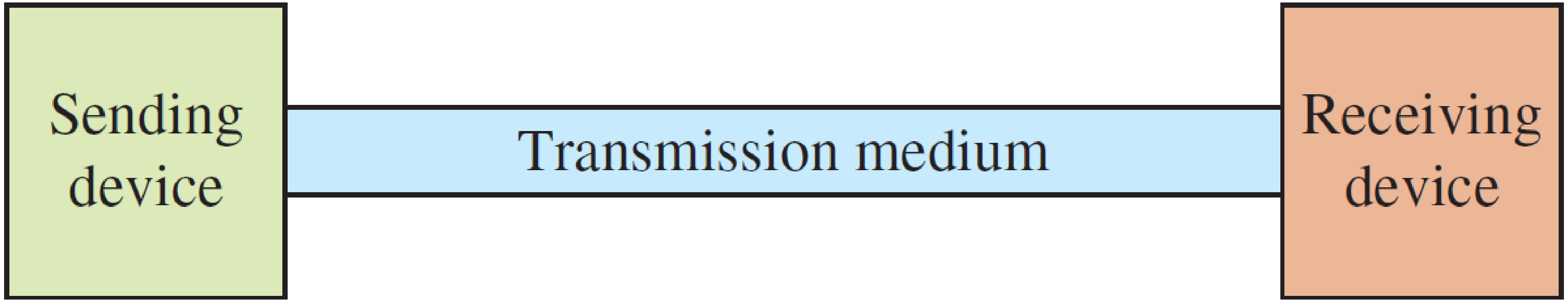
Internet of Things

Python & NodeMCU Serial Communication

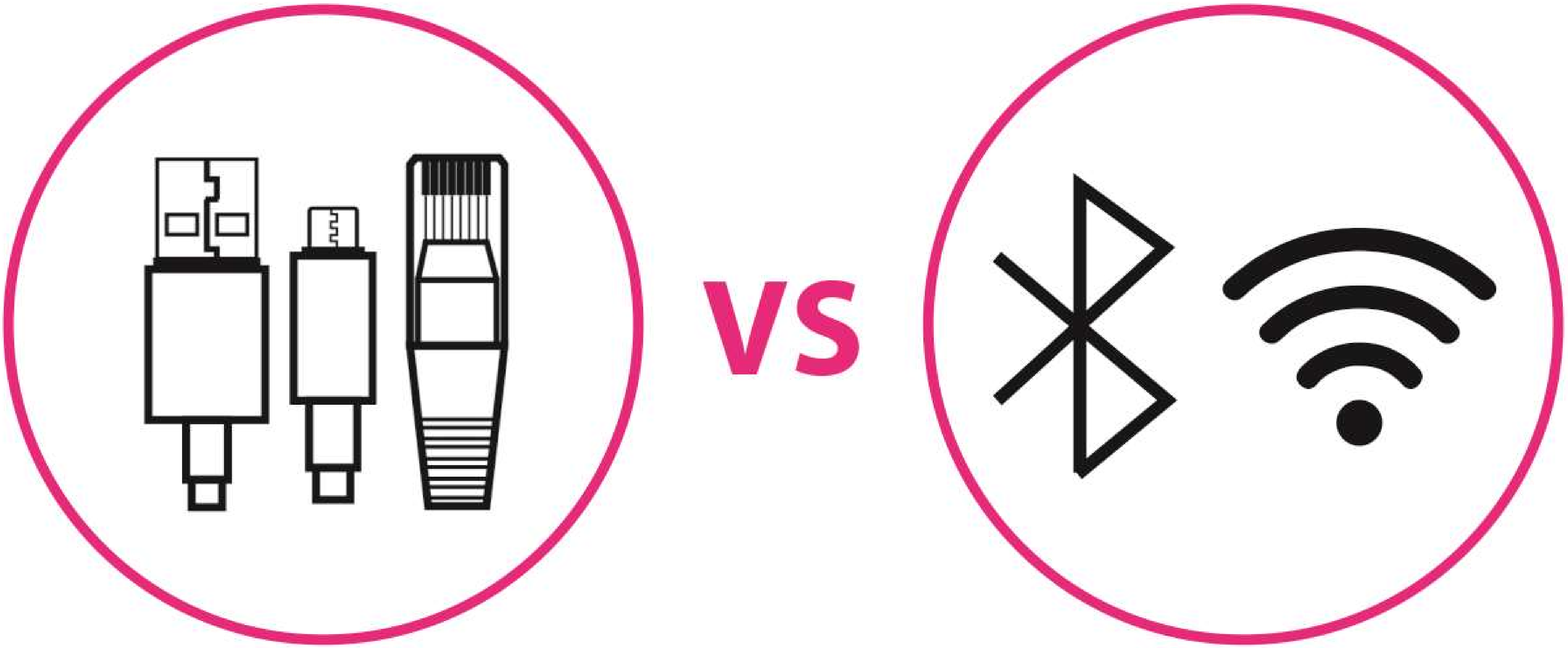
IoT Team, BFCAI

Data Transmission

- All data transmission systems in their most basic form have a sending device at one end and a receiving device at the other.

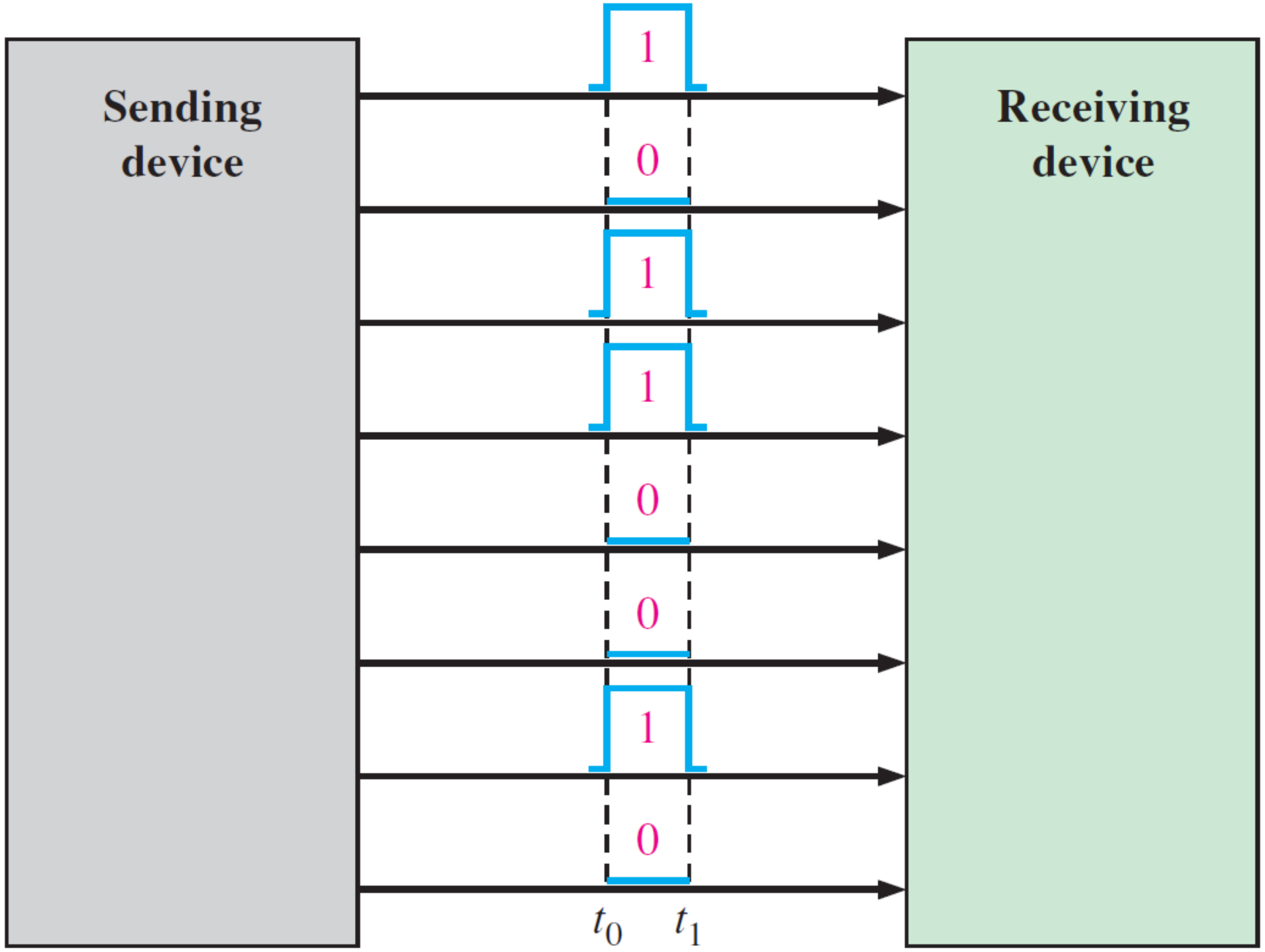


Wired vs. Wireless Communication



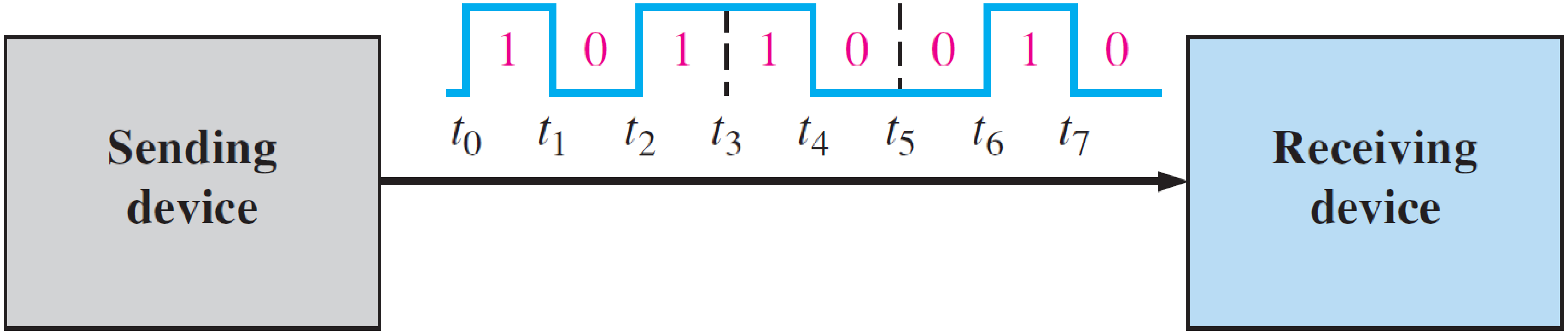
Parallel Communication

- In parallel communication, where many bits are sent at the same time.



Serial Communication

- Serial communication is simply a way to transfer data.
- The data will be sent sequentially, one bit at a time.



UART Protocol

- UART means “Universal Asynchronous Receiver Transmitter”.
- UART represents the hardware circuitry (module) being used for the serial communication.
- UART is sold/shipped as a standalone integrated circuit (IC) or as an internal module within microcontrollers.
- The UART protocol allows you to communicate between 2 boards.
- When you use serial communication between PC and Arduino, you’re using the UART protocol.

UART Protocol: Baud Rate

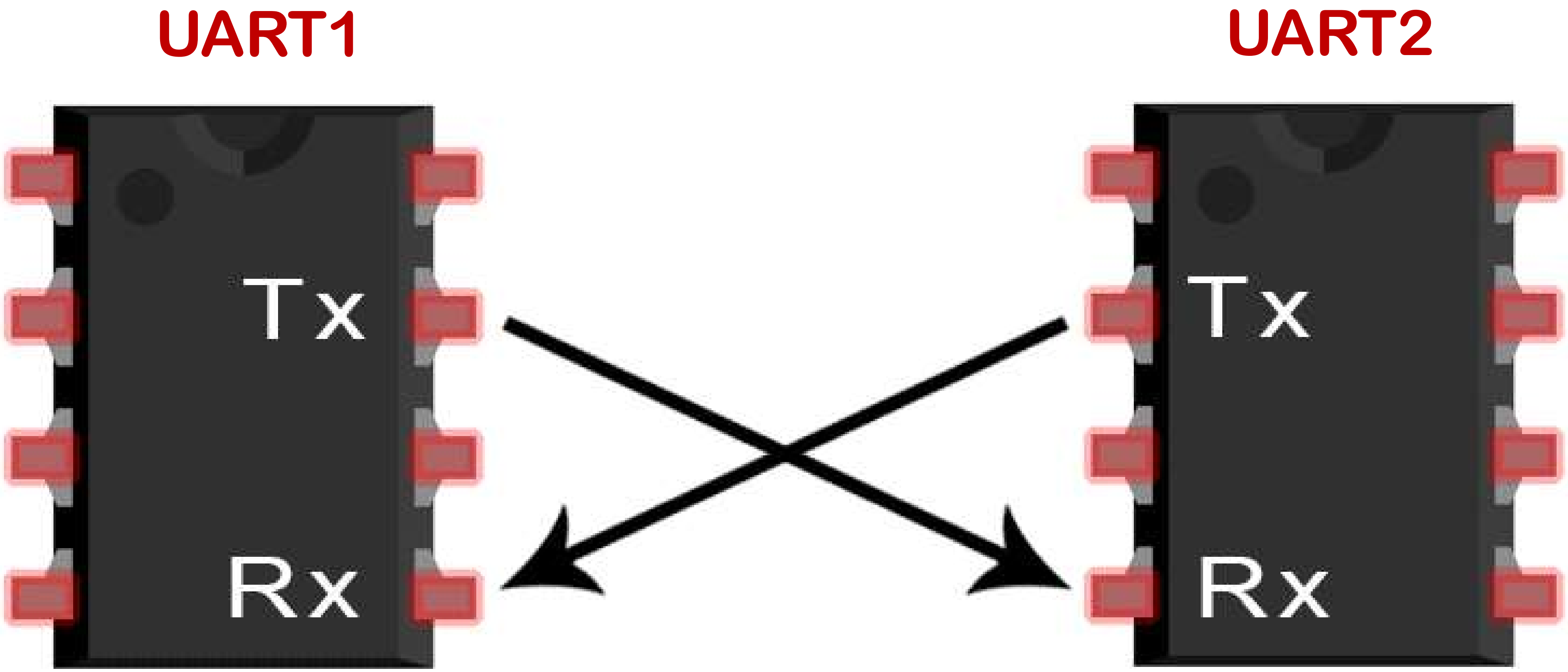
- The baud rate specifies how fast the data is sent over the bus and it is specified in bits-per-second or bps.
- You can actually choose any speed for the baud rate.
- However, there are specific values that are known as industry standards.
- The most common and widely-used standardized value is 9600.

```
Serial.begin(9600);
```

- In the serial port context, “9600 baud” means that the serial port is capable of transferring a maximum of 9600 bits per second.
- Other standard baud rates include: 1200, 2400, 4800, 19200, 38400, 57600 and 115200.

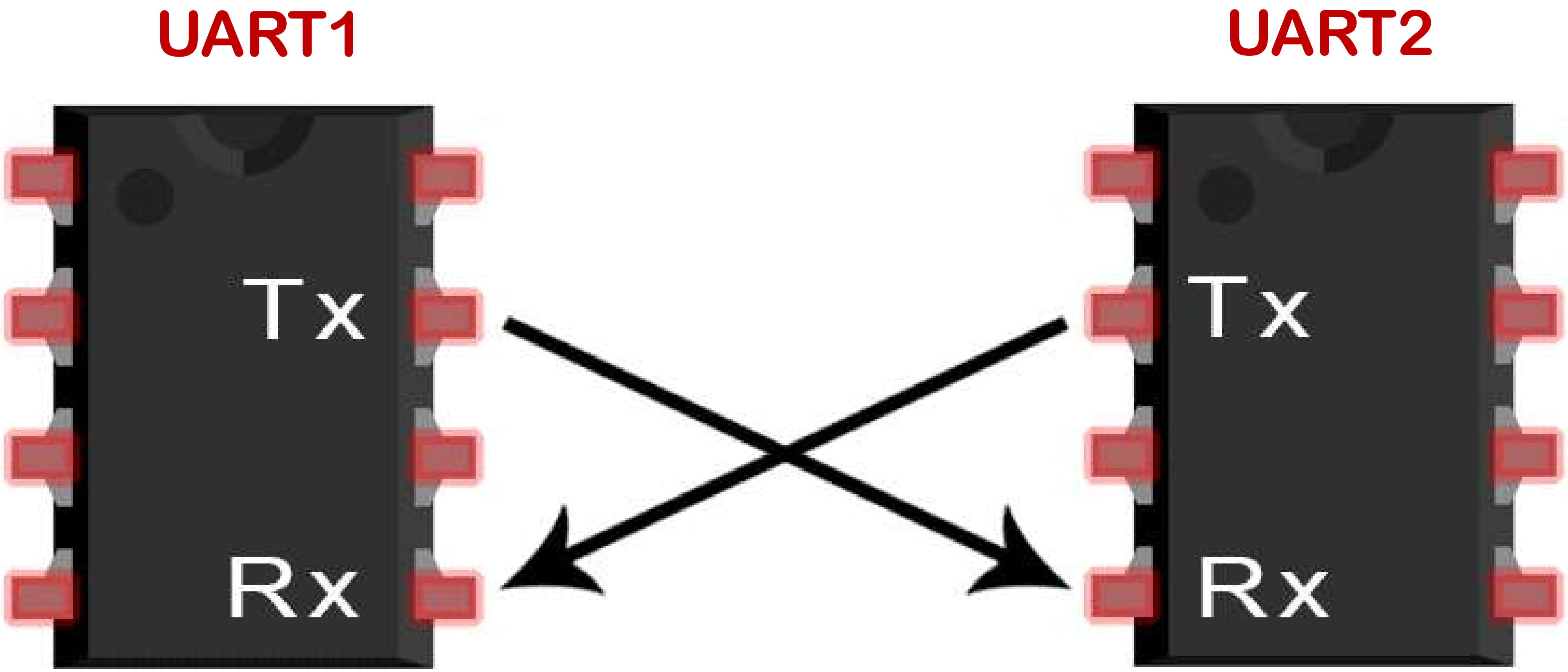
UART Protocol: Transmitter and Receiver

- When device A wants to transmit data to device B, it will share data via its transmitter's pin and device B receiver will receive the sent data.



UART Protocol: Transmitter and Receiver

- In UART communication, both transmitter and receiver must agree on the exact same baud rate for a successful data transmission.



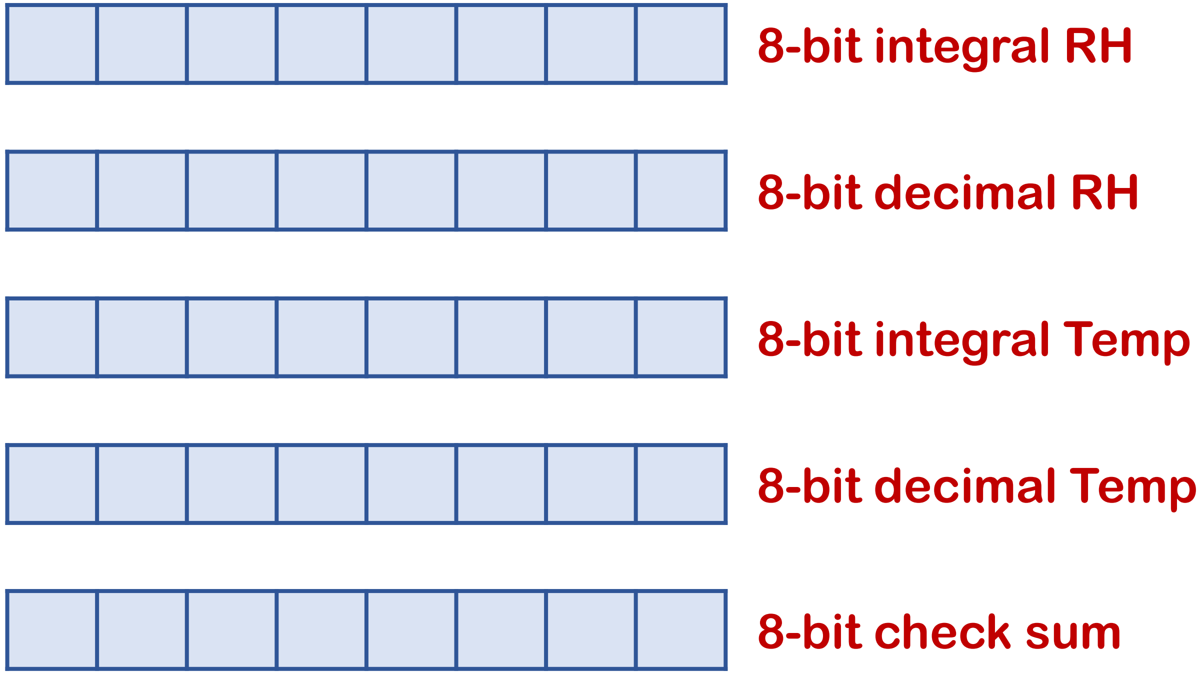
UART Protocol: Data Packet

- The data being transmitted/received in UART serial communication is organized into specific blocks called packets.
- UART packets usually start with “start bit” which is a logic LOW and is used to signal the receiver that there is a new coming packet.
- Data bits are the actual data bits being transmitted to receiver.
- Parity bit allows the receiver to check the correctness of the received data.
- Stop bits are used to signal the end of the data packet being sent.

Start Bit (1 Bit)	Data Bits (5 to 9 Bits)	Parity Bit (0 to 1 Bit)	Stop Bits (1 to 2 Bits)

DHT11: Temperature and Humidity Sensor

- The DHT11 sensor measures humidity and temperature values serially over a single wire.
- It sends a 40-bit data stream containing both temperature and humidity.



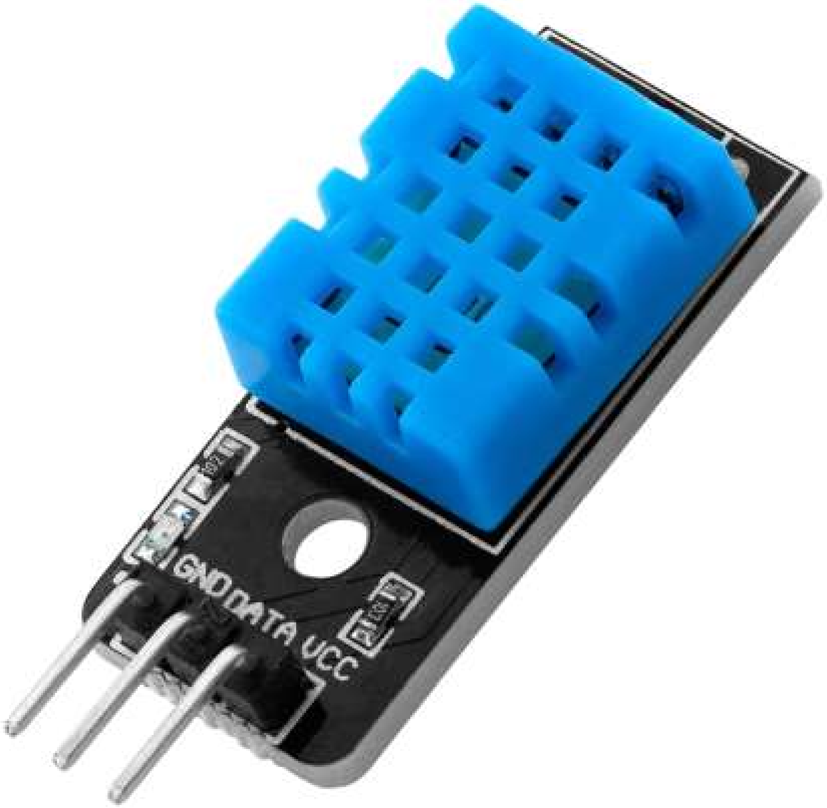
8-bit integral RH

8-bit decimal RH

8-bit integral Temp

8-bit decimal Temp

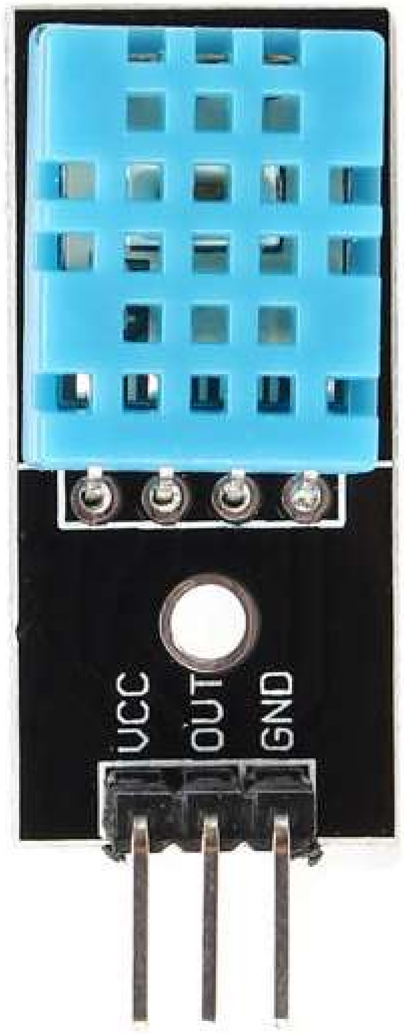
8-bit check sum



DHT11: Specifications

Criteria	Description
Operating Voltage	3.3V to 5.5V
Communication	Serial
Output Signal	Digital
Temperature Range	0°C to 50°C
Temperature Accuracy	±2°C
Humidity Range	20% to 90%
Humidity Accuracy	±5%
Refresh Rate	~ 2 seconds

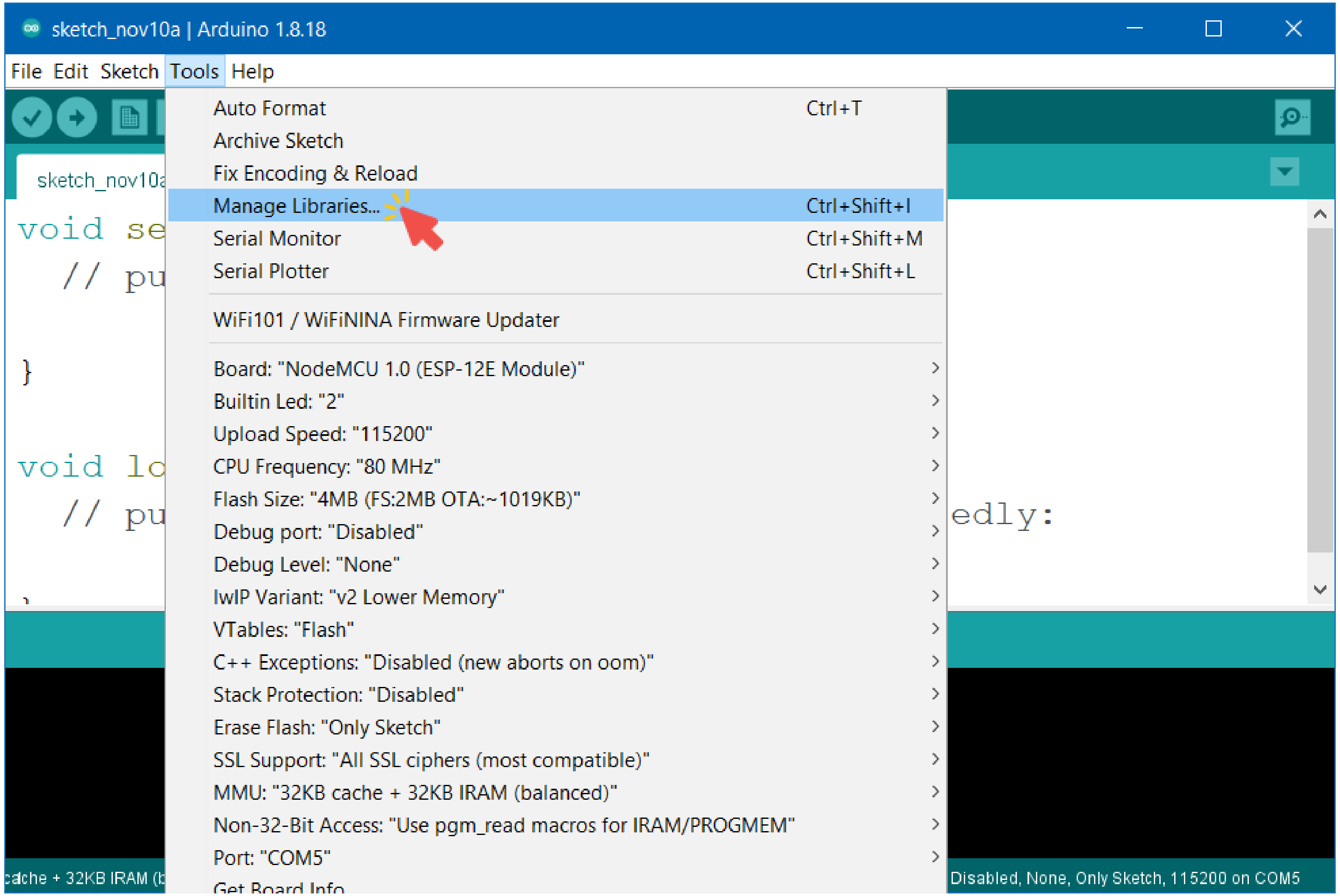
DHT11: Pinout



VCC S GND

DHT11: Installing Library

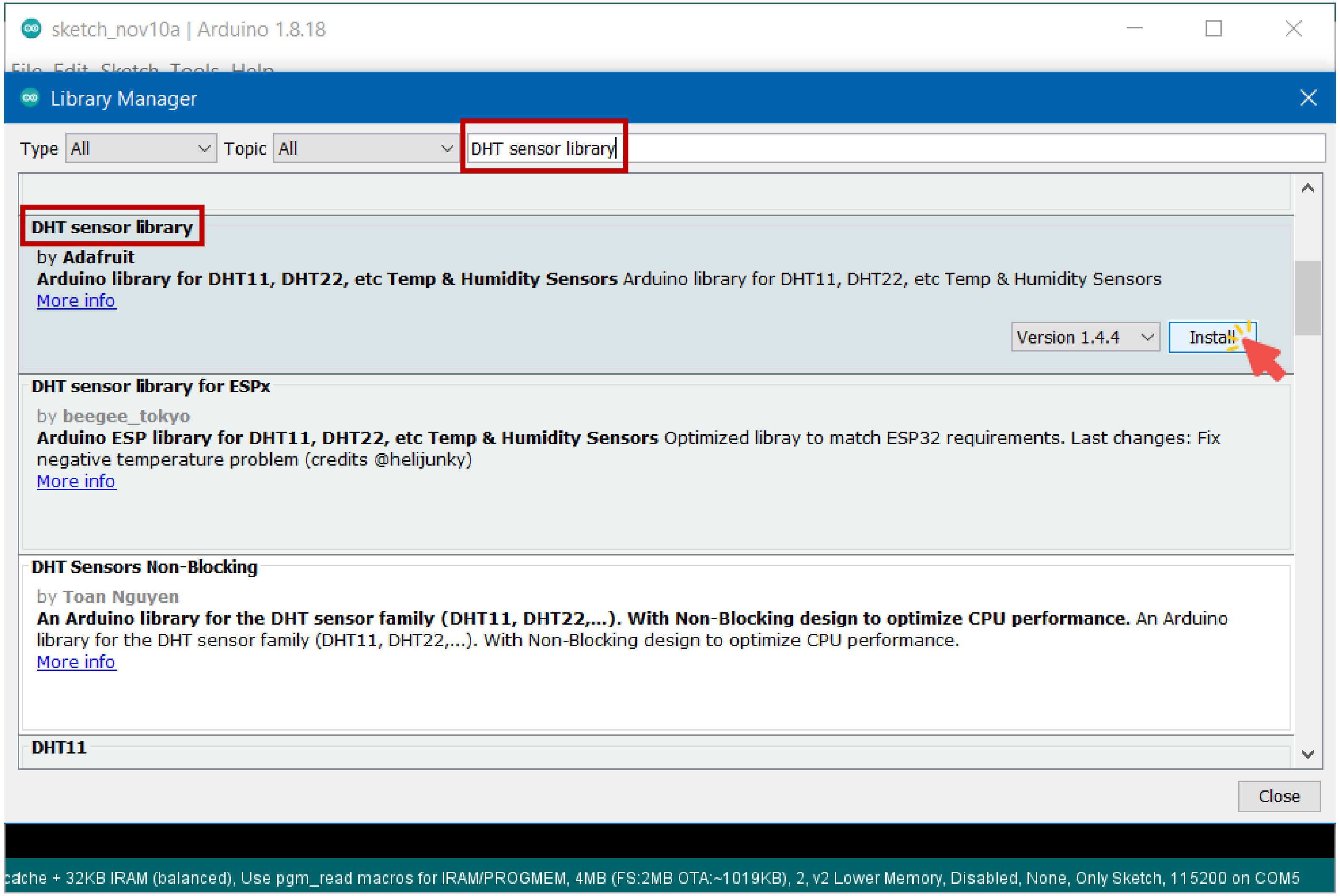
- Go to Tools → Manage Libraries.


sketch_nov10a | Arduino 1.8.18
File Edit Sketch Tools Help
sketch_nov10a
void se
// pu
}
void lo
// pu
edly:
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Get Board Info
cache + 32KB IRAM (b
Disabled, None, Only Sketch, 115200 on COM5

Board: "NodeMCU 1.0 (ESP-12E Module)"
Builtin Led: "2"
Upload Speed: "115200"
CPU Frequency: "80 MHz"
Flash Size: "4MB (FS:2MB OTA:~1019KB)"
Debug port: "Disabled"
Debug Level: "None"
lwIP Variant: "v2 Lower Memory"
VTables: "Flash"
C++ Exceptions: "Disabled (new aborts on oom)"
Stack Protection: "Disabled"
Erase Flash: "Only Sketch"
SSL Support: "All SSL ciphers (most compatible)"
MMU: "32KB cache + 32KB IRAM (balanced)"
Non-32-Bit Access: "Use pgm_read macros for IRAM/PROGMEM"
Port: "COM5"

DHT11: Installing Library

- Search DHT sensor library by Adafruit, and install it.



Type: All

Topic: All

Search: DHT sensor library

DHT sensor library
by **Adafruit**
Arduino library for DHT11, DHT22, etc Temp & Humidity Sensors Arduino library for DHT11, DHT22, etc Temp & Humidity Sensors
More info

DHT sensor library for ESPx
by **beegee_tokyo**
Arduino ESP library for DHT11, DHT22, etc Temp & Humidity Sensors Optimized libray to match ESP32 requirements. Last changes: Fix negative temperature problem (credits @helijunky)
More info

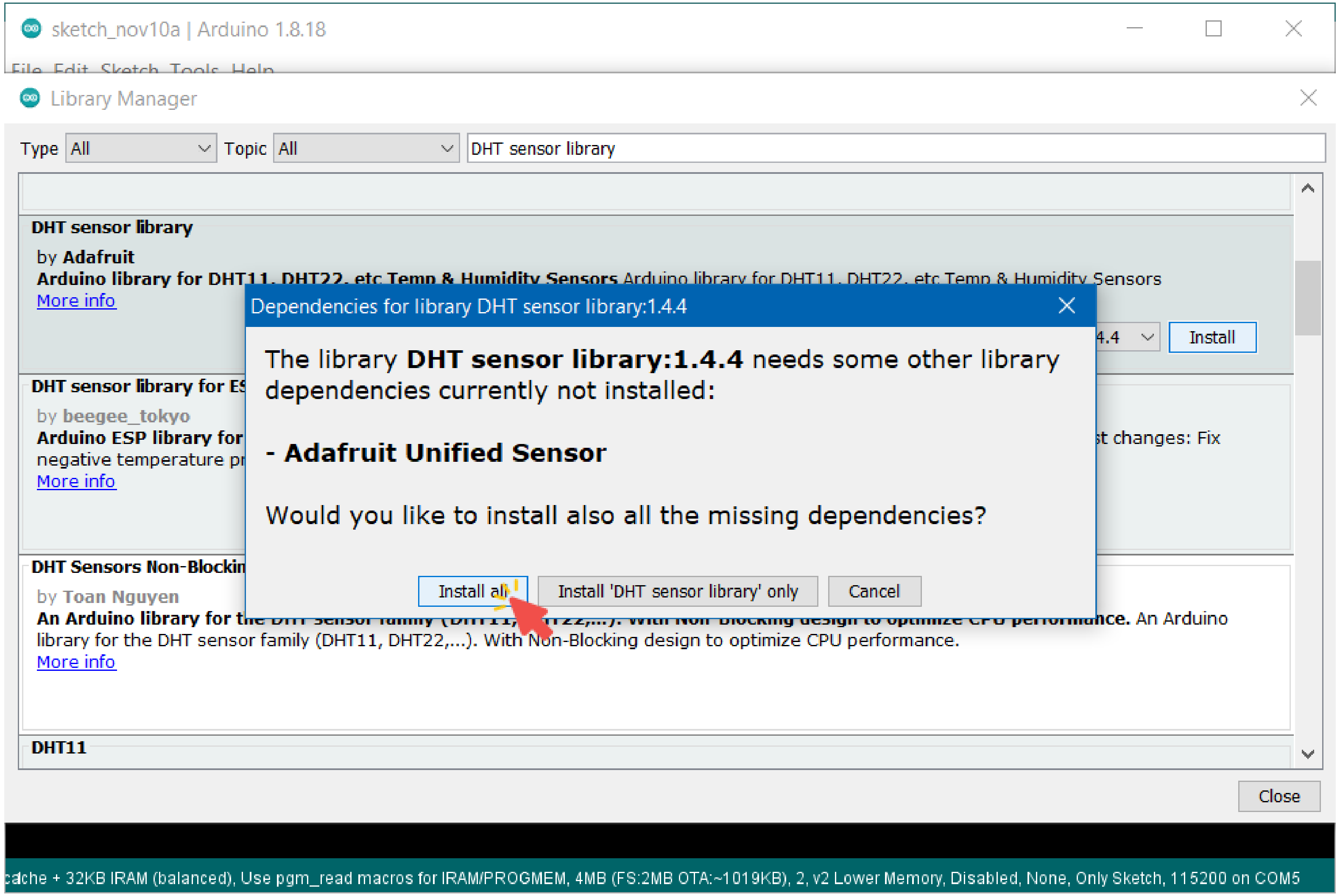
DHT Sensors Non-Blocking
by **Toan Nguyen**
An Arduino library for the DHT sensor family (DHT11, DHT22,...). With Non-Blocking design to optimize CPU performance. An Arduino library for the DHT sensor family (DHT11, DHT22,...). With Non-Blocking design to optimize CPU performance.
More info

DHT11

```
ache + 32KB IRAM (balanced), Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM5
```

DHT11: Installing Library

- Click Install all, if this message appears.



DHT sensor library

DHT sensor library
by **Adafruit**
Arduino library for DHT11, DHT22, etc Temp & Humidity Sensors Arduino library for DHT11, DHT22, etc Temp & Humidity Sensors
More info

DHT sensor library for ES
by **beegee_tokyo**
Arduino ESP library for ...t changes: Fix
negative temperature pr
More info

DHT Sensors Non-Blockin
by **Toan Nguyen**
An Arduino library for the DHT sensor family (DHT11, DHT22,...). With Non-Blocking design to optimize CPU performance. An Arduino library for the DHT sensor family (DHT11, DHT22,...). With Non-Blocking design to optimize CPU performance.
More info

DHT11

Dependencies for library DHT sensor library:1.4.4

The library **DHT sensor library:1.4.4** needs some other library dependencies currently not installed:

- Adafruit Unified Sensor

Would you like to install also all the missing dependencies?

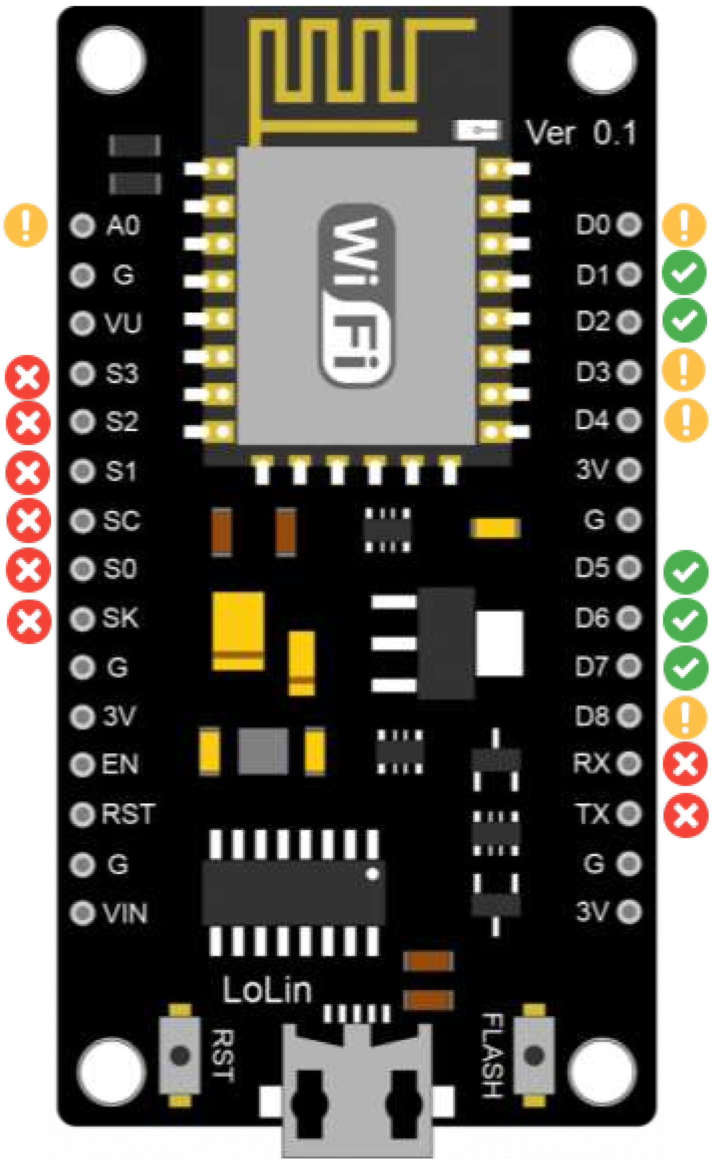
```
cache + 32KB IRAM (balanced), Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM5
```

DHT11: Hardware Components

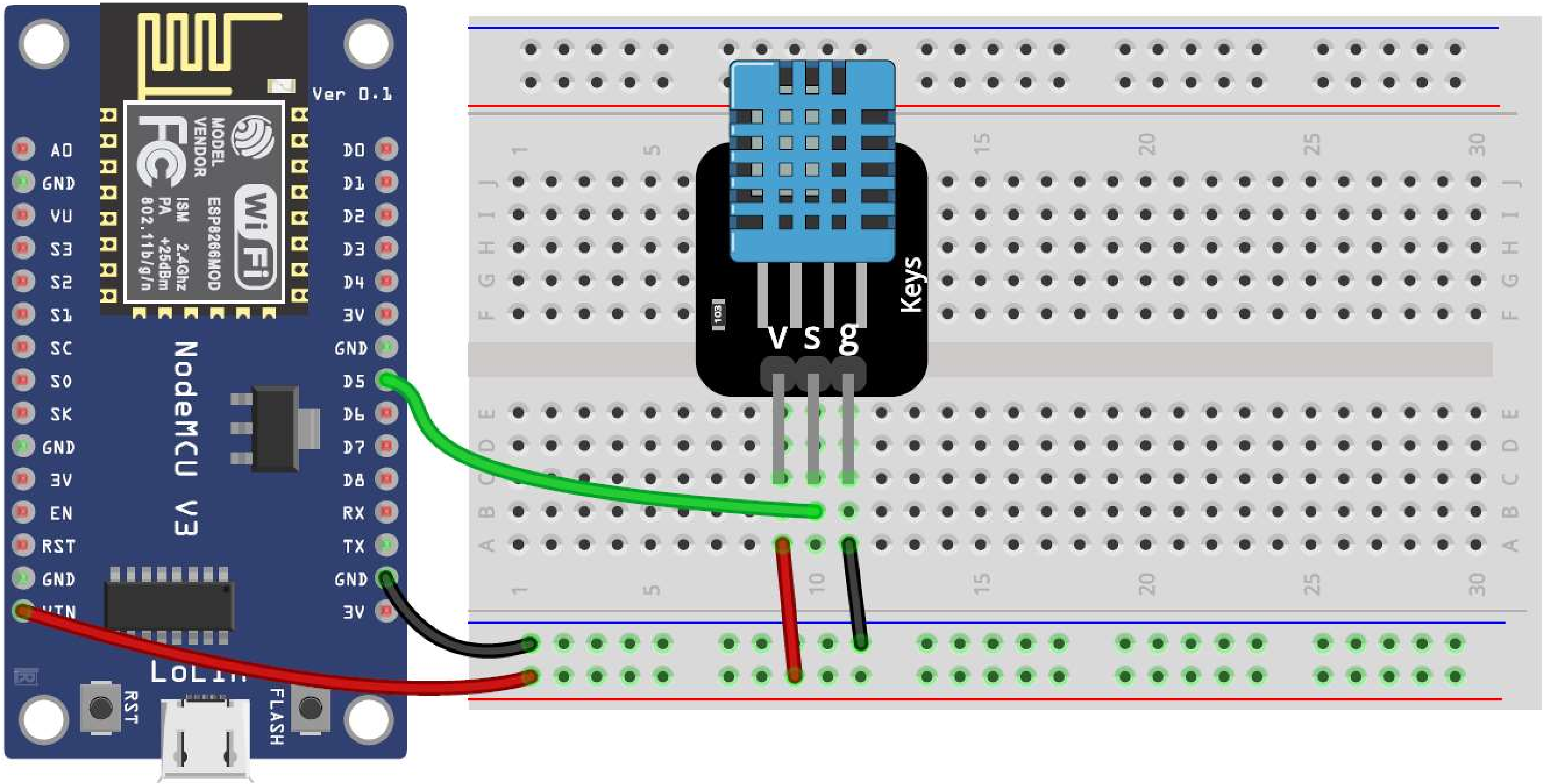
- NodeMCU ESP8266
- DHT11 Sensor
- Jumpers
- Breadboard

DHT11: NodeMCU ESP8266 Pinout

PIN	GPIO	Why Not Safe?
D0	GPIO16	HIGH at boot Used to wake up from deep sleep
D1	GPIO5	-
D2	GPIO4	-
D3	GPIO0	Connected to FLASH button Boot fails if pulled LOW
D4	GPIO2	HIGH at boot Boot fails if pulled LOW
D5	GPIO14	-
D6	GPIO12	-
D7	GPIO13	-
D8	GPIO15	Required for boot Boot fails if pulled HIGH

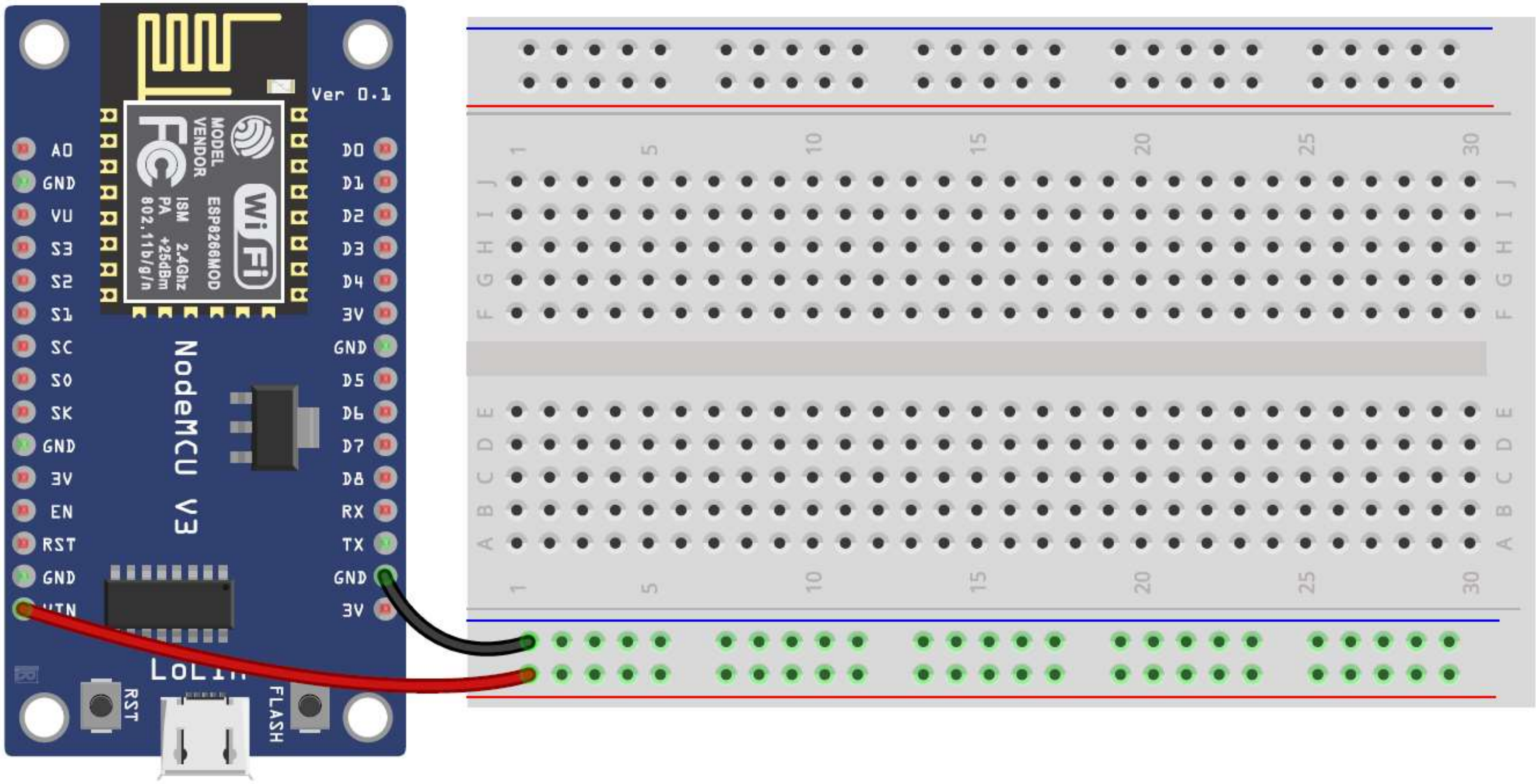


DHT11: Circuit



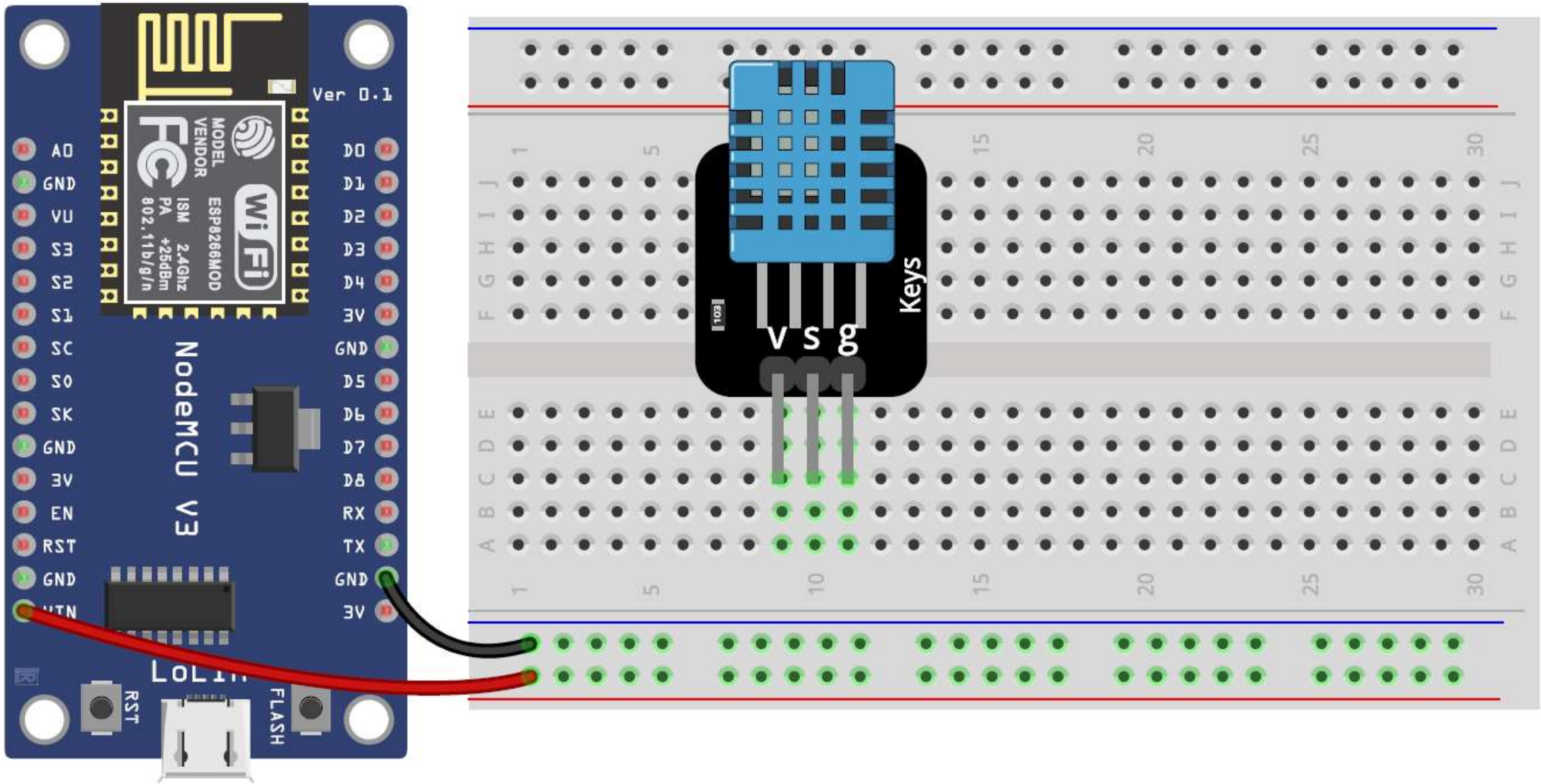
DHT11: Steps

1. Connect breadboard power (+) and ground (-) rails to NodeMCU VIN and ground (GND), respectively.



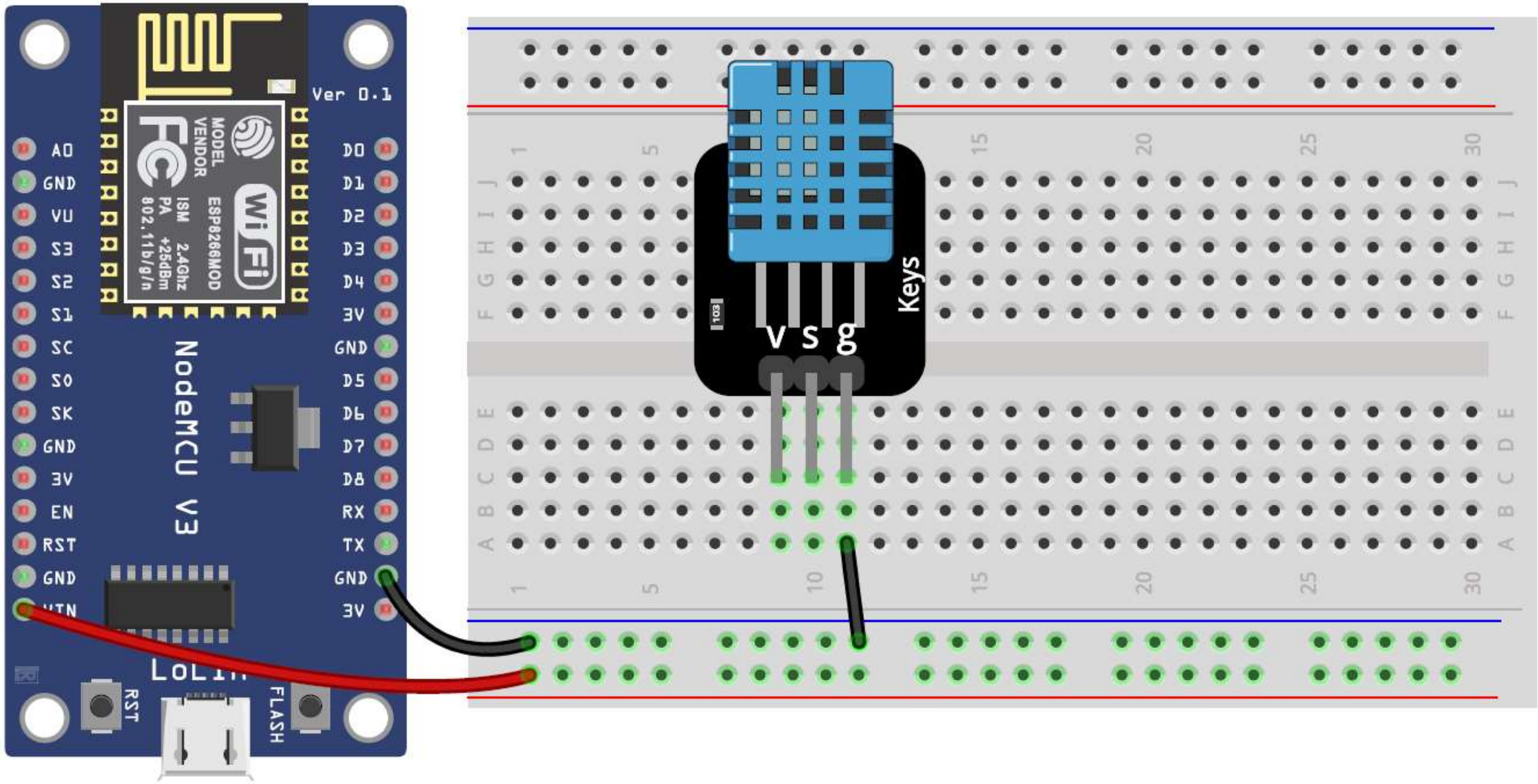
DHT11: Steps

2. Plug the DHT11 sensor into the breadboard.



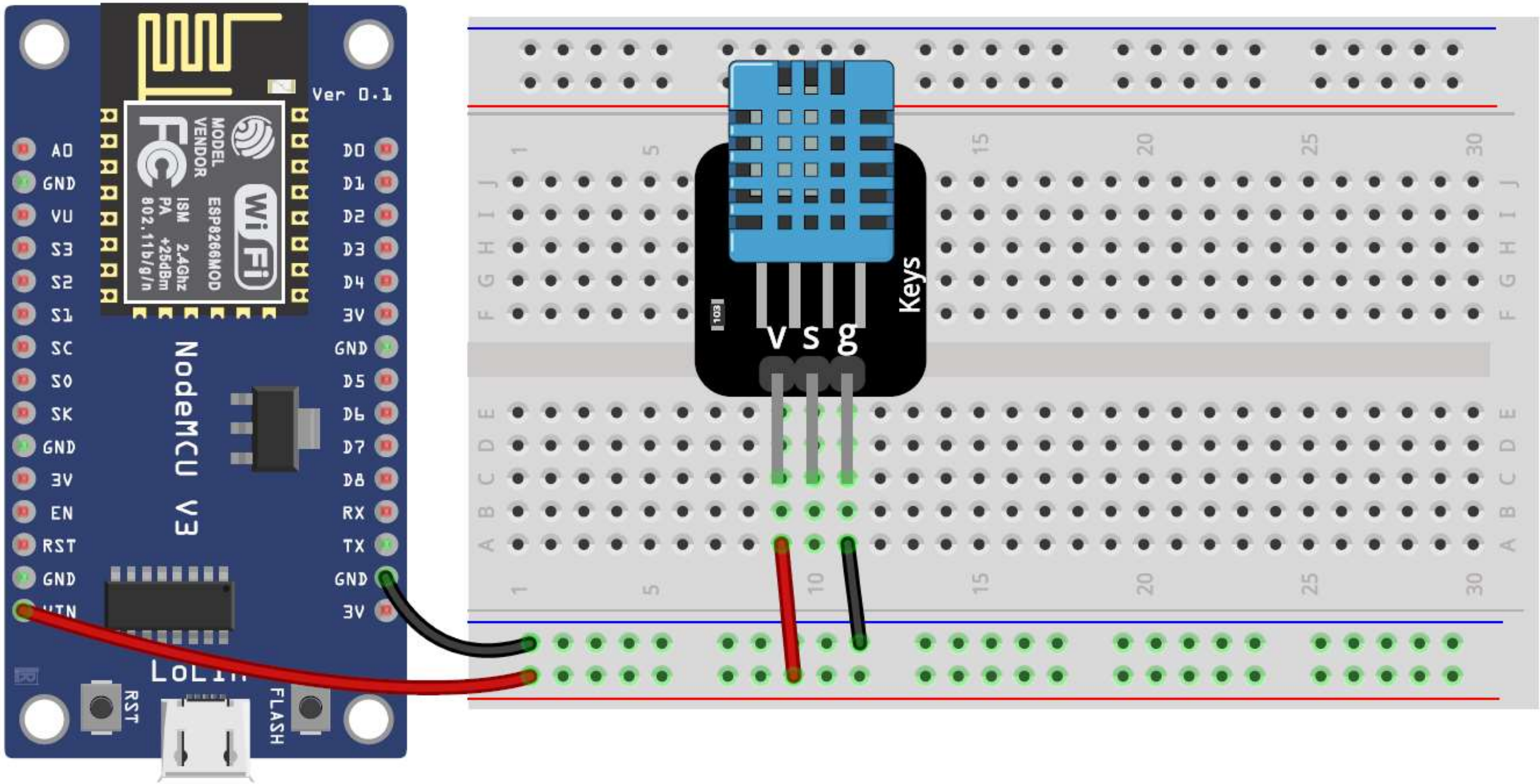
DHT11: Steps

3. The sensor GND pin connects to the ground on NodeMCU.



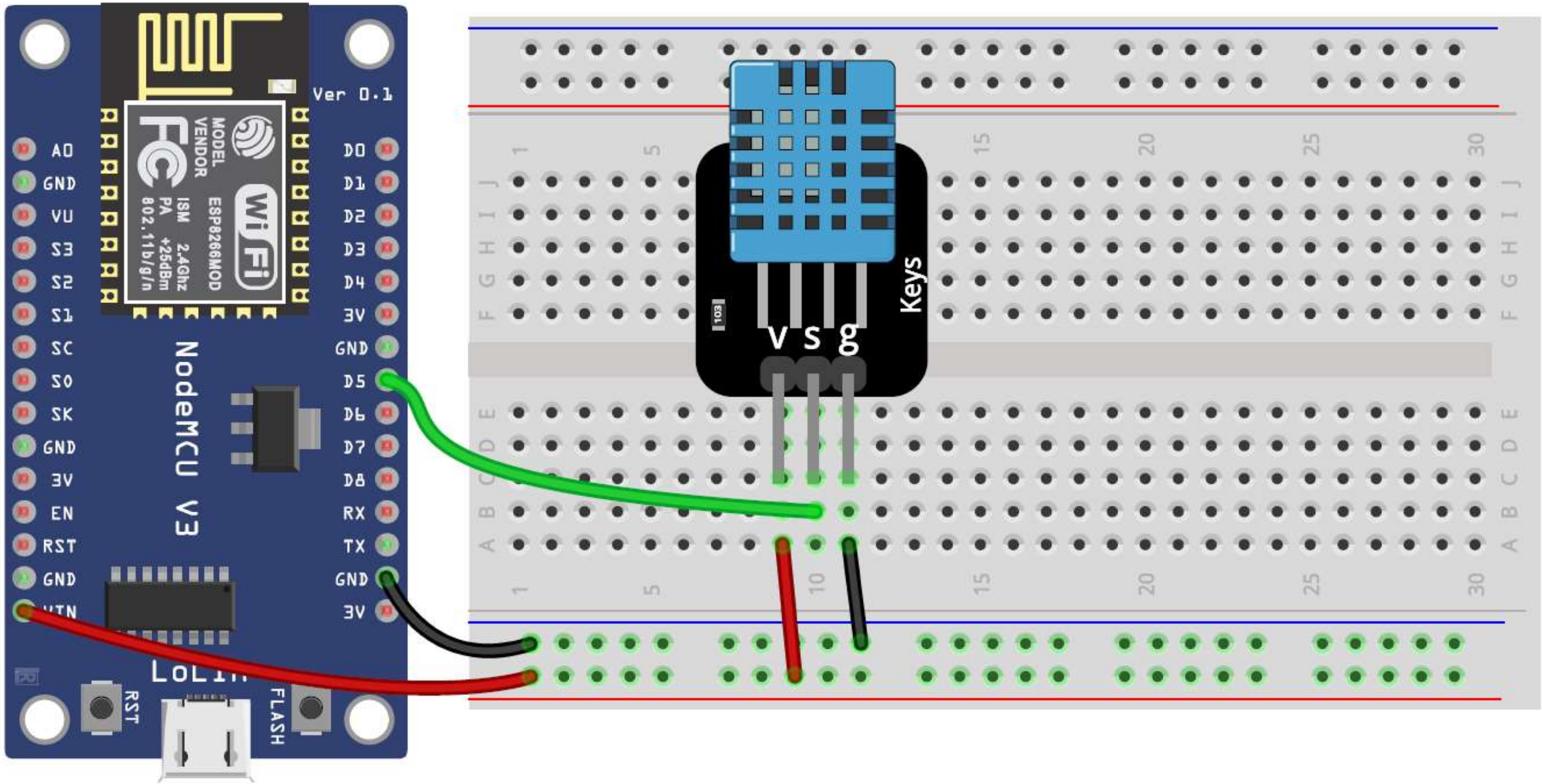
DHT11: Steps

4. The sensor Power pin connects to the VCC on NodeMCU.



DHT11: Steps

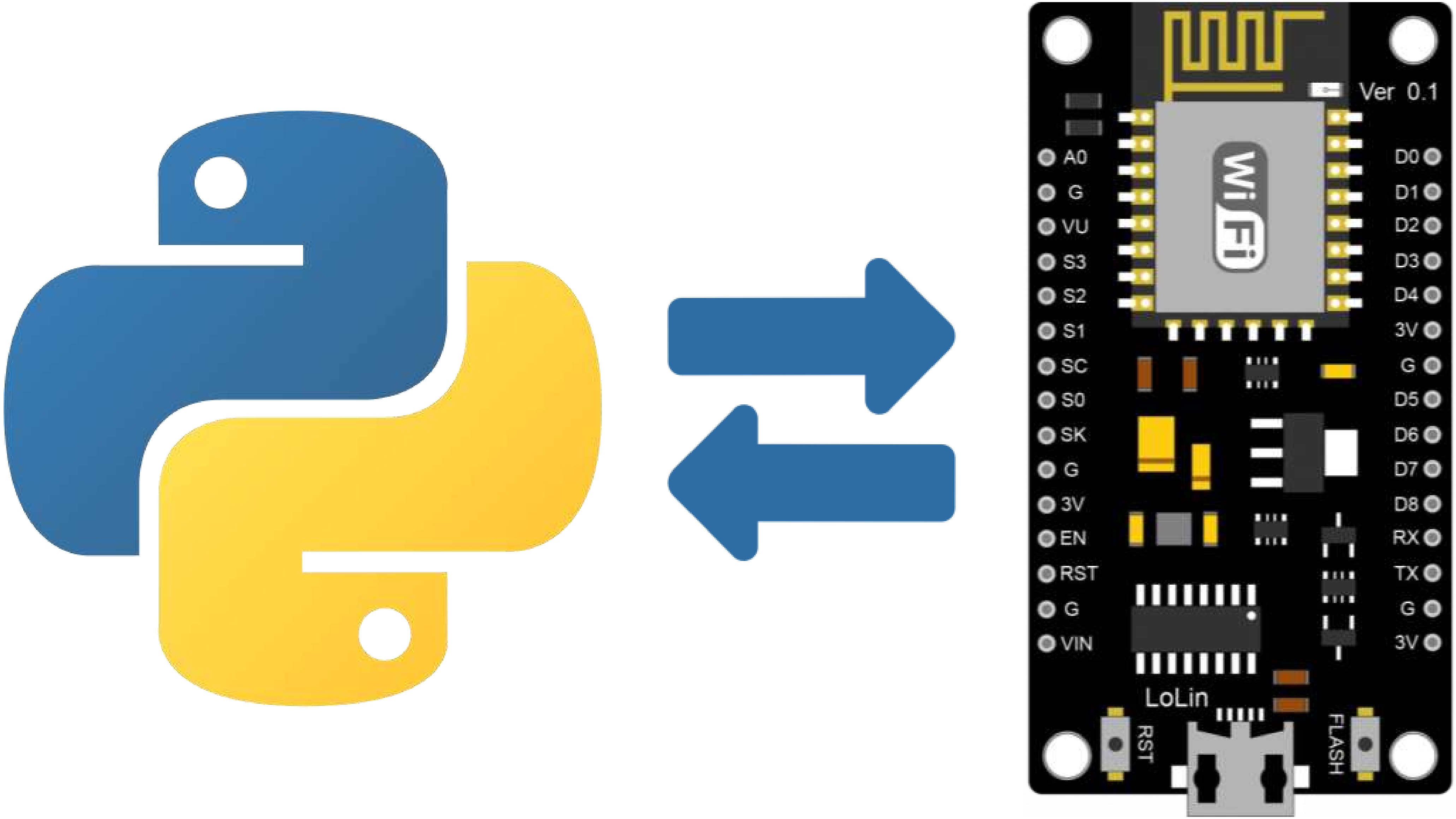
5. Wire up the sensor Data pin to the analog pin D5 on NodeMCU.



DHT11: Code

```
#include "DHT.h"                                        // Import DHT library
#define DHT_PIN D5                                      // Digital pin connected to the DHT sensor
DHT dht(DHT_PIN, DHT11);                                // Initialize DHT sensor

void setup() {
  Serial.begin(9600);                                   // Start serial monitor
  dht.begin();                                          // Start DHT sensor
}

void loop() {
  delay(2000);                                          // Wait a few seconds between measurements

  float h = dht.readHumidity();                         // Read humidity
  float t = dht.readTemperature();                      // Read temperature as Celsius

  // Check if any reads failed (to try later)
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor.");
    return;
  }

  // Print temperature
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print("°C ");

  // Print humidity
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.println("%");
}
```

NodeMCU & Python Serial Communication

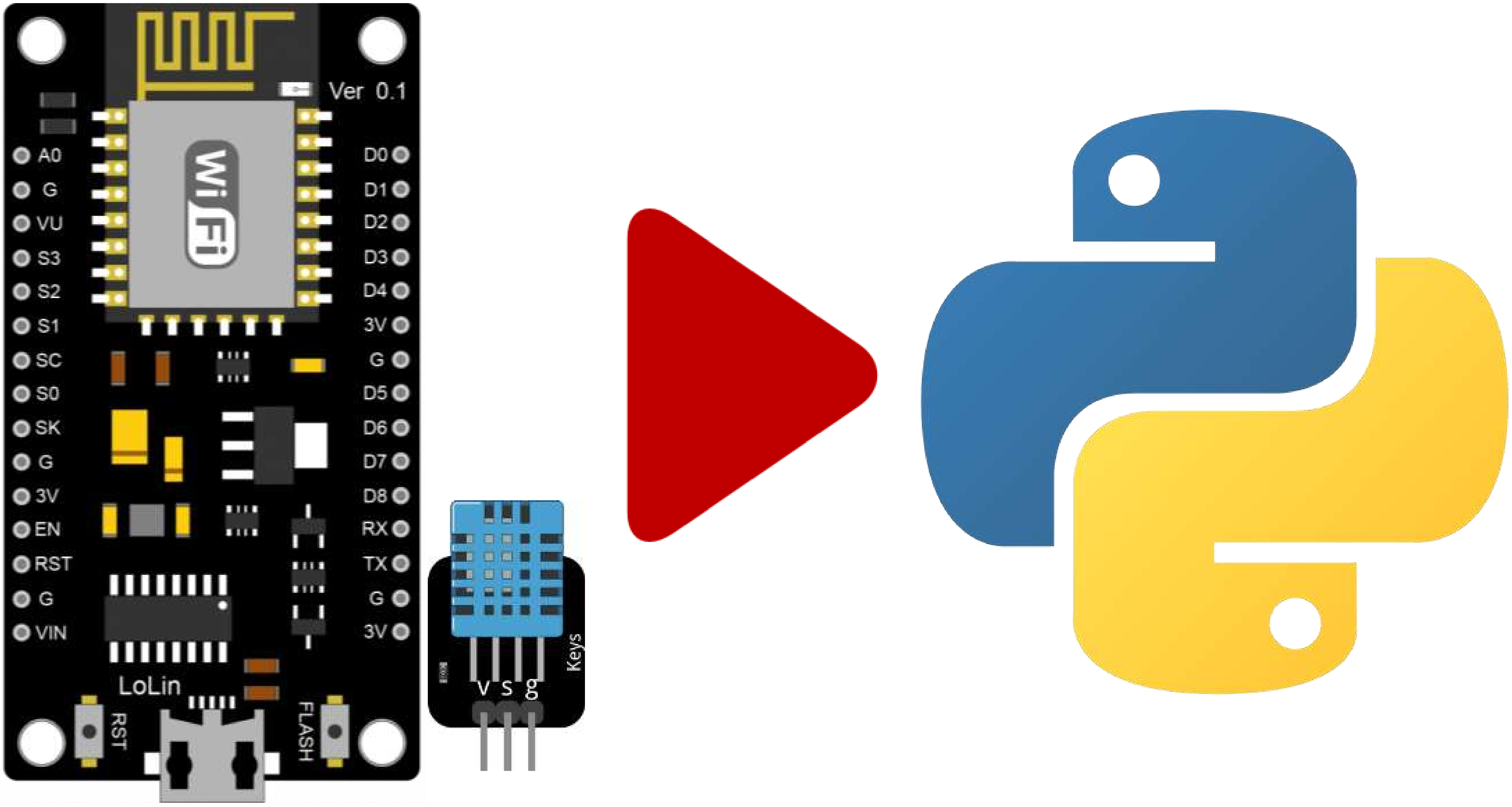


NodeMCU & Python Serial Communication

- The objective of this part is to establish a serial connection between a Python program and an Arduino/NodeMCU/ESP-32 program.
- In the Python program, we will use the `PySerial` module to be able to establish the serial connection.
- The easiest way to install `PySerial` is by using `pip`.

  ```
  >> pip install pyserial
  ```
- We will need to know the port and the value of baud rate, to be used later in the Python program.

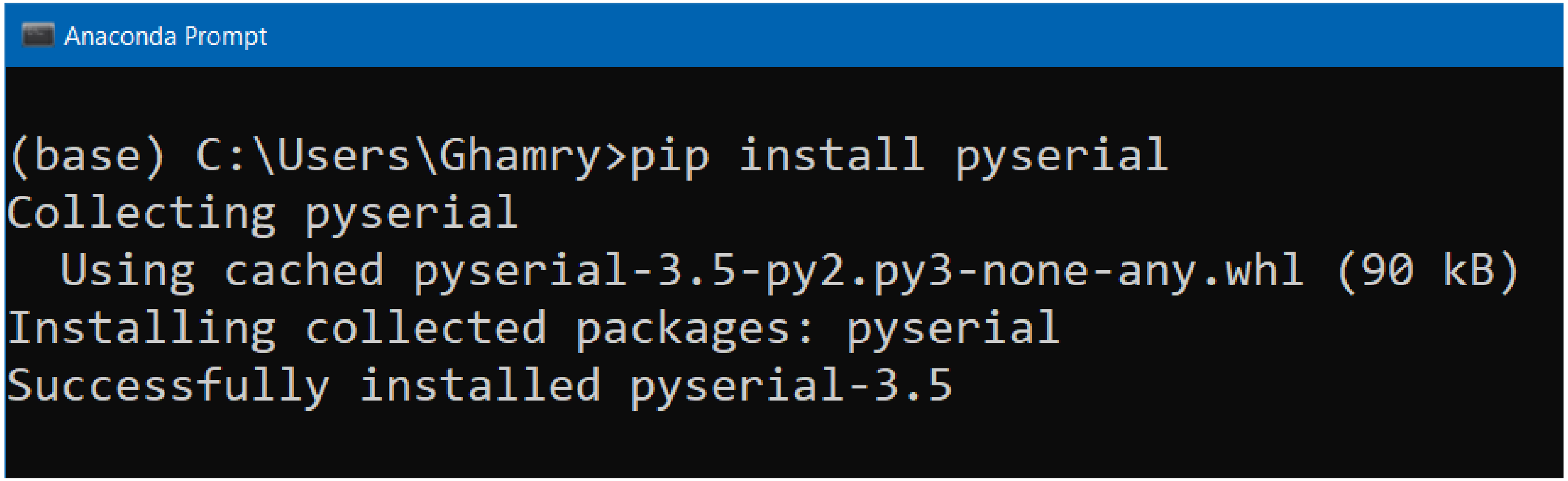
NodeMCU & Python Serial Communication



NodeMCU & Python Serial Communication: Installing PySerial

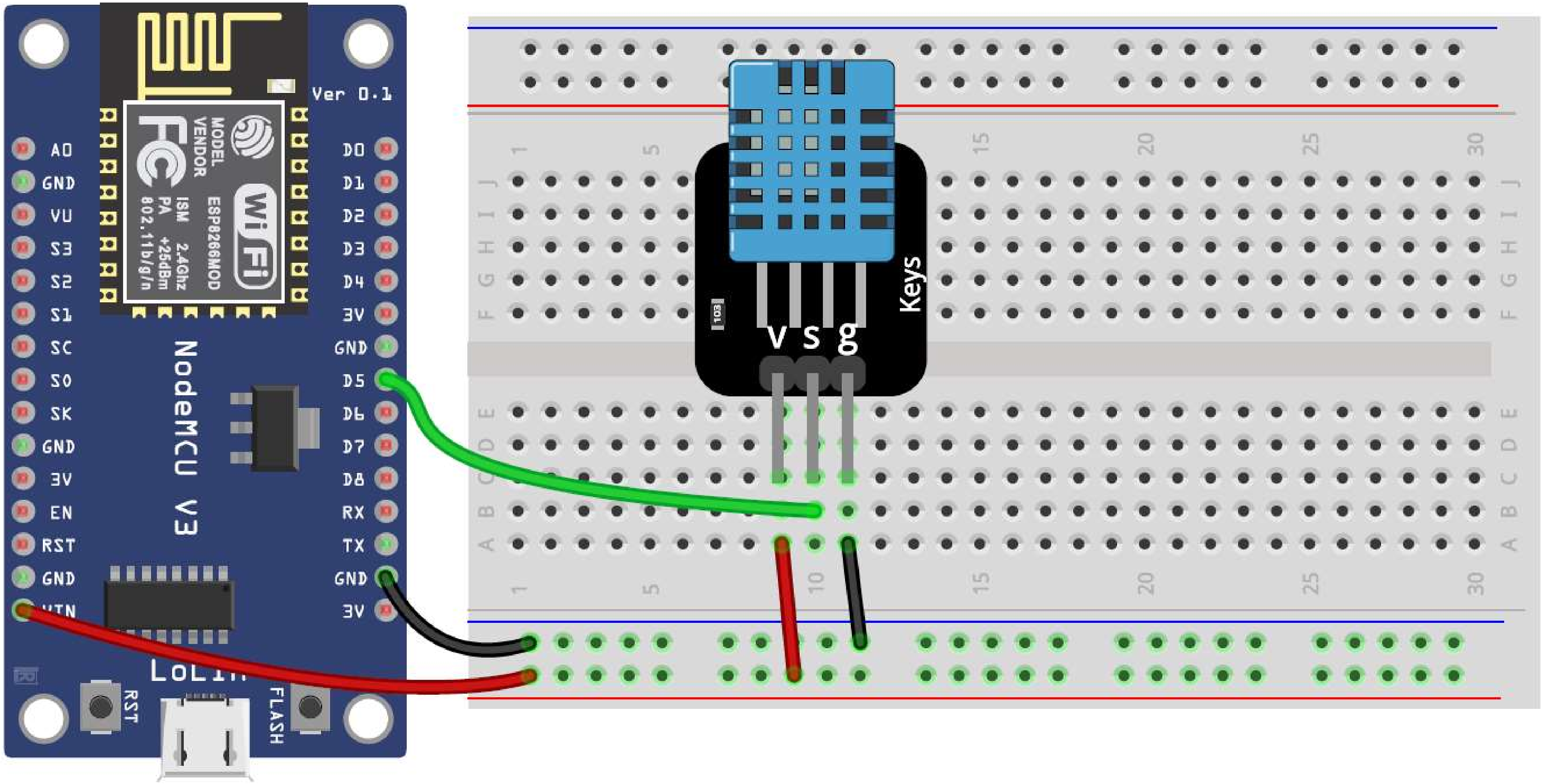
- To establish a serial connection between a Python program and an NodeMCU program, you can use the PySerial library, which allows communication with serial ports.

  `>> pip install pyserial`



```
(base) C:\Users\Ghamry>pip install pyserial
Collecting pyserial
  Using cached pyserial-3.5-py2.py3-none-any.whl (90 kB)
Installing collected packages: pyserial
Successfully installed pyserial-3.5
```

NodeMCU & Python Serial Communication: Circuit



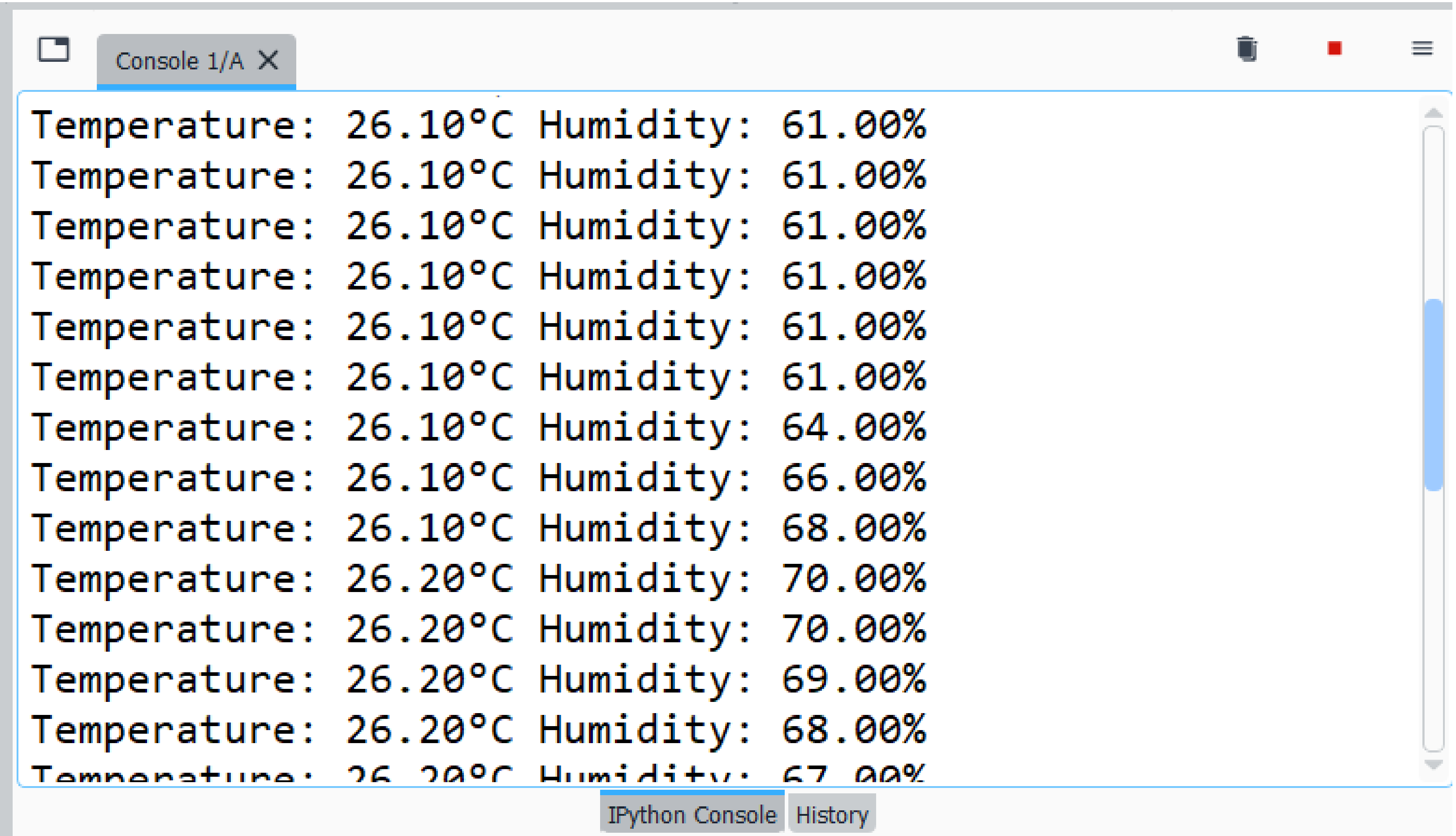
NodeMCU & Python Serial Communication: NodeMCU Program

```
#include "DHT.h"                                        // Import DHT library
#define DHT_PIN D5                                      // Digital pin connected to the DHT sensor
DHT dht(DHT_PIN, DHT11);                                // Initialize DHT sensor

void setup() {
  Serial.begin(9600);                                   // Start serial monitor
  dht.begin();                                          // Start DHT sensor
}

void loop() {
  delay(2000);                                          // Wait a few seconds between measurements

  float h = dht.readHumidity();                         // Read humidity
  float t = dht.readTemperature();                      // Read temperature as Celsius

  // Check if any reads failed (to try later)
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor.");
    return;
  }

  // Print temperature
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print("°C ");

  // Print humidity
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.println("%");
}
```

NodeMCU & Python Serial Communication: Python Program

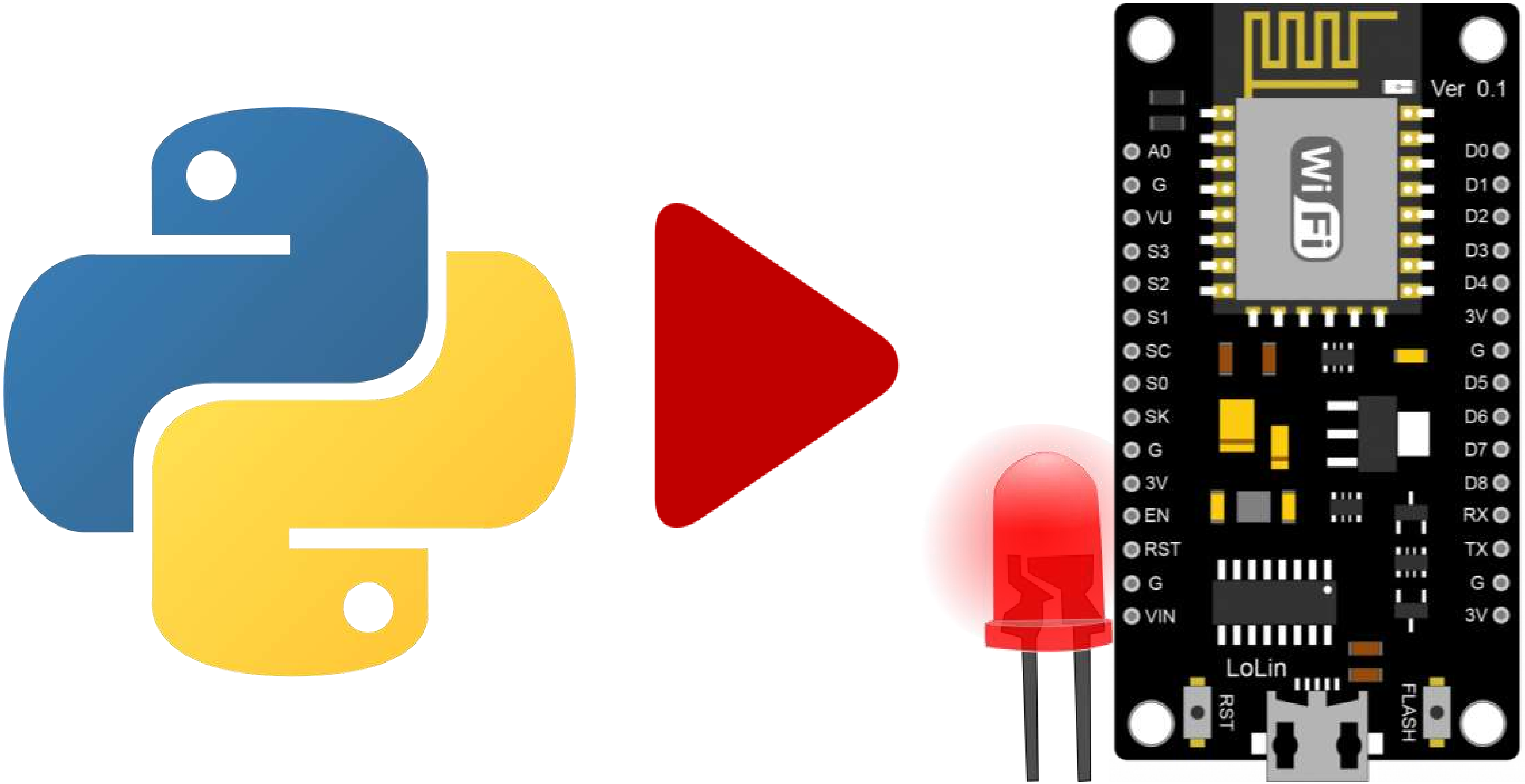
```python
# Import the PySerial library for serial communication
import serial

# Initialize serial communication
ser = serial.Serial('COM5', 9600)

try:
    while True:
        # Check if there is data available in the input buffer
        if ser.in_waiting > 0:
            # Read all bytes until a newline character is detected
            line = ser.readline()

            # Decode the bytes into a UTF-8 string
            line = line.decode('utf-8', errors='ignore')

            # Remove whitespaces from the beginning and the end
            line = line.strip()

            # Print data
            print(line)
except:
    # Close the serial connection
    ser.close()
    print("Serial connection closed.")
```

NodeMCU & Python Serial Communication: Output

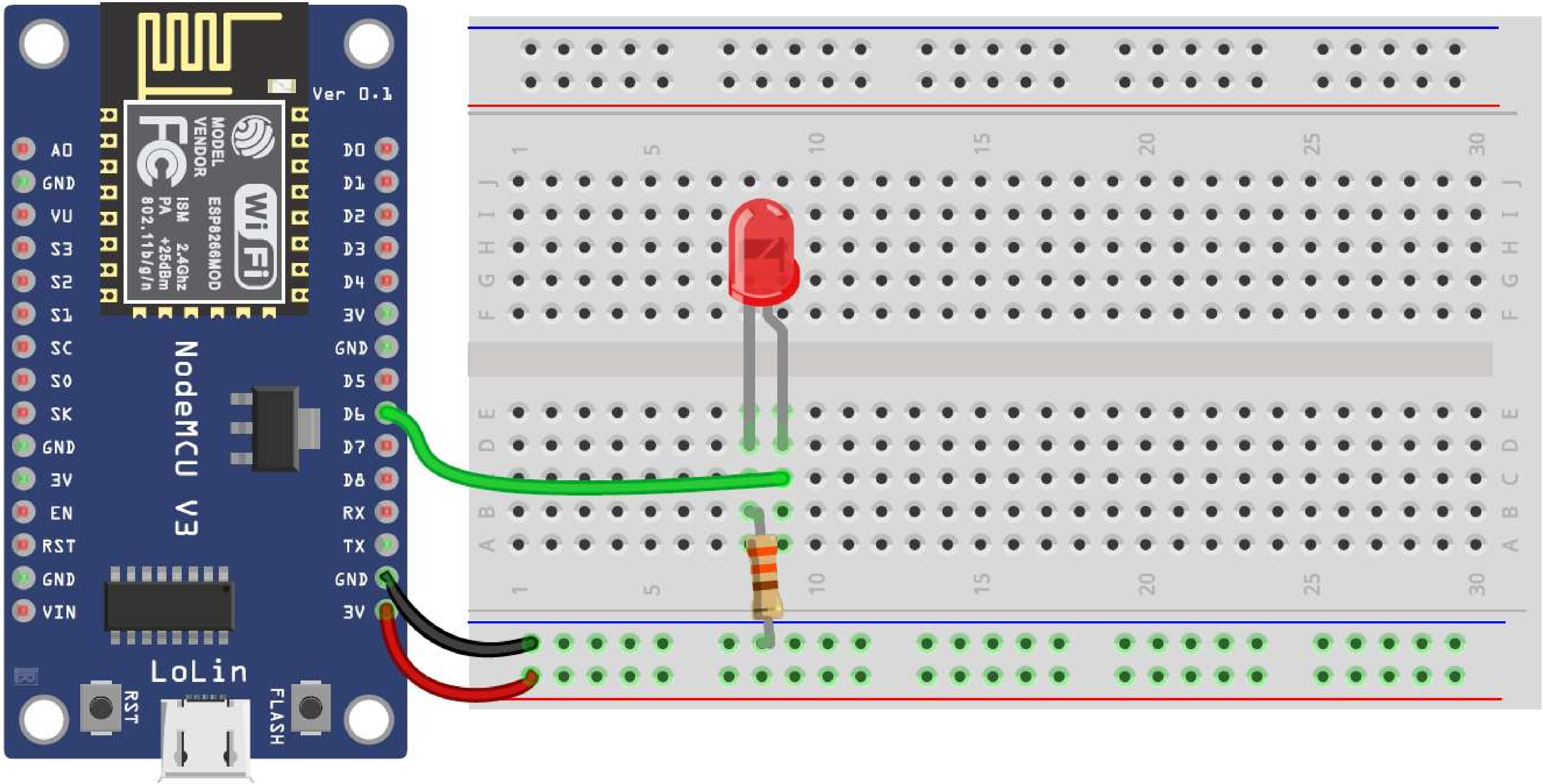


```
Temperature: 26.10°C Humidity: 61.00%
Temperature: 26.10°C Humidity: 61.00%
Temperature: 26.10°C Humidity: 61.00%
Temperature: 26.10°C Humidity: 61.00%
Temperature: 26.10°C Humidity: 61.00%
Temperature: 26.10°C Humidity: 61.00%
Temperature: 26.10°C Humidity: 64.00%
Temperature: 26.10°C Humidity: 66.00%
Temperature: 26.10°C Humidity: 68.00%
Temperature: 26.20°C Humidity: 70.00%
Temperature: 26.20°C Humidity: 70.00%
Temperature: 26.20°C Humidity: 69.00%
Temperature: 26.20°C Humidity: 68.00%
Temperature: 26.20°C Humidity: 67.00%
```

Python & NodeMCU Serial Communication



Python & NodeMCU Serial Communication: Circuit



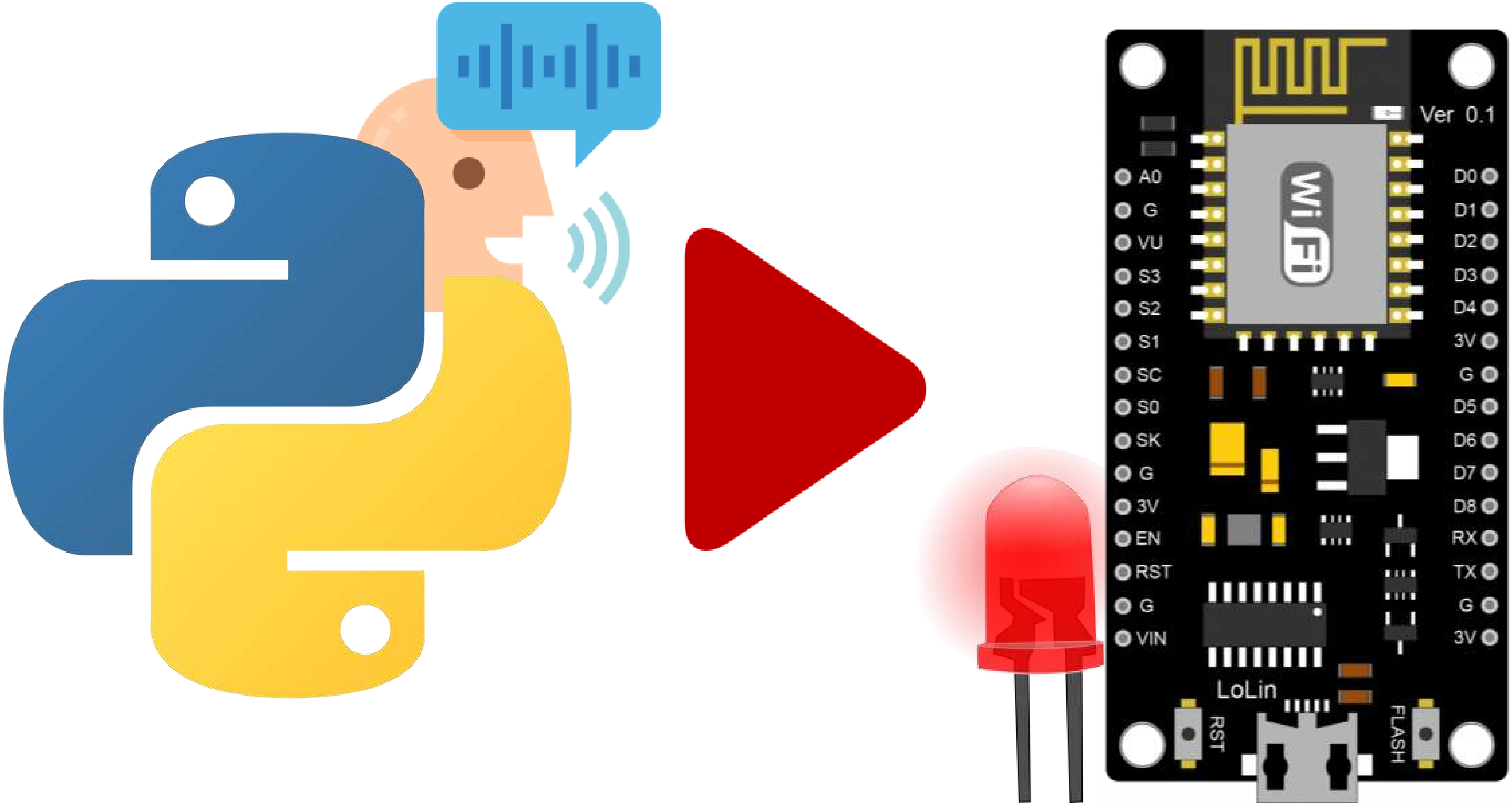
Python & NodeMCU Serial Communication: Python Program

```python
# Import the PySerial library for serial communication
import serial

# Initialize serial communication
ser = serial.Serial('COM5', 9600)

try:
    while True:
        # Get command from the user
        cmd = input('Enter the command: ')

        # Send command to NodeMCU
        ser.write(cmd.encode())
except:
    # Close the serial connection
    ser.close()
    print("Serial connection closed.")
```

Python & NodeMCU Serial Communication: NodeMCU Program

```
#define LED_PIN D6                        // Define LED pin

void setup() {
  Serial.begin(9600);                     // Start serial monitor
  pinMode(LED_PIN, OUTPUT);               // Initialize the pin D6 as an output
}

void loop() {
  // Read the incoming byte if available
  if(Serial.available()){                 // Check if there is a message available
    char cmd = Serial.read();             // Read the incoming byte

    if(cmd == '1')                        // If command is '1'
      digitalWrite(LED_PIN, HIGH);        // Turn on LED
    else if(cmd == '0')                   // If command is '0'
      digitalWrite(LED_PIN, LOW);         // Turn off LED
  }
}
```

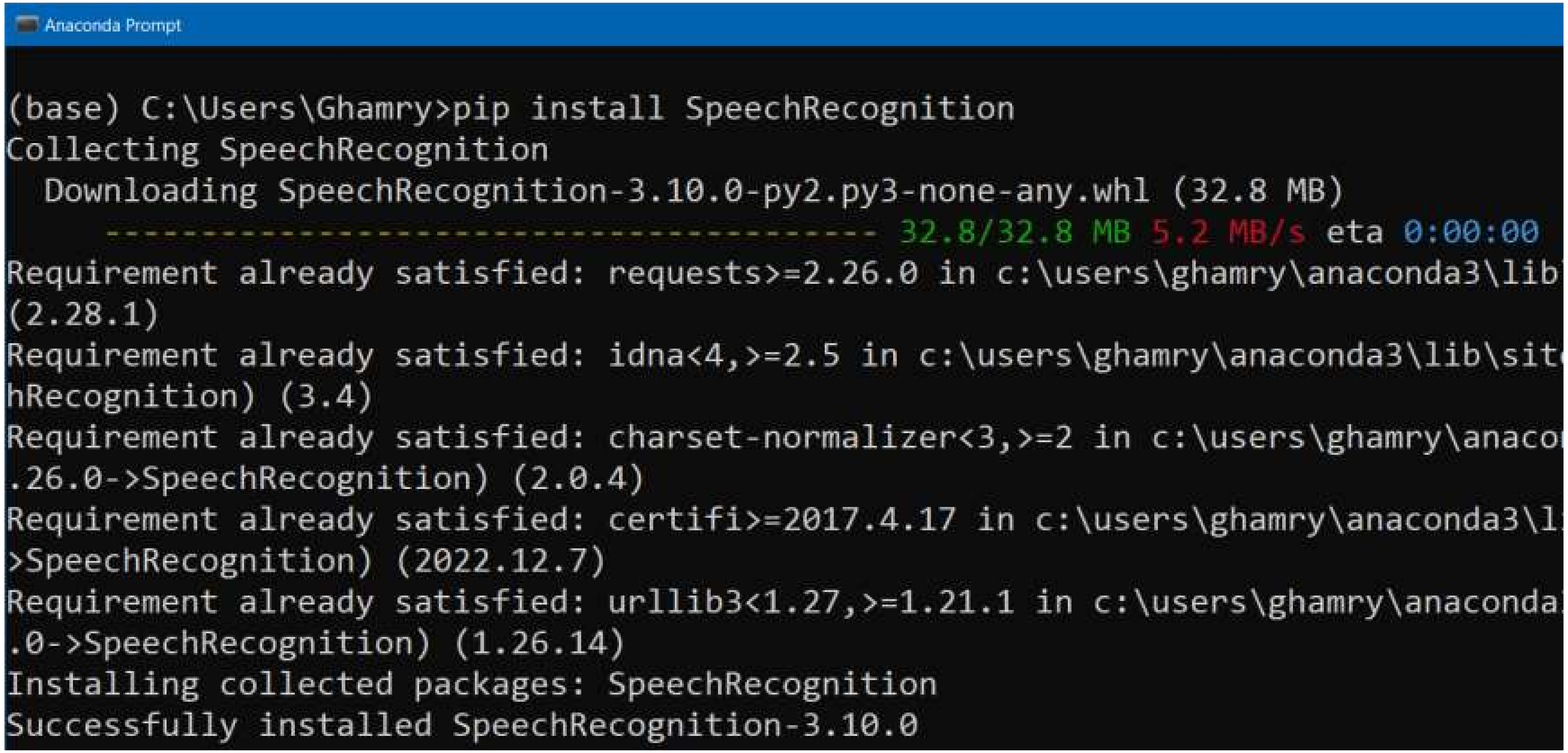
Voice-Controlled Lamp



Voice-Controlled Lamp: Installing SpeechRecognition

- To convert speech to text in Python, you can use `SpeechRecognition` library, which provides an interface to various speech recognition engines.

  `>> pip install SpeechRecognition`

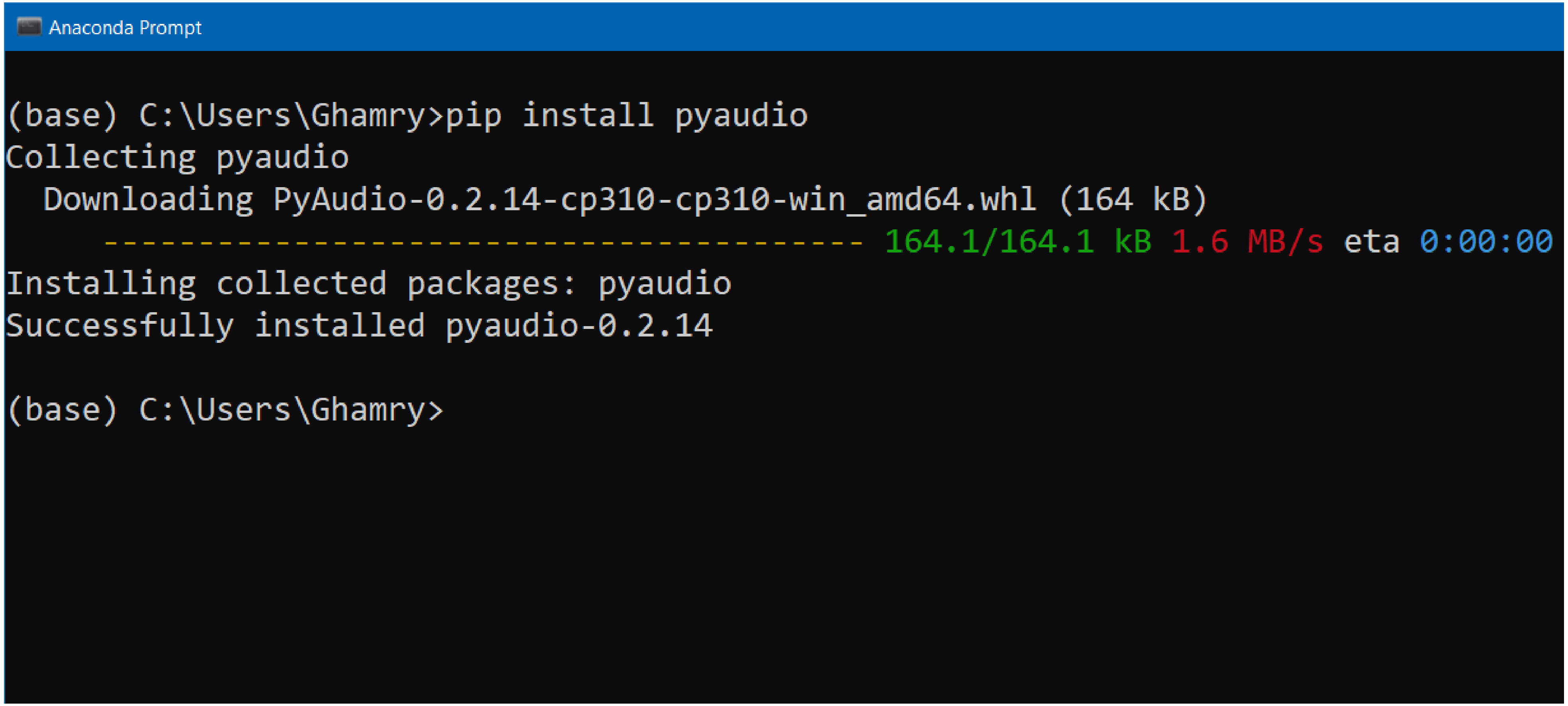


```
(base) C:\Users\Ghamry>pip install SpeechRecognition
Collecting SpeechRecognition
  Downloading SpeechRecognition-3.10.0-py2.py3-none-any.whl (32.8 MB)
     ---------------------------------------- 32.8/32.8 MB 5.2 MB/s eta 0:00:00
Requirement already satisfied: requests>=2.26.0 in c:\users\ghamry\anaconda3\lib
(2.28.1)
Requirement already satisfied: idna<4,>=2.5 in c:\users\ghamry\anaconda3\lib\sit
hRecognition) (3.4)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\ghamry\anaco
.26.0->SpeechRecognition) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\ghamry\anaconda3\l
>SpeechRecognition) (2022.12.7)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\ghamry\anaconda
.0->SpeechRecognition) (1.26.14)
Installing collected packages: SpeechRecognition
Successfully installed SpeechRecognition-3.10.0
```

Voice-Controlled Lamp: Installing PyAudio

- The `SpeechRecognition` library relies on `PyAudio` library.

  ```
  >> pip install pyaudio
  ```



```
(base) C:\Users\Ghamry>pip install pyaudio
Collecting pyaudio
  Downloading PyAudio-0.2.14-cp310-cp310-win_amd64.whl (164 kB)
     ---------------------------------------- 164.1/164.1 kB 1.6 MB/s eta 0:00:00
Installing collected packages: pyaudio
Successfully installed pyaudio-0.2.14

(base) C:\Users\Ghamry>
```

Voice-Controlled Lamp: Python Program

```python
import serial
import speech_recognition as sr

ser = serial.Serial('COM5', 9600)               # Initialize serial communication
recognizer = sr.Recognizer()                    # Initialize the recognizer

try:
    while True:
        try:
            # Capture audio from the microphone for 2 seconds
            with sr.Microphone() as source:
                print("Say something.")
                audio = recognizer.listen(source, phrase_time_limit=2)

            # Use Google Web Speech API to recognize the speech
            print('Processing voice ...')
            text = recognizer.recognize_google(audio, language='ar-EG')
            print(f'You said: {text}')

            # Send command to NodeMCU
            if text == 'نور اللمبه':
                ser.write('1'.encode())
            elif text == 'اطفي اللمبه':
                ser.write('0'.encode())
        except sr.UnknownValueError:
            print("Google Web Speech API could not understand audio.")
        except sr.RequestError:
            print("Could not request results from Google Web Speech API.")
        finally:
            repeat = input('\nRepeat? ')
except:
    ser.close()
    print("Serial connection closed.")
```

Voice-Controlled Lamp: NodeMCU Program

```
#define LED_PIN D6                        // Define LED pin

void setup() {
  Serial.begin(9600);                     // Start serial monitor
  pinMode(LED_PIN, OUTPUT);               // Initialize the pin D6 as an output
}

void loop() {
  // Read the incoming byte if available
  if(Serial.available()){                 // Check if there is a message available
    char cmd = Serial.read();             // Read the incoming byte

    if(cmd == '1')                        // If command is '1'
      digitalWrite(LED_PIN, HIGH);        // Turn on LED
    else if(cmd == '0')                   // If command is '0'
      digitalWrite(LED_PIN, LOW);         // Turn off LED
  }
}
```

References and Tutorials

- DHT11 Sensor Interfacing with NodeMCU
- Interfacing of DHT11 Sensor With ESP8266 nodemcu
- DHT11 Temperature & Humidity sensor on NodeMCU
- Interface DHT11 DHT22 with NodeMCU Using Web Server
- ESP8266 DHT11/DHT22 Temperature and Humidity Web Server
- pySerial Documentation
- ESP32 / ESP8266 Arduino: Serial communication with Python
- Raspberry Pi Arduino Serial Communication
- The Ultimate Guide To Speech Recognition With Python
- A Guide to Speech Recognition in Python