Programming Study electron

Sungwoo Nam

2018.2.28

Electron

- Develop desktop application using javascript, HTML, CSS.
- Runs on Windows, Linux, Mac
- Based on Chrome

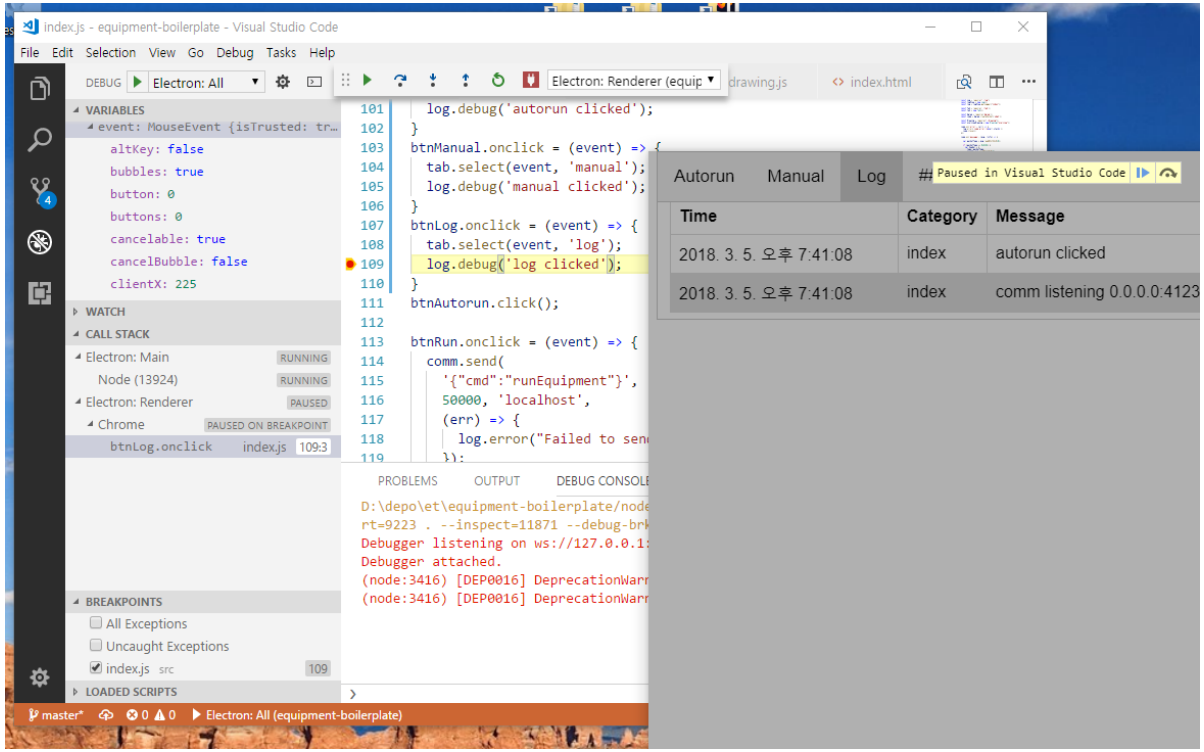
Develop environment

- Install Node.js
- Install VSCode

```
PS D:\depo\et> git clone https://github.com/SungwooNam/equipment-boilerplate.git ecb
Cloning into 'ecb'...
...
Resolving deltas: 100% (67/67), done.
PS D:\depo\et> cd ecb
PS D:\depo\et\ecb> npm install
> electron-chromedriver@1.8.0 install D:\depo\et\ecb\node_modules\electron-chromedriver
> node ./download-chromedriver.js
successfully dowloaded and extracted!
> electron@1.8.2 postinstall D:\depo\et\ecb\node_modules\electron
> node install.js
added 442 packages in 27.565s
PS D:\depo\et\ecb> npm run start
```

Debugging

- ./.vscode/launch.json
- F9 breakpoint, F5 run



Electron Main - Renderer

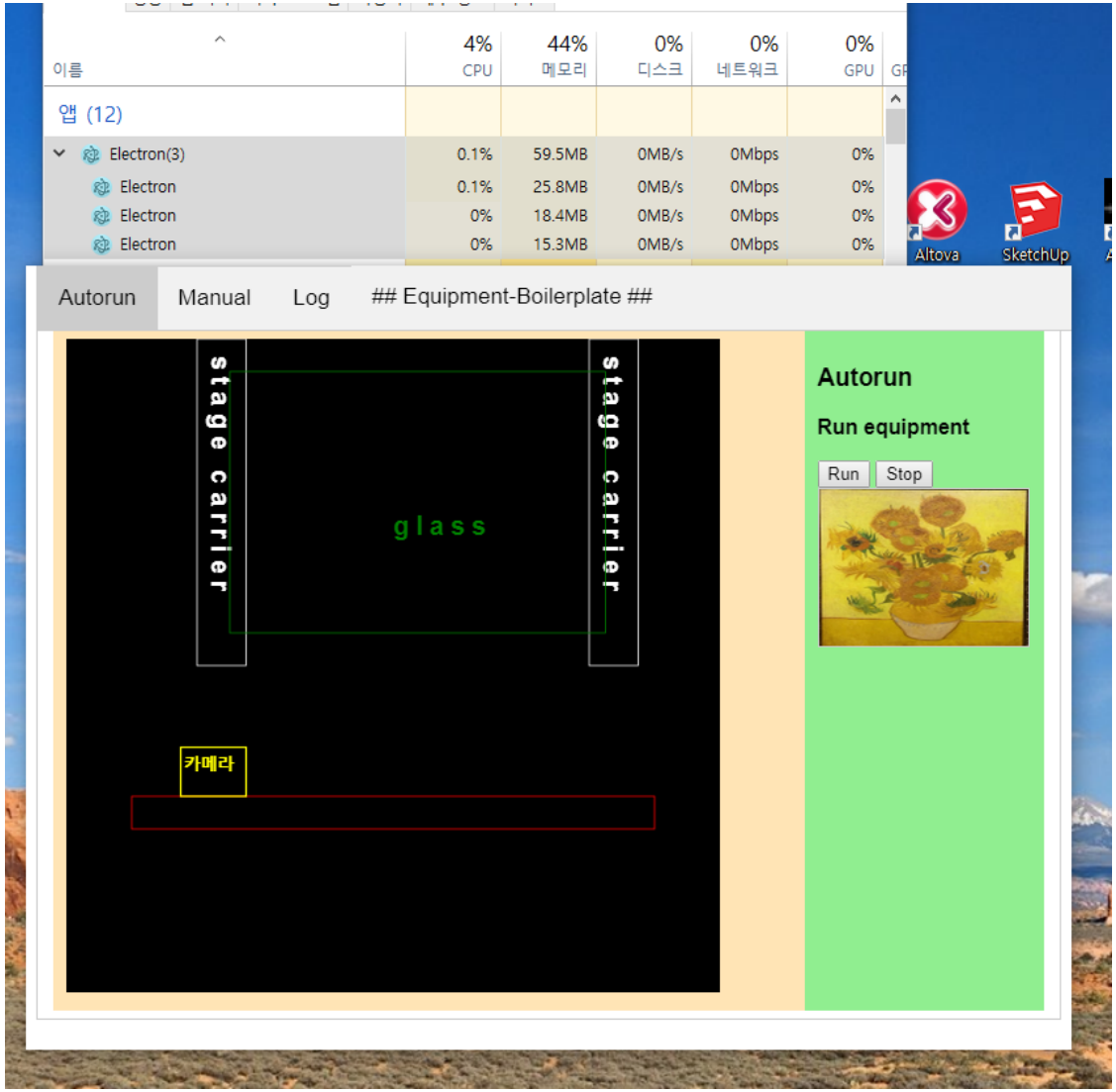
```python
// main.js
const electron = require('electron')
const app = electron.app
const path = require('path')
const url = require('url')

let mainWindow

function createWindow() {
  mainWindow = new electron.BrowserWindow({
    width: 800,
    height: 600,
    frame: false
  })

  mainWindow.loadURL(url.format({
    pathname: path.join(__dirname, 'index.html'),
    protocol: 'file:',
    slashes: true
  }))

  ...
}

app.on('ready', createWindow)
```

```html
<!-- index.html-->
<!DOCTYPE html>
<html>
<head>
…
  <link href="css/styles.css" type="text/css"
rel="stylesheet" />
</head>
<body>
…
  <script>require('./index.js')</script>
</body>
</html>
```

```css
body {
  font-family: Arial;
}
…
```

```python
// index.js
const dgram = require('dgram');
const comm = dgram.createSocket('udp4');
comm.on('message', (msg, rinfo) => {
  var packetType = msg.readUInt32LE(0);
…
```

Electron Main - Renderer



Log usage

```
const Log = require("./log")
const logView = new Log();
const log = logView.getLogger("index");
const comm = dgram.createSocket('udp4');

comm.on('error', (err) => {
  log.error(`comm error:\n${err.stack}`);
});

comm.on('message', (msg, rinfo) => {
  …
  log.info(`comm got: ${msg} from
${rinfo.address} : ${rinfo.port}`);
});

comm.on('listening', () => {
  const address = comm.address();
  log.info(`comm listening ${address.address} :
${address.port}`);
});

comm.bind(41234);

btnAutorun.onclick = (event) => {
  tab.select(event, 'autorun');
  log.debug('autorun clicked');
}
```

```
...
[2018-03-03T01:10:44.488] [INFO] index - comm listening
0.0.0.0:41234
[2018-03-03T01:10:46.370] [DEBUG] index - log clicked
[2018-03-03T01:19:16.796] [DEBUG] index - autorun clicked
...
```

```
[2018-03-05T14:21:27.780] [ERROR] index - Cannot support
image with 160x120x32
...
```

Autorun | Manual | Log | ## Equipment-Boilerplate ##

Time	Category	Message
2018. 3. 5. 오후 5:52:25	index	autorun clicked
2018. 3. 5. 오후 5:52:25	index	comm listening 0.0.0.0:41234
2018. 3. 5. 오후 5:52:26	index	log clicked

Log using log4js

```
// log.js
var log4js = require('log4js');

class Log {
  constructor() {
    …
    log4js.configure('./config/log4js.json');
  }

  getLogger( category ) {
    return log4js.getLogger( category );
  }
…

module.exports = Log;
```

```
...
[2018-03-03T01:10:44.488] [INFO] index - comm listening
0.0.0.0:41234
[2018-03-03T01:10:46.370] [DEBUG] index - log clicked
[2018-03-03T01:19:16.796] [DEBUG] index - autorun clicked
...
```

```
[2018-03-05T14:21:27.780] [ERROR] index - Cannot support
image with 160x120x32
...
```

```
{
   "appenders": {
        "app": {
             "type": "file",
             "filename": "log/app.log",
             "maxLogSize": 10485760,
             "numBackups": 3
        },
        "errorFile": {
             "type": "file",
             "filename": "log/errors.log"
        },
        "errors": {
             "type": "logLevelFilter",
             "level": "ERROR",
             "appender": "errorFile"
        }
   },
   "categories": {
        "default": {
             "appenders": [
                  "app",
                  "errors"
                  ],
             "level": "DEBUG"
        }
     }
```

Log – display using table

```
log4js.addLayout('showImmediate', function (config) {
 return ( log ) => {
   var table = document.getElementById("logTable");
   if (table != null) {
     var row = document.createElement("tr");
     row.innerHTML =
       `<td>${log.startTime.toLocaleString()}</td>
       <td>${log.categoryName}</td>
       <td>${log.data}</td>`;
     table.append(row);
   }
 }
});
log4js.configure('./config/log4js.json');
```

```
{
"appenders": {
  ...
  "out": {
    "type": "stdout",
    "layout": {
      "type": "showImmediate"
    }
...
```

```
<div id="log" class="tabcontent">
  <table id="logTable">
    <col width="30%">
    <col width="10%">
    <col width="60%">
    <tr>
      <th>Time</th>
      <th>Category</th>
      <th>Message</th>
    </tr>
  </table>
</div>
```

Tab – HTML & CSS

```html
<div class="tab">
  <button class="tablinks" id="btnAutorun">Autorun</button>
  <button class="tablinks" id="btnManual">Manual</button>
  <button class="tablinks" id="btnLog">Log</button>
  <label class="tab remaining">## Equipment-Boilerplate
  ##</button>
</div>

<div id="autorun" class="tabcontent">
  <div class="flex-container">
  ...
</div>

<div id="manual" class="tabcontent">
  <div class="flex-container">
  ...
</div>

<div id="log" class="tabcontent">
  <table id="logTable">
  ...
</div>
```

```css
.tab button{
  background-color: inherit;
  ...
}

.tab button:hover {
  background-color: #ddd;
}

.tab button.active {
  background-color: #ccc;
}

.tabcontent {
  display: none;
  ...
}
```

Tab - javascript

```javascript
class Tab {
  constructor() {
    this.content = document.getElementsByClassName("tabcontent");
    this.links = document.getElementsByClassName("tablinks");
  }

  select(evt, tabId) {
    for (var i = 0; i < this.content.length; i++) {
      this.content[i].style.display = "none";
    }
    document.getElementById(tabId).style.display = "block";

    for (var i = 0; i < this.links.length; i++) {
      this.links[i].className = this.links[i].className.replace(" active", "");
    }
    evt.currentTarget.className += " active";
  }
}
module.exports = Tab;
```

```javascript
const Tab = require('./tab');
const tab = new Tab();
btnAutorun.onclick = (event) => { tab.select(event, 'autorun');}
btnManual.onclick = (event) => { tab.select(event, 'manual'); }
btnLog.onclick = (event) => { tab.select(event, 'log'); }
btnAutorun.click();
```

Socket – receive UDP JSON

```
{
  “cmd” : “ioUpdate”,
  “address” : “768”,
  “value” : [ “1”, “0”, “1”, “0”, “1”]
}
```

```
const dgram = require('dgram');
const comm = dgram.createSocket('udp4');

comm.on('message', (msg, rinfo) => {
  var jm = JSON.parse(msg);

  if (jm != null && jm.cmd == 'ioUpdate') {
     if (jm.address == 0x300) {

       var inport = document
         .getElementById('ioIn')
         .getElementsByTagName('input');

       for (var i = 0; i < inport.length; ++i) {
         inport[i].checked =
           jm.value[i] == '1' ? true : false;
       }
     ...
}

comm.bind(41234);
```

```
string IOUpdate2String(int addr, vector<int>& io )
{
  Poco::JSON::Object j;
  j.set("cmd", "ioUpdate");
  j.set("address", addr);

  Poco::JSON::Array v;
  for (auto i : io) { v.add(i); }
  j.set("value", v);

  std::stringstream ss; j.stringify(ss);
  return ss.str();
}

IDatagramPoint* p = ...
SocketAddress remote("127.0.0.1:41234");
string msg = IOUpdate2String(
  0x300,
  vector<int>{ 1, 0, 1, 0, 1 } );
point->SendTo(
  msg.data(), (int)msg.size(),
  remote);
```

Socket – send UDP JSON

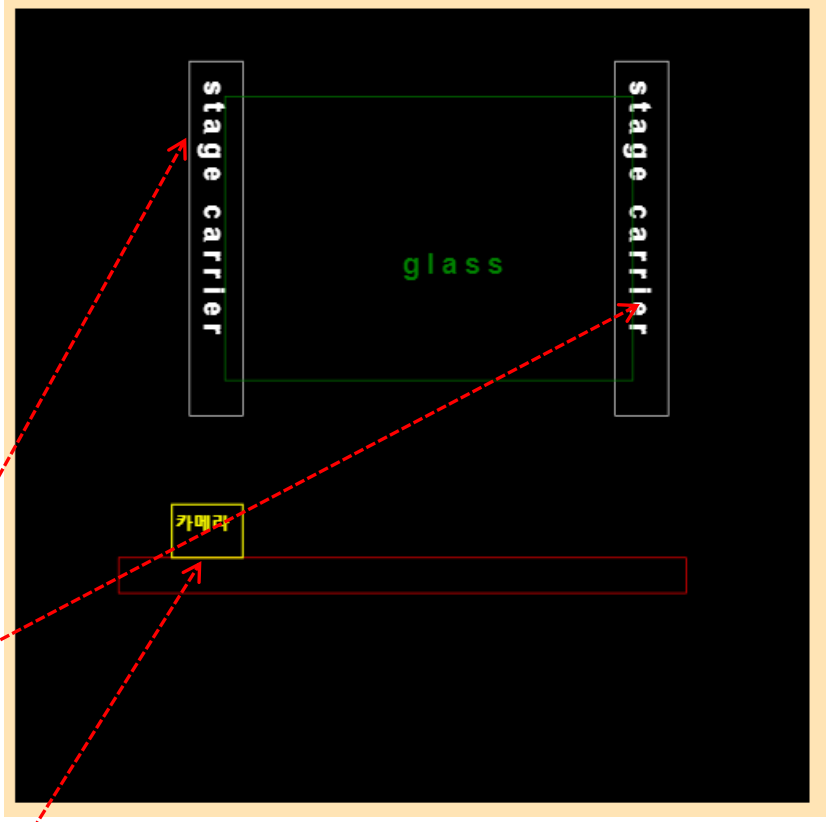
```
{
  “cmd” : “runEquipment”
}
```

```js
const dgram = require('dgram');
const comm = dgram.createSocket('udp4');

comm.bind(41234);

btnRun.onclick = (event) => {
  comm.send(
    '{"cmd":"runEquipment"}',
    50000, 'localhost',
    (err) => {
      log.error("Failed to send runEquipment");
  });
}
```

```
IDatagramPoint::Ptr point =
  m_Fab->CreateDatagramPoint(
    SocketAddress("127.0.0.1:50000"),
    [=](IDatagramPoint* p) mutable
    {
      char buffer[MAX_PATH];
      SocketAddress sender;
      int n = p->ReceiveFrom(
        buffer, sizeof(buffer)-1, sender);
      buffer[n] = '\0';

      Poco::JSON::Parser parser
      auto obj = parser.parse(buffer)
        .extract<Poco::JSON::Object::Ptr>();
      string cmd = obj->get("cmd").toString();
      if (cmd == “runEquipment")
      {
      ...
```

SVG - Graphic

```
<svg id="overview" version="1.1" baseProfile="full" xmlns="htt
    width="500" height="500"
    viewBox="-1000 -1000 2000 2000"
    style="background-color: black;">

    <defs>
        <g id="y_axis_stage">
            <rect x="0" y="0" width="150" height="1000" style=
            <text x="70" y="50" style="letter-spacing:20; fill
        </g>
        <g id="x_shift_camera">
            <rect x="0" y="0" width="200" height="150" style="
            <text x="10" y="70" style="letter-spacing:1; fill:
        </g>
    </defs>

    <g id="overview_stage"
        transform="translate(0 0) rotate(0 0 0)">
        <use xlink:href="#y_axis_stage" x="-600" y="-1000" />
        <use xlink:href="#y_axis_stage" x="600" y="-1000" />
        <rect x="-500" y="-900" width="1150" height="800" styl
        <text x="0" y="-400" style="letter-spacing:20; fill:
    </g>

    <rect x="-800" y="400" width="1600" height="100" style="st

    <g id="overview_shift"
        transform="translate(0 0) rotate(0 0 0)">
        <use xlink:href="#x_shift_camera" x="-650" y="250" />
    </g>
</svg>
```



SVG – moving objects

```
<g id="overview_stage"
   transform="translate(0 0) rotate(0 0 0)">
    <use xlink:href="#y_axis_stage" x="-600" y=
    <use xlink:href="#y_axis_stage" x="600" y="
    <rect x="-500" y="-900" width="1150" height
    <text x="0" y="-400" style="letter-spacing:
</g>

<rect x="-800" y="400" width="1600" height="100

<g id="overview_shift"
   transform="translate(0 0) rotate(0 0 0)">
    <use xlink:href="#x_shift_camera" x="-650"
</g>
```

```
comm.on('message', (msg, rinfo) => {

  if (jm != null && jm.cmd == 'servoUpdate') {
    var ypos = -parseInt(jm.value[0]);
    var xpos = parseInt(jm.value[1]);
    overview_stage.setAttribute(
      "transform", `translate(0 ${ypos}) rotate(0 0 0)`);
    overview_shift.setAttribute(
      "transform", `translate(${xpos} 0 ) rotate(0 0 0)`);
    return;
  }
```

SVG – mouse movement

```
class Drawing {
  constructor( idOfOverviewSVG ) {
    this.id = idOfOverviewSVG;
    this.svg = document.getElementById( idOfOverviewSVG );
    var v = this.svg.getAttribute( "viewBox").split(" ");
    this.view = {
      x: parseInt(v[0]),
      y: parseInt(v[1]),
      width: parseInt(v[2]),
      height: parseInt(v[3]),
    }
    this.mouseDown = 0;
    this.svg.onmousemove = (event) => { this.onMouseMove(event); }
    ...
  }

  onMouseMove( event )
  {
    if (this.mouseDown == 1) {
      this.view.x += event.movementX;
      this.view.y += event.movementY;
      this.svg.setAttribute("viewBox",
        `${this.view.x} ${this.view.y} ${this.view.width} ${this.view.height}`);
      }
  }
```

Canvas - Image

```html
<article class="main">
  <h3>Autorun</h3>
  <p>Run equipment</p>
  <button id="btnRun">Run</button>
  <button id="btnStop">Stop</button>
  <canvas
    id="cameraImage"
    width="160" height="120“
    style="border: 1px solid #cccccc;"
  />
</article>
```


```javascript
window.onload = function () {
  initCameraImage();
}

function initCameraImage() {
  var canvas = document.getElementById(
    "cameraImage");
  var context = canvas.getContext("2d");

  var image = new Image();
  image.src = "./sunflower.png";
  image.onload = (arg) => {
    context.drawImage(image, 0, 0);
  };
}
```

Canvas – fragmented packet

```
struct ImageHeader {
  uint32_t Type;
  uint16_t Length;
  uint16_t Width, Height, BPP;
  uint16_t RegionX, RegionY, RegionWidth, RegionHeight;
};

vector< vector<uint8_t> > GenerateImagePacket(
  const cv::Mat& m )
{
  ...
  // make 64kbyte fragmented image packets

  return packets;
}

auto packets = GenerateImagePacket(
  CreateFilledImage(160, 120, (rand() | 0xFF000000)));
for (auto packet : packets) {
  point->SendTo(packet.data(), (int)packet.size(), remote);
}
```

Canvas – parse binary packet

```
comm.on('message', (msg, rinfo) => {
  var packetType = msg.readUInt32LE(0);
  if (packetType == 0x6182) {
    var header = {
      type: packetType,
      length: msg.readUInt16LE(4),
      width: msg.readUInt16LE(6),
      height: msg.readUInt16LE(8),
      bpp: msg.readUInt16LE(10),
      regionX: msg.readUInt16LE(12),
      regionY: msg.readUInt16LE(14),
      regionWidth: msg.readUInt16LE(16),
      regionHeight: msg.readUInt16LE(18),
    };

    var canvas = document.getElementById("cameraImage");
    var context = canvas.getContext("2d");
    context.putImageData(
      new ImageData(
        new Uint8ClampedArray(msg.buffer, 20),
        header.regionWidth, header.regionHeight
      ),
      header.regionX, header.regionY,
      0, 0, header.regionWidth, header.regionHeight
    );
```