Diseño e implementación de vehículo aspirador y evasor de obstáculos para el traslado de mercancías en el sector industrial – ASPTRA

Santisteban Llaxa Juan Alfredo [(1)]

jsantistebanllaxa08@gmail.com

Universidad Nacional Pedro Ruiz Gallo

Resumen. El objetivo del robot "Vehículo aspirador y evasor de obstáculos para el traslado de mercancías en el sector industrial - ASPTRA" es el reconocimiento y evasión de obstáculos a una distancia media de 25 cm, así también indicará la detección de los obstáculos a través de un buzzer, así como el estado de encendido del aspirador a través de un led rojo y el estado de movimiento con un led verde; ya que como sabemos uno de los principales problemas en la logística es la contaminación por partículas, además del tiempo empleado en el transporte de mercancía en distintas empresas. Tomando en consideración estos problemas y para poder contrarrestarlos, se propone este robot con la finalidad de disminuir el tiempo utilizado en el transporte de mercancía y además ayudar a reducir la contaminación por residuos sólidos.

Así también se implementará la función de seguidor de rutas especificadas y la propia categorización de las rutas por tipo de paquete que se pretende transportar.

En cuanto al diseño del robot, se utilizó la plataforma Tinkercad, en el cual se pueden realizar prototipos de robots en base a herramientas que facilitan su elaboración. Primero se diseñó la base del vehículo, en la cual se encuentra el chasis y las ruedas, lo que facilitará el desplazamiento del robot por diversos lugares para transportar objetos y aspirar.

En el caso del diseño de las conexiones, se utilizó el programa Fritzing, el cual permite hacer uso de componentes y poder conectarlos, de tal manera que podamos tener conocimiento cuáles serían los pines y materiales que se utilizarán en el proyecto.

El robot transportador de objetos y aspirador consta de módulos que serán controlados por un aplicativo móvil a través de conexión bluetooth. Y para el funcionamiento de estos, se utilizó el programa Arduino, en el cuál podemos programar cada acción que realizarán cada uno de ellos.

Las necesidades de hoy en día exigen a la logística ofrecer un servicio rápido y eficaz; en muchos casos se ve sobrepasada por las excesivas horas que deben cumplir los trabajadores, lo que genera en consecuencia un bajo rendimiento laboral.

El robot "ASPTRA" pretende igualar las condiciones de logística entre las medianas y pequeñas empresas locales, con las grandes empresas ubicadas en los países desarrollados, además incluiremos el proceso de aspirar el polvo y suciedad durante el recorrido y de esta manera automatizada, ahorrar el tiempo dedicado a estas tareas.

Abstract. The objective of the robot "Vehicle vacuum cleaner and obstacle avoidance for the transport of goods in the industrial sector - ASPTRA" is the recognition and avoidance of obstacles at an average distance of 25 cm, and also indicate the detection of obstacles through a buzzer, as well as the power status of the vacuum cleaner through a red LED and the state of movement with a green LED; because as we

know one of the main problems in the country is pollution by garbage, in addition to the time spent in the transport of goods in different companies. Taking into consideration these problems and in order to counteract them, this robot is proposed in order to reduce the time used in the transport of goods and also help reduce solid waste pollution.

Thus, the function of following specified routes and the categorization of the routes by type of package to be transported will also be implemented.

As for the design of the robot, the Tinkercad platform was used, in which robot prototypes can be made based on tools that facilitate their elaboration. First, the base of the vehicle was designed, in which the chassis and wheels are located, which will facilitate the movement of the robot through different places to transport objects and vacuum.

In the case of the design of the connections, we used the Fritzing program, which allows us to make use of components and connect them, so that we can know what would be the pins and materials to be used in the project.

The robot transporter of objects and vacuum cleaner consists of modules that will be controlled by a mobile application through Bluetooth connection. And for the operation of these, the Arduino program was used, in which we can program each action to be performed by each of them.

Today's needs require logistics to offer a fast and efficient service; in many cases it is overwhelmed by the excessive hours that workers must meet, which consequently generates a low work performance.

The robot "ASPTRA" aims to equalize the conditions of logistics between medium and small local companies, with large companies located in developed countries, also include the process of vacuuming dust and dirt during the journey and in this automated way, save time spent on these tasks.

Palabras clave: Robot, recolector, aspirador, Arduino, Tinkercad, Fritzing, Bluetooth

1. INTRODUCCIÓN

En un almacén completamente manual, el tiempo destinado a moverse por el almacén para preparar los pedidos puede representar hasta la mitad del tiempo de trabajo de un empleado. Al minimizar el movimiento que necesita efectuar una persona para realizar su trabajo, se incrementará la eficiencia del almacén y de la propia empresa hasta un 80% además de reducir la mano de obra. (Anaya Tejero, 2017)

El transporte y la logística es un sector muy complejo que tiene un impacto muy significativo en los precios, el medio ambiente y el consumo de energía. Si la globalización implica transportar cada vez más productos a mayores distancias, el manejo óptimo de todos los recursos implicados puede no sólo significar mejores resultados financieros, sino la supervivencia de la propia empresa. (Dorta González, 2013)

Según la ONU en el 2015, los residuos de envases plásticos representaron el 47% de los residuos plásticos generados en todo el mundo, de los cuales la mitad parece haber provenido de Asia. Mientras que China sigue siendo el mayor generador mundial de residuos de envases plásticos. Estados Unidos es el mayor generador de residuos de envases plásticos per cápita, seguidos por Japón y la Unión Europea. La ciudad de Chiclayo produce al día un promedio de 400 toneladas de residuos sólidos, las que al paralizarse el servicio de limpieza se han acumulado en “cerros” de basura que ocasionan la presencia de moscas y otros

insectos. (Agencia Peruana de Noticias Andina, n.d.)

(Kuo et al., 2014) proponen la construcción de un robot de limpieza que proporciona servicios de monitorización remota inalámbrica. Diseñaron e implementaron un sistema que integra el robot móvil en casa con el teléfono/Tablet PC mediante la modificación de un robot de limpieza comercial.

2. MATERIALES Y MÉTODOS

El proyecto tiene como finalidad diseñar e implementar un robot aspirador y transportador de objetos, que de acuerdo a parámetros que serán controlador por una aplicación Android conectado por Bluetooth y por tanto el robot será capaz de realizar ciertas acciones como, desplazarse, transportar y aspirar.

A continuación, se detallan los módulos con los que cuenta este proyecto:

a) Módulo de locomoción

Este módulo permite el movimiento a través de motorreductores en conjunto con las ruedas, controlados a partir del componente Driver L298N. Persiguiendo las funcionalidades de, ir hacia adelante, hacia atrás, derecha e izquierda, así como ir en retroceso, y finalmente detenerse. Dicho driver y por consiguiente los motorreductores serán alimentados a través de una batería externa.

b) Módulo de detección de obstáculos

Este módulo permite obtener información del entorno donde el vehículo se va a desplazar, en concreto, permite detectar ciertos cuerpos, paredes, u obstáculos en general, a través del sensor ultrasónico HC-SR04. Este sensor captará los obstáculos a una distancia menor a 25 cm, lo que permitirá tomar una nueva opción de desplazamiento al vehículo.

c) Módulo de aspiración

Este módulo permite recoger impurezas, polvo y demás, del suelo, en un área específica del recorrido; a través de un 'Fan PC'. Este módulo será alimentado a través de una batería externa, además de ser necesario un módulo de relé simple para esta alimentación.

d) Módulo de comunicación

Este módulo permite asignar y delegar órdenes al vehículo, a través del componente módulo Bluetooth HC-05 y su conexión con la aplicación móvil de forma inalámbrica por medio de una señal Bluetooth.

e) Módulo de estado

Este módulo permite mostrar el estado en que se encuentra el robot a través de diferentes indicadores, como un led verde cuando está en proceso de movimiento, un led rojo para conocer el encendido de la aspiradora y también un buzzer, el cual se activará cada vez que el robot detecte un obstáculo.

La metodología se desarrolla en cuatro etapas, desde el diseño del robot, elección de los componentes y sus conexiones, programación lógica y el aplicativo Android.

A. DISEÑO DEL ROBOT

Figura 1. Vista Frontal

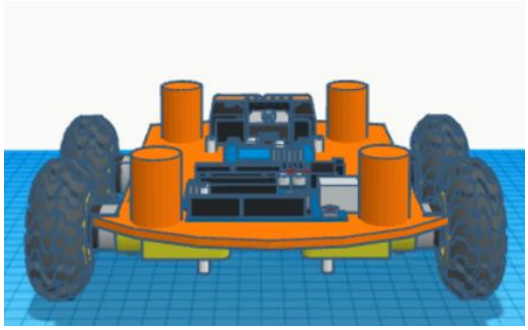


Figura 2. Vista Lateral

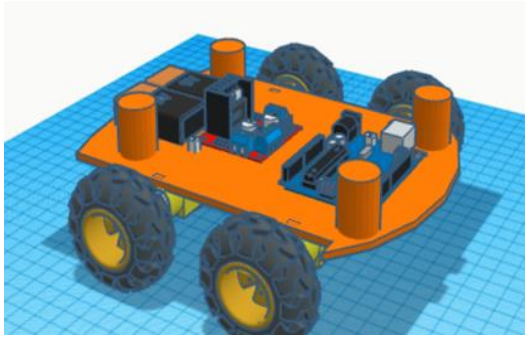


Figura 3. Vista Trasera

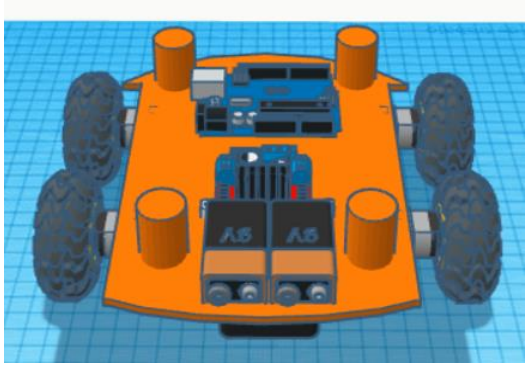


Figura 4. Vista Inferior

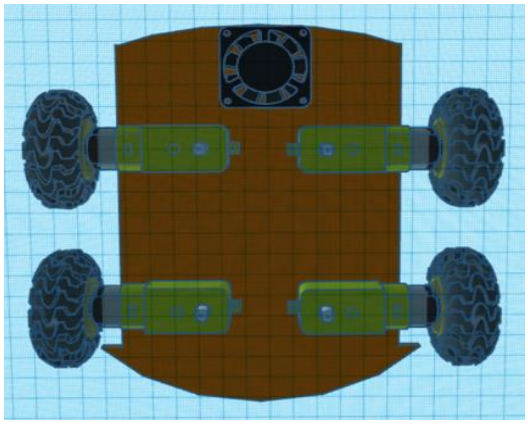


Figura 5. Vista Frontal Completa

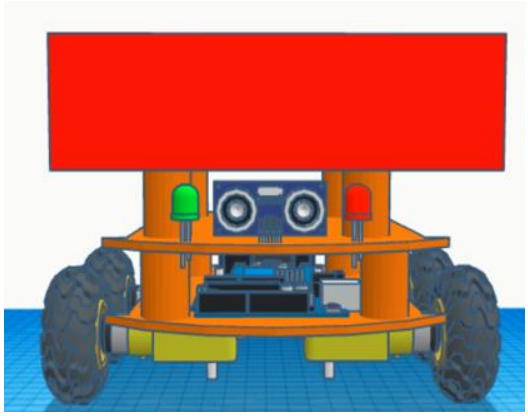
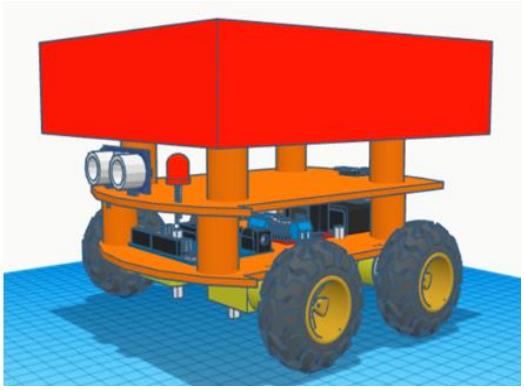


Figura 6. Vista Lateral Completa



B. HARDWARE Y CONEXIONES FÍSICAS

Basado en los requerimientos del proyecto, ASPTRA, se seleccionan los elementos a utilizar.

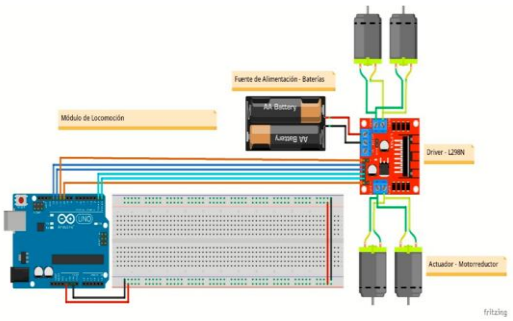
El prototipo será de un vehículo con un microcontrolador Arduino Uno como centro principal del robot. Se le sumará el sensor ultrasónico para percibir los obstáculos presentes delante de él; además, de principales actuadores, como un ventilador, buzzer, leds, los motorreductores y llantas; sin olvidar de los principales módulos, teniendo al Driver L298N, módulo Bluetooth y módulo relé simple.

A continuación, se detallan las conexiones por cada módulo incluido en el robot.

- **Conexión del módulo de locomoción**
  La primera conexión de este módulo se dará entre el Driver L298N y sus pines 1N1, 1N2, 1N3, 1N4 con el Arduino y sus pines digitales 13,12,1,0, respectivamente que se encargarán del sentido del movimiento de los motorreductores, además, los pines ENA y ENB del driver serán conectados a los pines digitales 11 y 10 respectivamente, los cuales se encargarán de la velocidad del movimiento de los motorreductores. Seguidamente los motorreductores que albergarán las ruedas del lado izquierdo

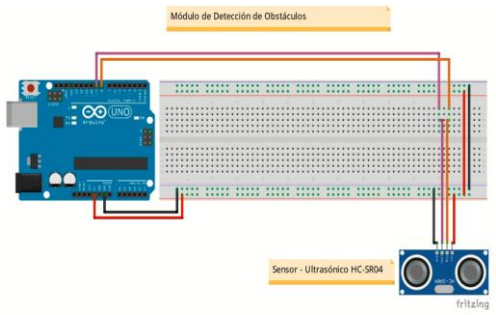
estarán conectados al Driver L298N por medio de los pines OUT1 y OUT2; los motorreductores del lado derecho estarán conectados a los pines OUT3 y OUT4. Por último, el driver L298N será alimentado por una batería externa a través de sus pines 12V y GND.

Figura 7. Diagrama del módulo de locomoción



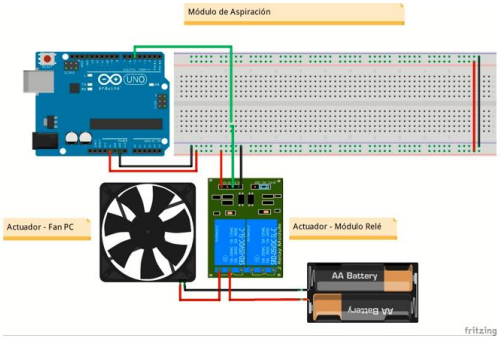
- **Conexión del módulo de detección de obstáculos**
  La conexión se dará entre el sensor ultrasónico HC-SR04 a través de sus pines Vcc y GND conectados al protoboard en sus líneas respectivas; el pin ECHO irá conectado al pin digital 9 de Arduino y el pin TRIG irá conectado al pin 8.

Figura 8. Diagrama del módulo de detección de obstáculos



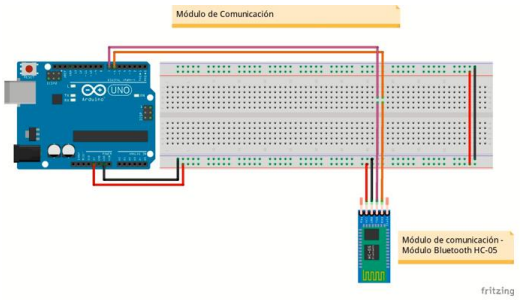
- **Conexión del módulo de aspiración**
  La conexión se dará entre el Fan PC y su fuente de alimentación externa, primero el pin GROUND del Fan PC hacia el negativo de la fuente y el pin VOLTAGE tendrá de intermediario al módulo relé. Este módulo sí se conectará a través de su pin K-In a la fuente externa y el pin K-On al pin VOLTAGE del Fan PC, además sus pines GND y Vcc irán conectados a sus líneas respectivas del protoboard y su pin IN1 se conectará al Arduino en su pin digital 5.

Figura 9. Diagrama del módulo de aspiración



- **Conexión del módulo de comunicación**
  La conexión se dará entre el módulo Bluetooth HC-05 y sus pines Vcc y GND a sus líneas respectivas del protoboard y sus pines TXD y RXD irán conectados al Arduino en sus pines digitales 7 y 6.

Figura 10. Diagrama del módulo de comunicación



- **Conexión del módulo de estado**
  Este módulo dará las indicaciones a través del led rojo que irá conectado al pin digital 4 del Arduino, el led verde que irá conectado al pin 2 y el buzzer irá conectado al pin 3; estos tres actuadores

además conectarán su pin GND a su línea respectiva del protoboard.

Figura 11. Diagrama del módulo de estado

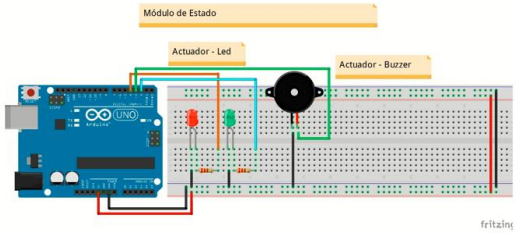
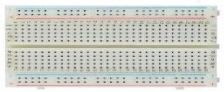
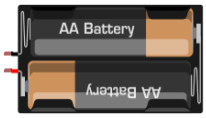


Tabla 1. Elementos seleccionador para el prototipo

Componentes	Imagen Referencial
Microcontrolador Arduino Uno	
HC-SR04	
Fan PC	
Buzzer	
Módulo Relé 5V	
Driver Puente H L298N	
Motorreductor	
Ruedas	
Leds	
HC - 05	
Protoboard	
Pilas 1.5 V	
Baterías 9V	

C. PROGRAMACIÓN LÓGICA

Se diseña el diagrama de flujo por cada módulo y se desarrolla el código, por cada módulo del prototipo, siguiendo la lógica del módulo y los elementos seleccionados.

Figura 12. Diagrama de flujo del módulo de locomoción

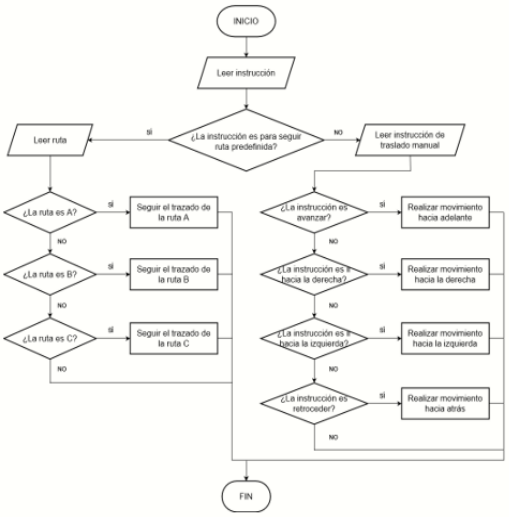


Figura 13. Código del módulo de locomoción

```
// MOD. LOCOMOCION: CREACION DE FUNCIONES
void vehiculoAvanza(int velocidad){
    // funcion para giro antihorario de los 4 motorreductores
  encenderLedVerde();

  analogWrite(ENA, velocidad);          // velocidad mediante PWM en ENA
  digitalWrite(IN1, LOW);               // IN1 a cero logico
  digitalWrite(IN2, HIGH);              // IN2 a uno logico
  analogWrite(ENB, velocidad);          // velocidad mediante PWM en ENB
  digitalWrite(IN3, LOW);               // IN3 a cero logico
  digitalWrite(IN4, HIGH);              // IN4 a uno logico
}
```

```
void vehiculoVolteaIzquierda(int velocidad){
  // funcion para giro antihorario de los motores del lado derecho
  encenderLedVerde();

  analogWrite(ENA, velocidad);              // velocidad mediante PWM en ENA
  digitalWrite(IN1, LOW);                   // IN1 a cero logico
  digitalWrite(IN2, HIGH);                  // IN2 a uno logico
  analogWrite(ENB, 0);
                    // deshabilita motores del lado izquierdo
}

void vehiculoVolteaDerecha(int velocidad){
    // funcion para giro antihorario de los motores del lado izquierdo
  encenderLedVerde();

  analogWrite(ENB, velocidad);              // velocidad mediante PWM en ENB
  digitalWrite(IN3, LOW);                   // IN3 a cero logico
  digitalWrite(IN4, HIGH);                  // IN4 a uno logico
  analogWrite(ENA, 0);                      // deshabilita motores del lado derecho
}

void vehiculoRetrocede(int velocidad){
       // funcion para giro horario de los 4 motorreductores
  encenderLedVerde();

  analogWrite(ENA, velocidad);              // velocidad mediante PWM en ENA
  digitalWrite(IN1, HIGH);                  // IN1 a cero logico
  digitalWrite(IN2, LOW);                   // IN2 a uno logico
  analogWrite(ENB, velocidad);              // velocidad mediante PWM en ENB
  digitalWrite(IN3, HIGH);                  // IN3 a cero logico
  digitalWrite(IN4, LOW);                   // IN4 a uno logico
}

void vehiculoDetente(){
                // funcion que detiene los 4 motorreductores
  analogWrite(ENA, 0);                      // deshabilita motores del lado derecho
  analogWrite(ENB, 0);
                    // deshabilita motores del lado izquierdo

  apagarLedVerde();
}

void vehiculoRutaA(){           // creacion de ruta a seguir llamada 'A'
  vehiculoAvanza(VelAvance);
  delay(2000);
  vehiculoVolteaIzquierda(VelGiro);
  delay(600);
  vehiculoAvanza(VelAvance);
  delay(1500);
  vehiculoVolteaDerecha(VelGiro);
  delay(1400);
  vehiculoAvanza(VelAvance);
  delay(6000);
  vehiculoDetente();
}

void vehiculoRutaB(){           // creacion de ruta a seguir llamada 'B'
  vehiculoRetrocede(VelAvance);
  delay(1500);
  vehiculoVolteaIzquierda(VelGiro);
  delay(2100);
  vehiculoAvanza(VelAvance);
  delay(3000);
  vehiculoDetente();
}

void vehiculoRutaC(){           // creacion de ruta a seguir llamada 'C'
  vehiculoAvanza(VelAvance);
  delay(2000);
  vehiculoVolteaDerecha(VelGiro);
  delay(1050);
  vehiculoAvanza(VelAvance);
  delay(1500);
  vehiculoVolteaIzquierda(VelGiro);
  delay(500);
  vehiculoAvanza(VelAvance);
  delay(2500);
  vehiculoDetente();
}
```

Figura 14. Diagrama de flujo del módulo de detección de obstáculos

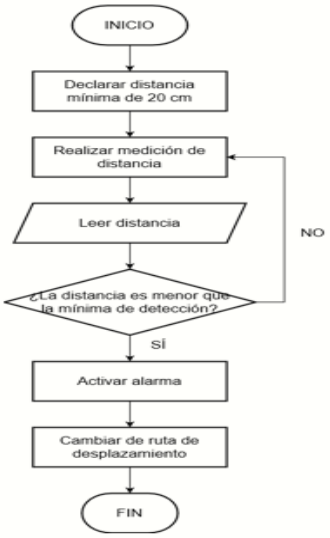


Figura 15. Código del módulo de detección de obstáculos

```
// MOD. DETECCION DE OBSTACULOS: CREACION DE FUNCIONES
long obtenerDistancia(){
        // creacion de ruta a seguir llamada 'B'
  digitalWrite(pinTRIG, HIGH);
  delay(1);
  digitalWrite(pinTRIG, LOW);
           // generacion del pulso a enviar al pin conectado al
 trigger del sensor

  DURACION = pulseIn(pinECHO, HIGH);
     // con funcion pulseIn se espera un pulso alto en echo

  DISTANCIA = DURACION / 58.2;
           // distancia medida en centimetros

  return DISTANCIA;
}
```

Figura 16. Diagrama de flujo del módulo de aspiración

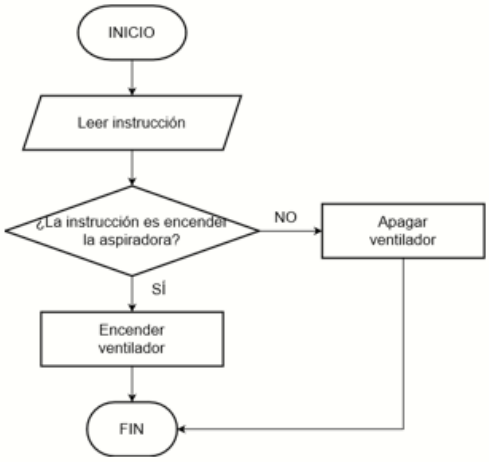


Figura 17. Código del módulo de aspiración

```
// MOD. ASPIRACION: CREACION DE FUNCIONES
void encenderFan(){
                    // creacion de funcion con objetivo: ENCENDER EL VENTILADOR
  digitalWrite(pinFAN, HIGH);
  encenderLedRojo();
}
void apagarFan(){
                    // creacion de funcion con objetivo: APAGAR EL VENTILADOR
  digitalWrite(pinFAN, LOW);
  apagarLedRojo();
}
```

Figura 18. Diagrama de flujo del módulo de comunicación

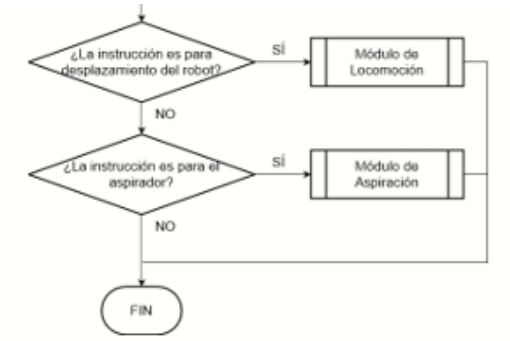


Figura 19. Código del módulo de comunicación – Configuración

```ino
#include <SoftwareSerial.h>

SoftwareSerial miBT(7, 6);
    // establecer pines TXD al pin 7 y RXD al pin 6

void setup() {
  Serial.begin(9600);
        // asignar velocidad de comunicacion con el Serial
  miBT.begin(38400);
         // asignar velocidad de comunicacion con HC-05
}

void loop() {
  if(miBT.available()){
    Serial.write(miBT.read());    // lee HC-05 y envia a Arduino
  }

  if(Serial.available()){
    miBT.write(Serial.read());    // lee Arduino y envia a HC-05
  }
}
```

Figura 20. Diagrama de flujo del módulo de estado

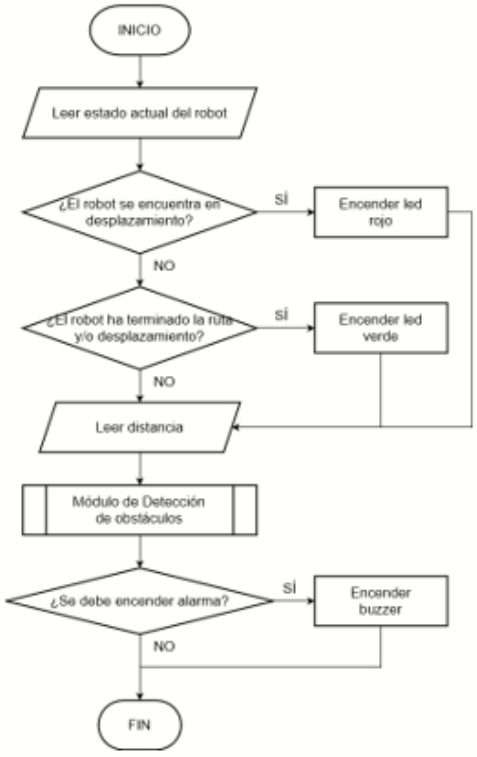


Figura 21. Código del módulo de estado

```ino
// MOD. ESTADO: CREACION DE FUNCIONES
void buzzerParpadea(){
                  // creacion de funcion con objetivo: ENCENDER EL BUZZER
  analogWrite(pinBUZZER, 60);             // encender el buzzer con un voltaje de 1.8v
  delay(1000);                            // demora de 1.5 segundos
  analogWrite(pinBUZZER, 0);              // apagar buzzer
  delay(200);                             // demora de 0.5 segundos
  analogWrite(pinBUZZER, 60);             // encender el buzzer con un voltaje de 1.8v
  delay(1000);                            // demora de 1.5 segundos
  analogWrite(pinBUZZER, 0);              // apagar buzzer
  delay(200);                             // demora de 0.5 segundos
}
void encenderLedRojo(){
                  // creacion de funcion con objetivo: ENCENDER EL LED RED
  analogWrite(pinLedRojo, 250);         // encender Led Rojo
}
void apagarLedRojo(){
                  // creacion de funcion con objetivo: APAGAR EL LED RED
  analogWrite(pinLedRojo, 0);           // apagar Led Rojo
}
void encenderLedVerde(){
                  // creacion de funcion con objetivo: ENCENDER EL LED GREEN
  analogWrite(pinLedVerde, 200);        // encender Led Verde
}
void apagarLedVerde(){
                  // creacion de funcion con objetivo: AOAGAR EL LED GREEN
  analogWrite(pinLedVerde, 0);          // apagar Led Verde
}
```

Figura 22. Código integrado del robot

```ino
#include <SoftwareSerial.h>

// MOD. LOCOMOCION: DECLARACION DE PINES COMO CONSTANTES
const int IN1 = 13;               // L298N: pin IN1 a pin digital 13
const int IN2 = 12;               // L298N: pin IN2 a pin digital 12
const int ENA = 11;               // L298N: pin ENA a pin digital PWM 11
const int IN3 = 2;                // L298N: pin IN3 a pin digital 1
const int IN4 = 4;                // L298N: pin IN4 a pin digital 0
const int ENB = 10;               // L298N: pin ENA a pin digital PWM 10
const int VelAvance = 110;        // constante para la velocidad de avance
const int VelGiro = 90;           // constante para la velocidad de giro
const int VelRetroceso = 70;      // constante para la velocidad de retroceso
// MOD. DETECCION DE OBSTACULOS: DECLARACION DE PINES COMO CONSTANTES
const int pinTRIG = 8;            // HC-SR04: pin trigger a digital pin 8
const int pinECHO = 9;            // HC-SR04: pin echo a digital pin 9
const long distMinima = 25.0;     // DISTANCIA MINIMA ESTABLECIDA
// MOD. ASPIRACION: DECLARACION DE PINES COMO CONSTANTES
const int pinFAN = 5;
          // FAN-PC a traves de MODULO RELE: pin 5 en pin 5
// MOD. COMUNICACION: DECLARACION DE PINES COMO CONSTANTES
SoftwareSerial miBT(7, 6);        // HC-05: pin TXD al pin 7 y RXD al pin 6
// MOD. ESTADO: DECLARACION DE PINES COMO CONSTANTES
const int pinLedRojo = 14;        // LED RED: pin voltage a pin digital 4
const int pinLedVerde = 15;       // LED GREEN: pin voltage a pin digital 4
const int pinBUZZER = 3;          // BUZZER: pin voltage a digital PWM pin 3

// CONSTANTES PARA CONTROL POR APLICATIVO
const char cstControlManual = 'q';
const char cstControlAutomatico = 'e';
const char cstAvanza = 'w';
const char cstDerecha = 'd';
const char cstIzquierda = 'a';
const char cstRetrocede = 's';
const char cstDetente = 'x';
const char cstRutaA = '1';
const char cstRutaB = 'm';
const char cstRutaC = 'n';
const char cstAspiradorON = 'o';
const char cstAspiradorOFF = 'p';

// DECLARACION DE VARIABLES
int DURACION;
                  // variable para almacenar valor de duracion de pulso
long DISTANCIA;                   // variable para almacenar valor de distancia
char DATO = 0;
                // variable para almacenar valor de dato a recibir del HC-05

void setup(){
  Serial.begin(9600);
        // inicializacion de comunicacion serial a 9600 bps
  miBT.begin(38400);
        // inicializacion velocidad de comunicacion con HC-05
  //MOD. LOCOMOCION: DECLARACION DE TIPO
  pinMode(IN1, OUTPUT);             // IN1 como salida
  pinMode(IN2, OUTPUT);             // IN2 como salida
  pinMode(ENA, OUTPUT);             // ENA como salida
  pinMode(IN3, OUTPUT);             // IN3 como salida
  pinMode(IN4, OUTPUT);             // IN4 como salida
  pinMode(ENB, OUTPUT);             // ENB como salida
  //MOD. DETECCION DE OBSTACULOS: DECLARACION DE TIPO
  pinMode(pinTRIG, OUTPUT);         // trigger como salida
  pinMode(pinECHO, INPUT);          // echo como entrada
  pinMode(pinBUZZER, OUTPUT);       // buzzer como salida
  //MOD. ASPIRACION: DECLARACION DE TIPO
  pinMode(pinFAN, OUTPUT);          // fan como salida
  //MOD. ESTADO: DECLARACION DE TIPO
  pinMode(pinLedRojo, OUTPUT);      // LED RED como salida
  pinMode(pinLedVerde, OUTPUT);     // LED GREEN como salida
}

void loop(){
  if(miBT.available()){
        // condicional para constatar comunicacion de HC-05 con aplicativo
    char DATO_NUEVO = miBT.read();
    if(DATO != DATO_NUEVO){
      DATO = DATO_NUEVO;
        // asignar la lectura del dato o la variable enviada por aplicativo
    }

    if(DATO == cstAspiradorON){
      encenderFan();
    }

    if(DATO == cstAspiradorOFF){
      apagarFan();
    }
```

```
    //LOCOMOCION AUTOMATICA
    // Seguir ruta A
      if(DATO == cstRutaA){
        vehiculoRutaA();
      }
    // Seguir ruta B
      if(DATO == cstRutaB){
        vehiculoRutaB();
      }
    // Seguir ruta C
      if(DATO == cstRutaC){
        vehiculoRutaC();
      }

    //LOCOMOCION MANUAL
    char DAT_TEMP;
    // Avanzar
      if(DATO == cstAvanza){
        DISTANCIA = obtenerDistancia();
        Serial.println(DISTANCIA);
        while(DISTANCIA > distMinima){
          vehiculoAvanza(VelAvance);
          DISTANCIA = obtenerDistancia();
          DAT_TEMP = miBT.read();
            if(DAT_TEMP == cstDerecha || DAT_TEMP == cstIzquierda || DAT_TEMP ==
cstRetrocede || DAT_TEMP == cstDetente){
              DISTANCIA = -1.0;
            }
          delay(200);
        }
        if(DISTANCIA < distMinima && DISTANCIA > 0.0){
          vehiculoDetente();
          buzzerParpadea();
        }
      }
      if(DTSTANCIA == -1.0){
        DATO = DAT_TEMP;
        DTSTANCIA = obtenerDistancia();
      }
    delay(1);
    // Giro Izquierda
      if(DATO == cstIzquierda){
        DISTANCIA = obtenerDistancia();
        while(DTSTANCIA > distMinima){
          vehiculoVolteaIzquierda(VelGiro);
          DTSTANCIA = obtenerDistancia();
          DAT_TEMP = miBT.read();
            if(DAT_TEMP == cstAvanza || DAT_TEMP == cstDerecha || DAT_TEMP ==
cstRetrocede || DAT_TEMP == cstDetente){
              DTSTANCIA = -1.0;
            }
          delay(200);
        }
        if(DTSTANCIA < distMinima && DTSTANCIA > 0.0){
          vehiculoDetente();
          buzzerParpadea();
        }
      }
      if(DISTANCIA == -1.0){
        DATO = DAT_TEMP;
        DISTANCIA = obtenerDistancia();
      }
    delay(1);
    // Giro Derecha
      if(DATO == cstDerecha){
        DTSTANCIA = obtenerDistancia();
        while(DISTANCIA > distMinima){
          vehiculoVolteaDerecha(VelGiro);
          DISTANCIA = obtenerDistancia();
          DAT_TEMP = miBT.read();
            if(DAT_TEMP == cstAvanza || DAT_TEMP == cstIzquierda || DAT_TEMP ==
cstRetrocede || DAT_TEMP == cstDetente){
              DISTANCIA = -1.0;
            }
          delay(200);
        }
        if(DISTANCIA < distMinima && DISTANCIA > 0.0){
          vehiculoDetente();
          buzzerParpadea();
        }
      }
      if(DTSTANCIA == -1.0){
        DATO = DAT_TEMP;
        DTSTANCIA = obtenerDistancia();
      }
    delay(1);
    // Retroceder
      if(DATO == cstRetrocede){
        vehiculoRetrocede(VelRetroceso);
      }
    delay(1);
    // Detener
      if(DATO == cstDetente){
        vehiculoDetente();
      }
    delay(1);
  }
}


// MOD. LOCOMOCION: CREACION DE FUNCIONES
void vehiculoAvanza(int velocidad){
    // funcion para giro antihorario de los 4 motorreductores
  encenderLedVerde();

  analogWrite(ENA, velocidad);          // velocidad mediante PWM en ENA
  digitalWrite(IN1, LOW);               // IN1 a cero logico
  digitalWrite(IN2, HIGH);              // IN2 a uno logico
  analogWrite(ENB, velocidad);          // velocidad mediante PWM en ENB
  digitalWrite(IN3, LOW);               // IN3 a cero logico
  digitalWrite(IN4, HIGH);              // IN4 a uno logico
}

void vehiculoVolteaIzquierda(int velocidad){
  // funcion para giro antihorario de los motores del lado derecho
  encenderLedVerde();

  analogWrite(ENA, velocidad);                // velocidad mediante PWM en ENA
  digitalWrite(IN1, LOW);                     // IN1 a cero logico
  digitalWrite(IN2, HIGH);                    // IN2 a uno logico
  analogWrite(ENB, 0);
                    // deshubilita motores del lado izquierdo
}

void vehiculoVolteaDerecha(int velocidad){
    // funcion para giro antihorario de los motores del lado izquierdo
  encenderLedVerde();

  analogWrite(ENB, velocidad);                // velocidad mediante PWM en ENB
  digitalWrite(IN3, LOW);                     // IN3 a cero logico
  digitalWrite(IN4, HIGH);                    // IN4 a uno logico
  analogWrite(ENA, 0);
                    // deshabilita motores del lado derecho
}

void vehiculoRetrocede(int velocidad){
      // funcion para giro horario de los 4 motorreductores
  encenderLedVerde();

  analogWrite(ENA, velocidad);                // velocidad mediante PWM en ENA
  digitalWrite(IN1, HIGH);                    // IN1 a cero logico
  digitalWrite(IN2, LOW);                     // IN2 a uno logico
  analogWrite(ENB, velocidad);                // velocidad mediante PWM en ENB
  digitalWrite(IN3, HIGH);                    // IN3 a cero logico
  digitalWrite(IN4, LOW);                     // IN4 a uno logico
}

void vehiculoDetente(){
                  // funcion que detiene los 4 motorreductores
  analogWrite(ENA, 0);
                    // deshabilita motores del lado derecho
  analogWrite(ENB, 0);
                    // deshabilita motores del lado izquierdo

  apagarLedVerde();
}
```

```
void vehiculoRutaA(){           // creacion de ruta a seguir llamada 'A'
  vehiculoAvanza(VelAvance);
  delay(2000);
  vehiculoVolteaIzquierda(VelGiro);
  delay(600);
  vehiculoAvanza(VelAvance);
  delay(1500);
  vehiculoVolteaDerecha(VelGiro);
  delay(1400);
  vehiculoAvanza(VelAvance);
  delay(6000);
  vehiculoDetente();
}

void vehiculoRutaB(){           // creacion de ruta a seguir llamada 'B'
  vehiculoRetrocede(VelAvance);
  delay(1500);
  vehiculoVolteaIzquierda(VelGiro);
  delay(2100);
  vehiculoAvanza(VelAvance);
  delay(3000);
  vehiculoDetente();
}

void vehiculoRutaC(){           // creacion de ruta a seguir llamada 'C'
  vehiculoAvanza(VelAvance);
  delay(2000);
  vehiculoVolteaDerecha(VelGiro);
  delay(1050);
  vehiculoAvanza(VelAvance);
  delay(1500);
  vehiculoVolteaIzquierda(VelGiro);
  delay(500);
  vehiculoAvanza(VelAvance);
  delay(2500);
  vehiculoDetente();
}

// MOD. DETECCION DE OBSTACULOS: CREACION DE FUNCIONES
long obtenerDistancia(){        // creacion de ruta a seguir llamada 'B'
  digitalWrite(pinTRIG, HIGH);
  delay(1);
  digitalWrite(pinTRIG, LOW);
            // generacion del pulso a enviar al pin conectado al trigger
del sensor

  DURACION = pulseIn(pinECHO, HIGH);
      // con funcion pulseIn se espera un pulso alto en echo

  DISTANCIA = DURACION / 58.2;
            // distancia medida en centimetros

  return DISTANCIA;
}

// MOD. ASPIRACION: CREACION DE FUNCIONES
void encenderFan(){
                        // creacion de funcion con objetivo: ENCENDER EL V
ENTILADOR
  digitalWrite(pinFAN, HIGH);
  encenderLedRojo();
}
void apagarFan(){
                        // creacion de funcion con objetivo: APAGAR EL V
ENTILADOR
  digitalWrite(pinFAN, LOW);
  apagarLedRojo();
}

// MOD. ESTADO: CREACION DE FUNCIONES
void buzzerParpadea(){
                        // creacion de funcion con objetivo: ENCENDER EL BUZZ
ER
  analogWrite(pinBUZZER, 60);
            // encender el buzzer con un voltaje de 1.8v
  delay(1000);                          // demora de 1.5 segundos
  analogWrite(pinBUZZER, 0);            // apagar buzzer
  delay(200);                           // demora de 0.5 segundos
  analogWrite(pinBUZZER, 60);
            // encender el buzzer con un voltaje de 1.8v
  delay(1000);                          // demora de 1.5 segundos
  analogWrite(pinBUZZER, 0);            // apagar buzzer
  delay(200);                           // demora de 0.5 segundos
}
void encenderLedRojo(){
                    // creacion de funcion con objetivo: ENCENDER EL LED R
ED
  analogWrite(pinLedRojo, 250);         // encender Led Rojo
}
void apagarLedRojo(){
                    // creacion de funcion con objetivo: APAGAR EL LED R
ED
  analogWrite(pinLedRojo, 0);           // apagar Led Rojo
}
void encenderLedVerde(){
                  // creacion de funcion con objetivo: ENCENDER EL LED GR
EEN
  analogWrite(pinLedVerde, 200);        // encender Led Verde
}
void apagarLedVerde(){
                    // creacion de funcion con objetivo: AOAGAR EL LED GR
EEN
  analogWrite(pinLedVerde, 0);          // apagar Led Verde
}
```

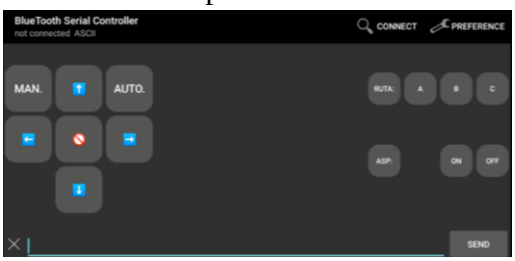
D. APLICATIVO ANDROID

El aplicativo Android que se utilizó para la comunicación por bluetooth fue “BlueTooth Serial Controller”

El cual logró brindar la facilidad de edición para presentación, así como, el correcto envío de los comandos especificados para cada uno de los botones añadidos.

Dentro de los botones añadidos se encuentran divididos en 2 secciones, en el lado izquierdo se observan los botones para arranque manual o automático del robot, además, de los botones principales de movimiento manual; en el lado derecho se encuentran las rutas que el vehículo puede seguir, además de los botones de referencia para encender y apagar el aspirador.

Figura 23. Captura de pantalla de la aplicación

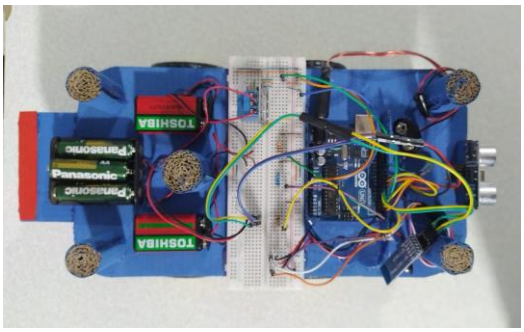


3. RESULTADOS

Como resultado del proceso de investigación y la implementación del proyecto se obtiene el diseño para el prototipo de vehículo aspirador y evasor de obstáculos para el traslado de mercancías en el sector industrial.

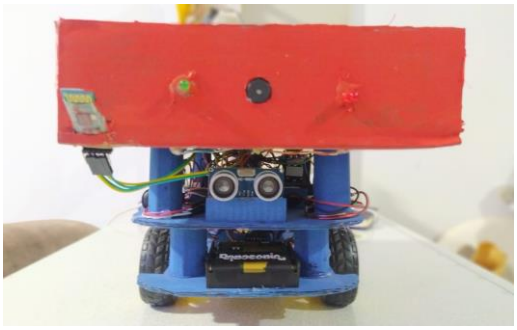
En la figura 24 se puede observar la conexión final de todos los componentes, en la plataforma superior la disposición de las baterías, el microcontrolador Arduino uno, el sensor ultrasónico HC-SR04, así como el módulo bluetooth HC-05; y en la plataforma inferior la conexión de los componentes Driver Puente H L298N y el Fan PC.

Figura 24. Conexión final de componentes en robot ASPTRA



En la figura 25 se muestra el robot terminado, con las conexiones establecidas, además de la lógica de los módulos grabados en el microcontrolador Arduino uno y con un dispositivo Android con la aplicación de control instalada, se procedió a realizar las pruebas del prototipo.

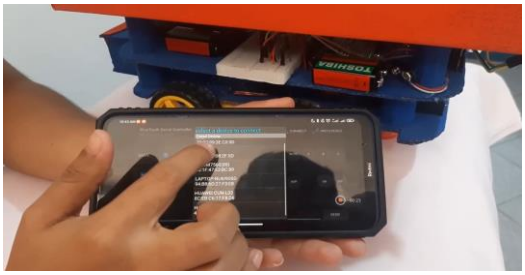
Figura 25. Robot terminado



Las pruebas se realizaron en cada módulo del prototipo todas controlados por medio de bluetooth, desde una aplicación móvil Android; obteniendo los siguientes resultados:

- ✓ En el *módulo de comunicación*, en la figura 26 se observa la conexión en el módulo de comunicación, mediante un aplicativo móvil “BlueTooth Serial Controller” conectado con el módulo Bluetooth HC-05, que permite controlar las funcionalidades del robot.

Figura 26. Conexión de robot ASPTRA con aplicativo móvil a través del módulo Bluetooth



- ✓ En el *módulo de locomoción modo manual*, como observamos en las siguientes figuras, se presenta el cómo nuestro robot ASPTRA logra de manera exitosa avanzar, retroceder, así también girar a la izquierda y derecha de manera manual; pulsando los botones configurados en el aplicativo móvil, según la elección del usuario.

Figura 27. Robot ASPTRA avanzando

Figura 28. Robot ASPTRA girando a la derecha

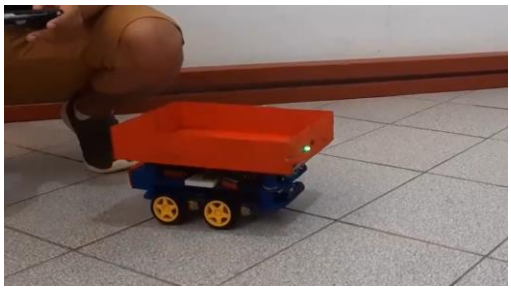


Figura 29. Robot ASPTRA girando a la izquierda

- ✓ En el *módulo de locomoción modo automático*, como observamos en las siguientes figuras, se presenta el cómo nuestro robot ASPTRA sigue las rutas preestablecidas A, B y C en el aplicativo móvil; cada una de estas serán ejecutadas pulsando los botones para rutas configurados.

Figura 30. Robot ASPTRA llegando al punto final de la ruta A

Figura 31. Robot ASPTRA siguiendo la trayectoria para la ruta C

- ✓ En el *módulo de detección de obstáculos*, en la figura se observa cómo el vehículo robot logra detenerse cuando detecta un obstáculo a una distancia no menor de 25 cm.

Figura 32. Robot ASPTRA detectando un obstáculo

- ✓ En el *módulo de aspiración*, en la figura se muestra cómo el vehículo robot logra aspirar de manera exitosa los residuos presentes en cierta área en específico, logrando almacenarlos dentro del compartimento situado en la parte trasera.

Figura 33. Robot ASPTRA luego de haber aspirado la suciedad

- ✓ En el *módulo de estado*, En la figura se observa cómo el vehículo robot al detectar un obstáculo a una distancia no menor de 25 cm, activará el sonido del buzzer; así también se indicará mediante un led rojo el encendido de la aspiradora y cuando el robot está en movimiento mediante un led verde.

Figura 34. Robot ASPTRA con led rojo indicando el prendido del aspirador

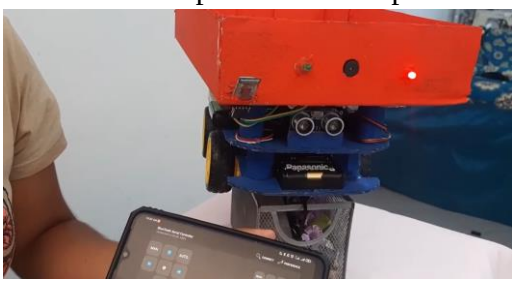


Figura 35. Robot ASPTRA con led verde indicando el movimiento

4. DISCUSIÓN

El prototipo de robot denominado ha sido construido de manera exitosa, además de lograr haber realizado todas las pruebas de funcionamiento con un nivel óptimo, dichas pruebas fueron gestionadas por el microcontrolador Arduino Uno y el aplicativo móvil destinado para su correcto funcionamiento.

Se presentaron algunos inconvenientes con respecto a las diversas fuentes de alimentación, ya sea por caídas de voltaje, así como, por el alto consumo de algunos de los componentes principales, si bien se lograron resolver estos inconvenientes, se espera que existan mejores fuentes de alimentación para lograr una mayor autonomía del robot.

En el futuro se espera mejorar el diseño y la funcionalidad que realiza el robot, por la parte del diseño y arquitectura, se pretende obtener mayor fuerza en los piezas del chasis para lograr movilizar paquetes de mayor peso, así como, mejorar resistencia de los componentes que integran el robot; por la parte de funcionalidad, se podrían utilizar un mayor número de baterías o conseguir baterías recargables con un buen soporte de consumo para los componentes, además, si se obtuvieran mayores pines para entrada digital, se podrían añadir funcionalidades complementarias como el seguidor de líneas, movimiento o añadidos de sensores ultrasónicos para no dejar puntos ciegos al robot, etc.

5. CONCLUSIONES

Este proyecto de robot "Vehículo aspirador y evasor de obstáculos para el traslado de mercancías en el sector industrial - ASPTRA" ha cumplido la función de seguidor de rutas especificadas, traslado a nivel manual y el agregado de ser un vehículo aspirador con el fin de recoger impurezas, dicho proyecto se ha diseñado e implementado con las plataformas Arduino y Android.

Con el prototipo del presente robot se pretende eliminar las excesivas horas que deben cumplir los trabajadores, lo que genera en consecuencia un bajo rendimiento laboral; al presionar diferentes botones los trabajadores tendrán automatizados los procesos que retrasaban el trabajo.

La implementación del proyecto es una muestra de las potencialidades de las plataformas utilizadas (Arduino y Android), en el campo de la automatización de procesos logísticos y de limpieza, específicamente en el uso de vehículos para transporte de mercancías y absorción de impurezas en un almacén logístico.

6. REFERENCIAS BIBLIOGRÁFICAS

Anaya Tejero, J. (2017). Logística Integral. La gestión operativa de la empresa. Madrid: Editorial ESIC.

Dorta González, P. (2013). Transporte y Logística Internacional. España: Universidad de Las Palmas de Gran Canaria.

Kuo, G. H., Cheng, C. Y., & Wu, C. J. (2014). Design and implementation of a remote monitoring cleaning robot. CACS 2014 - 2014 International Automatic Control Conference, Conference Digest, 281–286.