Instituto Politécnico Nacional

Escuela Superior De Computo

Técnicas de Programación para Robots Móviles

Grupo: 7BV1

Fecha de entrega: 24 de marzo de 2025

Profesor: Marco Antonio Ortiz Castillo

Practica 3

Dominguez Beltran Hugo

Olivares Lifshitz Daniel Stanislav

Ramirez Salas Carlos

Lopez Zapata Luis Enrique

Introducción

En esta práctica se implementó un sistema de control de motores DC y la lectura de sensores ultrasónicos para medir distancias en un robot móvil. Se utilizó un microcontrolador Arduino para manejar el control de velocidad y dirección de los motores mediante señales PWM, además de la obtención de distancias usando sensores ultrasónicos laterales (HC-SR04) y un sensor ultrasónico frontal con comunicación I2C.
El objetivo principal fue diseñar y programar un sistema que permita controlar el movimiento del robot (avanzar, retroceder, girar) variando la velocidad y monitoreando distancias para posibles futuras aplicaciones de navegación o evitación de obstáculos.

Descripción del Hardware

El hardware del robot está compuesto por una serie de componentes que permiten que el robot se desplace, tome decisiones basadas en su entorno y se comunique con otros dispositivos.

Microcontroladores y Módulos

Arduino UNO: Microcontrolador principal encargado del procesamiento de señales, control de motores y lectura de sensores ultrasónicos mediante pines digitales y comunicación I2C.

Sensores

Sensores ultrasónicos HC-SR04 (laterales): Utilizados para medir distancias laterales. Funcionan enviando un pulso ultrasónico y midiendo el tiempo de retorno para calcular la distancia al objeto más cercano. Se conectan a pines digitales (TRIG y ECHO) para el disparo y recepción.

Sensor ultrasónico I2C frontal (dirección 0x77): Sensor ultrasónico con comunicación I2C que permite medir distancias frontales con mayor facilidad de integración.

Actuadores

Motores DC (4 unidades): Motores controlados mediante señales PWM para modificar la velocidad, y señales digitales para la dirección de giro.

Driver de motores (asumido basado en control de PWM y pines digitales): Controla la alimentación y dirección de los motores mediante señales digitales y PWM provenientes del Arduino.

Pines

Sensores laterales: TRIG_LEFT (pin 4), ECHO_LEFT (pin 5), TRIG_RIGHT (pin 2), ECHO_RIGHT (pin 3).
Sensor I2C frontal: Dirección I2C 0x77.
Motores: PWM en pines 6, 9, 10, 11; dirección en pines 7, 8, 12, 13.

Descripción del Software

Se utilizó Arduino IDE para programar el microcontrolador en lenguaje C/C++. El código se compiló y cargó directamente al Arduino UNO mediante conexión USB.

```
#include <Arduino.h>
#include <Wire.h>

int contador_pwm  =0;
int index_count = 0;


// Pines sensores laterales (HC-SR04)
#define TRIG_LEFT 4
#define ECHO_LEFT 5
#define TRIG_RIGHT 2
#define ECHO_RIGHT 3

// Dirección del sensor frontal I2C
#define ULTRASOUND_I2C_ADDR 0x77
#define DISDENCE_L 0  // Registro de lectura de distancia

// Pines motores
const static uint8_t pwm_min = 50;
const static uint8_t motorpwmPin[4] = {10, 9, 6, 11};
const static uint8_t motordirectionPin[4] = {12, 8, 7, 13};

// Inicialización de pines de motor
void Motor_Init() {
  for (uint8_t i = 0; i < 4; i++) {
    pinMode(motordirectionPin[i], OUTPUT);
    pinMode(motorpwmPin[i], OUTPUT);
  }
}

// Control de motores: adelante/atrás por motor y velocidad
void SetMotorStates(bool m0, bool m1, bool m2, bool m3, uint8_t velocidad) {
  bool motores[4] = {m0, m1, m2, m3};
  for (uint8_t i = 0; i < 4; ++i) {
    digitalWrite(motordirectionPin[i], motores[i]);
    analogWrite(
```

```
      motorpwmPin[i], //Motor al cual se le esta modificando
      //velocidad == 0 ? 0 : map(velocidad, 0, 100, pwm_min, 255)
      velocidad
      );
  }
}

// Obtener distancia de sensor ultrasónico digital (HC-SR04)
float GetDistanceUltrasonico(uint8_t trig, uint8_t echo) {
  long t;
  float d;
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);
  t = pulseIn(echo, HIGH, 20000);  // Timeout agregado
  if (t == 0) return 9999;
  d = (0.0343 / 2) * t;
  return d;
}

// Obtener distancia del sensor I2C frontal (convertido a cm)
float GetDistanceFrente() {
  uint8_t data[2];
  Wire.beginTransmission(ULTRASOUND_I2C_ADDR);
  Wire.write(DISDENCE_L);
  if (Wire.endTransmission() != 0) return 9999;
  Wire.requestFrom(ULTRASOUND_I2C_ADDR, 2);
  if (Wire.available() < 2) return 9999;
  data[0] = Wire.read();
  data[1] = Wire.read();
  uint16_t distanciaRaw = (uint16_t)data[1] << 8 | data[0];
  return distanciaRaw / 10.0;  // Convertir de mm a cm
}

void setup() {
  Serial.begin(115200);
  Wire.begin();
  pinMode(TRIG_LEFT, OUTPUT);
  pinMode(ECHO_LEFT, INPUT);
  pinMode(TRIG_RIGHT, OUTPUT);
  pinMode(ECHO_RIGHT, INPUT);
  Motor_Init();
}

void loop() {
```

```
contador_pwm = contador_pwm + 1;

if(contador_pwm > 255){
  contador_pwm = 0;
  index_count++;
  delay(1000);
}

if(contador_pwm<90 && index_count ==0){
  Serial.print("PWM:= ");
  Serial.println(contador_pwm);
  delay(50);
  SetMotorStates(1, 0, 0, 1, contador_pwm);

}

if(contador_pwm == 90){
  SetMotorStates(0, 0, 0, 0, 0);
  delay(5000);

}

if(contador_pwm>90){
  if(index_count == 0){

        SetMotorStates(0, 1, 1, 0, contador_pwm); // Hacia Atras
        delay(25);


      }

    if(index_count == 1){
        SetMotorStates(1, 0, 0, 1, contador_pwm); // Hacia Adelante
        delay(25);


      }

    if(index_count == 2){
      SetMotorStates(1, 1, 1, 1, contador_pwm); // Girar a la izquierda
      delay(25);

    }

    if(index_count ==3){
```

```
        SetMotorStates(0, 0, 0, 0, contador_pwm); // Girar a la derecha
        delay(25);

    }
}
```

Funcionamiento General

El código está diseñado para controlar un robot móvil equipado con cuatro motores DC y tres sensores ultrasónicos: dos sensores laterales tipo HC-SR04 conectados por pines digitales y un sensor ultrasónico frontal con comunicación I2C.

Inicialización

En la función setup(), se configuran los pines digitales para los sensores ultrasónicos laterales (pines TRIG y ECHO), se inicializa la comunicación serial a 115200 baudios para monitoreo y se inicia la comunicación I2C con Wire.begin() para el sensor ultrasónico frontal. También se llama a la función Motor_Init() para configurar los pines de control de los motores como salidas.

Control de motores

La función Motor_Init() inicializa cuatro pares de pines para el control de dirección y velocidad (PWM) de los motores, definiendo cada pin como salida.

La función SetMotorStates(bool m0, bool m1, bool m2, bool m3, uint8_t velocidad) recibe cuatro booleanos que representan el estado de dirección de cada motor y un valor de velocidad PWM. Dentro de esta función, se itera sobre los cuatro motores configurando sus pines de dirección mediante digitalWrite y ajustando la velocidad con analogWrite en los pines PWM correspondientes.

Lectura de sensores ultrasónicos

- La función GetDistanceUltrasonico(uint8_t trig, uint8_t echo) mide la distancia utilizando un sensor HC-SR04 conectado por pines digitales. Primero, genera un pulso de disparo (trig en HIGH por 10 microsegundos), luego mide el tiempo que tarda en recibir el eco (pulseIn(echo, HIGH, 20000)). Finalmente, convierte el tiempo medido a distancia en centímetros mediante la fórmula del tiempo de vuelo del sonido.

- La función GetDistanceFrente() lee el sensor ultrasónico frontal usando el protocolo I2C. Envía una solicitud al registro de distancia (DISDENCE_L) y recibe dos bytes que se combinan para obtener la distancia en milímetros, que luego convierte a centímetros.

Lógica principal en loop()

El ciclo principal aumenta progresivamente un contador contador_pwm que representa el valor PWM para controlar la velocidad de los motores. Este contador se incrementa en cada iteración hasta llegar a 255, donde se reinicia a 0 y se incrementa un índice index_count que determina el modo de movimiento del robot.

- Cuando contador_pwm es menor a 90 y index_count es 0, el robot avanza hacia adelante con velocidad proporcional a contador_pwm, activando los motores 0 y 3 en dirección hacia adelante mediante la función SetMotorStates(1, 0, 0, 1, contador_pwm).

- Si contador_pwm alcanza 90, el robot se detiene completamente (SetMotorStates(0, 0, 0, 0, 0)) y espera 5 segundos (delay(5000)).

- Para valores de contador_pwm mayores a 90, el comportamiento cambia dependiendo del valor de index_count:

  - index_count == 0: El robot se mueve hacia atrás activando los motores 1 y 2 con dirección invertida.

  - index_count == 1: El robot avanza de nuevo, activando motores 0 y 3.

  - index_count == 2: El robot gira a la izquierda activando todos los motores en dirección hacia adelante, simulando un giro.

  - index_count == 3: El robot detiene todos los motores (aunque el valor PWM sigue aumentando).

Cada modo mantiene su movimiento durante una pequeña pausa de 25 ms antes de la siguiente iteración del ciclo, para permitir la ejecución progresiva.

Modos de Operación

Incremento progresivo de velocidad: La velocidad PWM incrementa de 0 a 255, modificando la velocidad de los motores.
Movimiento hacia adelante: Motores activados para avanzar con velocidad PWM variable.
Movimiento hacia atrás: Motores activados para retroceder con velocidad PWM variable.
Giros: Motores activados para girar a la izquierda o derecha en función del índice de operación.
Detención: El robot se detiene por un intervalo al alcanzar ciertos valores del contador PWM.

Resultados
El sistema logró controlar exitosamente los motores en diferentes modos de movimiento y variación de velocidad, verificable mediante monitoreo serial del valor PWM. La lectura de sensores ultrasónicos funcionó correctamente, obteniendo distancias en centímetros para los sensores laterales y frontal.

El uso de la comunicación I2C para el sensor ultrasónico frontal simplifica la integración y reduce el uso de pines. La modularización del control de motores mediante funciones facilita la extensión futura para incluir la lógica de evitación de obstáculos.

El bloque comentado muestra una posible extensión para evitar obstáculos utilizando las lecturas ultrasónicas, lo que puede ser implementado en prácticas futuras para lograr autonomía básica.

Conclusiones

Esta práctica permitió implementar un control básico pero funcional de un robot móvil, integrando el manejo de motores DC con modulación PWM y la adquisición de datos de sensores ultrasónicos. Se comprobó la capacidad del Arduino para controlar múltiples motores y sensores simultáneamente, y la flexibilidad del entorno Arduino IDE para programar funciones de control y lectura.

La experiencia destaca la importancia de una correcta configuración de pines y parámetros de comunicación para lograr una integración efectiva de hardware y software en sistemas robóticos móviles.

La práctica sienta las bases para desarrollar algoritmos más complejos de navegación autónoma basados en las mediciones de distancia y el control de velocidad y dirección.

Bibliografía

1. **Arduino. (n.d.).** *Arduino - Getting Started*.

   - Recuperado de https://www.arduino.cc/en/Guide/HomePage
   - Esta documentación oficial de Arduino ofrece una introducción al entorno de desarrollo y a la programación de placas Arduino, como el Arduino UNO utilizado en el robot.