

Electrical and Computer Systems Engineering Department

Computer Design Laboratory

ENCS411

Report 1

Experiment 3 : Serial Communication with Arduino

Prepared by

Anas Nimer_1180180

Instructor:

Dr. Abdallatif Abuissa

Teacher assistant:

Eng. Motasem Diab

BIRZEIT

July – 2020

Abstract:

In this experiment, we will illustrate the basic concepts of serial communication and to practically apply it using Arduino and we will describe UART chip structure and how it works also we will talk about translate parallel data to and from serial data.

Table of Contents:

Abstract:.....	II
1. Theory.....	V
1.1. UART:	V
1.2. Serial communication:	VI
1.2.1. Start Bit:.....	VI
1.2.2. Stop Bits:	VI
1.2.3. Parity:.....	VII
1.2.4. Data Frame:	VII
1.3. Serial Communication Functions	VII
1.3.1. Serial.available().....	VII
1.3.2. Serial.begin()	VII
1.3.3. Serial.read().....	VII
1.3.4. Serial.readBytes(buffer,length).....	VII
1.3.5. Serial.readString().....	VII
1.3.6. Serial.parseInt().....	VIII
1.3.7. Serial.paseFloat().....	VIII
1.3.8. Serial.print()	VIII
1.3.9. Serial.write().....	VIII
2. Procedure & Discussion:	IX
2.1. Part (1) : Basic Communication Between Arduino & PC:.....	IX
2.2. Part (2) : Basic Communication Between 2 Arduinos:.....	XI
2.3. Part (3) : Push Button & LED using 2 Arduinos:	XIII
2.4. Part (4) : Visualization of serial communication using Serial Plotter:.....	XV
3. TO DO:	XVII
4. Conclusion:	XIX
5. References:.....	XX
6. Appendix:.....	XXI
6.1. Part1:.....	XXI
6.2. Part2:	XXII
6.3. Part3:.....	XXIII
6.4. Part4:	XXIV
6.5. Todo:.....	XXV

List of figure:

Figure 1 (UART Communication).....	V
Figure 2 (Serial Communication).....	VI
Figure 3 (Communication between Arduino & PC design)	IX
Figure 4 (Communication between Arduino & PC code)	X
Figure 5 (Communication Between 2 Arduinos design)	XI
Figure 6 (Sender code)	XII
Figure 7 (Receiver code).....	XII
Figure 8 (Connecting 2 Arduino with a push Button).....	XIII
Figure 9 (Transmitter connected with button code)	XIII
Figure 10 (Receiver connected with LED code).....	XIV
Figure 11 (serial communication connection design).....	XV
Figure 12 (serial plotter output).....	XV
Figure 13 (Transmit A Code).....	XVI
Figure 14 (Receiver Code).....	XVI
Figure 15 (TO DO design).....	XVII
Figure 16 (TO DO Transmitter Code).....	XVII
Figure 17 (TO DO Receiver Code).....	XVIII

1. Theory

1.1. UART:

UART stand for Universal Asynchronous Receiver/Transmitter , the main purpose from UART is to transmit and receive serial data, two UART needed to make communication between them directly using 2 wires from Tx pin of the transmitting UART to the Rx pin of the receiving UART, The transmitting UART converts parallel data from a controlling device like a CPU into serial form, transmits it in serial to the receiving UART, which then converts the serial data back into parallel data for the receiving device . [1]

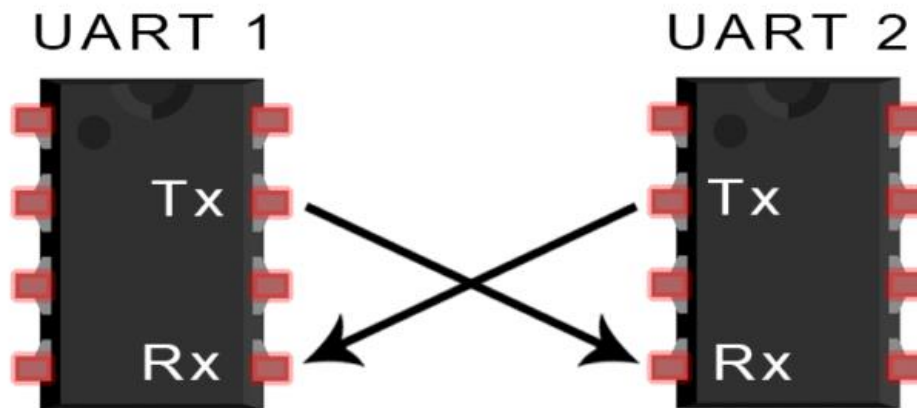


Figure 1 (UART Communication)

1.2. Serial communication:

In telecommunication and 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.

Serial communication is used for all long-haul communication and most computer networks, where the cost of cable and synchronization difficulties make parallel communication impractical.[2]

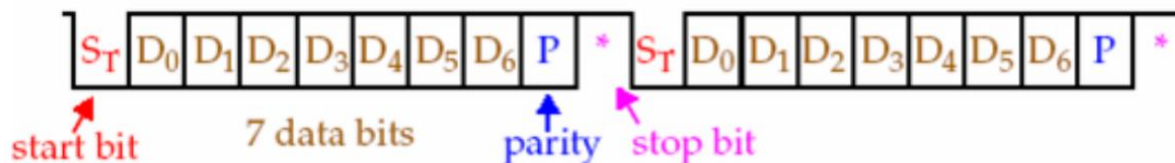


Figure 2 (Serial Communication)

Asynchronous form used to transmit data from UART without clock signal , instead of that data transmitted packet include start bit , end bit and parity so the receiving UART knows when to start reading the bits which will be discuss them later .

1.2.1. Start Bit:

Used to tell receiver that the serial data is start by transfer value form high (which is the normal case when there is no transmitting data) to low for one clock cycle .

1.2.2. Stop Bits:

To signal the end of the data packet, the sending UART drives the data transmission line from a low voltage to a high voltage for at least two-bit durations. [1]

1.2.3. Parity:

Parity describes the evenness or oddness of a number also till UART if the receiving bits contains errors (if data frame have changed during transmission) or not, there are three cases for Parity bit :

- 1) **Even parity** : parity bit is a 0 ,the 1-bits in the data frame total to an even number .
- 2) **Odd parity** : parity bit is a 1,the 1-bits in the data frame total to an odd number .
- 3) **Error in data** : if the parity bit is a 0, and the total is odd; or the parity bit is a 1, and the total is even .

1.2.4. Data Frame:

The data frame contains the actual data being transferred. It can be 5 bits up to 8 bits long if a parity bit is used. If no parity bit is used, the data frame can be 9 bits long. [1]

1.3. Serial Communication Functions

Serial is used for communication Arduino board and a computer or other devices through pin 0 RX and pin 1 TX as well as with the computer via USB and we will talk about some functions used in communication process such as :-

1.3.1. Serial.available()

Get the number of bytes available for reading from the serial part from the stored data in the receive buffer .

1.3.2. Serial.begin()

sets the baud rate for serial data communication. The baud rate signifies the data rate in bits per second.[3]

1.3.3. Serial.read()

Read incoming serial data and return the first byte of incoming serial data or -1 if data is not available .

1.3.4. Serial.readBytes(buffer,length)

Reads characters from the serial port into a buffer, returns the number of bytes placed in the buffer.

1.3.5. Serial.readString()

Returns A String read from the serial buffer.

1.3.6. Serial.parseInt()

Looks for the next valid integer in the incoming serial.

1.3.7. Serial.parseFloat()

Returns the first valid floating-point number from the Serial buffer .

1.3.8. Serial.print()

Prints data to the serial port as human-readable ASCII text.

1.3.9. Serial.write()

Writes binary data to the serial port .

2. Procedure & Discussion:

2.1. Part (1) : Basic Communication Between Arduino & PC:

At this part we connect the circuit in figure 3 below and then we run the code in the appendix [A1] .

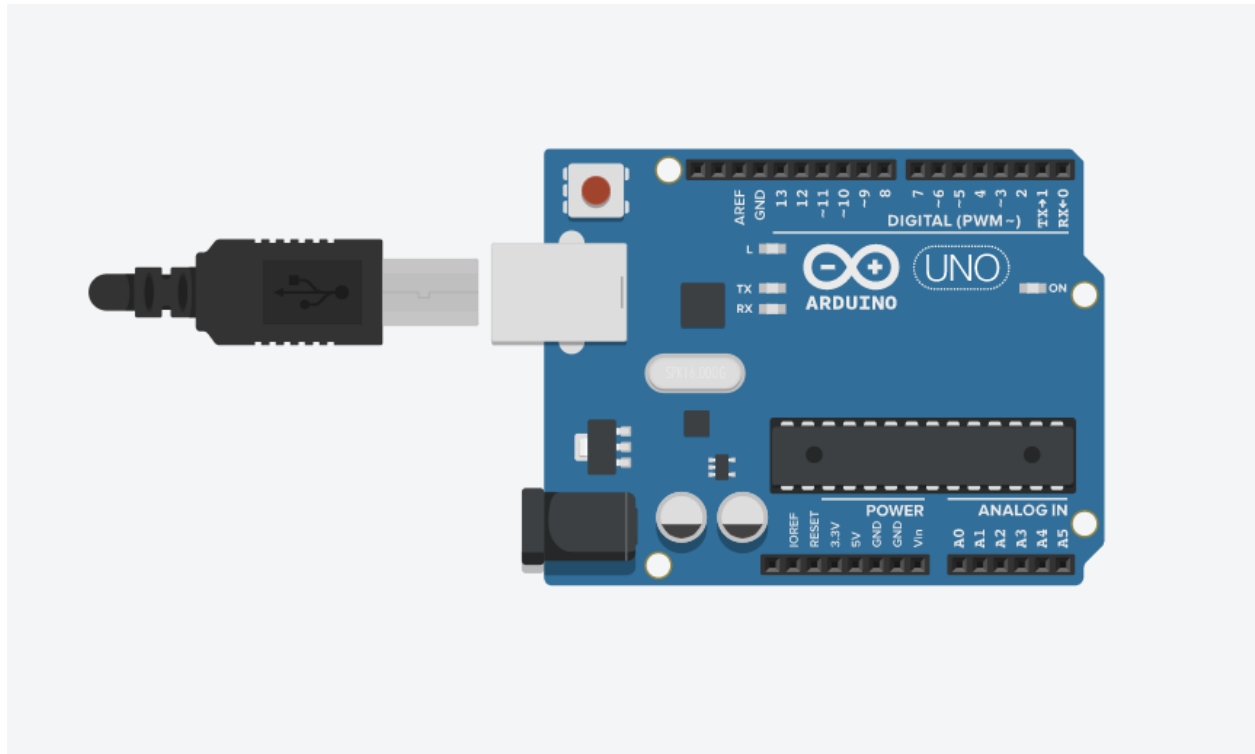
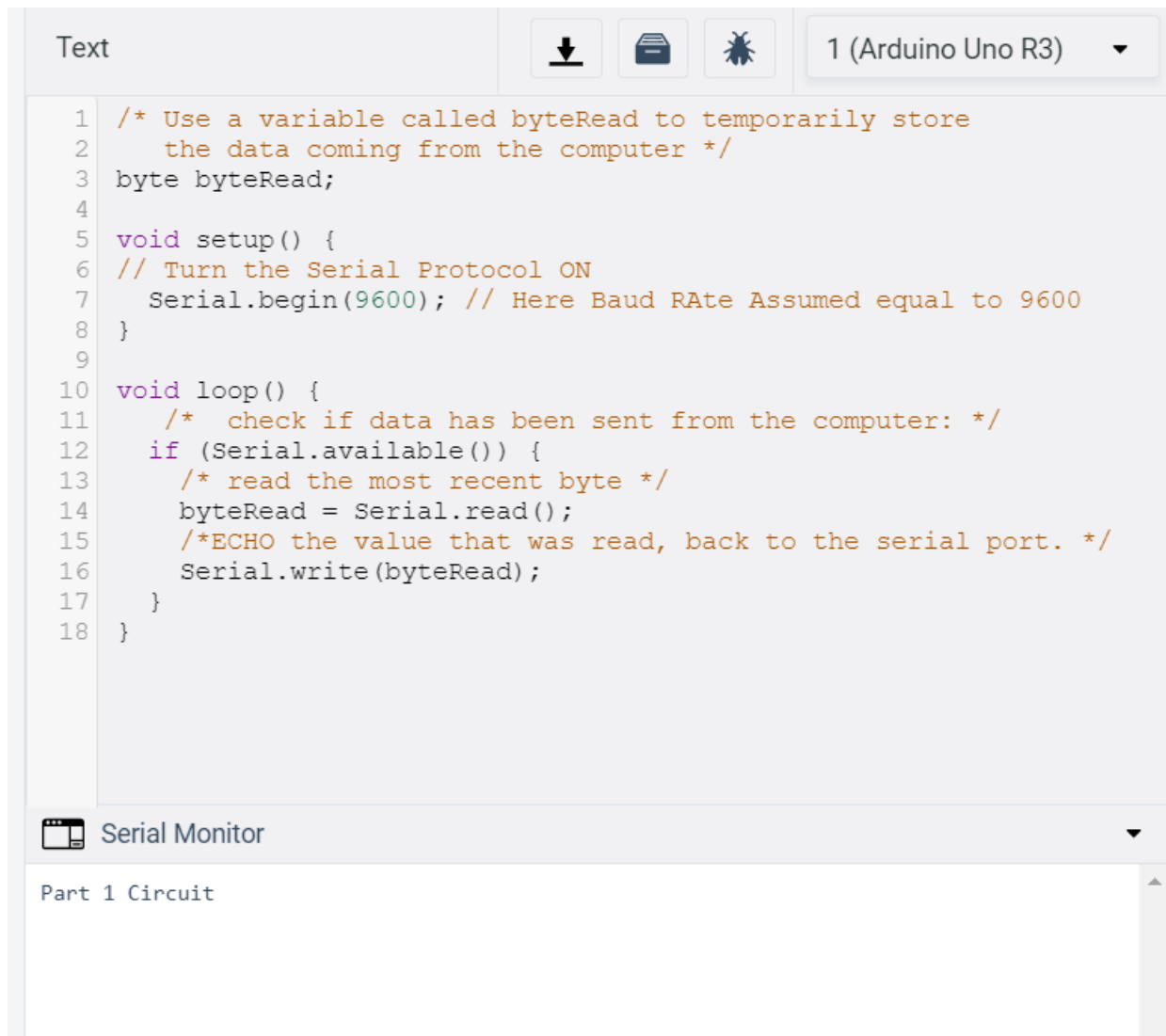


Figure 3 (Communication between Arduino & PC design)



The image shows a screenshot of an IDE interface. At the top, there is a 'Text' tab with icons for download, save, and a bug. A dropdown menu shows '1 (Arduino Uno R3)'. Below this is a code editor with the following C++ code:

```
1  /* Use a variable called byteRead to temporarily store
2     the data coming from the computer */
3  byte byteRead;
4
5  void setup() {
6    // Turn the Serial Protocol ON
7    Serial.begin(9600); // Here Baud RATE Assumed equal to 9600
8  }
9
10 void loop() {
11   /* check if data has been sent from the computer: */
12   if (Serial.available()) {
13     /* read the most recent byte */
14     byteRead = Serial.read();
15     /*ECHO the value that was read, back to the serial port. */
16     Serial.write(byteRead);
17   }
18 }
```

Below the code editor is a 'Serial Monitor' window with a dropdown arrow. The window contains the text 'Part 1 Circuit'.

Figure 4 (Communication between Arduino & PC code)

As shown in figure 4 firstly, a **byteRead** variable was defined to temporarily store the data coming from the computer , then baud rate was assumed equal to 9600 using `Serial.begin()` method , in loop function if statement checked if data has been sent from the computer , if it `Serial.read()` function read data and store it temporary in the variable then print it on the serial monitor using `Serial.write()` function .

2.2. Part (2) : Basic Communication Between 2 Arduinos:

In this part a simple information was sent between the 2 Arduinos

[“Hello World!”] sentence was sent order to become familiar with the connections

So the circuit connected such that TX in the First Arduino connected with RX in the second one and Vice versa [TX1 ↔ RX2 & RX1 ↔ TX2]. The ground must be common between both of them. as shown in figure 5 below .

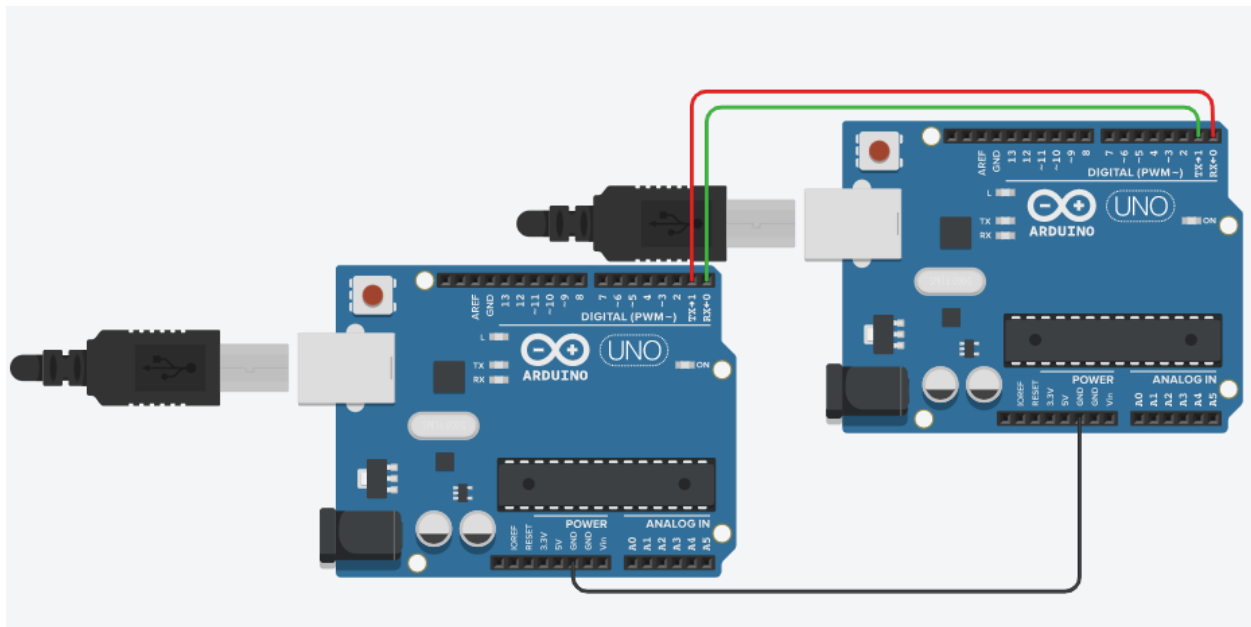


Figure 5 (Communication Between 2 Arduinos design)

And then we run the code (sender code in appendix A[2.1] & receiver code in A[2.2])

And the serial monitors of the sender and receiver shown in figure 6&7 above.

```
1  /* ##### Sender Code ##### */
2  void setup() {
3      // Begin the Serial at 9600 Baud
4      Serial.begin(9600);
5  }
6
7  void loop() {
8
9      Serial.write("Hellow word"); //Write the serial data
10     delay(1000);
11     Serial.println(); // Print serial data on the Serial Monitor
12 }
```

Serial Monitor

Hellow word
Hellow word
Hellow word
Hellow word
Hellow word
Hellow word

Figure 6 (Sender code)

Text

2 (Arduino Uno R3)

```
1  /* ##### Receiver #####*/
2  void setup() {
3      // Begin the Serial at 9600 Baud
4      Serial.begin(9600);
5  }
6
7  void loop() {
8      delay(1000); // here delay is important because transmit is faster
9      byte v = Serial.read();// assign serial data to variable v
10     delay(1000);
11     Serial.write(v);//print serial data on the Serial Monitor
12 }
```

Serial Monitor

Hellow word
Hel

Figure 7 (Receiver code)

Sender code write data to the serial port as binary then print it with new line on the Serial Monitor Receiver code receive the data using Serial.read() method and store it in byte variable then print it on the Serial Monitor , here delay is important because in general transmitter is faster than receiver also it's important that both transmitter and receiver have **the same baud rate** .

2.3. Part (3) : Push Button & LED using 2 Arduinos:

In this part we connect a push button with Arduino1 and whenever it's pushed, the value 1 is sent to the Arduino2. , and Arduino2 receives the data and whenever logic 1 is read, a LED will be turned on. Then we run the tow codes (Transmitter code in appendix [3.1] , and receiver code in appendix [3.2])

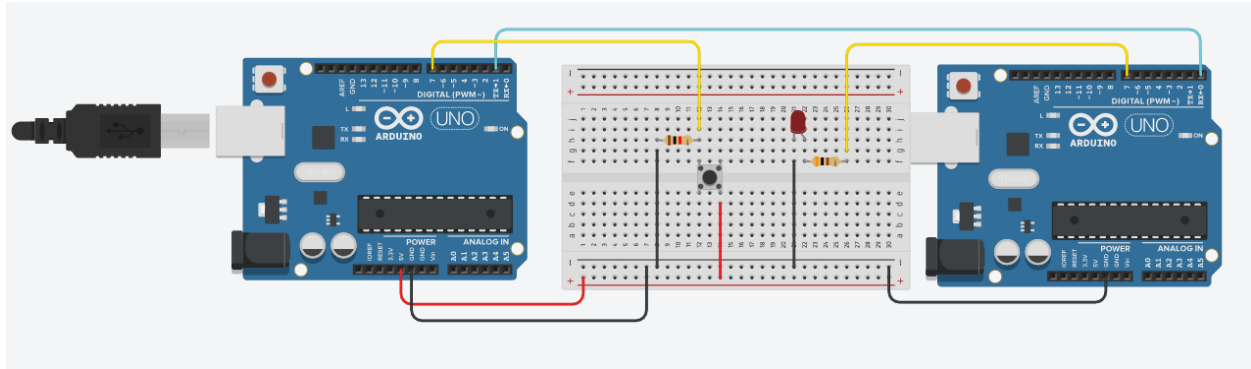


Figure 8 (Connecting 2 Arduino with a push Button)

```
1 /* ##### Transmitter ##### */
2 int pin =7; // define pin nuber that used
3 int n =0;
4 void setup() {
5   pinMode(pin, INPUT); // define interrupt number and when should be triggerd
6   Serial.begin(9600); // Setup the Serial at 9600 Baud
7 }
8 void loop() {
9   n = digitalRead(7); // assign pin 7 value to variable n
10  if(n == HIGH ) { // if the button bushed
11    Serial.println('1'); // print 1 on the Serial Monitor
12    delay(1000);
13  }
14 }
```

Serial Monitor

1
1

Figure 9 (Transmitter connected with button code)

```

1  /* ##### Receiver ##### */
2  int pin1 =7; // assign pin number used to pin variable
3  void setup() {
4      // Begin the Serial at 9600 Baud
5      Serial.begin(9600);
6      pinMode(pin1, OUTPUT); // define interrupt number and when should be triggered
7  }
8  void loop() {
9      if (Serial.available()) // if there's a serial data
10     {
11         //Read the serial data and store in var
12         char data = Serial.read();
13         //Print data on Serial Monitor
14         Serial.println(data);
15         if( data == '1'){ // if there's data if it equal to 1 turn on the light
16             digitalWrite(pin1, HIGH);
17         }
18         delay(1000);
19         digitalWrite(pin1,LOW); // after the specified delay turn off the light
20     }
21 }

```

 Serial Monitor

1

1

Figure 10 (Receiver connected with LED code)

As shown in Figure 9 the transmitter send data depending on the push button if it's pushed then sent **1** and print 1 on the Serial Monitor , where the receiver in Figure 10 accept serial data from the transmitter if the transmitted serial data equal 1 then by using interrupt the LED bulb will be turned on and after a specified delay time it will be turned off .

- **NOTE :** the comments was written in the code and as seen in the figures which it explain the code step by step with the additional explains after each figure .

2.4. Part (4) : Visualization of serial communication using Serial Plotter:

The idea of this part is to see how the serial data are transferred as start-data-parity-stop bits. This can be done using Serial plotter in Arduino IDE. So the circuit was connected as in figure 11 below and then run the tow codes (Transmitter code in appendix [4.1] , and receiver code in appendix [4.2]).

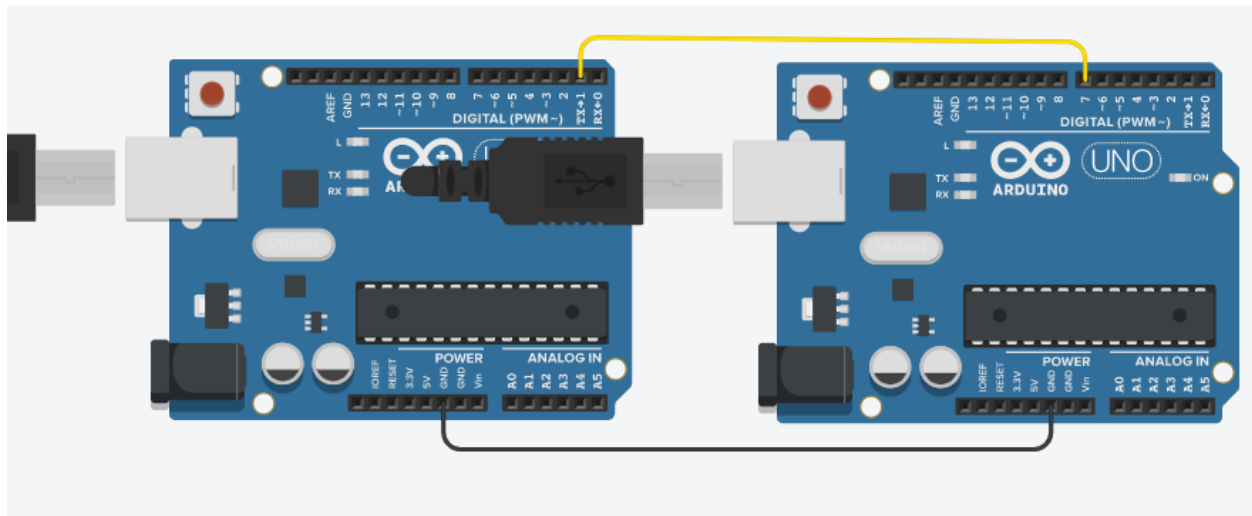


Figure 11 (serial communication connection design)

The output of the serial plotter is shown in figure 12 below :-

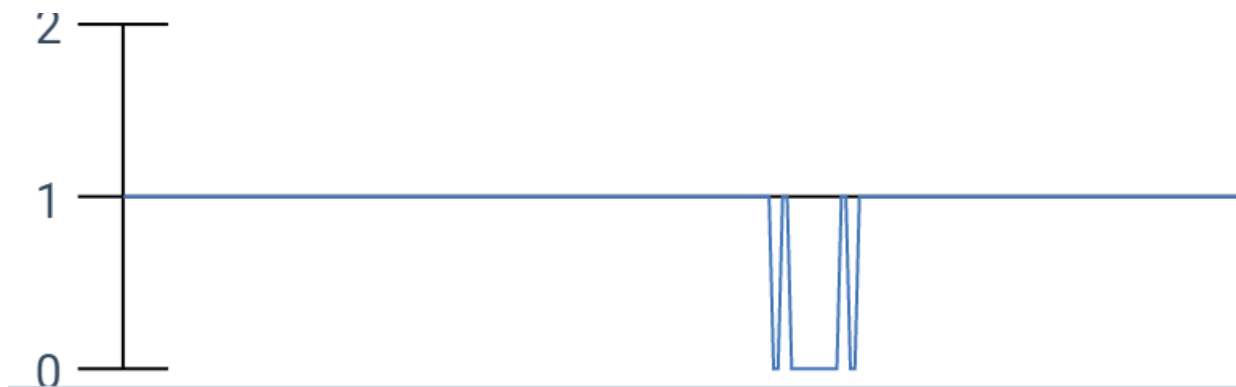


Figure 12 (serial plotter output)

As shown in figure 12 the plotter start with value which mean there is now data or in other word it's the value of the stopped bit then when data received it's become for only one bit cycle then show value as binary from 0 and 1 and it's :-

010000010 which is equal to A in ASCII code that letter has been sent in transmitter code as shown in figure 13 below (Appendix [4.1]) :-

```
1 /*##### Transmitter #####*/|
2 void setup() {
3   // Setup the Serial at 300 Baud
4   Serial.begin(300);
5 }
6 void loop() {
7   Serial.write('A'); //Write the character A => will be transmitted as Byte [41H]
8   delay(500);
9 }
```

Figure 13 (Transmit A Code)

Figure 13 transmit A to the receiver in figure 14 below (Appendix [4.2]) :-

```
1 int readPin = 7;
2 void setup() {
3   // Begin the Serial at 19200 Baud
4   Serial.begin(19200);
5   pinMode(readPin, INPUT);
6 }
7 void loop() {
8   Serial.println(digitalRead(readPin));
9 }
```

Figure 14 (Receiver Code)

Receiver print the value of the interrupt 7 which is connect with the transmitter TX pin and print these value on the Serial Monitor .

3. TO DO:

The aim of these task is to ask user to enter speed value of the motor which taken from Arduino 1 that transit speed value to receive it in Arduino that will run the motor, the circuit was built as shown in figure 15 below :-

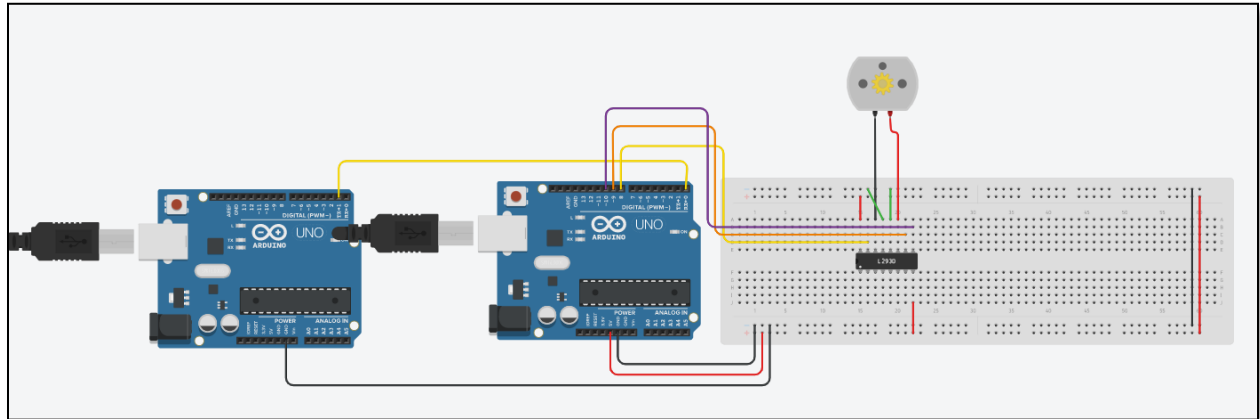


Figure 15 (TO DO design)

Transmitter and receiver code shown below in figure 16 & 17 (Appendix [5.1] & [5.2]) :-

```
Text [Download] [Save] 1 (Arduino Uno R3)
1 // ##### Transmitter #####
2 int speed; // define speed variable
3 void setup()
4 {
5   Serial.begin(9600);
6 }
7
8 void loop()
9 {
10  if(Serial.available()){
11    speed = Serial.parseInt(); /*read speed and put it
12                               in speed variable*/
13
14    if(speed >= 0 && speed <= 255){ /*check if speed between
15                                   0 & 255 */
16      Serial.write(speed);
17    }
18  }
19 }
```

Figure 16 (TO DO Transmitter Code)

As shown in figure 16 for transmitter baud rate defined equal to 9600 , then a helper sentence shown on the Serial Monitor and using if statement to check if there's available data from the user ,if it's store data in speed variable and check if it's not less than 0 and isn't greater than 255 because less than 0 and grater then 255 didn't change the DC motor motion.

```
Text [Download] [Save] 2 (Arduino Uno R3)
1 // ##### Receiver #####
2 int pin3 = 9;          /* define pins and enable variable */
3 int pin4 = 8;
4 int enable34 = 10;
5 void setup()
6 {
7   Serial.begin(9600);
8   pinMode(pin3, OUTPUT); // make pin3 output
9   pinMode(pin4, OUTPUT); // make pin4 output
10  pinMode(enable34, OUTPUT); // make enable34 output
11 }
12 void loop()
13 {
14   if(Serial.available()){
15     int speed = Serial.read(); // read speed
16     analogWrite(enable34, speed); // write speed in enable34
17     digitalWrite(pin3, HIGH); // make pin3 high
18     digitalWrite(pin4, LOW); // make pin4 low
19   }
20 }
```

Figure 17 (TO DO Receiver Code)

Receiver take speed value from transmitter convert it to integer. After that the DC-motors run where depends on the speed that the user has entered.

4. Conclusion:

In this experiment we learn about UART and how to transmit and receive serial data using it also we learn how it can read asynchronous data because serial communication include start and stop bit in addition to detect if the data accurate using parity bit, in addition to that learn about the special functions that used with serial data we apply practically about this in different parts and cases such as : communication between Arduino & PC , communication between 2 Arduinos , Push button & led with Arduinos and how to understand serial plotter .

5. References:

- [1] Lab Manual [Accessed 9:20 July 2020].
- [2] https://en.wikipedia.org/wiki/Serial_communication [Accessed 10:13 July 2020].
- [3] <https://www.javatpoint.com/arduino-serial-serial-begin> [Accessed 11:40 July 2020].
- [3] <https://www.javatpoint.com/arduino-serial-serial-begin> [Accessed 11:40 July 2020].
- [4] https://www.tinkercad.com/things/7Cp3A6Vuv5M-exp31/editel?sharecode=5fqghuqRTWfFK_M2WQEU0eGeI-PYIMv5r6ZPfvuzvjs [part1].
- [5] https://www.tinkercad.com/things/1wWgbFzld5-exp3-2/editel?sharecode=2LYU_93XHZ57HM9lhaeKTEEz6sOEkHOnoZH-jJQ_RtI [part2].
- [6] https://www.tinkercad.com/things/i0yEG3Uc33V-exp3-3/editel?sharecode=7PxQu-n4Xh0ASNcheC-mKV63F4ohcT1hsaBKxL1wt_A [part3].
- [7] <https://www.tinkercad.com/things/1b8AULWwA1r-exp3-4/editel?sharecode=8HsZMbNHCGvoJxoeGRE4HaVurxvQWujLkV2-NOEbCMI> [part4].
- [8] https://www.tinkercad.com/things/kUSgCK88T6r-todo2anas/editel?sharecode=vszVDA0UZgoQgvjIXK_ldAZpkXPCsosyLcoD3mfRpNU [todo].

6. Appendix:

6.1. Part1:

```
/* Use a variable called byteRead to temporarily store
the data coming from the computer */
byte byteRead;
void setup() {
// Turn the Serial Protocol ON
Serial.begin(9600);
}
void loop() {
/* check if data has been sent from the computer: */
if (Serial.available()) {
/* read the most recent byte */
byteRead = Serial.read();
/*ECHO the value that was read, back to the serial port. */
Serial.write(byteRead);
}
}
```

6.2. Part2:

- **Part2.1:**

```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  Serial.println("Hello World!"); //Write the serial data
  delay(1000);
}
```

- **Part2.2:**

```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  if (Serial.available()) {
    byte data = Serial.read(); //Read the serial data and store in var
    Serial.write(data); //Print data on Serial Monitor
  }
}
```

6.3. Part3:

- **Part3.1:**

```
int mode = 0;
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  if (Serial.available()) {
    int data = Serial.parseInt();
    Serial.write(data);
  }
}
```

- **Part3.2:**

```
void setup()
{
  pinMode(7, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  if(Serial.available()){
    byte data = Serial.read();
    Serial.println(data);
  }
}
```

6.4. Part4:

- **Part4.1:**

```
void setup()
{
  Serial.begin(300);

}

void loop()
{
  Serial.write('A'); //Write the character A => will be transmitted as Byte [41H]
  delay(500);
}
```

- **Part4.2:**

```
int readPin = 7;
void setup() {
  // Begin the Serial at 19200 Baud
  Serial.begin(19200);
  pinMode(readPin, INPUT);
}
void loop() {
  Serial.println(digitalRead(readPin));
}
```


6.5. Todo:

- **Part1:**

```
// ##### Transmitter #####
int speed; // define speed variable
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  if(Serial.available()){
    speed = Serial.parseInt(); /*read speed and put it
                               in speed variable*/

    if(speed >= 0 && speed <= 255){ /*check if speed between
                                   0 & 255 */
      Serial.write(speed);
    }
  }
}
```

- **Part2:**

```
// ##### Receiver #####
int pin3 = 9;    /* define pins and enable variable */
int pin4 = 8;
int enable34 = 10;
void setup()
{
  Serial.begin(9600);

  pinMode(pin3, OUTPUT); // make pin3 output
  pinMode(pin4, OUTPUT); // make pin4 output
  pinMode(enable34, OUTPUT); // make enable34 output
}
void loop()
{
```

```
if(Serial.available()){  
    int speed = Serial.read(); // read speed  
    analogWrite(enable34, speed); // write speed in enable34  
    digitalWrite(pin3, HIGH); // make pin3 high  
    digitalWrite(pin4, LOW); // make pin4 low  
}  
}
```