

piRover Builds with K2

P03 – Remote Drive – Servo Extension

Rev 1.0

Goal:

The goal of Project 03, the final project, is to complete the course by implementing your own version of this smartphone controller. Part 2 of the project adds the buzzer and servo functionality to the drive functionality created in part 1. This document contains the details on servo use along with required code to implement the required left, center, and right servo functions. You will create a `piRover_servo` module and integrate this module and functionality into your P03 Remote Drive project.

Prerequisites:

This is an extension to P03 Remote Drive. Drive and buzzer functionality should be complete.

- P03 Remote Drive

Performance Outcomes:

1. Create a module to support servo operation
2. Interface with Bluetooth to enable remote servo actions.

Resources:

1. See prerequisite lessons

Materials:

1. piRover
2. `remote_drive.py` (See P03 document)
3. `piRover_buzzer.py` (See P03 document)
4. `piRover_Bluetooth.py` (See P03 document)
5. `piRover_drive.py` (See P03 document)
6. Smart phone

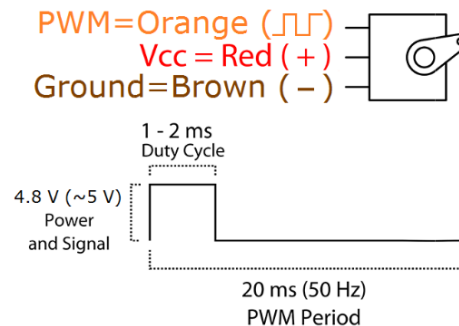
Part 1 – Set Up

1. Prepare your workspace for this project extension.
 - a. Connect to your piRover using VNC. Access your piRover folder and launch VS Code.
 - b. Create a new **`piRover_servo.py`** file in the `12.RemoteDrive` directory.

piRover Builds with K2

Part 2 – Servo Investigation

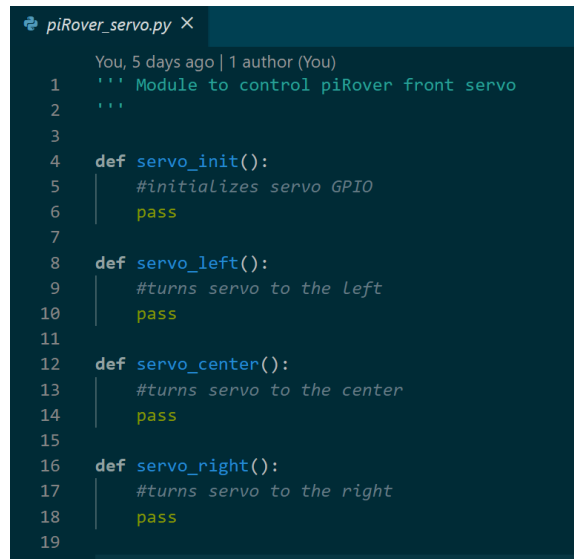
1. Begin by viewing a data sheet for the [SG90 Micro Servo](#). This small, inexpensive servo are common in many small robotics and drone projects. The code and output signal required to drive this servo is the same as larger servo devices.
2. Review the timing diagram for the servo signal to determine the duty cycle for the left, center, and right positions.



Position "0" (1.5 ms pulse) is middle, "90" (~2 ms pulse) is all the way to the right, "-90" (~1 ms pulse) is all the way to the left.

3. To control the front servo on your piRover the left, mid, and right actions will be used. You'll need to create a PWM port on the required GPIO pin to enable the servo motion.
4. Review the documentation in the [Yahboom Expansion Board Manual](#) to determine the GPIO pin used to control the front servo.
5. The piRover_servo module interface is shown on page 3. You'll need to create the PWM port on the appropriate pin and then control the servo position using the following duty cycle values. The servo should be centered during your initialization process.
 - a. Left: $1\text{ms}/20\text{ms} = 5\%$
 - b. Mid: $1.5\text{ms}/20\text{ms} = 7.5\%$
 - c. Right: $2\text{ms}/20\text{ms} = 10\%$

piRover Builds with K2



```
piRover_servo.py X
You, 5 days ago | 1 author (You)
1  ''' Module to control piRover front servo
2  ...
3
4  def servo_init():
5      #initializes servo GPIO
6      pass
7
8  def servo_left():
9      #turns servo to the left
10     pass
11
12  def servo_center():
13     #turns servo to the center
14     pass
15
16  def servo_right():
17     #turns servo to the right
18     pass
19
```

Figure 1 - piRover Servo Module Interface

6. Open the piRover_servo.py file created during set up. Create the module including the functions shown above.
7. Integrate the piRover_servo.py module into the Remote Drive solution so that servo buttons control the front servo position.
8. Submit this work with the final PO3 project submission.