

Laboratorio de Programación de Sistemas Embebidos en Red

José Luis Cánovas Sánchez
Ezequiel Santamaría Navarro

18 de enero de 2016

Resumen

En nuestra práctica final del laboratorio hemos integrado todos los sensores estudiados en la primera parte con un servidor web integrado en una raspberry pi. Por eso dividimos la memoria en describir los sensores y arduino (la primera parte de las prácticas), y la segunda en el código final que comunica la raspberry pi y arduino (la segunda parte de trabajo propio).

Índice

1. Práctica 1. Introducción a Arduino. Descripción del hardware	2
1.1. Led	3
1.2. Sensor de luz	3
1.3. NTC	3
1.4. LDR	3
1.5. LCD	3
1.6. Generador señal tipo GPS	3
2. Práctica 2. Montaje de un servidor web de monitorización. Descripción del software	4
2.1. Montaje de red	4
2.2. Protocolo serie	4
2.3. Código arduino	4
2.4. Servicio web	4
2.4.1. Parte servidor de python	4
2.4.2. Parte cliente de javascript	5
3. Notas adicionales	5
3.1. Aproximación a la temperatura real	5
3.2. Lector de pulsaciones	5
4. Código	5
4.1. Código Arduino	5
4.2. Código Raspberry Pi	8

1. Práctica 1. Introducción a Arduino. Descripción del hardware

Para poder utilizar directamente un cable entre la Raspberry Pi y Arduino, sin necesidad de un divisor de tensión para pasar de 5v de Arduino UNO a 3.3v de la Raspberry Pi, usamos la versión de Arduino DUE, que usa en sus pines 3.3v. Esto sólo afectará a la comunicación Arduino-Raspberry y al Led, porque son los únicos que se conectan directamente a un pin de Arduino, porque el resto de sensores los conectamos al pin de 5v que ofrece Arduino DUE, y la entrada de datos se realiza por los puertos analógicos.

Vamos a describir brevemente en cada apartado cómo integramos cada sensor con el Arduino. En la Figura 1 se ve el sistema global donde se puede comprobar cada esquema individual.

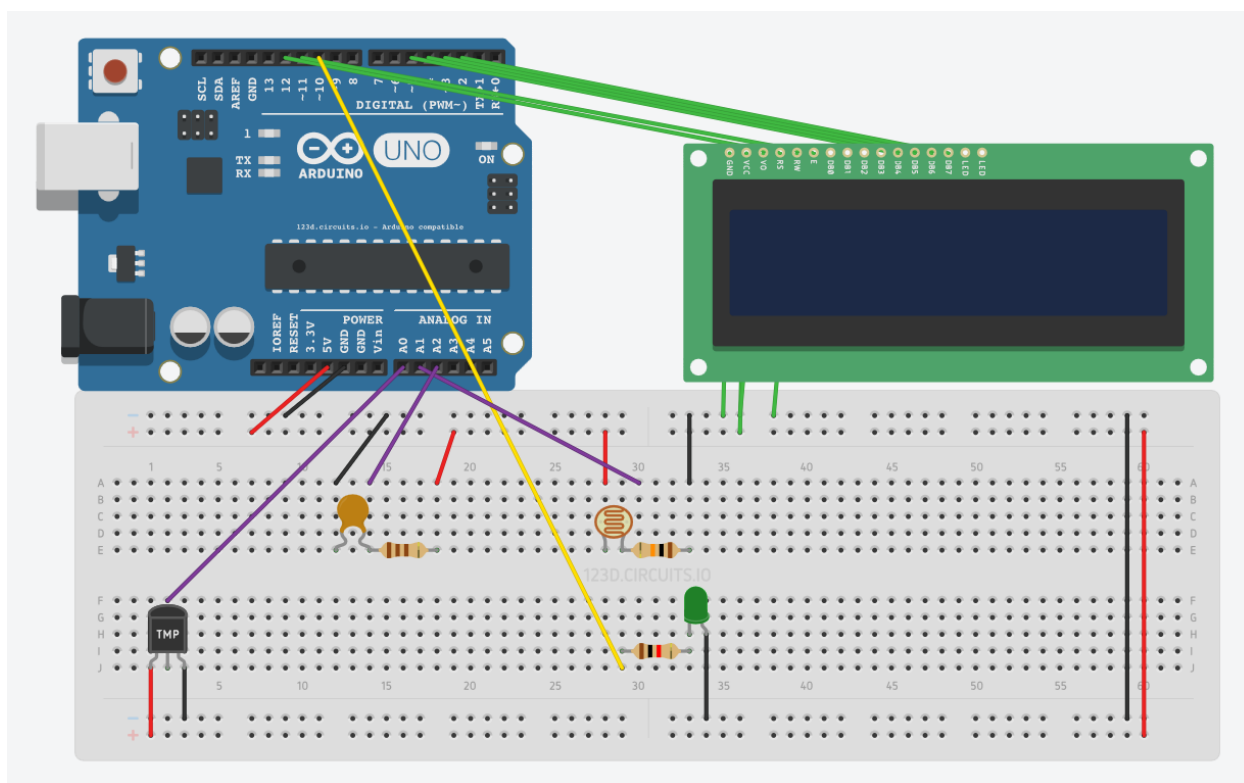


Figura 1: Esquema global

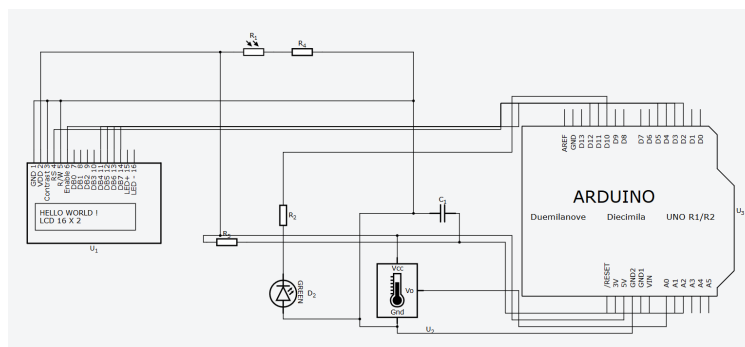


Figura 2: Esquema global

1.1. Led

1.2. Sensor de luz

1.3. NTC

1.4. LDR

1.5. LCD

1.6. Generador señal tipo GPS

```
1 #include <stdbool.h>
2 #include <LiquidCrystal.h>
3
4 // initialize the library with the numbers of the interface pins
5 LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
6
7 unsigned short r1 = 2047;
8 unsigned short r2 = 2047;
9
10 #define B11 0x01
11 #define B10 0x02
12 #define B9 0x04
13 #define B8 0x08
14 #define B7 0x010
15 #define B6 0x020
16 #define B5 0x040
17 #define B4 0x080
18 #define B3 0x0100
19 #define B2 0x0200
20 #define B1 0x0400
21
22 unsigned short G1() {
23     r1 >>= 1;
24     r1 |= (((r1&B4) >> 7) ^ (r1&B11) ? B1 : 0);
25     return r1&B11;
26 }
27
28 unsigned short G2() {
29     r2 >>= 1;
30     r2 |= (((r2 & B3) >> 8) ^ ((r2 & B4) >> 7) ^ ((r2 & B7) >> 4) ^ ((r2 & B9) >> 2) ^ ((r2 &
        B10) >> 1) ^ (r2 & B11) ? B1 : 0);
31     return (((r2&B4) >> 5) ^ (r2&B9)) >> 2;
32 }
33
34 void setup() {
35     // put your setup code here, to run once:
36     lcd.begin(16, 4);
37     pinMode(9, OUTPUT);
38 }
39
40 void loop() {
41     // put your main code here, to run repeatedly:
42     //lcd.setCursor(0,0);
43     //lcd.print(G1()^G2());
44     digitalWrite(9, G1()^G2());
45     //delay(500);
46 }
```

serialGPS.ino

2. Práctica 2. Montaje de un servidor web de monitorización. Descripción del software

En la segunda parte de la práctica hemos puesto como objetivo desarrollar algo en el ámbito de internet de las cosas. Básicamente nuestro reto ha sido poner en internet (en nuestro caso usando HTTP y servicios Web) los distintos sensores y el monitor LCD utilizados en la práctica anterior.

Para ello nos hemos servido de un Arduino Due (ya que el voltaje de sus pines es compatible con la Raspberry Pi), de una Raspberry Pi y de un router Wifi que hace de punto de acceso entre la Raspberry Pi e internet.

2.1. Montaje de red

La Raspberry Pi, por su gran capacidad de cómputo, hace de centro de datos. Recibe información del Arduino Due a través de el puerto serie, y lo pone en internet a través de un puerto Ethernet.

2.2. Protocolo serie

El protocolo serie es muy simple, y se basa en transmitir unas tramas con la siguiente información (en ese orden):

- 4 bytes que codifican en ASCII hexadecimal el tamaño de la trama.
- 1 byte que codifica en ASCII el tipo del paquete. Puede ser
- Varios bytes con el payload de la trama, en ASCII. No pueden contener el byte 0.
- 1 byte nulo (valor 0) que inidica fin de la trama. Se utiliza para sincronizar en caso de error.

2.3. Código arduino

Lo primero que hemos hecho ha sido reutilizar parte del código de la primera práctica. En un bucle continuo, leemos los sensores de temperatura y luz y con la información obtenida mandamos frames por el protocolo serie a la Raspberry Pi.

Por otro lado, también nos encargamos de leer frames que puedan venir de la Rapsberry Pi. Si llega un frame de tipo 'L', le cambiamos el valor al *analog output* del led, haciendo que ilumine más o menos. O por si llega un frame de tipo 'D', en cuyo caso limpiamos el display y ponemos el nuevo texto en la pantalla.

2.4. Servicio web

El servicio web tiene dos partes, un código escrito en python que corre en la Rapsberry PI y otro código escrito en javascript que se ejecuta en un teléfono móvil, o un pc.

2.4.1. Parte servidor de python

En esta parte, se espera a que un navegador web estándar haga una petición de tipo GET para obtener el documento index.html. Además, está a la espera de recibir peticiones GET al documento 'luz' o al documento 'tmp'.

Cada vez que recibe una petición al documento luz, lee las tramas a través del puerto serie para obtener del Arduino la última lectura del sensor de luz. Después, contesta la petición con el valor del sensor. Lo mismo sucede si se recibe una petición al documento tmp, solo que se devuelve la temperatura.

Por otro lado, el servidor de python también está a la espera de las peticiones PUT 'led' y 'display'. En este caso, en vez de devolver un valor leyendo por el puerto serie, lo que hace es escribir en el puerto serie una trama de tipo 'L' (si led) o de tipo 'D' (si display) para alterar el brillo del led o el texto del monitor LCD.

2.4.2. Parte cliente de javascript

En el explorador web, se ejecuta continuamente una petición de tipo GET 'luz' y GET 'tmp', para actualizar en tiempo real la oscuridad del fondo (dependiendo del nivel de luz), o el valor de un número dependiendo de la temperatura.

También se pone una barra de desplazamiento para modificar el brillo del led, y una barra de texto para cambiar el mensaje del display. Cuando se mueve la barra de desplazamiento se ejecuta un PUT 'led', y cuando se cambia el texto en la barra se ejecuta un PUT 'display', que llegan al servidor web de python.

3. Notas adicionales

3.1. Aproximación a la temperatura real

Siguiendo las indicaciones de los fabricantes de los sensores de temperatura hemos llegado a lecturas que no pueden corresponderse con la realidad (porque aparecían disparates), así que hemos tenido que aproximar la temperatura con la experimentación.

3.2. Lector de pulsaciones

4. Código

4.1. Código Arduino

Dos ficheros, uno con el código de los sensores de la primera parte, usando el Arduino UNO, y el segundo el código de Arduino DUE con el protocolo de comunicación y lectura de sensores.

Código sensores Arduino UNO:

```
1 #define pinLed 10
2 #define tempResN A2
3 #define tempResP A0
4 #define luzRes A1
5 #include <LiquidCrystal.h>
6
7 LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
8
9 void setup() {
10     // Led:
11     pinMode(pinLed, OUTPUT);
12
13     // Debug
14     Serial.begin(9600);
15
16     // LCD
17     lcd.begin(16, 2);
18 }
19
20 void loop() {
21     delay(20);
22
23     // Temperaturas:
24     Serial.println("Resistencia positiva:");
```

```

25 int res = analogRead(tempResP);
26 double temp = (4.88*res-400)/19.5-10.0;
27 Serial.println(temp);
28
29 Serial.println(" Resistencia negativa:");
30 res = analogRead(tempResN);
31 Serial.println((1.-res/1024.)*100.-25.);
32
33 // Luz:
34 Serial.println(" Resistencia de luz:");
35 res = analogRead(luzRes);
36 Serial.println(128-(unsigned char)(res/4));
37
38 // Led:
39 res = res/4 > 127 ? 127 : res/4;
40 analogWrite(pinLed,(128-res)*2-1);
41
42 // LCD:
43 lcd.setCursor(0,0);
44 lcd.print("Temp = ");
45 lcd.print(temp);
46
47 }

```

codigoSensoresArduinoUNO.ino

Código Arduino DUE:

```

1 #define LED_PIN 10
2 #define luzRes A1
3 #define tempResN A2
4 #include <Streaming.h>
5 #include <LiquidCrystal.h>
6
7 struct paquete {
8     paquete() {
9         this->reserved = 0;
10    }
11
12    char longitud[4];
13    char tipo;
14    char payload[555];
15    char reserved;
16    void leer() {
17        char test;
18        Serial1.readBytes(&test, 1);
19        while (test != '\255')
20            Serial1.readBytes(&test, 1);
21        Serial1.readBytes(longitud, 4);
22        l = hex2int(longitud, 4);
23        Serial1.readBytes(&tipo, 1);
24        Serial1.readBytes(payload, l-5);
25        if (l-5 < 555)
26            payload[l-5] = 0;
27    }
28
29    void enviar() {
30        if (tipo == 'l' || tipo == 'L' || tipo == 't')
31            this->enviar(7);
32    }
33
34    void enviar(int longitud) {
35        Serial1.write("\255", 1);
36        Serial1.write(this->longitud, 4);
37        Serial1.write(&tipo, 1);

```

```

38     Serial1.write(payload, longitud-5);
39 }
40
41 long length() {
42     return l;
43 }
44 private:
45     long l;
46 };
47
48 unsigned long hex2int(char *a, unsigned int len)
49 {
50     unsigned int i;
51     unsigned long val = 0;
52
53     for(i=0; i<len; i++)
54         if(a[i] <= 57)
55             val += (a[i]-48)*(1<<(4*(len-1-i)));
56         else
57             val += (a[i]-87)*(1<<(4*(len-1-i)));
58     return val;
59 }
60
61 void int2hex(char * a, unsigned int n) {
62     char x1 = n%16;
63     char x2 = (n>>4)%16;
64     a[1] = x1+'0';
65     a[0] = x2+'0';
66
67     if (x1 >= 10)
68         a[1] = (x1-10)+'A';
69     if (x2 >= 10)
70         a[0] = (x2-10)+'A';
71 }
72
73 struct paquete luz;
74 struct paquete tmp;
75 LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
76
77 void setup() {
78     strcpy(luz.longitud, "000");
79     luz.longitud[3] = '7';
80     luz.tipo = 'l';
81
82     strcpy(tmp.longitud, "000");
83     tmp.longitud[3] = '7';
84     tmp.tipo = 't';
85
86     // set the data rate for the SoftwareSerial port
87     Serial1.begin(4800);
88     Serial.begin(9600);
89     //Serial1.println("");
90     pinMode(LED_PIN, OUTPUT);
91     analogWrite(LED_PIN, 255);
92
93     lcd.begin(20, 4);
94     lcd.setCursor(0, 0);
95     lcd.print("abcdefghijklmnopqrstuvwxyz0123456789abcdefghijklmnopqrstuvwxyz0123456789");
96 }
97
98 void loop() { // run over and over
99     struct paquete p;
100     if (Serial1.available()) {
101         p.leer();
102         if (p.tipo == 'L') {

```

```

103     Serial.write(p.payload,2);
104     Serial.write("\n",1);
105     long intensidad = hex2int(p.payload,2);
106     Serial.println(intensidad);
107     analogWrite(LED_PIN, intensidad);
108 } else if (p.tipo == 'D') {
109     Serial.println(p.payload);
110     lcd.setCursor(0,0);
111     lcd.print(p.payload);
112 }
113 } else {
114     int res = analogRead(luzRes);
115     res = (unsigned char)(res/4);
116     int2hex(luz.payload, res);
117     luz.enviar();
118
119     res = analogRead(tempResN);
120     int2hex(tmp.payload, res/4);
121     tmp.enviar();
122
123     delay(50);
124 }
125 }

```

pser.ino

4.2. Código Raspberry Pi

Dos ficheros, el servidor web escrito en Python, implementando el protocolo de comunicación con Arduino DUE, y el fichero *index.html* de la página web que muestran la información de los sensores o entrada de datos para pantalla y led.

Código servidor Python:

```

1 import sys
2 import BaseHTTPServer
3 from SimpleHTTPServer import SimpleHTTPRequestHandler
4 import RPi.GPIO as GPIO
5 import time
6 import serial
7 GPIO.cleanup() ## Hago una limpieza de los GPIO
8 GPIO.setmode(GPIO.BCM)
9 #GPIO.setup(21, GPIO.OUT) ## GPIO 17 como salida
10
11 port = serial.Serial("/dev/ttyAMA0", baudrate=4800, timeout=3.0)
12
13 class paquete:
14     puerto = None
15     def __init__(self, s):
16         self.puerto = s
17
18     def enviar(self):
19         bytes = self.long + self.tipo + self.payload
20         length = int(self.long)
21         if (length > len(bytes)):
22             bytes += "0"*(length - len(bytes))
23         self.puerto.write("\255"+bytes)
24
25     def leer(self):
26         print self.long + self.tipo + self.payload
27         sync = self.puerto.read(1)
28         while sync != "\255":
29             sync = self.puerto.read(1)
30         self.long = self.puerto.read(4)

```



```

31         self.tipo = self.puerto.read(1)
32         self.payload = self.puerto.read(int(self.long,16)-5)
33
34         long = "0007"
35         tipo = "L"
36         payload = "0"
37
38
39
40 luz = 0
41 tmp = 0
42 class PUTHandler(SimpleHTTPRequestHandler):
43     p = paquete(port)
44     def do_PUT(self):
45         if self.path == "/led":
46             length = int(self.headers['Content-Length'])
47             content = self.rfile.read(length)
48             self.send_response(200)
49             x = int(content)%256
50             self.p.tipo = "L"
51             self.p.payload = hex(x)[2:]
52             self.p.payload = (2-len(self.p.payload))*"0"+self.p.payload
53             self.p.enviar()
54         elif self.path == "/display":
55             length = int(self.headers['Content-Length'])
56             content = self.rfile.read(length)
57             content = content[0:min(length,80)]
58             if length < 80:
59                 content = content + " "*(80-len(content))
60             print content
61             content = content[0:20]+content[40:60]+content[20:40]+content[60:80]
62             print len(content)
63             self.send_response(200)
64             self.p.long = "0055"
65             self.p.tipo = "D"
66             self.p.payload = content
67             self.p.enviar()
68
69
70     def do_GET(self):
71         global luz
72         global tmp
73         if self.path in ["/luz", "/tmp"]:
74             while self.p.puerto.inWaiting() > 0:
75                 self.p.leer()
76                 if (self.p.tipo == "l"):
77                     luz = self.p.payload
78                 elif (self.p.tipo == "t"):
79                     tmp = self.p.payload
80             self.send_response(200)
81             self.send_header("Content-type", "text/plain")
82             self.end_headers()
83             self.wfile.write([luz,(1.-int(tmp,16)/256.)*200.-2.][self.path == "/tmp"])
84         return
85     return SimpleHTTPRequestHandler.do_GET(self)
86
87
88 HandlerClass = PUTHandler
89 ServerClass = BaseHTTPServer.HTTPServer
90 Protocol = "HTTP/1.0"
91
92 if sys.argv[1:]:
93     port = int(sys.argv[1])
94 else:
95     port = 8000

```

```

96 server_address = ('', port)
97
98 HandlerClass.protocol_version = Protocol
99 httpd = ServerClass(server_address, HandlerClass)
100
101 sa = httpd.socket.getsockname()
102 print "Serving HTTP on", sa[0], "port", sa[1], "..."
103 try:
104     httpd.serve_forever()
105 except:
106     GPIO.cleanup()
107     print "limpiando"

```

restserver.py

Código página web:

```

1 <html>
2
3 <meta name="viewport" content="width=device-width, initial-scale=1.0">
4 <meta charset="UTF-8">
5
6 <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
7 <body>
8 <div class="temperatura" id="tmp">25</div>
9 <div class="centro">
10 <input min="0" max="255" type="range" id="myRange" value="100" oninput="change(this.value)"
    onchange="change(this.value);">
11 <input type="text" value="Ayy lmao" onkeyup="display(this.value);">
12 </div>
13 </body>
14
15 <style>
16 body, html {
17     padding: 0;
18     margin: 0;
19     width: 100vw;
20 }
21
22 .centro {
23     position: relative;
24     display: flex;
25     display: -webkit-flex;
26
27     flex-flow: column;
28     -webkit-flex-flow: column;
29     position: fixed;
30     width: 95vw;
31     height: 20vh;
32     left: 2.5vw;
33     top: 40vh;
34 }
35
36 .centro > * {
37     width: 100%;
38     margin-top: 30px;
39 }
40
41 .temperatura {
42     font-size: 20px;
43     position: fixed;
44     left: 20px;
45     top: 20px;
46 }
47

```

```

48 </style>
49
50 <script>
51 function change(e) {
52     console.log(e);
53     $.ajax({
54         url: '/led',
55         type: 'PUT',
56         data: e,
57         success: function(result) {
58             console.log(result);
59         }
60     });
61 }
62
63 function display(e) {
64     console.log(e);
65     $.ajax({
66         url: '/display',
67         type: 'PUT',
68         data: e,
69         success: function(result) {
70             console.log(result);
71         }
72     });
73 }
74
75 setInterval(function () {
76     //console.log("lmao");
77     $.ajax({
78         url: "/luz",
79         type: "GET",
80         success: function(result) {
81             result = ""+result;
82             //console.log(result+result+result);
83             if (result.length < 2)
84                 result = "0"+result;
85             $("body").css({"background-color": "#"+result+result+result});
86         }
87     });
88     $.ajax({
89         url: "/tmp",
90         type: "GET",
91         success: function(result) {
92             result = ""+result;
93             //console.log(result+result+result);
94             result = parseFloat(result);
95             $("#tmp").get(0).innerHTML = result.toFixed(2)+" C";
96         }
97     });
98 }, 50);
99
100
101 </script>
102 </html>

```

index.html