Automated Device Testing for Windows Phone 7

Demonstration Application Manual

Axelerate Solutions, inc

Abstract

The purpose of this document is to walk the reader through the necessary steps to launch the current proof-of-concept demonstration application. This includes launching the single-device and two-device demonstrations. No technical background is required apart from this document.

Introduction

The proof-of-concept demonstration application implements several key features for automated device testing, including GUI component detection, three styles of user input simulation, testing system configuration and more. There also exists a two-device implementation of this automated testing system. This document will walk you through the necessary steps to install, deploy, configure, and execute this demonstration application for any given Windows Phone 7 device.

Dependencies

Hardware

You will need physical access to the Adept Cobra S350 Robotic Arm (found in lab-room G, number 1501) and the attached encasing. The robotic platform is the entire case, which includes a protective see-through box around the top of the robot, and a cabinet that includes all of the necessary electronics and computer equipment. The monitor, mouse, and keyboard found next to the robot are all connected to the computer within this platform. No other computers are connected to the robot. It is not recommended to remote-access into the computer for safety reasons; you should always be able to see what the robot is doing to prevent breaking devices. Several safety components are also included with this platform, as discussed below.

The robotic arm is connected through a proprietary cable to a control box, which houses all the necessary electronics to control the robot. Connected to this box is the controller computer, which is a 32-bit Windows 7 machine. This computer runs the demonstration application software, and currently has it installed on the default user's desktop.

The surface where the phone is mounted is a black metal block that is spring loaded to handle push-events from the robot. On this black block, there are four mounting points per device, with the ability to mount two devices at once. Note that these mounting points can be changed by using either a small wrench or a hex key.

Software

The controller computer, found within the robotic platform, only has one account and is not connected to corp-net. The default account user name is ***Adept*** and the password is ***ms-adept1***. This information is found physically printed on the front of the computer within the robot's lower cabinet. In general, it is recommended to leave all previously opened windows alone, as several groups and users are using the system to test with each day. The computer does not have a lock-out time, and is usually always left unlocked.

The computer runs a 32-bit Windows 7 installation, with Visual Studio 2010 and the Adept Desktop 4.2 client. AForge.net is also installed, which is a visions library required to run the demonstration software. AForge.net can be downloaded and installed using the installer client, found on their official open-source website. Please note that this library was used only for the demonstration proof-of-concept, and should not be used in the final system.

The demonstration software can be found on the desktop of the controller when logged-in into the default user's account. The first directory "AutoTest_Demo" contains the latest release of the single-phone demonstration code. The second directory "AutoTest_Backup" is a simpler version that has been proven to be more stable, but lacks several features. The third directory "AutoTest_MultiDevice" is the same as the demonstration application, but supports two devices and can run two scripts independently between each device. Preferably use the "AutoTest_Demo" for single-phone demos, and "AutoTest_MultiDevice" for multi-phone demos. All source code is held within each of these directories.

Robot Initialization & Notes

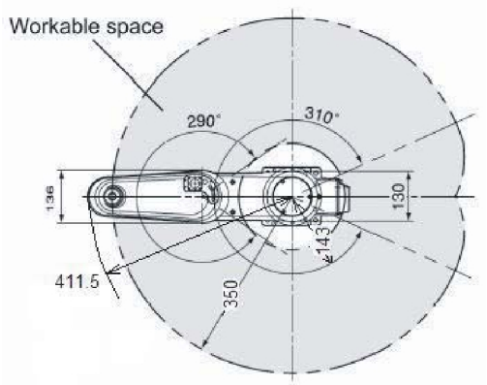
Initialization

The robot platform has three E-STOPs (Emergency-Stop – press any of these buttons in case of an emergency) that must be in the ready position and a key in the correct lock position to have the robot ready for use. The first E-STOP is the door-safety, which is only in the ready position when the doors of the robot platform are firmly closed. The second E-STOP is a red button on the front of the bottom cabinet. Make sure that the button is not pushed-in, and if it is, you can release it by rotating it. The third and final E-STOP is the red button found on the remote controller device, usually placed next to the controller computer's monitor. Again, make sure that the button is not pushed-in, and if it is, you can release it by rotating it. Once the doors are closed and all E-STOPs are in the ready state (released), make sure the controller key, found on the bottom cabinet doors, is rotated towards the right, pointing to a computer icon. At this point, the robot is ready for use but is not yet active.

To activate the robot, there are two methods: press the "on" button found on the top-left of the remote that looks similar to a vertical straight line, above another button that has a circle on it or press the "3. Connect Robot" button in the demonstration software. Once either of these are done, the

platform will make a high-pitch “beep” noise making you aware you are turning it on. You have five to ten seconds to press a flashing white button on the front of the platform before the robot ignores your command and de-activates itself.

The robot has a wide range of motions and positions, but is safely encased within the robotic platform. Please see the below image for the range of positions the robot can achieve.



Range of positions the robot can achieve.

The robot at this point is both ready and active, and can be given commands to move around. Please review the V+ programming guide and Adept DeskTop documentation to learn how to send manual commands via the Adept DeskTop 4.2 client or by the remote-control box.

Robot Probe

To mimic human input events, and to record the device screen, a special probe was built for the robot arm. The probe has a metal cap at the bottom that has a thin layer of conducting materials that mimics human skin. The surface is not abrasive, and will not scratch the screen. The probe also has two cameras built-in: an endoscope to better see pixel-level information and a large HD web-cam, to see the screen as a whole. The probe also has a spring mechanism to prevent breaking any device’s screen when applying too much pressure. The surface where the phone is mounted also has a spring system to dampen high pressure events (the robot pushes too far into the phone), but will automatically shut off the robotic arm if it is triggered.

Coordinate System

The coordinate system of the robot is based on three dimensions defined by three axis: X, Y, and Z. If you face the robot directly from the front, the X+ direction is towards you, Y+ is to your right, and Z+ is up. There is a sticker in the back-left of the robotic platform to show you these axis directions and a sticker on the base of the robotic arm itself showing the directions of the Z axis.

Remote Control Robotic Arm

It is possible, and required, to move the robot manually when setting your configuration variables (as discussed below). Using the remote controller box (a large remote-controller box that has several buttons on the left and right and a large digital screen in between), you can manually move the robot and its probe at multiple locations. For example, when defining the best camera position for single-device configuration, you must move it manually first using the remote-controller then save the position in the demonstration application.

To go into manual control mode, the robot must be active, and the mode set within the controller box must be that of "World". This can be set by cycling through the right-screen options by pressing the button "Mode" which is found on the left of the device. When releasing manual control back to the computer, which must be explicitly done, you must cycle back to "Comp." for computer control.

Once in manual control, you can move the robotic probe location based on X, Y, Z locations by changing their values using the +(plus) and –(minus) buttons found on the right of the screen. You may have to cycle through the screens (by pressing the "Disp." button on the left) to see the values of X, Y, and Z locations.

Configuration

Hardware

You must first place the phone into the platform such that the phone's screen is laid flat, in parallel with the mounting platform. You can do this by tightly screwing it in position using rubber padding and a digital balance kept next to the robot. Make sure to measure both pitch (X-axis) and roll (Y-axis) of the surface so that no single corner is higher than another.

The phone should be left on, unlocked, and have the screen time-out set to none. This can be done by going to the settings menu in Windows Phone 7 (right of the main screen), select "lock & wallpaper", then select from the drop-down box titled "screen time-out" to "never". You can also turn off the password on this options screen. The device theme can be set to either "light" or "dark", though "light" is better supported because of certain properties of the visions algorithm. You may do this in the settings screen, by selecting "theme" then selecting "light" from the drop-down menu titled "Background".

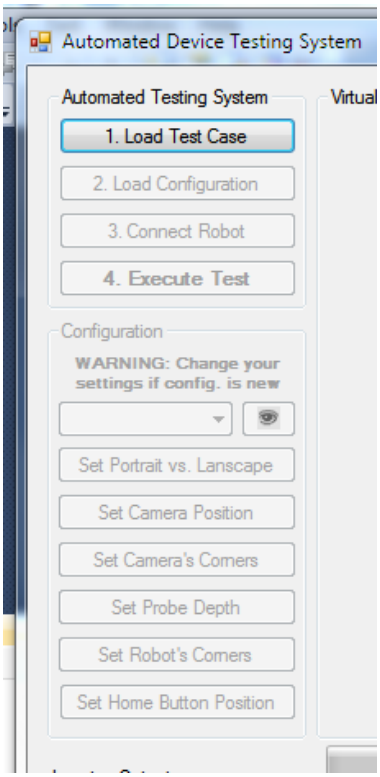
The phone does not have to be connected via cable to the computer, though it can help by preventing the battery from being drained over time. No cable-based features exist in the current demonstration applications.

A large sheet of cardboard may be found above or near the platform. It is critically important to place this cardboard above the robotic arm, and shield as much light from the florescent tubes found directly above from reflecting onto the device's screen.

Once the phone is correctly configured and placed into position, close all of the robotic arm's platform doors and launch the software.

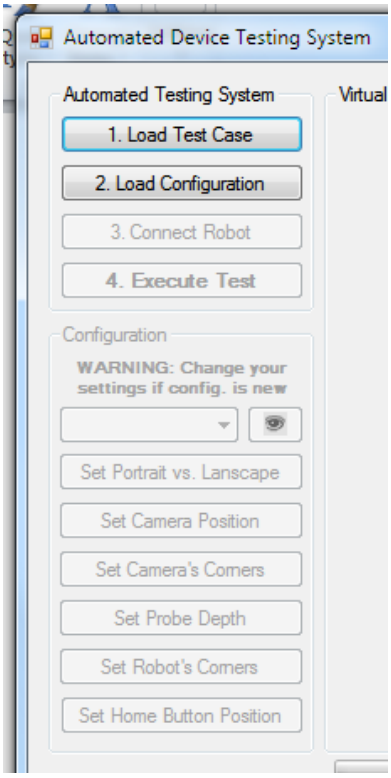
Software

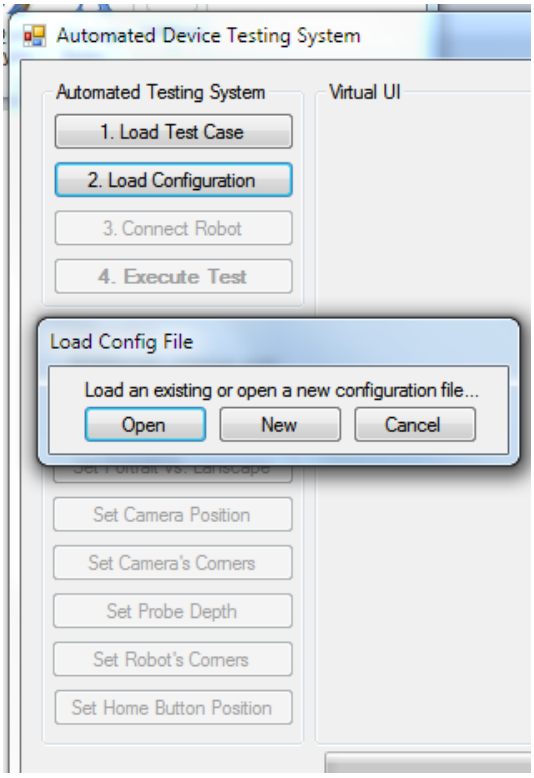
The software project is a source-code solution, and not an executable. You will have to build the solution and execute the debug version for each demo (though this binary could be saved and executed as a stand-alone application if placed in the right directory). The solution is titled "AutoTest_Demo" for the single-device demonstration, "AutoTest_Backup" or "AutoTest_MultiDevice" for different demonstration types as discussed above. Open the solution with Visual Studio 2010 and press the "Run as Debug" button (a green triangle) found on the top of the window. After this, the application should be launched and you will see a plain window with most UI forms being disabled, except for one button:



Click on the "1. Load Test Case" button to load a test case file. These files end in "*.tpf" (Testing Procedure File) and describe the actions the robot should do, and the graphical user interface elements it expects to see after each movement. The file format is discussed in more detail in the Design Report document. There are several testing scripts available ready for use. These are as follows:

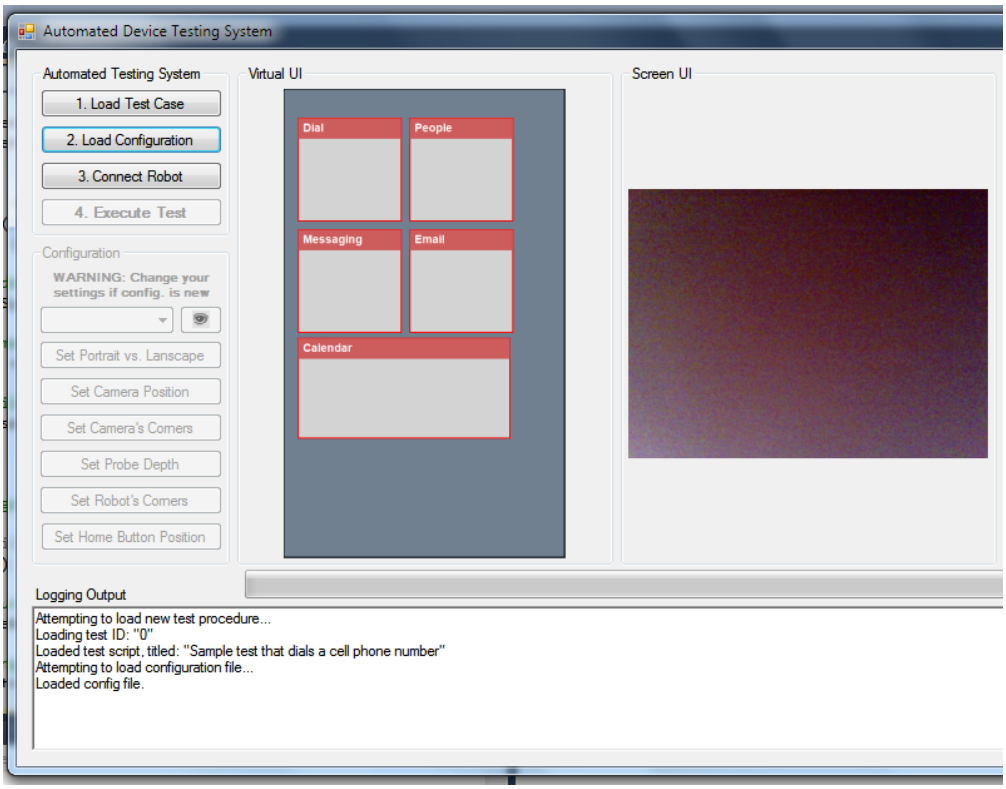
- *0 - Call number.tpf:* Call a given phone number, as defined in the script; does not require a SIM chip, and actually expects the application to fail placing the phone call
- *1 - Flick Sample.tpf:* Flick through the calendar to demonstrate the flicking ability
- *2 - Call contact.tpf:* Call a given contact, in this case, a contact named Jeremy Bridon who is the first member of the user's contacts list



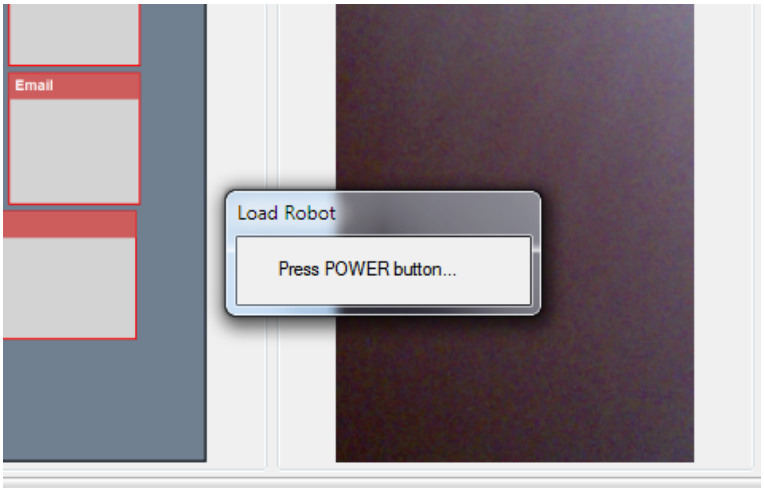


Click on "2. Load Configuration" button to load the configuration file. These files are specific to each phone you are testing with and end in "*.tcf" (Testing Configuration File). If you are using a new phone, or have changed the positioning of the phone, it is higly recommended you either replace the existing configuration file or create a new configuration file. These configuration file formats is discussed in the Design Report document. Note that actual change of configuration variables is discussed in the next section.

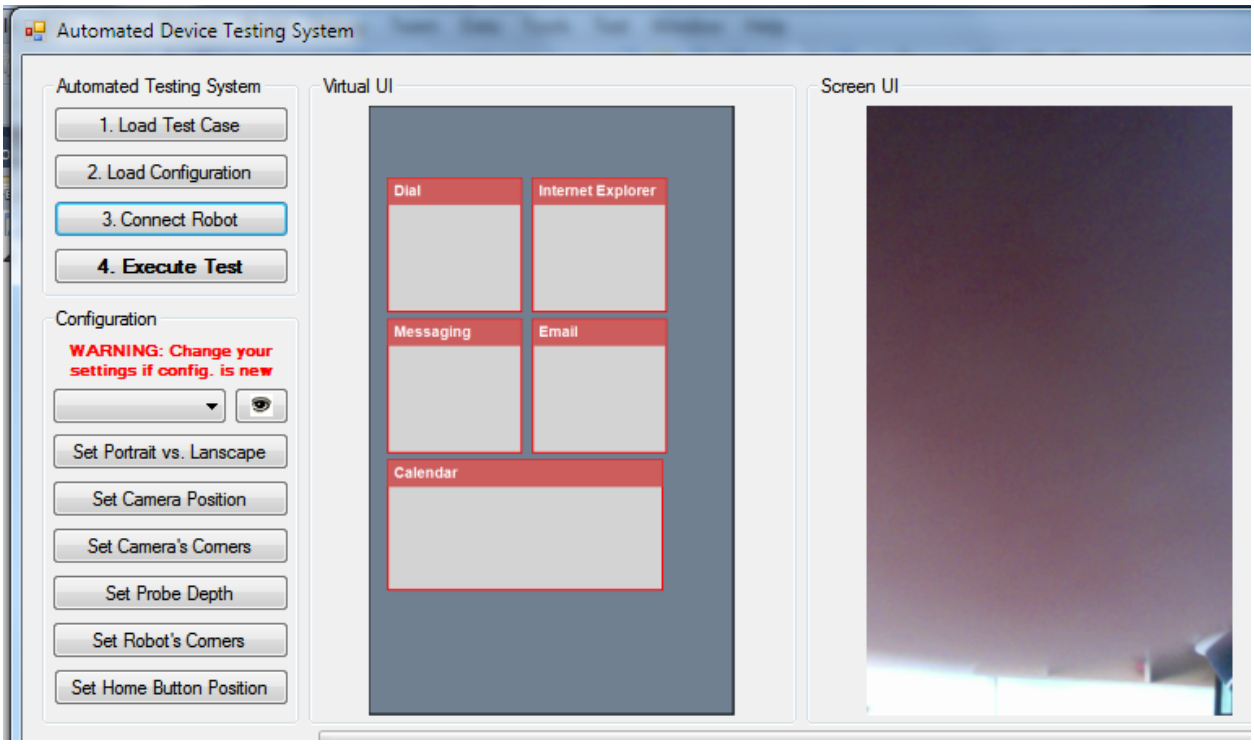
Please note that the description of the testing configuration script file format is found in the design documentation, and not in this manual.



Once the test case and configuration files have been loaded, you will now have the application present the default screen we are expecting from the device in the left box titled “Virtual UI” and what the HD camera is seeing in the middle-box titled “Screen UI”.



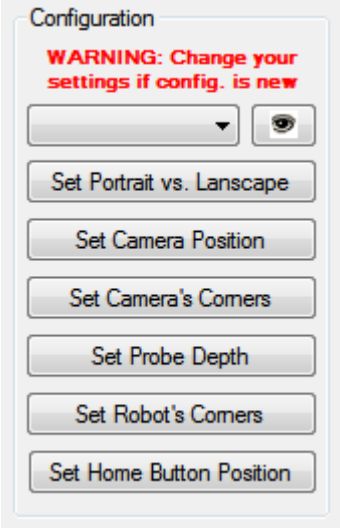
“3. Connect Robot” button is enabled; press it to connect to the robot. Make sure you press the flashing white power button if the robot is not yet active. If you fail to do so, the software will not be able to connect correctly to the software-serial controller of the robot. If there is a failure in communication initialization, you may need to restart this process.



At this point, the application is ready to execute the test case script, but you should configure the robot as best as possible before pressing the "4. Execute Test" button. This is discussed in the next section.

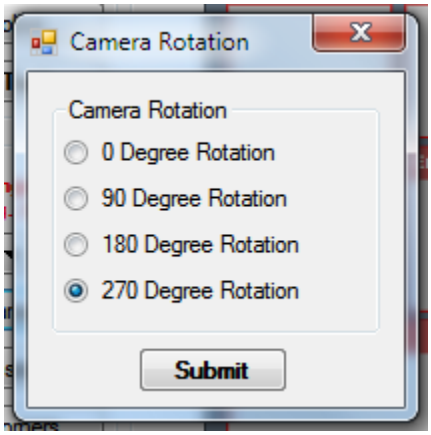
Software & Hardware Configuration

Configuring the robot to work correctly with the target device can be tedious and take about ten minutes when changing all configuration variables. There are several aspects of configuration, which includes selecting the correct camera handle, portrait vs. landscape orientation, camera's expected corner location, robot's expected corner locations, probe's depth, and home button location.

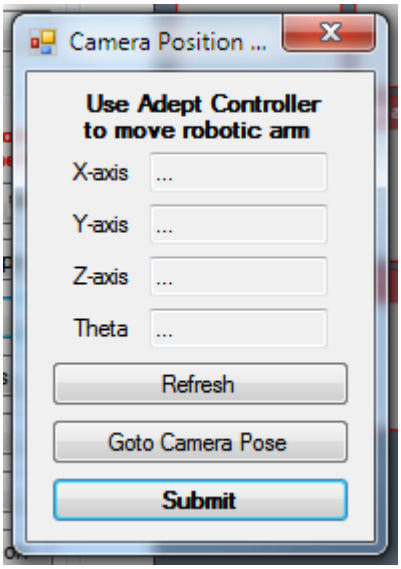


If this is the very first time you are running this application for a new phone, you *must* reconfigure all variables. The first is a drop-down box of all camera devices. You must select the best camera device that is able to take a picture of the entire device screen. If you click on the “eye icon” button, you will be able to preview the camera. It is known to fail, so you may have to restart the application if you press two times.

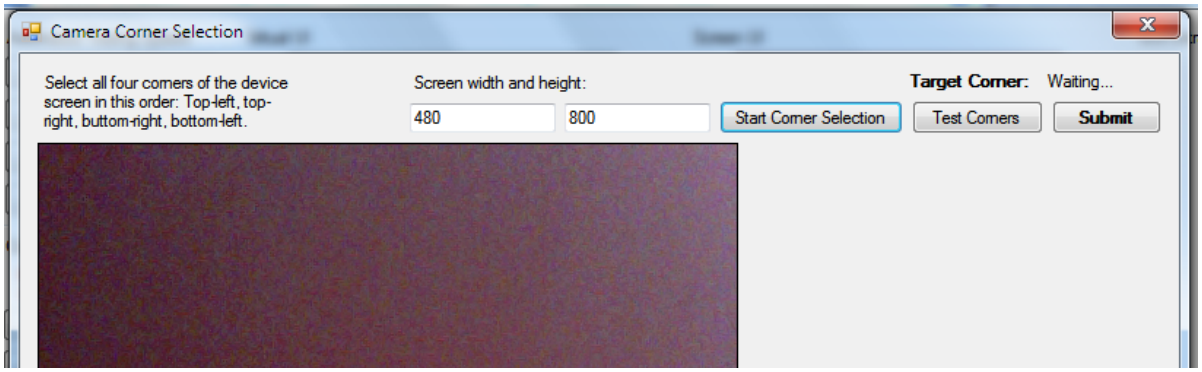
The second “Set Portrait vs. Landscape” configuration variable tells which orientation the camera should output to. Sometimes, you can see more if you tell the camera to orient-itself at 270 degrees, because the camera will see the device screen length-wise, thus recording more screen pixels.



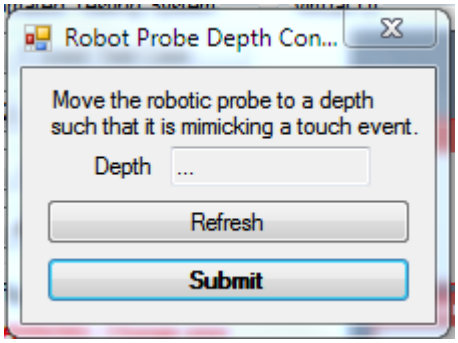
Next, you may consider resetting the camera’s position and orientation. This is done by clicking on the “Set Camera Position” button, and manually moving the camera by the remote controller. The remote controller usage is explained in detail above in the section titled “Remote Control Robotic Arm”. Make sure that you rotate the robot to best align with the device screen. Once this is done, simply press the “Submit” button.



Next, you must define where the corners of the device's screens are. You can do this by moving the robotic arm back into the camera position, then pressing the "Set Camera's Corners". A window will appear which asks for important information: the size (in pixels) of the target device's screen and the screen's corners. You may change the size of the screen by changing the values of the text box, and you may change the corner locations by pressing the "Start Corner Selection". When this button is pressed, it will ask you to press each of the four corners of the device screen. *You must* click each corner in the image preview as closely as possible to the corner, preferably within the screen, but in the following order: top-left, top-right, bottom-right, bottom-left. The ordering is critical. Once this is done, you may click the "Submit" button. The button "Test Corners" will move the robot to the positions you defined, as well as the positions the system thinks is the physically-defined positions (as discussed in later sections).



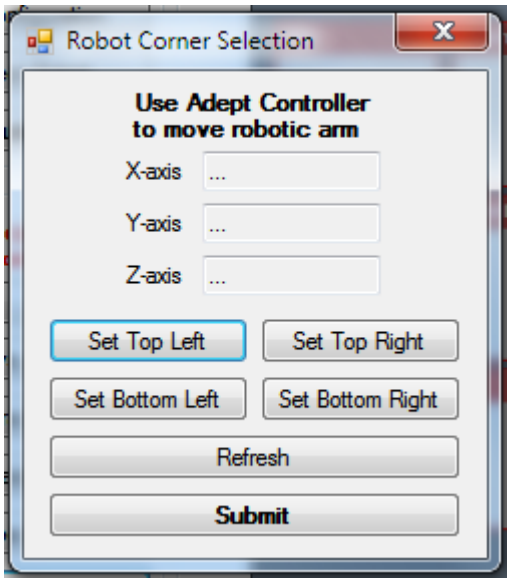
You must also define the depth the robot must reach to mimic a touch-event. Simply click the "Set Probe Depth" button, switch the controls of the robot to manual mode, move the robot over the center of the screen, then *move the depth of the probe to physically touch the screen*. Once this is done, press the "Submit" button to save this into the configuration file. Please note that the robot must physically be touching the screen, since the submitted depth will not be changed when applying one of the three pre-programmed motions.



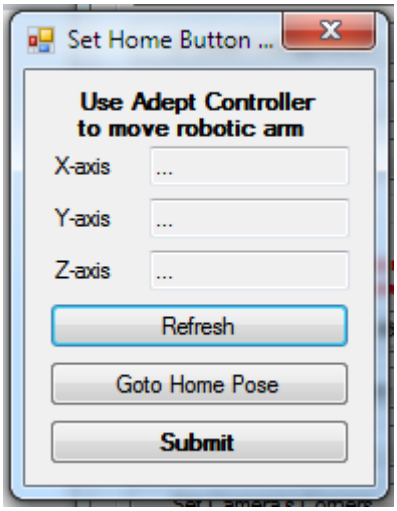
To tell the robot where the physical buttons are, you will have to manually move the probe above (depth does not matter) each of the four corners. Once places over one of the four corners,

simply press the appropriate “Set Top Left”, “Set Top Right”, “Set Bottom Left”, or “Set Bottom Right” button. As noted in the camera corner-selection window, you may verify that all four corners are correctly being detected and moved to by pressing the “Test Corners" button in the “Camera Corner Selection” button.

It can be helpful to launch the endoscope camera, with lights off, to better see the pixel-level information so that you can accurately place the probe right above the corner of the device’s screen.



Finally, the last configuration that must be set would be the location of the home button. This is usually found as a physical button with the Windows flag or wave logo on the bottom-center of the device. Simply move the device manually over this location, slightly above the actual button with enough space to not press too deeply, since the application will press about 2.5 to 5 millimeters down into the device.



Once all of these configuration variables are set, the application is ready to be used and

launched by simply pressing the “4. Execute Test” button at the middle-left of the application.

Running the Demo Application

When all configuration variables are set, and the robot is both enabled, powered, and set to be controlled by the computer, the application can be ran by pressing the “4. Execute Test” button. This will activate the testing cycle, which is a series of movements and virtual screens you are expecting and comparing with.

The main window has three image previews: the left-most (Virtual UI) is a preview of what the user-interface should look like after the current motion. The middle-window (Screen UI) is a live connection to the HD camera, which should have, when the robot is verifying the window, a full view of the device’s screen without the edges of the device being visible. The third right-most preview (Extracted UI) is the User Interface extraction algorithm output. There is also a text-box at the bottom of the window that contains logging and movement information for the current action.

The Virtual UI preview screen renders the output of the GUI extraction algorithm by placing several dozen yellow or green boxes on top of the HD-camera preview. The yellow boxes represent unlikely, but possible, UI components. Green boxes, with labels, represent very likely, but not guaranteed, UI components. Based on internal configuration, there may or may not be many false-positives. Based on the current virtual UI verification algorithm, based on the Design Report documentation, these false-positives are not at all hindering the verification process and may actually help. Note that when doing multi-device demonstrations, the sensitive of the algorithm is much higher and is done so because the device’s screens at the time of the development process, were giving high-levels of reflection.

Please note that all motions are also rendered on-screen onto the “Virtual UI” preview box. A red “X” shape is a touch/push event taking place at that location on-screen. Two circles and a line connecting them is a drag event. A line with a single circle representing a flick motion originating from that circle’s position.

The first action the robot will always take will be to reset the robot probe to the home screen. Make sure this does not fail, and if it does, simply reconfigure the camera home-button location and depth of the probe. This may be a common problem if the depth is not correctly set.

After the home screen is validated, the robot will continue walking through the procedure script, executing a motion, then verifying the user interface. There are three types of motions: push, drag, and flick. Push is a simple push-button event at a given location. A drag is a simple point-to-point drag event. A flick is similar to a drag, but the robot quickly raises up the probe, similar to a human finger flicking across the screen: a quick drag-then-lift motion.

After a motion is executed, the robotic arm resets itself to the camera-location, which is the best location for the camera to take a picture of the screen. Once the robot is correctly placed, the device screen is extracted and the verification algorithm is applied: the GUI components on-screen are

extracted and then compared to the expected virtual screen. This is explained in detail in the Design Report document.

If any of the procedure script steps fails, the test will stop and will re-enable all configuration and the “execute test” buttons. If the testing procedure continues without any failure and reaches the final step, a log message is printed about the success state, some relate information, and then enables all configuration and scripts buttons again. A progress bar, above the log text box, can be found showing how many steps are left for the current test procedure script.

Multi-Device Demonstration

Running the multi-device demonstration is similar to the standard demonstration. Instead of a single testing script and configuration file, there are now two of both, one testing script and one configuration for each device. Devices must be placed onto the robotic platform in the same method, with both devices laid as flat as possible side-by-side. When setting the configuration file, the user must select the appropriate radio button representing either one of the two configuration indices. Even if either device screens do not conform to the 480x800 pixel ratio, do not change these during the configuration process. This is a feature left undeveloped because of image and movement scaling issues.

Note that the format of multi-device testing scripts are the exact same, but both files must have the same number of steps. Several examples are included within the “AutoTest_MultiDevice” folder demonstrating sample scripts. These two scripts may be ran without any interaction (i.e. without calling from phone A to phone B) or have heavy interaction (call from phone A to phone B, and have phone B respond, and have phone A disconnect).

Conclusion

The above document details how to run the proof-of-concept demonstration manual. This code was written purely as a way of demonstrating the basic features of an automated device testing system for Windows Phone 7 devices. This application is by no means deployment-quality code, but is stable enough to be able to run as many testing instances as possible without restarting the application.

To correctly run the demonstration, you must resolve any missing dependencies for both hardware and software, as well as understand the safety procedures of the robot platform. Depending on the demo, you may have to run different executables, as well as different testing scripts and configuration files. If using any new phones, you must reconfigure all configuration variables to fit the device properties appropriately.