Constructing a UX Testing Platform using Embedded Computing Systems

Ariana Beeby Erik Steinmetz
Department of Math, Statistics, and Computer Science
Augsburg University
2211 Riverside Ave, Minneapolis, MN 55454
{beebya|steinmee}@augsburg.edu

Abstract

In this work we demonstrate how to create a platform for UI/UX experiments based on a Raspberry Pi computer configured with a touchscreen display.

User interface studies are often conducted by personal observation of a user engaging with a piece of software. Many kinds of experiments can be conducted without the need for human oversight at the time of the experiment. This work is intended to build a device to monitor user interactions in an unsupervised kiosk mode, providing a platform for these kinds of experiments.

The work presented in this paper explains the hardware configuration and software stack used to set up the kiosk. We present a detailed look at the software design, including which programs were chosen along with the configuration settings necessary for the software and hardware components to combine and create a versatile experimentation platform.

1 Introduction

The goal is to set up the Raspberry Pi as a kiosk allowing users unsupervised interactions with the computer in a controlled and limited environment. This environment enables UI/UX experiments to be conducted such as observing First Click placement and length of engagement. To allow entirely unsupervised user interactions, the machine will run software which automatically records and documents desired user events. It will also have a "default" display mode to which it returns after a set amount of time, reducing the effects of a previous user's interactions on future engagements. This eliminates the need for a human observer of the user, whose presence may interfere with the natural flow of the subject's interaction with the software. A machine may also be able to offer a more accurate and consistent record of the interactions than a human observer.

2 Background

User interface and experience testing can be broken down into a number of unique tests that focus on individual elements necessary for the user experience, including both performance and quality-focused tests. Some examples include task completion, clickstream analysis, and First Click testing [1][2]. With some of these tests, a moderator is required to have direct interaction with participants as they are guided through each task, requiring more time and resources to be put towards testing. The presence of a moderator could also have a undesired effect on the outcome of each test as well. Other tests only require observation of the participants and their interactions with the software. For First Click testing, the participant's first interaction with a given situation is recorded, with a desired outcome on the tester's part. This results in the tracking of correct and incorrect interactions based off the first click [1][2].

3 Development Process and Architecture

The software created for the kiosk setup of the Raspberry Pi is based on a tutorial guide outlined on the official Raspberry Pi website [3]. A bash script is created that blocks aspects of the desktop environment and cursor allowing for a clean display – kiosk mode – and opens a specified website on the Chromium browser. In order to run the data collection software created specifically for this kiosk, a command was added to the script which opens and runs a python script in the background. Completing the setup a service file is created that ensures the device will boot automatically in its kiosk mode by executing the bash script on power-up.

The focus of the data collection in the system described in this paper is purely to get the locations and times of the interactions with the kiosk system. Unlike First Click testing, data collection will continue after the first interaction, allowing for analysis of the overall experience, including engagement time.

For the creation of the data collection software, a library that could monitor input events from the screen on the site was required. A number of libraries were explored, including xdotool, a shell command that is used in the kiosk bash script. When initially implemented, xdotool's tracking of the mouse-click behavior did not detect any touches, but the mouse-enter behavior would detect when the cursor would briefly appear on screen on touch. This resulted in the logging of two duplicate events, as the appearing and disappearing of the cursor on touch counted as separate mouse-enter behaviors.

Ultimately, the Pynput [4] and logging Python libraries were utilized. Pynput is a high-level library that allows for the simulation and monitoring of keyboard and mouse events. Though not extensive in its functionality, pynput fits the needs of the data-collection this device requires, and has streamlined usability. Similar to xdotool's mouse-click behavior, pynput's click detection does not read a touch on the screen as a click. However, as the cursor only appears on touch, the event is detected as a movement of the mouse. Unlike xdotool, a touch event is only detected once and at the moment the initial contact is made, or the cursor is first moved. This eliminates the logging of redundant information. The Python logging library was used to record these events and the specific time of the events to a logging file. This file can be written and read while the monitoring program is active, allowing the device to remain in kiosk mode while observations are being made.

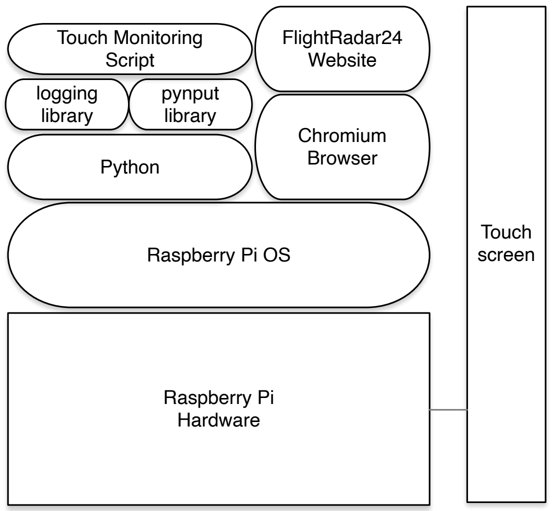

Figure 1: The Hardware-Software Stack

The overall architecture of the system is shown in Figure 1, including the Raspberry Pi hardware and the various software components as outlined above.

The current version of the kiosk runs on a seven inch touchscreen with the computer mounted on the back, so it can act as a portable experimentation system. This setup is seen in Figure 2.

4 Results

When powered up, the kiosk boots and after about thirty seconds ends up in its default state, showing the `flightradar24.com` website that the user should interact with, as shown in Figure 3.

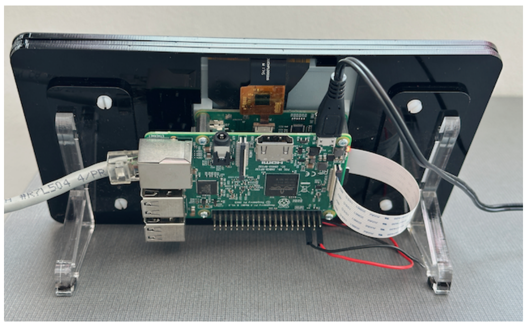
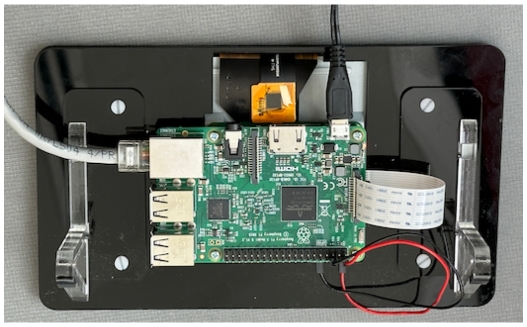

Figure 2: Kiosk Hardware with Touchscreen and Computer

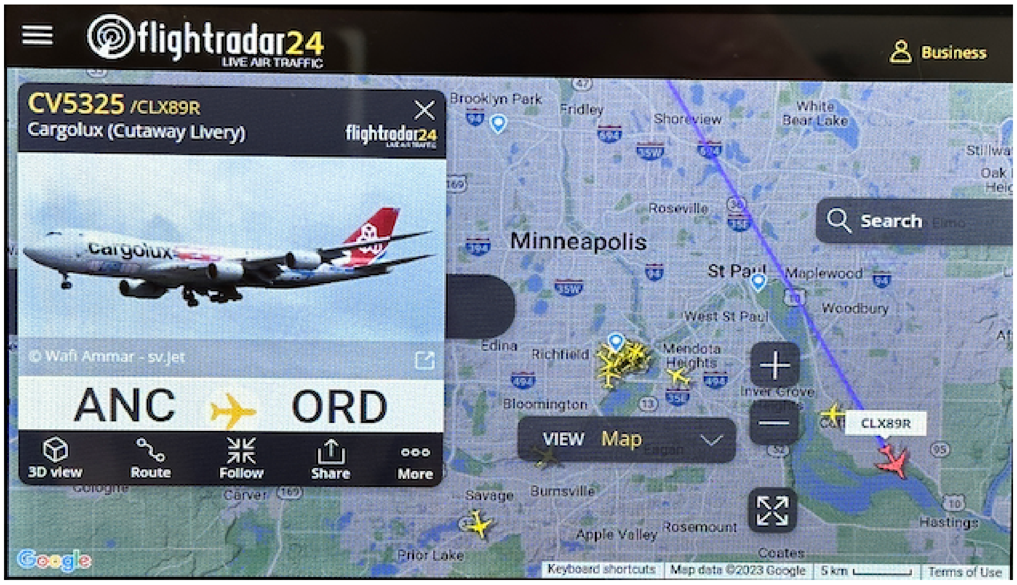

Figure 3: Flight Radar Screenshot

As the user interacts with the kiosk, each of their taps is recorded in a log for examination and analysis at a later time by the user interaction experimenter. The log tracks the time of the tap, as well as the location on the screen using the corresponding x and y coordinates. An example log file is shown in Figure 4.

The user can customize the kiosk.sh file to specify the desired site to start the kiosk in. They can also specify in the Python script the file path and desired name for their log. For the user's experimentation, the device can be left in a location where participants can interact with the kiosk. Each interaction, or touch, will be logged in the file, which can be accessed remotely by the user through an ssh connection with the device. As changes are made to the site for each test, or different sites are specified in kiosk.sh, all that is required of the user is to stop and restart kiosk.service, or restart the device,and specify a separate file in data-collection script to track the new interactions.

```
2023-03-17 14:27:14,108  move to (539, 248)
2023-03-17 14:27:15,118  move to (512, 245)
2023-03-17 14:27:16,216  move to (571, 229)
2023-03-17 14:28:13,617  move to (588, 304)
2023-03-17 14:28:14,545  move to (600, 309)
2023-03-17 14:28:15,505  move to (565, 362)
2023-03-17 14:28:16,489  move to (554, 432)
2023-03-17 14:28:17,425  move to (590, 354)
2023-03-17 14:28:18,869  move to (602, 313)
2023-03-17 14:28:19,417  move to (573, 355)
```

Figure 4: Sample Log Entries

5 Conclusions and Future Work

This first build is a success as it creates a clean kiosk display on a Raspberry Pi device, and monitors and records touch events on the device while in the kiosk mode. However, the touch event data is rudimentary, only recording the x and y coordinates of the touch. This data requires more work on the researches part to mirror the coordinates over the website display in order to see what elements were interacted with. It also does not account for touches that would result in the leaving of the page, which could result in data that is not applicable to the desired testing.

In order to more accurately monitor events, future versions of this kiosk will continue to use the Raspberry Pi, but attached to a larger screen. The data-collection software will also be expanded upon, allowing for finer-grained control. In order to expand the software, the exploration of a different, lower-level recording and monitoring library will be required.

References

[1] Bailey, R. W., Wolfson, C. A., Nall, J., and Koyani, S. Performance-based usability testing: Metrics that have the greatest impact for improving a system's usability. In *Human Centered Design* (Berlin, Heidelberg, 2009), M. Kurosu, Ed., Springer Berlin Heidelberg, pp. 3–12.

[2] Dumas, J. S., and Fox, J. E. Usability testing. In *The Human-Computer Interaction Handbook, Third Edition*. CRC Press, 2012, pp. 1222–1241.

[3] Foundation, R. P. How to use a raspberry pi in kiosk mode. `https://www.raspberrypi.com/tutorials/how-to-use-a-raspberry-pi-in-kiosk-mode/`. Accessed in March 2023.

[4] Palmer, M. pynput 1.7.6 library. `https://pypi.org/project/pynput/`, 2022. Accessed in March 2023.