CamJam EduKit Worksheet Seven

Project Traffic Light Simulator for UK Pelican Crossing traffic lights.

Description In this project, you will program a traffic light simulator using the circuit from Worksheet Six.

Equipment Required

The circuit built in CamJam EduKit Worksheet Six which includes three LEDs, a button, and a buzzer.

Exercise

In this worksheet, you are going to program the Pico and EduKit to act like a standard UK Pelican Crossing traffic lights which uses a flashing amber light and a button press to allow pedestrians to cross. The LEDs will act as signals to the vehicles; the buzzer will act as a signal to the pedestrians.

The standard sequence is as follows:

Action	Signal to Vehicles	Signal to Pedestrians	Timings
	Steady Green	Red Standing Figure	
Pedestrian presses the button	Steady Amber (Cars should stop if they can)	Red Standing Figure	3 seconds
	Steady Red (Cars must stop)	Red Standing Figure	1 second
Pedestrians can start walking	Steady Red	Green Waking Figure and Beeping	4 to 7 seconds
Pedestrians should not start to cross	Steady Red	Flashing Green Figure and no sound	2 seconds
	Flashing Amber (Cars can start to go again if the crossing is clear)	Flashing Green Figure	6 seconds
	Flashing Amber	Red Standing Figure	1 second
	Steady Green (Cars can proceed)	Red Standing Figure	At least 20 seconds

On the last step, the button can be pressed during the 20 seconds, but the lights should not change immediately. After the 20 seconds, the process can start again.

Using your knowledge from the previous worksheets, the Code Hints, and the outline code, write your traffic light code.

Code Hints

Here are some reminders of some of the code you will need:

`import library`	Remember to import required libraries at the start of the code. You will need time and picozero.
`colour = LED(xx)`	Sets pin *xx* to be used as an LED pin.
`time.sleep(x)`	Wait for *x* seconds.
`colour.on()`	Turns on the LED called *colour.*

`while condition`
`  xxxx`
`  yyyy`
	While the condition is true (e.g. x<1), run the commands *xxxx* and *yyyy*.

`if condition:`
`  xxxx`
`else:`
`  yyyy`
	If *condition* is true, run code block *xxxx*, otherwise run code block *yyyy*.

Code Outline

Plug your Pico into your computer. Restart Thonny and create a new file. Type in the following code. There are comments where you need to fill in some code yourself.

```
# CamJam EduKit 1 - Basic
# Worksheet 7 - Traffic Lights Template

# Import Libraries
import time
from picozero import LED, Button, Buzzer

# Set up variables for the LED, Buzzer and switch pins

# Define a function for the initial state (Green LED on, the rest off)
# (If you have the second 'pedestrian LEDs,
# turn the red on & green off)
def startgreen():
    # Remember all code in the function is indented

# Turn the green off and the amber on for 3 seconds
# ('Pedestrian' red LED stays lit)
def steadyamber():
    # Remember all code in the function is indented

# Turn the amber off, and then the red on for 1 second
def steadyred():
    # Remember all code in the function is indented

# Sound the buzzer for 4 seconds
```

```
# (If you have the 'pedestrian' LEDs, turn the red off and green on)
def startwalking():
    # Make the buzzer buzz on and off, half a second of
    # sound followed by half a second of silence

# Turn the buzzer off and wait for 2 seconds
# (If you have a second green 'pedestrian' LED, make it flash on and
# off for the two seconds)
def dontwalk():
    # Remember all code in the function is indented

# Flash the amber on and off for 6 seconds
# (And the green 'pedestrian' LED too)
def flashingambergreen():
    # Remember all code in the function is indented

# Flash the amber for one more second
# (Turn the green 'pedestrian' LED off and the red on)
def flashingamber():
    # Remember all code in the function is indented

# Go through the traffic light sequence by calling each function
# one after the other.
def trafficlightqequence():
    # Remember all code in the function is indented

print("Traffic Lights")
# Initialise the traffic lights
startgreen()

# Here is the loop that waits at lease 20 seconds before
# stopping the cars if the button has been pressed
while True:  # Loop around forever
    buttonnotpressed = True  # Button has not been pressed
    start = time.time()  # Records the current time
    while buttonnotpressed:  # While the button has not been pressed
        time.sleep(0.1)  # Wait for 0.1s
        if button.ispressed:  # If the button is pressed
            now = time.time()
            buttonnotpressed = False  # Button has been pressed
            if (now - start) <= 20:  # If under 20 seconds
                time.sleep(20 - (now - start))  # Wait until 20s is up
                trafficlightqequence()  # Run the light sequence
```

Save the file to the Pico as `7-trafficlight.py`.

Running the Code

Click the green Run icon on the top menu bar.

If errors are reported, check your code, and edit it to fix the bugs.

Press the button on the breadboard while the green LED is lit and see what happens.

Debugging

Debugging is the process of finding out what 'error' is causing your Python code to either crash, or not behave in the manner you expect.

Syntax errors are violations of the 'grammar' of the Python language. They are easily detected by the Python Interpreter when you run the code. Line numbers of the code in question will be displayed.

'Logic errors' are ones caused by there being a difference in meaning between your thinking and the code you have written. What you 'mean' inside your head is one thing, but the Python Interpreter cannot read your mind (yet!). It only sees what you typed in. It looks at your code and then triggers actions in strict accordance with Python grammar rules.

There is a very simple debugging tool available to you. Just insert additional `print()` lines at 'important' places in the code. You can print out variable values and see how the code flows by using these statements.

Challenge 1

Use a second CamJam EduKit, or buy additional green and red LEDs and two 330Ω resistors and use them as the red and green pedestrian figures. Alter your traffic light code to show the red and green figures correctly as shown in the standard traffic light sequence.

Challenge 2

Change your code to simulate a 'Puffin Crossing' traffic lights which uses steady amber lights and different timings.

Action	Pedestrian	Traffic	Timings (Seconds)
Button Pressed	Red	Green	Minimum 6 to 15 Maximum 60
	Red	Amber	3
	Red	Red	1, 2 or 3
Pedestrians can cross	Green	Red	4 to 9
Pedestrians should not start crossing	Red	Red	1 to 5
	Red	Red	0 to 30
If pedestrians are still detected	Red	Red	0 to 3
	Red	Red	0 to 3
	Red	Red & Amber	2

Challenge 3 (If you have CamJam EduKit 2)

Puffin Crossing traffic lights have kerb side detectors monitoring the presence of pedestrians at the crossing. If someone presses the request button and then walks away, the request will be cancelled. Use the ultrasonic distance sensor from EduKit 2 to simulate this situation.

Challenge 4 (If you have CamJam EduKit 2)

Puffin Crossing traffic lights have an on-crossing detector ensuring that the signal for vehicles remains red until pedestrians have finished crossing (within certain limits). Use the ultrasonic distance sensor from EduKit 2 to simulate this situation.