



**UNIVERSIDAD ESAN**  
**FACULTAD DE INGENIERÍA**

# **Manual Técnico de TomatAIT**

Robot estacionario con cinta transportadora para detección de madurez de tomates

**Curso:** Robótica

**Docente:** Calderón Niquin, Marks

## **Integrantes:**

Andrade Ortiz Alessandro

Condor García Kheyla

Mantilla Huaman Joe

**LIMA-PERÚ**

**2025**

## ÍNDICE

<b>1. Lista de Componentes:</b>	<b>3</b>
1.1. Componentes en de Cinta Transportadora:	3
1.2. Componentes de Módulo de Detección:	5
1.3. Componentes Electrónicos:	6
<b>2. Construcción de Robot Estacionario:</b>	<b>9</b>
2.1. Ensamblaje de Cinta Transportadora:	9
2.2. Ensamblaje de Módulo de Detección:	10
2.3. Ensamblado de componentes electrónicos:	11
2.4. Ensamblado Final:	13
• Fijación del Motor a la Cinta Transportadora:	13
• Montaje del Módulo de Detección sobre la Cinta:	13
• Posicionamiento del Sensor Ultrasónico (Detección de Entrada):	14
• Integración del Mecanismo de Actuación:	14
<b>3. Instalación del Software:</b>	<b>15</b>
3.1. Código Arduino MEGA:	15
3.2. Código en Raspberry Pi Pico W:	21
3.3. Instalación del software en la Raspberry Pi 5:	24
<b>4. Uso básico de TomatAIT:</b>	<b>28</b>

## 1. Lista de Componentes:

### 1.1. Componentes en de Cinta Transportadora:



Figura 1: Vista en Corel Draw de Carril principal

La Figura 1 muestra los componentes estructurales más largos, diseñados para ser cortados con láser.

- **Rieles Laterales (Dos piezas largas con orificios):** Estas piezas conforman los **laterales longitudinales** del marco de la cinta transportadora. Su función es establecer la **longitud total** de la cinta y servir como soporte principal para los ejes de tracción y los rodamientos en sus extremos. Son la base de apoyo estructural.
- **Plataforma de Base (Una pieza rectangular grande):** Este rectángulo se utiliza como la **plataforma interna** que se coloca debajo de la banda azul, proporcionando una superficie plana y firme sobre la cual el material transportado (como el tomate) se deslizará, evitando que la banda se hunda.

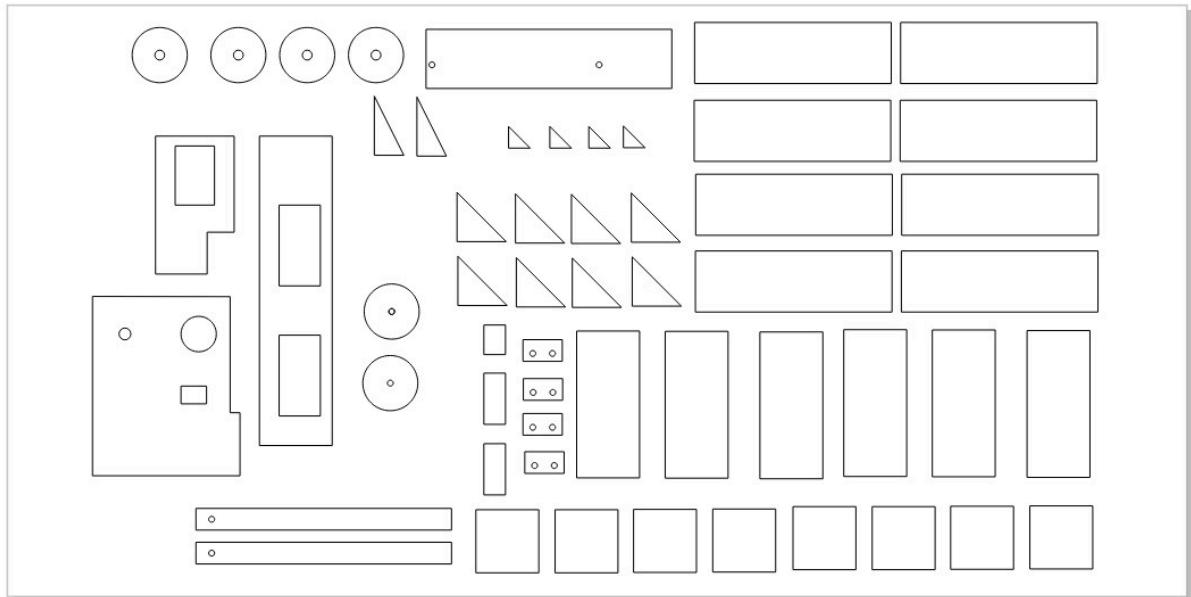


Figura 2: Vista en Corel Draw de Soportes

La Figura 2 muestra una colección de piezas auxiliares y soportes optimizados para el corte (nesting). Estas piezas aportan el soporte vertical y los puntos de montaje.

- **Soportes Verticales y Patas:** Se observan piezas rectangulares de diferentes tamaños que actúan como **patas o soportes verticales** para elevar la estructura del carril del suelo (visible en la Figura 3).
- **Soportes para Ejes y Rodamientos (Piezas circulares y cuadradas pequeñas):** Los círculos y pequeños rectángulos perforados están diseñados para fijar y asegurar los **rodamientos** y los **ejes** dentro de los rieles laterales de la Figura 1, garantizando que giren libremente.
- **Refuerzos y Piezas de Ensamblaje (Triángulos y piezas variadas):** Las pequeñas piezas triangulares y rectangulares más cortas se utilizan para dar **rigidez transversal** a la estructura y funcionar como **uniones encajables** (\textit{finger joints}) que conectan los rieles laterales con los soportes verticales.

### 1.2. Componentes de Módulo de Detección:

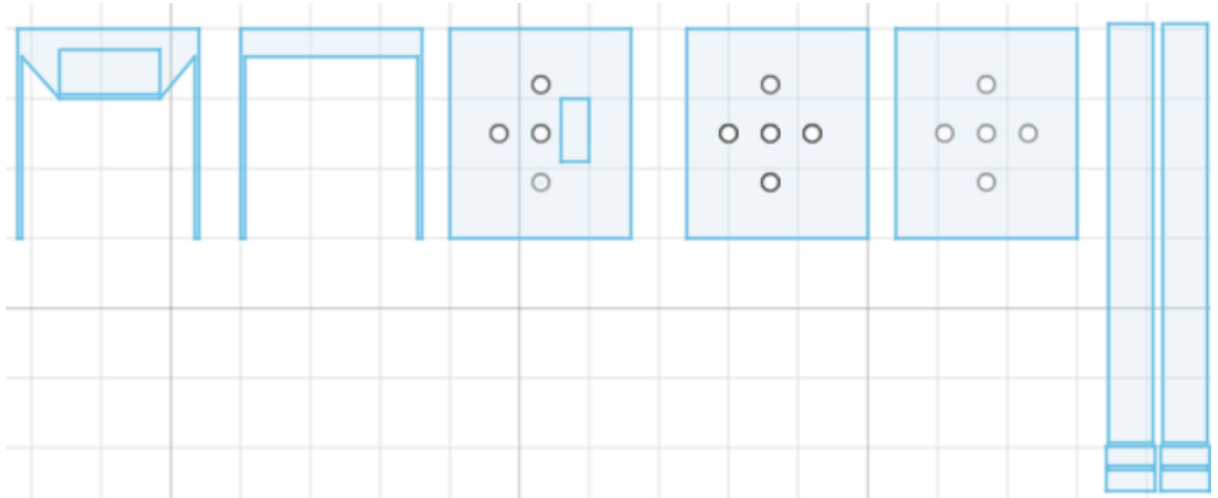


Figura 3: Vista en Fusion 360

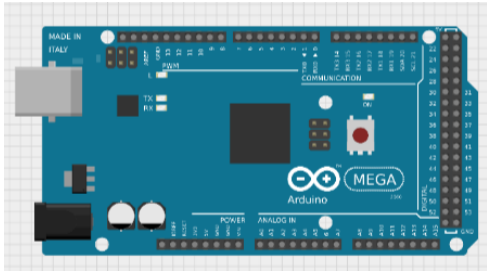
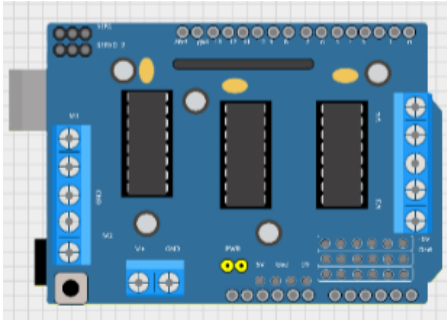

La Figura 3 muestra la vista de diseño de las piezas estructurales del Módulo de Detección, las cuales serán ensambladas para formar la caja que alberga la electrónica.



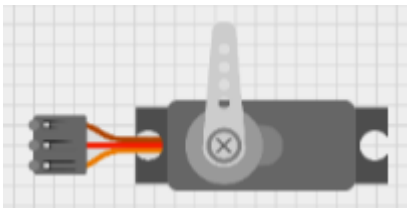

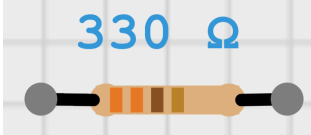
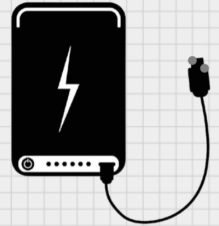
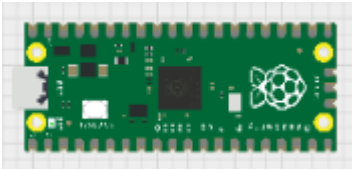
Los componentes principales que se utilizarán en el ensamblaje son:

- **Paredes Laterales (Las piezas más largas a la izquierda):** Son las paredes que definen la **altura y profundidad** del módulo. Estas piezas tienen grandes cortes rectangulares en la parte superior, que actúan como marcos o huecos para montar la placa superior de control y proporcionar acceso a la electrónica interior. En el ensamblaje (Figura 5), estas se convierten en los lados izquierdo y derecho del módulo central.
- **Soportes Inferiores/Patas (Dos piezas planas con base curva):** Estas son las "patas" del módulo, que se extienden hacia adelante. Su función principal es **dar estabilidad** y elevar el módulo para que pueda colocarse **sobre la cinta transportadora**.
- **Placas de Montaje Interno (Las piezas cuadradas con círculos):** Estas son cruciales para el montaje. Contienen orificios precisos para fijar componentes clave:

- **Orificio Central Grande:** Diseñado para la **lente de la cámara** o el sensor de visión que mirará hacia abajo, sobre la cinta.
- **Orificios Periféricos:** Utilizados para montar **sensores de distancia** (como el ultrasónico) o para atornillar la estructura interna de la electrónica (Arduino, motor driver).
- **Paneles Frontal/Trasero (Piezas delgadas a la derecha):** Se utilizan para cerrar la parte superior del módulo o actuar como tapas, algunas con ranuras para cables o interruptores.

### 1.3. Componentes Electrónicos

Imagen	Descripción
<p style="text-align: center;">Arduino Mega</p> 	<p>Placa de control principal que ejecuta el código, procesa la señal de los sensores y emite las señales de control.</p>
<p style="text-align: center;">Motor Driver (L293d)</p> 	<p>Chip/Módulo controlador de motor (Puentes H L293D). Recibe señales de bajo voltaje del Arduino y suministra la potencia necesaria (corriente) a los motores DC.</p>
<p style="text-align: center;">Motor DC</p> 	<p>Motorreductor DC con caja de engranajes, encargado de suministrar el torque y movimiento a la cinta transportadora.</p>
<p style="text-align: center;">Sensor Ultrasonico</p>	<p>Sensor Ultrasónico (HC-SR04), utilizado para</p>

	<p>medir la distancia del objeto (tomate) y determinar si se encuentra en la zona de detección.</p>
<p>Interruptor</p> 	<p>Interruptor de encendido/apagado general que activa y desactiva el funcionamiento completo del sistema y del motor.</p>
<p>Servomotor</p> 	<p>Actuador de control angular, utilizado para el mecanismo de clasificación o desvío de los objetos sobre la cinta.</p>
<p>LEDs (13 uds)</p> 	<p>Diodos emisores de luz (13 unidades) utilizados para optimizar la iluminación.</p>
<p>Resistencias (13 uds)</p> 	<p>Resistencias de 330 Ohms (13 unidades) utilizadas para limitar la corriente y proteger tanto a los LEDs como a los pines de salida del Arduino.</p>
<p>Fuente de Energía</p> 	<p>Fuente de alimentación externa de <b>12 Voltios</b>, requerida para suministrar la alta potencia (voltaje y corriente) necesaria para el Motor Driver (L293D) y el motor DC.</p>
<p>Raspberry Pi Pico</p> 	<p>Su función es actuar como el <b>punto de comunicación inter-placas</b> para el envío de los resultados del modelo YOLOv8 (desde la RPi 5) hacia el Arduino (para la actuación del clasificador).</p>
<p>Raspberry Pi 5</p>	<p>Realiza el <b>procesamiento de la cámara</b> (captura y gestión de imágenes) y la ejecución del modelo de <b>visión YOLOv8</b> para la</p>

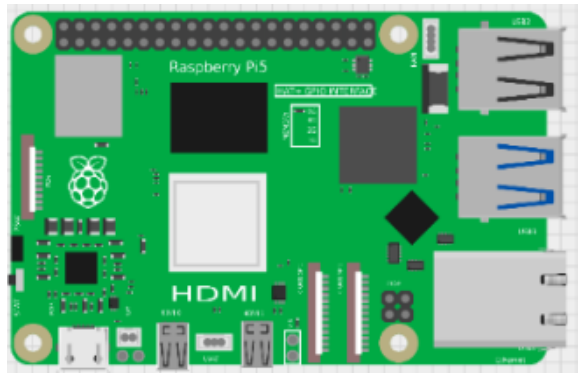

	<p>detección de madurez.</p>
<p>Argom Tech USB HD Webcam</p> 	<p>Captura continuamente el flujo de video/imágenes sobre la cinta transportadora. El procesamiento de estas imágenes es cargado a la <b>Raspberry Pi 5</b> para ser analizado por el modelo YOLOv8.</p>

Tabla 1: Componentes electrónicos principales de TomatAIT

## 2. Construcción de Robot Estacionario

### 2.1. Ensamblaje de Cinta Transportadora

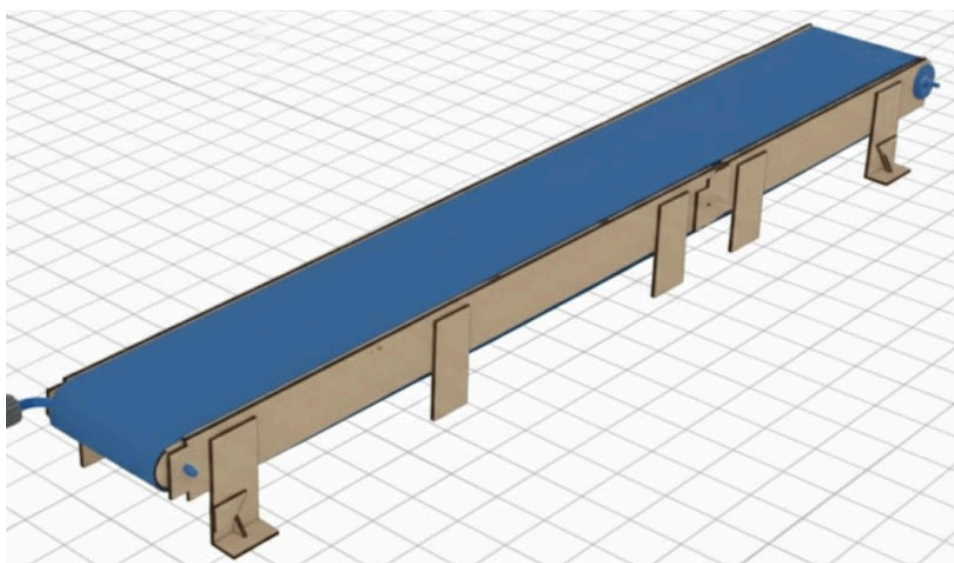




Figura 4: Cinta Transportadora ensamblada

El objetivo de ensamblaje es crear la estructura robusta y funcional que se muestra en la Figura 3, utilizando uniones machihembradas o encajables sin necesidad de adhesivos excesivos.

- **Montaje del Marco Longitudinal:** Se toman los dos **Rieles Laterales** (Figura 1) y se insertan en las ranuras de los **Soportes Verticales y Patas** más grandes (Figura 2). Este paso define la altura y la longitud de la cinta.
- **Fijación de la Plataforma de Base:** La **Plataforma de Base** (el rectángulo grande de la Figura 1) se coloca internamente, encajando en las ranuras inferiores de los rieles laterales, asegurando que la banda de la cinta tenga un apoyo firme en el centro.
- **Instalación de Ejes y Rodamientos:** En los extremos de los rieles laterales, se utilizan los **Soportes para Ejes y Rodamientos** (Figura 2) para fijar el eje de tracción (donde se conectará el motor) y el eje pasivo, que mantendrán la tensión de la banda.
- **Colocación de la Banda:** Una vez que la estructura es rígida, la **banda de la cinta** (el material azul) se coloca y se tensa alrededor del eje de tracción y el eje pasivo.

El resultado final es la **Cinta Transportadora ensamblada** (Figura 4), una unidad mecánica completa lista para ser acoplada al motor y a la estructura de la caja de control/detección.

## 2.2. Ensamblaje de Módulo de Detección

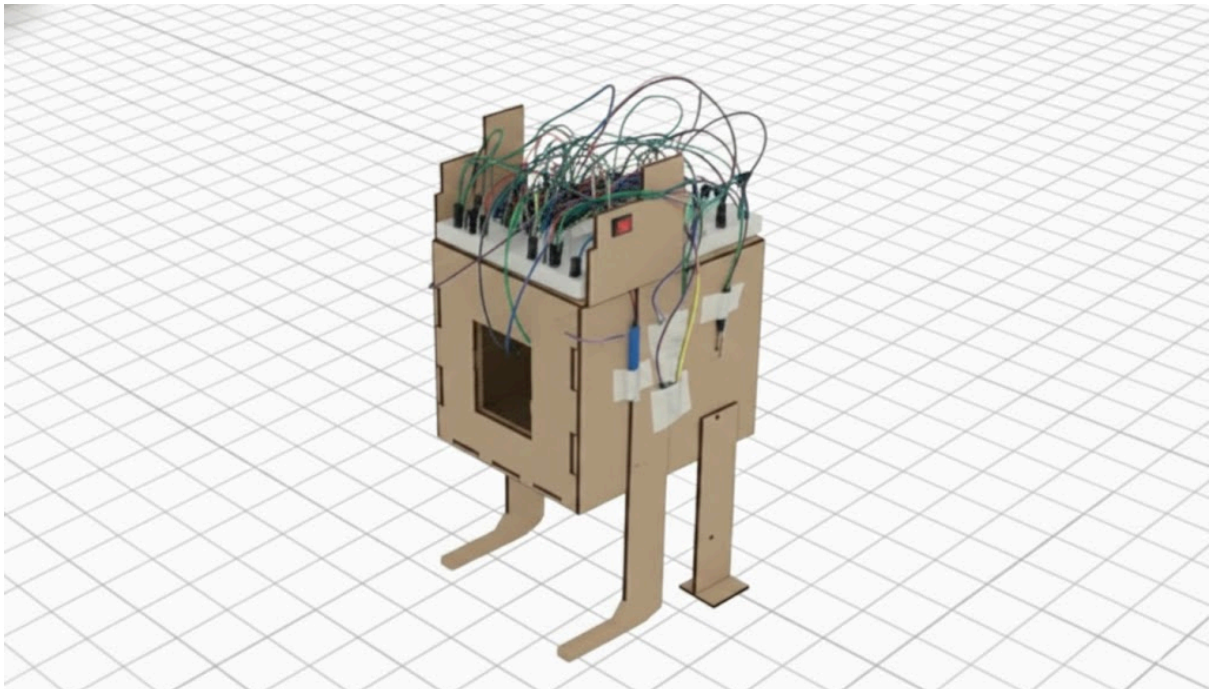


Figura 5: Módulo de Detección Ensamblado (referencial)

El ensamblaje de la Figura 5 tiene como objetivo construir una estructura de caja robusta y elevada a partir de las piezas planas cortadas con láser de la Figura 3.

- **Ensamblaje del Cuerpo Central (Caja):** Las **Paredes Laterales** (piezas largas) se unen perpendicularmente con el **Panel Frontal** y las **Placas de Montaje Interno** mediante uniones encajables. Esto forma la caja cuadrada que define el volumen interno para la electrónica.
- **Montaje de Soportes y Estabilización:** Los **Soportes Inferiores/Patas** se encajan en la parte inferior de la caja central. Estos soportes aseguran que la caja se mantenga estable y permiten que el módulo se "monte a horcajadas" sobre los rieles laterales de la cinta transportadora.
- **Integración de Sensores:** Una de las **Placas de Montaje Interno** se fija en el plano inferior del módulo, con el orificio central alineado sobre la cinta para permitir que la cámara o el sensor de visión enfoque el objeto transportado.
- **Cierre Superior:** La placa superior se fija en la parte superior, proporcionando una superficie para el **montaje y cableado** del circuito de control (visible en la Figura 5, con cables saliendo de la parte superior).

El resultado final (Figura 5) es un soporte vertical autónomo de madera (MDF) con una cavidad interna, listo para recibir los componentes electrónicos (Raspberry Pi, Arduino, drivers y sensores) y colocarse sobre la Cinta Transportadora.

### 2.3. *Ensamblado de componentes electrónicos*

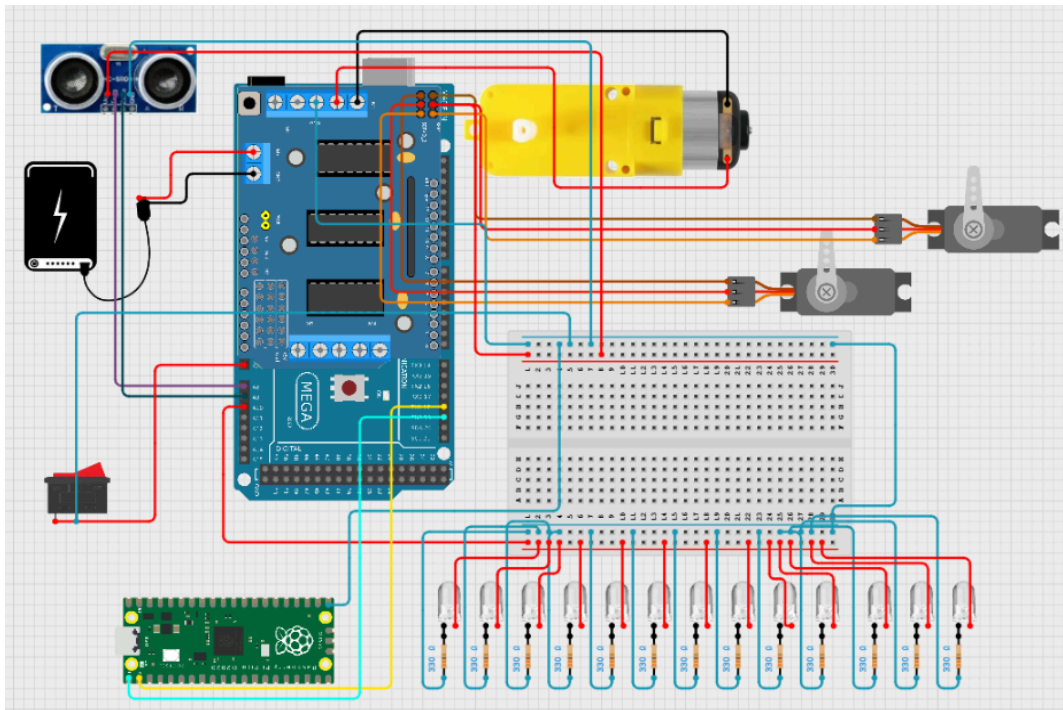


Figura 6: Conexiones de Componentes Electrónicos

El ensamblaje electrónico se centra en la integración del control de potencia (Driver/Motor) con la lógica de control (Arduino Mega) y el procesamiento de IA (Raspberry Pi 5).

- Integración del Módulo de Potencia y Control

Driver L293D (o similar): El Driver se monta sobre el Arduino Mega (o se conecta vía cables) y se alimenta de la Fuente de Energía Externa de 12V. Este driver gestiona la potencia del Motor DC (motorreductor de la cinta) y los Servomotores (actuadores de clasificación).

Motor DC y Servomotores: El motor de la cinta se conecta a las salidas del Driver L293D, mientras que los Servomotores se conectan directamente a los pines PWM del Arduino Mega o del shield (ej., pines 9 y 10), ya que el L293D no los controla.

- Integración de la Raspberry Pi 5 y Visión

Raspberry Pi 5 (RPi 5): Esta placa se instala para gestionar el procesamiento de la cámara. Recibe la señal de la Webcam USB (Argom Tech), donde se carga y ejecuta el modelo YOLOv8 para clasificar la madurez del tomate.

Comunicación Interna (MicroPython): La RPi 5, una vez que obtiene el resultado del modelo, utiliza un canal de comunicación (posiblemente a través de un módulo MicroPython / Pico o serial) para enviar la orden de acción (ej., "Maduro", "Verde", "Mohoso") al Arduino Mega.

- Sensores e Indicadores

Sensores: El Sensor de Distancia Ultrasónico y el Interruptor se cablean directamente a las entradas analógicas o digitales del Arduino Mega.

LEDs e Iluminación: Los 13 Diodos emisores de luz se conectan a los pines de salida del Mega (a través de sus Resistencias de  $330\Omega$  para limitar la corriente). Estos LEDs se utilizan para optimizar la iluminación del área de detección o como indicadores de estado del sistema.

#### 2.4. Ensamblado Final

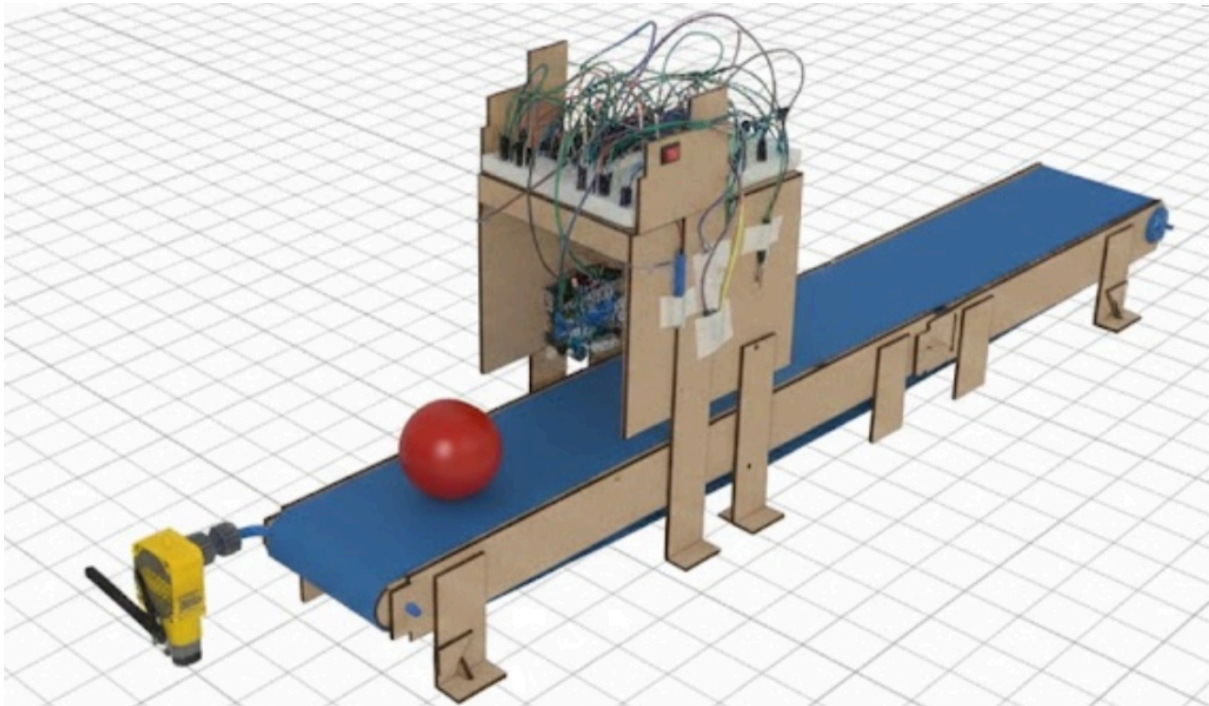


Figura 7: Vista general de ensamblado final (referencia)

Esta fase describe cómo las subestructuras de madera y los módulos electrónicos se unen para formar el prototipo final.

- **Fijación del Motor a la Cinta Transportadora**
  - El **Motor DC con caja reductora** se fija a un soporte de madera independiente, diseñado para alinear su eje con el **eje de tracción** de la **cinta transportadora** (Figura 3).
  - La unión entre el eje del motor y el eje de la cinta se realiza mediante un **acople flexible** (coupling) para compensar ligeras desalineaciones y garantizar una transmisión de movimiento suave y constante.
- **Montaje del Módulo de Detección sobre la Cinta**

- El **Módulo de Detección ensamblado** (Figura 5), que ya contiene la electrónica interna (Arduino Mega, RPi 5 y el cableado de la tapa superior), se coloca directamente sobre la estructura de la cinta transportadora.
- Las **patas curvas** del Módulo de Detección (Figura 5) están diseñadas para encajar y dar estabilidad a horcajadas sobre los **rieles laterales** de la Cinta Transportadora (Figura 3). Esta ubicación central permite que el sensor de visión y el mecanismo de actuación (servomotores) queden perfectamente **alineados sobre la banda azul**.
- **Posicionamiento del Sensor Ultrasónico (Detección de Entrada)**
  - El **Sensor de Distancia Ultrasónico** se fija en el **extremo de entrada** de la Cinta Transportadora.
  - Su función es **detectar la llegada** de un objeto (como el tomate) y emitir la señal al Arduino Mega para activar el proceso de lectura de la cámara y el motor de la cinta, iniciando el ciclo de clasificación.
- **Integración del Mecanismo de Actuación**
  - Los **Servomotores**, si son parte del mecanismo de desvío, se montan en el exterior del Módulo de Detección o justo después de este, en un punto donde puedan empujar o desviar el objeto según la clasificación de la Raspberry Pi 5.
  - El **Interruptor (Switch)** se instala en el orificio reservado de la placa superior del Módulo de Detección, actuando como el **punto de control manual** del sistema.

El sistema final, visible en la vista general, es una unidad compacta y autónoma que permite el flujo, la detección visual y la eventual clasificación de objetos.

### 3. Instalación del Software

#### 3.1. Código Arduino MEGA

```
1  #include <AFMotor.h>
2  #include <Servo.h>
3
4  // =====
5  //  CONFIGURACIÓN DE HARDWARE (ARDUINO MEGA)
6  //  =====
7
8  // 1. MOTOR DE LA CINTA (Shield L293D - Puerto M1)
9  AF_DCMotor motorCinta(1);
10
11 // 2. SERVOS
12 Servo servoMaduro; // Pin 9
13 Servo servoVerde;  // Pin 10
14
15 const int PIN_SERVO_MADURO = 9;
16 const int PIN_SERVO_VERDE  = 10;
17
18 // 3. SENSORES Y BOTONES
19 const int switchPin = A7; // Interruptor Maestro
20 const int trigPin   = A8; // Sensor Ultrasonico Trig
21 const int echoPin   = A9; // Sensor Ultrasonico Echo
22 const int pinLed     = A10; // Led indicador
23
24 // =====
25 //  CALIBRACIÓN DE TIEMPOS
26 //  =====
27
28 const int DISTANCIA_STOP = 12;
29
30 // VELOCIDAD DE TRABAJO (BAJADA A 70)
```

```
31 const int VELOCIDAD_TRABAJO = 70;
32
33 // TIEMPOS DE VIAJE
34 // Base para MADURO: 2 segundos
35 const unsigned long TIEMPO_VIAJE_MADURO = 2000;
36
37 // Base para VERDE: 1 segundo MÁS que el Maduro (Antes era +2000)
38 // Esto hace que active 1 segundo antes que la configuración previa
39 const unsigned long TIEMPO_VIAJE_VERDE = TIEMPO_VIAJE_MADURO + 1000;
40
41 // Tiempos de retención del brazo
42 const unsigned long TIEMPO_RETENCION_MADURO = 3000;
43 const unsigned long TIEMPO_RETENCION_VERDE = 5000;
44
45 // Tiempo de posicionamiento bajo cámara
46 const unsigned long TIEMPO_POSICIONAMIENTO = 800;
47 // =====
48
49 void setup() {
50     Serial.begin(9600);
51     Serial.println("SISTEMA TOMATAIT: LISTO (VELOCIDAD 70)");
52
53     // Comunicación con PICO W
54     Serial1.begin(9600);
55
56     // CONFIGURACIÓN MOTOR
57     motorCinta.setSpeed(VELOCIDAD_TRABAJO);
58     motorCinta.run(RELEASE);
59
60     pinMode(switchPin, INPUT_PULLUP);
61     pinMode(trigPin, OUTPUT);
62     pinMode(echoPin, INPUT);
63     pinMode(pinLed, OUTPUT);
64
65     servoMaduro.attach(PIN_SERVO_MADURO);
```



```
66  servoVerde.attach(PIN_SERVO_VERDE);
67
68  // === POSICIONES INICIALES DIFERENTES ===
69  // Maduro: Reposo en 180 (Lado A)
70  servoMaduro.write(180);
71
72  // Verde: Reposo en 0 (Lado B - Contrario)
73  servoVerde.write(0);
74 }
75
76 void loop() {
77  // === 1. VERIFICACIÓN MAESTRA ===
78  if (digitalRead(switchPin) == HIGH) {
79      apagarSistema();
80      return;
81  }
82
83  // === 2. BANDA AVANZANDO ===
84  digitalWrite(pinLed, HIGH);
85
86  // Velocidad constante a 70
87  motorCinta.setSpeed(VELOCIDAD_TRABAJO);
88  motorCinta.run(FORWARD);
89
90  // --- FILTRO DE SENSOR ---
91  long distancia = medirDistancia();
92
93  if (distancia > 0 && distancia <= DISTANCIA_STOP) {
94      // Doble confirmación
95      delay(50);
96      long confirmación = medirDistancia();
97      if (confirmación > 0 && confirmación <= DISTANCIA_STOP) {
98          Serial.println("TOMATE DETECTADO. POSICIONANDO...");
99          // Avanzamos para ponerlo bajo la cámara
100         if (!esperarSeguro(TIEMPO_POSICIONAMIENTO)) return;
```

```

101      // PAUSA LÓGICA PARA FOTO (Banda sigue rodando suave a 70)
102      Serial.println("Esperando clasificación IA (En movimiento)...");
103      while(Serial1.available() > 0) { Serial1.read(); }
104      bool datoRecibido = false;
105      char clasificación = 'N';
106      while (!datoRecibido) {
107          if (digitalRead(switchPin) == HIGH) { apagarSistema(); return; }
108          if (Serial1.available() > 0) {
109              clasificación = Serial1.read();
110              if (clasificación != '\n' && clasificación != '\r' &&
clasificación != ' ') {
111                  datoRecibido = true;
112              }
113          }
114          delay(10);
115      }
116
117      Serial.print("Clasificación: "); Serial.println(clasificación);
118
119      // === ACTUACIÓN SERVOS ===
120      motorCinta.setSpeed(VELOCIDAD_TRABAJO);
121      motorCinta.run(FORWARD);
122
123      if (clasificación == 'R') {
124          Serial.println("Viajando a MADURO...");
125          if (!esperarSeguro(TIEMPO_VIAJE_MADURO)) return;
126          Serial.println("SERVO MADURO (Normal)");
127
128          // Maduro usa lógica normal (180 -> 90 -> 180)
129          golpearNormal(servoMaduro, TIEMPO_RETENCION_MADURO);
130      }
131      else if (clasificación == 'U' || clasificación == 'H') {
132          Serial.println("Viajando a VERDE...");
133          if (!esperarSeguro(TIEMPO_VIAJE_VERDE)) return;
134          Serial.println("SERVO VERDE (Invertido)");
135

```

```
136          // Verde usa lógica invertida (0 -> 90 -> 0)
137          golpearInvertido(servoVerde, TIEMPO_RETENCION_VERDE);
138      }
139      else {
140          Serial.println("Descarte.");
141          if (!esperarSeguro(5000)) return;
142      }
143      Serial.println("Fin de ciclo. Esperando salida del sensor...\n");
144      esperarSeguro(2000);
145  }
146  }
147  delay(50);
148  }
149
150  // =====
151  //  FUNCIONES AUXILIARES
152  // =====
153
154  void apagarSistema() {
155      digitalWrite(pinLed, LOW);
156      motorCinta.run(RELEASE);
157      // Guardar brazos en sus respectivos lados
158      servoMaduro.write(180);
159      servoVerde.write(0);
160  }
161
162  bool esperarSeguro(unsigned long tiempoMs) {
163      unsigned long inicio = millis();
164      while (millis() - inicio < tiempoMs) {
165          if (digitalRead(switchPin) == HIGH) {
166              apagarSistema();
167              return false;
168          }
169          delay(10);
170      }
```

```
171     return true;
172 }
173
174 long medirDistancia() {
175     digitalWrite(trigPin, LOW); delayMicroseconds(2);
176     digitalWrite(trigPin, HIGH); delayMicroseconds(10);
177     digitalWrite(trigPin, LOW);
178     long duración = pulseIn(echoPin, HIGH, 30000);
179     if (duración == 0) return 999;
180     return (duración * 0.034 / 2);
181 }
182
183 // --- FUNCIÓN GOLPE NORMAL (Para Maduro) ---
184 // Reposo: 180 -> Activo: 90
185 void golpearNormal(Servo &s, unsigned long tiempoRetencion) {
186     s.write(135);
187     if(!esperarSeguro(tiempoRetencion)) return;
188     s.write(180);
189     if(!esperarSeguro(500)) return;
190 }
191
192 // --- FUNCIÓN GOLPE INVERTIDO (Para Verde) ---
193 // Reposo: 0 -> Activo: 90
194 void golpearInvertido(Servo &s, unsigned long tiempoRetencion) {
195     s.write(25);
196     if(!esperarSeguro(tiempoRetencion)) return;
197     s.write(0);
198     if(!esperarSeguro(500)) return;
199 }
```

### 3.2. Código en Raspberry Pi Pico W

```
1 import network
2 import time
3 from machine import UART, Pin
```

```
4 from umqtt.simple import MQTTClient

5 # ===== CONFIGURACIÓN =====

6 SSID = "iPhone de Joe Jose"      # Tu WiFi

7 PASSWORD = "jugodemaracuya"     # Tu Contraseña

8 BROKER_IP = "172.20.10.9"        # <--- ¡VERIFICA ESTA IP CON IPCONFIG!

9 TOPIC = b"robotica/frutas"      # Tema MQTT

10 # =====

11 # Comunicación con Arduino Mega

12 # GP0 (TX Pico) -> Arduino Mega Pin 19 (RX1)

13 # GP1 (RX Pico) -> Arduino Mega Pin 18 (TX1)

14 uart = UART(0, baudrate=9600, tx=Pin(0), rx=Pin(1))

15

16 # 1. Conectar al WiFi

17 wlan = network.WLAN(network.STA_IF)

18 wlan.active(True)

19 wlan.connect(SSID, PASSWORD)

20

21 print("Conectando al WiFi (Hotspot iPhone)...")

22 intentos = 0

23 while not wlan.isconnected() and intentos < 20:

24     time.sleep(1)

25     intentos += 1

26     print(".")

27

28 if wlan.isconnected():

29     print('WiFi Conectado:', wlan.ifconfig())

30 else:

31     print('Error: No se pudo conectar al WiFi. Revisa la clave.')

32

33 # Función para procesar mensajes

34 def mensaje_recibido(topic, msg):
```

```
35     try:
36         texto = msg.decode('utf-8')
37         print(f" Recibido: {texto}")
38
39         letra = 'N'
40         if texto == "ripe":
41             letra = 'R'
42         elif texto == "unripe" or texto == "half_ripe":
43             letra = 'U'
44         elif texto == "mold" or texto == "rotten":
45             letra = 'M'
46
47         if letra != 'N':
48             # Enviar al Arduino Mega
49             uart.write(letra + '\n')
50             print(f"Enviada '{letra}' al Arduino Mega")
51
52     except Exception as e:
53         print(f"Error procesando: {e}")
54
55 # 2. Conectar al Broker (Laptop)
56 try:
57     client = MQTTClient("Pico_Mensajero", BROKER_IP)
58     client.set_callback(mensaje_recibido)
59     client.connect()
60     client.subscribe(TOPIC)
61     print(f"Conectado a la Laptop ({BROKER_IP}). Esperando órdenes...")
62 except Exception as e:
63     print(f"❌ Error conectando a MQTT: {e}")
64     print("    -> Verifica la IP y el Firewall de Windows.")
65
```

```
66 # Bucle Principal
67 while True:
68     try:
69         client.check_msg()
70         time.sleep(0.01)
71     except OSError:
72         try:
73             print("Reconectando...")
74             client.connect()
75             client.subscribe(TOPIC)
76         except:
77             pass
```

### 3.3. *Instalación del software en la Raspberry Pi 5*

```
1  import cv2
2  from ultralytics import YOLO
3  import sys
4  import paho.mqtt.client as mqtt
5  import time
6  # ===== CONFIGURACIÓN =====
7  # IP DE TU LAPTOP (Broker Mosquitto)
8  # Debe ser la misma que pusiste en la Pico W
9  BROKER_ADDRESS = "172.20.10.9"
10 TOPIC = "robotica/frutas"
11
12 # Umbral de confianza para disparo automático (75%)
13 CONF_UMBRAL = 0.75
14 # =====
15
16 print("Cargando modelo YOLO (best.pt)...")
17 try:
18     model = YOLO('./best.pt')
```

```
19 except Exception as e:
20     print(f"Error cargando modelo: {e}")
21     sys.exit(1)
22
23 # Configurar Cliente MQTT
24 client = mqtt.Client("RaspberryPi_Auto")
25
26 def run_auto_system():
27     print(f"Conectando a MQTT en {BROKER_ADDRESS}...")
28     try:
29         client.connect(BROKER_ADDRESS, 1883, 60)
30         print("CONEXIÓN MQTT EXITOSA.")
31     except Exception as e:
32         print(f"✗ Error conectando al Broker: {e}")
33         print("    -> Verifica que Mosquitto esté corriendo en tu
34 Laptop.")
35         print("    -> Verifica que el Firewall de Windows permita el
36 puerto 1883.")
37         sys.exit(1)
38
39     print("Iniciando cámara...")
40     # Usamos backend V4L2 y formato MJPG para más velocidad
41     cap = cv2.VideoCapture(0, cv2.CAP_V4L2)
42     cap.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc('M', 'J', 'P',
43 'G'))
44     cap.set(cv2.CAP_PROP_FRAME_WIDTH, 800)
45     cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 600)
46     cap.set(cv2.CAP_PROP_BUFFERSIZE, 1)
47
48     if not cap.isOpened():
49         print("Error crítico: No se detecta cámara USB.")
50         sys.exit(1)
51
52     print("\nSISTEMA AUTOMÁTICO ACTIVO.")
```



```
53     print("Esperando tomate estático frente a la cámara...")
54
55     while True:
56         ret, frame = cap.read()
57         if not ret: continue
58
59         # Mostrar ventana (Si tienes monitor en la Pi)
60         cv2.imshow("Camara Clasificadora", frame)
61         if cv2.waitKey(1) & 0xFF == ord('q'): break
62
63         # 1. ANALIZAR CADA FRAME
64         # verbose=False para no llenar la consola de basura
65         results = model(frame, verbose=False)
66
67         mejor_clase = None
68         mayor_conf = 0.0
69
70         for result in results:
71             for box in result.bboxes:
72                 conf = float(box.conf)
73                 # Solo nos interesa si supera el 75% de certeza
74                 if conf > CONF_UMBRAL and conf > mayor_conf:
75                     mayor_conf = conf
76                     cls_id = int(box.cls)
77                     mejor_clase = model.names[cls_id]
78
79         # 2. SI ENCONTRAMOS UN TOMATE CLARO
80         if mejor_clase:
81             print(f"DETECTADO: {mejor_clase} (Confianza:
82 {mayor_conf:.2f})")
83
84         # Enviar MQTT a la Laptop
85         client.publish(TOPIC, mejor_clase)
86
```

```
87         print(f" Mensaje '{mejor_clase}' enviado. Pausando 5
88 seg...")
89
90         # 3. PAUSA (COOLDOWN)
91
92         # Esperamos 5 segundos para que:
93         # - La Laptop reciba el mensaje.
94         # - La Pico W se lo pase al Arduino.
95         # - El Arduino mueva la banda y clasifique.
96
97         time.sleep(5)
98
99         # Limpiar buffer de cámara (leer frames viejos para tener
100 imagen fresca)
101
102         for _ in range(5): cap.read()
103
104         print("Listo para el siguiente...")
105
106
107     cap.release()
108
109     cv2.destroyAllWindows()
110
111
112 if __name__ == "__main__":
113     run_auto_system()
```

#### 4. Uso básico de TomatAIT

- **Preparación e Inicio**

- **Encendido de Energía:** Conecte la **Fuente de Energía Externa de 12V** al Driver L293D y, por separado, encienda la Raspberry Pi 5.
- **Activación del Sistema:** Active el **Interruptor (Switch)** en el Módulo de Detección. Esto energiza el circuito de control del Arduino Mega y enciende los **LEDs** utilizados para optimizar la iluminación del área de detección.

- **Monitoreo de Espera:** La Raspberry Pi 5 y el Arduino Mega inician, pero el **motor DC** de la cinta permanece **detenido**. El sistema está esperando la detección de un tomate.
- **Detección y Transporte**
  - **Carga del Tomate:** Coloque un **tomate** en la parte inicial de la Cinta Transportadora, frente al **Sensor Ultrasónico**.
  - **Detección de Entrada:** El Sensor Ultrasónico detecta el tomate. Esta señal informa al Arduino Mega que debe iniciar el proceso de clasificación.
  - **Movimiento de la Cinta:** El Arduino Mega activa el Driver L293D para encender el **Motor DC**, transportando el tomate hacia el **Módulo de Detección**.
- **3. Procesamiento y Clasificación**
  - **Captura de Imagen:** Mientras el tomate pasa por debajo del Módulo de Detección, la **Webcam USB** captura continuamente imágenes y las envía a la **Raspberry Pi 5**.
  - **Inferencia de IA:** La Raspberry Pi 5 ejecuta el modelo de **visión YOLOv8** sobre las imágenes para determinar la clase de madurez (Verde, Pintón o Maduro).
  - **Comunicación de Resultado:** La Raspberry Pi 5 utiliza el canal de comunicación (MicroPython/Pico) para enviar el resultado final de la clasificación al **Arduino Mega**.
- **Actuación y Detención**
  - **Decisión del Motor:** El Arduino Mega recibe la orden. Si el tomate está **dentro del rango de detección efectivo**, el motor continúa moviéndose.

- **Clasificación (Actuación Futura):** En una implementación completa, el Arduino Mega activaría el **Servomotor** para desviar el tomate a su contenedor según la clasificación recibida (ej., desviación para "Maduro").
- **Detención de la Cinta:** Una vez que el tomate abandona la zona de detección/actuación, el Arduino Mega detiene el **Motor DC** (liberando el control del motor, RELEASE), y el sistema vuelve al estado de espera para el siguiente tomate.