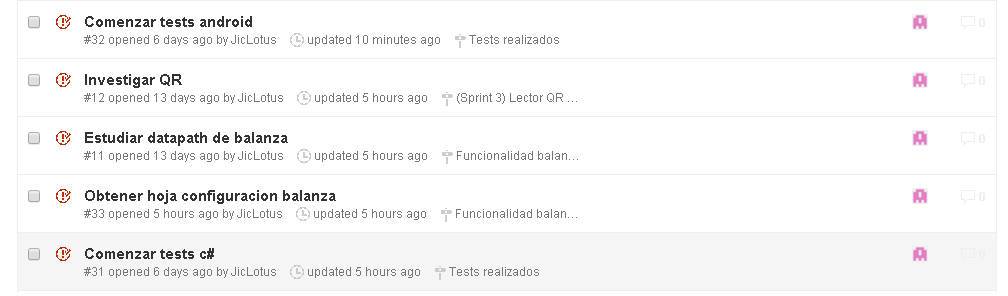
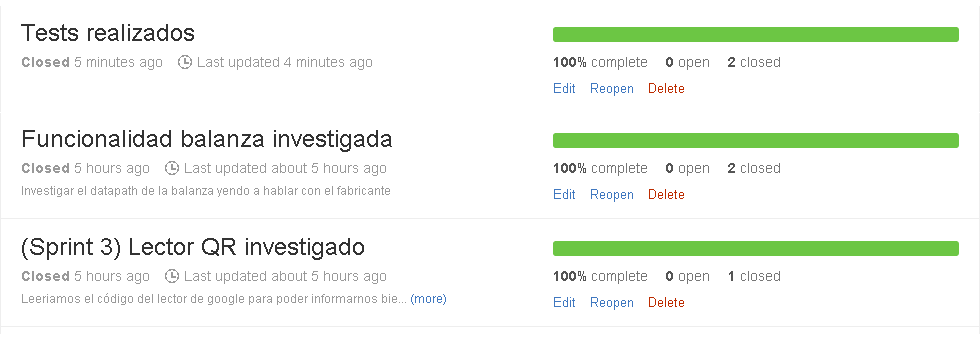
**Calendario Sprint 3**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Investigación funcionalidad balanza bobinas | Investigar el datapath de la balanza yendo a hablar con el fabricante | 7 | Realizado | 7/01/2016 |  |
| Realizar tests | Realizar los tests respectivos a la aplicación C# y android | 5 | Realizado | 7/01/2016 |  |
| Investigación lector QR | Leeríamos el código del lector de google para poder informarnos bien e implementarlo en nuestra aplicación | 5 | Realizado | 8/01/2016 | Sprint3 |

Issues sprint3:



Hitos Sprint3:

**Funcionalidades balanza de bobina**



La balanza provista por el cliente, es una balanza electrónica con la capacidad de pesar bobinas de papel de hasta 1000kg. El modelo de la misma es UMC 555 GEA y fue vendida por CAIR SRL.

Una vez estudiado la hoja de datos del modelo de la balanza, se visito al fabricante de dicha balanza, para confirmar las configuraciones apropiadas.

En la Ilustracion1 podemos observar la hoja de configuración entregada por el fabricante de balanzas electrónicas CAIR SRL, en donde allí detalla las configuraciones que deben aplicarse para la salida en puerto COM.

Ilustración

Configuración:

- 9600 bautios

- 7 bits de paridad impar

Conexiones Borneras:

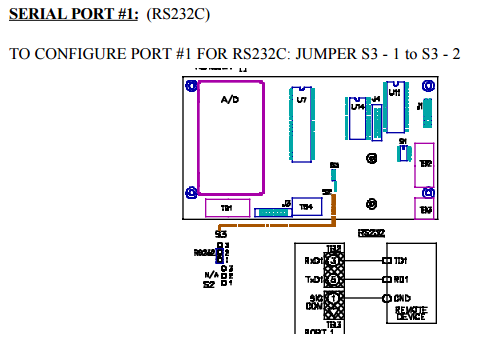
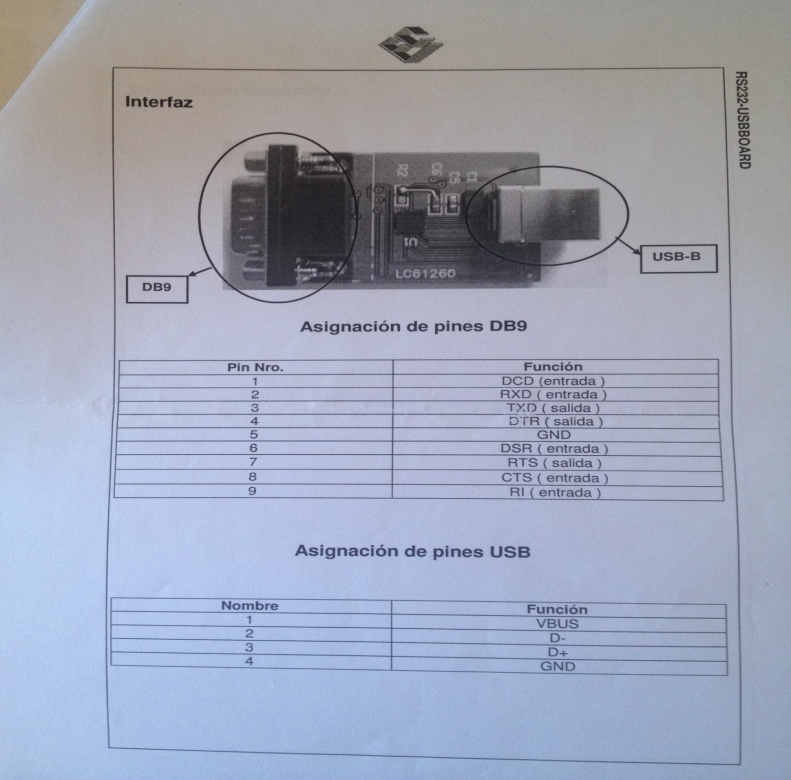
Como se puede observar en la ilustración 2, se ve a grandes razgos el circuito del cabezal digital que toma el peso de la balanza. Allí se indica cuales son las borneras de Tx, Rx y GND para enviar por RS232 (DB9) la salida del peso de la misma.

Ilustración   
Seccion 6 (Serial OutPut)

Para llevarlo a la PC, lo que se hizo fue comprar un adaptador de DB9 a usb.

A nivel programación lo que se debió hacer es incluir una librería para C#, System.IO.Ports y poder recibir el dato transmitido en el puerto COM conectado.

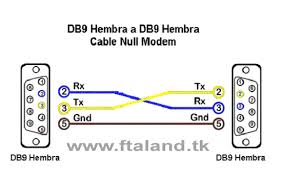
Adaptador DB9-USB



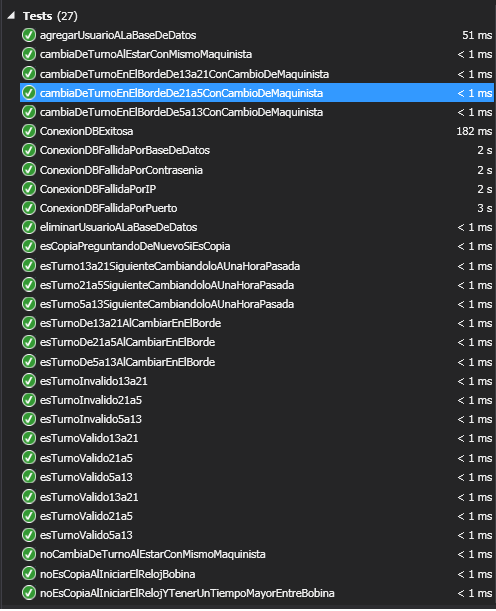
Ilustración

En la ilustración 3 se pueden observar cual fue el adaptador adquirido y cuales son los pines del Tx,Rx y GND para conectar a las borneras de la balanza.

Los pines en un DB9 y las conexiones que deben hacerse para los Tx, Rx y GND son las siguientes:

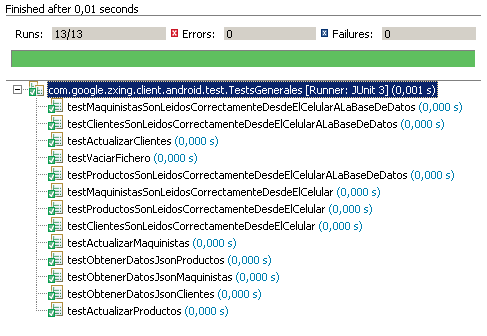


**Tests C#**

Como puede verse en la ilustración 4 se ha empezado realizando los primeros 27 tests principales respecto del modelo del proyecto.

Ilustración

Se hicieron modelos de tests en base a la conexión de la base de datos, a los turnos de los maquinistas, como deberían respetarse las entradas y salidas y en base a eso los tipos de copia según el horario en las impresiones de los rotulos.

**Tests Android**

Ilustración

Como puede verse en la ilustración 5 se implementaron tests básicos, aun con sus clases no implementadas sobre las futuras funcionalidades que tendrá la aplicación de android.

**Código QR**

****

Para la leer codigos QR me base en un proyecto open source llamado Zxing cuyo repositorio es el siguiente:

<https://github.com/zxing/zxing>

ZXing ("paso de cebra") es un multi-formato de biblioteca de procesamiento de imágenes de código de barras de código abierto 1D / 2D implementado en Java, con los puertos a otros idiomas.

**Formatos Soportados**

|  |  |  |
| --- | --- | --- |
| **1D product** | **1D industrial** | **2D** |
|  |  |  |
|  |
| UPC-A | Code 39 | QR Code |
| UPC-E | Code 93 | Data Matrix |
| EAN-8 | Code 128 | Aztec (beta) |
| EAN-13 | Codabar | PDF 417 (beta) |
|  | ITF |  |
|  | RSS-14 |  |
|  | RSS-Expanded |  |

Son los formatos de códigos que soporta el proyecto de Zxing, entre los cuales esta el QR code, que es el que iríamos a usar.

Lo que hay que hacer es incluir el proyecto que hace la lectura del código QR a nuestro proyecto java que procesa y realiza la conexión con la base de datos.

El código de Zxing se lo pondrá como librería, de esta forma se lo podrá reutilizar.