Microprocessors Lab

5- Serial Communication Unit

Instructor: Dr. Mohsen Raji

Graders:

- Mohammad Hossein Hashemi
- Reza Hesami
- Negin Shirvani

Shiraz University – Fall 2022

Based on the slides by:

- Hossein Dehghanipour
- Mohammad Hossein Allah Akbari

** Reminder

What is GPIO?

GPIO is the Input/Output that we normally use. They gave it a fancy name just to make it look cooler.

Honestly, GPIO a little more than that, but currently, that's all we need to know.

If you want to feel more "nerdish" or boast to your friends about how much you know, visit this link [Wikipedia].

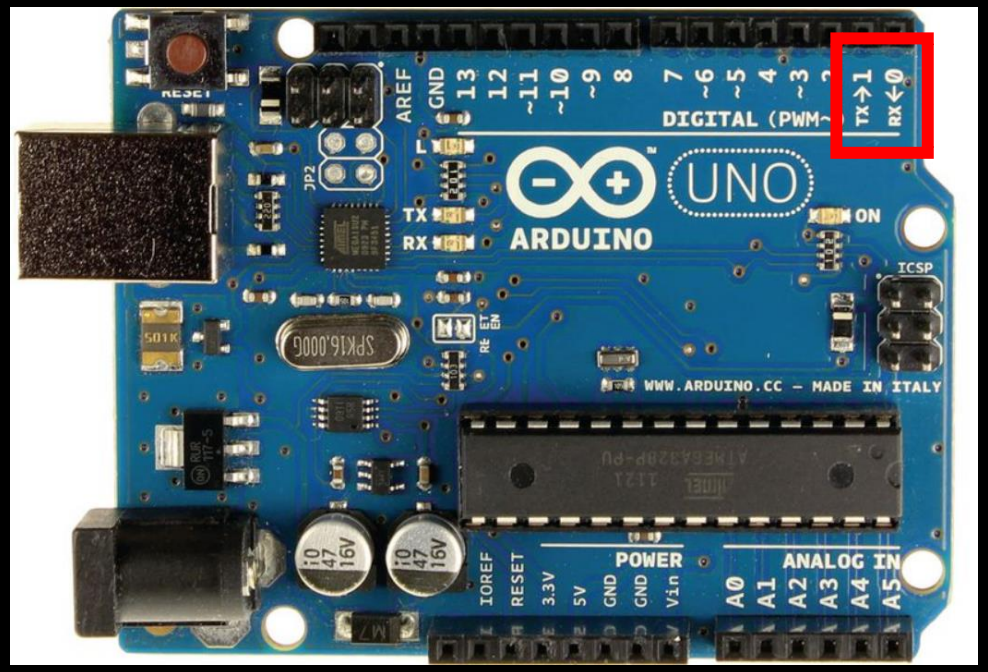
Serial Communication Unit
In order to transfer data between Arduino and other boards/peripherals/devices, we can use a unit called Serial Communication Unit.

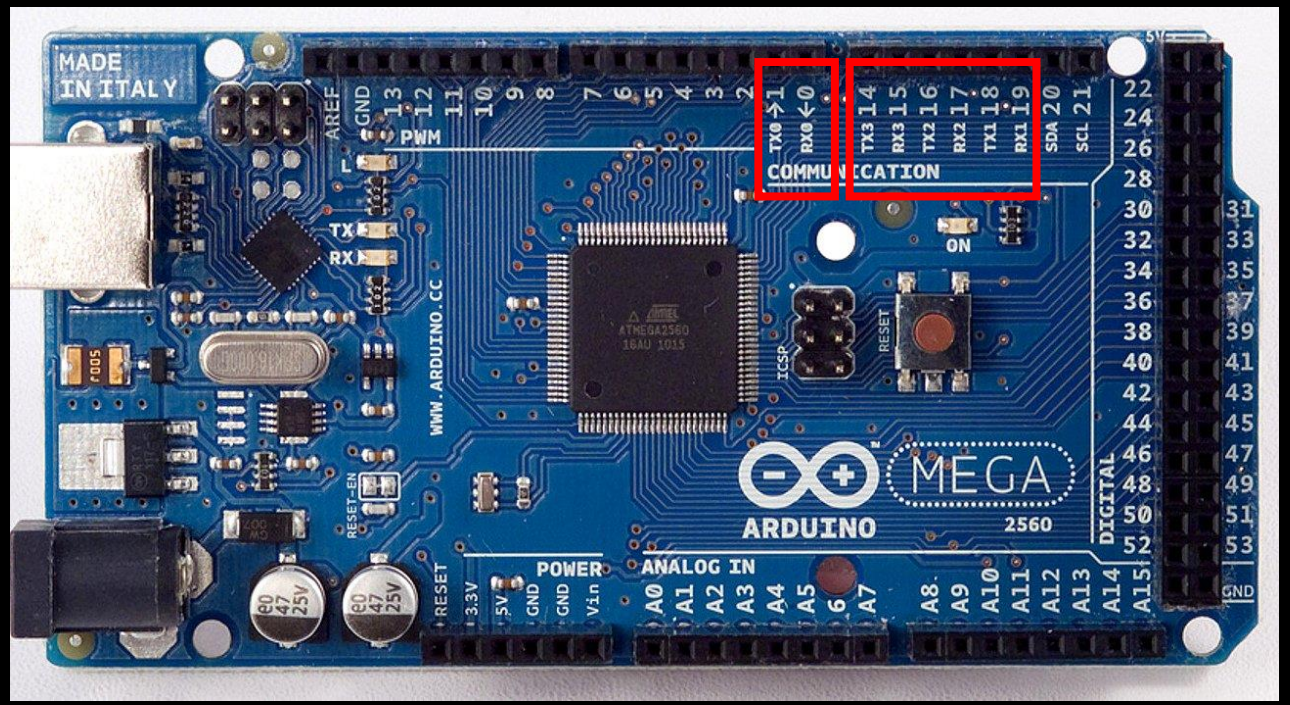
We use some libraries that make the job easier for us, but, we can manually do this task by changing the registers (that you got familiar with in the Micro Processor course).

Arduino Uno has 1 SCU.
Arduino Mega has 4 SCUs.

RX and TX are the ports that Arduino uses to Receive Data and Transfer Data, respectively.

Arduino supports both USART and UART.





How do we use it?

That's pretty easy.

List of All of The Functions

There are several functions for this job, but we are going to use a few simple ones.

Just a few; you know, Not so many.

All Functions

Functions

if(Serial)
available()
availableForWrite()
begin()
end()
find()
findUntil()
flush()
parseFloat()
parseInt()
peek()
print()
println()
read()
readBytes()
readBytesUntil()
readString()
readStringUntil()
setTimeout()
write()
serialEvent()

What we need

begin()
available()
read()
readString()
write()
print()
println()
parseInt()
parseFloat()

Picture Reference: Fullmetal Alchemist.

Serial Class

- In order to use the Serial ports we must use the Serial class.
- Arduino Uno has 1 Serial Communication unit called:
  - Serial
- Arduino Mega has 4 Serial Communication units called:
  - Serial
  - Serial1
  - Serial2
  - Serial3

begin()

Arduino Mega using all four of its Serial ports (Serial, Serial1, Serial2, Serial3), with different baud rates:

```
void setup() {
  Serial.begin(9600);
  Serial1.begin(38400);
  Serial2.begin(19200);
  Serial3.begin(4800);

  Serial.println("Hello Computer");
  Serial1.println("Hello Serial 1");
  Serial2.println("Hello Serial 2");
  Serial3.println("Hello Serial 3");
}
```

Description

Sets the data rate in bits per second (baud) for serial data transmission. You can, specify other rates rather than the ones are specified - for example, to communicate over pins 0 and 1 with a component that requires a particular baud rate.

An optional second argument configures the data, parity, and stop bits. The default is 8 data bits, no parity, one stop bit.

Syntax

```
Serial.begin(speed)
Serial.begin(speed, config)
```

Parameters

Serial: serial port object. See the list of available serial ports for each board on the Serial main page.

speed: in bits per second (baud rate).
Allowed data types: long.

config: sets data, parity, and stop bits.
Valid values are:
SERIAL_5N1, SERIAL_6N1, SERIAL_7N1, SERIAL_8N1 (the default),
SERIAL_5N2, SERIAL_6N2, SERIAL_7N2, SERIAL_8N2, SERIAL_5E1: even parity,
SERIAL_6E1, SERIAL_7E1, SERIAL_8E1, SERIAL_5E2, SERIAL_6E2,
SERIAL_7E2, SERIAL_8E2, SERIAL_5O1: odd parity, SERIAL_6O1
SERIAL_7O1, SERIAL_8O1, SERIAL_5O2, SERIAL_6O2, SERIAL_7O2,
SERIAL_8O2

Returns

Nothing

available()

- Description
  - Get the number of bytes (characters) available for reading from the serial port. This is data that's already arrived and stored in the serial receive buffer (which holds 64 bytes).
- Syntax

  Serial.available()
- Parameters
  - *Serial*: serial port object. See the list of available serial ports for each board on the Serial main page.
- Returns
  - The number of bytes available to read.

read()

```
int incomingByte = 0;
// for incoming serial data

void setup() {
  Serial.begin(9600);
// opens serial port, sets baud rate to 9600 bps
}

void loop() {
  // send data only when you receive data:
  if (Serial.available() > 0) {
    // read the incoming byte:
    incomingByte = Serial.read();

    // say what you got:
    Serial.print("I received: ");
    Serial.println(incomingByte, DEC);
  }
}
```

Description

Reads incoming serial data.

Syntax

Serial.read()

Parameters

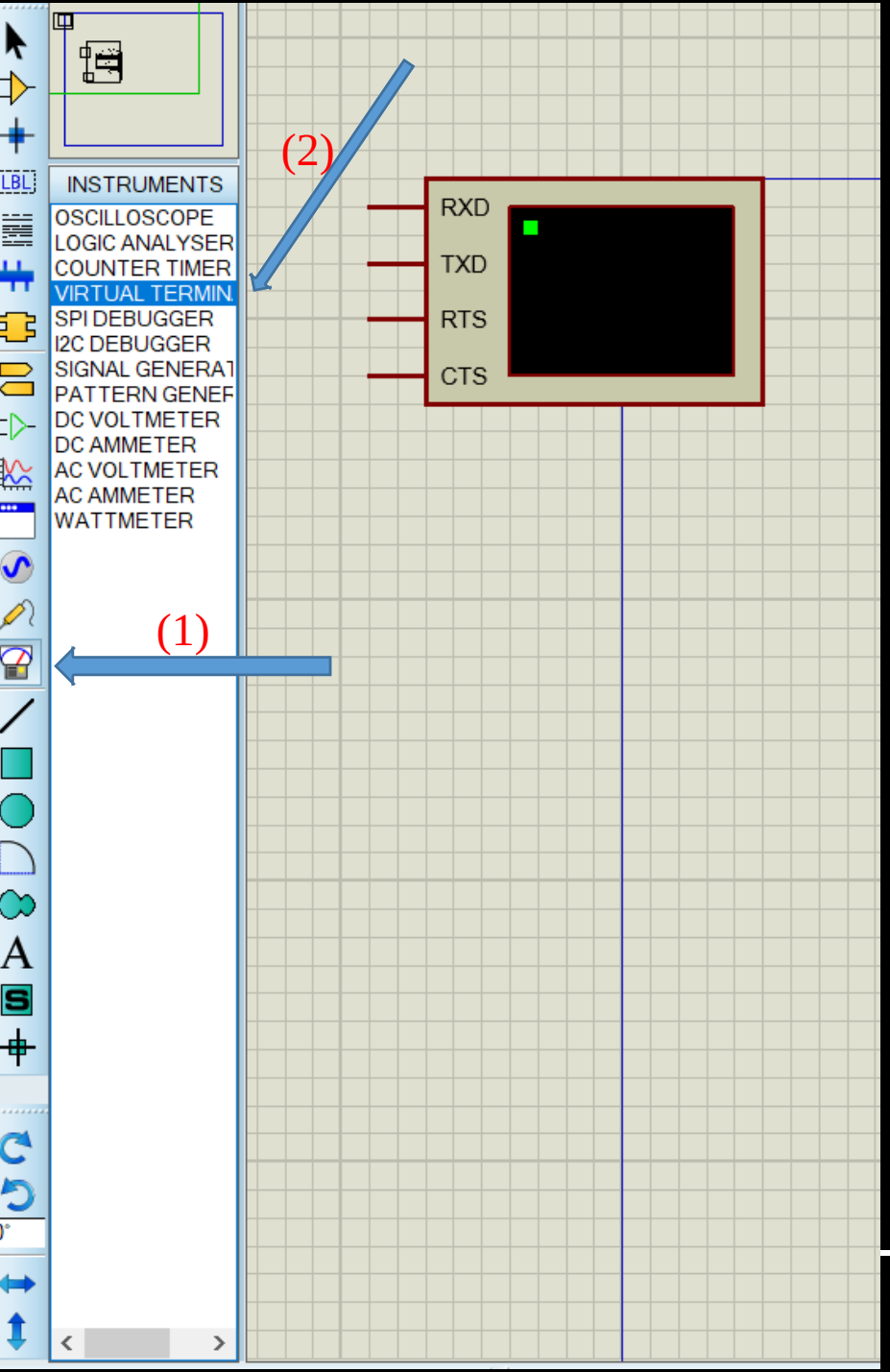
Serial: serial port object. See the list of available serial ports for each board on the Serial main page.

Returns : int

The first byte of incoming serial data available (or -1 if no data is available).

Before You Continue, Please read the following links:

- Serial.write()
- Serial.print()
- Serial.println()
- Serial.parseFloat()
- Serial.parseInt()
- Serial.readString()



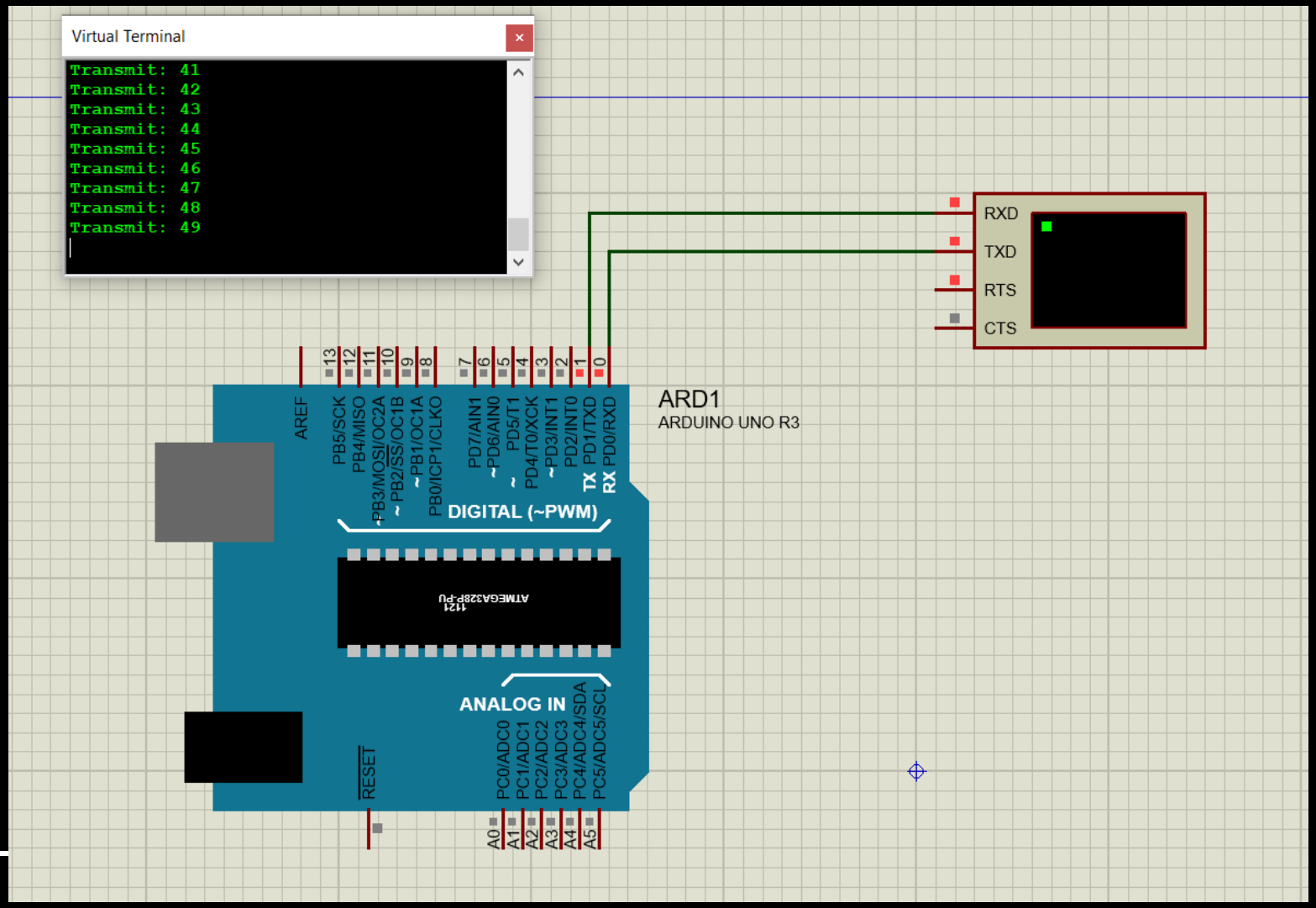
Virtual Terminal

- In the Proteus, there is a tool called Virtual Terminal. We can send String or Integer data using this terminal. You can find this device as the picture has depicted below.

- The Terminal has 4 pins:
  - RXD: Receive Data
  - TXD: Transmit Data
  - RTS: Request to Send
  - CLS: Clear to Send

For further readings about terminal, click on Link 1 .

Sample 1

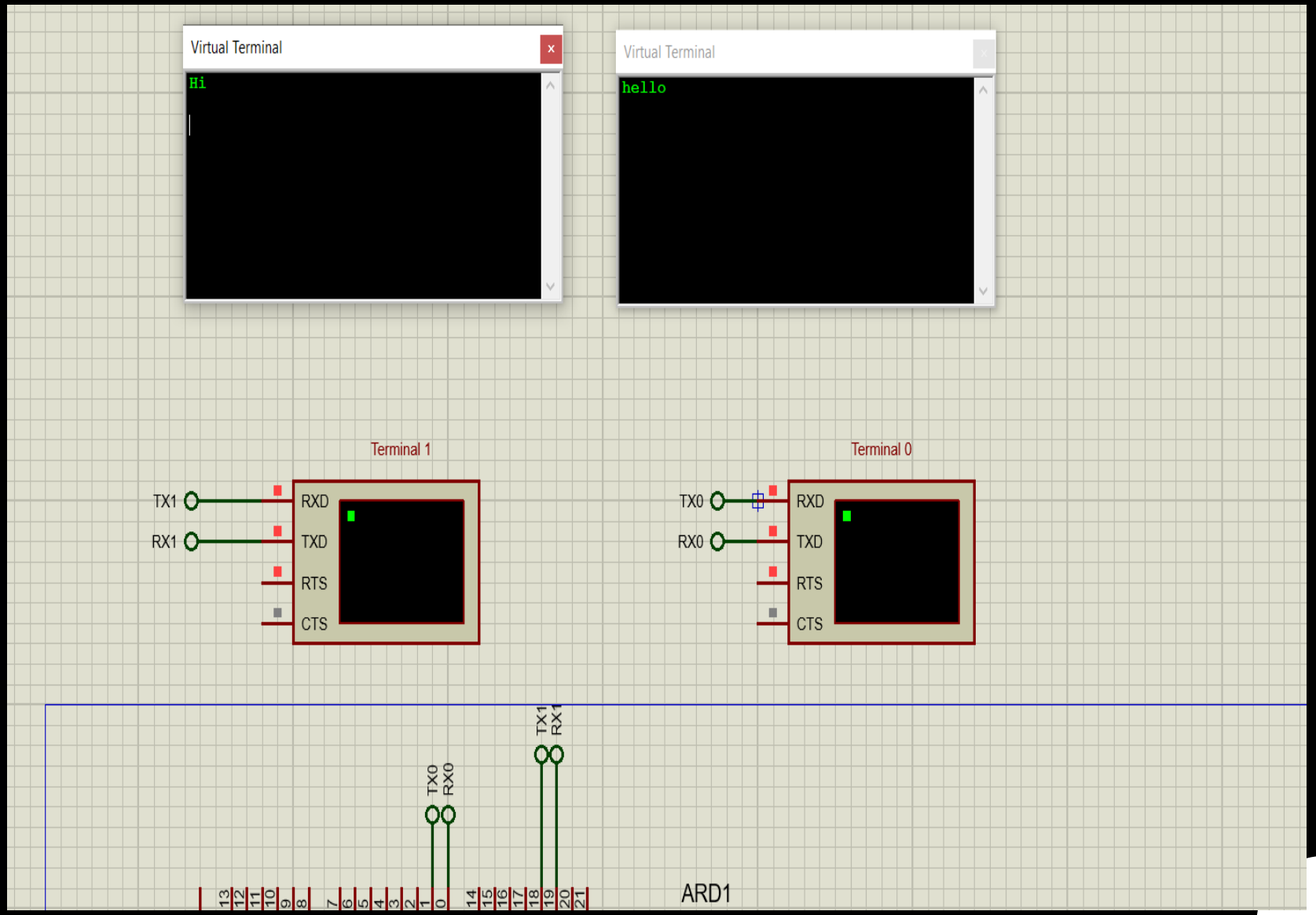



```arduino
#include <Arduino.h>
int i = 0 ;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600); //
}

void loop() {
  // put your main code here, to run repeatedly:
    if(Serial.available() > 0 ){
      Serial.print("Transmit: ");
      Serial.println(i++,DEC);

    delay(100);
  }

}
```

Sample II

There are some problems with this task… run the code to realize what I mean.

```
void loop() {
  // put your main code here, to run repeatedly:
    String str1 = Serial1.readString();
    String str0 = Serial.readString();
    if ( str1 != "" && str1 != "\n"){
      Serial.println(str1);
      delay(100);
    }

    if ( str0 != "" && str0 != "\n"){
      Serial1.println(str0);
      delay(100);
    }
```



SoftwareSerial

Functions

- SoftwareSerial()
- available()
- begin()
- isListening()
- overflow()
- peek()
- read()
- print()
- println()
- listen()
- write()

- If you haven't noticed yet, Arduino Uno has only one USART unit, which makes it impossible for us to connect Serial peripherals to this board. So what's the solution? There is a library called SoftwareSerial. This library allows us to use other pins as USART ports. It simulates another USART unit on other pins.
- The functions are pretty much the same with our standard serial unit but slightly different.
- Reading about this part is also UP TO YOU because bringing everything that is on the website into the documentation is redundant. This link gives you whatever you need to know about this library. Documentation Link

Bonus Score: Send the keyword “I am as fast as Avangard” to your TA. Luckily, if you are the first person who has sent this message to the TA, it means that you are the first student who has started reading the slides faster and earlier than the others, you will receive a premium package of receiving 0.5 point out of 20 as a bonus.

If you have any questions, Google it. The more you search the faster you will become in finding your answers online.
You may one day be able to get to your target even faster than the Avangard Hypersonic Missile (which can reach 27 Mochs).

Picture Reference: https://www.deviantart.com/tunaozcelik/art/Rainbow-Six-Siege-Wallpaper-723960147