RaspberryPI Demo Sensor Kit: Fase III

Juan Lasso, Edward Montaño

Junio, 2020

Resumen

Durante el desarrollo de esta fase del proyecto se realizo la implementación de una interfaz gráfica de usuario (GUI, por sus siglas en ingles) para gestionar la ejecución de las tareas implementadas en la fase anterior del proyecto. Se describen las distintas partes graficas de la interfaz, y su funcionamiento con su respectivo codigo en Python. Por ultimo se concluye que, a apesar de ser un lenguage muy versátil en campos como por ejemplo, el análisis numérico, las librerías disponibles en Python para la realización de dichas interfaces son bastante limitadas. ***Palabras Clave: Python, GUI, Interfaz, Layout.***

Introducción

Python es un lenguage caracterizado por su versatilidad a la hora de trabajar en el análisis numérico y otros campos, principalmente por la amplia variedad de librerías desarrolladas, que facilitan en gran medida la realización de dichas tareas. A la hora de realizar gráficos e interfaces de usuario también existen este tipo de librerías. Para este caso, se trabajara con la librería *Tkinter*. Las librerías de diseño de interfaces para Python son señaladas constantemente por ser bastante limitadas, es por eso que distintos programadores las usan únicamente son fines experimentales y recurren a otros lenguajes como javascript para realizar interfaces mucho mas profesionales. Sin embargo, dichas librerías son bastante intuitivas así que esta fase del proyecto se centra en aprovechar tanto la versatilidad de Python para trabajar con distintos componentes junto con herramientas intuitivas para el diseño de interfaces de usuario.

1. Implementación de la interfaz gráfica

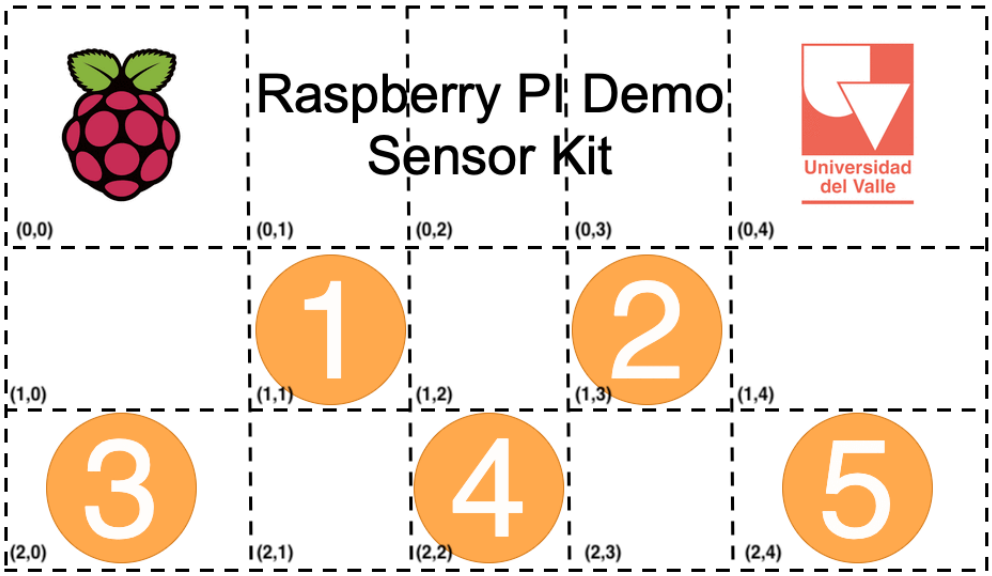
Como se menciono anteriormente, esta interfaz se realizo con la ayuda de la librería de Python *Tkinter*. Primero para trabajar con esta librería es necesario crear una variable "window" principal a la que, se le modifican sus atributos o se le agregan nuevos componentes como botones, cuadros de texto, entre otros. Así que lo primero que se realiza en la interfaz es, importar las librerías a usar y crear la ventana principal.

```
import os
import tkinter as tk
from tkinter import *

window = tk.Tk()
window.geometry("950x650")
window.title("Raspberry Demo GUI")
```

Luego, como la interfaz realizada trabaja con imágenes, primero se deben crear las variables que almacenen la ruta a dichas imágenes. Una anotación importante es que las imágenes deben estar en formato .gif para que las librería las reconozca. Estas imagenes seran usadas para logos presentes en la interfaz y botones.

```
boton1         = PhotoImage(file="button1.gif")
boton2         = PhotoImage(file="button2.gif")
boton3         = PhotoImage(file="button3.gif")
boton4         = PhotoImage(file="button4.gif")
boton5         = PhotoImage(file="button5.gif")
ejecutarButton = PhotoImage(file="Ejecutar2.gif")
backButton     = PhotoImage(file="back.gif")
photoPi        = PhotoImage(file="raspberry-pi-logo.gif")
photoUv        = PhotoImage(file="Logo-uv.gif")
photoBlank     = PhotoImage(file="blank.gif")
```

Ahora, empezamos a agregar nuevos elementos a nuestra GUI. lo primero que se agrega son espacios en blanco llamados "blanks". Esto se hace pues nuestra interfaz trabaja con un Layout Grid. Esto significa que la interfaz ubica los elementos son un sistema de coordenadas que no son absolutas, sino que dependen de las filas y las columnas que ubiquen los demás elementos en la ventana. En la siguiente imagen se observa el sistema de coordenadas de la pantalla principal de nuestra GUI.



Luego se crean los "blanks" que se agregaran posteriormente a la interfaz.

```
blankSpace = 100
blank1 = Label(window,image=photoBlank, height=blankSpace, width=blankSpace)
blank3 = Label(window,image=photoBlank, height=blankSpace, width=blankSpace)
blank5 = Label(window,image=photoBlank, height=blankSpace, width=blankSpace)
blank7 = Label(window,image=photoBlank, height=blankSpace, width=blankSpace)
blank9 = Label(window,image=photoBlank, height=blankSpace, width=blankSpace)
```

Luego se crean las variables que almacenan las imagenes tanto del logo de la Raspberry Pi como de la Universidad del Valle

```
piLogo = Label(window,image=photoPi, height=200, width=240)
uvLogo = Label(window,image=photoUv, height=200, width=240)
```

Se crean las variables que almacenas los Titulos, o textos grandes creados como Labels. Estos son el texto principal de la ventana y el texto que indica el numero de la actividad que se realiza en cada ventana.

```
titulo = Label(window,text="\nRaspberry PI Demo\nSensor Kit\n",fg="black",bg="white",font=("Arial",50))
numeroId = Label(window, text="test", fg="black", bg="white", font=("Arial", 60))
```

Ahora la parte mas importante de nuestra interfaz, los botones. Los botones principales de nuestra interfaz son los que nos permiten navegar a la pagina de cada actividad. Estos tienen el numero de la actividad, evidentemente, pero mas importante tienen asociado un método llamado paginaACtividad() que recibe el numero de una determinada actividad y modifica los elementos en pantalla de manera que muestra una pagina distinta y permite ejecutar una actividad distinta dependiendo de que pagina este activa. A continuación se crean los botones, con su respectivo tamaño y asociándoles un comando a cada uno con el numero de su actividad. Además, se crea el botón "back" que permite regresar a la pagina principal luego de haber navegado a una pagina de actividad con el método regresarPaginaPrincipal().

```
buttonSize = 142
btn1 = Button(window, image=boton1, height=buttonSize, width=buttonSize, borderwidth=0,
command =lambda: paginaActividad(1))
btn2 = Button(window, image=boton2, height=buttonSize, width=buttonSize, borderwidth=0,
command =lambda: paginaActividad(2))
btn3 = Button(window, image=boton3, height=buttonSize, width=buttonSize, borderwidth=0,
command =lambda: paginaActividad(3))
btn4 = Button(window, image=boton4, height=buttonSize, width=buttonSize, borderwidth=0,
command =lambda: paginaActividad(4))
btn5 = Button(window, image=boton5, height=buttonSize, width=buttonSize, borderwidth=0,
command =lambda: paginaActividad(5))
back = Button(window, image=backButton, height=buttonSize, width=buttonSize, borderwidth=0,
command =lambda: regresarPaginaPrincipal())
```

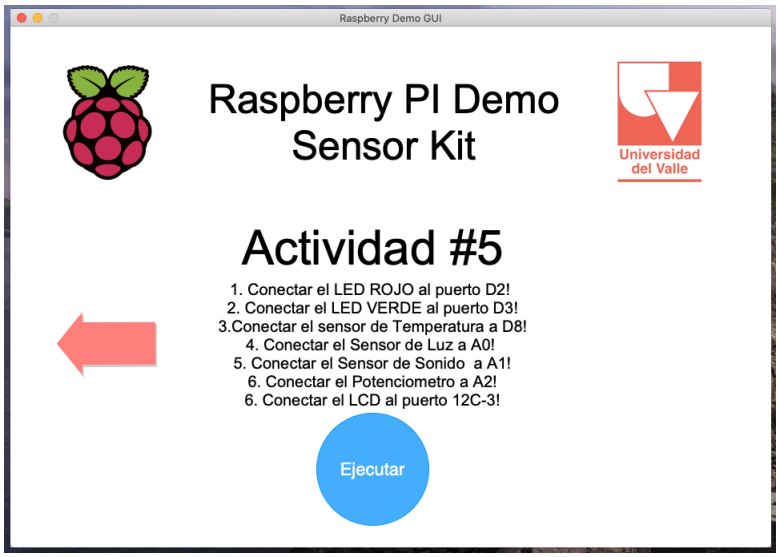
La forma en que funciona el método de paginaActividad() es relativamente sencilla. Con el fin de crear una nueva "pagina" de nuestra interfaz es necesario esconder todos los elementos que no son necesarios y revelar los botones y el texto que si es necesario mostrar. Para esto se crea un arreglo llamado buttons con todos los elementos presentes en la pagina principal de la interfaz. Luego de esto, se configura el texto que identifica en que pagina de actividad nos encitáramos y se revelan tanto las instrucciones de la actividad actual como su botón de ejecución.

```
buttons = [btn1,btn2,btn3,btn4,btn5,blank1,blank3,blank5,blank7,blank9]
def paginaActividad(numeroActividad):
    for btn in buttons:
        btn.grid_forget()
    back.grid(row=2, column = 0)
    numeroId.configure(text = "Actividad #"+str(numeroActividad))
    numeroId.grid(row =1, column = 1)
    if numeroActividad == 1:
        btnActividad1.grid(row=3, column=1)
        textoActividad1.grid(row=2, column=1)
    elif numeroActividad ==2:
        btnActividad2.grid(row=3, column=1)
        textoActividad2.grid(row=2, column=1)
    elif numeroActividad ==3:
        btnActividad3.grid(row=3, column=1)
        textoActividad3.grid(row=2, column=1)
    elif numeroActividad ==4:
        btnActividad4.grid(row=3, column=1)
        textoActividad4.grid(row=2, column=1)
    elif numeroActividad ==5:
        btnActividad5.grid(row=3, column=1)
        textoActividad5.grid(row=2, column=1)
```

Luego se crean los botones de ejecución. Dichos botones funcionan con el método ejecutarActividad(). Este método, que también recibe un numero de 1 a 5, ejecuta un determinado script de Python que contiene el codigo de cada respectiva actividad implementado en la fase anterior de proyecto.

```
btnActividad1 = Button(window, image=ejecutarButton, height=buttonSize, width=buttonSize-1,
borderwidth=0, command =lambda: ejecutarActividad(1))

btnActividad2 = Button(window, image=ejecutarButton, height=buttonSize, width=buttonSize-1,
borderwidth=0, command =lambda: ejecutarActividad(2))

btnActividad3 = Button(window, image=ejecutarButton, height=buttonSize, width=buttonSize-1,
borderwidth=0, command =lambda: ejecutarActividad(3))

btnActividad4 = Button(window, image=ejecutarButton, height=buttonSize, width=buttonSize-1,
borderwidth=0, command =lambda: ejecutarActividad(4))

btnActividad5 = Button(window, image=ejecutarButton, height=buttonSize, width=buttonSize-1,
borderwidth=0, command =lambda: ejecutarActividad(5))
```

En la siguiente imagen se observa la pagina de una actividad de ejemplo, en este caso la actividad 5.



El método de ejecutarActividad() funciona gracias a la librería os que viene instalada por defecto en Python. Esto permite ejecutar un script de Python desde otro script como si de una consola de comandos se tratase. De esta manera se puede sencillamente ejecutar los scripts implementados en la fase anterior del proyecto presionando únicamente los botones. Este método recibe un entero que determina el script a ejecutar.

```
def ejecutarActividad(numeroActividad):

    if numeroActividad == 1:
        os.system("python3 actividad1.py")

    elif numeroActividad == 2:
        os.system("python3 actividad2.py")

    elif numeroActividad == 3:
       os.system("python3 actividad3.py")

    elif numeroActividad == 4:
        os.system("python3 actividad4.py")

    elif numeroActividad == 5:
        os.system("python3 actividad5.py")
```

Como se aprecia en la imagen anterior existe un botón llamado "back" que permite regresar a la pantalla de selección de actividades al ser presionado. Este tiene asociado el método regresarPaginaPrincipal(), que se encarga de esconder los elementos de la ventana actuales y revelar nuevamente los botones de selección junto con los espacios de separación.

```
def regresarPaginaPrincipal():

numeroId.grid_forget()
back.grid_forget()
btnActividad1.grid_forget()
btnActividad2.grid_forget()
btnActividad3.grid_forget()
btnActividad4.grid_forget()
btnActividad5.grid_forget()
textoActividad1.grid_forget()
textoActividad2.grid_forget()
textoActividad3.grid_forget()
textoActividad4.grid_forget()
textoActividad5.grid_forget()
titulo.grid(row=0, column=1, columnspan=3)
piLogo.grid(row=0, column=0)
uvLogo.grid(row=0, column=4)
blank1.grid(row=1, column=0)
blank1.grid(row=1, column=0)
blank3.grid(row=1, column=2)
blank5.grid(row=1, column=4)
blank7.grid(row=2, column=1)
blank9.grid(row=2, column=3)
btn1.grid(row=1, column=1)
btn2.grid(row=1, column=3)
btn3.grid(row=2, column=0)
btn4.grid(row=2, column=2)
btn5.grid(row=2, column=4)
```

El ultimo paso es inicializar nuestra ventana con sus componentes iniciales y ejecutar el bucle que leerá constantemente los eventos que accione el usuario.

```
titulo.grid(row = 0, column = 1,columnspan=3)

setElement(piLogo, 0, 0)
setElement(uvLogo, 0, 4)
setElement(blank1, 1, 0)
setElement(blank3, 1, 2)
setElement(blank5, 1, 4)
setElement(blank7, 2, 1)
setElement(blank9, 2, 3)
setElement(btn1, 1, 1)
setElement(btn2, 1, 3)
setElement(btn3, 2, 0)
setElement(btn4, 2, 2)
setElement(btn5, 2, 4)



#Finalmente se crea le ventana de la GUI junto con el bucle principal
window.configure(background='white')
window.resizable(width=False, height=False)
window.mainloop()
```

2. Conclusiones

- A pesar de ser bastante intuitiva, durante el desarrollo de esta fase se observo como la librería usada presentaba problemas a la hora de ejecutar la interfaz en sistemas operativos diferentes. Esto debido a que cada sistema tiene parámetros diferentes a la hora de organizar los elementos en las ventanas que se muestran al usuario.
- Para interfaces experimentales y que no requieran una apariencia profesional con animaciones o imagenes de alta calidad, librerías como Tkinter son perfectas pues son muy fáciles de usar, y muchas personas las usan, lo que permite hayan mucha información acerca de su uso en foros o pagina dedicadas a realizar tutoriales de programación.

3. Bibliografía

https://docs.python.org/3/library/tk.html

https://stackoverflow.com/questions/2416170/pros-and-cons-of-tkinter-and-wxwidgets