DATA COMMUNICATION

LAB 2: SERIAL COMMUNICATION

I. Introduction

In data transmission, serial communication is the process of sending data one bit at a time, sequentially, over a communication channel or computer bus. This is in contrast to 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.

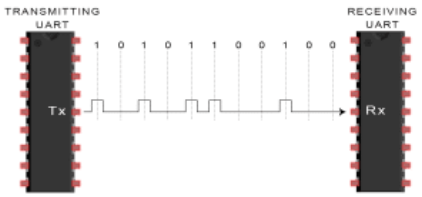

Figure 1: Data transmission using Serial communication

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.

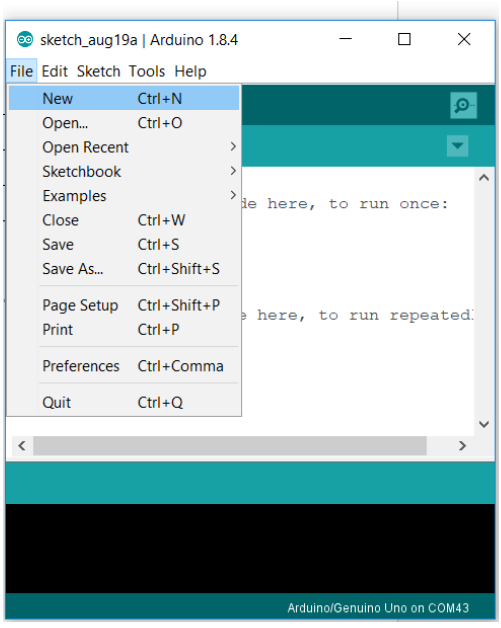

Figure 2: Create an empty project on Arduino by clicking File/New

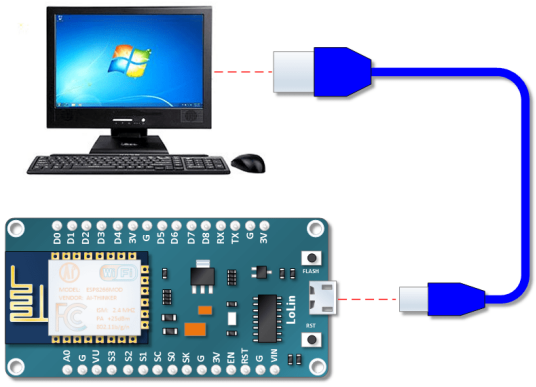

Figure 3: Serial connection (UBS cable) between Node MCU and PC

II. 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.

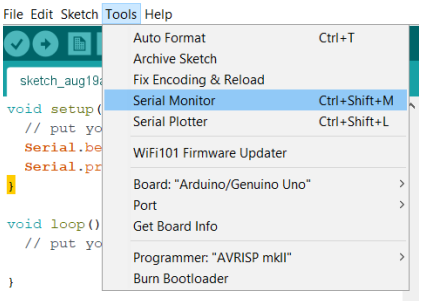

Figure 4: Select Serial Monitor to open the PC terminal

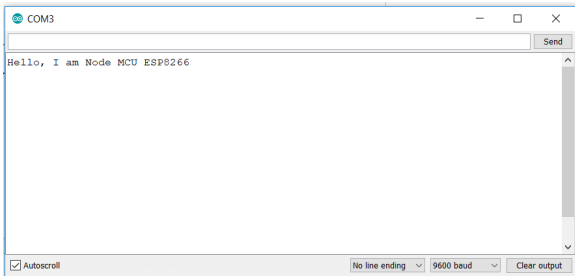

Figure 5: PC terminal: The speed is set to 9600 baud

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!!!

III. 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);
  }
}
```

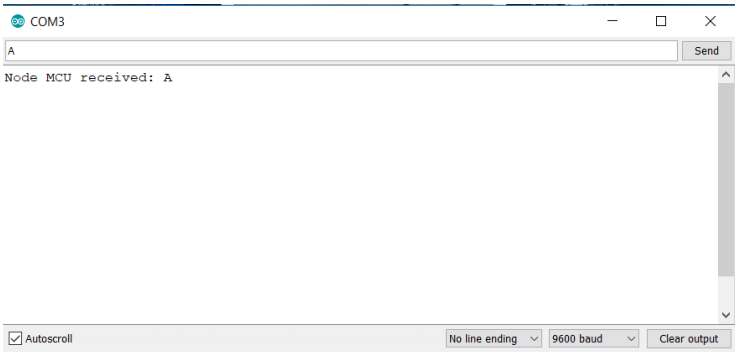

Figure 6: The result when typing letter A in the box, then click on Send button

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

IV. 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 {….}

```
#define WAITING 0
#define ON      1
#define OFF     2

short state;
char tmp;

void setup(){
  Serial.begin(9600);
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, HIGH);

  state = WAITING;
  tmp = 0;
}
void loop(){
  switch (state){
    case WAITING:{
```

```
      if (Serial.available()){
        char tmp = Serial.read();
        switch (tmp){
          case 'O':{
            state = ON;
            break;
          }
          default:{
            state = OFF;
            break;
          }
        }
      }
      break;
    }
    case ON:{
      digitalWrite(LED_BUILTIN, LOW);
      state = WAITING;
      break;
    }
    case OFF:{
      digitalWrite(LED_BUILTIN, HIGH);
      state = WAITING;
      break;
    }
  }
  delay(10);
}
```

2. 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**.

```
#define WAITING 0
#define ON      1
#define OFF     2
```

```
short state;
char tmp;


void setup(){
  Serial.begin(9600);
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, HIGH);


  state = WAITING;
  tmp = 0;
}
void loop(){
  switch (state){
    case WAITING:{
      if (Serial.available()){
        char tmp = Serial.read();
        switch (tmp){
          case 'O':{
            state = ON;
            break;
          }
          case 'F':{
            state = OFF;
            break;
          }
        }
      }
      break;
    }
    case ON:{
      digitalWrite(LED_BUILTIN, LOW);
      state = WAITING;
```

```
      break;
    }
    case OFF:{
      digitalWrite(LED_BUILTIN, HIGH);
      state = WAITING;
      break;
    }
  }
  delay(10);
}
```

3. 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 RESET     0
#define WAIT_NEXT 1
#define LED_ON    2
#define LED_OFF   3

short state;

unsigned short num;
char  c;

long tmr1;


void Timer1_set(unsigned int sec){
  tmr1 = sec * 100;
}

bool Timer1_tick(){
  if (tmr1 > 0){
    return (--tmr1) == 0;
  }
  else
    return false;
}

void setup() {
<...>
```

```
  num = 0;
  state = RESET;
  c = 0;
}

void loop() {
  switch(state){
    case RESET:{
      num = 0;
      if (Serial.available()){
        c = Serial.read();
        switch(c){
          case 'O':{
            num++;
            state = WAIT_NEXT;
            Timer1_set(5);
            break;
          }
        }
      }
      break;
    }
    case WAIT_NEXT:{
      if (Timer1_tick() == true){
        state = RESET;
        break;
      }

      if (Serial.available()){
        c = Serial.read();
        switch(c){
          case 'N':{
            if (num == 1)
              state = LED_ON;
            else
              state = RESET;
            break;
          }
          case 'F':{
            if (num == 2)
              state = LED_OFF;
            else
              num++;
            break;
          }
        }
      }

      break;
    }

    case LED_ON:{
      digitalWrite(LED_BUILTIN, LOW);
      state = RESET;
      break;
    }
```

```
        case LED_OFF:{
          digitalWrite(LED_BUILTIN, HIGH);
          state = RESET;
          break;
        }
      }

      delay(10);
    }
```

V. 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.

```
<Timer functions are the same as those of Problem 3>


char c;               // store read character
char c1, c2, c3, c4;   // store read characters like stack


void pushchartoStack(){
  c1 = c2;
  c2 = c3;
  c3 = c4;
  c4 = c;
}


boolean ispassvalid(){
  if ((c1 == 'O') &&
      (c2 == 'P') &&
      (c3 == 'E') &&
      (c4 == 'N'))
    return true;
  else
    return false;
```

```
}

void setup() {
<init functions for Serial and LED>
  c = c1 = c2 = c3 = c4 = 0;
  state = WAIT_1SHARP;
}

void loop() {
  switch(state){
    case WAIT_1SHARP:{
      if (Serial.available() > 0){
        c1 = 0;       // this to ensure the input string "##" is invalid
        c = Serial.read();
        if (c == '#'){
          state = WAIT_2SHARP;
          Timer1_set(5);
        }
      }
      break;
    }

    case WAIT_2SHARP:{
      if (Timer1_tick() == true){
        state = WAIT_1SHARP;
        break;
      }

      if (Serial.available() > 0){
        c = Serial.read();
        if (c == '#'){       // read second '#'
          if (ispassvalid())
            state = DOOR_OPEN;
          else
```

```
            state = DOOR_CLOSE;
          }
          else
            pushchartoStack();
        }
        break;
      }

      case DOOR_OPEN:{
        digitalWrite(LED_BUILTIN, LOW);
        state = WAIT_1SHARP;
        break;
      }
      case DOOR_CLOSE:{
        digitalWrite(LED_BUILTIN, HIGH);
        state = WAIT_1SHARP;
        break;
      }
    }

    delay(10);
}
```