# **[Demo 31: How to use Arduino ESP32 CAN interface](http://www.iotsharing.com/2017/09/how-to-use-arduino-esp32-can-interface.html)**

**1. Introduction**  
- CAN stands for Controller Area Network (CAN bus). This protocol is very popular in automotive domain. In order to understand more about history, benefits, characteristics, message format of CAN, you can refer:  
<http://www.ni.com/white-paper/2732/en/>  
<http://www.ti.com/lit/an/sloa101b/sloa101b.pdf>  
- As you knew in [Introduction to ESP32](http://www.iotsharing.com/2017/05/introduction-to-esp32.html), ESP32 also supports CAN interface. So I am going to make a demo for this with Arduino.  
- In this demo, 2 ESP32 modules will be used: first module will send the string "hellocan" to second module. The second module will convert the string to upper case and respond it back to first module and first module will show the result to Terminal.  
**2. Hardware**  
- ESP32 only supplies CAN controller. So you need CAN transceiver for this demo. I bought 2 CAN trasceivers [here](https://vi.aliexpress.com/item/SN65HVD230-CAN-bus-transceiver-communication-module-for-arduino/32687564210.html?spm=a2g14.search0303.3.28.TRuG6q).

[](https://1.bp.blogspot.com/-nuM3-7ENLbE/WaqHAVw-vjI/AAAAAAAAEPk/HsVOT7jnwdkSv9iFcQOgCIfjSmYJ3xuQQCEwYBhgL/s1600/esp32_CAN_1.jpg)

**Figure: ESP32 CAn transceiver**

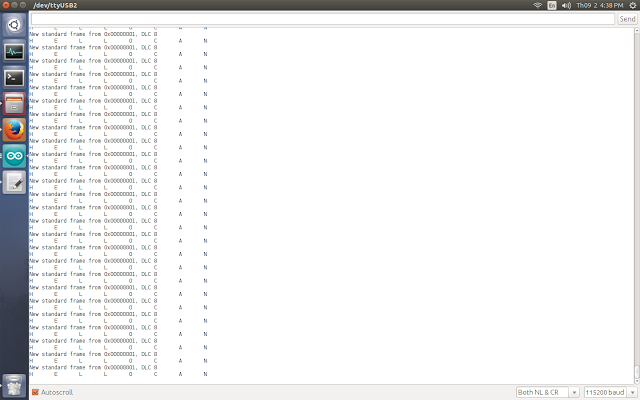
- ESP32 GPIO5 will act as CAN\_Tx.  
- ESP32 GPIO4 will act as CAN\_Rx.  
- So you can connect pins belows (ESP32\_X: module ESP32 X, CAN\_X: module CAN X, where X is 1 or 2 since we have 2 modules):  
**ESP32\_1 IO5 - CAN\_1 CTX**  
**ESP32\_1 IO4 - CAN\_1 CRX**  
**CAN\_1 CANH - CAN\_2 CANH**  
**CAN\_1 CANL - CAN\_2 CANL**  
**ESP32\_2 IO5 - CAN\_2 CTX**  
**ESP32\_2 IO4 - CAN\_2 CRX**  
**3. Software**  
- In order to make this demo, I used CAN driver which is made by [Thomas Barth (Thanks Thomas :))](https://github.com/ThomasBarth/ESP32-CAN-Driver/tree/master/components/can)  
- Download the CAN library for ESP32 [here](https://github.com/nhatuan84/arduino-esp32-can-demo). Unzip it and copy to **"Arduino/libraries"** folder.  
- Source code for first ESP32 module: receiving string, converting to upper case and respond back:

|  |
| --- |
| #include <ESP32CAN.h>  #include <CAN\_config.h>  /\* the variable name CAN\_cfg is fixed, do not change \*/  CAN\_device\_t CAN\_cfg;  void setup() {  Serial.begin(115200);  Serial.println("iotsharing.com CAN demo");  /\* set CAN pins and baudrate \*/  CAN\_cfg.speed=CAN\_SPEED\_1000KBPS;  CAN\_cfg.tx\_pin\_id = GPIO\_NUM\_5;  CAN\_cfg.rx\_pin\_id = GPIO\_NUM\_4;  /\* create a queue for CAN receiving \*/  CAN\_cfg.rx\_queue = xQueueCreate(10,sizeof(CAN\_frame\_t));  //initialize CAN Module  ESP32Can.CANInit();  }  void loop() {  CAN\_frame\_t rx\_frame;  //receive next CAN frame from queue  if(xQueueReceive(CAN\_cfg.rx\_queue,&rx\_frame, 3\*portTICK\_PERIOD\_MS)==pdTRUE){  //do stuff!  if(rx\_frame.FIR.B.FF==CAN\_frame\_std)  printf("New standard frame");  else  printf("New extended frame");  if(rx\_frame.FIR.B.RTR==CAN\_RTR)  printf(" RTR from 0x%08x, DLC %d\r\n",rx\_frame.MsgID, rx\_frame.FIR.B.DLC);  else{  printf(" from 0x%08x, DLC %d\n",rx\_frame.MsgID, rx\_frame.FIR.B.DLC);  /\* convert to upper case and respond to sender \*/  for(int i = 0; i < 8; i++){  if(rx\_frame.data.u8[i] >= 'a' && rx\_frame.data.u8[i] <= 'z'){  rx\_frame.data.u8[i] = rx\_frame.data.u8[i] - 32;  }  }  }  //respond to sender  ESP32Can.CANWriteFrame(&rx\_frame);  }  } |

- Source code for second ESP32 module: sending the string that need to be convert to upper case, receiving response and show it to Terminal

|  |
| --- |
| #include <ESP32CAN.h>  #include <CAN\_config.h>  /\* the variable name CAN\_cfg is fixed, do not change \*/  CAN\_device\_t CAN\_cfg;  void setup() {  Serial.begin(115200);  Serial.println("iotsharing.com CAN demo");  /\* set CAN pins and baudrate \*/  CAN\_cfg.speed=CAN\_SPEED\_1000KBPS;  CAN\_cfg.tx\_pin\_id = GPIO\_NUM\_5;  CAN\_cfg.rx\_pin\_id = GPIO\_NUM\_4;  /\* create a queue for CAN receiving \*/  CAN\_cfg.rx\_queue = xQueueCreate(10,sizeof(CAN\_frame\_t));  //initialize CAN Module  ESP32Can.CANInit();  }  void loop() {  CAN\_frame\_t rx\_frame;  //receive next CAN frame from queue  if(xQueueReceive(CAN\_cfg.rx\_queue,&rx\_frame, 3\*portTICK\_PERIOD\_MS)==pdTRUE){  //do stuff!  if(rx\_frame.FIR.B.FF==CAN\_frame\_std)  printf("New standard frame");  else  printf("New extended frame");  if(rx\_frame.FIR.B.RTR==CAN\_RTR)  printf(" RTR from 0x%08x, DLC %d\r\n",rx\_frame.MsgID, rx\_frame.FIR.B.DLC);  else{  printf(" from 0x%08x, DLC %d\n",rx\_frame.MsgID, rx\_frame.FIR.B.DLC);  for(int i = 0; i < 8; i++){  printf("%c\t", (char)rx\_frame.data.u8[i]);  }  printf("\n");  }  }  else  {  rx\_frame.FIR.B.FF = CAN\_frame\_std;  rx\_frame.MsgID = 1;  rx\_frame.FIR.B.DLC = 8;  rx\_frame.data.u8[0] = 'h';  rx\_frame.data.u8[1] = 'e';  rx\_frame.data.u8[2] = 'l';  rx\_frame.data.u8[3] = 'l';  rx\_frame.data.u8[4] = 'o';  rx\_frame.data.u8[5] = 'c';  rx\_frame.data.u8[6] = 'a';  rx\_frame.data.u8[7] = 'n';    ESP32Can.CANWriteFrame(&rx\_frame);  }  } |

**4. Result**

[](https://1.bp.blogspot.com/-aoKC-qLvmxA/WaqHBfTtfCI/AAAAAAAAEPo/Rj9ecF_g1msfqKJeYWiQnZs-um-p3ReuQCLcBGAs/s1600/esp32_CAN.png)

**Figure: ESP32 CAN demo**