Module Name	**IoT Software Development Lab**
# Module Description and Learning Objectives	1
# Learning Objectives	3
# Quizzes	4 (inc. Final)
# Assignments	4 (inc. Final)
# Discussions	0 (but optional reflection question available in each LO)
Faculty Name/s	Pat Paulson
Email Addresses	ppaulson@winona.edu
Pre-Req Knowledge or Courses – What should students know prior to taking this module?	Completion of IoT Software Development Concepts or prior programming experience is recommended. Students must own a personal computer (i.e., PC) or have access to computer (i.e., Mac OS) with full permissions to install software. Students must buy a Raspberry Pi Pico kit to practice exercises (about $25/kit) using link below, or equivalent: https://www.amazon.com/gp/product/B09XHTHZ8N

Module Description and Learning Objectives

Description

This module will provide students with a description of Internet of Things (IoT) devices to include microcontrollers, sensors, actuators, and other hardware. This module builds on the IoT Software Development Concepts. Please purchase a Freenove Basic Starter Kit for Raspberry Pi Pico, or equivalent. You may use the link below or any other vendor including Amazon.com:

https://www.amazon.com/gp/product/B09XHTHZ8N

Learning Objectives

After completing this module students will be able to:

1. Validate IoT digital actuators
2. Validate IoT analog sensors
3. Validate IoT analog actuators

Module Introduction – Internet of Things Labs

This unit has students apply computer science programming concepts, microcontrollers, sensors, and actuators to monitor the environment, automate tasks, and make more effective decisions.

Learning Objective 1 – Validate IoT Digital Actuators

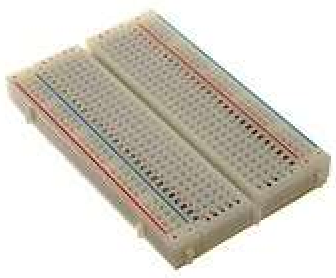
Introduction

The "Internet of things" is a powerful methodology to combine physical components with programming concepts to monitor and control the physical environment in a low-cost manner.
By combining hardware such as microcontrollers, resistors, LEDs, and breadboards with a knowledge of programming you can develop many ways to control these devices. In this lab you will build a circuit from various components, then employ software to control these components, then begin to explore enhancements to

these systems. You will try new ideas to control these devices and learn to validate your work. This is like the development processes undertaken by many organizations when they bring a new product or service to market.

Knowledge-Definitions

Breadboard: A construction base used to build semi-permanent prototypes of electronic circuits.



Source: Wikipedia.org, 2022

Digital Actuator: Any piece of computer hardware equipment which turns a digital electrical control signal into a human-perceptible form. It includes LEDs, displays, speakers, and other sensory technologies.

Digital Sensor: An electronic or electromechanical sensor, where data is digitally converted and transmitted. (Wikipedia.org)

Jumper Wire: An electrical wire with a connector or pin at each end used to interconnect the components of a breadboard.

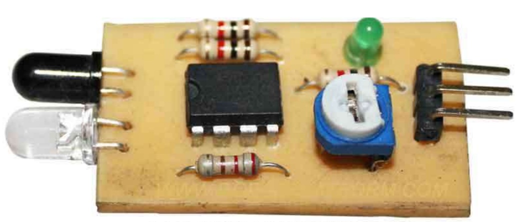
Light-Emitting Diode: A semiconductor light source that emits light when current flows through it.

Output Device: Any piece of computer hardware equipment which converts information into a human-perceptible form. (Wikipedia.org)

Push Button Switch: A momentary or non-latching switch which causes a temporary change in the state of an electrical circuit only while the switch is physically actuated. (Wikipedia.org)

Sensor: A device that detects and responds to some type of input from the physical environment. The specific input could be light, heat, motion, moisture, pressure, or any one of a great number of other environmental phenomena. The output is generally a signal that is converted to human-readable display at the sensor location or transmitted electronically over a network for reading or further processing. (Techtarget.com)

InfraRed Sensor



IR SENSOR (TRANSCEIVER)

Source: Wikipedia.com, 2022

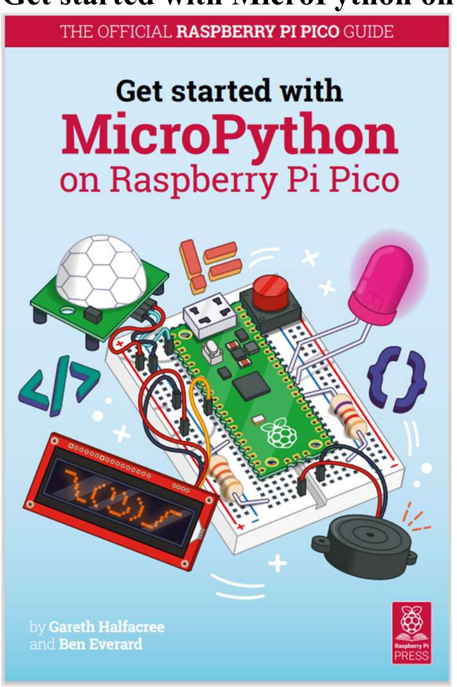
Reference:

1. **Freenove starter kit includes the following parts:**

Source: GitHub - Freenove/Freenove_Basic_Starter_Kit_for_Raspberry_Pi_Pico: Apply to FNK0064

2. **For a complete overview of the Freenove Basic Starter Kit for Raspberry Pi Pico see FreeNoveStarter_Python_Tutorial.pdf**
Source: Freenove_Basic_Starter_Kit_for_Raspberry_Pi_Pico/Python_Tutorial.pdf at main · Freenove/Freenove_Basic_Starter_Kit_for_Raspberry_Pi_Pico · GitHub

3. **Get started with MicroPython on Raspberry Pi Pico by Gareth Halfacre and Ben Everard**



Source: https://hackspace.raspberrypi.com/books/micropython-pico/pdf/download

Practice

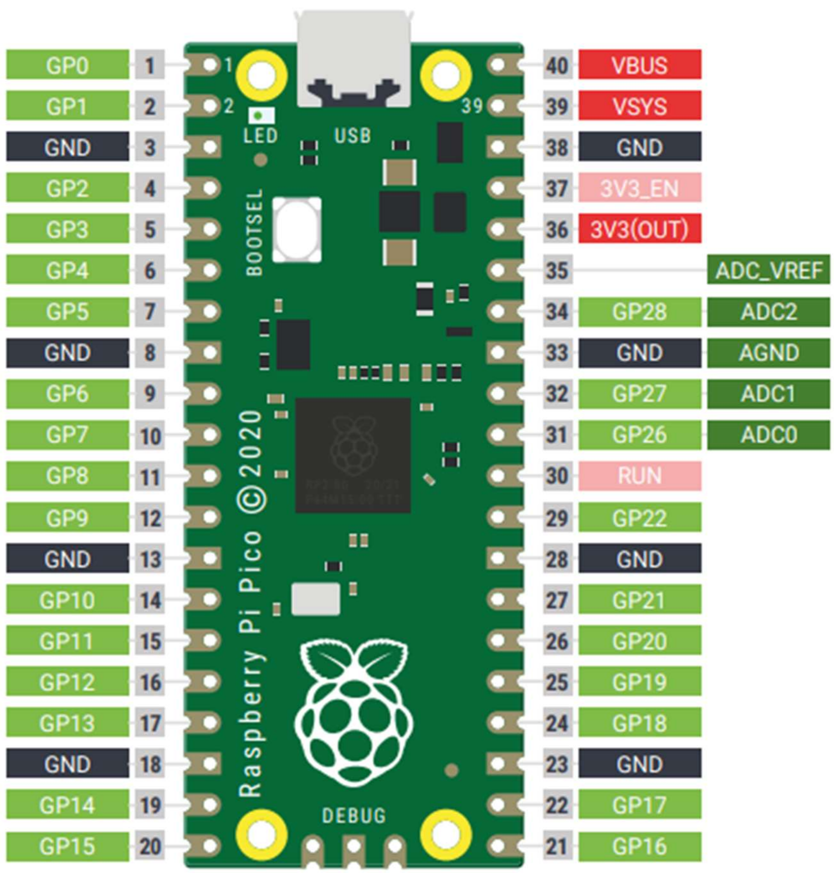
Programming Exercises

Assessment - Lab01 Program01

You will need the following prerequisites:

1) Computer with full permission (i.e., Administrator) and Thonny installed
2) Raspberry Pi Pico with soldered headers (any board type)
3) Breadboard
4) Micro USB cable (or whatever cable required by your Pico device)
5) 3.3-volt LED (any color)
6) 220-ohm resistor
7) Black jumper wire

Raspberry Pi Pico Pin Reference Diagram



Source: Get Started with MicroPython on Raspberry Pi Pico

STEPS

Create the physical circuit

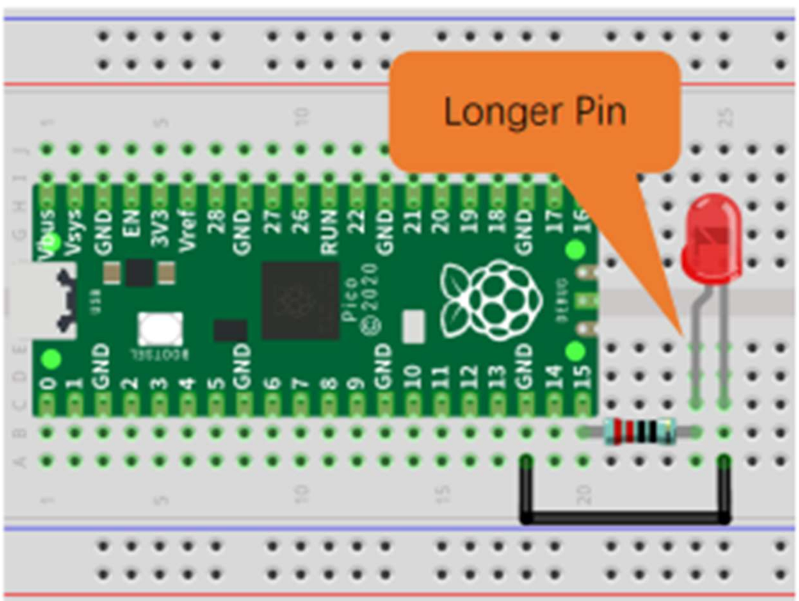
1) Mount the Pico on the breadboard with the micro-USB connector facing the edge

2) Mount the 3.3-volt LED to the breadboard, placing the lead wires in adjacent rows.

3) Connect the black jumper wire from pin 18 (GND) to the same row of the breadboard where the short leg of 3.3-volt LED is plugged into.

4) Connect the 220 ohm resistor from pin 20 (GP15) to the same row of the breadboard where the long leg of the 3.3 volt LED is plugged into.
   ref: https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-resistor-color-code

5) When completed, your circuit will look something like this:



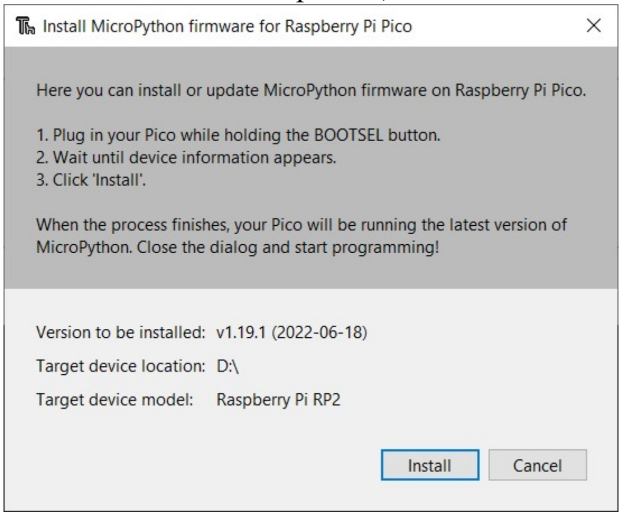
Source: Freenove.com

6) Connect the micro-USB cable to the Pico

7) Holding down the Boot/Sel button plug connect the USB-A cable end to your laptop

8) In the Thonny toolbar, press the Stop/Restart button

9) When the ‘Install MicroPython Firmware…’ dialog box appears select ‘Install’.
   Once the firmware is updated, select ‘Close’



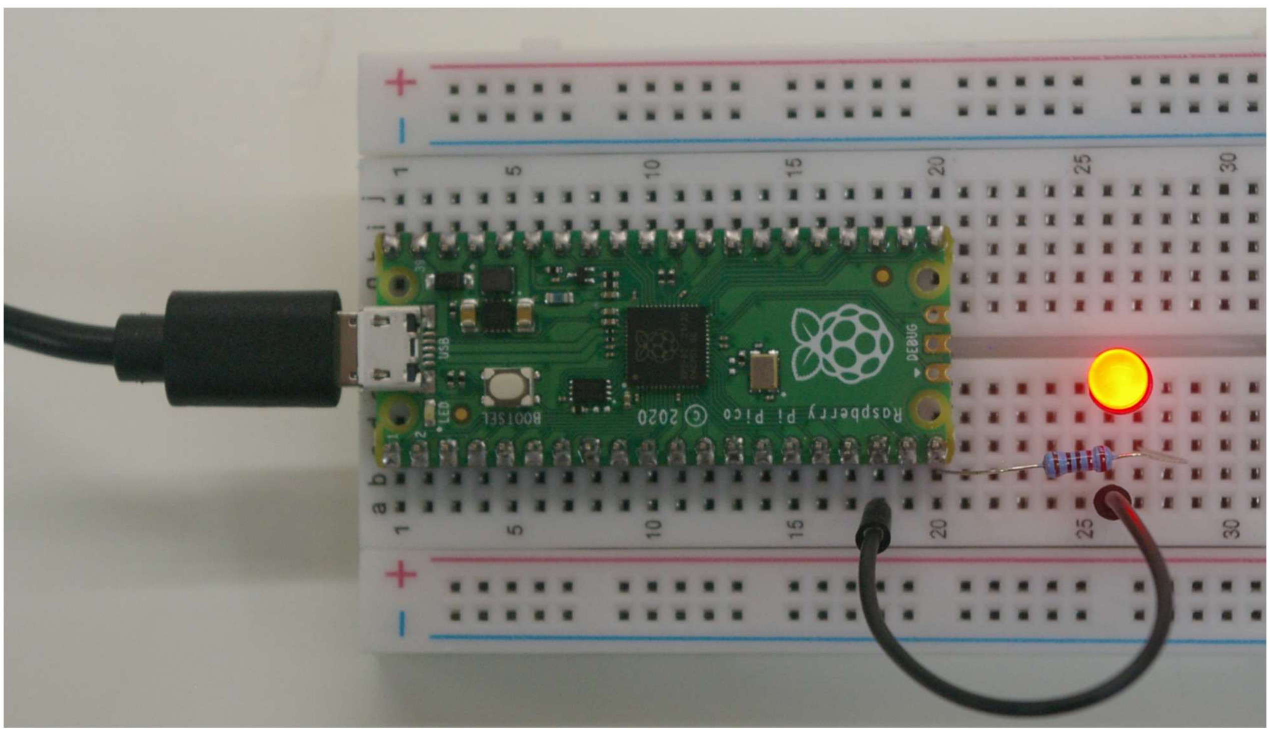
STEPS

Create a program to turn LED on

From the Thonny menu select File>New and type the program code below:

```
from machine import Pin                    # library to interface with Pico hardware
GPIO = 15                                  # connect GPIO pin 15 to LED
led = Pin(GPIO, Pin.OUT)                   # set GPIO pin to output mode
led.value(1)                               # turn led on, state (1)
print("GP", GPIO ,  "is in state", led.value())  # display pin status
```

1. Save the program to your computer as Lab01Program01.py
2. Click the [Play] button to run the program.
3. Observe the output in the shell window, and check that the LED is on.
4. If running the program does not turn the LED on, troubleshoot your wiring, then troubleshoot your Lab01Program01.py
   Note that an LED is a ‘polarized’ device-there is a positive and negative terminal. You may need to reverse the terminals.



LED connected to GP15 turned on by software command

Use the NEXT (>) arrow to submit screenshot to Assignment Folder (formerly Dropbox).

Assessment - Lab01 Program02

You will need the following prerequisites:
Note-items 1 to 7 are the same items in Lab01 Program01, items 8 to 10 are new

1) Computer with full permission (i.e., Administrator) and Thonny installed
2) Raspberry Pi Pico with soldered headers (any board type)
3) Breadboard
4) Micro USB cable (or whatever cable required by your Pico device)
5) 3.3-volt LED (any color)
6) 220-ohm resistor
7) Black jumper wire

8) Black jumper wire
9) Green and red jumper wires
10) (2) 10k ohm resistors
11) Push button switch

STEPS

Create the physical circuit

Begin with the circuit from Lab01 Program01, add components 8 through 11 as follows to create the physical circuit

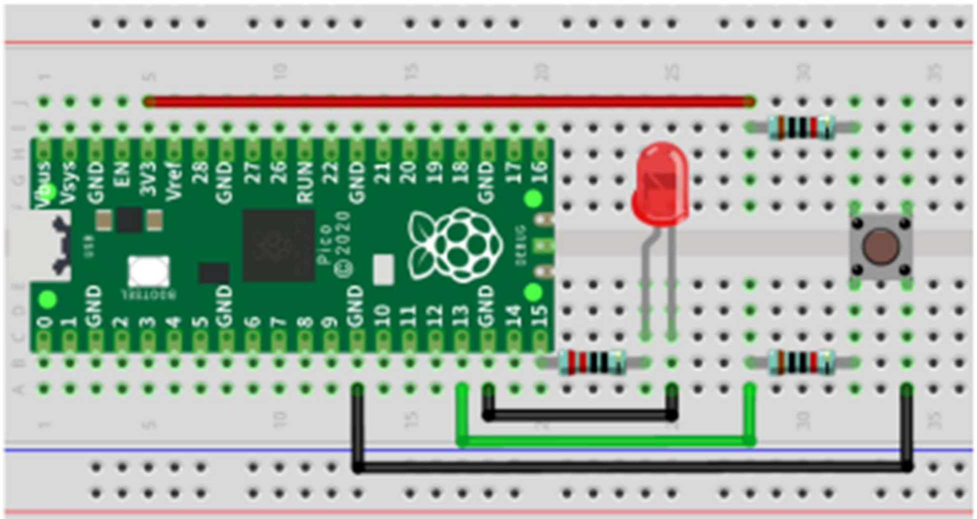
1) Being mindful of the pin orientation, mount the push button switch on the breadboard

2) Mount two 10k ohm resistors to the breadboard, placing one lead wire in the same row as the push button switch left-side leads.

3) Connect the red jumper wire from pin 36 (3.3V) to the other end of the upper 10k ohm resistor.

4) Connect the green jumper wire from pin 17 (GP13) to the lower 10k ohm resistor.

5) Connect the additional black jumper wire from pin 13 (GND) to the push button switch right-side lead.
6) When completed your circuit will look something like below:



Push button switch, digital actuator to control LED

STEPS

Create a program that allows the push button switch to toggle the LED on and off

1. From the Thonny menu select File>New and type the program code below:

```
from machine import Pin
import time

led = Pin(15, Pin.OUT)
button = Pin(13, Pin.IN, Pin.PULL_UP) # connect GPIO pin13 to push button

print("GPIO setup complete")

def reverseGPIO():                     # create toggle function
    if led.value():                    # check led state
        led.value(0)                   # turn led off, state(0)
    else:
        led.value(1)                   # turn led on, state(1)

print("toggle function ready")         # function ready
print("Press button to toggle LED")    # instruct user

try:
    while True:                        # create infinite loop
        if not button.value():
            time.sleep_ms(20)          # debounce switch
            if not button.value():
                reverseGPIO()
                while not button.value():
                    time.sleep_ms(20)
except:
    pass                               # continue on error
```

2. Save the program to your computer as Lab01Program02.py
3. Click the [Play] button to run the program.
4. Observe the output in the shell window.
5. Validate that pressing the push button switch turns the LED on and off.
6. If pressing the push button does not turn the LED on and off, troubleshoot your wiring, then troubleshoot your Lab01Program02.py

Thonny - C:\Labs\Lab01Program02.py @ 10 : 42

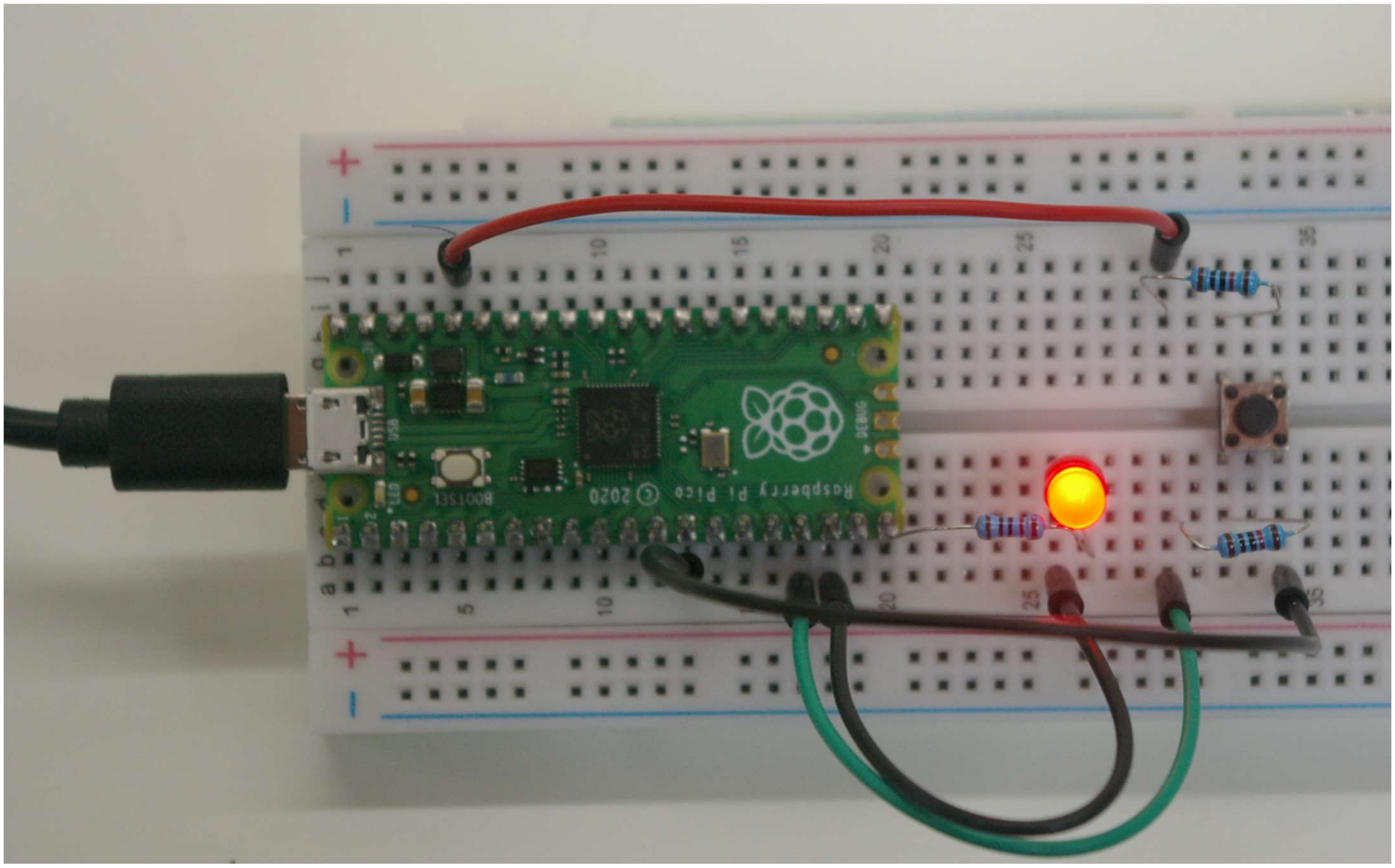
File Edit View Run Tools Help

Lab01Program01.py × Lab01Program02.py ×

```python
from machine import Pin
import time

led = Pin(15, Pin.OUT)
button = Pin(13, Pin.IN, Pin.PULL_UP) # connect GPIO pin13 to push button

print("GPIO setup complete")

def reverseGPIO():                    # create toggle function
    if led.value():                   # check led state
        led.value(0)                  # turn led off, state(0)
    else:
        led.value(1)                  # turn led on, state(1)

print("toggle function ready")        # function ready
print("Press button to toggle LED")   # instruct user

try:
    while True:                       # create infinite loop
        if not button.value():
            time.sleep_ms(20)         # debounce switch
            if not button.value():
                reverseGPIO()
                while not button.value():
                    time.sleep_ms(20)
except:
    pass                              # continue on error
```

Shell ×

```
>>> %Run -c $EDITOR_CONTENT

 GPIO setup complete
 toggle function ready
 Press button to toggle LED
```



Push button switch circuit added to control LED

Use the NEXT (>) arrow to submit screenshot to Assignment Folder (formerly Dropbox).

Assessment - Lab01 Program03

You will need the same prerequisites from Lab01 Program01:

1) Computer with full permission (i.e., Administrator) and Thonny installed
2) Raspberry Pi Pico with soldered headers (any board type)
3) Breadboard
4) Micro USB cable (or whatever cable required by your Pico device)
5) 3.3-volt LED (any color)
6) 220-ohm resistor
7) Black jumper wire

STEPS

Use the same physical circuit as Lab01 Program01

1) Using Lab01Program01.py create a new file, Lab01Program03.py by modifying line 4 to create a program that turns the LED off.

```
from machine import Pin                # library to interface with Pico hardware
GPIO = 15                              # connect GPIO pin 15 to LED
led = Pin(GPIO, Pin.OUT)               # set GPIO pin to output mode
led.value(0)                           # turn led off, state (0)
print("GP", GPIO ,  "is in state", led.value())  # display pin status
```

Use the NEXT (>) arrow to submit screenshot to Assignment Folder (formerly Dropbox).

Assessment - Lab01 Program04

You will use the same prerequisites as Lab01Program02

1) Computer with full permission (i.e., Administrator) and Thonny installed
2) Raspberry Pi Pico with soldered headers (any board type)
3) Breadboard
4) Micro USB cable (or whatever cable required by your Pico device)
5) 3.3-volt LED (any color)
6) 220-ohm resistor
7) Black jumper wire

8) Black jumper wire
9) Green and red jumper wires
10) (2) 10k ohm resistors
11) Push button switch

STEPS

Use the same physical circuit as Lab01 Program02

1) Using Lab01Program02.py create a new file Lab01Program04.py by adding a print statement to the reverseGPIO() function which writes the current state of the LED to the shell.
   Note state (0) is off and state (1) is on.

```
from machine import Pin
import time

led = Pin(15, Pin.OUT)
button = Pin(13, Pin.IN, Pin.PULL_UP) # connect GPIO pin13 to push button

print("GPIO setup complete")

def reverseGPIO():                    # create toggle function
    if led.value():                   # check led state
        led.value(0)                  # turn led off, state(0)
    else:
        led.value(1)                  # turn led on, state(1)
    print("LED changed to state", led.value())

print("toggle function ready")        # function ready
print("Press button to toggle LED")   # instruct user

try:
    while True:                       # create infinite loop
        if not button.value():
            time.sleep_ms(20)         # debounce switch
            if not button.value():
                reverseGPIO()
```

```
                    while not button.value():
                        time.sleep_ms(20)

except:
    pass                              # on error continue
```

Use the NEXT (>) arrow to submit screenshot to Assignment Folder (formerly Dropbox).

Assessment – Quiz 1

1) Describe a push button switch
2) Provide an example of a digital actuator.
3) Describe an output device
4) Describe an example of a digital actuator.
5) Describe a jumper wire
6) What are the 5 band colors on a 220-ohm resistor?
7) What are the 5 band colors on a 10k-ohm resistor?

Use the NEXT (>) arrow to submit quiz answers to Assignment Folder (formerly Dropbox).

Learning Objective 2 – Validate IoT Analog Sensors

Introduction

The "Internet of things" is a powerful methodology to combine physical components with programming concepts to monitor and control the physical environment in a low-cost manner.
By combining hardware such as microcontrollers, resistors, LEDs, and breadboards with a knowledge of programming you can develop many ways to control these devices. In this lab you will build a circuit from various components, then employ software to control these components, then begin to explore enhancements to these systems. You will try new ideas to control these devices and learn to validate your work. This is like the development processes undertaken by many organizations when they bring a new product or service to market.

Computers are digital devices, working on a binary system of 0's and 1's. In this lab we will introduce analog components such as temperature sensors known as thermistors. Analog devices have been around for much longer than digital devices and are common. For computers and software to work with analog devices, there is a need to use analog-to-digital converters.

Knowledge

Analog Actuator: Any piece of computer hardware equipment which turns an analog electrical control signal into a human perceptible form. It includes lights, speakers, linear actuators, rotary actuators, and other sensory technologies.

Analog Sensor: A device that produces a variable output signal for the purpose of sensing a physical phenomenon. (Wikipedia.org)

Analog-to-Digital Converter (ADC): A system that converts an analog signal, such as a sound picked up by a microphone or light entering a digital camera, into a digital signal. (Wikipedia.org)

Thermistor: A type of resistor whose resistance is strongly dependent on temperature, more so than in standard resistors. The word thermistor is a portmanteau of thermal and resistor. (Wikipedia.org)

Practice

Programming Exercises

Assessment - Lab02 Program01

You will need the following prerequisites:

1) Computer with full permission (i.e., Administrator) and Thonny installed
2) Raspberry Pi Pico with soldered headers (any board type)
3) Breadboard
4) Micro USB cable (or whatever cable required by your Pico device)

STEPS

For this lab you will use an analog temperature device and an ADC that is already built into the Raspberry Pi Pico. No additional hardware is needed.

Plug your micro-USB cable into the Pico, plug the other end into a USB-A connection on your computer.

STEPS

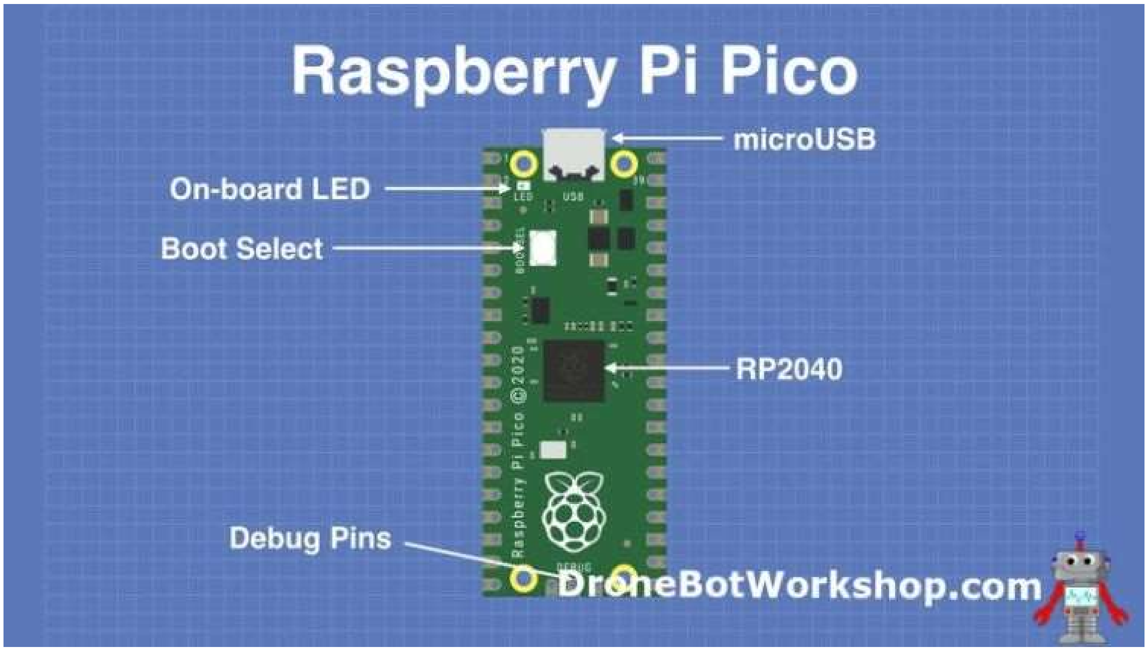
Create a program that reads the built-in temperature sensor, converts the temperature to Fahrenheit, prints the output to the Thonny Shell, and then repeats the process every 2 seconds.

From the Thonny menu select File>New and type the program code below:

```
import machine                      # library to interface with Pico hardware
import utime                        # library that provides time functions

sensor_temp=machine.ADC(4)          # connect to the ADC on channel 4

conversion_factor = 3.3 / (65535)   # conversion factor 3.3 volt scale, 16-bit ADC

while True:
    reading = sensor_temp.read_u16() * conversion_factor   # convert to voltage
    temperature = 27 - (reading - 0.706)/0.001721          # convert to degrees C
    print(temperature , "degrees C")                       # send output to Shell
    utime.sleep(2)                                         # wait 2 seconds
```

1. Save the program to your computer as Lab02Program01.py
2. Click the [Play] button to run the program.
3. Observe the output in the shell window, and check that the temperature value appears reasonable.
4. Validate that the temperature sensor is working by placing your finger on the RP2040 chip on top of the Pico. After a few seconds the temperature should begin increasing.

Source: Get Started With MicroPython on Raspberry Pi Pico



Temperature reading increasing with finger placed on RP2040

Use the NEXT (>) arrow to submit screenshot to Assignment Folder (formerly Dropbox).

Assessment - Lab02 Program02

You will need the following prerequisites:

1) Computer with full permission (i.e., Administrator) and Thonny installed
2) Raspberry Pi Pico with soldered headers (any board type)
3) Breadboard
4) Micro USB cable (or whatever cable required by your Pico device)
5) Thermistor
6) 10k-ohm resistor
7) Jumper wires-red, black, blue

STEPS

Create the physical circuit.

Note that you can continue adding components to the breadboard, as all these labs are designed to be cumulative, and use different GPIO pins. This allows you to go back and run a prior lab without needing to move or change components.

1) Mount the Pico on the breadboard with the micro-USB connector facing the edge

2) Mount the thermistor to the breadboard, placing the lead wires in adjacent rows.

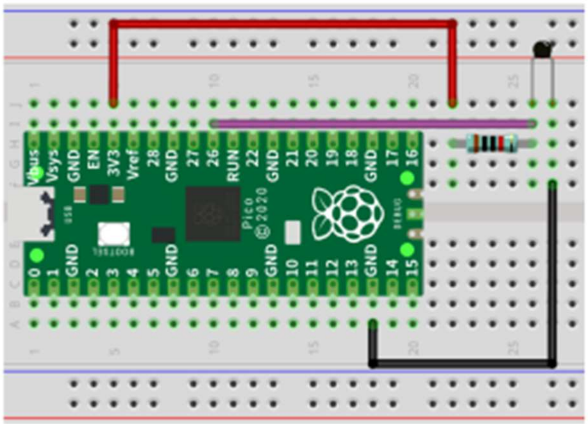
3) Mount the 10k ohm resistor with the right leg in the same row as the left leg of the thermistor, and the left leg two rows further left

4) Connect the black jumper wire from pin 18 (GND) to the same row of the breadboard where the right leg of the thermistor is plugged into.

5) Connect the red jumper wire from pin 5 (3V3) to the same row of the breadboard where the left leg of the 10k ohm resistor is plugged into.

6) Connect the blue jumper wire from pin 31 (GP26_A0) to the same row as the left leg of the thermistor and the right leg of the 10k ohm resistor

When completed, your circuit will look something like this:



Source: Freenove.com

STEPS

Create a program that reads the analog thermistor sensor, converts the temperature to Fahrenheit, prints the output to the Thonny Shell, and then repeats the process every 2 seconds.

From the Thonny menu select File>New and type the program code below:

```
from machine import Pin, ADC   # library to interface with Pico
import time                    # library that provides time functions
import math                    # library that provides math functions


adc=ADC(26)   #use GPIO 26



try:
    while True:
        adcValue = adc.read_u16()
        voltage = adcValue / 65535.0 * 3.3
        Rt = 10 * voltage / (3.3-voltage)
        tempK = (1 / (1 / (273.15+25) + (math.log(Rt/10)) / 3950))
        tempC = int(tempK - 273.15)
        time.sleep(2) # wait 2 seconds
        tempF = int(tempC * 9/5 + 32)   # convert to degrees F

        # send output to Shell
        print("ADC:", adcValue, "  Volts:%0.2f"%voltage,
              " Temp:" + str(tempC) + "C" + " Temp:" + str(tempF) + "F")
except:
    pass
```

Save the program to your computer as Lab02Program02.py

1. Click the [Play] button to run the program.

2. Observe the output in the Shell window, and check that the temperature value appears reasonable. Is the temperature value the same, higher, or lower that the temperature readings seen in Lab02Program01? What could account for any differences?
3. Validate that the temperature sensor is working by placing your fingers on the thermistor. After a few seconds the temperature should begin increasing.

Source: Freenove.com

Thonny - C:\Labs\Lab02Program02.py @ 7:5

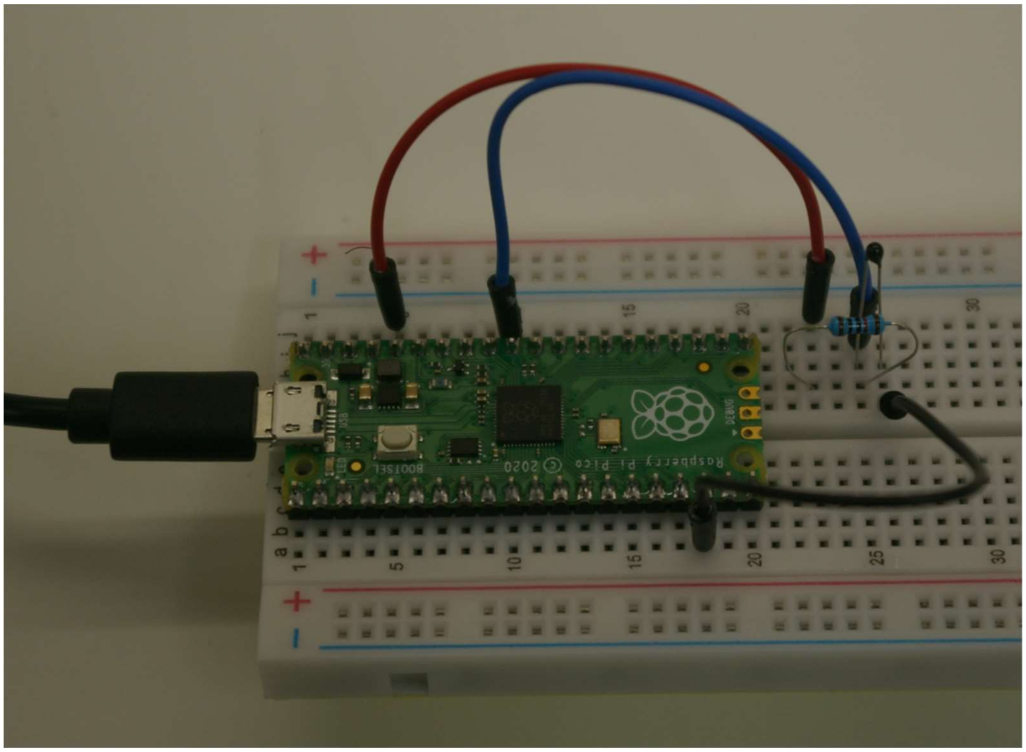
File Edit View Run Tools Help

Lab02Program02.py

```
from machine import Pin, ADC   # library to interface with Pico
import time                    # library that provides time functions
import math                    # library that provides math functions

adc=ADC(26)  #use GPIO 26

try:
    while True:
        adcValue = adc.read_u16()
        voltage = adcValue / 65535.0 * 3.3
        Rt = 10 * voltage / (3.3-voltage)
        tempK = (1 / (1 / (273.15+25) + (math.log(Rt/10)) / 3950))
        tempC = int(tempK - 273.15)
        time.sleep(2) # wait 2 seconds
        tempF = int(tempC * 9/5 + 32)   # convert to degrees F

        # send output to Shell
        print("ADC:", adcValue, "  Volts:%0.2f"%voltage,
              " Temp:" + str(tempC) + "C" + " Temp:" + str(tempF) + "F")
except:
    pass
```

Shell

```
ADC: 32167   Volts:1.62  Temp:25C Temp:77F
ADC: 32135   Volts:1.62  Temp:25C Temp:77F
ADC: 32055   Volts:1.61  Temp:25C Temp:77F
ADC: 29319   Volts:1.48  Temp:29C Temp:84F
ADC: 25878   Volts:1.30  Temp:34C Temp:93F
ADC: 26422   Volts:1.33  Temp:34C Temp:93F
ADC: 27190   Volts:1.37  Temp:32C Temp:89F
```



Analog Thermistor Sensor Circuit

Use the NEXT (>) arrow to submit screenshot to Assignment Folder (formerly Dropbox).

Assessment - Lab02 Program03

You will need the same prerequisites from Lab02 Program01:

1) Computer with full permission (i.e., Administrator) and Thonny installed
2) Raspberry Pi Pico with soldered headers (any board type)
3) Breadboard
4) Micro USB cable (or whatever cable required by your Pico device)

STEPS

For this lab you will use an analog temperature device and an ADC that is already built into the Raspberry Pi Pico. No additional hardware is needed. This is the same as in Lab02 Program 01.

1) Using Lab02Program01.py create a new file, Lab02Program03.py by adding a statement to convert the temperature in degrees C to degrees F. Make sure to also modify the print statement so that the output temperature is correctly labeled as degrees F.

```
import machine                      # library to interface with Pico hardware
import utime                        # library that provides time functions

sensor_temp=machine.ADC(4)          # connect to the ADC on channel 4

conversion_factor = 3.3 / (65535)   # conversion factor 3.3 volt scale, 16-bit ADC

while True:
    reading = sensor_temp.read_u16() * conversion_factor   # convert to voltage
    temperature = 27 - (reading - 0.706)/0.001721          # convert to degrees C
    # ADD STATEMENT BELOW TO CONVERT FROM DEGREES C TO DEGREES F
    temperature = (temperature * 9/5) + 32                 # convert to degrees F
    # MODIFY PRINT STATEMENT TO SHOW DEGREES F
    print(temperature , "degrees F")                       # send output to Shell
    utime.sleep(2)                                         # pause 2 seconds
```

```
import machine                          # library to interface with Pico hardware
import utime                            # library that provides time functions

sensor_temp=machine.ADC(4)              # connect to the ADC on channel 4

conversion_factor = 3.3 / (65535)       # conversion factor 3.3 volt scale, 16-bit ADC

while True:
    reading = sensor_temp.read_u16() * conversion_factor   # convert to voltage
    temperature = 27 - (reading - 0.706)/0.001721          # convert to degrees C
    # ADD STATEMENT BELOW TO CONVERT FROM DEGREES C TO DEGREES F
    temperature = (temperature * 9/5) + 32                 # convert to degrees F
    # MODIFY PRINT STATEMENT TO SHOW DEGREES F
    print(temperature , "degrees F")                       # send output to Shell
    utime.sleep(2)                                         # pause 2 seconds
```

```
>>> %Run -c $EDITOR_CONTENT
 59.61343 degrees F
 58.77078 degrees F
 58.77078 degrees F
 58.77078 degrees F
```

Sensor Output Displayed in Degrees F (Fahrenheit)

Use the NEXT (>) arrow to submit screenshot to Assignment Folder (formerly Dropbox).

Assessment - Lab02 Program04

You will need the same prerequisites from Lab02 Program02:

1) Computer with full permission (i.e., Administrator) and Thonny installed
2) Raspberry Pi Pico with soldered headers (any board type)
3) Breadboard
4) Micro USB cable (or whatever cable required by your Pico device)
5) Thermistor
6) 10k-ohm resistor
7) Jumper wires-red, black, blue

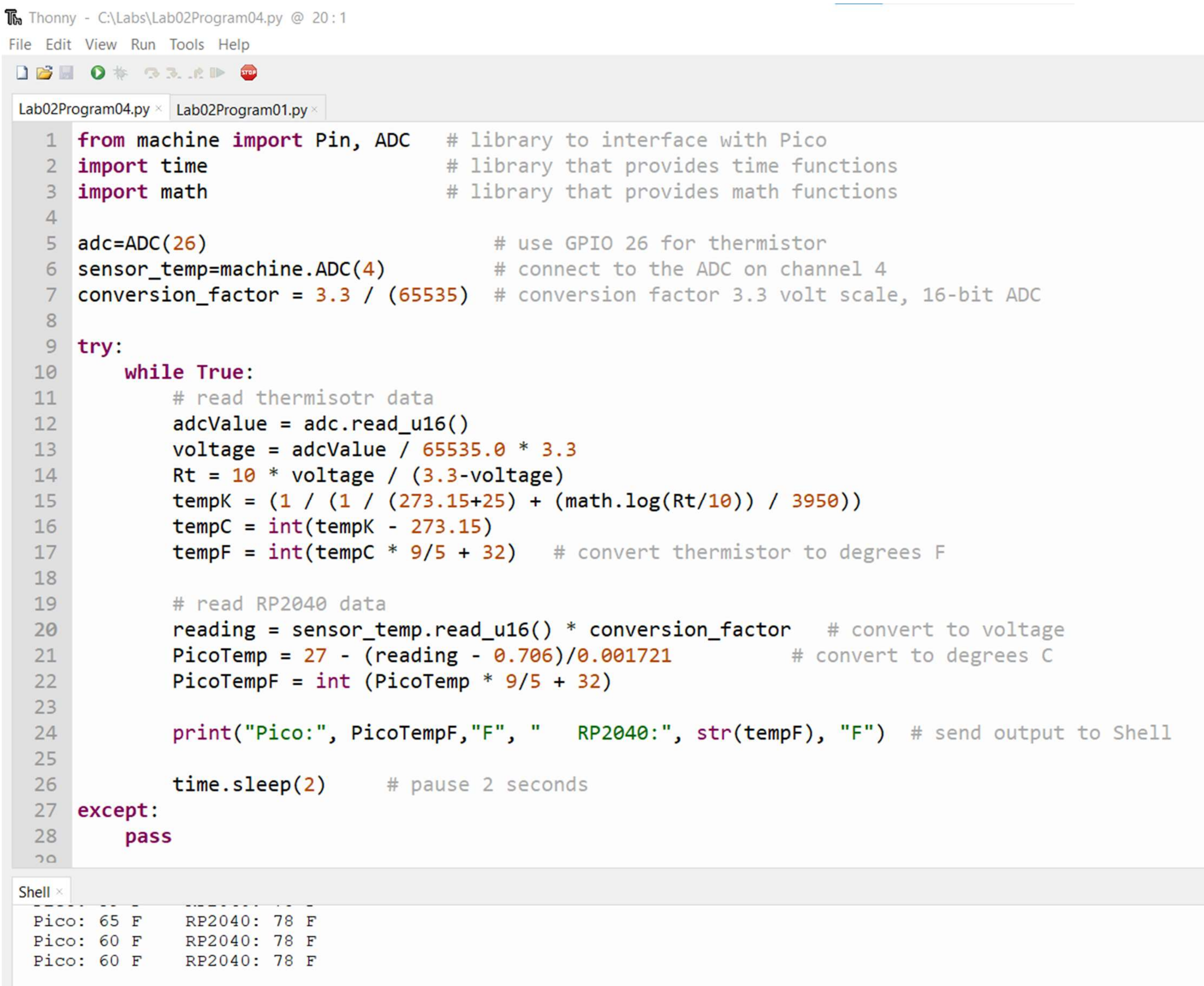
STEPS

Use the same physical circuit as Lab02 Program02.

1) Using Lab02Program02.py create a new file, Lab02Program04.py that displays both the temperature readings from the thermistor and the RP2040, both in degrees F, on the same line of output. Refer to Lab02Program01.py for the way to display the RP2040 temperature sensor output.

```
from machine import Pin, ADC   # library to interface with Pico
import time                    # library that provides time functions
import math                    # library that provides math functions

adc=ADC(26)                        # use GPIO 26 for thermistor
sensor_temp=machine.ADC(4)         # connect to the ADC on channel 4
conversion_factor = 3.3 / (65535)  # conversion factor 3.3 volt scale, 16-bit ADC

try:
    while True:
        # read thermisotr data
        adcValue = adc.read_u16()
        voltage = adcValue / 65535.0 * 3.3
        Rt = 10 * voltage / (3.3-voltage)
        tempK = (1 / (1 / (273.15+25) + (math.log(Rt/10)) / 3950))
        tempC = int(tempK - 273.15)
        tempF = int(tempC * 9/5 + 32)   # convert thermistor to degrees F


        # read RP2040 data
        reading = sensor_temp.read_u16() * conversion_factor   # convert to voltage
        PicoTemp = 27 - (reading - 0.706)/0.001721          # convert to degrees C
        PicoTempF = int (PicoTemp * 9/5 + 32)

        print("Pico:", PicoTempF,"F", "   RP2040:", str(tempF), "F")  # output results

        time.sleep(2)     # pause 2 seconds
```

```
except:
    pass
```




```
from machine import Pin, ADC   # library to interface with Pico
import time                    # library that provides time functions
import math                    # library that provides math functions

adc=ADC(26)                         # use GPIO 26 for thermistor
sensor_temp=machine.ADC(4)          # connect to the ADC on channel 4
conversion_factor = 3.3 / (65535)  # conversion factor 3.3 volt scale, 16-bit ADC

try:
    while True:
        # read thermisotr data
        adcValue = adc.read_u16()
        voltage = adcValue / 65535.0 * 3.3
        Rt = 10 * voltage / (3.3-voltage)
        tempK = (1 / (1 / (273.15+25) + (math.log(Rt/10)) / 3950))
        tempC = int(tempK - 273.15)
        tempF = int(tempC * 9/5 + 32)    # convert thermistor to degrees F

        # read RP2040 data
        reading = sensor_temp.read_u16() * conversion_factor   # convert to voltage
        PicoTemp = 27 - (reading - 0.706)/0.001721          # convert to degrees C
        PicoTempF = int (PicoTemp * 9/5 + 32)

        print("Pico:", PicoTempF,"F", "   RP2040:", str(tempF), "F")  # send output to Shell

        time.sleep(2)      # pause 2 seconds
except:
    pass
```

```
Pico: 65 F    RP2040: 78 F
Pico: 60 F    RP2040: 78 F
Pico: 60 F    RP2040: 78 F
```

Use the NEXT (>) arrow to submit screenshot to Assignment Folder (formerly Dropbox).

Assessment – Quiz 2

1) Describe an Analog Actuator
2) Describe an Analog Sensor
3) What is an Analog-to-Digital Converter?
4) Describe a thermistor

Use the NEXT (>) arrow to submit quiz answers to Assignment Folder (formerly Dropbox).

Learning Objective 3 – Validate IoT Analog Actuators

Introduction

This module will provide descriptions and resources for learners to program their Raspberry Pi Pico using the MicroPython language.

Knowledge

NPN Bipolar Junction Transistor: A semiconductor device that allows a small current injected at one of its terminals to control a much larger current flowing between the terminals, making the device capable of amplification or switching. (Wikipedia.org).

Piezoelectric Buzzer: An audio signaling device that uses the piezoelectric effect driven by an oscillating circuit. Typically used to confirm user input.

Practice

Programming Exercises

Assessment - Lab03 Program01

You will need the following prerequisites:

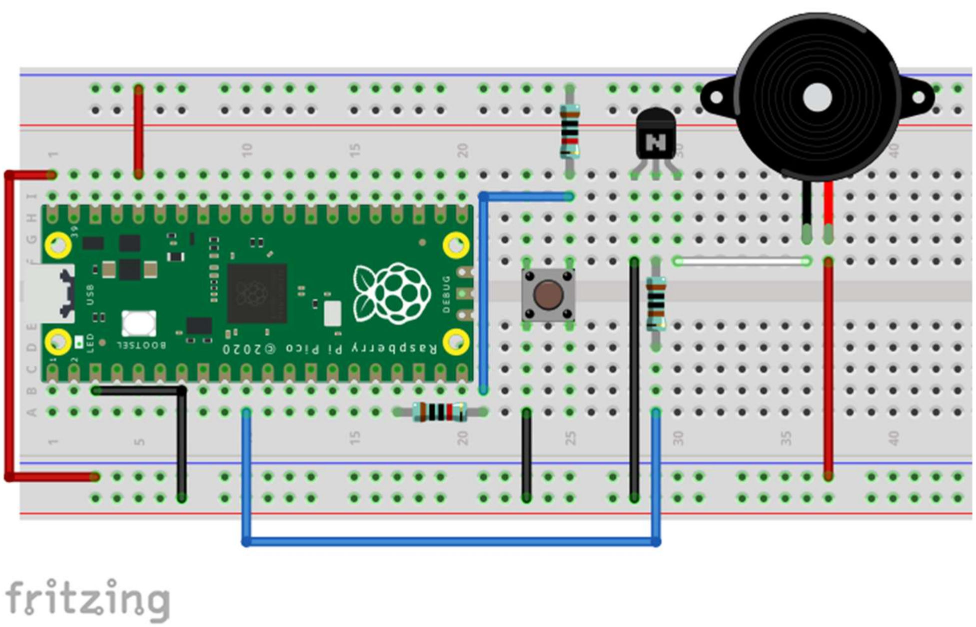
1) Computer with full permission (i.e., Administrator) and Thonny installed
2) Raspberry Pi Pico with soldered headers (any board type)
3) Breadboard
4) Micro USB cable (or whatever cable required by your Pico device)
5) Piezoelectric Passive Buzzer
6) 10k ohm resistors, quantity 2
7) 1k ohm resistor
8) NPN Bipolar Junction Transistor (BJT) (8050)
9) Black jumper wires, quantity 3
10) Red jumper wires, quantity 3
11) Blue jumper wires, quantity 2
12) White jumper wire
13) Push button switch

STEPS

Create the physical circuit. Note that you can continue adding components to the breadboard, as all these labs are designed to be cumulative, and use different GPIO pins. This allows you to go back and run a prior lab without needing to move or change components.

1) Mount the Pico on the breadboard with the micro-USB connector facing the edge
2) Mount the push button switch on the breadboard, leaving two empty rows between it and the right edge of the Pico.

3) Mount the NPN BJT Transistor on the breadboard, leaving two empty rows between it and the push button switch.
4) Mount the Piezoelectric Passive Buzzer on the breadboard, leaving five empty rows between it and the NPN BJT Transistor.
5) Connect a 10k ohm resistor from pin 17 (GP13) to the breadboard row at the right end of the Pico.
6) Connect a blue jumper wire from this breadboard row at the right end of the Pico to the top half breadboard row that is in line with the right pins of the push button switch.
7) Connect a 10k ohm resistor from the top row (+) upper breadboard power rail to the row that is in line with the right pins of the push button switch.
8) Connect a 1k ohm resistor from a row in line with the middle terminal of the NPN BJT Transistor directly below to the lower half of the breadboard.
9) Connect a blue jumper wire from the same row of the 1k resistor to pin10 (GP7)
10) Connect a black jumper wire from pin 3 (GND) to the bottom row of the lower breadboard power rail.
11) Connect a black jumper wire from the row in line with the left edge of the push button switch to the bottom row of the lower breadboard power rail.
12) Connect a black jumper wire from the row in line with the left terminal of the NPN BJT Transistor to the bottom row of the lower breadboard power rail.
13) Connect a red jumper wire from pin 36 (3V3) to the upper row of the upper breadboard power rail.
14) Connect a red jumper wire from pin 40 (5V) to the top row of the lower breadboard power rail.
15) Connect a red jumper wire from the top row of the lower breadboard power rail to the row that is in line with the right terminal (+) of the Piezoelectric Passive Buzzer.
16) Connect a white jumper wire from a row in line with the right terminal of the NPN BJT Transistor to the row that is in line with the left terminal (-) of the Piezoelectric Passive Buzzer.
17) When completed the circuit will look something like this:



Piezoelectric Passive Buzzer with Push Button Switch to actuate

STEPS

Create a program that activates the Piezoelectric Passive Buzzer with a varying tone for 3 seconds and prints a message to the Thonny Shell that an alert will sound. After 3 seconds, print a message to the shell that the test is complete.

1. From the Thonny menu select File>New and type the program code below:

```
import math
import utime

PI = math.pi                               # use constant pi
button = Pin(13, Pin.IN, Pin.PULL_UP)      # button controls buzzer by GPIO 13
passiveBuzzer = PWM(Pin(7))                # Passive Buzzer powered by GPIO 7
passiveBuzzer.freq(1000)                   # Passive Buzzer base frequency

def alert():                               # create sinusoidal frequency for buzzer
    for x in range(0, 36):
        sinVal  = math.sin(x * 10 * PI / 180)
        toneVal = 1500+int(sinVal*500)
        passiveBuzzer.freq(toneVal)
        utime.sleep_ms(10)  # time delay modifies sound patterns, default is 10


print("Setup complete. \n")
t = utime.ticks_ms()    # set start time
print("An alert will sound for 3 seconds! \n")


try:
    while True:
        if not utime.ticks_diff(utime.ticks_ms(), t) >= 3000:   # execute for 3 sec
            button.value(1)     # turn on virtual button
            passiveBuzzer.duty_u16(4092*2)
            alert()     #  call sinusoidal frequency function
        else:                                  # after 3 seconds turn off buzzer
            print("The test is complete.")
            passiveBuzzer.duty_u16(0)
            button.value(0)                    # turn off power to virtual button
            exit()                             # exit the program

except:
    passiveBuzzer.deinit()   # free up resources
```

Save the program to your computer as Lab03Program01.py

2) Click the [Play] button to run the program.
3) Observer the output in the shell window.
4) Validate that the buzzer sounds with a varying frequency for 3 seconds.
5) In line 16 of the code, vary the sleep time from 10 (ms) to values between 6 and 50-you should note a distinct change in the buzzer output.
6) If nothing happens when you run the program, first check and troubleshoot the component wiring, then troubleshoot your Lab03Program01.py

Thonny - C:\Labs\Lab03Program01.py @ 35 : 53

File Edit View Run Tools Help

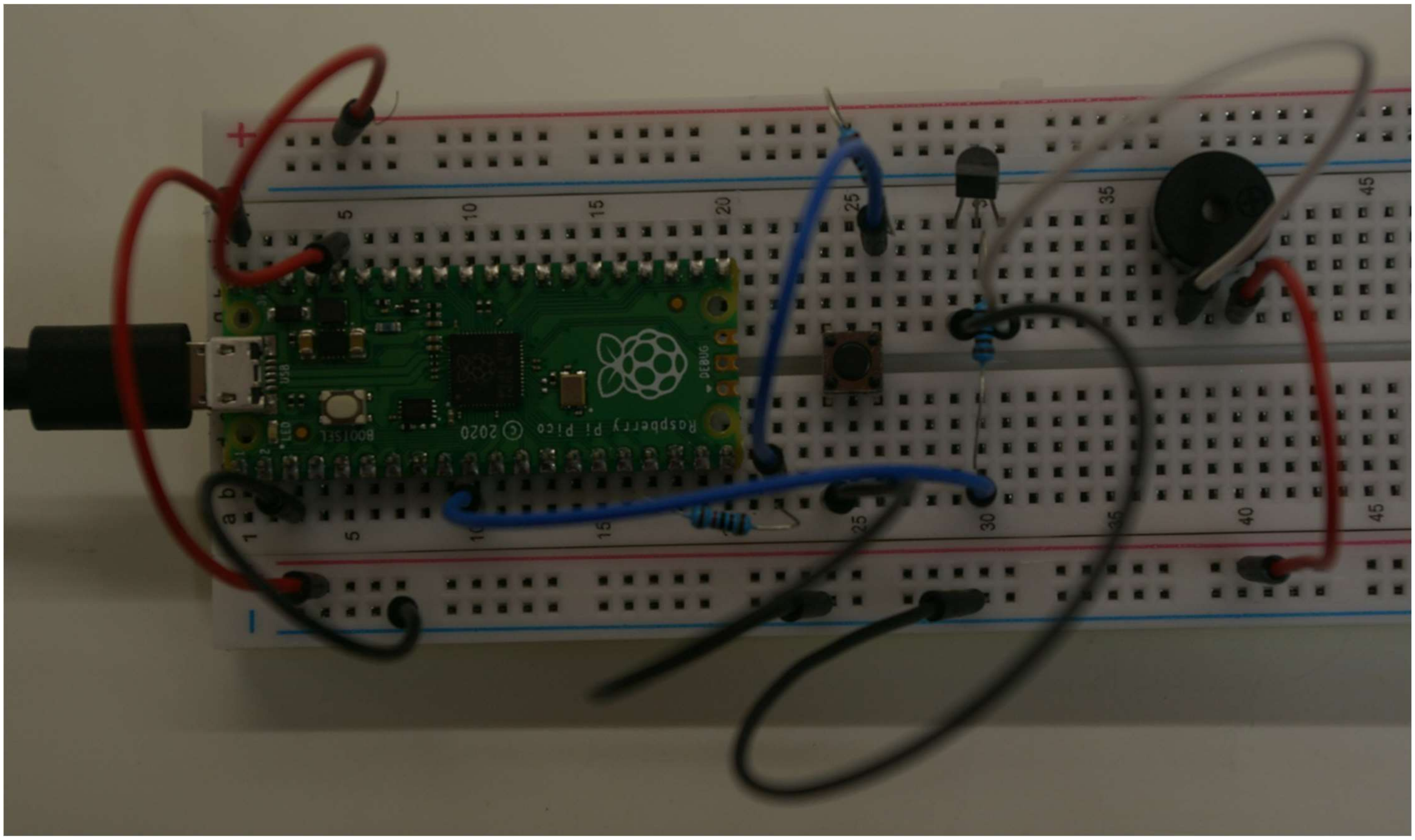
Lab03Program01.py

```python
# PgP 7/4/2022 software turns on passive buzzer with varying tone for 3 seconds
from machine import Pin,PWM
import math
import utime

PI = math.pi                              # use constant pi
button = Pin(13, Pin.IN, Pin.PULL_UP)     # button controls buzzer by GPIO 13
passiveBuzzer = PWM(Pin(7))               # Passive Buzzer powered by GPIO 7
passiveBuzzer.freq(1000)                  # Passive Buzzer base frequency

def alert():                              # creates sinusoidal frequency for buzzer
    for x in range(0, 36):
        sinVal  = math.sin(x * 10 * PI / 180)
        toneVal = 1500+int(sinVal*500)
        passiveBuzzer.freq(toneVal)
        utime.sleep_ms(50)  # time delay modifies sound patterns, default is 10

print("Setup complete. \n")
t = utime.ticks_ms()    # set start time
print("An alert will sound for 3 seconds! \n")

try:
    while True:
        if not utime.ticks_diff(utime.ticks_ms(), t) >= 3000:   # execute for 3 sec
            button.value(1)                   # turn on virtual button
            passiveBuzzer.duty_u16(4092*2)
            alert()                           #  call sinusoidal frequency function
        else:                                           # after 3 seconds turn off buzzer
            print("The test is complete.")
            passiveBuzzer.duty_u16(0)
            button.value(0)                             # turn off power to virtual button
            exit()                                      # exit the program

except:
    passiveBuzzer.deinit()       # free up resources
```

Shell

```
>>> %Run -c $EDITOR_CONTENT
 Setup complete.

 An alert will sound for 3 seconds!

 The test is complete.
>>>
```

Piezoelectric Buzzer Sounds for 3 Seconds, Then Stops



Piezoelectric Passive Buzzer with Push Button Switch Circuit

Use the NEXT (>) arrow to submit screenshot to Assignment Folder (formerly Dropbox).

Assessment - Lab03 Program02

You will need the following prerequisites:

1. Computer with full permission (i.e., Administrator) and Thonny installed
2. Raspberry Pi Pico with soldered headers (any board type)
3. Breadboard with same components as Lab03 Program01
4. Micro USB cable (or whatever cable required by your Pico device)

STEPS

For this lab you will use the same circuit as in Lab03 Program01. No additional hardware is needed.

STEPS

Create a program that sounds the piezoelectric passive buzzer with a varying tone when the push button switch is pressed. Print statements to the shell after setup is complete telling the user how to proceed, and the current time in year, month, date, hour, minute, second, day of week, day of year format.

1. From the Thonny menu select File>New and type the program code below:

```
from machine import Pin,PWM
import math
import utime

PI = math.pi                                # use constant pi
button = Pin(13, Pin.IN, Pin.PULL_UP)       # button controls buzzer by GPIO 13
passiveBuzzer = PWM(Pin(7))                 # Passive Buzzer powered by GPIO 7
passiveBuzzer.freq(1000)                    # Passive Buzzer base frequency


def alert():                                # create sinusoidal frequency for buzzer
    for x in range(0, 36):
        sinVal  = math.sin(x * 10 * PI / 180)
        toneVal = 1500+int(sinVal*500)
        passiveBuzzer.freq(toneVal)
        utime.sleep_ms(10)  # time delay modifies sound patterns, default is 10


print("Setup complete. Press button to sound the alert! \n")
print("Year,Month,Date,Hour,Minute,Second, Day of Week, Day of Year")
print(utime.localtime())

try:
    while True:              # infinite loop, pressing button energizes buzzer
        if not button.value():
            passiveBuzzer.duty_u16(4092*2)  # set buzzer volume
            alert()
        else:
            passiveBuzzer.duty_u16(0)   # turn off buzzer
```

```
except:
    passiveBuzzer.deinit()  # free up resources
```

Use the NEXT (>) arrow to submit screenshot to Assignment Folder (formerly Dropbox).

Assessment - Lab03 Program03

You will need the same prerequisites from Lab03 Program01:

1) Computer with full permission (i.e., Administrator) and Thonny installed
2) Raspberry Pi Pico with soldered headers (any board type)
3) Breadboard
4) Micro USB cable (or whatever cable required by your Pico device)
5) Piezoelectric Passive Buzzer
6) 10k ohm resistors, quantity 2
7) 1k ohm resistor
8) NPN Bipolar Junction Transistor (BJT) (8050)
9) Black jumper wires, quantity 3
10) Red jumper wires, quantity 3
11) Blue jumper wires, quantity 2
12) White jumper wire
13) Push button switch

STEPS

Use the same physical circuit as Lab03 Program01.

1) Using Lab03Program01.py create a new file, Lab03Program03.py by modifying code in the alert() function that changes the output tone pattern when the push button switch is pressed. Note that there are several different ways to accomplish this. Modifying the statements on lines 13, 14 and 16 can all affect the tone, and can be done in combination. Feel free to try several ways. The simplest method is to modify the sleep time in line 16 as described in Lab03Program01. Another way is to vary the base and variable frequency values in line 14.

```
from machine import Pin,PWM
import math
import utime

PI = math.pi                              # use constant pi
button = Pin(13, Pin.IN, Pin.PULL_UP)     # button controls buzzer by GPIO 13
passiveBuzzer = PWM(Pin(7))               # Passive Buzzer powered by GPIO 7
passiveBuzzer.freq(1000)                  # Passive Buzzer base frequency

def alert():                              # creates sinusoidal frequency for buzzer
    for x in range(0, 36):
        sinVal  = math.sin(x * 10 * PI / 180)
        toneVal = 3500+int(sinVal*1000)   # MODIFY THE BASE AND VARIABLE FREQUENCY
        passiveBuzzer.freq(toneVal)
        utime.sleep_ms(50)  # time delay modifies sound patterns, default is 10


print("Setup complete. \n")
t = utime.ticks_ms()    # set start time
```

```
print("An alert will sound for 3 seconds! \n")

try:
    while True:
        if not utime.ticks_diff(utime.ticks_ms(), t) >= 3000:   # execute for 3 sec
            button.value(1)                  # turn on virtual button
            passiveBuzzer.duty_u16(4092*2)
            alert()                          #  call sinusoidal frequency function
        else:                                          # after 3 seconds turn off buzzer
            print("The test is complete.")
            passiveBuzzer.duty_u16(0)
            button.value(0)                            # turn off power to virtual button
            exit()                                     # exit the program

   except:
       passiveBuzzer.deinit()       # free up resources
```

Use the NEXT (>) arrow to submit screenshot to Assignment Folder (formerly Dropbox).

Assessment - Lab03 Program04

You will need the same prerequisites as Lab03 Program02:

1. Computer with full permission (i.e., Administrator) and Thonny installed
2. Raspberry Pi Pico with soldered headers (any board type)
3. Breadboard with same components as Lab03 Program01
4. Micro USB cable (or whatever cable required by your Pico device)

STEPS

Use the same circuit as in Lab03 Program01. No additional hardware is needed.

1. Using Lab03Program02.py create a new file, Lab03Program04.py by modifying the code in the alert() function that changes how long the piezoelectric passive buzzer tone persists after the push button switch is released. Hint: modify line 13. Realize that the 'for loop' executes 36 times with a delay of 10 milliseconds in Lab03Program02.py. Therefore, the buzzer tone persists for 36 x 10 or 360 milliseconds-which is just under half a second.

```
from machine import Pin,PWM
import math
import utime

PI = math.pi                                # use constant pi
button = Pin(13, Pin.IN, Pin.PULL_UP)       # button controls buzzer by GPIO 13
passiveBuzzer = PWM(Pin(7))                 # Passive Buzzer powered by GPIO 7
passiveBuzzer.freq(1000)                    # Passive Buzzer base frequency


def alert():                                # create sinusoidal frequency for buzzer
    for x in range(0, 200):        # INCREASE LOOP COUNTER TO INCREASES BUZZER ON TIME
        sinVal  = math.sin(x * 10 * PI / 180)
        toneVal = 1500+int(sinVal*500)
        passiveBuzzer.freq(toneVal)
        utime.sleep_ms(10)  # time delay modifies sound patterns, default is 10


print("Setup complete. Press button to sound the alert! \n")
print("Year,Month,Date,Hour,Minute,Second, Day of Week, Day of Year")
print(utime.localtime())

try:
    while True:              # infinite loop, pressing button energizes buzzer
        if not button.value():
            passiveBuzzer.duty_u16(4092*2)  # set buzzer volume
            alert()
        else:
            passiveBuzzer.duty_u16(0)   # turn off buzzer
```

```
except:
    passiveBuzzer.deinit()  # free up resources
```

Use the NEXT (>) arrow to submit screenshot to Assignment Folder (formerly Dropbox).

Assessment – Quiz 3

1) What is a NPN Bipolar Junction Transistor?
2) Describe a piezoelectric buzzer
3) What are the 5 band colors on a 1k-ohm resistor?

Use the NEXT (>) arrow to submit quiz answers to Assignment Folder (formerly Dropbox).

FINAL ASSIGNMENT

Using your Raspberry Pi Pico, breadboard and photoresistor along with required jumper wires and resistors create a MicroPython program that reads the light level and prints the reading to the Shell every second.

Use the NEXT (>) arrow to submit screenshot to Assignment Folder (formerly Dropbox).

FINAL EXAM

1) Describe a push button switch
2) Provide an example of a digital actuator.
3) Describe an output device
4) Describe an example of a digital actuator.
5) Describe a jumper wire
6) What are the 5 band colors on a 220-ohm resistor?
7) What are the 5 band colors on a 10k-ohm resistor?
8) Describe an Analog Actuator
9) Describe an Analog Sensor
10) What is an Analog-to-Digital Converter?
11) Describe a thermistor
12) What is a NPN Bipolar Junction Transistor?
13) Describe a piezoelectric buzzer
14) What are the 5 band colors on a 1k-ohm resistor?

Use the NEXT (>) arrow to submit quiz answers to Assignment Folder (formerly Dropbox).

Summary and Additional Resources

Summary

This module covers three main objectives:

1. Validate IoT digital actuators
2. Validate IoT analog sensors
3. Validate IoT analog actuators

Additional Resources

Whenever possible, pictures have been provided. Links to the original source material contain more information about the topic.

Instructor Resources

Faculty Notes

Pre-requisite: Completion of IoT Software Development Concepts or prior programming experience is recommended. Students must own a personal computer (i.e., PC) or have access to computer (i.e., Mac OS) with full permissions to install software. Students must buy a Raspberry Pi Pico kit to practice exercises (about $25/kit) using link below, or equivalent: https://www.amazon.com/gp/product/B09XHTHZ8N

Quiz 1 Key

1) Describe a push button switch

Push Button Switch: A momentary or non-latching switch which causes a temporary change in the state of an electrical circuit only while the switch is physically actuated.

2) Provide an example of a digital actuator.

Digital Actuator: A piece of computer hardware equipment which turns a digital electrical control signal into a human-perceptible form, such as a light-emitting diode (LED).

3) Describe an output device

Output Device: An output device is any piece of computer hardware equipment which converts information into a human-perceptible form.

4) Describe an example of a digital actuator.

Light Emitting Diode (LED): A light-emitting diode (LED) is a semiconductor light source that emits light when current flows through it.

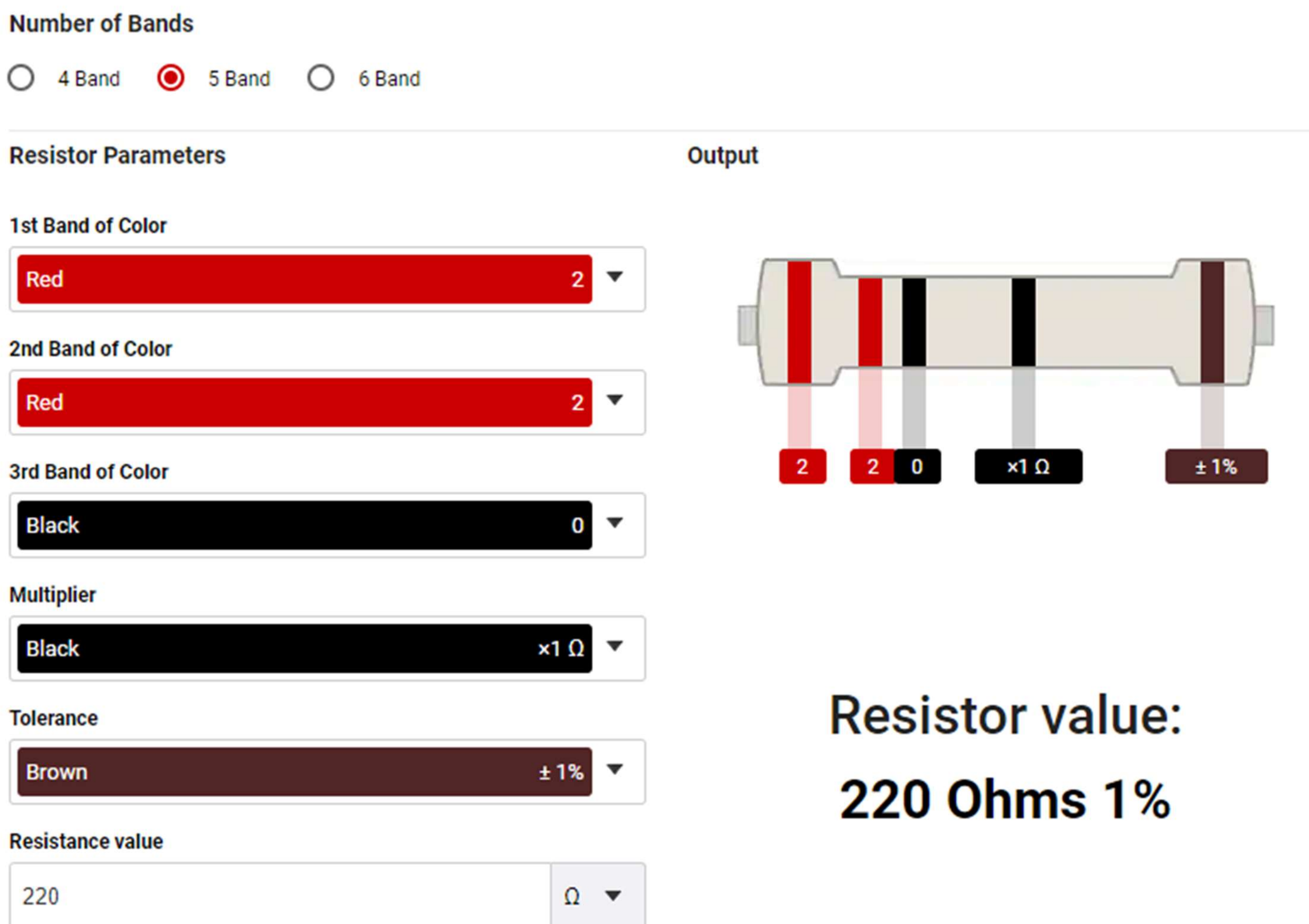
5) Describe a jumper wire

Jumper wire: A jumper wire is an electrical wire with a connector or pin at each end used to interconnect the components of a breadboard

6) What are the 5 band colors on a 220-ohm resistor?

220 ohm Resistor Band Colors: Refer to the Digi-Key Band Resistor Color Code Calculator at: https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-resistor-color-code

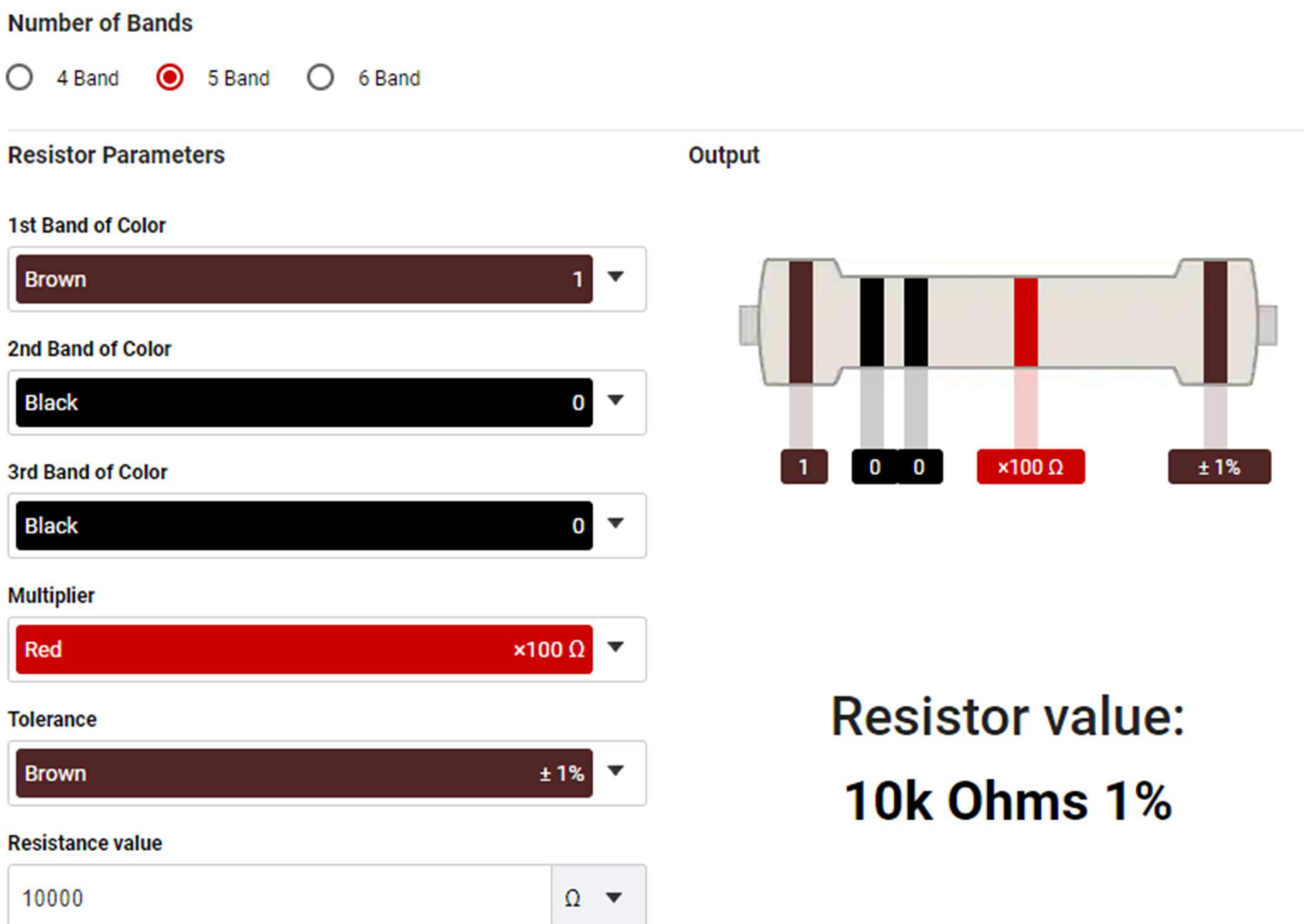
For a 5 band resistor- First Red band equates to '2' and Second Red band equates to '2' and Third Black band equates to '0' and Fourth Black band has a multiplier of '10' so the correct bands are:
Red(2) Red(2) Black(0) Black(1) = 220 ohms
The fifth band is the tolerance, which is probably 1%(Brown)



7) What are the 5 band colors on a 10k-ohm resistor?

10k ohm Resistor Band Colors: Refer to the Digi-Key Band Resistor Color Code Calculator at: https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-resistor-color-code

For a 5 band resistor- First Brown band equates to '1' and Second Black band equates to '0' and Third Black band equates to '0' and Fourth Red band has a multiplier of '100' so the correct bands are:
Brown(1) Black(0) Black(0) Red(100) = 10k ohms
The fifth band is the tolerance, which is probably 1%(Brown)



Source: Digi-Key.com

Quiz 2 Key

1) Describe an Analog Actuator

Analog Actuator: Any piece of computer hardware equipment which turns an analog electrical control signal into a human perceptible form. It includes lights, speakers, linear actuators, rotary actuators, and other sensory technologies.

2) Describe an Analog Sensor

Analog Sensor: A device that produces a variable output signal for the purpose of sensing a physical phenomenon. (Wikipedia.org).

3) What is an Analog-to-Digital Converter?

Analog-to-Digital Converter: A system that converts an analog signal, such as a sound picked up by a microphone or light entering a digital camera, into a digital signal.

4) Describe a thermistor

Thermistor: A type of resistor whose resistance is strongly dependent on temperature, more so than in standard resistors.

Quiz 3 Key

1) What is a NPN Bipolar Junction Transistor?

NPN Bipolar Junction Transistor: A semiconductor device that allows a small current injected at one of its terminals to control a much larger current flowing between the terminals, making the device capable of amplification or switching.

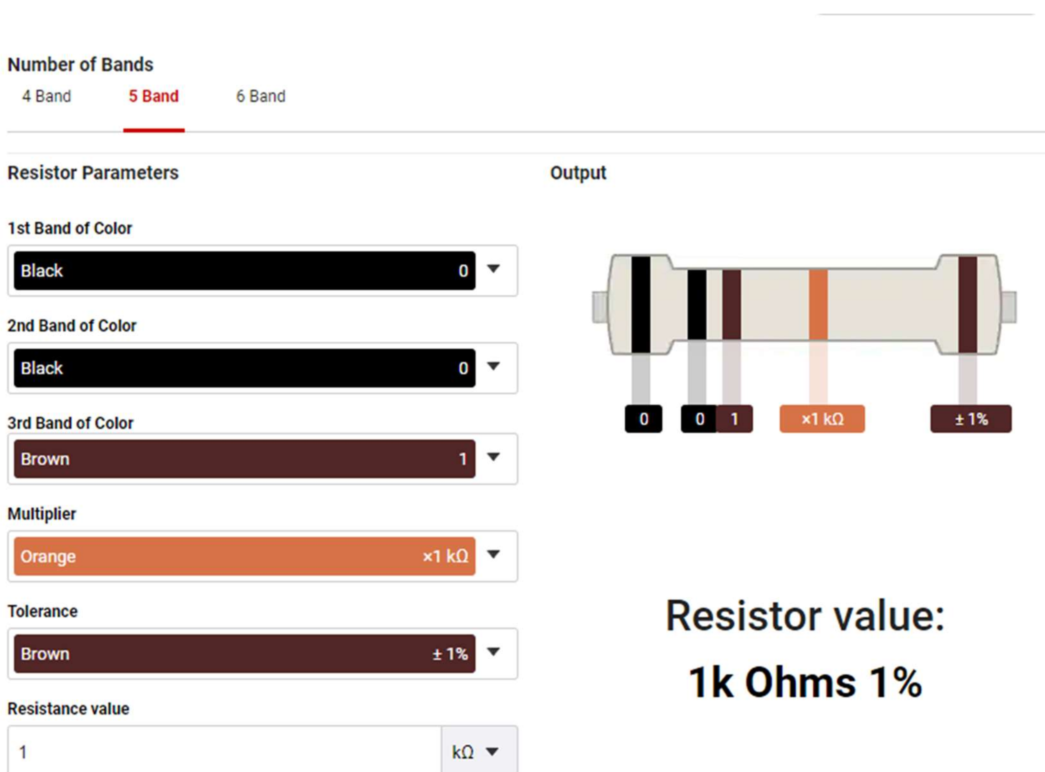
2) Describe a piezoelectric buzzer

Piezoelectric buzzer: An audio signaling device that uses the piezoelectric effect driven by an oscillating circuit. Typically used to confirm user input.

3) What are the 5 band colors on a 1k-ohm resistor?

1k ohm Resistor Band Colors: Refer to the Digi-Key Band Resistor Color Code Calculator at:
https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-resistor-color-code

For a 5 band resistor- First Black band equates to '0' and Second Black band equates to '0' and Third Brown band equates to '1' and Fourth Orange band has a multiplier of '1k' so the correct bands are:
Black(0) Black(0) Brown(1) Orange(1k) = 1k ohms
The fifth band is the tolerance, which is probably 1%(Brown)



FINAL EXAM KEY

1) Describe a push button switch

Push Button Switch: A momentary or non-latching switch which causes a temporary change in the state of an electrical circuit only while the switch is physically actuated.

2) Provide an example of a digital actuator.

Digital Actuator: A piece of computer hardware equipment which turns a digital electrical control signal into a human-perceptible form, such as a light-emitting diode (LED).

3) Describe an output device

Output Device: An output device is any piece of computer hardware equipment which converts information into a human-perceptible form.

4) Describe an example of a digital actuator.

Light Emitting Diode (LED): A light-emitting diode (LED) is a semiconductor light source that emits light when current flows through it.

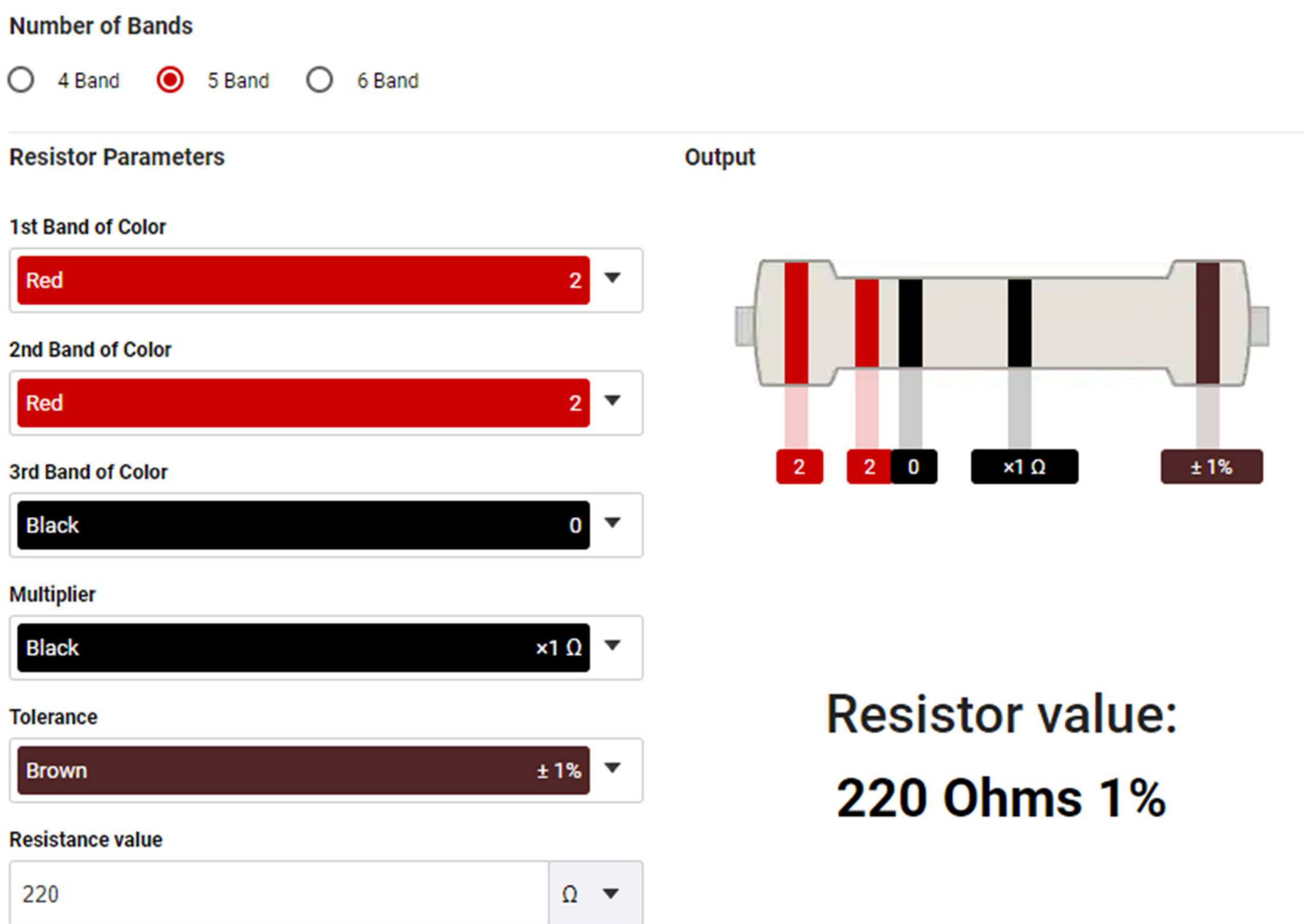
5) Describe a jumper wire

Jumper wire: A jumper wire is an electrical wire with a connector or pin at each end used to interconnect the components of a breadboard

6) What are the 5 band colors on a 220-ohm resistor?

220 ohm Resistor Band Colors: Refer to the Digi-Key Band Resistor Color Code Calculator at: https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-resistor-color-code

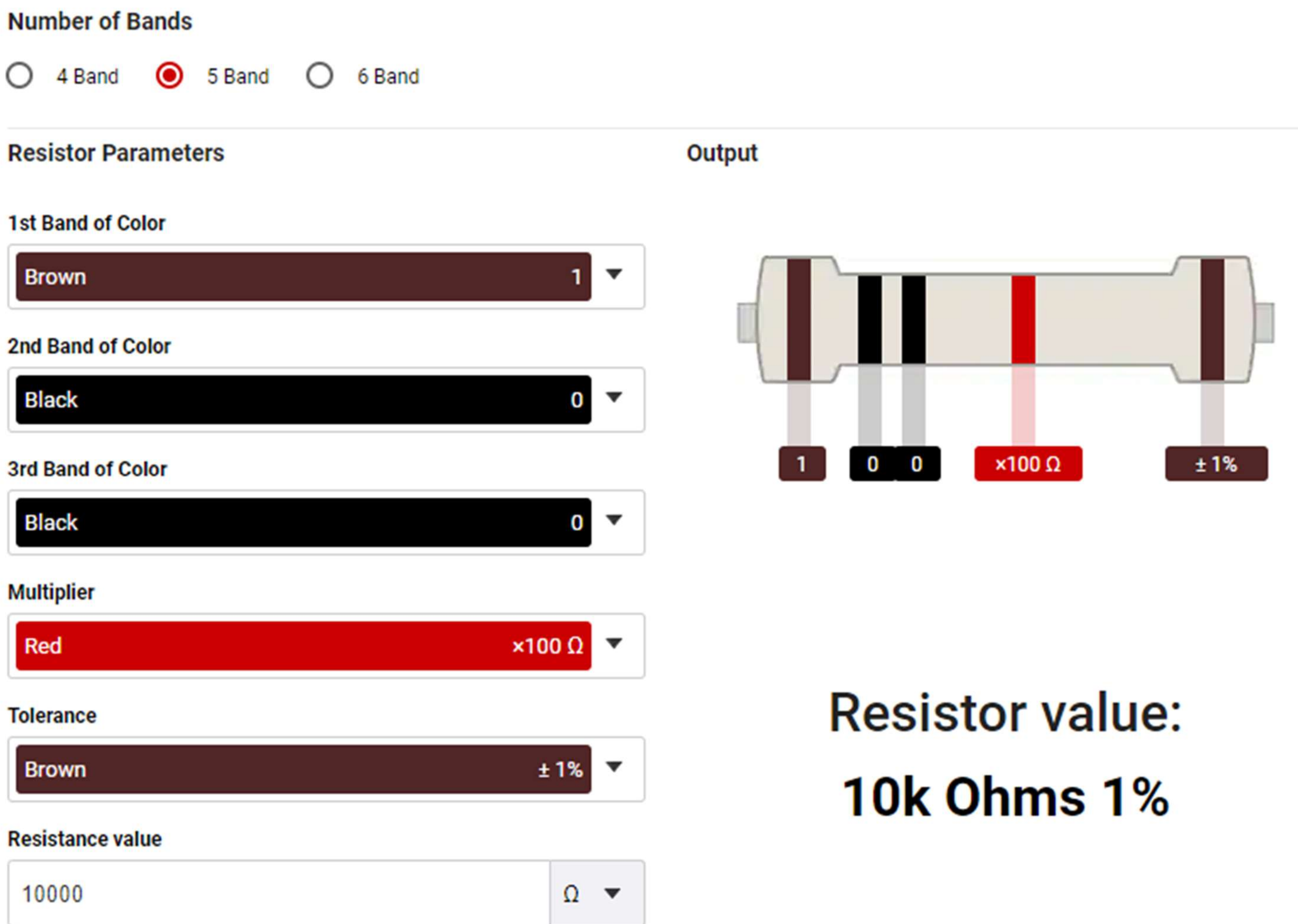
For a 5 band resistor- First Red band equates to '2' and Second Red band equates to '2' and Third Black band equates to '0' and Fourth Black band has a multiplier of '10' so the correct bands are:
Red(2) Red(2) Black(0) Black(1) = 220 ohms
The fifth band is the tolerance, which is probably 1%(Brown)



7) What are the 5 band colors on a 10k-ohm resistor?

10k ohm Resistor Band Colors: Refer to the Digi-Key Band Resistor Color Code Calculator at: https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-resistor-color-code

For a 5 band resistor- First Brown band equates to '1' and Second Black band equates to '0' and Third Black band equates to '0' and Fourth Red band has a multiplier of '100' so the correct bands are:
Brown(1) Black(0) Black(0) Red(100) = 10k ohms
The fifth band is the tolerance, which is probably 1%(Brown)



Source: Digi-Key.com

8) Describe an Analog Actuator

Analog Actuator: Any piece of computer hardware equipment which turns an analog electrical control signal into a human perceptible form. It includes lights, speakers, linear actuators, rotary actuators, and other sensory technologies.

9) Describe an Analog Sensor

Analog Sensor: A device that produces a variable output signal for the purpose of sensing a physical phenomenon. (Wikipedia.org).

10) What is an Analog-to-Digital Converter?

Analog-to-Digital Converter: A system that converts an analog signal, such as a sound picked up by a microphone or light entering a digital camera, into a digital signal.

11) Describe a thermistor

Thermistor: A type of resistor whose resistance is strongly dependent on temperature, more so than in standard resistors.

12) What is a NPN Bipolar Junction Transistor?

NPN Bipolar Junction Transistor: A semiconductor device that allows a small current injected at one of its terminals to control a much larger current flowing between the terminals, making the device capable of amplification or switching.

13) Describe a piezoelectric buzzer

Piezoelectric buzzer: An audio signaling device that uses the piezoelectric effect driven by an oscillating circuit. Typically used to confirm user input.

14) What are the 5 band colors on a 1k-ohm resistor?

1k ohm Resistor Band Colors: Refer to the Digi-Key Band Resistor Color Code Calculator at: https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-resistor-color-code

For a 5 band resistor- First Black band equates to '0' and Second Black band equates to '0' and Third Brown band equates to '1' and Fourth Orange band has a multiplier of '1k' so the correct bands are:
Black(0) Black(0) Brown(1) Orange(1k) = 1k ohms
The fifth band is the tolerance, which is probably 1%(Brown)