

SFT server

The server for *SMART FAIRY TALE* is based on *Node.js* running on a raspberry pi. This document briefly explains how to set up the pi properly. *Node.js* is a JavaScript runtime environment which makes it possible to run JavaScript code outside a web browser. Additionally we also used four packages for *Node.js*:

- *express* for serving the website
- *onoff* for controlling the GPIO's of the pi
- *socket.io* to establish websocket connections to the clients
- *forever* restarts the server automatically if it crashes

installation

change directory to root:

```
cd~
```

download node.js:

```
wget https://nodejs.org/dist/latest-v11.x/node-v11.15.0-linux-armv6l.tar.gz
```

unpack node.js:

```
tar -xzf node-v11.15.0-linux-armv6l.tar.gz
```

check version:

```
node-v11.15.0-linux-armv6l/bin/node -v
```

change directory to node:

```
cd node-v11.15.0-linux-armv6l/
```

copy all to */usr/local*:

```
sudo cp -R * /usr/local/
```

for testing:

```
export PATH=$PATH:/usr/local/bin  
node -v  
npm -v
```

download and install *express*:

```
npm install express --save
```

download and install *onoff*:

```
npm install onoff --save
```

download and install *forever*:

```
npm install forever --save
```

download and install *socket.io*:

```
npm install socket.io --save
```

setup server

create necessary directories:

```
cd ~  
mkdir nodeserver  
cd nodeserver  
mkdir public
```

create *package.json*:

```
npm init
```

the *package.json* file holds all dependencies

add `server.js` as entry point:

```
sudo nano package.json
```

create an empty `server.js` file:

```
nano server.js
```

copy/paste to `server.js` file:

```
console.log("hello server!");  
var express = require('express');  
var server = app.listen(3000); // port  
  
app.use(express.static('public')); // the directory that holds the 'index.html'  
  
var socket = require('socket.io');  
var io = socket(server);
```

start server for testing:

```
cd nodeserver  
forever server.js
```

use `CTRL + c` for stopping the server

websocket snippets

receiving and sending a message (server):

```
io.sockets.on('connection',newConnection); // set handler for new connection

// new connection callback
function newConnection(socket) {
    // print ip adress
    console.log(socket.handshake.address);
    // sending data e.g. for initializiation
    var sendData = {
        sendData1: someValue,
        sendData2: anotherValue
    }

    socket.emit('messageName',sendData);

    // set handler for receiving new message with the name 'messageName'
    socket.on('anotherMessageName', handler);

    function handler(data){
        var receivedData1 = data.val1;
        var receivedData2 = data.val2;

        // send message to all sockets
        var dataForAll = {
            dataForAll1: someValue,
            dataForAll2: anotherValue
        }
        io.sockets.emit('yetAnotherMessageName',data);
    }
}
```

add socket.io module inside `index.html`:

```
<script src="/socket.io/socket.io.js"></script>
```

add to client javascript, setup is called on load:

```
var socket;

function setup(){
  socket = io.connect('http://hostname:port');
  // set handler for message receive
  socket.on('messageName', handler);
}
```

receiving data (client):

```
function handler(data){
  var someLocalValue = data.someValue;
  var anotherLocalValue = data.anotherValue;
}
```

sending data (client):

```
function someEvent(somevalue, anothervalue){
  var data = {
    val1: someValue,
    val2: anotherValue
  }

  socket.emit('messageName',data);
}
```

onoff snippets

```
//include onoff object for gpios
var Gpio = require('onoff').Gpio;

//initialie pins
var myOutputPin = new Gpio(1, 'out');
var myInputPin = new Gpio(2, 'in', 'rising');

...
// STATE can be 1 or 0
myOutputPin.writeSync(STATE);
...
// observe the input pin
myInputPin.watch((err,value) =>
    if(err){
        throw err;
    }
    // do stuff on input rising edge
    console.log("rising");
);
...

// 'unexport' all pins when server is shut down

function unexportOnColose(){
    myOutputPin.writeSyn(0);
    myOutputPin.unexport();
    myInputPin.unexport();
}

process.on('SIGINT', unexportOnClose);
```

links

[onoff API](#)

[Node.js Websocket tutorial](#)

[setup node server video](#)