Lab 5 Node-Red

**OBJECTIVE:**

Learn how to use Node-RED to communicate with the Raspberry Pi’s GPIO pins. You will create a Node-RED ‘flow’ to control LEDs, Log weather data to Evernote on button press, Send ultrasonic sensor data to mobile as a SMS.

## COMPONENTS:

1) RPi 3

2) Breadboard

3) 330 Ω Resistor x 1

4) Push Button x 1

5) LED x 1

6) HC-SR04 Ultrasonic Distance Sensor x 1

7) Connecting Wires

## Node-RED:

Node-RED is a flow-based programming tool for wiring together hardware devices, APIs and online services in new and interesting ways. Developed by IBM Emerging Technology and the open source community. It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click. It is built on Node.js. Node Red can be installed on Raspberry Pi and its server can be run on it.

For Raspbian pre-installed with Node-RED, you need to upgrade it. Use the following commands (without ‘$’):

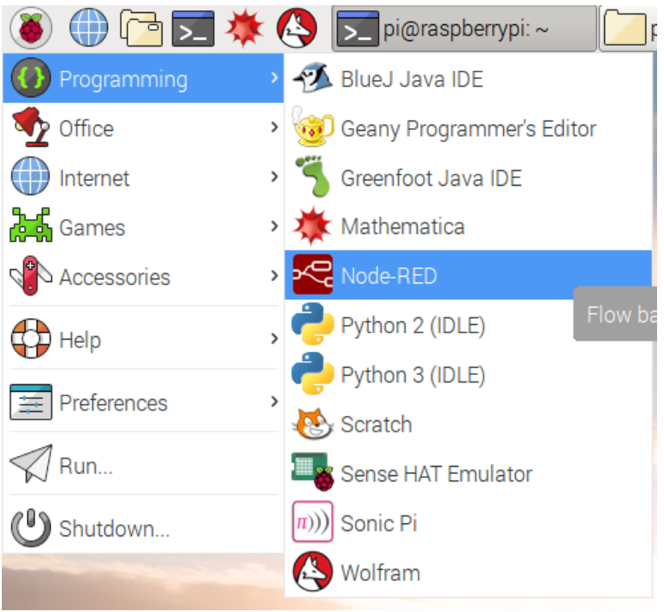
sudo apt-get update

update-nodejs-and-nodered

### **To freshly install Node-Red on your Raspberry Pi:2**[**¶**](http://localhost:8888/notebooks/Desktop/Important%20Documents%20/IoT%20final/Combined/Lab%205/Lab%205.ipynb#To-freshly-install-Node-Red-on-your-Raspberry-Pi:2)

Before installing node red, first install npm and node.js on your raspberry pi. Refer <http://www.instructables.com/id/Install-Nodejs-and-Npm-on-Raspberry-Pi/>For Node Red, refer <https://nodered.org/docs/hardware/raspberrypi>. Type the given commands in your terminal.

Open Node Red by typing node-red-start in your terminal, or by (Figure 23):



### Ultrasonic Sensor:

HC-SR04 is a commonly used module for non-contact distance measurement for distances from 2cm to 400cm. This sensor uses ultrasonic sound to measure distance just like bats and dolphins do. Ultrasonic sound has such a high pitch that humans cannot hear it. This particular sensor sends out an ultrasonic sound that has a frequency of about 40 kHz. The sensor has two main parts: a transducer that creates an ultrasonic sound and another that listens for its echo. The time difference between transmission and reception of ultrasonic signals is calculated. Using the speed of sound and ‘Speed = Distance/Time' equation, the distance between the source and target can be easily calculated.

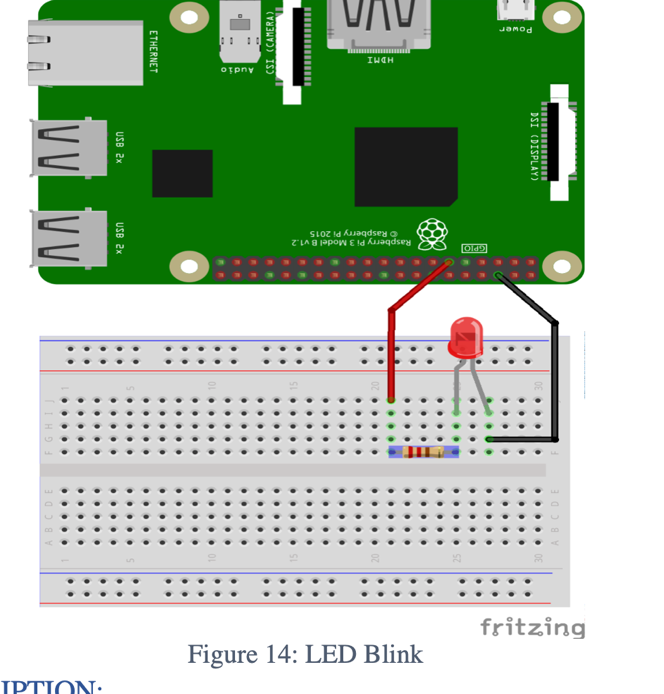
Distance Calculation: Distance = Speed \* Time/2

The "2" is in the formula because the sound has to travel back and forth. First the sound travels away from the sensor, and then it bounces off of a surface and returns back. we need only half of this. Therefore time is taken as time/2. Speed of sound at sea level = 340 m/s

### Thus, Distance = 17150 \* Time (unit cm)

### Module 1: Use Node-RED to control LED via the Raspberry Pi’s GPIO pins

## CIRCUIT DIAGRAM:

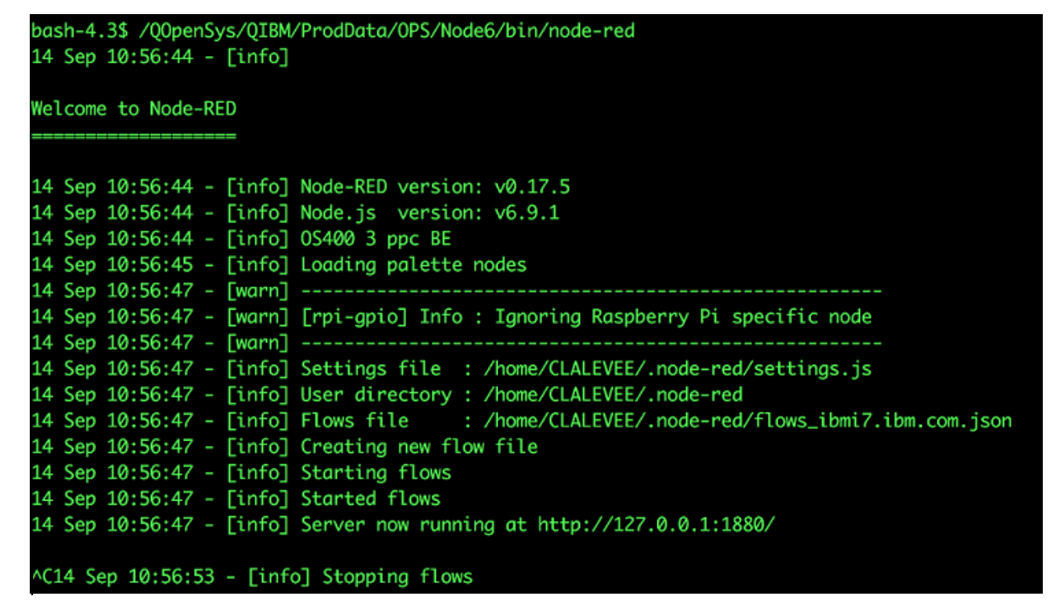


### CIRCUIT DESCRIPTION

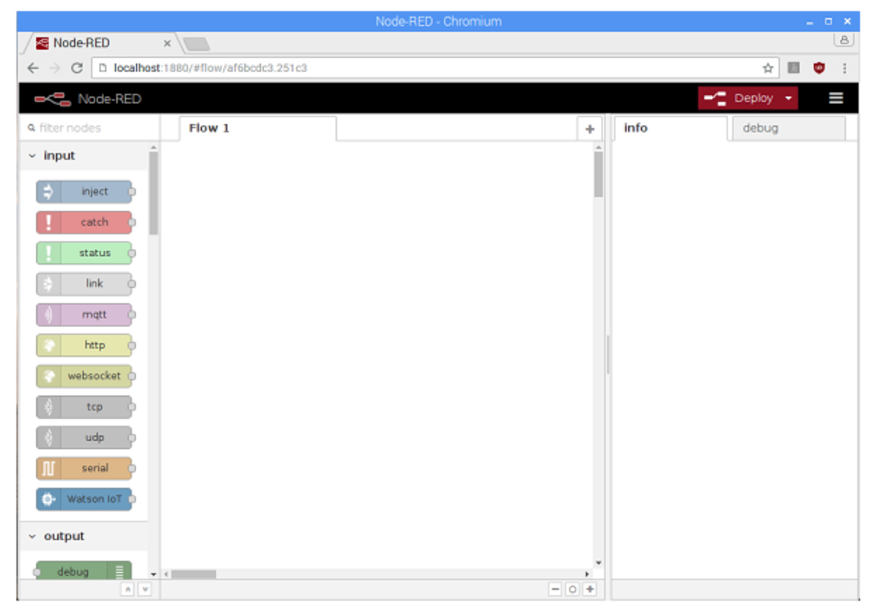
1. BCM PIN CONFIGURATION IS USED.
2. Positive leg of LED to one end of resistor.
3. Other leg of resistor to GPIO 17
4. Negative leg of LED to Ground pin

### CREATING THE PROGRAM:

1. Start up your Raspberry Pi. Click on the Raspberry icon, then the Programming menu to open Node-RED (Figure 23) You should see a window displaying information about Node-RED starting up

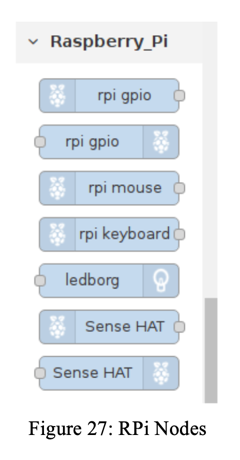


1. Now go to the Internet menu and open Chromium Web Browser. If chromium is not installed, type $ sudo apt-get install chromium-browser --yes in your terminal.
2. In Chromium, locate the address bar at the top and type in localhost:1880, then press Enter. This will display the Node-RED interface. (Your Raspberry Pi does not need to be connected to the internet to use Node-RED: localhost is the address the Raspberry Pi uses to refer to itself and :1880 means that it is looking at port 1880.)



1. Programs in Node-RED are called flows. You can see that your blank page is labelled as Flow 1 in the tab at the top. You can create as many flows as you want and they can all run at the same

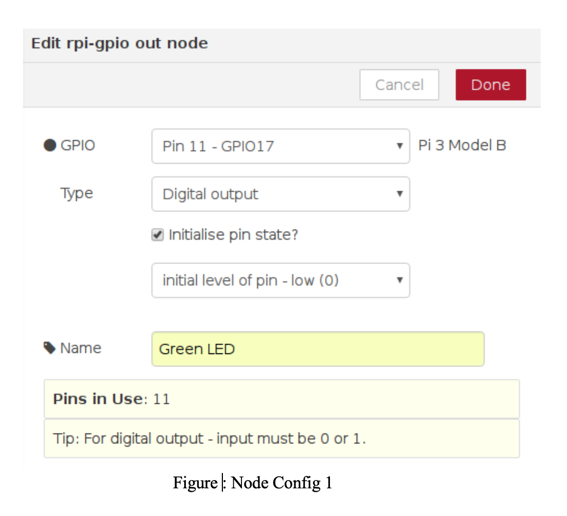
The coloured blocks on the left side of the interface are the nodes. Scroll right down to the bottom of the list and you will see some nodes labelled Raspberry Pi.



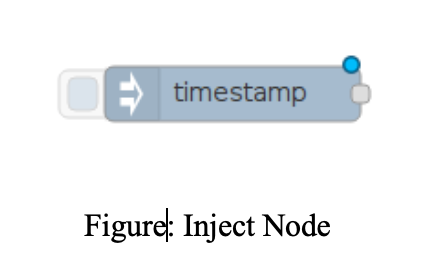
1. You will see two nodes with the label rpi gpio: these are the ones we will use to talk to the GPIO pins on the Raspberry Pi. The first one in the list, with the raspberry icon on the left, is for inputs. Using a button push to control something would be an example of an input. The second node, with the raspberry icon on the right, is for outputs. Switching on an LED would be an example of an output. Drag an output node onto the blank page in the middle.



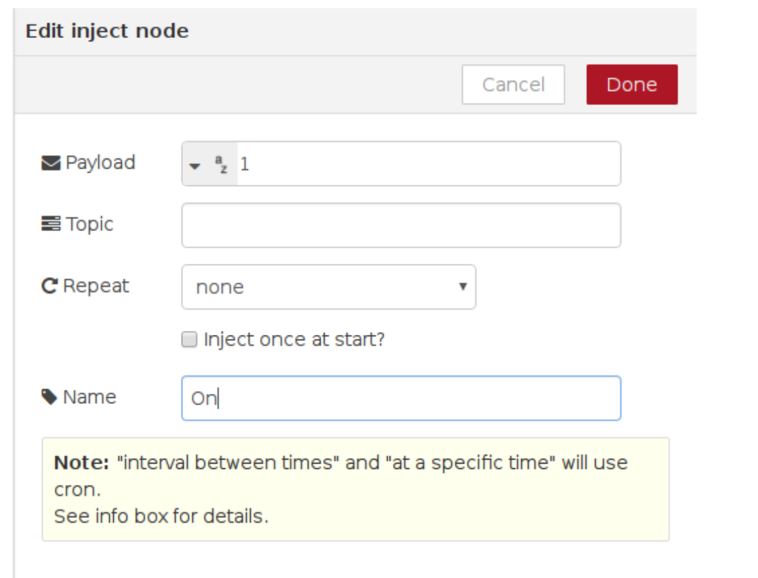
1. Double-click on the node and a box will appear to let you configure the node. Change the GPIO pin to be GPIO17 and tick Initialise pin state?. Leave the setting for Initial level of pin on low. Give the node a name. When you are finished, click Done



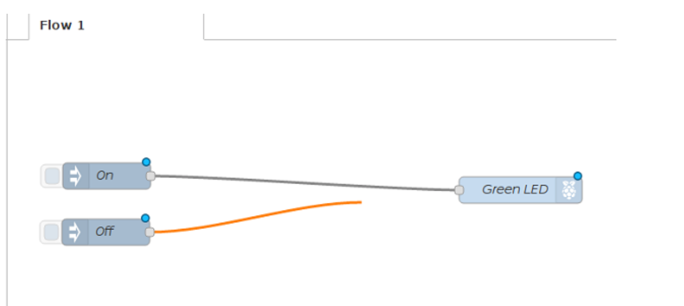
1. Now scroll back up to the list of nodes. To turn the LED on and off, we need an input. In Node-RED we can inject messages into the flow and cause things to happen as a result. Drag an inject node onto the flow



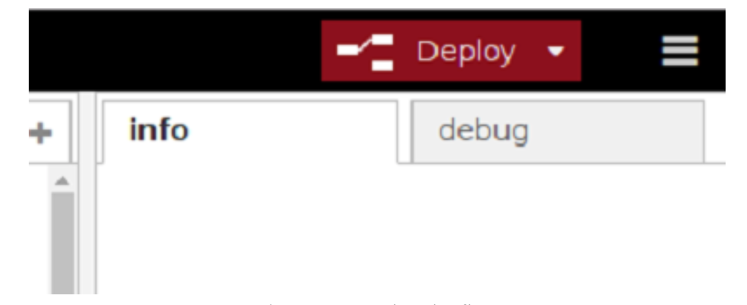
1. Double-click on the inject node. Use the drop down next to Payload to change the data type to string and type 1 in the Payload box - this will be the message. Type On in the Name box. Press Done



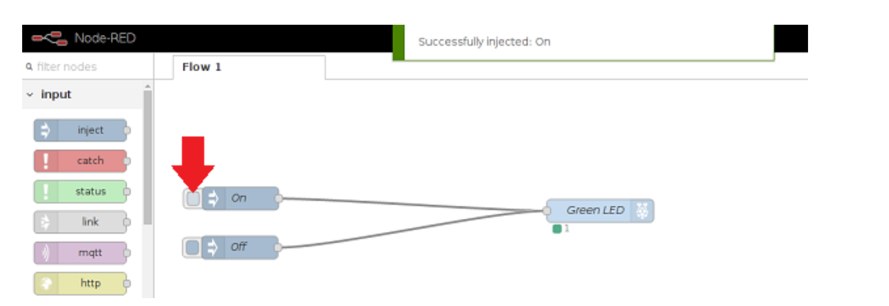
1. Repeat the previous steps to create another inject node, except this time add 0 as the payload message, and call this node Off.and call this node Off.
2. Now look for the grey dot on the right side of the inject nodes. Click and drag from the grey dot on the On node to the grey dot on your LED node to join them up. Repeat for the Off node, also joining it to the LED node.



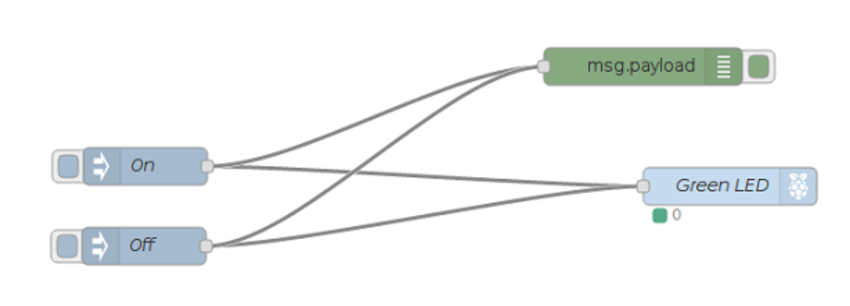
1. Our flow is finished, so we can deploy it. Click on the big red Deploy button on the top right of the screen. A message should pop up at the top saying “Successfully deployed”. This is similar to pressing the green flag in Scratch or F5 to run your code in Python.



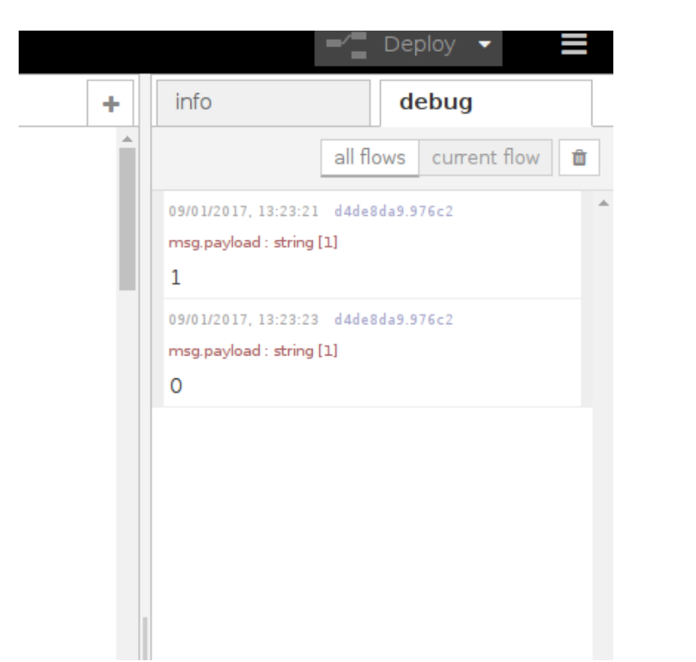
1. Now click on the blue square on the left of the On node to inject the message 1. The Green LED node receives the message and your LED should light up. You should be able to turn the LED off by clicking the blue square on the Off node, which injects the message 0.



1. You can also ask Node-RED to display debugging information by wiring up your nodes to a Debug node, which can be found under output. Drag in a debug node and wire your two inject nodes to it, then click Deploy. When you click the buttons to inject the message, Node-RED will show you what was injected in the Debug tab on the right side of the screen. Click the tab to display the messages



For example, this is the output if you click first on the On node and then on the Off node.

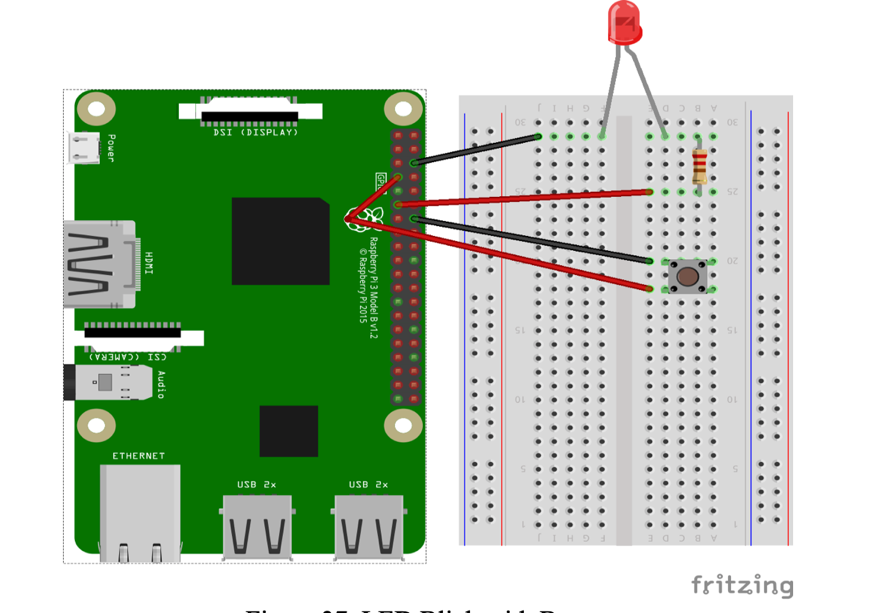


Lab 5A:

# **Objective:**[**¶**](http://localhost:8888/notebooks/Desktop/Important%20Documents%20/IoT%20final/Combined/Lab%205/Lab%205A.ipynb#Objective:)

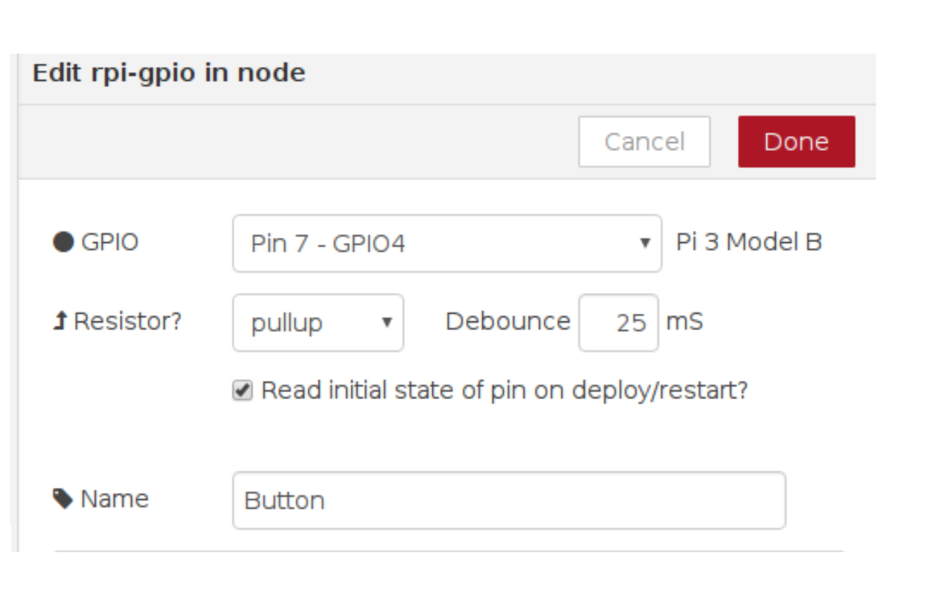
Adding a Button or input Device learn to interface using Node-red

### CIRCUIT DIAGRAM:



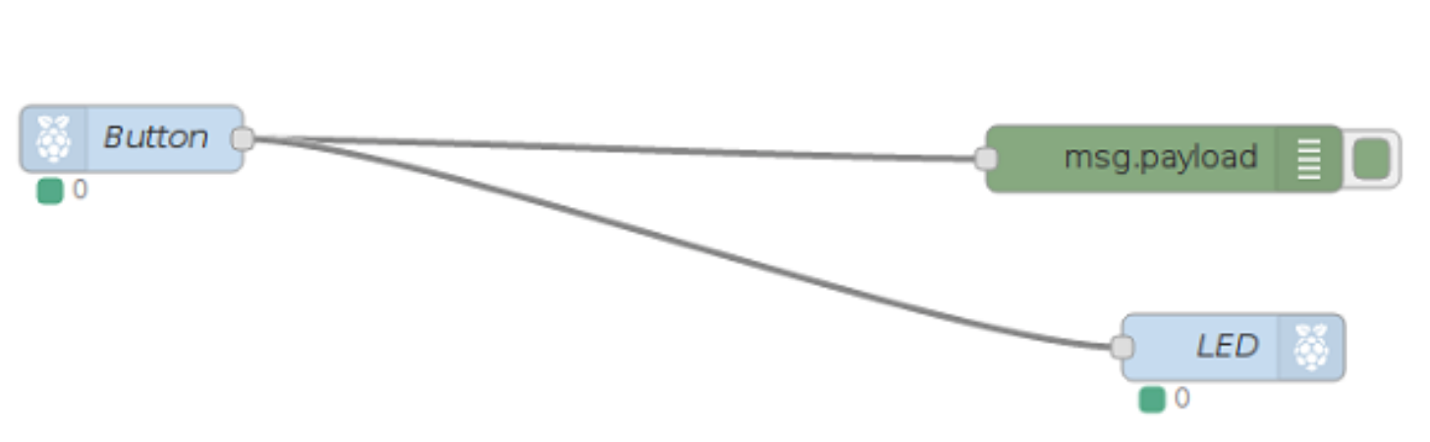
### CREATING THE PROGRAM:

1. Now let’s add a button to control the LED. Wire up your button to the Raspberry Pi as shown in Figure 37, so that your LED is still connected to GPIO pin 17, and your button is connected to GPIO pin 4
2. Remove your Turn on and Turn off inject nodes by clicking the node and pressing Delete on the keyboard. We no longer need these, as we will be controlling the LED using a physical button.
3. Now we need to add a Raspberry Pi GPIO input node. This is the node with the raspberry icon on the left. Set up this node to receive input from your physical button as follows:



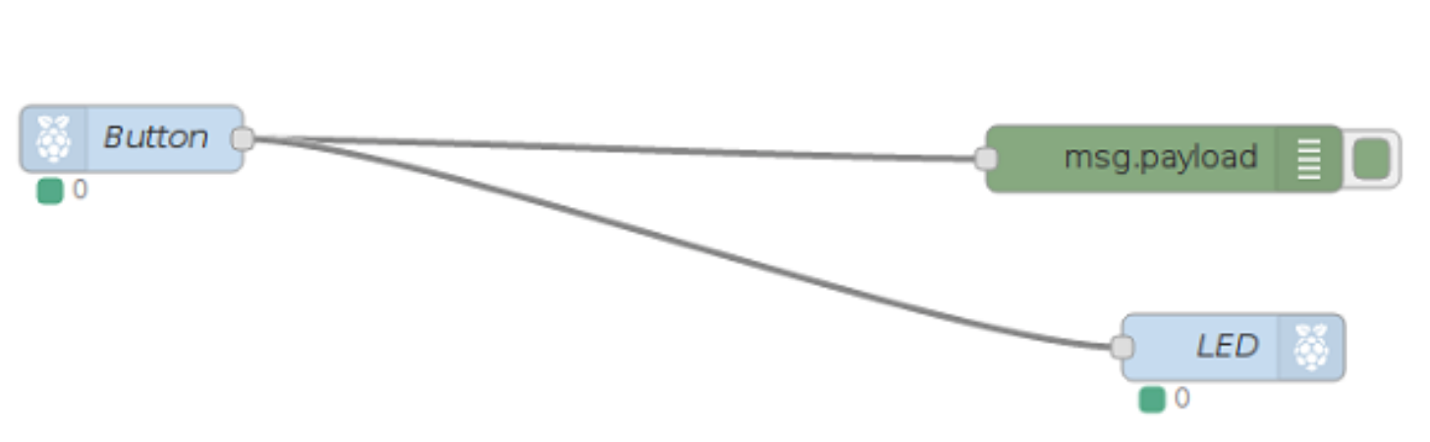
Specifying pullup means that GPIO pin 4 will be set to HIGH, and pressing the button will cause it to go LOW.

1. Now join up your button node output to the existing debug and LED nodes. Deploy the flow and test it by pressing the button.

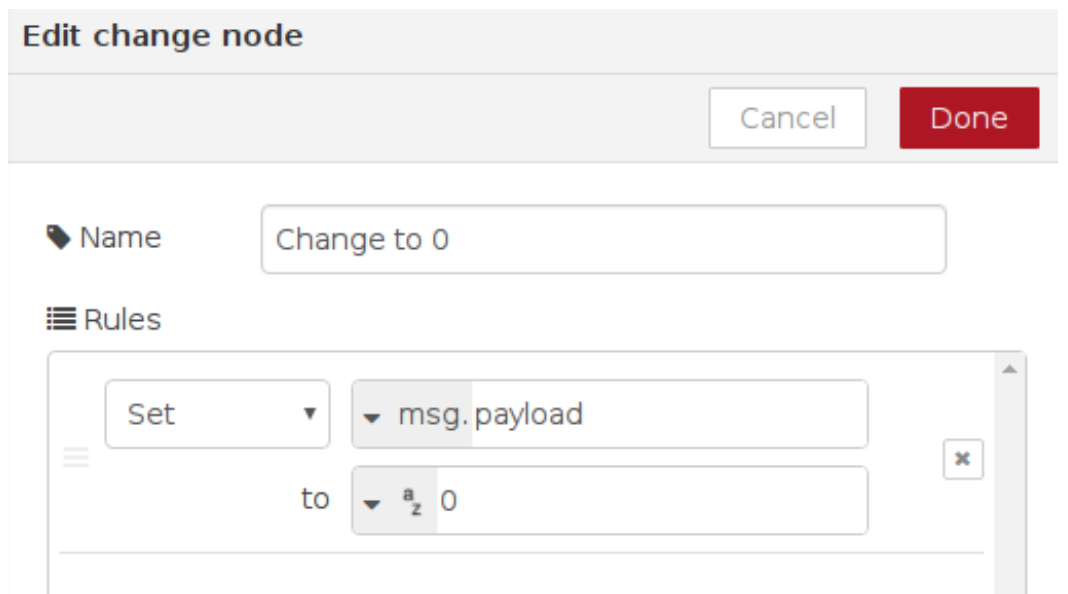


You will notice that the LED is lit to start with, and that pressing the button switches it off. That’s not quite right! This is because we are using pullup as explained in the previous step, so the button pin will be HIGH by default. HIGH generates the message 1, and this turns the LED on. When we press the button, we cause the pin to go LOW, generating a 0 message which turns off the LED. We need to reverse the values. We want the LED to receive the message 0 by default and the message 1 when the button is pressed.

1. Remove the connection you made between the button node and the LED node by clicking on the line and pressing Delete on the keyboard.
2. Add a switch node. This can be found in the function section. This node is similar to the if/elif/else type constructs you may have seen in Scratch or Python. You can configure it to have multiple output paths (outlined in red on the screenshot) depending on the value passed in. In this case we will set up the node so that if the property msg.payload is equal to 1, the first path will be followed. Click the small Add button at the bottom to add a second path, and for this path select otherwise in the drop down. This path will be followed if the input was anything other than 1. Click Done when you are finished.



1. You should see your newly created switch node, with two dots on the right side for outputs. Note that the title If input is 1 is simply a description of what the node does, and has no effect on its function.
2. Join up the GPIO input button node to the input (left side) of the switch node
3. Now drag in a yellow change node from the functions section and double-click on it to configure it. We will use this node to change the message being sent. Remember: when we created the switch node, the first output was set to be followed if the input message was 1. We will use the change node to change the message to 0.



1. Press Done, then draw a line from first output of the switch node to the change node. Then connect the output of the change node to the LED node:
2. Now add another change node to set the msg.payload to 1. Connect this node to output 2 of the switch node and then to the LED node. This tells the flow that when the otherwise branch is followed (i.e. the msg.payload is not already 1), we would like to change it to 1. When you are ready, deploy your flow, and then push the physical button to confirm that it works properly

### Final Diagram

