SNS COLLEGE OF TECHNOLOGY

Coimbatore-35

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A++' Grade

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

19ECT213- IoT SYSTEM ARCHITECTURE

II ECE / IV SEMESTER

UNIT 2 – MICROCONTROLLER AND INTERFACING TECHNIQUES FOR IoT DEVICES

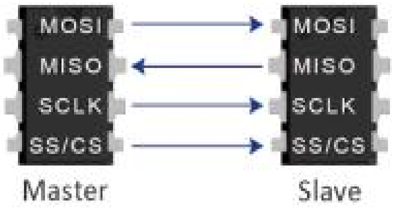
TOPIC 8 –-PWM Technique-Serial Communication

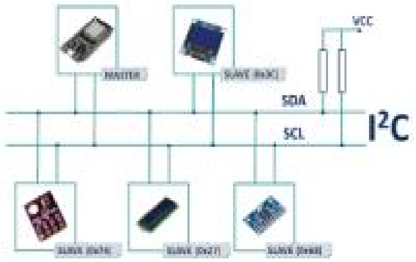
Interfacing

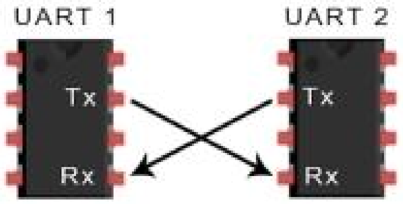
- The process of connecting external peripheral devices such as sensors , actuators and other external modules with different communication protocols with microcontroller is called interfacing.

- 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**.

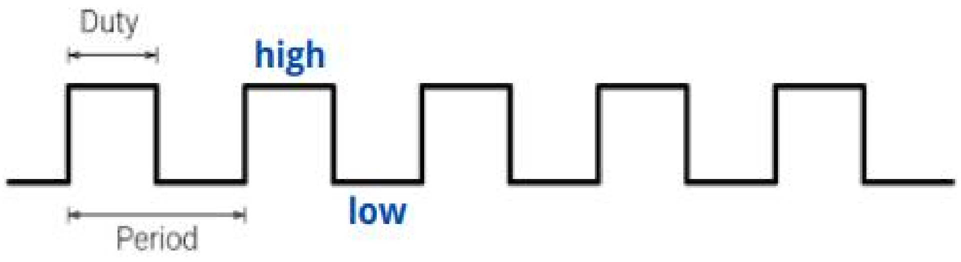






Pulse Width Modulation

- 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**".

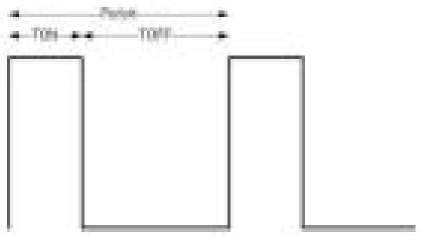
Duty cycle and frequency

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

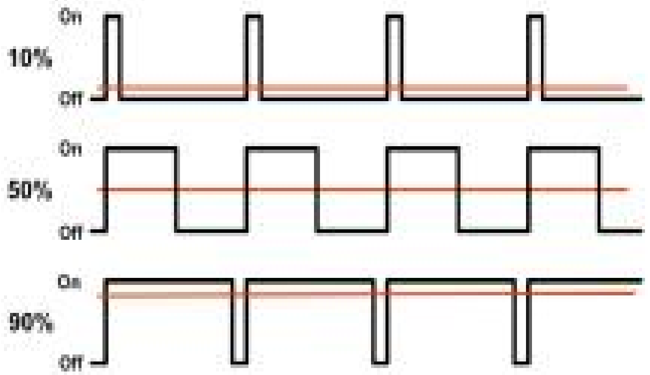
- The The frequency of a PWM signal determines how fast a PWM completes one period.
- **frequency = 1 / Period**



Period = TON + TOFF

Frequency = 1 / Period

Duty Cycle = $\frac{TON}{TON + TOFF} * 100$



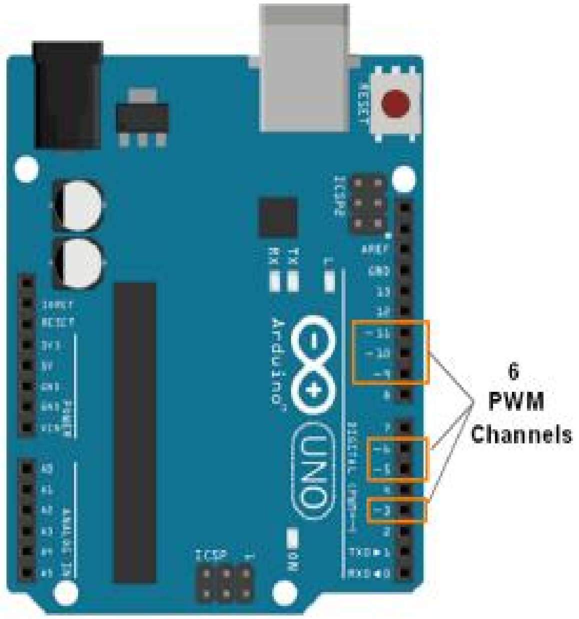
Different duty cycles

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



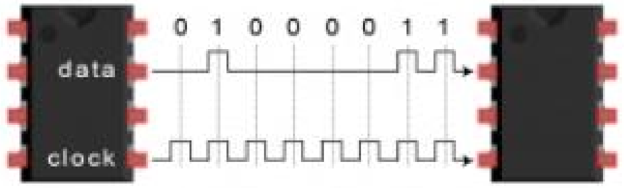
Parallel vs Serial Communication

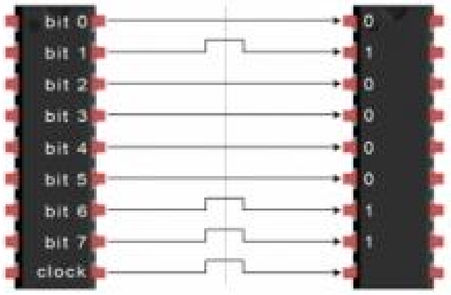
- In Communication the data is sent as bit.
- A bit is a binary representaion 1 or 0 . O is 0V or low and 1 is 5V or high.

- 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.

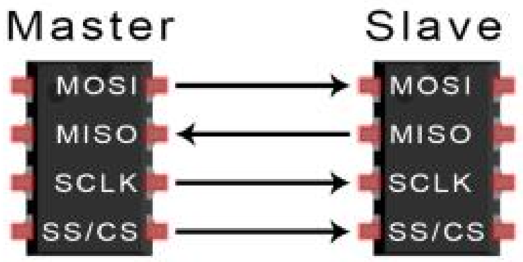




SPI protocol

- 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, displa 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.

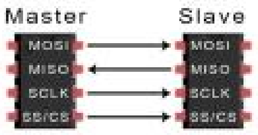
SS/CS (Slave Select/Chip Select) – Line for the master to select which slave to send data to.

SPI protocol

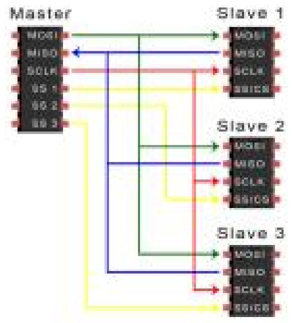
- **Clock pulse**
  - The Clock pulse is innitiated by the master - Slave receives the pulse through **Sclk pin**.
  - The Clock signal in SPI can be modified with two properites
    - => Clock polarity - It is set to allow the bits to be output and sampled on either the **rising or falling edge** of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

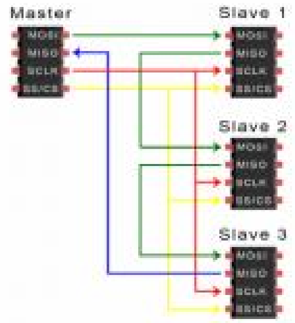
Daisy chaining method is adopted to connect multiple slaves to single slave select



single slave

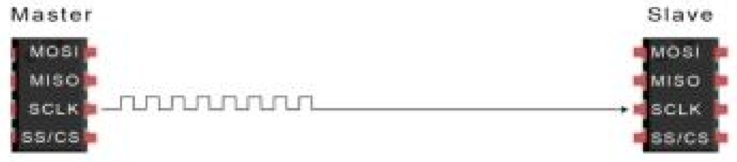


Master with mulltiple slave select

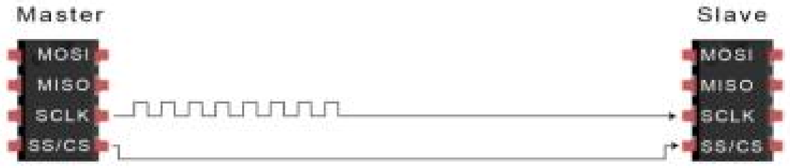


Master with single slave select

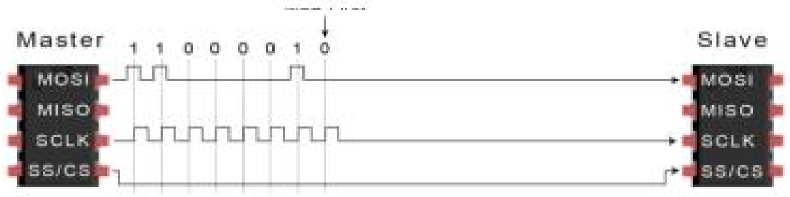
Steps in SPI protocol



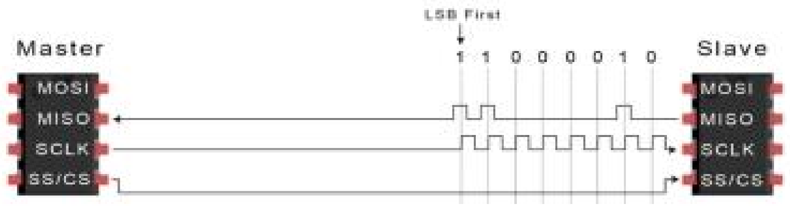
master initiates clock signal



Slave select pin shifts from high to low



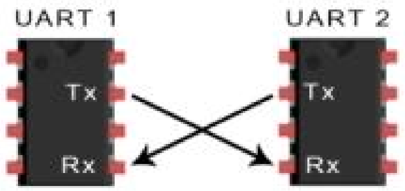
master sends data to slave through mosi pin with MSB first



slave sends data to master through miso pin with LSB first

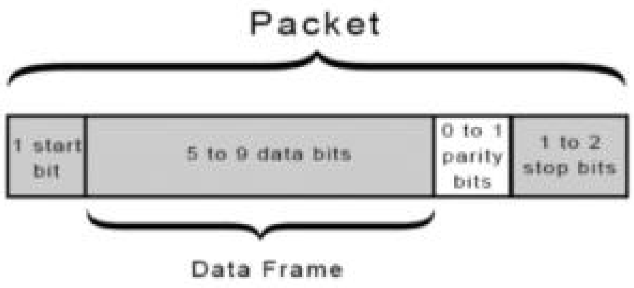
UART

- 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)**



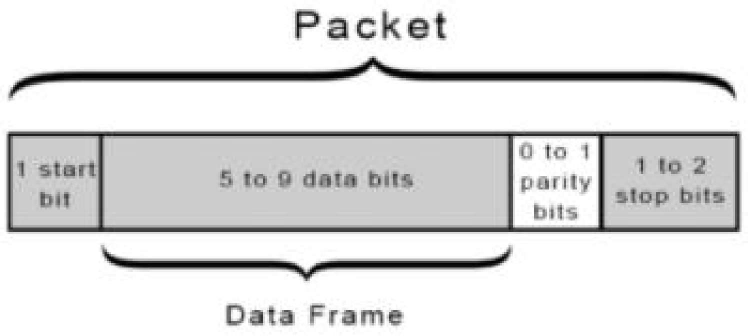
UART - Data packet

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.



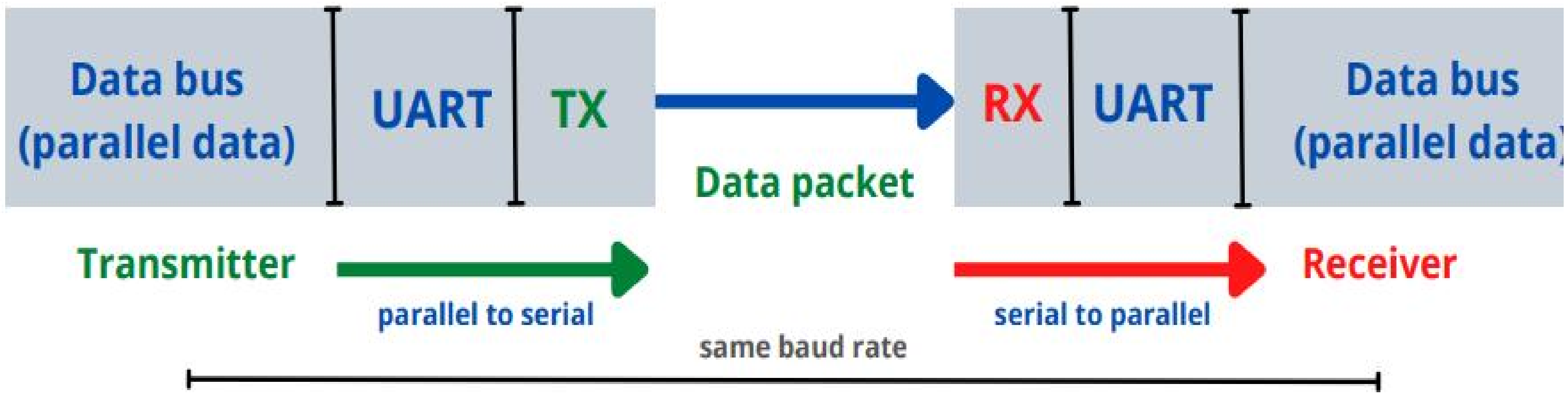
Parity bit

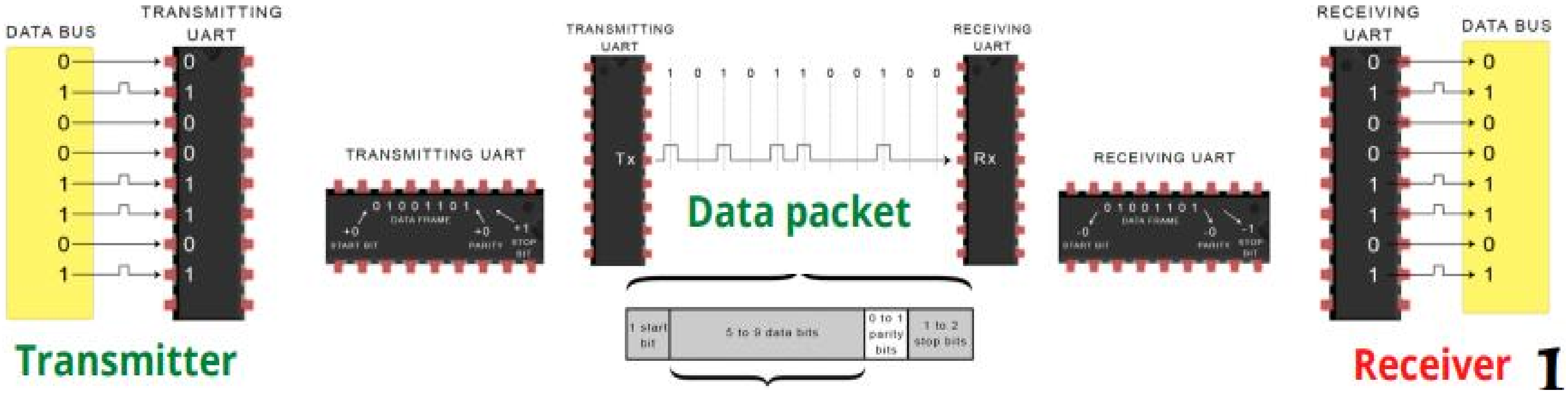
- 0 if sum of all 1's in data frame is even
- 1 if sum of all 1's in data frame is odd
- It detects change in data during transfer.

Stop bit

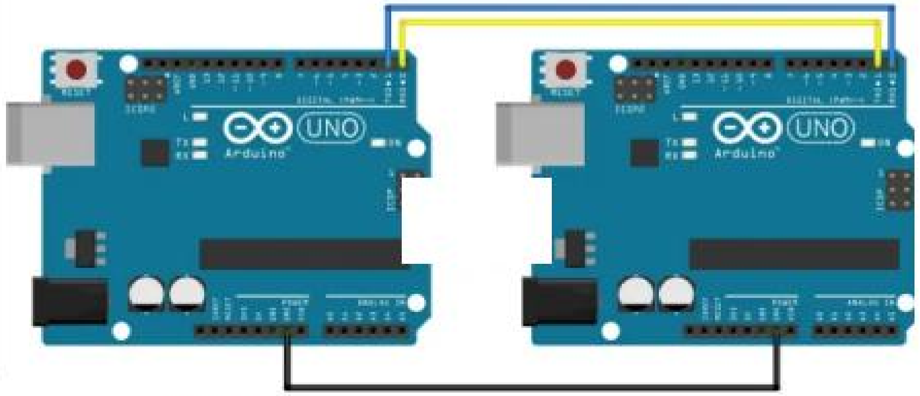
- To signal the end of the data packet.
- Tx pulls low to high pulse for 2 bit duration.
- After the stop signal , the sampled data is processed.

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

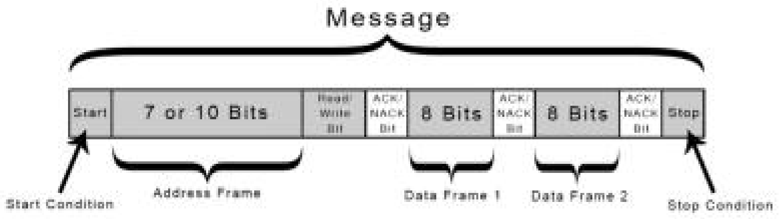
```
char Mymessage[10]; //Initialized variable to store recieved
data

 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.

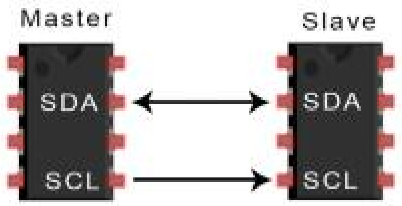
- In I2C the data is transmitted as **messages** , which is of several fragments

  - 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

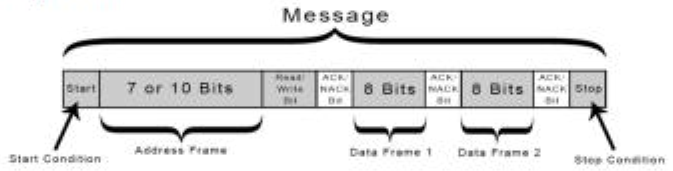
Start Signal The SDA line switches from **high to low** before SCL line switches from high to low.

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.



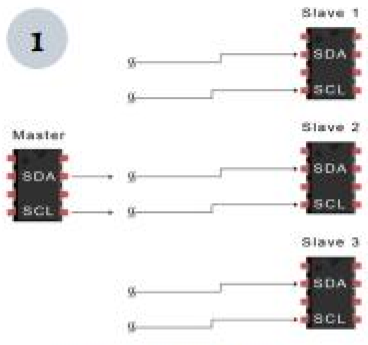
Data Frame

- 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.

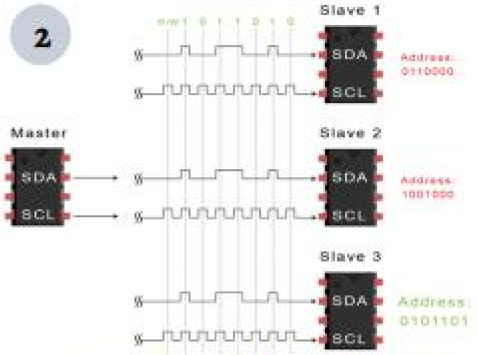
Stop Signal

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

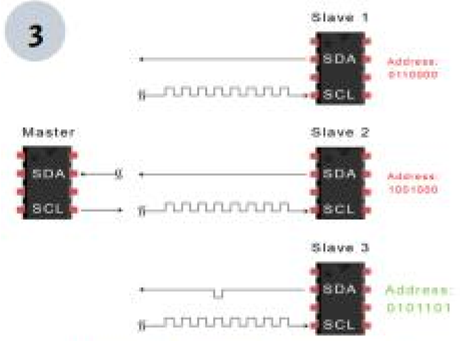
I2C protocol steps



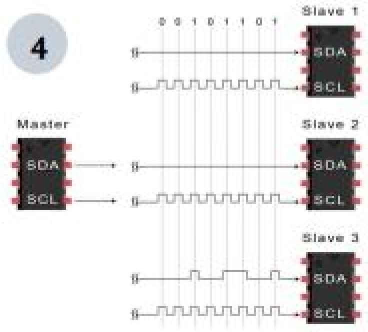
master initiates the start pulse (high to low)



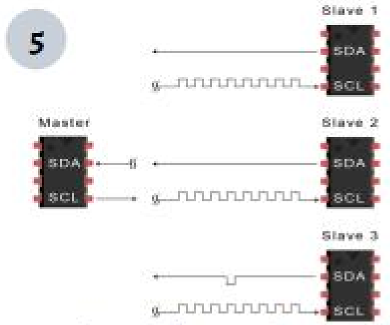
master sends 7 bit address along with R/W bit



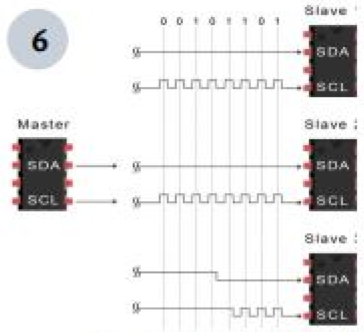
acknowledgement from receiver (low pulse)



Master sends or receive data frame



acknowledgement from receiver (low pulse)



master initiates the stop pulse (low to high)

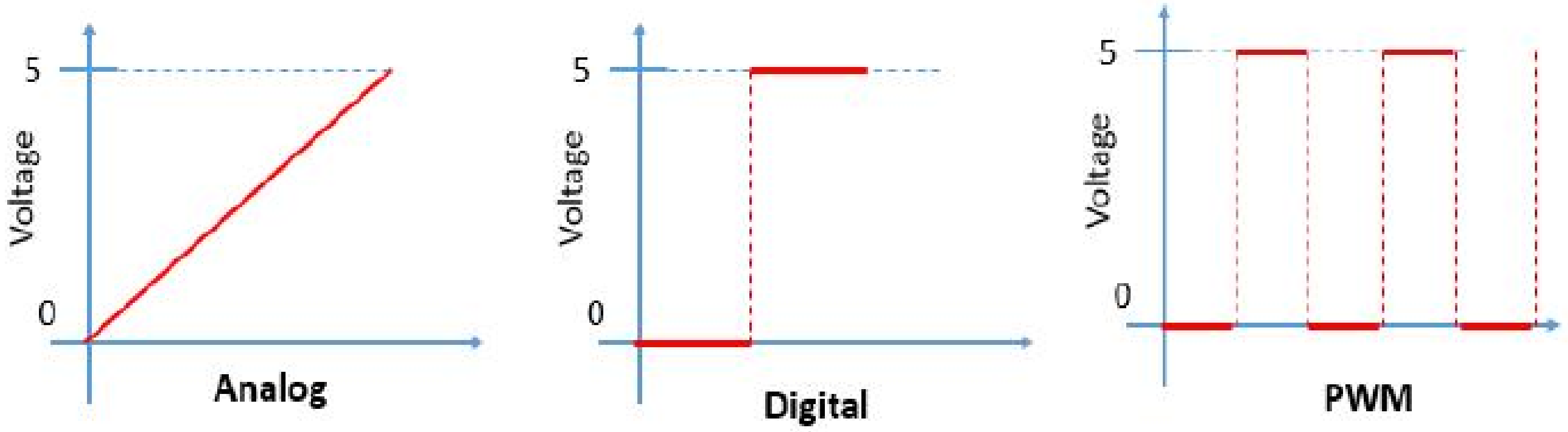
Difference between Analog, Digital and PWM Pins

In **analog pins,** you have unlimited possible states between 0 and 1023. This allows you to read sensor values. For example, with a light sensor, if it is very dark, you'll read 1023, if it is very bright you'll read 0 If there is a brightness between dark and very bright you'll read a value between 0 and 1023.

In **digital pins**, you have just two possible states, which are on or off. These can also be referred as High or Low, 1 or 0 and 5V or 0V. For example, if an LED is on, then, its state is High or 1 or 5V. If it is off, you'll have Low, or 0 or 0V.

PWM pins are digital pins, so they output either 0 or 5V. However these pins can output "fake" intermediate voltage values between 0 and 5V, because they can perform "Pulse Width Modulation" (PWM). PWM allows to "simulate" varying levels of power by oscillating the output voltage of the Arduino.

Fundamentals of Arduino Programming

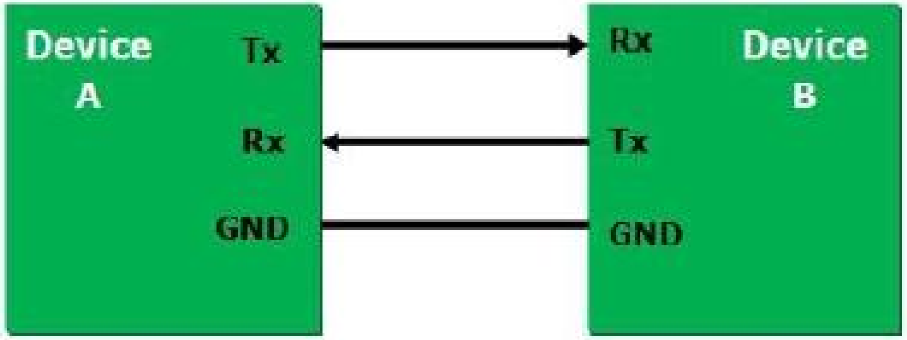


Difference between Analog, Digital and PWM Pins

Introduction to Communications

Serial (UART) communications:

- Serial communication on Arduino pins Tx/Rx uses TTL logic levels which operates at either 5V/3.3V depending the type of the board used.
- Tx/Rx pins should not be connected to any source which operates more than 5V which can damage the Arduino board.
- Serial communication is basically used for communication between Arduino board and a computer or some other compatible devices.



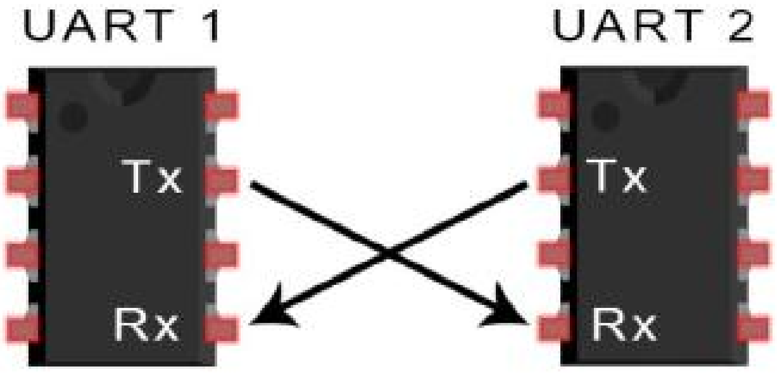
Introduction to Communications

Serial (UART) communications:

▪Every Arduino board will have at least one serial port known as UART.

▪Serial communicates on digital pins Rx(pin 0) and Tx(pin 1) with the computer via USB, pin 0 and pin 1 cannot be used for digital input or output.

▪The built in serial monitor can be used to communicate with an Arduino board by selecting same baud rate that is used in the call to begin () which will come across in the later part of the chapter



Introduction to Communications

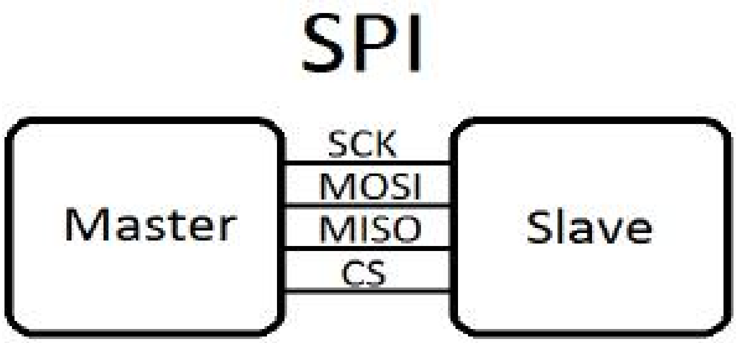
SPI communications

- Serial communication Interface (SPI) is a synchronous data protocol used by large microcontrollers for communicating with one or more peripheral devices for a shorter distance and also used for communication between two devices.
- With SPI there will be always one master device which is a microcontroller like Arduino which controls the functionalities of other peripheral devices.
- Devices have three lines in common which are as follows
  - MISO (Master in Slave Out)- Slave line for sending data to the master.
  - MOSI (Master Out Slave In)- Master sending data to peripherals
  - SCK Serial clock) - clock pulses which synchronize data transmission generated by the master And one of the specific line for every device is
  - SS (slave select) - pin on each device that the master can use to enable and disable specific devices.

Introduction to Communications

SPI communications

- When device SS pin is low, communication happens with the master, if SS pin is high device ignores the maser. This allows multiple SPI devices sharing the the same MISO, MOSI and CLK lines.
- To program a new SPI device some key points to be noted which are
  - Maximum SPI speed of the device used?
  - How data is shifted like MSB/LSB?
  - Data clock is idle when high/low.



Introduction to Communications

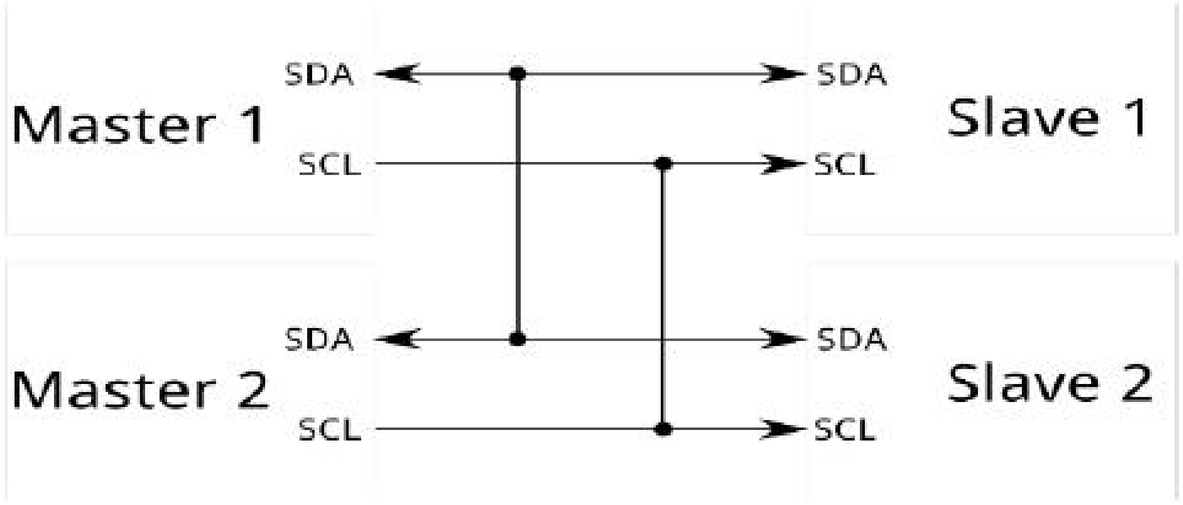
I^2C communications

- Inter-Integrated circuit or I^2C (I squared C) is one of the best protocol used when a workload of one Arduino (Master Writer) is shared with another Arduino (Slave receiver).
- The I^2C protocol uses two lines to send and receive data which are a serial clock pin (SCL) which writes data at regular intervals and a serial data pin (SDA) over which data sent between devices.
- When the clock signal changes from LOW to HIGH the information, the address corresponds to a specific device and a command is transferred from board to the I^2C device over the SDA line.
- This information is sent bit by bit which is executed by the called device, executes and transmits the data back.

Introduction to Communications

I^{2}C communications

- If the device require execution from another a slave device, the data is transferred to the board on the same line using pulse generated from Mater on SCL as timing.
- Each slave should have unique identity and both Master and slave turns out communicating on the same data line. In this way many of the Arduino boards are communicated using just two pins of microcontroller with each unique address of a device.


Master 1
SDA
SCL
Master 2
SDA
SCL
SDA
SCL
Slave 1
SDA
SCL
Slave 2