Kennedy Anukam
CPE 406
Assignment 8
April 22, 2021

Assignment Description:

In the first portion of the lab, I will build Project 6.1 described in the Freenove guide. In this project, I will create a doorbell. When the push button switch is pressed, the buzzer should sound. I will utilize the code provided in the Freenove guide to achieve the functionality. What I hope to learn from this section of the lab is how an active buzzer works.

In the next part of the lab, I will build Project 6.2 described in the Freenove guide. In this part of the lab, I will create the buzzer alarm circuit. Alongside the circuit, I will utilize the code provided by Freenove to complete the functionality. I will replace the active buzzer with the passive buzzer and leave the rest of the circuit alone. What I hope to learn from this section of the lab is how a passive buzzer differs from an active buzzer.

Problems Encountered:

A problem that I encountered in the lab had to do with using the wrong piece of hardware in Project 6.1. I was initially using the PNP transistor when I was supposed to be using the NPN transistor. I fixed this problem by double checking the part number and realizing I was using the 8550 transistor instead of the 8050 transistor.

Lessons Learned:

A lesson that I learned from this lab was how transistors work in more detail. I learned that transistors can be used to act as a switch. In this lab, an NPN transistor is being used to drive the buzzer. When the GPIO pin is high, current flows through the resistor and the transistor conducts current. The transistor is acting as a switch for the buzzer.

Description of Completed Lab:

Project 6.1

Circuit Wiring

To begin wiring the circuit, I first added the active buzzer to the breadboard. Figure 1 shows a snippet of the circuit with the active buzzer.

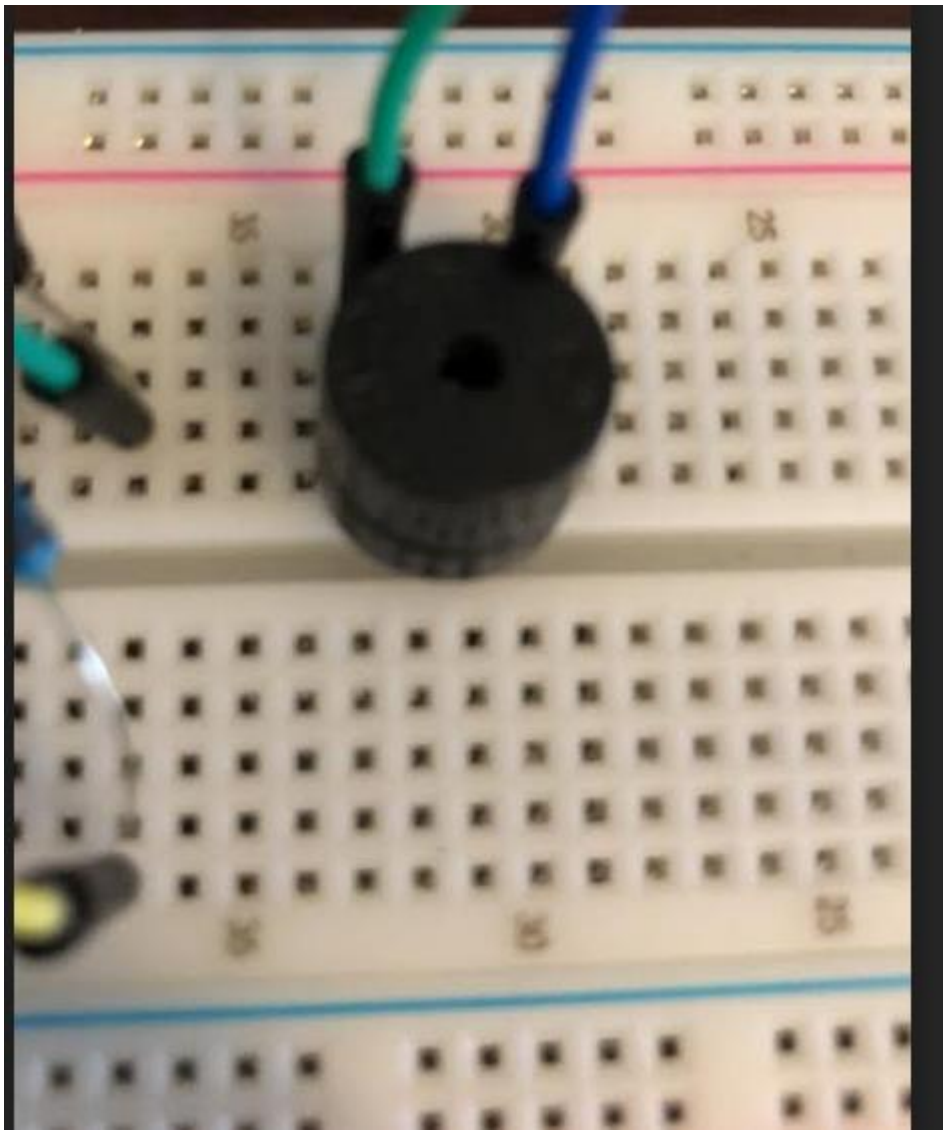

Figure 1: The active buzzer connected to the breadboard. The connections are to the 5V pin and the transistor.

The next part of the circuit that I added was the transistor. The transistor has three pins, the base, collector, and emitter. Figure 2 shows a snippet of the circuit including the transistor.

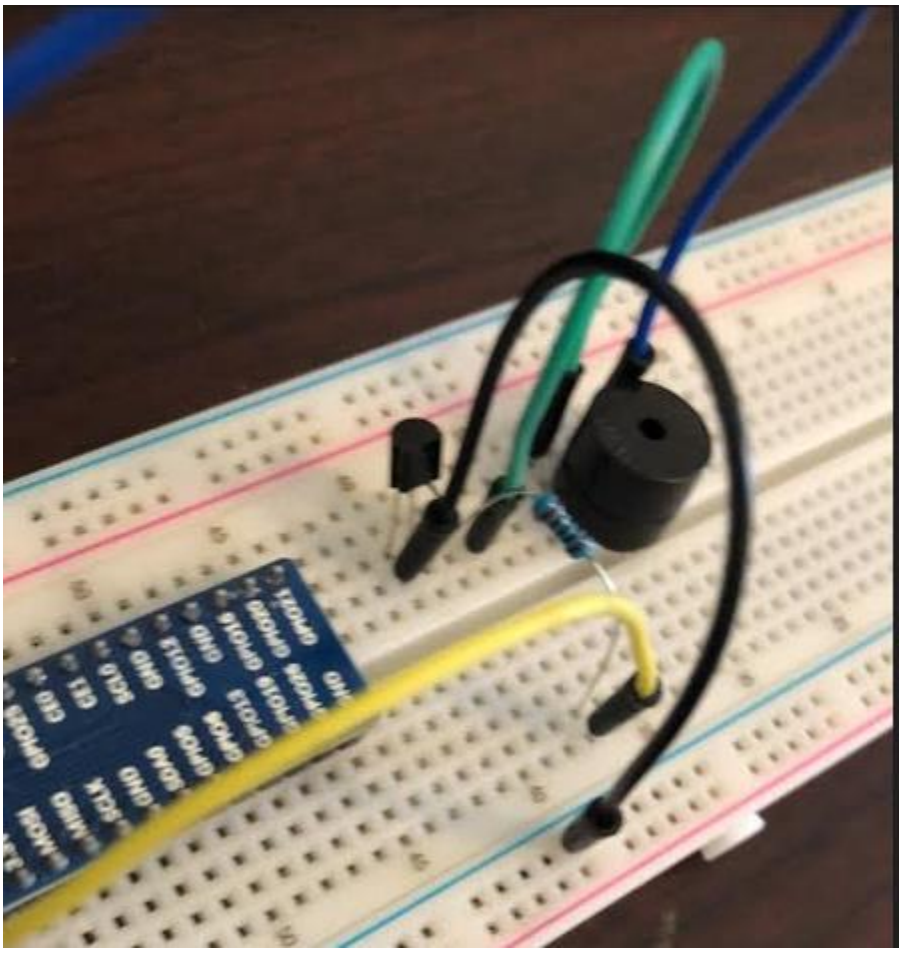

Figure 2: The transistor added to the circuit. The transistor is connected to the buzzer.

The final part of the circuit is the push button switch with a pull-up resistor. Figure 3 shows the push button switch utilized in the circuit.

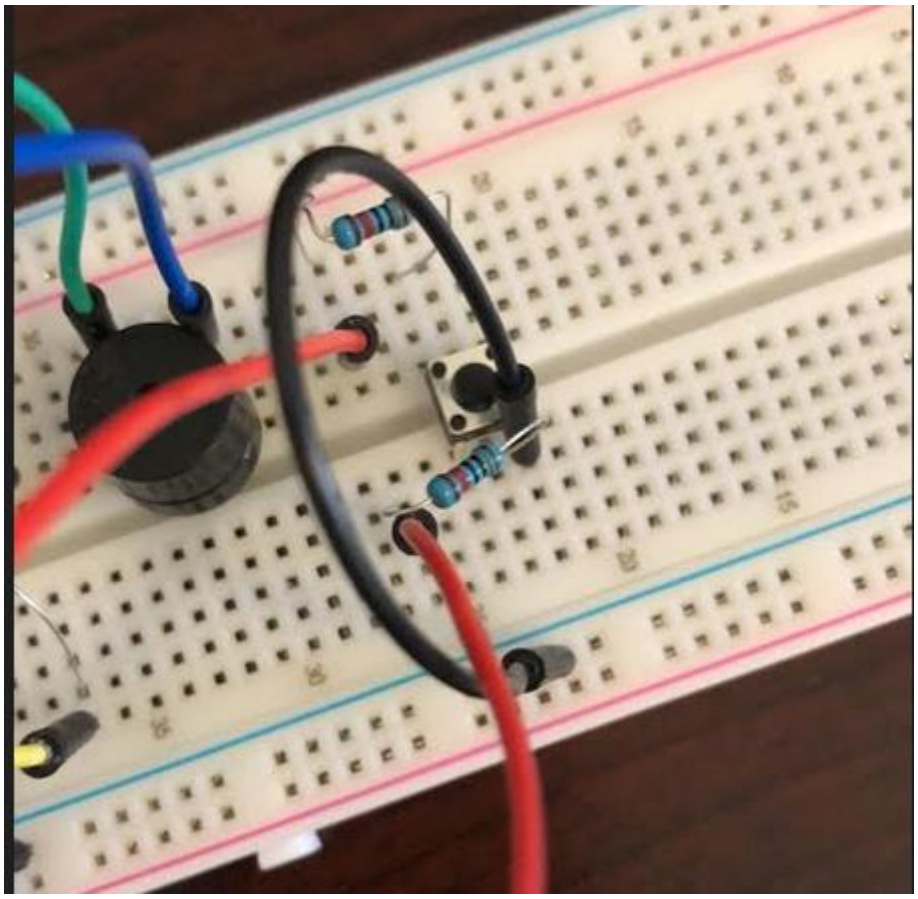

Figure 3: The push button switch for the circuit. The switch is connected to a GPIO pin which drives the transistor GPIO pin high when pressed utilizing the code.

Figure 4 shows the entire circuit.

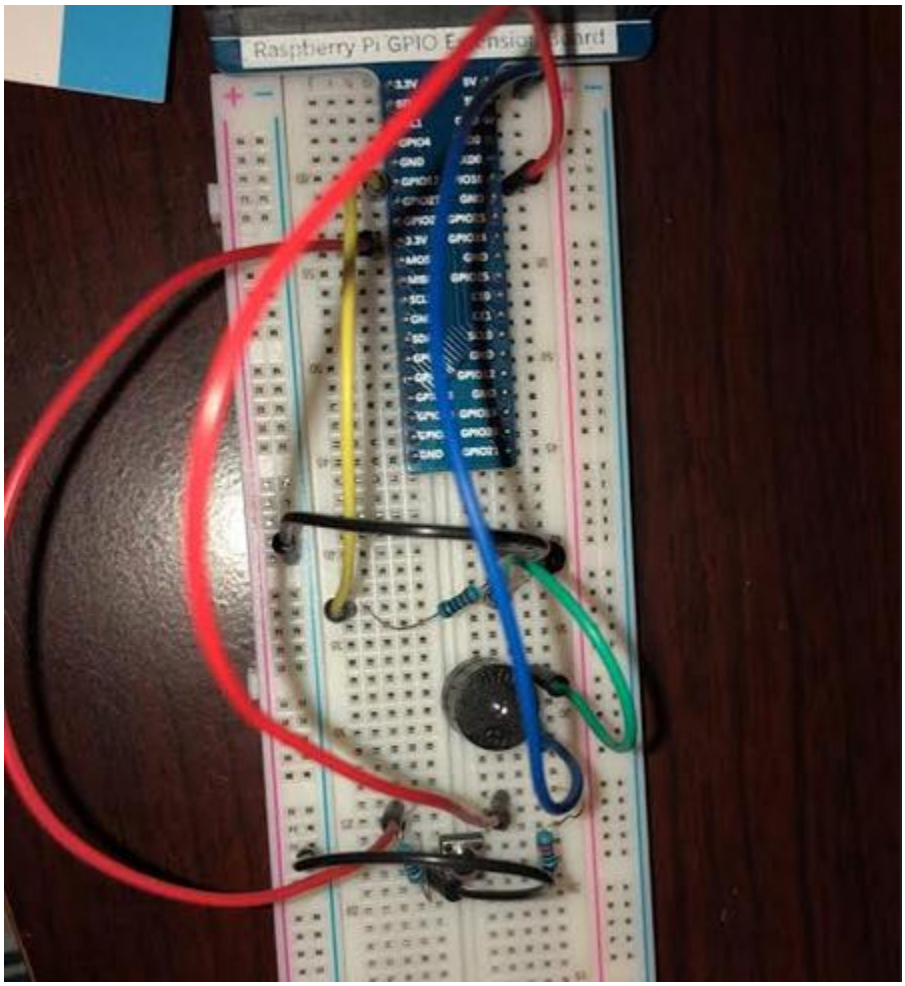

Figure 4: The complete circuit to accomplish the doorbell functionality.

Code & Results

How the code works is when the pull up resistor is pressed, the pin connected to the active buzzer is driven high. This triggers the buzzer to make noise.

```
#!/usr/bin/env python3
########################################################################
# Filename    : Doorbell.py
# Description : Make doorbell with buzzer and button
# auther      : www.freenove.com
# modification: 2019/12/28
########################################################################
# modified by K. Anukam 4/22/2021 rev 1.2
import RPi.GPIO as GPIO

buzzerPin = 11    # define buzzerPin
buttonPin = 12    # define buttonPin

def setup():
    print("Kennedy Anukam HW 8")
    GPIO.setmode(GPIO.BOARD)        # use PHYSICAL GPIO Numbering
    GPIO.setup(buzzerPin, GPIO.OUT)   # set buzzerPin to OUTPUT mode
    GPIO.setup(buttonPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)    # set
buttonPin to PULL UP INPUT mode

def loop():
    while True:
        if GPIO.input(buttonPin)==GPIO.LOW: # if button is pressed
            GPIO.output(buzzerPin,GPIO.HIGH) # turn on buzzer
            print ('buzzer turned on >>>')
        else : # if button is relessed
            GPIO.output(buzzerPin,GPIO.LOW) # turn off buzzer
            print ('buzzer turned off <<<')

def destroy():
    GPIO.cleanup()                     # Release all GPIO

if __name__ == '__main__':     # Program entrance
    print ('Program is starting...')
    setup()
    try:
        loop()
    except KeyboardInterrupt:  # Press ctrl-c to end the program.
        destroy()
```

VIDEO

Project 6.1

Circuit Wiring

This circuit for this portion is identical to project 6.1 with 1 major distinction. The active buzzer is replaced with a passive buzzer. Figure 5 shows the altered circuit.

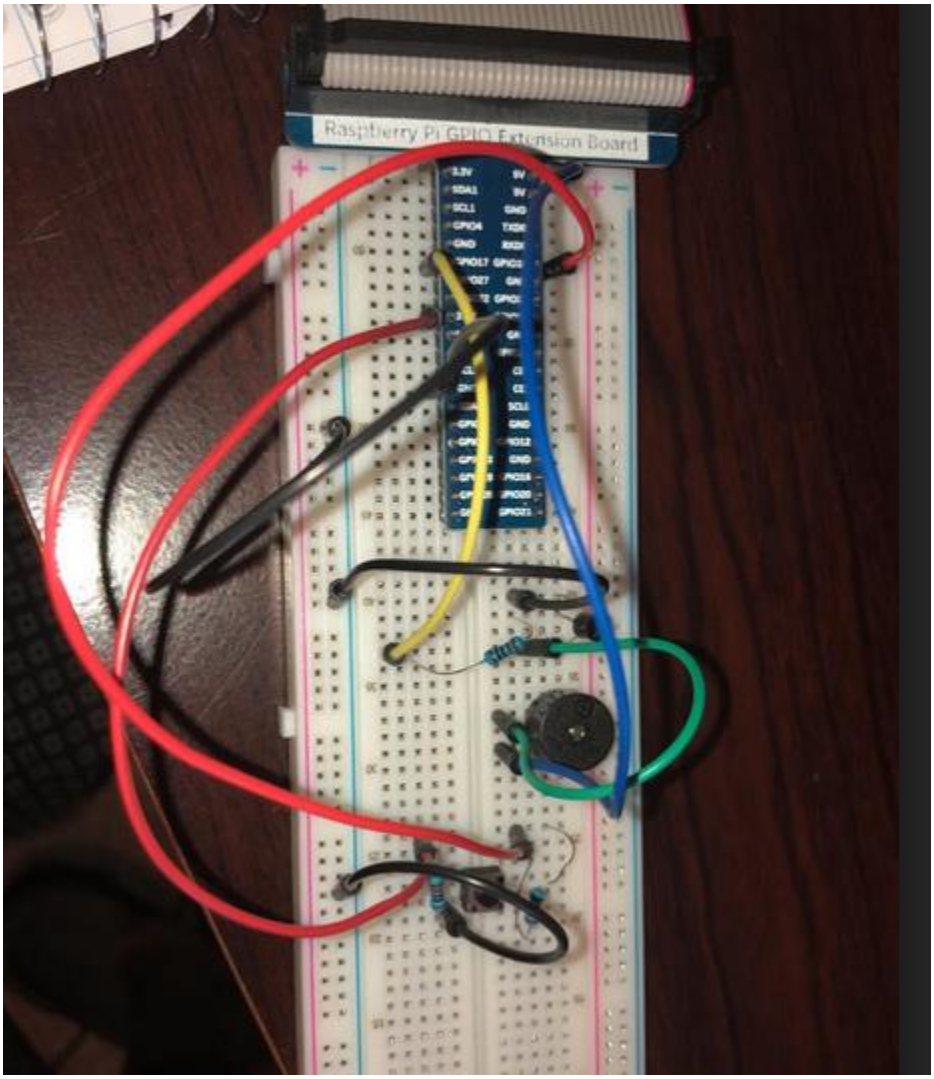

Figure 5: The modified circuit with the active buzzer replaced with a passive buzzer.

Code & Results

How the code works is when the pull up resistor is pressed, the pin connected to the passive buzzer is given a varying frequency.

```
#!/usr/bin/env python3
########################################################################
# Filename    : Alertor.py
# Description : Make Alertor with buzzer and button
# Author      : www.freenove.com
# modification: 2019/12/27
########################################################################
# modified by K. Anukam 2021/04/17 rev 1.2
import RPi.GPIO as GPIO
import time
import math

buzzerPin = 11  # define the buzzerPin
buttonPin = 12  # define the buttonPin


def setup():
    print("Kennedy Anukam AS07")
    global p
    GPIO.setmode(GPIO.BOARD)  # Use PHYSICAL GPIO Numbering
    GPIO.setup(buzzerPin, GPIO.OUT)  # set RGBLED pins to OUTPUT mode
    GPIO.setup(buttonPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)  # Set buttonPin
to INPUT mode, and pull up to HIGH level, 3.3V
    p = GPIO.PWM(buzzerPin, 1)
    p.start(0)


def loop():
    while True:
        if GPIO.input(buttonPin) == GPIO.LOW:
            alertor()
            print('alertor turned on >>> ')
        else:
            stopAlertor()
            print('alertor turned off <<<')


def alertor():
    p.start(50)
    for x in range(0, 361):  # Make frequency of the alertor consistent with
the sine wave
        sinVal = math.sin(x * (math.pi / 180.0))  # calculate the sine value
        toneVal = 2000 + sinVal * 500  # Add to the resonant frequency with a
Weighted
        p.ChangeFrequency(toneVal)  # Change Frequency of PWM to toneVal
        time.sleep(0.001)
```

```
def stopAlertor():
    p.stop()


def destroy():
    GPIO.output(buzzerPin, GPIO.LOW)  # Turn off buzzer
    GPIO.cleanup()  # Release GPIO resource


if __name__ == '__main__':  # Program entrance
    print('Program is starting...')
    setup()
    try:
        loop()
    except KeyboardInterrupt:  # Press ctrl-c to end the program.
        destroy()
```

VIDEO

Active Buzzer vs. Passive Buzzer

A primary difference between an active buzzer and a passive buzzer has to do with the oscillators from them. With active buzzers, the oscillator is built in. With passive buzzers, an external oscillator is needed. PWM can be used for passive buzzers to obtain different frequencies, thus outputting different sounds. Active buzzers usually only make one sound and are less limited on the range of sounds they can make in comparison to passive buzzers.

An application where a passive buzzer has an advantage over an active buzzer is an alarm clock. With some alarm clocks, various frequencies are used instead of just one sound. Since various frequencies are outputted to wake up the person, a passive buzzer has the advantage.

An application where an active buzzer has an advantage over a passive buzzer is a timer. With timers, usually only one sound is outputted. The sound is stopped and continued. An active buzzer has the advantage here since only one sound is being outputted.