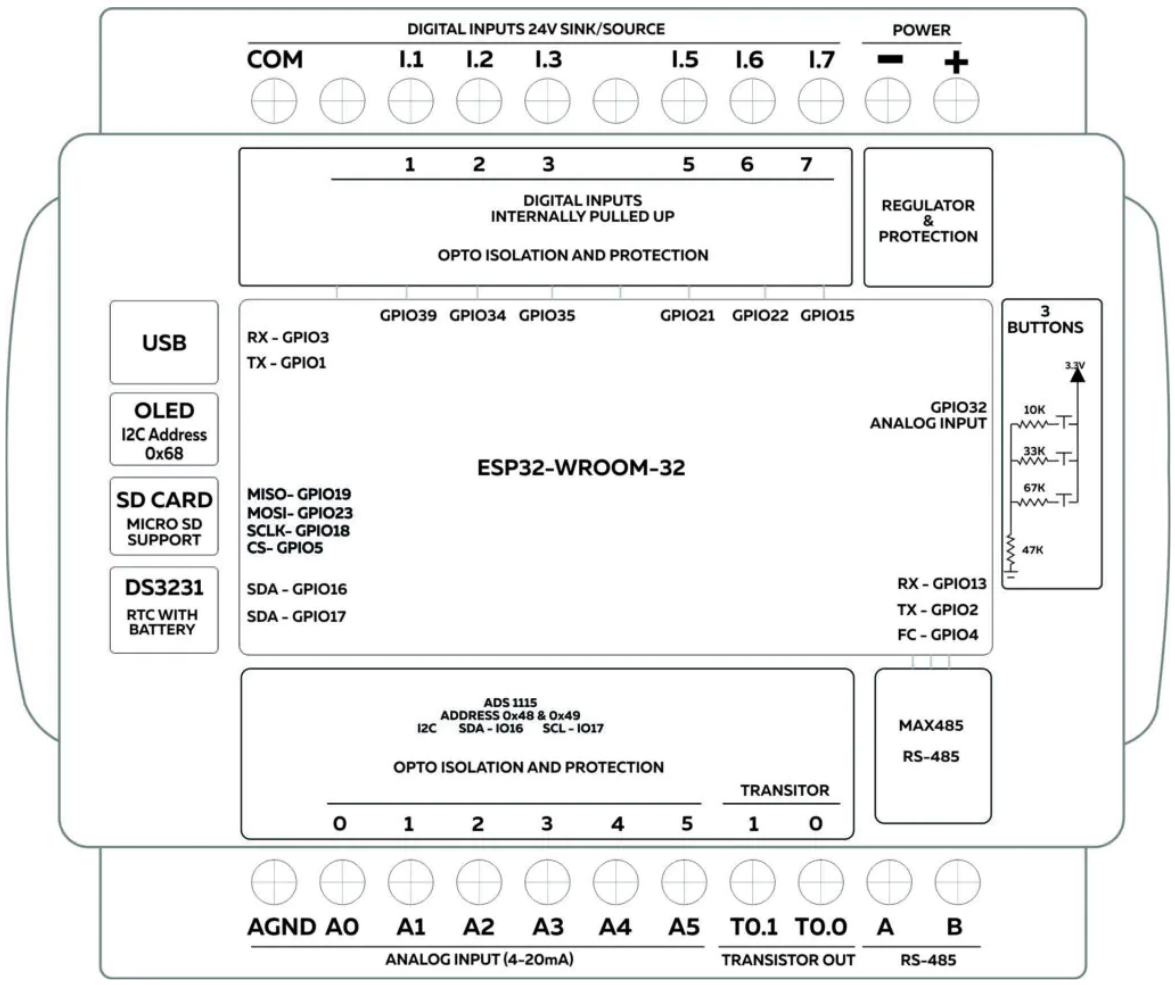
NORVI IIOT-AE04-I – Ficha de Datos

Características del Producto



- Módulo ESP32-WROOM32
- Pantalla OLED de 0.96 integrada
- Soporte para Tarjeta Micro SD
- RTC DS3231 con respaldo de batería
- Botón integrado en el panel frontal
- Entradas Digitales
- Entrada Analógica
- Salidas a Transistor
- Montaje en Carril DIN
- Soporta puertos de expansión

Expansiones Compatibles

- Entrada Digital
- Entrada Analógica
- Salida Transistor

Inicio

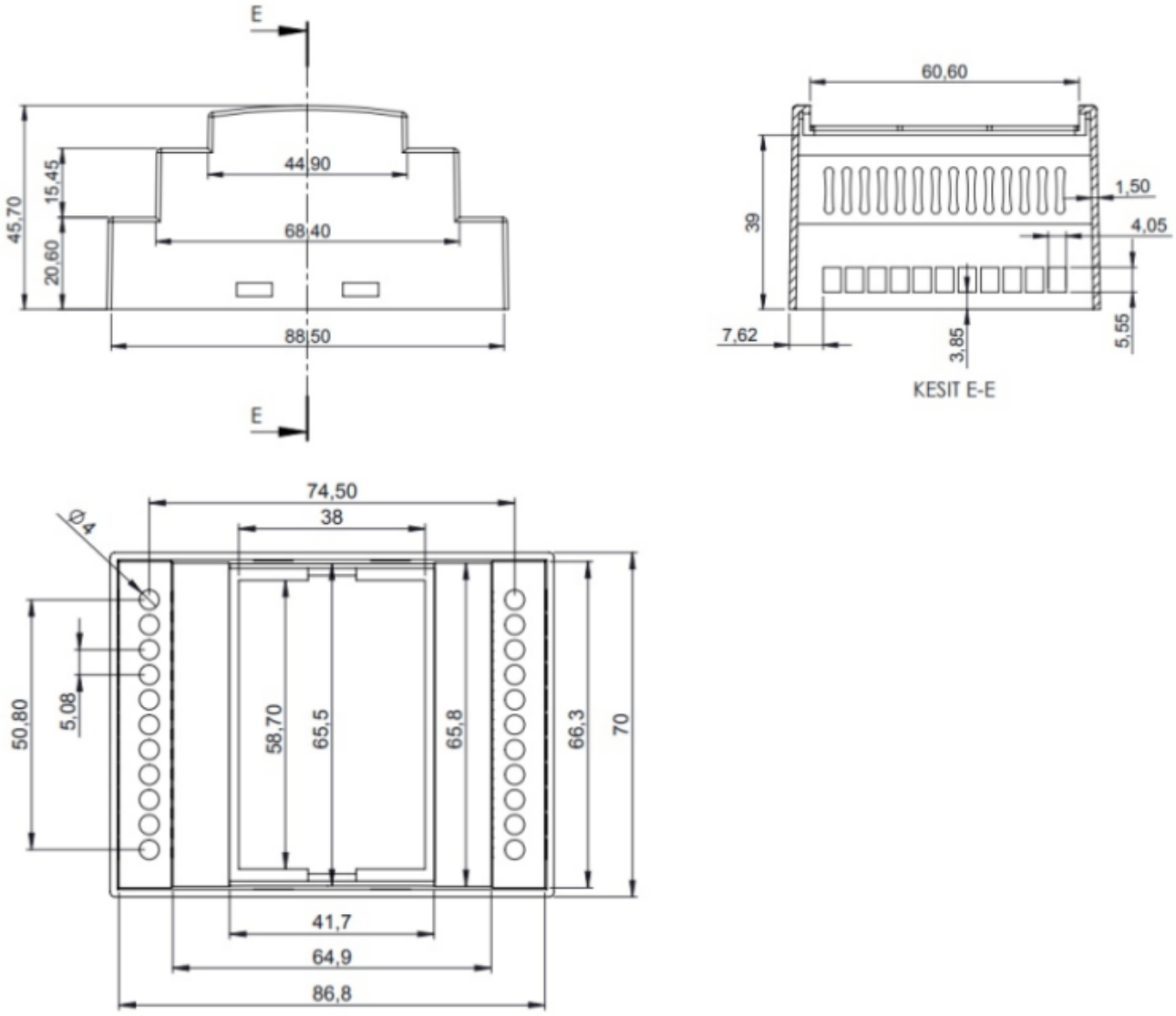
Gama del Producto	NORVI IIOT
Tipo de Producto	Controlador Programable
Certificaciones	EN 61131-2:2007 EN 61010-1:2010+A1:2019 EN IEC 61010-2-201:2018 2014/30/EU- Compatibilidad Electromagnética (EMC) Anexo III, Parte B, Modulo C
Tensión de Alimentación	24V DC
Comunicación	WiFI 2.5GHz / Bluetooth RS-485
Entradas y Salidas	6 x Entradas Digitales 6 x Entradas Analógicas con 4-20 mA 2 x Salidas Transistor
Pantallas e Indicadores Visuales	0.96 Pantalla OLED e indicadores

Complementarios

Código de Producto Unificado	NORVI IIOT-AE04-I
Números de Referencia de los Productos	NORVI IIOT-AE04-I

Características Mecanicas

Carcasa	NORVI 204
Montaje / Método de Instalacion	RIEL DIN / PESTAÑAS DE SUJECIÓN
Tipo de Terminal	TERMINAL DE TORNILLOS
Disposición de Terminales	Arriba y Abajo
Largo	90.50 mm
Alto	56.60 mm
Ancho	60.60 mm



Entorno

Grado de Proteccion IP	IP20
Altitud Operativa	0–2000 metros
Temperatura de Operación	– -10… +85° C (14…185 °F)
Altitud de Almacenaje	0–3000 metros
Resistencia a Golpes	15 gn for 11ms
Resistencia a Descargas Electrostáticas	4 kV en contacto 8 kV en aire
Resistencia a Campos Electromagnéticos	10 V/m (80 MHz …… 1GHz) 3 V/m (1.4 MHz… 2 GHz) 1 V/m (2 MHz, 3 GHz)

Caracteristicas Eléctricas

Dispositivos alimentados por la red eléctrica

Rated Supply Voltage (V)	24V DC
Current Consumption (mA)	400mA
Recommended Power Source	1A, 24V DC

Procesamiento

SOC / MCU	ESP32-WROOM32
Flash Memory	4MB
ROM	448 KB
SRAM	520 KB
PSRAM	NO DISPONIBLE

Periféricos

Soporte para Tarjeta Micro SD

Tipo de Tarjeta	micro SD
Interfaz	SPI
SD CARD CS	GPIO5
MISO	GPIO19
MOSI	GPIO23
SCLK	GPIO18
Detección SD	NO CONECTADO

RTC Interno

Chip RTC	DS3231
Tipo de Batería de Respaldo	CR2032
Interfaz	I2C
Dirección I2C	0x68
SCL Pin	GPIO17
SDA Pin	GPIO16

Botones Integrados

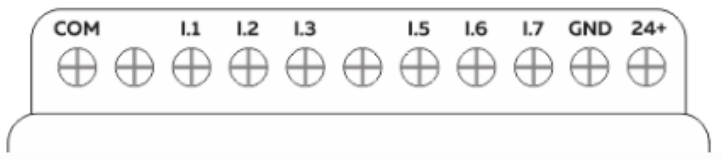
Pin de Botón 1	GPIO32 Nivel de Entrada Analógica 1
Pin de Botón 2	GPIO32 Nivel de Entrada Analógica 2
Pin de Botón 3	GPIO32 Nivel de Entrada Analógica 3

Pantalla OLED

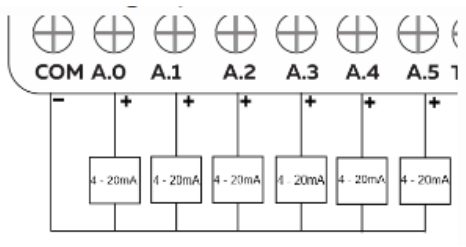
Controlador de Pantalla	SSD1306
Tamaño de Pantalla	0.96 inch
SCL PiN	GPIO17
SDA Pin	GPIO16
RESET Pin	NO CONECTADO

ENTRADAS Y SALIDAS

Entradas Digitales

Numero de Entradas Digitales	6
Polaridad de Entradas Digitales	Sink and Source
Tensión Máxima de Entrada Digital	32V DC
Tensión Mínima de Entrada Digital	18V DC
Frecuencia Máxima de Conmutación	1 kHZ
Disposición de Terminales	Entrada Digital 1 – GPIO39 Entrada Digital 2 – GPIO34 Entrada Digital 3 – GPIO35 Entrada Digital 5 – GPIO21 Entrada Digital 6 – GPIO22 Entrada Digital 7 – GPIO15 

Entradas Analógicas

Numero de Entradas Analógicas	6
Rango de Medición de la Entrada Analógica	4 – 20mA
Conversor analógico-digital (ADC)	ADS1115
Comunicación Analógica a Digital (Analog to Digital Converter "ADC")	I2C
Convertidor analógico-digital (ADC) Dirección	0x48, 0x49
Disposición de Terminales	A0 : Entrada Analógica 0 – ADS1115 – 0x48 – AIN0 A1 : Entrada Analógica 1 – ADS1115 – 0x48 – AIN1 A2 : Entrada Analógica 2 – ADS1115 – 0x48 – AIN2 A3 : Entrada Analógica 3 – ADS1115 – 0x48 – AIN3 A4 : Entrada Analógica 4 – ADS1115 – 0x49 – AIN0 A5 : Entrada Analógica 5 – ADS1115 – 0x49 – AIN1 

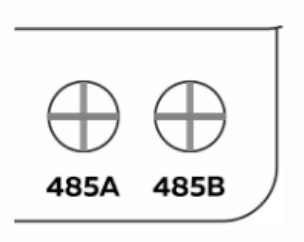
Salidas Transistor

Numero de Salidas Transistor	2
Tipo de Salida Transistor	COLECTOR ABIERTO
Corriente Máxima Sink/Source (mA)	100mA

Tensión Máxima Aplicable	40V DC
Frecuencia Máxima de Conmutación	1 kHz
Disposición de Terminales	T0.0 – GPIO26 T0.1 – GPIO27 

Canales de Comunicación

RS-485 Comunicación

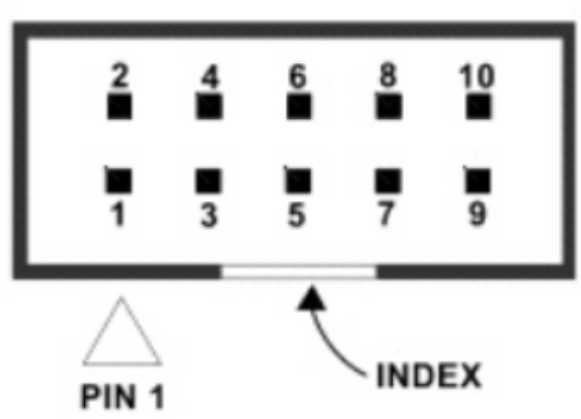
Modo de Comunicación	HALF-DUPLEX
Transceptor	MAX485
Unidad de Carga	1/4
Control de Flujo / Pin de Control de Dirección	GPIO4
Pin TX	GPIO2
Pin RX	GPIO13
Disposición de Terminales	

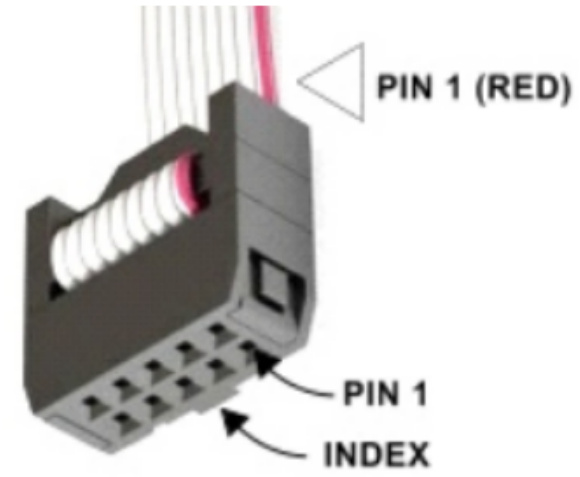
GPIO Map

GPIO	Descripción	Usage
0	emite una señal PWM en el arranque	NRST
1	salida de depuración en el arranque	
2	RS485	TX
3	ALTO (HIGH) en el arranque	
4	RS-485	Control de Flujo (Flow Control)
5	emite una señal PWM en el arranque	
12		
13	RS485	RX
14	emite una señal PWM en el arranque	
15	sólo entrada	Entrada Digital 5
16	SDA	I2C
17	SCL	I2C
18	SCLK	
19	MISO	
21	sólo entrada	Entrada Digital 3
22	sólo entrada	Entrada Digital 4
23	MOSI	
25	Puerto de Expansión	Pin 1
26		Salida Transistor 0.0

27		Salida Transistor 0.1
28		
32	Analog Input	Botones
33		
34	sólo entrada	Entrada Digital 1
35	sólo entrada	Entrada Digital 2
38		
39	sólo entrada	Entrada Digital 0

Puerto de Expansión





PIN	**ESP32 Conexión**
1	IO25
2	TXD0
3	NO CONECTADO
4	RXD0
5	BOOT IO0
6	IO32
7	NO CONECTADO
8	SCL – GPIO17
9	TIERRA
10	SDA – GPIO16

NORVI IIOT-AE04-I – Programa de Pruebas

Introducción

Esta guía está destinada a probar las características y el funcionamiento básico del dispositivo, el NORVI IIOT-AE04-I (modelo actual).

NORVI IIOT-AE04-I programa de pruebas.

Tabla de Instrucciones de la Prueba

Grabe el firmware de prueba antes de probar el dispositivo. Siga las instrucciones dadas en la guía "Guía para Grabar el Firmware de Prueba" para grabar el código binario.

Prueba de componente/ característica	**Test - Prueba**	**Salida Esperada / Salidas Esperadas**
Alimentación	Suministre una alim. de 24V CC.	Pantalla, tarjeta de memoria y RTC
Pantalla, tarjeta de memoria y RTC	Encienda el dispositivo mediante un cable USB o una fuente de alimentación de 24 V CC.	La pantalla comienza con el logotipo de Norvi. Se muestra el modelo del dispositivo. Se muestra el estado del RTC. Se muestra el estado de la tarjeta de memoria. Aparece una última pantalla con el estado de las entradas, salidas y pulsadores. Los indicadores LED del lado de salida se iluminan siguiendo un patrón.
Entradas Digitales	Encienda el dispositivo con una fuente de alimentación de 24 V CC. Conecta los pines GND y COM y suministra los 24V DC a cada entrada digital una a una.	Consulte los resultados esperados de la comprobación de la pantalla más arriba. En el estado de las entradas, el estado de las seis entradas digitales será 1. (Como las entradas tienen pull-up interno.) El estado de la entrada cambia de 1 a 0, y el LED indicador de entrada empieza a iluminarse correspondientemente.
Entradas de Corriente y Salidas a Transistor	Alimente el dispositivo utilizando una fuente de alimentación de 24V CC.	El estado de las seis entradas analógicas será 0. La alternancia del estado de salida (de 0 a 1) se observa en la pantalla para las 2 salidas a transistor, que sigue el patrón de parpadeo del LED indicador de salida. Siempre que estos LEDs están encendidos, significa que el transistor respectivo está encendido.
Entradas de corriente y salidas de transistor (continuación)	Después de encender el aparato, para comprobar el funcionamiento de las seis entradas analógicas (tensión), suministre una corriente entre 4 y 20 mA a cada entrada de corriente. (Consulta este enlace para ver la conexión de los cables). Para comprobar el funcionamiento de los dos transistores, se realiza una prueba de tensión con un multímetro. Para ello, mantén la sonda positiva del multímetro en la patilla +24V del dispositivo. A continuación, toca la sonda negativa con las dos patillas de salida del transistor, una a una, tras un intervalo de 15 segundos.	En la pantalla aparece la tensión detectada por el dispositivo Norvi. (Puede confirmar estos valores de corriente con un multímetro). El multímetro muestra una lectura de 24 V CC siempre que el transistor esté encendido. (El estado del transistor se indica mediante el indicador LED del lado de salida respectivo y el estado de salida en la pantalla).
Pulsadores	Pulse los tres pulsadores, de uno en uno.	El estado analógico de 4 dígitos del pulsador se visualiza correspondientemente en la pantalla. Estado analógico 1 para el botón superior Estado analógico 2 para el botón central Estado analógico 3 para el botón inferior

Comunicación RS-485	Para esta prueba se necesita un conversor USB a RS-485. Conecta los pines A y B RS-485 del dispositivo Norvi con los respectivos pines A y B del conversor USB a RS-485. Conecte el extremo USB del conversor USB a RS-485 al PC. Encienda el dispositivo Norvi con un cable USB. Abra la aplicación Arduino IDE. Seleccione el puerto COM correcto del conversor USB a RS-485 en el IDE de Arduino y abra el Monitor serie. Envía el número '5' en el monitor serie.	En el monitor serie se observa que se imprime la sentencia "485 Success". Esto indica que la operación Tx del RS-485 está funcionando correctamente en el dispositivo Norvi. Una vez recibido el número "5", todos los LED del lado de salida Los indicadores se iluminarán simultáneamente durante unos segundos. Luego, más tarde, continuarán brillando en su patrón anterior. Esto indica que la operación RX del RS-485 está funcionando correctamente en el dispositivo Norvi.

Tabla de pruebas

Programa de Pruebas

```
/*
 * IIOT_AE04_FINAL TEST
 *
 */

#include <SPI.h>
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_ADS1X15.h>
#include "SD.h"
#include "RTClib.h"

#define ANALOG_PIN_0 32

#define INPUT1 39
#define INPUT2 34
#define INPUT3 35
#define INPUT4 21
#define INPUT5 22
#define INPUT6 15

#define OUTPUT1 26
#define OUTPUT2 27

#define RS485_RX 13
#define RS485_TX 2
#define RS485_FC 4

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

#define OLED_RESET     -1 // Reset pin # (or -1 if sharing Arduino reset pin)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

RTC_DS3231 rtc;
char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};

Adafruit_ADS1115 ads1;
```

```
Adafruit_ADS1115 ads2;

int analog_value = 0;

int readSwitch(){
  analog_value = analogRead(ANALOG_PIN_0);
  return analog_value
; //Read analog
}

unsigned long int timer1 = 0;

// ================================================ SETUP ================================================
void setup() {
  Serial.begin(115200);

  pinMode(RS485_FC, OUTPUT);
  digitalWrite(RS485_FC, HIGH);  // RS-485
  Serial.println("Hello");
  Serial1.begin(9600, SERIAL_8N1, RS485_RX, RS485_TX);

  pinMode(OUTPUT1, OUTPUT);
  pinMode(OUTPUT2, OUTPUT);

  pinMode(5, OUTPUT);
  digitalWrite(5,HIGH);

  pinMode(INPUT1, INPUT);
  pinMode(INPUT2, INPUT);
  pinMode(INPUT3, INPUT);
  pinMode(INPUT4, INPUT);
  pinMode(INPUT5, INPUT);
  pinMode(INPUT6, INPUT);

  Wire.begin(16,17);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3D for 128x64
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }
  display.display();

  //int ads1 valu = 0;
  if (!ads1.begin(0x48)) {
    Serial.println("Failed to initialize ADS 1 .");
    while (1);
  }

  if (!ads2.begin(0x49)) {
    Serial.println("Failed to initialize ADS 1 .");
    while (1);
  }

  Wire.begin(16,17);

  RTC_Check();
  delay(1000);
```

```
  SD_CHECK();
  delay(1000);

  adcAttachPin(32);
  digitalWrite(RS485_FC, HIGH);   // RS-485
}

void loop() {

  int16_t adc0, adc1, adc2, adc3;
  Serial.println("------------------------------------------------------------");
  Serial.print(digitalRead(INPUT1));
  Serial.print(digitalRead(INPUT2));
  Serial.print(digitalRead(INPUT3));
  Serial.print(digitalRead(INPUT4));
  Serial.print(digitalRead(INPUT5));
  Serial.print(digitalRead(INPUT6));
  Serial.println("");
  Serial.println("");
  Serial.print("Push button  ");
  Serial.println(readSwitch());
  Serial.println("");

  adc0 = ads1.readADC_SingleEnded(0);
  adc1 = ads1.readADC_SingleEnded(1);
  adc2 = ads1.readADC_SingleEnded(2);
  adc3 = ads1.readADC_SingleEnded(3);

  Serial.println("------------------------------------------------------------");
  Serial.print("AIN1: "); Serial.print(adc0); Serial.println("  ");
  Serial.print("AIN2: "); Serial.print(adc1); Serial.println("  ");
  Serial.print("AIN3: "); Serial.print(adc2); Serial.println("  ");
  Serial.print("AIN4: "); Serial.print(adc3); Serial.println("  ");

  adc0 = ads2.readADC_SingleEnded(0);
  adc1 = ads2.readADC_SingleEnded(1);
  adc2 = ads2.readADC_SingleEnded(2);
  adc3 = ads2.readADC_SingleEnded(3);

  Serial.println("------------------------------------------------------------");
  Serial.print("AIN5: "); Serial.print(adc0); Serial.println("  ");
  Serial.print("AIN6: "); Serial.print(adc1); Serial.println("  ");

  digitalWrite(OUTPUT1, HIGH);
  digitalWrite(OUTPUT2, LOW);
  delay(100);
  digitalWrite(OUTPUT1, LOW);
  digitalWrite(OUTPUT2, HIGH);
  delay(100);
  digitalWrite(OUTPUT1, LOW);
  digitalWrite(OUTPUT2, LOW);
  delay(100);

  Serial.println("------------------------------------------------------------");
  digitalWrite (RS485_FC, HIGH);                    // Make FLOW CONTROL pin HIGH
  delay(100);
```

```
  Serial1.println(F("RS485 01 SUCCESS"));    // Send RS485 SUCCESS serially
  delay(100);                                // Wait for transmission of data
  digitalWrite(RS485_FC, LOW) ;                    // Receiving mode ON

  delay(100);

  while (Serial1.available()) {  // Check if data is available
    char c = Serial1.read();     // Read data from RS485
    Serial.write(c);             // Print data on serial monitor
  }

}
void displayTime(void) {
  DateTime now = rtc.now();

  Serial.print(now.year(), DEC);
  Serial.print('/');
  Serial.print(now.month(), DEC);
  Serial.print('/');
  Serial.print(now.day(), DEC);
  Serial.print(" ");
  Serial.print(daysOfTheWeek[now.dayOfTheWeek()]);

  Serial.print(now.hour(), DEC);
  Serial.print(':');
  Serial.print(now.minute(), DEC);
  Serial.print(':');
  Serial.print(now.second(), DEC);
  Serial.println();
  delay(1000);

}
void RTC_Check(){
  if (! rtc.begin()) {
    Serial.println("Couldn't find RTC");
  }
 else{
 if (rtc.lostPower()) {

    Serial.println("RTC lost power, lets set the time!");
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));

  }

  int a=1;
  while(a<6)
  {
  displayTime();   // printing time function for oled
  a=a+1;
  }
 }
}

void SD_CHECK(){
  uint8_t cardType = SD.cardType();
  //spi.begin(SCK, MISO, MOSI, CS);

    if(SD.begin(5))
```

```
{
 Serial.println("Card Mount: success");
 Serial.print("Card Type: ");

   if(cardType == CARD_MMC){
       Serial.println("MMC");
   } else if(cardType == CARD_SD){
       Serial.println("SDSC");
   } else if(cardType == CARD_SDHC){
       Serial.println("SDHC");
   } else {
       Serial.println("Unknown");
   }

 int cardSize = SD.cardSize() / (1024 * 1024);
 Serial.printf("Card Size: %lluMB\n", cardSize);

 }
   if(!SD.begin(-1))
   {
   Serial.println("NO SD card");
   }

}
```

NORVI IIOT-AE04-I – Guía del Usuario

Programación

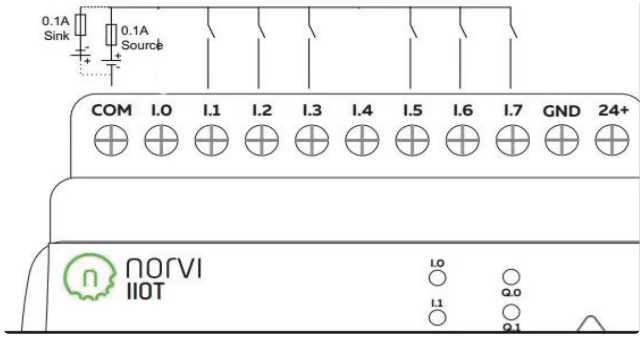
El NORVI IIOT-AE04-I tiene un mini puerto USB para la conexión en serie con el SoC para la programación. Cualquier IDE de programación compatible con ESP32 se puede utilizar para programar el controlador. Siga esta Guía para programar controladores NORVI basados en ESP32 con el IDE Arduino.

SoC: ESP32-WROOM32
Programming Port: USB UART

Entradas Digitales

Cableado de Entradas Digitales

Las entradas digitales del NORVI IIOT-AE04-I pueden configurarse como conexiones tanto Sink como Source. La polaridad inversa de la entrada digital debe suministrarse al terminal común.



Cableado de la entrada digital NORVI IIOT-AE04-I

Programación de Entradas Digitales

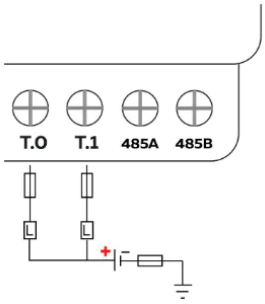
La lectura del GPIO correspondiente del ESP32 da el valor de la Entrada Digital. Cuando las entradas están en estado OFF, el GPIO pasa a HIGH, y cuando la entrada está en estado ON, el GPIO pasa a LOW.

Consulte la tabla de asignación GPIO en la hoja de datos para la entrada digital GPIO

```
#define INPUT1 39
void setup() {
   Serial.begin(115200);
   Serial.println("Device Starting");
   pinMode(INPUT1, INPUT);
}
void loop() {
   Serial.print(digitalRead(INPUT1));
   Serial.println("");
   delay(500);
}
```

Salidas Transistor

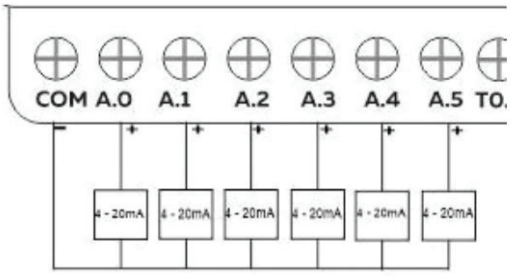
Cableado de Salidas de Transistor



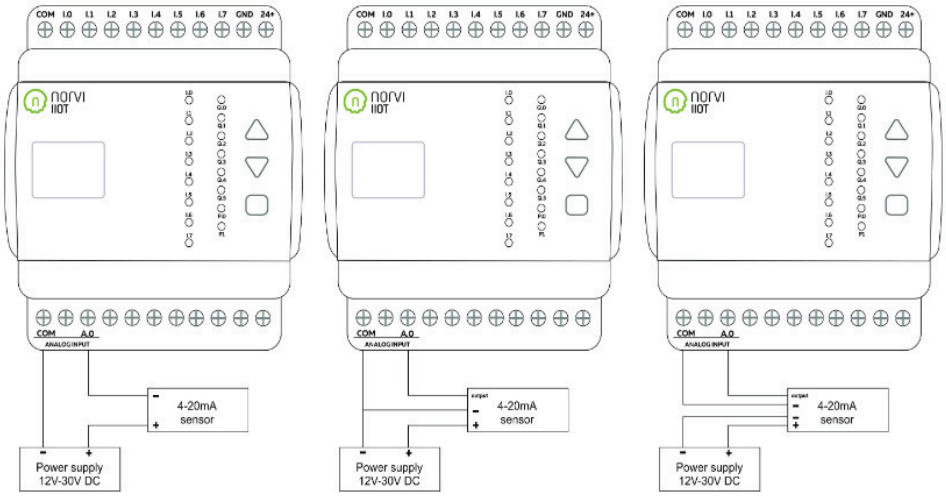
NORVI-IIOT-AE04-I Transistor Output Wiring

Entrada Analógica 0 – 10V

Cableado de Entradas Analógicas

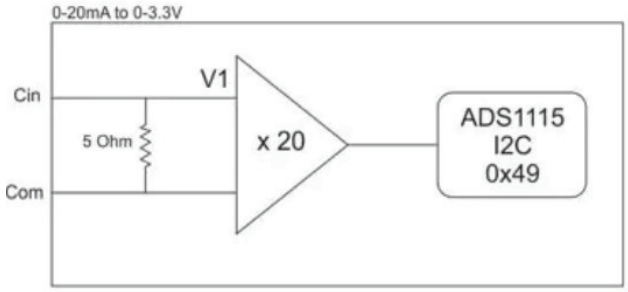


NORVI-IIOT-AE04-I Analog Input Wiring



Lectura de la Entrada Analógica

Leyendo la dirección I2C correspondiente del ADC se obtiene el valor de la Entrada Analógica.



Here's a general formula:

$$Cout = \left(\frac{ADC\ Reading}{32767}\right) \times \left(\frac{Vref}{20 \times 5\Omega}\right)$$

Cout = the output current you want to calculate.

ADC Reading = the serial reading obtained from the ADS115.

32767 = the maximum positive value in the digital output range.

Vref = the Full-scale Input Voltage of the ADS1115. Vref is set to ±4.096 volts.

5Ω = the shunt resistor value.

20 = the OPAMP multiplier.

If the ADC reading is 4478,

$$Cout = \left(\frac{4478}{32767}\right) \times \left(\frac{4096mV}{20 \times 5\Omega}\right)$$

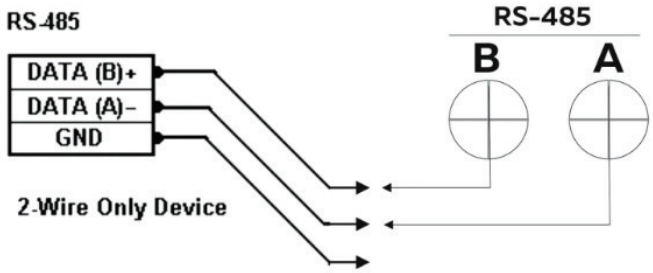
$$Cout = 5.6\text{mA}$$

Programación de Entradas Analógicas

```
#include <Adafruit_ADS1X15.h>
#include <Wire.h>

Adafruit_ADS1115 ads1;
Adafruit_ADS1115 ads2;

void setup() {
   Serial.begin(115200);
   Serial.println("Device Starting");

   Wire.begin(16,17);
   ads1.begin(0x48);
   ads2.begin(0x49);
   ads1.setGain(GAIN_ONE);
   ads2.setGain(GAIN_ONE);
}
void loop() {
  Serial.print("Analog 0 ");
  Serial.println(ads1.readADC_SingleEnded(0));
  delay(10);
  Serial.print("Analog 1 ");
  Serial.println(ads1.readADC_SingleEnded(1));
  delay(10);
  Serial.print("Analog 2 ");
  Serial.println(ads1.readADC_SingleEnded(2));
  delay(10);
  Serial.print("Analog 3 ");
  Serial.println(ads1.readADC_SingleEnded(3));
  delay(10);
  Serial.print("Analog 4 ");
  Serial.println(ads2.readADC_SingleEnded(0));
  delay(10);
  Serial.print("Analog 5 ");
  Serial.println(ads2.readADC_SingleEnded(1));
  delay(10);
  Serial.println("");
  delay(500);
}
```

Comunicación RS-485

Cableado RS-485



Driver	MAX485
UART RX	GPIO13
UART TX	GPIO2
Control de Flujo	GPIO4

Conexiones GPIO de RS-485

Programación RS-485

La conexión RS-485 de la serie NORVI-IIOT-AE04 utiliza un modo semidúplex de transmisor MAX485 con comunicación UART.

```
#define RXD 13
#define TXD 2
#define FC  4

void setup() {
 Serial.begin(9600);
 pinMode(FC, OUTPUT);
 Serial1.begin(9600, SERIAL_8N1,RXD,TXD);
}

void loop() {
  digitalWrite(FC, HIGH);                 // Make FLOW CONTROL pin HIGH
  Serial1.println("RS485 01 SUCCESS");     // Send RS485 SUCCESS serially
  delay(500);                             // Wait for transmission of data
  digitalWrite(FC, LOW);                  // Receiving mode ON

  while (Serial1.available()) {  // Check if data is available
    char c = Serial1.read();     // Read data from RS485
    Serial.write(c);             // Print data on serial monitor
  }
  delay(1000);
}
```

Pantalla OLED Integrada

Controlador de Pantalla	SSD1306
Comunicación	I2C
Dirección del Module	0x3C
Resolución	128 x 64

0,96 Especificaciones de la pantalla OLED

Consulte la tabla de asignación de GPIO en la hoja de datos para el GPIO I2C de la pantalla OLED.

Biblioteca apoyada por la **Adafruit_SSD0306 Library.**

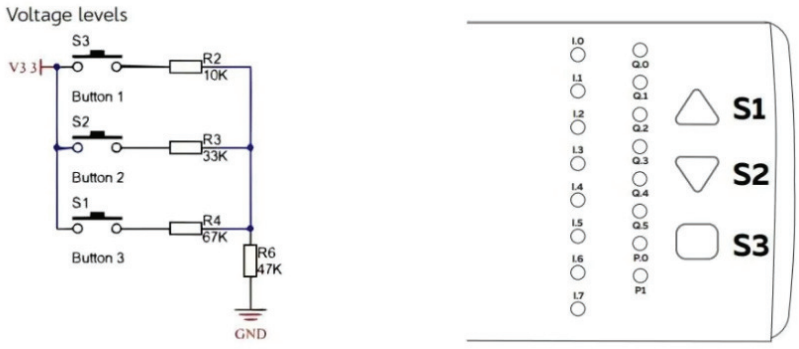
Wire.begin (SDA, SCL) es necesario para inicializar I2C en el pin correctos

Programación Pantalla OLED

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
#define OLED_RESET     -1 // Reset pin # (or -1 if sharing Arduino reset pin)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

void setup() {
  Wire.begin(16,17);
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3C for 128x64
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }
  // Show initial display buffer contents on the screen --
  // the library initializes this with an Adafruit splash screen.
  display.display();
  delay(2000); // Pause for 2 seconds
  }

void loop() {
}
```

Botones Integrados

Modo de Lectura	ADC (Analog to Digital Conversion)
Analogico IO (In/Out)	GPIO32



Botón incorporado esquema interno

Botones de Programación

```
#define buttonPin 32
int  buttonState = 0;

void setup() {
  Serial.begin(9600);
  pinMode(buttonPin,INPUT);
}

void loop() {
  Serial.print("Button: ");
  buttonState = analogRead(buttonPin);
  delay(50);
  Serial.print(analogRead(buttonPin));
  Serial.print("\tAnalog: ");
  delay(1000);
}
```

RTC Interno

Chip RTC	DS3231
Tipo de Batería de Reserva	CR2032
Interfaz	I2C
Dirección I2C	0x68
Pin SCL	GPIO17
Pin SDA	GPIO16

Programación del RTC

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "RTClib.h"

RTC_DS3231 rtc;
char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};

#define SCREEN_WIDTH 128  // OLED display width, in pixels
#define SCREEN_HEIGHT 64  // OLED display height, in pixels
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

void setup()  {
    Serial.begin(9600);
    if (! rtc.begin()) {
       Serial.println("Couldn't find RTC");
       while (1);
    }
    if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C))  {
        Serial.println(F("SSD1306 allocation failed"));
        for(;;); // Don't proceed, loop forever
```

```
    }
    rtc.adjust(DateTime(__DATE__, __TIME__));
    display.display();
    delay(2);
    display.clearDisplay();
    display.setTextColor(WHITE);
    display.setTextSize(2);
    display.setCursor(0,5);
    display.print("  Clock ");
    display.display();
    delay(3000);
}

void loop() {
    DateTime now = rtc.now();
    display.clearDisplay();
    display.setTextSize(2);
    display.setCursor(75,0);
    display.println(now.second(), DEC);

    display.setTextSize(2);
    display.setCursor(25,0);
    display.println(":");

    display.setTextSize(2);
    display.setCursor(65,0);
    display.println(":");
    display.setTextSize(2);
    display.setCursor(40,0);
    display.println(now.minute(), DEC);

    display.setTextSize(2);
    display.setCursor(0,0);
    display.println(now.hour(), DEC);

    display.setTextSize(2);
    display.setCursor(0,20);
    display.println(now.day(), DEC);

    display.setTextSize(2);
    display.setCursor(25,20);
    display.println("-");

    display.setTextSize(2);
    display.setCursor(40,20);
    display.println(now.month(), DEC);

    display.setTextSize(2);
    display.setCursor(55,20);
    display.println("-");

    display.setTextSize(2);
    display.setCursor(70,20);
    display.println(now.year(), DEC);

    display.setTextSize(2);
    display.setCursor(0,40);
    display.print(daysOfTheWeek[now.dayOfTheWeek()]);
    display.display();
 }
```

Compatible con Tarjetas Micro SD

SD CARD CS	GPIO5
MISO	GPIO19
MOSI	GPIO23
SCLK	GPIO18

Programación de Tarjeta SD

```
#include <SD.h>
#define PIN_SPI_CS 5 // The ESP32 pin GPIO5

File myFile;

void setup() {
   Serial.begin(9600);
   if (!SD.begin(PIN_SPI_CS)) {
     Serial.println(F("SD CARD FAILED, OR NOT PRESENT!"));
     while (1); // stop the program
   }
   Serial.println(F("SD CARD INITIALIZED."));
   if (!SD.exists("esp32.txt")) {
     Serial.println(F("esp32.txt doesn't exist. Creating esp32.txt file..."));
     // create a new file by opening a new file and immediately close it
     myFile = SD.open("esp32.txt", FILE_WRITE);
     myFile.close();
   }
   // recheck if file is created or not
   if (SD.exists("esp32.txt"))
   Serial.println(F("esp32.txt exists on SD Card."));
   else
   Serial.println(F("esp32.txt doesn't exist on SD Card."));
}

void loop() {
}
```

REINICIO Y USB