

piRover Builds with K2

Digital Inputs - Introduction

Rev 1.1

Overview:

In this virtual lab, the instructor will review GPIO inputs covered in the RAM155 course. You'll then create a simple logic switch using a voltage divider circuit connect to one of the control board's servo channels. You'll measure the voltage applied to the GPIO input to validate that it does not exceed the Raspberry Pi's 3.3-volt limitation and measure the input voltage with the switch open and closed.

You'll continue by writing GPIO code that reads the switch status and controls outputs accordingly.

Prerequisites:

Prior to beginning the instruction provided in this lesson you must have completed the following:

1. Digital Outputs

Performance Outcomes:

1. Identify voltage levels required for valid GPIO inputs.
2. Convert from a 5-volt logic level to a 3.3-volt logic level system using a voltage divider.
3. Construct basic digital switch using 10K ohm voltage divider.
4. Recognize an active-low switching circuit.

Resources:

1. [Raspberry Pi GPIO Electrical Specification](#)

Materials:

1. piRover with fully charged battery
2. RAM155 Digital Multimeter
3. RAM205 Parts Kit

Directions:

1. The instructor will review the electrical specification document provided in the resources section, specifically reviewing the circuit diagram of Figure 1.
2. Review the circuit of Figure 2. What is the signal voltage if S1 is open? What is the signal voltage if S1 is closed? Give the specification in Figure 1 in the specifications document, will this provide valid

piRover Builds with K2

digital signals to the GPIO port enabling you to read GPIO.HIGH and GPIO.LOW values in code?

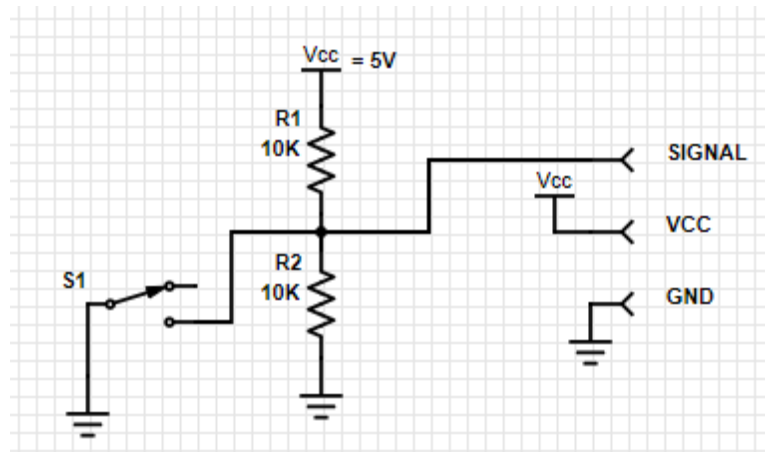
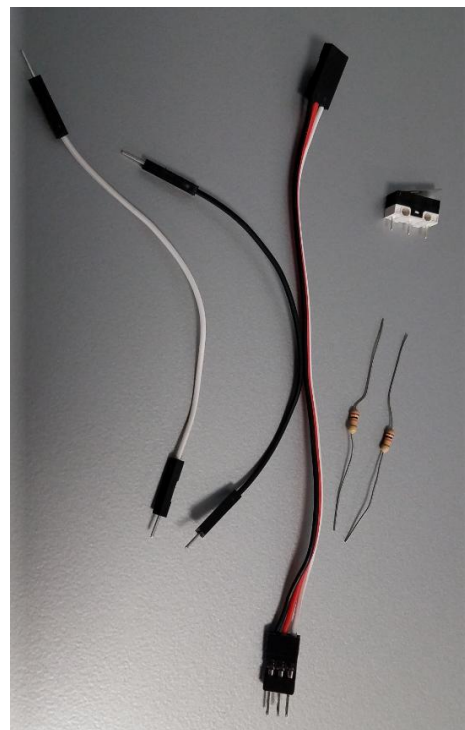


Figure 2

3. Complete the following table based on your analysis of the voltage divider circuit in Figure 2.

S1 State	V_{CC}	V_{SIGNAL}	V_{IL}	V_{IH}	Logic
Open					
Closed					

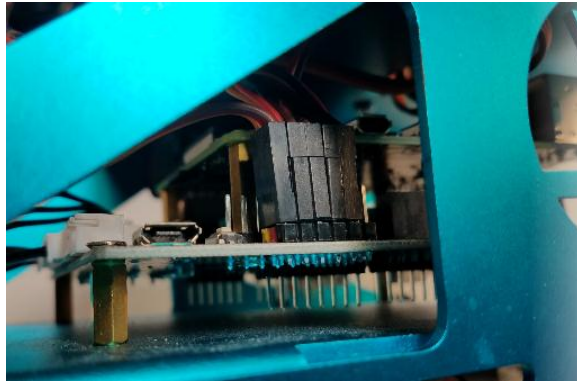
4. Next, you'll connect and test the circuit of Figure 2. You'll need the following parts from your RAM205 kit.
- Servo extension cable – Male to Female
 - 10K ohm resistors (x2)
 - Micro switch
 - Connector wires (x2) – male to male



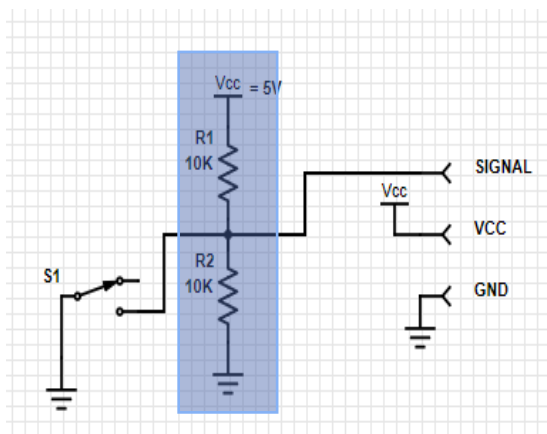
5. Connect the servo cable to position 5 on the servo extension header on

piRover Builds with K2

the Yahboom controller board. Recall that the servo cable on position 4 is now driving the amber warning light.

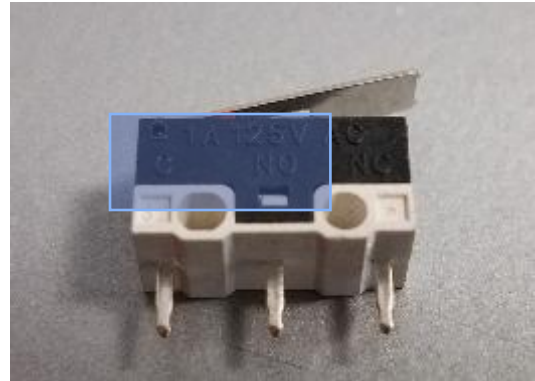
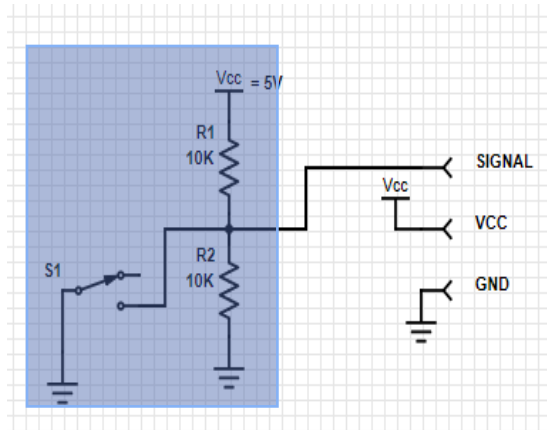


6. Insert the servo extension cable into the breadboard and connect only the 10K voltage divider network. Do not connect to the signal input at this point. Use your DVM to verify the required voltage level.

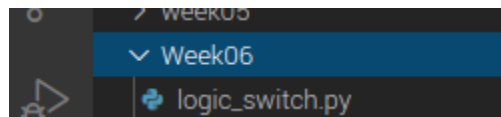


piRover Builds with K2

- Now connect the switch using the normally open connections so that making the switch pulls the signal to ground. Use your voltmeter to verify.



- Once you have verified the function of the digital switch, connect the signal to the signal pin of the servo extension cable.
- Locate the expansion board manual or GPIO documentation for the controller board. What GPIO pin will be configured as an input to read the status of the switch?
- Boot the piRover. Open your connection and launch VS Code.
- Create a week06 folder and add a logic_switch.py file.

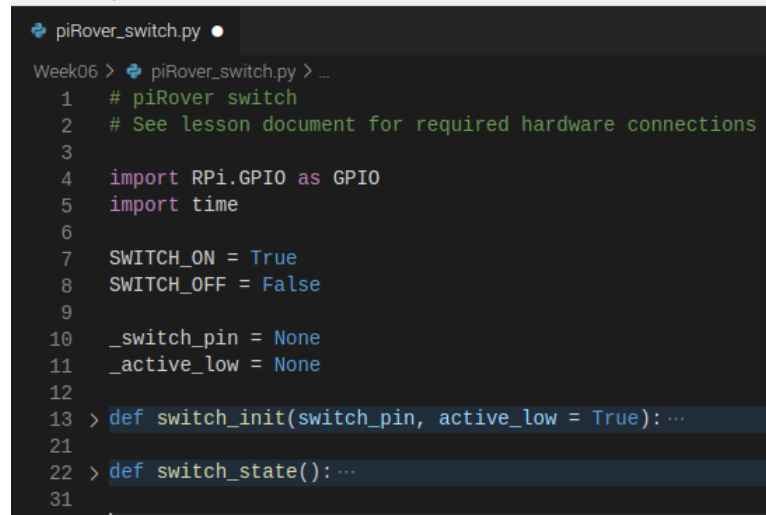


- Add the following code to the logic_switch.py file. Continue creating the solution by setting the buzzer pin to an output and the switch pin to an input. Be sure to turn the buzzer off initially.

```
Week06 > logic_switch.py > ...
1  ''' Demonstrating GPIO input using logic switch
2  Keith E. Kelly
3  2/12/21
4  '''
5  #import required libraries
6  import RPi.GPIO as GPIO
7  import time
8
9  SWITCH_PIN = 22
10 BUZZER_PIN = 24
11
12 # Configure GPIO setting
13 GPIO.setwarnings(False)
14 GPIO.setmode(GPIO.BOARD)
15
```

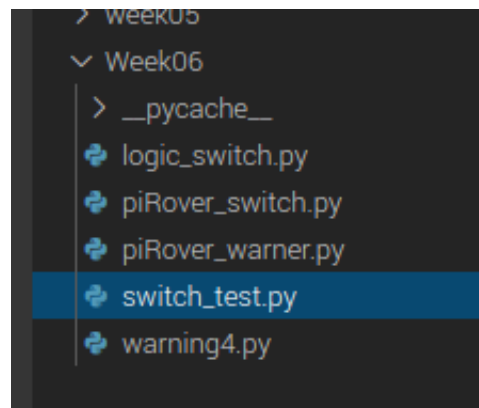
piRover Builds with K2

13. Finally, create a loop that constantly polls the switch input and updates the buzzer to sound when the switch is depressed. The instructor will review multiple versions of this code.
14. The next step is to create a module for the switch like the warner module created in the last activity. Follow along with the instructor as the piRover_switch.py module is implemented. The outline is shown below.



```
piRover_switch.py
Week06 > piRover_switch.py > ...
1  # piRover switch
2  # See lesson document for required hardware connections
3
4  import RPi.GPIO as GPIO
5  import time
6
7  SWITCH_ON = True
8  SWITCH_OFF = False
9
10 _switch_pin = None
11 _active_low = None
12
13 > def switch_init(switch_pin, active_low = True):...
21
22 > def switch_state():...
31
```

15. Copy the piRover_warner module from week05 and create a new switch_test.py solution. This program will use both the piRover_switch and piRover_warner modules to demonstrate switch functionality.



16. Enter the switch test code provided on the following page. Run the code to test. Note how simple this main code becomes as the complexities of both input and output are encapsulated in the piRover modules.

piRover Builds with K2

```
switch_test.py
Week06 > switch_test.py > ...
1  ''' Demonstrating GPIO using logic switch and warner
2  Keith E. Kelly
3  2/12/21
4  '''
5  #import required libraries
6  from piRover_switch import *
7  from piRover_warner import *
8  import time
9
10 SWITCH_PIN = 22
11
12 switch_init(SWITCH_PIN)
13 warn_init()
14
15 while True:
16     if switch_state() == SWITCH_ON:
17         warn_start()
18     else:
19         warn_stop()
20         time.sleep(.5)
21
```

Assessment:

Demonstrate the switch_test.py to the instructor for a lab check. Submit the following files along with other required files in your weekly submission at the bottom of this week's Moodle section.

- piRover_switch.py
- piRover_warner.py
- switch_test.py