Article **ECORFAN Journal Democratic Republic of Congo**
June 2019 Vol.5 No.8 1-6

Application to monitoring a USB control with Python in Windows, Mac OS and Raspbian OS

Aplicación para monitoreo de un control USB con Python en Windows, Mac OS y Raspbian OS

ABRIL-GARCÍA, José Humberto†*, ALCANTAR-MARTINEZ, Adelina del Carmen, LÓPEZ-ROMO, José Alonso and MEZA-IBARRA, Iván Dostoyewski

Universidad Tecnológica de Hermosillo, Information Technologies Engineering

ID 1st Author: *José Humberto, Abril-García* / **ORC ID:** 0000-0003-3494-6817, **Researcher ID Thomson:** F-4252-2018, **arXiv ID:** jhabril, **CVU CONACYT ID:** 204935

ID 1st Coauthor: *Adelina del Carmen, Alcántar-Martínez* / **ORC ID:** 0000-0003-2715-9209, **Researcher ID Thomson:** F-6771-2018, **arXiv ID:** adelina-alcantar, **CVU CONACYT ID:** 640868

ID 2nd Coauthor: *José Alonso, Lopez-Romo* / **ORC ID**: 0000-0001-7428-1480, **Researcher ID Thomson:** R-5616-2018, **arXiv:** alonsolopez2, **CVU CONACYT:** 944227

ID 3rd Coauthor: *Ivan Dostoyewski, Meza-Ibarra* / **ORC ID:** 0000-0001-6139-032X, **Researcher ID Thomson:** F-3550-2018, **arXiv ID:** imeza, **CVU CONACYT ID:** 769494

DOI: 10.35429/EJDRC.2019.8.5.1.6 Received May 18, 2019; Accepted June 20, 2019

Abstract

The present work shows the development of an application in Python to detect the actions carried out on a USB control, with the aim of achieving a multiplatform application that can be used as a base for the creation of more robust and complex applications such as the development of video games or control and monitoring of devices. The methodology used was SCRUM in the first sprint a program was written for CLI that prints messages based on the buttons pressed, in the second sprint a GUI was developed to show a USB control and to indicate graphically when a button was pressed, in the last sprint a single application was integrated, the necessary tests were made and the code was published in a repository in GIT for reference and future use. The application was developed for Windows, macOS and Raspbian to test Python portability.

Python, Windows, Mac OS, Raspbian, USB gamepad

Resumen

El presente trabajo muestra el desarrollo de una aplicación en Python para detectar las acciones realizadas sobre un control USB, con el objetivo de lograr una aplicación multiplataforma que puede ser usada como base para la creación de proyectos más robustos y complejos, como el desarrollo de video juegos o control y monitoreo de dispositivos. La metodología utilizada fue SCRUM, en el primer sprint se escribió un programa para CLI que imprime mensajes dependiendo de los botones presionados, en el segundo se desarrolló una GUI que mostrara un control USB y que indicara de manera gráfica cuando se presionara un botón, finalmente se integró una sola aplicación, se hicieron las pruebas necesarias y se publicó el código en un repositorio en GIT para referencia y uso futuro. La aplicación fue desarrollada para Windows, macOS y Raspbian para probar la portabilidad de Python.

Python, Windows, Mac Os, Raspbian, Control USB

Citation: ABRIL-GARCÍA, José Humberto, ALCANTAR-MARTINEZ, Adelina del Carmen, LÓPEZ-ROMO, José Alonso and MEZA-IBARRA, Iván Dostoyewski. Application to monitoring a USB control with Python in Windows, Mac OS and Raspbian OS. ECORFAN Journal-Democratic Republic of Congo 2019, 5-8: 1-6.

*Correspondence to Author (email: abril@uthermosillo.edu.mx)
† Researcher contributing first author.

Introduction

Today the use of Python to develop several kinds of application is growing at a high rate, we can use Python for Web and Internet Development, scientific and numeric computing, education Desktop GUIs, software development, video games, and Business Applications like ERP and e-commerce systems, are examples of the multiple uses of Python. In this work we present an application to monitoring a USB control on Windows, Mac OS and Raspberry Pi (Foundation R. P., 2018). Our aim in this paper is to develop an application that can be used like a start point in more complex and robust aplications. Figure 1 shows the general diagram of the project.

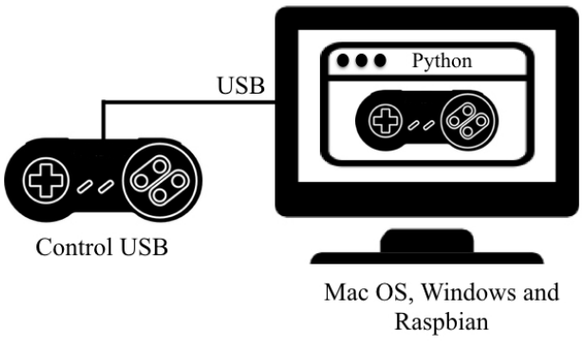


Figure 1 General Diagram

In the literature, we can find applications that make use of Python and Pygame for the development of applications in wireless networks (Cherry, 2015) and in the development of video games (Rachana Kumari, 2017). The main contribution of this work is the development of a base prototype that allows the students to know the main characteristics of the Python programming language, as well as to have a guide for the development of more complex projects that allow the integration of multiple technologies. The rest of the paper is organized as follows: Section 2 describes the tools used for this project; Section 3 gives conclusion to this work. Furthermore, analysis and future use for this application are provided.

Description of tools

Python (Foundation P. S., Welcome to Python.org, 2018) (Nagar, 2018) is an object-oriented, high-level programming language with integrated dynamic semantics primarily for web and app development.

It is extremely attractive in the field of Rapid Application Development because it offers dynamic typing and dynamic binding options. Python is a cross-platform language, a program written on a Macintosh computer will run on a Linux system and viceversa. Python programs can run on any computer, as long as the machine has the Python interpreter installed

Pip (Foundation P. S., pip · PyPI, 2018) is a package management system used to install and manage software packages written in Python. Pip was used to upgrade and install packages required on this project.

Pygame (pygame, 2018) is a cross-platform set of modules designed for writing multimedia applications like games built on top of the SDL library. It includes computer graphics and sound libraries designed to be used with the Python programming language.

Raspbian (FOUNDATION, 2018) is a free operating system based on Debian optimized for the Raspberry Pi hardware. An operating system is the set of basic programs and utilities that make Raspberry Pi run.

Github (Conservancy, 2018), (GitHub, 2018) is a web-based hosting service for version control using Git. It is mostly used for computer code. It offers all of the distributed version control and source code management (functionality of Git. It provides access control and collaboration features such as bug tracking, feature requests and task management, in this project we use Github to make the code avaible for any user.

Hardware and Software specificications

The hardware used in this project was:

– Pentium(R) Dual-Core CPU E5700 @ 3.00 GHz, 4 GB RAM, 32 bits.

– 2.7 GHz Intel Core i5, 8 GB 1867 MHz DDR3
– Raspberry Pi (c) 2011.12

The software used in this project:

– macOS High Sierra version 10.13.5
– Python 2.7.10 for Mac Os
– IDLE 2.7.10 for Mac Os
– Windows Home Basic 32 bits Service Pack 1

ISSN: 2414- 4924

– Python 3.6.5 for Windows
– IDLE 3.6.5 for Windows
– Raspbian GNU/Linux 9 (stretch)
– Python for Raspbian 2.7.13
– IDLE 2.7.13

Methodology

After installing all the tools required developing on Pyton, and using SCRUM team methodology; A Command Line Interface CLI application was developed to identify the control pad and buttons on sprint 1. On sprint 2, two Graphical User Interfaces were developed for SNES and NES. Finally on sprint 3, integration was done to merge the prototypes made on sprint one and two.

Sprint 1: The code generated on this stage runs with modifications for each platform.

Windows code:

```
# Get joystick axes for Win

x = joy.get_axis(0)

if(str(x)!="-0.00784301757812"): print(x)

y = joy.get_axis(1)

if(str(y)!="-0.00784301757812"): print(y)
```

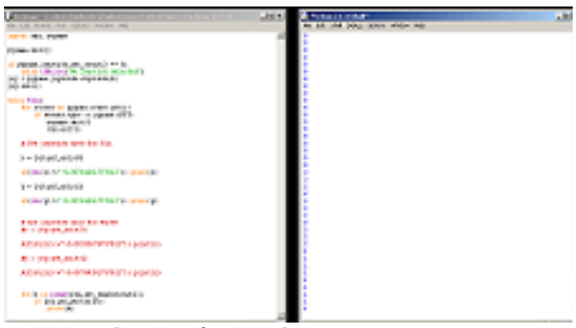
Mac OS code:

```
# Get joystick axes for MacOS

x = joy.get_axis(3)

if(str(x)!="-0.00393676757812"): print(x)

y = joy.get_axis(4)

if(str(y)!="-0.00784301757812"): print(y)
```

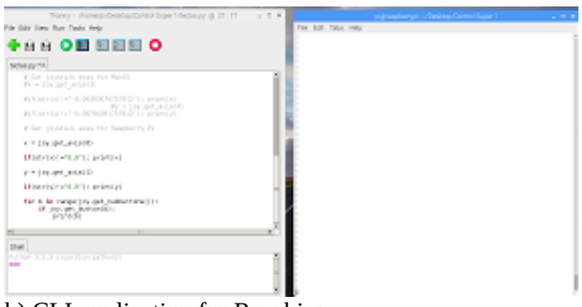
Raspbian code:

```
# Get joystick axes for Raspberry Pi

x = joy.get_axis(0)

if(str(x)!="0.0"): print(x)

y = joy.get_axis(1)

if(str(y)!="0.0"): print(y)
```

Figure 2 shows the code in execution on each platform.



a) CLI application for Windows



b) CLI application for Raspbian

c) CLI application for macOS

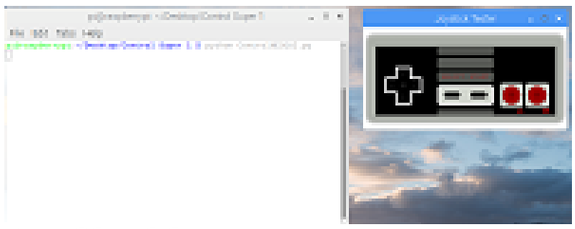
Figure 2 Python CLI application for a) Windows, b) Raspbian and c) macOS

Sprint 2 The code generated could be executed without modifications in all the platforms; two GUI's were designed for NES and SNES. On figure 3 we can see the GUI's running on Windows, MacOs and Rapsbian.

a) Windows NES GUI

ISSN: 2414- 4924

b) macOS SNES GUI

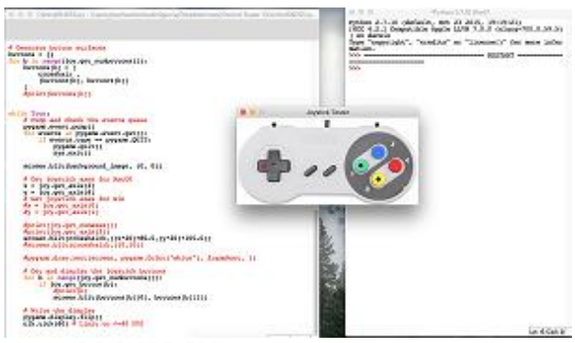


c) macOS NES GUI

Figure 3 Python GUI application for a) Windows NES, b) masOS SNES GUI and c) Raspbian NES GUI

Sprint 3: Finally, the prototypes developed on sprint 1 and 2 were merged on a final version; this version requires small changes on each platform. On Figure 4, we can see the programs running while the user presses key pad left, Y, B, L and R buttons for SNES control and key pad left, key pad up, select, start, A and B buttons for NES control.

a) Windows SNES



b) macOS SNES GUI

c) macOS NES GUI

Figure 4 Python application for a) Windows SNES, b) masOS SNES GUI, c) Raspbian NES GUI, while the user presses some buttons

Results and Conclusions

With the development of this work a Python application compatible was created with any platform that supports the language, and has served as a guide to be able to venture into the development and creation of various types of applications such as video games, in the same way a repository where students and the general public can download the code freely, study it and adapt it to their needs. In the same way, Python's multi-platform capabilities were tested, obtaining applications that with minimal modifications offer the same result in different platforms. Figure 5 shows the final version running on macOS. On appendix A we see the code for SNES and on appendix B the code for the NES applications.

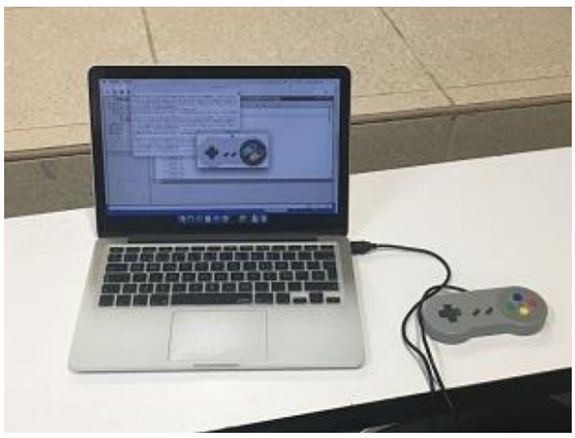


Figure 5 Application for macOS

ISSN: 2414- 4924

Appendix A: SNES Code

```
import sys, pygame, os
os.environ['SDL_VIDEO_CENTERED'] = '1'
pygame.init()

clk = pygame.time.Clock()

if pygame.joystick.get_count() == 0:
   raise IOError("No joystick detected")
joy = pygame.joystick.Joystick(0)
joy.init()

def buttonX(boton):
   if(boton==0): return 298
   elif(boton==1): return 335
   elif(boton==2): return 298
   elif(boton==3): return 260
   elif(boton==4): return 75
   elif(boton==5): return 295
   elif(boton==8): return 154
   elif(boton==9): return 195

def buttonY(boton):
   if(boton==0): return 70
   elif(boton==1): return 98
   elif(boton==2): return 127
   elif(boton==3): return 98
   elif(boton==4 or boton==5): return 11
   elif(boton==8): return 108
   elif(boton==9): return 110

size = width, height = 389, 187
screen = pygame.display.set_mode(size)
pygame.display.set_caption("Joystick Tester")
background_image =
pygame.image.load('cnes1.png').convert()
frameRect = pygame.Rect((0, 0), (width, height))

crosshair = pygame.surface.Surface((10, 10))
crosshair.fill(pygame.Color("magenta"))
pygame.draw.circle(crosshair, pygame.Color("black"),
(5,5), 5, 0)
crosshair.set_colorkey(pygame.Color("magenta"))

crosshairb = pygame.surface.Surface((10, 10))
crosshairb.fill(pygame.Color("magenta"))
pygame.draw.circle(crosshairb,pygame.Color("red"),
(5,5), 5, 0)
crosshairb.set_colorkey(pygame.Color("magenta"))

buttons = {}
for b in range(joy.get_numbuttons()):
   buttons[b] = [
      crosshair ,
      (buttonX(b), buttonY(b))
   ]

while True:
   pygame.event.pump()
   for events in pygame.event.get():
      if events.type == pygame.QUIT:
         pygame.quit()
         sys.exit()
   screen.blit(background_image, (0, 0))

   # Get joystick axes for MacOS
   #x = joy.get_axis(3)
   #y = joy.get_axis(4)

   # Get joystick axes for Win and Raspberry
   # x = joy.get_axis(0)
   # y = joy.get_axis(1)

screen.blit(crosshairb,((x*20)+80-5,(y*20)+105-5))

for b in range(joy.get_numbuttons()):
      if joy.get_button(b):

         screen.blit(buttons[b][0], buttons[b][1])


   pygame.display.flip()
   clk.tick(40)
```

Appendix B: NES Code

```
import sys, pygame
pygame.init()

clk = pygame.time.Clock()

if pygame.joystick.get_count() == 0:
   raise IOError("No joystick detected")
joy = pygame.joystick.Joystick(0)
joy.init()

def buttonX(boton):
   if(boton==1): return 295
   elif(boton==2): return 250
   elif(boton==8): return 148
   elif(boton==9): return 193

def buttonY(boton):
   if(boton==1 or boton==2): return 112
   elif(boton==8 or boton==9): return 112

size = width, height = 350, 175
screen = pygame.display.set_mode(size)
pygame.display.set_caption("Joystick Tester")
background_image =
pygame.image.load('image.png').convert_alpha()

crosshair = pygame.surface.Surface((10, 10))
crosshair.fill(pygame.Color("magenta"))
pygame.draw.circle(crosshair, pygame.Color("blue"),
(5,5), 5, 0)
crosshair.set_colorkey(pygame.Color("magenta"))

crosshairb = pygame.surface.Surface((10, 10))
crosshairb.fill(pygame.Color("magenta"))
pygame.draw.circle(crosshairb,pygame.Color("red"),
(5,5), 5, 0)
crosshairb.set_colorkey(pygame.Color("magenta"))

buttons = {}
for b in range(joy.get_numbuttons()):
   buttons[b] = [ crosshair , (buttonX(b), buttonY(b))]
```

ISSN: 2414- 4924

```
While True:

    pygame.event.pump()
    for events in pygame.event.get():
        if events.type == pygame.QUIT:
            pygame.quit()
            sys.exit()
    screen.fill([255, 255, 255])
    screen.blit(background_image, (0, 0))

    # Get joystick axes for MacOS
    # x = joy.get_axis(3)
    # y = joy.get_axis(4)

    # Get joystick axes for Win and Raspberry
    # x = joy.get_axis(0)
    # y = joy.get_axis(1)
    screen.blit(crosshairb,((x*20)+64,(y*20)+96))


    for b in range(joy.get_numbuttons()):
        if joy.get_button(b):
            screen.blit(buttons[b][0], buttons[b][1])

    pygame.display.flip()
    clk.tick(40)
```

References

Cherry, Y. B. (2015). *Implementing Packet Transfer in wireless Networks Using PyGame.* IEEE.

Conservancy, S. F. (2018). *Git*. Retrieved 2018, from Git: https://git-scm.com/

Foundation, P. S. (2018). *pip · PyPI*. Retrieved 08 2018, from pip · PyPI: https://pypi.org/project/pip/

Foundation, P. S. (2018). *Welcome to Python.org*. Retrieved 08 2018, from Welcome to Python.org: https://www.python.org/

FOUNDATION, R. P. (2018). *Download Raspbian for Raspberry Pi*. Retrieved 2018, from Download Raspbian for Raspberry Pi: https://www.raspberrypi.org/downloads/raspbian/

Foundation, R. P. (2018). *Raspberry Pi — Teach, Learn, and Make with Raspberry Pi*. Retrieved 08 2018, from Raspberry Pi — Teach, Learn, and Make with Raspberry Pi: https://www.raspberrypi.org/

GitHub. (2018). Retrieved 2018, from GitHub: https://github.com/

Nagar, S. (2018). Introduction to Python for Engineers and Scientists. In S. Nagar, *Introduction to Python for Engineers and Scientists* (pp. 1-11). New York, USA: Apress.

pygame. (2018). Retrieved 2018, from pygame: https://www.pygame.org/news

Rachana Kumari, F. (2017). *ANALYZING THE PyGameGUI MODULES AVAILABLE IN PYTHON*. IEEE Conferences.

ISSN: 2414- 4924