GoPiGo Tutorials and Resources

Contents

NOTE: All Python code is compatible with Python 3.5. This is the current version of Python on GoPiGo Raspbian for Robots. If you build a Raspberry Pi OS (Currently Buster) and install the GoPiGo3 software, you will be using Python 3.7.

Run Python Code from the Terminal

Run all Python scripts from the terminal.

```
python3 <scriptname>
```

GoPiGo Library Documentation

This is the location for the documentation on the GoPiGo Python library.

https://gopigo3.readthedocs.io/en/master/api-basic/easygopigo3.html#easygopigo3

GoPiGo Tutorials

- Video Streaming Robot
- Basic Robot Control
- GoPiGo Scratch Control Panel
- Build Your Own GoPiGo License Plate (3D Printed)

Modular Robotics GoPiGo Documentation

This has the latest documentation for the GoPiGo3. It has basic tutorials and a reference to get you started programming the robot.

https://readthedocs.org/projects/gopigo3/downloads/pdf/latest/ (pdf version)

https://gopigo3.readthedocs.io/en/latest/ (Web version)

Code Examples and Projects on the GoPiGo

On your GoPiGo3, there are some code examples.

1) In File Manager → go to **\Dexter\GoPiGo3** or **\Dexter\GoPiGo**
2) **\Dexter\GoPiGo3\Software\Python** is a good place to start.

GoPiGo projects explained in more detail.

https://www.dexterindustries.com/GoPiGo/projects/python-examples-for-the-raspberry-pi/

GoPiGo3 Python Tutorials

https://edu.modrobotics.com/course/view.php?id=16

Sample GoPiGo Python3 Programs in WNCCNASA GitHub

There are sample Python3 programs in the GitHub Code folder. They use the Easy GoPiGo library. Movements and the led's on the GiPiGo are a good place to start programming.

https://gopigo3.readthedocs.io/en/master/api-basic/easygopigo3.html#easygopigo3 (Easy GoPiGo Library)

Learning Python

There are hundreds of tutorials and videos on learning Python. Here is a good resource for learning Python.

https://www.w3schools.com/python/default.asp

General Raspberry Pi Resources

Run this command at the terminal to determine which Raspberry Pi model you have.

```
cat /sys/firmware/devicetree/base/model
```

MagPi magazine is free on pdf. They have several good general books on the Raspberry Pi.

1. https://magpi.raspberrypi.org/books (Raspberry Pi Books)
2. https://magpi.raspberrypi.org/issues (MagPi Magazine)
3. Conquer the Command Line (MagPi Book about the bash shell on a Pi)

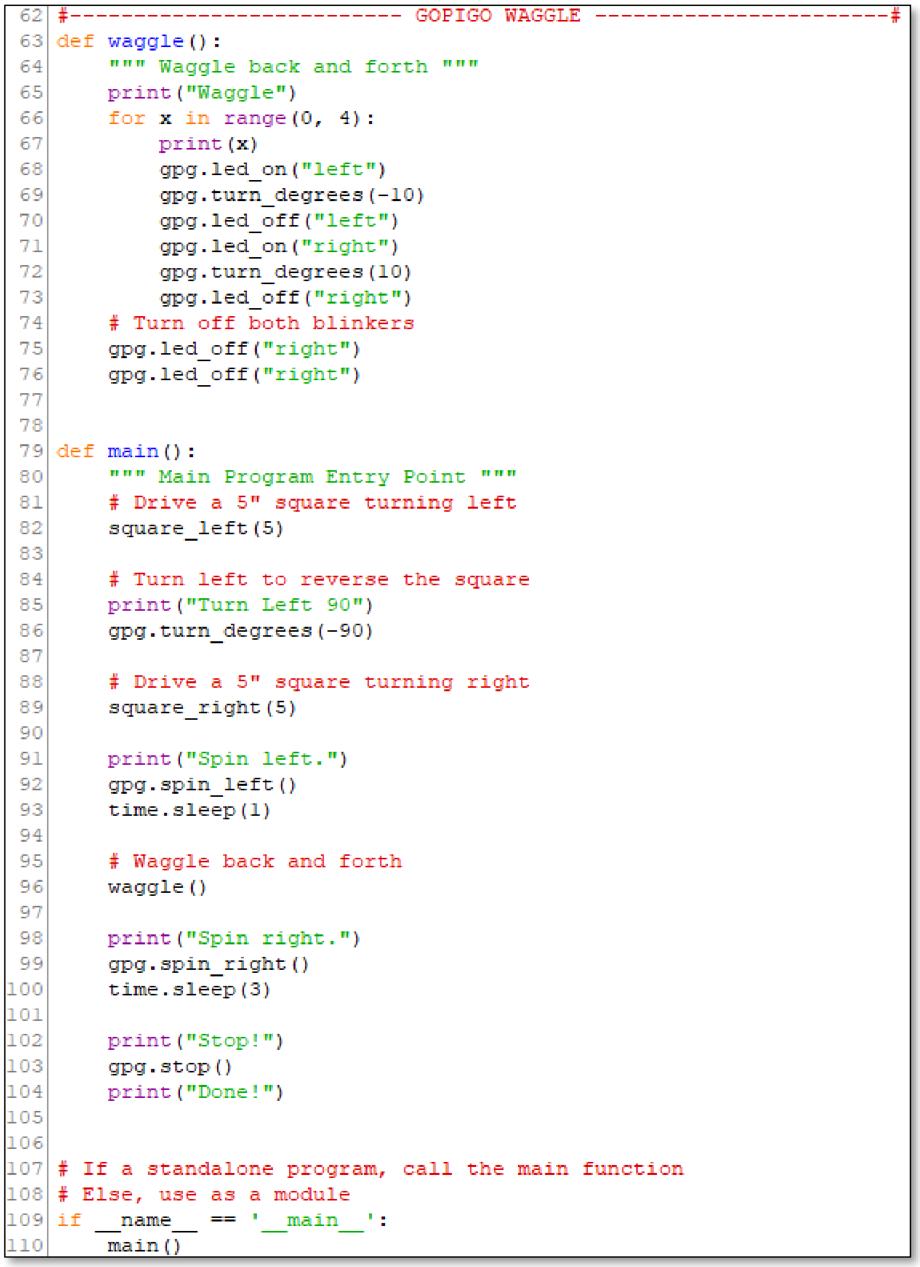
Easy Movement Tutorial

It is time to do some tutorials to learn the GoPiGo library better.

Learning points

- Functions, Loops, Movement, GoPiGo Blinkers

```python
#!/usr/bin/env python3
"""
    Name: easy_movement.py
    Author: William A Loring
    Created: 09-18-21 Revised:
    Purpose: Demonstrate a sampling of GoPiGo dead reckoning movements
"""
# This uses the EasyGoPiGo3 library.  You can find more information on the li
# here:  https://gopigo3.readthedocs.io/en/master/api-basic/easygopigo3.html#

# Import the time library for the sleep function
import time
# Import GoPiGo3 library
from easygopigo3 import EasyGoPiGo3

# Create an instance of the GoPiGo3 class
# GPG is the GoPiGo3 object used to access methods and properties
gpg = EasyGoPiGo3()


#------------------------- SQUARE RIGHT ------------------------#
def square_right(distance):
    """
        Drive a right square based on the distance argument
    """
    # Loop four times,  Loop starts at 0,
    # Ends at 1 less than the last number
    # The loop increments 0, 1, 2, 3
    print("Square Right")
    for x in range(0, 4):
        # Print the loop counter
        print(x)
        gpg.led_off("right")
        gpg.drive_inches(
            distance,   # How far to drive in inches
            True        # Blocking, nothing else can happen while moving
        )
        gpg.led_on("right")
        # Turn right 90 degrees, positive number is right
        gpg.turn_degrees(90)
    # Turn both blinkers off
    gpg.led_off("right")
    gpg.led_off("left")


#-------------------------- SQUARE LEFT ------------------------#
def square_left(distance):
    """
        Drive a left square based on the distance argument
    """
    print("Square Left")
    for x in range(0, 4):
        print(x)
        gpg.led_off("left")
        gpg.drive_inches(distance, True)
        gpg.led_on("left")
        # Turn left 90 degrees, - is left
        gpg.turn_degrees(-90)
    gpg.led_off("left")


```



```python
#-------------------------- GOPIGO WAGGLE -----------------------#
def waggle():
    """ Waggle back and forth """
    print("Waggle")
    for x in range(0, 4):
        print(x)
        gpg.led_on("left")
        gpg.turn_degrees(-10)
        gpg.led_off("left")
        gpg.led_on("right")
        gpg.turn_degrees(10)
        gpg.led_off("right")
    # Turn off both blinkers
    gpg.led_off("right")
    gpg.led_off("right")


def main():
    """ Main Program Entry Point """
    # Drive a 5" square turning left
    square_left(5)

    # Turn left to reverse the square
    print("Turn Left 90")
    gpg.turn_degrees(-90)

    # Drive a 5" square turning right
    square_right(5)

    print("Spin left.")
    gpg.spin_left()
    time.sleep(1)

    # Waggle back and forth
    waggle()

    print("Spin right.")
    gpg.spin_right()
    time.sleep(3)

    print("Stop!")
    gpg.stop()
    print("Done!")


# If a standalone program, call the main function
# Else, use as a module
if __name__ == '__main__':
    main()
```

Driving School Remote Control Console

You have seen this before in Intro to Robotics. Here it is again!

You will find all of the assignments from Intro to Robotics in the mBot folder in GitHub. These can be used to see the shape that you want your code to be.

Requirements

1. Each movement will have its own function. This is demonstrated in the example program: square_left, square_right, orbit_right, and waggle.
2. **DRY:** Don't Repeat Yourself (Reuse functions, build bigger functions from smaller functions.
3. Create a console based menu to choose which function you wish to perform.

Stage 1

1. **Square** - Trace the path of a square that is 1-foot square. It will start and end in the same place and the same orientation.
2. **Rectangle** - Trace the path of a rectangle that is 1-foot x 2-foot. It will start and end in the same place and the same orientation.
3. **Sentry** - Trace a 1-foot square around an object. Start the square one way, then turn around and go back the other way. Return to the beginning point and orientation.
4. **Retrace** - Move in a 1-foot square forward, then move in reverse to retrace that same square backwards to the beginning point and orientation. One solution would be to build a Reverse block that uses negative numbers for motor movement.
5. **ForwardReverse** - Move forward 12", turn 180°, move backwards 12" (which will be the same direction), turn 180° again, and then continue to move forward 12". The robot should move in one direction, but do part of the trip moving backwards.
6. **OrbitRight and OrbitLeft -** Orbit your GoPiGo around a central point.

Stage 2

1. **Pentagon** - Move your robot in a 12" pentagon. Start and end in the same place and the same orientation.
2. **Octagon** - Move your robot in a 12" octagon. Start and end in the same place and the same orientation.
3. **Equilateral Triangle** – Move your robot in a 12" equilateral triangle. Start and end in the same place and the same orientation.

4. **5-Point Star** – Teach your robot to trace a 5-point 12” star. Start and end at the same location and orientation. Look up the inside angle and subtract from 180 degrees.
5. **3-PointTurn** - Using 3 or more turns, teach your robot how to make a 3-point turn, like a regular car. You don’t have to do curves, you can use straight angles if you wish.

Stage 3

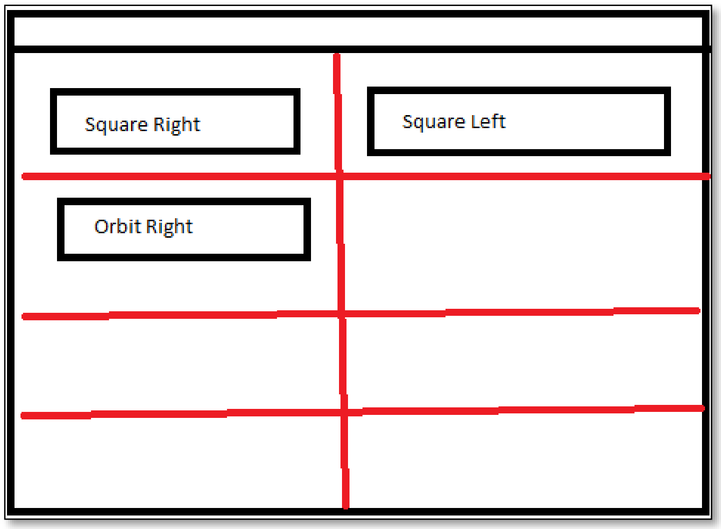
1. **Circle** - your robot will trace the path of a circle that is 1 foot in diameter. It will start and end in the same location, and in the same orientation.
   HINT: Adjust the power of your left and right motors to create a left half circle block and right half circle block. Put those together to make your curved shapes.
2. **S-Shape** – your robot will trace two half-circles to create an S-shaped curve. Your robot will start and end in the same orientation, and the two half-circles will be the same size.
3. **Figure-8** - Move in a figure-8 shape.

Driving School Remote Control Tkinter

Copy your console menu based program and convert it to Tkinter OOP. You can use all of the functions you created with the console program as methods in your Tkinter Driving School.

Draw It

When designing a GUI, start with a sketch. The example sketch shows a grid of 2 columns and multiple rows. You can have as many columns and rows as you wish.



Pseudocode

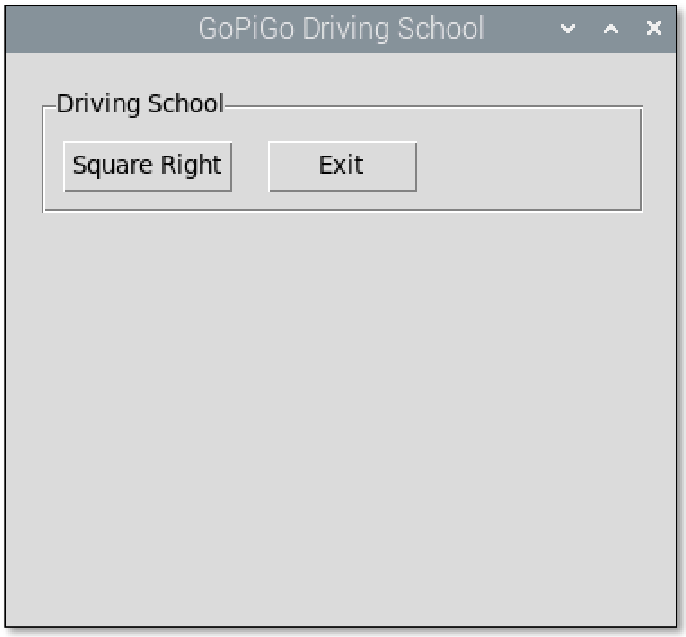
```
Create interface
Add functions from console program
Attach methods to buttons
```

Tkinter Driving School Sample Code

Some sample code to get you started.

```python
#!/usr/bin/env python3
"""
    Name:    rc_tkinter_driving_school.py
    Author:  William A Loring
    Created: 12/04/2021
    Purpose: GoPiGo Tkinter based driving school program
"""

# ------------------------------------------------
# History
# ------------------------------------------------
# Author     Date           Comments


from tkinter import *         # Import tkinter for GUI
from tkinter.ttk import *     # Add ttk themed widgets
import sys                    # Used to as sys.exit() to exit the program
import easygopigo3 as easy    # Import EasyGoPiGo3 library


class GoPiGoGUI:
    def __init__(self):
        """ Initialize the program """
        self.gpg = easy.EasyGoPiGo3()  # Create EasyGoPiGo3 object
        self.gpg.set_speed(200)        # Set initial speed
        self.window = Tk()             # Create Tkinter window
        self.window.title("GoPiGo Driving School")
        # Set the window size and location
        # 375x320 pixels in size, location at 50x50
        self.window.geometry("375x320+50+50")
        self.create_widgets()  # Create and layout widgets
        mainloop()             # Start Tkinter program main loop

#---------------------------- CREATE WIDGETS -----------------------------#
    def create_widgets(self):
        """ Create and layout widgets """
        # Create main label frame to hold remote control widgets
        self.main_frame = LabelFrame(
            self.window,
            text="Driving School",
            relief=GROOVE)

        # Pack the frame to the width (X) of the window
        self.main_frame.pack(fill=X, padx=10, pady=(10))
        # Keep the frame size regardless of the widget sizes
        self.main_frame.pack_propagate(False)

        # Create widgets and attach them to the correct frame
        btn_square = Button(
            self.main_frame,
            text="Square Right",
            command=self.square_right)

        btn_exit = Button(
            self.main_frame,
            text="Exit",
            command=self.exit_program)

        # Grid the widgets
        btn_square.grid(row=0, column=0,)
        btn_exit.grid(row=0, column=1)

        # Set padding between frame and window
        self.main_frame.pack_configure(padx=10, pady=(10))
        # Set padding for all widgets in frames
        pad = 10
        for child in self.main_frame.winfo_children():
            child.grid_configure(padx=pad, pady=pad)

#------------------------- DRIVE RIGHT SQUARE -----------------------------#
    def square_right(self):
        for i in range(4):
            self.gpg.drive_inches(12)
            self.gpg.turn_degrees(90)

#----------------------------- EXIT PROGRAM --------------------------------#
    def exit_program(self):
        print("\nExiting")
        sys.exit()


# Create program object
gopigo_gui = GoPiGoGUI()
```

Example run.



Random Numbers

Create and display random numbers on your GoPiGo terminal.

Obstacle Avoidance

Another blast from the past! Base this on the mBlock program samples in the repository.

Minimum Requirements

Start with a simple Obstacle Avoidance program. Add in other requirements as you work on this project.

- Use the shape of your mBot mBlock or Arduino code to help your GoPiGo learn autonomous obstacle avoidance.
- Use your servo to look right and left to gauge the distance.
- Use random numbers to change the obstacle avoidance.