1.2.21. Display LCD y memorias

En este apartado vamos a utilizar un Display LCD para mostrar datos procedentes de nuestra placa ZUM BT o Arduino UNO compatible.

Hay diferentes tipos de displays LCD. Podemos clasificarlos en función del número caracteres que puede representar (los hay por ejemplo de 16x2 como el de la Figura 1.2.21-1 derecha o 16x4 como el de la Figura 1.2.21-1 izquierda). También los podemos clasificar en función de la manera que tienen de recibir datos (con comunicación I2C, SPI o serie)

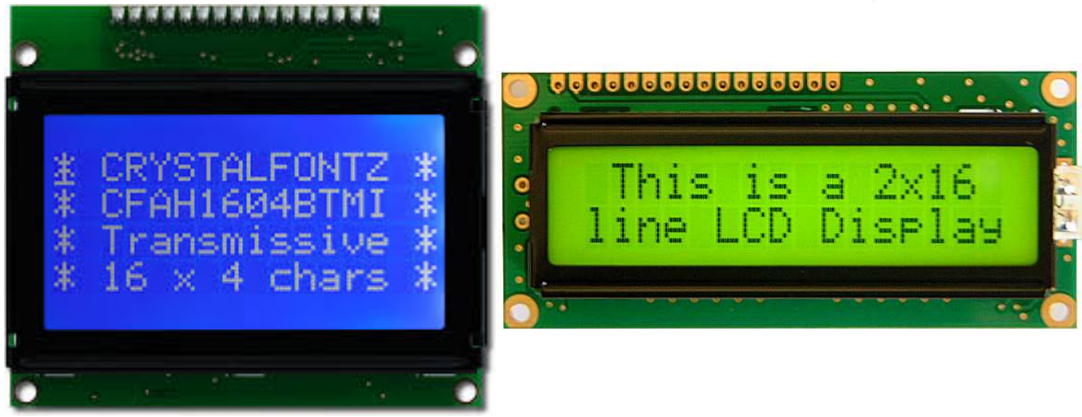


Figura 1.2.21-1 Display de 16x4 caracteres (izquierda), Display de 16x2 caracteres (derecha)

Nosotros utilizaremos un display 16x2 caracteres (16 columnas 2 filas) con una comunicación I2C, pero utilizar otro tipo de displays es también muy sencillo con Arduino y hay multitud de tutoriales en la web de cómo hacerlo.

Componentes

- Tarjeta ZUM BT o Arduino UNO compatible
- Display LCD de BQ o cualquiera con comunicación I2C

1.2.21.1. Hola Mundo

Vamos a empezar programando una aplicación muy sencilla que muestre un mensaje de saludo por nuestro display.

Conexionado

Seguiremos el conexionado mostrado en la Figura 1.2.21-2

- Pin SCL Display – Pin A5 Arduino
- Pin SDA Display – Pin A4 Arduino
- Pin VCC Display – Cualquier Pin VCC Arduino
- Pin GND Display → Cualquier Pin GND Arduino

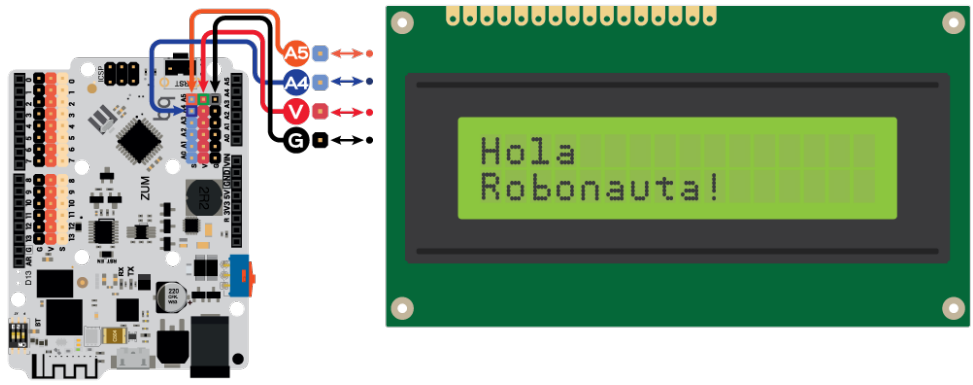


Figura 1.2.21-2 Conexionado por I2C con LCD de 16x2 caracteres (de diwo.bq.com)

Abriremos un proyecto en Bitbloq y añadiremos la placa Arduino y el display, En Bitbloq solo hace falta unir el pin SCL del display con el A5 y el pin SDA del display con el pin A4 (ver Figura 1.2.21-3).

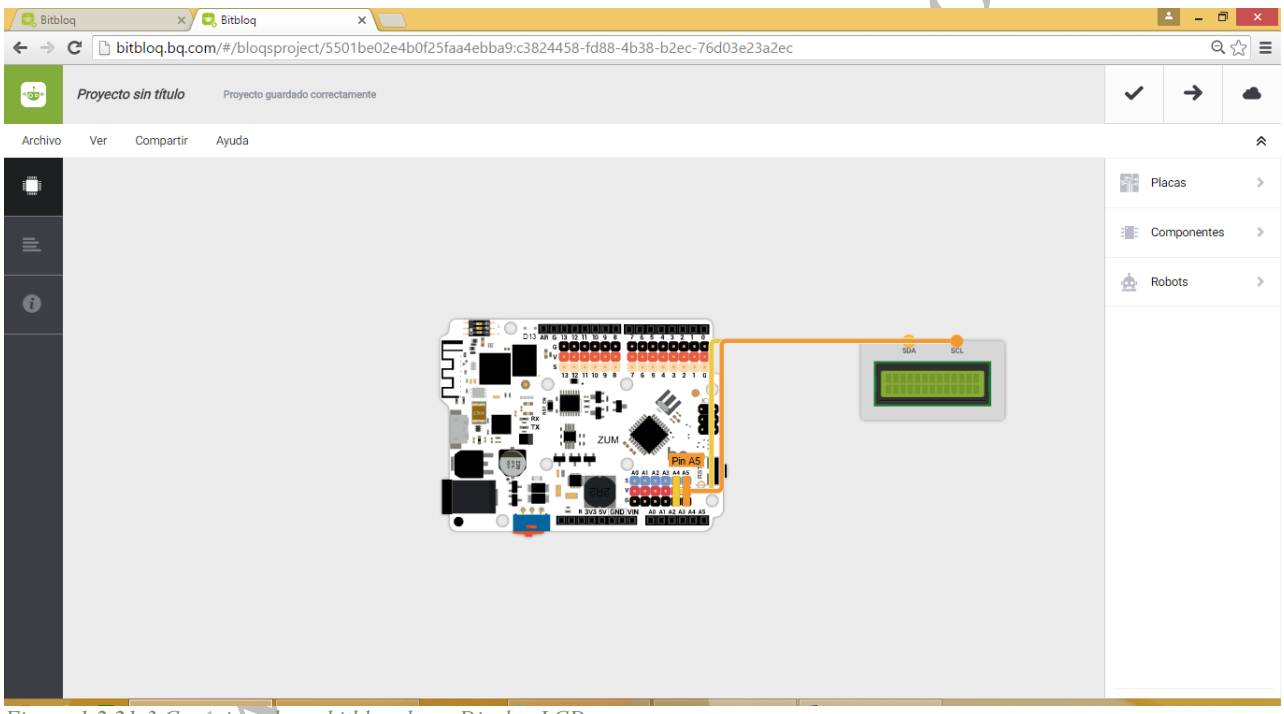


Figura 1.2.21-3 Conexionado en bitbloq de un Display LCD

Programación

Para poder ver bien el texto los displays tienen una luz de retroiluminación. Esta luz puede encenderse o apagarse para ahorrar energía. Por eso, lo primero que haremos, como vemos en la Fg, es encender esta retroiluminación y posteriormente mandar el mensaje que queremos que muestre. Como puedes ver hemos colocado los bloques en la parte Setup del código ya que en esta actividad solo vamos a mostrar un mensaje, y con mandárselo una vez al display este ya lo mantiene todo el tiempo (recuerda que el código escrito en el Setup solo se ejecuta una vez justo antes del Loop que se ejecuta una y otra vez).

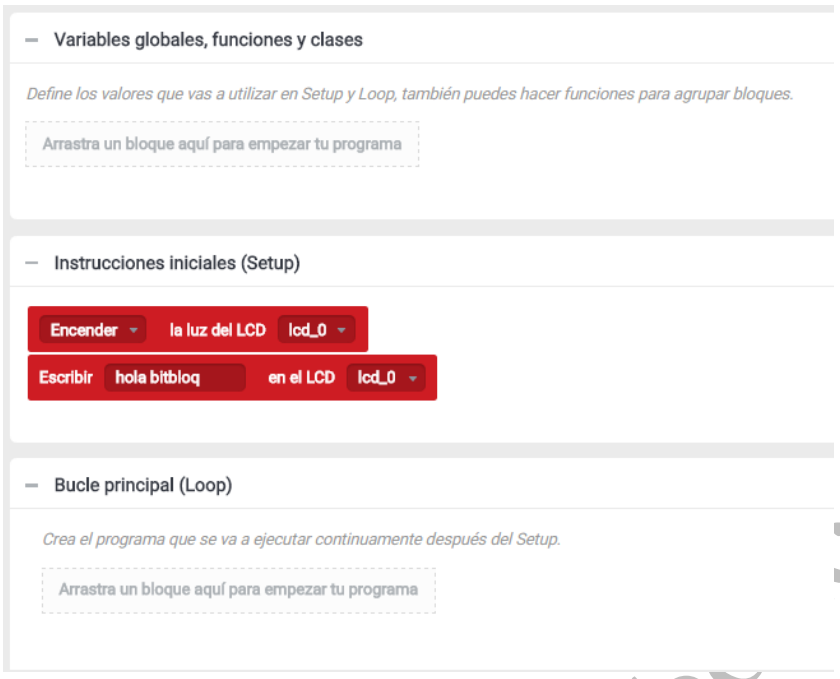


Figura 1.2.21-4 Código de bloques de la actividad Hola Mundo del display

El código Arduino se muestra en la Figura 1.2.21-5. Como vemos hay más cosas que tenemos que tener en cuenta cuando escribimos código en Arduino para el uso de un display. Además de indicar que tipo de display estamos utilizando (línea 11) deberemos utilizar comandos para:

- Borrar el contenido del display cuando queramos escribir algo nuevo y que no se vea lo que había antes (clear de la línea 12).
- Activar la luz de retroiluminación (SetBackLight de la línea 13).
- Mandar mensaje de texto (print de la línea 14).

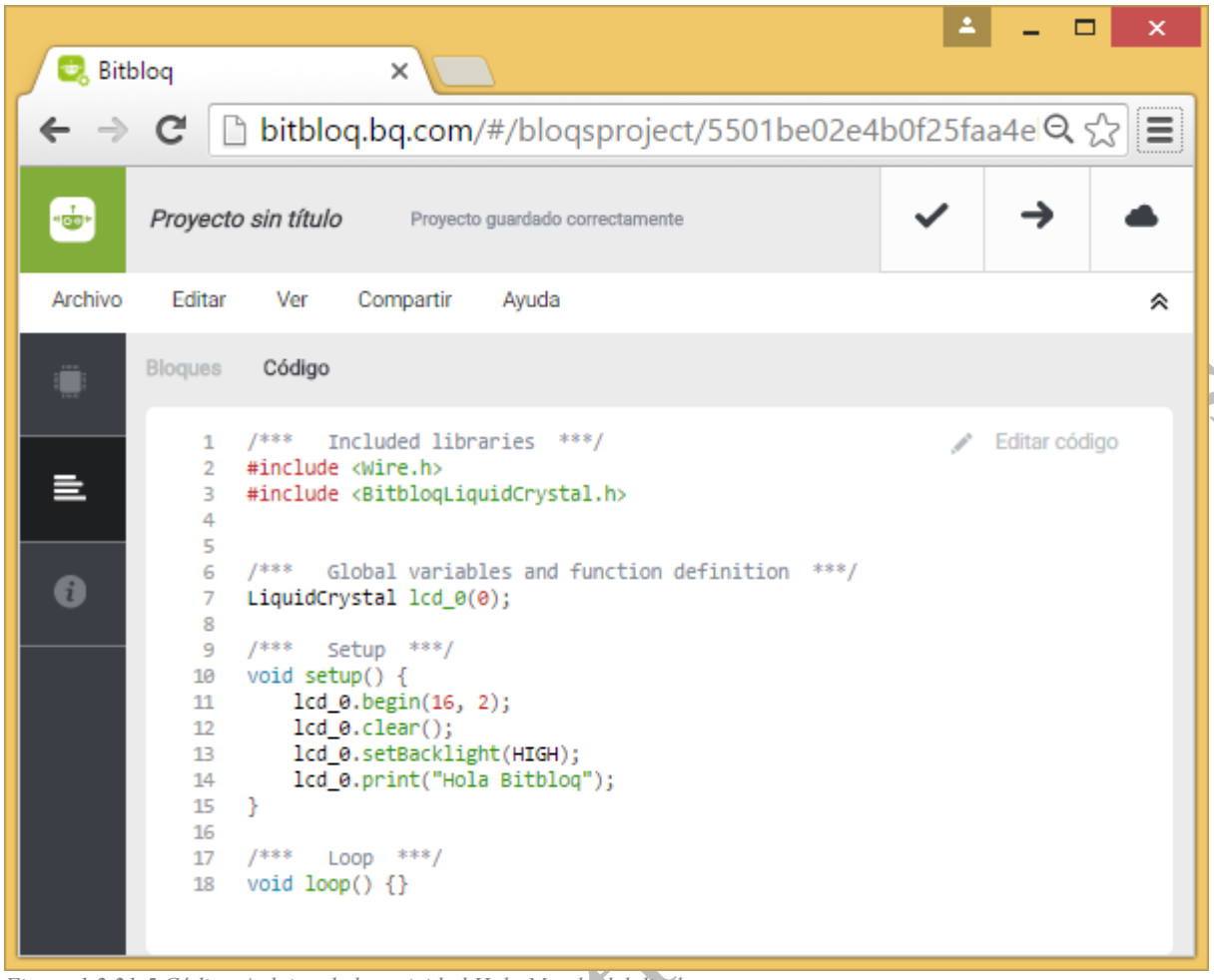


Figura 1.2.21-5 Código Arduino de la actividad Hola Mundo del display

Algo de teoría

Hay tres conceptos fundamentales en electrónica (y por supuesto en robótica) que se usan en los display y que es conveniente que repasemos:

- El protocolo de comunicaciones como es el I2C. Este protocolo define la manera en la que se comunican dos o más dispositivos por medio de dos cables de comunicación como vemos en la Figura 1.2.21-6 (un cable de datos, otro de señal de reloj para sincronizar dispositivos) y los cables de alimentación (VCC y GND). Este protocolo es tipo maestro/esclavo, en donde cada esclavo tiene una dirección, estilo DNI, y cuando un maestro necesita algo de un esclavo escribe su DNI en el bus de datos y espera que éste le conteste.

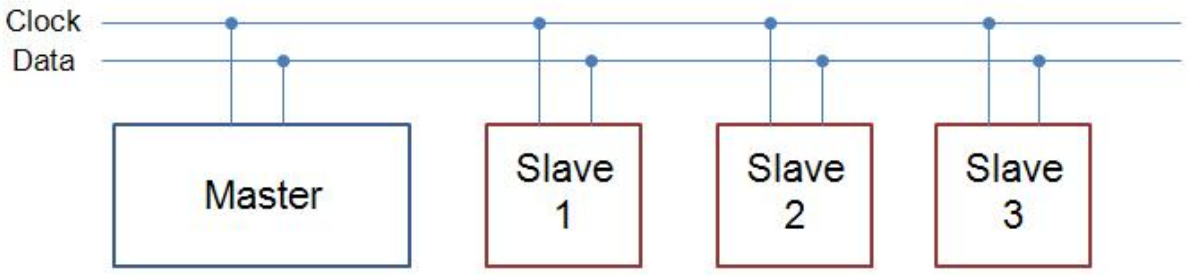


Figura 1.2.21-6Cables de comunicación del protocolo I2C. A estos cables hay que añadir el GND y VCC de alimentación, como hemos hecho con el display LCD.

- El uso de memorias: un display contiene memorias internas para almacenar los caracteres que puede representar. Cuando se le realiza una petición de escribir un carácter, realmente lo que hacemos es acceder (mediante una dirección en hexadecimal) a una memora ROM que tiene el LCD donde tiene almacenado como se dibujan los caracteres (ver).

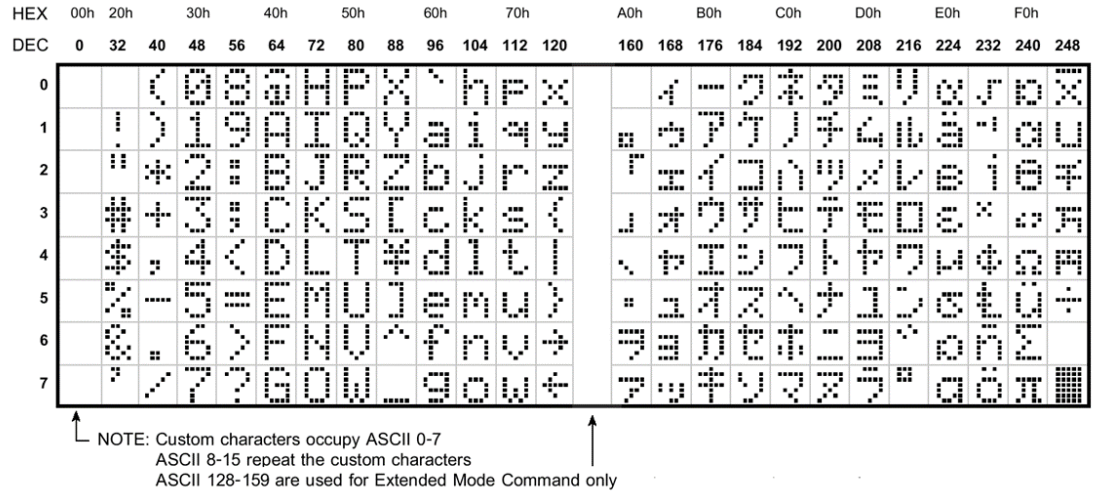


Figura 1.2.21-7 Memoria ROM de un LCD. Como podemos ver, cada dirección de memoria (en hexadecimal) corresponde a un caracter.

- La tecnología LCD (liquid crystal display) utilizada también en pantallas y monitores de todo tipo. Básicamente esta tecnologia se basa en la polarización de la luz: el cristal líquido deja pasar o no la luz (o si es una pantalla en color, como las teles, deja pasar o no una franja de longitud de onda).

1.2.21.2. Mostrar por lcd lo que mandamos por el puerto serie

Vamos a complicar un poco más la actividad, mostrando por el LCD un mensaje que hayamos mandado desde nuestro PC a la Placa ZUM BT o Arduino UNO compatible. Esto puede servir por ejemplo para comprobar que las comunicaciones entre nuestro PC y nuestro robot funcionan bien.

Conexionado

Al conexionado anterior vamos a añadir el cable USB que conecta el PC con la placa de Arduino.

- Pin SCL Display – Pin A5 Arduino
- Pin SDA Display – Pin A4 Arduino
- Pin VCC Display – Cualquier Pin VCC Arduino
- Pin GND Display → Cualquier Pin GND Arduino
- USB PC→ USB Arduino

En Bitbloq abriremos un nuevo proyecto y conectaremos el LCD y el USB como aparece en la Figura 1.2.21-8.

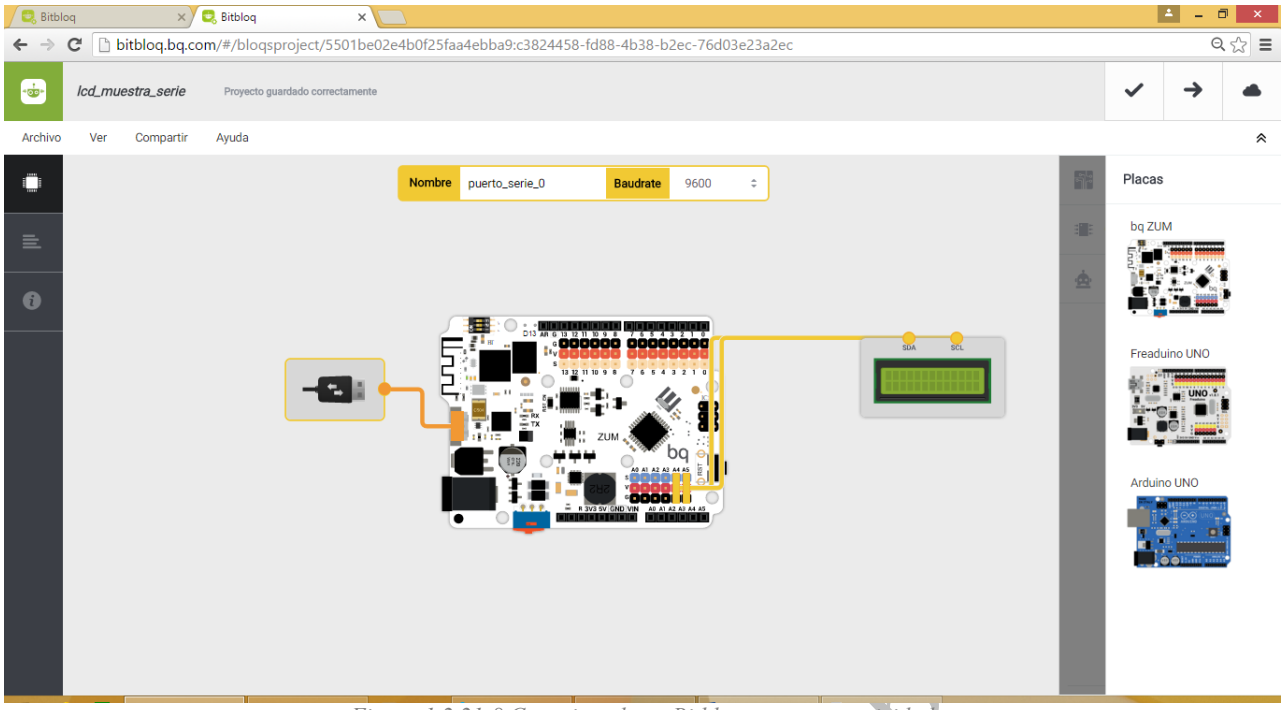
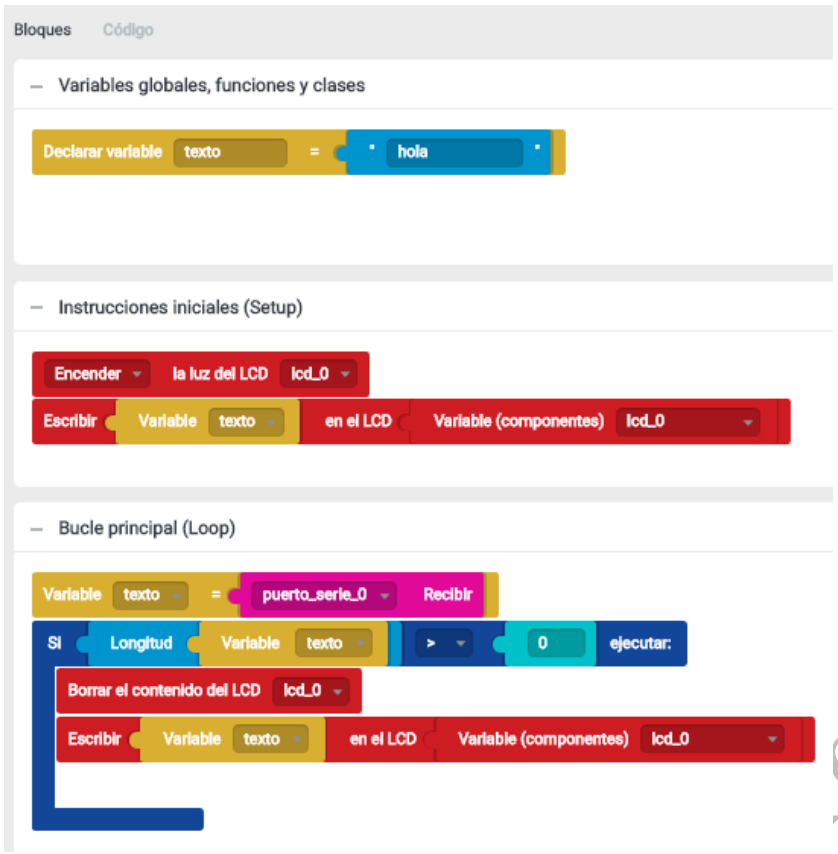


Figura 1.2.21-8 Conexionado en Bitbloq para esta actividad

Programación

Vamos a realizar un programa por bloques que reenvíe lo que llega por el puerto serie al LCD. Para ello utilizaremos una variable de texto en donde guardaremos todo lo que se recibe por el puerto serie Si esta variable no está vacía, es decir, tiene almacenado uno o más caracteres entonces enviaremos su contenido por el bus I2c.



Declaración de la variable *texto* donde almacenaremos los caracteres que llegan por el puerto serie

Encendemos la retroilumniacion del LCD y mostramos HOLA (contenido hasta este momento en la variable *texto*)

Leemos del puerto serie y lo almacenamos en *texto* Si el número de caracteres de esa variable (longitud) es mayor que cero entonces limpiamos el lcd y escribimos el contenido de *texto*.

El código en el lenguaje Arduino es el siguiente:

```
/***  Included libraries  ***/
#include <Wire.h>
#include <BitbloqLiquidCrystal.h>
#include <SoftwareSerial.h>
#include <BitbloqSoftwareSerial.h>


/***  Global variables and function definition  ***/
LiquidCrystal lcd_0(0);
bqSoftwareSerial puerto_serie_0(0, 1, 9600);
String texto = "hola ";

/***  Setup  ***/
void setup() {
    lcd_0.begin(16, 2);
    lcd_0.clear();
    lcd_0.setBacklight(HIGH);
    lcd_0.print(texto);
}

/***  Loop  ***/
void loop() {
    texto = puerto_serie_0.readString();
    if (texto.length() > 0) {
        lcd_0.clear();
        lcd_0.print(texto);
    }
}
```

1.2.21.3. Muestra el valor de una señal analógica

Los displays son muy útiles para mostrar información del estado interno de un robot sin que sea necesario conectarlo a un PC u otro dispositivo. En esta actividad leeremos el valor de un potenciómetro y lo mostraremos por el display.

Conexionado

Al conexionado de la actividad anterior vamos a añadir un potenciómetro, que como recordaremos de actividades anteriores, es un sensor analógico.

- Pin SCL Display – Pin A5 Arduino
- Pin SDA Display – Pin A4 Arduino
- Pin VCC Display – Cualquier Pin VCC Arduino
- Pin GND Display → Cualquier Pin GND Arduino
- USB PC→ USB Arduino
- Potenciómetro→ PIN A1 Arduino

Abriremos bitbloq y generaremos un proyecto con los componentes y conexionado anteriormenete indicado (véase Figura 1.2.21-9).

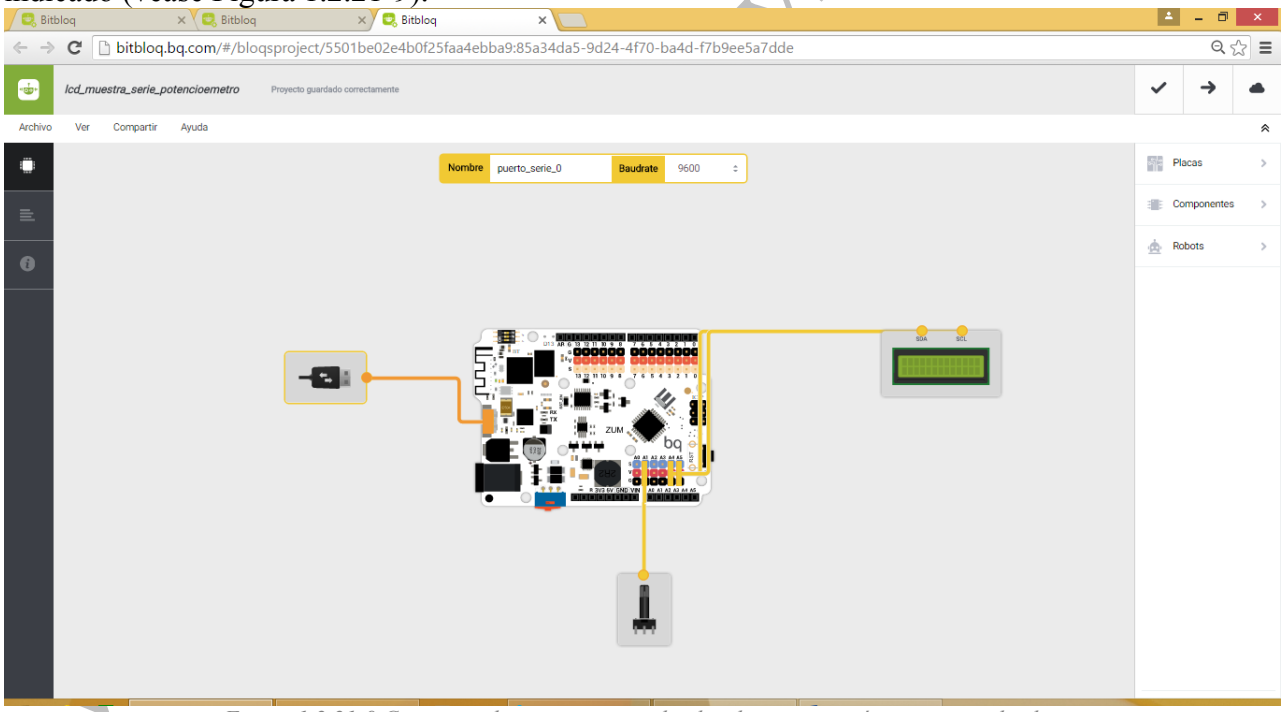
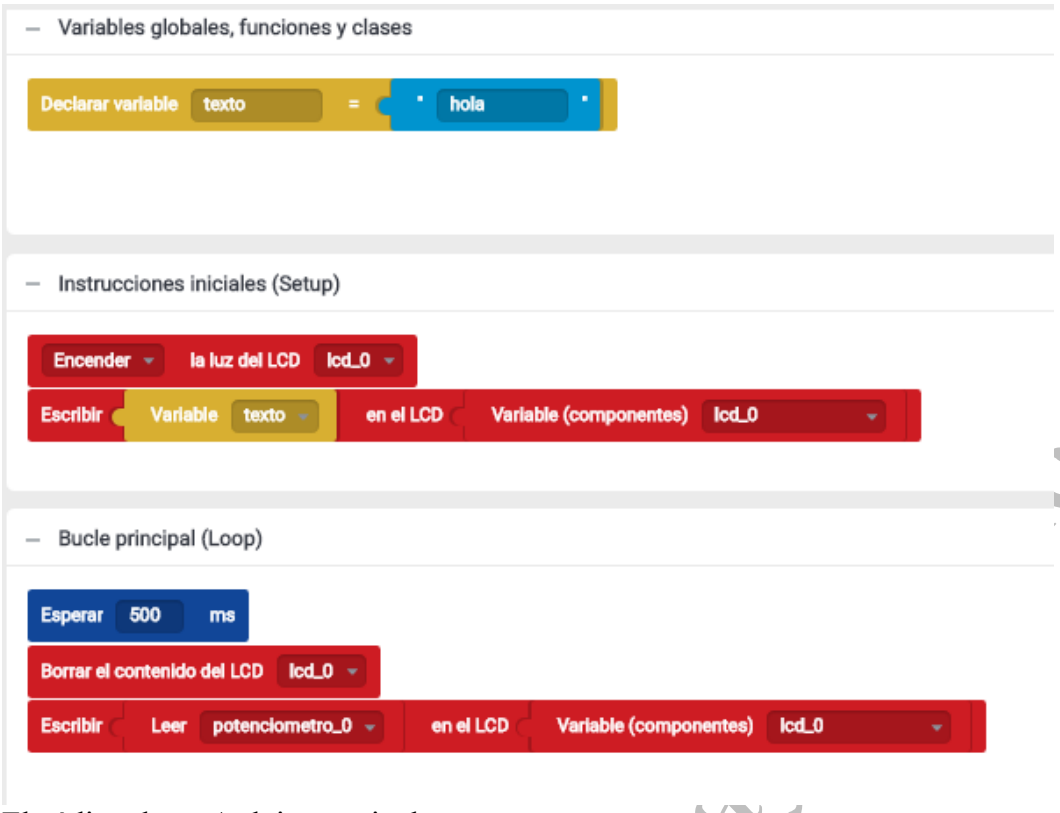


Figura 1.2.21-9 Conexionado para mostrar el valor de un potenciómetro en un display

Programación

En este caso deberemos tendremos una variable donde almacenaremos el valor del potenciómetro. Simplemente lo que haremos, como vemos en código de bloques siguiente, es ir leyendo del potenciómetro y escribiendo el valor en el LCD (como siempre habrá que borrar antes de escribir para que no se superponga nada).



El código de en Arduino equivalente es:

```
/***  Included libraries  ***/
#include <Wire.h>
#include <BitbloqLiquidCrystal.h>
#include <SoftwareSerial.h>
#include <BitbloqSoftwareSerial.h>


/***  Global variables and function definition  ***/
LiquidCrystal lcd_0(0);
int potenciometro_0 = A1;
bqSoftwareSerial puerto_serie_0(0, 1, 9600);
String texto = "hola ";

/***  Setup  ***/
void setup() {
    lcd_0.begin(16, 2);
    lcd_0.clear();
    pinMode(potenciometro_0, INPUT);
    lcd_0.setBacklight(HIGH);
    lcd_0.print(texto);
}

/***  Loop  ***/
void loop() {
    delay(500);
    lcd_0.clear();
    lcd_0.print(analogRead(A1));
}
```