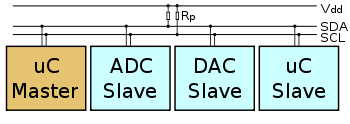
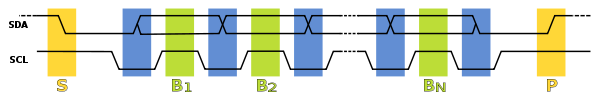
# 1. Write a brief summary about I2C protocol.

* I2C is a serial protocol for two-wire interface to connect low-speed devices like microcontrollers, EEPROMs, A/D and D/A converters, I/O interfaces and other similar peripherals in embedded systems.
* I2C has more than one master, only upper bus speed is defined and only two wires with pull-up resistors are needed to be connect almost unlimited number of I2C devices.
* I2C can use even slower microcontrollers with general-purpose I/O pins since they only need to generate correct Start and Stop conditions  in addition to functions for reading and writing a byte.
* Each slave device has a unique address. Transfer from and to master device is serial and it is split into 8-bit packets.
* The initial I2C specifications defined maximum clock frequency of 100 kHz. Fast mode uses 400 kHz. There is also a High speed mode which can go up to 3.4 MHz and there is also a 5 MHz ultra-fast mode.
* I2C uses only two wires: SCL (serial clock) and SDA (serial data). Both need to be pulled up with a resistor to +Vdd.I2C level shifters are used to connect to two I2C buses with different voltages.



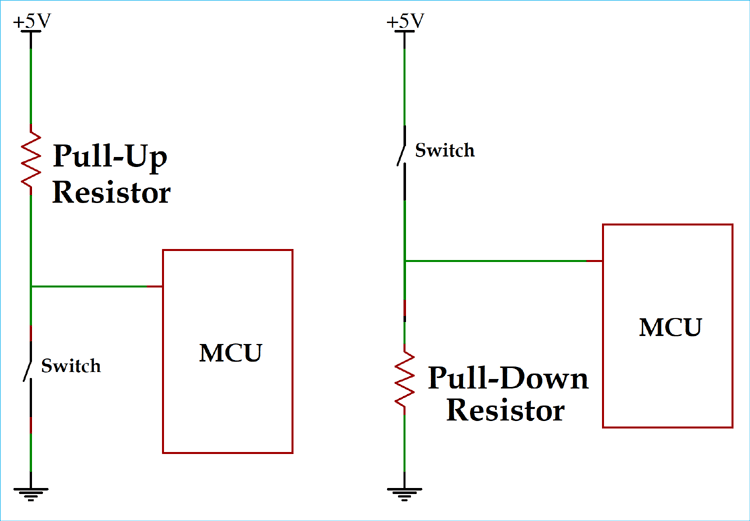
* I2C communication is using transfers of 8 bits or bytes. Each I2C slave device has a 7-bit unique address.
* Some devices have fixed I2C address while others have few address lines which determine lower bits of the I2C address. This makes it very easy to have all I2C devices on the bus with unique I2C address.
* 7-bit address represents bits 7 to 1 while bit 0 is used to signal reading from or writing to the device. If bit 0 (in the address byte) is set to 1 then the master device will read from the slave I2C device.
* Master device needs no address since it generates the clock (via SCL) and addresses individual I2C slave devices.
* In normal state both lines (SCL and SDA) are high. The communication is initiated by the master device.
* It generates the Start condition (S) followed by the address of the slave device (B1). If the bit 0 of the address byte was set to 0 the master device will write to the slave device (B2).
* Otherwise, the next byte will be read from the slave device. Once all bytes are read or written (Bn) the master device generates Stop condition (P). This signals to other devices on the bus that the communication has ended and another device may use the bus.
* Most I2C devices support repeated start condition. This means that before the communication ends with a stop condition, master device can repeat start condition with address byte and change the mode from writing to reading.
* I2C specifications are flexlible , I2C bus can communicate with slow devices and can also use high speed modes to transfer large amounts of data. Because of many advantages, I2C bus will remain as one of the most popular serial interfaces to connect integrated circuits on the board.



2. Write short text about, pull up resistor, pull down resistor, open drain, active low, active high.

pull up resistor

* A pull-up resistor is a resistor used to ensure a known state for a signal a pull-up resistor effectively establishes an additional loop over the critical components, ensuring that the voltage is well-defined even when the switch is open.
* A pull-up resistor connects unused input pins (AND and NAND gates) to the dc supply voltage, (Vcc) to keep the given input HIGH.
* To control the current flow, we need those pull-down or pull up resistors. A pull-up resistor allow controlled current flow from supply voltage source to the digital input pins.



pull down resistor

* Similarly to pull-up resistors, pull-down resistors ensure the voltage between VCC and a microcontroller pin is actively controlled when the switch is open. However, instead of pulling a pin to a high value, such resistors pull the pin to a low valued instead.
* The pull-down resistors could effectively control current flow from digital pins to the ground.
* A pull-down resistor connects unused input pins (OR and NOR gates) to ground, (0V) to keep the given input LOW.

 open drain

* The term “open drain” means there's a current sink, but on a FET device, for example, a MOSFET, the transistor will switch to ground when it's active, thus “sinking” current.
* An open-drain or open-collector output pin is driven by a single transistor, which pulls the pin to only one voltage.When the output device is off, the pin is left floating.
* Open-drain outputs can be useful for analog weighting, summing, and limiting as well as digital logic. An open drain terminal is connected to ground when a high voltage (logic 1) is applied to the gate, yet presents a high impedance when a low voltage (logic 0) is applied to the gate.

**Active low**

* It means the output is active low.when the output is active it has a logic low value, rather than a logic high. So, when the corresponding value is present on the inputs (1001 for example) the corresponding output (9) will be low and the rest will be high.
* The primary advantage to active low is safety. It just describes how the pin is activated. If it's an active-low pin, you must "pull" that pin LOW by connecting it to ground.

**Active High**

* In Active-high circuit, Both inputs are normally tied to ground (LOW), and the latch is triggered by a momentary HIGH signal on either of the inputs.
* An active high device is a device that either outputs a HIGH signal when triggered on or that accepts a high signal as input to turn on. It really depends on whether the device is an input or an output device. For an active high pin, you connect it to your HIGH voltage (usually 3.3V/5V).

 3.Short text about Linux booting process and the role of Kernel.

**Linux booting process**

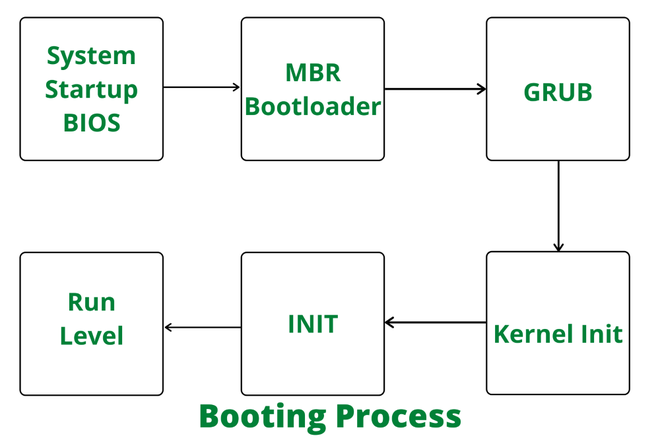
* There are many processes running in the background when we press the power button of the system. It is very important to learn the booting process to understand the working of any operating system.
* The Bootloader is the tool that loads the system software on the device and determines the priority for processes that run on the phone.
* Unlocking the Bootloader allows you to install custom firmware on your Android phone and gives you full access privileges to make modifications to the phone.
* Booting is the process of restarting a computer or its operating system software. It starts with switching on the computer and ends when the Operating System is loaded into main memory and the computer is ready to take commands from the user.
* Booting is of two types :

1. Cold booting: When the computer is started after having been switched off.

2. Warm booting: When the operating system alone is restarted after a system crash or freeze.

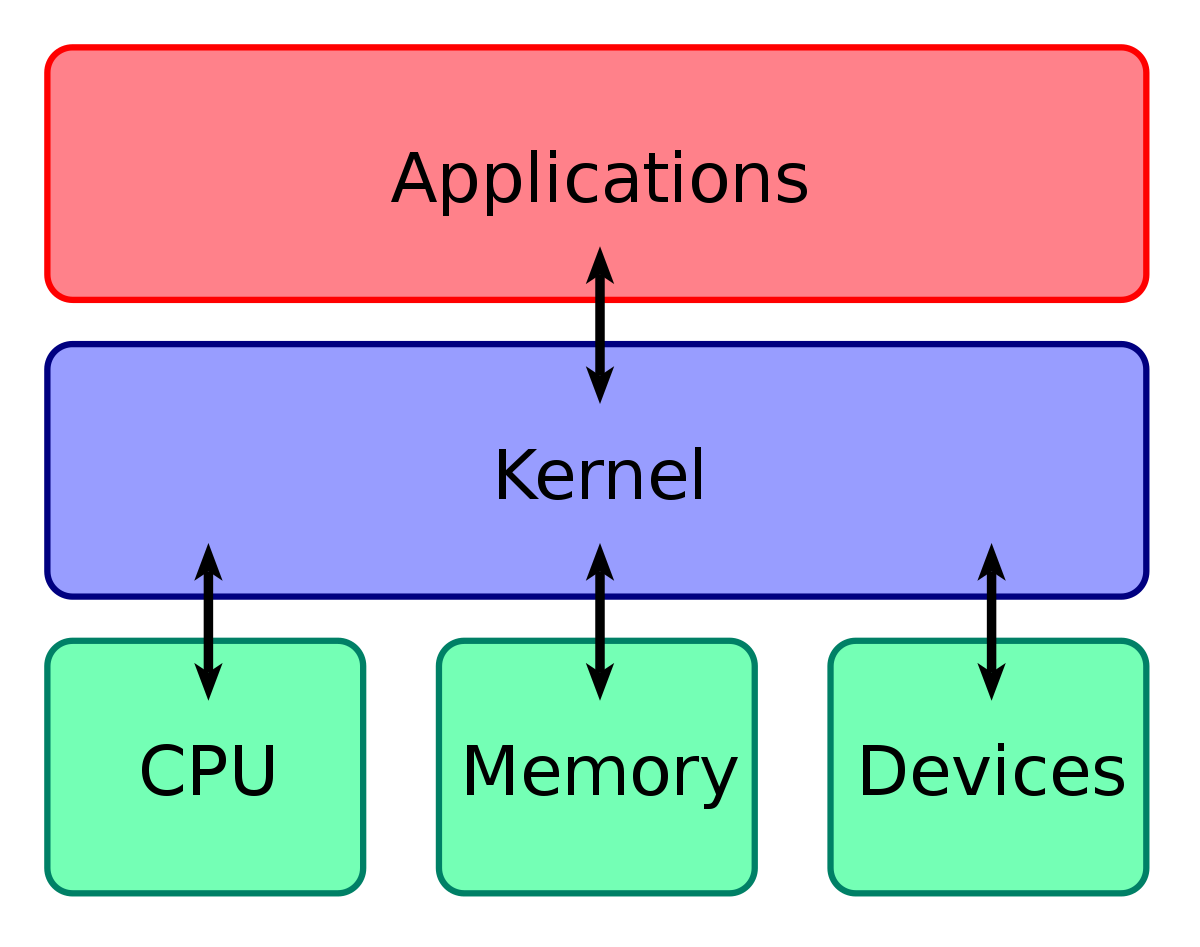
**Stages of Linux Boot Process**

* The machine’s BIOS or boot microcode hundreds and runs a boot loader.
* Boot loader finds the kernel image on the disk and loads it into memory, to start the system.
* The kernel initializes the devices and their drivers
* The kernel mounts the basis filesystem.
* The kernel starts a program referred to as init with a method ID zero
* init sets the remainder of the system processes in motion.
* For some purpose, init starts a method permitting you to log in, typically at the top or close to the top of the boot sequence.



**Role of Kernel**

* The kernel is the essential center of a computer operating system (OS). It is the core that provides basic services for all other parts of the OS.
* It is the main layer between the OS and hardware, and it helps with process and memory management, file systems, device control and networking.
* The Kernel is responsible for low-level tasks such as disk management, memory management, task management, etc.
* It provides an interface between the user and the hardware components of the system. When a process makes a request to the Kernel, then it is called System Call.
* Kernel acts as a bridge between applications and data processing performed at hardware level using inter-process communication and system calls.
* Kernel loads first into memory when an operating system is loaded and remains into memory until operating system is shut down again.



4.Text about first impression on Zephyr RTOS.

* Zephyr is originated from Virtuoso RTOS for digital signal processors (DSPs). The Zephyr Project source code is maintained in a Git repository.
* Zephyr is provided as source code and build scripts for different target architectures and configurations, and not as a binary image.
* The Zephyr OS is based on a small-footprint kernel designed for use on resource-constrained and embedded systems, from simple embedded environmental sensors and LED wearables to sophisticated embedded controllers, smart watches, and IoT wireless applications.
* The security functionality in Zephyr hinges mainly on the inclusion of cryptographic algorithms, and on its monolithic system design.
* The cryptographic features are provided through a set of cryptographic libraries. Applications can choose TinyCrypt2 or mbedTLS based on their needs.
* The Zephyr kernel supports multiple architectures, including ARM Cortex-M, Intel x86, ARC, NIOS II and RISC V, and a large number of supported boards