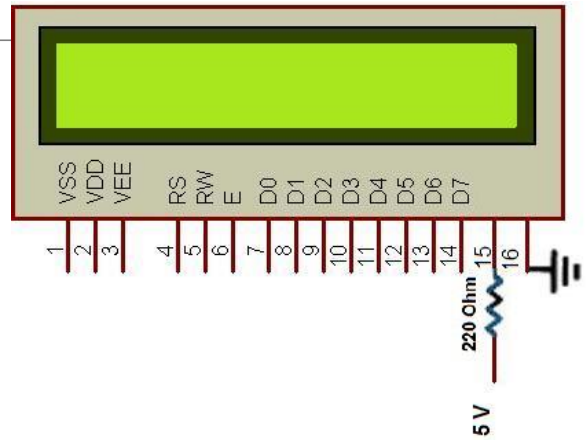
*Embedded System Design & Application*

# Serial Communication

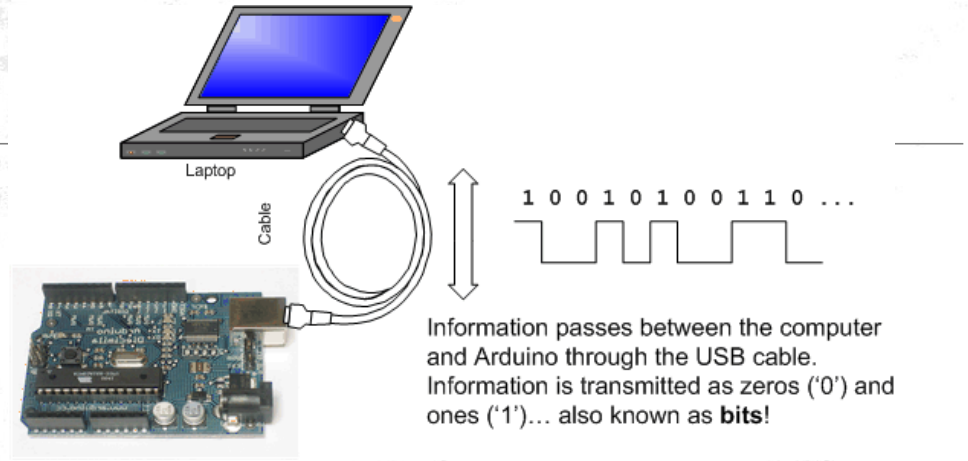
Dr. Sarwan Singh

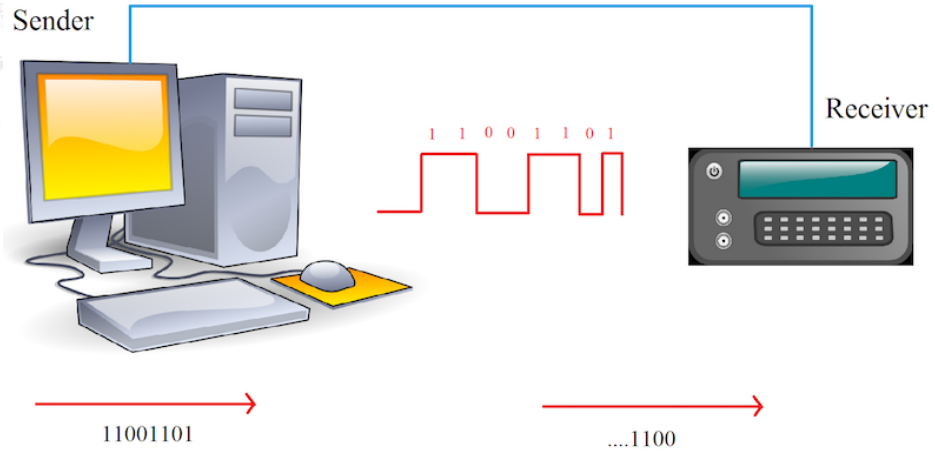
NIELIT Ropar



# Agenda

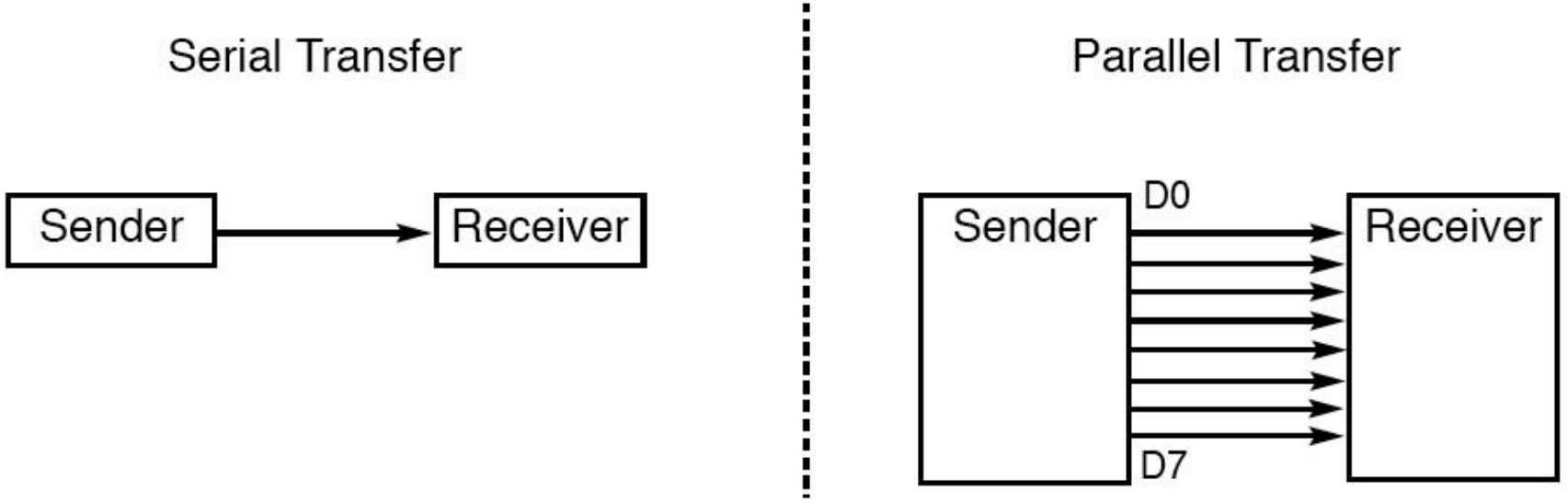
- Introduction
- Software Serial
- Interfacing devices on serial port
  - magnetic card reader
  - Bar code scanner




Communication is exchange of information between two devices

# Basics of Serial Communication




Serial versus Parallel Data Transfer

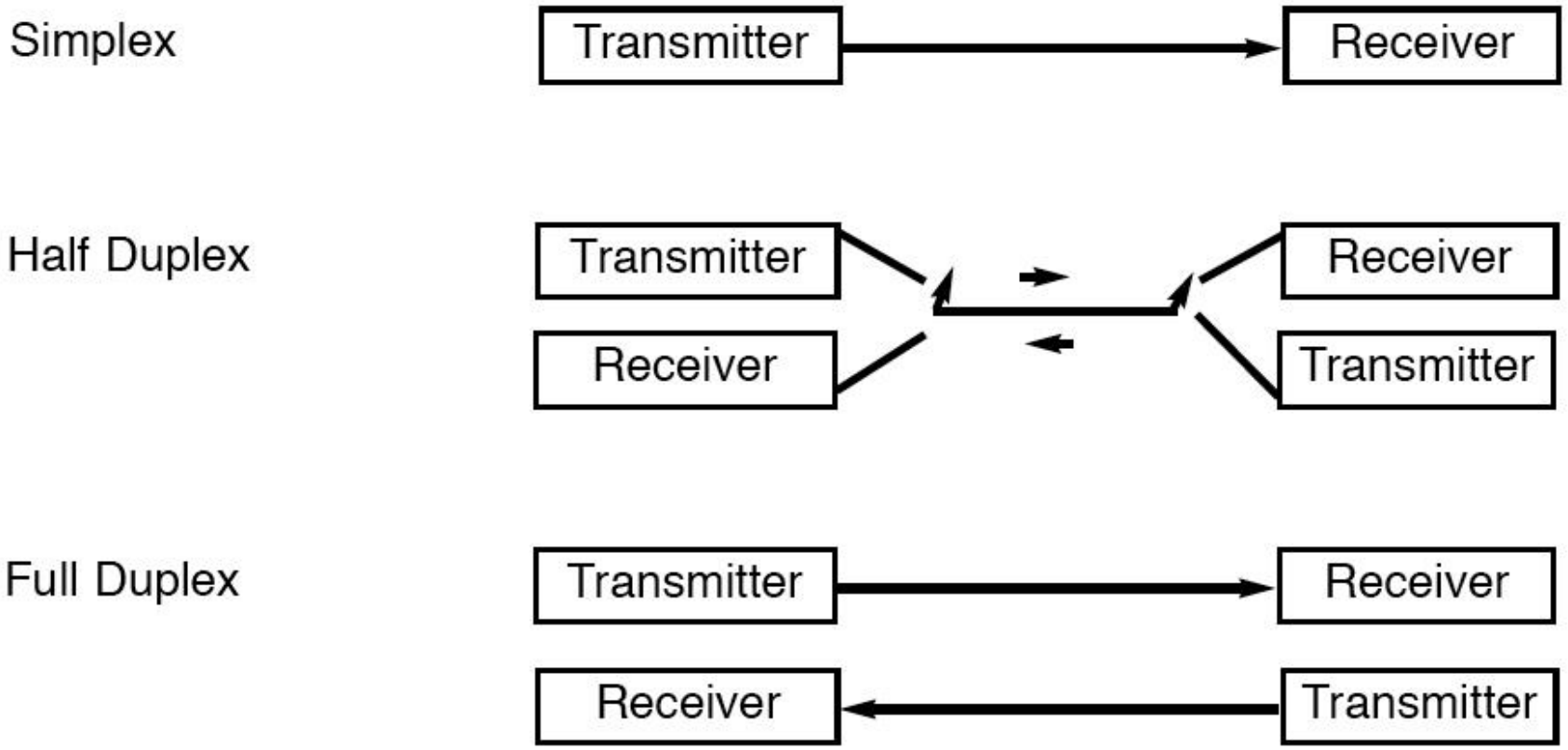
# BASICS OF SERIAL COMMUNICATION

- serial communication uses single data line making it much cheaper
- enables two computers in different cities to communicate over the telephone
- byte of data must be converted to serial bits using a parallel-in-serial-out shift register and transmitted over a single data line
- receiving end there must be a serial-in-parallel-out shift register
- if transferred on the telephone line, it must be converted to audio tones by *modem*
- for short distance the signal can be transferred using wire

# BASICS OF SERIAL COMMUNICATION

- Two methods, asynchronous and synchronous
  - *synchronous* method transfers a block of data (characters) at a time
  - *asynchronous* method transfers a single byte at a time
- Uses special IC chips called
  - **UART** (universal asynchronous receiver-transmitter) and
  - **USART** (universal synchronousasynchronous receiver-transmitter)
- 8051 chip has a built-in UART

# BASICS OF SERIAL COMMUNICATION




Simplex, Half-, and Full-Duplex Transfers

# BASICS OF SERIAL COMMUNICATION

- Half- and full-duplex transmission
  - if the data can be transmitted and received, it is a *duplex* transmission
  - *simplex* transmissions the computer only sends data
  - duplex transmissions can be half or full duplex
  - depends on whether or not the data transfer can be simultaneous
  - If one way at a time, it is *half duplex*
  - If can go both ways at the same time, it is full duplex
  - full duplex requires two wire conductors for the data lines (in addition to the signal ground)

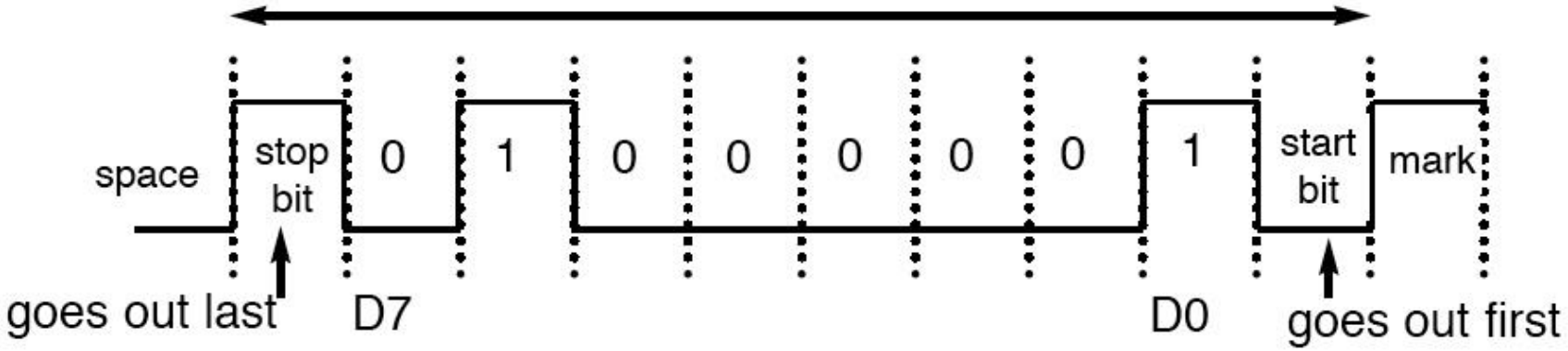
# BASICS OF SERIAL COMMUNICATION

- Asynchronous serial communication and data framing
  - data coming in 0s and 1s
  - to make sense of the data sender and receiver agree on a set of rules
  - Protocol
    - how the data is packed
    - how many bits/character
    - when the data begins and ends

# BASICS OF SERIAL COMMUNICATION

- Start and stop bits
  - asynchronous method, each character is placed between start and stop bits  called *framing*
  - start bit is always one bit
  - stop bit can be one or two bits
  - start bit is always a 0 (low)
  - stop bit(s) is 1 (high)
  - LSB is sent out first

# BASICS OF SERIAL COMMUNICATION




Framing ASCII “A” (41H)

# BASICS OF SERIAL COMMUNICATION

- in modern PCs one stop bit is standard
- when transferring a text file of ASCII characters using 1 stop bit there is total of 10 bits for each character
- 8 bits for the ASCII code (1 parity bit), 1 bit each for the start and stop bits
- for each 8-bit character there are an extra 2 bits, which gives 20% overhead

# BASICS OF SERIAL COMMUNICATION

- Data transfer rate
  - rate of data transfer *bps* (bits per second)
  - widely used terminology for bps is *baud rate*
  - baud and bps rates are not necessarily equal
  - baud rate is defined as the number of signal changes per second

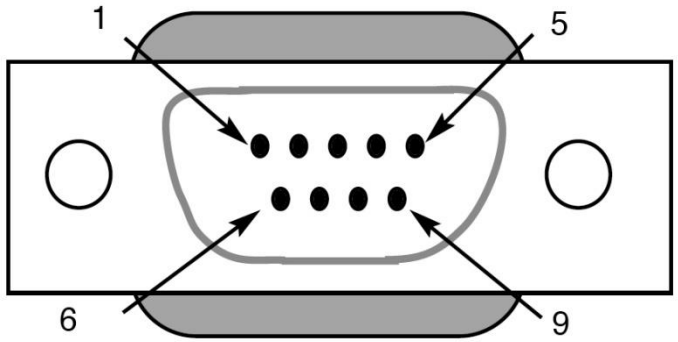
# BASICS OF SERIAL COMMUNICATION

- RS232 standards
  - most widely used serial I/O interfacing standard
  - input and output voltage levels are not TTL compatible
  - 1 bit is represented by -3 to -25 V
  - 0 bit is +3 to +25 V
  - -3 to +3 is undefined
  - to connect RS232 to a microcontroller system must use voltage converters such as MAX232 to convert the TTL logic levels to the RS232 voltage levels, and vice versa
  - MAX232 IC chips are commonly referred to as line drivers

# BASICS OF SERIAL COMMUNICATION

| Pin | Description |
|---|---|
| 1 | Data carrier detect ($\overline{DCD}$) |
| 2 | Received data (RxD) |
| 3 | Transmitted data (TxD) |
| 4 | Data terminal ready (DTR) |
| 5 | Signal ground (GND) |
| 6 | Data set ready ($\overline{DSR}$) |
| 7 | Request to send ($\overline{RTS}$) |
| 8 | Clear to send ($\overline{CTS}$) |
| 9 | Ring indicator (RI) |

IBM PC DB-9 Signals

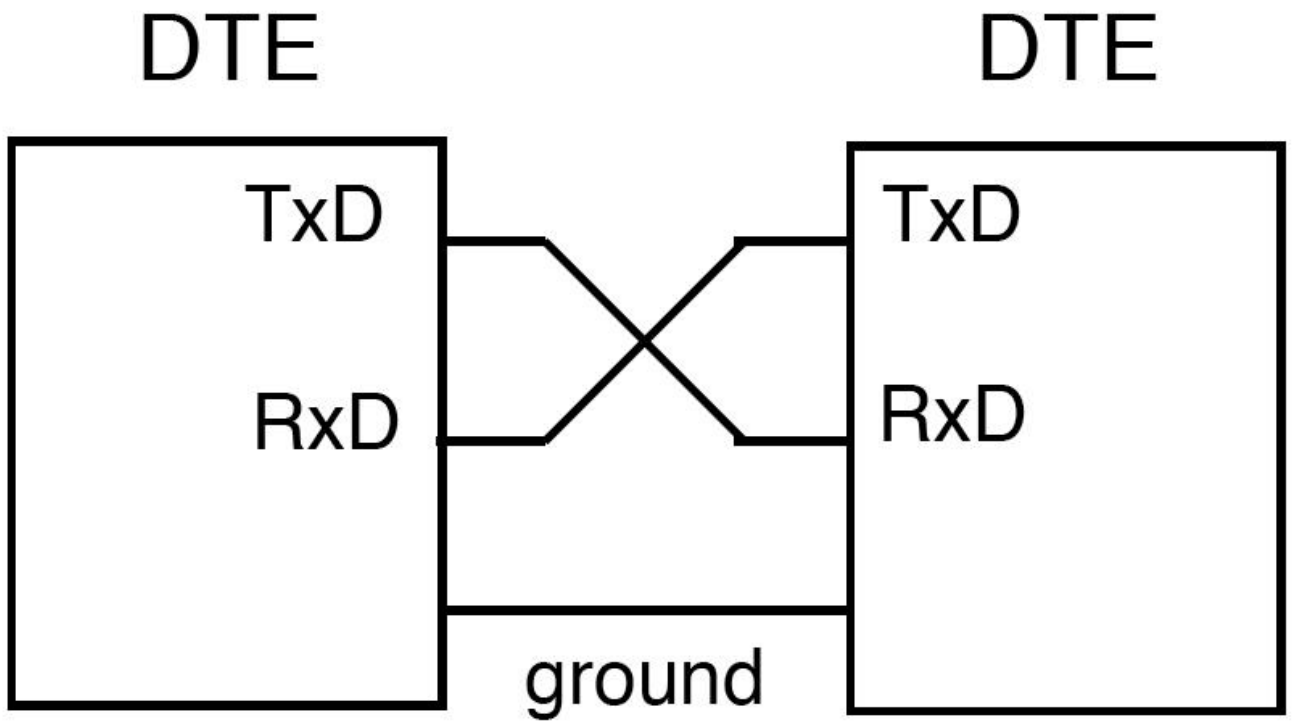



DB-9 9-Pin Connector

# BASICS OF SERIAL COMMUNICATION

- Data communication classification
  - DTE (data terminal equipment)
  - DCE (data communication equipment)
  - DTE - terminals and computers that send and receive data
  - DCE - communication equipment responsible for transferring the data
  - simplest connection between a PC and microcontroller requires a minimum of three pins, TxD, RxD, and ground

# BASICS OF SERIAL COMMUNICATION




Null Modem Connection

# BASICS OF SERIAL COMMUNICATION

- Examining RS232 handshaking signals
  - many of the pins of the RS-232 connector are used for handshaking signals
  - they are not supported by the 8051 UART chip

# BASICS OF SERIAL COMMUNICATION

- PC/compatible COM ports
  - PC/compatible computers (Pentium) microprocessors normally have two COM ports
  - both ports have RS232-type connectors
  - COM ports are designated as COM 1 and COM 2 (replaced by USB ports)
  - can connect the 8051 serial port to the COM 2 port

# Arduino Serial Communication

- The main purpose of this serial communication is to transfer the sketch from computer to Arduino, send information to computer etc.
- The most common type of serial communication protocol is UART i.e. Universal Asynchronous Receiver Transmitter. The UART connection requires a 9 pin DE – 9 Connector.
- Arduino UNO has an on board serial to USB converter and hence we can directly connect the Arduino to the computer.
- Using this USB connection and Arduino IDE, we can send data to Arduino or receive data from Arduino. This data can be monitored with the help of Serial Monitor in the Arduino IDE.

# Software Serial

- The Arduino Uno has built-in support for serial communication on pins 0 and 1 (which also goes to the computer via the USB connection).
- The native serial support happens via a piece of hardware (built into the chip) called a UART.
- This hardware allows the Atmega chip to receive serial communication even while working on other tasks, as long as there room in the 64 byte serial buffer.
- The SoftwareSerial library has been developed to allow serial communication on other digital pins of the Arduino, using software to replicate the functionality (hence the name "SoftwareSerial").

Reference : https://www.arduino.cc/en/Reference/SoftwareSerial

# SoftwareSerial

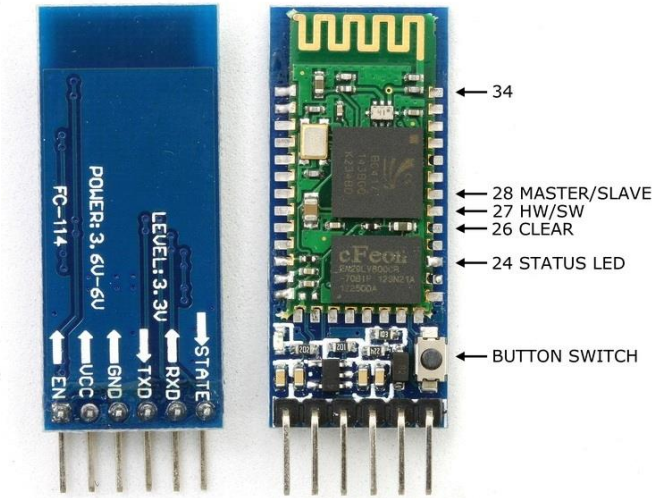
- It is possible to have multiple software serial ports with speeds up to 115200 bps.

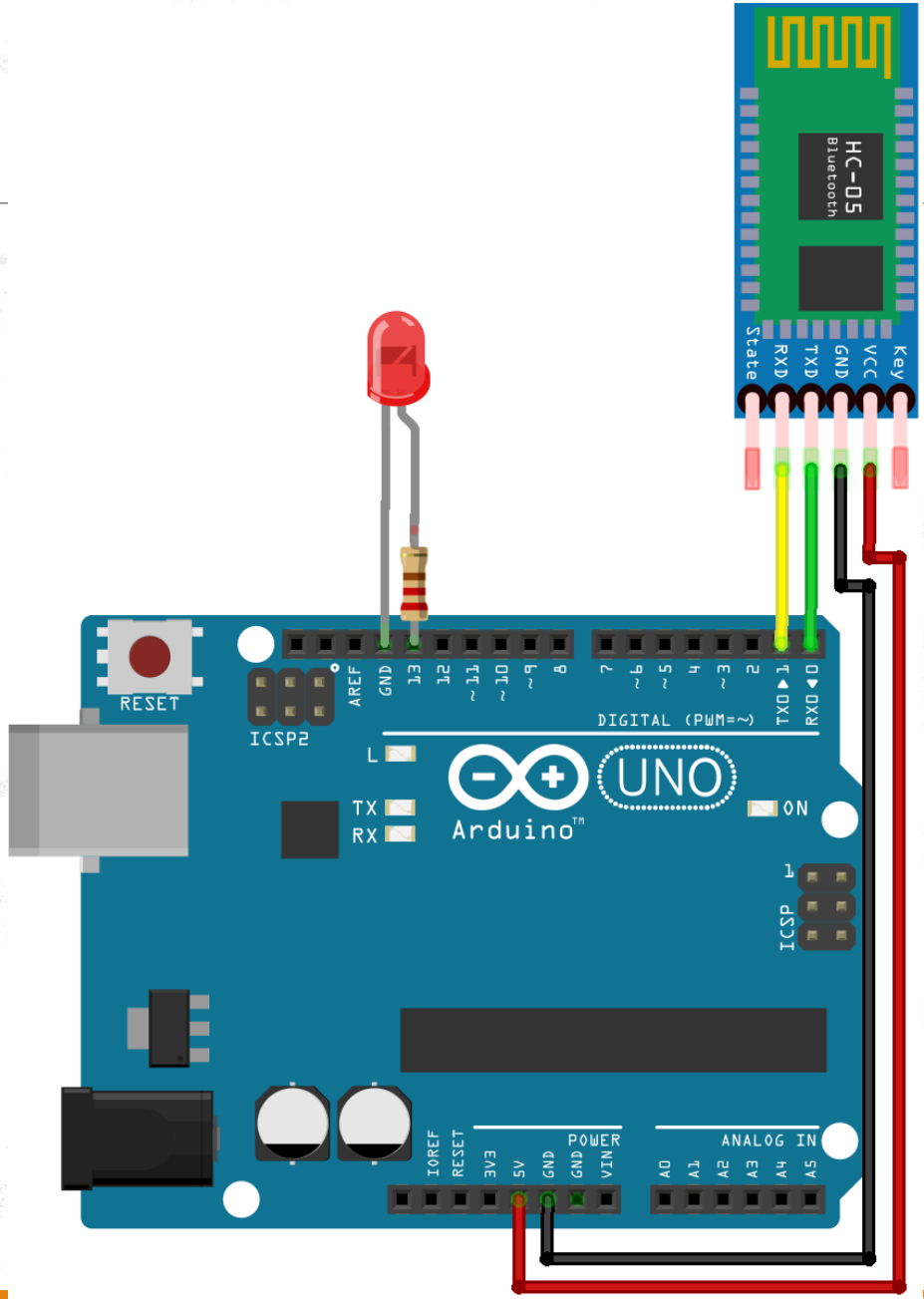
Limitations

- If using multiple software serial ports, only one can receive data at a time.
- Not all pins on the Mega and Mega 2560 support SoftwareSerial

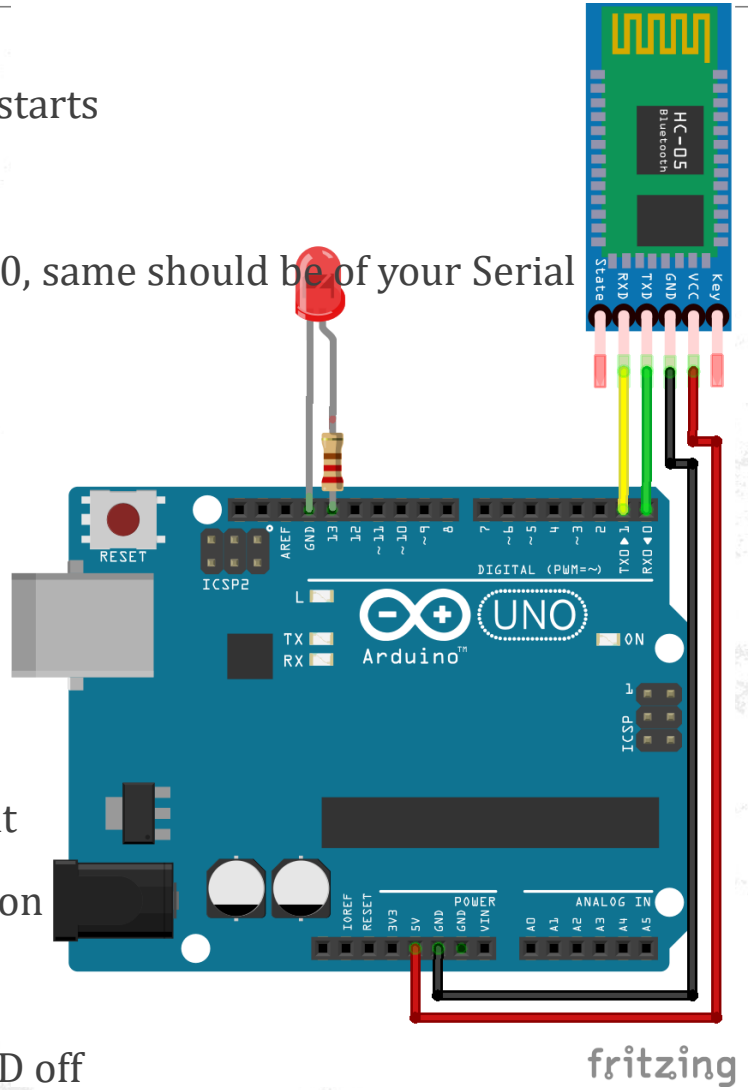
# Serial comm. examples

- Interfacing Magnetic Card Reader/Bar Code Reader
- Interfacing Blutooth-HC-05


HC-05
Bluetooth
State
RXD
TXD
GND
VCC
Key
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
UNO
Arduino™
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
fritzing

```
1.  /*Arduino Turn LED On/Off using Serial Commands
2.  a (to turn the LED on)
3.  b (to turn the LED off)   */
4.   void setup()                  // run once, when the sketch starts
5.  {
6.   Serial.begin(9600);          // set the baud rate to 9600, same should be of your Serial
    Monitor
7.   pinMode(13, OUTPUT);
8.  }
9.  void loop()
10. {   while(Serial.available())
11.     {
12.      char inChar = (char)Serial.read(); //read the input
13.      if(inChar == 'a'){       //in case of 'a' turn the LED on
14.        digitalWrite(13, HIGH);
15.      }else if(inChar == 'b'){  //incase of 'b' turn the LED off
16.        digitalWrite(13, LOW);
```



# Happy Coding

JOURNEY BEGINS FROM HERE……

- Github Repo

https://github.com/sarwansingh/IoT/tree/master/ClassExamples/CEPTAM_ESDA_Sept24