**WORLD ROBOT OLYMPIAD**

**FUTURES ENGINEERS**

**EQUIPO: “LOS MÁQUINAS”**

**GITHUB: # MADRID-LOS-MÁQUINAS**

WRO Futures Engineers: Coche Autónomo: Modelo: LEGO-Arduino (Chasis V2).

**Alumnos:**

Marcos Gomez Brito

Leo Tejada Eslava

Miguel Martínez Simarro

Imagen que contiene interior, tabla, pequeño, cámara

Descripción generada automáticamente

**1.- Introducción:**

Se trata de un vehículo construido con piezas de LEGO Technics para el Chasis y LEGO Mindstorms EV3 y Arduino para su control.

La propulsión la realizará con un motor mediano de LEGO y la dirección de cremallera estará controlada por una placa Arduino UNO y un servo.

El microcontrolador de EV3 para la propulsión cuenta con un procesador ARM9 de 300 MHz y 64 MB de RAM.

El microcontrolador de Arduino UNO es el ATMEL 328P con Procesador a 16MHz de 8 bits de, 32Kb de Memoria Flash, 2Kb de Memoria SRAM y 1Kb de Memoria persistente EEPROM.

Utilizará sensores ultrasónicos para hacer el WallFollower y una cámara Pixy inteligente para detectar los objetos y evitarlos, haciendo así lo que llamamos Slalom de pilares Verdes y Rojos. La programación esta hecha con Arduino para la Dirección y EV3-G de LabView para la propulsión.

**2.- Material utilizado:**

1 Microcontrolador: Brick de Lego MIndstorm EV3 procesador ARM9 de 300 MHz y 64 MB de RAM para la Propulsión.

1 microcontrolador de Arduino UNO es el ATMEL 328P para la Dirección.

1 Motor mediano de Lego Mindstorms

1 microservo para la Dirección.

3 sensores Ultrasónicos

1 Sensor Giróscopo

Piezas de LEGO Technics, incluyendo un diferencial para la propulsión trasera.

Cámara Pixy v2.

**3.- Desarrollo de la Ingeniería:**

Para crear un programa en Arduino que permita a un robot sortear obstáculos de color rojo a la izquierda y verde a la derecha utilizando una cámara Pixy y al mismo tiempo rodear una pared con un sensor ultrasónico, necesitarás combinar el uso de dos sensores y escribir un código que tome decisiones basadas en la información proporcionada por estos sensores. A continuación, te proporcionaré un esquema general de cómo podrías abordar este proyecto, pero ten en cuenta que la implementación exacta dependerá de tu configuración y hardware específicos.

1. Configura la Pixy Camera:
   * Conecta la cámara Pixy a tu Arduino utilizando el protocolo de comunicación que prefieras (por ejemplo, I2C, UART, SPI).
   * Instala la biblioteca de Pixy en el entorno de desarrollo de Arduino para poder comunicarte con la cámara.
   * Configura la Pixy para que detecte objetos rojos y verdes. Esto generalmente se hace mediante la configuración de filtros de color en la cámara.
2. Configura el Sensor Ultrasónico:
   * Conecta el sensor ultrasónico a tu Arduino. El sensor ultrasónico generalmente tiene dos pines: uno para el envío de pulsos ultrasónicos y otro para recibir la señal de eco.
   * Utiliza una biblioteca de Arduino, como la biblioteca "NewPing", para medir la distancia desde el sensor ultrasónico a la pared.
3. Escribe el programa:
   * En el bucle principal del programa, obtén información de la Pixy Camera para detectar objetos rojos y verdes. Puedes utilizar la biblioteca de Pixy para esto.
   * Utiliza las coordenadas de los objetos detectados para determinar si un objeto rojo está a la izquierda y un objeto verde está a la derecha del robot.
   * Utiliza el sensor ultrasónico para medir la distancia a la pared frente al robot.
   * Basándote en la información de la Pixy y el sensor ultrasónico, toma decisiones sobre cómo mover el robot para evitar los obstáculos y rodear la pared.
   * Puedes utilizar motores y controladores de motores para controlar el movimiento del robot.

Aquí tienes el bucle principal de tu programa:

#include <Pixy.h>

#include <NewPing.h>

Pixy pixy;

NewPing sonar(triggerPin, echoPin);

void setup() {

// Inicializa la Pixy y el sensor ultrasónico aquí

}

void loop() {

// Obtén información de la Pixy

pixy.update();

// Lee la distancia desde el sensor ultrasónico

int distancia = sonar.ping\_cm();

// Procesa la información y toma decisiones de movimiento

if (pixy.getBlocks()) {

// Verifica la posición de los objetos y la distancia a la pared

// Toma decisiones para evitar obstáculos y rodear la pared

}

// Controla el movimiento del robot en función de tus decisiones

}

Ten en cuenta que este es solo un esquema general, y la lógica específica para sortear obstáculos y rodear la pared dependerá de tus requisitos y de la configuración de tu robot. Deberás ajustar y personalizar el código para que se adapte a tu hardware y necesidades particulares. Además, ten en cuenta las limitaciones de tiempo de procesamiento y memoria de Arduino al diseñar tu programa.

El código usado es:

#include <Pixy2.h>

#include <Servo.h>

// This is the main Pixy object

Pixy2 pixy;

Servo direccion;

const int PuntoMedio = 110; const int PuntoMinimoDerecha = 40; const int PuntoMaximoIzquierda = 180; //uint16\_t m\_signature; int velocidad = 30; //Esta velocidad es la que le ponemos al LEGO EV3, por lo que a más velocidad deberá necesitar menos tiempo para esquivar los obstaculos. int tiempoDeGiro = -62.5 \* velocidad + 4875; const int TriggerFrontPin = 8; const int EchoFrontPin = 9; int durationFront; int distanceFrontCm; const int TriggerDchaPin = 3; const int EchoDchaPin = 4; int durationDcha; int distanceDchaCm; const int distDchaSetPoint = 20; void setup() { Serial.begin(115200); Serial.print("Starting...\n"); pixy.init(); direccion.attach(2); direccion.write(PuntoMedio); Serial.println("PosicionPuntoMedio"); delay(2000); direccion.write(PuntoMinimoDerecha); Serial.println("PosicionMinimoDerecha"); delay(2000); direccion.write(PuntoMaximoIzquierda); Serial.println("PosicionMaximoIzquierda"); delay(2000); direccion.write(PuntoMedio); Serial.println("PosicionPuntoMedio"); pinMode(TriggerFrontPin, OUTPUT); pinMode(EchoFrontPin, INPUT); pinMode(TriggerDchaPin, OUTPUT); pinMode(EchoDchaPin, INPUT); } void loop() { direccion.write(PuntoMedio); Serial.println("Recto"); delay(1500); digitalWrite(TriggerDchaPin,LOW); delayMicroseconds(2); digitalWrite(TriggerDchaPin,HIGH); delayMicroseconds(5);//Entre 5 y 10ms dependiendo del sensor digitalWrite(TriggerDchaPin,LOW); int durationDcha=pulseIn(EchoDchaPin, HIGH); int distanceDchaCm=0.017\*durationDcha; Serial.print(durationDcha); Serial.print(" microseconds, "); Serial.print(distanceDchaCm); Serial.println(" cm por la derecha"); delay(100); if (distanceDchaCm >= distDchaSetPoint) { direccion.write(PuntoMinimoDerecha); Serial.println("Derecha"); delay(3000);//tiempoDeGiro } else { direccion.write(PuntoMaximoIzquierda); Serial.println("Izquierda"); delay(3000);//tiempoDeGiro } Slalom(); } void mideDistFront() { digitalWrite(TriggerFrontPin, LOW); delayMicroseconds(2); digitalWrite(TriggerFrontPin, HIGH); delayMicroseconds(5);//Entre 5 y 10ms dependiendo del sensor digitalWrite(TriggerFrontPin, LOW); int durationFront = pulseIn(EchoFrontPin, HIGH); int distanceFrontCm = 0.017 \* durationFront; Serial.print(durationFront); Serial.print(" microseconds, "); Serial.print(distanceFrontCm); Serial.println("cm de frente"); delay(100); } void mideDistDcha() { digitalWrite(TriggerDchaPin, LOW); delayMicroseconds(2); digitalWrite(TriggerDchaPin, HIGH); delayMicroseconds(5);//Entre 5 y 10ms dependiendo del sensor digitalWrite(TriggerDchaPin, LOW); int durationDcha = pulseIn(EchoDchaPin, HIGH); int distanceDchaCm = 0.017 \* durationDcha; Serial.print(durationDcha); Serial.print(" microseconds, "); Serial.print(distanceDchaCm); Serial.println("cm por la derecha"); delay(100); } void Slalom() { int i; // grab blocks! pixy.ccc.getBlocks(); // If there are detect blocks, print them! if (pixy.ccc.numBlocks) { Serial.print("Detected "); Serial.println(pixy.ccc.numBlocks); for (i = 0; i < pixy.ccc.numBlocks; i++) { Serial.print("La Signatura es = "); Serial.println(pixy.ccc.blocks[i].m\_signature); if (pixy.ccc.blocks[i].m\_signature == 1) { direccion.write(PuntoMinimoDerecha); delay(tiempoDeGiro); } else if (pixy.ccc.blocks[i].m\_signature == 3) { direccion.write(PuntoMaximoIzquierda); delay(tiempoDeGiro); } delay(250); } } }

1. **Comunicación con Pixy2**: Asegurar de que la comunicación con la cámara Pixy2 esté configurada correctamente. Debes haber conectado la cámara Pixy2 al puerto adecuado de Arduino y haber instalado la biblioteca correspondiente. Verifica que la cámara esté configurada para detectar los objetos rojos y verdes de acuerdo con tus necesidades.
2. **Medición de distancia ultrasónica**: La función **mideDistDcha()** está duplicada y no se llama en el bucle principal (**loop()**). Si planeas utilizar la distancia medida por el sensor ultrasónico en tu lógica de toma de decisiones, asegúrate de que esta función se llame en el bucle principal.
3. **Lógica de toma de decisiones**: El código actual parece mover el servo en función de lo que detecta la cámara Pixy2. Si detecta un objeto con una firma de 1, gira a la derecha; si detecta un objeto con una firma de 3, gira a la izquierda. Asegúrate de que esta lógica sea coherente con tus objetivos, es decir, gira hacia el lado correcto según el color detectado (rojo o verde).
4. **Tiempo de giro**: El tiempo de giro se calcula en función de la velocidad del motor. Asegúrate de que este cálculo sea adecuado para tu configuración específica y que el robot gire el ángulo deseado.
5. **Tiempos de espera**: Los tiempos de espera (**delay()**) se utilizan para controlar el tiempo que el robot gira. Asegúrate de que estos tiempos sean los adecuados para tu aplicación y que el robot tenga suficiente tiempo para realizar las acciones necesarias.
6. **Otras consideraciones de hardware**: Asegúrate de que los pines utilizados para la comunicación con la cámara Pixy2 y el sensor ultrasónico sean los correctos según tu configuración.
7. **Manejo de errores**: Considera agregar manejo de errores o excepciones para manejar situaciones inesperadas, como la pérdida de comunicación con la cámara Pixy2 o problemas con el sensor ultrasónico.
8. **Comentarios y documentación**: Puede ser útil agregar comentarios al código para explicar la lógica detrás de las decisiones y las funciones utilizadas, lo que facilitará la depuración y el mantenimiento en el futuro.

Principio del formulario

**4.- Fotos y Diseño Vehículo:**

Un circuito electrónico

Descripción generada automáticamente con confianza media

Imagen que contiene interior, tabla, camioneta, pequeño

Descripción generada automáticamente

Imagen que contiene interior, pequeño, camioneta, artículos

Descripción generada automáticamente

Imagen que contiene podadora, camioneta, hombre, pequeño

Descripción generada automáticamente