piRover Builds with K2

piRover – Infrared Input Example Rev 1.0

Overview:

In the last activity you learned to use a pushbutton connected to a GPIO input pin to control your code. This activity is an extension from pushbutton with similar code but using the infrared detector connected to the controller board as the input device.

Note: This activity is normally limited to a demonstration by the instructor. To demonstrate this code, you must have an infrared remote controller – for example, a TV remote control. A jumper must also be installed on the controller board to connect the infrared sensor to the GPIO pin. See the hardware document for additional detail.

The requirements for the solution are listed below.

- The user will see a welcome message indicating that this is the Infrared as an Input example.
- The initial state of the LED module is off, and the user is prompted to press a button on his or her infrared remote control. When the infrared signal is sensed on the GPIO pin, the LED module is toggled on and off.

Prerequisites:

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

1. Pushbutton

Performance Outcomes:

1. Locate hardware components on the controller board using the provided documentation.
2. Install a jumper.
3. Use the GPIO.input() function to read the logic level on a GPIO port configured as an input.
4. Control code execution based on GPIO input.

Resources:

1. How Remote Controls Work

Materials:

1. piRover
2. jumper
3. Infrared remote control

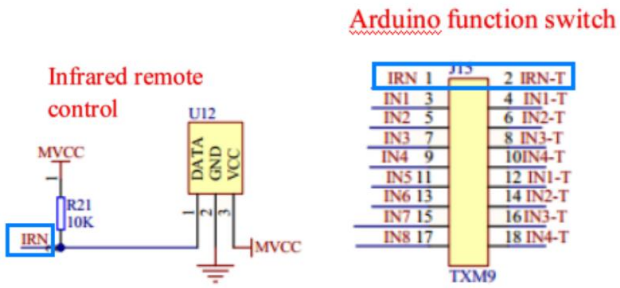
piRover Builds with K2

Part 1 – Set Up

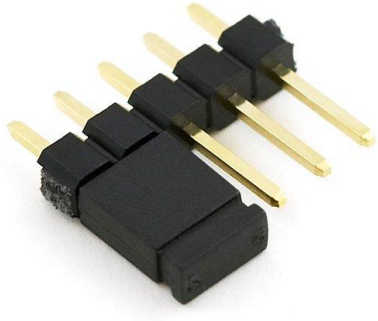
1. Prepare your workspace for this activity.
    a. Connect to your piRover using VNC.
    b. Access your piRover folder
    c. Launch VS Code
    d. Create a **05.InfraredInput** directory
    e. Create a new **infrared_input.py** file in the 05.InfraredInput directory.

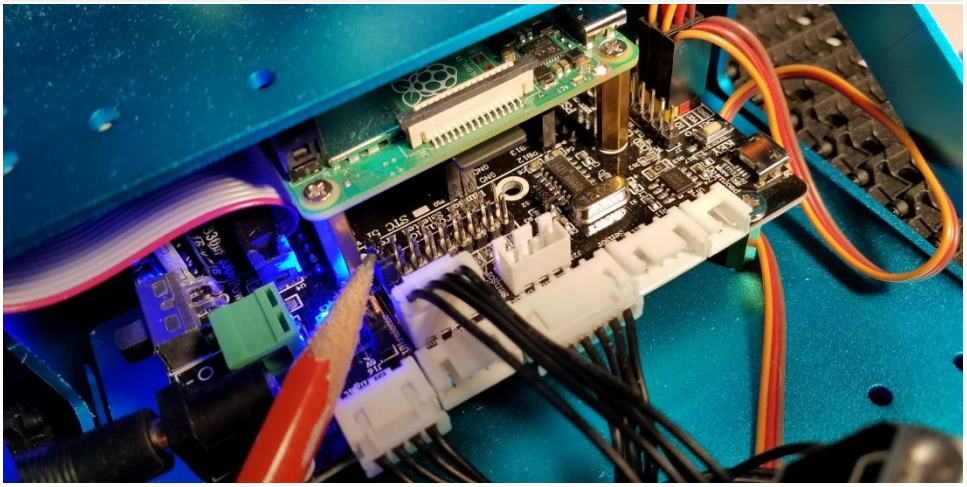
Part 1 – Investigate the hardware

2. In this activity you will be using an infrared remote controller to provide an input signal to the piRover. An infrared remote controller is not included in your kit, but any TV remote control should work as an input device.
3. With the instructor, review the function of an infrared remote control using this resource - How Remote Controls Work.
4. In this GPIO activity, the controller boards infrared sensor is connected to a GPIO pin and will apply high or low voltage onto the GPIO pin as the infrared pulse train is detected.
5. The infrared sensor is located in the back of the controller board. Use the Yahboom Expansion Board Manual to locate this component.
6. Review the documentation in the to determine which GPIO pin is connected to the infrared receiver. Note that a jumper is required in the first position of the "Arduino function switch" header



7. Several jumpers were provided in the kit. Install a jumper in the first position as shown on the following page.





8. Review the hardware documentation to determine the pin and GPIO port used for the infrared input. Table 1 entries are missing below. You should be able to determine the values at this point using the documentation.

Input	Board Pin	GPIO Reference
Infrared Receiver		

Table 1

9. Your goal in this activity is to toggle the LED when any activity is seen on the infrared input. The system can be programmed to read individual remote control buttons but this is beyond the scope of this introductory course.

Part 2 – Investigate the software – infrared_input.py

1. With the infrared_input.py file open in VS Code, follow along with the instructor as this solution is built.

infrared_input.py

```python
''' Demonstrating infrared as input
Note: This demo requires an infrared device
to test. Additionally, a jumper is required on the
Arduino function switch - see hardware documentation.

Keith E. Kelly
10/10/20
'''
#import required libraries
import RPi.GPIO as GPIO
import time

#create constants to represent piRover LED pin numbers
RED_PIN = 15
GREEN_PIN = 13
BLUE_PIN = 18
#create constants to represent piRover pushbutton pin number
IR_PIN = 3

# Configure GPIO setting
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)

# Set pin LED pins as output
GPIO.setup(RED_PIN, GPIO.OUT)
GPIO.setup(GREEN_PIN, GPIO.OUT)
GPIO.setup(BLUE_PIN, GPIO.OUT)
# Set IR pin as input
GPIO.setup(IR_PIN, GPIO.IN)

```

```
print("This solution demonstrates the IR sensor as an input.")
print("You need an IR transmitter like tv remote.")
print("A jumper is required at first of the Arduino function sw
print()
print("Press a button on the remote to light the LEDs")

while True:
    #get state of IR in and update LEDs
    state = GPIO.input(IR_PIN)
    print(state)
    #wait for pin to go low
    #pulses so we need to read frequently
    print("Waiting for IR pulse")
    while state == 1:
        state = GPIO.input(IR_PIN)
        time.sleep(.1)
    print("Pulse detected")
    #IR key was press so light LED for 1 sec
    GPIO.output(RED_PIN, True)
    GPIO.output(GREEN_PIN, True)
    GPIO.output(BLUE_PIN, True)
    time.sleep(1)
    #turn off and start again
    GPIO.output(RED_PIN, False)
    GPIO.output(GREEN_PIN, False)
    GPIO.output(BLUE_PIN, False)
    time.sleep(1)
```

2. Run the infrared_input.py code. Does it function?

Assessment:

There is no assessment associated with this activity. The code provides another example of configuring and using a GPIO port as an input to your code.