| Regularity (2) | Content (4) | Viva-voce (2) | Timely Submission (2) | Total (10) | Dated Sign of Subject Teacher |
|---|---|---|---|---|---|
| | | | | | |

---

**Assignment No. 06** Date of Performance: ……

---

**Title:** Design a web interface to control connected LEDs remotely using Raspberry-Pi/Beagle board/Arduino.

---

**Objective:** To design the web interface for IoT.

---

**Theory:** One of the function of IOT is remote control of devices, being able to trigger action from a remote location. This is otherwise known as remote configuration. Whether it's a home, an office, or an industrial site, connected devices rely on some form of automation through sensors or machines and need a mechanism to control their operation remotely. Here we will build a small IOT simulation using RPi and showcase the most common operation switching on and off a remote device.

**Hardware Requirements:**

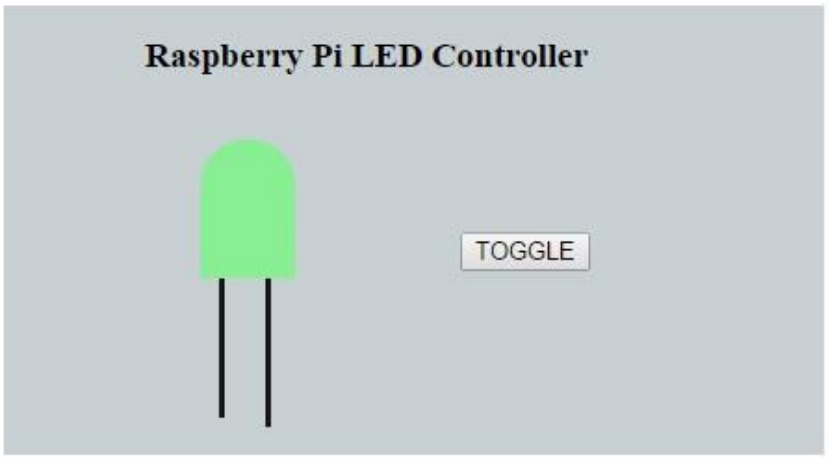
- RPi model with Raspbian OS
- 1 LED AND 1490OHM Resistor

**Software Requirements:**

- RPi GPIO, GPIO Python library for Raspbian OS.
- Pubnub Python SDK
- Pubnub JavaScript SDK

We will create 2 subsystems. One will be a controller, basic web application in the form of a webpage which can display the current states of device and send control messages to it and second one is the actual device simulated as on LED and controlled via Raspberry Pi.

**Web application:** The web interface looks like this:




This is a very simple webpage with a visual indicator for the device and a button to toggle ON/OFF state of LED. In the background, we have pubnub JavaScript API that performs 2 operations upon receiving certain events.

1. Sends a request to toggle the state of device.
2. Receives response with current state of the device.

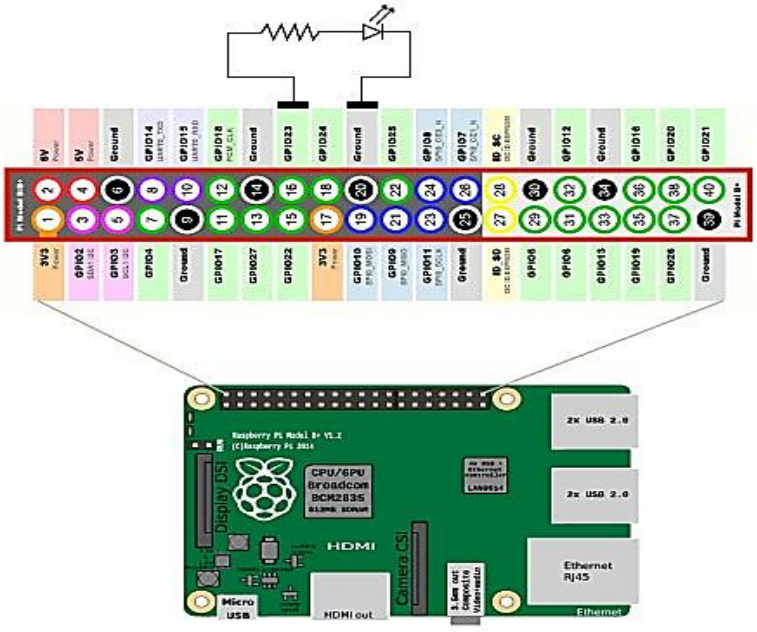
**Button click event:** When the toggle button is clicked, webpage sends a TOGGLE request message to the device via 'gpio-raspberry-control' channel.

```
$('#toggle').click(function(e){
        pubmsg = {"req" : "toggle"};
        pubnub.publish(
                {
                        channel: 'gpio-raspberry-control',
                        message:  pubmsg
                }
        );
});
```

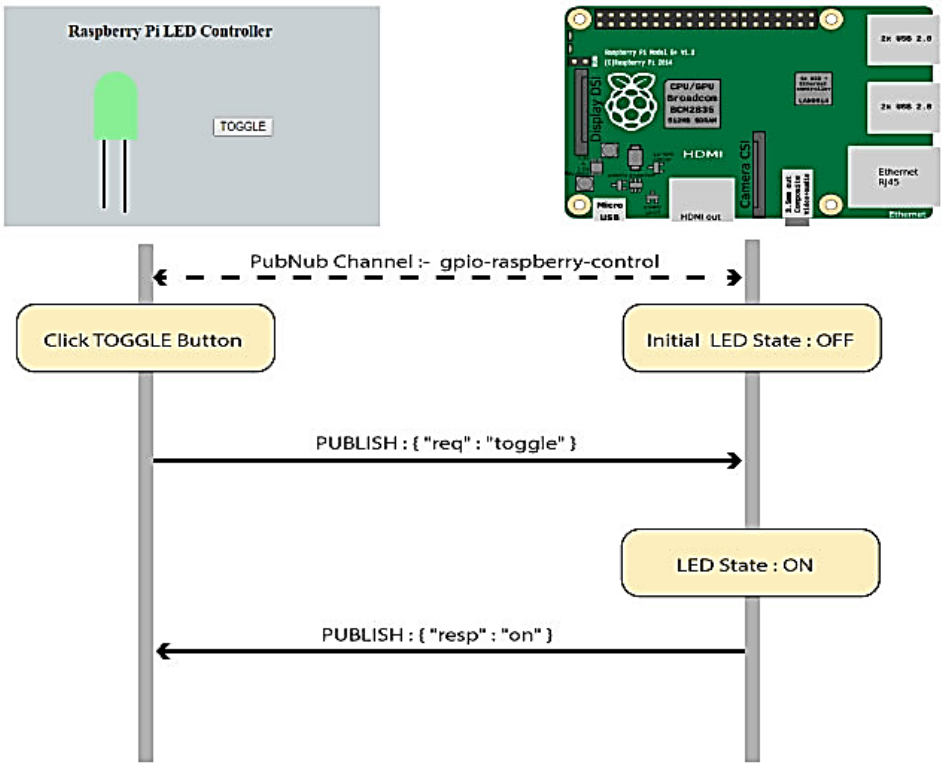
**PubNub Channel Subscribe Callback Event:** On receiving the toggle request, the Raspberry Pi toggles the state of the LED and sends a response back to the web page with the current state. Web page updates the visual indicator for the LED based on the received state information.

```
pubnub.subscribe({
        channel: 'gpio-raspberry-control',
        message: function(m){
                console.log(m)
                if('resp' in m) {
                        if('on' ==  m['resp']){
                                $('#led').removeClass('dim');
                                $('#led').addClass('glow');
                        } else {
                                $('#led').removeClass('glow');
                                $('#led').addClass('dim');
                        }
                }
        }
});
```

**RPi and LED:** Here is the schematic for the raspberry Pi connections to be used in this application.



We are going to use the Raspberry Pi GPIO Python library to send the control messages to Raspberry Pi GPIO ports. This library works well with the python environment available by default with Raspbian OS. When a toggle request is received, the application checks the current state of the GPIO driving pin of the LED, toggles its state and then sends the new state back to the web application as a response message. The exchange of messages between the web application and Raspberry Pi can be visualized as follows.




```
glow = False
#PubNub Channel Subscribe Callback
def gpioCallback(msg,channel):
        global glow
        respstring = ''
        command = msg
        print "Command is: " + str(command)
        if ('req' in command):
                if(command['req'] == 'toggle'):
                        if(glow):
                                glow = False;
                                respstring = 'off'
                        else:
                                glow = True
                                respstring = 'on'
                        GPIO.output(16, glow)
                        respmsg = {"resp": respstring}
                        pubnub.publish(pubnubChannelName, respmsg)
```

---

**Outcome:** Thus we have studied how to control various sensors and actuators with the help of web interface for IOT application.

---