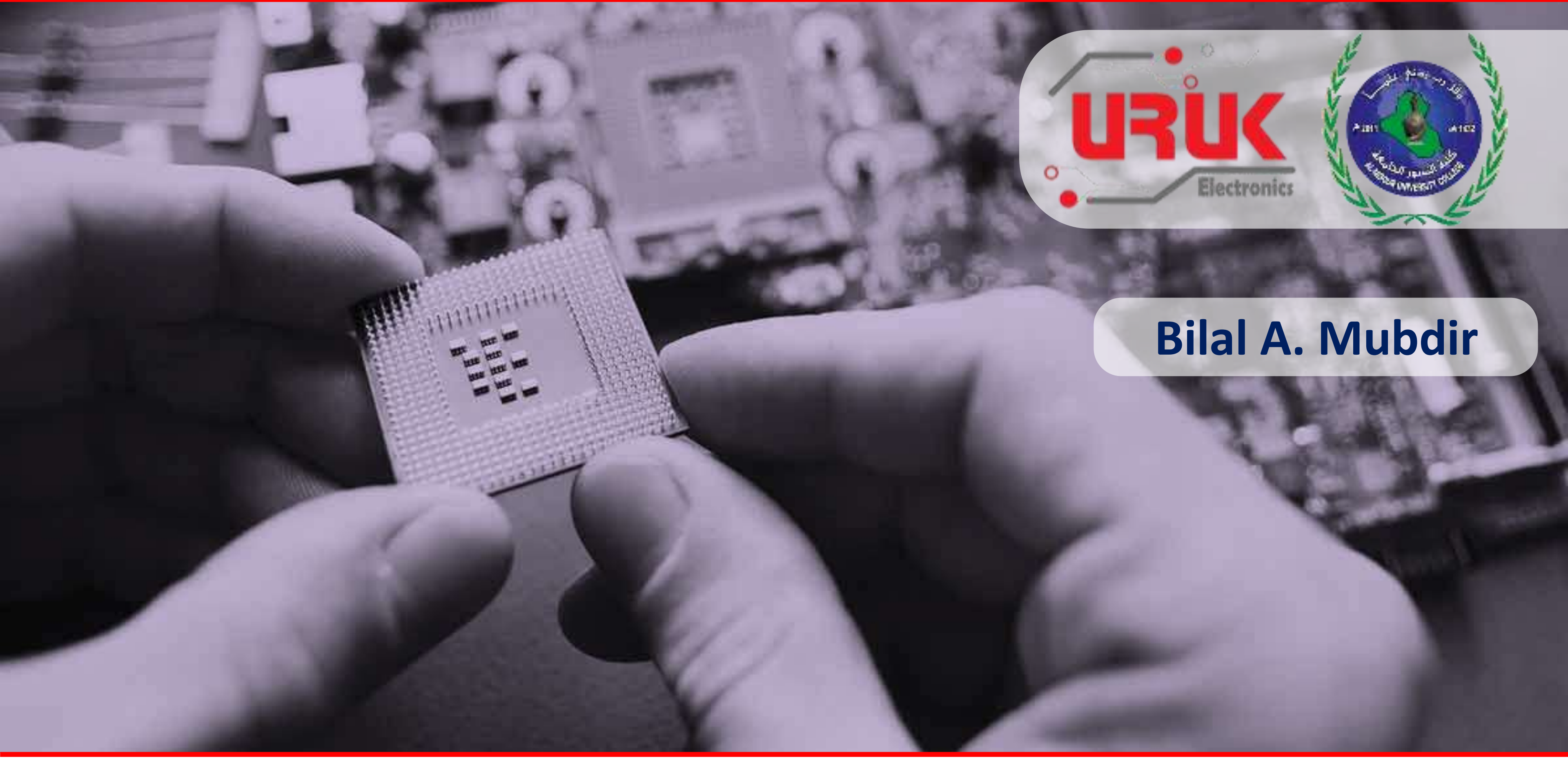
Workshop on **Embedded Systems Technology**

Embedded System Real Time Application



Bilal A. Mubdir

Training & Development Department
training@uruktech.com

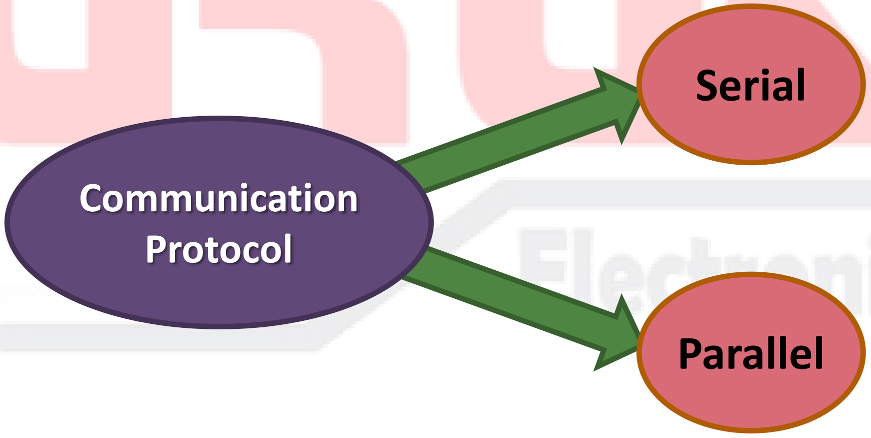
www.uruktech.com

Lecture Contents

- Communication in Embedded Systems
- Synchronous & Asynchronous Serial
- Rules of Asynchronous Serial
- Framing Data in Serial
- Wiring & Hardware of Serial Port
- Arduino Serial Functions
- What is Relay?
- Bluetooth Communication
- Real Time Application

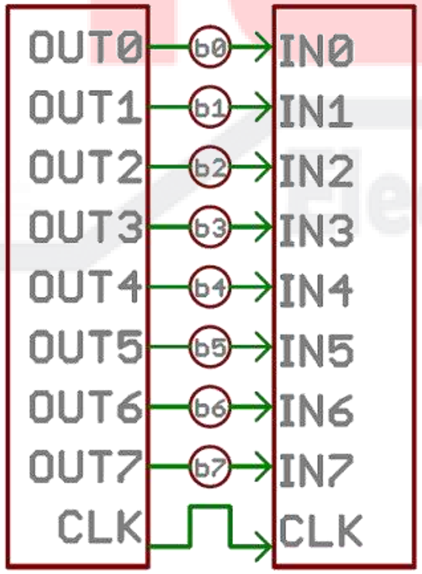
Communication in Embedded Systems

- Embedded Systems is all about **interlinking circuits** (microcontroller or other integrated circuits) to create a symbiotic system.
- In order for those individual circuits to **swap their information**, they must share a **common communication protocol**.



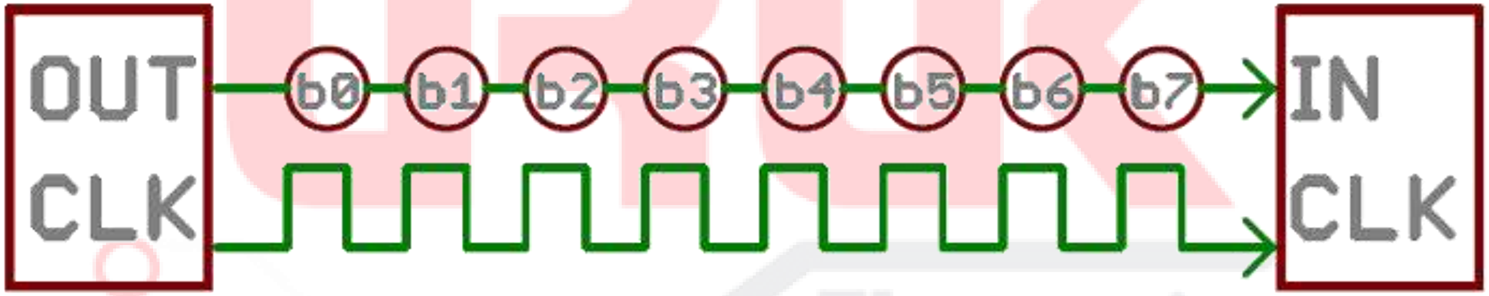
Communication in Embedded Systems

- **Parallel interfaces** transfer **multiple bits at the same time**.
- They usually require **buses of data** - transmitting across eight, sixteen, or more wires.
- Data is transferred in huge, crashing waves of 1's and 0's.



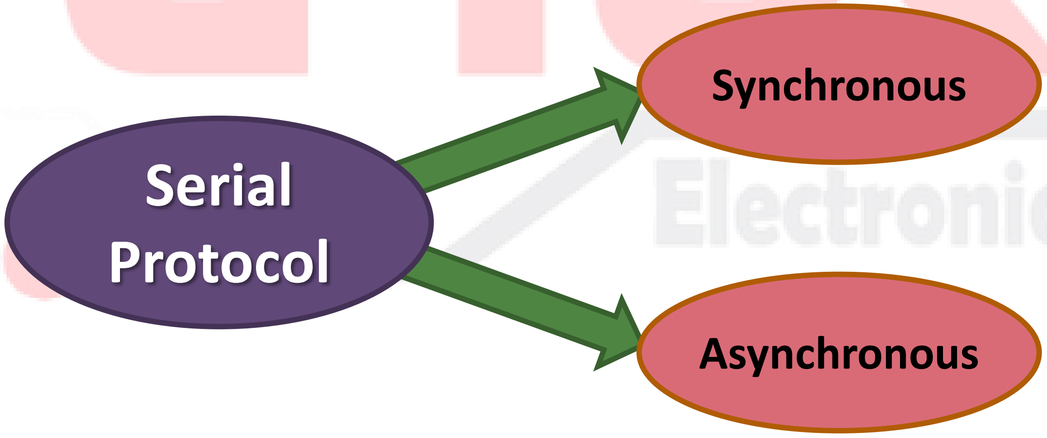
Communication in Embedded Systems

- **Serial interfaces** stream their data, one **single bit at a time**. These interfaces can operate on as little as **one wire**, usually **never more than four**.



Synchronous & Asynchronous Serial

- Over the years, dozens of serial protocols have been crafted to meet particular needs of embedded systems. USB (universal serial bus), and Ethernet, are a couple of the more well-known computing serial interfaces.



Synchronous & Asynchronous Serial

- A **synchronous** serial interface always **pairs its data line(s)** with a **clock signal**, so all devices on a synchronous serial bus **share a common clock**. It requires at least one extra wire between communicating devices.

Examples of synchronous interfaces:

- SPI
- I2C

Synchronous & Asynchronous Serial

- Asynchronous means that data is transferred **without support from an external clock signal**.
- This transmission method is perfect for **minimizing the required wires** and I/O pins, but it does mean we need to put some extra effort into reliably transferring and receiving data. The serial protocol we'll be discussing in this lecture is the most common form of asynchronous transfers. It is so common, in fact, that when most folks say **"serial"** they're talking about this protocol.

Rules of Asynchronous Serial

- The asynchronous serial protocol has a number of **built-in rules** that help ensure robust and error-free data transfers. These mechanisms, which we get for **avoiding the use of the external clock signal**, are:

❑ Data bits,

❑ Synchronization bits,

❑ Parity bits,

❑ and Baud rate.

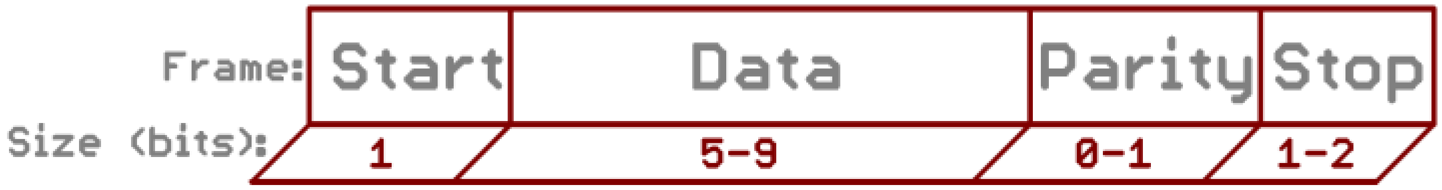
- The protocol is **highly configurable**. The critical part is making sure that both devices on a serial bus are configured to use the exact same protocols.

Rules of Asynchronous Serial

- The baud rate specifies **how fast data is sent over a serial line**.
- It's usually expressed in units of **bits-per-second (bps).**
- Baud rates can be just about any value within reason. The only requirement is that both devices operate at the same rate.
- One of the more common baud rates, especially for simple stuff where speed isn't critical, is **9600 bps**.
- Other "standard" baud are 1200, 2400, 4800, 19200, 38400, 57600, and 115200.

Framing Data in Serial

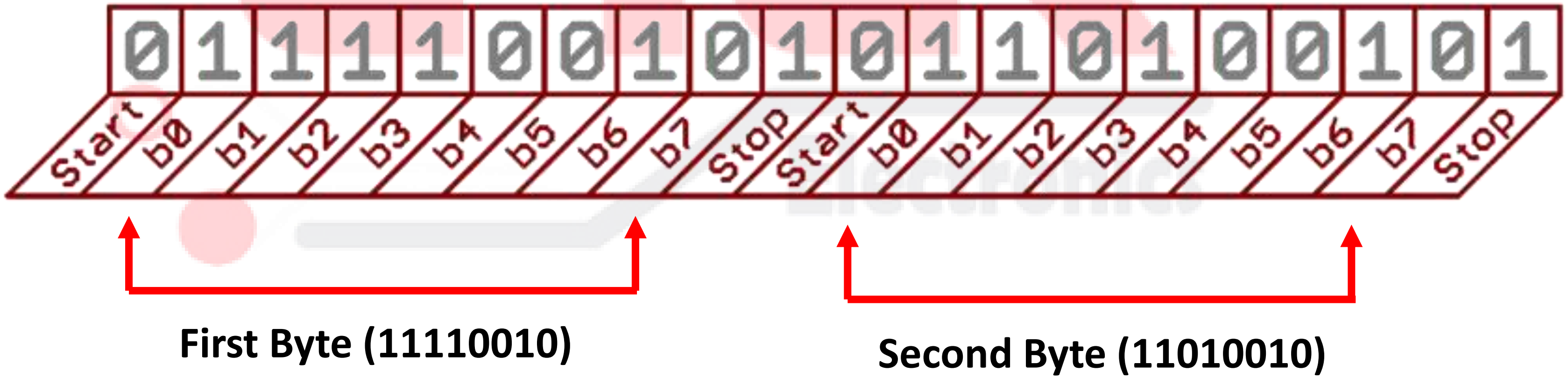
- Each block (usually a byte) of data transmitted is actually sent in a packet or frame of bits. Frames are created by appending synchronization and parity bits to our data.



Framing Data in Serial

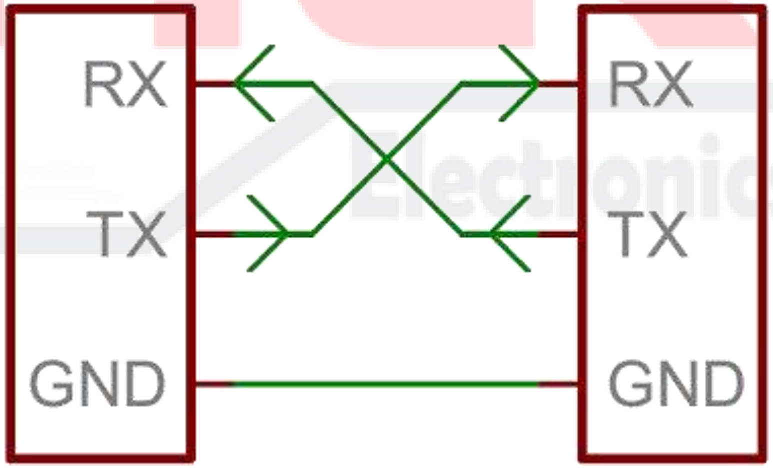
9600 8N1 (an example)

- 9600 8N1 - 9600 baud, 8 data bits, no parity, and 1 stop bit - is one of the more commonly used serial protocols. So, what would a packet or two of 9600 8N1 data look like?



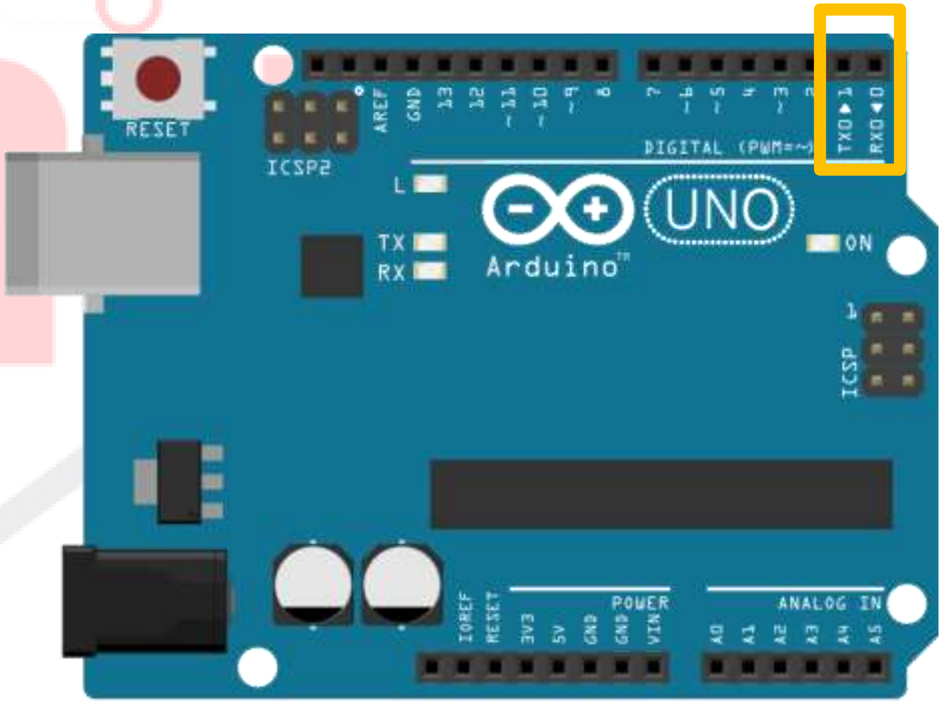
Wiring & Hardware of Serial Port

- A serial bus consists of just **two wires** - one for **sending data** and another for **receiving**. As such, serial devices should have two serial pins: the receiver, **RX**, and the transmitter, **TX**.



Arduino Serial Functions

- Serial communication on **pins TX/RX** uses TTL logic levels (5V or 3.3V depending on the board).

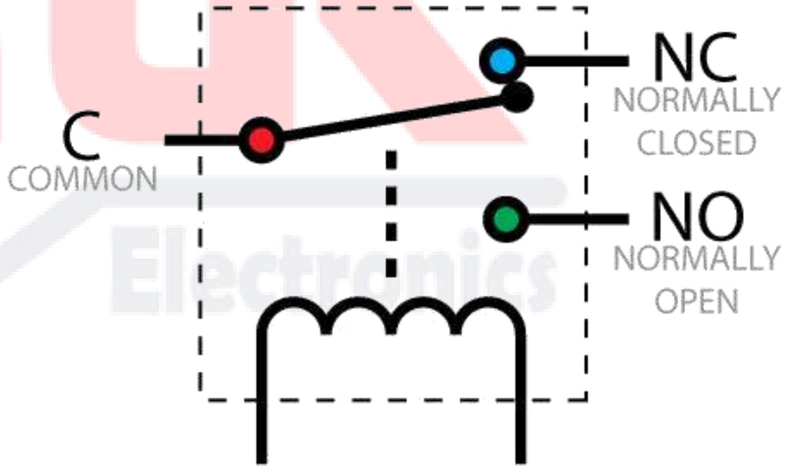


Functions:

- begin()
- available()
- flush()
- print()
- println()
- read()

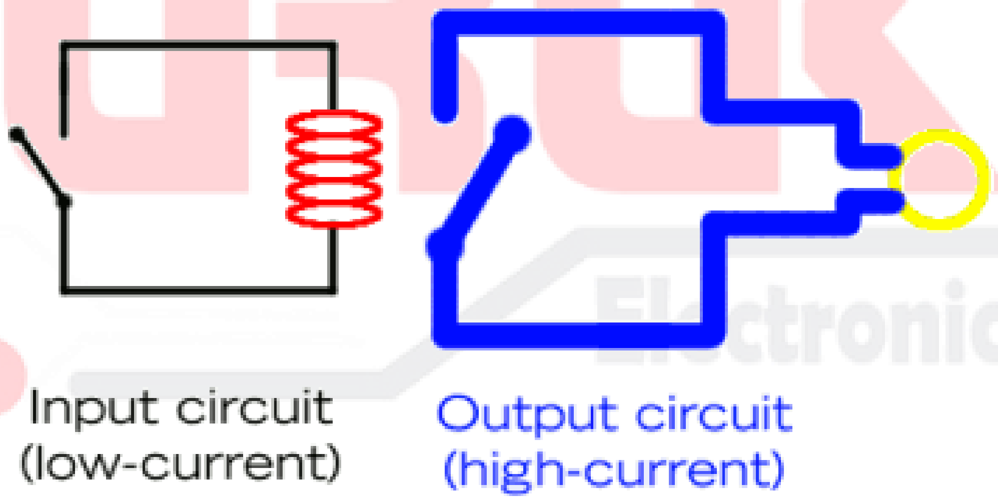
What is Relay?

- A relay is an **electromagnetic switch** operated by a relatively small electric current that can turn on or off a much larger electric current.

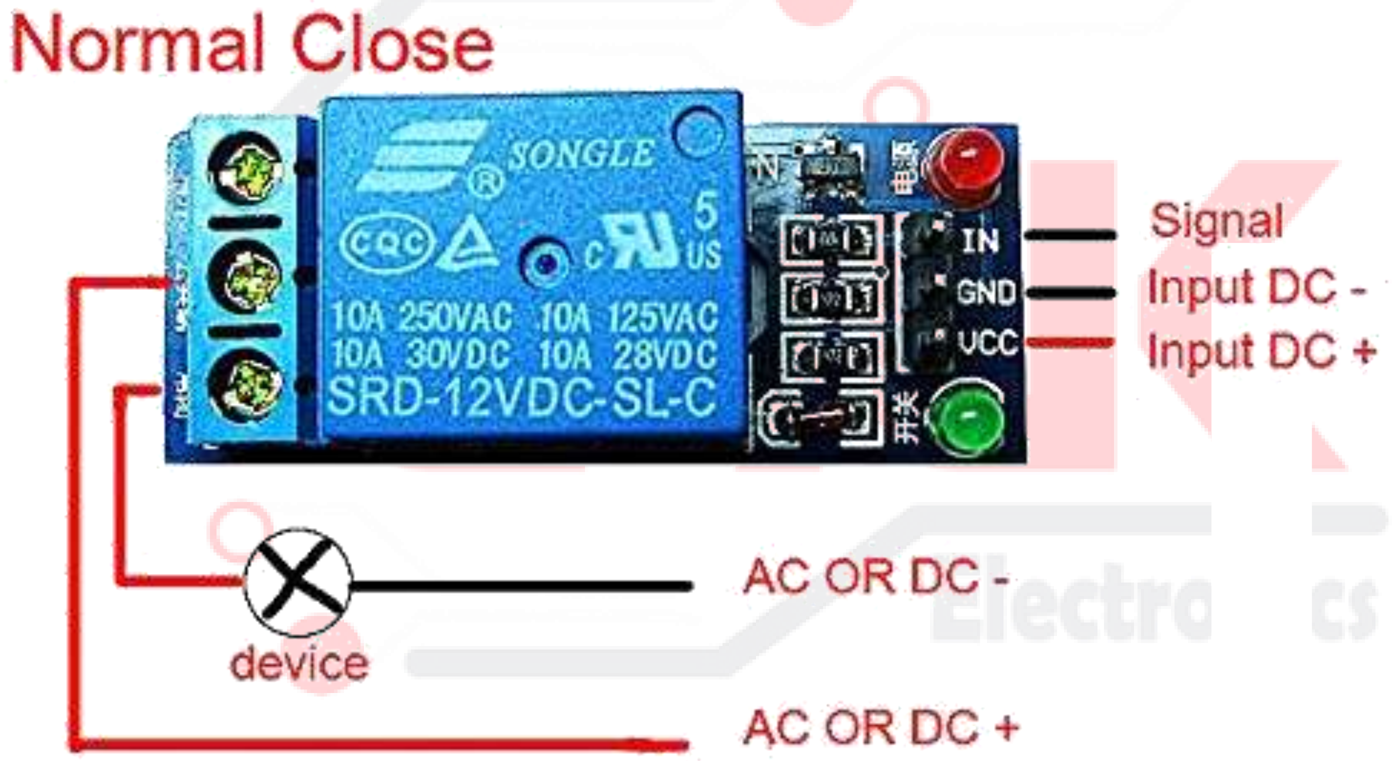


What is Relay?

- A relay is an **electromagnetic switch** operated by a relatively small electric current that can turn on or off a much larger electric current.



What is Relay?



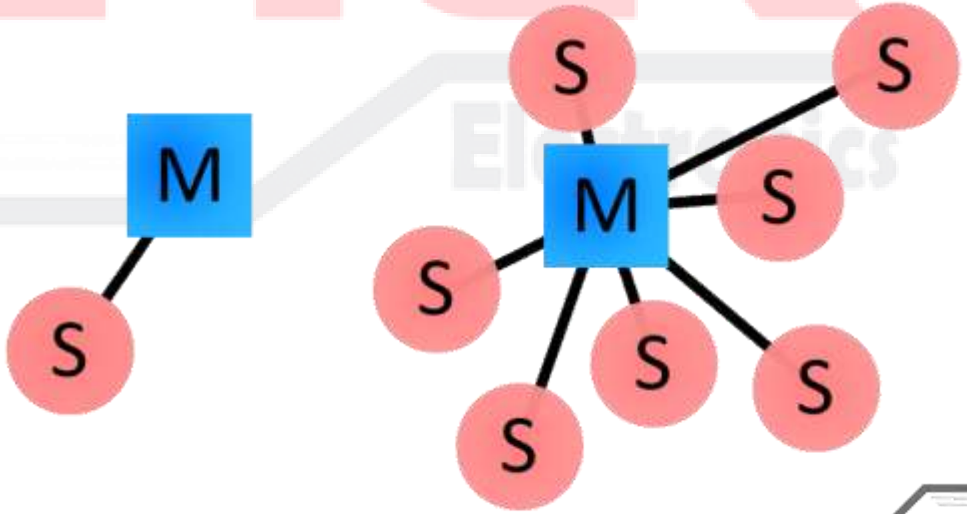
Bluetooth Communication

Bluetooth is a similar radio-wave technology, but it's mainly designed for **communicating over short distances** less than about 10m or 30ft. The Bluetooth protocol operates at 2.4GHz in the same unlicensed ISM frequency band where RF protocols like ZigBee and WiFi also exist. Typically, you might use it to download photos from a digital camera to a PC, to hook up a wireless mouse to a laptop, to link a hands-free headset to your cellphone so you can talk and drive safely at the same time, and so on.

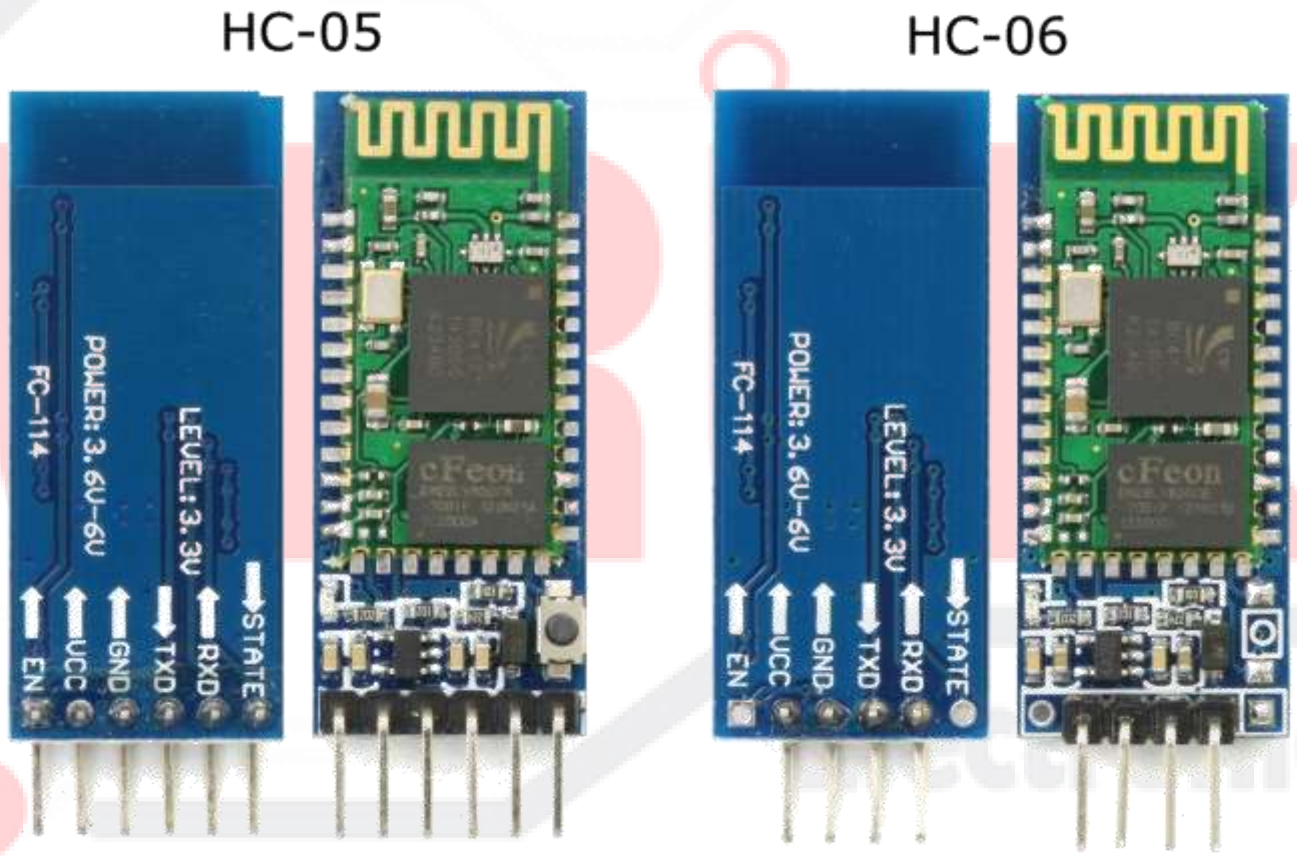
Bluetooth Communication

Masters, Slaves, and Piconets

- Bluetooth networks (commonly referred to as piconets) use a **master/slave model** to control when and where devices can send data. In this model, a single **master** device can be connected to up to **seven different slave** devices. Any slave device in the piconet can only be connected to a single master.

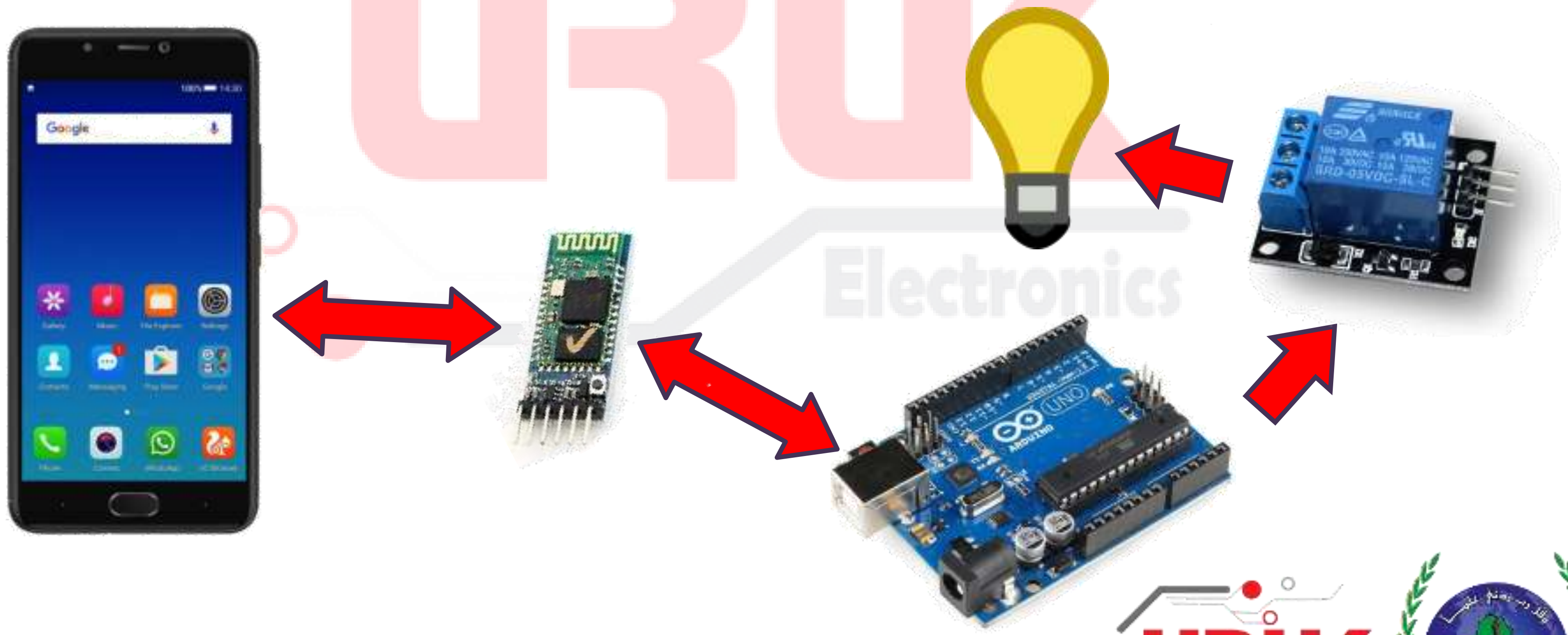


Bluetooth HC-05/06 Modules



Real Time Application

- Our application is to **turning ON/OFF** the **Hall's lighting** based on a **voice command** from the smart phone!



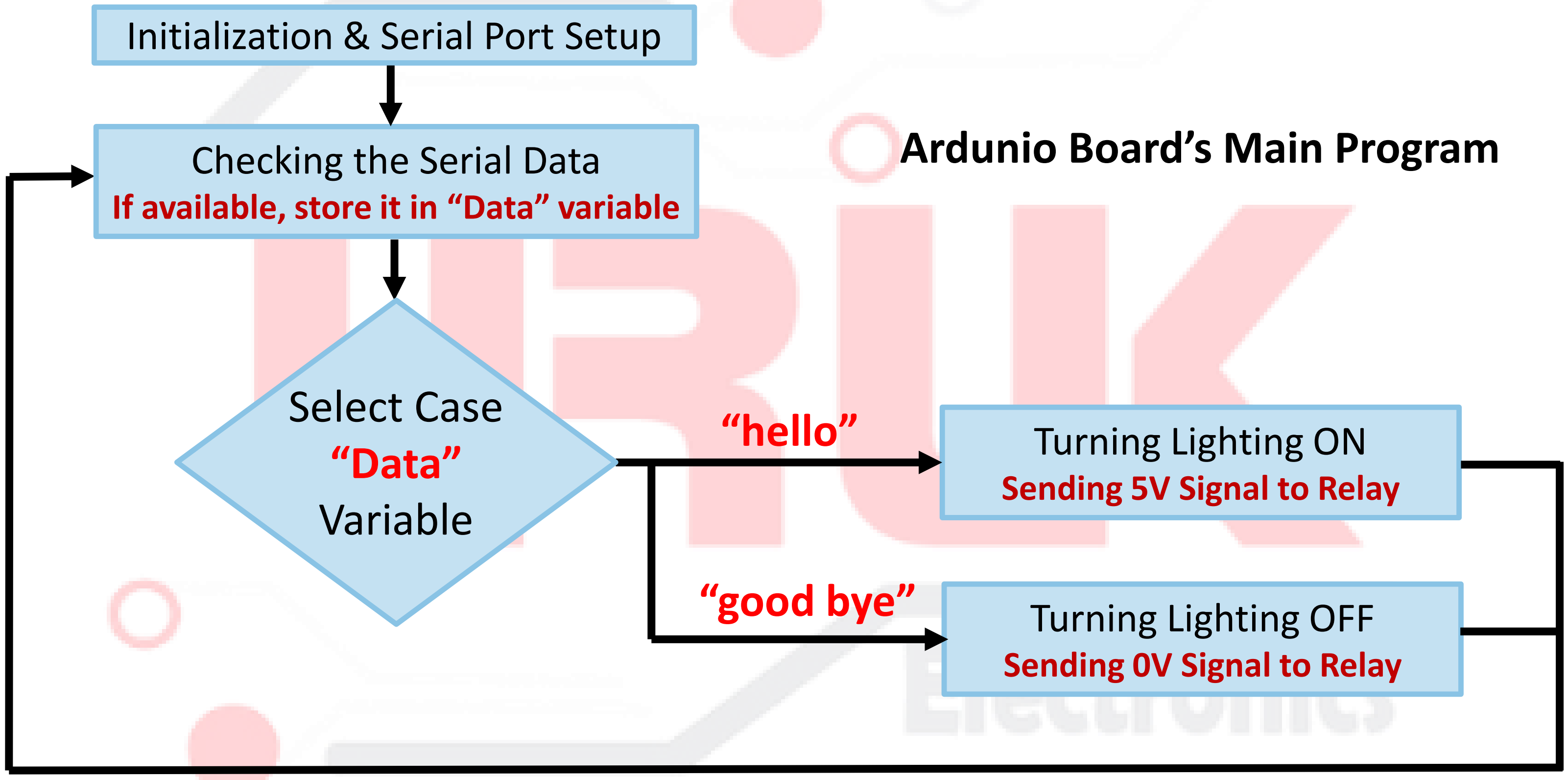
Real Time Application

- HC-05 module connected as below

HC-05 module's Side	Arduino's Side
Tx	Rx
Rx	Tx (stepped down to 3.3V)
VCC	5V
GND	GND

- Relay connected to Pin6
- Send any **text data** from **Mobile's by using Voice Recognition Application** to the Bluetooth after pairing it, turning the Lighting ON or OFF accordingly.

Real Time Application


Initialization & Serial Port Setup
Checking the Serial Data
If available, store it in "Data" variable
Select Case
"Data"
Variable
"hello"
Turning Lighting ON
Sending 5V Signal to Relay
"good bye"
Turning Lighting OFF
Sending 0V Signal to Relay
Ardunio Board's Main Program

Thank you …

Q&A