### **What is Serial Communication?**

**Serial communication** is the process of transmitting data one bit at a time. In contrast, parallel communication is where data bits are sent as a whole. Parallel data transmission is faster than serial transmission but with a number of disadvantages:

* It needs more wires and therefore can be more expensive to implement
* The greater number of wires limit it to shorter transmission distances
* It is susceptible to clock skew, which limits the speed of transmission to the slowest of the links
* Crosstalk is also an issue due to the proximity of the wires

Serial data transmission answers all the above problems, most especially the first one, as cost and limited pins are common issues in microcontroller system design.

There are generally two types of serial transmission: **asynchronous** and **synchronous** which literally means "not synced" and "synced" respectively. In asynchronous data transfer, there is no clock signal transmitted and received. The common UART (also known as RS-232 or SCI) device found in microcontrollers is asynchronous. When the communication is synchronous, there is an extra line for the clock signal. I2C and SPI are both considered synchronous.

The clock signal is used to ensure that the correct data bits are received. In SCI, the effect of the absence of a clock signal is mitigated by adding overhead. The added overhead affects throughput (the rate of production) since they don't carry useful information. However, the absence of the clock signal also means that SCI is simpler to setup and implement as compared  to the complex hardware needed by synchronous protocols.

For the rest of this post, serial will now refer to asynchronous data transmission used by UART devices found in Arduino and PIC microcontrollers. Synchronous data transmission protocols are discussed in their respective articles: [SPI](https://www.teachmemicro.com/spi-primer/), [I2C](https://www.teachmemicro.com/i2c-primer/).

### Using UART

#### Hardware Serial

Arduino boards communicate with the PC via USB cable or to other serial devices using the serial pins. Each arduino board has at least one serial port and is commonly assigned to digital pins 0 and 1 (Tx and Rx respectively). Using these pins while loading a program to the board causes an error because it's the same pins that are used for USB connection.

The arduino serial port uses TTL signal levels, which means the logic 1 is +5V and logic 0 is 0 V. A computer's serial port uses different voltage levels: the logic 1 is -3 to -25 V while the logic 0 is +3 to +25 V. This is why it's a bad idea to connect the arduino's serial pins directly to a PC's serial port.

The USB port (which is also serial) also uses different voltage levels and protocol. Arduino boards have USB to TTL chips included to interface the board to the computer via USB. Some ATMega micros have bootloaders that doesn't require a USB to TTL chip.

CODE

void setup() {

  // initialize serial communication at 9600 bits per second:

  Serial.begin(9600);

}

// the loop routine runs over and over again forever:

void loop() {

  // read the input on analog pin 0:

  int sensorValue = analogRead(A0);

  // print out the value you read:

  Serial.println(sensorValue);

  delay(1);        // delay in between reads for stability}

**I2C Protocol**

The I2C protocol allows embedded engineers to connect multiple slave devices with one or more master. Similar to the SPI protocol, it is intended primarily for short-distance communication within two ICs (Integrated Circuits) on the same printer circuit board (PCB).

It requires only two bi-directional wires for transmitting and receiving data bits. In terms of data rates, the I2C protocol supports speed up to 3.4 Mbps - which is pretty fast.

Here are some of the important advantages of using the I2C protocol:

* Flexibility of using multiple master devices to speed up communication and improve the design functionality
* Chip addressing eliminates the need of CS (Chip Select) Lines
* Supports a robust error handling mechanism with the ACK/NACK feature.

However, there are some limitations. For example, it takes up much more real estate on a PCB as it uses pull-up resistors.

*Uses of the I2C Protocol:*

* Changes the various color setting, such as hue, on a monitor.
* Controls the LED/LCD displays in cell phones
* Helps switching on /off the power supply of internal components

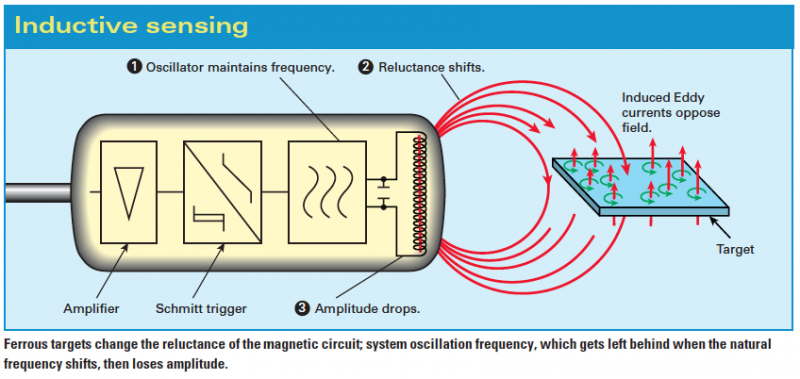
INDUCTIVE AND CAPACITIVE SENSORS

### **Inductive sensors**

These non-contact proximity sensors detect ferrous targets, ideally mild steel thicker than one millimeter. They consist of four major components: a ferrite core with coils, an oscillator, a Schmitt trigger, and an output amplifier. The oscillator creates a symmetrical, oscillating magnetic field that radiates from the ferrite core and coil array at the sensing face. When a ferrous target enters this magnetic field, small independent electrical currents called eddy currents are **induced** on the metal’s surface. This changes the reluctance (natural frequency) of the magnetic circuit, which in turn reduces the oscillation amplitude. As more metal enters the sensing field the oscillation amplitude shrinks, and eventually collapses. (This is the “Eddy Current Killed Oscillator” or ECKO principle.) The Schmitt trigger responds to these amplitude changes, and adjusts sensor output. When the target finally moves from the sensor’s range, the circuit begins to oscillate again, and the Schmitt trigger returns the sensor to its previous output.

If the sensor has a **normally open** configuration, its output is an **on** signal when the target enters the sensing zone. With normally closed, its output is an off signal with the target present. Output is then read by an external control unit (e.g. PLC, motion controller, smart drive) that converts the sensor on and off states into useable information. Inductive sensors are typically rated by frequency, or on/off cycles per second. Their speeds range from 10 to 20 Hz in ac, or 500 Hz to 5 kHz in dc. Because of magnetic field limitations, inductive sensors have a relatively narrow sensing range — from fractions of millimeters to 60 mm on average — though longer-range specialty products are available.

To accommodate close ranges in the tight confines of industrial machinery, geometric and mounting styles available include shielded (flush), unshielded (non-flush), tubular, and rectangular “flat-pack”. Tubular sensors, by far the most popular, are available with diameters from 3 to 40 mm.

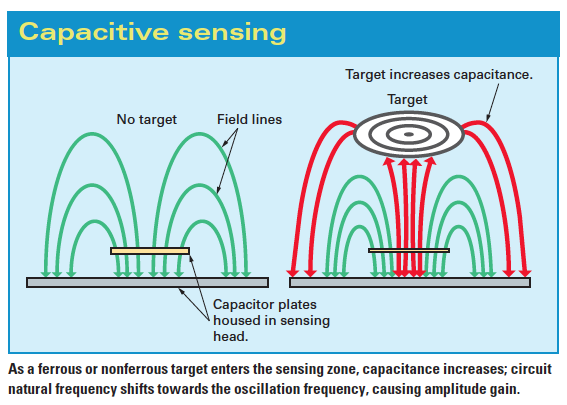


But what inductive sensors lack in range, they make up in environment adaptability and metal-sensing versatility. With no moving parts to wear, proper setup guarantees long life. Special designs with IP ratings of 67 and higher are capable of withstanding the buildup of contaminants such as cutting fluids, grease, and non-metallic dust, both in the air and on the sensor itself. It should be noted that metallic contaminants (e.g. filings from cutting applications) sometimes affect the sensor’s performance. Inductive sensor housing is typically nickel-plated brass, stainless steel, or PBT plastic.

### Capacitive sensors

Capacitive proximity sensors can detect both metallic and non-metallic targets in powder, granulate, liquid, and solid form. This, along with their ability to sense through nonferrous materials, makes them ideal for sight glass monitoring, tank liquid level detection, and hopper powder level recognition.

In capacitive sensors, the two conduction plates (at different potentials) are housed in the sensing head and positioned to operate like an open capacitor. Air acts as an insulator; at rest there is little capacitance between the two plates. Like inductive sensors, these plates are linked to an oscillator, a Schmitt trigger, and an output amplifier. As a target enters the sensing zone the capacitance of the two plates increases, causing oscillator amplitude change, in turn changing the Schmitt trigger state, and creating an output signal. Note the difference between the inductive and capacitive sensors: inductive sensors oscillate until the target is present and capacitive sensors oscillate when the target is present.



Because capacitive sensing involves charging plates, it is somewhat slower than inductive sensing ... ranging from 10 to 50 Hz, with a sensing scope from 3 to 60 mm. Many housing styles are available; common diameters range from 12 to 60 mm in shielded and unshielded mounting versions. Housing (usually metal or PBT plastic) is rugged to allow mounting very close to the monitored process. If the sensor has normally-open and normally-closed options, it is said to have a complimentary output. Due to their ability to detect most types of materials, capacitive sensors must be kept away from non-target materials to avoid false triggering. For this reason, if the intended target contains a ferrous material, an inductive sensor is a more reliable option.

### **How to use the HC-SR04 Ultrasonic Sensor**

**HC-SR04 distance sensor** is commonly used with both microcontroller and microprocessor platforms like Arduino, ARM, PIC, Raspberry Pie etc. The following guide is universally since it has to be followed irrespective of the type of computational device used.

  Power the Sensor using a regulated +5V through the Vcc ad Ground pins of the sensor. The current consumed by the sensor is less than 15mA and hence can be directly powered by the on board 5V pins (If available). The Trigger and the Echo pins are both I/O pins and hence they can be connected to I/O pins of the microcontroller. To start the measurement, the trigger pin has to be made high for 10uS and then turned off. This action will trigger an ultrasonic wave at frequency of 40Hz from the transmitter and the receiver will wait for the wave to return. Once the wave is returned after it getting reflected by any object the Echo pin goes high for a particular amount of time which will be equal to the time taken for the wave to return back to the sensor.

The amount of time during which the Echo pin stays high is measured by the MCU/MPU as it gives the information about the time taken for the wave to return back to the Sensor. Using this information the distance is measured as explained in the above heading.

### Applications

* Used to avoid and detect obstacles with robots like biped robot, obstacle avoider robot, path finding robot etc.
* Used to measure the distance within a wide range of 2cm to 400cm
* Can be used to map the objects surrounding the sensor by rotating it
* Depth of certain places like wells, pits etc can be measured since the waves can penetrate through water

### Ultrasonic Sensor Pin **Configuration**

|  |  |  |
| --- | --- | --- |
| **Pin Number** | **Pin Name** | **Description** |
| 1 | Vcc | The Vcc pin powers the sensor, typically with +5V |
| 2 | Trigger | Trigger pin is an Input pin. This pin has to be kept high for 10us to initialize measurement by sending US wave. |
| 3 | Echo | Echo pin is an Output pin. This pin goes high for a period of time which will be equal to the time taken for the US wave to return back to the sensor. |
| 4 | Ground | This pin is connected to the Ground of the system. |
|  |  |  |

**CODE**

const int pingPin = 7; // Trigger Pin of Ultrasonic Sensor

const int echoPin = 6; // Echo Pin of Ultrasonic Sensor

void setup() {

Serial.begin(9600); // Starting Serial Terminal

}

void loop() {

long duration, inches, cm;

pinMode(pingPin, OUTPUT);

digitalWrite(pingPin, LOW);

delayMicroseconds(2);

digitalWrite(pingPin, HIGH);

delayMicroseconds(10);

digitalWrite(pingPin, LOW);

pinMode(echoPin, INPUT);

duration = pulseIn(echoPin, HIGH);

inches = microsecondsToInches(duration);

cm = microsecondsToCentimeters(duration);

Serial.print(inches);

Serial.print("in, ");

Serial.print(cm);

Serial.print("cm");

Serial.println();

delay(100);

}

long microsecondsToInches(long microseconds) {

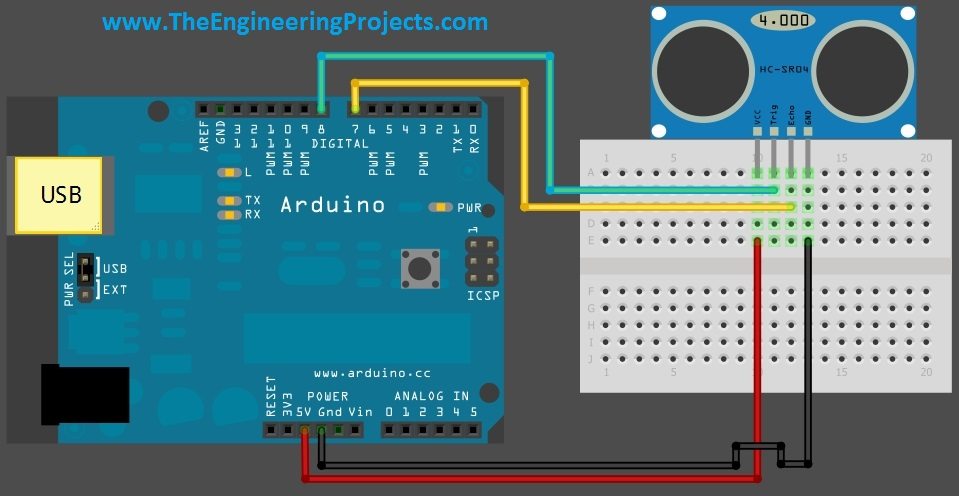
return microseconds / 74 / 2;

}

long microsecondsToCentimeters(long microseconds) {

return microseconds / 29 / 2;

}



[**What is the difference between baud rate and bit rate?**](https://stackoverflow.com/questions/20534417/what-is-the-difference-between-baud-rate-and-bit-rate)

Bits per second is straightforward. It is exactly what it sounds like. If I have 1000 bits and am sending them at 1000 bps, it will take exactly one second to transmit them.

Baud is symbols per second. If these symbols — the indivisible elements of your data encoding — are not bits, the baud rate will be lower than the bit rate by the factor of bits per symbol. That is, if there are 4 bits per symbol, the baud rate will be ¼ that of the bit rate.

This confusion arose because the early analog telephone modems weren't very complicated, so bps was equal to baud. That is, each symbol encoded one bit. Later, to make modems faster, communications engineers invented increasingly clever ways to send more bits per symbol.¹

## Analogy

**System 1, bits:** Imagine a communication system with a telescope on the near side of a valley and a guy on the far side holding up one hand or the other. Call his left hand "0" and his right hand "1," and you have a system for communicating one binary digit — one bit — at a time.

**System 2, baud:** Now imagine that the guy on the far side of the valley is holding up playing cards instead of his bare hands. He is using a subset of the cards, ace through 8 in each suit, for a total of 32 cards. Each card — each symbol — encodes 5 bits: 00000 through 11111 in binary.²

## Analysis

The System 2 guy can convey 5 bits of information per card in the same time it takes the System 1 guy to convey one bit by revealing one of his bare hands.

You see how the analogy seems to break down: finding a particular card in a deck and showing it takes longer than simply deciding to show your left or right hand. But, that just provides an opportunity to extend the analogy profitably.

A communications system with many bits per symbol faces a similar difficulty, because the encoding schemes required to send multiple bits per symbol are much more complicated than those that send only one bit at a time. To extend the analogy, then, the guy showing playing cards could have several people behind him sharing the work of finding the next card in the deck, handing him cards as fast as he can show them. The helpers are analogous to the more powerful processors required to produce the many-bits-per-baud encoding schemes.

That is to say, by using more processing power, System 2 can send data 5 times faster than the more primitive System 1.