**DATA COMMUNICATION**

**LAB 2: SERIAL COMMUNICATION**

# Introduction

In [data transmission](https://en.wikipedia.org/wiki/Data_transmission), serial communication is the process of sending [data](https://en.wikipedia.org/wiki/Data) one [bit](https://en.wikipedia.org/wiki/Bit) at a time, sequentially, over a [communication channel](https://en.wikipedia.org/wiki/Communication_channel) or [computer bus](https://en.wikipedia.org/wiki/Computer_bus). This is in contrast to [parallel communication](https://en.wikipedia.org/wiki/Parallel_communication), where several bits are sent as a whole, on a link with several parallel channels.

In this lab, a serial communication is used for data transmission between the Arduino board and a computer or other devices. All Arduino boards have at least one serial port (also known as a UART or USART) named Serial. It communicates on digital pins 0 (RX) and 1 (TX) as well as with the computer via USB. Thus, if you use these functions, you cannot also use pins 0 and 1 for digital input or output.

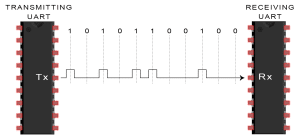


Figure 1: Data transmission using Serial communication

The process of sending data using serial communication is demonstrated in Figure 1. It is worth noting that the Tx pin of the transmitter must be connected to the Rx pin of the receiver.

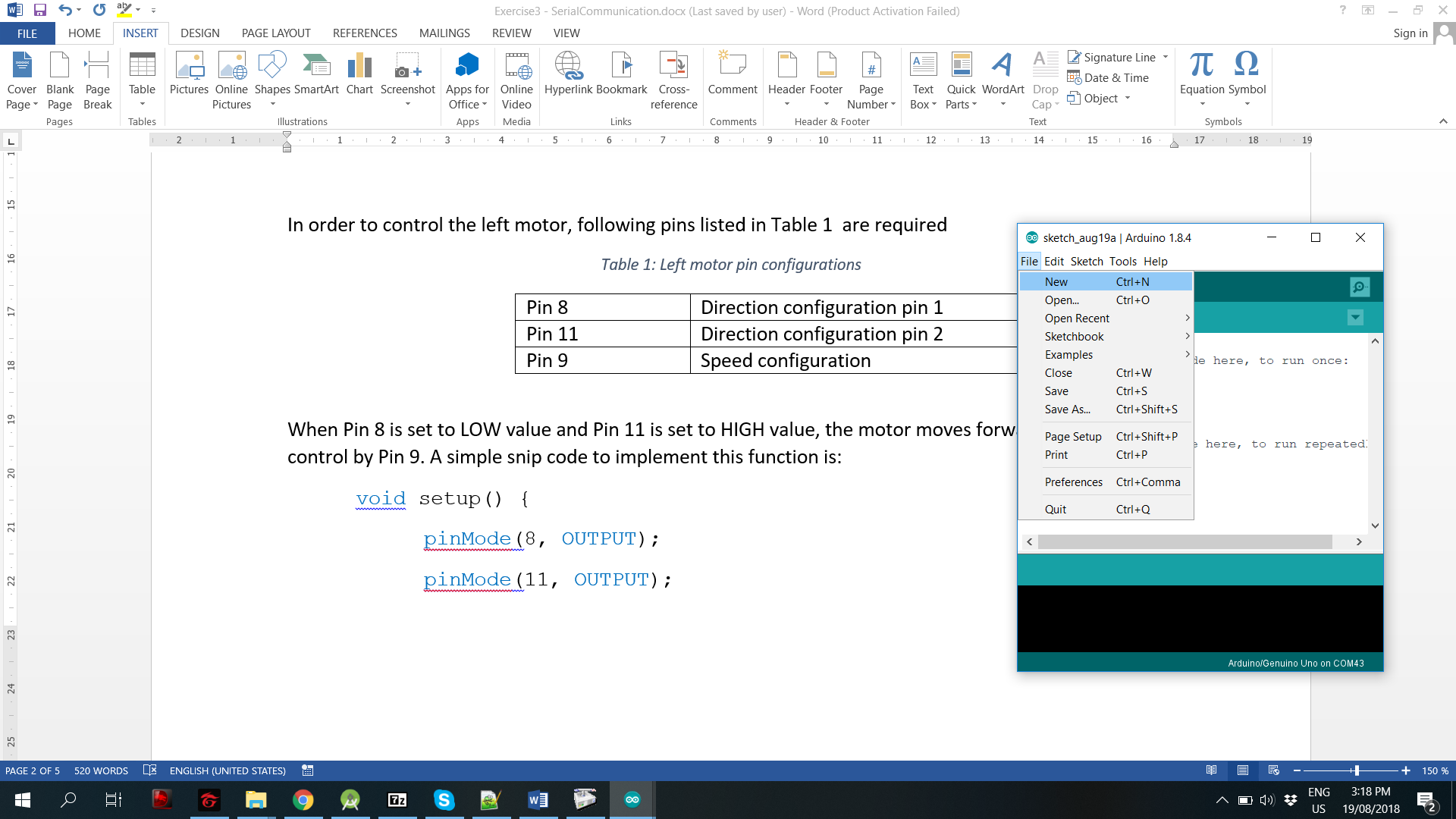


Figure 2: Create an empty project on Arduino by clicking File/New

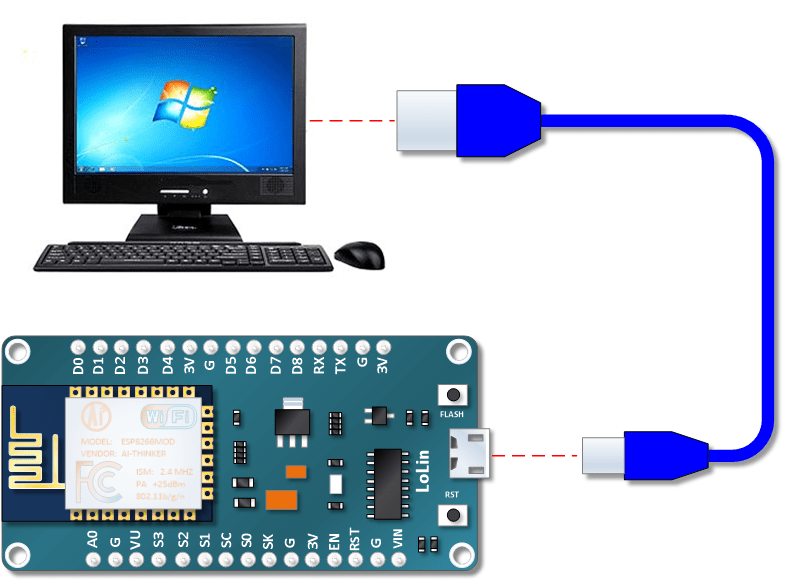


Figure 3: Serial connection (UBS cable) between Node MCU and PC

# Serial Interface in Arduino Uno

In this section, a manual to send data from Arduino board to the computer is presented briefly. Firstly, create a new project on Arduino by clicking on File and select New. An empty project is created as Figure 2. The first step in serial communication is set the speed of data transmission, which is expressed in baud, or baud-rate. ***In computer communication, the baud rate and bits per second (bps) are equivalent***. An example bellow will set the speed of the serial to 9600 bps and send “Hello, I am Node MCU ESP8266” to the PC. In fact, the USB connection between your NodeMCU board and PC is the Serial connection (see Figure 3).

**void** setup() {

// Set the speed to 9600bps

**Serial**.**begin**(9600);

**delay**(1000);

//Send a sentence to PC

**Serial**.**println**("Hello, I am Node MCU ESP8266");

}

Now, you can save your project and then, click on Tools/Port, chose a correct USB port which is used to connect to the board before uploading the code.

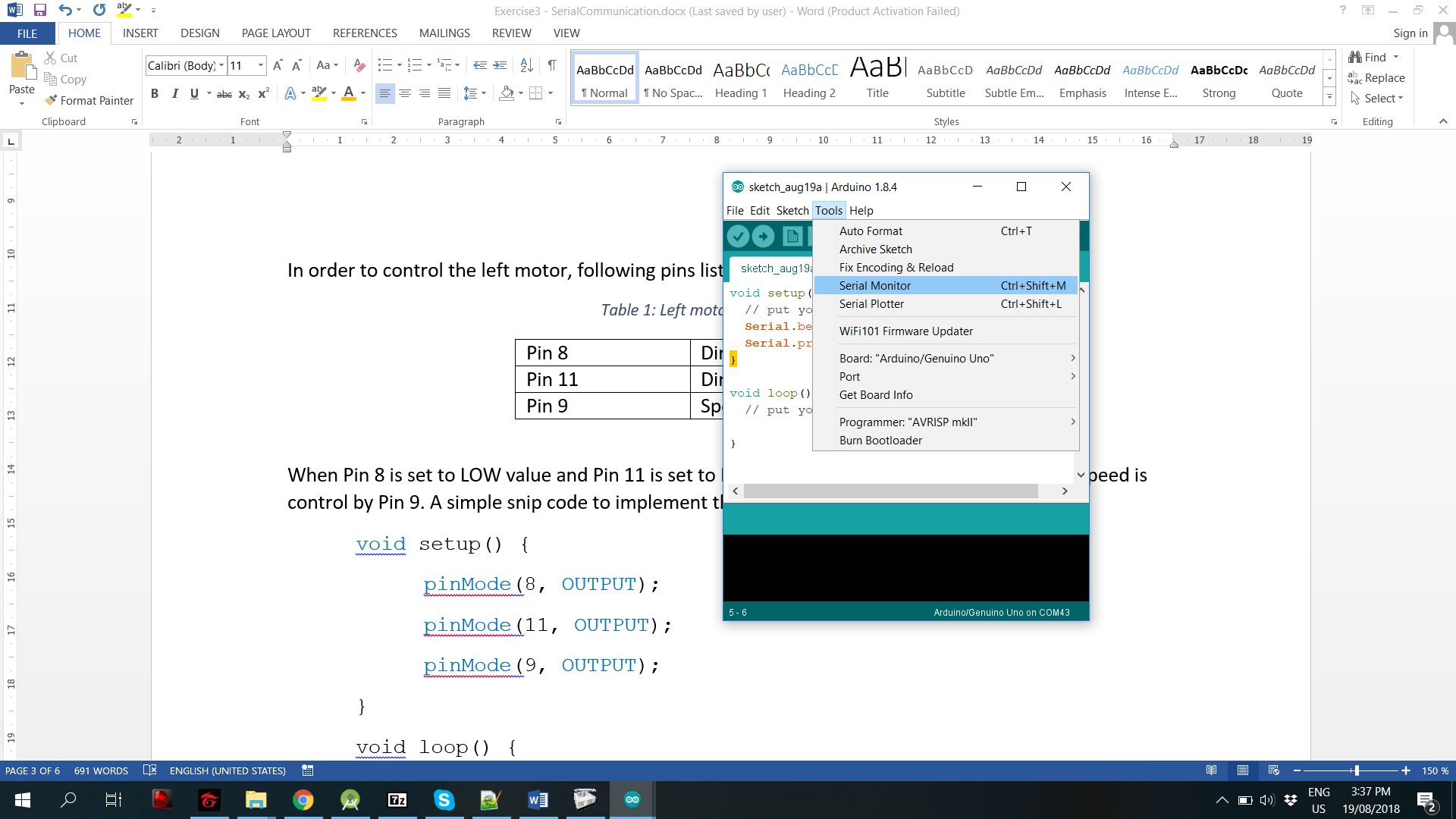


Figure 4: Select Serial Monitor to open the PC terminal

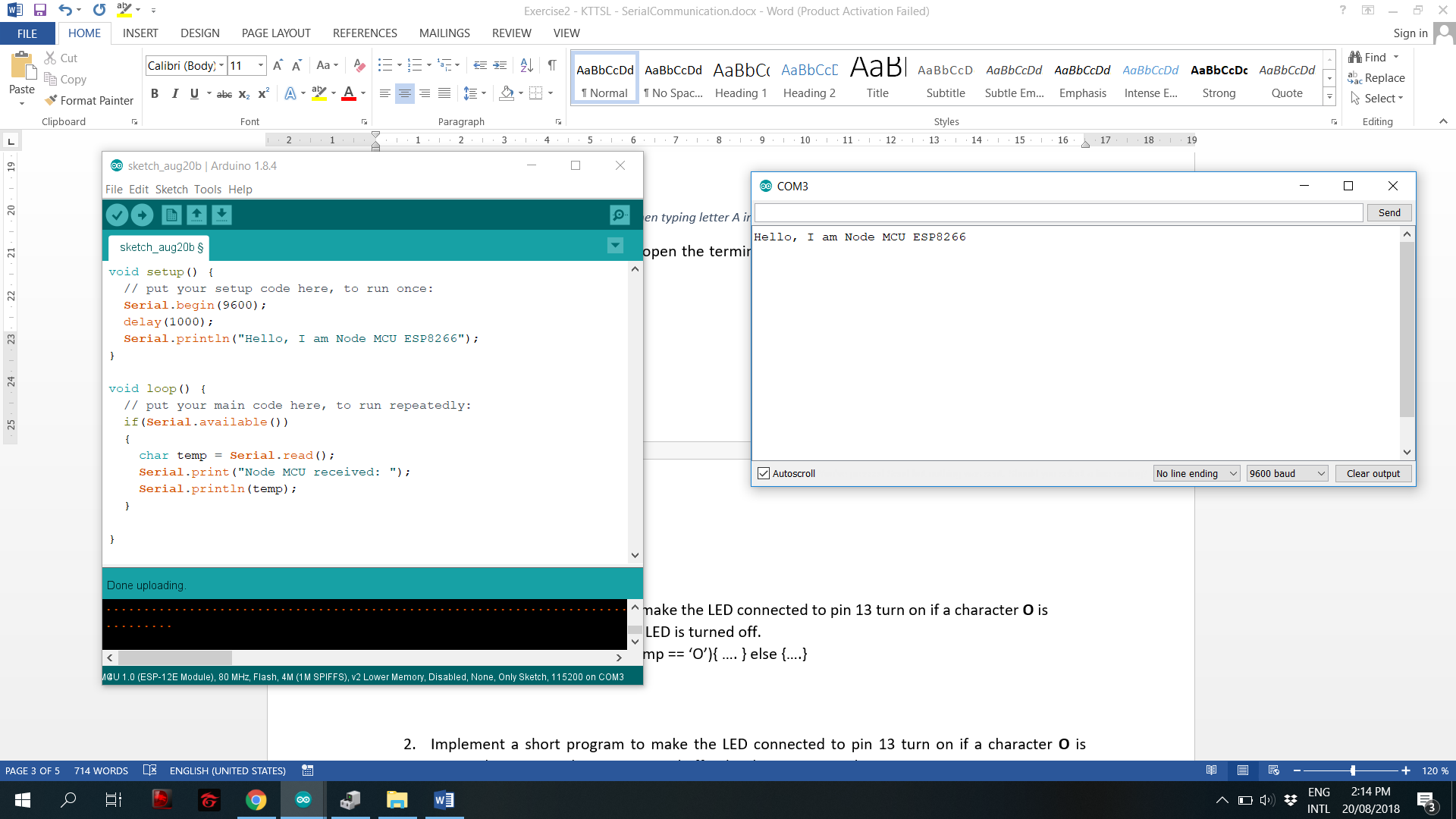


Figure 5: PC terminal: The speed is set to 9600 baud

Finally, click on Tools, select Serial Monitor as showed Figure 4 and then, a PC terminal will be opened as depicted in Figure 5. ***Please make sure that the speed of the terminal is set to 9600 baud***. Then, you can see the result on the terminal!!!

# Receive a Character from PC Terminal

From PC terminal, you can send some data to the Arduino board by pressing any charater on your PC keyboard and the click buttion Send. However, some code need to be implemented in the Arduino board as following: the board keep checking if there is a character sent to it. If there is a character, it will read this character and send it back to the PC terminal.

**void** loop() {

if(**Serial**.**available**())

{

char temp = **Serial**.**read**();

**Serial**.**print**("Node MCU received: ");

**Serial**.**println**(temp);

}

}

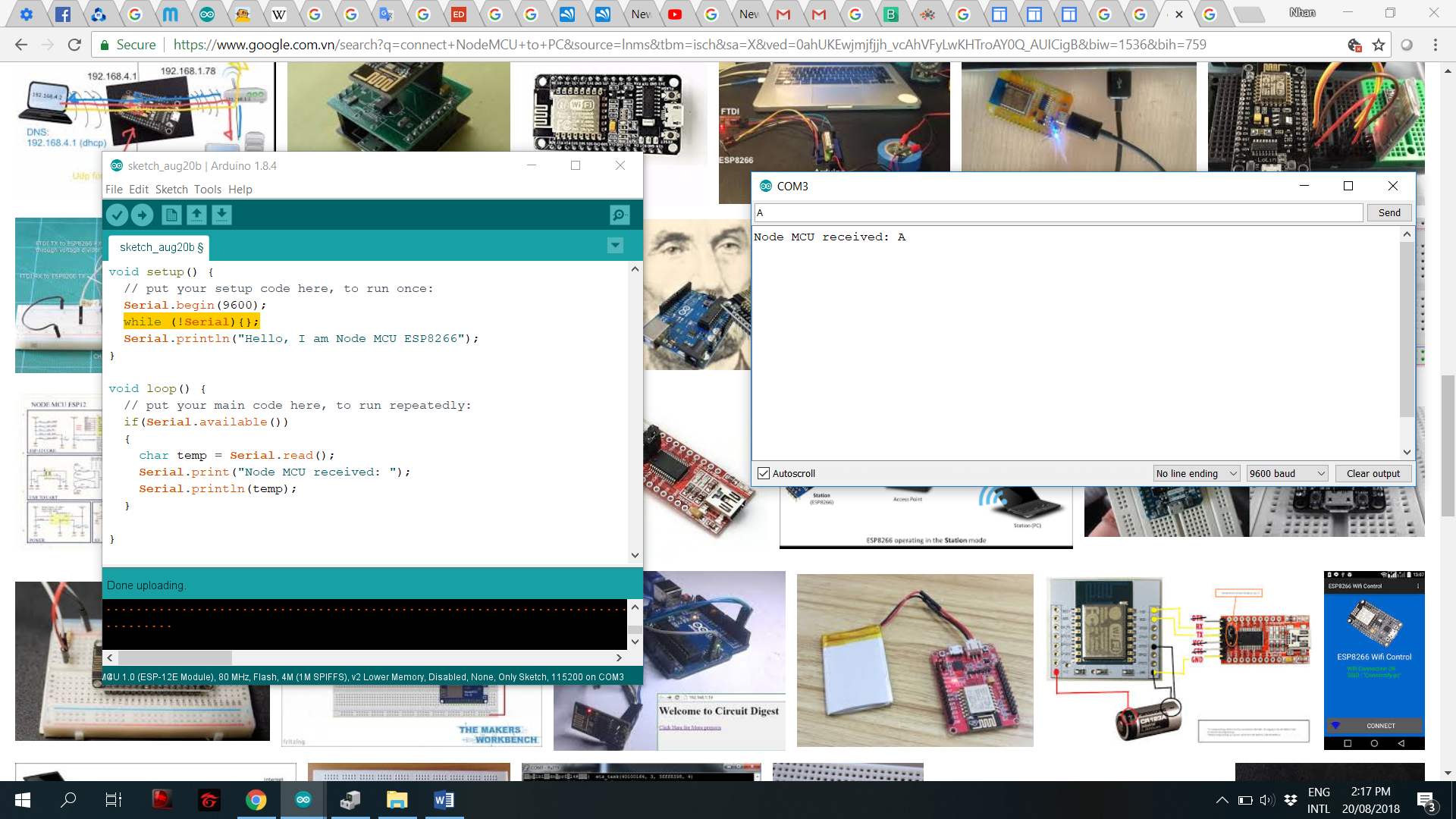


Figure 6: The result when typing letter A in the box, then click on Send button

After uploading to the board, you can open the terminal again by clicking on Tools/Serial Monitor. The results can be found in Figure 6

# Exercise

1. Implement a short program to make the LED connected to pin 13 turn on if a character **O** is received. Other characters, the LED is turned off.

Hint: Use the if statement: if(temp == ‘O’){ …. } else {….}

void loop() {

if (Serial.available()){

char temp = Serial.read();

if(('a' <= temp && temp <= 'z') || ('A' <= temp && temp <= 'Z'))

charReceived = temp;

}

if (charReceived == 'O')

digitalWrite(ledPin,LOW);

else if (charReceived == 'F')

digitalWrite(ledPin,HIGH);

delay(10);

}

1. Implement a short program to make the LED connected to pin 13 turn on if a character **O** is received. However, the LED is turned off only when receiving character **F**.

void loop() {

if (Serial.available()){

char temp = Serial.read();

if(('a' <= temp && temp <= 'z') || ('A' <= temp && temp <= 'Z'))

charReceived = temp;

}

if (charReceived == 'O')

digitalWrite(ledPin,LOW);

else if (charReceived == 'F')

digitalWrite(ledPin,HIGH);

delay(10);

}

1. Implement a short program to make the LED connected to pin 13 turn on if ***a sequence of characters*** **ON** is received. However, the LED is turned off only when receiving a sequence **OFF**.Moreover, there is a time-out for receiving a command, which is stated when letter O is received. After 5s, if a command is not finished or valid, the system is reset to the beginning, waiting for a new character **O**

#define ledPin 16

#define LED\_ON 1

#define LED\_OFF 0

#define READY 2

#define INIT 3

char charReceived;

int state;

bool OFF\_check;

int T, T\_OUT;

char cmdBuffer[4] = "";

void setup() {

Serial.begin(9600);

pinMode(ledPin,OUTPUT);

state = INIT;

T = 0;

T\_OUT = 500;

delay(1000);

}

void loop() {

if (Serial.available()){

char temp = Serial.read();

if((32 <= temp && temp <= 126))

// charReceived = temp;

strncat(cmdBuffer, &temp, 1);

}

switch(state){

case READY:

if (strcmp("O", cmdBuffer) == 0){

state = LED\_ON;

Serial.println("WATTING");

}

break;

case LED\_ON:

if (strcmp("O", cmdBuffer) == 0) T++;

else if (strcmp("ON", cmdBuffer) == 0){

state = READY;

digitalWrite(ledPin,LOW);

Serial.println("LED\_ON");

T = 0;

cmdBuffer[0] = 0;

}

else if (strcmp("OF", cmdBuffer) == 0){

state = LED\_OFF;

}

else {

state = READY;

T = 0;

cmdBuffer[0] = 0;

}

break;

case LED\_OFF:

if (strcmp("OF", cmdBuffer) == 0) T++;

else if (strcmp("OFF", cmdBuffer) == 0){

state = READY;

digitalWrite(ledPin,HIGH);

Serial.println("LED\_OFF");

T = 0;

cmdBuffer[0] = 0;

}

else {

state = READY;

T = 0;

cmdBuffer[0] = 0;

}

break;

default:

state = READY;

digitalWrite(ledPin,HIGH);

Serial.println("INIT");

break;

}

if (T > T\_OUT) {

state = READY;

Serial.println("TIME OUT");

T = 0;

cmdBuffer[0] = 0;

}

delay(10);

}

# Extra exercise

Assume that we have a door with secret code. The security is check when the character **#** is sent. Use the available LED to demonstrate the state of the door (ON – means security is valid, OFF – means security is invalid). However, when # is sent, only the last 4 characters are used to check the security. For instance, if the security code is OPEN, the sequence **123OPEN#** or **abcOPEN#** are all passed. There is also a timeout when the first character is received. After this timeout, the whole system is reset to the initiate state.

#define ledPin 16

#define READY 1

#define INIT 0

#define CHECK 2

#define SECRET\_CODE "OPEN"

int state;

int T, T\_OUT;

char temp;

char cmdBuffer[32] = "";

void setup() {

Serial.begin(9600);

pinMode(ledPin,OUTPUT);

state = INIT;

T = 0;

T\_OUT = 500;

delay(1000);

}

void loop() {

if (Serial.available()){

temp = Serial.read();

Serial.println(temp);

}

switch(state){

case READY:

if (temp =='#') state = CHECK;

else if((32 <= temp && temp <= 126))

strncat(cmdBuffer, &temp, 1);

if (cmdBuffer[0] != 0) T++;

break;

case CHECK:

if (strlen(cmdBuffer) >= strlen(SECRET\_CODE) && strcmp(SECRET\_CODE, cmdBuffer+strlen(cmdBuffer)-4) == 0) {

state = READY;

digitalWrite(ledPin,LOW);

Serial.println("Security is VALID");

}

else{

state = READY;

digitalWrite(ledPin,HIGH);

Serial.println("Security is INVALID");

}

T = 0;

cmdBuffer[0] = 0;

break;

default:

state = READY;

digitalWrite(ledPin,HIGH);

Serial.println("INIT");

break;

}

if (T > T\_OUT) {

state = READY;

Serial.println("TIME OUT");

T = 0;

cmdBuffer[0] = 0;

}

delay(10);

}