Controller area network (CAN) experiment

Author:- Alok Kumar Mishra

Akmishra_99@yahoo.com

Wednesday, October 15, 2025

Controller Area Network (CAN) is a serial communication protocol that allows microcontrollers and devices to communicate in a vehicle or industrial automation system without a host computer. It reduces wiring by using a bus architecture, and it ensures reliable, high-priority data transmission through a message-based protocol with a non-destructive arbitration process,for further description Please see

https://en.wikipedia.org › wiki › CAN_bus

Here we are going to experiment sending CAN packet between two Arduino UNO R4 WiFi. One Arduino will transmit CAN messages and another Arduino will receive it , and then we'll add third Teensy 4.1 node to it.

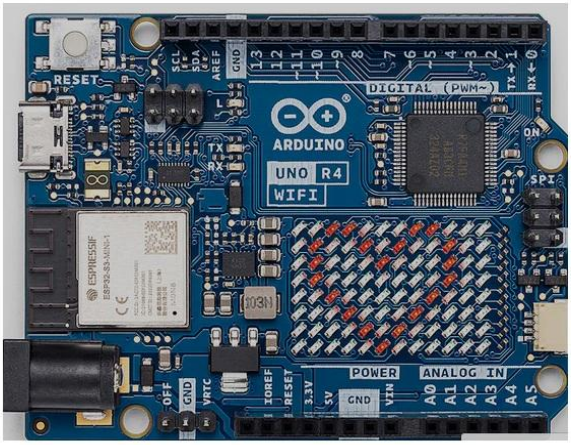


Figure 1 Arduino UNO R4 WiFi

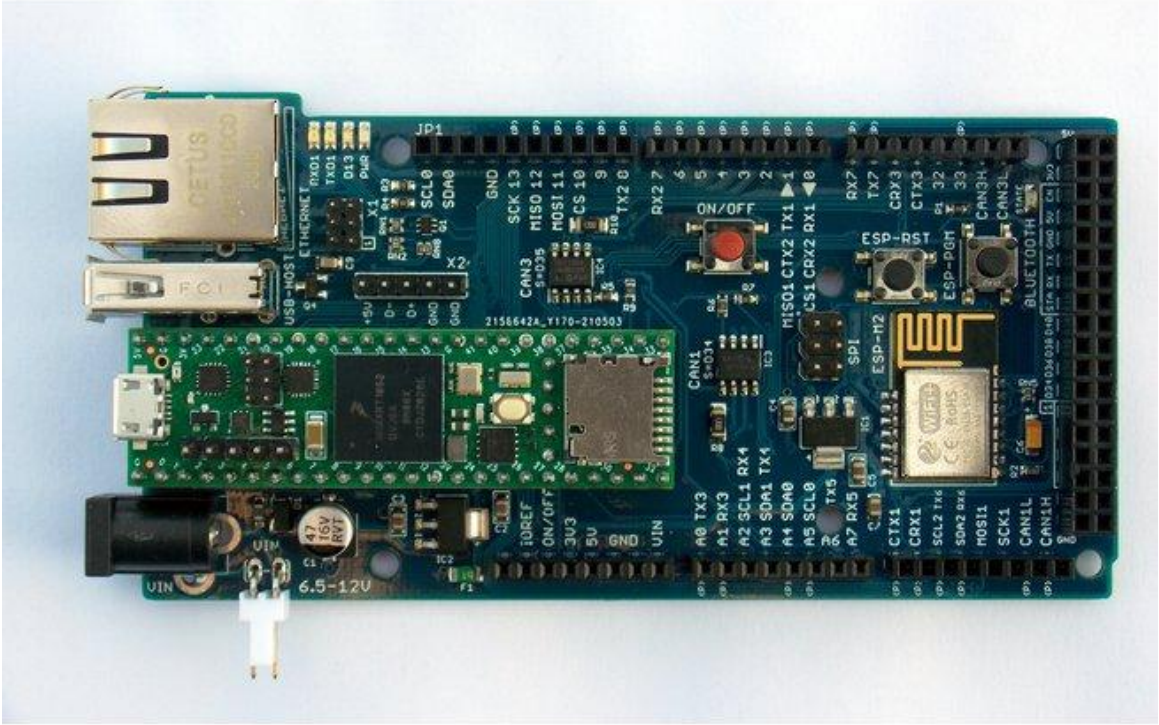


Figure 2 Teensy 4.1 mounted on expansion board (top view)

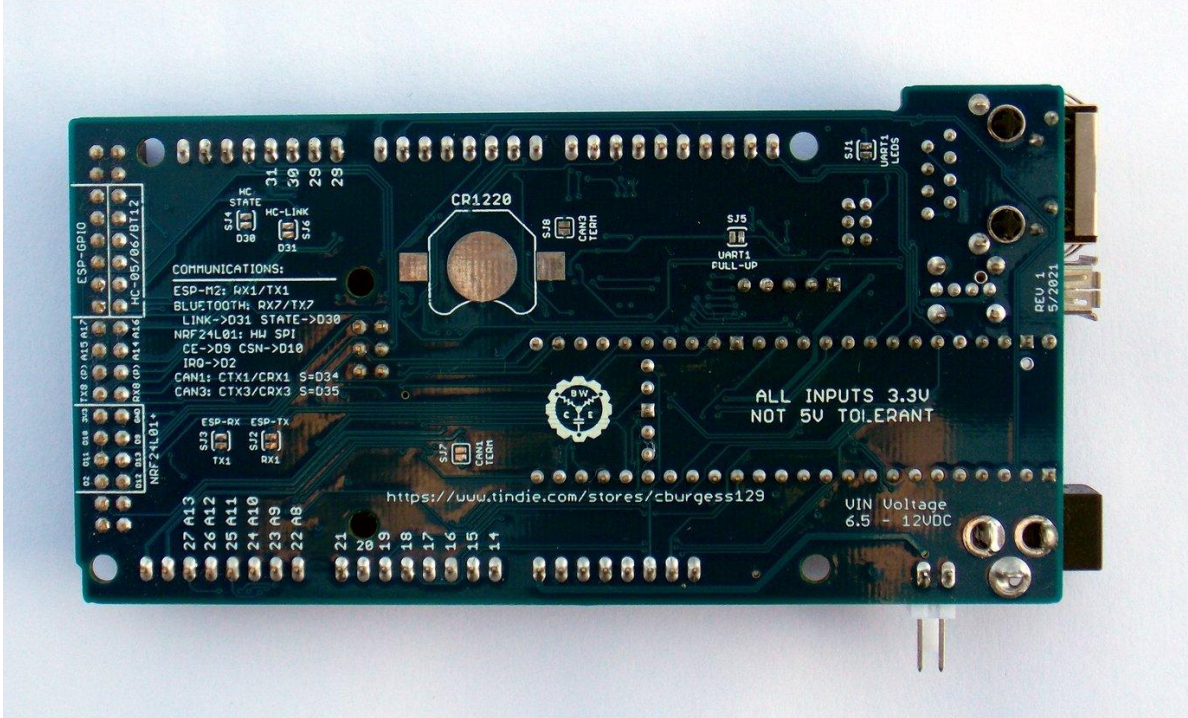


Figure 3 Teensy 4.1 mounted on expansion board (bottom view)

Arduino UNO R4 WiFi has CAN (CAN2.0) hardware, all we need is CAN transreceiver to connect to CAN bus,we are going to use WWZMDiB TJA1050 CAN Bus Transceiver Module

Figure 4 TJA1050 CAN Bus Transceiver Module

Here is circuit diagram of TJA1050 module

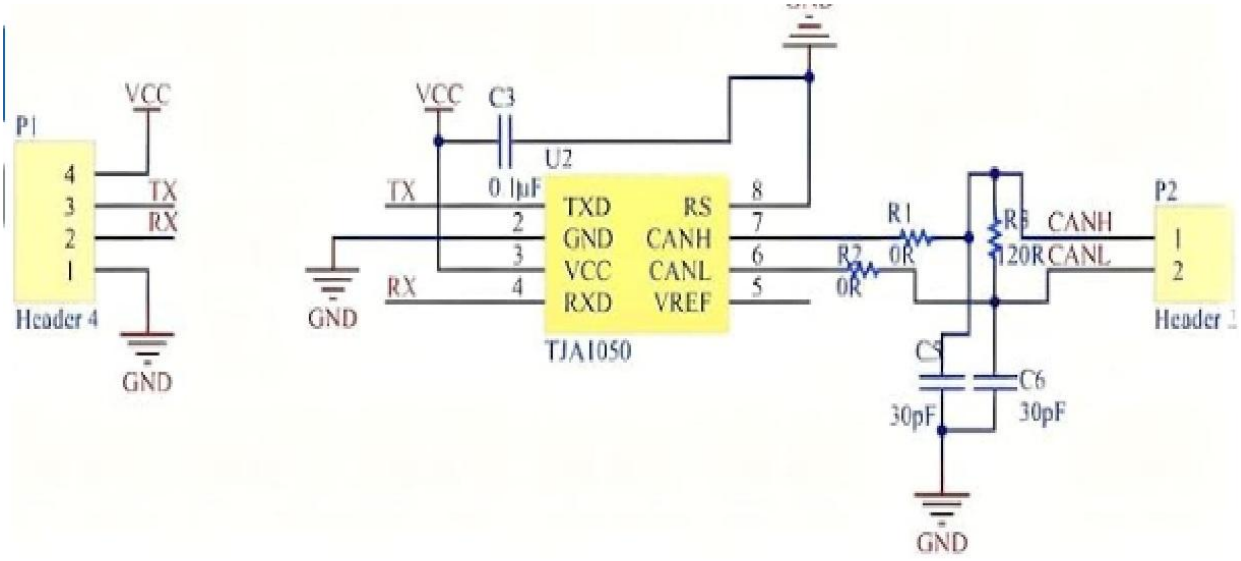


Figure 5 TJA1050 circuit

CAN bus requires 120 Ohm termination resistor, this module has 120 Ohm termination resistor built in. We are going to use two of these, teensy 4.1 with expansion board does not require CAN Transreceiver as it has one on expansion board,we are going to use CAN1 on teensy 4.1 ,at first we are going to experiment with two UNO R4 WiFi.

UNO R4 WiFi has one Hardware CAN bus port,

The CAN bus uses two wires: CAN high (CANH) and CAN low(CANL). On the UNO R4 WiFi, these pins are:

D13/CANRX0 (receive)

D10/CANTX0 (transmit)

Figure 6 2 node CAN bus

Figure 6 shows connection to Transreceiver and UNO R4 WiFi

After soldering pin headers to both Transreceiver (TJA1050) connect it to UNO R4 WiFi as shown in Figure 6

Connect CANH on two TJA1050 to each other and then connect CANL to each other TJA1050

Upload following Arduino code to one of Arduino UNO R4 WiFi

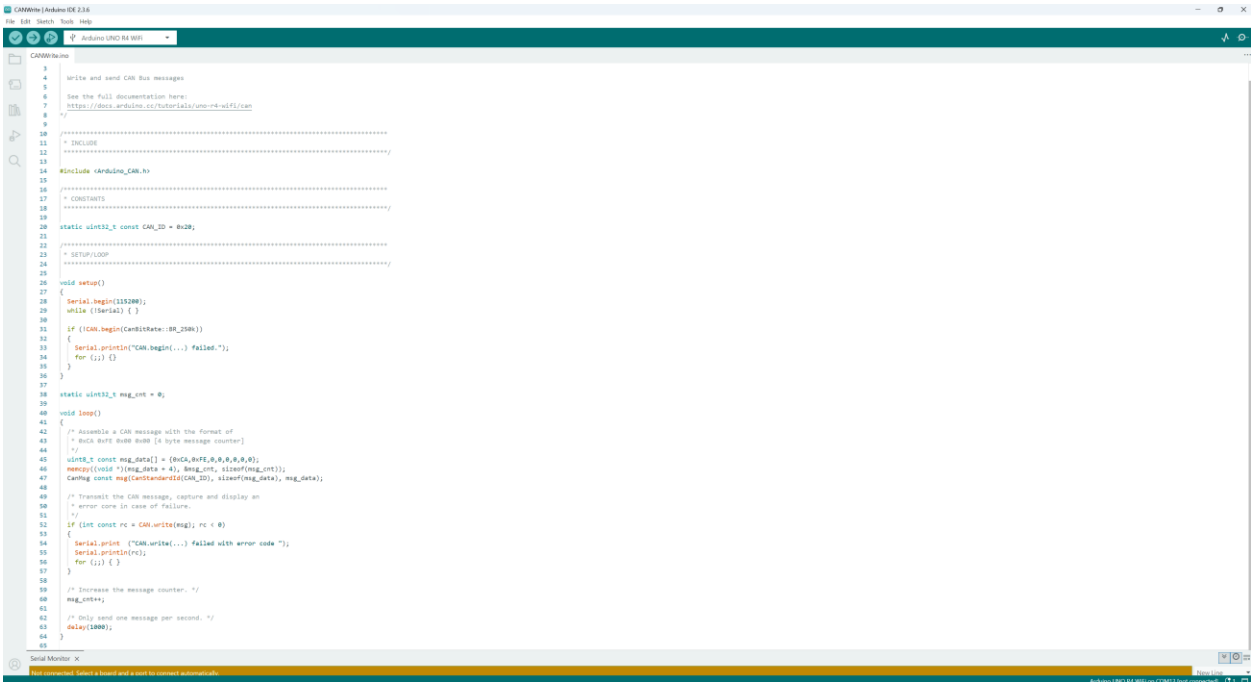
This file CANRead.info (also available in Arduino menu, file->Examples->Arduino_CAN->CANRead)

Figure 7 CANread.ino

Here it is

```
/* CANRead

Receive and read CAN Bus messages

  See the full documentation here:

  https://docs.arduino.cc/tutorials/uno-r4-wifi/can

*/

/**********************************************************************************
********

 * INCLUDE

 **********************************************************************************
********/

#include <Arduino_CAN.h>

void setup()

{

  Serial.begin(115200);

  while (!Serial) { }

  if (!CAN.begin(CanBitRate::BR_250k))

  {

    Serial.println("CAN.begin(...) failed.");

    for (;;) {}

  }

}

void loop()

{

  if (CAN.available())

  {

    CanMsg const msg = CAN.read();

    Serial.println(msg);

  }

}
```

Here is file CANWrite.ino



```
/*
  Write and send CAN Bus messages

  See the full documentation here:
  https://docs.arduino.cc/tutorials/uno-r4-wifi/can
*/

/**********************************************************************************
 * INCLUDE
 **********************************************************************************/

#include <Arduino_CAN.h>

/**********************************************************************************
 * CONSTANTS
 **********************************************************************************/

static uint32_t const CAN_ID = 0x20;

/**********************************************************************************
 * SETUP/LOOP
 **********************************************************************************/

void setup()
{
  Serial.begin(115200);
  while (!Serial) { }

  if (!CAN.begin(CanBitRate::BR_250k))
  {
    Serial.println("CAN.begin(...) failed.");
    for (;;) {}
  }
}

static uint32_t msg_cnt = 0;

void loop()
{
  /* Assemble a CAN message with the format of
   * 0xCA 0xFE 0x00 0x00 [4 byte message counter]
   */
  uint8_t const msg_data[] = {0xCA, 0xFE, 0, 0, 0, 0, 0, 0};
  memcpy((void *)(msg_data + 4), &msg_cnt, sizeof(msg_cnt));
  CanMsg const msg(CanStandardId(CAN_ID), sizeof(msg_data), msg_data);

  /* Transmit the CAN message, capture and display an
   * error core in case of failure.
   */
  if (int const rc = CAN.write(msg); rc < 0)
  {
    Serial.print  ("CAN.write(...) failed with error code ");
    Serial.println(rc);
    for (;;) { }
  }

  /* Increase the message counter. */
  msg_cnt++;

  /* Only send one message per second. */
  delay(1000);
}
```

Figure 8 CANwrite.ino

File CANwrite.ino

```
/*
  CANWrite

  Write and send CAN Bus messages

  See the full documentation here:
  https://docs.arduino.cc/tutorials/uno-r4-wifi/can
*/

/**********************************************************************
*********
 * INCLUDE

**********************************************************************
********/

#include <Arduino_CAN.h>

/**********************************************************************
*********
 * CONSTANTS

**********************************************************************
********/
```

```
static uint32_t const CAN_ID = 0x20;

/**************************************************************************
*********
 * SETUP/LOOP

**************************************************************************
********/

void setup()
{
  Serial.begin(115200);
  while (!Serial) { }

  if (!CAN.begin(CanBitRate::BR_250k))
  {
    Serial.println("CAN.begin(...) failed.");
    for (;;) {}
  }
}

static uint32_t msg_cnt = 0;

void loop()
{
  /* Assemble a CAN message with the format of
```

```
   * 0xCA 0xFE 0x00 0x00 [4 byte message counter]
   */
  uint8_t const msg_data[] = {0xCA,0xFE,0,0,0,0,0,0};
  memcpy((void *)(msg_data + 4), &msg_cnt, sizeof(msg_cnt));
  CanMsg const msg(CanStandardId(CAN_ID), sizeof(msg_data), msg_data);

  /* Transmit the CAN message, capture and display an
   * error core in case of failure.
   */
  if (int const rc = CAN.write(msg); rc < 0)
  {
    Serial.print  ("CAN.write(...) failed with error code ");
    Serial.println(rc);
    for (;;) { }
  }

  /* Increase the message counter. */
  msg_cnt++;

  /* Only send one message per second. */
  delay(1000);
}
```

Upload CANread.ino to one of microcontroller UNO R4 WiFI and then upload CANWrite.ino to second UNO R4 WiFi and open a putty session on com port on which UNO R4 WiFi with read sketch (CANRead.ino) is running and you”ll see data being read as shown in Figure 10

As wee can see that first two bytes received is 0XCA and 0xFE followed by msg_cnt variable which is incremented after 1 second (1000 milliseconds) and written to CANbus

Here is picture of entire two CAN node setup

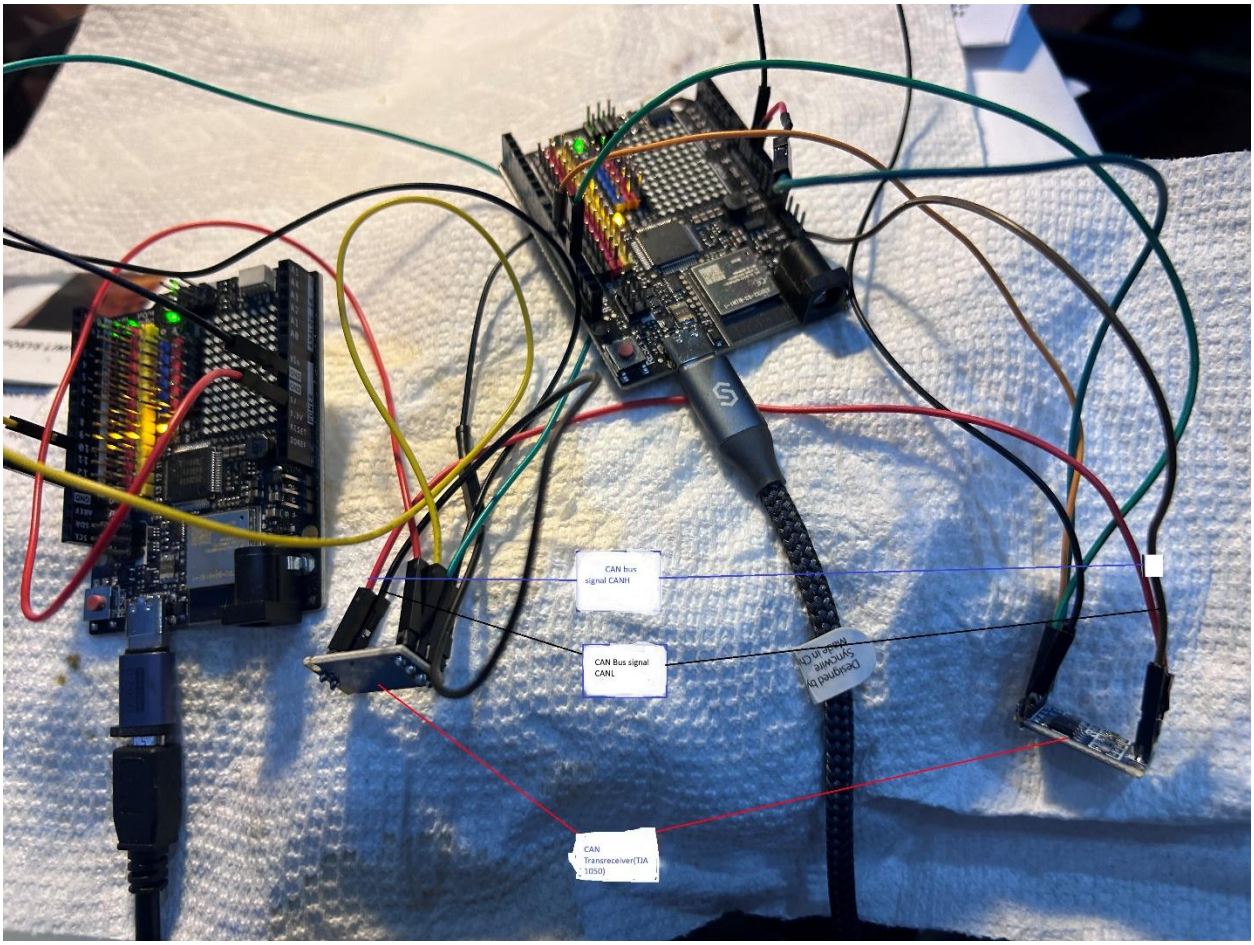


Figure 9 2 node UNO R4 WiFi setup

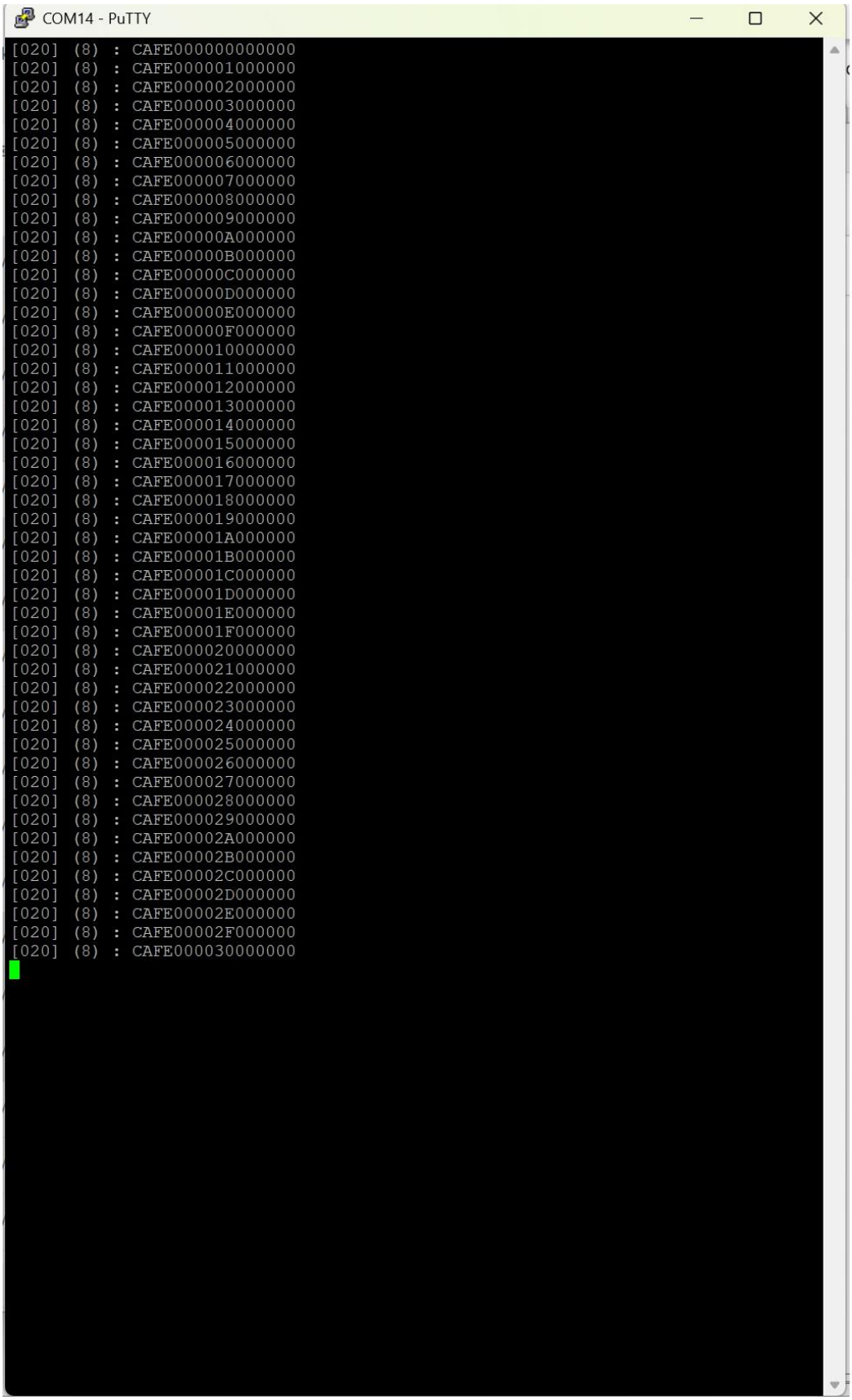


Figure 10 CAN read output on putty

Adding third node (teensy 4.1 with expansion board)

Here is block diagram of how teensy 4.1 is connected

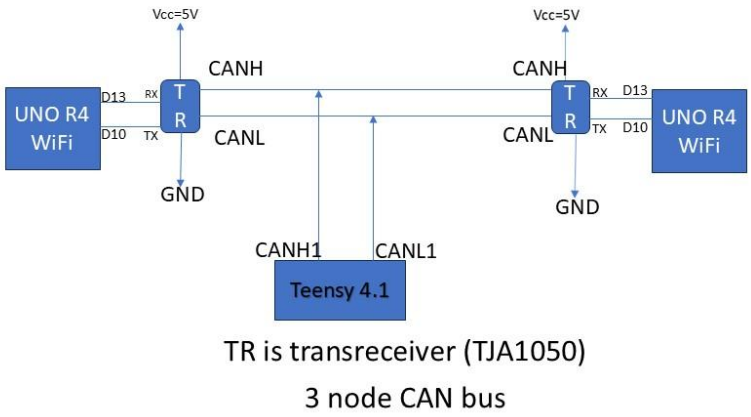


Figure 11 three node CAN setup with Teensy 4.1

Connect teensy CANH1 to CANH bus of Figure 11 and Teensy CANL1 to CANL bus of Figure 11,we are using Teensy CAN1 interface.

Here program (Teensy_4_1_canread.ino) which we need to upload to Teensy4.1 with expansion board

```
#include <FlexCAN_T4.h>

FlexCAN_T4<CAN1, RX_SIZE_256, TX_SIZE_16> can1;

CAN_message_t msg;


void setup() {
```

```
  // put your setup code here, to run once:
  Serial.begin(115200);
  can1.begin();
  can1.setBaudRate(250000);
  Serial.println("Started");
}

void loop() {
  // put your main code here, to run repeatedly:
 if ( can1.read(msg) ) {
    Serial.print("CAN1 ");
    Serial.print("MB: "); Serial.print(msg.mb);
    Serial.print("  ID: 0x"); Serial.print(msg.id, HEX );
    Serial.print("  EXT: "); Serial.print(msg.flags.extended );
    Serial.print("  LEN: "); Serial.print(msg.len);
    Serial.print(" DATA: ");
    for ( uint8_t i = 0; i < 8; i++ ) {
      Serial.print(msg.buf[i]); Serial.print(" ");
    }
    Serial.print("  TS: "); Serial.println(msg.timestamp);
  }
}
```

Connect CAN bus CANH to teensy 4.1 expansion board CANH1 and CANL to CANL1 of teensy with expansion board

Here is output (Teensy 4.1) reading CAN bus messages

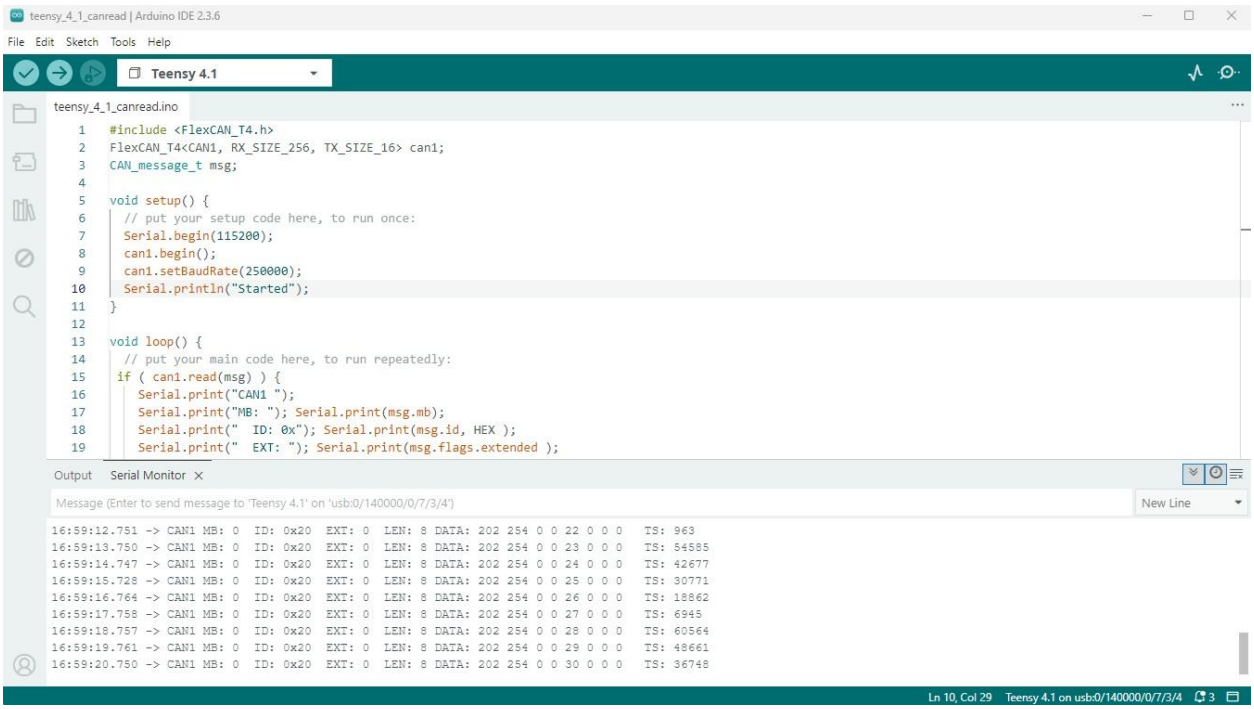


Figure 12 output of CAN read on teensy4.1 with expansion board

Following lines display message received by teensy 4.1 from CAN bus

```
16:59:12.751 -> CAN1 MB: 0  ID: 0x20  EXT: 0  LEN: 8 DATA: 202 254 0 0 22 0 0 0   TS: 963
16:59:13.750 -> CAN1 MB: 0  ID: 0x20  EXT: 0  LEN: 8 DATA: 202 254 0 0 23 0 0 0   TS: 54585
16:59:14.747 -> CAN1 MB: 0  ID: 0x20  EXT: 0  LEN: 8 DATA: 202 254 0 0 24 0 0 0   TS: 42677
16:59:15.728 -> CAN1 MB: 0  ID: 0x20  EXT: 0  LEN: 8 DATA: 202 254 0 0 25 0 0 0   TS: 30771
16:59:16.764 -> CAN1 MB: 0  ID: 0x20  EXT: 0  LEN: 8 DATA: 202 254 0 0 26 0 0 0   TS: 18862
16:59:17.758 -> CAN1 MB: 0  ID: 0x20  EXT: 0  LEN: 8 DATA: 202 254 0 0 27 0 0 0   TS: 6945
16:59:18.757 -> CAN1 MB: 0  ID: 0x20  EXT: 0  LEN: 8 DATA: 202 254 0 0 28 0 0 0   TS: 60564
16:59:19.761 -> CAN1 MB: 0  ID: 0x20  EXT: 0  LEN: 8 DATA: 202 254 0 0 29 0 0 0   TS: 48661
16:59:20.750 -> CAN1 MB: 0  ID: 0x20  EXT: 0  LEN: 8 DATA: 202 254 0 0 30 0 0 0   TS: 36748
```

Figure 13 output from teensy4.1

Here is picture of 3 node CAN setup, signals on breadboard implements CAN bus

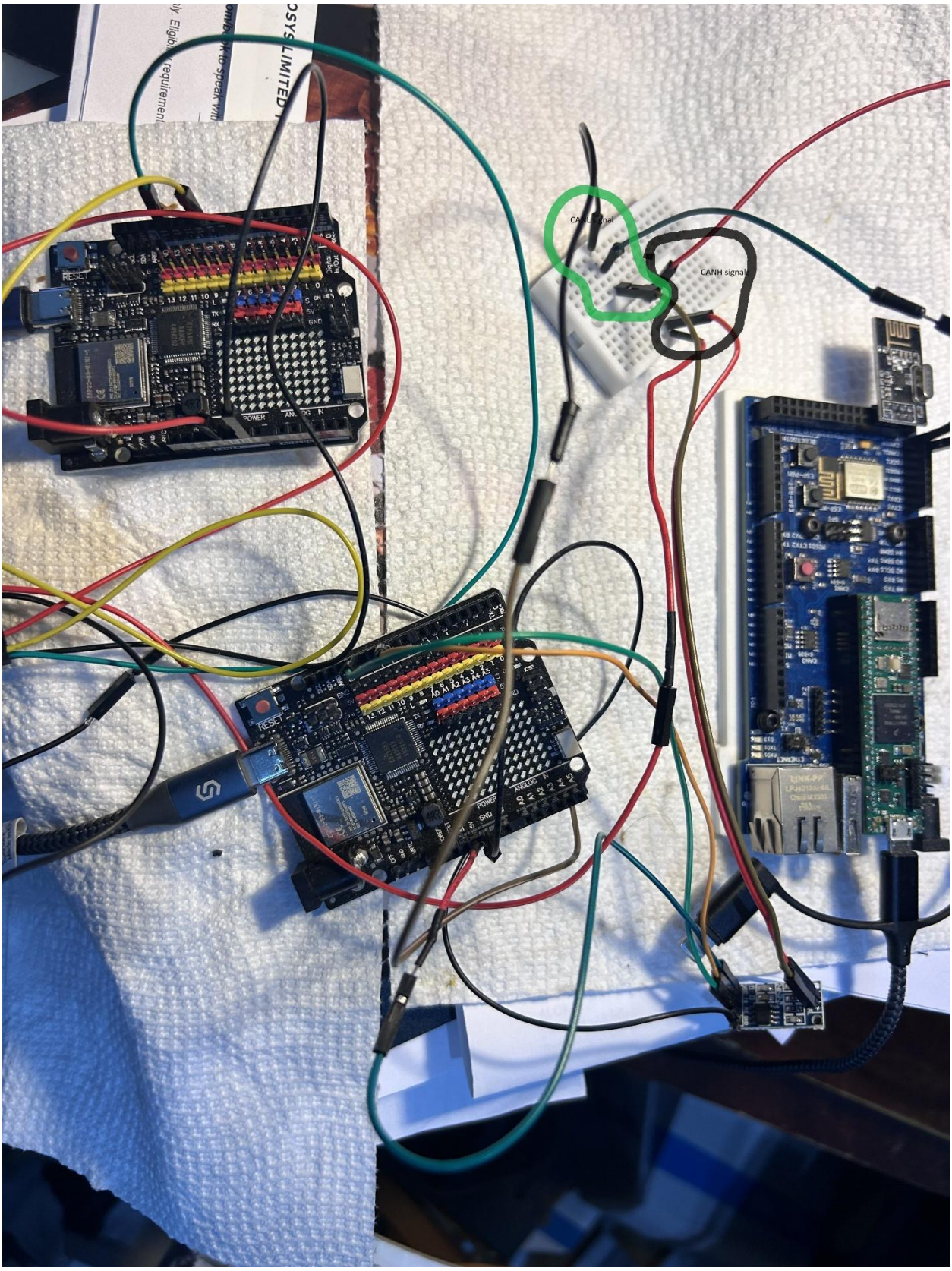


Figure 14 3 node CAN bus setup