/\* El robot (Bluetooth)! \*/

/\* En esta experiencia vamos a juntar lo aprendido en las experiencias

\* anteriores sobre el uso de motores servo, sensor de distancia y buzzers

\* con el objetivo de construir un pequeño robot que controlaremos con control

\* remoto (bluetooth)

\*/

/\*

Se cargan las librerias necesarias

\*/

#include <SoftwareSerial.h> //Librer+ia para asignar otros puertos seriales

#include <Servo.h> //Para control de los motores servo

/\*

Se definen los pines que vamos a usar

\*/

int buzzPin = 8; // Pin del buzzer

int button\_pin = 7; // Pin donde se conecta el botón

int led\_pin = 13; // Pin que avisará estados

int sensorizqPin = 0; // Entrada analogica que lee al sensor de distancia ("ojo izquierdo")

int sensorderPin = 1; // Entrada analogica que lee al sensor de distancia ("ojo derecho")

int servoizqPin = 9; // Pin del servo izquierdo

int servoderPin = 6; // Pin del servo derecho

/\*

Se declaran variables para el manejo del robot

\*/

// Variables que se envian a los servo para el movimiento

int ang\_izq = 90;

int ang\_der = 90;

/\*

Variable para definir la direccion de movimiento.

0 -> Reposo

1 -> Avanzar

2 -> Retroceder

3 -> Virar izquierda

4 -> Virar derecha

\*/

int estado = 0;

boolean apagar = true; // Variable para controlar encendido y apagado (condición inicial)

boolean modoauto = false; // Variable para controlar modo de funcionamiento

char comando = 0; // Variable para almacenar el comando recibido por bluetooth

int button\_read; // Variable Botón apagado/encendido

// Mediciones del sensor

int sensorizqRead; // Variable que almacena el voltaje leido en el sensor de la izquierda

int sensorderRead; // Variable que almacena el voltaje leido en el sensor de la izquierda

int dist\_izq; // Variable para la distancia medida por el sensor de la izquierda

int dist\_der; // Variable para la distancia medida por el sensor de la derecha

// Tolerancias de distancia. Las usaremos para decidir que hace el robot

int tolerancia\_sup = 18;

int tolerancia\_inf = 8;

// Variables para manejo de tiempo.

int intervalo; // Variable auxiliar (para comparar tiempos de sonido de buzzer)

unsigned long tiempo\_previo = 0; // Variable para medir tiempo (inicial)

unsigned long tiempo\_actual; //Variable para medir tiempo de ejecución del programa ( millis() )

// Creamos dos objetos de tipo Servo para manejar las ruedas

Servo miServo\_izq; // Servo de la rueda izquierda

Servo miServo\_der; // Servo de la rueda derecha

// Creamos un puerto serial por software en los pines 4 y 2.

SoftwareSerial BT1(4, 2); // RX, TX

void setup() {

miServo\_izq.attach(servoizqPin);

miServo\_der.attach(servoderPin);

pinMode(buzzPin, OUTPUT);

pinMode(led\_pin, OUTPUT);

pinMode(button\_pin, INPUT);

pinMode(servoizqPin, OUTPUT);

pinMode(servoderPin, OUTPUT);

BT1.begin(9600); //Comunicación con el módulo bluetooth

/\* Para efectos de este robot, el módulo se encuentra previamente configurado \*/

/\* Debe contarse con un celular con una aplicación compatible, como Bluetooth SPP disponible

\* para Android, muy fácil e intuitiva de usar. Gran herramienta

\*/

}

void loop() {

if (BT1.available()) { //Revisa si hay información desde el bluetooth. Funciona para el encendido inicial via bluetooth (espera un retorno de línea)

comando = BT1.read();

BT1.flush();

}

button\_read = digitalRead(button\_pin); //Leemos al botón

// revisar si prende, apaga o cambia modo con BT o boton.

// Actualiza estado de operacion

//

if (comando == 'o' || button\_read == LOW) { //Apaga/Enciende

apagar = !apagar;

estado = 0;

comando = 0;

digitalWrite(led\_pin, !apagar); //Robot encendido, enciende luz

tone(buzzPin, 2000,20);

delay(200); //Delay para que notemos el cambio de modo

}

else if (comando == 'm') { //Cambio de modo auto/manual

modoauto = !modoauto;

estado = 0;

comando = 0;

tone(buzzPin, 2000,20);

delay(200); //Delay para que notemos el cambio de modo

}

movimiento(estado);

if (apagar == false) { /\* Se ejecuta si el robot está encendido -> apagar = false \*/

// Mide distancia y suena

Dist\_Sensor\_Buzzer();

/\* \*\*\*Servos\*\*\* \*/

/\* Modo auto \*/

if (modoauto == true) {

// if ....

// state = 1;

// Valores utilizados fueron encontrados de forma empírica (acá usamos los parámetros de tolerancia)

// Estos parámetros se pueden variar para hacer que el robot haga acciones a determinadas distancias

// Como tenemos 3 situaciones: avanzar, viraje y retroceder y 2 sensores de distancia, tenemos 3^2 posibles

// situaciones que pueden ocurrir (con el conector AND(&&)).

// Comentaremos lo que hará el Robot en cada una (lo que sería más lógico).

// Se pueden agregar más condiciones si se quiere mejorar el algoritmo de movimiento(IA)

// Con este simple algoritmo estamos haciendo que el robot busque siempre pasar obstáculos y buscar la zona que le permita avanzar mejor y más cómodo

/\* Condiciones de viraje de Servos \*/

if (dist\_izq > tolerancia\_sup && dist\_der > tolerancia\_sup) {

// avanzar

estado = 1;

}

else {

if (dist\_izq < tolerancia\_inf || dist\_der < tolerancia\_inf) {

//retrocede

estado = 2;

}

else {

if (dist\_izq > dist\_der) {

//dobla izq

estado = 3;

movimiento(estado);

delay(random(500,1000));

}

else {

//dobla der

estado = 4;

movimiento(estado);

delay(random(300,1000));

}

}

}

delay(200);

}

/\* Modo manual controlado por Bluetooth \*/

else { // Modo manual

/\* Definimos las letras siguientes por gusto\*/

/\* Si se cambian, no olvidar cambiarlas en la aplicación que controla al robot\*/

if (comando == 'w') {

//avanzar

estado = 1;

}

else if (comando == 'a') {

//Virar izquierda

estado = 3;

}

else if (comando == 'd') {

//Virar derecha

estado = 4;

}

else if (comando == 's') {

//retrocede

estado = 2;

}

else if (comando == 'f') {

//Reposo

estado = 0;

}

}

}

}

/\* Funciones utilizadas \*/

void movimiento(int estado){

if (estado == 0){

ang\_izq = 90;

ang\_der = 90;

}

else if (estado == 1){

ang\_izq = 60;

ang\_der = 120;

}

else if (estado == 2){

ang\_izq = 110;

ang\_der = 70;

}

else if (estado == 3){

ang\_izq = 100;

ang\_der = 100;

}

else if (estado == 4){

ang\_izq = 80;

ang\_der = 80;

}

miServo\_izq.write(ang\_izq);

miServo\_der.write(ang\_der);

}

/\*

Funcion que ejecuta la medicion de distancia y el sonido del buzzer. Es lo mismo que se hizo en

un ejemplo anterior, pero ahora con dos sensores.

\*/

void Dist\_Sensor\_Buzzer() {

sensorizqRead = analogRead(sensorizqPin); //Lectura de la señal del sensor de la izquierda

sensorderRead = analogRead(sensorderPin); //Lectura de la señal del sensor de la derecha

dist\_izq = pow(3027.4 / sensorizqRead, 1.2134); //Transformamos a distancia lo leído

dist\_der = pow(3027.4 / sensorderRead, 1.2134); //Transformamos a distancia lo leído

tiempo\_actual = millis(); //Tiempo de ejecución del programa

intervalo = 18 \* (dist\_izq + dist\_der) - 280 ; //Nuestra función para pitidos del buzzer según distancia

/\* Buzzer \*/

if (intervalo <= 50) { //Condición para evitar números negativos y el siseo del buzzer en pequeñas distancias

intervalo = 50;

}

if (intervalo > 2500) { //Condición para evitar largos períodos de tiempo entre sonidos

intervalo = 2500;

}

if (tiempo\_actual - tiempo\_previo >= intervalo) { //Controlar la rapidez de los pitidos del buzzer

tiempo\_previo = tiempo\_actual;

tone(buzzPin, 4000, 20);

delay(20);

/\*

Es necesario esperar con un delay debido a que la funcion servo interfiere con el funcionamiento

de la funcion tone. Esto se debe a que ambas usan el mismo timer, que es un recurso del microprocesador.

\*/

}

}

Confirmar Navegación

¿Estás seguro/a de que quieres abandonar esta página?

Permanecer en esta página  Abandonar esta página

Confirmar Navegación

¿Estás seguro/a de que quieres abandonar esta página?

Permanecer en esta página  Abandonar esta página