# Arduino UNO R4 Minima with CAN Bus Module for Sensor Data Transmission

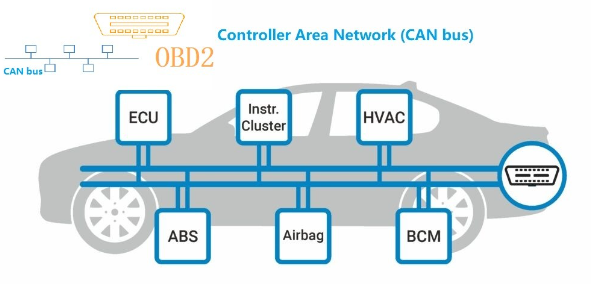
## Description

### Know about CAN Bus

First of all, let us know about CAN BUS.

**CAN bus,** short for **Controller Area Network bus**, is a robust and widely used communication protocol primarily employed in automotive and industrial applications. It was originally developed by Bosch in the 1980s to address the increasing complexity of automotive electronic systems.

CAN bus is a multi-master, message-oriented protocol designed for high-speed, reliable communication between electronic control units (ECUs) in vehicles or industrial systems. It uses a differential serial bus to transmit data between nodes without the need for a centralized controller. One of the key features of CAN bus is its reliability and robustness. It employs error detection and error handling mechanisms, including cyclic redundancy checks (CRCs), to ensure data integrity even in noisy environments. It supports a scalable network architecture, allowing for easy expansion by adding more nodes without significant changes to the overall system. This makes it suitable for applications ranging from simple vehicle subsystems to complex automotive or industrial networks It offers deterministic real-time communication, making it suitable for applications where timing constraints are critical, such as engine control, transmission systems, anti-lock braking systems (ABS), and electronic stability control (ESC) in vehicles. And it is a cost-effective solution for communication within vehicles and industrial machinery compared to alternatives like Ethernet or USB. Its simplicity and widespread adoption contribute to lower implementation costs.



### Project Description

In this project, we'll explore how to interface an Arduino UNO R4 Minima with a CAN bus module to transmit sensor data. We'll utilize a raindrop sensor and a soil moisture sensor to collect environmental data, and then send this data over the CAN bus to another device or system for further processing or monitoring. This project provides a hands-on experience with sensor integration and communication protocols, helping users understand the basics of CAN bus communication.

## How-To Guide

### 1. Gather Components

* 1 x Arduino Uno R4 Minima
* 1 x Breadboard
* 1 x Raindrop Sensor
* 1 x Soil Moisture Sensor
* 1 x CAN Bus Module
* 20 x Jumper wires

### 2. Assemble the Circuit

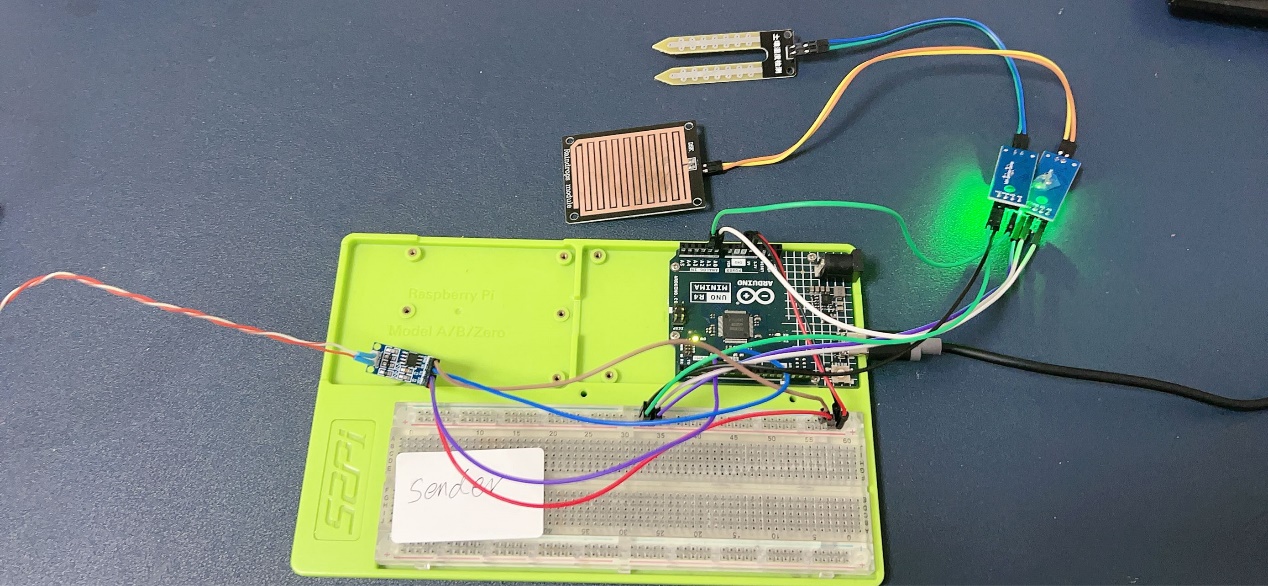
* Connect the raindrop sensor and soil moisture sensor to the Arduino Uno R4 Minima as per their respective datasheets or pinout diagrams.
* Connect the CAN bus module to the Arduino Uno R4 Minima according to its datasheet or pinout diagram.
* Ensure proper power and ground connections for all components.

#### Wiring Diagram

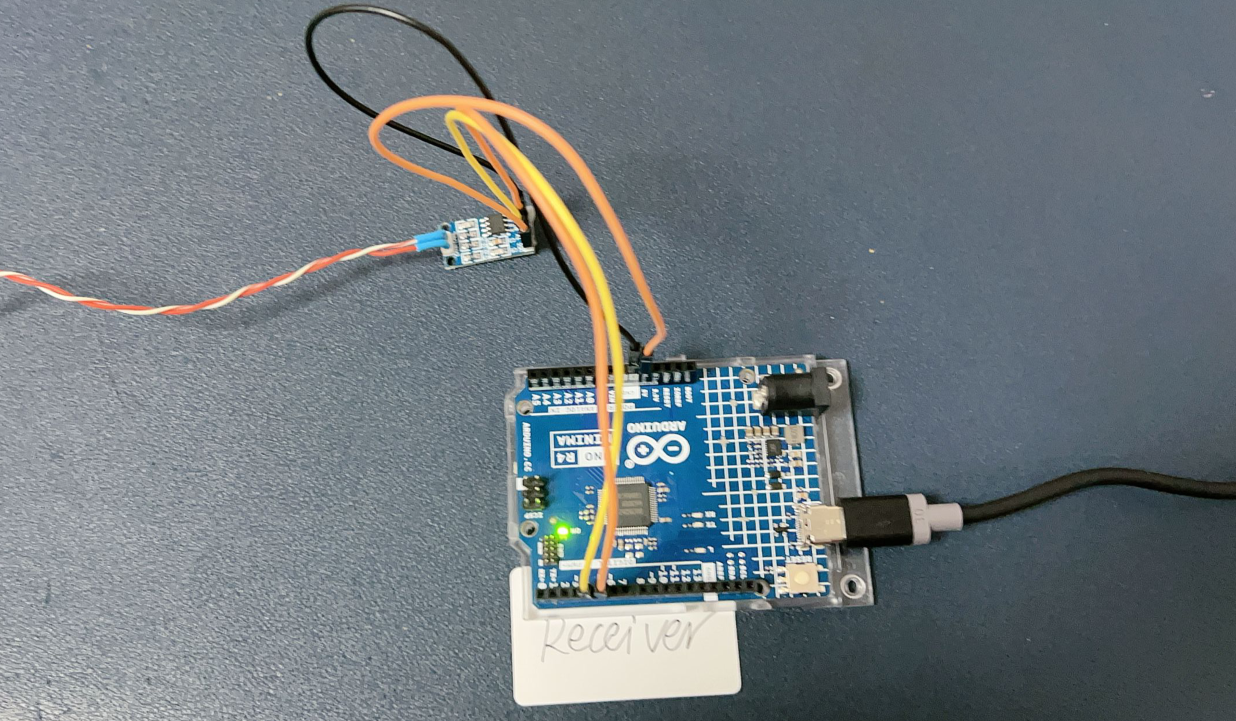
|  |  |
| --- | --- |
| Arduino UNO R4 Minima | Rain drop sensor |
| 5V | VCC |
| GND | GND |
| NC - Not connect | DO |
| A0 | AO |

|  |  |
| --- | --- |
| Arduino UNO R4 Minima | Soil Moisture |
| 5V | VCC |
| GND | GND |
| NC - Not connect | DO |
| A1 | AO |

|  |  |
| --- | --- |
| Arduino UNO R4 Minima | CAN Bus Module |
| 3.3V | VCC |
| GND | GND |
| D5 (CANRX0) | CANRX |
| D4 (CANTX0) | CANTX |

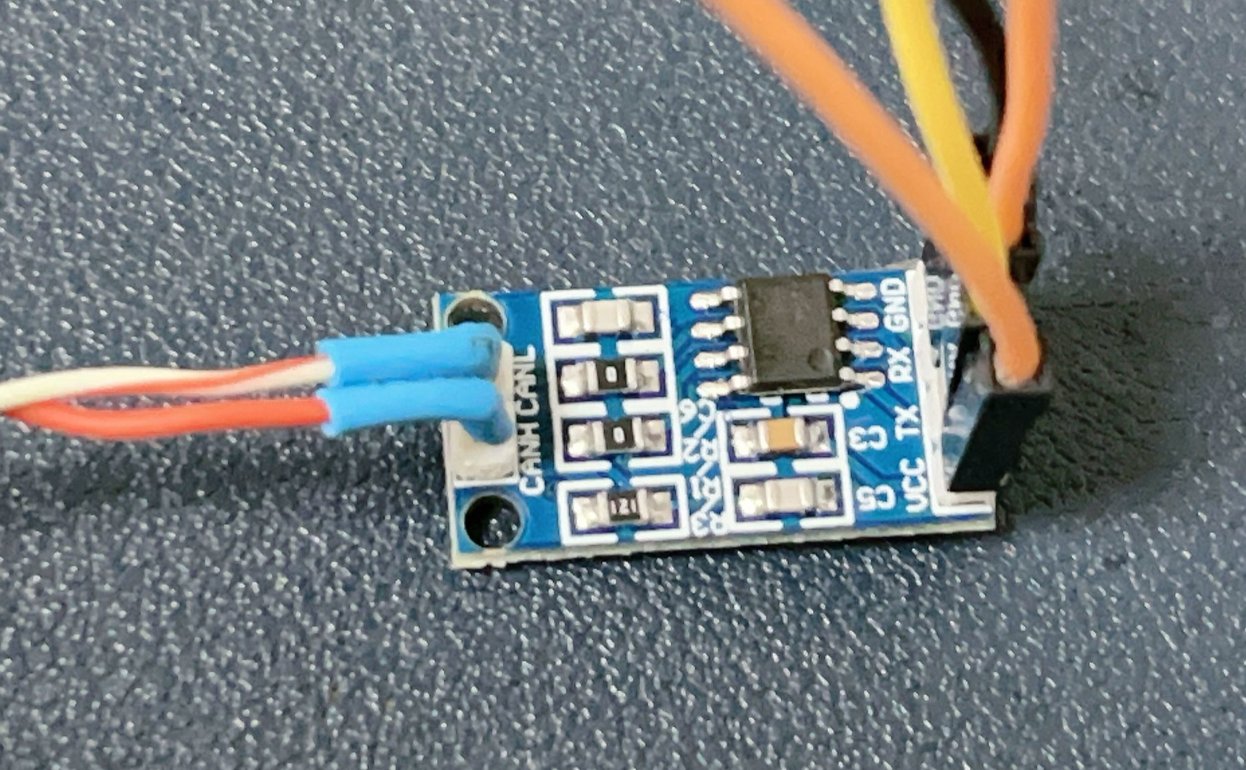


Receiver:



CAN BUS module:

|  |  |
| --- | --- |
| CAN Transceiver 1 | CAN Transceiver 2 |
| CANH (HIGH) | CANH (HIGH) |
| CANL (LOW) | CANL (LOW) |



### 3. Write the Arduino Code

#### Data Sender

Open the Arduino IDE on your computer and create a new sketch. Then, use the following code:

#include <Arduino\_CAN.h>

const int raindropPin = A0;   // Analog pin for raindrop sensor

const int moisturePin = A1;   // Analog pin for soil moisture sensor

int raindropValue = 0;        // Variable for storing raindrop sensor value

int moistureValue = 0;        // Variable for storing soil moisture sensor value

void setup() {

  Serial.begin(115200);

while (!Serial) {}

  // Initialize CAN bus

   if (!CAN.begin(CanBitRate::BR\_250k))

  {

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

    for (;;) {}

  }

}

void loop() {

  raindropValue = analogRead(raindropPin); // Read raindrop sensor value

  moistureValue = analogRead(moisturePin); // Read soil moisture sensor value

  // Create CAN message

 uint8\_t const msg\_data[] = {raindropValue, moistureValue, 0, 0, 0, 0, 0, 0}; // create a 4 bytes data

   // Send CAN message

  CAN.sendMsgBuf(0x100, 0, 8, canMsg);

  delay(500); // Delay for stability

}

#### Data Receiver

#include <Arduino\_CAN.h>

void setup() {

  if (!CAN.begin(CanBitRate::BR\_250k)) {

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

    for (;;) {}

  }

  // init serial port

  Serial.begin(9600);

  while (!Serial) {

      ;  // wait for serial port to connect. Needed for native USB port only

    }

}

void loop() {

  if (CAN.available())

  {

    CanMsg const msg = CAN.read();

    // read data from sender part

    int raindropValue = (int) msg.data[0];

    int moistureValue = (int) msg.data[1];

    Serial.print("Received data: ");

    Serial.print("Rain drop sensor: ");

    Serial.print(raindropValue);

    Serial.print("\tSoil Moisture sensor: ");

    Serial.println(moistureValue);

    delay(1000);

  }

}

### 4. Understand the Code

This code reads analog values from the raindrop sensor and soil moisture sensor, and sends these values over the CAN bus during 1000ms.

The library used is built into the Board Package, so no need to install the library if you have the Board Package installed.

To initialize the library, use **CAN.begin(CanBitRate::BR\_250k)**, where a CAN bit rate is specified. Choose between:

* BR\_125k (125000)
* BR\_250k (250000)
* BR\_500k (500000)
* BR\_1000k (1000000)

The Arduino\_CAN library is used to initialize and communicate over the CAN bus.

To send a CAN message, you can create a **CanMsg** object, which should contain the **CAN\_ID**, **size** and **message data**. Below is an example on how to create such object.

 uint8\_t const msg\_data[] = {raindropValue, moistureValue, 0, 0, 0, 0, 0, 0}; // create a 4 bytes data

After you have crafted a CAN message, we can send it off, by using the **CAN.write()** method. The following example creates a CAN message that increases each time void loop() is executed.

The code creates a CAN message containing the sensor data and sends it over the CAN bus.

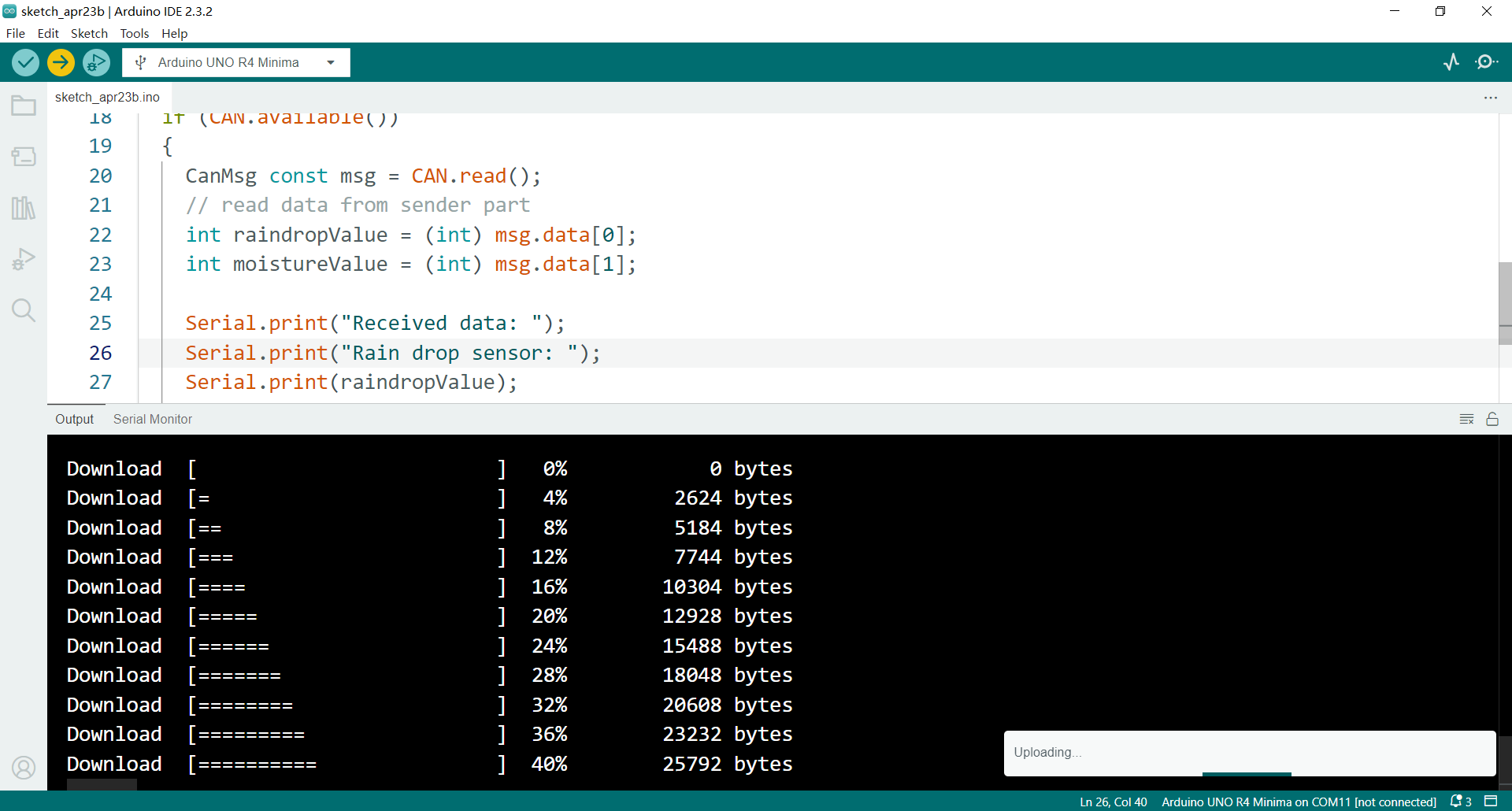
### 5. Test the Project

Upload the code to your Arduino Uno R4 Minima.

* Sender part

Do not worry about the error message, because the receiver part is not available right now.

And upload the receiver part demo code, and reset both of them, you will get the right feedback.

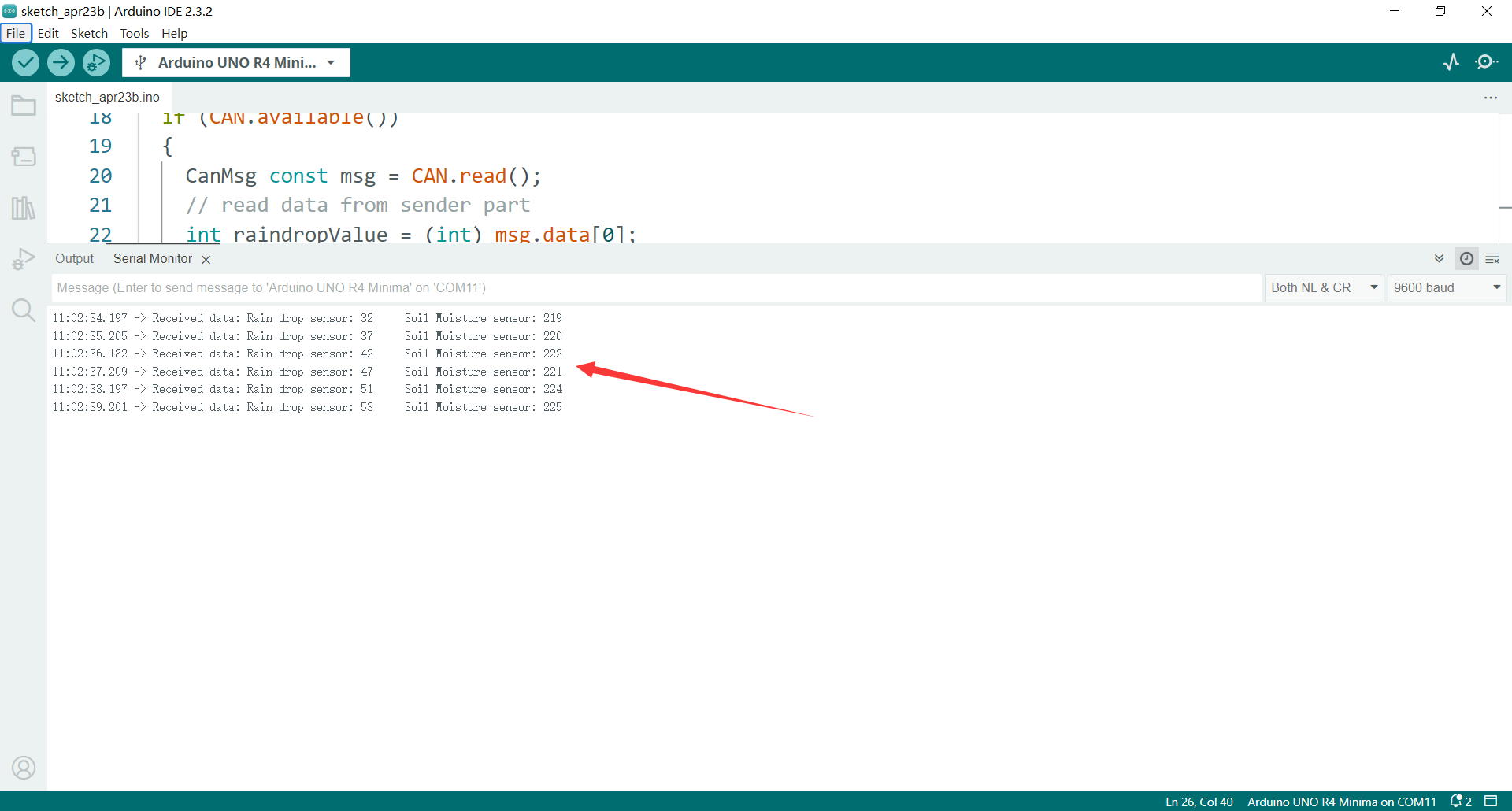


Connect the CAN bus module to a CAN bus network or to another device capable of receiving CAN messages.

* From sender’s serial monitor:



From receiver’s serial monitor:



### 6. Experiment and Learn

Adjust the code to include additional sensors or modify the data format being sent over the CAN bus.

Explore other features of the Arduino\_CAN library to customize CAN bus communication.

Integrate the project with a CAN bus monitoring tool to visualize and analyze the transmitted data.

By following these steps, you can create a sensor data transmission system using an Arduino UNO R4 Minima and a CAN bus module, enabling communication between sensors and other devices over a CAN bus network.