San Francisco Bay University

CE450 Fundamentals of Embedded Engineering
Lab 4 Buzzer Control

Objectives:

In this lab, students will learn how to control on-board buzzer in Python program on Raspberry Pi board and do hands-on exercise through lab assignments.

Introduction:

One buzzer is available in Sunfounder accessory box. If a control signal generated from one of pins in GPIO port is logic low, it will switch buzzer on through a BJT, otherwise, off.

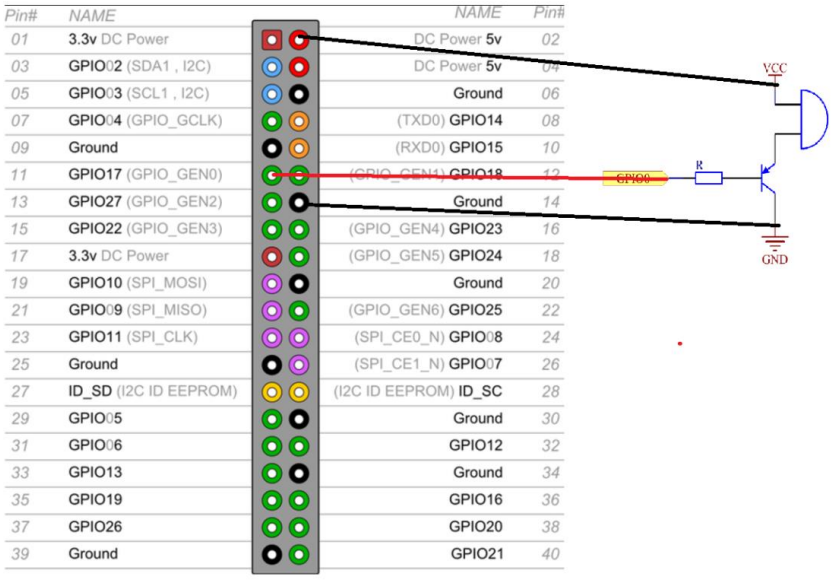
Equipment:

The equipment you require is as follows:

- Laptop & Raspberry Pi 4B model Board
- SunFounder Super Starter Kit V2.0 for Raspberry Pi

The Laboratory Procedure:

1. **Hardware connection**



2. **Control program in Python**

```
# Python program
import RPi.GPIO as GPIO
import time

BeepPin = 11    # pin11

def setup():
        GPIO.setmode(GPIO.BOARD)        # Numbers GPIOs by physical location
        GPIO.setup(BeepPin, GPIO.OUT)   # Set BeepPin's mode is output
        GPIO.output(BeepPin, GPIO.HIGH) # Set BeepPin high(+3.3V) to off beep

def loop():
        while True:
                GPIO.output(BeepPin, GPIO.LOW)      # Switch on Buzzer
                time.sleep(0.1)                     # 0.1s delay
                GPIO.output(BeepPin, GPIO.HIGH)
                time.sleep(0.1)

def destroy():
        GPIO.output(BeepPin, GPIO.HIGH)    # beep off
        GPIO.cleanup()                     # Release resource


print 'Press Ctrl+C to end the program...'

setup()
try:
        loop()
except  KeyboardInterrupt:  # When 'Ctrl+C' is pressed, the child program destroy()
will be executed.
        destroy()
```

**Note: Hardware connection reference and running command*
https://learn.sunfounder.com/lesson-6-buzzer/
https://learn.sunfounder.com/category/super-kit-v3-0-for-raspberry-pi/

The Laboratory Assignments:

1. Implement the buzzer control based on the above example program

2. Add one LED to the buzzer control circuit to make it on when the buzzer is ringing and off if the buzzer is silent

```
# Python program
import RPi.GPIO as GPIO
import time

BeepPin = 11    # pin11
ledPin = 15     # pin15     Led one pin to GND, another to GPIO

on = True
off = False

def setup():
          GPIO.setmode(GPIO.BOARD)         # Numbers GPIOs by physical location
          GPIO.setup(BeepPin, GPIO.OUT)    # Set BeepPin's mode is output
          GPIO.output(BeepPin, GPIO.HIGH) # Set BeepPin high(+3.3V) to off beep


           GPIO.setup(ledPin, GPIO.OUT)    # set ledPin's mode is output
           GPIO.output(ledPin, GPIO.LOW)    # set ledPin low to off the led

def loop():
          while True:
                    buzzer_ctrl(on)
                    time.sleep(0.1)
                    buzzer_ctrl(off)
                    time.sleep(0.1)

def buzzer_ctrl(switch):
           if(switch): # switch on buzzer to make both beep and led on
                    GPIO.output(BeepPin, GPIO.LOW) # Switch on Buzzer
                    GPIO.output(ledPin, GPIO.HIGH) # set ledPin high(+3.3) to on led

           else: # switch off buzzer to make both beep and led off
                    GPIO.output(BeepPin, GPIO.HIGH)
                    GPIO.output(ledPin, GPIO.LOW)


def destroy():
          GPIO.output(BeepPin, GPIO.HIGH)    # beep off
           GPIO.output(ledPin, GPIO.LOW)      # led off
          GPIO.cleanup()                     # Release resource



print 'Press Ctrl+C to end the program...'

setup()
try:
          loop()
except    KeyboardInterrupt:  # When 'Ctrl+C' is pressed, the child program
destroy() will be  executed.
          destroy()
```