CamJam EduKit Worksheet Six

Project Generate Morse code SOS using a Buzzer.

Description In this project, you will learn how to wire and sound a buzzer and use it to produce Morse code. You will be using user-defined 'functions'.

Equipment Required

- The circuit built in CamJam EduKit Worksheet Five, plus the following:
  - The Buzzer from the CamJam EduKit.
  - 2 x M/F jumper wires.

Additional Parts

You will be adding a buzzer to the LED and button circuit that you made in CamJam EduKit Worksheet Five. Let us look at the additional components.

Note: Please do not skip this section, as you need to know how to connect the buzzer.

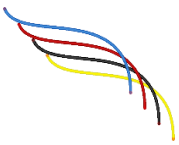
Buzzer

The buzzer supplied in the EduKit is an 'active' buzzer, which means that it only needs an electric current to make a noise. In this case, you are using the Raspberry Pi Pico to supply that current.

Note that the buzzer needs to be wired with the correct 'polarity' for it to function. Using the wrong wiring may damage it. The longer leg is positive, the shorter leg is negative.

Buzzers can be loud, so you may want to use adhesive tape to cover the 'opening hole' of the buzzer to reduce the volume!

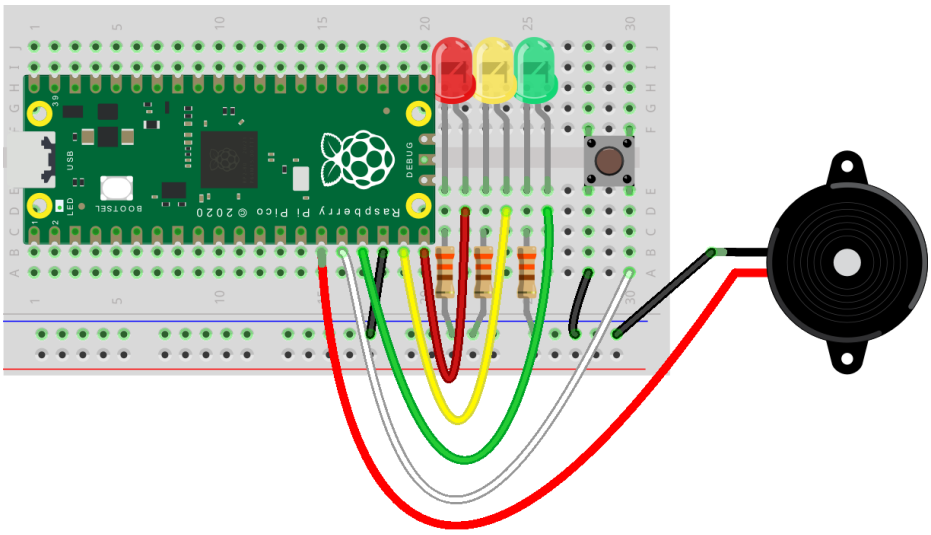
The Jumper Wires



You are not going to plug the buzzer into the breadboard for this circuit (because there isn't room!). Instead, you are going to use two M/F (male/female) jumper wires – these have a pin at one end and a hole at the other. The pins of the jumpers will go into the breadboard, and the pins of the buzzer will go into the holes at the other end of the jumpers.

The buzzer will freely move around, so be careful that it doesn't become detached from the jumper wires.

Building the Circuit



Before you connect additional components to the circuit, you should unplug the USB cable from your computer. Leave the LED and switch circuit from Worksheet 5 in place.

Pin GP11 will be an output pin, and when it is set to 'on', the buzzer will sound.

The longer leg of the buzzer is connected via the red jumper wire to pin GP11. The shorter leg is connected by the black jumper wire to the common ground rail.

After you finish adding the additional components, double check the wiring to ensure that the circuit is correct. Using the wrong wiring can damage the Pico and other components.

Concepts

You are going to be using a user-defined 'function' in the code below. These are blocks of code that you may want to run more than once. By using functions, you only need to write the code once, and 'call' that function from within your code each time you want to run that block of code.

To define a function, you first need to tell Python that you are writing a function. To do this, use the 'def' command followed by the name of the function (you can choose almost any name), followed by brackets and a colon, like so ():

For example:

```
def hello():
```

Everything indented after this line will be included in the function, for example:

```
    print("Hello World!")
```

To use the function, you must 'call' it by simply using the name of your new function with brackets:

```
hello()
```

Now, every time Python sees `hello()` in your code, it will print '`Hello World!`'

Code

Connect your Pico to your computer. Restart Thonny and create a new file. Type in the following code below exactly as seen, including the important indentation.

```
# CamJam Edukit 1 - Basic
# Worksheet 6 - Morse Code

# Import Libraries
import time  # Proves time related commands
from picozero import Buzzer  # The picozero Buzzer function

# Set pin GP11 as a buzzer
buzzer = Buzzer(11)

# Define some 'user-defined 'functions'
def dot():  # A single Morse dot
    buzzer.on()
    time.sleep(0.1)
    buzzer.off()
    time.sleep(0.1)

def dash():  # A single Morse dash
    buzzer.on()
    time.sleep(0.3)
    buzzer.off()
    time.sleep(0.1)

def letterSpace():  # The space between letters
    time.sleep(0.2)

def wordSpace():  # The space between words
    time.sleep(0.6)

def morseS():  # The Morse for S, ...
    dot()
    dot()
    dot()

def morseO():  # The Morse for O, ---
    dash()
    dash()
    dash()

print("Morse Code")
```

```
# Prompt the user for input
loop_count = input("How many times would you like to send an SOS?")
loop_count = int(loop_count)  # Ensure that loop_count is an integer

while loop_count > 0:  # Loop around the chosen number of times
    morseS()
    letterSpace()
    morseO()
    letterSpace()
    morseS()
    wordSpace()
    loop_count = loop_count - 1
```

Save the file to the Pico as `6-morsecode.py`.

Running the Code

Click the green Run icon on the top menu bar.

You will be prompted for the number of times you want to repeat 'SOS'.

Challenge

Using the above code as your template, write another program that will allow you to sound any Morse code you choose. Use the following rules:

- The length of a dot is one unit of time.
- The length of a dash is three units.
- The space between the parts of each letter is one unit.
- The space between letters is three units.
- The space between words is seven units.

The letter and number codes are:

A	•–	K	–•–	U	••–	0	–––––
B	–•••	L	•–••	V	•••–	1	•––––
C	–•–•	M	––	W	•––	2	••–––
D	–••	N	–•	X	–••–	3	•••––
E	•	O	–––	Y	–•––	4	••••–
F	••–•	P	•––•	Z	––••	5	•••••
G	––•	Q	––•–			6	–••••
H	••••	R	•–•			7	––•••
I	••	S	•••			8	–––••
J	•–––	T	–			9	––––•

Advanced Challenge

Using what you have learned so far, especially from Worksheet 5, make your own Morse machine by making the buzzer sound when you press the button that is already on the breadboard.

Note: Do not disassemble this circuit as it will be used in the following worksheets.