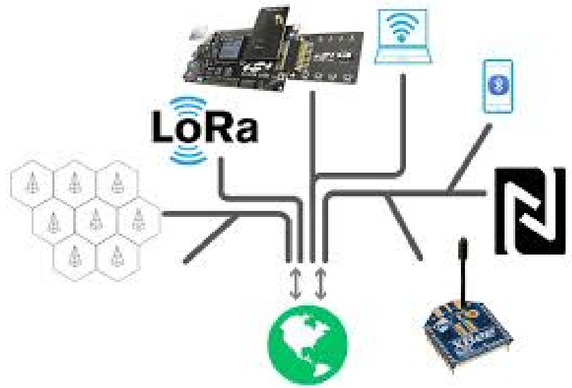
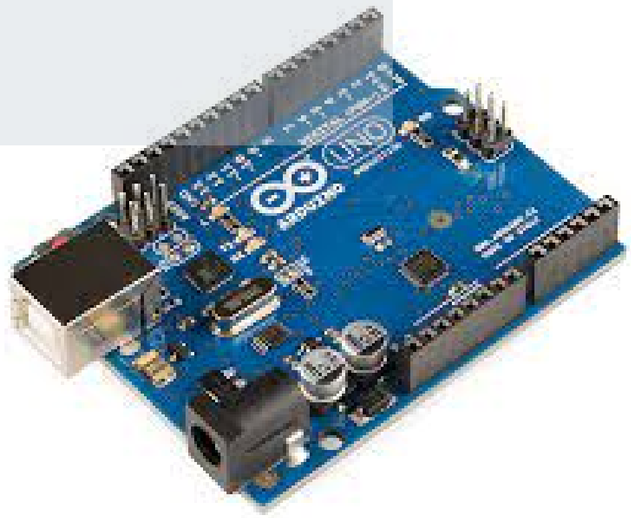
SnS College Of Technology

Department Of Ece

IOT System Architecture

Arduino Interfacing and communication



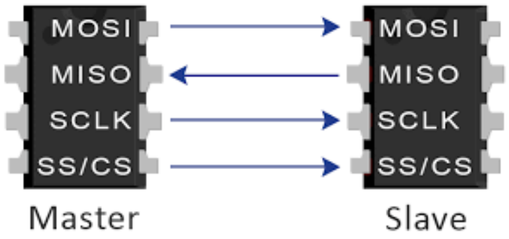


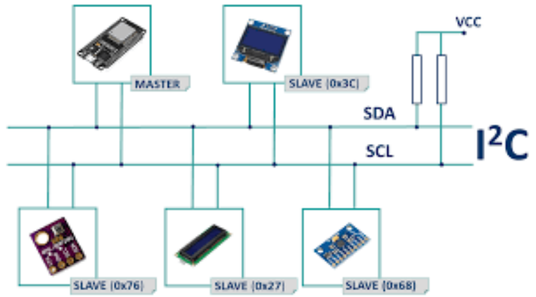
Interfacing

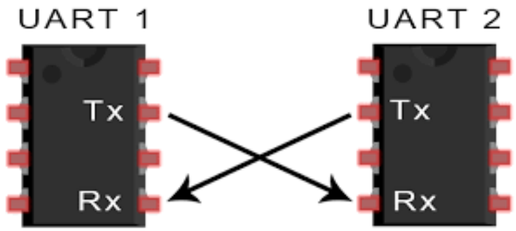
- The process of connecting external peripheral devices such as sensors , actuators and other external modules with different communication protocols with microcontroller is called interfacing.

- The Arduino UNO supports different protocols for inter communication.
- **SPI**
- **I2C**
- **UART**

- It also contains internal timers , pwm to produce **time delay , dimming of lights , speed control of servo motors**.

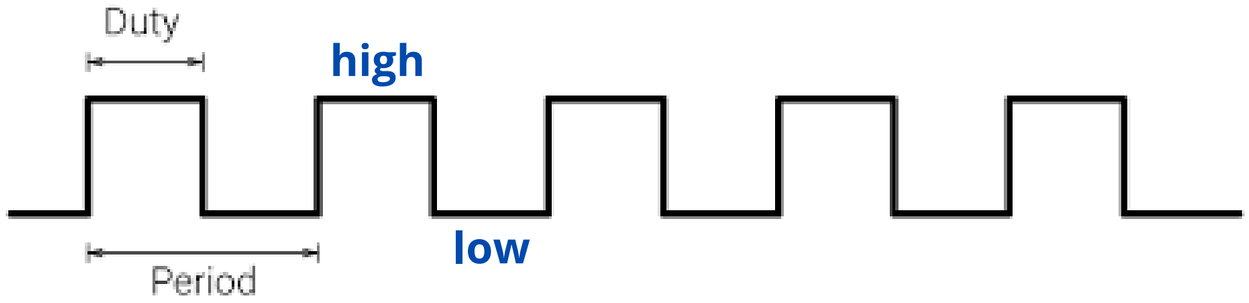






Pulse Width Modulation

- PWM stands for Pulse Width Modulation
- Used in Inverters, dimming led , rgb led , Controlling speed of motors
- **PWM is used to produce Analog signals from a digital device like microcontroller**.
- Consider an led , which has only two states - **ON or OFF**
  - To dim the led , the supply must be turned on and off continously for certain time interval.
  - This can be done by **PWM** which is a **square wave**.



- The time to complete one on and off cycle is called **Period**.
- The duration at which the signals stays high is called the “**on time**” and the duration at which the signal stays low is called as the “**off time**”.

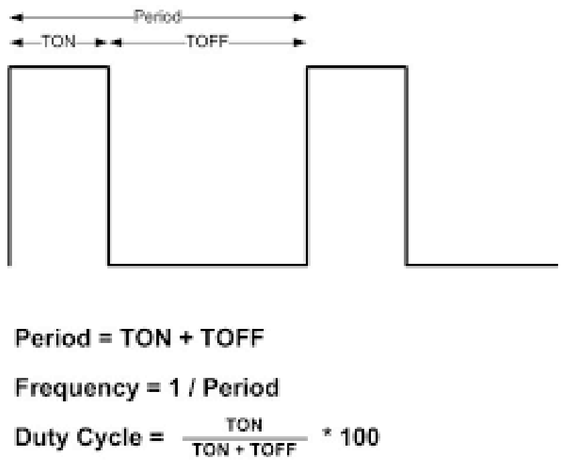
Duty cycle and frequency

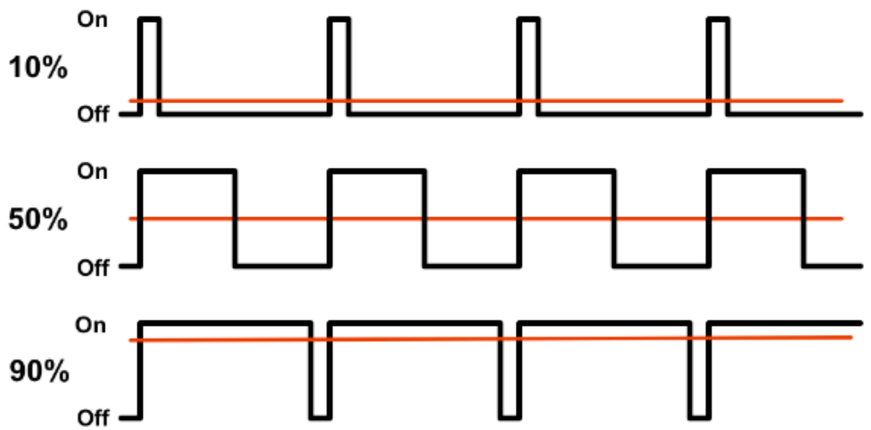
Duty Cycle

- The percentage of time in which the PWM signal remains **HIGH** (on time) is called as **duty cycle**.
- By controlling the **Duty cycle from 0% to 100%** the “on time” of PWM signal and width of signal can be altered.
- By altering duty cycle , the **average output** can be obtained to dim led or control speed of motors.

Frequency

- The The frequency of a PWM signal determines how fast a PWM completes one period.
- **frequency = 1 / Period**

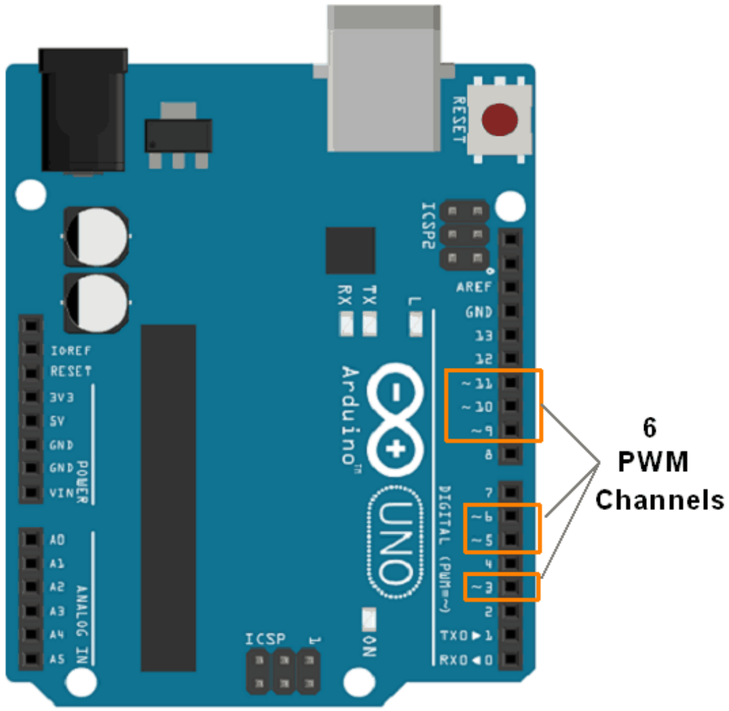




Different duty cycles

PWM in Arduino

- Arduino Uno has **6** **8-bit** PWM channels.
- The pins with symbol **'~'** represents that it has PWM support.]
- The digital pins 3 , 5 , 6 , 9 , 10 , 11 provides PWM output.



Arduino PWM Funcitons

- **analogWrite (pin, duty cycle)**
- It is used to generate PWM or output analog value to a specified PWM channel.
- pin – pin on which we want to generate pwm or analog signal.
- duty cycle – it lies in between **0 (0%, always off) – 255 (100%, always on).**
- **e.g. analogWrite (3, 127)** //generates pwm of 50% duty cycle

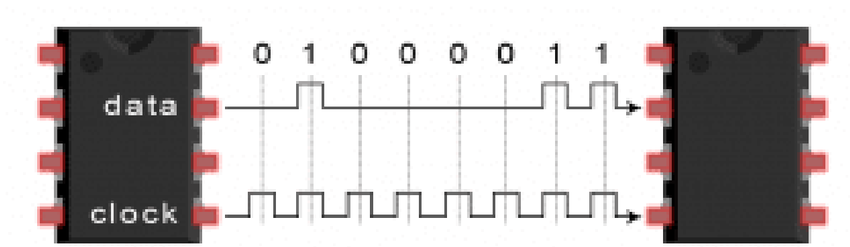
Parallel vs Serial Communication

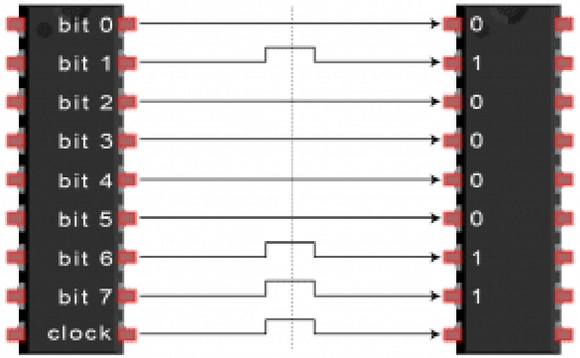
- In Communication the data is sent as bit.
- A bit is a binary representaion 1 or 0 . O is 0V or low and 1 is 5V or high.

- In **parallel communication**, the bits of data are sent all at the same time, each through a separate wire. Here for a single clock pulse multiple bits are transferred.

- In **serial communication**, the bits are sent one by one through a single wire. Here Each bit is sychronized with clock pulse. Here for a single clock pulse single bit is transferred.

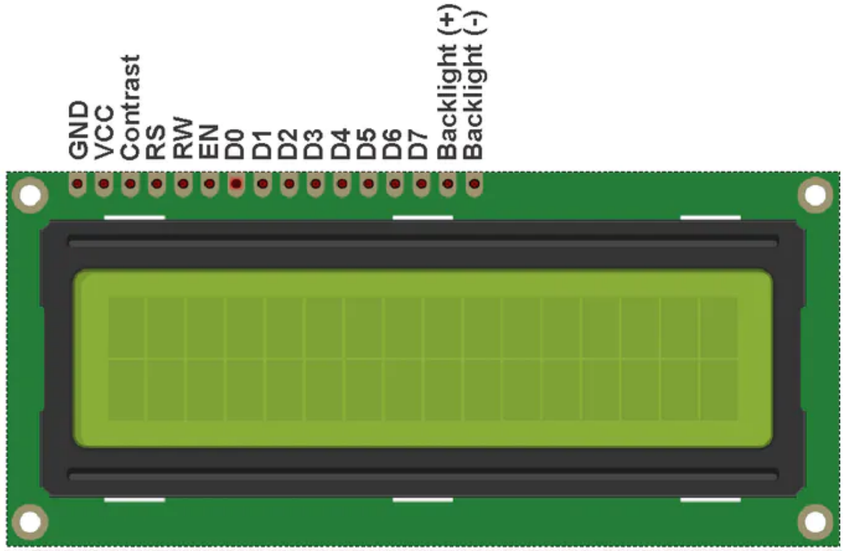
- The data transmission rate of parallel is greater as compared to serial. The speed of data depends upon frequency of clock pulse.





LCD interfacing

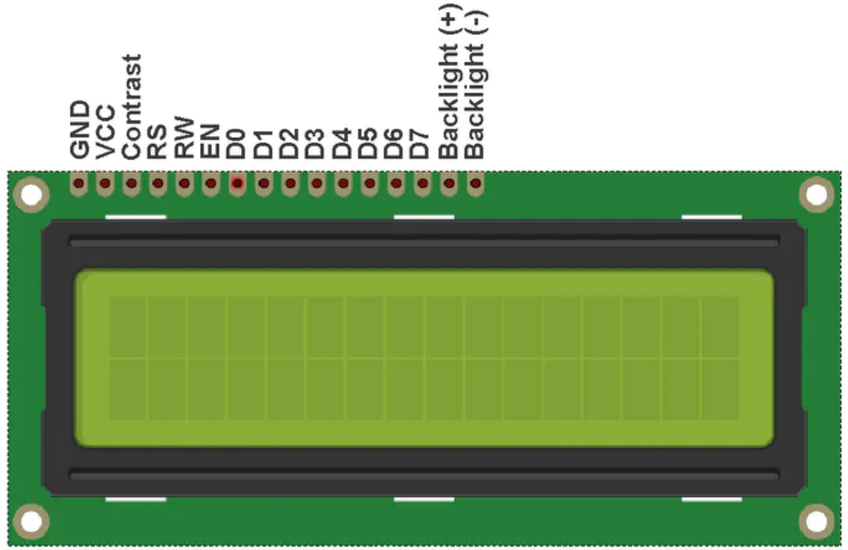
- LCD is a seven segment display that is used to display the output.
- It is available in dimensions **8×1, 8×2, 8×4, 16×1, 16×2 20×1, 20×2, 20×4, 24×1, 24×2, 24×4, 32×1, 32×2, 40×1, 40×2, 40×4 .**
- **Hitachi HD44780** 16X2 is the commonly used lcd for arduino.
- 16 represents **columns (0 - 15)** 2 represents **rows (0 - 1)**.



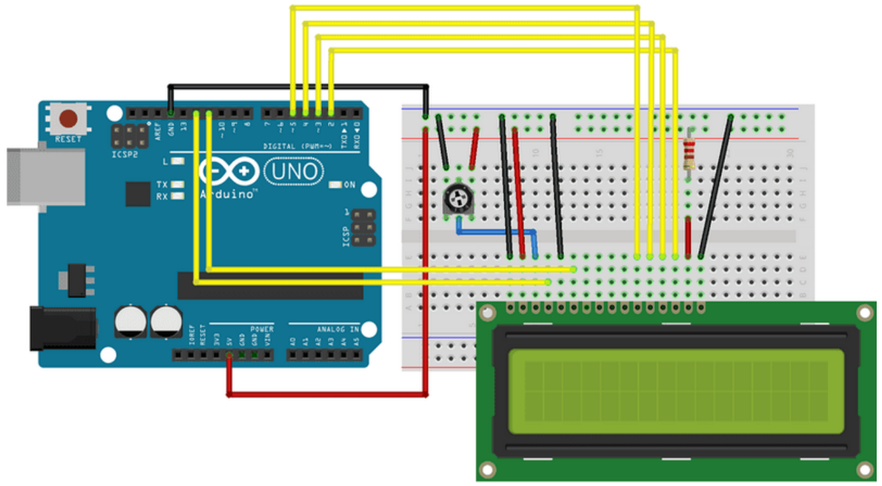
- pin 1 - **GND** - Ground
- pin 2 - **Vcc** - power supply (5v)
- pin 15 - Backlight (+) - 5V
- pin 16 - Backlight (-) - Gnd
- pin 3 - **Contrast** (adjustable with potentiometer)
- pin 4 - **Register Select**
- pin 5 - **Read or Write**
- pin 6 - **Enable**
- pin 7 to pin 14 (D0 - D7) - **Data pins**

LCD pins configuration

- **Contrast (pin 3)** - connected to potentiometer to adjust the text contrast . depending upon potentiometer output contrast is adjusted.
- **Register Select (pin 4)** - switch between two registers (data or instruction registers) . Data register holds the display data of the screen . Instruction register holds the next operation to execute.
- **Read / Write (pin 5)** - High - Reads from register . Low - write to register.
  It is permanently grounded to write.
- **Enable (pin 6)** - enables inputing data into the data pins.
- **Data pins (D0 - D7)** - It has two modes 4 bit or 8 bit mode.
  - In **4 bit mode** , last four significant bits (D4 - D7) are used.
  - In **8 bit mode** , all bits are used (D0 - D4).
  - 4 bit mode is commonly used.



Connection and coding

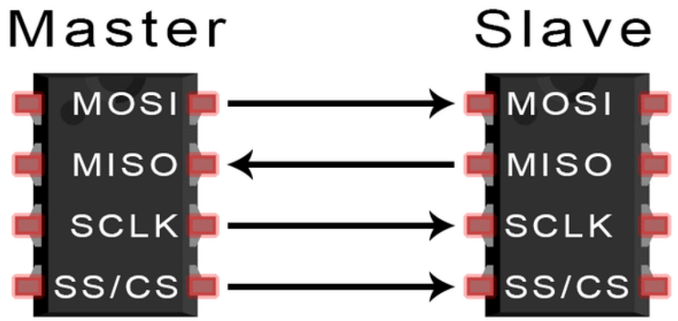


- pin 1 - **GND** - Ground
- pin 2 - **Vcc** - power supply (5v)
- pin 15 - Backlight (+) - 5V
- pin 16 - Backlight (-) - Gnd
- pin 3 - **Contrast** (10k potentiometer)
- pin 4 - **Register Select** - **D11**
- pin 5 - **Read or Write - Grounded**
- pin 6 - **Enable - D12**
- pin 11 to pin 14 (D4 - D7) - **D0 , D1 , D2 , D3**

```
// include the library
#include <LiquidCrystal.h>

// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
const int rs = 11, en = 12, d4 = 3, d5 = 2, d6 = 1, d7 = 0;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
 // set up the LCD's number of columns and rows and begin
 lcd.begin(16, 2);
 lcd.print("hello world!");  // prints hello world
 lcd.clear();  // clears the display
 lcd.write("hello");  //similiar to print
 lcd.setCursor(1,1);  //set cursor to 1st row , 1st column
 lcd.scrollLeft();
 lcd.scrollRight();  //scroll disply towards left or right
 lcd.display() ;  // turn display on
 lcd.noDisplay();  // turn display off
 lcd.blink();  // sets up blinking cursor
 lcd.cursor();  // sets up a underscore cursor at next writing position
 delay(1000);
}

void loop() {
}
```

SPI protocol

- SPI stands for Serial Peripheral Interface.
- SD card reader modules, RFID card reader modules all use SPI to communicate with microcontrollers.
- It is Synchronous protocol - Data is Synchronized with clock pulse.
- SPI follows a Master - Slave relationship.
- The master is the controlling device (usually a microcontroller), while the slave (usually a sensor, display, or memory chip) takes instruction from the master.
- The benefit of SPI is the fact that data can be transferred without interruption.



- **Here one master can control single or multiple slaves.**

MOSI (Master Output/Slave Input) – Line for the master to send data to the slave.

MISO (Master Input/Slave Output) – Line for the slave to send data to the master.

SCLK (Clock) – Line for the clock signal.

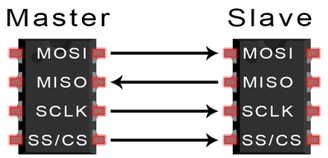
SS/CS (Slave Select/Chip Select) – Line for the master to select which slave to send data to.

SPI protocol

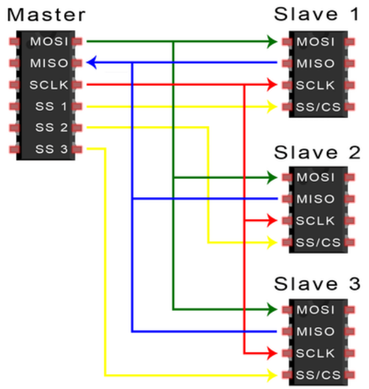
- **Clock pulse**
  - The Clock pulse is innitiated by the master - Slave receives the pulse through **Sclk pin**.
  - The Clock signal in SPI can be modified with two properites
    - => Clock polarity - It is set to allow the bits to be output and sampled on either the **rising or falling edge** of the clock cycle .
    - => Clock pulse - It is set for the output and sampling to occur on either the **first edge or second edge of the clock cycle** , irrespective of either rising or falling.

- **Slave select (CS/SS)**
  - It is active low pin
  - To select a slave it is set to low
  - Arduino has single SS pin

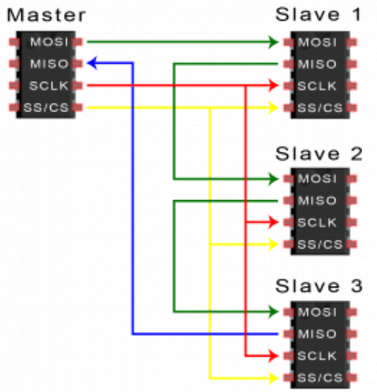
Daisy chaining method is adopted to connect multiple slaves to single slave select



single slave

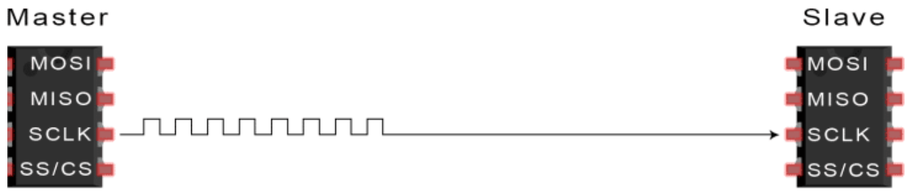


Master with mulltiple slave select

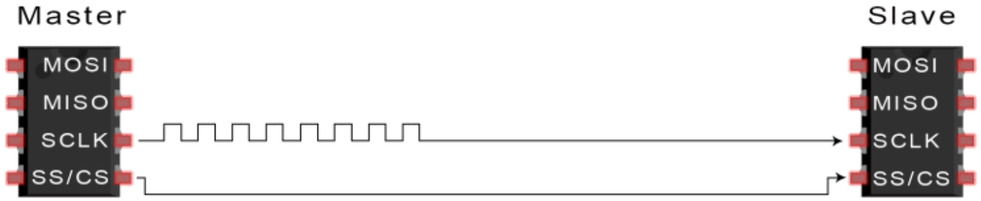


Master with single slave select

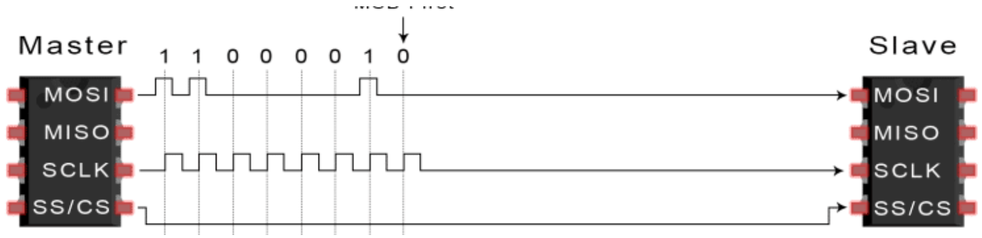
Steps in SPI protocol



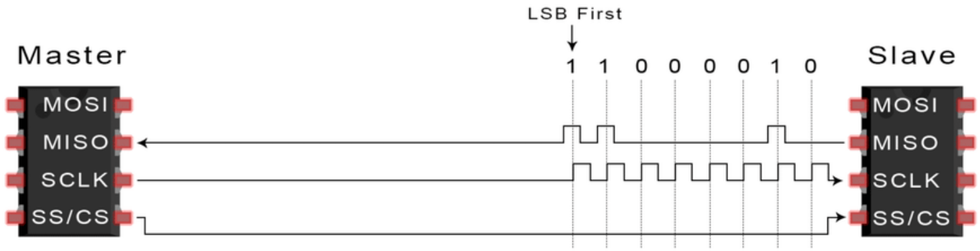
master initiates clock signal



Slave select pin shifts from high to low

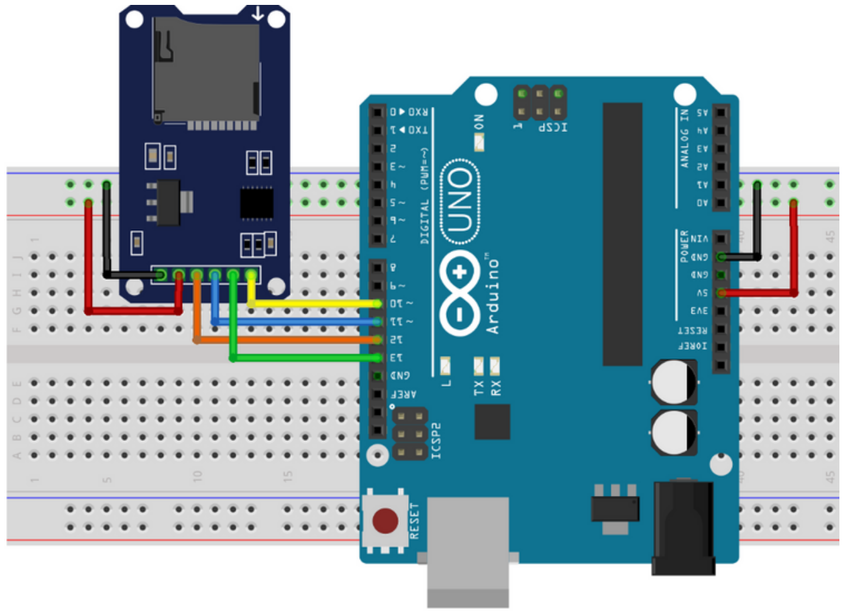


master sends data to slave through mosi pin with MSB first



slave sends data to master through miso pin with LSB first

UNO SPI interfacing with Sd card



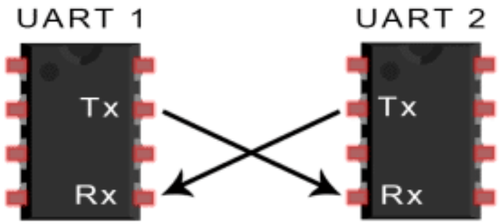
GND - GND
VCC - VCC
MISO - D12 (MISO pin of arduino)
MOSI - D11 (MOSI pin of arduino)
SCK - D13 (CLK pin of arduino)
CS - D10 (Slave select pin of arduino)

```
#include <SPI.h>
#include <SD.h>
File myFile;  // creating file object
int chipSelect = 4;
void setup() {
 Serial.begin(9600);
 while (!Serial) {
  ; // wait for serial port to connect. Needed for native USB
port only
 }
Serial.print("Initializing SD card...");
 if (!SD.begin()) {
  Serial.println("initialization failed!");
  while (1);
 }
 Serial.println("initialization done.");

 myFile = SD.open("myfile.txt", FILE_WRITE);
 if (myFile) {
  Serial.print("Writing to myfile.txt...");
  myFile.println("content");
  // close the file:
  myFile.close();
  Serial.println("done.");
 } else {
  Serial.println("error opening test.txt");
 }
```

```
 myFile = SD.open("myfile.txt");
  if (myFile) {
   Serial.println("myfile.txt:");

    while (myFile.available()) {
     Serial.write(myFile.read());
    }
    // close the file:
    myFile.close();
  } else {
    Serial.println("error opening myfile.txt");
 //error
  }
 }
 void loop() {
 }
```

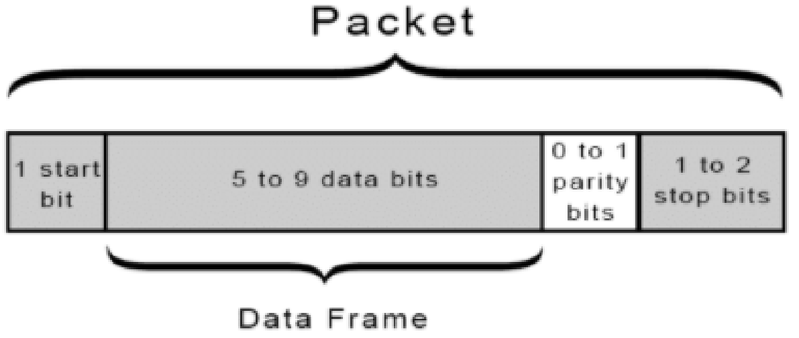
UART

- UART stands for **Universal Asynchronous Receiver/Transmitter.**
- It is used to connect **GPS modules, Bluetooth modules** etc.
- It's not a communication protocol like SPI and I2C, but a physical circuit in a microcontroller.
- UART's main purpose is to **transmit and receive** serial data.



- UART uses **two wires** for communication. **(Tx , Rx)**
- It is asynchronous , independent of clock pulse .
- Rather it uses a predefined **baud rate** for sampling.
- Baud rate is a measure of the speed of data transfer, expressed in **bits per second (bps).**
- **Both UARTs must operate at about the same baud rate.**

- In UART , **data packets** are used to send data to receiver.
- The data packet consists of
  - **Start bit (1bit)**
  - **Data bit (6 - 9 bit) - { Data Frame**
  - **Parity bit (1 or 0)**
  - **Stop bit (1 or 2 bits)**



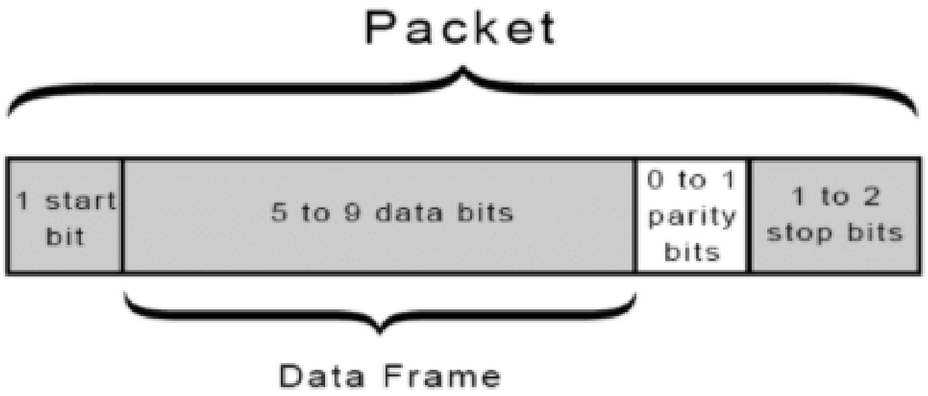
UART - Data packet

Start bit

- Normally held at a high voltage level when it’s not transmitting data.
- UART pulls the Tx from high to low for one clock cycle to start transfer.
- Rx detects the high to low voltage transition, it begins sampling data frame at baud rate frequency.

Data frame

- The data frame contains the actual data.
- It can be 5 bits up to 8 bits long.
- The data is sent with the least significant bit first.



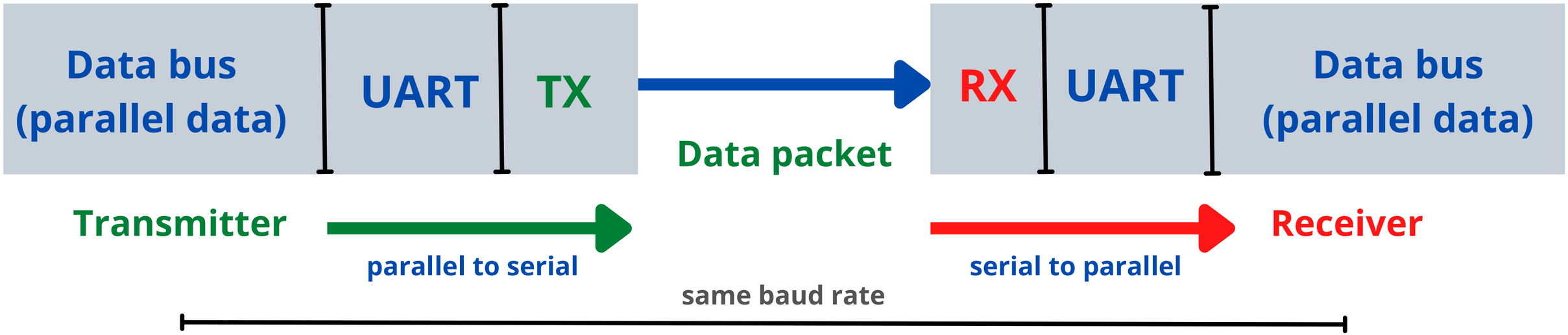
Parity bit

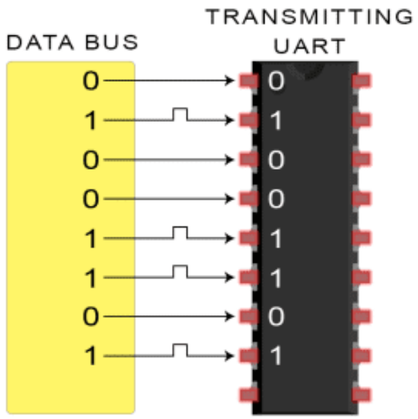
- 0 if sum of all 1's in data frame is even
- 1 if sum of all 1's in data frame is odd
- It detects change in data during transfer.

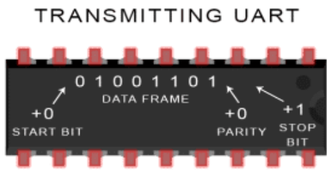
Stop bit

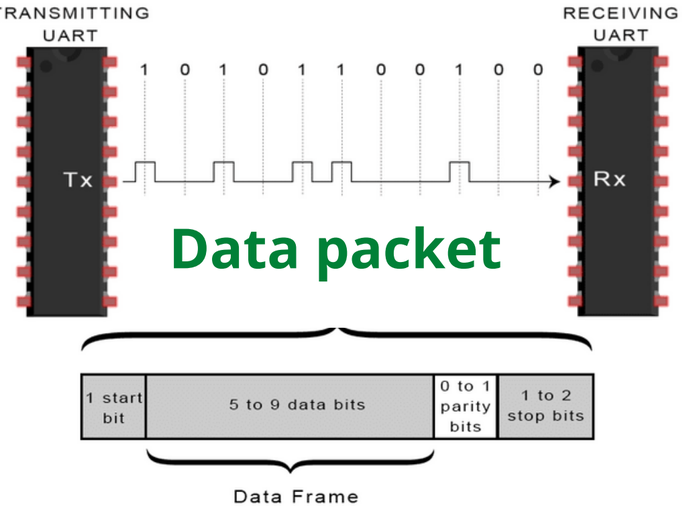
- To signal the end of the data packet.
- Tx pulls low to high pulse for 2 bit duration.
- After the stop signal , the sampled data is processed.

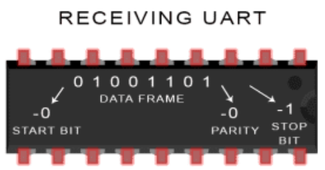
UART - Data transmission

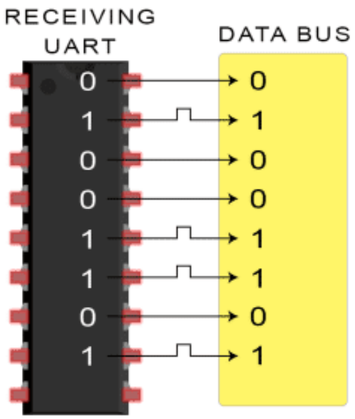




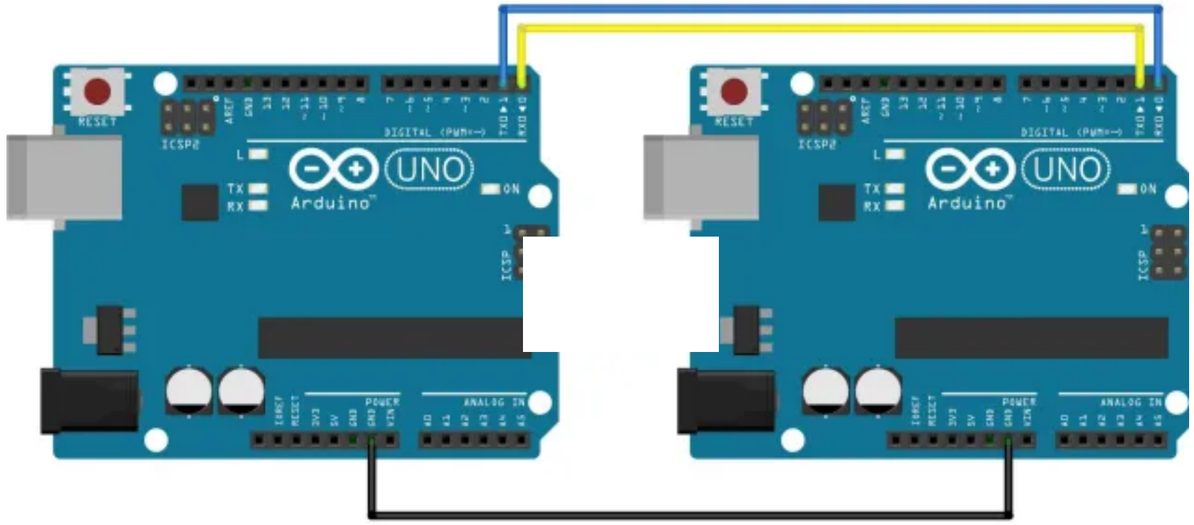








connecting two Arduino via UART



Sender Receiver

Arduino 1 (TX) - Arduino 2 (RX)

Arduino 2 (TX) - Arduino 1(RX)

Arduino 1 (GND) - Arduino 2 (GND)

Sender

```
char Mymessage[5] = "Hello"; //String data

 void setup() {
  // Begin the Serial at 9600 Baud
  Serial.begin(9600);
 }

 void loop() {
  Serial.write(Mymessage,5); //Write the serial data
  delay(1000);
 }
```

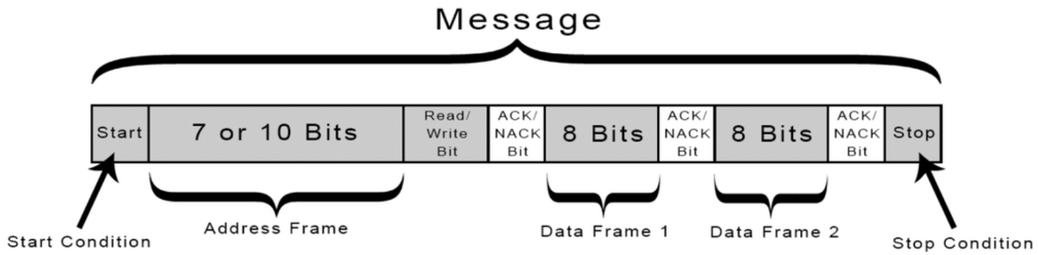
Receiver

```
char Mymessage[10]; //Initialized variable to store recieved
data

 void setup() {
  // Begin the Serial at 9600 Baud
  Serial.begin(9600);
 }

 void loop() {
  Serial.readBytes(Mymessage,5); //Read the serial data and
store in var
  Serial.println(Mymessage); //Print data on Serial Monitor
  delay(1000);
 }
```

I2C protocol

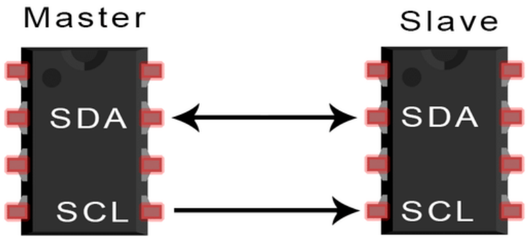
- I2C Stands for **Inter integrated Circuit**.
- It can connect multiple slaves to a single master and multiple masters controlling single, or multiple slaves.
- I2C uses two wires to transmit data **(SDA and SCL)** .
- I2C is a serial communication protocol , uses **single wire** for data.
- It is **Sychronous** , the output bit is synced with clock signal.

- In I2C the data is transmitted as **messages** , which is of several fragments
  - Start Condition
  - Address frame (7 or 10 bits)
  - Read or Write Bit
  - Dataframe (8 bits)
  - Acknowledgement or no-Acknowledgement bit
  - Stop signal



SDA (Serial Data) – The line for the master and slave to send and receive data.

SCL (Serial Clock) – The line that carries the clock signal.



I2C - Messages

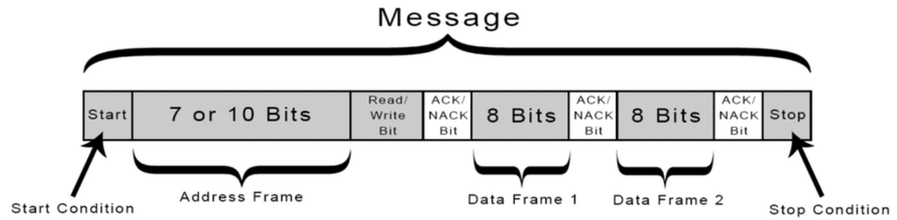
Start Signal The SDA line switches from **high to low** before SCL line switches from high to low.

Address Frame A **7 or 10 bit** sequence unique to each slave that identifies the slave.

Read or Write bit **LOW** = master sending data to slave **HIGH** - master requesting data from slave

Acknowledgement bit

- Each frame in a message is followed by an acknowledge/no-acknowledge bit.
- Receiver provides if address is matched or data is received.
- By pulling the **SDA line low for one bit** , ACK is sent.



Data Frame

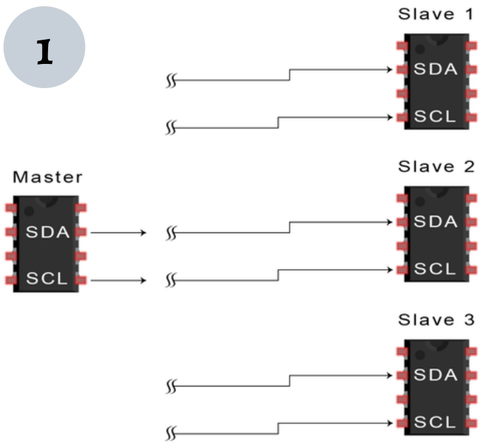
- The data frame is **8 bits** long, and sent with the **most significant bit first**.
- Each data frame is immediately followed by an acknowledgement bit.

Stop Signal

- The SDA line switches from **low to high** before SCL line switches from low to high.

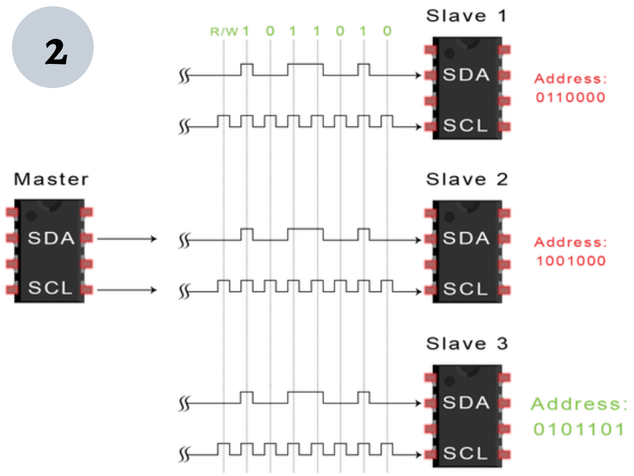
I2C protocol steps

1



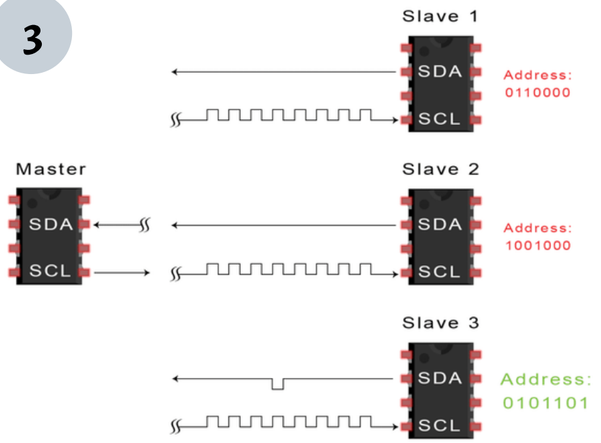
master initiates the start pulse (high to low)

2



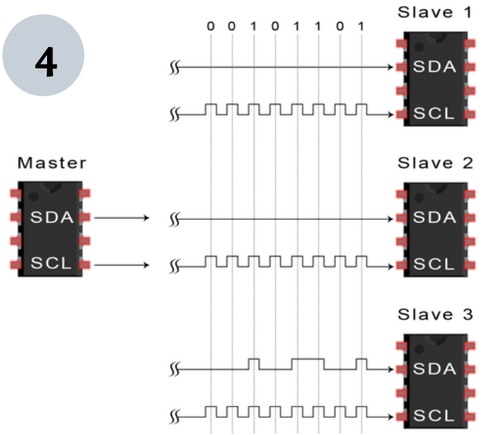
master sends 7 bit address along with R/W bit

3



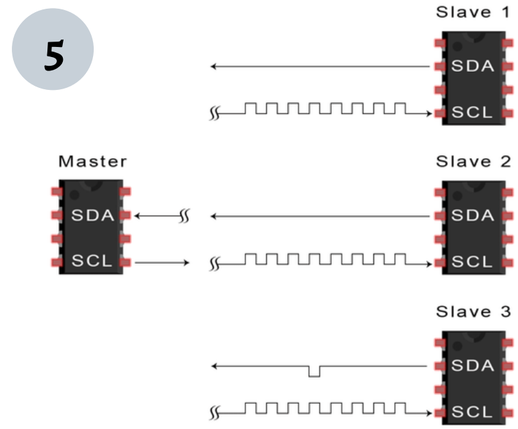
acknowledgement from receiver (low pulse)

4



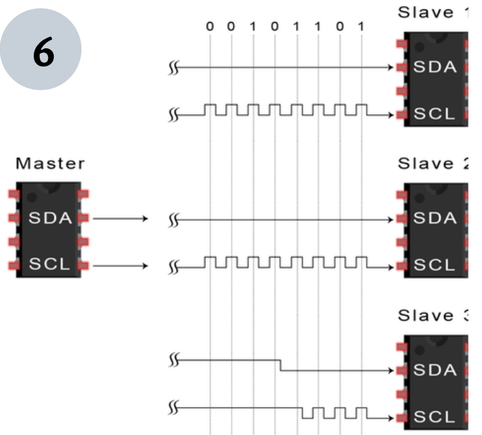
Master sends or receive data frame

5



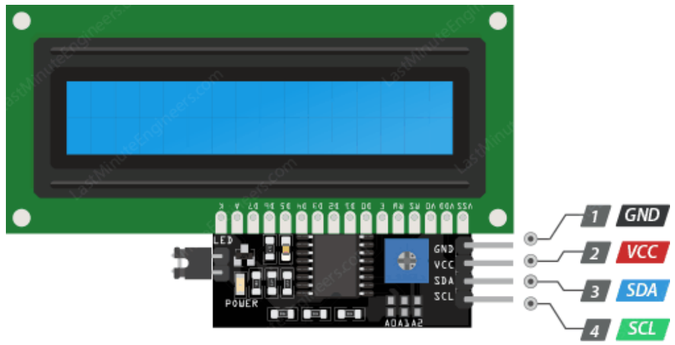
acknowledgement from receiver (low pulse)

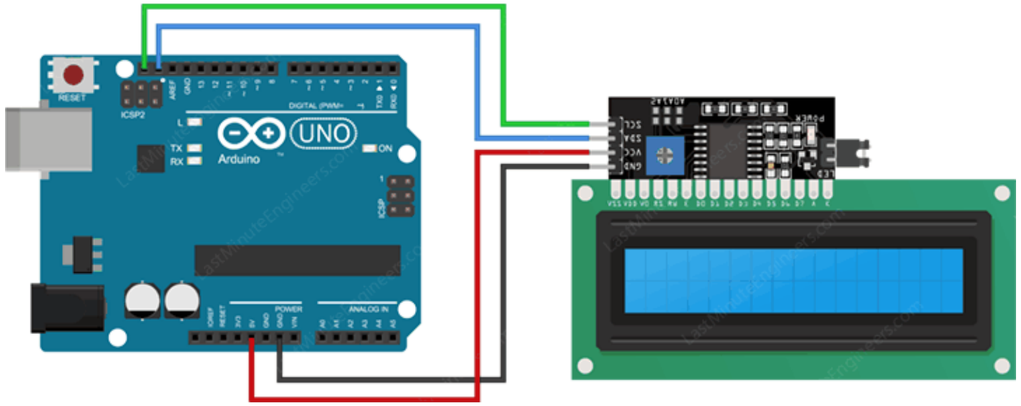
6



master initiates the stop pulse (low to high)

interfacing I2C lcd with Arduino






```
#include <LiquidCrystal_I2C.h> // include library for i2c lcd

LiquidCrystal_I2C lcd(0x3F,16,2);  // sets address and dimensions

void setup() {
  lcd.init();
  lcd.clear();
  lcd.backlight();
  lcd.setCursor(2,0);   //Set cursor to character 2 on line 0
  lcd.print("Hello world!");
  lcd.setCursor(2,1);   //Move cursor to character 2 on line 1
}

void loop() {
}
```

	SCL	SDA
Arduino Uno	A5	A4

The module is an 8-Bit I/O Expander chip – PCF8574. This chip converts the I2C data from an Arduino into the parallel data required by the LCD display.


Thank
you