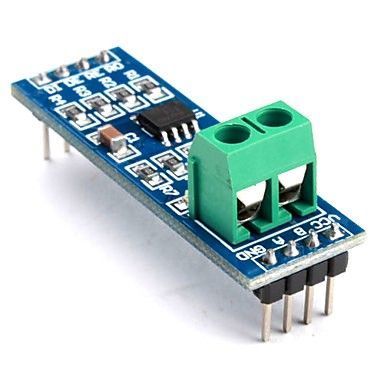
RS-485 Haberleşme Testi:



metin, elektronik donanım, bilgisayar donanımı, elektronik mühendisliği içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

elektronik donanım, elektronik mühendisliği, kablo, Elektrik kabloları içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

metin, bilgisayar, ekran görüntüsü, ekran, görüntüleme içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

metin, bilgisayar, ekran, görüntüleme, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

RS485 GÖNDERİCİ KOD:

#include <SoftwareSerial.h>

#define RO 10 // RS485 modülü RX pini (Arduino'ya TX olacak)

#define DI 11 // RS485 modülü TX pini (Arduino'ya RX olacak)

SoftwareSerial rs485(RO, DI); // RX, TX

void setup() {

Serial.begin(9600); // Seri Monitör

rs485.begin(9600); // RS485 haberleşmesi

}

void loop() {

rs485.println("Merhaba RS485!");

Serial.println("Veri gönderildi: Merhaba RS485!");

  delay(1000);

}

RS485 ALICI:

#include <SoftwareSerial.h>

#define RO 10 // RS485 modülü RX pini (Arduino'ya TX olacak)

#define DI 11 // RS485 modülü TX pini (Arduino'ya RX olacak)

SoftwareSerial rs485(RO, DI); // RX, TX

void setup() {

Serial.begin(9600); // Seri Monitör

rs485.begin(9600); // RS485 haberleşmesi

}

void loop() {

if (rs485.available()) {

String receivedData = rs485.readString();

Serial.print("Gelen Veri: ");

Serial.println(receivedData);

    }

}

TOF1020 Test:



The TOF10120 range sensor provides accurate and repeatable long-distance measurements for high-speed autofocus (AF). Innovative TOF time-of-flight technology makes the sensor performance independent of the reflectivity of the target object.

TOF1020 KOD:

#include <SoftwareSerial.h>

SoftwareSerial tofSerial(10, 11); // RX, TX (TOF1020 bağlantıları)

int referenceDistance = -1; // Başlangıçta bilinmeyen referans mesafesi

int currentDistance = 0; // Anlık mesafe

int positionOffset = 0; // Referansa göre fark (konum değişimi)

void setup() {

Serial.begin(115200); // PC ile haberleşme

tofSerial.begin(9600); // TOF1020'nin UART hızı

delay(2000); // Sensörün stabil olması için bekleme süresi

// İlk okuma ile referans mesafeyi belirle

referenceDistance = getDistance();

Serial.print("Referans Mesafe Belirlendi: ");

Serial.print(referenceDistance);

Serial.println(" mm");

}

void loop() {

currentDistance = getDistance(); // Güncel mesafe oku

if (currentDistance > 0) { // Geçerli bir mesafe ölçümü varsa

positionOffset = currentDistance - referenceDistance; // Konum değişimi hesapla

Serial.print("Mesafe: ");

Serial.print(currentDistance);

Serial.print(" mm, Konum Farkı: ");

Serial.print(positionOffset);

Serial.println(" mm");

}

delay(100); // Sensörü çok sık sorgulamamak için bekleme süresi

}

// Sensörden UART üzerinden mesafeyi oku

int getDistance() {

char buffer[10];

int index = 0;

while (tofSerial.available() && index < 10) {

char c = tofSerial.read();

if (c == '\n') break; // Satır sonu geldiğinde çık

buffer[index++] = c;

}

buffer[index] = '\0'; // Stringin sonunu belirle

return atoi(buffer); // String'i integer'a çevirip döndür

}

TF MINI V2 :



This product is based on ToF (Time of Flight) principle and integrated with unique optical and electrical designs, so as to achieve stable, precise, high sensitivity and high-speed distance detection.

TF MINI V2  KOD:

#include <SoftwareSerial.h>

SoftwareSerial tfminiSerial(10, 11); // RX, TX (TF Mini V2 bağlantıları)

int referenceDistance = -1; // Başlangıçta bilinmeyen referans mesafesi

int currentDistance = 0; // Anlık mesafe

int positionOffset = 0; // Referansa göre fark (konum değişimi)

void setup() {

Serial.begin(115200); // PC ile haberleşme

tfminiSerial.begin(115200); // TF Mini V2'nin varsayılan UART hızı

delay(2000); // Sensörün stabil olması için bekleme süresi

// İlk okuma ile referans mesafeyi belirle

referenceDistance = getDistance();

Serial.print("Referans Mesafe Belirlendi: ");

Serial.print(referenceDistance);

Serial.println(" mm");

}

void loop() {

currentDistance = getDistance(); // Güncel mesafe oku

if (currentDistance > 0) { // Geçerli bir mesafe ölçümü varsa

positionOffset = currentDistance - referenceDistance; // Konum değişimi hesapla

Serial.print("Mesafe: ");

Serial.print(currentDistance);

Serial.print(" mm, Konum Farkı: ");

Serial.print(positionOffset);

Serial.println(" mm");

}

}

// TF Mini V2'den UART üzerinden mesafeyi oku

int getDistance() {

static uint8\_t buffer[9]; // TF Mini V2 veri paketi 9 bayttır

int distance = -1;

if (tfminiSerial.available() >= 9) { // 9 baytlık veri beklenmeli

if (tfminiSerial.read() == 0x59) { // İlk başlık baytı (0x59)

if (tfminiSerial.read() == 0x59) { // İkinci başlık baytı (0x59)

for (int i = 0; i < 7; i++) {

buffer[i] = tfminiSerial.read(); // Kalan veriyi oku

}

uint16\_t checksum = 0x59 + 0x59;

for (int i = 0; i < 7; i++) {

checksum += buffer[i];

}

if (checksum % 256 == buffer[6]) { // Checksum doğrulaması

distance = buffer[0] + (buffer[1] << 8); // Mesafe hesapla

}

}

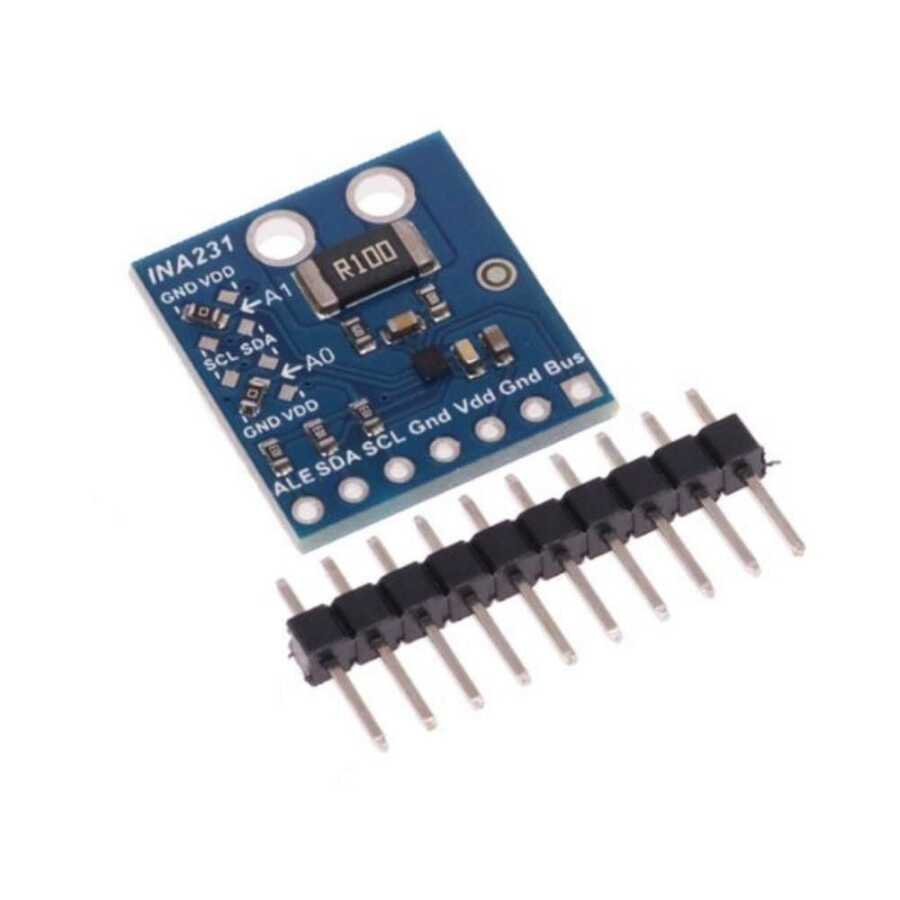
}

}

return distance;

}

INA231 I2C VOLTAJ:



The INA231 is a current-shunt and power monitor with a 1.8-V compliant I2C interface that features 16 programmable addresses.

INA231 KOD:

#include <Wire.h>

#define INA231\_ADDR 0x40 // I²C adresi (A0 ve A1 GND'ye bağlıysa)

void setup() {

Wire.begin();

Serial.begin(115200);

// INA231 yapılandırma (continuous mode)

Wire.beginTransmission(INA231\_ADDR);

Wire.write(0x00); // Konfigürasyon Registeri (0x00)

Wire.write(0x40); // Mode: Continuous

Wire.write(0x27);

Wire.endTransmission();

}

void loop() {

int16\_t shunt\_voltage, bus\_voltage\_raw;

float bus\_voltage, current, power;

// 🔹 ⿡ Bus Voltage Oku (Gerilim)

Wire.beginTransmission(INA231\_ADDR);

Wire.write(0x02); // Bus Voltage Register

Wire.endTransmission();

Wire.requestFrom(INA231\_ADDR, 2);

if (Wire.available() == 2) { // Veri gelmezse hata önler

bus\_voltage\_raw = Wire.read() << 8 | Wire.read();

bus\_voltage = ((bus\_voltage\_raw >> 3) \* 1.25) / 1000.0; // V cinsine çevir

Serial.print("Ham Gerilim: "); Serial.print(bus\_voltage\_raw); Serial.println(" V");

} else {

Serial.println("HATA: Bus Voltage okunamadı!");

return;

}

// 🔹 ⿢ Shunt Voltage Oku (Akım için)

Wire.beginTransmission(INA231\_ADDR);

Wire.write(0x01); // Shunt Voltage Register

Wire.endTransmission();

Wire.requestFrom(INA231\_ADDR, 2);

if (Wire.available() == 2) {

shunt\_voltage = Wire.read() << 8 | Wire.read();

float shunt\_voltage\_mV = shunt\_voltage \* 2.5 / 100000.0; // mV to V dönüşümü

float shunt\_resistance = 0.1; // Shunt direnci (örn: 0.1 ohm)

current = shunt\_voltage\_mV / shunt\_resistance; // I = V / R

} else {

Serial.println("HATA: Shunt Voltage okunamadı!");

return;

}

// 🔹 ⿣ Güç Hesaplama (P = V \* I)

power = bus\_voltage \* current;

// 🔹 ⿤ Seri Monitöre Yazdır

Serial.print("Gerilim: "); Serial.print(bus\_voltage); Serial.println(" V");

Serial.print("Akım: "); Serial.print(current); Serial.println(" A");

Serial.print("Güç: "); Serial.print(power); Serial.println(" W");

Serial.println("----------------------------");

  delay(1000);

}

VL53L3C :



This sensor is a carrier/breakout board for ST’s VL53L3CX laser-ranging sensor, which offers fast and accurate ranging **up to 5 m**. It uses the time of flight (ToF) of invisible, eye-safe laser pulses to measure absolute distances to **multiple targets** simultaneously, independent of ambient lighting conditions and target characteristics like color, shape, and texture (though these things will affect the maximum range). Distance measurements can be read through a digital I²C interface.

VL53L3C KOD:

#include <Wire.h>

#include "PWFusion\_VL53L3C.h"

VL53L3C TOF;

void setup() {

// Initialize I2C and serial port

Wire.begin();

Wire.setClock(400000);

Serial.begin(9600);

Serial.println(F("Initilizing VL53L3"));

TOF.begin();

Serial.println(F("VL53L3 Boot Complete"));

// Set the timing budget (sample period) to 100ms

TOF.setTimingBudget(100000);

// Start sampling data

TOF.startMeasurement();

Serial.println(F("Starting TOF measurements"));

}

void loop() {

MeasurmentResult measResult;

// Has the VL53L3 finished a new distance mesaurement

if (TOF.dataIsReady())

{

// Get the last measurement result and trigger a new measurement

TOF.getMeasurmentData(&measResult);

TOF.startNextMeasurement();

// The VL53L3 can track up to four onject simultaneously. Prind the

// measured distance to each detected onject.

Serial.print(F("Range Measurement: "));

for (int i=0; i<measResult.numObjs; i++)

{

Serial.print(measResult.rangeData[i].Range);

Serial.print(F("mm "));

}

Serial.println();

  }

}

Hall Motor PR-520EV-17355:

kablo, elektronik donanım, Elektrik kabloları, dizüstü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

LDS02RR MOTOR:



LDS02RR- Hall Motor PR-520EV-17355 KOD:

#define TACHO\_PIN 2 // Lidar hız sinyali (TACHO veya FG çıkışı)

volatile int pulse\_count = 0;

unsigned long last\_time = 0;

float motor\_speed\_rpm = 0;

// Kesme fonksiyonu (Her darbeyi sayar)

void countPulse() {

pulse\_count++;

}

void setup() {

Serial.begin(115200);

pinMode(TACHO\_PIN, INPUT\_PULLUP);

attachInterrupt(digitalPinToInterrupt(TACHO\_PIN), countPulse, RISING); // Her yükselen kenarda sayar

}

void loop() {

unsigned long current\_time = millis();

if (current\_time - last\_time >= 1000) { // Her 1 saniyede bir ölçüm yap

noInterrupts();

int count = pulse\_count;

pulse\_count = 0;

interrupts();

// Genellikle lidarın 1 devir için kaç darbe gönderdiği modeline göre değişir (örneğin 6 darbe = 1 tam devir)

int pulses\_per\_revolution = 6; // Lidar modeline bağlı, değiştirilebilir

motor\_speed\_rpm = (count / (float)pulses\_per\_revolution) \* 60.0;

Serial.print("Lidar Motor Hızı: ");

Serial.print(motor\_speed\_rpm);

Serial.println(" RPM");

last\_time = current\_time;

  }

}

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

metin, ekran görüntüsü, kitap, bilgisayar içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Analog 2 AXIS Gimbal Control with PWM(Using ADC):

Elektrik kabloları, elektronik mühendisliği, kablo, elektronik donanım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

KOD (ARDUINO UNO):

// Joystick girişleri

#define JOYSTICK\_X A0 // X ekseni (Roll)

#define JOYSTICK\_Y A1 // Y ekseni (Pitch)

// STorM32 PWM çıkışları

#define ROLL\_PWM 9 // Roll için PWM çıkışı

#define PITCH\_PWM 10 // Pitch için PWM çıkışı

void setup() {

pinMode(ROLL\_PWM, OUTPUT);

pinMode(PITCH\_PWM, OUTPUT);

pinMode(JOYSTICK\_X, INPUT);

pinMode(JOYSTICK\_Y, INPUT);

Serial.begin(9600); // Seri monitör için

}

void loop() {

// Joystick değerlerini oku

int xValue = analogRead(JOYSTICK\_X); // 0-1023

int yValue = analogRead(JOYSTICK\_Y); // 0-1023

// Analog değerleri PWM sinyaline çevir (1000 - 2000 mikro saniye)

int rollPWM = map(xValue, 0, 1023, 1000, 2000);

int pitchPWM = map(yValue, 0, 1023, 1000, 2000);

// PWM sinyallerini gönder

sendPWM(ROLL\_PWM, rollPWM);

sendPWM(PITCH\_PWM, pitchPWM);

// Seri monitörden değerleri görmek için

Serial.print("Roll PWM: "); Serial.print(rollPWM);

Serial.print(" | Pitch PWM: "); Serial.println(pitchPWM);

delay(10); // Stabil çalışma için kısa bekleme

}

// PWM sinyali gönderen fonksiyon

void sendPWM(int pin, int pulseWidth) {

digitalWrite(pin, HIGH);

delayMicroseconds(pulseWidth);

digitalWrite(pin, LOW);

delayMicroseconds(20000 - pulseWidth);

}

KOD (STM32):

/\* USER CODE BEGIN Header \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* @brief : Main program body

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* Copyright (c) 2025 STMicroelectronics.

\* All rights reserved.

\*

\* This software is licensed under terms that can be found in the LICENSE file

\* in the root directory of this software component.

\* If no LICENSE file comes with this software, it is provided AS-IS.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

#include "main.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

/\* USER CODE END Includes \*/

/\* Private typedef -----------------------------------------------------------\*/

/\* USER CODE BEGIN PTD \*/

/\* USER CODE END PTD \*/

/\* Private define ------------------------------------------------------------\*/

/\* USER CODE BEGIN PD \*/

/\* USER CODE END PD \*/

/\* Private macro -------------------------------------------------------------\*/

/\* USER CODE BEGIN PM \*/

/\* USER CODE END PM \*/

/\* Private variables ---------------------------------------------------------\*/

TIM\_HandleTypeDef htim3;

/\* USER CODE BEGIN PV \*/

TIM\_HandleTypeDef htim3; // Timer 3 (PWM için)

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

void SystemClock\_Config(void);

static void MX\_GPIO\_Init(void);

static void MX\_TIM3\_Init(void);

/\* USER CODE BEGIN PFP \*/

void ReadJoystickAndUpdatePWM(void);

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

// PWM genişlik değerleri (mikrosaniye cinsinden)

#define PWM\_MIN 1000 // 1ms minimum PWM değeri (Servo ESC için)

#define PWM\_MAX 2000 // 2ms maksimum PWM değeri (Servo ESC için)

// PWM sinyalini belirli bir duty cycle ile ayarlayan fonksiyon

void set\_PWM\_duty\_cycle(TIM\_HandleTypeDef \*htim, uint32\_t channel, uint16\_t pulse)

{

\_\_HAL\_TIM\_SET\_COMPARE(htim, channel, pulse);

}

/\* USER CODE END 0 \*/

/\*\*

\* @brief The application entry point.

\* @retval int

\*/

int main(void)

{

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

/\* MCU Configuration--------------------------------------------------------\*/

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

HAL\_Init();

/\* USER CODE BEGIN Init \*/

/\* USER CODE END Init \*/

/\* Configure the system clock \*/

SystemClock\_Config();

/\* USER CODE BEGIN SysInit \*/

/\* USER CODE END SysInit \*/

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

MX\_TIM3\_Init();

/\* USER CODE BEGIN 2 \*/

void move\_pitch(int pitch\_angle)

{

// Pitch açısını PWM değerine dönüştür (1ms - 2ms arasında)

uint16\_t pitch\_pwm = PWM\_MIN + ((PWM\_MAX - PWM\_MIN) \* (pitch\_angle + 90)) / 180;

// Pitch için PWM sinyalini uygula

set\_PWM\_duty\_cycle(&htim3, TIM\_CHANNEL\_2, pitch\_pwm);

}

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

while (1)

{

move\_pitch(0); // Pitch ekseni orta konum

HAL\_Delay(1000);

move\_pitch(30); // Pitch ekseni +30 dereceye hareket etsin

HAL\_Delay(1000);

move\_pitch(-30); // Pitch ekseni -30 dereceye hareket etsin

HAL\_Delay(1000);

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

}

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

void SystemClock\_Config(void)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

/\*\* Configure the main internal regulator output voltage

\*/

\_\_HAL\_RCC\_PWR\_CLK\_ENABLE();

\_\_HAL\_PWR\_VOLTAGESCALING\_CONFIG(PWR\_REGULATOR\_VOLTAGE\_SCALE1);

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSI;

RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;

RCC\_OscInitStruct.HSICalibrationValue = RCC\_HSICALIBRATION\_DEFAULT;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_NONE;

if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Initializes the CPU, AHB and APB buses clocks

\*/

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_HSI;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV1;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_0) != HAL\_OK)

{

Error\_Handler();

}

}

/\*\*

\* @brief TIM3 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_TIM3\_Init(void)

{

/\* USER CODE BEGIN TIM3\_Init 0 \*/

/\* USER CODE END TIM3\_Init 0 \*/

TIM\_ClockConfigTypeDef sClockSourceConfig = {0};

TIM\_MasterConfigTypeDef sMasterConfig = {0};

TIM\_OC\_InitTypeDef sConfigOC = {0};

/\* USER CODE BEGIN TIM3\_Init 1 \*/

/\* USER CODE END TIM3\_Init 1 \*/

htim3.Instance = TIM3;

htim3.Init.Prescaler = 83;

htim3.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim3.Init.Period = 1999;

htim3.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim3.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_ENABLE;

if (HAL\_TIM\_Base\_Init(&htim3) != HAL\_OK)

{

Error\_Handler();

}

sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;

if (HAL\_TIM\_ConfigClockSource(&htim3, &sClockSourceConfig) != HAL\_OK)

{

Error\_Handler();

}

if (HAL\_TIM\_PWM\_Init(&htim3) != HAL\_OK)

{

Error\_Handler();

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

if (HAL\_TIMEx\_MasterConfigSynchronization(&htim3, &sMasterConfig) != HAL\_OK)

{

Error\_Handler();

}

sConfigOC.OCMode = TIM\_OCMODE\_PWM1;

sConfigOC.Pulse = 0;

sConfigOC.OCPolarity = TIM\_OCPOLARITY\_LOW;

sConfigOC.OCFastMode = TIM\_OCFAST\_DISABLE;

if (HAL\_TIM\_PWM\_ConfigChannel(&htim3, &sConfigOC, TIM\_CHANNEL\_1) != HAL\_OK)

{

Error\_Handler();

}

if (HAL\_TIM\_PWM\_ConfigChannel(&htim3, &sConfigOC, TIM\_CHANNEL\_2) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN TIM3\_Init 2 \*/

HAL\_TIM\_PWM\_Start(&htim3, TIM\_CHANNEL\_1);

HAL\_TIM\_PWM\_Start(&htim3, TIM\_CHANNEL\_2);

/\* USER CODE END TIM3\_Init 2 \*/

HAL\_TIM\_MspPostInit(&htim3);

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_GPIO\_Init(void)

{

/\* USER CODE BEGIN MX\_GPIO\_Init\_1 \*/

void HAL\_TIM\_MspPostInit(TIM\_HandleTypeDef \*htim)

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

if (htim->Instance == TIM3)

{

\_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();

GPIO\_InitStruct.Pin = GPIO\_PIN\_4 | GPIO\_PIN\_5;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

GPIO\_InitStruct.Alternate = GPIO\_AF2\_TIM3;

HAL\_GPIO\_Init(GPIOB, &GPIO\_InitStruct);

}

}

/\* USER CODE END MX\_GPIO\_Init\_1 \*/

/\* GPIO Ports Clock Enable \*/

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

/\* USER CODE BEGIN MX\_GPIO\_Init\_2 \*/

/\* USER CODE END MX\_GPIO\_Init\_2 \*/

}

/\* USER CODE BEGIN 4 \*/

/\* USER CODE END 4 \*/

/\*\*

\* @brief This function is executed in case of error occurrence.

\* @retval None

\*/

void Error\_Handler(void)

{

/\* USER CODE BEGIN Error\_Handler\_Debug \*/

/\* User can add his own implementation to report the HAL error return state \*/

\_\_disable\_irq();

while (1)

{

}

/\* USER CODE END Error\_Handler\_Debug \*/

}

#ifdef USE\_FULL\_ASSERT

/\*\*

\* @brief Reports the name of the source file and the source line number

\* where the assert\_param error has occurred.

\* @param file: pointer to the source file name

\* @param line: assert\_param error line source number

\* @retval None

\*/

void assert\_failed(uint8\_t \*file, uint32\_t line)

{

/\* USER CODE BEGIN 6 \*/

/\* User can add his own implementation to report the file name and line number,

ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

/\* USER CODE END 6 \*/

}

#endif /\* USE\_FULL\_ASSERT \*/

STM32 PARAMETERS:

metin, ekran görüntüsü, yazılım, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

GETTING GPS SIGNAL BY USING MINI GPS REFERENCED CLOCK:

kablo, elektronik mühendisliği, elektronik donanım, Elektrik kabloları içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.metin, bina, dış mekan içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

metin, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, yazılım, çizgi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

USING THIS GPS SIGNAL AS A REFERANCE FOR OUR GIMBALL PWM SIGNAL BY RTC PINOUT:

metin, ekran görüntüsü, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, çizgi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

STM32 KOD:

/\* USER CODE BEGIN Header \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* @brief : Main program body

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* Copyright (c) 2025 STMicroelectronics.

\* All rights reserved.

\*

\* This software is licensed under terms that can be found in the LICENSE file

\* in the root directory of this software component.

\* If no LICENSE file comes with this software, it is provided AS-IS.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

#include "main.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

/\* USER CODE END Includes \*/

/\* Private typedef -----------------------------------------------------------\*/

/\* USER CODE BEGIN PTD \*/

/\* USER CODE END PTD \*/

/\* Private define ------------------------------------------------------------\*/

/\* USER CODE BEGIN PD \*/

/\* USER CODE END PD \*/

/\* Private macro -------------------------------------------------------------\*/

/\* USER CODE BEGIN PM \*/

/\* USER CODE END PM \*/

/\* Private variables ---------------------------------------------------------\*/

RTC\_HandleTypeDef hrtc;

TIM\_HandleTypeDef htim3;

/\* USER CODE BEGIN PV \*/

TIM\_HandleTypeDef htim3; // Timer 3 (PWM için)

volatile uint32\_t ppm\_last\_capture = 0;

volatile uint32\_t ppm\_pulse\_width = 0;

volatile uint32\_t ppm\_channels[8]; // 8 kanal PPM desteği

volatile uint8\_t ppm\_channel\_index = 0;

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

void SystemClock\_Config(void);

static void MX\_GPIO\_Init(void);

static void MX\_TIM3\_Init(void);

static void MX\_RTC\_Init(void);

/\* USER CODE BEGIN PFP \*/

void ReadJoystickAndUpdatePWM(void);

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

// PWM genişlik değerleri (mikrosaniye cinsinden)

#define PWM\_MIN 1000 // 1ms minimum PWM değeri (Servo ESC için)

#define PWM\_MAX 2000 // 2ms maksimum PWM değeri (Servo ESC için)

// PWM sinyalini belirli bir duty cycle ile ayarlayan fonksiyon

void set\_PWM\_duty\_cycle(TIM\_HandleTypeDef \*htim, uint32\_t channel, uint16\_t pulse)

{

\_\_HAL\_TIM\_SET\_COMPARE(htim, channel, pulse);

}

/\* USER CODE END 0 \*/

/\*\*

\* @brief The application entry point.

\* @retval int

\*/

int main(void)

{

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

/\* MCU Configuration--------------------------------------------------------\*/

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

HAL\_Init();

/\* USER CODE BEGIN Init \*/

/\* USER CODE END Init \*/

/\* Configure the system clock \*/

SystemClock\_Config();

/\* USER CODE BEGIN SysInit \*/

/\* USER CODE END SysInit \*/

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

MX\_TIM3\_Init();

MX\_RTC\_Init();

/\* USER CODE BEGIN 2 \*/

void move\_gimbal(int roll\_angle, int pitch\_angle)

{

// Gelen açıları PWM değerine dönüştür (1ms - 2ms arasında)

uint16\_t roll\_pwm = PWM\_MIN + ((PWM\_MAX - PWM\_MIN) \* (roll\_angle + 90)) / 180;

uint16\_t pitch\_pwm = PWM\_MIN + ((PWM\_MAX - PWM\_MIN) \* (pitch\_angle + 90)) / 180;

// PWM sinyalini uygula

set\_PWM\_duty\_cycle(&htim3, TIM\_CHANNEL\_1, roll\_pwm);

set\_PWM\_duty\_cycle(&htim3, TIM\_CHANNEL\_2, pitch\_pwm);

}

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

while (1)

{

move\_gimbal(0, 0); // Orta konum

HAL\_Delay(1000);

move\_gimbal(30, -20); // Roll: +30, Pitch: -30

HAL\_Delay(1000);

move\_gimbal(-30, 20); // Roll: -30, Pitch: +30

HAL\_Delay(1000);

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

}

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

void SystemClock\_Config(void)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

/\*\* Configure the main internal regulator output voltage

\*/

\_\_HAL\_RCC\_PWR\_CLK\_ENABLE();

\_\_HAL\_PWR\_VOLTAGESCALING\_CONFIG(PWR\_REGULATOR\_VOLTAGE\_SCALE1);

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSI|RCC\_OSCILLATORTYPE\_LSI;

RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;

RCC\_OscInitStruct.HSICalibrationValue = RCC\_HSICALIBRATION\_DEFAULT;

RCC\_OscInitStruct.LSIState = RCC\_LSI\_ON;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_NONE;

if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Initializes the CPU, AHB and APB buses clocks

\*/

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_HSI;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV1;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_0) != HAL\_OK)

{

Error\_Handler();

}

}

/\*\*

\* @brief RTC Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_RTC\_Init(void)

{

/\* USER CODE BEGIN RTC\_Init 0 \*/

/\* USER CODE END RTC\_Init 0 \*/

/\* USER CODE BEGIN RTC\_Init 1 \*/

/\* USER CODE END RTC\_Init 1 \*/

/\*\* Initialize RTC Only

\*/

hrtc.Instance = RTC;

hrtc.Init.HourFormat = RTC\_HOURFORMAT\_24;

hrtc.Init.AsynchPrediv = 127;

hrtc.Init.SynchPrediv = 255;

hrtc.Init.OutPut = RTC\_OUTPUT\_DISABLE;

hrtc.Init.OutPutPolarity = RTC\_OUTPUT\_POLARITY\_HIGH;

hrtc.Init.OutPutType = RTC\_OUTPUT\_TYPE\_OPENDRAIN;

if (HAL\_RTC\_Init(&hrtc) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Enable the reference Clock input

\*/

if (HAL\_RTCEx\_SetRefClock(&hrtc) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN RTC\_Init 2 \*/

/\* USER CODE END RTC\_Init 2 \*/

}

/\*\*

\* @brief TIM3 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_TIM3\_Init(void)

{

/\* USER CODE BEGIN TIM3\_Init 0 \*/

/\* USER CODE END TIM3\_Init 0 \*/

TIM\_ClockConfigTypeDef sClockSourceConfig = {0};

TIM\_MasterConfigTypeDef sMasterConfig = {0};

TIM\_OC\_InitTypeDef sConfigOC = {0};

/\* USER CODE BEGIN TIM3\_Init 1 \*/

/\* USER CODE END TIM3\_Init 1 \*/

htim3.Instance = TIM3;

htim3.Init.Prescaler = 83;

htim3.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim3.Init.Period = 1999;

htim3.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim3.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_ENABLE;

if (HAL\_TIM\_Base\_Init(&htim3) != HAL\_OK)

{

Error\_Handler();

}

sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;

if (HAL\_TIM\_ConfigClockSource(&htim3, &sClockSourceConfig) != HAL\_OK)

{

Error\_Handler();

}

if (HAL\_TIM\_PWM\_Init(&htim3) != HAL\_OK)

{

Error\_Handler();

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

if (HAL\_TIMEx\_MasterConfigSynchronization(&htim3, &sMasterConfig) != HAL\_OK)

{

Error\_Handler();

}

sConfigOC.OCMode = TIM\_OCMODE\_PWM1;

sConfigOC.Pulse = 0;

sConfigOC.OCPolarity = TIM\_OCPOLARITY\_LOW;

sConfigOC.OCFastMode = TIM\_OCFAST\_DISABLE;

if (HAL\_TIM\_PWM\_ConfigChannel(&htim3, &sConfigOC, TIM\_CHANNEL\_1) != HAL\_OK)

{

Error\_Handler();

}

if (HAL\_TIM\_PWM\_ConfigChannel(&htim3, &sConfigOC, TIM\_CHANNEL\_2) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN TIM3\_Init 2 \*/

HAL\_TIM\_PWM\_Start(&htim3, TIM\_CHANNEL\_1);

HAL\_TIM\_PWM\_Start(&htim3, TIM\_CHANNEL\_2);

/\* USER CODE END TIM3\_Init 2 \*/

HAL\_TIM\_MspPostInit(&htim3);

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_GPIO\_Init(void)

{

/\* USER CODE BEGIN MX\_GPIO\_Init\_1 \*/

void HAL\_TIM\_MspPostInit(TIM\_HandleTypeDef \*htim)

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

if (htim->Instance == TIM3)

{

\_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();

GPIO\_InitStruct.Pin = GPIO\_PIN\_4 | GPIO\_PIN\_5;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

GPIO\_InitStruct.Alternate = GPIO\_AF2\_TIM3;

HAL\_GPIO\_Init(GPIOB, &GPIO\_InitStruct);

}

}

/\* USER CODE END MX\_GPIO\_Init\_1 \*/

/\* GPIO Ports Clock Enable \*/

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();

/\* USER CODE BEGIN MX\_GPIO\_Init\_2 \*/

void MX\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE(); // GPIOA saatini etkinleştir

// PPM sinyali için PA0 yapılandırması

GPIO\_InitStruct.Pin = GPIO\_PIN\_0;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP; // Alternatif fonksiyon modu

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

GPIO\_InitStruct.Alternate = GPIO\_AF1\_TIM2; // PA0 için TIM2\_CH1 alternatifi

HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);

}

/\* USER CODE END MX\_GPIO\_Init\_2 \*/

}

/\* USER CODE BEGIN 4 \*/

void HAL\_TIM\_IC\_CaptureCallback(TIM\_HandleTypeDef \*htim)

{

if (htim->Instance == TIM2 && htim->Channel == HAL\_TIM\_ACTIVE\_CHANNEL\_1)

{

uint32\_t current\_capture = HAL\_TIM\_ReadCapturedValue(htim, TIM\_CHANNEL\_1);

if (current\_capture >= ppm\_last\_capture)

{

ppm\_pulse\_width = current\_capture - ppm\_last\_capture;

}

else

{

ppm\_pulse\_width = (0xFFFF - ppm\_last\_capture) + current\_capture;

}

ppm\_last\_capture = current\_capture;

// PPM çerçevesini algılamak için uzun pulse kontrolü

if (ppm\_pulse\_width > 3000) // 3ms üzerindeki genişlik çerçeve başlangıcı olabilir

{

ppm\_channel\_index = 0; // Yeni çerçeve başlangıcı

}

else

{

ppm\_channels[ppm\_channel\_index++] = ppm\_pulse\_width;

}

// RTC ile senkronizasyon (GPS saatine göre zaman damgalama)

RTC\_TimeTypeDef sTime;

HAL\_RTC\_GetTime(&hrtc, &sTime, RTC\_FORMAT\_BIN);

printf("RTC Time: %02d:%02d:%02d | PPM Channel %d: %lu us\n",

sTime.Hours, sTime.Minutes, sTime.Seconds, ppm\_channel\_index, ppm\_pulse\_width);

}

}

/\* USER CODE END 4 \*/

/\*\*

\* @brief This function is executed in case of error occurrence.

\* @retval None

\*/

void Error\_Handler(void)

{

/\* USER CODE BEGIN Error\_Handler\_Debug \*/

/\* User can add his own implementation to report the HAL error return state \*/

\_\_disable\_irq();

while (1)

{

}

/\* USER CODE END Error\_Handler\_Debug \*/

}

#ifdef USE\_FULL\_ASSERT

/\*\*

\* @brief Reports the name of the source file and the source line number

\* where the assert\_param error has occurred.

\* @param file: pointer to the source file name

\* @param line: assert\_param error line source number

\* @retval None

\*/

void assert\_failed(uint8\_t \*file, uint32\_t line)

{

/\* USER CODE BEGIN 6 \*/

/\* User can add his own implementation to report the file name and line number,

ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

/\* USER CODE END 6 \*/

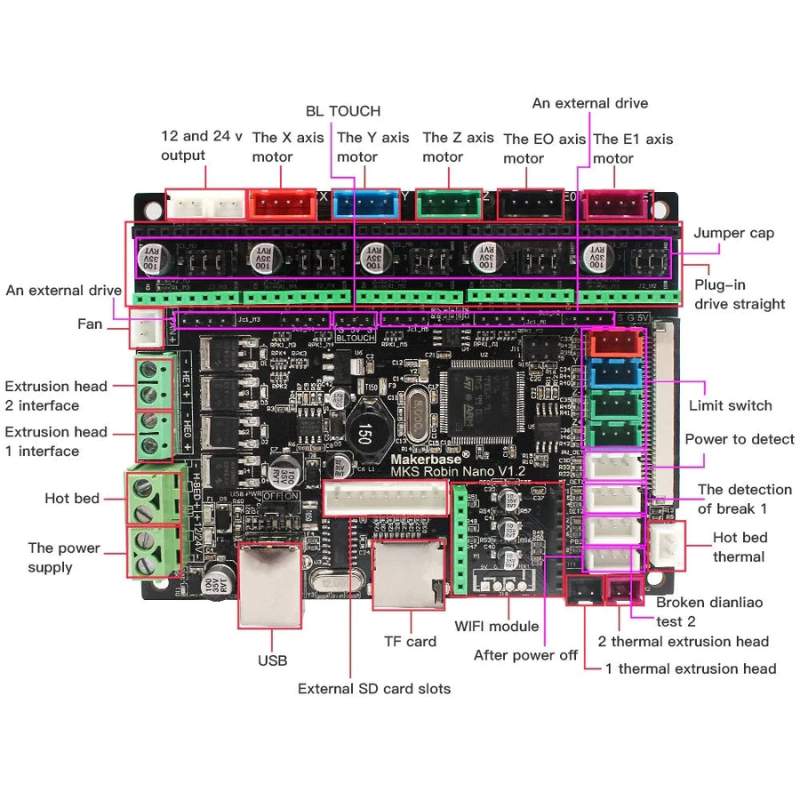
}

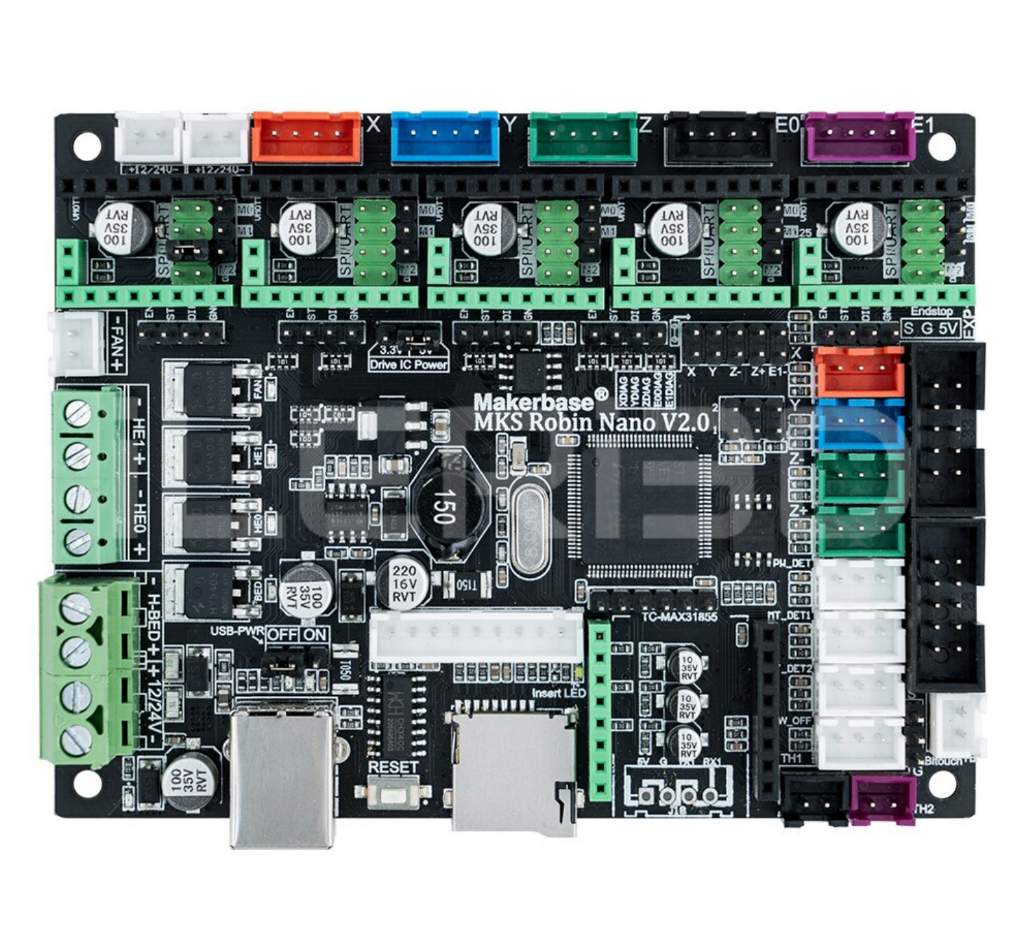
#endif /\* USE\_FULL\_ASSERT \*/

metin, ekran görüntüsü, sayı, numara, yazılım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

NEMA17 MOTOR CONTROL OF GIMBALL’S 3 AXIS STEP MOTORS USING MKS Robin Nano V1.3 AND MKS NANO 2.0





elektronik donanım, makine, kablo, Elektrik kabloları içeren bir resim

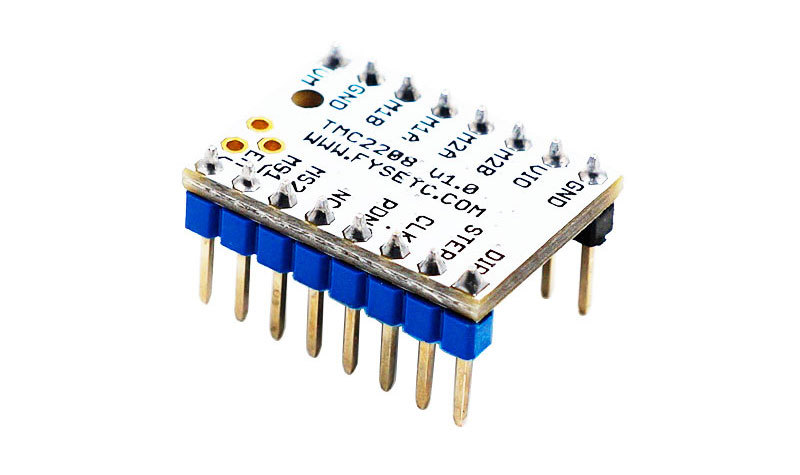
Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

metin, iç mekan, bilgisayar monitörü, çıkış cihazı içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

MOTOR DRIVERS THAT WE TESTED:

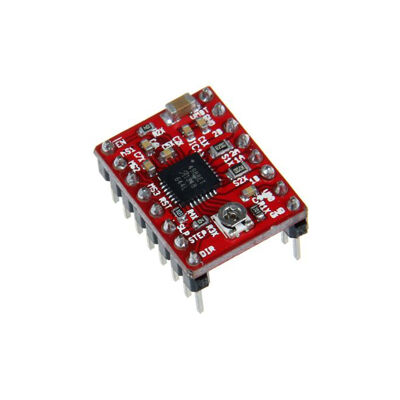
TMC2208



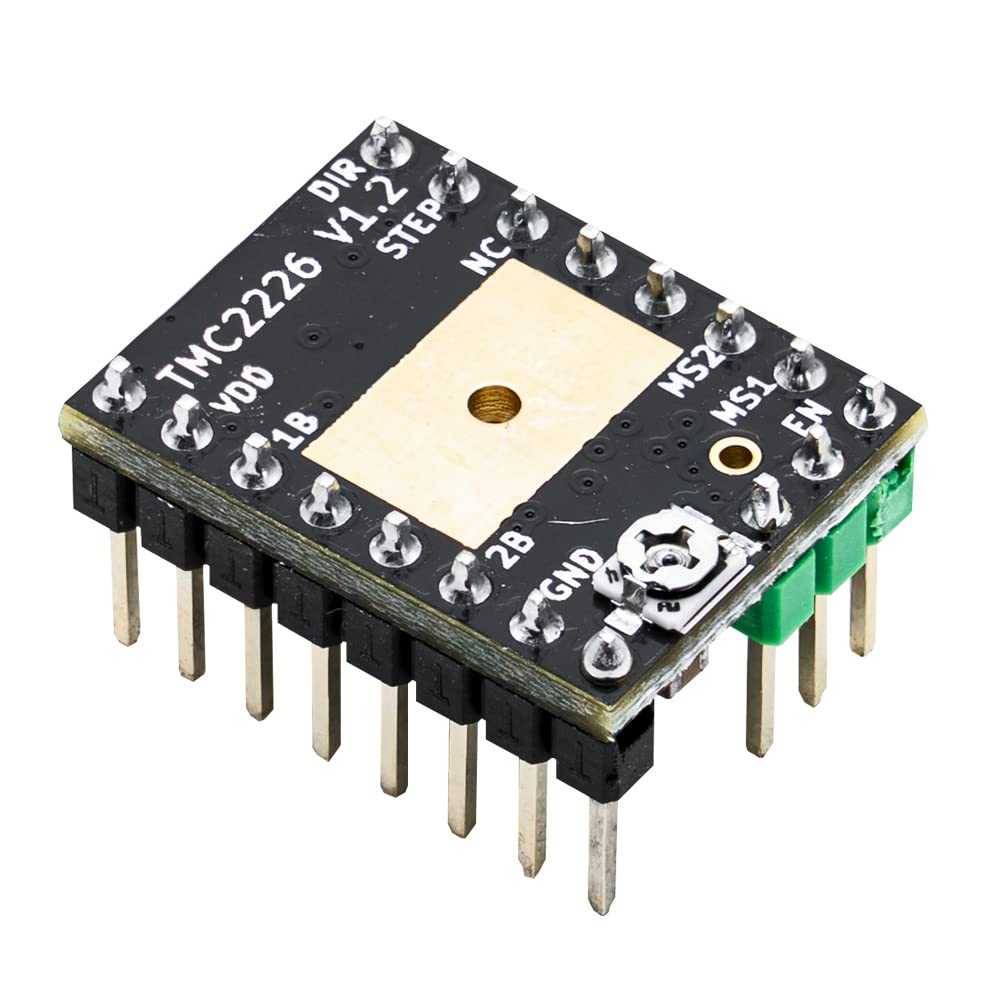
DRV8825



A4988



TMC2226



2 AXIS GIMBALL CONTROL WITH RADIO FREQUENCY:

elektronik donanım, elektronik mühendisliği, elektronik cihaz, bilgisayar içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

metin, elektronik donanım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

NEWPORT GTS30V AND 2 IMS300V

metin, iç mekan, elektronik donanım, ofis malzemesi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

makine, Araba parçası, elektronik donanım, kablo içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

elektronik cihaz, elektronik donanım, küçük alet, iç mekan içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

SMC100CC:

elektronik donanım, kablo, elektronik mühendisliği, iç mekan içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

NEWPORT ESP301:

metin, elektronik donanım, iç mekan, tasarım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

metin, ekran görüntüsü, yazılım, multimedya içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.