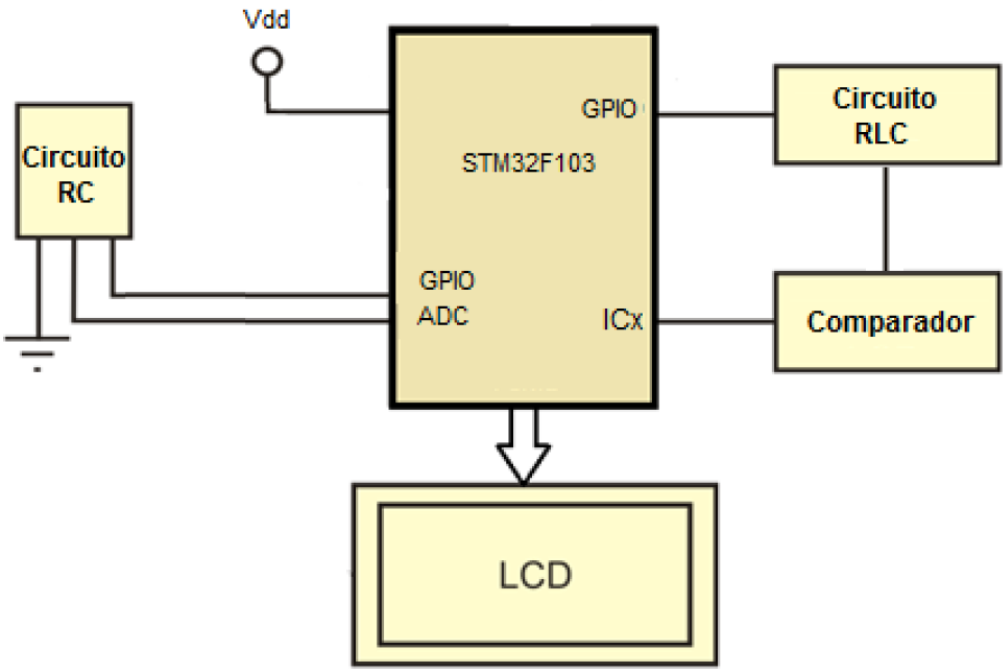
## Parte 1: Diseño del firmware, Simulación y Depuración

*Desarrollar el firmware del MCU para implementar la medición de capacidad e inductancia como se muestra la figura.*
*Los componentes a medir (incógnitas) se encuentran en los circuitos pasivos RC y RLC. En particular en el circuito RC el valor a determinar será la capacidad C y en el circuito RLC el valor a medir será la inductancia L. El sistema deberá mostrar los valores de capacidad e inductancia medidos en el display LCD cada 0.5 seg.*




*Investigar un método de medida y realizar la simulación con Proteus utilizando un esquema dedicado a dicha simulación (no tiene que utilizar el circuito esquemático completo con el cual diseñará el PCB).*
*Realizar en el informe una descripción detallada del firmware, de la modularización, de la forma en que adquiere, procesa y presenta los datos en pantalla. Mostrar en el informe como validó su solución propuesta.*

# Interpretación

Se desarrollará un sistema capaz de medir inductancia y capacitancia de un circuito RLC y un circuito RC respectivamente, una vez calculados dichos valores se imprimirán en un display LCD y se actualizarán cada 0,5 s. El cálculo de los valores se realizará de manera secuencial, es decir, se medirá un circuito y después el otro y posteriormente se imprimirán.

Los principales componente s para la realización del sistema son los siguientes:

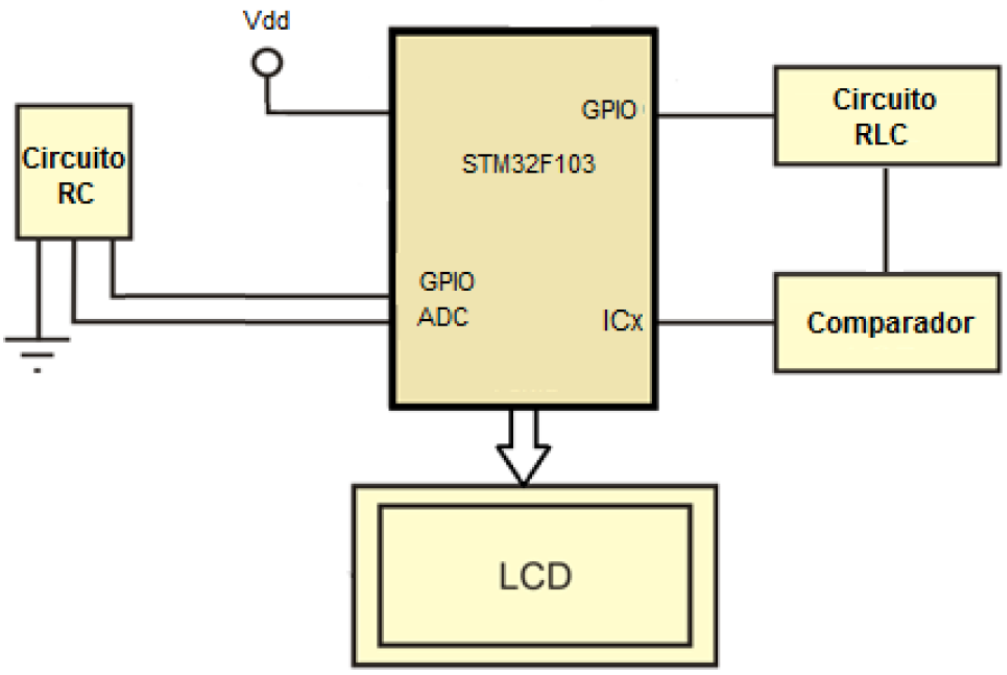
-MCU: STM32F103C6

-Display LCD:

-Comparador analogico: LM393N

-resistencias, capacitores e inductores.

Para el conexionado se seguirá el esquema mostrado a continuación:



# Propuesta e implementación

Medición de inductancia en circuito RLC:

Para la medición de la inductancia se utilizará alguna de las entradas de input capture disponibles en nuestro MCU, el principio de funcionamiento se basa en medir la frecuencia de oscilación del circuito RLC durante la etapa transitoria que se genera al aplicar un pulso de tensión continua.

Para poder calcular esta frecuencia se tomarán dos flancos consecutivos de la señal proveniente del circuito RLC, cabe aclarar que esta señal tiene una forma senoidal por lo que hara falta un comparador analogico para poder transformar esta señal analogica en una digital que pueda ser input de nuestro MCU, en este caso utilizaremos el LM339(equivalente al LM393N). Una vez tomados los valores del contador del TIMER en los dos instantes de flanco de la señal, se calculará la diferencia entre ambos y se dividirá por la frecuencia del TIMER, para poder calcular la frecuencia, el calculo seria el siguiente:

$$t0 \rightarrow primer\ instante,\ t \rightarrow segundo\ instante$$

$$f_{RLC} = \frac{(t - t0)}{f_{timer}}$$

Una vez obtenida la frecuencia del circuito podemos calcular el valor de la inductancia a partir de la siguiente expresión:

$$f_{RLC} = \frac{1}{2\pi\sqrt{LC}}$$

Al ser C un valor conocido podemos calcular el valor de la inductancia como:

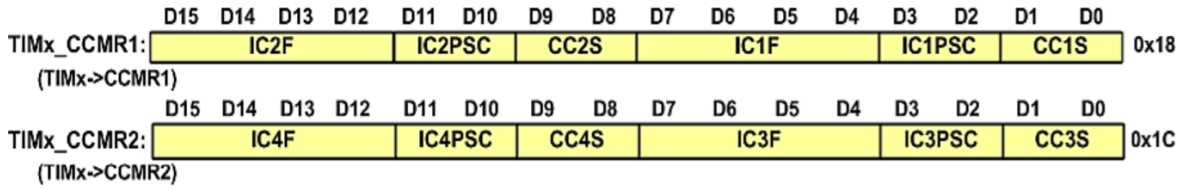
$$L = \frac{1}{(f_{RLC}2\pi)^2 C}$$

Aclaración: las expresiones utilizadas se obtienen de resolver las ecuaciones diferenciales que representan el circuito RLC, es este informe no se muestran ya que no forma parte del enfoque del mismo.

## Pseudocódigo

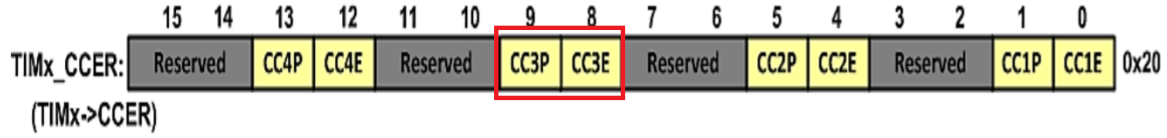
```
medir_L(){
        //energizo el circuito
        Contador_timer =0;
        whlie(no llegue un flanco){}
        t0 = contador_timer;
        whlie(no llegue un flanco){}
        t = contador_timer;
        //desenergizo el circuito
        T = (t1-t0)/(frecuencia_timer);
        F = 1/T;
        L = 1/ ((F*2*PI) * (F*2*PI) * (C));
}
```

Para poder realizar la medición se inicializa el timer de la siguiente forma:
En modo input capture con el canal CH3 (PA2)




| CC3S | Mode |
|---|---|
| 00 | Output Compare mode |
| 01 | Input from TIMx_CH3 |
| 10 | Input from TIMx_CH4 |
| 11 | Input from TRC |

Y la captura en rising edge (flanco de subida)




CC3P = 0
CC3E = 1

Una vez realizado el cálculo se imprimirá en el display LCD.
Impresión en LCD:

Para imprimir en el LCD se implementó una función que recibe como parámetro el valor y unidad a medir, así como también la posición en la que se quiere imprimir.
La función separa el número medido en sus dígitos y lo envía a imprimir uno por uno y en orden al LCD.

## Pseudocodigo

```
void imprimir(dato, x, y, unidad){
  int aux,cont=0;
  unsigned char vec[16];
  lcd_gotoXY(x,y);
  while (dato > 0){
    aux = dato mod 10;
    dato=dato / 10;
    vec[cont++] = (aux + '0');
  }
  while(cont>0){
    lcd_sendData(vec[--cont]);
  }
  lcd_string(unidad,2);
}
```

## Medición de capacitancia en circuito RC:

Para la medición de la capacitancia se utilizará una entrada analógica perteneciente al ADC, el principio de funcionamiento se basa en medir el tiempo que tarda el circuito en alcanzar una tensión igual al 63% de la carga máxima. Como en nuestro caso la carga máxima la brinda el microcontrolador esta es de 3.3V. Entonces el 63% de la misma será 2.079V. Una vez calculado este tiempo este va a ser el valor de $\tau$ y como conocemos el valor de la resistencia y el $\tau$ para este circuito es conocido ($\tau$=R*C, esto sale de resolver la ecuación diferencial) Podemos despejar de esta ecuación el valor de la capacitancia.

Para poder calcular esta tensión el ADC contiene 11 bits para muestrear la tensión, por lo tanto si 3.3V es el máximo de tensión, el valor del adc será 4095 (pero porque el último tiene un error de 3/2), entonces con una regla de tres sé que para conseguir 2.079 de precisión necesito que el adc llegue a 2580. Para poder saber el tiempo que pasó hasta llegar a $\tau$ usaremos un timer como contador. Gracias a este podremos despejar el valor de la capacitancia de la siguiente manera:

$$Capacitancia = \frac{\tau}{frecuenciaDelMicrocontrolador * Resistencia}$$

Para configurar el ADC, (en nuestro caso utilizamos el ADC1) debemos primero activar el clock del adc, después configurar el pin A1 como salida analogica, es decir poner un 0 en el pin 1 del GPIOA->CRL, luego hay que poner en 1 el bit ADON del registro ADC1->CR2 para prender el ADC, después se selecciona la frecuencia con la que vamos a muestrear, en nuestro caso elegimos la máxima frecuencia por lo tanto ponemos un 001 en el registro ADC1->SMPR2, una vez hecho esto se debe poner un delay de un microsegundo, en nuestro caso pusimos un delay más grande para asegurarnos que el circuito inicie descargado.

Para medir la capacitancia primero se debe activar el timer para que comience a contar colocando un 1 en el registro TIM2->CR1 y se debe resetear el valor del registro TIM2->CNT, luego se debe seleccionar el canal con el que vamos a muestrear en este caso

como solo estamos utilizando un ADC colocamos un 1 en el registro ADC1->SQR3 seguidamente comenzamos la conversión escribiendo un 1 en ADC1->CR2. Nuestra implementación funciona con polling, por lo tanto nos quedaremos esperando a que se escriba un 1 el flag EOC del registro ADC1->SR, por último leemos el valor del registro ADC1->DR que es el valor del dato.
Este proceso se repetirá hasta que llegue el valor medido llegue a 2580 significa que el contador del timer llegó al valor de $\tau$ entonces con ese dato calculo el valor de la capacitancia de la forma anteriormente explicada.

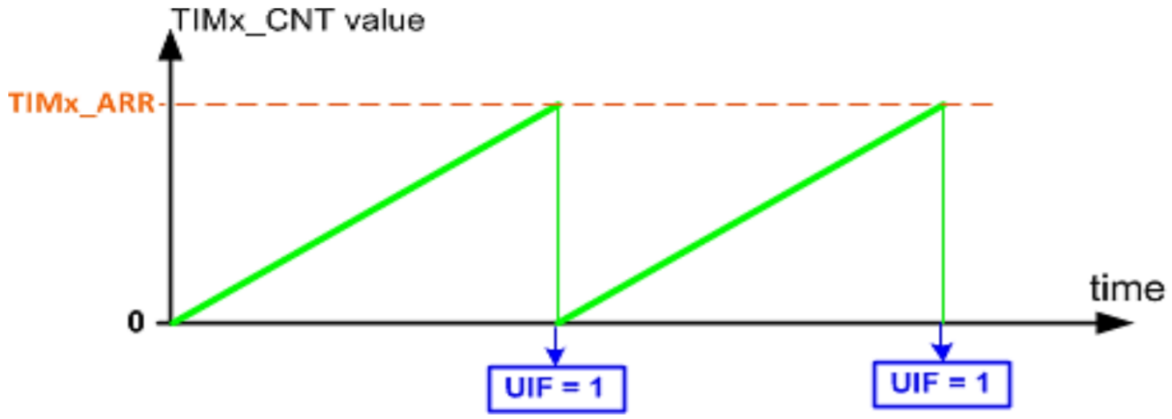
## Pseudocódigo

```
inicializar_RC{
  //activar clock para el ADC y el Timer
  //colocar Pin A0 como entrada analogica
  //colocar PIn B0 como salida y dejarla en cero para descargar el circuito
  //configurar el timer para contar
  //activar el adc
  //configurar
  //delay necesario para el adc y para asegurarnos que el circuito se encuentra inicialmente descargado

}


medir_RC(){
        //inicializar el contador
        //activar el contador
        //activar el pin B0
        while(hasta que que la conversión medida sea menor que 2580 ){
                //elegir el canal 1 como entrada
                //iniciar conversión
                //espero a que termine la conversión
        }
        //cálculo el contador


        //imprimir en el lcd el contador
        //desactivar el contador

}
```

## Temporización:

Para poder temporizar el programa y que la lectura se realice cada 0.5segundos utilizaremos el timer2, para que el contador utilizado para medir capacitancia e inductancia se mantenga a la hora de generar el delay. haciéndolo de esta forma sólo conseguiremos un delay máximo de 0.91ms (en nuestro caso al tener que usar una frecuencia de 8MHz por

problemas de simulación el máximo será de 8.19ms), esta se calcula como el valor máximo del registro TIM2_ARR que es el registro de desborde.




y el máximo de tiempo se da porque el valor máximo que puede tomar el el ARR es de 65535, y para calcular el tiempo del delay es

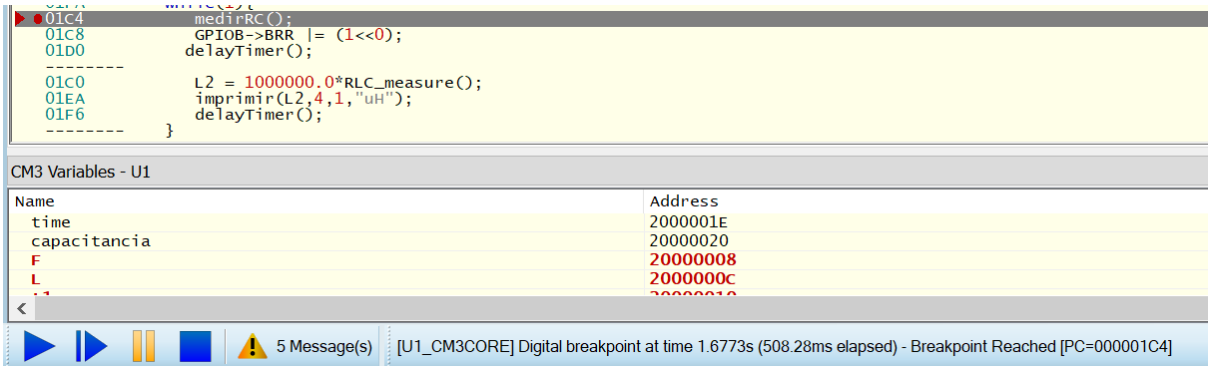
$$Delay = \frac{(ValorARR+1)}{frecuenciaMicro}$$

ahora para poder alcanzar los 0.5 segundos debemos ejecutar varias veces la ecuación del delay, decidimos ejecutar 31 veces el delay después de calcular el RC y otras 30 luego de calcular el RLC, esto con la idea de aprovechar ese tiempo para que el circuito se descargue, entonces la ecuación para conocer el valor del ARR seria:

$$\frac{Delay}{2} = 30 * \frac{(ValorARR+1)}{frecuenciaMicro}$$

$$ValorARR = \frac{Delay*frecuenciaMicro}{62} - 1$$

Para nuestro caso el valor de ARR debería  ser de 64515.



Aqui podemos ver que la simulación tarda 508 ms en realizar el loop, para que sea incluso más preciso deberíamos sacar un delay de la última ejecución, pero por razones de modularización y como no se notaría la diferencia en condiciones reales lo dejamos de esta manera.

## Codigo C

El código que realiza la medición de la capacitancia está implementado en el archivo llamado **rc.c** con su respectivo archivo de cabecera **rc.h**:

**rc.c**

```
  #include "rc.h"

  volatile uint16_t adcResult=0,time=0;
  volatile uint32_t capacitancia;

  void medirRC(){
    TIM2->CNT=0;
    TIM2->CR1=1;
    GPIOB->BSRR |= (1<<0);
    while(adcResult<2580){
        ADC1->SQR3 = 1; //choose channel 1 as input
        ADC1->CR2 = 1; //ADCON = 1 star conversion
        while((ADC1->SR&(1<<1)) == 0);
        adcResult=ADC1->DR;
    }
    time=TIM2->CNT;
    adcResult=0;
    capacitancia= time/(8);     //divido 8 porque el micro esta en 8MHz por problemas de
simulacion
    imprimir(capacitancia,4,0,"nF");
    TIM2->CR1=0;
  }

  void init_RC(){
    TIM2->CCMR2 = 0x001;
    TIM2->CCER = (1<<8);
    TIM2->ARR = 65535;
    ADC1->CR2 = 1; //ADON = 1 (power-up)
    ADC1->SMPR2 = 1<<3; // SMP1 = 001 (7.5 ADC clock cycles)
    delay_us(10000);
  }
```

**rc.h**

```
#include <stm32f103x6.h>
#include <stdio.h>
#include <delay.h>
#include <lcd.h>
#include <math.h>

void init_RC(void);
void medirRC(void);
```

El código que realiza la medición de la inductancia está implementado en el archivo llamado **rlc.c** con su respectivo archivo de cabecera **rlc.h**:

**rlc.c**

```
#include "rlc.h"
volatile float F = 0,L = 0,x1,x2,x3,t1,t0,T;

void RLC_init(){

	GPIOB->CRL = 0x44444443;
	GPIOA->ODR |= (1<<2);

	TIM2->CCMR2 = 0x001; /* Pin TIM2_CH3 as input for channel 3 */
	TIM2->CCER = (0x1<< 8); /*CC3P = 0 (rising), CC3E = 1 */
}
float RLC_measure(){
	GPIOB -> ODR|=(1<<0);
	TIM2->CNT=0;
	TIM2->CR1 = 1; /* start counting up */
	 TIM2->CCR3 =0;
	while((TIM2->SR  & (1<<3)) == 0); /* wait until the CC3IF flag*/
	t0 = TIM2 -> CCR3;
	while((TIM2->SR  & (1<<3)) == 0); /* wait until the CC3IF flag*/
	t1 = TIM2 -> CCR3; /* read the captured value */
	GPIOB -> ODR&=~(1<<0);
	T = (t1-t0)/(8000000);
	F = 1/T;
	L = 1/ ((F*2*PI) * (F*2*PI) * (C)) ;

return L;
}
```

**rcl.h**

```
#include <stm32f103x6.h>
#include <stdio.h>
#include <stdint.h>
#include "lcd.h"
#include <math.h>
#define PI 3.14159265359
#define C 0.00001

float RLC_measure(void);
void RLC_init(void);
```

**delay.c**

```
#include "delay.h"
void delay_us (uint32_t t)
{
  volatile unsigned long l = 0;
  uint32_t i;
  for( i= 0; i < t; i++)
  for(l = 0; l < 6; l++)
  { }
}

void delay()
{
  TIM2->ARR = 64515;
  TIM2->SR = 0; /* clear the UIF flag */
  TIM2->CR1 = 1; /* up counting */
  while((TIM2->SR & (1<<0)) == 0); /* wait until the UIF flag is set */
  TIM2->CR1 = 0; /*stop counting */
}


void delayTimer(){
    uint32_t i;
    for(i=0;i<31;i++){
            delay();
    }
}
```

**delay.h**

```
#include <stm32f103x6.h>
#include <stdio.h>
#include <stdint.h>
void delay_us (uint32_t );

void delay(void);
void delayTimer(void);
```

**main.c**

```
#include <stm32f103x6.h>
#include <stdio.h>
#include <delay.h>
#include <lcd.h>
#include <math.h>
#include <rlc.h>
#include <rc.h>




volatile            uint32_t L2=0,aux,j;
int main (void)
{

  RCC->APB2ENR |= 0xFC | (1<<9) | (1<<14);
  RCC->APB1ENR |= (1<<0);

  RLC_init();

  GPIOA->CRL = 0x44444804; //PA1 as analog input
  GPIOB->CRL = 0x44444443; //PB0 as output
  GPIOB->BRR = (1<<0);
  init_RC();
  lcd_init();
  clearLCD();
  while(1){
    medirRC();
    GPIOB->BRR |= (1<<0);
   delayTimer();

    L2 = 1000000.0*RLC_measure();
    imprimir(L2,4,1,"uH");
    delayTimer();
  }

}
```

**lcd.c**

```
#include "lcd.h"
void lcd_init()
{
  GPIOB->CRL = 0x33333333;
  GPIOB->CRH = 0x33333333;
  GPIOB->BRR = (1<<LCD_EN); //same as GPIOB>ODR &= ~(1<<LCD_EN); /* LCD_EN =
0 */
```

```
  GPIOB->BRR = (1<<LCD_RW); //same as GPIOB>ODR &= ~(1<<LCD_RW); /* LCD_RW
= 0 */
  delay_us(3000); /* wait 3ms */
  lcd_sendCommand(0x38); /* init. LCD 2 line,5´7 matrix */
  lcd_sendCommand(0x0C); /* display on, cursor on */
  lcd_sendCommand(0x01); /* clear LCD */
  delay_us(2000); /* wait 2ms */
  lcd_sendCommand(0x06); /* shift cursor right */
}

void lcd_sendCommand (uint8_t cmd)
{
  GPIOB->BRR = (1<<LCD_RS); /* RS = 0 for command */
  lcd_putValue(cmd);
}
/**************************
Prints in LCD display
**************************/
void lcd_sendData (uint8_t data)
{
  GPIOB->BSRR = (1<<LCD_RS); /* RS = 1 for data */
  lcd_putValue(data);
}

void lcd_putValue (uint8_t value)// 8-bit
{
  GPIOB->BRR = 0x1FE0; /* clear PB4-PB12 */
  GPIOB->BSRR = (value<<5)&0x1FE0; /* put value on PB4-PB12 , shidteo 5 para poder
enmascarar correctamente el valor*/
  GPIOB->BSRR = (1<<LCD_EN); /* EN = 1 for H-to-L pulse */
  delay_us(2); /* make EN pulse wider */
  GPIOB->BRR = (1<<LCD_EN); /* EN = 0 for H-to-L pulse */
  delay_us(100); /* wait */
}

void lcd_string(uint8_t* data, uint8_t nBytes)//Outputs string to LCD
{
register uint8_t i;

        // check to make sure we have a good pointer
        if (!data) return;

        // print data
        for(i=0; i<nBytes; i++)
        {
                lcd_sendData(data[i]);
        }
}
```

```
void lcd_gotoXY(uint8_t x, uint8_t y) //Cursor to X Y position
{
	register uint8_t DDRAMAddr;
	// remap lines into proper order
	switch(y)
	{
	case 0: DDRAMAddr = LCD_LINE0_DDRAMADDR+x; break;
	case 1: DDRAMAddr = LCD_LINE1_DDRAMADDR+x; break;
	case 2: DDRAMAddr = LCD_LINE2_DDRAMADDR+x; break;
	case 3: DDRAMAddr = LCD_LINE3_DDRAMADDR+x; break;
	default: DDRAMAddr = LCD_LINE0_DDRAMADDR+x;
	}
	// set data address
	lcd_sendCommand(1<<LCD_DDRAM | DDRAMAddr);

}

void lcd_clr(void)	//Clears LCD
{
	lcd_sendCommand (1<<LCD_CLR);
}


void imprimir(uint32_t dato, uint32_t x, uint32_t y, char* medida){
  int aux,cont=0;
  unsigned char vec[16];
  lcd_gotoXY(x,y);
  while (dato > 0){
    aux = dato - ((dato / 10)*10);
    dato=dato / 10;
    vec[cont++] = (uint32_t)(aux + '0');
  }
  while(cont>0){
    lcd_sendData((uint8_t)vec[--cont]);
  }
  lcd_string(medida,2);
}

void clearLCD(){
  lcd_clr();
  lcd_gotoXY(0,0);
  lcd_string("C: ",3);
  lcd_gotoXY(0,1);
  lcd_string("L: ",3);
}
```

**lcd.h**

```
#define LCD_RS 15
#define LCD_RW 14
#define LCD_EN 13

#define LCD_LINE0_DDRAMADDR		0x00
#define LCD_LINE1_DDRAMADDR		0x40
#define LCD_LINE2_DDRAMADDR		0x14
#define LCD_LINE3_DDRAMADDR		0x54

#define LCD_DDRAM		7

#define LCD_CLR			0

#include <stm32f103x6.h>
#include "delay.h"
#include <stdint.h>

void lcd_init(void);
void lcd_sendCommand (uint8_t);
void lcd_sendData (uint8_t );
void lcd_putValue (uint8_t );
void lcd_string(uint8_t* , uint8_t );
void lcd_gotoXY(uint8_t x, uint8_t y);
void lcd_clr(void);

void imprimir(uint32_t, uint32_t, uint32_t,char*);
void clearLCD(void);
```

## Parte 2: Diseño de Hardware: Circuito impreso (PCB)

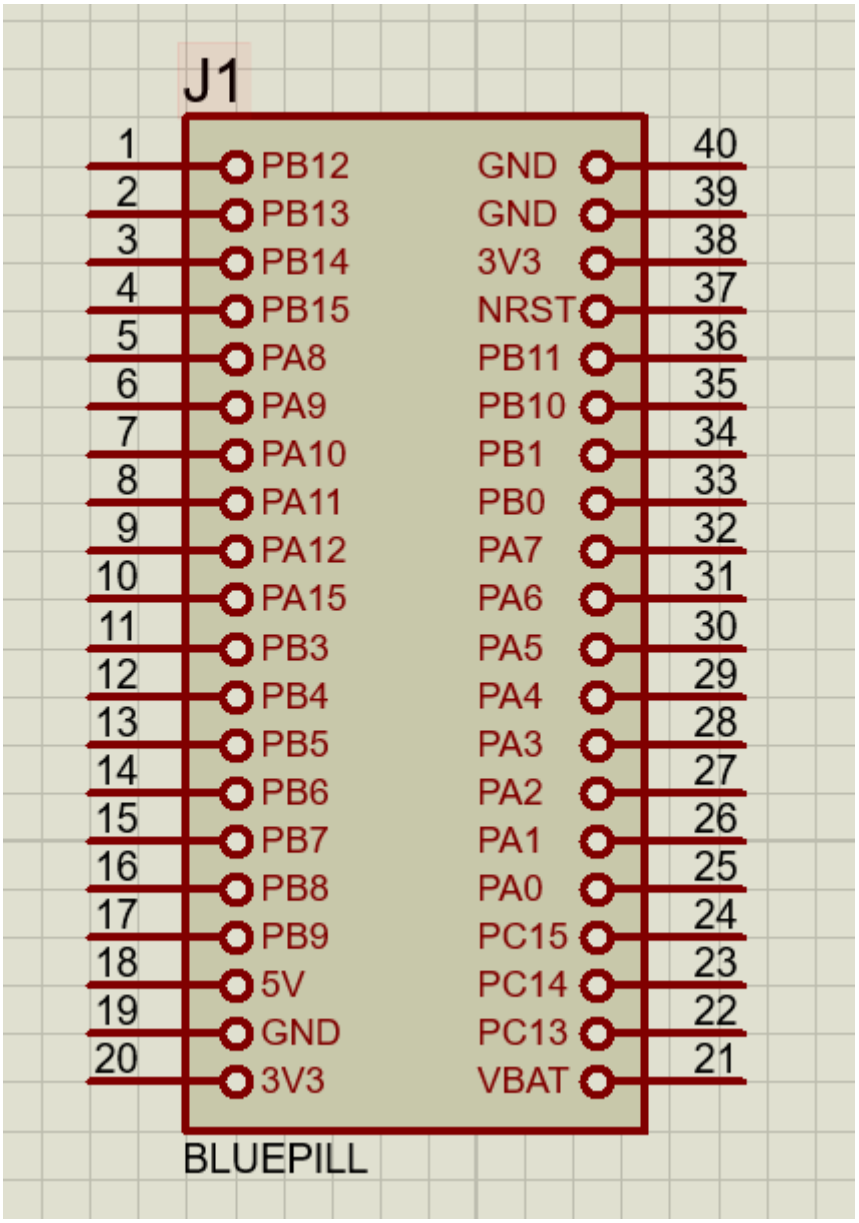
*En primer lugar diseñar el esquema eléctrico completo con todos los componentes y sus conexiones. Tener en cuenta que el PCB contendrá los conectores necesarios para conectar la placa BluePill completa, la cual proveerá de alimentación al resto de los circuitos y los conectores para colocar los elementos pasivos C y L a medir.*
*Luego diseñar el circuito PCB del sistema implementado según los siguientes requerimientos para la fabricación artesanal:*

*Simple faz de tamaño máximo: 10x10cm*
*Pads o vías: 1.8 mm de corona, 0.7mm los agujeros*
*Ancho de pista y separación: mínimo 0.7mm, preferido 1mm.*
*Sin serigrafía de componentes y sin mascara antisoldante.*

*Realizar en el informe un resumen de los procedimientos realizados para diseñar el PCB y adjuntar imágenes de la capa superior (capa de componentes), capa inferior (capa de soldadura) y vistas 3D del circuito completo.*
*En la entrega a través del campus virtual, deberá adjuntar un .zip con los proyectos.*

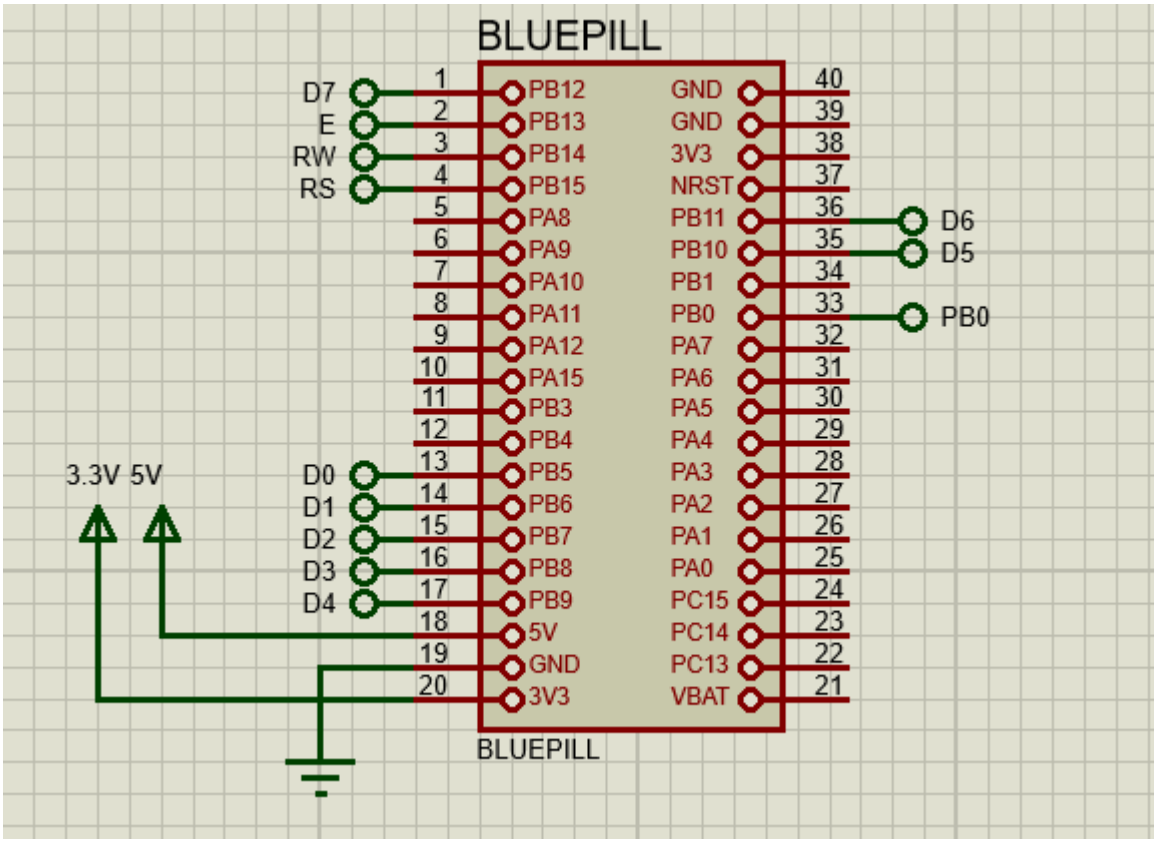
## Esquemático:

Teniendo en cuenta que se utilizará una placa BluePill completa, lo primero en el desarrollo del esquemático es la creación del componente de la placa BluePill en Proteus. Para ello utilizamos como referencia el componente CONN-DIL40, el cual es un conector que posee 40 puertos, 20 de cada lado, que entre sí poseen una distancia de 0.1 pulgadas. Luego de personalizar nuestro componente llegamos a este resultado:

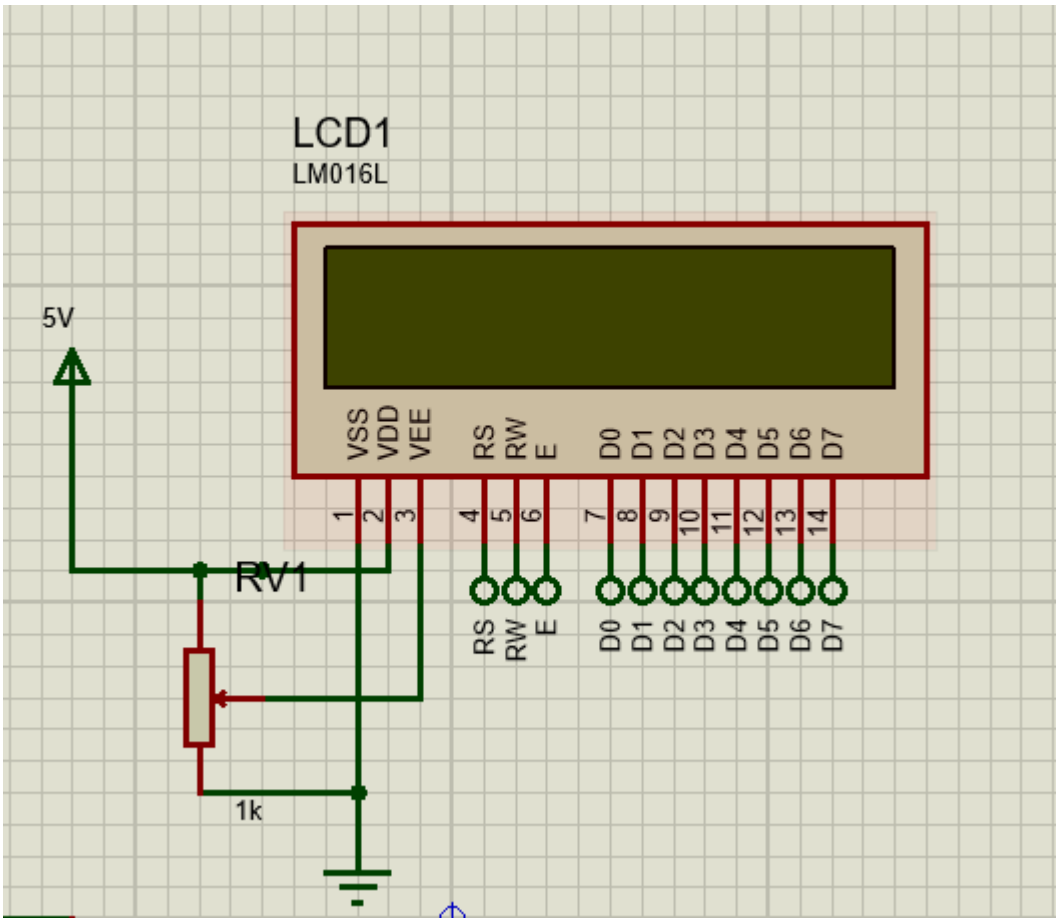



Con nuestro componente ya creado y listo para utilizar, generamos las partes restantes del esquemático y las conexiones necesarias para que la placa bluepill alimente los demás circuitos.

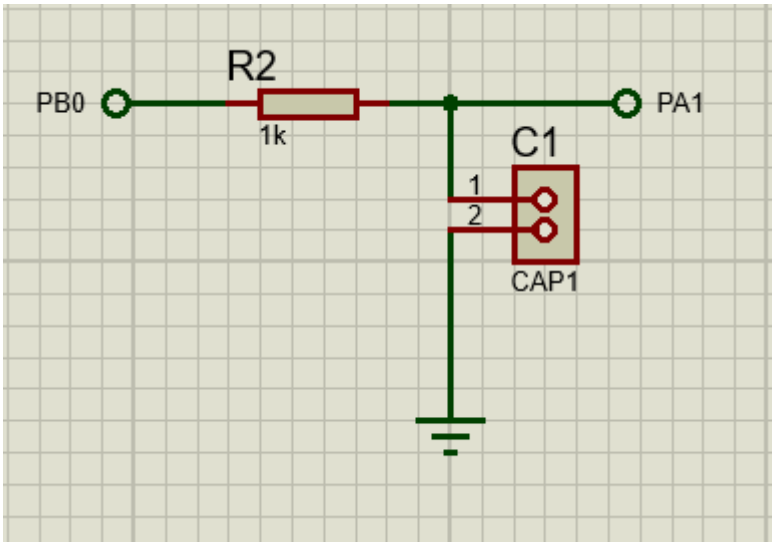
De la misma forma que hemos conectado nuestro microcontrolador, procedemos a conectar la bluepill de la siguiente forma.

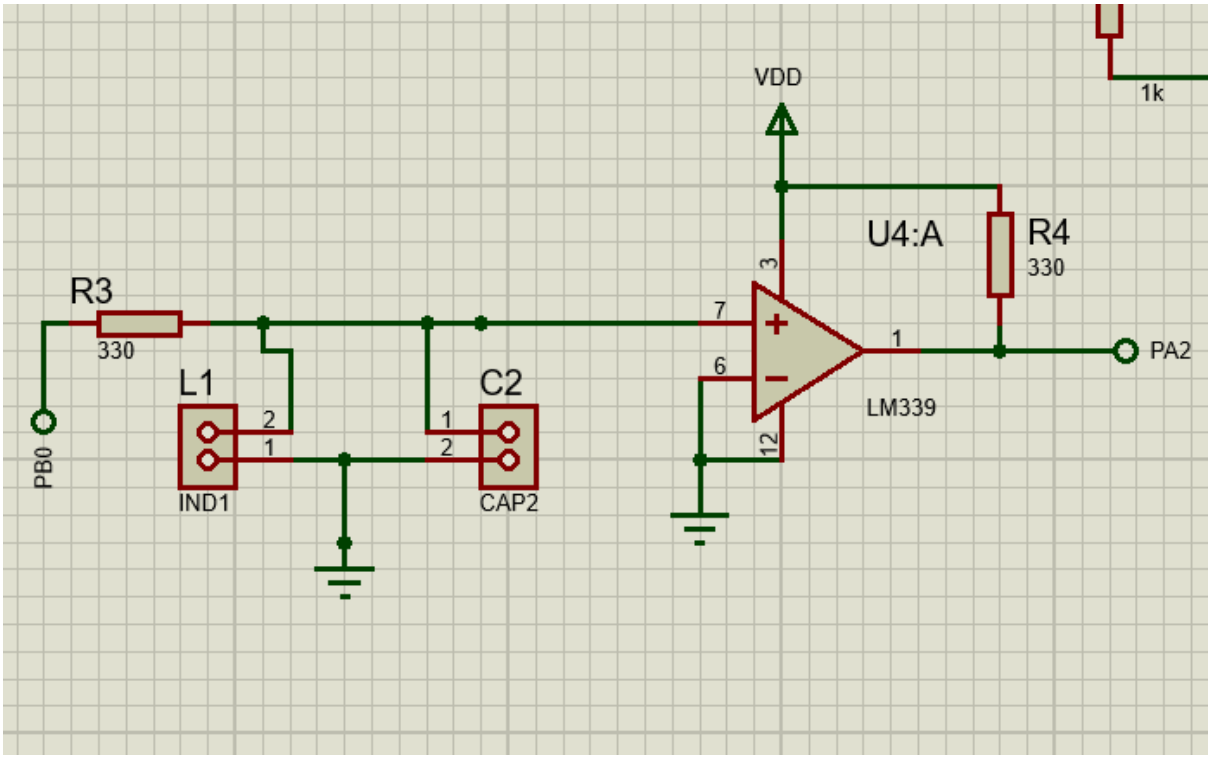



Al módulo LCD no se le generan cambios más que conectarlo a una resistencia variable para controlar la intensidad del brillo.

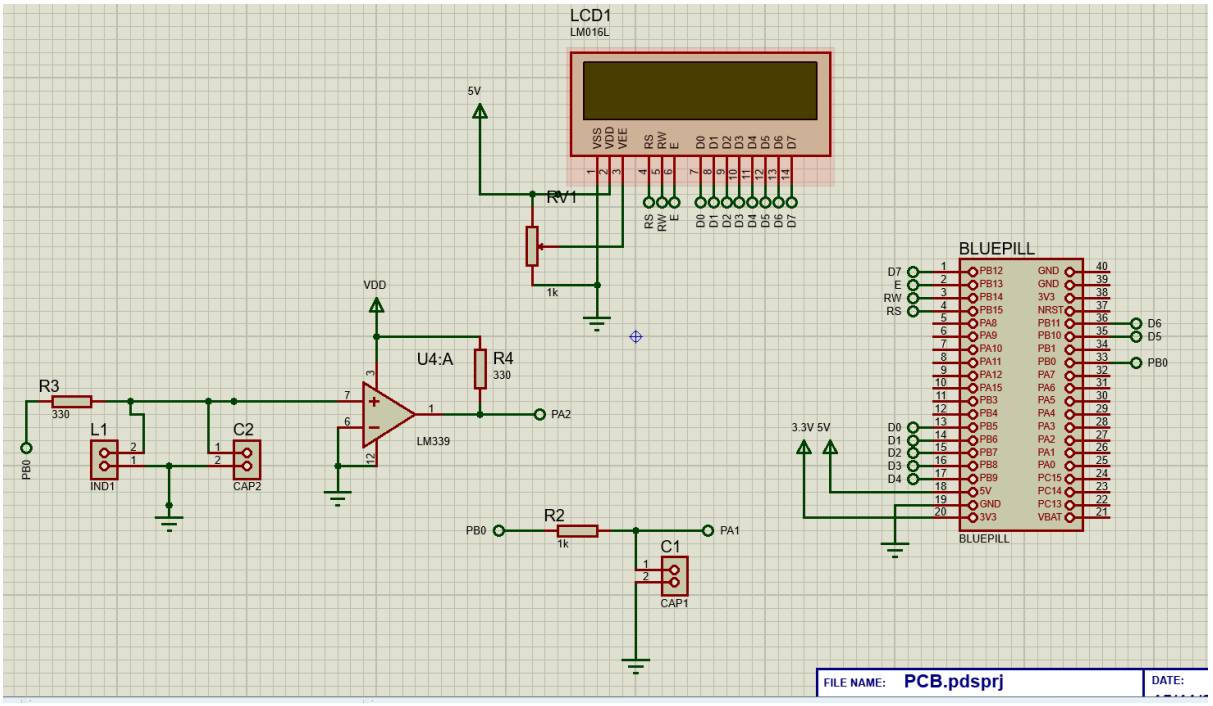


Sin embargo en nuestros circuitos RLC y RC si se generan cambios. En el lugar donde antes se encontraban los capacitores e inductores, se colocarán conectores para que así se puedan conectar distintos tipos de inductores y capacitores, y de esta forma, poder experimentar con varios tipos de instrumentos. Para utilizarlos estos deberán ser insertados en los conectores correspondientes de la placa.



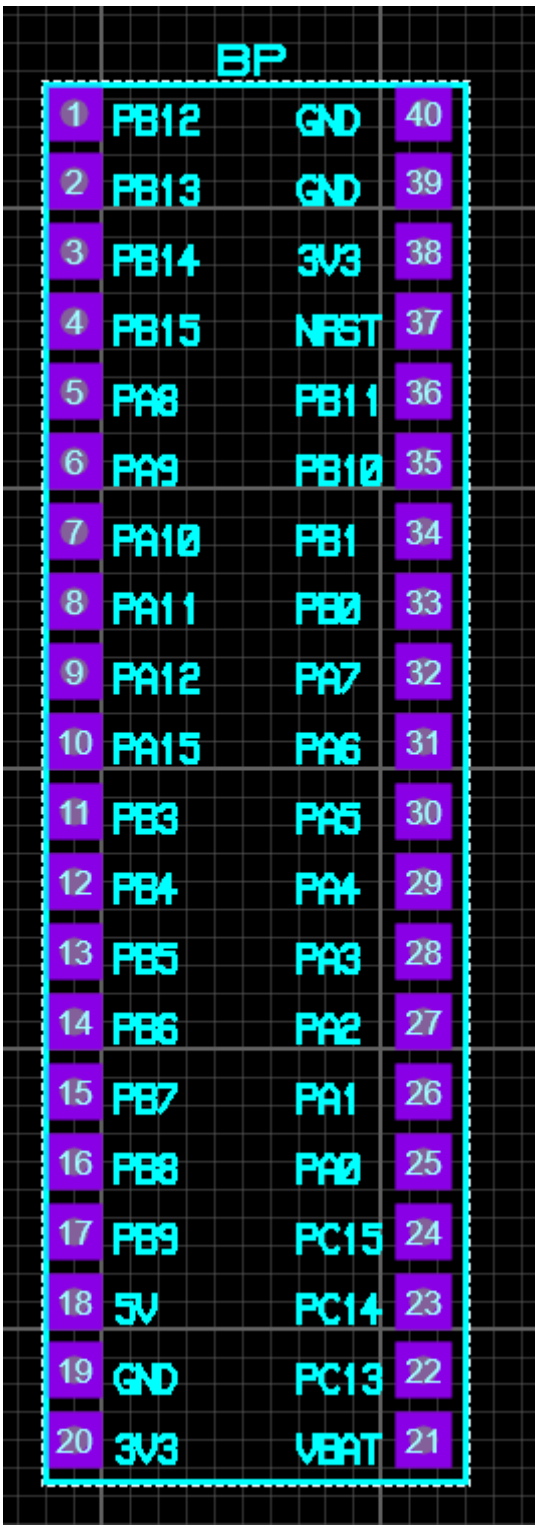




Finalmente el esquemático completo resulta de la siguiente forma:

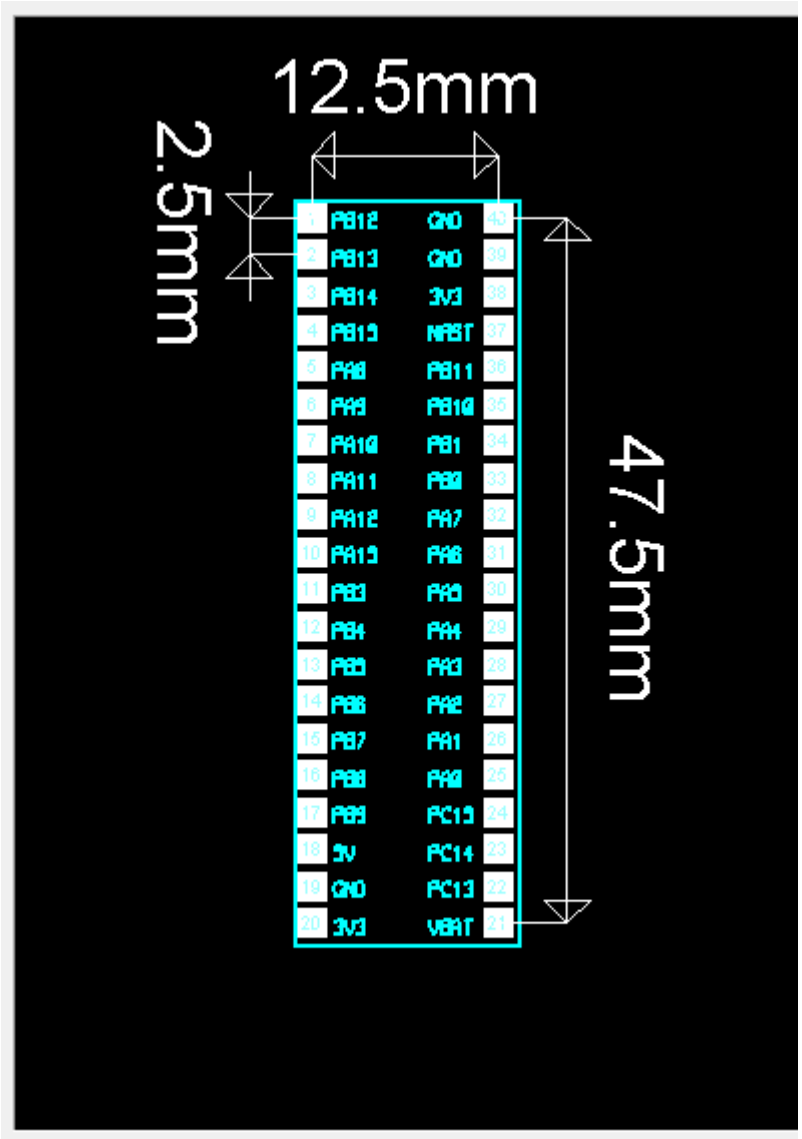



## PCB:

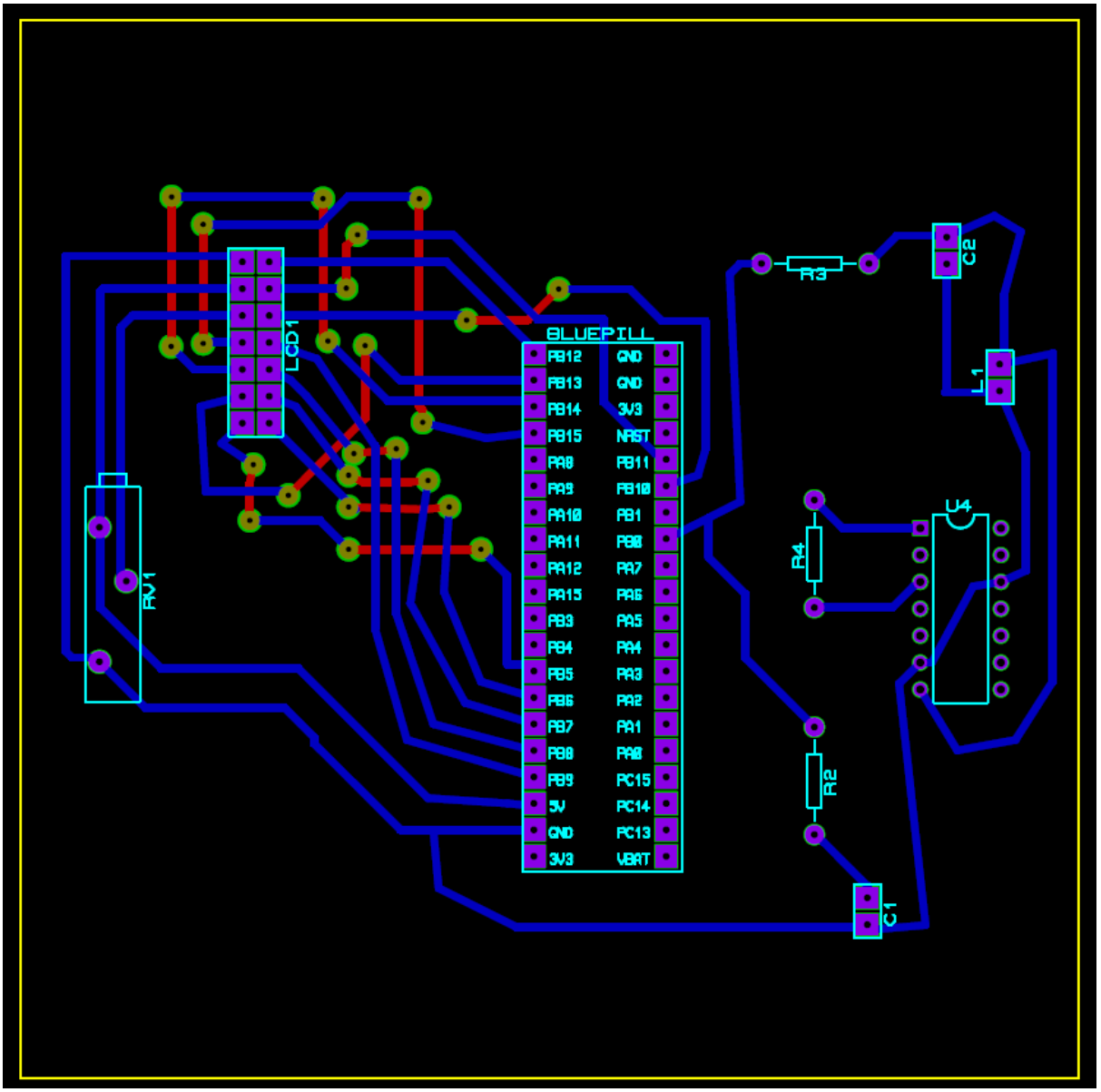
Lo primero a tener en cuenta en la realización de nuestro PCB es que los componentes utilizados en el esquemático tengan su representación de tipo PCB. Para esto, decidimos personalizar el PCB de nuestra BLUEPILL y creamos un PACKAGE de la siguiente forma:




Esta representación PCB está inspirada en la representación del conector CONN-DIL40 y utiliza las mismas medidas de los PADS que ese conector. La medida de los pads es de 80 mils el cuadrado y 40 mils el agujero de la mecha del taladro. Podemos ver más medidas en la siguiente ilustración.




Luego de generado nuestro PCB de BLUEPILL el paso siguiente es el de crear nuestra plaqueta simple faz de 10cmx10cm, el cual es realizado con la herramienta rectángulo bajo la capa BOARD EDGE.
Ya teniendo la plaqueta, se insertan todos los componentes del esquemático en nuestro espacio de trabajo de PCB, el lugar donde se insertan es importante para más tarde no intentar que nuestras pistas de material conductor se junten o se genere una intersección.
Una vez conectado los componentes se procede a conectarlos mediante pistas y pads en el caso de que se deba pasar por arriba de un conductor. Para esto utilizamos pistas de 0.7mm (el mínimo indicado por la consigna) y vías de 70x30 (esto significa 1.8 mm de corona y 0.7mm los agujeros). Ya que nuestra plaqueta es simple faz solo poseemos conductor de un solo lado de la plaqueta, por lo tanto, los cruces que se hagan del lado de los componentes deberían ser conectados por un cable. En nuestro PCB los conductores rojos representan estos cables que se encuentran del lado superior de la plaqueta.

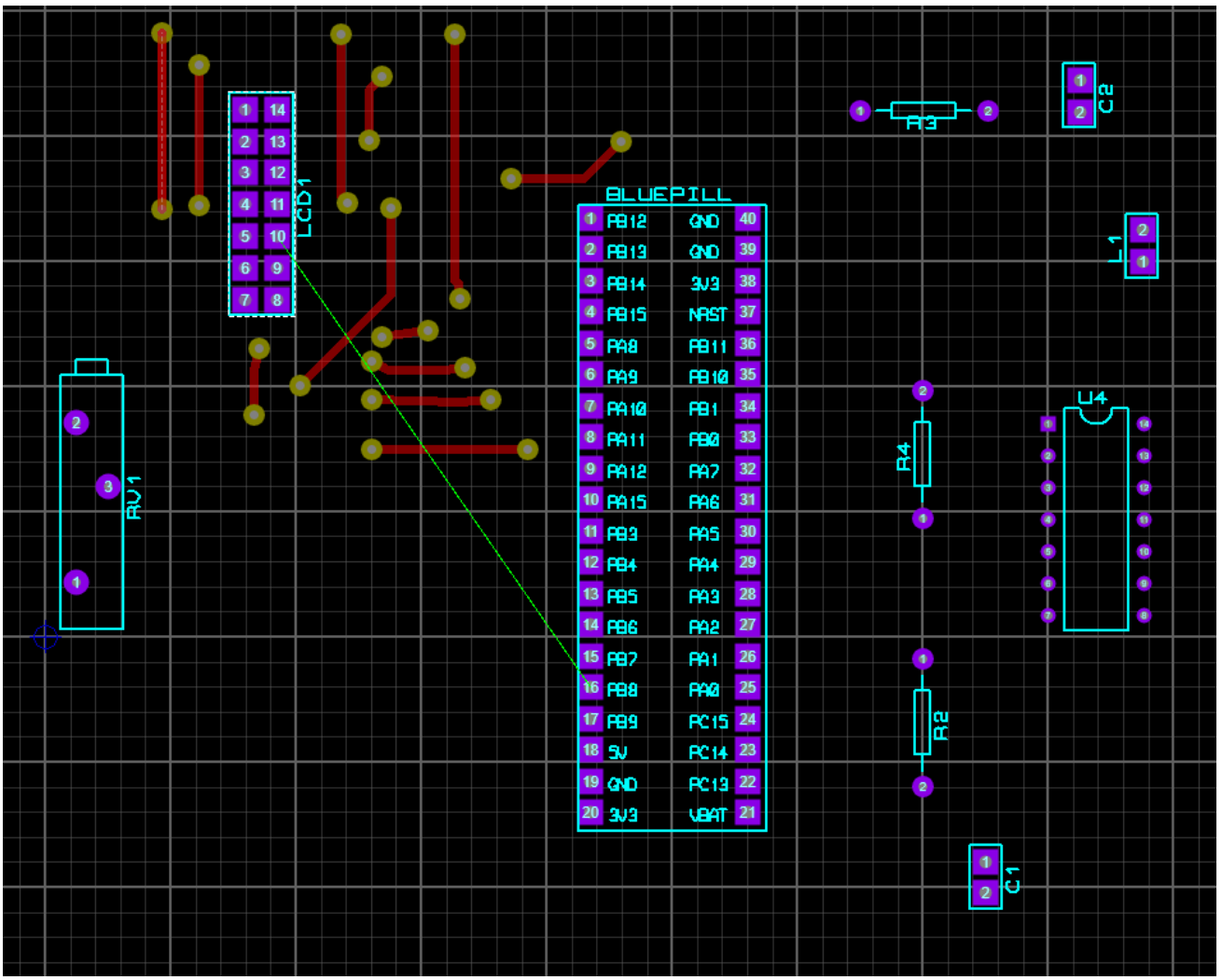


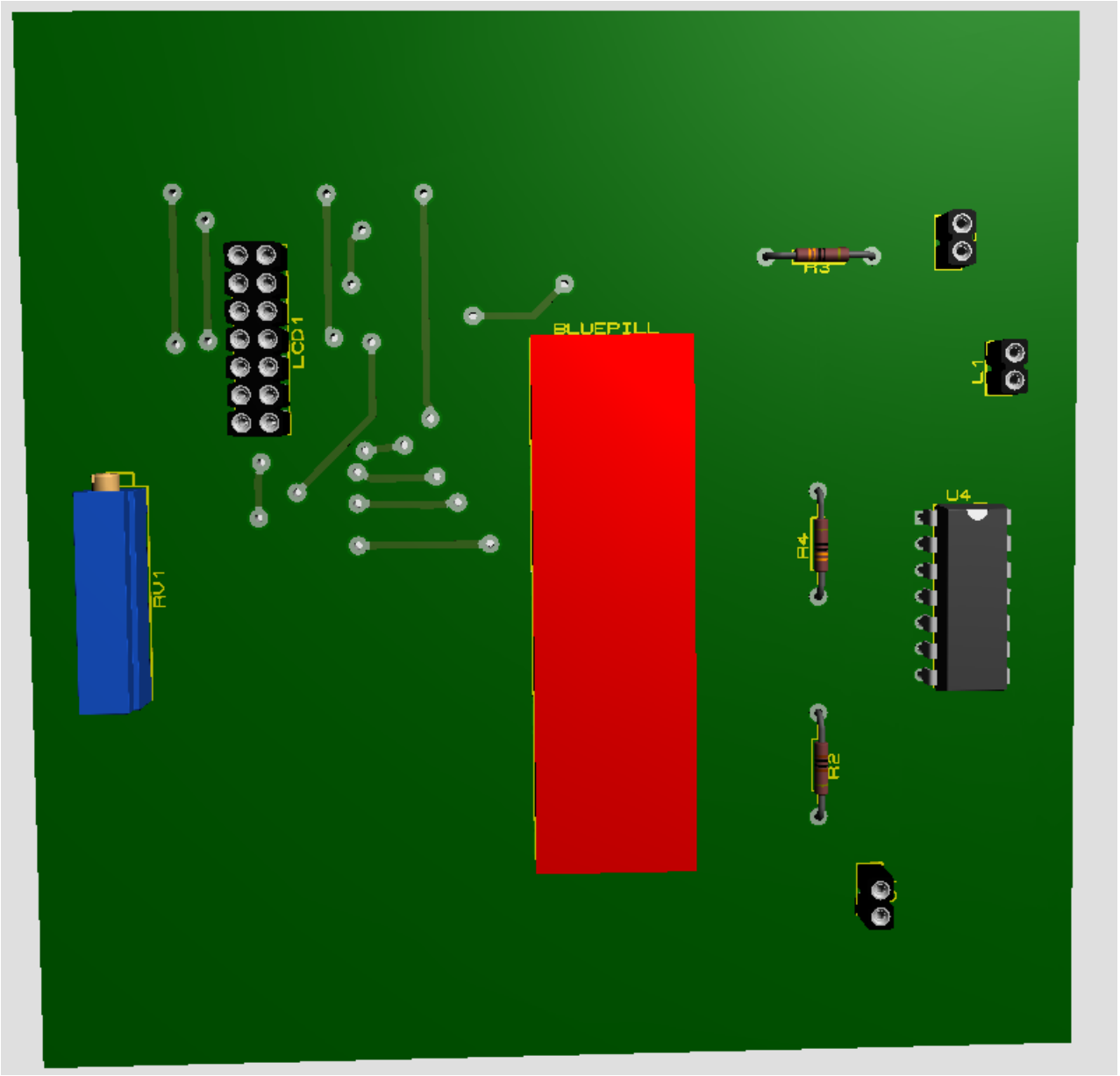

Como podemos ver el único lugar donde se generó intersección fue en el conexionado del LCD. Al utilizar cables del lado de los componentes y una plaqueta simple faz, este diseño puede resultar no intuitivo a simple vista, por lo que se le podría generar optimizaciones al ubicar los componentes o utilizar una plaqueta con más capas.

La configuración de reglas de diseño nos permite verificar que la separación de las pistas no sea menor a 0.7mm por lo tanto las reglas de la consigna se mantienen válidas.

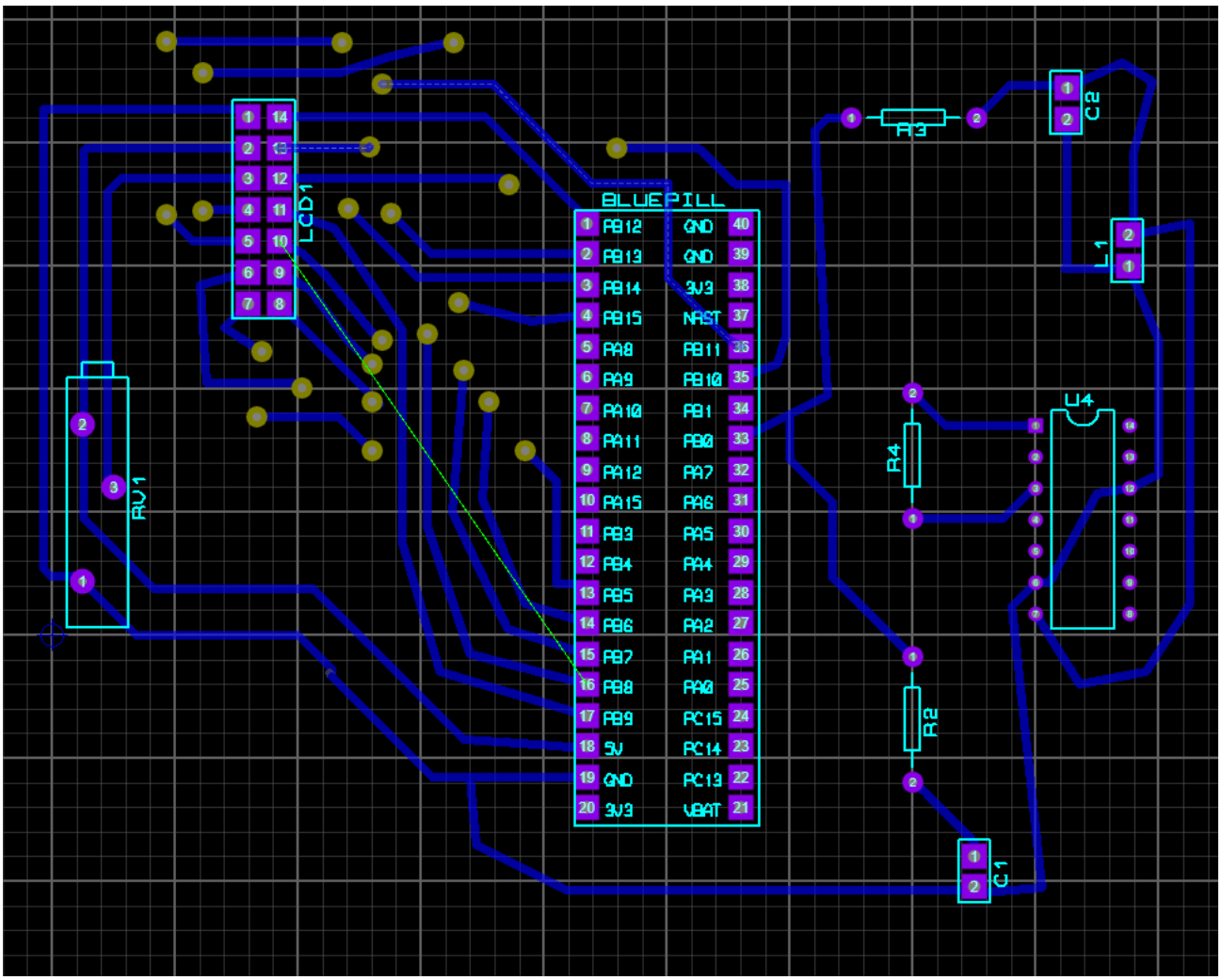
Finalmente se muestran imágenes de la capa de componentes y de la capa de soldadura, en 2d y 3d:

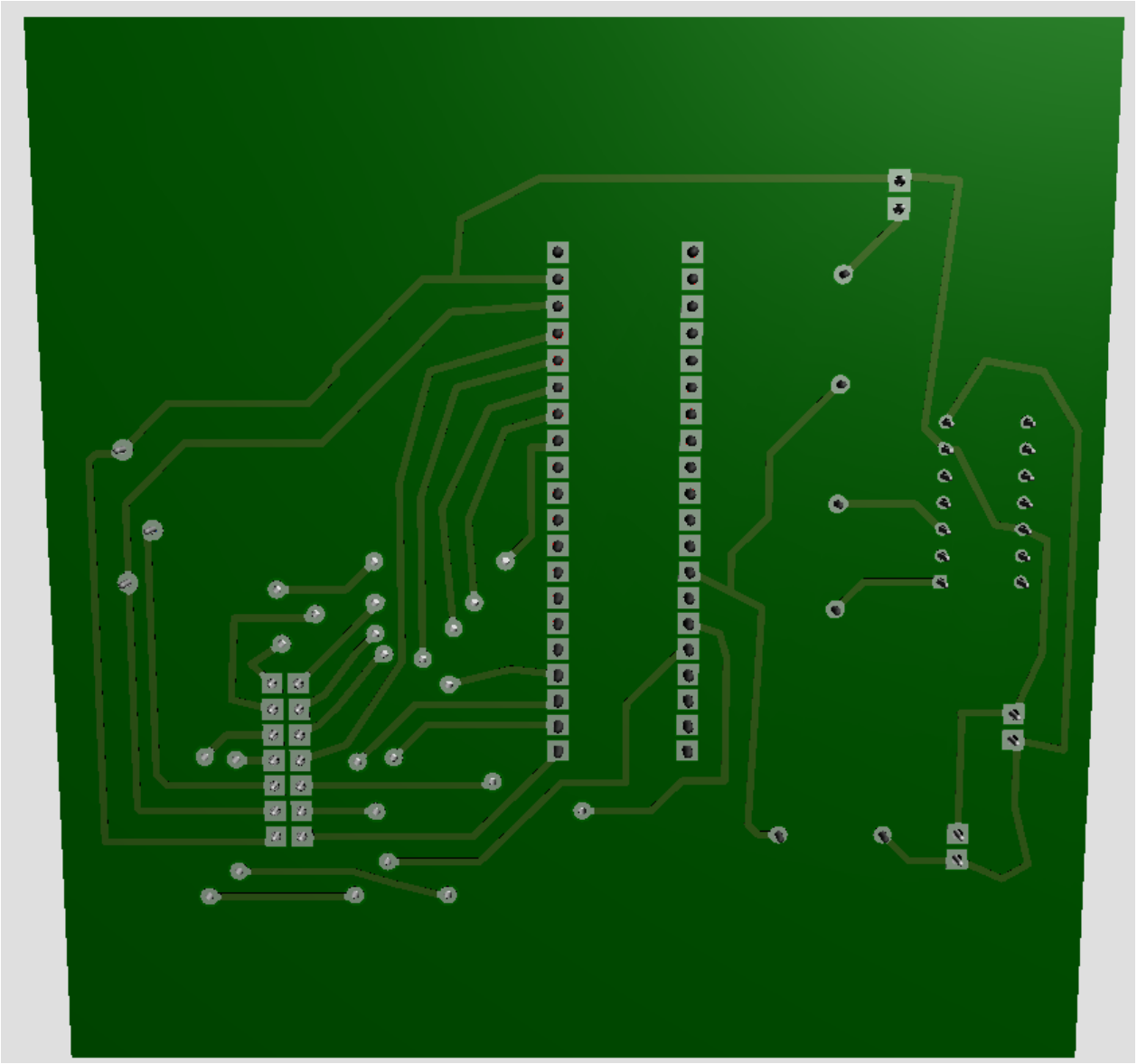
## Capa de Componentes:


LCD1
BLUEPILL
PB12 GND
PB13 GND
PB14 3V3
PB15 NRST
PA8 PB11
PA9 PB10
PA10 PB1
PA11 PB0
PA12 PA7
PA15 PA6
PB3 PA5
PB4 PA4
PB5 PA3
PB6 PA2
PB7 PA1
PB8 PA0
PB9 PC15
5V PC14
GND PC13
3V3 VBAT
RV1
R3
C2
L1
U4
R4
R2
C1




**Capa de soldadura:**


LCD1
BLUEPILL
PB12 GND
PB13 GND
PB14 3V3
PB15 NRST
PA8 PB11
PA9 PB10
PA10 PB1
PA11 PB0
PA12 PA7
PA15 PA6
PB3 PA5
PB4 PA4
PB5 PA3
PB6 PA2
PB7 PA1
PB8 PA0
PB9 PC15
5V PC14
GND PC13
3V3 VBAT
RV1
R3
C2
L1
U4
R4
R2
C1



## Validación:

Github: https://github.com/Pedro-Molina/Cap_Ind

En el siguiente video se muestra la correcta medición de la inductancia y capacitancia.
Video: https://youtu.be/XQxyYjv4KJU