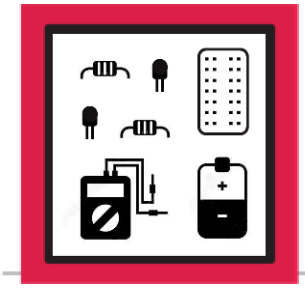


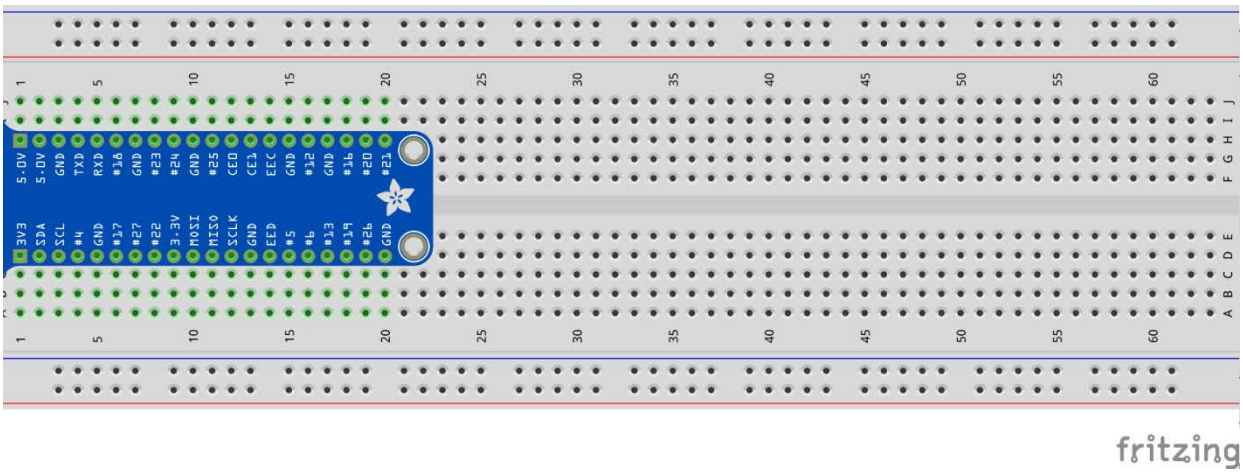
LESSON B-13

ACTIVITY #1:
LEVEL SHIFTER & THE RGB LED

In this activity, you will install and connect the level shifting IC, connect its output to a GPIO input, and build an RGB LED circuit. You will also build a small program to test everything out before adding the IR sensors in Activities #2 and #3.

Step #1

The circuit for this lesson will be made of almost entirely different components from previous lessons. The first step will be to ensure the Raspberry Pi is powered down so you can remove any components from previous lessons. Remove all components from the breadboard except for the wedge and ribbon cable. When completed your breadboard should look like this:

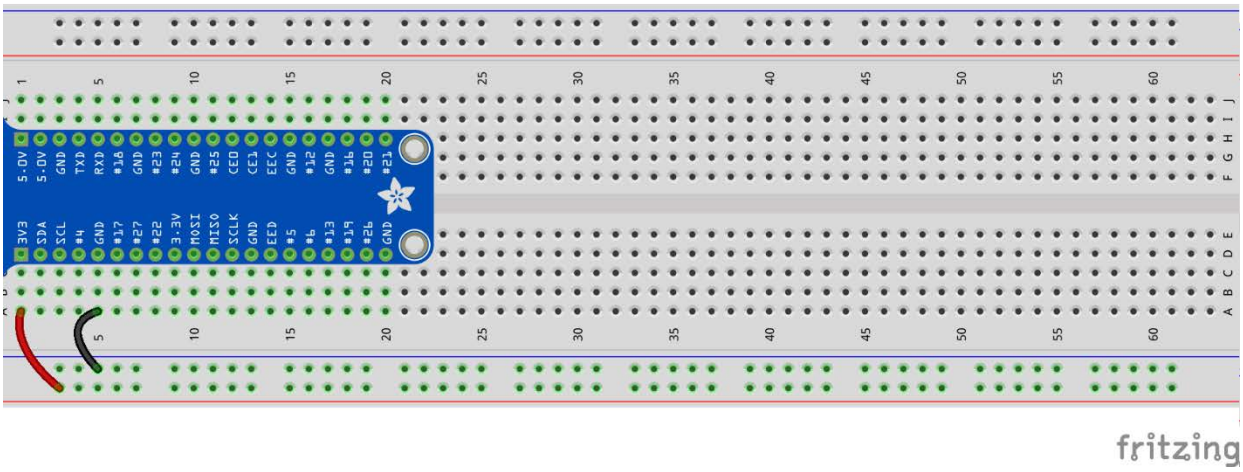


Step #2

The next step will be to add 3.3V power and ground to the P1 side buses. Connect two short jumper wires between the points below:

3.3V – connect A1 to P1-3

Ground – connect A5 to N1-5



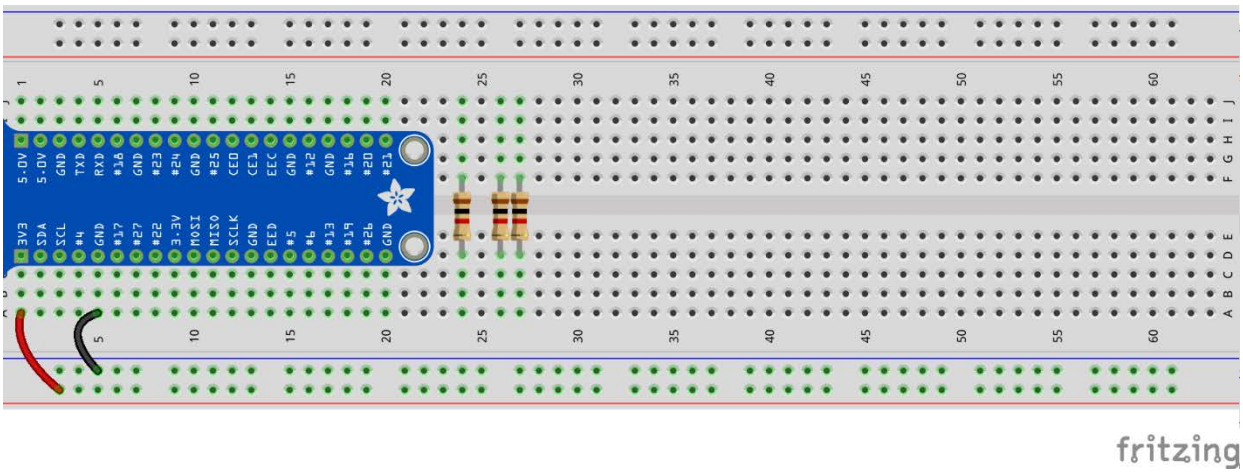
Step #3

You will now add the current limiting resistors for the RGB LED. You will be using 1K-Ohm resistors to reduce the current flowing through the RGB LED, as the LED elements are already very bright even with the small amount of current allowed by the 1K-Ohm resistors. Add three 1K-Ohm resistors between these three sets of points:

D24 to F24

D26 to F26

D27 to F27



These resistors must be very close together due to the length of the legs of the RGB LED. Inspect the legs of each resistor closely to ensure no legs are touching any others before proceeding to the next step.

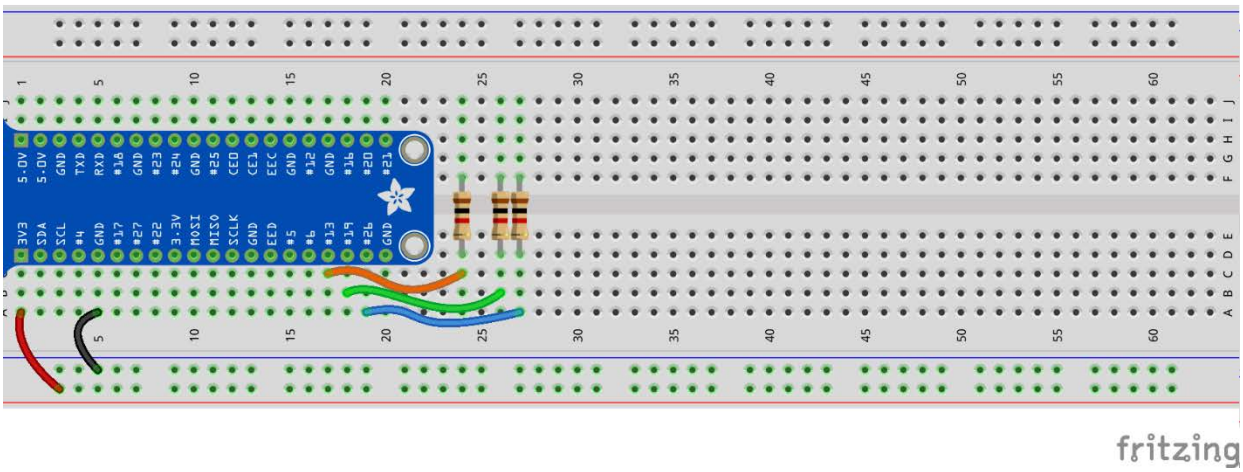
Step #4

You are now ready to add the GPIO output wires that will drive the red, green, and blue elements of the RGB LED. Connect three short jumper wires between the following points:

GPIO13 – C17 to C24

GPIO19 – B18 to B26

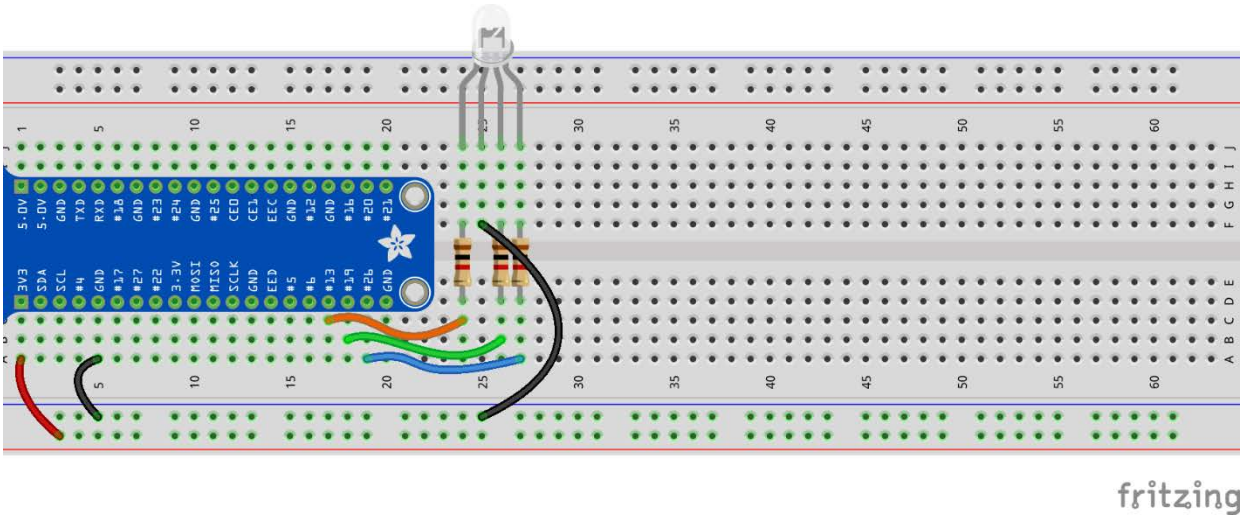
GPIO26 – A19 to A27



Step #5

It's now time for the RBG LED and its common cathode ground connection. Remember that all color elements of your RGB LED share one cathode or ground connection, so it only takes one ground connection for all three colors of your LED to have a path to ground. The longest leg of the LED is the common cathode. Insert the common cathode leg into J25, making sure to rotate the LED so it's anode legs connect to J24, J26, and J27.

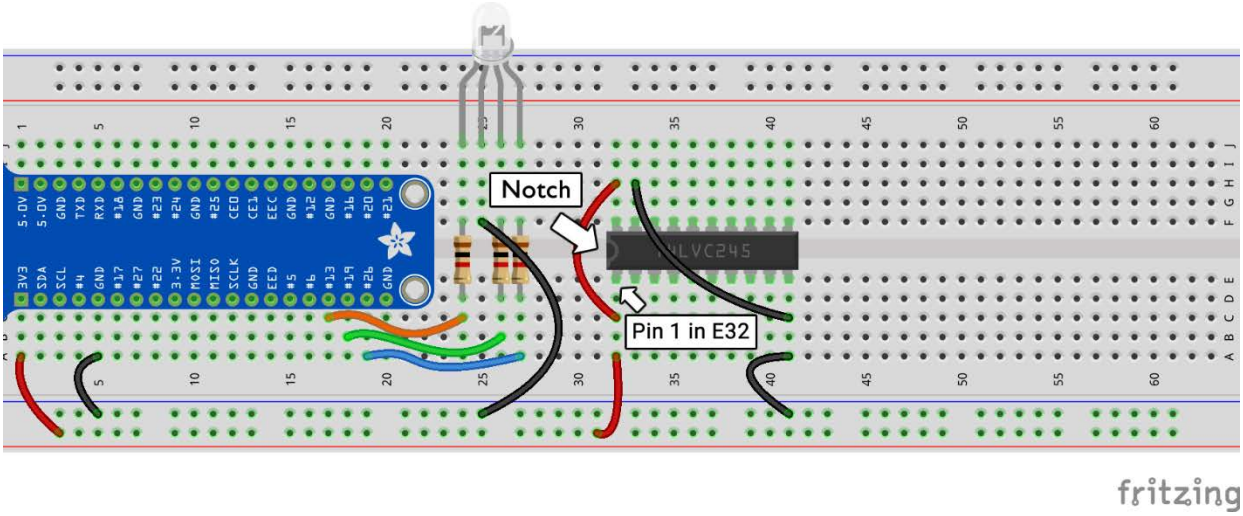
During this step you will also install the ground connection for the RGB LED. Insert a short jumper wire between breadboard points N1-25 and F25:



Step #6

Next, the 74LVC245 level shifting IC needs to be installed so you can convert 5V signals down to 3.3V, making them safe for the Raspberry Pi GPIO pins. The level shifting IC has 10 pins per side so it can be a little difficult to get properly aligned, but make sure all pins are properly aligned in the breadboard before applying pressure across the top of the IC to seat all of the pins. The IC's pins are very thin and can easily be damaged by improper insertion into the breadboard. If you are at all unsure about this process, please watch the video on the Level B Resource Page prior to attempting installation.

With pin 1 located at E32, insert the 74LVC245 across the center of the breadboard in rows 32 through 41.



Install four short jumper wires between the following locations to supply power and ground to the IC:

3.3V - P1-31 to A32

3.3V – C32 to H32

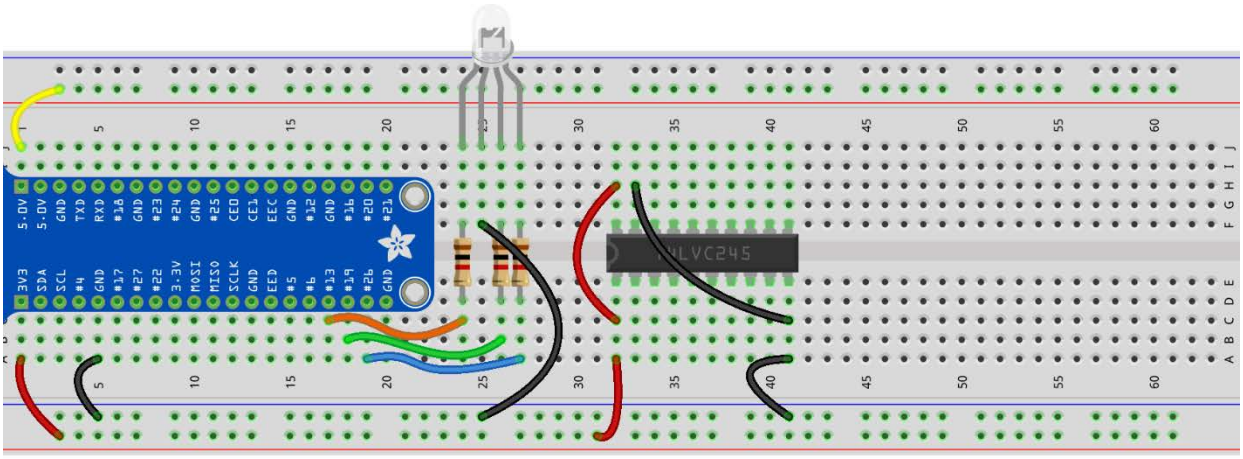
Ground – N1-41 to A41

Ground – C41 to H33

The IC is now connected to power and ground and it has been configured to accept 5V inputs on the A side and output 3.3V signals on the B side.

Step #7

Make 5V available on the P2 power rail so it can be used to test the level shifter as well as power the infrared sensors that will be added in upcoming activities. Connect a short jumper wire from J1 to P2-3.



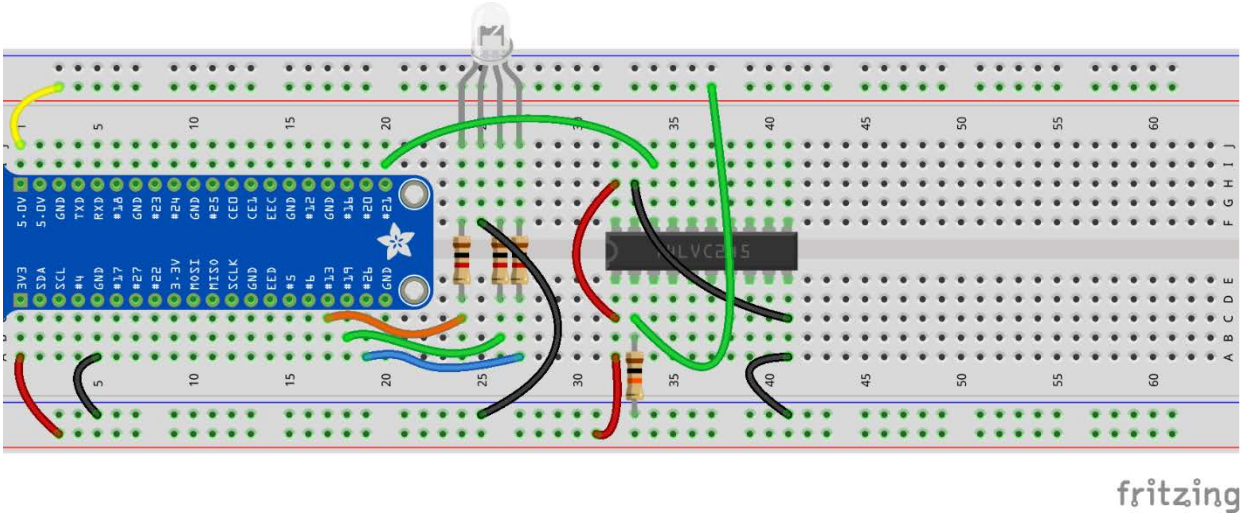
Step #8

You are now ready to connect an input and output channel to the level shifter. The input of the 74LVC245 will float when not pulled up or down. You will use a 10K-Ohm resistor between the input and ground to ensure that the 74LVC245 sees a low unless we pull the input high using a connection to 5V. Make the following connections:

10K-Ohm resistor – N1-33 to B33

Output – long jumper wire from I20 to I34

Input – long jumper wire from C33 to P2-37



The input to the level shifter will now be held high by a direct connection to the 5V power rail. When disconnected the pull-down resistor will take over and the input will be grounded.

Step #9

Double check all of your connections in the last step and power on your Raspberry Pi. It's now time to write a program that will change the color of the RGB LED based on whether the level shifter is receiving a high or a low. Open Thonny and create a new program that consists of the following lines:

```
import RPi.GPIO as GPIO
import time

rgb = [13,19,26]
red = 13
green = 19
blue = 26

GPIO.setmode(GPIO.BCM)
GPIO.setup(rgb, GPIO.OUT)
GPIO.setup(21, GPIO.IN)

try:
    while True:
        if GPIO.input(21) == False:
            GPIO.output(green, GPIO.LOW)
            GPIO.output(red, GPIO.HIGH)
        else:
            GPIO.output(red, GPIO.LOW)
            GPIO.output(green, GPIO.HIGH)
        time.sleep(.1)

except KeyboardInterrupt:
    GPIO.cleanup()
```

There are no new concepts above, but here is an overview of everything happening in the program in case you have any questions:

The RPi.GPIO and time modules are imported. A variable called **rgb** is created that contains all output pins so they can be setup using only one line. The variables **red**, **green**, and **blue** are set equal to the GPIO pin numbers that control each of those colors in the LED. Color names can then be used throughout the program instead of using the GPIO pins numbers.

The GPIO pin numbering is set to BCM, each of the pins listed in **rgb** is designated as an output, and pin 21 is declared as an input. The try/except format is used to catch keyboard exceptions and a **GPIO.cleanup** runs if an exception is encountered.

The **`try`** loop uses a **`while True:`** loop to keep checking the input over and over. If the input on **`GPIO21`** is low or False, then the green element will turn off and red will illuminate. The else condition will trigger if **`GPIO21`** is high or True during a check. In that case the red element will turn off and the green element will turn on. The **`time.sleep`** command is added to slow the loop checks down to avoid unnecessarily checking of the GPIO which can lead to unnecessary load, and heat, on the processor of the Raspberry Pi.

Step #10

Run the program. The LED will initially be green to indicate that **`GPIO21`** is high or 3.3V coming from the level shifter IC.

Carefully disconnect the jumper wire at location P2-37 and the input will go low, indicated by the LED turning red.

Reconnect the jumper wire to P2-37 and the LED will change back to green to indicate a high is present on the input.

Remove and reinstall the connection at P2-37 a few more times to ensure the program and circuit are acting as expected. If so, feel free to proceed to the next activity where you will reconfigure this circuit to get 5V input from an IR Obstacle sensor.

If the circuit or program is not behaving as outlined above, shut down the Pi and recheck all wiring and the IC's pin 1 orientation. If everything looks good on the circuitry, then proceed to reboot the Pi and double-check your program. Do not continue to the next activity until this program and circuit are behaving as expected.