piRover Builds with K2

piRover Drive Module   Rev 1.0

Goal:

In this activity you will be introduced to using Python modules. Here each Python file as code supporting a specific object or purpose. A main module is created and import statements are used to access code in the associated module files.

The instructor will start with a tic tac toe game. The initial purpose is to demonstrate more advanced code. Next, the instructor will deconstruct the single module solution into multiple modules.

Next, you will use your keyboard and piRover drive fake file create during the last activity to create a drive solution with two modules. The keyboard file in the main module that will call functions imported from the piRover drive module.

Finally, you will work to integrate move and speed code into the piRover Drive module to create the desired behavior.

Prerequisites:

This assessment requires content and code solutions from the following:

- Drive Interface Design

Performance Outcomes:

1. Create a main solution module
2. Create other modules supporting specific objects or purpose.
3. Import functions from modules
4. Replace piRover Drive function "stubs" with GPIO code creating motion and speed.

Resources:

1. See prerequisite lessons

Materials:

1. piRover
2. keyboard_drive.py
3. piRover_drive_fake.py

piRover Builds with K2

Set Up

1. Prepare your workspace for this project.

   a. Connect to your piRover using VNC. Access your piRover folder and launch VS Code.

   b. Create a **11.piRoverDriveModule** directory.

   c. Copy your **keyboard_drive.py** file from the prior activity here.

   d. Copy your **piRover_drive_fake.py** file from the prior activity here.

Part 1 – tic tac toe Review and Deconstruction

1. Follow along with the instructor as he or she investigates this game solution.

2. Follow along with the instructor as the game solution is deconstructed to include a board.py module.

Part 2 – Keyboard Module

3. The class will discuss modifications to the key commands based on the design work during the last activity.

4. The instructor will demonstrate his solution using more complex commands and string functions. Python Lists are introduced here.

5. Open you keyboard_drive.py file and revise as demonstrated to connect the piRover_drive_fake.py file.

6. Test this revised solution to verify the appropriate print functions are being called from the piRover_drive_fake.py module when a key combination is entered by the user.

7. The revised requirements for the **keyboard_drive.py** module are below.

   a. Welcomes the user to the keyboard move solution

   b. Provides directions to the user including a list of keyboard commands

   c. Captures the command

   d. Use a selection structure to call the appropriate speed and motion functions from the piRover_Drive module.

   e. Includes a method for the user to exit the solution.

   f. Includes a help function to redisplay directions (see b).

   g. No move action is required until the piRover_drive_fake is replaced below.

piRover Builds with K2

Part 3 – piRover Drive

8. Copy your **piRover_drive_fake.py** code to piRover_drive.py

9. Review prior move solutions and copy the required GPIO code to produce motion.

10. Review the PWM activity. Included a **left_speed** and **right_speed** PWM object in this module. Recall the requirement to use the **global** keyword.

11. Include an **init()** function and revise the keyboard file to call this function.

12. As a class discuss accelerate and decelerate function. How will this speed value be adjusted? Revise your drive module as required.

13. Continue to test and revise your code.

14. Submit your current solution at the end of class time using the link provided.

15. Continue to develop your solution. Submit your final keyboard drive solution with then end of the week submission.

Submission:

16. Submit your revised keyboard_drive.py and piRover_drive.py files. Be sure these two files create a functional solution. The instructor will download and run to test.