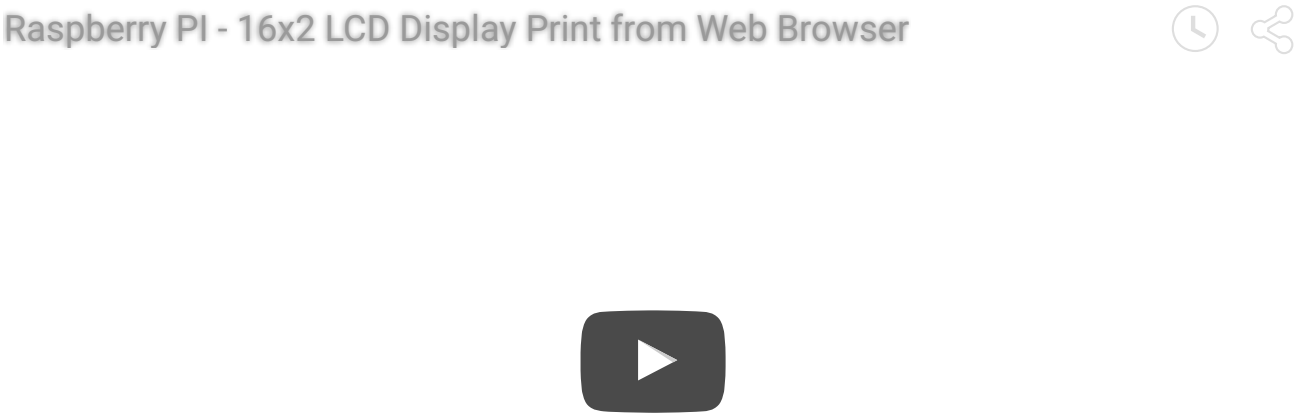
Raspberry Pi, 16×2 LCD and Node.js – Print stuff

In this post, we will interface Raspberry Pi and a 16 X 2 Liquid Crystal Display. And then we will write a Node.js program to print some information on the LCD using the lcd node module. We will see how to print a Digital clock, get the Pi's IP Address and print the same and how we can interact with the world wide web and display the same on the LCD.

Below is a quick demo of how you can print stuff using a Web Browser



You can find the completed code here.

So let us get started.

Prerequisites

If you are new to Raspberry pi and have not yet installed Node.js on it, I would recommend going through Getting Started with Raspberry pi and Node.js.

If you are new to electronics devices and circuits, I would recommend going through the video lectures from All About Circuits.

Components needed

1. 1 – Raspberry pi B+
2. 1 – Standard 16×2 LCD
3. 1 – Breadboard
4. 1 – 1K Ohm potentiometer

If you are new to Raspberry Pi GPIO, please refer this.

Understanding a 16×2 LCD

A typical 16×2 LCD consists of 16 connectors. There are 5 sets of operational pins.

1. The Data pins
2. The Register select pin
3. The Read/Write pin
4. The Enable pin
5. The Contrast pin

The **data pins** are straight forward. They are sending data to the display (toggled high/low). We will only be using write mode and not reading any data.

The **register select** pin has two uses. When pulled low it can send commands to the LCD (like position to move to, or clear the screen). This is referred to as writing to the instruction or command register. When toggled the other way (1) the register select pin goes into a data mode and will be used to send data to the screen.

The **read/write** pin will be pulled low (write only) as we only want to write to the LCD based on this setup.

The **enable** pin will be toggled to write data to the registers.

The **contrast** pin is used to control the brightness of the LCD.

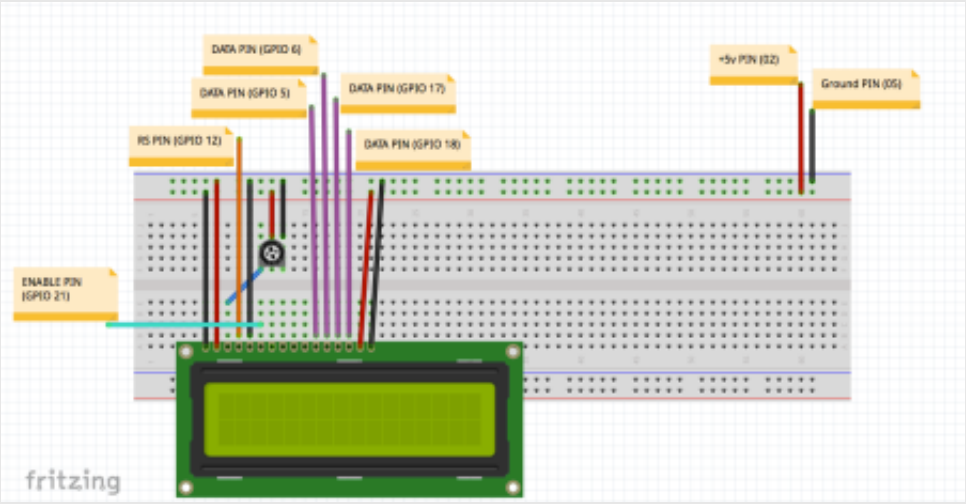
Apart from the above, there are 2 pairs of 5v and ground pins. One set is used to power the LCD and the second set is to control the back light.

LCD Pinout

The above information is taken from Wiring the Cobbler to the LCD.

Setup LCD

Now that we have a basic understanding of the LCD, we will set it up with our Raspberry Pi B+.



As you can see from the above diagram, We will connect

- Pin 5 from the PI – GND to the first row of the breadboard
- Pin 2 from the PI – 5V to the second row of the breadboard

From now on, we will use these 2 rows for power and ground.

Next, we will connect

- Pin 1 and Pin 16 of the LCD to Row 1 – GND
- Pin 2 and Pin 15 of the LCD to Row 2 – +5V

Next, we will connect the variable potentiometer

- The negative end of the variable potentiometer to Row 1 of the breadboard – GND
- The positive end of the variable potentiometer to Row 2 of the breadboard – +5V
- The third pin on the variable potentiometer to Pin 3 of the LCD

The above setup will help us configure the contrast of the Backlight.

Now, power up the PI and you should see the LCD powered up. Before we proceed further, we need to make sure that contrast settings are correct. Else you will not see any display when you start printing text.

Adjust the variable potentiometer as shown below till you see the black boxes.

If you do not see the black boxes, recheck your connection. There is no point going further if this step fails.

Now that the LCD is setup, we will complete the remaining connections. Connect

- RS Pin – Pin 4 of LCD to GPIO Pin 12 of the Pi
- R/W Pin – Pin 5 of LCD to GND Pin – Row 1 of breadboard
- E Pin – Pin 6 of LCD to GPIO Pin 21 of the Pi
- Data Pin D4 – Pin 11 of LCD to GPIO Pin 5 of the Pi
- Data Pin D5 – Pin 12 of LCD to GPIO Pin 6 of the Pi
- Data Pin D6 – Pin 13 of LCD to GPIO Pin 17 of the Pi
- Data Pin D7 – Pin 14 of LCD to GPIO Pin 18 of the Pi

Make sure that all connections are proper before proceeding.

Examples

Now that we have completed our hardware setup, we will print stuff on the LCD. Yay!!

Login to your pi via ssh – terminal/putty. As soon as you ssh into pi, you will be landing inside the */home/pi* folder. We will create a new folder here

named *node_programs*. And inside this folder, we will be maintaining all our programs. Run

mkdir node_programs

To step inside that folder, run

cd node_programs

For this post, we will create a new folder named printToLCD and will step inside this folder. Run

mkdir printToLCD && cd printToLCD

Note : You can run multiple commands separated by a &&.

First we will initialize a new node project here. Run

npm init

Fill it up as applicable.

Now, we will use a node module named *lcd* to interact with the LCD from inside our Node.js code. This is the simplest way to interact with the LCD from our Pi. Run

npm install lcd --save

Next, we will create a new file named *index.js*. And we will open the same in the nano editor. Run

nano index.js

The first example we are going to do is to show a Digital Clock.

Digital Clock

Paste the below code into the nano editor

~/node_programs/printToLCD/index.js

```
var Lcd = require('lcd'),
lcd = new Lcd({
```

```
1
2  var Lcd = require('lcd'),
3    lcd = new Lcd({
4      rs: 12,
5      e: 21,
6      data: [5, 6, 17, 18],
7      cols: 8,
8      rows: 2
9    });
10 lcd.on('ready', function() {
11   setInterval(function() {
12     lcd.setCursor(0, 0);
13     lcd.print(new Date().toString().substring(16, 24));
14   }, 1000);
15 });
16 // If ctrl+c is hit, free resources and exit.
17 process.on('SIGINT', function() {
18   lcd.clear();
19   lcd.close();
20   process.exit();
21 });
22
```

Things to notice

Line 1 : We require the LCD module

Line 2 : We init the module with our LCD – PI pin config. As you can read, rs stands for Register select, e stands for enable, data stands for the 4 data pins and finally the total rows and cols. Even though we have a 16 x 2 LCD, we will be using only the 8 x 1

part to print.

Line 10 : LCD takes a few minutes to startup, so we have this handy ready method, inside which we write our printing logic.

Line 11 : Every 1000 milliseconds, we execute the logic on line 12 and 13.

Line 12 : Position the cursor at 0th row and 0th col of the LCD

Line 13 : Prints the time part of the Date object

Line 18 : Cleans resources on exit.

Let us save the file now. To save the program, press (ctrl+x). This will ask you to save the file. Press Y and press enter key to complete the operation.

Execute the program by running

sudo node index.js

And you should see

Simple and easy right!

Scrolling Text

Next, we will see how to scroll a text. First, create a new file

```
nano scrollingText.js
```

And paste the below code into the nano editor

~/node_programs/printToLCD/scrollingText.js

```
var Lcd = require('lcd'),
lcd = new Lcd({
```

```
1
2
3
4  var Lcd = require('lcd'),
5    lcd = new Lcd({
6      rs: 12,
7      e: 21,
8      data: [5, 6, 17, 18],
9      cols: 16,
10     rows: 1
11   });
12 lcd.on('ready', function() {
13   lcd.setCursor(16, 0);
14   lcd.autoscroll();
15   print('Hello, World! ** ');
16 });
17 function print(str, pos) {
18   pos = pos || 0;
19   if (pos === str.length) {
20     pos = 0;
21   }
```

```
22    lcd.print(str[pos]);
23    setTimeout(function() {
24      print(str, pos + 1);
25    }, 300);
26  }
27  // If ctrl+c is hit, free resources and exit.
28  process.on('SIGINT', function() {
29    lcd.clear();
30    lcd.close();
31    process.exit();
32  });
33
34
35
```

We init the LCD as we did in the above program and instead of using the lcd.print() we have written our own method here that will print one character at a time (line 23) and when we reach the end, we set the position back to the first character (line 19). And we repeat the process of printing a character every 300 milliseconds (line 25) emulating a marquee.

Save the program and run

sudo node scrollingText.js

And you should see

Print Two Rows

Next, we will print text in 2 rows, one after the other. Create a new file

nano printTwoRows.js

And paste the below code inside the nano editor

~/node_programs/printToLCD/printTwoRows.js

```
var Lcd = require('lcd'),
lcd = new Lcd({
```

```
1
2  var Lcd = require('lcd'),
3    lcd = new Lcd({
4      rs: 12,
5      e: 21,
6      data: [5, 6, 17, 18],
7      cols: 16,
8      rows: 2
9    }); // Pi
```

```
10 lcd.on('ready', function() {
11   lcd.setCursor(0, 0); // col 0, row 0
12   lcd.print(new Date().toISOString().substring(11, 19)); // print time
13   lcd.once('printed', function() {
14     lcd.setCursor(0, 1); // col 0, row 1
15     lcd.print(new Date().toISOString().substring(0, 10)); // print date
16     lcd.once('printed', function() {
17       lcd.clear();
18       lcd.close();
19     });
20   });
21 });
22
```

Once we have init the LCD, we will wait for the ready event. Next, we set the cursor to first row and first col and print the time. We wait on the printed event and then we set the cursor to second row first col and print the date.

Save the file and run

sudo node printTwoRows.js

And the output would look like

Raspberry PI - 16x2 LCD Display Print Two Rows

Print IP Address

Till now we have been printing static content. Now, we will query the OS and get the IP address of PI. And we will print the IP Address on to the LCD.

First create a new file

nano ipAddress.js

And paste the below code in the nano editor

~/node_programs/printToLCD/ipAddress.js

```
var Lcd = require('lcd'),
lcd = new Lcd({
```

```
1
2
3
4  var Lcd = require('lcd'),
5    lcd = new Lcd({
6      rs: 12,
7
```

```
    e: 21,
    data: [5, 6, 17, 18],
    cols: 16,
    rows: 1
  });
lcd.on('ready', function() {
  lcd.setCursor(16, 0);
  lcd.autoscroll();
  require('dns').lookup(require('os').hostname(), function(err, add, fam) {
    var text = "My IP Address is : " + add + " ** ";
    print(text);
  });
});
function print(str, pos) {
  pos = pos || 0;
  if (pos === str.length) {
    pos = 0;
  }
  lcd.print(str[pos]);
  setTimeout(function() {
    print(str, pos + 1);
  }, 300);
}
// If ctrl+c is hit, free resources and exit.
process.on('SIGINT', function() {
  lcd.clear();
  lcd.close();
  process.exit();
```

```
36 });
37
38
39
```

The above code is almost same as the scrolling text one, except for line 13. Here we query the OS and get the IP address. This IP address is then used for printing.

Save the above file and run

sudo node ipAddress.js

Sweet right!

Stock Ticker

Now, we will Fire a request to the Google Finance API and then print the same on the LCD. Create a new file

nano printStockData.js

Paste the below code into the nano editor

~/node_programs/printToLCD/printStockData.js

```
var http = require('http');
var Lcd = require('lcd');
```

```
1
2
3
4
5
6
7
8  var http = require('http');
9  var Lcd = require('lcd');
10 var lcd = new Lcd({
11   rs: 12,
12   e: 21,
13   data: [5, 6, 17, 18],
14   cols: 16,
15   rows: 1
16 });
17 function getQuote(stock) {
18   http.get({
19     host: 'www.google.com',
20     port: 80,
21     path: '/finance/info?client=ig&q=' + stock
22   }, function(response) {
23     response.setEncoding('utf8');
24     var data = "";
25     response.on('data', function(chunk) {
```

```
26      data += chunk;
27    });
28    response.on('end', function() {
29      if (data.length > 0) {
30        try {
31          var data_object = JSON.parse(data.substring(3));
32        } catch (e) {
33          return;
34        }
35        var quote = {};
36        quote.ticker = data_object[0].t;
37        quote.exchange = data_object[0].e;
38        quote.price = data_object[0].l_cur;
39        quote.change = data_object[0].c;
40        quote.change_percent = data_object[0].cp;
41        quote.last_trade_time = data_object[0].lt;
42        quote.dividend = data_object[0].div;
43        quote.yield = data_object[0].yld;
44        console.log(JSON.stringify(quote, null, 4));
45        lcd.setCursor(16, 0);
46        lcd.autoscroll();
47        var text = "Ticker : " + quote.ticker + " , Price : " + quote.price + " , Change(%) : " +
quote.change_percent + " ** ";
48
          print(text);
49
      }
50
    });
51
  });
52
}
53
```

```
54 function print(str, pos) {
55   pos = pos || 0;
56   if (pos === str.length) {
57     pos = 0;
58   }
59   lcd.print(str[pos]);
60   setTimeout(function() {
61     print(str, pos + 1);
62   }, 300);
63 }
64 // If ctrl+c is hit, clear, free resources and exit.
65 process.on('SIGINT', function() {
66   lcd.clear();
67   lcd.close();
68   process.exit();
69 });
70 getQuote('goog');
71
72
73
74
75
76
77
```

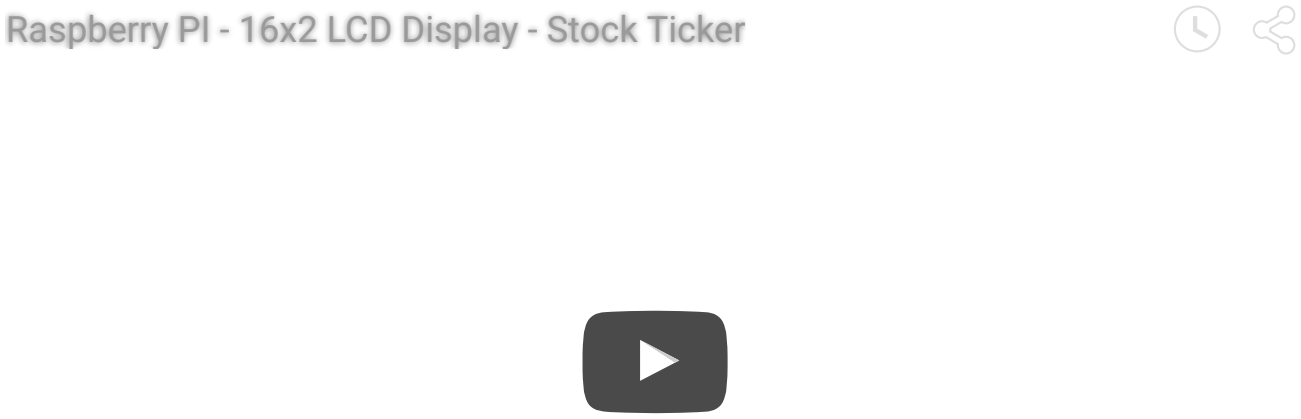
The above program is again derived from the scrolling text program. Except for the data that will be printed. In the above program, we init the LCD and at the very end (line 77), we trigger the getQuote(). This function will contact the Google Finance API and get the ticker for the provided stock. We then parse this response and print the

same to the console (line 43) as well as the LCD (line 48).

Save the file and run

sudo node printStockData.js

And you should see



Sweet Right!!

LCD and IOT

For the final example, we will host a web application. This will contain a text box, where a user can enter text. Upon submission of this text, we will display it on the LCD.

For this program, we will use another node module named tiny-router. This is a simple lightweight alternative of Express.js targeted at devices like PI. Let us first install it locally. Run

npm install tiny-router --save

And now, create a new file

nano server.js

And paste the below code into the nano editor

~/node_programs/printToLCD/server.js

```
var router = require('tiny-
router');
```

```
1
2
3
4
5
6
7
8
9  var router = require('tiny-router');
10 var fs = require('fs');
11 var Lcd = require('lcd');
12 var lcd = new Lcd({
13   rs: 12,
14   e: 21,
15   data: [5, 6, 17, 18],
16   cols: 16,
17   rows: 1
18 });
19 lcd.blink();
20 router.get('/', function(req, res) {
21   fs.readFile('./index.html', 'utf8', function(err, contents) {
22     if (err) {
```

```
23      return console.log(err);
24    }
25    res.send(contents);
26   });
27  });
28  router.get('/print/{text}', function(req, res) {
29   var text = req.body.text;
30   lcd.clear();
31   lcd.setCursor(0, 0);
32   print(decodeURI(text));
33   res.send({
34    status: true,
35    text: text
36   });
37  });
38  router.listen(2000);
39  console.log('Server listening on port 2000');
40  var timer;
41  function print(str, pos) {
42   pos = pos || 0;
43   timer = setTimeout(function() {
44    if (pos === str.length - 1) {
45     clearTimeout(timer);
46     str = '';
47    } else {
48     print(str, pos + 1);
49    }
50   }, 300);
```

```
51    lcd.print(str[pos]);
52  }
53  // If ctrl+c is hit, clear, free resources and exit.
54  process.on('SIGINT', function() {
55    lcd.clear();
56    lcd.close();
57    process.exit();
58  });
59
60
61
62
63
64
65
66
```

Things to notice

Line 1,2,4 : We require the dependent modules

Line 5 : We init the code

Line 13 : We show a blinking cursor (*for effect*)

Line 15 : We listen to the / route and dispatch the home page (*which we will create in a moment*).

Line 24 : This is the method that gets invoked when a user submits a text. Here we capture the text and print it on line 30. The logic is similar to the scrolling text program.

Save and close the *server.js*.

Now, we will create the home page. Create a new file

nano index.html

And paste the below code in the nano editor

~/node_programs/printToLCD/index.html

```
<!DOCTYPE html>
<html lang="en">
```

```
1
2
3
4  <!DOCTYPE html>
5  <html lang="en">
6  <head>
7    <meta charset="utf-8">
8    <meta http-equiv="X-UA-Compatible" content="IE=edge">
9    <meta name="viewport" content="width=device-width, initial-scale=1">
10   <title>Title Page</title>
11   <!-- Bootstrap CSS -->
12   <link href="//netdna.bootstrapcdn.com/bootstrap/3.1.1/css/bootstrap.min.css"
13 rel="stylesheet">
14   <!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and media queries -
   ->
15   <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
16   <!--[if lt IE 9]>
17 <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
18 <script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>
19 <![endif]-->
20   <style type="text/css">
21
```

```
  .disp-none {display: none;}
  </style>
</head>
<body class="container">
  <h1 class="text-center">LCD Remote Controller</h1>
  <hr/>
  <form role="form">
    <div class="form-group">
      <label for="printText">Print Text</label>
      <input type="email" class="form-control" id="printText" placeholder="Enter Text">
    </div>
    <div class="alert alert-warning disp-none" id="required">Print Text cannot be empty</div>
    <div class="alert alert-danger disp-none" id="error">Oops something went wrong! </div>
    <div class="alert alert-success disp-none" id="success">Success!! Check your LCD display</div>
    <button type="button" id="submit" class="btn btn-success">Submit</button>
  </form>
  <!-- jQuery -->
  <script src="//code.jquery.com/jquery.js"></script>
  <script type="text/javascript">
  $('#submit').click(function() {
    $('.alert').hide();
    var val = $('#printText').val().trim();
    if (val.length > 0) {
      $.get('/print/' + val, function(resp) {
        if (resp.status) {
          $('#success').show();
```

```
50          } else {
51            $('#errror').show();
52          }
53        });
54      } else {
55        $('#required').show();
56      }
57    });
58    </script>
59  </body>
60  </html>
61
62
63
```

A simple Bootstrap, jQuery page that will read a text from the textbox and post it to the server.

Save the file and run

sudo node server.js

And you should see

Simple and easy!

Hope this post gave you an idea how to work with a 16 x 2 LCD and Raspberry PI B+

Thanks for reading! Do comment. @arvindr21

Series Navigation

<< Raspberry Pi, Camera and Node.js – Live Streaming with Websockets

#IoTRaspberry pi, an Ultrasonic Sensor and Node.js – Measure Distances >>

November 17, 2014Arvind Ravulavaru