DEPARTAMENTO DE INGENIERÍA E INVESTIGACIONES TECNOLÓGICAS

SISTEMAS OPERATIVOS AVANZADOS

TRABAJO PRÁCTICO IoT – Android MoeBot

JEFE DE CÁTEDRA
GRACIELA DE LUCA

PROFESORES
ESTEBAN CARNUCCIO
GERARDO GARCIA
WALDO VALIENTE
MARIANO VOLKER

2° CUATRIMESTRE - AÑO 2017

APELLIDO Y NOMBRE	DNI
Accattoli, Apolo	33477386
Alonso, Rodolfo	33474354
Maidana, Diego	36258785
Sánchez, Martín	33458911

Contenido

INTRODUCCIÓN

MoeBot es un proyecto que tiene como objetivo facilitar el delivery de los pedidos que se llevan a cabo en un bar/comercio. La idea principal es de dotar al manager de un negocio de bebidas de una herramienta que le permita automatizar el flujo de entrega de los pedidos a los consumidores de manera muy sencilla y rápida.

El proyecto está dividido en dos grandes partes: una aplicación MoeBot para Android y un robot basado en Arduino. Mediante la aplicación se puede ir cargando los pedidos de los consumidores a medida que vayan ordenando y una vez que estén preparados indicar al robot inicia el reparto.

El robot que se utiliza para el reparto es un seguidor de líneas, en este caso línea negra. Por lo que, a fines de esta experiencia, vamos a implementar un circuito compuesto por cinta aislante negra simulando el entorno de un bar, donde cada mesa va a ser una "estación" donde el robot llevará su pedido correspondiente. Como punto de partida, definiremos una "Zona de Abastecimiento", que es donde el robot se encontrará esperando a que se preparen los pedidos, hasta que reciba la orden de iniciar el reparto.

Es importante aclarar que el alcance de este proyecto generar la inteligencia detrás de la gestión de pedidos y no el reparto en sí. Por lo tanto, a fines prácticos toda la experiencia será realizada sin tener en cuenta factores como el peso a transportar y la carga de productos que el robot Arduino tiene que entregar a cada una de las "estaciones. Esto posiblemente será tomado en cuenta y trabajado en otra etapa del proyecto.

¿COMO FUNCIONA? PASO A PASO

Básicamente el robot MoeBot es un robot seguidor de línea basado en Arduino, por lo tanto, como su nombre indica, es un vehículo guiado automatizado, que sigue una línea visual incrustada en el suelo. La línea visual es la trayectoria en la que va el robot seguidor de línea y será una línea negra sobre una superficie blanca. Para esto se utilizan sensores de luz CNY70, que serán detallados en el apartado de Hardware.

El otro gran componente del proyecto es la aplicación MoeBot para Android. Mediante un sensor de Bluetooth HC-06 incorporado al robot, este será emparejado con el sensor de Bluetooth del dispositivo móvil donde se estará ejecutando la aplicación Android.

1- Ya con la aplicación descargada en instalada en el dispositivo móvil del usuario, el mismo inicia la aplicación MoeBot.
2- Una vez iniciada, procede a emparejar el dispositivo con el robot Arduino encargado del delivery de pedidos, mediante conexión Bluetooth. El roboto posee un sensor HC-06 para realizar esta comunicación con el sensor Bluetooth del dispositivo móvil.
3- Ya emparejados ambos dispositivos, el usuario procede a cargar los pedidos. Los datos que debe suministrar son: **N° de mesa, Tipo de Bebidas y Cantidad**.

ACLARACIÓN: La cantidad de pedidos que se van a cargar en la aplicación es determinada por el usuario. Cuando él lo crea conveniente, va a dar la orden para iniciar el reparto. No hay participación en este punto tanto de la aplicación como del robot Arduino.

Una vez completado el detalle del pedido, procede a guardar el mismo en la lista de pedidos.

4- Cuando se hayan cargado una cantidad determinada de pedidos, todos se hayan abastecido y el usuario lo considere apropiado, este va a descargar la lista de pedidos al robot Arduino, y esto será entendido como orden para iniciar el reparto. El robot Arduino recibirá una secuencia de mesas a entregar los pedidos. A esa secuencia de unos y ceros, el Arduino interpreta como “Hoja de Ruta” del recorrido que debe hacer. Por ejemplo, en un escenario con 4 “estaciones” (mesas), la gente de las mesas 2 y 3, piden unos tragos. Una vez cargados estos pedidos en la aplicación el robot recibe la orden para que salga a entregarlos, a través de una secuencia de ceros y unos, en este caso [0 1 1 0].

Esta secuencia viene ordenada de tal forma que la posición de cada número es equivalente al número de mesa. Por otro lado, cada uno de esos números representa si hay pedido (1) o no hay pedido (0). Para nuestro ejemplo, esto se interpretaría como que el robot tiene pedidos en la mesa 2 y 3. Tomaremos como escenario de referencia un circuito conformado por líneas negras similar al de la **Figura 1.**

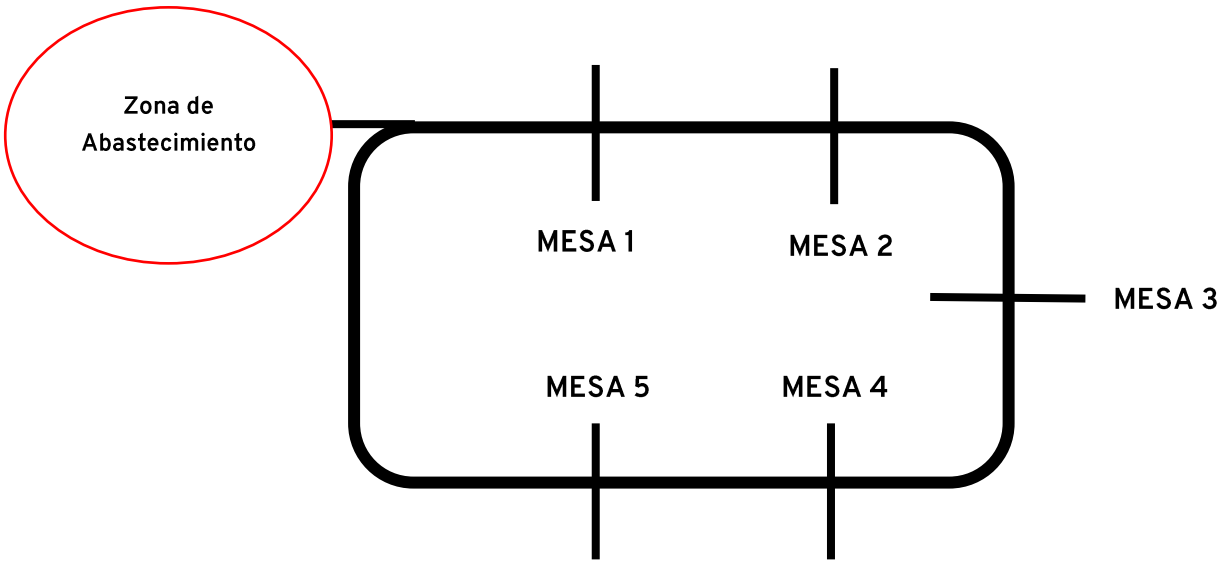


Figura 1 – Circuito de Reparto de Pedidos

5- El robot comienza a movilizarse siguiendo el circuito de líneas, y una vez que el detecta una estación (mesa) la cual tiene un pedido asignado, se frena y queda esperando a que se retiren los productos, por un lapso definido de antemano. Pasado este tiempo, se reinicia el recorrido.
6- Puede pasar que el sensor ultrasónico del robot detecte un obstáculo en medio del camino que le imposibilite seguir avanzando. Cuando esto suceda, comenzará a sonar una alarma (buzzer) avisando que necesita ayuda para quitar dicho obstáculo y proseguir con el recorrido. El lapso también es definido por el usuario de antemano.
7- Cuando el robot no tenga más pedidos que repartir, vuelve a la zona de abastecimiento, a la espera de seguir recibiendo nuevos pedidos.

HARDWARE - COMPONENTES

A continuación, se listan todos los componentes de Hardware que utilizamos para la construcción del robot de reparto de pedidos:

- **1 x Arduino Uno**: Placa con el microcontrolador ATmega328, 14 pines digitales de I/O (6 salidas PWM), 6 entradas análogas, 32k de memoria Flash y Reloj de 16MHz de velocidad.
- **1 x Sensor Ultrasónico HC-SR04**: Realiza un pulso y mide el tiempo que tarda en rebotar el eco para medir distancia. En nuestro proyecto se usa para detectar obstáculo frontal. Es un sensor que arroja un valor digital codificado por ancho de pulso.
- **3 x Sensor infrarrojo CNY70:** Contiene un diodo emisor de luz infrarroja y un receptor sensible que arroja una señal analógica indicando la detección o no de rebote de la emisión en el receptor. Los utilizamos en el proyecto como entrada digital para ir sensando la posición de la línea.
- **1 x Puente H doble L298:** Controla el giro de los motores. Recibe, además de las señales High o Low (Digitales) para cada motor, también una señal PWM por motor controlando su velocidad.
- **2 x Motor con caja reductora:** Realizan el giro de las ruedas, controlados por el puente H.
- **1 x Buzzer:** Speaker piezoeléctrico que realiza el sonido mediante la orden “tone” de Arduino que genera la frecuencia de onda deseada en el pin que alimenta dicho actuador. Se utiliza el mismo clock interno que para generar señales PWM.
- **1 x Módulo Bluetooth HC-06**, esclavo solamente. Espera el establecimiento de conexión y luego recibe y envía datos byte a byte de forma serial. Está configurado para transmitir a 9600 Baudios

Para la Alimentación:

- **4 x Pila AA** para las ruedas y el puente H.
- **1 s Batería de 9V** para la placa Arduino y el resto de los componentes (Que se alimentan a través del Arduino).

HARDWARE - DIAGRAMAS

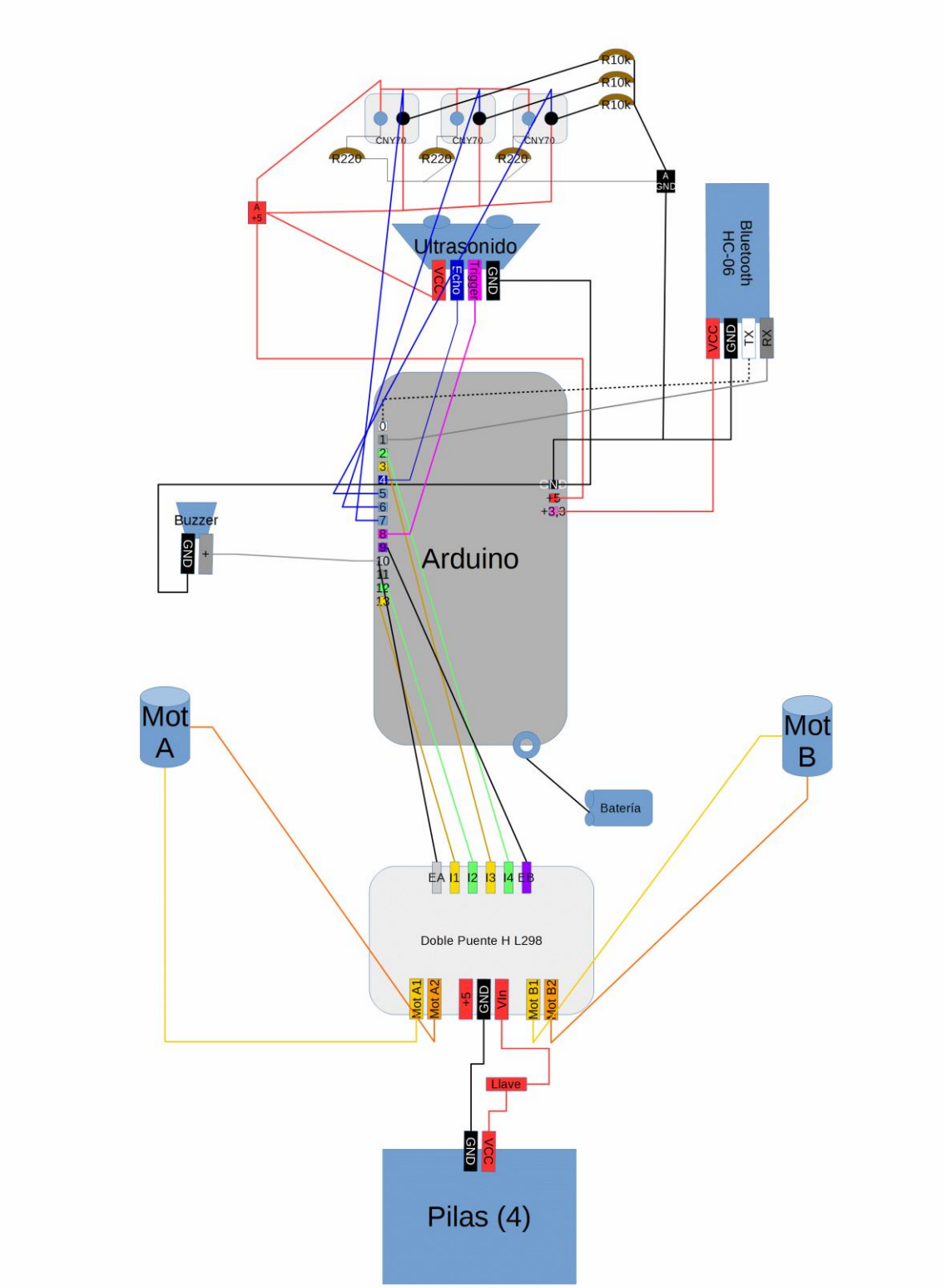


Figura 2 - Diagrama de Componentes

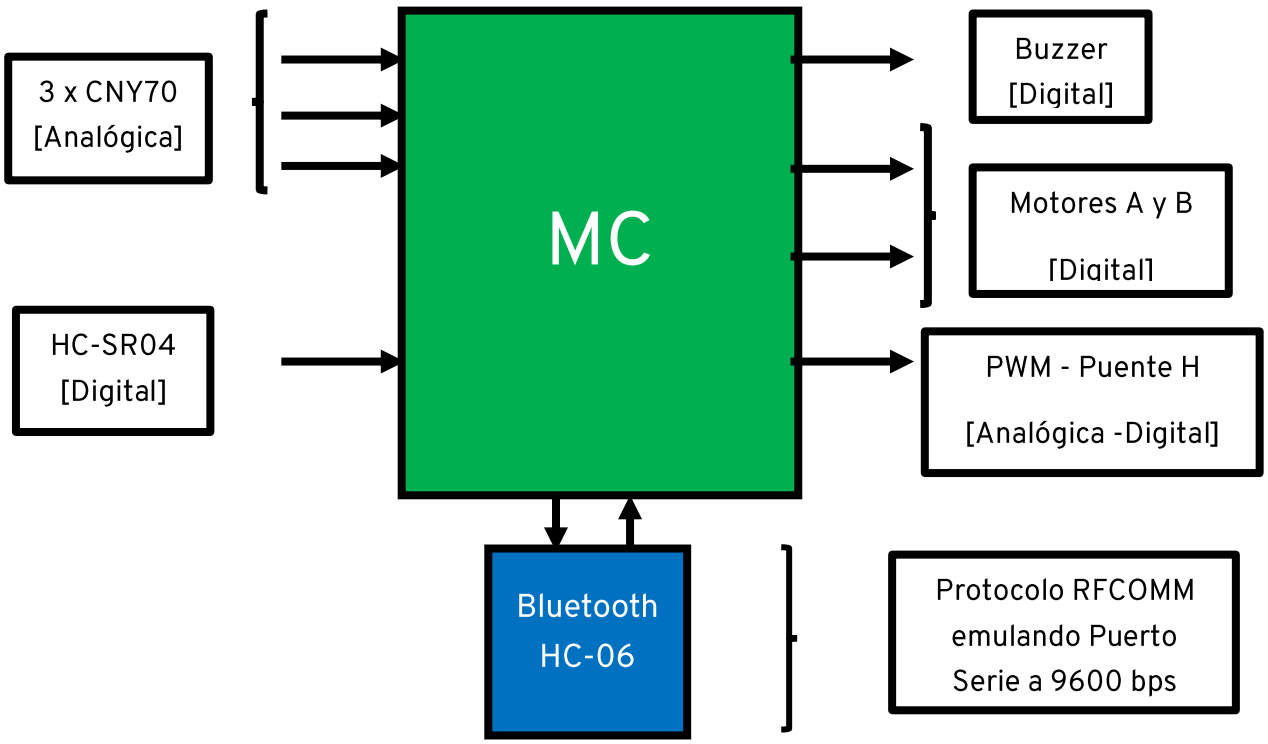


Figura 3 - Diagrama de Señales de Entrada y Salida

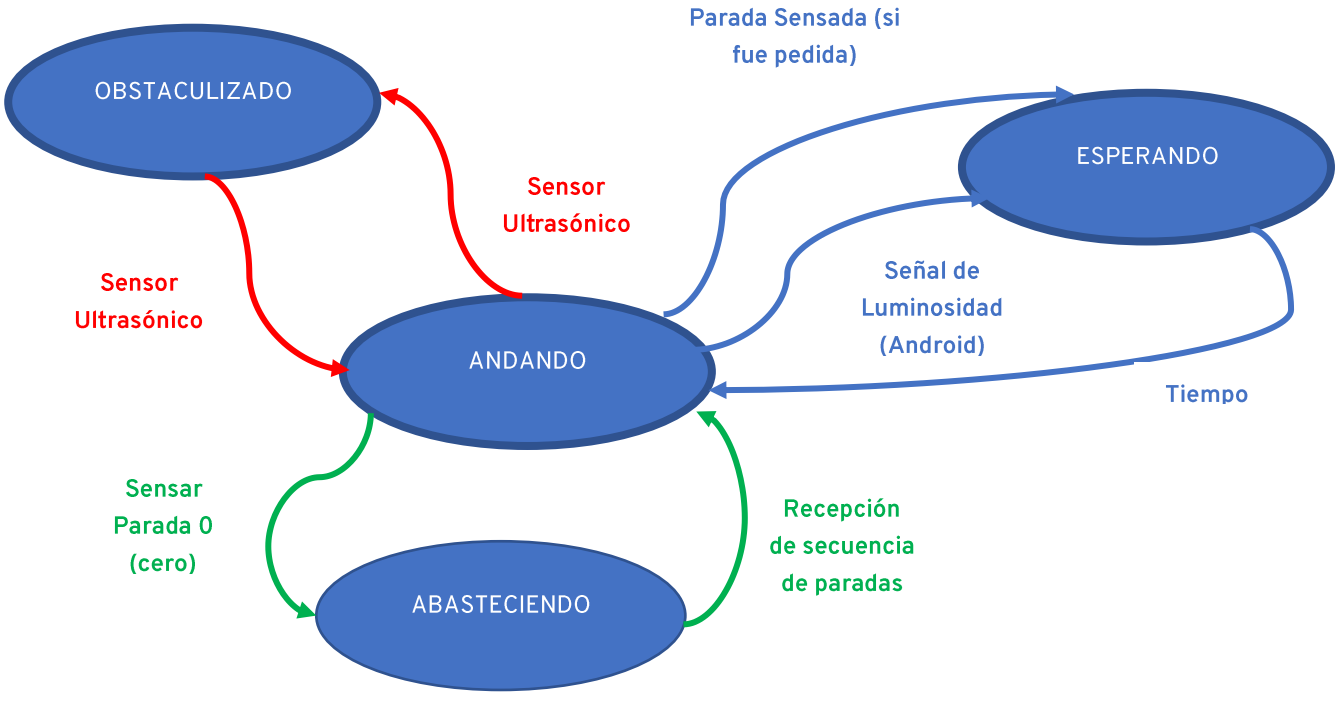


Figura 4 – Diagrama de Estados

APLICACIÓN ANDROID

En este apartado, haremos un repaso de las Activities pertenecientes la aplicación MoeBot de Android.

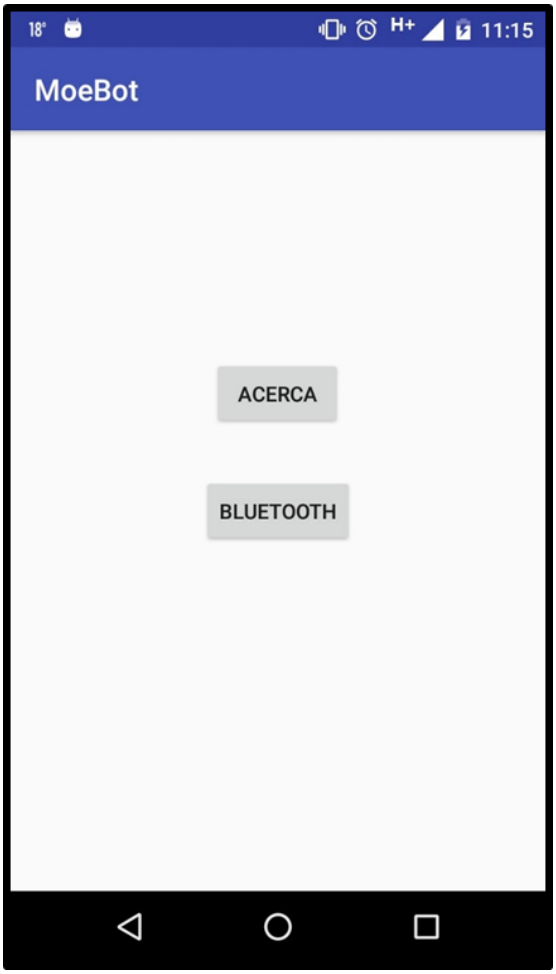
Una vez instalada MoeBot en un dispositivo móvil aparecerá un icono ilustrativo que hará referencia a la misma.

1) Activity Inicio

Esta actividad es la primera con la que nos toparemos luego de haber iniciado la aplicación Android.

2) Activity Menu

Luego de haberse completado la carga de la aplicación, encontraremos dos opciones:



ACERCA → Aquí se encuentra a modo informativo los datos del equipo detrás del proyecto MoeBot, el objetivo del mismo y la versión de la aplicación.

BLUETOOTH → Esta opción desplegará la lista de los dispositivos Bluetooth que se encuentran disponibles para emparejar, entre los cuales encontraremos el robot Arduino.

3) Activity Acerca

Detalles mencionados en la opción “Acerca” del punto anterior.

4) Activity Opciones Bluetooth

Si se elige la opción Bluetooth de la Activity Menu, se presentan tres opciones:

DESACTIVAR → Desactiva el sensor de Bluetooth del dispositivo móvil, para proceder a desemparejar.

DISPOSITIVOS EMPAREJADOS → Nos muestra una lista con los dispositivos que ya están emparejados (Activity Lista Dispositivos)

BUSCAR DISPOSITIVOS → Comienza la búsqueda de dispositivos con Bluetooth, para proceder a emparejar.

5) Activity Lista Dispositivos

Se muestra la lista de dispositivos emparejados, tal cual se menciona en el detalle de la opción Dipositivos Emparejados, del punto anterior.

6) Activity Selección

Aquí podremos cargar los pedidos a medida que van llegando. Para completar cada uno de los pedidos deberemos indicar a que Estación corresponde (Mesa), el tipo y la cantidad de bebidas. Una vez seleccionado estos detalles, tendremos otras tres opciones:

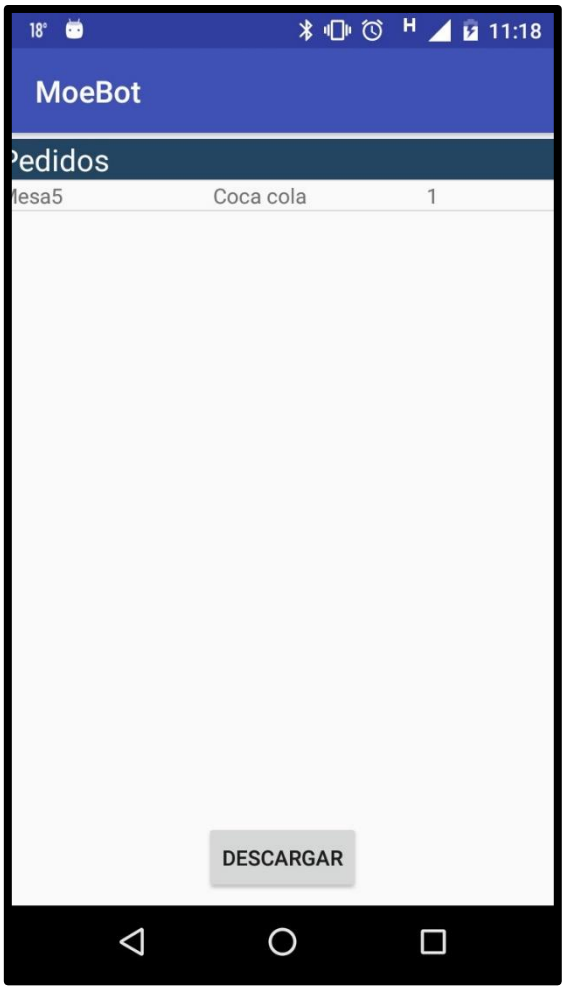
CARGAR → Genera el pedido con los cambios realizados y lo guarda en la lista de pedidos. Esto también se puede realizar mediante el sensor Shake del dispositivo móvil.

LISTA DE PEDIDOS → Nos muestra la lista con todos los pedidos realizados. Sirve como "Hoja de Ruta" para el reparto del robot Arduino.

LIMPIAR LISTA → Limpia el buffer de pedidos. Se recomienda realizarlo entre entrega y entrega.

7) Activity Lista de Pedidos

Aquí visualizaremos la lista con todos los pedidos realizados y que están esperando por ser entregados. Una vez que se abastezcan los mismos, y si el usuario lo considera pertinente, se le tiene que dar la orden al robot Arduino, para que inicie con el reparto. Esto se hace mediante la opción **DESCARGAR.**


18°
11:18
MoeBot
edidos
esa5
Coca cola
1
DESCARGAR

BIBLIOGRAFÍA - CONSULTA

Para el armado de este proyecto, se ha utilizado diversas fuentes de consulta. A continuación dejamos algunos de los links que nos fueron de utilidad:

http://www.monarcaelectronica.com.ar/

https://forum.arduino.cc/

http://seguidorlineaarduino.blogspot.com.ar/

https://jhosman.com/index.php/2013/06/04/construir-un-robot-seguidor-de-linea-con-arduino-softwarelibre-hardwarelibre/

https://developer.android.com/studio/intro/index.html?hl=es-419

https://www.tuexpertomovil.com/2013/12/29/como-emparejar-un-movil-android-con-un-dispositivo-bluetooth/

ANEXO 1 – CODIGO SEGUIDOR DE LINEAS (ARDUINO)

```
#define cantidadParadas 6
/*****************************************************************/
/*****   Variables Asociadas a PINES ******************************/
/*****************************************************************/

//Motor A (Derecho)
int IN1 = 13;   // Input1 conectada al pin 13
int IN2 = 12; // Input2 conectada al pin 12
int ENA = 10; // ENA para control de velocidad del motor

//Motor B (Izquierdo)
int IN3 = 3;  // Input3 conectada al pin 3
int IN4 = 2;  // Input4 conectada al pin 2
int ENB = 9; // ENB para control de velocidad del motor

//Sensores de Línea Infrarrojos
int sensora = 7; //Sensor Izquierdo
int sensorb = 6; //Sensor Central<
int sensorc = 5; //Sensor Derecho
int Valora = 0; //Valor de Lectura
int Valorb = 0;
int Valorc = 0;

//Buzzer-Speaker
int speakerPin = 11;

//Ultrasonido
int Trigger = 8; //para el pulso ultrasónico
int Echo = 4; //Para el sensor de rebote ultrasónico
```

```
//Otras Variables Globales
long distancia;
long tiempo;
unsigned long millisSpeaker;
unsigned long millisSpeakerAnt = 0;
unsigned long millisEspera;
unsigned long millisEsperaAnt = 0;
unsigned long millisUltrasonido;
unsigned long millisUltrasonidoAnt = 0;
unsigned long millisParada;
unsigned long millisParadaAnt = 0;
unsigned long millisProx;
unsigned long millisProxAnt = 0;
unsigned long millisLum;
unsigned long millisLumAnt = 0;


int nota; //Sonido que hace el buzzer
int esperando;
bool abasteciendo;
bool obstaculizado;
bool paradas[cantidadParadas]; //Para saber en cuales parar
int parada = 0; //Acumulador de paradas detectadas
int direccion = 0;
int contadorDeteccionesDerechas = 0; //Dado que los sensores CNY70 suelen tirar lecturas erráticas, me aseguro con al menos 3 o 4 seguidas antes de cambiar la dirección
int contadorDeteccionesIzquierdas = 0;
/*
 Dirección determina cómo están andando las ruedas para evitar seteos innecesarios.
 0 Parado
```

```
 1 Andand Rueda Derecha
 2 Andando Rueda Izquierda
 3 Andando Ambas Ruedas
 4 Andando Lento
*/
int contadorRecepcion = 0; //Contador de recepción de paradas en la comunicación BT/Serial
char c; //Caracter recibido por BT

void setup() {
 Serial.begin(9600);//Inicializar la comunicación serial
 /*****************************************************************/
 /*****  Inicialización de PINES *********************************/
 /*****************************************************************/

 pinMode (IN1, OUTPUT); //Pines de Motor
 pinMode (IN2, OUTPUT);
 pinMode (IN4, OUTPUT);
 pinMode (IN3, OUTPUT);
 pinMode (ENA, OUTPUT);
 pinMode (ENB, OUTPUT);
 pinMode (Trigger, OUTPUT); /*activación del pin salida para el pulso ultrasónico*/
 pinMode (Echo, INPUT); /*activación del pin entrada del rebote del ultrasonido*/
 pinMode (sensora, INPUT);//Definir el sensor(pin5) como entrada
 pinMode (sensorb, INPUT);//Definir el sensor(pin6) como entrada
 pinMode (sensorc, INPUT);//Definir el sensor(pin7) como entrada
 abasteciendo = true;
 obstaculizado = false;
 esperando = 0;

 for (int i = 0; i < cantidadParadas; i++) {
  paradas[i] = false;
```

```
 }
}

void loop() {
 /*****************************************************/
 //Recepción de transmisión por BT
 /*****************************************************/
 if (Serial.available() > 0) {
  c = Serial.read();
  //Verbose para Debuggin por Serial
  //Serial.print("Leido ");
  //Serial.write(c);
  //Serial.print("\n");
 }

 //Comprobación de señales enviadas por BT
 if (c == 'P') { //Si recibe la señal de proximidad
  millisProx=millis();
  if (millisProx-millisProxAnt>=3000){
  //Generar Alarma
  tone(speakerPin, 290, 1500);
  millisProxAnt=millisProx;
  }
  c = 'X'; //Desechamos lo leído para evitar re-lectura.
 }

 if (c == 'L') { //Si recibe la señal de Luminosidad
  millisLum=millis();
  if (millisLum-millisLumAnt>=3000){
  //Provocar Espera pequeña
  digitalWrite (IN2, LOW); //Frena Rueda derecha
```

```
  digitalWrite (IN1, LOW);
  digitalWrite (IN4, LOW); //Frena Rueda Izquierda
  digitalWrite (IN3, LOW);
  direccion = 0; //Informe de frenado
  esperando = 3;
    millisProxAnt=millisProx;
  }
  //Podría llamarse una espera superlarga y luego cortarla con otra señal cuando se reestablece la luz
  c = 'X'; //Desechamos lo leído para evitar re-lectura.

 }

 /*--------------Comprobación de Obstáculos--------------*/
 /*--------------Comprobación de Obstáculos--------------*/
 /*--------------Comprobación de Obstáculos--------------*/
 millisUltrasonido = millis(); //Medición del obstáculo cada décima de segundo, por problemas de ecos.
 if (millisUltrasonido - millisUltrasonidoAnt >= 100) { //Si se mide distancia
  millisUltrasonidoAnt = millisUltrasonido;
  //Lectura de Distancia
  digitalWrite(Trigger, LOW); /* Por cuestión de estabilización del sensor*/
  delayMicroseconds(5);
  digitalWrite(Trigger, HIGH); /* envío del pulso ultrasónico*/
  delayMicroseconds(10);
  tiempo = pulseIn(Echo, HIGH); /* Función para medir la longitud del pulso entrante. Mide el tiempo que transcurrido entre el envío
 del pulso ultrasónico y cuando el sensor recibe el rebote, es decir: desde que el pin 4 empieza a recibir el rebote, HIGH, hasta que
 deja de hacerlo, LOW, la longitud del pulso entrante*/
  distancia = int(0.017 * tiempo); /*fórmula para calcular la distancia obteniendo un valor entero*/
  /*Monitorización en centímetros por el monitor serial*/
```

```
  //Serial.print(distancia);
  // Serial.print("\n");
  if (distancia < 10 && distancia != 0 ) { //Y la distancia es menos de 10 cm (Y distinta de cero porque a veces el sensor se cuelga y da cero sin razón conocida)
    obstaculizado = true; //Obstaculizado ----------------------------
    //Frenar Ruedas
    analogWrite(ENA, 0);
    analogWrite(ENB, 0);
    direccion = 0; //Informe de frenado
    millisSpeaker = millis();
    if (millisSpeaker - millisSpeakerAnt >= 1000) {
      millisSpeakerAnt = millisSpeaker;
      if (nota == 294) {
        nota = 494;
      }
      else
      {
        nota = 294;
      }

      tone(speakerPin, nota, 500);
      Serial.write('X');
    }
  }
  else { //Si la distancia no es menos de 10 no está obstaculizado
    obstaculizado = false;
  }
 }
 else  { //Si no se midió distancia no hay que hacer nada en especial, se conserva el estado
 }
```

```
 /*----------------Código Seguidor con Estaciones-------------------*/

 /*----------------Código Seguidor con Estaciones-------------------*/

 /*----------------Código Seguidor con Estaciones-------------------*/

 if (!obstaculizado) { //Si no está obstaculizado

  //noTone(speakerPin); //Speaker en silencio (Aún así no se por qué hace un pequeño sonido)

  if (!abasteciendo) { //Si no está en estación de abastecimiento

   if (esperando == 0) { //Si no está esperando

    //Funcionamiento de Seguidor de Líneas

    Valora = digitalRead(sensora); //Leer y almacenar el valor del sensor Derecho

    Valorb = digitalRead(sensorb); //Leer y almacenar el valor del sensor Central

    Valorc = digitalRead(sensorc); //Leer y almacenar el valor del sensor Izquierdo

    delay(20);//Esperar 20 ms

    if (Valora == 1 && Valorb == 0 && Valorc == 1 && direccion != 3) { //B-N-B, Ambas Ruedas andando

     analogWrite(ENA, 190);

     analogWrite(ENB, 190);

     digitalWrite (IN2, HIGH); //Anda Rueda derecha

     digitalWrite (IN1, LOW);

     delay(50); //Por alguna razón los motores arrancan a destiempo, uso esto para intentar sincronizarlo.

     digitalWrite (IN4, HIGH); //Anda Rueda Izquierda

     digitalWrite (IN3, LOW);

     direccion = 3;

     //Serial.write('A');

     contadorDeteccionesDerechas = 0; //Cuando se detecta ir para adelante, se resetean los cambios de dirección, para ignorar las lecturas erráticas.

     contadorDeteccionesIzquierdas = 0;

    }

    if ((Valora == 1 && Valorb == 0 && Valorc == 0 || Valora == 1 && Valorb == 1 && Valorc == 0) && direccion != 1) { //B-N-N,BBN Gira para Derecha

     contadorDeteccionesIzquierdas++;

     contadorDeteccionesDerechas = 0;
```

```
      if (contadorDeteccionesIzquierdas > 2) {
        analogWrite(ENA, 180); //Anda Rueda Izquierda
        analogWrite(ENB, 0);
        digitalWrite (IN2, HIGH);
        digitalWrite (IN1, LOW);
        digitalWrite (IN4, HIGH);
        digitalWrite (IN3, LOW);
        direccion = 1;
       // Serial.write('I');
      }
    }
    if ((Valora == 0 && Valorb == 0 && Valorc == 1 || Valora == 0 && Valorb == 1 && Valorc == 1) && direccion != 2) { //N-N-B,NBB Gira para Izquierda
      contadorDeteccionesDerechas++;
      contadorDeteccionesIzquierdas = 0;
      if (contadorDeteccionesDerechas > 2) {
        analogWrite(ENA, 0);
        analogWrite(ENB, 180); //Anda Rueda Derecha
        digitalWrite (IN2, HIGH);
        digitalWrite (IN1, LOW);
        digitalWrite (IN4, HIGH);
        digitalWrite (IN3, LOW);
        direccion = 2;
        //Serial.write('D');
      }

    }
    if (Valora == 0 && Valorb == 0 && Valorc == 0) { //N-N-N, Estación detectada (O hubo crash), frenar ruedas
      //Millis para detección de paradas, sino con cada una detecta innumerables paradas
      millisParada = millis();
      if (millisParada > millisParadaAnt + 1000) {
```

```
    millisParadaAnt = millisParada;
    parada++;
    if (parada % cantidadParadas == 0) { //Si es parada cero
     abasteciendo = true; //Abasteciendo, entra en estado receptor
     contadorRecepcion = 0; //Preparamos el contador para la recepción de paradas
     digitalWrite (IN2, LOW); //Frena Rueda derecha
     digitalWrite (IN1, LOW);
     digitalWrite (IN4, LOW); //Frena Rueda Izquierda
     digitalWrite (IN3, LOW);
     direccion = 0; //Informe de frenado
     //Envío de señal de llegada a Abastecimiento por BT
     Serial.write('0');
    }
    else if (paradas[parada % cantidadParadas] == true) { //Si valía parar en dicha parada
     digitalWrite (IN2, LOW); //Frena Rueda derecha
     digitalWrite (IN1, LOW);
     digitalWrite (IN4, LOW); //Frena Rueda Izquierda
     digitalWrite (IN3, LOW);
     direccion = 0; //Informe de frenado
     esperando = 5;
     //Envío de señal de llegada a Parada por BT
     Serial.write((parada % cantidadParadas) + '0');
    }
   }
  }
 } else { //Si está esperando
  direccion = 0;
  //Un loop fijo de "esperando" segundos para la espera en parada
  nota = 261;
  while (esperando > 0) {
   millisEspera = millis();
```

```
        if (millisEspera - millisEsperaAnt >= 1000) {
          millisEsperaAnt = millisEspera;
          tone(speakerPin, nota, 500);
          nota += 40;
          esperando--;
        }
      }
      //Aviso de parada parada%cantidadParadas completada
    }
  } else {//Si está abasteciendo
    // Recepción de listas de paradas
    if (contadorRecepcion < (cantidadParadas - 1)) { //Si aún no recibió todas las paradas
      if (c == '0' || c == '1') { //Si es un 0 o 1, los caracteres de lista de paradas
        contadorRecepcion++;
        paradas[contadorRecepcion] = c - '0';
        c = 'X'; //Para evitar volver a leer el mismo hasta que se asigne en la lectura arriba.
      }
    }
    if (contadorRecepcion == (cantidadParadas - 1)) { //Si ya recibió todas las paradas
      abasteciendo = false; //Deja el modo de abastecimiento
      //Serial.write('R');//Aviso de Recepción y salida a Repartir
    }
  }
 }
}
```

ANEXO 2 – INFORMACIÓN COMPLEMENTARIA

Sensores CNY70

VISHAY

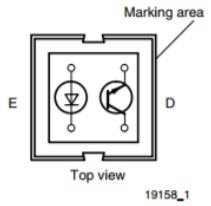
www.vishay.com

CNY70

Vishay Semiconductors

Reflective Optical Sensor with Transistor Output

21835



19158_1

FEATURES

- Package type: leaded
- Detector type: phototransistor
- Dimensions (L x W x H in mm): 7 x 7 x 6
- Peak operating distance: < 0.5 mm
- Operating range within > 20 % relative collector current: 0 mm to 5 mm
- Typical output current under test: I_C = 1 mA
- Emitter wavelength: 950 nm
- Daylight blocking filter
- Lead (Pb)-free soldering released
- Material categorization: For definitions of compliance please see www.vishay.com/doc?99912

Pb-free

e4

RoHS COMPLIANT

DESCRIPTION

The CNY70 is a reflective sensor that includes an infrared emitter and phototransistor in a leaded package which blocks visible light.

APPLICATIONS

- Optoelectronic scanning and switching devices i.e., index sensing, coded disk scanning etc. (optoelectronic encoder assemblies).

PRODUCT SUMMARY

PART NUMBER	DISTANCE FOR MAXIMUM CTR_{rel} (1) (mm)	DISTANCE RANGE FOR RELATIVE I_{out} > 20 % (mm)	TYPICAL OUTPUT CURRENT UNDER TEST (2) (mA)	DAYLIGHT BLOCKING FILTER INTEGRATED
CNY70	0	0 to 5	1	Yes

Notes

(1) CTR: current transfere ratio, I_{out}/I_{in}

(2) Conditions like in table basic charactristics/sensors

ORDERING INFORMATION

ORDERING CODE	PACKAGING	VOLUME (1)	REMARKS
CNY70	Tube	MOQ: 4000 pcs, 80 pcs/tube	-

Note

(1) MOQ: minimum order quantity

ABSOLUTE MAXIMUM RATINGS (T_{amb} = 25 °C, unless otherwise specified)

PARAMETER	TEST CONDITION	SYMBOL	VALUE	UNIT
COUPLER				
Total power dissipation	$T_{amb} \leq 25$ °C	P_{tot}	200	mW
Ambient temperature range		T_{amb}	- 40 to + 85	°C
Storage temperature range		T_{stg}	- 40 to + 100	°C
Soldering temperature	Distance to case 2 mm, t £ 5 s	T_{sd}	260	°C
INPUT (EMITTER)				
Reverse voltage		V_R	5	V
Forward current		I_F	50	mA
Forward surge current	$t_p \leq 10$ µs	I_{FSM}	3	A
Power dissipation	$T_{amb} \leq 25$ °C	P_V	100	mW
Junction temperature		T_j	100	°C

Sensor Ultrasónico

Ultrasonic Ranging Module HC - SR04

Product features:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:
(1) Using IO trigger for at least 10us high level signal,
(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.
Test distance = (high level time×velocity of sound (340M/S) / 2,

Wire connecting direct as following:

- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

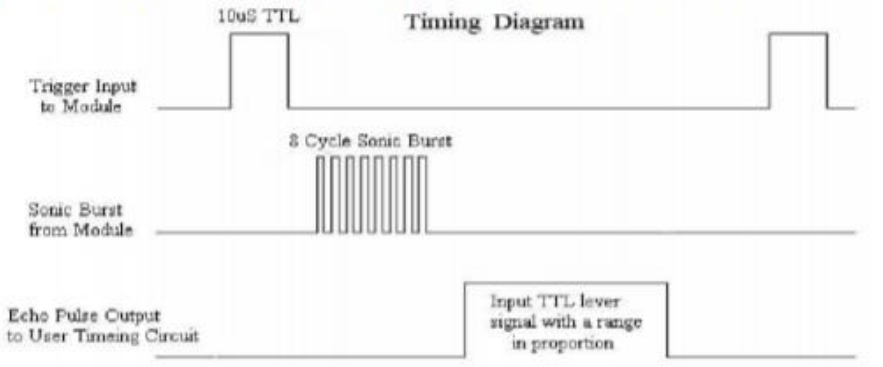
Electric Parameter

Working Voltage	**DC 5 V**
Working Current	**15mA**
Working Frequency	**40Hz**
Max Range	**4m**
Min Range	**2cm**
MeasuringAngle	**15 degree**
Trigger Input Signal	**10uS TTL pulse**
Echo Output Signal	**Input TTL lever signal and the range in proportion**
Dimension	**45*20*15mm**

Vcc Trig Echo GND

Timing diagram

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: uS / 58 = centimeters or uS / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.



Sensor Bluetooth HC-06

Bluetooth to serial HC-06 wireless module

Product Description:

1，Mainstream CSR Bluetooth chip, Bluetooth V2.0 protocol standards

2，serial module operating voltage 3.3V.

3，the baud rate for 1200, 2400, 4800, 9600, 19200, 38400 and users can be set

4，core module size: 28mm x 15 mm x 2.35mm.

5，the working current: 40mA

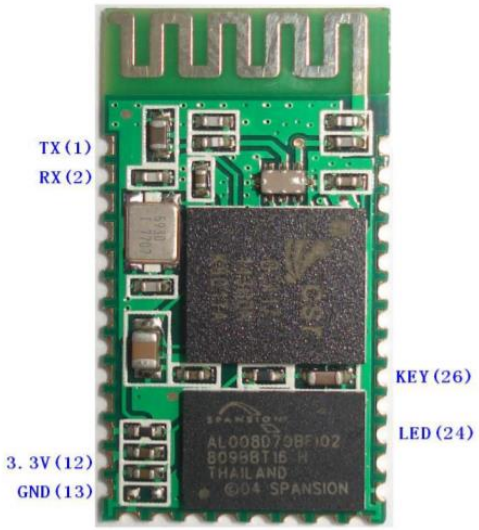
6，Sleep current: <1mA

7，for the GPS navigation system, utility meter reading system, the industrial field, collecting and controlling system.

8，with a Bluetooth laptop computer to the Bluetooth adapter, PDA and other devices to seamlessly connect

The module's host and slave, the host and slave pairing communication from the machine and from the machine or between the host and the host can not communicate, communication function and computers, mobile phones and other Bluetooth pairing purchase default slave , requires that the host needs to be indicated]

Main distinction: 1, if the chip is not specified on, the lights flash slow main fast from; September 2,2009, all manufactured host will be playing in the IC a hook or paste There are the "main" characters, there is no hook or not affixed to the word "master" is the slave. The date of manufacture can be obtained from the Bluetooth address]



Motores con caja reductora

Descripción del Producto

Motorreductor especial para aplicaciones de robótica. Implementado en carritos seguidores de linea, evasores de obstáculos.

- **Voltaje de operación: 3V~12VDC (recomendado 6V a 8V)**
- Torque máximo: 800gf cm/min (3V)
- Este motor cumple EMC, capacidad anti-interferencia. Puede operar con microcontrolador sin interferencias.
- Tamaño: 7 x 2.2 x 1.8cm (approx.)

Caracteristicas

Voltaje de operación	Parámetro	3V DC	5V DC	6V DC
Parámetros del motor	RPM		125R / minuto	
(Sin caja reductora)	Corriente		80-100mA	
Parámetros de caja reductora	Reducción		48:1	
	Velocidad sin carga	125R/minuto	200R/minuto	230R/minuto
	Velocidad con carga	95R/minuto	152R/minuto	175R/minuto
	Torque de salida	0.8kg.cm	1.0kg.cm	1.1kg.cm
	Velocidad carro Sin carga	25.9meter/minuto	41.4meter/minuto	47.7meter/minuto
	Corriente	110-130mA	120-140mA	130-150mA
	Max. diámetro de llanta		6.5cm	
	Dimensiones		70mm x 22mm x 18mm	
	Peso		50g	
	Ruido		<65dB	

Puente H – L298

DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)

Multiwatt15

PowerSO20

ORDERING NUMBERS : L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

L298

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_I, V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

THERMAL DATA

Symbol	Parameter		PowerSO20	Multiwatt15	Unit
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	Max.	–	3	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	13 (*)	35	°C/W

(*) Mounted on aluminum substrate

Placa Arduino UNO

Overview

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.
The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.
Revision 2 of the Uno board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into DFU mode.
Revision 3 of the board has the following new features:

- 1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible both with the board that use the AVR, which operate with 5V and with the Arduino Due that operate with 3.3V. The second one is a not connected pin, that is reserved for future purposes.
- Stronger RESET circuit.
- Atmega 16U2 replace the 8U2.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V

Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Buzzer

Buzzer

Features

- Black in colour
- With internal drive circuit
- Sealed structure
- Wave solderable and washable
- Housing material: Noryl

Applications

- Computer and peripherals
- Communications equipment
- Portable equipment
- Automobile electronics
- POS system
- Electronic cash register

Specifications:

Rated Voltage	: 6V DC
Operating Voltage	: 4 to 8V DC
Rated Current*	: ≤30mA
Sound Output at 10cm*	: ≥85dB
Resonant Frequency	: 2300 ±300Hz
Tone	: Continuous
Operating Temperature	: -25°C to +80°C
Storage Temperature	: -30°C to +85°C
Weight	: 2g

*Value applying at rated voltage (DC)