Assignment 4

Assigned: Tuesday 05/14/2019

Due: Friday 05/24/2019, by 11:59PM

Turn in: Canvas

File to turn in: assignment4.py

Note 1: make sure to add comments on the status of the code.

Warnings:

- "*Before connecting power, get into the habit of checking that there is nothing conductive in contact that could cause a short circuit with you Raspberry Pi. A quick check that there is nothing nearby could save you from damaging your Pi."*

- *"Electricity can kill! Only experiment with low voltage and currents, and never work with mains. If you are ever in doubt you should check with someone suitably qualified."*

- *"Be extremely careful when working with circuits (especially 5V) that connect to the GPIO pins as they are not protected on the Raspberry Pi and the external power supply."*

Objective: Interrupt-driven Input/Output on raspberry Pi 3B+ with LEDs and pushbuttons.

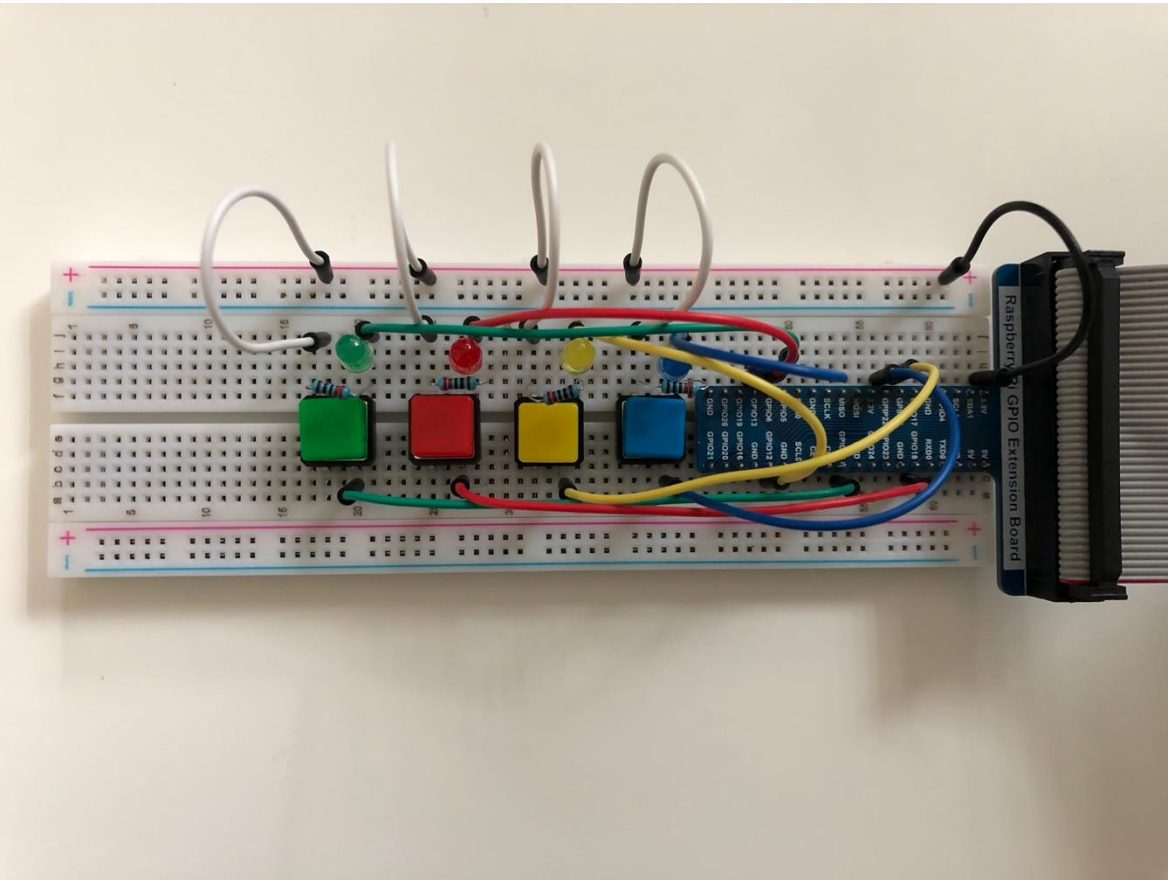
Main idea: In this assignment, you are going to wire up four inputs to pushbuttons, and four outputs to light-emitting diodes, and then writing a python script to light up some LEDs in blink mode and stop them from blinking when some events happen.

1. Please complete the Raspberry Pi setup by following set up guide document which is provided for you already.

2. Wire up four inputs to pushbuttons, and four outputs to light-emitting diodes (LEDs).
    a. All of the LEDs are suitable for use with the 220Ω resistors at 3.3V, at a reasonable brightness. Connect one side of the resistor to one side of the LED, and the other side of the resistor to pushbutton. Also make sure that power pin (3.3V) is also connected to the board.

3. To make the connections to the RPi3 GPIO, use a Pi Cobbler which is a ribbon cable, and a Raspberry Pi GPIO Extension Board that connects to a solderless breadboard, where you can add your own components.

4. Port selection is as followed (last page shows the GPIO pin layout for raspberry Pi 3B+):
    a. Pin #22 / GPIO 25: green pushbutton
    b. Pin #12 / GPIO 18: red pushbutton
    c. Pin #13 / GPIO 27: yellow pushbutton
    d. Pin #15 / GPIO 22: blue pushbutton
    e. Pin #29 / GPIO 5: green LED
    f. Pin #31 / GPIO 6: red LED
    g. Pin #32 / GPIO 12: yellow LED
    h. Pin #33 / GPIO 13: blue LED



5. After wiring up all the pushbuttons and LEDs and resistors to the GPIO pins, write a python script using ***RPi.GPIO*** module in python in which:

    a. Pressing Yellow button activates blink mode for both Red and Green LEDs in which when these two buttons (Red and green) are pressed simultaneously, both Red and Green LEDs start blinking.
        i. Red and Green LEDs stop blinking when Blue button is pressed, or when activated timer reaches a specific threshold.

b. Also, Pressing Blue button activates blink mode for both Red and Green LEDs in which when these two buttons (Red and green) are pressed simultaneously, both Red and Green LEDs start blinking.
    i. Blinking Red and Green buttons finishes when Yellow button is pressed, or when activated timer reaches a specific threshold.

6. Create a directory with the name of ***gpio*** in your home directory *(/home/pi/gpio*) and save your code with the name of ***assignment4.py*** in this directory.

7. Copy and paste the following script in a file and save it in the directory *(/home/pi/gpio/*) with the name of ***gpint*** (without any extension).

```
#!/bin/sh
### BEGIN INIT INFO
# Provides:          gpint
# Required-Start:    $remote_fs
# Required-Stop:     $remote_fs
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Interrupt-based GPIO LED/pushbutton daemon
# Description:       Listens for button events and lights up LEDs
### END INIT INFO

# Do NOT "set -e"

# PATH should only include /usr/*     if it runs after the mountnfs.sh script
PATH=/sbin:/usr/sbin:/bin:/usr/bin
DESC="gpint daemon"
NAME=gpint
#DAEMON=/usr/bin/$NAME
DAEMON=/home/pi/gpio/assignment4.py
ARGS=""
PIDFILE=/var/run/$NAME.pid
SCRIPTNAME=/etc/init.d/$NAME

export QUIET=1

# Exit if the package is not installed
[ -x "$DAEMON" ] || exit 0

# Read configuration variable file if it is present
[ -r /etc/default/$NAME ] && . /etc/default/$NAME

# work out daemon args

# Load the VERBOSE setting and other rcS variables
. /lib/init/vars.sh

# Define LSB log_* functions.
# Depend on lsb-base (>= 3.2-14) to ensure that this file is present
# and status_of_proc is working.
. /lib/lsb/init-functions
```

```
#
# Function that starts the daemon/service
#
do_start()
{
	# Return
	#   0 if daemon has been started
	#   1 if daemon was already running
	#   2 if daemon could not be started

	start-stop-daemon --start --quiet --pidfile $PIDFILE --user root --exec $DAEMON --test > /dev/null \
		|| return 1

	start-stop-daemon --start --quiet --pidfile $PIDFILE --user root --make-pidfile --background --no-close --exec
$DAEMON -- \
		$ARGS \
		|| return 2
      sleep 1
}
#
# Function that stops the daemon/service
#
do_stop()
{
	# Return
	#   0 if daemon has been stopped
	#   1 if daemon was already stopped
	#   2 if daemon could not be stopped
	#   other if a failure occurred
	start-stop-daemon --stop --retry=TERM/30/KILL/5 --pidfile $PIDFILE --user root #--exec $DAEMON # don't pass --exec
since is /usr/bin/python for scripts
	RETVAL="$?"
	[ "$RETVAL" = 2 ] && return 2
      sleep 1
	# Many daemons don't delete their pidfiles when they exit.
	rm -f $PIDFILE
	return "$RETVAL"
}

case "$1" in
 start)
	[ "$VERBOSE" != no ] && log_daemon_msg "Starting $DESC" "$NAME"
	do_start
	case "$?" in
		0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
		2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
	esac
	;;
 stop)
	[ "$VERBOSE" != no ] && log_daemon_msg "Stopping $DESC" "$NAME"
	do_stop
	case "$?" in
		0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
		2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
	esac
	;;
 status)
	status_of_proc "$DAEMON" "$NAME" && exit 0 || exit $?
```

```
        ;;
 restart|force-reload)
        log_daemon_msg "Restarting $DESC" "$NAME"
        do_stop
        case "$?" in
         0|1)
                do_start
                case "$?" in
                        0) log_end_msg 0 ;;
                        1) log_end_msg 1 ;; # Old process is still running
                        *) log_end_msg 1 ;; # Failed to start
                esac
                ;;
         *)
                # Failed to stop
                log_end_msg 1
                ;;
        esac
        ;;
 *)
        echo "Usage: $SCRIPTNAME {start|stop|status|restart|force-reload}" >&2
        exit 3
        ;;
esac

:
```

8. Execute the following commands in terminal:

    a. ***chmod a+x /home/pi/gpio/assignment4.py***

    b. ***sudo ln -s /home/pi/gpio/gpint /etc/init.d***

    c. ***sudo /etc/init.d/gpint start***
        i. **This command starts running your code in background.**
        ii. **For stopping your code from running in background you can execute the following command:**
            1. ***sudo /etc/init.d/gpint stop***

    d. ***sudo ln -s /etc/init.d/gpint /etc/rc5.d/S01gpint***

9. After following step 8, the bash script (***gpint*** file) runs your Python code in background continuously and listens for button events to light up the LEDs. You can test your code by pressing pushbuttons (which triggers an interrupt) to see if your script outputs the correct results.

10. Following is the structure you need to use for your code, so your script is able to work with the predefined bash scrip:

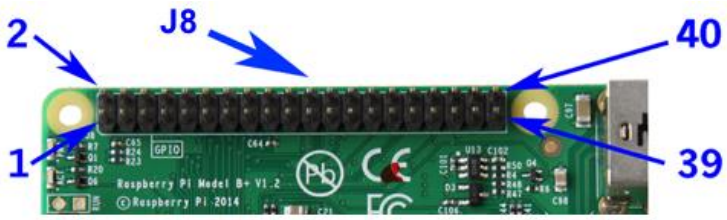
```
#!/usr/bin/python
# Assignment 4 #

###  import Python Modules  ###
import threading

### Set GPIO pins and all setups needed  ###

### Define a function based on assignement description ###

def handle(pin):
        # light corresponding LED when pushbutton of same color is pressed
        # btn2led is a dictionary which saves inforamtion regarding each pushbutton
        # and its corresponding LED which needs to be defined at start of your code
        GPIO.output(btn2led[pin], not GPIO.input(pin))

        t = None
        if pin == BTN_G or pin == BTN_R:
                # when green and red pressed simultaneously, enter blink mode
                if GPIO.input(BTN_G) and GPIO.input(BTN_R):
                        #print "starting thread"
                        t = threading.Thread(target=blink_thread)
                        t.daemon = True
                        t.start()

## Tell GPIO library to look out for an event on each pushbutton and pass handle ###
### function to be run for each pushbutton detection ###
```

Assignment submission:

- Submit your Python code in Canvas.
    - Your code needs to be commented extensively.
- Create and submit a 2-minute video of running and testing your code based on the assignment description.
    - Note that you need to state your name and student ID in the video. You also have the option of appearing in the video but are not required to do so.
    - You need to explain your code and your approach in detail.

GPIO (General Purpose I/O):

- The Raspberry Pi 3B+ board contains a single 40-pin expansion header labeled as ‘J8’ providing access to 26 GPIO pins (pins 1, 2, 39 & 40 are also labeled below):



J8 Pinout (40-pin Header)

- The following figure shows the GPIO pin layout for Raspberry Pi 3B+: