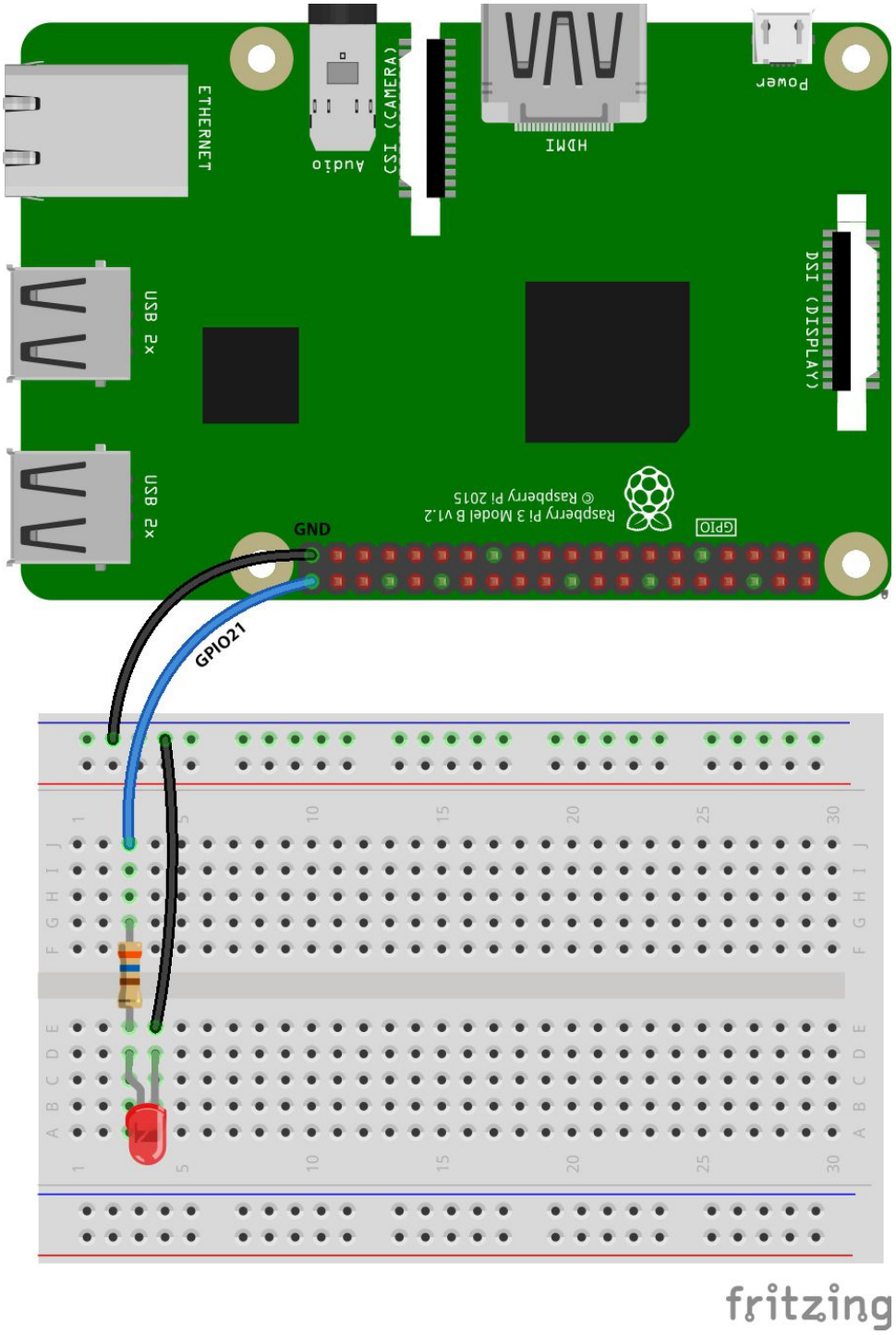
Making Sense - Reference Sheet
SFPC Detroit 2019

LED Circuit



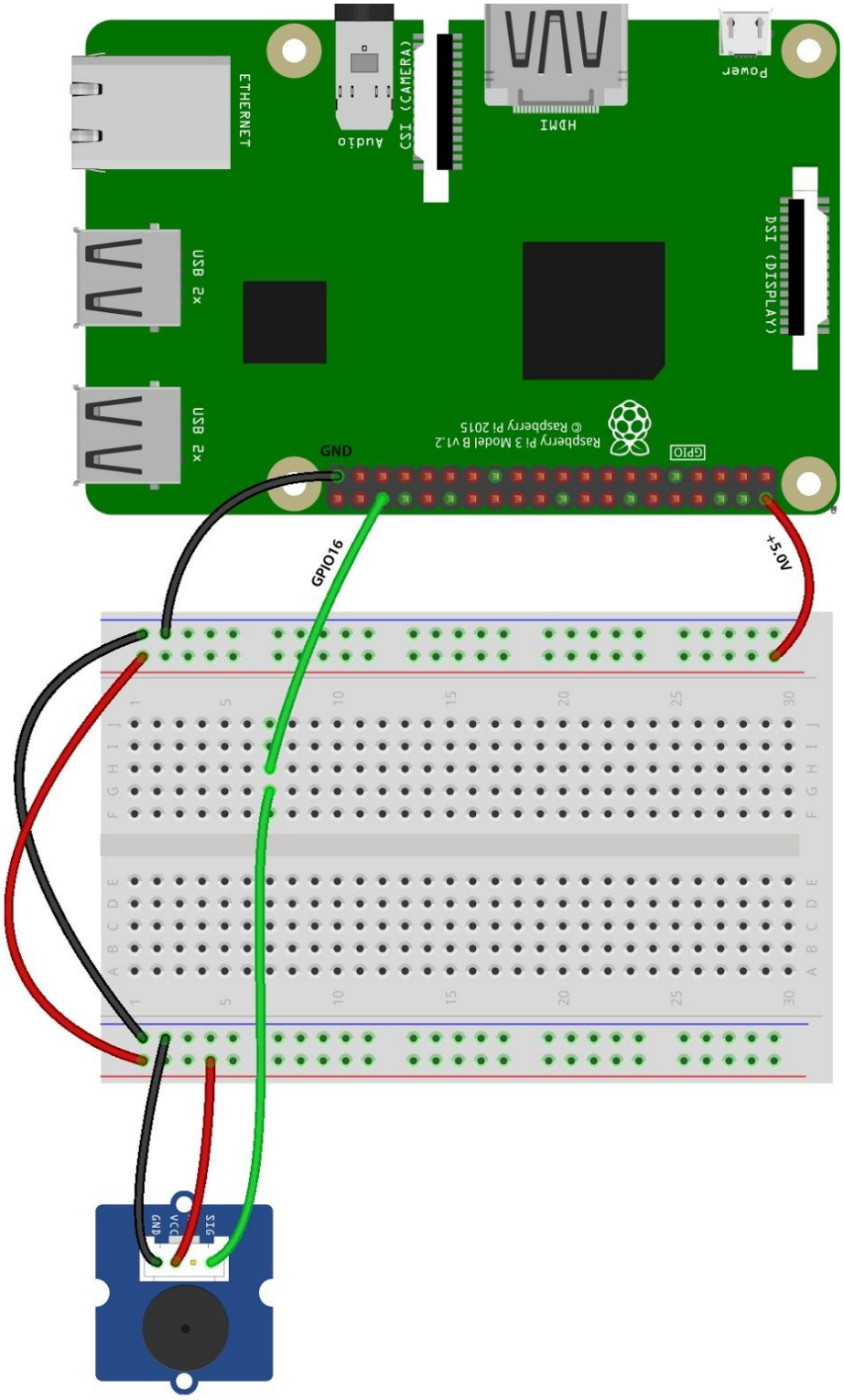
LED Code

```
from gpiozero import LED
from time import sleep

led = LED(21)

while True:
    led.on()
    sleep(.5)
    led.off()
    sleep(.5)
```

Code Breakdown

`from gpiozero import LED`	Imports the LED functions from the GPIO library
`from time import sleep`	Imports the sleep functions from the time library
`led = LED(21)`	Initializes pin 21 as an LED and defines variable name "led"
`while True:`	Run the indented code forever
`  led.on()`	Turns "led" on
`  sleep(.5)`	Waits 0.5 seconds before next line of code
`  led.off()`	Turns "led" off
`  sleep(.5)`	Waits 0.5 seconds before next line of code, in this case back to led.on()

Button Circuit



***IMPORTANT NOTE!**
Always connect GND first and +5.0v last.
Always double check your wiring is correct before connecting +5.0v.

Otherwise you could fry your Pi!

LED + Button Code

```
from gpiozero import LED, Button
from time import sleep

led = LED(21)
button = Button(16)

button.when_pressed = led.off
button.when_released = led.on
```

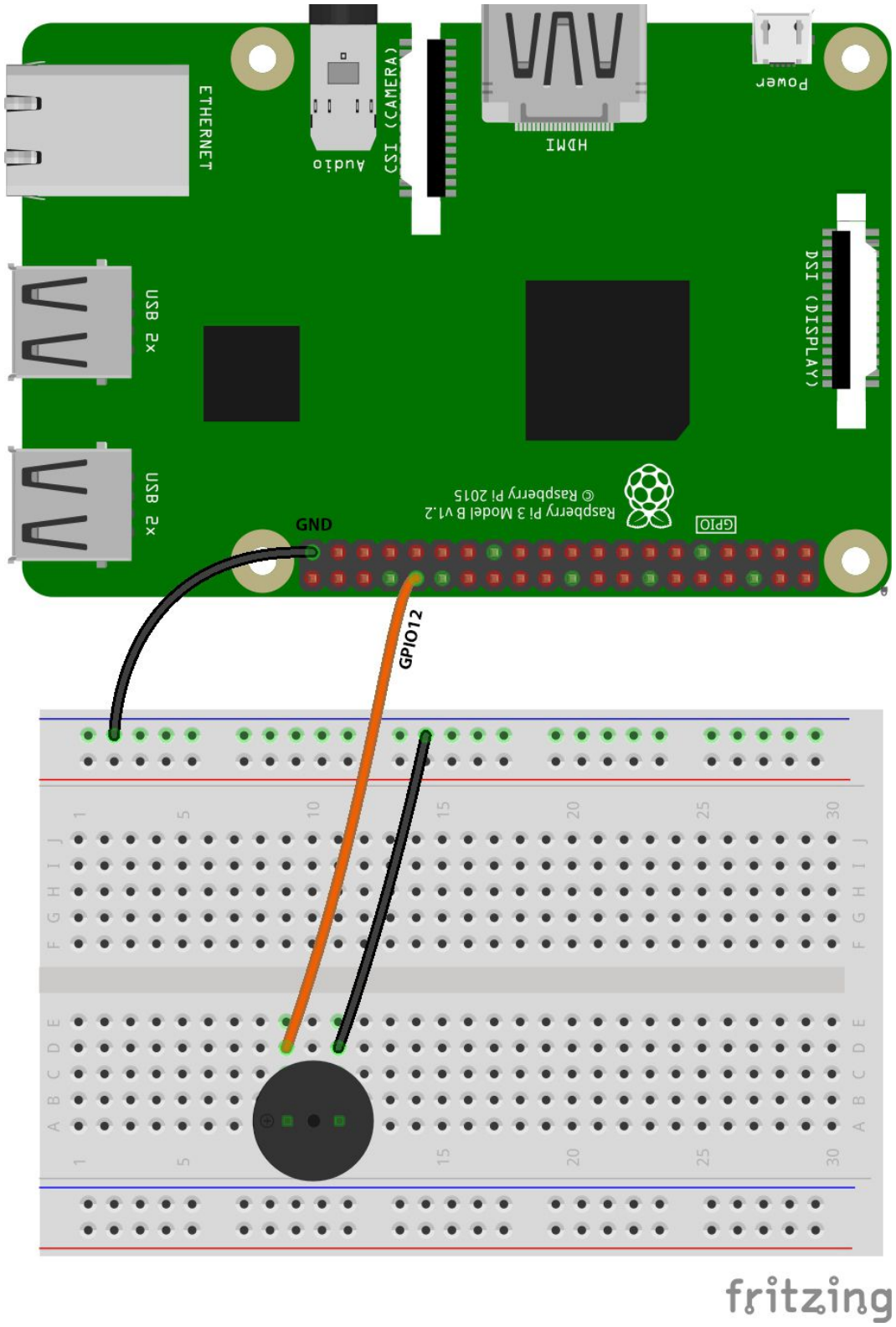
Code Breakdown

`from gpiozero import LED, Button`	Imports the LED and Button functions from the GPIO library
`from time import sleep`	Imports the sleep functions from the time library
`led = LED(21)`	Initializes pin 21 as an LED and defines variable name "led"
`button = Button(16)`	Initializes pin 16 as a Button and defines variable name "button"
`button.when_pressed = led.off`	When button pin value goes high, turn led off
`button.when_released = led.on`	When button pin value goes low, turn led on *See comment below about why this seems backwards

Note on why capacitive touch sensors output high when touched

Capacitive touch sensors output high to the signal pin when untouched (the circuit is closed and complete). When touched, the circuit is grounded by your finger, and the button outputs low to the signal pin (the circuit is open and incomplete). This is the opposite of how a regular passive push button works,where when pushed down the circuit is physically closed and completed,and allows a high signal to travel through to the signal pin.

Piezo Circuit



LED + Button + Piezo Code

```
from gpiozero import LED, Button
from time import sleep

led = LED(21)
button = Button(16)
piezo = LED(12)

button.when_pressed = piezo.off
button.when_released = piezo.on
```

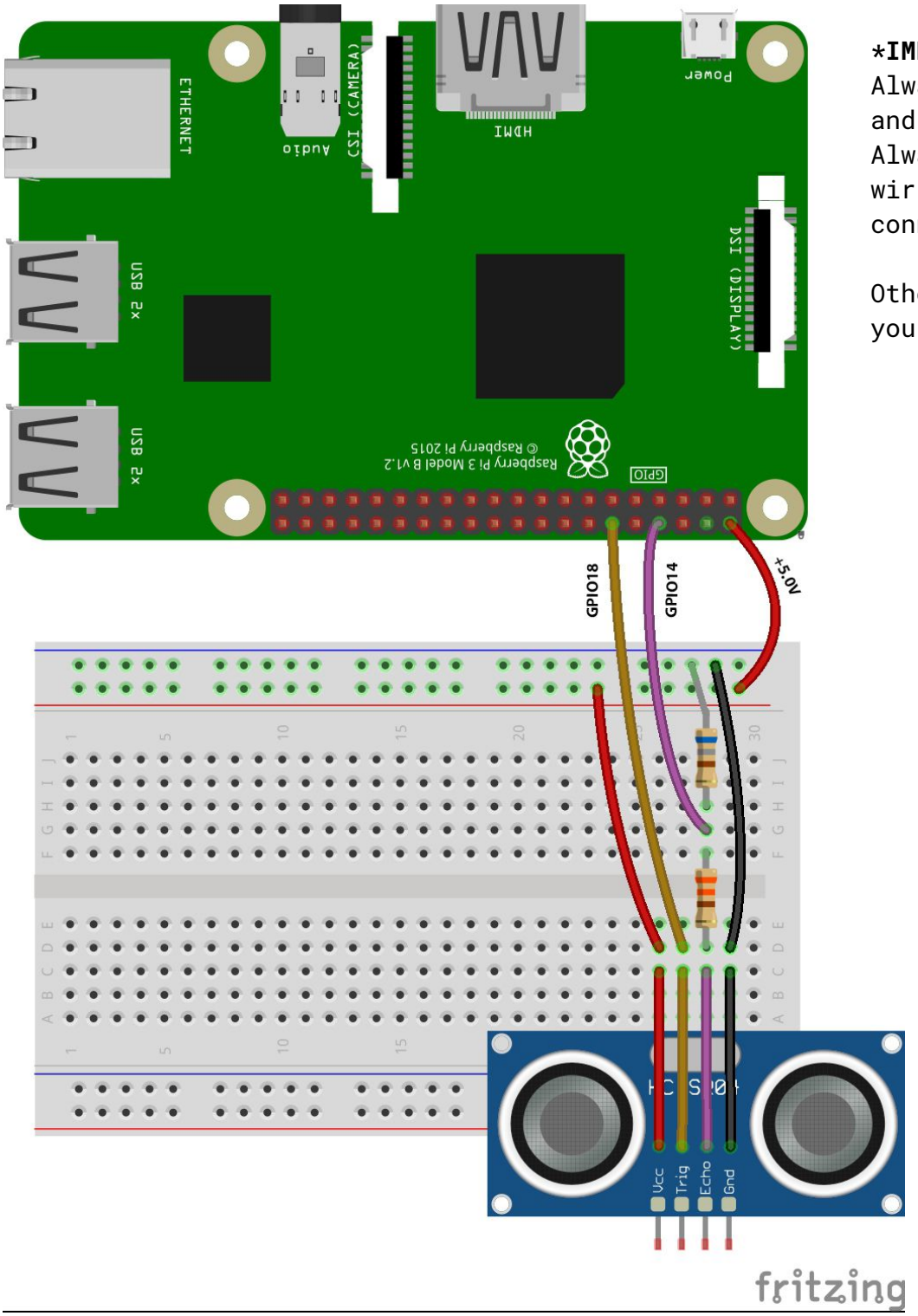
Code Breakdown

`from gpiozero import LED, Button`	Imports the LED and Button functions from the GPIO library
`from time import sleep`	Imports the sleep functions from the time library
`led = LED(21)`	Initializes pin 21 as an LED and defines variable name "led"
`button = Button(16)`	Initializes pin 16 as a Button and defines variable name "button"
`piezo = LED(12)`	Initializes pin 12 as an LED and defines variable name "piezo". We'll use the LED functions to run our piezo
`button.when_pressed = piezo.off`	When button pin value goes high, turn led off
`button.when_released = piezo.on`	When button pin value goes low, turn led on

Note on why capacitive touch sensors output high when touched

Capacitive touch sensors output high to the signal pin when untouched (the circuit is closed and complete). When touched, the circuit is grounded by your finger, and the button outputs low to the signal pin (the circuit is open and incomplete). This is the opposite of how a regular passive push button works,where when pushed down the circuit is physically closed and completed,and allows a high signal to travel through to the signal pin.

Distance Sensor Circuit



***IMPORTANT NOTE!**
Always connect GND first and +5.0v last.
Always double check your wiring is correct before connecting +5.0v.

Otherwise you could fry your Pi!

Distance Sensor Code

```
from gpiozero import LED, Button, DistanceSensor
from time import sleep

distanceMeters = DistanceSensor(trigger = 18, echo = 14)

while True:
    distanceInches = distanceMeters.distance * 39.37
    distanceInches = round(distanceInches, 1)

    print('Distance: ', distanceInches, 'in')

    sleep(.1)
```

Code Breakdown

`from gpiozero import LED, Button, DistanceSensor`	Imports the LED, Button, and DistanceSensor functions from the GPIO library
`from time import sleep`	Imports the sleep functions from the time library
`distanceMeters = DistanceSensor(trigger = 18, echo = 14)`	Initializes pins 8 and 11 as a DistanceSensor echo and trigger, respectively, and defines variable name "distanceMeters"
`while True:`	Run the indented code forever
`    distanceInches = distanceMeters.distance * 39.37`	Define variable "distanceInches" as distanceMeters times 39.37
`    distanceInches = round(distanceInches, 1)`	Define new value of distanceInches as previous value of distanceInches rounded to one decimal place
`    print('Distance: ', distanceInches, 'in')`	Prints "Distance: XX.X in."
`    sleep(.1)`	Waits 0.1 seconds before running next line of code, in this case back to initializing distanceInches

Distance Sensor + LED Code

```
from gpiozero import LED, Button, DistanceSensor
from time import sleep

led = LED(21)
distanceMeters = DistanceSensor(trigger = 18, echo = 14, threshold_distance = 0.15)

while True:
    distanceInches = distanceMeters.distance * 39.37
    distanceInches = round(distanceInches, 1)

    print('Distance: ', distanceInches, 'in')

    distanceMeters.when_in_range = led.on
    distanceMeters.when_out_of_range = led.off

    sleep(.1)
```

Code Breakdown

`from gpiozero import LED, Button, DistanceSensor`	Imports the LED, Button, and DistanceSensor functions from the GPIO library
`from time import sleep`	Imports the sleep functions from the time library
`led = LED(21)`	Initializes pin 21 as an LED and defines variable name "led"
`distanceMeters = DistanceSensor(trigger = 18, echo = 14, threshold_distance = 0.15)`	Initializes pins 8 and 11 as a DistanceSensor echo and trigger, respectively, and defines variable name "distanceMeters". Also sets threshold distance to 0.15m (~6in)
`while True:`	Run the indented code forever
`    distanceInches = distanceMeters.distance * 39.37`	Define variable "distanceInches" as distanceMeters times 39.37

`distanceInches = round(distanceInches, 1)`	Define new value of distanceInches as previous value of distanceInches rounded to one decimal place
`print('Distance: ', distanceInches, 'in')`	Prints "Distance: XX.X in."
`distanceMeters.when_in_range = led.on`	When distance sensed is less than threshold distance, turn on led
`distanceMeters.when_out_of_range = led.off`	When distance sensed is greater than threshold distance, turn off led
`sleep(.1)`	Waits 0.1 seconds before running next line of code, in this case back to initializing distanceInches

Distance Sensor + LED + Piezo Code

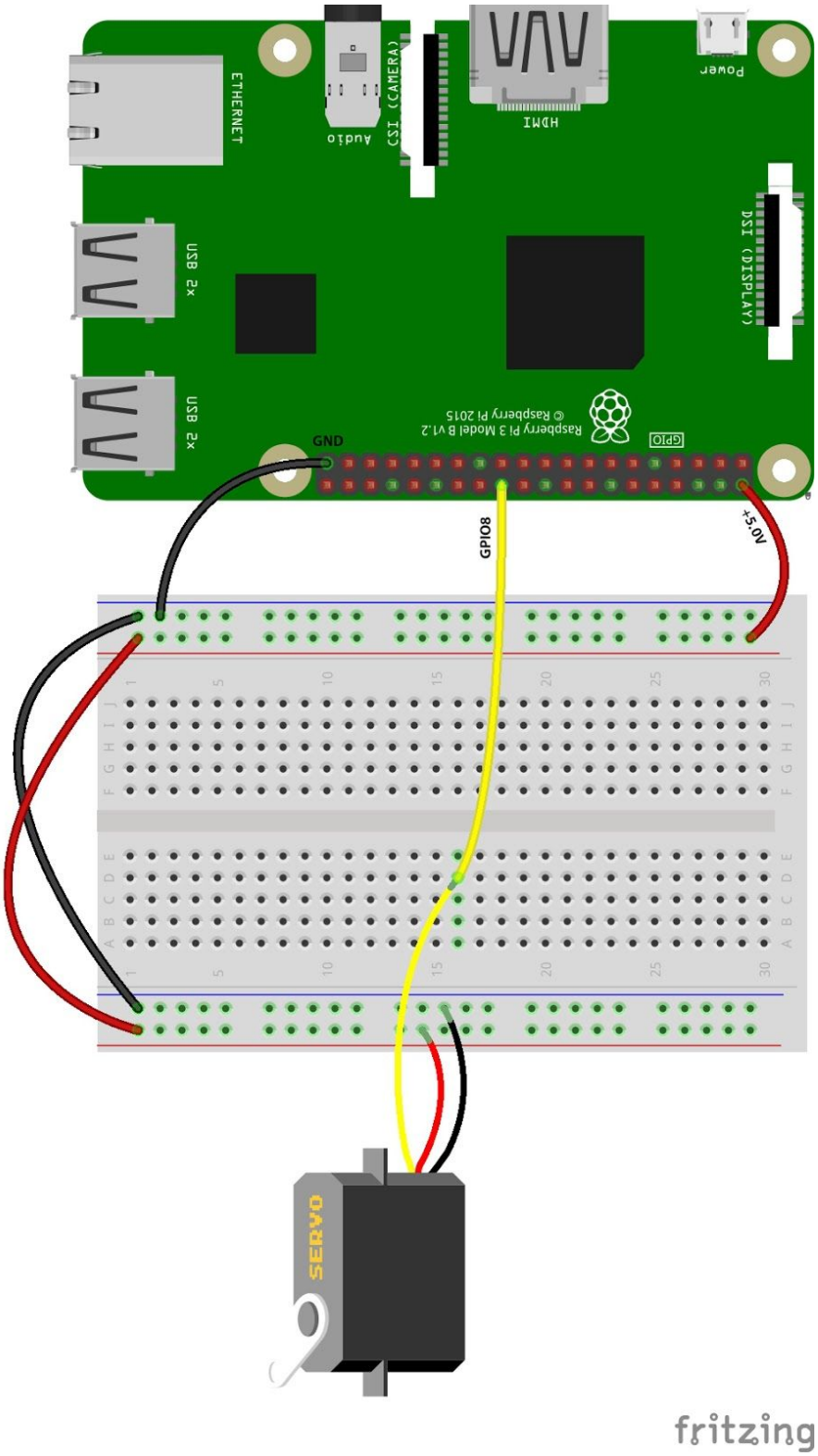
```
from gpiozero import LED, Button, DistanceSensor
from time import sleep

led = LED(21)
piezo = LED(12)
distanceMeters = DistanceSensor(trigger = 18, echo = 14, threshold_distance = 0.15)

while True:
    distanceInches = distanceMeters.distance * 39.37
    distanceInches = round(distanceInches, 1)

    print('Distance: ', distanceInches, 'in')

    distanceMeters.when_in_range = led.on
    distanceMeters.when_out_of_range = led.off

    if distanceInches <= 3:
        piezo.on()
    else:
        piezo.off()

    sleep(.1)
```

Code Breakdown

`from gpiozero import LED, Button, DistanceSensor`	Imports the LED, Button, and DistanceSensor functions from the GPIO library
`from time import sleep`	Imports the sleep functions from the time library
`led = LED(21)`	Initializes pin 21 as an LED and defines variable name "led"
`piezo = LED(12)`	Initializes pin 12 as an LED and defines variable name "piezo", we'll use the LED functions to run our piezo
`distanceMeters = DistanceSensor(trigger = 18, echo = 14, threshold_distance = 0.15)`	Initializes pins 8 and 11 as a DistanceSensor echo and

	trigger, respectively, and defines variable name "distanceMeters". Also sets threshold distance to 0.15m (~6in)
`while True:`	Run the indented code forever
`    distanceInches = distanceMeters.distance * 39.37`	Define variable "distanceInches" as distanceMeters times 39.37
`    distanceInches = round(distanceInches, 1)`	Define new value of distanceInches as previous value of distanceInches rounded to one decimal place
`    print('Distance: ', distanceInches, 'in')`	Prints "Distance: XX.X in."
`    distanceMeters.when_in_range = led.on`	When distance sensed is less than threshold distance, turn on led
`    distanceMeters.when_out_of_range = led.off`	When distance sensed is greater than threshold distance, turn off led
`    if distanceInches <= 3:`	If distance is less than or equals to 3 inches, run the indented code below, otherwise skip to the indented code below else
`        piezo.on()`	Turn piezo on
`    else:`	If if statement is not satisfied, run the indented code below
`        piezo.off()`	Turn piezo off
`    sleep(.1)`	Waits 0.1 seconds before running next line of code, in this case back to initializing distanceInches

Servo Circuit



***IMPORTANT NOTE!**
Always connect GND first and +5.0v last.
Always double check your wiring is correct before connecting +5.0v.

Otherwise you could fry your Pi!

Servo + Button Code

```
from gpiozero import LED, Button, DistanceSensor, Servo
from time import sleep

button = Button(16)
servo = Servo(8)

while True:
    if button.is_pressed == False:
        servo.max()
    else: indented code below
        servo.min()
```

Code Breakdown

`from gpiozero import LED, Button, DistanceSensor, Servo`	Imports the LED, Button, DistanceSensor, and Servo functions from the GPIO library
`from time import sleep`	Imports the sleep functions from the time library
`button = Button(16)`	Initializes pin 16 as a Button and defines variable name "button"
`servo = Servo(8)`	Initializes pin 18 as a Servo and defines variable name "servo"
`while True:`	Run the indented code forever
`if button.is_pressed == False:`	If capacitive touch button is touched (False meaning the signal is low run the indented code below, otherwise skip to the indented code below else
`    servo.max()`	Turn the servo to its maximum position
`else:`	If if statement is not satisfied, run the indented code below
`    servo.min()`	Turn the servo to its minimum position

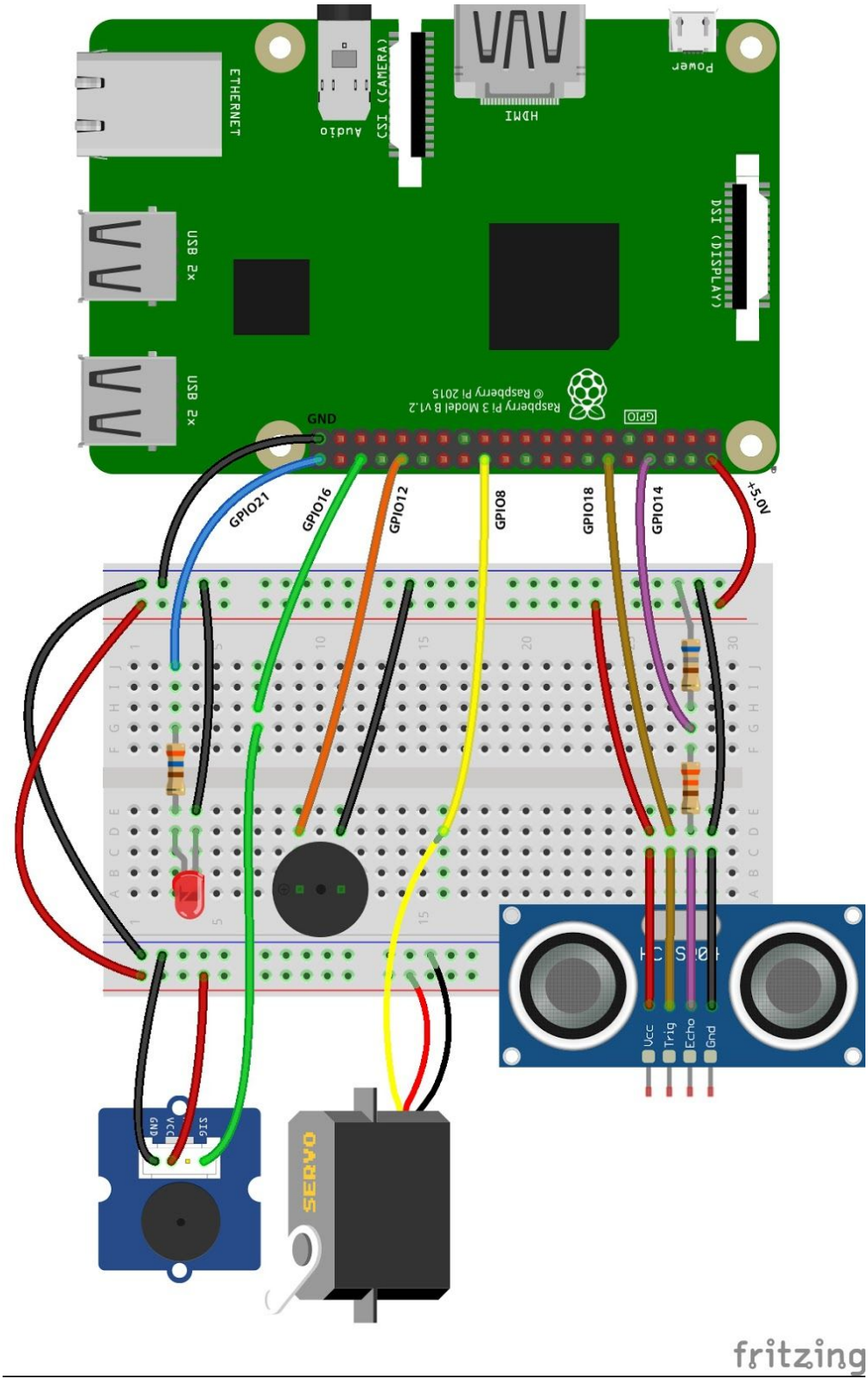
Servo Wave + Button Code

```
from gpiozero import LED, Button, DistanceSensor, Servo
from time import sleep

button = Button(16)
servo = Servo(8)

while True:
    if button.is_pressed == False:
        servo.max()
        sleep(.5)
        servo.min()
        sleep(.5)

    else: indented code below
        servo.mid()
```

Code Breakdown

`from gpiozero import LED, Button, DistanceSensor, Servo`	Imports the LED, Button, DistanceSensor, and Servo functions from the GPIO library
`from time import sleep`	Imports the sleep functions from the time library
`button = Button(16)`	Initializes pin 16 as a Button and defines variable name "button"
`servo = Servo(8)`	Initializes pin 18 as a Servo and defines variable name "servo"
`while True:`	Run the indented code forever
`if button.is_pressed == False:`	If capacitive touch button is touched (False meaning the signal is low run the indented code below, otherwise skip to the indented code below else
`servo.max()`	Turn the servo to its maximum position
`sleep(.5)`	Waits 0.5 seconds before next line of code
`servo.min()`	Turn the servo to its minimum position
`sleep(.5)`	Waits 0.5 seconds before next line of code
`else:`	If if statement is not satisfied, run the indented code below
`servo.mid()`	Turn the servo to its middle position

All Components Circuit



***IMPORTANT NOTE!**
Always connect GND first and +5.0v last.
Always double check your wiring is correct before connecting +5.0v.

Otherwise you could fry your Pi!