piRover Builds with K2

Project 03 – Remote Drive   Rev 1.0

Goal:

The course started with you evaluating the piRover using the Yahboom smart phone app. You will complete the course by implementing your own version of this controller.

In part 1, you will investigate the piRover Bluetooth module provided by the instructor. The module will provide message inputs to your application when a smart phone is connected and application buttons are pressed. You will use input and selection patterns developed in earlier lessons to decode the direction message and call functions from your piRover Drive module.

In part 2, you will work with the instructor to create additional modules to control the buzzer and the front servo. These modules will be integrated into your main remote drive application so that user presses will activate features.

Finally, in part 3, you will create an LED module on your own. This module provides the final functionality for the Yahboom smartphone application. Part 3 the final assessment for the course. An additional document will be provided. Do your best and do you own work. Do not assist others.

Prerequisites:

This assessment requires content and code solutions from the following:

- Keyboard Drive

Performance Outcomes:

1. Create a solution using multiple modules
2. Interface with Bluetooth to enable remote drive.
3. Create a module to support buzzer operation
4. Create a module to support servo operation
5. Create a module to support LED operation

Resources:

1. See prerequisite lessons

Materials:

1. piRover
2. piRover_Bluetooth.py (provided)
3. piRover_drive.py (created during prior lesson)
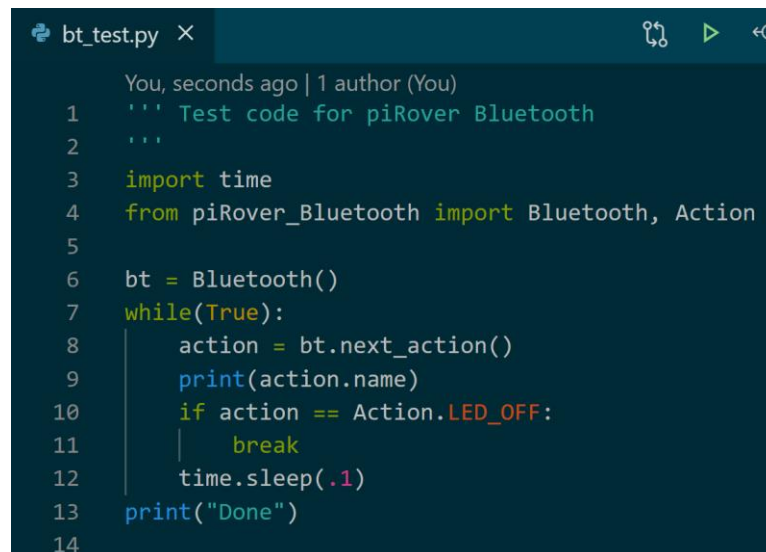4. Smart phone

piRover Builds with K2

Part 1 – 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 **12.RemoteDrive** directory.

   c. Create a new **remote_drive.py** file in the 12.RemoteDrive directory. This is the solution file for this project.

   d. Locate the **piRover_drive.py** code in the 11.piRoverDriveModule directory. Copy this file the new 12.RemoteDrive directory.

   e. Download the Bluetooth file into the solution directory.
      wget -O piRover_Bluetooth.py http://bit.ly/K2-piRover-bluetooth

Part 2 – piRover_Bluetooth Investigation

1. Create a new **bt_test.py** file in the solution directory.

2. Enter the following code to test the piRover Bluetooth interface and messaging.

```python
''' Test code for piRover Bluetooth
'''
import time
from piRover_Bluetooth import Bluetooth, Action

bt = Bluetooth()
while(True):
    action = bt.next_action()
    print(action.name)
    if action == Action.LED_OFF:
        break
    time.sleep(.1)
print("Done")
```

3. Connect your smart phone to the piRover and run the test code. Note line 10 where the action value is checked for a specific value (Action.LED_OFF)

4. Open the piRoverBluetooth.py file and review the Action definitions. They are shown in Figure 1 on the following page for your reference. You will implement piRover code to respond to these user messages. (Note: OUTFIRE is not used.)

piRover Builds with K2

```
class Action(Enum):
    STOP            = 0
    FORWARD         = 1
    BACKWARD        = 2
    LEFT            = 3
    RIGHT           = 4
    LEFT_ALT        = 5
    RIGHT_ALT       = 6

    BEEP            = 8
    SPEED_UP        = 9
    SPEED_DOWN      = 10
    SERVO_LEFT      = 11
    SERVO_MID       = 12
    SERVO_RIGHT     = 13
    LED_OFF         = 14
    LED_RED         = 15
    LED_GREEN       = 16
    LED_BLUE        = 18
    OUTFIRE         = 19
```
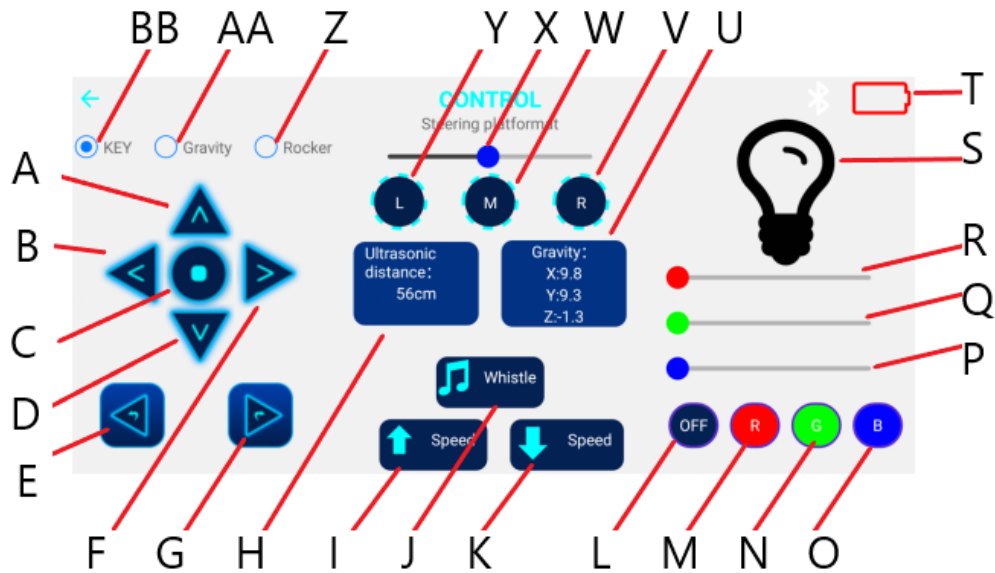
Figure 1– piRover Bluetooth Actions

Figure 2 – Yahboom Smartphone App Interface

piRover Builds with K2

Part 3 – Bluetooth Drive

1. Open the piRover_Drive.py file and modify the init() function name to drive_init() as shown below. The renaming is required due to later modules also requiring an initialize function. Each must have a unique name to prevent "name collisions".

```
def drive_init(): ⋯

def forward(): ⋯

def backward(): ⋯
```

2. Review the keyboard_drive.py located in your prior lesson. The initial remote_drive.py solution for this project is almost identical to this solution except the input is captured from the piRover_Bluetooth module rather than the keyboard.

3. Enter the code in the remote_drive.py file that enables you to drive the piRover using the smart phone application. This requires code similar to the keyboard_drive reviewed in the prior step along with the Bluetooth integration demonstrated in Part 2.

4. You will need to decide on how left vs. left_alt and right vs. right_alt are implemented.

piRover Builds with K2

5. The requirements for the remote_drive.py file are:

   a. Imports the piRover_Bluetooth module. See the example in Part 2.

   b. Imports the piRover_drive module. See your keyboard_drive solution.

   c. Welcomes the user to the Remote Drive solution and specifies that the smart phone app must be installed and connected to Bluetooth.

   d. Continually captures next actions from the Bluetooth module, selects appropriate move based on the action, and calls the require move method in the piRover Drive module.

6. Run, test, and debug your solution.

Part 4 – Bluetooth Buzzer

7. Add a piRover_buzzer.py module to your solution.

8. Implement the following member functions.

```python
piRover_buzzer.py
1    ''' Module to control piRover buzzer
2    '''
3
4    def buzzer_init():
5        #initializes buzzer GPIO
6        pass
7
8    def buzzer_on():
9        #turns buzzer on
10       pass
11
12   def buzzer_off():
13       #turns buzzer off
14       pass
15
16   def buzzer_beep():
17       #Buzzer beeps for 1 sec"
18       pass
19
20   def buzzer_beeps():
21       #Buzzer continuously beeps
22       #PWM port is required
23       pass
```

9. Complete the buzzer module by adding the required GPIO code. Ignore buzzer_beeps initially. This function requires that a PWM port is used. Get basic code functional and then add PWM pulse functionality as time permits.

piRover Builds with K2

10. Integrate with the Remove Drive solution by importing the module and specifying the buzzer_beep function if the beep action is selected.

11. Run, test, and debug your code.

Part 5 – Bluetooth Servo

12. Add a piRover_servo.py module to your solution.

13. The instructor will present detail on servo control and assist with creating the required code that implements the servo functions below.

```python
''' Module to control piRover front servo
'''

def servo_init():
    #initializes servo GPIO
    pass

def servo_left():
    #turns servo to the left
    pass

def servo_center():
    #turns servo to the center
    pass

def servo_right():
    #turns servo to the right
    pass
```

14. Integrate with the Remove Drive solution by importing the module and specifying the servo_left, servo_center, and servo_right functions if one of the servo actions is selected by the user.

15. Run, test, and debug your code.

Part 6 – Bluetooth LED

16. This section is the final assessment for the course. You will create an LED module on your own. An additional document will be provided specifying requirements. Do your best and do you own work. Do not assist others.

Part 7 – Project Extension

17. User a combination of keys or key timing to specify an exit condition for the application. Exit the application when the user enters this key combination. Timing should be used to distinguish an exit from a normal combination of key presses.

18. Add a special feature. Be sure to include comments in your code identifying the special feature and how it is used.

Evaluation

19. See the grading rubric on the following page.

piRover Builds with K2

Section	Description	Points	Score
P03 - Remote Drive - Project Evaluation			
Main	Files remote_drive, piRover_Bluetooth, piRover_drive, piRover_buzzer, piRover_servo and piRover_led submitted in a zipped solution folder.	4	4
	Docstring identifies solution and file function. Name and date included.	2	2
	Solution imports required modules	4	4
	Solution calls initialization code required for each module.	4	4
	Solution welcomes user and specifies smartphone connection requirements.	2	2
	Solution continuously captures Bluetooth input.	4	4
	Solution call appropriate module function based on Bluetooth input.	4	4
	No Python syntax errors	2	2
	Valid logic and structure used. Runs as specified.	2	2
Drive	docstring identifies file solution.	2	2
	piRover drive contains required global constants and variables.	2	2
	drive_init initializes GPIO and PWM	4	4
	Motion functions provide basic drive moves	4	4
	Speed functions enable speed up and speed down	2	2
	No Python syntax errors	2	2
	Valid logic and structure used. Runs as specified.	2	2
Buzzer	docstring identifies file solution.	2	2
	piRover buzzer contains required global constants and variables	2	2
	buzzer_init initializes GPIO	4	4
	Buzzer functions implemented as defined	2	2
	Beeps function modifies initialization and functions as required	2	2
	No Python syntax errors	2	2
	Valid logic and structure used. Runs as specified.	2	2
Servo	docstring identifies file solution.	2	2
	piRover servo contains required global constants and variables	2	2
	servo_init initializes GPIO and PWM	4	4
	Servo functions implemented as defined	2	2
	No Python syntax errors	2	2
	Valid logic and structure used. Runs as specified.	2	2
LED	docstring identifies file solution.	2	2
	piRover LED contains required global constants and variables	2	2
	led_init initializes GPIO and PWM	4	4
	LED functions implemented as defined	6	6
	PWM ports enable LED dimming as required	2	2
	No Python syntax errors	2	2
	Valid logic and structure used. Runs as specified.	2	2
Other	Bluetooth key combination causes program to exit	2	2
	Special Feature defined and implemented	2	2
TOTAL		100	100

piRover Builds with K2

Submission:

1. Review the grading rubric on the prior page. Partial credit is awarded. Use comment lines to identify issues.

2. Zip your solution file folder with all required files.

3. Submit your zip file to the Moodle link by the deadline specified.