Universidad Modelo

Ingeniería Mecatrónica

Alumno: Rodrigo Gómez Paredes

Séptimo semestre

Asignatura: Visión por computadora.

Reporte proyecto AGV.

Fecha de entrega: 11 de noviembre de 2025

Introducción

El presente proyecto tiene como propósito el desarrollo de un vehículo guiado autónomo (AGV, por sus siglas en inglés) capaz de desplazarse de manera autónoma siguiendo una ruta predefinida mediante sensores de línea, e integrar además un sistema de visión por computadora para la detección de marcadores ArUco.

El objetivo principal de este trabajo es combinar principios de mecatrónica, control y visión artificial, aplicando herramientas de diseño CAD, programación embebida y procesamiento de imágenes en un mismo sistema funcional.

El proyecto se concibe como una plataforma de prueba para futuras implementaciones de navegación inteligente, localización visual y comunicación entre sistemas. En una primera etapa, se enfocó en el diseño y construcción del robot AGV, asegurando su movilidad y estabilidad. Posteriormente, se desarrolló el código de control para permitir el seguimiento de trayectorias mediante sensores infrarrojos, logrando un desplazamiento autónomo.

Finalmente, se integró un módulo de detección visual basado en la librería OpenCV de Python, encargado de identificar marcadores ArUco en el entorno. Esto permite que el AGV pueda reconocer puntos de referencia o zonas específicas, ampliando sus capacidades hacia la navegación guiada por visión, un paso importante dentro del campo de la robótica móvil inteligente.

Metodología:

1. Diseño CAD

Se elaboró un modelo tridimensional del robot AGV utilizando el software Autodesk Fusion 360. Este diseño permitió definir las dimensiones generales del vehículo, así como la disposición óptima de cada componente. Durante esta etapa se consideraron factores como la estabilidad del robot, el peso de los elementos, el centro de gravedad y el espacio necesario para integrar los motores, sensores, controladores y baterías. El diseño CAD sirvió también como base para la fabricación de las piezas estructurales y para anticipar posibles interferencias entre componentes.

2. Ensamble

Con el modelo CAD como referencia, se procedió al ensamblaje físico del robot AGV. Se montaron los motores de tracción, ruedas, chasis, baterías y la placa controladora, asegurando conexiones mecánicas firmes y una adecuada distribución del peso. En esta fase se realizaron pruebas iniciales de alimentación eléctrica y verificación de polaridad en los motores, además de comprobar la correcta instalación de los sensores y el sistema de control principal.

3. Programación

Una vez completado el ensamble, se desarrolló el código de control encargado de gestionar el movimiento del robot. Este programa permitió al AGV seguir una trayectoria marcada mediante el uso de sensores infrarrojos que detectan una línea guía en el suelo. El código fue implementado en el microcontrolador principal, integrando rutinas de lectura de sensores, control de motores, y toma de decisiones para la navegación autónoma. También se añadieron funciones de seguridad y control básico de velocidad.

4. Pruebas y calibraciones

Con el sistema programado, se inició la fase de pruebas y calibraciones. Durante esta etapa se ajustaron los umbrales de detección de los sensores infrarrojos, las ganancias de control y los parámetros de velocidad del AGV, con el fin de lograr un desplazamiento estable y preciso sobre la trayectoria. Asimismo, se realizaron pruebas repetitivas en diferentes condiciones de iluminación y superficie para asegurar un comportamiento robusto del sistema de navegación.

5. Detector de ArUcos

Finalmente, una vez que el AGV fue capaz de desplazarse de manera autónoma, se integró un sistema de visión por computadora mediante un código en Python que utiliza la librería OpenCV para la detección de marcadores ArUco. Este módulo permite al robot identificar marcadores visuales y, potencialmente, reconocer posiciones o zonas específicas dentro de su entorno, ampliando sus capacidades de localización y navegación.

Códigos:

Código para seguir las líneas en el piso:

```
int val1=700;
int val2=700;
int val3=700;
int val4=700;
int val5=700;

int M1A = 5; //MOTOR A 5
int M2A = 6; //        6

int M1B = 9; //MOTOR B  9
int M2B = 10; //        10

int s1 = A7; //SENSORES DE PISO
int s2 = A6;
int s3 = A5;
int s4 = A4;
int s5 = A3;

int vs1 = 0; //VARIABLES
int vs2 = 0;
int vs3 = 0;
int vs4 = 0;
int vs5 = 0;
```

```
void setup() {
  //Declarar pines de comunicación
  pinMode (11, OUTPUT);
  pinMode (12, OUTPUT);
  pinMode (13, OUTPUT);
  pinMode (14, OUTPUT);
  pinMode (15, OUTPUT);
  //Declarar pines de motores
  pinMode (M1A, OUTPUT);
  pinMode (M2A, OUTPUT);
  pinMode (M1B, OUTPUT);
  pinMode (M2B, OUTPUT);
  //Declarar pines de sensores
  pinMode (s1, INPUT);
  pinMode (s2, INPUT);
  pinMode (s3, INPUT);
  pinMode (s4, INPUT);
  pinMode (s5, INPUT);
  Serial.begin (9600);
}

void loop() {
  vs1 = analogRead(s1);
  vs2 = analogRead(s2);
  vs3 = analogRead(s3);
  vs4 = analogRead(s4);
  vs5 = analogRead(s5);
```

```
  if (vs1 >= val1 && vs2 >= val2 && vs3 <= val3 && vs4 >=val4 && vs5 >= val5){
   // Serial.println( "adelante");
    adelante();
    digitalWrite(11,LOW);
    digitalWrite(12,LOW);
    digitalWrite(13,HIGH);
    digitalWrite(14,LOW);
    digitalWrite(15,LOW);
  }

  if (vs1 >= val1 && vs2 <= val2 && vs3 >= val3 && vs4 >=val4 && vs5 >= val5){
    // Serial.println( "izquierda");
    izquierda();
    digitalWrite(11,LOW);
    digitalWrite(12,LOW);
    digitalWrite(13,LOW);
    digitalWrite(14,HIGH);
    digitalWrite(15,LOW);
  }

  if (vs1 >= val1 && vs2 >= val2 && vs3 >= val3 && vs4 <=val4 && vs5 >= val5){
    //Serial.println( "derecha");
    derecha();
    digitalWrite(11,LOW);
    digitalWrite(12,HIGH);
    digitalWrite(13,LOW);
    digitalWrite(14,LOW);
    digitalWrite(15,LOW);
  }

  if (vs1 <= val1 && vs2 >= val2 && vs3 >= val3 && vs4 >= val4 && vs5 >= val5){
   // Serial.println( "izquierda_2");
    izquierda_2();
    digitalWrite(11,LOW);
    digitalWrite(12,LOW);
    digitalWrite(13,LOW);
    digitalWrite(14,LOW);
    digitalWrite(15,HIGH);
  }
```

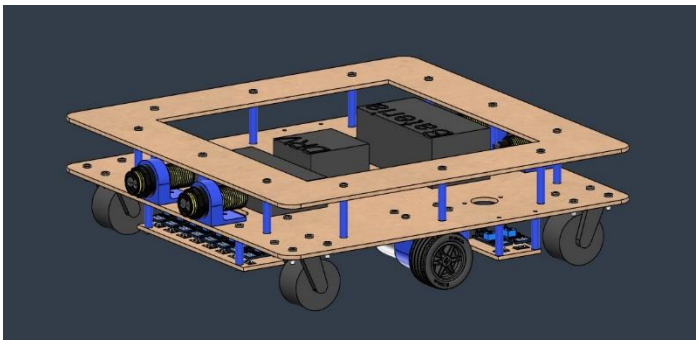
```
  if (vs1 >= val1 && vs2 >= val2 && vs3 >= val3 && vs4 >= val4 && vs5 <= val5){
   // Serial.println( "derecha_2");
    derecha_2();
    digitalWrite(11,HIGH);
    digitalWrite(12,LOW);
    digitalWrite(13,LOW);
    digitalWrite(14,LOW);
    digitalWrite(15,LOW);
  }

  if ((vs1 <= 500 && vs2 <= 500) && vs3 <=500 && (vs4 <=500 || vs5 <= 500)){
    //parar();
    digitalWrite(11,HIGH);
    digitalWrite(12,HIGH);
    digitalWrite(13,HIGH);
    digitalWrite(14,HIGH);
    digitalWrite(15,HIGH);
 }
}
```

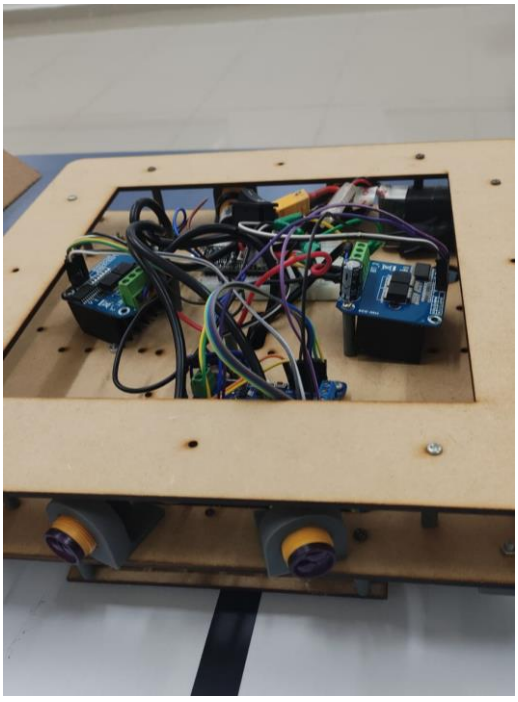
```
112  void adelante() {
113    analogWrite (M1A, LOW);
114    digitalWrite (M2A, 90);
115    analogWrite (M1B, 90);
116    digitalWrite (M2B,LOW );
117  }
118  void parar() {
119    digitalWrite (M1A, LOW);
120    digitalWrite (M2A, LOW);
121    digitalWrite (M1B, LOW);
122    digitalWrite (M2B, LOW);
123  }
124  void izquierda() {
125    analogWrite (M1A, LOW);
126    digitalWrite (M2A, 90);
127    analogWrite (M1B, 70);
128    digitalWrite (M2B, LOW);
129  }
130
131  void izquierda_2() {
132    analogWrite (M1A, LOW);
133    digitalWrite (M2A, 90);
134    analogWrite (M1B, LOW);
135    digitalWrite (M2B, LOW);
136  }
137
138  void derecha() {
139    digitalWrite (M1A, LOW);
140    digitalWrite (M2A, 70);
141    digitalWrite (M1B, 90);
142    digitalWrite (M2B, LOW);
143  }
144
145  void derecha_2() {
146    digitalWrite (M1A, LOW);
147    digitalWrite (M2A, LOW);
148    digitalWrite (M1B, 90);
149    digitalWrite (M2B, LOW);
150  }
```

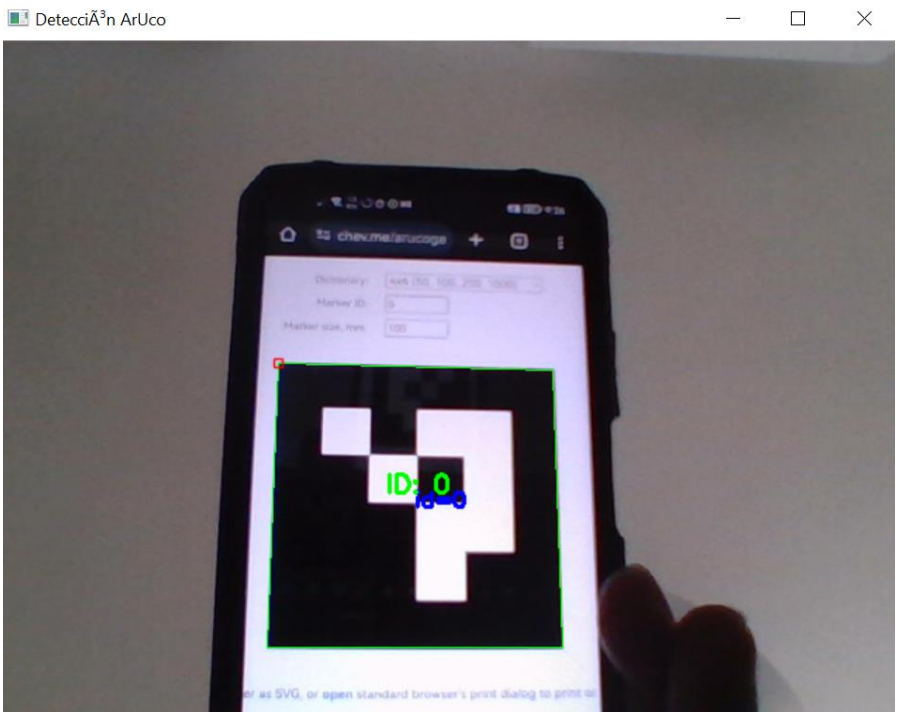
Código para detectar ArUcos

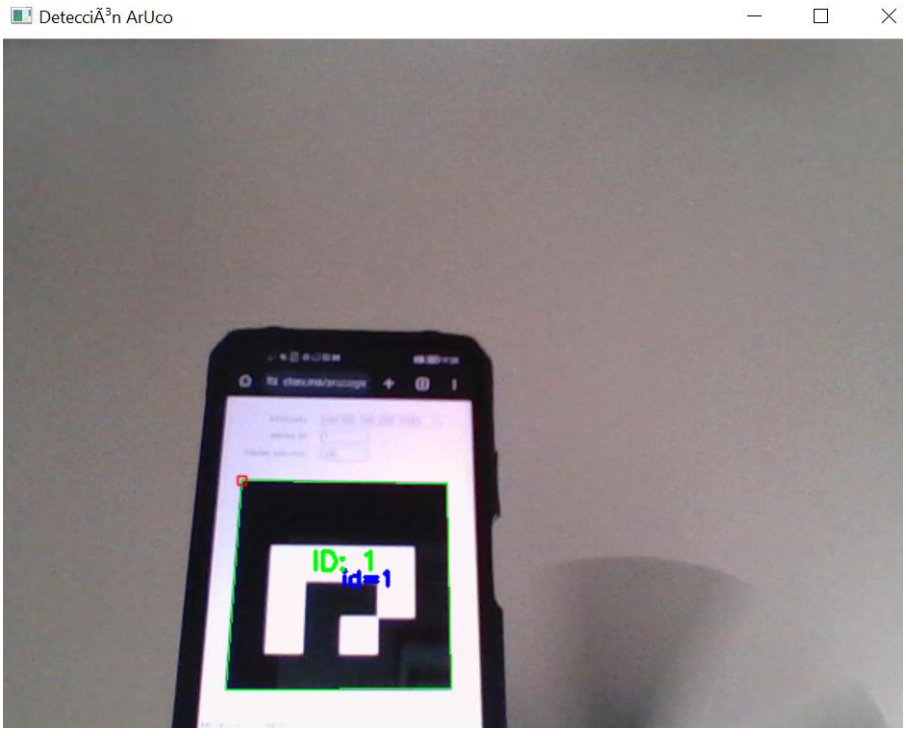
```python
#Codigo de prueba para detectar ArUco
import cv2
# --- Inicializar la cámara ---
cap = cv2.VideoCapture(0)  # Cambia el número si tienes más de una cámara

# --- Cargar el diccionario de marcadores ArUco ---
aruco_dict = cv2.aruco.getPredefinedDictionary(cv2.aruco.DICT_4X4_50)
parameters = cv2.aruco.DetectorParameters()

# --- Crear el detector ---
detector = cv2.aruco.ArucoDetector(aruco_dict, parameters)

print("Presiona 'q' para salir")

while True:
    ret, frame = cap.read()
    if not ret:
        break

    # --- Detectar marcadores ---
    corners, ids, rejected = detector.detectMarkers(frame)

    # --- Si se detectan ---
    if ids is not None:
        # Dibujar los marcadores y sus IDs
        cv2.aruco.drawDetectedMarkers(frame, corners, ids)
        for i in range(len(ids)):
            c = corners[i][0]
            x, y = int(c[:, 0].mean()), int(c[:, 1].mean())

            # Mostrar ID en la imagen
            cv2.putText(frame, f"ID: {ids[i][0]}", (x - 20, y - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0), 2)

            # --- Imprimir el ID en consola ---
            print(f"Marcador detectado -> ID: {ids[i][0]}")

    else:
        cv2.putText(frame, "Sin marcadores detectados", (20, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

    # --- Mostrar imagen ---
    cv2.imshow("Detección ArUco", frame)

    # --- Salir con 'q' ---
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Resultados:






DetecciÃ³n ArUco
ID: 0
id=0


DetecciÃ³n ArUco
ID: 1
id=1

Conclusiones:

El desarrollo del presente proyecto permitió integrar distintas áreas de la ingeniería mecatrónica, como el diseño mecánico, la electrónica de control y la visión por computadora, en un sistema funcional de navegación autónoma. A través del diseño CAD se logró planificar la estructura del robot AGV de manera eficiente, asegurando la correcta distribución de los componentes y un ensamblaje estable.

La etapa de programación y calibración resultó fundamental para conseguir que el robot siguiera correctamente la trayectoria marcada por los sensores infrarrojos, demostrando la importancia de los ajustes en los parámetros de control y lectura de sensores para garantizar un movimiento preciso y estable.

Por otro lado, la implementación del detector de marcadores ArUco en Python mediante la librería OpenCV aportó una nueva dimensión al proyecto, al incorporar la capacidad de reconocimiento visual. Este módulo permite al AGV identificar referencias en su entorno, lo que sienta las bases para futuras mejoras orientadas a la navegación visual, posicionamiento y toma de decisiones autónomas.

En conclusión, el proyecto cumplió con los objetivos planteados al lograr la construcción y funcionamiento de un AGV capaz de desplazarse de forma autónoma y detectar marcadores visuales. Asimismo, representa una plataforma sólida para el desarrollo de aplicaciones más avanzadas de robótica móvil e inteligencia artificial aplicada a la visión computacional.