Treball de Fi de Grau

Grau en enginyeria en tecnologies industrials

DISSENY I CONSTRUCCIÓ D'UN INSTRUMENT PORTÀTIL DETECTOR DELS COMPONENTS RGB DE LLUM INCIDENT I ENREGISTRAMENT AL NÚVOL DELS COMPONENTS DETECTATS.

ANNEX

Autor: Carlos Álvaro García
Director: Rosa Rodriguez Montañés

Convocatòria: Febrer 2018

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Sumari

1. Programació PIC

1.1. Main.c

```
/*
 Author: Carlos Álvaro García
 */

#include <pic16f690.h>
#include <xc.h>
#define _XTAL_FREQ 4000000
#include  <stdio.h>
#include  <stdbool.h>
#include "uart.h"
#include "TCS34725.h"

// PINS I2C

#define SDA_PIN RB4     //
#define SCL_PIN RB6     //
#define SDA_DIR TRISB4
#define SCL_DIR TRISB6


#pragma config FOSC = INTRCIO   // Oscillator Selection bits (INTOSCIO
oscillator: I/O function on RA4/OSC2/CLKOUT pin, I/O function on
RA5/OSC1/CLKIN)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT
disabled and can be enabled by SWDTEN bit of the WDTCON register)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT
disabled)
#pragma config MCLRE = OFF      // MCLR Pin Function Select bit (MCLR pin
function is digital input, MCLR internally tied to VDD)
#pragma config CP = OFF         // Code Protection bit (Program memory
code protection is disabled)
#pragma config CPD = OFF        // Data Code Protection bit (Data memory
code protection is disabled)
#pragma config BOREN = OFF      // Brown-out Reset Selection bits (BOR
disabled)
#pragma config IESO = OFF       // Internal External Switchover bit
(Internal External Switchover mode is disabled)
#pragma config FCMEN = OFF      // Fail-Safe Clock Monitor Enabled bit
(Fail-Safe Clock Monitor is disabled)

//variables a usar m?s tard
char buf[16];
unsigned int cdat;
unsigned int id;
int t;
int LECTURA;



int main(void)
```

```
{
  //tots els ports es posen a 0 inicialment
  PORTA = 0;
  PORTB = 0;
  PORTC = 0;

  TRISA = 0b00000100; //RA2 com a entrada (pendent d'interrupcions)
  TRISB = 0b00000000; //tots els pins com a sortida
  TRISC = 0b00000000; // tots els pins com a sortida

  //com que es vol treballar digitalment, ANSEL i ANSELH a 0
  ANSEL = 0X00; // digital
  ANSELH = 0X00; // digital

  INTF = 0; // Reset del flag de les interrupcions
  INTE = 1; // Habilitaci? les interrupcions externes
  PEIE = 1; //Habilitaci? de interrupcions de perif?rics
  GIE = 1; // Habilitaci? interrupcions globals
  // Habilitaci? de els pull-up interns, per no tenir problemes amb
  // els possibles punts flotants quan es connecten les interrupcions
externes
  OPTION_REG = 0;

 //configuraci? BLUETOOTH
     TRISBbits.TRISB5=1;      // Habilitem el pin 5B com a entrada (RX)
     TRISBbits.TRISB7=0;       // Habilitem el pin 5B com a entrada (RX)
     UART_Init(9600);// inicialitzaci? de la comunicaci? UART amb el
Bluetooth



/*En cas d'identificar el sensor, el configurem*/
 if(initialize) {

      wireWriteDataByte(TCS34725_ENABLE, TCS34725_ENABLE_PON);
            __delay_ms(10);
             wireWriteDataByte(TCS34725_ENABLE,  0b00000011);
             __delay_ms(10);

//inicialitzaci? dels par?metres del sensor
wireWriteDataByte(TCS34725_ATIME, TCS34725_WT_240MS); // es defineix el
temps
//d'integraci? com 240ms
    __delay_ms(10);
     wireWriteDataByte(TCS34725_CONTROL, TCS34725_GAIN_4X); //s'afegeix
un
     //guany can?nic al valor captat pel sensor (valor per defecte)
    }

__delay_ms(100);
```

```
  while (1)
  {

    __delay_ms(1000);


  }
}


void interrupt isr()
{


int C_data_L, C_data_H;
int R_data_L, R_data_H;
int G_data_L, G_data_H;
int B_data_L, B_data_H;
  if (INTF) //si el flag de les interrupcions val 1
  {
      //es deshabiliten interrupcions moment?neament
    INTE = 0;
__delay_ms(20);

//es sol?liciten els valors dels registres corresponents a les
components RGBC
   C_data_L = wireReadDataByte(TCS34725_CDATAL);
   C_data_H = wireReadDataByte(TCS34725_CDATAH);
   R_data_L = wireReadDataByte(TCS34725_RDATAL);
   R_data_H = wireReadDataByte(TCS34725_RDATAH);
   G_data_L = wireReadDataByte(TCS34725_GDATAL);
   G_data_H = wireReadDataByte(TCS34725_GDATAH);
   B_data_L = wireReadDataByte(TCS34725_BDATAL);
   B_data_H = wireReadDataByte(TCS34725_BDATAH);
    //es deixa el microcontrolador en espera fins passat el temps
d'intregraci?
    __delay_ms(240);


  //s'escriu un nombre arbitrari al BL (a l'smarphone) com a 'hand-shake'
    UART_Write(45);
    //s'obt? el valor en 16 bits del registre de Clear
  cdat=C_data_L+(C_data_H << 8);

  //es transmet el missatge al BL, ?s a dir, al dispositiu Smartpone
  //el missatge s'escriu com a text, i els valors en 16 bits del colors,
// es transformen primerament en un String (una cadena de car?cters)
// a trav?s d'un buffer

    UART_Write_Text("Color:(");
 __delay_ms(3);
 sprintf(buf,"%d",cdat);
 UART_Write_Text(buf);
 __delay_ms(3);
 UART_Write_Text(",");
sprintf(buf,"%d",(R_data_L+(R_data_H << 8)));
 __delay_ms(3);
```

```
  UART_Write_Text(buf);
    __delay_ms(3);
 UART_Write_Text(",");
   __delay_ms(3);
 sprintf(buf,"%d",(G_data_L+(G_data_H << 8)));
  UART_Write_Text(buf);
    __delay_ms(3);
  UART_Write_Text(",");
    __delay_ms(3);
sprintf(buf,"%d",(B_data_L+(B_data_H << 8)));
  UART_Write_Text(buf);
    __delay_ms(3);
   UART_Write_Text(")");

   }
// el flag de les interrupcions torna a 0
    INTF = 0;

    //s'habiliten les interrupcions altra vegada
    INTE = 1;

}
```

1.2. TCS34725.h

```
*
 * File:   APDS9960.h
 * Author: Rosa Rodriguez & Carlos Álvaro
 * Description:

 * to be used for PIC16F690 as a color sensor.
 * This library interfaces the tcs to pic16f690 over I2C. The library
 * relies on the (I2C) library.
 */

#ifndef TCS34725_H
#define    TCS34725_H

#include <stdbool.h>
#include <stdint.h>

#define TCS34725_ADDRESS_W          0x52
#define TCS34725_ADDRESS_R          0x53



#define TCS34725_COMMAND_BIT      0x80

#define TCS34725_ENABLE           (0x00)
#define TCS34725_ENABLE_AIEN      (0x10)    /* RGBC Interrupt Enable */
#define TCS34725_ENABLE_WEN       (0x08)    /* Wait enable - Writing 1
activates the wait timer */
#define TCS34725_ENABLE_AEN       0x02    /* RGBC Enable - Writing 1
```

```
actives the ADC, 0 disables it */
#define TCS34725_ENABLE_PON       0x01    /* Power on - Writing 1
activates the internal oscillator, 0 disables it */
#define TCS34725_ATIME            0x01    /* Integration time */
#define TCS34725_WTIME            0x03    /* Wait time (if
TCS34725_ENABLE_WEN is asserted) */
#define TCS34725_WTIME_2_4MS      (0xFF)    /* WLONG0 = 2.4ms   WLONG1 =
0.029s */
#define TCS34725_WTIME_204MS      (0xAB)    /* WLONG0 = 204ms   WLONG1 =
2.45s  */
#define TCS34725_WTIME_614MS      (0x00)    /* WLONG0 = 614ms   WLONG1 =
7.4s   */

#define TCS34725_CONTROL          0x0F    /* Set the gain level for the
sensor */
#define TCS34725_ID               (0x12)    /* 0x44 = TCS34721/TCS34725,
0x4D = TCS34723/TCS34727 */
#define TCS34725_STATUS           (0x13)
#define TCS34725_STATUS_AINT      (0x10)    /* RGBC Clean channel
interrupt */
#define TCS34725_STATUS_AVALID    (0x01)    /* Indicates that the RGBC
channels have completed an integration cycle */


#define TCS34725_CDATAL           0x14    /* Clear channel data */
#define TCS34725_CDATAH           0x15
#define TCS34725_RDATAL           0x16    /* Red channel data */
#define TCS34725_RDATAH           0x17
#define TCS34725_GDATAL           0x18    /* Green channel data */
#define TCS34725_GDATAH           0x19
#define TCS34725_BDATAL           0x1A    /* Blue channel data */
#define TCS34725_BDATAH           0x1B

#define TCS34725_GAIN_4X           0x01
#define TCS34725_WT_154MS           0xC0
#define TCS34725_WT_240MS           0x9C

bool initialize();
    /* Raw I2C Commands */

    int wireWriteDataByte(unsigned char , unsigned char );
    unsigned char wireReadDataByte(unsigned char);

#endif
```

1.3. TCS34725.c

```
/*This library interfaces the APDS-9960 to SK40C over I2C. The library
 * relies on the I2C library.
 * This library is only for gesture sensor.
 * the main function that are used for gesture test are:
 *              1) initialize()
 *              2) enableGestureSensor()
 *              3) isGestureAvailable()
 *              4) readGesture()
*/

#include <pic16f690.h>

// PINS I2C
#define SDA_PIN RB4     //
#define SCL_PIN RB6     //
#define SDA_DIR TRISB4
#define SCL_DIR TRISB6

#include "i2c.h"
#include "TCS34725.h"
#include <stdint.h>
#include <stdbool.h>
#include <xc.h>

//#include "lcd.h"
#define _XTAL_FREQ 4000000

/*Es comprova si el sensor es communica correctament:
 Si el registre que emmagatzema la ID del sensor s diferent de 0x44 (el
valor
 que el fabricant determina, torna False)
 */
bool initialize()
{

    unsigned char id=0;

    id= wireReadDataByte(TCS34725 ID);
    if(id!= 0x44) {
        return false;
    }


return true;
}



/*Fuhci per escriure un byte determinat al registre determinat*/
int wireWriteDataByte(unsigned char reg, unsigned char val)
{

   i2c_start();
   i2c_out_byte(TCS34725_ADDRESS_W);
```

```
   i2c_ack();
   i2c_out_byte(reg | TCS34725_COMMAND_BIT);
   i2c_ack();
   i2c_out_byte(val);
   i2c_ack();
   i2c_stop();


    return 1;
}

/*Llegir un byte del registre que s'ordeni*/
 unsigned char wireReadDataByte(unsigned char reg)
{
    /* Indicate which register we want to read from */
   char val = 0;
   i2c_start();
   i2c_out_byte(TCS34725_ADDRESS_W);
   i2c_ack();
   i2c_out_byte(reg | TCS34725_COMMAND_BIT);
   i2c_ack();
   i2c_start();
   i2c_out_byte(TCS34725_ADDRESS_R);
   i2c_ack();
   val = i2c_in_byte();
   i2c_nack();
   i2c_stop();
   return(val);

}
```

2. AWS Lambda

2.1. SetColors.js

```
// Create the DynamoDB service object

    // Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region

exports.handler = (event, context, callback) => {
    // TODO implement


var ddb = new AWS.DynamoDB({});
var data1= new Date();
var data2= data1.toString();
var params = {


  Item: {
    'vermell' : {N: event.vermell},
    'blau': {N: event.blau},
    'verd' : {N: event.verd},
    'clear' : {N: event.clear},
    'date' : {S: data2} ,
  },
  TableName: 'colors'
};

// Call DynamoDB to add the item to the table
ddb.putItem(params, function(err, data) {
  if (err) {

     callback(null, err);
  } else {
    callback(null, 'succeess');
  }
});

};
```

2.2. getColors.js

```
'use strict';

var AWS = require('aws-sdk'),
   documentClient = new AWS.DynamoDB.DocumentClient();

exports.handler = function(event, context, callback){
```

```
    var params = {
        TableName : "colors"
    };
    documentClient.scan(params, function(err, data){
        if(err){
            callback(err, null);
        }else{
            callback(null, data.Items);
        }
    });
}
```

3. Android App

3.1. AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.mcuhq.coloretsApp">

    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />

    <permission
        android:name="android.permission.BLUETOOTH"
        android:label="BLUETOOTH" />
    <permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <permission android:name="android.permission.ACCESS_COARSE_LOCATION"
/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/col"
        android:label="colorets"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name="com.mcuhq.coloretsApp.MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

        <activity android:name="com.mcuhq.coloretsApp.biblio.Biblio">

        </activity>

    </application>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE"/>
</manifest>
```

3.2. MainActivity.java

```
ackage com.mcuhq.coloretsApp;

(…)


//aws
import com.amazonaws.mobile.client.AWSMobileClient;


import com.amazonaws.mobile.client.AWSStartupHandler;
import com.amazonaws.mobile.client.AWSStartupResult;

public class MainActivity extends AppCompatActivity {
private colors color;
private colors color1;


    // GUI Components
    private TextView mBluetoothStatus;
    private TextView mReadBuffer;
    private Button enviarBiblio;
    private Button mOffBtn;
    private Button mListPairedDevicesBtn;
    private Button mDiscoverBtn;
    private Button col;
    private BluetoothAdapter mBTAdapter;
    private Set<BluetoothDevice> mPairedDevices;
    private ArrayAdapter<String> mBTArrayAdapter;
    private ListView mDevicesListView;
    private CheckBox mLED1;



    //global file
 //   private GlobalFile gf;
private JSONUtilsFile js;


    private final String TAG = MainActivity.class.getSimpleName();
    private Handler mHandler; // Our main handler that will receive
callback notifications
    private ConnectedThread mConnectedThread; // bluetooth background
worker thread to send and receive data
    private BluetoothSocket mBTSocket = null; // bi-directional client-
to-client data path

    private static final UUID BTMODULEUUID = UUID.fromString("00001101-
0000-1000-8000-00805F9B34FB"); // "random" unique identifier


    // #defines for identifying shared types between calling functions
    private final static int REQUEST_ENABLE_BT = 1; // used to identify
adding bluetooth names
    private final static int MESSAGE_READ = 2; // used in bluetooth
handler to identify message update
```

```
    private final static int CONNECTING_STATUS = 3; // used in bluetooth
handler to identify message status


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        //aws
js= new JSONUtilsFile();

        AWSMobileClient.getInstance().initialize(this, new
AWSStartupHandler() {
            @Override
            public void onComplete(AWSStartupResult awsStartupResult) {
                Log.d("wow1", "AWSMobileClient is instantiated and you
are connected to AWS!");
            }
        }).execute();
GlobalFile.INSTANCE.apiGetCallWithPath("colors");
color= new colors();




        setContentView(R.layout.activity_main);
        mBluetoothStatus = (TextView)findViewById(R.id.bluetoothStatus);
        mReadBuffer = (TextView) findViewById(R.id.readBuffer);
       // mScanBtn = (Button)findViewById(R.id.scan);
        //mOffBtn = (Button)findViewById(R.id.off);
        mDiscoverBtn = (Button)findViewById(R.id.discover);
        mListPairedDevicesBtn = (Button)findViewById(R.id.PairedBtn);
       // mLED1 = (CheckBox)findViewById(R.id.checkboxLED1);
        enviarBiblio= (Button) findViewById(R.id.enviar1);
        col = (Button) findViewById((R.id.color));


        //intent butó biblio
        Button anarbiblio = (Button) findViewById(R.id.biblioc);
        anarbiblio.setOnClickListener(new View.OnClickListener() {
                                          @Override
                                          public void onClick(View v) {
                                              Intent i = new
Intent(MainActivity.this,  Biblio.class);
                                              startActivity(i);

                                          }
                                      });

        enviarBiblio.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

GlobalFile.INSTANCE.apiPutCallWithPath("/colors",js.fillCommand(color));

            }
        });
```

```
        mBTArrayAdapter = new
ArrayAdapter<String>(this,android.R.layout.simple_list_item_1);
        mBTAdapter = BluetoothAdapter.getDefaultAdapter(); // get a
handle on the bluetooth radio

        mDevicesListView = (ListView)findViewById(R.id.devicesListView);
        mDevicesListView.setAdapter(mBTArrayAdapter); // assign model to
view
        mDevicesListView.setOnItemClickListener(mDeviceClickListener);

        // Ask for location permission if not already allowed
        if(ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED)
            ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_COARSE_LOCATION}, 1);


        mHandler = new Handler(){
            public void handleMessage(android.os.Message msg){
                if(msg.what == MESSAGE_READ){
                    String readMessage = null;

                    try {

                        readMessage = new String((byte[]) msg.obj, "UTF-
8");

                        if(readMessage.charAt(0)=='-')
                        {
                            if(readMessage.charAt(7)=='(')
                            {
                                if(readMessage.indexOf(')')>0) {
                                    Log.d("wow1", "ok");
                                    color.getcols(readMessage);

                                    col.setText("color: ("+
Integer.toString(color.r)+ ","+ Integer.toString(color.g)
+","+Integer.toString(color.b)+")");

col.setBackgroundColor(Color.argb(255, color.r, color.g, color.b));

                                }
                            }

                        }


                    } catch (UnsupportedEncodingException e) {
                        e.printStackTrace();
                    }
```

```
                }

                if(msg.what == CONNECTING_STATUS){
                    if(msg.arg1 == 1)
                        mBluetoothStatus.setText("Connected to Device: "
+ (String)(msg.obj));
                    else
                        mBluetoothStatus.setText("Connection Failed");
                }
            }
        };

        if (mBTArrayAdapter == null) {
            // Device does not support Bluetooth
            mBluetoothStatus.setText("Status: Bluetooth not found");
            Toast.makeText(getApplicationContext(),"Bluetooth device not
found!",Toast.LENGTH_SHORT).show();
        }
        else {



/*
            mScanBtn.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    bluetoothOn(v);
                }
            });
 turn of BT
            mOffBtn.setOnClickListener(new View.OnClickListener(){
                @Override
                public void onClick(View v){
                    bluetoothOff(v);
                }
            });
*/
            mListPairedDevicesBtn.setOnClickListener(new
View.OnClickListener() {
                @Override
                public void onClick(View v){
                    listPairedDevices(v);
                }
            });

            mDiscoverBtn.setOnClickListener(new View.OnClickListener(){
                @Override
                public void onClick(View v){
                    discover(v);
                }
            });
        }
    }

    private void bluetoothOn(View view){
        if (!mBTAdapter.isEnabled()) {
```

```
            Intent enableBtIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
            mBluetoothStatus.setText("Bluetooth enabled");
            Toast.makeText(getApplicationContext(),"Bluetooth turned
on",Toast.LENGTH_SHORT).show();

        }
        else{
            Toast.makeText(getApplicationContext(),"Bluetooth is already
on", Toast.LENGTH_SHORT).show();
        }
    }

    // Enter here after user selects "yes" or "no" to enabling radio
    @Override
    protected void onActivityResult(int requestCode, int resultCode,
Intent Data){
        // Check which request we're responding to
        if (requestCode == REQUEST_ENABLE_BT) {
            // Make sure the request was successful
            if (resultCode == RESULT_OK) {
                // The user picked a contact.
                // The Intent's data Uri identifies which contact was
selected.
                mBluetoothStatus.setText("Enabled");
            }
            else
                mBluetoothStatus.setText("Disabled");
        }
    }

    private void bluetoothOff(View view){
        mBTAdapter.disable(); // turn off
        mBluetoothStatus.setText("Bluetooth disabled");
        Toast.makeText(getApplicationContext(),"Bluetooth turned Off",
Toast.LENGTH_SHORT).show();
    }

    private void discover(View view){
        // Check if the device is already discovering
        if(mBTAdapter.isDiscovering()){
            mBTAdapter.cancelDiscovery();
            Toast.makeText(getApplicationContext(),"Discovery
stopped",Toast.LENGTH_SHORT).show();
        }
        else{
            if(mBTAdapter.isEnabled()) {
                mBTArrayAdapter.clear(); // clear items
                mBTAdapter.startDiscovery();
                Toast.makeText(getApplicationContext(), "Discovery
started", Toast.LENGTH_SHORT).show();
                registerReceiver(blReceiver, new
IntentFilter(BluetoothDevice.ACTION_FOUND));
            }
            else{
                Toast.makeText(getApplicationContext(), "Bluetooth not
on", Toast.LENGTH_SHORT).show();
```

```
            }
        }
    }

    final BroadcastReceiver blReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            String action = intent.getAction();
            if(BluetoothDevice.ACTION_FOUND.equals(action)){
                BluetoothDevice device =
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
                // add the name to the list
                mBTArrayAdapter.add(device.getName() + "\n" +
device.getAddress());
                mBTArrayAdapter.notifyDataSetChanged();
            }
        }
    };

    private void listPairedDevices(View view){
        mPairedDevices = mBTAdapter.getBondedDevices();
        if(mBTAdapter.isEnabled()) {
            // put it's one to the adapter
            for (BluetoothDevice device : mPairedDevices)
                mBTArrayAdapter.add(device.getName() + "\n" +
device.getAddress());

            Toast.makeText(getApplicationContext(), "Show Paired
Devices", Toast.LENGTH_SHORT).show();
        }
        else
            Toast.makeText(getApplicationContext(), "Bluetooth not on",
Toast.LENGTH_SHORT).show();
    }

    private AdapterView.OnItemClickListener mDeviceClickListener = new
AdapterView.OnItemClickListener() {
        public void onItemClick(AdapterView<?> av, View v, int arg2, long
arg3) {

            if(!mBTAdapter.isEnabled()) {
                Toast.makeText(getBaseContext(), "Bluetooth not on",
Toast.LENGTH_SHORT).show();
                return;
            }

            mBluetoothStatus.setText("Connecting...");
            // Get the device MAC address, which is the last 17 chars in
the View
            String info = ((TextView) v).getText().toString();
            final String address = info.substring(info.length() - 17);
            final String name = info.substring(0,info.length() - 17);

            // Spawn a new thread to avoid blocking the GUI one
            new Thread()
            {
                public void run() {
                    boolean fail = false;
```

```
                    BluetoothDevice device =
mBTAdapter.getRemoteDevice(address);

                    try {
                        mBTSocket = createBluetoothSocket(device);
                    } catch (IOException e) {
                        fail = true;
                        Toast.makeText(getBaseContext(), "Socket creation
failed", Toast.LENGTH_SHORT).show();
                    }
                    // Establish the Bluetooth socket connection.
                    try {
                        mBTSocket.connect();
                    } catch (IOException e) {
                        try {
                            fail = true;
                            mBTSocket.close();
                            mHandler.obtainMessage(CONNECTING_STATUS, -1,
-1)
                                    .sendToTarget();
                        } catch (IOException e2) {
                            //insert code to deal with this
                            Toast.makeText(getBaseContext(), "Socket
creation failed", Toast.LENGTH_SHORT).show();
                        }
                    }
                    if(fail == false) {
                        mConnectedThread = new
ConnectedThread(mBTSocket);
                        mConnectedThread.start();

                        mHandler.obtainMessage(CONNECTING_STATUS, 1, -1,
name)
                                .sendToTarget();
                    }
                }
            }.start();
        }
    };

    private BluetoothSocket createBluetoothSocket(BluetoothDevice device)
throws IOException {
        try {
            final Method m =
device.getClass().getMethod("createInsecureRfcommSocketToServiceRecord",
UUID.class);
            return (BluetoothSocket) m.invoke(device, BTMODULEUUID);
        } catch (Exception e) {
            Log.e(TAG, "Could not create Insecure RFComm Connection",e);
        }
        return  device.createRfcommSocketToServiceRecord(BTMODULEUUID);
    }

    private class ConnectedThread extends Thread {
        private final BluetoothSocket mmSocket;
        private final InputStream mmInStream;
        private final OutputStream mmOutStream;
```

```
        public ConnectedThread(BluetoothSocket socket) {
            mmSocket = socket;
            InputStream tmpIn = null;
            OutputStream tmpOut = null;

            // Get the input and output streams, using temp objects
because
            // member streams are final
            try {
                tmpIn = socket.getInputStream();
                tmpOut = socket.getOutputStream();
            } catch (IOException e) { }

            mmInStream = tmpIn;
            mmOutStream = tmpOut;
        }

        public void run() {
            byte[] buffer = new byte[1024];  // buffer store for the
stream
            int bytes; // bytes returned from read()
            // Keep listening to the InputStream until an exception
occurs
            while (true) {
                try {
                    // Read from the InputStream
                    bytes = mmInStream.available();
                    if(bytes != 0) {
                        buffer = new byte[1024];
                        SystemClock.sleep(100); //pause and wait for rest
of data. Adjust this depending on your sending speed.
                        bytes = mmInStream.available(); // how many bytes
are ready to be read?
                        bytes = mmInStream.read(buffer, 0, bytes); //
record how many bytes we actually read
                        mHandler.obtainMessage(MESSAGE_READ, bytes, -1,
buffer)
                                .sendToTarget(); // Send the obtained
bytes to the UI activity
                    }
                } catch (IOException e) {
                    e.printStackTrace();

                    break;
                }
            }
        }

        /* Call this from the main activity to send data to the remote
device */
        public void write(String input) {
            byte[] bytes = input.getBytes();           //converts entered
String into bytes
            try {
                mmOutStream.write(bytes);
            } catch (IOException e) { }
        }
```

```
        /* Call this from the main activity to shutdown the connection */
        public void cancel() {
            try {
                mmSocket.close();
            } catch (IOException e) { }
        }



        //aws 2




    }
```

3.3. GlobalFile.kt

```
package com.mcuhq.coloretsApp

(…)

/**
 * Created by carlos on 27/11/2018.
 */
//objecte singleton que crida internet (el núvol) i allotja en si mateix
el array de colors rebuts, a través i facilitant-se amb el
colormodelCreator

 object GlobalFile {

    public var list1=ArrayList<colormODEL>()
private var apiClient: ColorsApiClient? = null
   public fun apiGetCallWithPath(path: String) {
       var modelcreator: ColorModelCreator?=null;

        apiClient = ApiClientFactory()

.credentialsProvider(AWSMobileClient.getInstance().credentialsProvider)
                .build(ColorsApiClient::class.java)
       modelcreator= ColorModelCreator();

        val parameters = mapOf("lang" to "en_US")
        val headers = mapOf("Content-Type" to "application/json")

        val request = ApiRequest(ColorsApiClient::class.java.simpleName)
                .withPath(path)
                .withHttpMethod(HttpMethodName.GET)
                .withHeaders(headers)
                .withParameters(parameters)

        thread(start = true) {
            try {
                Log.e("wow1", "Accedint GET API path $path")
```

```
                val response = apiClient?.execute(request)
                val responseContentStream = response?.getContent()
                if (responseContentStream != null) {
                    val responseData =
IOUtils.toString(responseContentStream)
                   list1 =modelcreator.createColors(responseData);
                    Log.e("wow11", list1.toString());

                }

            } catch (ex: Exception) {
                Log.e("wow1", "Error invoking API")
                Log.e("Api error", ex.toString())
            }
        }

    }


    fun apiPutCallWithPath(path: String, body: String) {

        Log.e("wow15", "1time")
        apiClient = ApiClientFactory()

.credentialsProvider(AWSMobileClient.getInstance().credentialsProvider)
                .build(ColorsApiClient::class.java)
        val parameters = mapOf("lang" to "en_US")
        val headers = mapOf("Content-Type" to "application/json")
        val contentLenght = body.toByteArray().size

        val contentToUse = contentLenght.toString()

        Log.e("Body", body)
        Log.e("Body Lenght byte", contentToUse)
        Log.e("wow1", "E145I")
        val request = ApiRequest(ColorsApiClient::class.java.simpleName)
                .withPath(path)
                .withHttpMethod(HttpMethodName.PUT)
                .withHeaders(headers)
                .addHeader("Content-Length", contentLenght.toString())
                .withParameters(parameters)
                .withBody(body)

        thread(start = true) {
            try {
                Log.e("wow2", "Accedint PUT API path $path")
                val response = apiClient?.execute(request)
                val responseContentInt = response?.statusCode
                Log.e("Result: ", responseContentInt.toString())
            } catch (ex: Exception) {
                Log.e("wow2", "Error invoking API")
                Log.e("Api error", ex.toString())
            }
        }


    }
```

```
}
```

3.4. Colors.kt

```
package com.mcuhq.coloretsApp;

import android.util.Log;

/**
 * Created by trank on 10/11/2018.
 */


/*
Classe-objecte que transforma l'string rebut de la comunicació Bluetooth
i que obté les components RGBC, guardant-les en variables. (Kotlin)
*/
public class colors {
public int b,r,g,c;
    public float c1,r1,g1,b1;
    public void getcols(String str) {
String sub;
int indxofp;


try {
    //tractament strings
    indxofp = str.indexOf(')');
    String[] strl;
    sub = str.substring(8, indxofp);
    strl = sub.split(",");

    //si té 4 components (r,g,b,c)
    if(strl.length==4) {



        c1 = Float.parseFloat(strl[0]);
        if (c1 < 0) {
            c = Math.round(c1);
            c1 = c & 0x0000ffffL;

        }

//obtenció valors en 8 bits
        r1 = Float.parseFloat(strl[1]) / c1 * 255;
        g1 = Float.parseFloat(strl[2]) / c1 * 255;
        b1 = Float.parseFloat(strl[3]) / c1 * 255;
        b = Math.round(b1);
        r = Math.round(r1);
        g = Math.round(g1);


        Log.d("Log", Integer.toString(r) + "," + Integer.toString(g) +
"," + Integer.toString(b));
```

```
    }

    else {

        try {

            if(r!=0 & g!=0 & b!=0)

            {
             r=r; g=g; b=b;
            }

        }

        catch (Exception e) {
           r=0; g=0; b=0;

        }
        Log.d("log", "Bad measure");
    }

}

catch (Exception e) {
    r=0; g=0; b=0;

}


    }

}
```

3.5 Biblio.kt

```
package com.mcuhq.coloretsApp.biblio


import com.mcuhq.coloretsApp.R
/**
 * Created by trank on 26/11/2018.
 */
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import kotlinx.android.synthetic.main.activity_main.*
import kotlinx.android.synthetic.main.biblio.*
/*


Activitat pantalla de la biblioteca. Té associat l'arxiu biblio.xml
(Programat amb kotlin)
 */
class Biblio : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
```

```
        setContentView(R.layout.biblio)

        val fragmentAdapter = PageAdapter(supportFragmentManager)
        viewpager_main.adapter = fragmentAdapter

        tabs_main.setupWithViewPager(viewpager_main)
    }
}
```

3.6. FirstFragment .kt

```
package com.mcuhq.coloretsApp.biblio

(…)


import
com.mcuhq.coloretsApp.amazonaws.mobile.api.id8a6y7xdey6.ColorsApiClient;

import com.mcuhq.coloretsApp.GlobalFile


import kotlinx.android.synthetic.main.fragment_first.view.*



/**
 * A simple [Fragment] subclass.
 */

//fragment que allotja una listView on aniran els elements visuals,
adaptats pel colorAdapter. Sols afegeix els items que corresponguin
class FirstFragment : Fragment() {
    private var apiClient: ColorsApiClient? = null

    private var listview: ListView? = null
    private lateinit var viewOfLayout: View

private var arrayl: ArrayList<colormODEL>?=null;

    companion object {
        private val TAG: String = this::class.java.name
    }

    override fun onCreateView(inflater: LayoutInflater?, container:
ViewGroup?, savedInstanceState: Bundle?): View? {
        // Inflate the layout for this fragment
arrayl= ArrayList<colormODEL>();

        //agafem el view (element visual superior)
        val viewOfLayout = inflater!!.inflate(R.layout.fragment_first,
container, false)


        // Construct the data source
```

```
        val arrayOfColors = ArrayList<colormODEL>()


        //el list1 ha d'estar creat previament
        arrayOfColors!= GlobalFile.list1;

// Create the adapter to convert the array to views
        val adapterC = ColorAdapter(getActivity().getBaseContext(),
arrayOfColors)
// Attach the adapter to a ListView

        viewOfLayout.colorsList.adapter = adapterC




        for(i in GlobalFile.list1.orEmpty()) {
            if(i.getr()>= i.getb() && i.getr()>=i.getg()) {
                // l'adaptr afegeix items

            adapterC.add(i);}

        }


        return viewOfLayout;

    }

}
```

3.7. ColorModelCreator.kt

```
package com.mcuhq.coloretsApp.biblio

/**
 * Created by trank on 27/11/2018.
 */
import android.util.Log
import org.json.JSONArray

class ColorModelCreator {



    //classe intermedia entre rebre el objecte del núvol fins a salvar,
en un un format array, els colors
/*
Classe que transforma l'objecte JSON rebut del núvol (que conté tots els
elements de la base de dades) en un Array de colormODELS. (kotlin)
 */
    public fun createColors(data: String): ArrayList<colormODEL> {

        val list1= ArrayList<colormODEL>();
        val JSONobjects = JSONArray(data)
```

```
        val JSONobjectsLenght = JSONobjects.length()
        Log.e("APIJSON: ", data)
        Log.e("APIJSON Lenght", JSONobjectsLenght.toString())


        for (jsonObj in 0 until JSONobjectsLenght) {
val colorm: colormODEL;
val r: Int;
            val g: Int;
            val c: Int;
            val b: Int;


r=JSONobjects.getJSONObject(jsonObj)["vermell"].toString().toInt()


g=JSONobjects.getJSONObject(jsonObj)["verd"].toString().toInt();

b=JSONobjects.getJSONObject(jsonObj)["blau"].toString().toInt();

c=JSONobjects.getJSONObject(jsonObj)["clear"].toString().toInt();

            colorm= colormODEL(r, g, b, c);


             list1.add(colorm);

        }
    return list1
    }
}
```

3.8. ColorAdapter.java

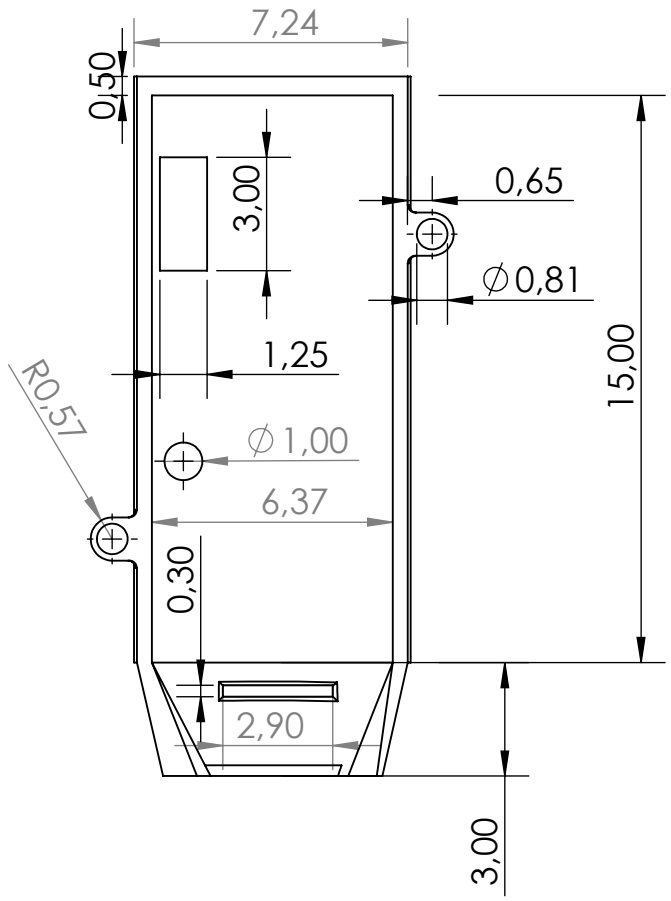
```
package com.mcuhq.coloretsApp.biblio;

(…)

/**
 * Created by trank on 27/11/2018.
 */

//classe que es dedica a generar un element visual a partir d'una llista
de models de color colorModel

public class ColorAdapter extends ArrayAdapter<colormODEL> {
    public ColorAdapter(Context context, ArrayList<colormODEL> users) {
        super(context, 0, users);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent)
{
        // Get the data item for this position
        colormODEL colormodel = getItem(position);
        // Check if an existing view is being reused, otherwise inflate
```

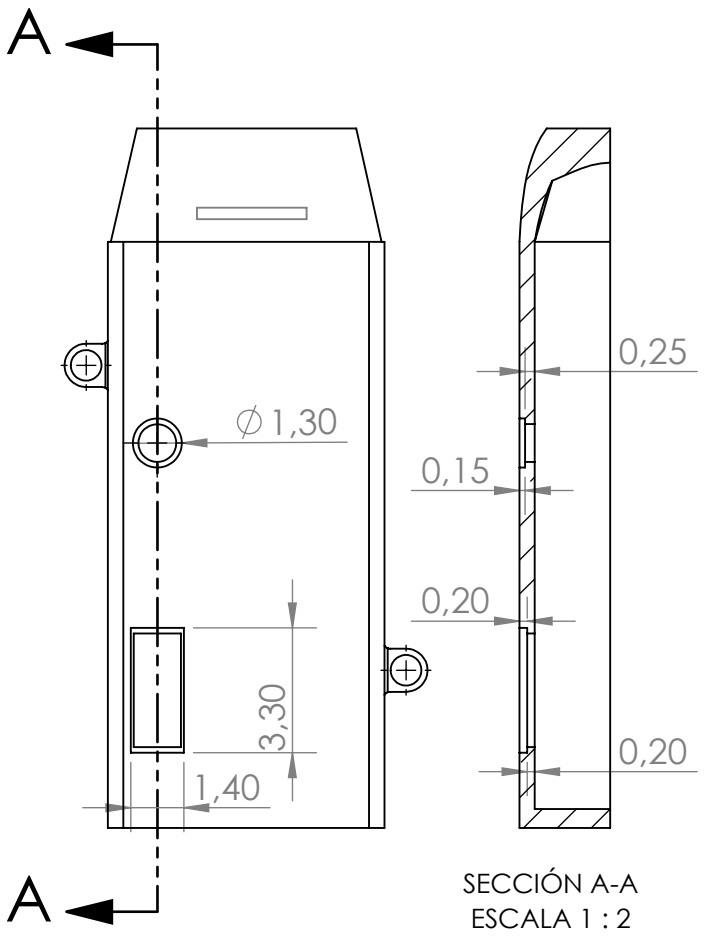
```
the view
        if (convertView == null) {
            convertView =
LayoutInflater.from(getContext()).inflate(R.layout.item_color, parent,
false);
        }
        // Lookup view for data population
        TextView rc = (TextView) convertView.findViewById(R.id.rc);
        TextView gc = (TextView) convertView.findViewById(R.id.gc);
        TextView bc = (TextView) convertView.findViewById(R.id.bc);
        LinearLayout ly= (LinearLayout)
convertView.findViewById(R.id.line);

ly.setBackgroundColor(Color.argb(255,colormodel.getr(),colormodel.getg(),
colormodel.getb()));
        // Populate the data into the template view using the data object
       rc.setText(Integer.toString(colormodel.getr()) + ",");
        gc.setText(Integer.toString(colormodel.getg())+ ",");
        bc.setText(Integer.toString(colormodel.getb())+ ")");

        // Return the complet
        // ed view to render on screen
        return convertView;    }}
```

3.9. ColormODEL.java

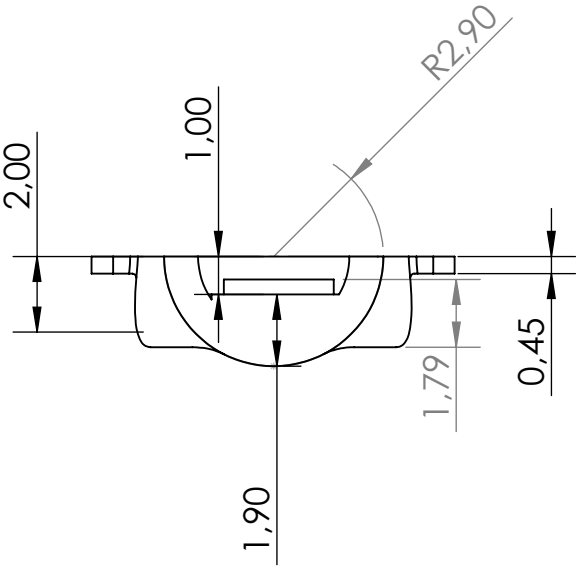
```
package com.mcuhq.coloretsApp.biblio;
//simple classe que allotja variables del color
    /*
    Classe que es dedica a generar elements visuals dinàmicament a
partir d'un Array d'objectes
    colormODEL, poblant així els ListView de la biblioteca. Cada
element visual generat té associat l'arxiu item_color.xml (programat
amb java)
     */
public class colormODEL  {
    private int r, g, b, c;
    public colormODEL(int r1, int g1, int b1, int c1) {
      r=r1; g=g1; b=b1; c=c1; }
    public int getr() {        return r;    }
    public int getb() {        return b;}
    public int getg() {        return g;     }
    public int getc() {        return c;     }}
```

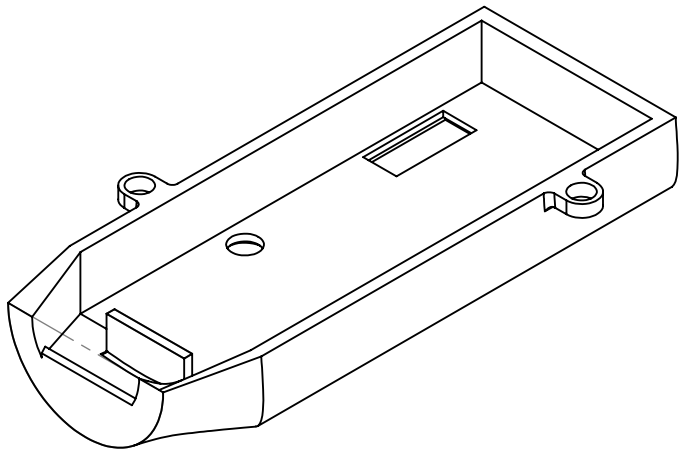
4. Plànol



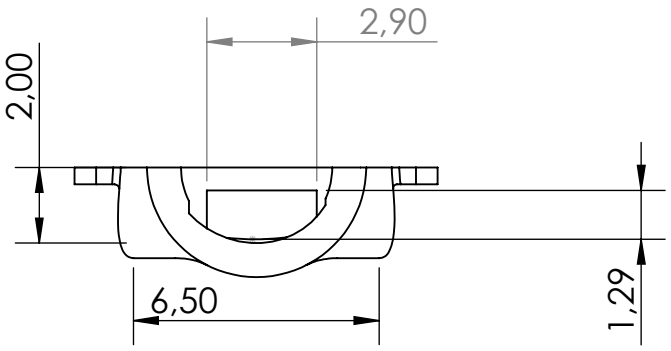


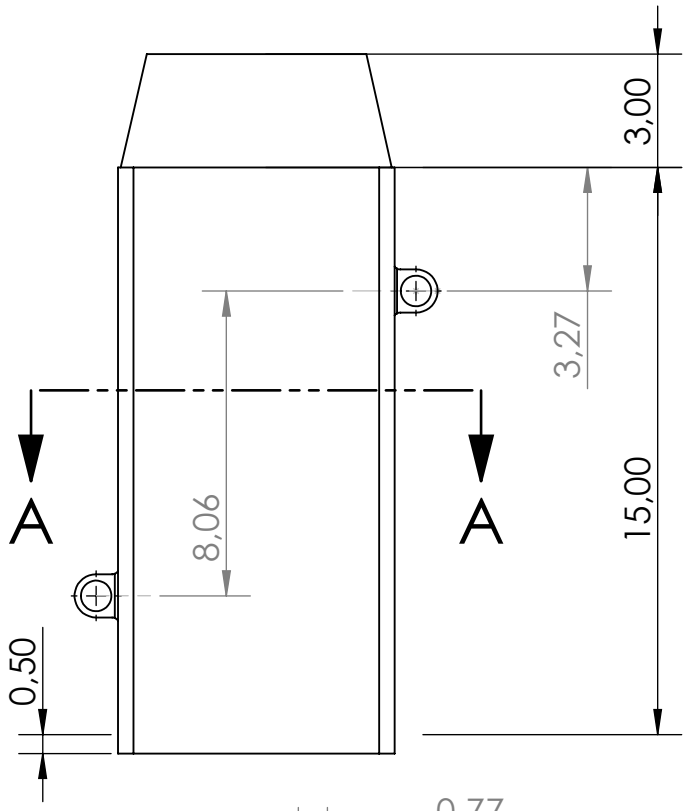
SECCIÓN A-A
ESCALA 1 : 2

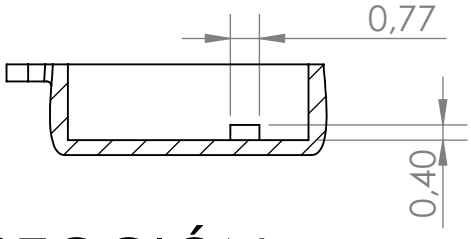


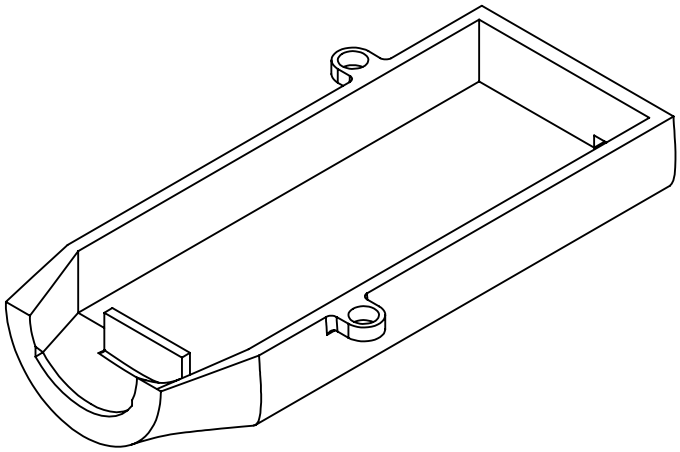


SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:	REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA	REVISIÓN	
APROB.			TÍTULO: Carcassa Superior		
FABR.					
CALID.		MATERIAL: FDA	N.º DE DIBUJO		A4
		PESO:	ESCALA1:2	HOJA 1 DE 2	









SECCIÓN A-A

ESCALA 1 : 2

SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:	REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA / REVISIÓN

	NOMBRE	FIRMA	FECHA
DIBUJ.			
VERIF.			
APROB.			
FABR.			
CALID.			

TÍTULO:

Carcassa inferior

MATERIAL: PLA	N.º DE DIBUJO	A4
PESO:	ESCALA:2	HOJA 2 DE 2