ПРОГРАМИРАНЕ ЗА ТАКТИЛНИ ДИСПЛЕИ С RASPBERRY PI И БИБЛИОТЕКА KIVY

Делян Генков

Технически университет - Габрово

PROGRAMMING FOR TOUCH-SCREEN DISPLAYS USING RASPBERRY PI AND KIVY LIBRARY

Delyan Genkov

Technical University - Gabrovo

Abstract

Touch screen displays are an essential part of our life. We all use smartphones, tablets, public devices, presentation screens and many other devices, equipped with touch screen displays on daily basis. There are many libraries and frameworks that can be used for programming for these displays. The paper presents one way to build such programs, using a library - Kivy, installed on a Raspberry Pi computer. The main goal of the paper is to share the undocumented behavior of the library and the experience of the author to overcome these unexpected features.

Keywords: Touch Screen Display, Raspberry Pi, Kivy, Python.

ВЪВЕДЕНИЕ

Много съвременни устройства използват тактилни дисплеи – мобилни телефони, таблети, екрани за автомобилни табла, публични дисплеи. Съществуват много програмни среди и библиотеки, които могат да се използват за прихващане и обработка на докосването на потребителя до дисплея.

В настоящия доклад се описва опитът на автора с една такава библиотека – Kivy – безплатна библиотека с отворен код, позволяваща използване с програмен език Python. Библиотеката е многоплатформена, поддържа Android, iOS, Linux, macOS и Windows.[1]

Въпреки че проектът се развива вече 12 години от общност, наброяваща над 150 души и наличието на документация на сайта на проекта, не всички функционалности работят както е описано, понякога документацията е доста повърхностна и не може да се разбере лесно как и какво точно трябва да се направи. Поради широкото и използване има много въпроси за конкретни функции или употреба в Интернет пространството, които получават различни отговори и трикове, които понякога работят, друг път – не.

При разработката на конкретен проект беше избрана тази библиотека за управление на устройство чрез тактилен дисплей и бяха открити доста различия на поведението от документацията, както и бяха преодоляни доста трудности.

ИЗЛОЖЕНИЕ

Основна цел на проекта беше замяна на блок с механични бутони с тактилен дисплей, като се симулират софтуерни бутони, при натискането на които да се изпраща команда към машината, но да бъдат добавени допълнителни функционалности. Част от тях са добавяне на екрани с допълнителна информация за компанията опериращa машините, извеждане на анкета за качеството на услугите, възпроизвеждане на рекламен клип, въвеждане на многоезичен интерфейс, както и екрани, съобщаващи за нефункционалност на машината и за режим на отдалечена диагностика.

За управлението на машината беше избран Raspberry Pi 4B компютър с 4 GB RAM памет и с операционна система Raspbian версия 11 (bullseye). Изборът беше продиктуван от наличието на HDMI порт за дисплея и USB портове за получаване на информация за докосване на екрана, както и от наличието на достатъчен брой входно/изходни портове за симулиране на 22 броя хардуерни бутони за машината.

Използваният дисплей е ZHIXIANDA 21.5 Inch Wide Screen FHD 1920x1080 With HDMI VGA USB Input Industrial Open Frame Touch Monitor, поради подходящия му размер за вграждане в машината. Той използва HDMI интерфейс за извеждане на информация и USB интерфейс за предаване на информацията за докосването към компютъра. При свързването и двете функции бяха автоматично разпознати от операционната система и не се наложи да се инсталират драйвери или да се калибрира системата, управлението на операционната система се изпълняваше успешно чрез докосвания на екрана.

За управление на машината бяха инсталирани библиотеката Kivy и необходимите допълнителни компоненти, според инсталационната процедура от документацията, както и необходимите компоненти за програмиране с Python. [4]

В Kivy графичните примитиви (бутони, текстови полета, изображения) се наричат с общото наименование Widget. Те могат да се добавят към подложки (Layouts), които имат различен начин на разполагане на обектите в тях.

Свойствата на всеки обект – цвят, размер и други се управляват, променяйки свойството на обекта в програмата на Python или чрез специален език – Kivy Language, чрез който във външен файл с разширение .kv се описват свойствата. Пример за такова описание е представен на фигура 1.

```
MyWidget:
    canvas:
        Color:
            rgba: 1, .3, .8, .5
```

***Фиг. 1**. Пример за Kivy Language*

Примерът създава фон (Canvas) с цвят 100% червено, 30% зелено, 80% синьо и 50% прозрачност. Еквивалентният код на Python е показан на фигура 2.

```
with self.canvas:
    Color(1,.3,.8,.5, mode='rgba')
```

***Фиг. 2**. Еквивалентен код на Python*

Поради относителната спешност на проекта и непознаването на Kivy Language, неговото използване беше изоставено и целия код на обектите в приложението беше описан на Python. В последствие беше установено, че избраният подход може би не е най-добрият, защото кодът на финалната програма надвиши 1300 реда и стана труден за ориентация, както и промяната на свойствата на някои обекти се затрудни.

Цялата структура на обектите е дървовидна – коренът на дървото е главният обект на приложението, като в него могат да се добавят различни обекти – деца (Children), които в себе си могат да съдържат други обекти.

Идеята на изграждането приложение беше то да има различни екрани за различните функционалности. Затова за главен обект беше избран ScreenManager – обект, който може да съдържа в себе си много екрани и да сменя активния при команда.

Отделните екрани съдържат различни обекти. В зависимост от начина на подреждане на обектите е удобно като контейнери да се използват подложки (Layouts). Поддържат се следните видове подложки:

- BoxLayout – хоризонтален или вертикален. Разпределя обектите по хоризонтала или по вертикала, разделяйки пространството между всички обекти;
- GridLayout – нарежда обектите в решетка с фиксиран брой редове и/или колони;
- AnchorLayout, където позицията на елемент може да се избира между горе, център и долу във вертикална позиция и ляво, център и дясно в хоризонтална;
- FloatLayout, където позицията на всеки елемент може да се посочва в

относителни или абсолютни координати.

Тези елементи могат да се наслагват един върху друг, за да се постигне желаната подредба. При докосване на екрана, всеки обект, който е в обсега на докосването получава събитие on_press и on_release, независимо на каква дълбочина се намира.

В нашия случай главният екран на приложението ще бъде с фиксиран размер и изисква подреждането на 18 еднакви по размер бутона, наредени в три колони. Освен тях са необходими още 5 допълнителни бутона, разположени на фиксирани места. Разположението на екрана е показано на фигура 3.

***Фиг. 3**. Основен екран на приложението*

Бутоните за избор на напитка са разположени под съответната картинка и са наредени автоматично в GridLayout. Добавянето на един бутон е показано на фигура 4, за другите е аналогично.

```
grid = GridLayout(cols=3, rows=6,
spacing = (50,50), padding = (80,
300, 80, 150))

btn1=Button(text='Button 1',
background_color = (0,0,0,0))

btn1.bind(on_press=coffee_btn)
btn1.bind(on_release=clear_out)
grid.add_widget(btn1)
```

***Фиг. 4**. Добавяне на бутони в GridLayout*

На горните три реда се създава GridLayout с три колони и 6 реда, задава се разстоянието между отделните бутони и отстоянието на елементите от границите на екрана. След това се създава бутон с име Button 1. Името е важно, защото после във функциите, присъединени към събитията на бутоните ще се разпознава кой бутон е натиснат или отпуснат по името на бутона. Чрез действието bind се назначават функциите, които се изпълняват при натискане и отпускане на бутон и накрая бутонът се добавя в подложката.

При добавяне на елементи в Layout трябва да се има предвид, че в масива, описващ поделементите на подложката – children[] елементът с индекс 0 е последния добавен бутон, а не първия.

Другите бутони са в полетата горе вляво под емблемата на клиента за портфолио, горе в средата за регулиране на захар, долу вляво за информация и долу вдясно за смяна на език. Освен тях има две текстови полета за сумата, която е пусната в машината и горе вдясно за изписване на текущия час. Всички тези елементи са наредени в един FloatLayout и позициите им са фиксирани твърдо. Добавянето на първия елемент е показано на фигура 5, за останалите е аналогично.

```
floatlay = FloatLayout()
button = Button(text='About',
size_hint = (.5, .05), pos = (0, 0))

button.bind(on_press=callback)
floatlay.add_widget(button)
```

***Фиг. 5**. Добавяне на бутон във FloatLayout*

Първият ред създава FloatLayout, а втория – бутон. Размерът на бутона в полето

size_hint е в проценти от размера на екрана. В случая размерът на бутона е половината от ширината (x – координатата) на екрана, а по височина – 5% от размера на дисплея. В последствие създаденият FloatLayout с добавените към него бутони се добавя като подчинен елемент на екрана.

След аранжирането на бутоните най-отгоре се добавя изображение, което скрива графичното представяне на бутоните. Върху него се добавят двете текстови полета за кредит и час. Кодът е показан на фигура 6.

```
screen1.add_widget(Image(source='imag
es/menu.jpg'))
lblCredit = Label(text='0.00')
lblCredit.color=(0, 0, 0, 1)
lblCredit.background_color=(0,0,0,0)
lblCredit.font_size=70
lblCredit.bold=True
lblCredit.pos=(90, 890)
```

***Фиг. 6**. Добавяне на изображение и текстово поле*

Първият ред добавя картинката, която е най-отгоре на екрана. След това се създава текстово поле с бял цвят на текста и прозрачен фон с голям размер шрифт, удебелен и се позиционира на абсолютна позиция на екрана. По същия начин се създава и текстовото поле за часа.

Повечето останали екрани са статични изображения с добавени един или няколко бутона, тяхното създаване е аналогично. По-голям интерес представляват екраните с анкетата, която се предлага на потребителя след избора на напитка и екранът с рекламното видео, който се визуализира след отказ от анкета.

На екрана с анкетата могат да се изобразяват до 4 въпроса, което е определено от размера на екрана и времето, което потребителят има, за да попълни анкетата. Въпросите могат да бъдат с по един възможен отговор или с по няколко възможни отговора. Тъй като трябва да се осигури възможност за задаване на въпросите и възможните отговори е дефиниран формат на текстов файл, в който собственика на машината да записва опциите за анкетата. При стартиране на програмата се изчита файла и се конструира анкетата с необходимите контроли и текст, както е показано на фигура 7.

```
Questions=4
Question1=Доволни ли сте от
качеството на напитките?
Type=radio
Answers=2
Answer1=Да
Answer2=Не
```

***Фиг. 7**. Формат на файла за анкета*

Най-отгоре се указва броя на въпросите, след това за всеки въпрос се указва текста на въпроса, типа на отговорите – дали е с радио-бутони (един възможен отговор) или с чек-бокс – няколко възможни отговора. След това се показват броя отговори и техните текстове. За всеки въпрос са предвидени до 4 възможни отговора и ако предложените отговори са по-малко, останалото пространство се допълва с празни текстови полета.

Интересното при избора на отговорите в Kivy е, че и в двата случая контролата се нарича checkbox. Ако трябва да има само един възможен отговор, трябва да се обединят няколкото възможни отговора в група с един и същ номер. Друг интересен момент е, че въпреки че иначе противоречи на правилата е възможно да няма нито един маркиран радиобутон. Проблемът, който беше забелязан беше, че библиотеката използва графични примитиви за радио бутоните и чек боксовете, които се изрязват от предварително записано графично изображение. Поисканата от клиента функционалност да увеличим размера на изображението не можеше да се осъществи по стандартен начин. За да стане възможно увеличаването на размера на чек-боксовете и радио-бутоните се наложи да се промени файла по подразбиране на библиотеката Kivy, който указва от коя точка и колко точки да бъдат изрязани от оригиналното изображение. Файлът се нарича defaulttheme.atlas, по подразбиране се намира в папка /usr/local/lib/python3.9/dist-packages/kivy/data/images/ и показва от кои координати на комбинираното изображение колко точки да се изрежат. За да се уголеми изображението на радиобутоните, се наложи да бъдат сменени всички координати на текстовете, съдържащи checkbox. Примерен текст на бутон е: "checkbox_radio_off": [196, 137, 26, 26]. Левите две стойност се увеличават спрямо подразбиращите се с 4

пиксела, а десните две се намаляват с 4.

Когато даден бутон бъде натиснат, програмата влиза в съответната функция, която е закачена при дефиницията на бутона. Възможно е функциите да са различни – при натискане и при отпускане на бутона, както е разработено в настоящата система. Един интересен ефект, който беше забелязан при настоящата разработка и не беше открит отговор при нито един Интернет форум беше, че при всяко натискане на бутон или чекбокс програмата влизаше два последователни пъти в прикачената функция и също така при отпускане влиза два пъти в съответната функция. Това наложи филтриране на натискането и отпускането – проверка дали това е първото натискане, при което да се предприема съответното действие и при второ задействане да пропуска извикването на функцията.

Друг интересен момент при работата на приложението беше с инициализацията на серийния порт. Разработваният софтуер си комуникира чрез USB интерфейс с външен контролер, свързан към LCD дисплей на машината за осигуряване на информация за текущия статус на машината и други входни параметри.

```
try:
   ser=serial.Serial("/dev/ttyUSB0",115200)
except:
   logging.info("ttyUSB0 not found!")
   try:

ser=serial.Serial("/dev/ttyUSB1",115200)
   except:
      logging.info("ttyUSB1 not found")
```

***Фиг. 8**. Инициализация на сериен порт*

Без промяна на физическия USB порт, в който е включен контролера, веднъж интерфейса се разпознава като устройство ttyUSB0, друг път като ttyUSB1. За преодоляване на този проблем, трябва да се направи проверка в началото на програмата какъв е текущият номер, както е показано на фигура 8.

ЗАКЛЮЧЕНИЕ

В заключение Kivy е една удобна библиотека за обработка на събития при работа с тактилен дисплей. Въпреки широкото си разпространение, документацията на проекта е доста оскъдна и някои функционалности имат различно от документираното поведение. В бъдеще са планирани подобни разработки за Android, както и разширяване на функционалността на съществуващата разработка.

БЛАГОДАРНОСТИ

Настоящият документ е изготвен с финансовата помощ на договор № 2203E за провеждане на научни изследвания по проект на тема: „Разработка и валидиране на решения за ефективно дистанционно обучение чрез използване на иновативни ИКТ технологии“ към Технически университет – Габрово.

REFERENCE

[1] Kivy, Kivy: The Open Source Python App development Framework., https://kivy.org/, date of usage: 20.10.2022.

[2] Raspberry Pi foundation, Raspberry Pi 4 - Your tiny, dual-display, desktop computer…and robot brains, smart home hub, media centre, networked AI core, factory controller, and much more, https://www.raspberrypi.com/products/raspberry-pi-4-model-b/, date of usage 20.10.2022.

[3] Focus Tehnology, Metal Case 10 10.4 11.6 12.1 15 17 19 21.5 Inch Pcap Touch Screen Monitor Industrial Open Frame LCD Monitor, https://zhixianda.en.made-in-china.com/product/NQCpSRJVvnhE/China-Metal-Case-10-10-4-11-6-12-1-15-17-19-21-5-Inch-Pcap-Touch-Screen-Monitor-Industrial-Open-Frame-LCD-Monitor.html, Date of usage: 20.10.2022.

[4] Kivy. Getting Started » Installing Kivy, https://kivy.org/doc/stable/gettingstarted/installation.html, Date of usage: 18.02.2022.