Connection Arduino Uno to a NRF2401

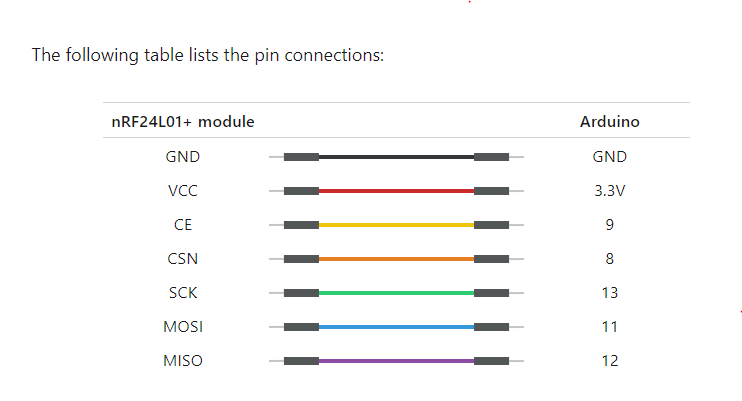
**Wiring a nRF24L01+ module to an Arduino**

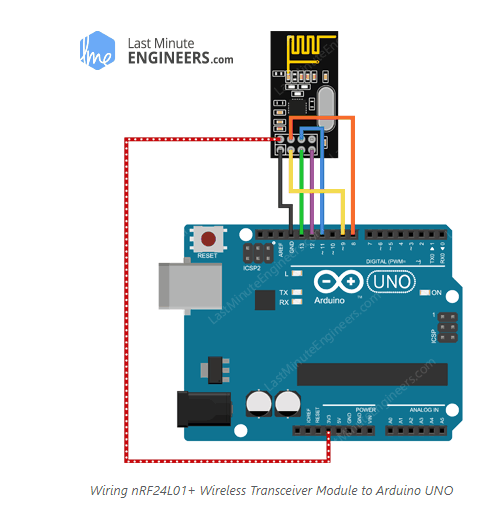
Now that we know everything about how the nRF24L01+ module works, we can start hooking it up to our Arduino.

To begin, connect the module’s VCC pin to Arduino’s 3.3V and the GND pin to ground. CSN and CE pins can be connected to any digital pin on an Arduino; in our case, they are connected to digital pins #8 and #9.

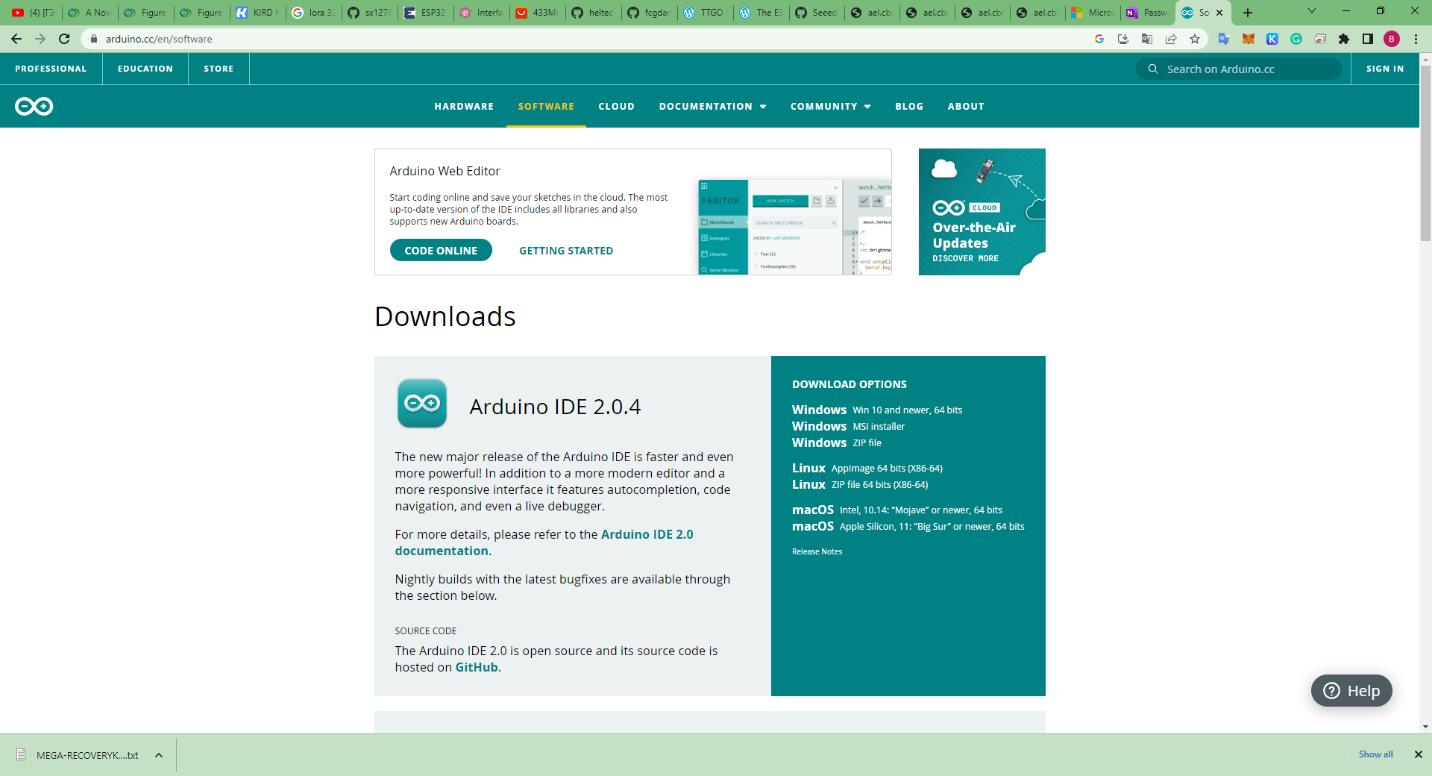
Let’s wire up the SPI pins. Note that each Arduino board has a unique set of SPI pins that must be connected accordingly. For Arduino boards such as the UNO/Nano V3.0, these pins are digital 13 (SCK), 12 (MISO), 11 (MOSI) and 10 (SS).

If you’re using a different Arduino board, check the official documentation for [SPI pin locations](https://www.arduino.cc/en/reference/SPI) before proceeding.





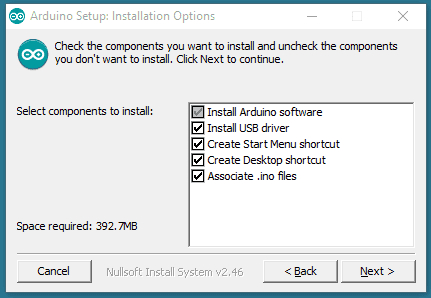
Download the Arduino Software (IDE)



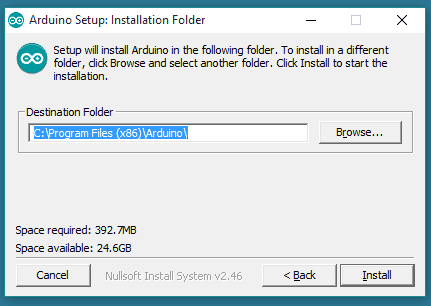
https://www.arduino.cc/en/software

Download the Arduino IDE that is compatible with your computer system.

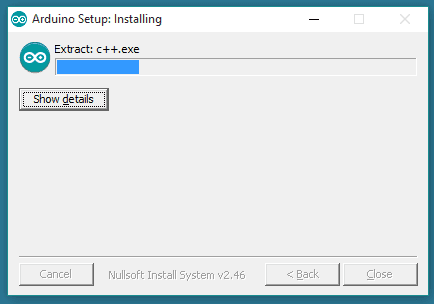
When the download finishes, proceed with the installation and please allow the driver installation process when you get a warning from the operating system.



Choose the components to install.



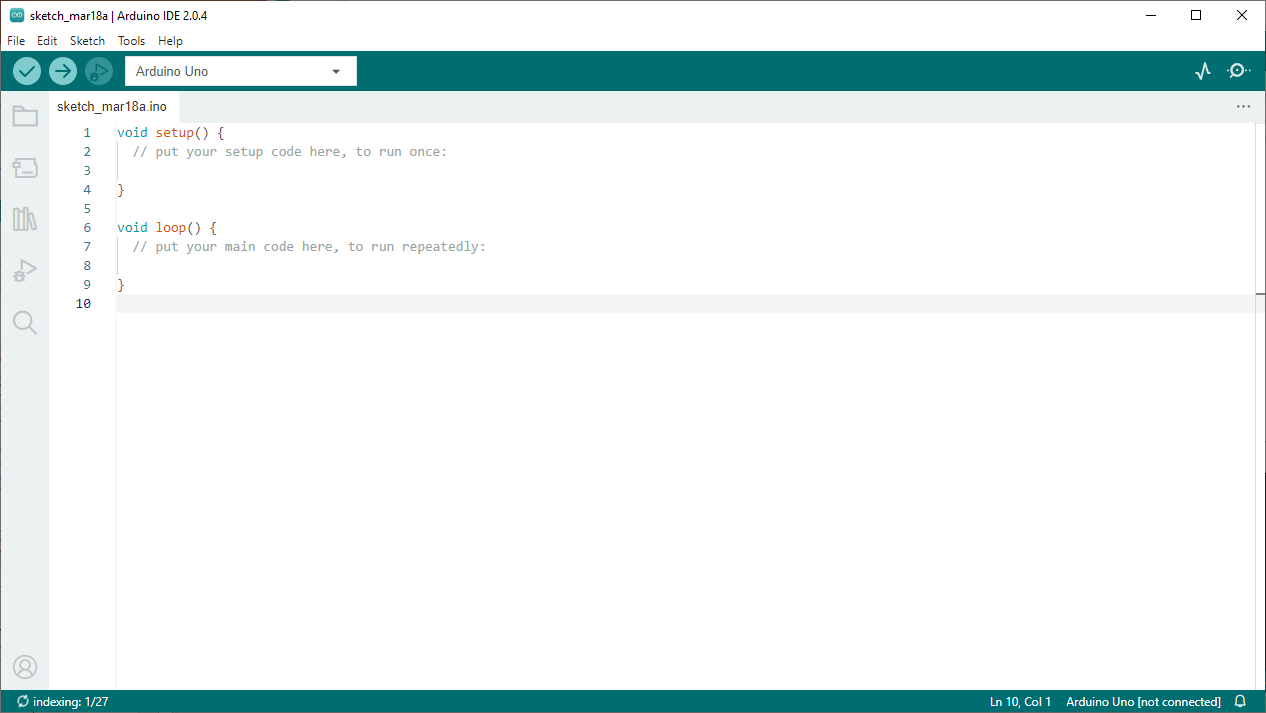
Choose the installation directory.



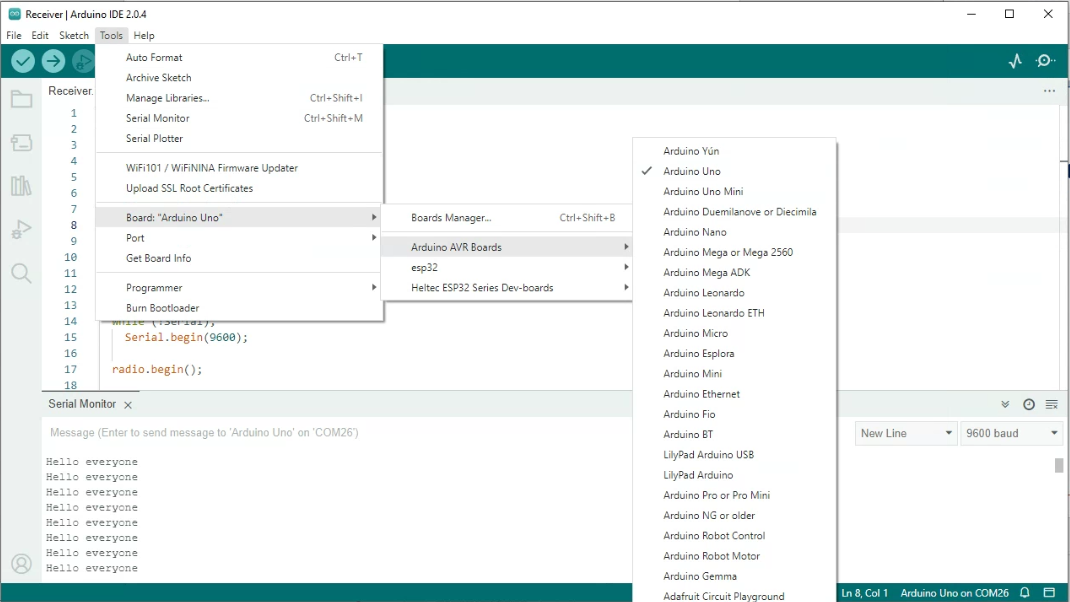
Installation in progress.

The process will extract and install all the required files to execute properly the Arduino Software (IDE)

Open the Arduino IDE.



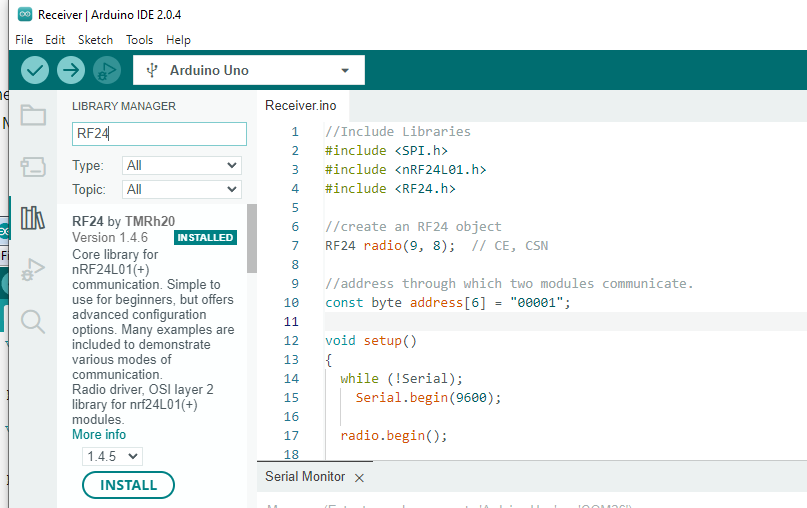
Select the Board(Arduino Uno).



## **Library Installation**

There are many libraries available for the nRF24L01+ module, but one of the most popular is [RF24](https://tmrh20.github.io/). This library has been around for a long time. It’s simple to use for beginners while still providing a lot for advanced users. We will use this library in our examples.

To install the library, navigate to **Sketch > Include Library > Manage Libraries…** Wait for the Library Manager to download the library index and update the list of installed libraries.



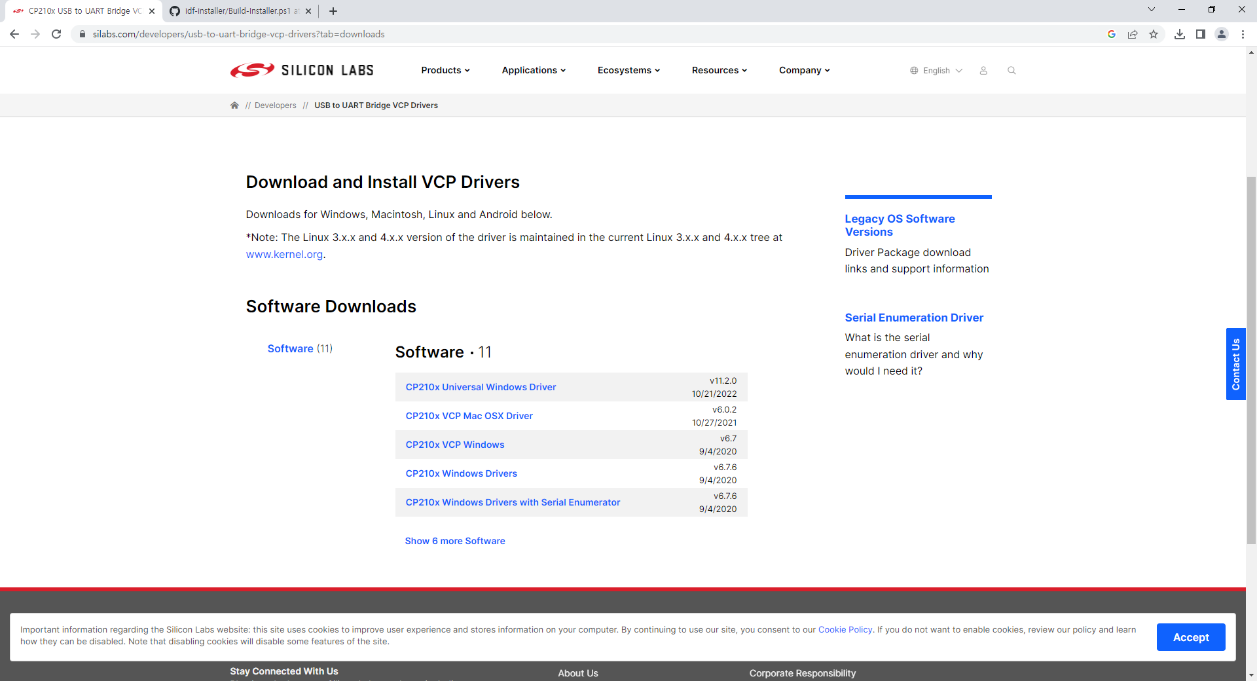
Copy and paste the below code here.

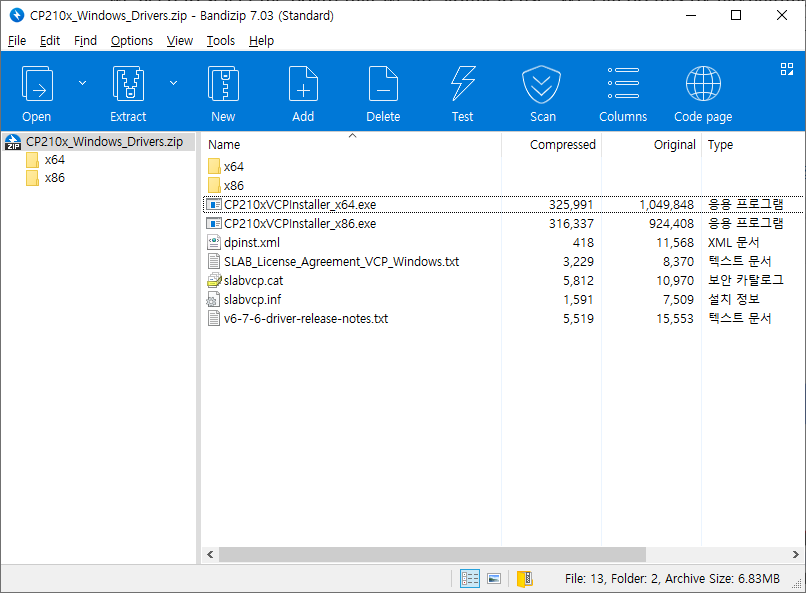
we need to select the board that we are going to use. We can do this by navigating to **Tools > Port > {Board}**. The board(s) that are connected to your computer should appear here, and we need to select it by clicking it. In this case, our board is displayed as **COM25**.

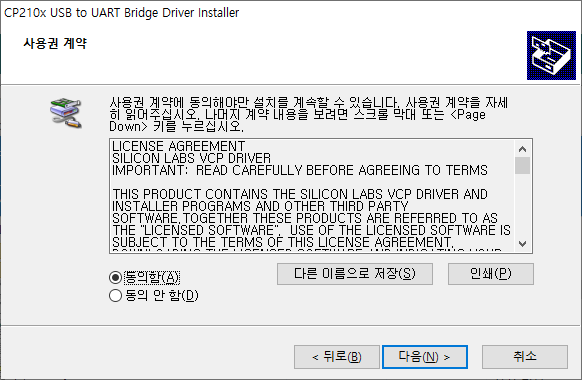
If you have connected your board to your computer but there is no com port

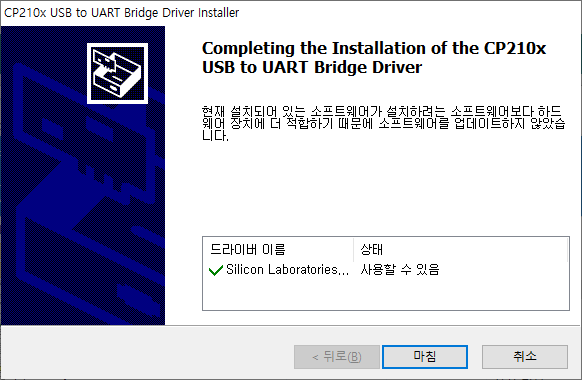
Download the usb driver and install your usb driver

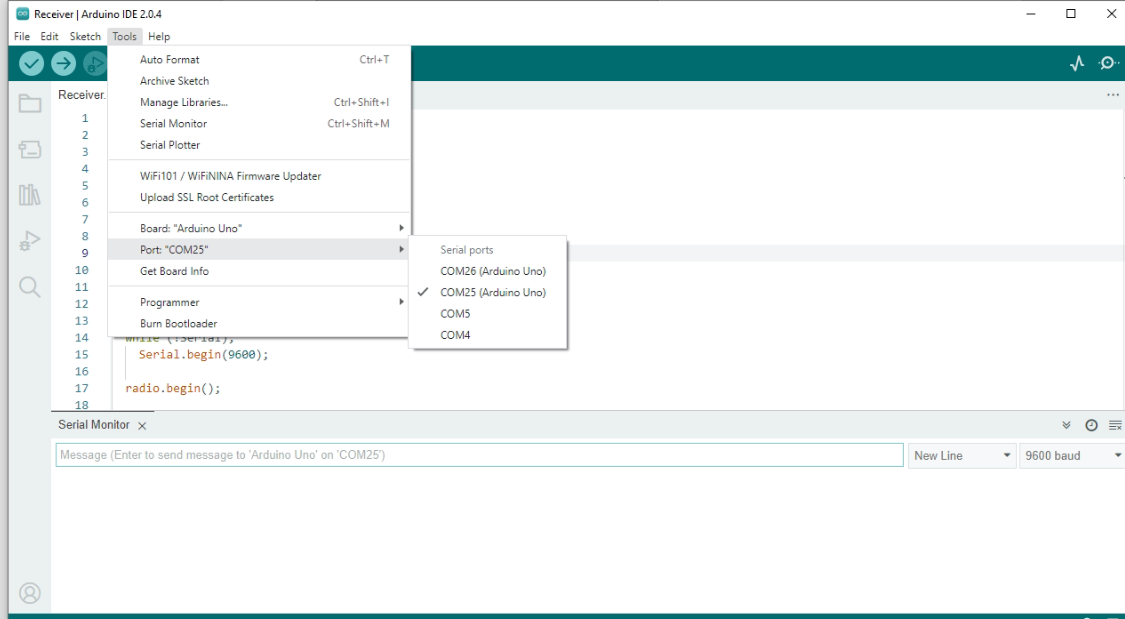
<https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads>



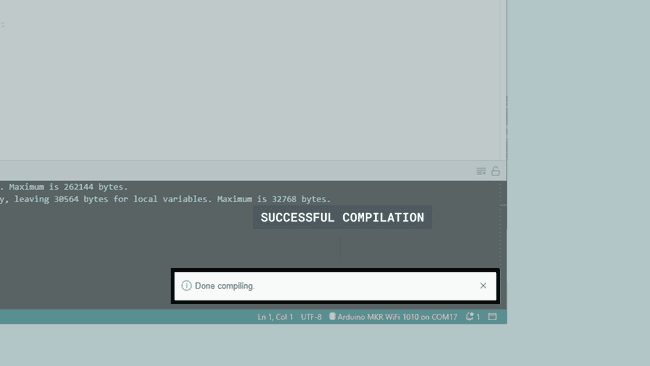






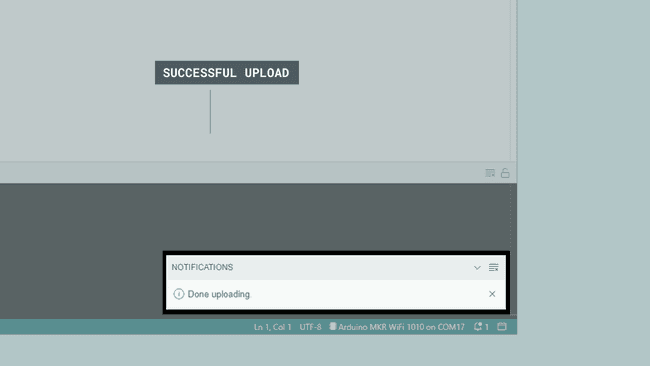


Click on the verify tool (checkmark). Since we are verifying an empty sketch, we can be sure it is going to compile. After a few seconds, we can see the result of the action in the console (black box in the bottom).



 With the board selected, we are good to go! Click on the **upload** button, and it will start uploading the sketch to the board.

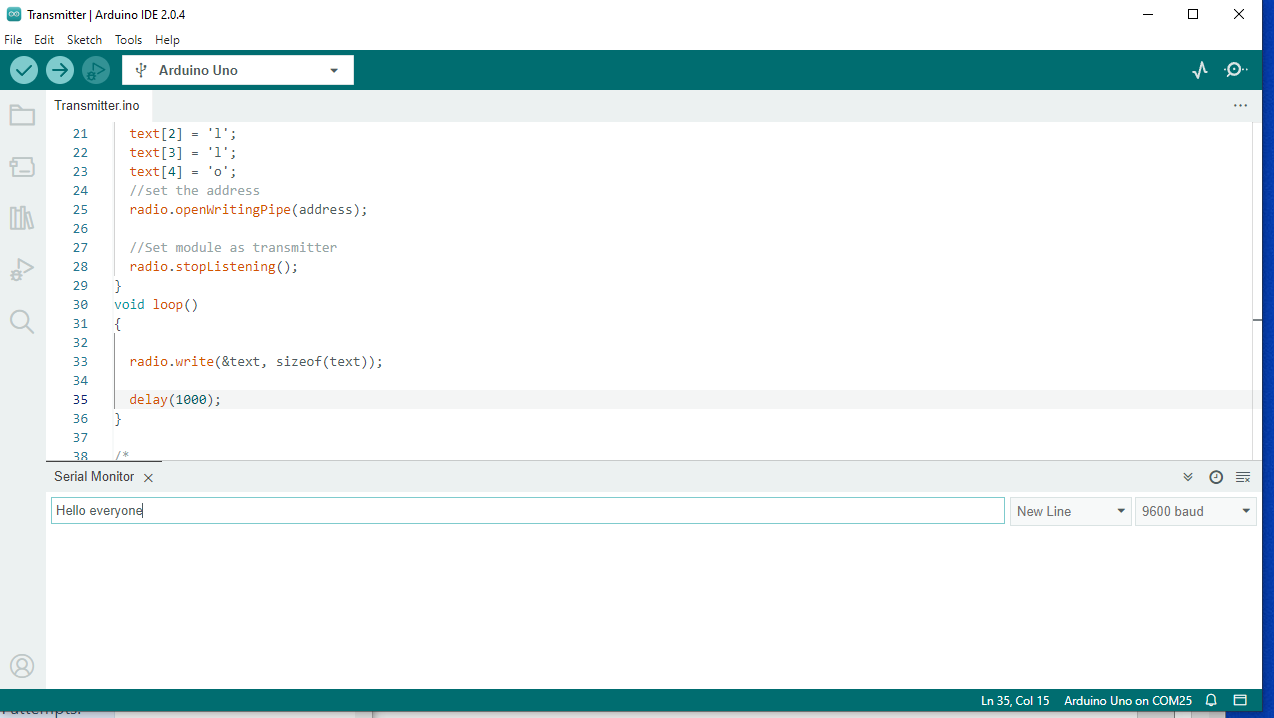
When it is finished, a notification pops up in the bottom right of your IDE window. Of course, sometimes there are some complications when uploading, and these errors will be listed here as well.



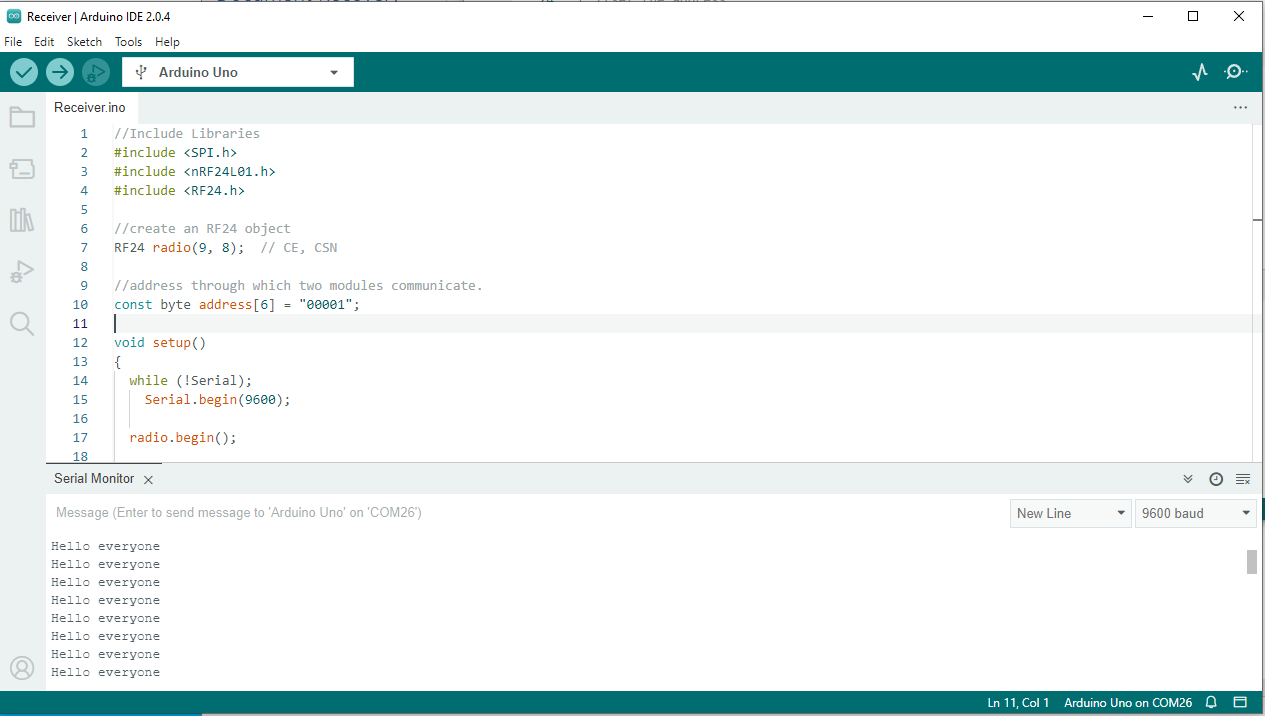
Congratulations, you have now uploaded a sketch to your Arduino board!

When it has finished uploading, click on the Serial Monitor button, located at the top right corner of the IDE. This will launch the Serial Monitor in the bottom of the IDE, replacing the console section.

Transmitter section

When data is written and sent on the serial monitor, it is sent by the nrf24l01 module

Receiver section

When new data is read by the nrf24l01 module, it is printed on the serial monitor 

**Run the 2 following code on a board according to the above instructions**

Source code.

Transmitter

//Include Libraries

#include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>

//create an RF24 object

RF24 radio(9, 8);  // CE, CSN

//address through which two modules communicate.

const byte address[6] = "00001";

  //Send message to receiver

  char text[100];

  bool stringComplete = false;  // whether the string is complete

void setup()

{

  Serial.begin(9600);

  radio.begin();

  text[0] = 'H';

  text[1] = 'e';

  text[2] = 'l';

  text[3] = 'l';

  text[4] = 'o';

  //set the address

  radio.openWritingPipe(address);

  //Set module as transmitter

  radio.stopListening();

}

void loop()

{

  radio.write(&text, sizeof(text));

  delay(1000);

}

/\*

  SerialEvent occurs whenever a new data comes in the hardware serial RX. This

  routine is run between each time loop() runs, so using delay inside loop can

  delay response. Multiple bytes of data may be available.

\*/

void serialEvent() {

  int i=0;

  memset(text, 0, sizeof(text));

  while (Serial.available()) {

    // get the new byte:

    char inChar = (char)Serial.read();

    // if the incoming character is a newline, set a flag so the main loop can

    // do something about it:

    if (inChar == '\n'){

      stringComplete = true;

    }

    else

      text[i++] = inChar; // add it to the inputString:

  }

}

Receiver

//Include Libraries

#include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>

//create an RF24 object

RF24 radio(9, 8);  // CE, CSN

//address through which two modules communicate.

const byte address[6] = "00001";

void setup()

{

  while (!Serial);

    Serial.begin(9600);

  radio.begin();

  //set the address

  radio.openReadingPipe(0, address);

  //Set module as receiver

  radio.startListening();

}

void loop()

{

  //Read the data if available in buffer

  if (radio.available())

  {

    char text[32] = {0};

    radio.read(&text, sizeof(text));

    Serial.println(text);

  }

}