


Figure 2 Process Control

4. TOOLS OF ROBOT

1. **Body:** The chassis of the robot is made of Smooth as well as threaded metal rods, Belt drive, bearings, and 3D printed parts.
    - a) **Metal Rods:** Two pairs of 8mm smooth stainless steel rods are used for the rails of dimensions 320x400mm and a pair of 10mm threaded rods are used for support in the base of length 320mm.
    - b) **Belt Drive:** GT2 belt drive setup is used.
    - c) **Bearings:** LM8UU 8mm standard linear bearings are used for the linear motion and F623ZZ flange type for belt drive motion.
    - d) **3D printed parts:** The parts are being made of PLA material and are used for holding together the body of the robot.
2. **Stepper Motor:** Two NEMA 17 motors are used for driving the GT2 belt drive system and positioning of the gantry robot in XY space.
3. **Motor Driver:** Two Pololu A4988 stepper motor drivers are used for each motor.
4. **Microcontroller:** Arduino Mega 2560 is used as gantry controller.
5. **Touch Actuator:** A servo motor coupled with a stylus is used as touch interface between the robot and touchscreen device.

5. ANDROID TOUCHSCREEN FEEDBACK

For detecting the touches, multiple methods were considered. The first option was to create an app which would output the touch coordinates to Android's log cat. This information would then be read by the software running on the computer to determine if the required action was performed successfully- or not. This was ruled out as a viable option as, while testing, it was found that touch response callbacks are not reliably called while the app is not in the foreground - as the touch response callbacks are processed at a high level - allowing the operating system to give foreground applications higher priority - this caused significant latencies between touches and the logical output and on lower end devices the OS terminated the application to free RAM space. Other options were considered such as using screen capture programs to track the touch pointers - this option was quickly ruled out due to performance considerations. The solution which was finally used was to access the touchscreen device as a HID device. This is made possible by the shell of the Android debugging bridge - which allows utilization of the get event utility - which allows access to all the various hardware components in a mobile device - including the touchscreen.

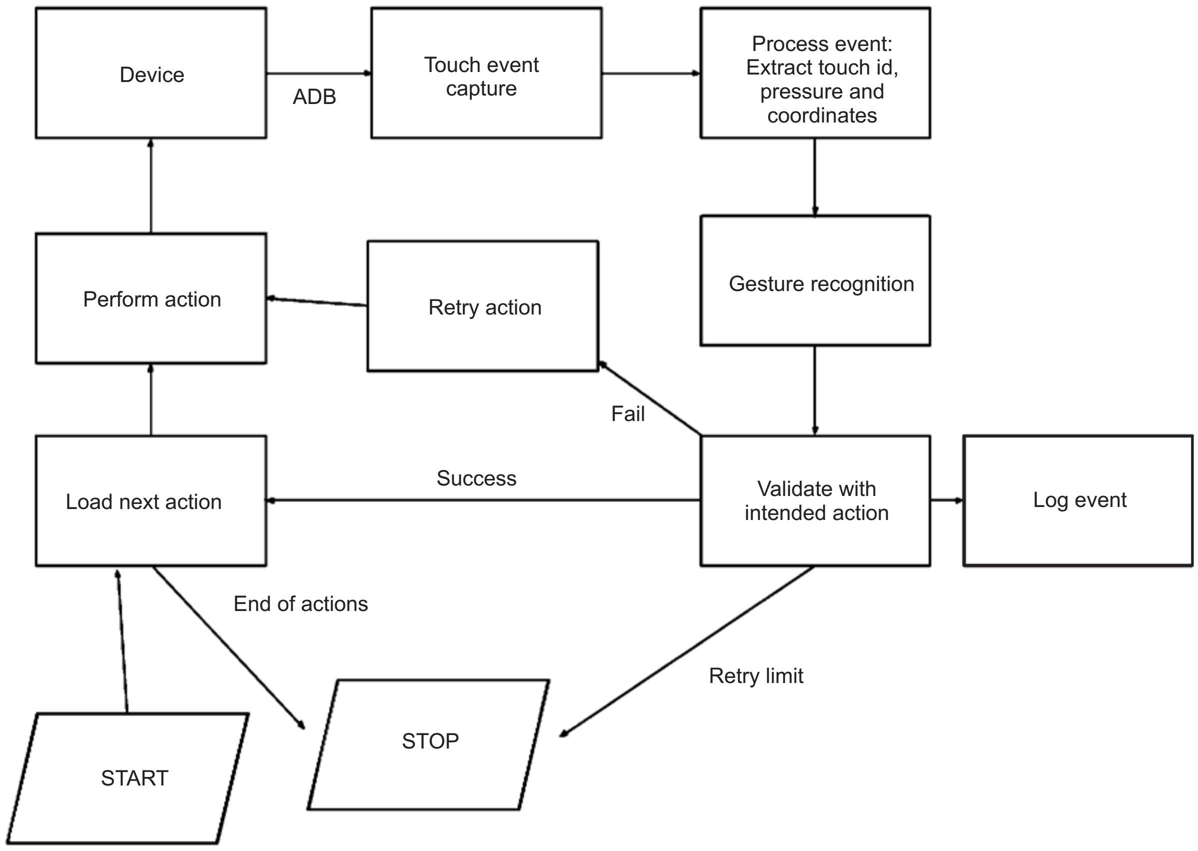


Figure 3: Flow Chart

The get event tool runs on the device and provides information about input devices and a live dump of kernel input events. This tool is typically used for ensuring device drivers are reporting the expected set of capabilities for each input device and are generating the desired stream of input events[3].

The command ADB shell su -- get even -lt /dev/input/event1 corresponds to the touch events generated by the touchscreen device on most android devices.

The values for ABS_MT_TRACKING_ID, ABS_MT_POSITION_X, A ABS_MT_POSITION_Y and the time are read by the PC based controller software, where the position_x and position_y are the absolute coordinates of where the touchscreen is touched, and in the case of multiple touches simultaneously, different tracking IDs are given to each touch point. All values are in hexadecimal and are converted into a decimal by the PC Software. The time printed is in the format <seconds>.<microseconds>.

6. PC BASED CONTROLLER SOFTWARE

The PC based controller software is a GUI application built using Python using QtPy as the GUI framework. It communicates with the android debugging bridge using a piped input from the ADB shell. It commands the gantry robot via the serial communication port at 19200 baud rate, using a subset of the g-code, also known as *RS-274* programming language [5].

The process starts by defining an action in G-code and the response that should be expected in terms of gesture or a log cat output. The action is performed on the start button is pressed in the GUI, as the program transmits the G-code to the robot controller. The python program interprets the touch coordinates from the device using the ADB as described earlier. It then detects the gesture based on the state flow diagram given below(fig. 4)[4]:

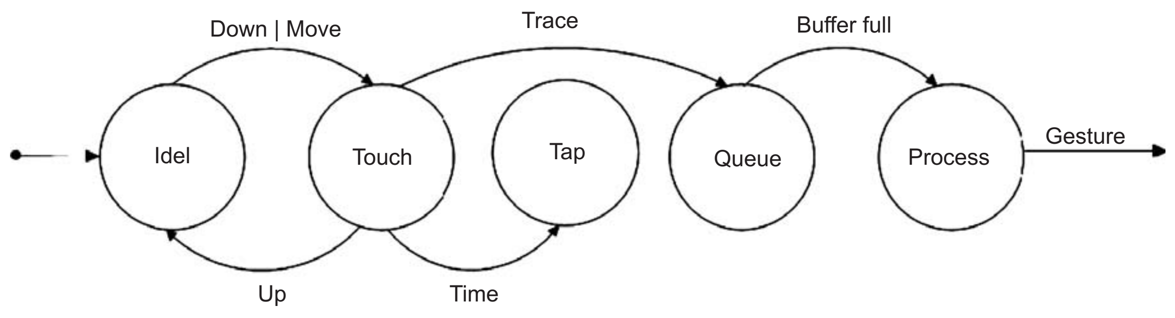


Figure 4: State Space Diagram

7. CONCLUSION

To conclude, we proposed, designed and constructed a working prototype of a system capable of allowing users to easily define complex testing processes for testing either their hardware or software and perform the same in a highly scalable manner. Our system will allow the user to quickly debug, diagnose and find solutions to the hardware/software issues due to the provision of the extensive logs and screenshot system. The implementation of this system is designed from the ground up to be low cost, allowing access to automated testing tools to small software development companies and hardware manufacturers and distributors. The system can also be upgraded to allow multiple end effectors allowing for an even higher range of motions such as pinches, multi-finger swipes, etc.

1. **Single tap:** When a touchdown event and a touch-up event occurs within 300 milliseconds of each other, it is recognized as a single tap.
2. **Double tap:** When two single tap events occur within 300 milliseconds of each other, it is detected as a double tap.
3. **Swipe:** When a touch point is traced across the screen, it is detected as a swipe. The starting and ending points are detected and the velocity vector is calculated to provide the swipe “strength” and direction. The strength is typically used for applications such as kinetic scrolling.

Once the gesture is detected, it is then validated against the conditions provided by the user. If the action is successfully validated, the program proceeds to the next action. If the action fails, it retries the action until the specified retry limit is reached.

All the above actions, its responses as well as the android’s log cat output are captured and saved. Additionally, the system will also take the device screenshot as required by the user program.

REFERENCES

[1] Aidan F. Browne, Wesley B. Williams, Keith Loftus and Cameron Nye, “Implementation of a Cartesian Robot for Remote Mission Critical Operator Training”- ©2016 IEEE.

[2] Mahir Abdelwahid Ibrahim Ismail, Mohammed Khalafalla Mohammed, “Gantry Robot Kinematic Analysis User Interface Based on Visual Basic and MATLAB” - Volume 4 Issue 2, February 2015 www.ijsr.net

[3] https://source.android.com/devices/input/getevent.html

[4] http://www.edn.com/electronics-blogs/mechatronics-in-design/4368079/So-you-want-to-build-an-H-bot-

[5] The NIST RS274NGC Interpreter - Version 3, Thomas R. Kramer Frederick M. Proctor Elena Messina - NISTIR 6556 August 17, 2000

[6] Multi-Touch Gesture Recognition using Feature Extraction - Francisco R. Ortega, Naphtali Rishe, Armando Barreto, Fatemeh Abyarjoo, Malek Adjouadi