datos

Gutierrez Cuautle Cruz Guadalupe
No. control 125872
fecha:25/05/2020

lcd

```
#include <LiquidCryst-
al.h>: permite invocar
la librería para el
manejo del lcd.
LiquidCrystal lcd(12,
11, 5, 4, 3, 2): Se
realiza la asignación de
los pines al lcd.
lcd.begin(16, 2):
Permite configurar el
tamaño del lcd.
lcd.print("Hola..."):
permite imprimir un
texto en el lcd.
lcd.clear(): Permite
limpiar lo escrito en el
lcd.
lcd.setCursor(0,1):
Permite posicionar el
cursor en una celda del
lcd en especifico.
}
```

El el módulo Arduino LCD KeyPad Shield/Teclado LCD está diseñado para ser compatible con las placas Arduino, y para proporcionar una interfaz fácil de usar que permite hacer menús, selecciones etc. Consta de un 2 filas de 16 caracteres blancos con la retroiluminación LCD azul. El teclado se compone de 5 teclas

sensor de temperatura LM35

```
const int sensorPin= A0;
void setup()
{
Serial.begin(9600);
}
void loop()
{
int value = analogRead-
(sensorPin);
float millivolts =
(value / 1023.0) * 5000;
float celsius =
millivolts / 10;
Serial.print(celsius);
Serial.println(" C");
delay(1000);
}
```

El LM35 es un circuito electrónico sensor que puede medir temperatura. Su salida es analógica, es decir, te proporciona un voltaje proporcional a la temperatura. El sensor tiene un rango desde −55°C a 150°C. Su popularidad se debe a la facilidad con la que se puede medir la temperatura

teclado 4x4

```
#include <Keypad.h>
const byte numRows= 4;
// Tiene 4 filas
const byte numCols= 4;
// Tiene 4 columnas
char keymap[numRows][nu-
mCols]=
{
{'1', '2', '3', 'A'},
{'4', '5', '6', 'B'},
{'7', '8', '9', 'C'},
{'*', '0', '#', 'D'}
};
byte rowPins[numRows] =
{9,8,7,6}; // Estos
terminales del Arduino
corresponden a Filas
byte colPins[numCols]=
{5,4,3,2}; // Estos
terminales del Arduino
corresponden a Columnas
void setup()
{
Serial.begin(9600);
}
```

Puede ser conectado a cualquier microcontrolador o tarjetas de desarrollo como Arduino. El teclado matricial 4x4 está formado por una matriz de pulsadores dispuestos en filas (L1, L2, L3, L4) y columnas (C1, C2, C3, C4), con la intención de reducir el número de pines necesarios para su conexión.

motor paso a paso

```
#include <Stepper.h>
#define STEPS 2048
Stepper stepper(STEPS,
8, 9, 10, 11);
void setup() {
 stepper.setSpeed(10);
}
void loop() {
 stepper.step(2048);
}
```

Un motor paso a paso es un dispositivo electromecánico que convierte una serie de pulsos eléctricos en desplazamientos angulares, lo que significa que es capaz de girar una cantidad de grados (paso o medio paso) dependiendo de sus entradas de control.
Este paso puede variar desde 90° hasta pequeños movimientos de 180°

driver

```
#include <Stepper.h>
//Importamos la librería
para controlar motores
paso a paso
```

driver (cont)

```
#define STEPS 200
//Ponemos el número de pasos que necesita para dar una vuelta. 200 en nuestro caso
// Ponemos nombre al motor, el número de pasos y los pins de control
Stepper stepper(STEPS, 8, 9, 10, 11); //Stepper nombre motor (número de pasos por vuelta, pins de control)
void setup()
{
  // Velocidad del motor en RPM
  stepper.setSpeed(100);
}
void loop()
{
  //Girar una vuelta entera en un sentido
  stepper.step(200);
  delay(500); //Pequeña pausa
  //Girar una vuelta entera en sentido contrario
  stepper.step(-200);
  delay(500); //Pequeña pausa
}
```

Un motor paso a paso unipolar es más sencillo que controlar. Utilizaremos el integrado ULN2803 que es un array de 8 transistores tipo Darlington capaz de soportar cargas de hasta 500mA (datasheet). Conectaremos los cuatro pins del Arduino a las entradas del ULN2803 y las salidas de este a las bobinas. Los comunes a 12V

servomotor

```
// Incluímos la librería para poder controlar el servo
#include <Servo.h>

// Declaramos la variable para controlar el servo
Servo servoMotor;

void setup() {
  // Iniciamos el monitor serie para mostrar el resultado
  Serial.begin(9600);

  // Iniciamos el servo para que empiece a trabajar con el pin 9
  servoMotor.attach(9);
}

void loop() {

  // Desplazamos a la posición 0º
  servoMotor.write(0);
  // Esperamos 1 segundo
  delay(1000);

  // Desplazamos a la posición 90º
  servoMotor.write(90);
  // Esperamos 1 segundo
  delay(1000);

  // Desplazamos a la posición 180º
  servoMotor.write(180);
  // Esperamos 1 segundo
  delay(1000);
}
```

permite mantener una posición que indiquemos, siempre que esté dentro del rango de operación del propio dispositivo. Por otro lado nos permite controlar la velocidad de giro, podemos hacer que antes de que se mueva a la siguiente posición espere un tiempo.

ultrasónico HC-SR04

```
const int Trigger = 2;
//Pin digital 2 para el Trigger del sensor
const int Echo = 3;
//Pin digital 3 para el echo del sensor
void setup() {
Serial.begin(9600);//iniciailzamos la comunicación
pinMode(Trigger, OUTPUT); //pin como salida
pinMode(Echo, INPUT);
//pin como entrada
digitalWrite(Trigger, LOW);//Inicializamos el pin con 0
}
```

El sensor de distancia ultrasónico HC-SR04 utiliza los ultrasonidos para determinar la distancia a un objeto. Ofrece una excelente precisión y lecturas estables en un paquete fácil de usar. La operación no se ve afectada por la luz solar o el material negro como los sensores de infrarojos, aunque los materiales blandos como las telas pueden ser difíciles de detectar.

Dispositivos Bluetooth

```
#include
<SoftwareSerial.h> //
Incluimos la librería
SoftwareSerial
SoftwareSerial
BT(10,11); // Definimos
los pines RX y TX del
Arduino conectados al
Bluetooth

void setup()
{
  BT.begin(9600); //
Inicializamos el puerto
serie BT (Para Modo AT
2)
  Serial.begin(9600); //
Inicializamos el puerto
serie
}

void loop()
{
  if(BT.available()) //
Si llega un dato por el
puerto BT se envía al
monitor serial
  {
    Serial.write(BT.re-
ad());
  }
```

Dispositivos Bluetooth (cont)

```
  if(Serial.available())
// Si llega un dato por
el monitor serial se
envía al puerto BT
  {
     BT.write(Serial.re-
ad());
  }
}
```

Funciona como dispositivo maestro y esclavo bluetooth
Configurable mediante comandos AT
Bluetooth V2.0+EDR
Frecuencia de operación: 2.4 GHz Banda ISM
Modulación: GFSK (Gaussian Frequency Shift Keying)
Potencia de transmisión: <=4dBm, Class 2
Sensibilidad: <=-84dBm @ 0.1% BER
Seguridad: Autenticación y encriptación
Perfiles Bluetooth: Puerto serie bluetooth.
Distancia de hasta 10 metros en condiciones óptimas

IoT

```
#include
<SoftwareSerial.h>
SoftwareSerial mySeri-
al(10, 11); // RX, TX
void setup() {
  // Open serial commun-
ications and wait for
port to open:
  Serial.begin(115200);
  while (!Serial) {
    ; // wait for serial
port to connect. Needed
for native USB port only
  }
  Serial.println("Goo-
dnight moon!");
  // set the data rate
for the SoftwareSerial
port
  mySerial.begin(115-
200);
  mySerial.println("H-
ello, world?");
}
void loop() { // run
over and over
  if (mySerial.avail-
able()) {
    Serial.write(mySer-
ial.read());
  }
  if (Serial.availab-
le()) {
    mySerial.write(Ser-
ial.read());
  }
}
```

Internet de las cosas (en inglés Internet of things, abreviado IoT) es un concepto que se refiere a la interconexión digital de objetos cotidianos con Internet un elemento que nos permite de forma sencilla y económica conectar cualquier cosa a Internet. Con un Arduino y un sencillo módulo ethernet o wifi podemos conectar a Internet sensores para informar, controlar motores o bombillas desde cualquier parte del mundo o mandar un SMS o email cada vez que se abra la puerta de casa.