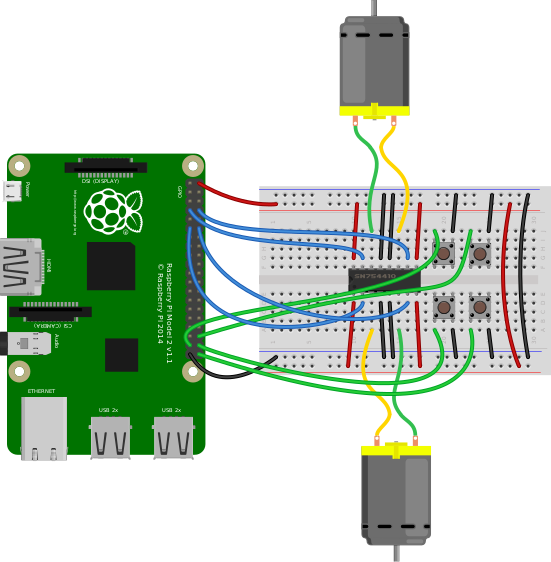
Milindi Shah J057

IOT Exp 10:

Programming with GPIOZero/any other library

**Q1. Button controlled robot :**



from gpiozero import Robot, Button

from time import sleep

from signal import pause

robot = Robot(left=(1,2), right=(3,4))

btn1 = Button(5)

btn2 = Button(6)

btn3 = Button(7)

btn4 = Button(8)

btn1.when\_pressed = robot.forward()

btn1.when\_released = robot.stop()

btn2.when\_pressed = robot.left()

btn2.when\_released = robot.stop()

btn3.when\_pressed = robot.right()

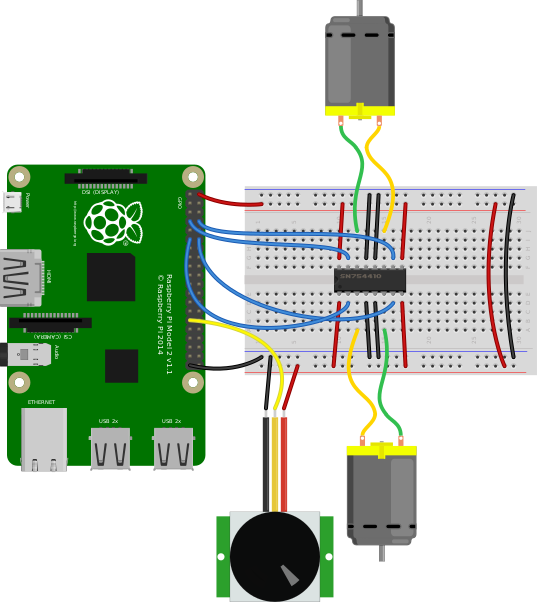
btn3.when\_released = robot.stop()

btn4.when\_pressed = robot.backward()

btn4.when\_released = robot.stop()

pause()

**Q2. Make a robot drive forward when it detects motion :**



from gpiozero import MotionSensor, Robot

from signal import pause

robot = Robot(left=(1,2), right=(3,4))

pir = MotionSensor(6)

pir.when\_motion = robot.forward()

pir.when\_no\_motion = robot.stop()

pause()

**Q3. Present the value of a potentiometer on an LED bar graph using PWM:**

from gpiozero import LEDBarGraph, MCP3008

from signal import pause

graph = LEDBarGraph(5, 6, 13, 19, 26, pwm=True)

pot = MCP3008(channel=0)

graph.source = pot

pause()

### Q4. Internet connection status indicator:

from gpiozero import LED, PingServer

from gpiozero.tools import negated

from signal import pause

green=LED(1)

red=LED(2)

internet = PingServer(‘google.com’)

green.source=internet

red.source=negated(green)

pause()

**Q5.You can read the Raspberry Pi’s own CPU temperature using the built-in** CPUTemperature class, and display this on a “bar graph” of LEDs:

from gpiozero import CPUTemperature, LED

cpu=CPUTemperature(min\_temp=50, max\_temp=90)

led=LED(1)

led.source=cpu

**Q6. Control LED using TimeofDay i.e. LED should be on between 7-8am**

from gpiozero import TimeOfDay, LED

from datetime import time

from signal import pause

lamp = LED(1)

morning = TimeOfDay(time(7), time(8))

lamp.source = morning

pause()

**Q7. DiskUsage  - Demonstrate use of these classes**

*DiskUsage(filesystem='/', \*, threshold=90.0, pin\_factory=None)*

Extends InternalDevice to provide a device which is active when the disk space used exceeds the threshold value.

The following example plots the disk usage on an LED bar graph:

*from gpiozero import LEDBarGraph, DiskUsage*

*from signal import pause*

*disk = DiskUsage()*

*print('Current disk usage: {}%'.format(disk.usage))*

*graph = LEDBarGraph(5, 6, 13, 19, 25, pwm=True)*

*graph.source = disk pause()*

Parameters:

*filesystem (str)* **–** A path within the filesystem for which the disk usage needs to be computed. This defaults to /, which is the root filesystem.

*threshold (float)* **–** The disk usage percentage above which the device will be considered “active” (see is\_active). This defaults to 90.0.

*pin\_factory* **-** Factory or None

*is\_active* **-** Returns True when the disk usage exceeds the threshold.

*usage* **-** Returns the current disk usage in percentage.

*value* **-** Returns the current disk usage as a value between 0.0 and 1.0 by dividing usage by 100.

**Q8. LoadAverage - Demonstrate use of these classes**

*LoadAverage(load\_average\_file='/proc/loadavg', \*, min\_load\_average=0.0, max\_load\_average=1.0, threshold=0.8, minutes=5, pin\_factory=None)*

Extends InternalDevice to provide a device which is active when the CPU load average exceeds the threshold value.

The following example plots the load average on an LED bar graph:

*from gpiozero import LEDBarGraph, LoadAverage*

*from signal import pause la = LoadAverage(min\_load\_average=0, max\_load\_average=2) graph = LEDBarGraph(5, 6, 13, 19, 25, pwm=True)*

*graph.source = la pause()*

Parameters:

*load\_average\_file (str)* **–** The file from which to read the load average. This defaults to the proc file /proc/loadavg. Whatever file is specified is expected to contain three space-separated load averages at the beginning of the file, representing 1 minute, 5 minute and 15 minute averages respectively.

*min\_load\_average (float)* **–** The load average at which value will read 0.0. This defaults to 0.0.

*max\_load\_average (float)* **–** The load average at which value will read 1.0. This defaults to 1.0.

*threshold (float)* **–** The load average above which the device will be considered “active”.This defaults to 0.8.

*minutes (int)* **–** The number of minutes over which to average the load. Must be 1, 5 or 15. This defaults to 5.

*pin\_factory* **-** Factory or None

*is\_active* **-** Returns True when the load\_average exceeds the threshold.

*load\_average* **-** Returns the current load average.

*value* **-** Returns the current load average as a value between 0.0 (representing the min\_load\_average value) and 1.0 (representing the max\_load\_average value). These default to 0.0 and 1.0 respectively.