Sistemas Embebidos de Internet de las cosas (IoT)

INTRODUCCIÓN A LINUX EMBEBIDO SOBRE INTEL GALILEO

PONTIFICIA UNIVERSIDAD JAVERIANA

NOVIEMBRE DE 2015, BOGOTÁ COLOMBIA

INGENIERÍA ELECTRÓNICA

1. OnBoard LED blink

```
var mraa = require('mraa');                                  //llamado de la librería
var myOnboardLed = new mraa.Gpio(3, false, true);   //Se asigna el Puerto del LED (solo para
                                                     //la Galileo G1 es de esta manera de lo
                                                     //contrario es: new mraa.Gpio(13))
myOnboardLed.dir(mraa.DIR_OUT);                  //Se configura el Puerto como salida
var ledState = true;                                         //Variable para el estado del LED
periodicActivity();                                          //Llamado de la funcion

function periodicActivity()                                  //función periodicActivity
{
  myOnboardLed.write(ledState?1:0);                          //Cambio de estado del LED
  ledState = !ledState;                                      //Se invierte la variable ledState
  setTimeout(periodicActivity,1000);                         //Se llama la función cada segundo
}
```

2. Button changes state of LED (no ISR)

```
var mraa = require('mraa');                                  //llamado de la librería
var led = new mraa.Gpio(3, false, true);         //Se asigna el Puerto del LED (solo para
                                                 //la Galileo G1 es de esta manera de lo
                                                 //contrario es: new mraa.Gpio(13))
led.dir(mraa.DIR_OUT);                           //Se configura el Puerto como salida
var ahora=false;                                             //Variable p
var antes = false;
var ledState=true;
var boton = new mraa.Gpio(32, false, true);                  //Se asigna el Puerto del boton
boton.dir(mraa.DIR_IN);                          //Se configura como entrada
periodicActivity();                              //Se hace el llamado de la función

function periodicActivity()                      //Función periodicActivity
{
  ahora = boton.read();                          //Se lee el Puerto del boton
    if ((ahora == 1) && (antes == 0)){           //Se verifica si hay un flanco de subida
        led.write(ledState?1:0);                 //Se cambia el estado del LED
        ledState = !ledState;                    //Se niega la variable ledState
    }
  antes = ahora;                                 //Se guarda el estado actual del boton
  setTimeout(periodicActivity,1);                //Se auto-llama la función cada milisegundo
}
```

3. Read an analog input (temperature) pint on the console

```
var mraa = require('mraa');                      //llamado de la librería
var analogPin0 = new mraa.Aio(0);                //Se configure el acceso al puerto (A0)
const B = 4275;                                  //B valor del termistor
periodicActivity();                              //Se hace el llamado de la función

function periodicActivity()                      //Función periodicActivity
{
    var analogValue = analogPin0.read();         //Se lee el valor del puerto
    resistance = 100000*(1023/(analogValue)-1);          //Se obtiene la
                                                         //resistencia del sensor
          //Se convierte la temperatura (según el datasheet)
```

```
    temperature=1.0/(Math.log(resistance/100000.0)/B+1/298.15)-273.15;
    console.log(temperature);                          //Se imprime la temperatura
    setTimeout(periodicActivity,1000);                 //Se auto-llama la función cada
}
```

4. Control a LED (PWM) depending on the light intensity

```
var mraa = require('mraa');                            //llamado de la librería
var analogPin0 = new mraa.Aio(1);                 //Se configure el acceso al puerto (A0)
var pwm3 = new mraa.Pwm(3);                            //Se inicializa el puerto PWM 3
pwm3.enable(true);                                     //Se habilita el puerto PWM 3
pwm3.period_us(2000);                            //Se configura el periodo en microsegundos
var value = 0;
periodicActivity();                                    //Se hace el llamado de la función

function periodicActivity()                            //Función periodicActivity
{
    var analogValue = analogPin0.read();               //Se lee el A0
    value = 1 - (1/712)*analogValue;                   //Conversión
    pwm3.write(value);                           //Se escribe un valor entre 0-1 (ciclo útil)
    setTimeout(periodicActivity,500);             //Se auto-llama la función cada 0.5 seg

}
```

5. Show the temperature on the LCD display, and modify the background color if the temperature is above certain threshold

```
const B = 3975;                                        //B valor del termistor
var mraa = require('mraa');                            //llamado de la librería
var lcd = require('jsupm_i2clcd');                     //Llamado librería LCD
var display = new lcd.Jhd1313m1(0, 0x3E, 0x62);        //Se configure la referencia del LCD
var analogPin0 = new mraa.Aio(0);                 //Se configure el acceso al puerto (A0)
periodicActivity();                                    //Se hace el llamado de la función

function periodicActivity()                            //Función periodicActivity
{
    var analogValue = analogPin0.read();               //Se lee el valor del puerto
    resistance = (1023-analogValue)*10000/analogValue;             //Se obtiene la
                                                                   //resistencia del sensor


          //Se convierte la temperatura (según el datasheet)
   temperature = 1/(Math.log(resistance/10000)/B+1/298.15)-273.15;

    if(temperature>25){                                //Limite de temperatura
        display.setColor(255, 0, 0);                   //Se enciende el LCD en rojo
        display.setCursor(0,0);                        //Se configura el cursor
        display.write("Temp:" + temperature);          //Se imprime en el LCD
    }else{
        display.setColor(0, 0, 255);                   //Se enciende el LCD en azul
        display.setCursor(0,0);                        //Se configura el cursor
        display.write("Temp:" + temperature);          //Se imprime en el LCD
    }

    console.log(temperature);                          //Se imprime la temperatura
    setTimeout(periodicActivity,1000);               //Se auto-llama la función cada segundo
}
```

6. Show the temperatura on the WEB

```
var mraa = require('mraa');                                          //llamado de la librería
var analogPin0 = new mraa.Aio(0);                               //Se configure el acceso al puerto (A0)
var myOnboardLed = new mraa.Gpio(3, false, true);    //Se asigna el Puerto del LED (solo para
                                                     //la Galileo G1 es de esta manera de lo
                                                     //contrario es: new mraa.Gpio(13))
```

```
myOnboardLed.dir(mraa.DIR_OUT);                                  //Se asigna el Puerto como salida
var ledState = false;                                            //Variable para estado del LED

const B = 4275;                                                  //B valor del termistor
var http = require('http');                                      //llamado de la librería http
http.createServer(function (req, res) {                          //Se crea el servidor

    var analogValue = analogPin0.read();                         //Se lee el valor del puerto A0
    resistance = 100000*(1023/(analogValue)-1);                  //Resistencia del sensor
                                                        //Se realiza el cálculo de la temperatura
    temperature=1.0/(Math.log(resistance/100000.0)/B+1/298.15)-273.15;

    if(req.url === "/ledOn"){                                    //Se lee le mensaje de
                                                                 //requerimiento del cliente
        myOnboardLed.write(1);                                   //Se enciende el LED
        ledState=true;
    }else if(req.url === "/ledOff"){                             //Se lee le mensaje de
                                                                 //requerimiento del cliente

        myOnboardLed.write(0);                                   //Se apaga el LED
        ledState=false;
    }
                                                                 //Página WEB
        res.writeHead(200, { "Content-Type": "text/html" });
        res.write("<!DOCTYPE html><html><body>");
        res.write("<h1>Temperatura</h1>");
        res.write("<p>Sensor value : " + temperature);
        res.write("<p><input type='button' onclick='location.reload();' value='Refresh
Temperature'/></p>");
        res.write("</p><p>LED Status : "+ ledState);
        res.write("<p><input type='button' onclick='window.location
=\"http://192.168.137.6:1337/ledOn\"' value='Turn LEDon'/></p>");
        res.write("<p><input type='button' onclick='window.location
=\"http://192.168.137.6:1337/ledOff\"' value='Turn LEDoff'/></p>");
        res.write("</body></html>");
        res.end();

}).listen(1337);                                                 //Servidor en el puerto 1337
```

7. Changing intensity of a LED

INDEX.HML

```
<!DOCTYPE html>
<html>
    <head>
        <title>Intensidad de un LED</title>
            <script src="js/jquery.min.js"></script>

        <link rel="stylesheet" type="text/css" href="js/jquery-ui.css"/>
        <script src="js/jquery-ui.js"></script>
        <script src="/socket.io/socket.io.js"></script>

        <script>

            $(document).ready(function(){

                $( "#slider_price" ).slider({
                        range: true,
                        min: 0,
                        max: 1,
                        step:0.00002,
                        values: [ 0, 25000000000000000000 ],
                        slide: function( event, ui ) {
                            $( "#app_min_price" ).text(ui.values[0] + "$");
                            socket.emit('valor',ui.values[0]);
                        },
                });
```

```
                    var socket = io.connect();

            });

        </script>
        <style>
            .leftSide{ float:left; min-width:25%; min-height:400px; margin:10px;
padding:15px; border:1px solid #333;}

            .rightSide{ float:right; min-width:65%;min-height:400px; margin:10px;
padding:15px;border:1px solid #333; }
        </style>
    </head>
    <body>
        <h2>Intensidad de un LED</h2>
        <div>
            <span id="app_min_price" >?</span> Ciclo util

            <br /><br />
            <div id="slider_price"></div>
            <br />
        </div>
    </body>
</html>
```

MAIN.JS

```
var socketio = require('socket.io');
var express = require('express');
var app = express();
app.use(express.static(__dirname));
var server = app.listen(8085);
var io = require('socket.io').listen(server);

var mraa = require('mraa'); //require mraa
var pwm3 = new mraa.Pwm(3);
pwm3.enable(true);                                          //Se habilita el puerto PWM 3
pwm3.period_us(2000);

//Controlamos el evento cuando se conecte e imprimimos en consola el mensaje recibido


//emitiendo el evento hello hacia el cliente con un mensaje inicial.
io.sockets.on('connection', function (socket) {

  socket.on('hello', function (msg) {
    console.log('Received: %s', msg);
    socket.emit('hello', 'Hello from server.');
  });

  socket.on('valor', function (msg) {
    console.log('Received.: %s', msg);
    pwm3.write(msg);
  });

});
```

8. Temperature tachometer

MAIN.JS

```
var express = require('express');
var app = express();
app.use(express.static(__dirname));
var server = app.listen(8085);
var io = require('socket.io').listen(server);
const B = 3975;
```

```
var mraa = require('mraa'); //require mraa
console.log('MRAA Version: ' + mraa.getVersion()); //write the mraa version to the Intel XDK
console

var myOnboardLed = new mraa.Gpio(3, false, true);  //LED hooked up to digital pin 13 (or
built in pin on Intel Galileo Gen2 as well as Intel Edison)
myOnboardLed.dir(mraa.DIR_OUT); //set the gpio direction to output

var analogPin0 = new mraa.Aio(0);

io.sockets.on('connection', function (socket) {

    setInterval(function () {
        socket.emit( 'temp' , getTemp());  //send temp every interval
    }, 50);

});

function getTemp(){

    var analogValue = analogPin0.read();                 //Se lee el valor del puerto
    resistance = (1023-analogValue)*10000/analogValue;             //Se obtiene la
   temperature = 1/(Math.log(resistance/10000)/B+1/298.15)-273.15;
    var fahrenheit=temperature+32;
    return  fahrenheit  ;

}
```

INDEX.HML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Thermometer Demo</title>

    <script src="js/jquery-1.11.1.min.js"></script>
    <script src="js/gauge.min.js"></script>
    <script src="/socket.io/socket.io.js"></script>
    <script>

    var socket = io.connect();
        socket.on('temp', function (data) {
            //console.log(data);
            Gauge.Collection.get('temp').setValue(data);
        });
    </script>

</head>
<body>

            <div>
                <canvas id="temp" width="200" height="200"
                   data-type="canv-gauge"
                   data-title="Temperature"
                   data-min-value="40"
                   data-max-value="100"
                   data-major-ticks="40 50 60 70 80 90 100"
                   data-minor-ticks="10"
                   data-stroke-ticks="true"
                   data-units=" degrees F"
                   data-value-format="3.1"
                   data-glow="true"
                   data-animation-delay="10"
                   data-animation-duration="200"
                   data-animation-fn="bounce"
                   data-colors-needle="#f00 #00f"
                   data-highlights="40 80 #06B70F, 80 90 #F5E940, 90 100 #FB0808"
```

```
            ></canvas>
          </div>

</body>
</html>
```