Article **Journal of Computational Systems and ICTs**
June 2022, Vol.8 No.21 11-18

Application to monitoring a SNes control with Ruby in Windows, Linux Ubuntu, OS X El Capitan, and Raspbian GNU/Linux 10

Aplicación para monitoreo de un control USB con Ruby en Windows, Linux Ubuntu, OS X El Capitan y Raspbian GNU/Linux 10

ESPARZA-CASTILLO, Ramón Ángel†, LÓPEZ-ROMO, José Alonso, CAZAREZ-CAMARGO, Noe and ABRIL- GARCÍA, José Humberto*

Universidad Tecnológica de Hermosillo, Information Technologies Engineering, Mexico.

ID 1st Author: *Ramón Ángel, Esparza-Castillo* / **ORC ID:** 0000-0001-6929-8798, **Researcher ID Thomson:** 3453138, **arXiv Author ID:** Ramon.Esparza, **CVU CONACYT ID ID:** 1070055

ID 1st Co-author: *José Alonso, López-Romo* / **ORC ID:** 0000-0001-7428-1480, **Researcher ID Thomson:** R-5616-2018, **arXiv Author ID:** alonsolopezr, **CVU CONACYT ID:** 944227

ID 2nd Co-author: *Noe, Cazarez-Camargo* / **ORC ID**: 0000-0002-9301-0819, **Researcher ID Thomson:** ABO-1496-2022, **arXiv Author ID:** suzuma, **CVU CONACYT ID:** 438961

ID 3rd Co-author: *José Humberto, Abril-García* / **ORC ID:** 0000-0003-3494-6817, **Researcher ID Thomson:** F-4252-2018, **arXiv Author ID:** jhabril, **CVU CONACYT ID:** 204935

DOI: 10.35429/JCSI.2022.21.8.11.18 Received January 20, 2022; Accepted June 30, 2022

Abstract

This paper shows the development of a Ruby application to detect the actions carried out on an USB control, with the goal of achieving the coding of multiplatform application that can be used as a basis for the creation of projects with larger scope and impact in this fiel, such as videogame development, and device control systems. The 'prototype models' methodology was the approach for this project. In the first prototype a CLI program was written that prints out messages in the terminal depending on the buttons that are simulated based on the keys pressed on the keyboard, and so on until the end of the project, in which a final application is delivered, in compliance with the necessary tests. This application code was published in a GITLAB repository for reference and future use. The application was developed to be compatible with Windows, Linux, macOS, and Raspbian operating systems to test Ruby portability.

Ruby, Gosu, Windows, Linux, Mac OS, Raspbian, USB gamepad

Resumen

El presente trabajo muestra el desarrollo de una aplicación en Ruby 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 de mayor alcance, como el desarrollo de videojuegos y control de dispositivos. La metodología utilizada fue 'modelos de prototipos', en el primer prototipo se escribió un programa de para CLI que imprime mensajes en terminal simulando que sepresionan botones en el mando, dependiendo de las teclas que se presionen en el teclado. El proyecto fue evolucionando hasta llegar finalmente a tener una sola aplicación, cumpliendo con las pruebas necesrias y se publicó el código en un repositorio GITLAB para referencia y uso futuro. La aplicación fue desarrollada para que sea compatible con los sistemas operativos Windows, Linux, macOS y Raspbian para probar la portabilidad de Ruby.

Ruby, Gosu, Windows, Linux, Mac OS, Raspbian, USB gamepad

Citation: ESPARZA-CASTILLO, Ramón Ángel, LÓPEZ-ROMO, José Alonso, CAZAREZ-CAMARGO, Noe and ABRIL-GARCÍA, José Humberto. Application to monitoring a SNes control with Ruby in Windows, Linux Ubuntu, OS X El Capitan, and Raspbian GNU/Linux 10. Journal of Computational Systems and ICTs. 2022. 8-21:11-18.

* Correspondence to Author (E-mail: abril@uthermosillo.edu.mx)
† Researcher contributing as first author.

Introduction

Today the use of Ruby to develop several kinds of applications is growing at a high rate, we can use Ruby for Web and Internet Development, scientific, education Desktop GUIs, software development, video games and e-commerce systems, are examples of the multiple uses of Ruby. In this work we present an application to monitor a SNes control with Ruby in Windows, Linux Ubuntu, OS X El Capitan, and Raspbian GNU/Linux 10. Our aim in this paper is to develop an application that can be used like a starting point in more complex and robust aplications. Figure 1 shows the general diagram of the project.

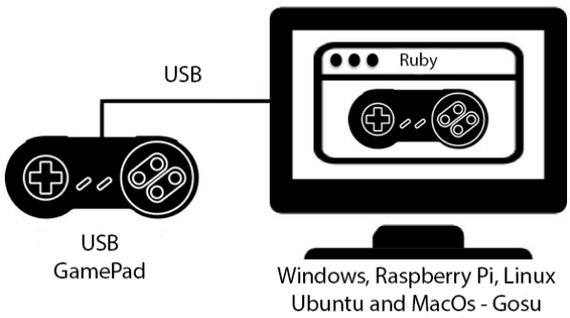

Figure 1 Diagram

Tools description

Ruby (Cooper, 2007) (Mahadevan, 2002) is a language of careful balance. Its creator, Yukihiro “Matz” Matsumoto, did a mix of several parts of his favorite languages (Perl, Smalltalk, Eiffel, Ada, and Lisp) to create a new language that did balanced functional programming with imperative programming.

He has often said that he was “trying to make Ruby natural, not simple,” in a way that mirrors life.

Since its public release in 1995, Ruby has drawn devoted coders worldwide. In 2006, Ruby achieved mass acceptance, with active user groups formed in the world’s major cities and Ruby-related conferences filled to maximum capacity. Ruby-Talk, the primary mailing list for discussion of the Ruby language, climbed to an average of 200 messages per day in 2006. This fluency has dropped in recent years, as has the size of the community, and thus it has pushed discussion from one central list into many smaller groups.

Ruby is ranked among the top 10 programming lenguages on most of the indices that measure the coding leguanges growth and popularity worldwide (TIOBE <the software quality company>, 2022). A large parth of this growth is attributed to the popularity of software written in Ruby, particularly the Ruby on Rails web framework.

Ruby is also completely free. Not only free of charge, but also free to use, copy, modify, and distribute.

RubyGems is a Ruby packaging system designed to facilitate the creation, sharing and installation (in some ways, it is a distribution packaging system like what, apt-get is to debian linux distribution, but targeted at Ruby software). Ruby comes with RubyGems by default since version 1.9, previous Ruby versions require RubyGems to manually.

The main place where Ruby packages are hosted is RubyGems.org, a public repository of gems (a package of Ruby code release to public via RubyGems.org) (RubyGems, 2022) that can be searched and installed onto your machine. You may browse and search for gems using the RubyGems website or use the gem command on the terminal with the CLI.

Gosu (Julian Raschke, 2020) (Sobkowicz, 2015) is a 2D game development library for Ruby and C++. It’s available for macOS, Windows, Linux (including Raspbian), and iOS. Gosu is focused, lightweight and has few dependencies (mostly SDL 2). It provides: a window and a main loop 2D graphics and text, powered by OpenGL or OpenGL ES sounds and music. keyboard, mouse, and gamepad input.

Gosu is mostly used to teach or learn Ruby or in small game development competitions. It’s also a great prototyping tool and even it has been used for indie game development.

About the development environment, VSCode has been used to code the project described in this article. VSCode is a free source code editor made by Microsoft for Windows, Linux and macOS.

ISSN-2444-5002

Its Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git commands (git is a distributed version control system). Besides that, VSCode users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality. VSCode source code comes from Microsoft's free and open source software VSCode project released under the permissive Expat (vscode/LICENSE.txt at main · microsoft/vscode · GitHub, 2022), but the compiled binaries are freeware (License - Visual Studio Code, 2022) for any use.

GitLab is a web-based DevOps lifecycle tool that delivers a Git-repository manager providing wiki, issue-tracking and continuous integration / continuousdeployment pipeline features, using an open-source license, developed by GitLab Inc. GitLab software was created by the Ukrainians Dmitriy Zaporozhets and Valery Sizov.

Hardware and Software specificications

The hardware used in this project was:

- Intel(R) Core(TM) i34025U CPU @ 1.90 GHz, 6 GB RAM, 64 bits.
- Intel(R) Core(TM) i3 CPU M 370 @ 2.40 GHz, 4GB RAM, 32 bits.
- MacBook Pro (Retina 13-inch, Early 2015), Procesador: 2.7 GHz Intel Core i5, Memoria 8 GB 1867 MHz DDR3, Macintosh HD, Intel Iris Graphics 6100 1536 MB.
- Raspberry PI Model 3B+ V1.3, RAM 1024Mb.
- Generic SNES USB Controller Gamepad.

The software used in this project:

Windows 10 Home

- Ruby 2.7.1p83 (2020-03-31 revision a0c7c23c9c) [x64-mingw32]
- Gosu (0.15.2 x64-mingw32)
- Visual Studio Code version 1.45.1

Linux Ubuntu 20.04 LTS

- Ruby 2.7.0p0 (2019-12-25 revision 647ee6f091) [x86_64-linux-gnu]
- Gosu (0.15.2)
- Visual Studio Code version 1.46.1

OS X El Capitan Version 10.11.6

- Ruby 2.7.1p83 (2020-03-31 revision a0c7c23c9c) [x86_64-darwin15].
- Gem 3.1.4.
- Gosu (0.15.2).
- Visual Studio Code Version: 1.47.3

Raspbian GNU/Linux 10 (buster)

- Ruby 2.5.5p157 (2019-03-15 revision 67260) [arm-linux-gnueabihf]
- Gem Version 2.7.7Gosu (0.15.2)
- Gosu (0.15.2)
- Visual Studio Code Version: 1.45.0

Methodology

After installing all the tools required to develop on Ruby and using Prototyping Model; A first GUI prototype was developed that prints on the console the buttons of the GUI screen that are simulated on the keyboard.

Figure 1

ISSN-2444-5002

In the second GUI prototype, it has the same functions as the first, adding a display to the buttons on the screen image, so now, apart from printing the buttons pressed on the console, it also shows a green box on each button indicating which one or which ones. they are pressured.

Figure 2

In the third GUI prototype, the printing of buttons on the console is eliminated to place it in the center of the control illustrated on the screen and leaving the display of the buttons of the second prototype which are simulated by the keyboard.

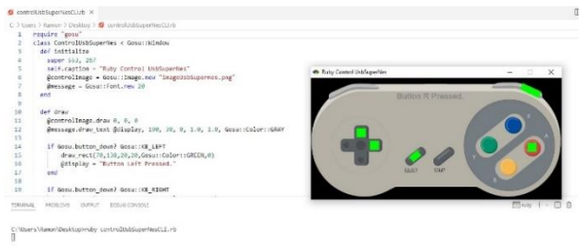

Figure 3

In the final development of the GUI, it has the same elements of the previous visual prototypes, only the keyboard simulation method is changed to SNesUSB command control and code has also been added so that it can run on different operating systems.

Figure 4

Results

Figure 5

Figure 6

The results shall be by section of the article.

ISSN-2444-5002

Appendix: Code prototype one

```
require "gosu"
class ControlUsbSuperNes < Gosu::Window
  def initialize
    super 553, 267
    @controlImage = Gosu::Image.new
"imageUsbSupernes.png"
  end

  def draw
    @controlImage.draw 0, 0, 0

    p 'Button Left Pressed.' if
Gosu.button_down? Gosu::KB_LEFT

    p 'Button Right Pressed.' if
Gosu.button_down? Gosu::KB_RIGHT

    p 'Button Up Pressed.' if
Gosu.button_down? Gosu::KB_UP

    p 'Button Down Pressed.' if
Gosu.button_down? Gosu::KB_DOWN

    p 'Button B Pressed.' if
Gosu.button_down? Gosu::KB_B

    p 'Button A Pressed.' if
Gosu.button_down? Gosu::KB_A

    p 'Button Y Pressed.' if
Gosu.button_down? Gosu::KB_Y

    p 'Button X Pressed.' if
Gosu.button_down? Gosu::KB_X

    p 'Button Select Pressed.' if
Gosu.button_down? Gosu::KB_SPACE

    p 'Button Start Pressed.' if
Gosu.button_down? Gosu::KB_RETURN

    p 'Button L Pressed.' if
Gosu.button_down? Gosu::KB_L

    p 'Button R Pressed.' if
Gosu.button_down? Gosu::KB_R

  end
end

ControlUsbSuperNes.new.show
```

Code prototype two

```
require "gosu"
class ControlUsbSuperNes < Gosu::Window
  def initialize
    super 553, 267
    self.caption = "Ruby Control
UsbSuperNes"
    @controlImage = Gosu::Image.new
"imageUsbSupernes.png"
    @message = Gosu::Font.new 20
  end

  def draw
    @controlImage.draw 0, 0, 0

    if Gosu.button_down? Gosu::KB_LEFT
        draw_rect(70,138,20,20,Gosu::Col
or::GREEN,0)
        p "Button Left Pressed."
    end

    if Gosu.button_down?
Gosu::KB_RIGHT
        draw_rect(130,138,20,20,Gosu::Co
lor::GREEN,0)
        p "Button Right Pressed."
    end

    if Gosu.button_down? Gosu::KB_UP
        draw_rect(100.7,110,20,20,Gosu::
Color::GREEN,0)
        p "Button Up Pressed."
    end

    if Gosu.button_down? Gosu::KB_DOWN
        draw_rect(100.7,167,20,20,Gosu::
Color::GREEN,0)
        p "Button Down Pressed."
    end

    if Gosu.button_down? Gosu::KB_B
        draw_rect(425,185,20,20,Gosu::Co
lor::GREEN,0)
        p "Button B Pressed."
    end

    if Gosu.button_down? Gosu::KB_A
        draw_rect(478,142,20,20,Gosu::Co
lor::GREEN,0)
        p "Button A Pressed."
    end

    if Gosu.button_down? Gosu::KB_Y
        draw_rect(383,137,20,20,Gosu::Co
lor::GREEN,0)
        p "Button Y Pressed."
    end

    if Gosu.button_down? Gosu::KB_X
        draw_rect(437.5,94,20,20,Gosu::C
olor::GREEN,0)
        p "Button X Pressed."
```

ISSN-2444-5002

```
    end

    if Gosu.button_down? Gosu::KB_SPACE
        rotate(50,230,170)
{draw_rect(224,159.5,15,23,Gosu::Color::
GREEN,0)}
        p "Button Select Pressed."
    end

    if Gosu.button_down? Gosu::KB_RETURN
        rotate(50,236,296)
{draw_rect(170,159.5,15,23,Gosu::Color::
GREEN,0)}
        p "Button Start Pressed."
    end

    if Gosu.button_down? Gosu::KB_L
        rotate(65,100,100)
{draw_rect(15,80,15,45,Gosu::Color::GREE
N,0)}
        p "Button L Pressed."
    end

    if Gosu.button_down? Gosu::KB_R
        rotate(295,100,100)
{draw_rect(321,396,15,45,Gosu::Color::GR
EEN,0)}
        p "Button R Pressed."
    end

  end
end

ControlUsbSuperNes.new.show
```

Code prototype tree

```
require "gosu"
class ControlUsbSuperNes < Gosu::Window
  def initialize
    super 553, 267
    self.caption = "Ruby Control
UsbSuperNes"
    @controlImage = Gosu::Image.new
"imageUsbSupernes.png"
    @message = Gosu::Font.new 20
  end

  def draw
    @controlImage.draw 0, 0, 0
    @message.draw_text @display, 190,
30, 0, 1.0, 1.0, Gosu::Color::GRAY

    if Gosu.button_down?
Gosu::KB_LEFT
        draw_rect(70,138,20,20,Gosu::Col
or::GREEN,0)
        @display = "Button Left
Pressed."
    end

    if Gosu.button_down?
Gosu::KB_RIGHT
        draw_rect(130,138,20,20,Gosu::Co
lor::GREEN,0)
        @display = "Button Right
Pressed."
    end

    if Gosu.button_down? Gosu::KB_UP
        draw_rect(100.7,110,20,20,Gosu::
Color::GREEN,0)
        @display = "Button Up Pressed."
    end

    if Gosu.button_down? Gosu::KB_DOWN
        draw_rect(100.7,167,20,20,Gosu::
Color::GREEN,0)
        @display = "Button Down
Pressed."
    end

    if Gosu.button_down? Gosu::KB_B
        draw_rect(425,185,20,20,Gosu::Co
lor::GREEN,0)
        @display = "Button B Pressed."
    end

    if Gosu.button_down? Gosu::KB_A
        draw_rect(478,142,20,20,Gosu::Co
lor::GREEN,0)
        @display = "Button A Pressed."
    end

    if Gosu.button_down? Gosu::KB_Y
        draw_rect(383,137,20,20,Gosu::Co
lor::GREEN,0)
        @display = "Button Y Pressed."
    end

    if Gosu.button_down? Gosu::KB_X
        draw_rect(437.5,94,20,20,Gosu::C
olor::GREEN,0)
        @display = "Button X Pressed."
    end

    if Gosu.button_down? Gosu::KB_SPACE
        rotate(50,230,170)
{draw_rect(224,159.5,15,23,Gosu::Color::
GREEN,0)}
        @display = "Button Select
Pressed."
    end

    if Gosu.button_down? Gosu::KB_RETURN
        rotate(50,236,296)
{draw_rect(170,159.5,15,23,Gosu::Color::
GREEN,0)}
        @display = "Button Start
Pressed."
    end

    if Gosu.button_down? Gosu::KB_L
        rotate(65,100,100)
{draw_rect(15,80,15,45,Gosu::Color::GREE
N,0)}
        @display = "Button L Pressed."
```

ISSN-2444-5002

```
    end

    if Gosu.button_down? Gosu::KB_R
        rotate(295,100,100)
{draw_rect(321,396,15,45,Gosu::Color::GR
EEN,0)}
        @display = "Button R Pressed."
    end

  end
end

ControlUsbSuperNes.new.show
```

USBSNES Ruby Final Code

```
require "gosu"
class ControlUsbSuperNes < Gosu::Window
  def initialize
    super 553, 267
    self.caption = "Ruby Control UsbSupe
rNes"
    @controlImage = Gosu::Image.new "ima
geUsbSupernes.png"
    @message = Gosu::Font.new 20

    if RUBY_PLATFORM =~ /win32/ || RUBY_
PLATFORM =~ /x64-mingw32/
        lsusb_results = `wmic path CIM_L
ogicalDevice where "Description like '%j
uego%'"`
        if lsusb_results.include?("juego
") || lsusb_results.include?("gamepad")
            @display = "Windows - Gamepa
d On"
        else
            @display = "Windows - No Gam
epad"
        end
    elsif RUBY_PLATFORM =~ /linux/
        lsusb_results = `lsusb`
        if lsusb_results.downcase.includ
e? "gamepad"
            @display = "Linux - Gamepad
On"
        else
            @display = "Linux - No Gamep
ad"
        end
    elsif RUBY_PLATFORM =~ /darwin/
        lsusb_results = `system_profiler
 SPUSBDataType`
        if lsusb_results.downcase.includ
e? "gamepad"
            @display = "MacOS X - Gamepa
d On"
        else
            @display = "MacOS X - No Gam
epad"
        end
    elsif RUBY_PLATFORM =~ /freebsd/
        lsusb_results = `lsusb`
        if lsusb_results.downcase.includ
e? "gamepad"
            @display = "FreeBSD - Gamepa
d On"
        else
            @display = "FreeBSD - No Gam
epad"
        end
    else
        #@display = "Unknown operating s
ystem - No gamepad detected"
    end

  end

  def draw
    @controlImage.draw 0, 0, 0
    @message.draw_text @display, 190, 30
, 0, 1.0, 1.0, Gosu::Color::GRAY

    if button_down? Gosu::GP_LEFT
        draw_rect(70,138,20,20,Gosu::Col
or::GREEN,0)
        @display = "Button Left Pressed.
"
    end

    if button_down? Gosu::GP_RIGHT
        draw_rect(130,138,20,20,Gosu::Co
lor::GREEN,0)
        @display = "Button Rigth Pressed
."
    end

    if button_down? Gosu::GP_UP
        draw_rect(100.7,110,20,20,Gosu::
Color::GREEN,0)
        @display = "Button Up Pressed."
    end

    if button_down? Gosu::GP_DOWN
        draw_rect(100.7,167,20,20,Gosu::
Color::GREEN,0)
        @display = "Button Down Pressed.
"
    end

    if button_down? Gosu::GP_BUTTON_0
        draw_rect(425,185,20,20,Gosu::Co
lor::GREEN,0)
        @display = "Button B Pressed."
    end

    if button_down? Gosu::GP_BUTTON_1
        draw_rect(478,142,20,20,Gosu::Co
lor::GREEN,0)
        @display = "Button A Pressed."
    end

    if button_down? Gosu::GP_BUTTON_2
        draw_rect(383,137,20,20,Gosu::Co
lor::GREEN,0)
        @display = "Button Y Pressed."
    end
```

ISSN-2444-5002

```
    if button_down? Gosu::GP_BUTTON_3
        draw_rect(437.5,94,20,20,Gosu::C
olor::GREEN,0)
        @display = "Button X Pressed."
    end

    if button_down? Gosu::GP_BUTTON_4
        rotate(50,230,170) {draw_rect(22
4,159.5,15,23,Gosu::Color::GREEN,0)}
        @display = "Button Select Presse
d."
    end

    if button_down? Gosu::GP_BUTTON_6
        rotate(50,236,296) {draw_rect(17
0,159.5,15,23,Gosu::Color::GREEN,0)}
        @display = "Button Start Pressed
."
    end

    if button_down? Gosu::GP_BUTTON_9
        rotate(65,100,100) {draw_rect(15
,80,15,45,Gosu::Color::GREEN,0)}
        @display = "Button L Pressed."
    end

    if button_down? Gosu::GP_BUTTON_7
        rotate(295,100,100) {draw_rect(3
21,396,15,45,Gosu::Color::GREEN,0)}
        @display = "Button R Pressed."
    end

  end
end

ControlUsbSuperNes.new.show
```

Results and conclusions

A GUI application was developed using the Ruby programming language with the GOSU library, in this application we simulate a control on the screen that when pressed on the remote is also reflected on the screen, indicating by means of green boxes which button is pressed and also indicating in the central part of the screen with text which was the button pressed, prior to the final result there were three prototypes which were giving way to the final result since in the first prototype it only showed the button simulated by the console. keyboard, in the second it illustrated the buttons on the screen and showed them on the console, and the third and last prototype added text in the central part of the screen indicating which button was pressed and reaching the end with an SNes command control USB.

References

Cooper, P. (2007). *Beginning Ruby: From Novice to Professional*. New York: Apress.

Coupe, C. (2020, 07 15). *Walkabout Shoes | Keeping Red Shoes alive since January 2014*. Retrieved from https://walkabout.mvmanila.com/

Gandy, D. (2020, 07 28). *Shoes! The easiest little GUI toolkit, for Ruby*. Retrieved from http://shoesrb.com/

Julian Raschke, J. L. (2020, 07 15). *Hello • Gosu*. Retrieved from https://www.libgosu.org

License - Visual Studio Code. (2022, 03 03). Retrieved from https://code.visualstudio.com/license

Mahadevan, S. (2002). *Making Use of RUBY*. Indianapolis, Indiana: Wiley Publishing, Inc.

RubyGems. (2022, 03 03). Retrieved from RubyGems.com

Sobkowicz, M. (2015). *Learn Game Programming with Ruby Bring Your Ideas to Life with Gosu*. United States of America: The Pragmatic Bookshelf.

TIOBE <the software quality company>. (2022, 03 03). Retrieved from https://www.tiobe.com/tiobe-index/

vscode/LICENSE.txt at main · microsoft/vscode · GitHub. (2022, 03 03). Retrieved from https://github.com/microsoft/vscode/blob/main/LICENSE.txt

ISSN-2444-5002