CURSO: MICROCONTROLADORES

UNIDAD 4: COMUNICACIÓN SERIE- ARDUINO

PROFESOR: JORGE ANTONIO POLANÍA

Un receptor asíncrono / transmisor universal (UART) es un bloque de circuitería responsable de implementar la comunicación serie. En esencia, la UART actúa como intermediario entre las interfaces paralelo y serie. En un extremo de la UART es un bus de ocho o lo que las líneas de datos (además de algunos pines de control), en el otro es los dos cables de serie - RX y TX.

El UART es el responsable del envío y recepción de datos en serie. Por el lado de transmisión, una UART debe crear el paquete de datos - añadiendo sincronización y bits de paridad - y enviar ese paquete por la línea TX con sincronización precisa (según la velocidad de transmisión). En el extremo de recepción, el UART tiene que probar la línea RX a tasas de acuerdo a la velocidad de transmisión y esperar los datos.

Las placas Arduino tienen al menos un puerto serie (también conocido como un UART o USART). Se comunica con los pines digitales 0 (RX) y 1 (TX), así como con el computador a través de USB. Por lo tanto, si utiliza estas funciones, no se puede también utilizar los pines 0 y 1 para la entrada o salida digital.

FUNCIONES

Serial.begin(tasa): Inicializa el puerto serie y asigna la tasa de baudios para la transmisión de datos serie. La típica tasa de baudios para comunicarse con el computador es 9600 aunque también soporta otras velocidades. Se puede utilizar el monitor incorporado de serie del entorno Arduino para comunicarse con una placa Arduino. En la ventana de edición:

Herramientas-->Monitor serie

seleccione la misma velocidad de transmisión utilizada en la llamada para comenzar (). Se utiliza para la comunicación entre la placa Arduino y un computador u otros dispositivos.

Nota: Cuando se usa la comunicación serie, los pines digitales 0 (Rx) y 1 (Tx) no pueden ser usados al mismo tiempo.

Serial.print(dato), Serial.println (): Envían datos por el puerto serie o el valor de una variable. Println finaliza una línea cuando ha terminado de transmitir.

Serial.available(). Devuelve el número de bytes a la espera de leerse.

Serial.read (). Devuelve un byte leído por el puerto serie, -1 si no hay datos.

Serial.write (). Escribe datos por el puerto serie.

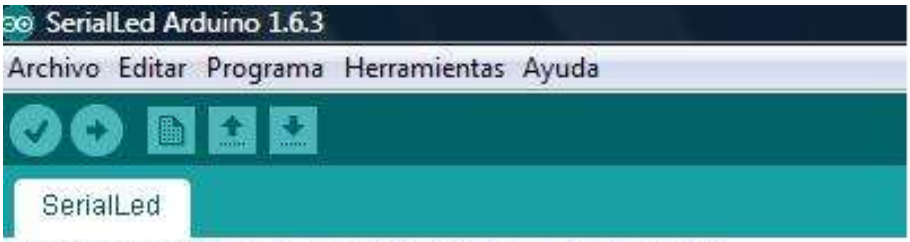
El objetivo de la siguiente práctica es adquirir habilidades para programar el Arduino Uno realizando algunas experiencias de comunicación serie de este sistema de desarrollo para el microcontrolador ATMEGA328 de ATMEL.

EQUIPO Y MATERIAL NECESARIO

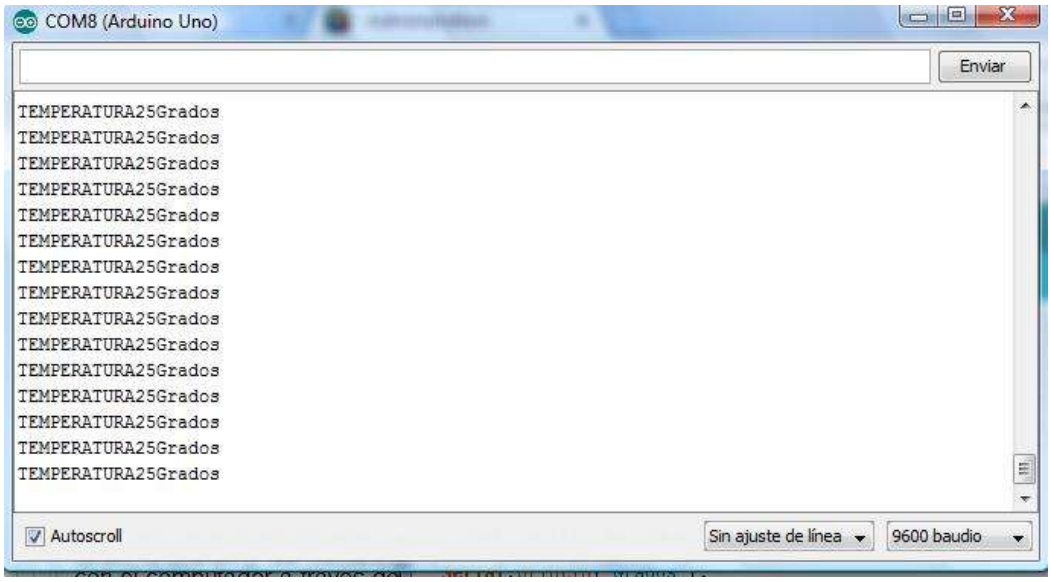
- Un computador
- Placa Arduino Uno
- Cable de conexión para usb al arduino
- Protoboard
- Un LED
- Un sensor de temperatura LM35
- Un motor DC de 6V
- Un teclado matricial 4x4
- Una resistencia a 1/4W de 330Ω
- 1 potenciómetro lineal de 10KΩ
- Conectores

1. MONITOR SERIE DEL ARDUINO

En el siguiente ejemplo se va a leer la temperatura que mide el sensor LM35 en el Monitor serie del Arduino. Se usa un led en pin 4 para indicar el control de temperatura y el sensor en la entrada análoga A0.



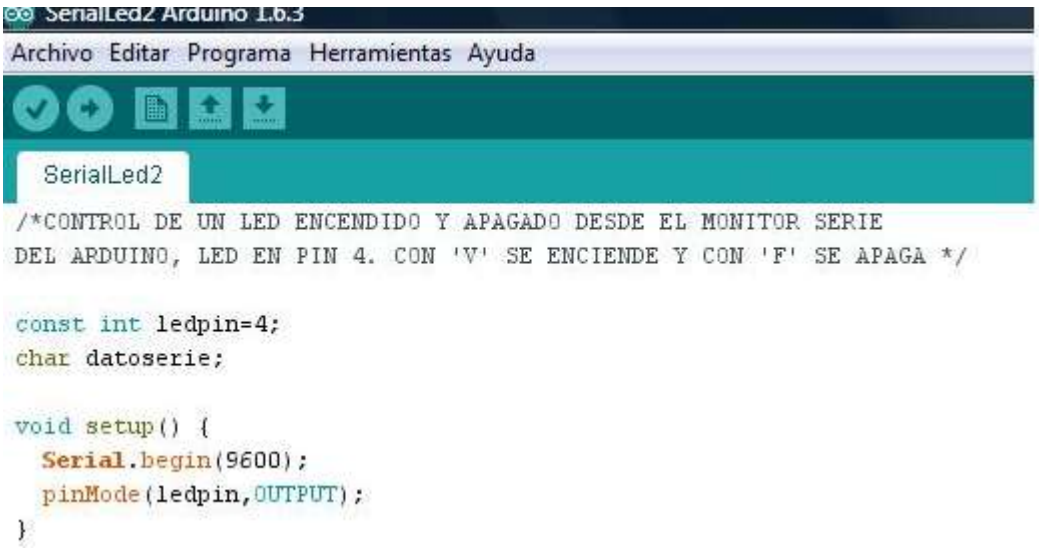

```
/*LEER TEMPERATURA DEL SENSOR LM35 EN MONITOR
SERIE DEL ARDUINO, SENSOR EN PIN A0 Y LED EN PIN 4 */

const int sensor=0;
const int ledpin=4;
long temp;
long valorsensor;

void setup() {
  Serial.begin(9600);
  pinMode(ledpin,OUTPUT);
}

void loop() {
  valorsensor=(analogRead(sensor))*(5000/1023);
  temp=valorsensor/10;
  Serial.print("TEMPERATURA");
  Serial.print(temp);
  Serial.println("Grados");
  delay(1000);
  if(temp>=26){
    digitalWrite(ledpin,HIGH);
  }
  else{
    digitalWrite(ledpin,LOW);
  }

}
```



Observar el monitor y comprobar que al pasar la temperatura de 26 grados el led se enciende y al bajar se apaga.

2. CONTROL DE UN LED POR EL MONITOR SERIE

El ejemplo consiste en encender o apagar un led colocado en el pin 4 desde el Monitor serie del Arduino, de tal forma que al enviarle un carácter 'V' se enciende y al enviarle el carácter 'F' se apaga.



```
void loop() {
  //comprobar si hay dato serie de entrada
  if(Serial.available()>0){
    datoserie=Serial.read();
    if(datoserie=='V'){
      digitalWrite(ledpin,HIGH);
    }
    if(datoserie=='F'){
      digitalWrite(ledpin,LOW);
    }
  }
}
```

3. MANEJO DE STRING

Cuando se trabaja con el puerto serie es muy común transmitir textos. Arduino utiliza la función sprintf () que tiene la siguiente sintaxis:

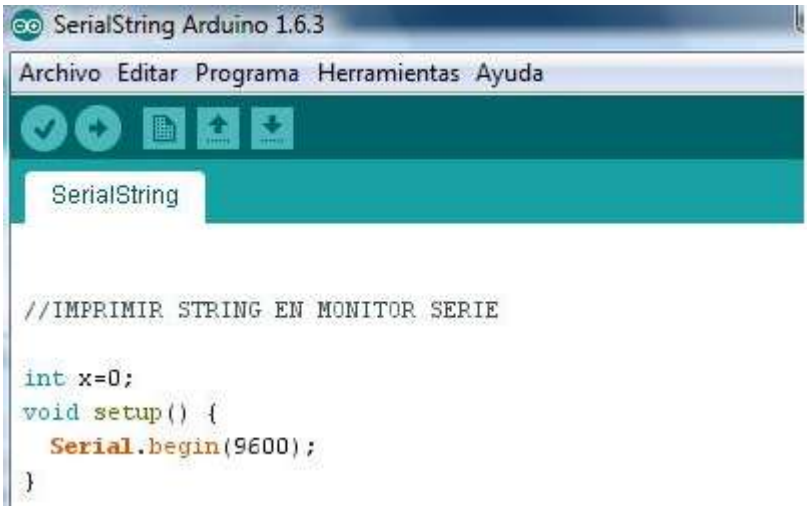
sprintf(stringResultado, stringFormato, argumentos);

StringResultado, es un arreglo char donde se almacenará el resultado.

StringFormato, especifica el formato, comienza con % y una letra para indicar el tipo (%d para enteros, %x para hexadecimal, %c para char).

Argumentos, lista de variables.

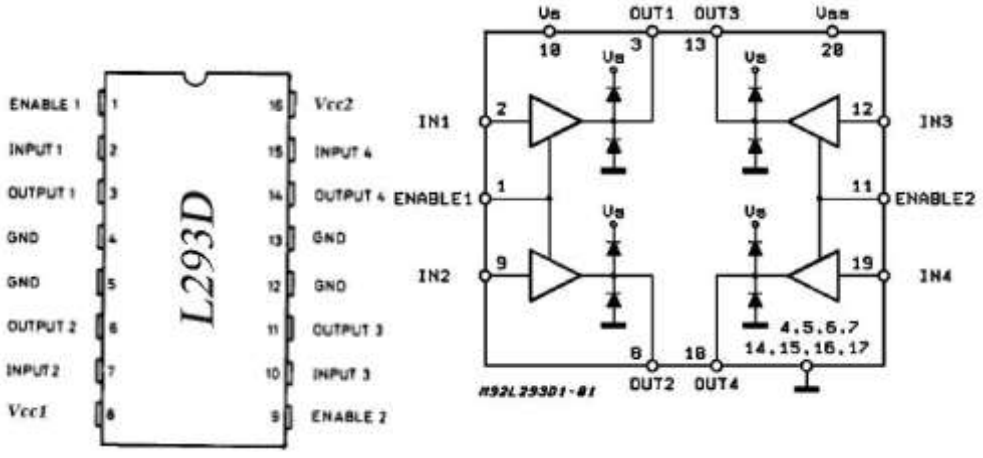
EJEMPLO:



```
void loop() {
   char texto[20];
   sprintf(texto,"Variable: %d\n",x);
   Serial.println(texto); //imprime Variable: 0
   x++;
   delay(1000);
}

    }
    if(datoserie=='F'){
      digitalWrite(ledpin,LOW);
    }
  }
}
```

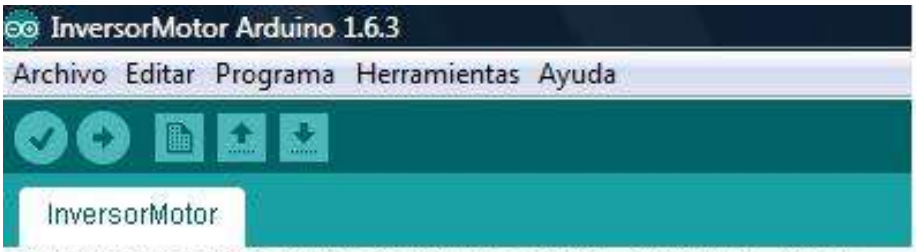
4. INVERTIR GIRO DE UN MOTOR DC

Invertir el giro de un motor de corriente continua usando el driver L293D para conexión del motor que soporta hasta 600 mA por canal. Se utiliza como control de giro un potenciómetro de 10K colocado en la entrada análoga A0 del conversor A/D de tal forma que cuando lea de 0 a 500 gire contrario al reloj, de 500 a 580 se pare y una lectura de más de 580 gire en al sentido del reloj (recuerde que el conversor A/D traduce la lectura análoga de 0 a 1023). A continuación se dan las características del driver L293D.



IN1	IN2	Function
High	High	Turn Anti-clockwise (Reverse)
High	Low	Turn clockwise (Forward)
High	High	Stop
High	Low	Stop
Low	X	Stop

A continuación se presenta el programa (sketch) para el Arduino.




```
/*INVERTIR GIRO DE UN MOTOR DC USANDO PUENTE H L293D
UTILIZANDO UN POT DE 10K CONECTADO EN EL PIN A0*/

const int pot=0;  //pot conectado a A0
const int pinmotor1=6;  // IN1 del L293 al pin 6
const int pinmotor2=5;  // IN2 del L293 al pin 5
int valorpot;

void setup() {
  pinMode(pinmotor1,OUTPUT);
  pinMode(pinmotor2,OUTPUT);
  //motor apagado inicialmente
  digitalWrite(pinmotor1,LOW);
  digitalWrite(pinmotor2,LOW);
}
```

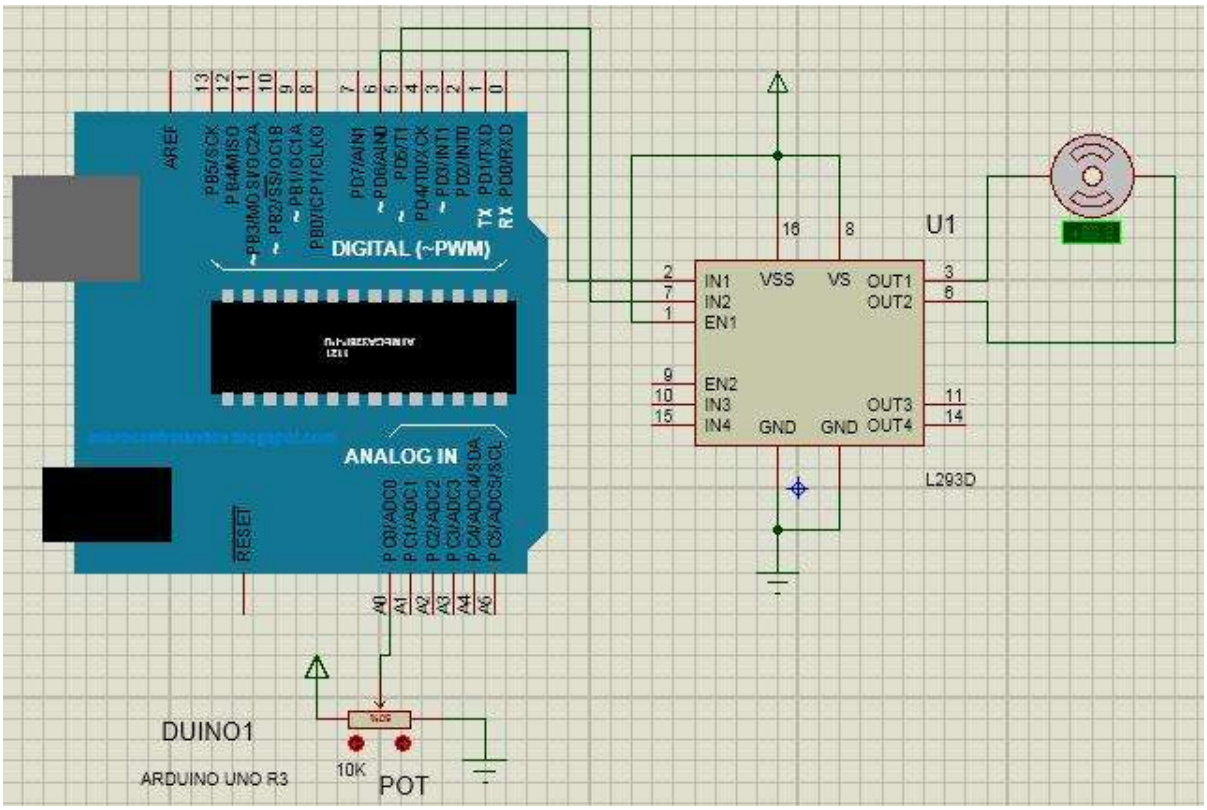
```
void loop() {
  valorpot=analogRead(pot);
  if(valorpot<500){
    //giro contrario al reloj
    digitalWrite(pinmotor1,HIGH);
    digitalWrite(pinmotor2,LOW);
  }
  if(valorpot>580){
    //giro en sentido del reloj
    digitalWrite(pinmotor1,LOW);
    digitalWrite(pinmotor2,HIGH);
  }
    else{
      //motor apagado pot entre 500 y 580
      digitalWrite(pinmotor1,LOW);
      digitalWrite(pinmotor2,LOW);
    }
  }
```

La implementación hardware se presenta a continuación. Se ha simulado en Proteus. Para descargar Proteus Versión 8 en este video está la forma de descargarlo e instalarlo.

https://www.youtube.com/watch?v=ls2MJla_gno

En este enlace se encuentra la forma de simular Arduino con Proteus.

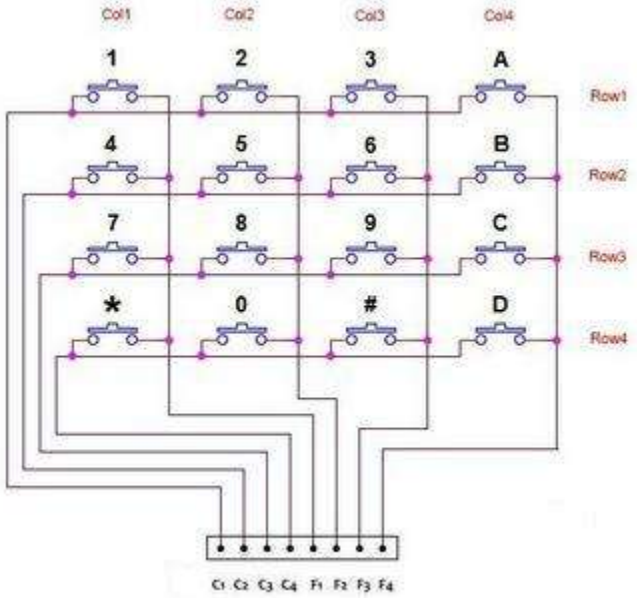
Simulación de Arduino con Proteus



5. LEER TECLADO MATRICIAL

El Arduino Uno no tiene la librería del teclado hay que bajarla copiarla y pegarla al a la carpeta C --> archivos de programa-->arduino-->libraries el archivo Keypad que está en siguiente enlace.

http://playground.arduino.cc/code/keypad

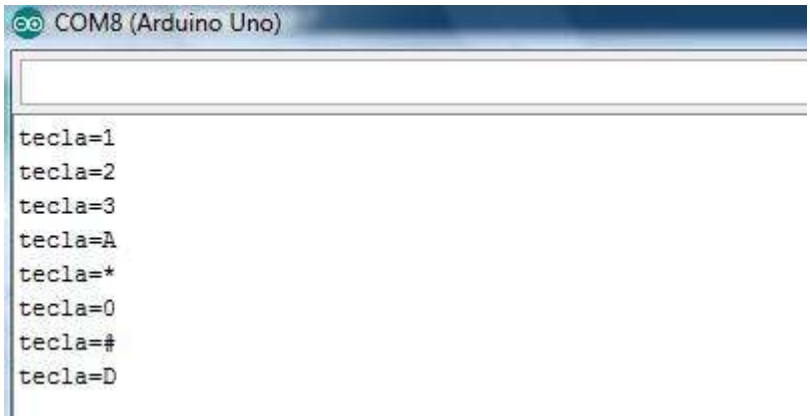


El siguiente ejemplo se lee el teclado por puerto serie:

```
//programa-->include library-->Keypad
#include <Keypad.h>
const byte rows=4;
const byte cols=4;
//configuración del teclado
char keys [rows][cols]={
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

byte rowspin[rows]={9,8,7,6};  //pines a las filas f1 f2 f3 f4
byte colspin[cols]={5,4,3,2};  //pines a las columnas c1 c2 c3 c4
Keypad keypad=Keypad(makeKeymap(keys),rowspin,colspin,rows,cols);

void setup() {
  Serial.begin(9600);
}
```

```
void loop() {
  char tecla=keypad.getKey();
  if(tecla){
    Serial.print("tecla=");
    Serial.println(tecla);
  }
  delay(10);
}
```

Comprobación de la configuración del teclado en el Monitor serie del Arduino.



HARDWARE IMPLEMENTADO.