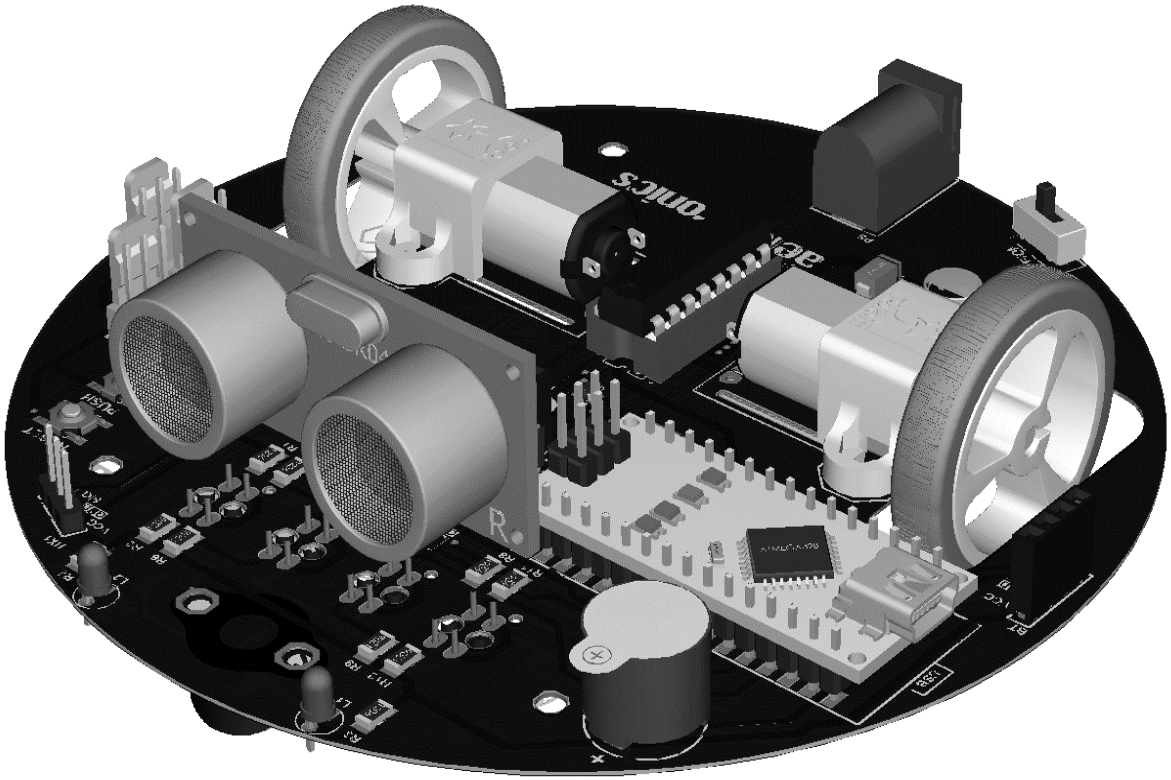

CHO-KOOS V2

MANUAL DE INSTRUCCIONES

INDICE

ÍNDICE DE FIGURAS

CAPITULO 1

Manual Cho-Koos v2

1.1-Descripción

Cho-koos v2 es un robot de programación basado en Arduino.

1.2-Características

- Soporte para Arduino.
- Movimiento flexible y de pequeño tamaño.
- Motor de engranajes en miniatura totalmente metálico.
- Seguidor de línea, Bluetooth, Sensor ultrasónico(HC-SR04), Buzzer, interfaz I2C, Infrarrojo.
- Fácil de instalar, fácil de usar.

1.3-Diagrama de Funciones

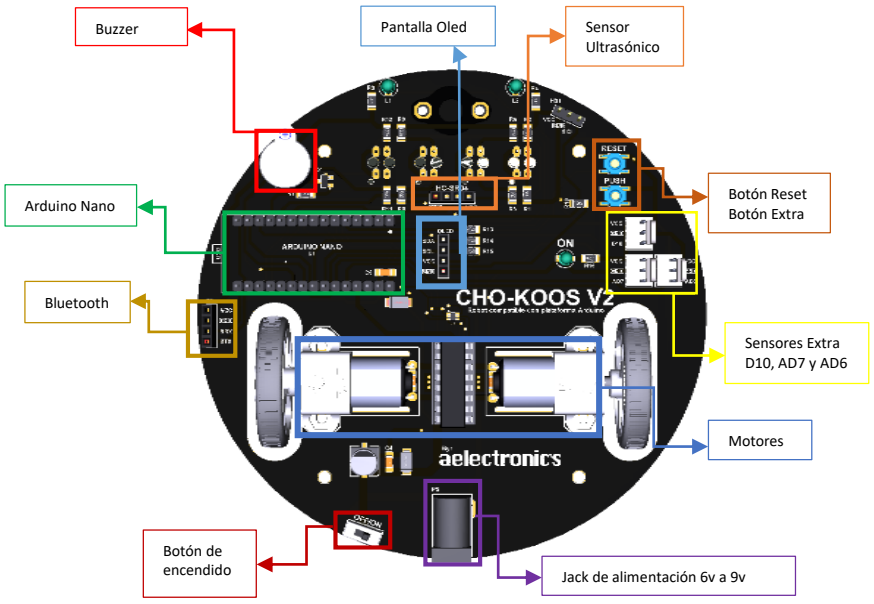

Fig. 1 Parte Superior Cho-Koos.

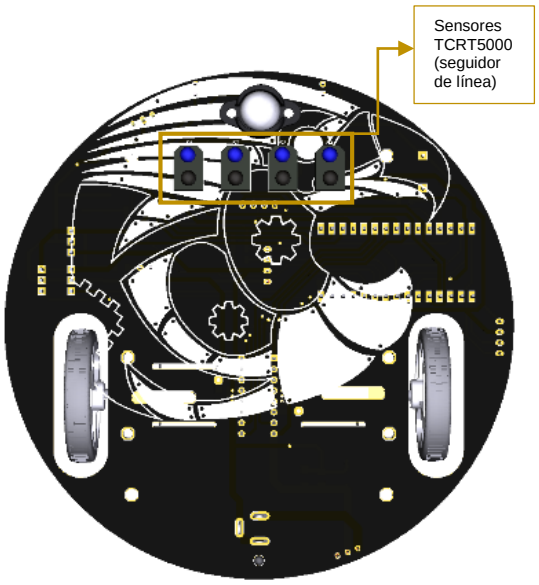

Fig. 2 Parte Inferior Cho-Koos.

1.4-Especificaciones:

- El voltaje de alimentación de 6 a 9v (se sugiere usar baterías de litio)
- TCRT5000 (Seguidor de línea) x 3
- Zumbador x 1
- Pantalla Oled I2C x 1
- Bluetooth x 1
- Sensor ultrasónico (HC-SR04) x 1
- Interfaz I2C (5v) x 1
- Puerto Sensor Analógico x 2 y Puerto Sensor Digital x 1
- Sensor infrarrojo x 1
- Motor reductor x 2
- Control de velocidad (PWM) de motores
- Programación por bloques.

1.5-Instalación de Librerías:

Seleccionar la pestaña "Programa", en la sección "Incluir Librería" finalmente en "Añadir biblioteca .ZIP…" (*Pag.3*).

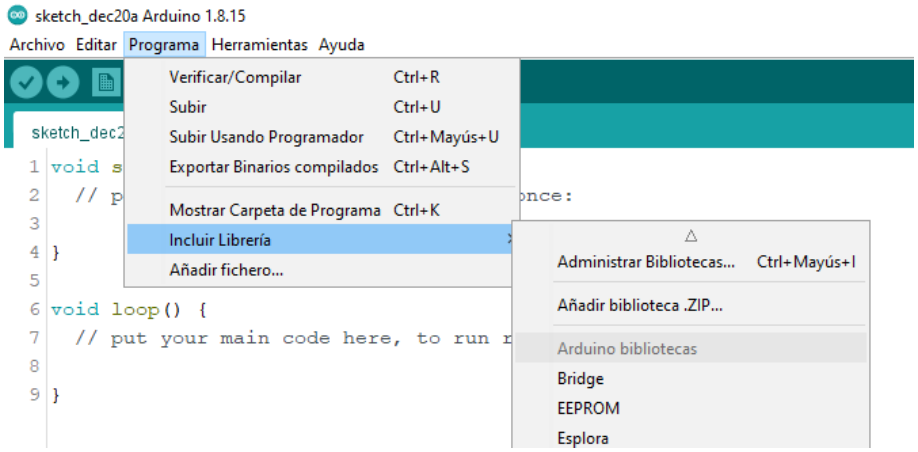

Fig. 3 Instalación de Librería.

Saldrá una ventana (*Fig. 4*) donde tenemos que buscar la librería en el lugar donde se almaceno y seleccionarla.

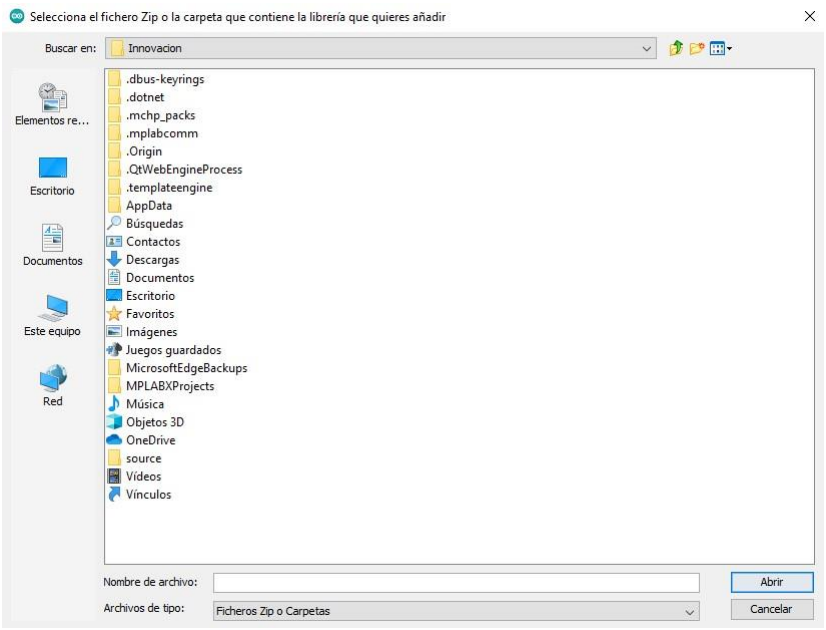

Fig. 4 Ventana de Selección.

CAPITULO 2

Códigos:

NOTA: *Al inicio de los códigos se tienen que poner los motores en bajo para que no estén habilitados desde el inicio.*

```
void setup() {
  pinMode(PWM1, OUTPUT);
  pinMode(INA, OUTPUT);
  pinMode(PWM2, OUTPUT);
  pinMode(INC, OUTPUT);
  digitalWrite(PWM1, LOW); //Deshabilitar los motores
  digitalWrite(PWM2, LOW); //Deshabilitar los motores
}
```

Fig. 5 Detener motores.

2.1-Control de motor:

El programa se encarga de hacer que avance hacia adelante, izquierda, detener, derecha y atrás en intervalos de 1 segundo.

```
int PWM1 = 3;
int INA = 4;
int PWM2 = 5;
int INC = 6;
int vel = 225;
void setup() {
  pinMode(PWM1, OUTPUT);
  pinMode(INA, OUTPUT);
  pinMode(PWM2, OUTPUT);
  pinMode(INC, OUTPUT);
  digitalWrite(PWM1, LOW);
  digitalWrite(PWM2, LOW);
}
void loop() {
  //RECTO
  analogWrite(PWM1, vel);
  digitalWrite(INA, HIGH);
  analogWrite(PWM2, vel);
  digitalWrite(INC, LOW);
  delay(1000);
  //IZQUIERDA
  analogWrite(PWM1, vel);
  digitalWrite(INA, LOW);
  analogWrite(PWM2, vel);
  digitalWrite(INC, LOW);
  delay(1000);
  //PARAR
  analogWrite(PWM1, vel);
  digitalWrite(INA, LOW);
```

```
  analogWrite(PWM2, vel);
  digitalWrite(INC, LOW);
  delay(1000);
  //DERECHA
  analogWrite(PWM1, vel);
  digitalWrite(INA, HIGH);
  analogWrite(PWM2, vel);
  digitalWrite(INC, HIGH);
  delay(1000);
  //REVERSA
  analogWrite(PWM1, vel);
  digitalWrite(INA, LOW);
  analogWrite(PWM2, vel);
  digitalWrite(INC, HIGH);
  delay(1000);
}
```

*Descargar código de: https://github.com/A-electronics/CHO-KOOS-v2/tree/main/Motores

2.2-Ultrasónico:

El programa realizara la lectura del sensor y el resultado lo devuelve al monitor serial.

```
const int EchoPin = 7;
const int TriggerPin = 8;
int tiempo = 0;
void setup() {
  Serial.begin(9600);
  pinMode(TriggerPin, OUTPUT);
  pinMode(EchoPin, INPUT);
  delay(200);
}
void loop() {
   int cm = ping(TriggerPin, EchoPin);
   Serial.print("Distancia: ");
   Serial.println(cm);
   delay(1000);
}
int ping(int TriggerPin, int EchoPin) {
   long duration, distanceCm;
   digitalWrite(TriggerPin, LOW);  //para generar un pulso limpio ponemos a LOW 4us
   delayMicroseconds(4);
   digitalWrite(TriggerPin, HIGH);  //generamos Trigger (disparo) de 10us
   delayMicroseconds(10);
   digitalWrite(TriggerPi|n, LOW);

   duration = pulseIn(EchoPin, HIGH);  //medimos el tiempo entre pulsos, en microsegundos
   distanceCm = duration * 10 / 292/ 2; //convertimos a distancia, en cm
   return distanceCm;
}
```

*Descargar código de:
https://github.com/A-electronics/CHO-KOOS-v2/tree/main/Modulo_Ultrasonico_HC-SR04

2.3-Buzzer:

Vamos a hacer uso de la instrucción ***tone*** cuyos parámetros son el pin de salida al cual tenemos conectado el zumbador, la frecuencia de la señal que vamos a generar para la nota y la duración en milisegundos (opcional). ***tone(pin, frecuencia) o tone(pin, frecuencia, tiempo).***

```
int PWM1 = 3;
int PWM2 = 5;
int pinBuzzer = 9;
int Do = 261;
int Re = 293;
int Mi = 329;
int Fa = 349;
int Sol = 392;
int La = 440;
int Si = 493;
int duracion = 100;
int retardo = 200;

void setup() {
  pinMode(PWM1, OUTPUT);
  pinMode(PWM2, OUTPUT);
  digitalWrite(PWM1, LOW); //Deshabilitar los motores
  digitalWrite(PWM2, LOW); //Deshabilitar los motores
}

void loop() {
  tone(pinBuzzer, Do, duracion);
  delay(retardo);
  tone(pinBuzzer, Re, duracion);
  delay(retardo);
  tone(pinBuzzer, Mi, duracion);
  delay(retardo);
  tone(pinBuzzer, Fa, duracion);
  delay(retardo);
  tone(pinBuzzer, Sol, duracion);
  delay(retardo);
  tone(pinBuzzer, La, duracion);
  delay(retardo);
  tone(pinBuzzer, Si, duracion);
  delay(retardo);
}
```

**Descargar código de:* https://github.com/A-electronics/CHO-KOOS-v2/tree/main/Buzzer

2.4-Pantalla oled:

En este código se necesita descargar la librería "*Adafruit SSD1306*" para hacer uso correcto de la pantalla OLED 128x64. Para instalarlo siga los pasos de la Pág.5 y Pág.6 (Fig. 3 y Fig.4).

Muestra la palabra CHO-KOOS en el centro de la pantalla.

```
#include <Adafruit_GFX.h> //Liberia para gráficos
#include <Adafruit_SSD1306.h> //Liberia para Oleds monocromáticos basados en controladores
SSD1306
/*Se declara el pin de reset, este es requerido por el constructor de la librería SSD1306
*para definir el tipo de OLED y la comunicación I2C en los (pines SDA, SCL)*/
#define OLED_RESET 13
Adafruit_SSD1306 display(OLED_RESET);
/*Se define una condición para saber si en la librería está definida la altura de 32 de lo
contrario
*no dejara compilar el código hasta que se cambie la altura correcta desde la librería*/
#if (SSD1306_LCDHEIGHT != 32)
#error("Altura incorrecta, cambie en la librería de Adafruit_SSD1306.h!");
#endif

void setup() {
pinMode(3,OUTPUT);
pinMode(5,OUTPUT);
delay(10);
digitalWrite(3,LOW);
digitalWrite(5,LOW);
/* Se inicia la comunicación I2C en la dirección 0x3C para la pantalla oled 128x32*/
display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
}
void loop() {
/*Se manda a llamar la función leer_oled para ejecutar su contenido*/
leer_oled();
}
/*Se declara la función leer_oled la cual define el tamaño, color, texto y la posición del
texto
que se mostrara en el display oled*/
void leer_oled() {
display.clearDisplay(); //Borra el buffer
display.display(); //Muestra la imagen
delay(2000); //Se muestre la imagen solo 2 segundos
display.clearDisplay(); //Borra el buffer
display.setTextSize(1); //Establece el tamaño de fuente, admite tamaños de 1 a 8
display.setTextColor(WHITE); //Establece el color
display.setCursor(35,10); //Establecer las coordenadas para mostrar la posición del texto
display.println("CHO-KOOS");
display.display(); //Muestra el texto
delay(2000); //Se muestre el texto por solo 2 segundos
}
```

Descargar código de:
https://github.com/A-electronics/CHO-KOOS-v2/tree/main/Display_Oled_128_32_SSD1306

2.5-Bluetooth:

Utilizamos el siguiente código para manejar el carro desde el teléfono mediante Bluetooth, mediante números mandamos las siguientes instrucciones: “0” detener el carro, “1” Avanzar, “2” Girar a la izquierda, “3” Girar a la derecha, “4” Reversa.

```
#include <SoftwareSerial.h> //Librería que permite establecer comunicación serie en otros
pins
//Aquí conectamos los pins RXD,TDX del módulo Bluetooth.
SoftwareSerial BT(13,12); //12 RX, 13 TX.
int estado = 48;
int PWM1 = 3;
int INA = 4;
int PWM2 = 5;
int INC = 6;
int vel = 245;
void setup() {
  BT.begin(9600);
}
void loop() {
     if(BT.available())
  {
    //Serial.write(BT.read());
    estado = BT.read();
  }
  Serial.println(estado);
  delay(100);
  if(estado == 49){    //RECTO
    analogWrite(PWM1,vel);
    digitalWrite(INA,HIGH);
    digitalWrite(PWM2,HIGH);
    digitalWrite(INC,LOW);
  }
  if(estado == 50){    //IZQUIERDA
    digitalWrite(PWM1,HIGH);
    digitalWrite(INA,LOW);
    digitalWrite(PWM2,HIGH);
    digitalWrite(INC,LOW);
  }
  if(estado == 48){     //Parar
    digitalWrite(PWM1,LOW);
    digitalWrite(INA,LOW);


    digitalWrite(PWM2,LOW);
    digitalWrite(INC,LOW);
  }
  if(estado == 51){    //DERECHA
    digitalWrite(PWM1,HIGH);
    digitalWrite(INA,HIGH);
    digitalWrite(PWM2,HIGH);
    digitalWrite(INC,HIGH);
  }
  if(estado == 52){    //REVERSA
    digitalWrite(PWM1,HIGH);
    digitalWrite(INA,LOW);
    digitalWrite(PWM2,HIGH);
    digitalWrite(INC,HIGH);
  }
}
```

*Descargar código de: https://github.com/A-electronics/CHO-KOOS-v2/tree/main/Bluetooth

2.6-Infrarrojo:

En este código se necesita descargar la librería "*IRremote*" para hacer uso correcto del sensor infrarrojo. Para instalarlo siga los pasos de la Pág.5 y Pág.6 (Fig. 3 y Fig.4).

Manda al monitor serial los datos obtenidos en formato hexadecimal dados por un control remoto.

```
#include <IRremote.h>    // importa librería IRremote
int PWM1 = 3;
int PWM2 = 5;
int SENSOR = 2;    // sensor KY-022 a pin digital 2
IRrecv irrecv(SENSOR);    // establece al 2 para objeto irrecv
decode_results codigo;    // crea objeto código de la clase decode_results

void setup() {
  pinMode(PWM1, OUTPUT);
  pinMode(PWM2, OUTPUT);
  pinMode(INA, OUTPUT);
  pinMode(INC, OUTPUT);
  Serial.begin(9600);     // inicializa comunicación serie a 9600 bps
  irrecv.enableIRIn();    // inicializa recepción de datos
}

void loop() {
  digitalWrite(PWM1, LOW);
  digitalWrite(PWM2, LOW);
  if (irrecv.decode(&codigo)) {   // si existen datos ya decodificados
    Serial.println(codigo.value, HEX);  // imprime valor en hexadecimal en monitor
    irrecv.resume();      // resume la adquisición de datos
  }
  delay (100);        // breve demora de 100 ms.
}
```

*Descargar código de: https://github.com/A-electronics/CHO-KOOS-v2/tree/main/Infrarojo

2.7-Sensores:

Utilizamos el siguiente código para detectar los valores mapeados (0-255) de los sensores y mostrarlos en el monitor serial.

```
int sensor4 = 0; //pines analógicos almacenados en una variable
int sensor3 = 1; //pines analógicos almacenados en una variable
int sensor2 = 2; //pines analógicos almacenados en una variable
int sensor1 = 3; //pines analógicos almacenados en una variable
int valorSensor4 = 0;
int valorSensor3 = 0;
int valorSensor2 = 0;
int valorSensor1 = 0;
void setup() {
  pinMode(3, OUTPUT);
```

```
  pinMode(5, OUTPUT);
  delay(10);
  pinMode(sensor1, INPUT);
  pinMode(sensor2, INPUT);
  pinMode(sensor3, INPUT);
  pinMode(sensor4, INPUT);
  Serial.begin(9600);
  analogWrite(3, 0);
  analogWrite(5, 0);
}

void loop() {
  valorSensor1 = analogRead(sensor1);
  valorSensor2 = analogRead(sensor2);
  valorSensor3 = analogRead(sensor3);
  valorSensor4 = analogRead(sensor4);
//Mapeo de
  valorSensor1 = map(valorSensor1, 0, 1023, 0, 255);
  valorSensor2 = map(valorSensor2, 0, 1023, 0, 255);
  valorSensor3 = map(valorSensor3, 0, 1023, 0, 255);
  valorSensor4 = map(valorSensor4, 0, 1023, 0, 255);

  Serial.print("Valor Sensor1 TCRT5000_1: ");
  Serial.println(valorSensor1);
  Serial.print("Valor Sensor2 TCRT5000_2: ");
  Serial.println(valorSensor2);
  Serial.print("Valor Sensor3 TCRT5000_3: ");
  Serial.println(valorSensor3);
  Serial.print("Valor Sensor4 TCRT5000_4: ");
  Serial.println(valorSensor4);
  Serial.println("");
  delay(1000);
}
```

**Descargar código de:* https://github.com/A-electronics/CHO-KOOS-v2/tree/main/Sensores