



[Projects \(/circuits/projects/\)](#)

[Contests \(/contest/\)](#)

CAN Protocol - Yes, We Can!

By [Fernando Koyanagi \(/member/Fernando+Koyanagi/\)](#) in [Circuits \(/circuits/\)](#) > [Microcontrollers \(/circuits/microcontrollers/projects/\)](#)

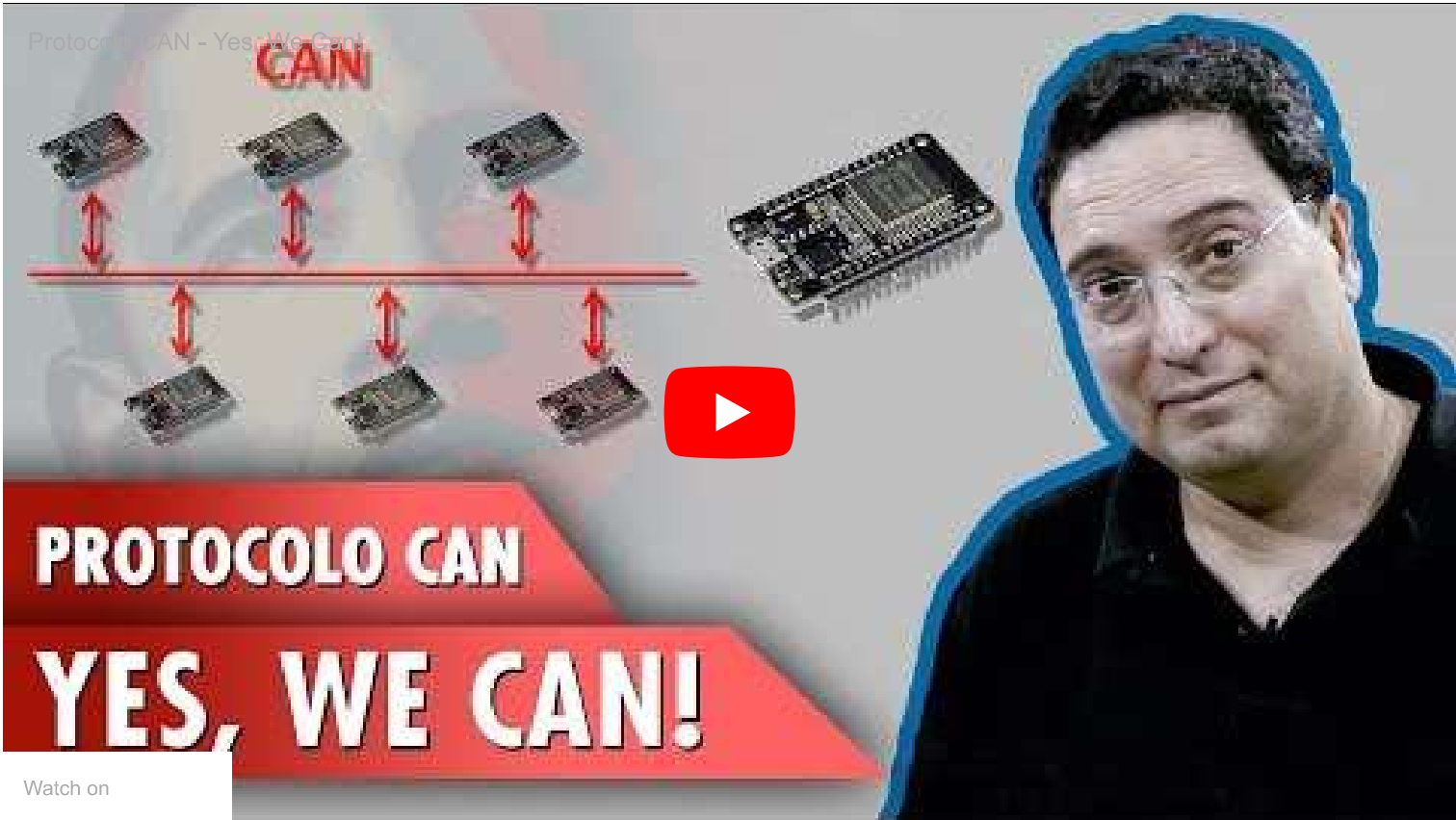
12,299

16

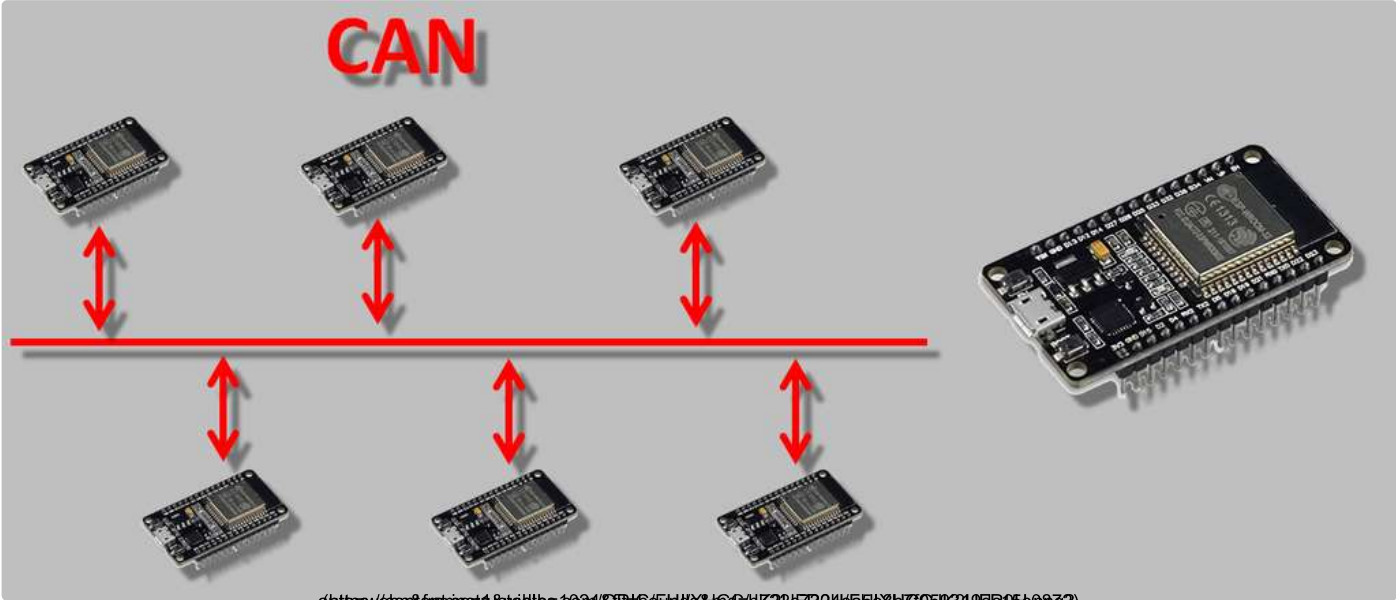
1

[Download](#)

[Favorite](#)



Watch on





By **Fernando Koyanagi**
(/member/Fernando+Koyanagi/)
Visit my Site!
(<http://www.fernandok.com>)

Follow

About: Do you like technology? Follow my channel on Youtube and my Blog. In them I put videos every week of microcontrollers, arduinos, networks, among other subjects. More About Fernando Koyanagi »
(/member/Fernando+Koyanagi/)

More by
the author:



Another subject recently suggested by my YouTube channel's followers was CAN (Controller Area Network) protocol, which is what we'll focus on today. It's important to explain that CAN is a simultaneous serial communication protocol. This means the synchronism between the modules connected to the network is performed in relation to the beginning of each message sent to the bus. We'll start off by introducing the basic concepts of the CAN protocol and perform a simple assembly with two ESP32s.

In our circuit, the ESPs can act as both Master and Slave. You can have multiple microcontrollers transmitting simultaneously, because the CAN deals with the collision of everything automatically. The source code of this project is super simple. Check it out!



Add Tip



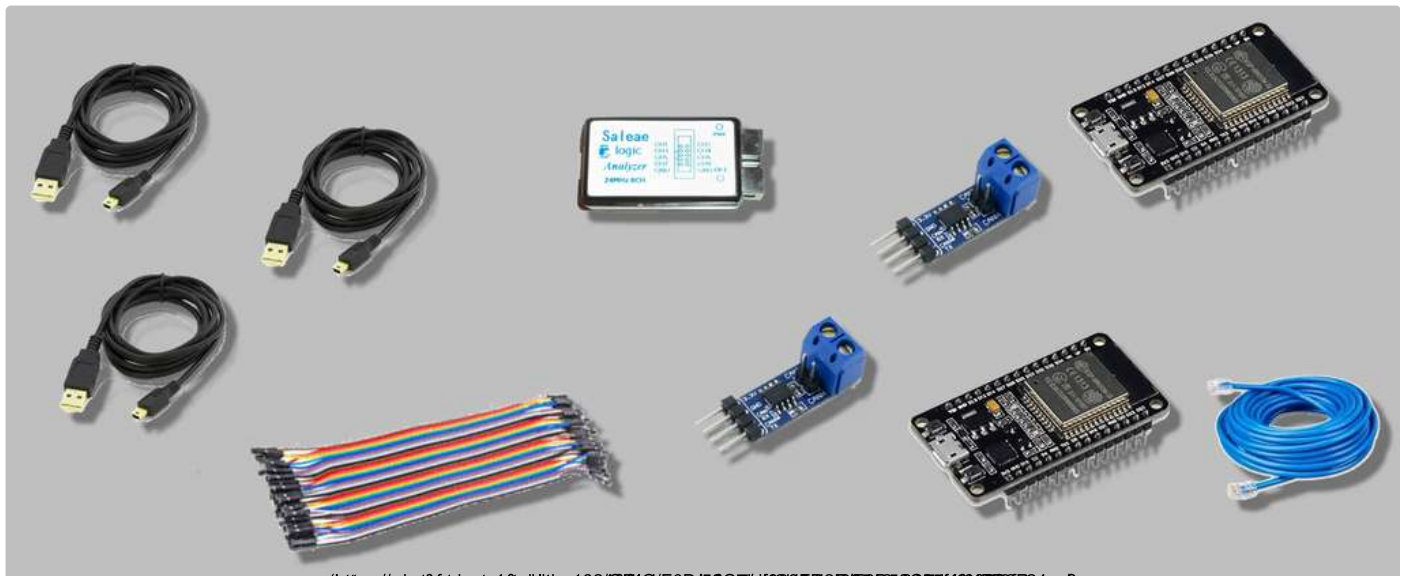
Ask Question



Comment

Download

Step 1: Resources Used



- Two modules of ESP WROOM 32 NodeMcu
- Two CAN transceivers from WaveShare
- Jumpers for connections
- Logical analyzer for capture
- Three USB cables for ESPs and analyzer
- 10 meters of twisted pair to serve as a bus



Add Tip



Ask Question



Comment

Download

Step 2: CAN (Controller Area Network)



- It was developed by Robert Bosch GmbH in the 1980s to serve the automotive industry.
- It has become widespread over the years due to its robustness and flexibility of implementation. It's being used with military equipment, agricultural machinery, industrial and building automation, robotics, and medical equipment.



Add Tip



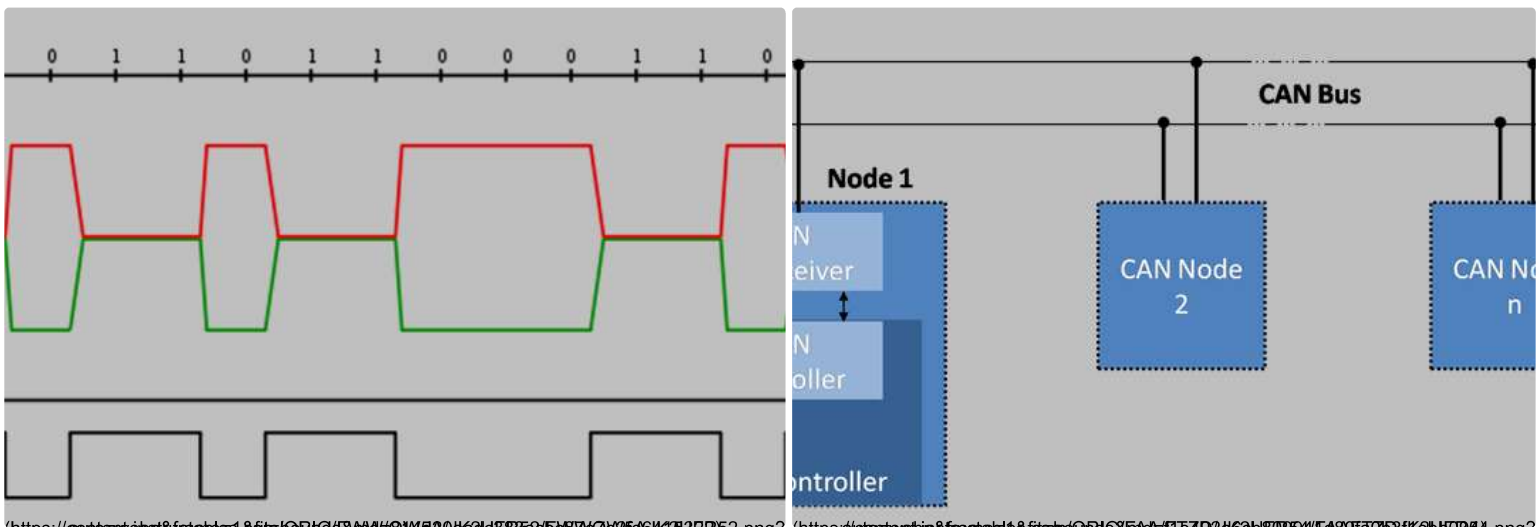
Ask Question



Comment

Download

Step 3: CAN - Features



- Two-wire serial communication
- Maximum of 8 bytes of useful information per frame, with fragmentation possible
- Address directed to the message and not to the node
- Assigning priority to messages and relaying of "on hold" messages
- Effective ability to detect and signal errors
- Multi-master capability (all nodes can request bus access)
- Multicast capability (one message for multiple receivers at the same time)

- Transfer rates of up to 1Mbit / s on a 40-meter bus (reduction of the rate with increase of busbar length)
- Flexibility of configuration and introduction of new nodes (up to 120 nodes per bus)
- Standard hardware, low cost, and good availability
- Regulated protocol: ISO 11898

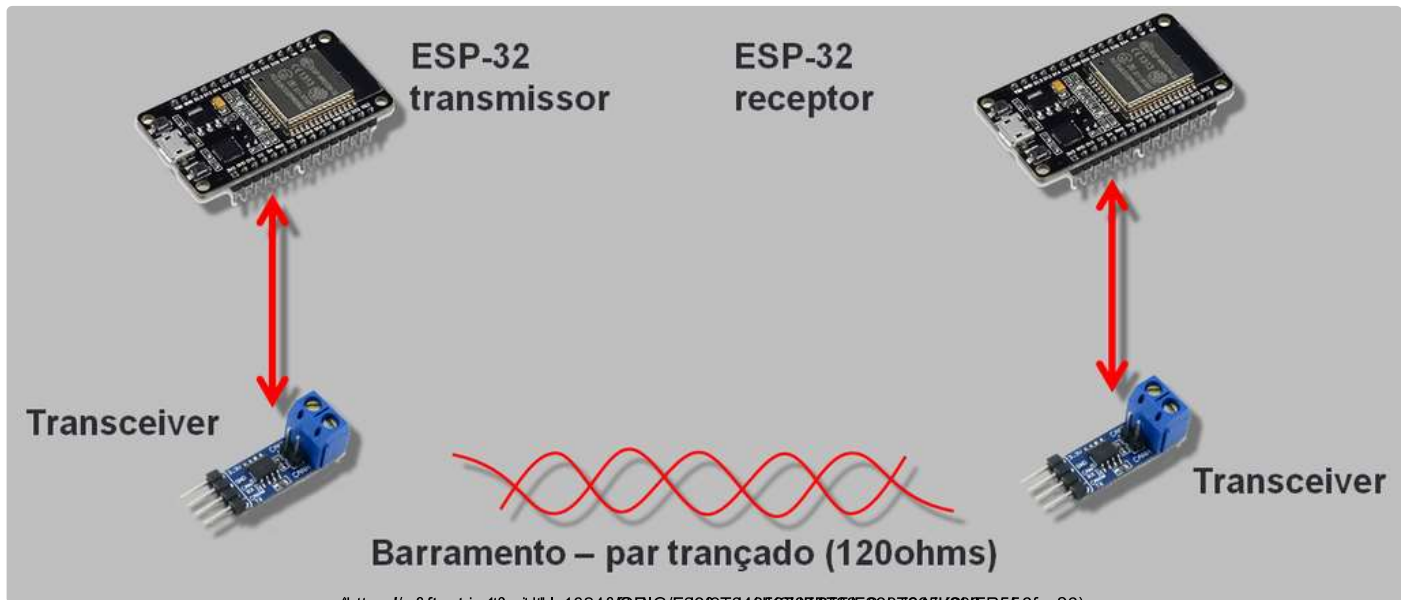
💡 Add Tip

❓ Ask Question

💬 Comment

Download

Step 4: Circuit Used



Here, I have the Transceivers. There is one on each side, and they're connected by a pair of wires. One is responsible for sending and the other for receiving data.

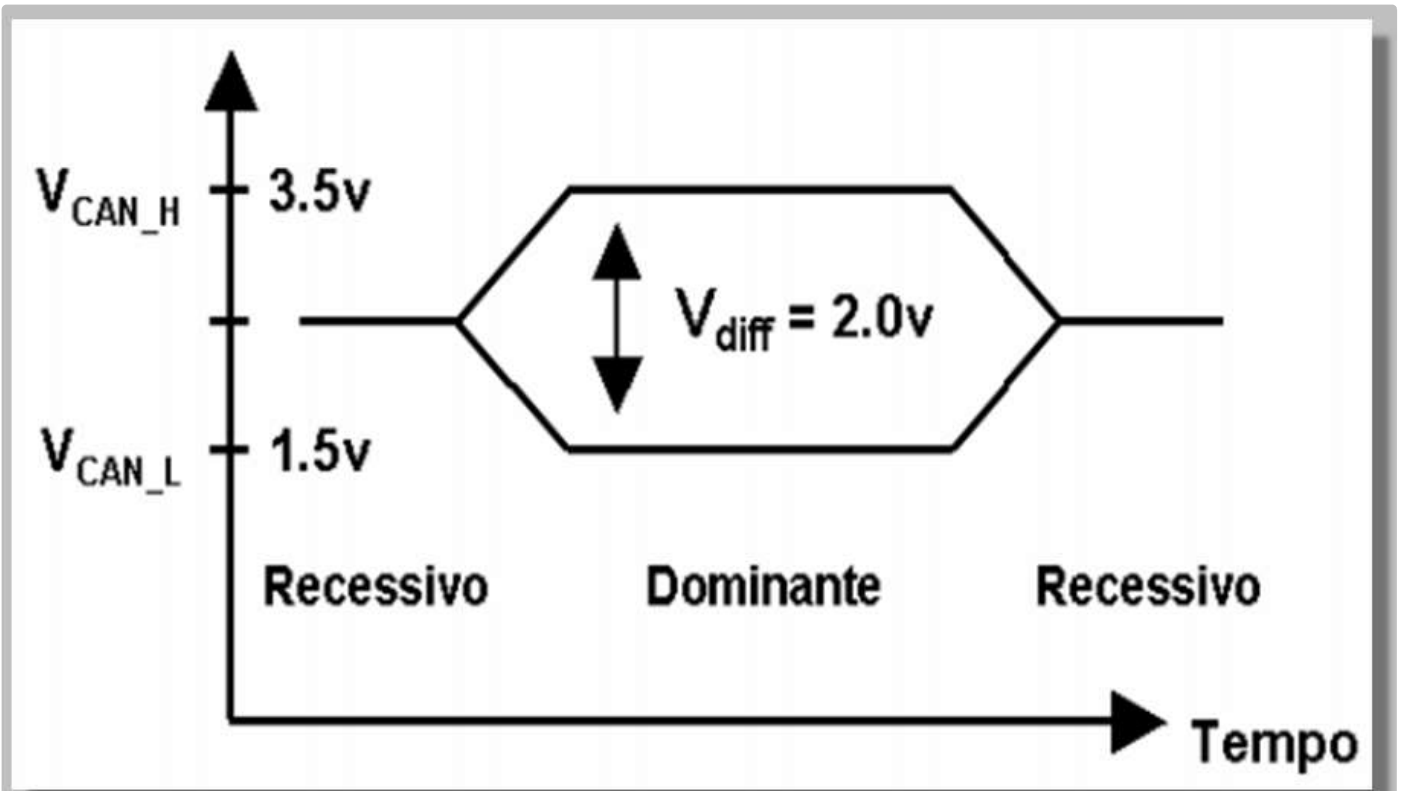
💡 Add Tip

❓ Ask Question

💬 Comment

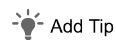
Download

Step 5: Transmission Line Voltages (Differential Detection)



In CAN, the dominant bit is Zero.

Line Differential Detection Reduces Noise Sensitivity (EFI)



Add Tip



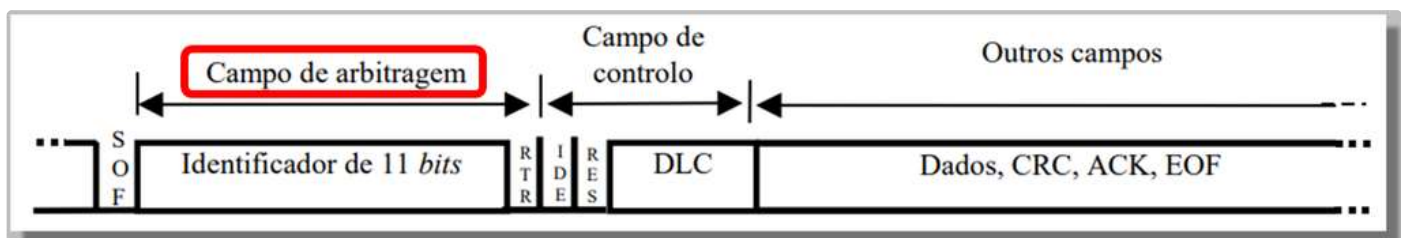
Ask Question



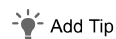
Comment

Download

Step 6: CAN Standards and Frames Format



Standard format with 11-bit identifier



Add Tip



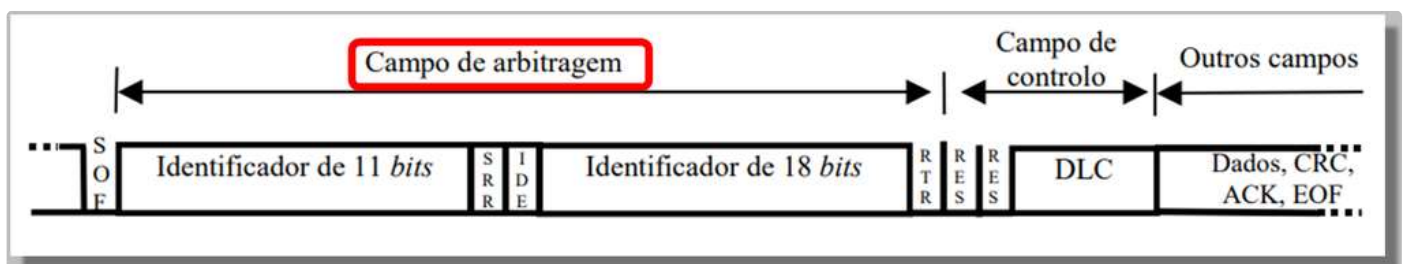
Ask Question



Comment

Download

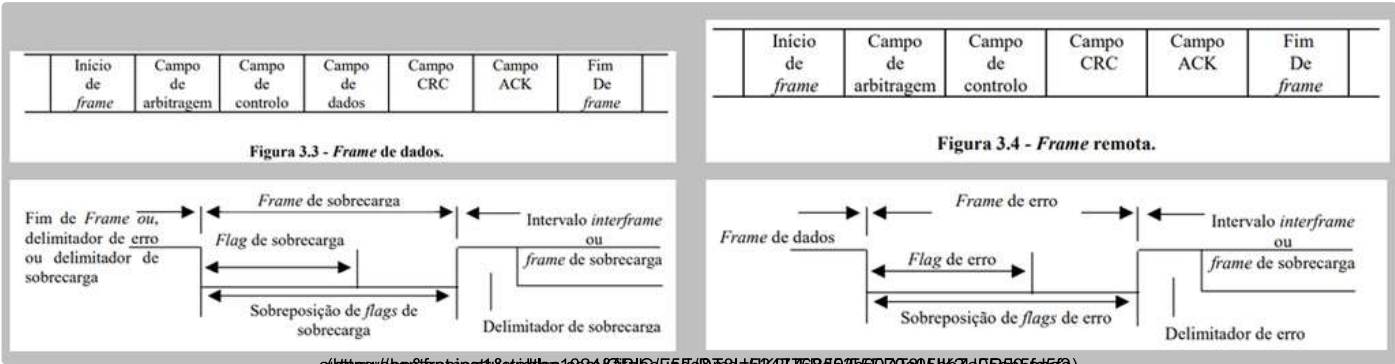
Step 7: CAN Standards and Frames Format



Step 8: CAN Standards and Frames Format

It is important to note that a protocol already calculates the CRC and sends ACK and EOF signals, which are things that are already done by the CAN protocol. This guarantees that the message sent won't arrive in the wrong way. This is because if it gives a problem in the CRC (Redundant Cyclic Check or Redundancy Check), which is the same as an information check digit, it will be identified by the CRC.

Step 9: Four Types of Frames (frames)



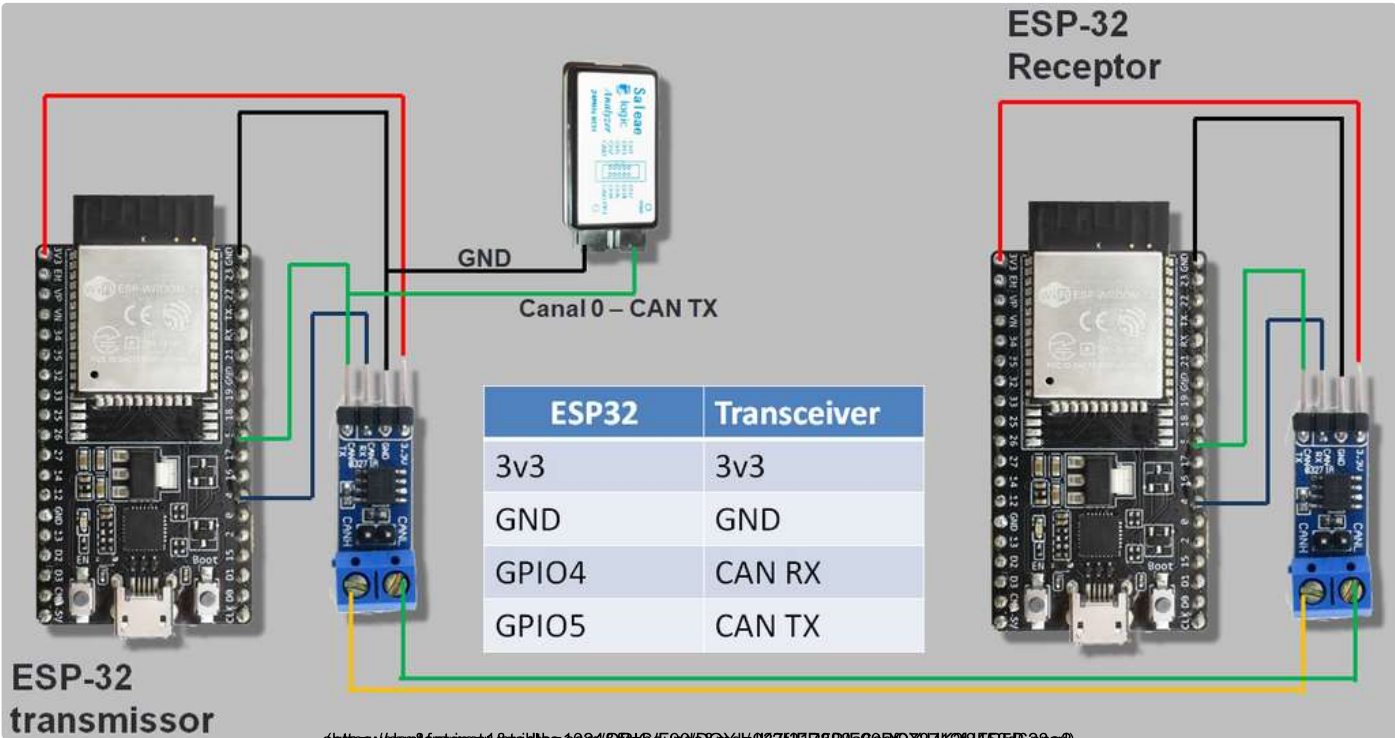
It is important to note that a protocol already calculates the CRC and sends ACK and EOF signals, which are things that are already done by the CAN protocol. This guarantees that the message sent won't arrive in the wrong way. This is because if it gives a problem in the CRC (Redundant Cyclic Check or Redundancy Check), which is the same as an information check digit, it will be identified by the CRC.

Four types of frames (frames)

The transmission and reception of data in the CAN are based on four types of frames. The frame types will be identified by variations in the control bits or even by changes in the frame writing rules for each case.

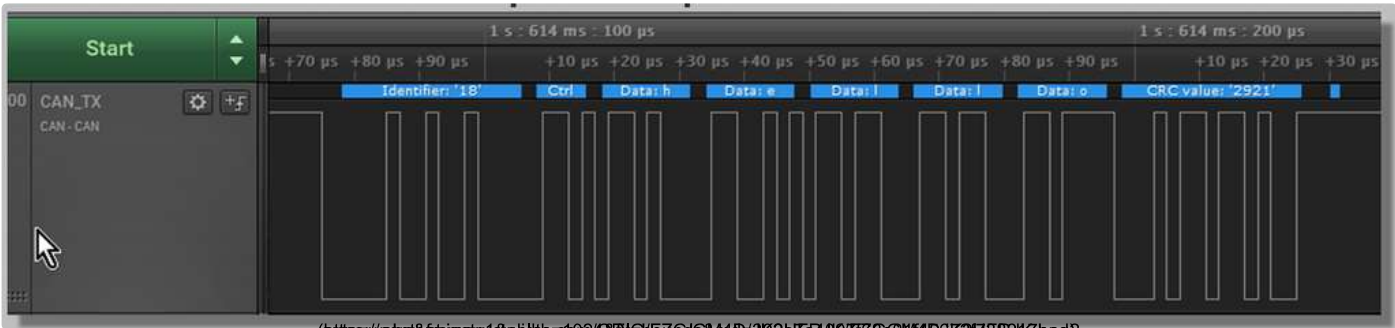
- Data Frame: Contains the transmitter data for the receiver (s)
- Remote Frame: This is a request for data from one of the nodes
- Error Frame: It is a frame sent by any of the nodes when identifying an error in the bus and can be detected by all nodes
- Overload Frame: Serves to delay traffic on the bus due to data overload or delay on one or more nodes.

Step 10: Circuit - Details of Connections



Add Tip Ask Question Comment Download

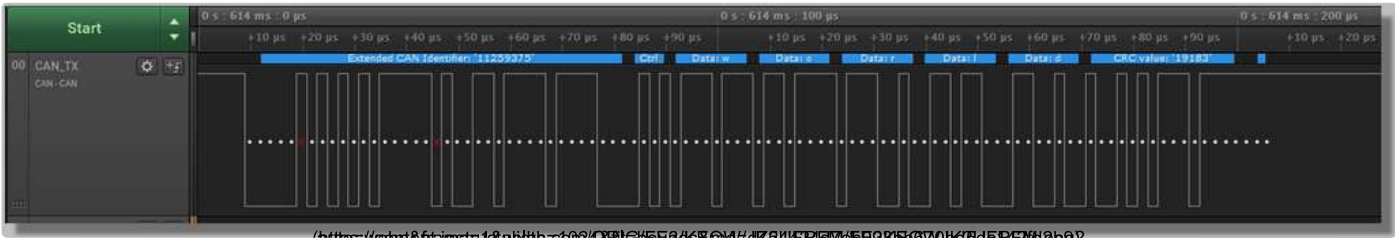
Step 11: Circuit - Data Capture



Wavelengths obtained for standard CAN with 11-bit ID

Add Tip Ask Question Comment Download

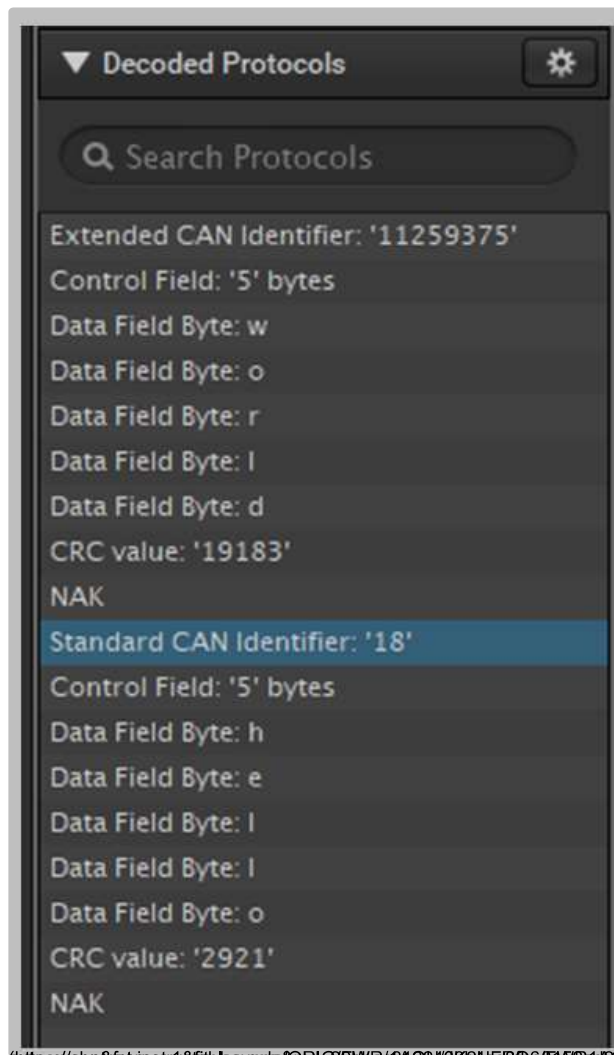
Step 12: Circuit - Data Capture



Wavelengths obtained for extended CAN with 29-bit ID

[Add Tip](#)[Ask Question](#)[Comment](#)[Download](#)

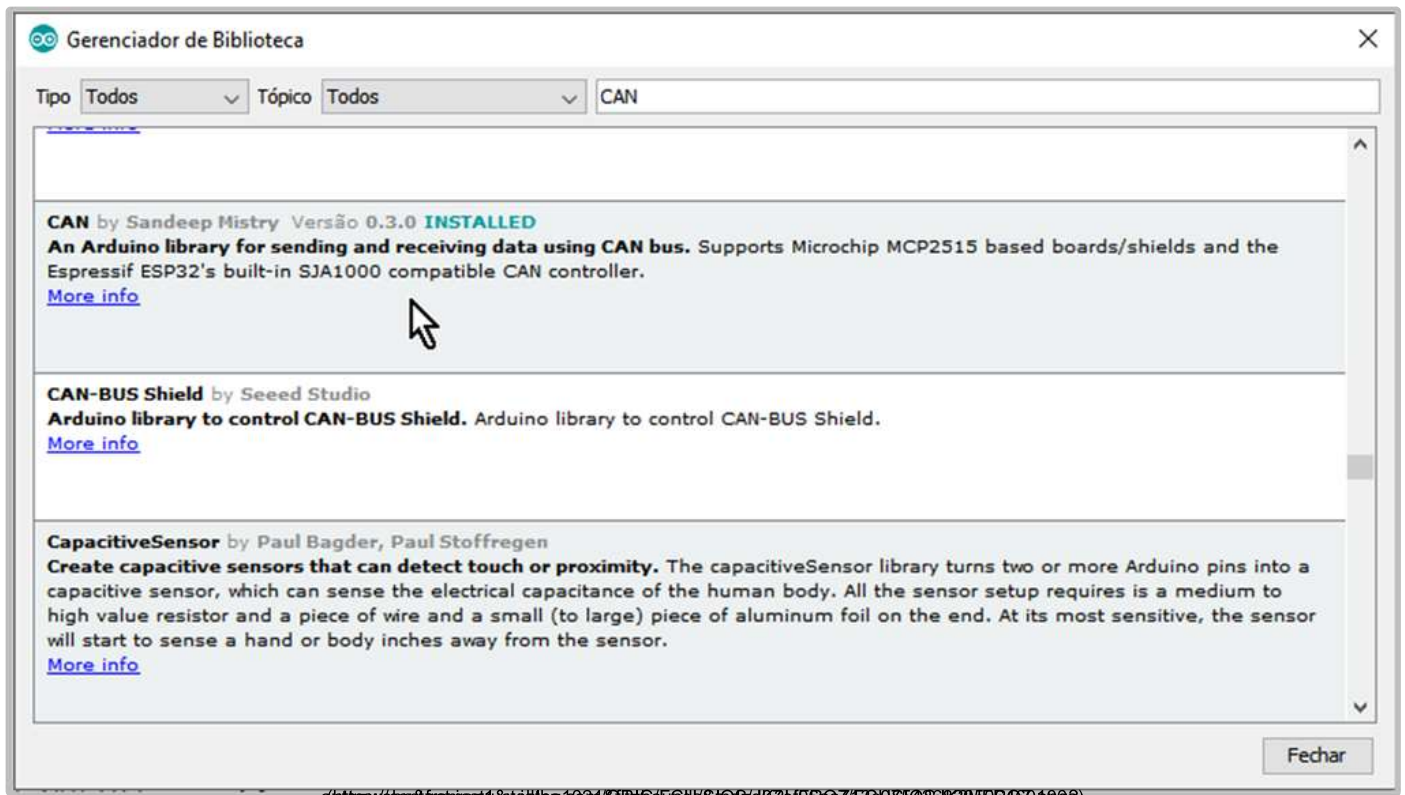
Step 13: Circuit - Data Capture



Data obtained by the logic analyzer

[Add Tip](#)[Ask Question](#)[Comment](#)[Download](#)


Step 14: Arduino Library - CAN



I show here the two options where you can install the CAN Driver Library

Arduino IDE Library Manager

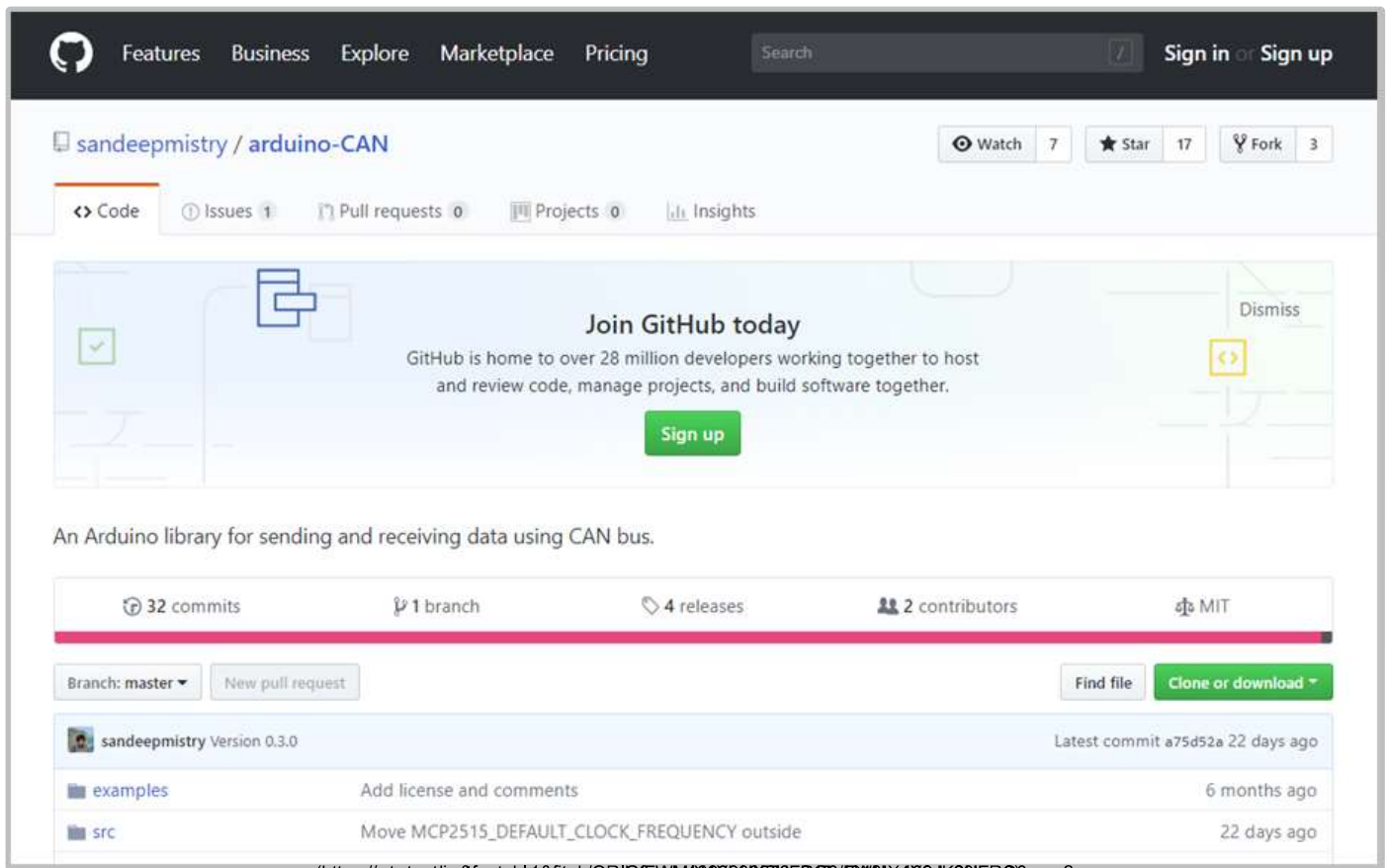
 Add Tip

 Ask Question





 Comment

Download

Step 15: Github



<https://github.com/sandeepmistry/arduino-CAN>

 Add Tip  Ask Question  Comment  Download

Step 16: Transmitter Source Code

Source Code: Includes and Setup ()





We'll include the CAN library, start the serial for debugging, and start the CAN bus at 500 kbps.

```
#include <CAN.h> // Inclui a biblioteca CAN

void setup() {
  Serial.begin(9600); // inicia a serial para debug
  while (!Serial);

  Serial.println("Transmissor CAN");

  // Inicia o barramento CAN a 500 kbps
  if (!CAN.begin(500E3)) {
    Serial.println("Falha ao iniciar o controlador CAN"); // caso não seja possível iniciar o controlador
    while (1);
  }
}
```

 Add Tip  Ask Question  Comment  Download

Step 17: Source Code: Loop (), Sending a Standard CAN 2.0 Packet

Using the standard CAN 2.0, we send a package. The 11-bit ID identifies the message. The data block must have up to 8 bytes. It starts the packet with ID 18 in hexadecimal. It packs 5 bytes and closes the function.

```
void loop() {
  // Usando o CAN 2.0 padrão
  //Envia um pacote: o id tem 11 bits e identifica a mensagem (prioridade, evento)
  //o bloco de dados deve possuir até 8 bytes
  Serial.println("Enviando pacote...");

  CAN.beginPacket(0x12); //id 18 em hexadecimal
  CAN.write('h'); //1ª byte
  CAN.write('e'); //2ª byte
  CAN.write('l'); //3ª byte
  CAN.write('l'); //4ª byte
  CAN.write('o'); //5ª byte
  CAN.endPacket(); //encerra o pacote para envio

  Serial.println("Enviado.");

  delay(1000);
}
```

[Add Tip](#)[Ask Question](#)[Comment](#)[Download](#)

Step 18: Source Code: Loop (), Sending an Extended CAN 2.0 Package

In this step, the ID has 29 bits. It starts sending 24 bits of ID and, once more, packs 5 bytes and quits.

```
//Usando CAN 2.0 Estendido
//Envia um pacote: o id tem 29 bits e identifica a mensagem (prioridade, evento)
//o bloco de dados deve possuir até 8 bytes

Serial.println("Enviando pacote estendido...");

CAN.beginExtendedPacket(0xabcdef); //id 11259375 decimal ( abcdef em hexa) = 24 bits preenchidos até aqui
CAN.write('w'); //1ª byte
CAN.write('o'); //2ª byte
CAN.write('r'); //3ª byte
CAN.write('l'); //4ª byte
CAN.write('d'); //5ª byte
CAN.endPacket(); //encerra o pacote para envio

Serial.println("Enviado.");

delay(1000);
}
```

[Add Tip](#)[Ask Question](#)[Comment](#)[Download](#)

Step 19: Receiver Source Code

Source Code: Includes and Setup ()

Again, we will include the CAN library, start the serial to debug, and start the CAN bus at 500 kbps. If an error occurs, this error will be printed.

```
#include <CAN.h> //Inclui a biblioteca CAN

void setup() {
  Serial.begin(9600); //inicia a serial para debug
  while (!Serial);

  Serial.println("Receptor CAN");

  // Inicia o barramento CAN a 500 kbps
  if (!CAN.begin(500E3)) {
    Serial.println("Falha ao iniciar o controlador CAN"); //caso não seja possível iniciar o controlador
    while (1);
  }
}
```


[Add Tip](#)

[Ask Question](#)

[Comment](#)
[Download](#)

Step 20: Source Code: Loop (), Getting the Package and Checking the Format

We tried to check the size of the packet received. The CAN.parsePacket () method shows me the size of this package. So if we have a package, we'll check whether it is extended or not.

```
void loop() {
  // Tenta verificar o tamanho do pacote recebido
  int packetSize = CAN.parsePacket();

  if (packetSize) {
    // Se temos um pacote
    Serial.println("Recebido pacote. ");

    if (CAN.packetExtended()) { //verifica se o pacote é estendido
      Serial.println("Estendido");
    }
  }
}
```


[Add Tip](#)

[Ask Question](#)

[Comment](#)
[Download](#)

Step 21: Source: Loop (), Checks to See If It Is a Remote Package

Here, we check if the received packet is a data request. In this case, there is no data.

```
if (CAN.packetRtr()) {
  //Verifica se o pacote é um pacote remoto (Requisição de dados), neste caso não há dados
  Serial.print("RTR ");
}
```


[Add Tip](#)

[Ask Question](#)

[Comment](#)
[Download](#)

Step 22: Source Code: Loop (), Data Length Requested or Received

If the received packet is a request, we indicate the requested length. We then obtain the Data Length Code (DLC), which indicates the length of the data. Finally, we indicate the length received.

```
Serial.print("Pacote com id 0x");
Serial.print(CAN.packetId(), HEX);

if (CAN.packetRtr()) { //se o pacote recebido é de requisição, indicamos o comprimento
    Serial.print(" e requisitou o comprimento ");
    Serial.println(CAN.packetDlc()); //obtem o DLC (Data Length Code, que indica o comprimento dos dados)
} else {
    Serial.print(" e comprimento "); // aqui somente indica o comprimento recebido
    Serial.println(packetSize);
}
```



Add Tip



Ask Question



Comment

Download

Step 23: Source Code: Loop (), If Data Is Received, It Then Prints

We print (on the serial monitor) the data, but only if the received packet is not a request.

```
//Imprime os dados somente se o pacote recebido não foi de requisição
while (CAN.available()) {
    Serial.print((char)CAN.read());
}
Serial.println();
}

Serial.println();
}
}
```



Add Tip



Ask Question



Comment

Download

Step 24: Download the Files

[PDF \(http://74.117.156.195/verify.php?arquivo=IntroducaoCANcomESP32.pdf\)](http://74.117.156.195/verify.php?arquivo=IntroducaoCANcomESP32.pdf).

[INO \(http://74.117.156.195/verify.php?arquivo=ProtocoloCAN.rar\)](http://74.117.156.195/verify.php?arquivo=ProtocoloCAN.rar).



Add Tip



Ask Question



Comment

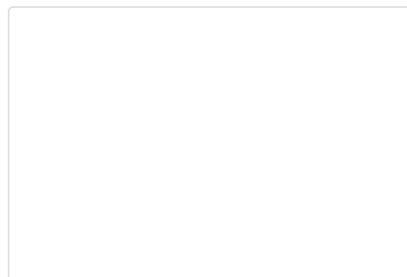
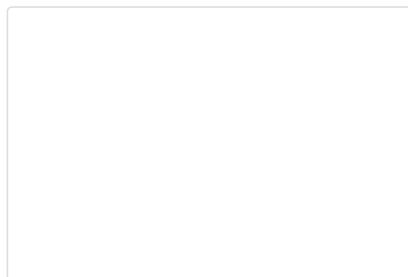
Download

Be the First to Share

Did you make this project? Share it with us!

I Made It!

Recommendations



(/Build-a-UV-Level-Monitoring-Budgie-Using-IoT-and-W/)

Build a UV Level Monitoring Budgie - Using IoT and Weather Data APIs (/Build-a-UV-Level-Monitoring-Budgie-Using-IoT-



(/)

47 4.1K


(/DIY-Weather-Station-With-ESP32/)


DIY Weather Station With ESP32 (/DIY-Weather-Station-With-ESP32/) by Giovanni Aggiustatutto


598 51K


(/contest/plywood2022/)

(/contest/pumpkin/)



Add Tip

Ask Question

Post Comment

We have a **be nice** policy.
Please be positive and constructive.

Add Images

Post

Comments

(/member/Devi+Boda/) Devi Boda (/member/Devi+Boda/) Question 2 years ago on Introduction

Answer

▲ Upvote

I followed the same steps but I am not receiving any data at receiver esp32. The serial monitor is showing blank. Please help me to resolve this issue.

Post Comment

Categories



Circuits
(/circuits/)



Craft
(/craft/)



Workshop
(/workshop/)



Cooking
(/cooking/)

About Us

Who We Are
(/about/)

Why Publish?
(/create/)


Resources


Sitemap (/sitemap/)

Help (/how-to-write-a-great-instructable/)

Contact (/contact/)

 Living
(/living/)

 Teachers
(/teachers/)

 Outside
(/outside/)

Find Us

[\(https://www.instagram.com/instructables/\)](https://www.instagram.com/instructables/) [\(https://www.pinterest.com/instructables\)](https://www.pinterest.com/instructables) [\(https://www.facebook.com/instructables\)](https://www.facebook.com/instructables) [\(https://www.twitter.com/instructables\)](https://www.twitter.com/instructables)

© 2022 Autodesk, Inc. [Terms of Service \(http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=21959721\)](http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=21959721)
[Privacy Statement \(http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=21292079\)](http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=21292079)
[Privacy settings](#)
[Legal Notices & Trademarks \(http://usa.autodesk.com/legal-notices-trademarks/\)](http://usa.autodesk.com/legal-notices-trademarks/)

[\(http://www.autodesk.com\)](http://www.autodesk.com)