

Electronic Basics

This document is a summary of the videos from the YouTube playlist [Electronic Basics - YouTube](#).

Electronic Basics #1

Link: [Electronic Basics #1: The Multimeter](#)


Ohm's Law

The video starts by explaining the ohm's law. Modern electronics depend on the relationship between voltage, current, and resistance. This relation is known as Ohm's law.

$$I = \frac{V}{R}$$

Multimeter

Multimeter is used to measure these parameters. In this video a digital multimeter is used.

- **Probes:** Digital multimeter utilizes two probes whose correct placement is crucial for accurate measurement. Put the black probe always in the common socket. But you might need to change the position of red probe depending on the measurement of voltage or current.
- **Measuring Resistance :** While you can measure the resistance through the DMM just by connecting the probes with both ends of the resistor, measuring it in a built circuit will not give proper result. It is because the measuring current from the DMM will not flow only through one resistor.
- **Continuity:** Right next to ohm Ω sign, there is a function called continuity with a symbol like Wi-Fi icon . This functionality checks if there's zero resistance between the two probes. If it is, then the meter will beep. Useful for checking cable breaks.
- **Measuring Voltage :** Make sure to set multimeter to AC or DC voltage mode, depending on power source. We measure voltage in parallel.
- **Measuring Current :** Move the red probe to the 10A socket (high-current port). Always use this socket when measuring high currents, as the other port only handles up to 500mA—exceeding this may blow the fuse. Switch to smaller

range for more precision results. If the fuse is burned, the video also shows how to change the fuse of DMM.

Electronic Basics #2

Link: [Electronic Basics #2: Dimming all kinds of LEDs!?](#)

The basic of dimming an LED is just to lower the voltage. Current is proportional to voltage, as voltage decreases current also decreases and lower current makes the LED glow dimmer.

Possible solutions for dimming the LED brightness are –

Potentiometer

As we don't want to use a variable bench power supply for the light to get dimmer or brighter, we can use potentiometer in series to control the current with a fixed voltage.

But using potentiometer to dim a high-power LED is not accurate, as potentiometer is variable resistor. So, it heats up and wastes energy. Also controlling high-power LEDs need expensive potentiometers. So, another solution is using PWM.

Pulse Width Modulation (PWM)

Pulse Width Modulation is a technique to control analog devices with digital signals by switching power on and off rapidly which results in the average voltage or current. Different average voltages can be gained by varying the width of duration of these pulses.

The **duty cycle** describes the on time in one period. The LED is fully bright with 100% duty cycle. In a fixed 5V digital signal, 50% duty cycle represent 2.5 volts, where the on time is 50% and off time is 50%. Again 20% duty cycle means 20% on time and 80% off time in a cycle which represents around one volt.

PWM using Arduino and Potentiometer

Arduino takes analog output from potentiometer (valuer range 0 – 1023) which can be mapped to range 0 – 255. Here 0 resembles 0V and 255 represents 5V. The `analogWrite` function of the Arduino can generate PWM signals.

PWM using IC-555, MOSFET and potentiometer

Without using Arduino PWM can be generated using IC-555. For higher voltage usage a MOSFET can be used and by finally adding the potentiometer the brightness can be controlled.

Electronic Basics #3

Link: [Electronic Basics #3: Programming an Attiny+Homemade Arduino Shield - YouTube](#)

In video explains the method of programming a microcontroller. In this case, a ATtiny85 chip to control different animations in an LED strip.

This whole process can be done with an ATmega328P which is used in an Arduino uno. But ATmega328P costs more than other chips that can be used for this project. ATtiny85 is a more affordable option used here.

Prerequisites

ATtiny85 is used by replacing ATmega328P which is placed by default on the Arduino. But ATmega328P is needed to program on ATtiny85

- Arduino IDE is needed to be installed to program the microcontroller. In the video Arduino IDE version 1.0.5 is used.
- Board data for ATtiny85 also needs to be installed. [GitHub repository for ATtiny85 board data](#). Extract the zip file and copy the **attiny** folder into 'Arduino/hardware/arduino'. This will create new board options inside arduino.
- Upload the Arduino Isp found in the example section to the ATmega328P.

Configuration of ATtiny85

In the IC – ATtiny85, pin 4 is ground, pin 8 is VCC and other 5 IOs are pin 2, 3, 5, 6 and 7. IO 2, 3, 4 are used as analog Input. IO 0 and 1 are used for generating PWM. Pin 1 is RESET.

Connection of ATtiny85 to Arduino

These are the following pin connection with Arduino –

Pin 13 -> IO 2

Pin 12 -> IO 1

Pin 11 -> IO 0

Pin 10 -> Reset

5V -> VCC

GND -> Ground

Electronic Basics #4

Link: [Electronic Basics #4: Arduino+Bluetooth+Android=Awesome](#)

This video talks about using a Bluetooth module to control the LED strip from mobile phone. Here instead of Arduino Uno r3, an Arduino nano is used and a Bluetooth module HC-05.

Arduino Nano TX and RX

The Bluetooth module uses 3.3V level. More than that can damage the module. The Bluetooth module sends a 3.3V to the receive pin (RX) of the Arduino. But the Arduino sends 5V through the transmit pin (TX) which can damage the Bluetooth module. Hence, the voltage divider rule is applied with 2k Ω and 4.7k Ω to convert the transmit voltage to 3.5V approximately.

The RGB LED

The LED has four legs where three cathodes are used to signal RED, GREEN and BLUE. Each cathode gets a 460 Ω and connects to the Arduino digital pins (R -> 8, G -> 9 and B -> 10).

Control App

An app called S2 terminal is used to connect with the Bluetooth module.

The Arduino sketch can be found [here](#). Before uploading the code, cut the TX and RX connection before uploading the code, otherwise, it won't work.

Electronic Basics #5

Link: [Electronic Basics #5: How to Multiplex](#)

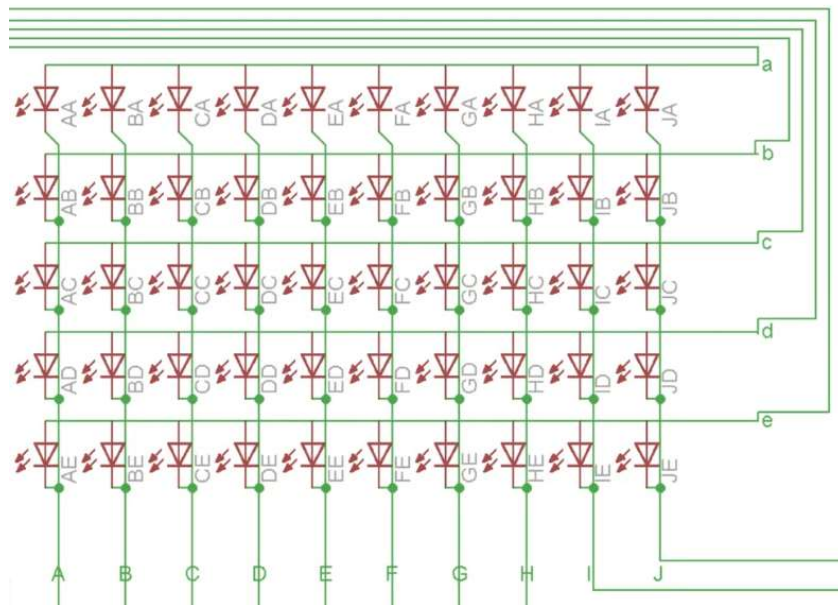
This video shows how to control each LED of a 10x5 LED strip individually. It shows the technique of controlling many LED with only few IOs in the Arduino Nano.

Prerequisites

Arduino Nano, TLC-5940 LED driver, five p-channel MOSFET, one 2k resistor, five 1k resistor.

10x5 matrix LED setup

Place the LEDs on the PCB in a 10x5 matrix. Connect All the cathodes (negative pins) in one column and connect the anodes (positive pins) in one row. Make sure that the columns and rows don't touch each other. In the end you get 10 cathode columns and 5 anode rows. Label the columns from A – J and rows from A – E which makes coding the Arduino easier. Here is the setup diagram of the LED strip -



In this configuration, if voltage given to 'a' and ground given to 'A' then LED 'AA' lights up. Every LED can be turned on individually like this. But if you only want to light up 'AA'

and 'JE' at the same time, then 'AE' and 'JA' also lights up. To solve this problem **Multiplexing** is used instead of lighting up individual LEDs.

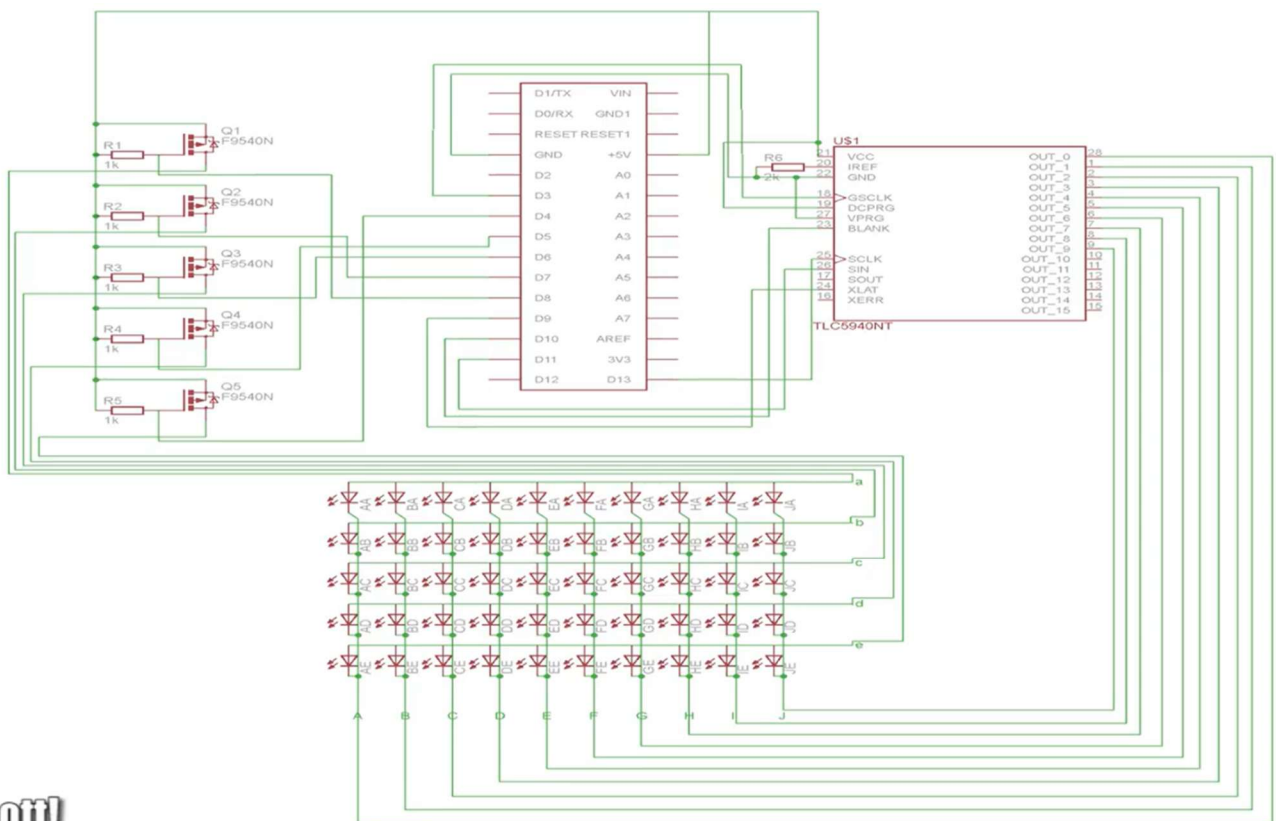
Multiplexing

Instead of dedicating a separate pin for each LED, multiplexing allows the system to activate LEDs in rapid succession, creating the illusion that multiple LEDs are lit simultaneously.

Instead of powering all LEDs at once, multiplexing turns them on and off in quick succession. This reduces power consumption and prevents excessive current draw.

The use of **P-channel MOSFETs and the TLC5940 LED driver** helps manage the switching process efficiently, ensuring smooth transitions between different LED states

Here is the schematic of the whole setup –



GreatScott!

The source of the MOSFET connected to VCC, ground to the IOs and drain to the anode rows.

TCL5940 configuration

Download the TLC5940 Arduino library from Arduino's official website. Place it in the library folder and upload the code to TLC.

Download the Arduino sketch for LED matrix [here](#) and run it.

Electrical Basics #6

Link: [Electronic Basics #6: Standalone Arduino Circuit](#)

This video demonstrates how to build a standalone Arduino circuit using an ATmega328p microcontroller. The video covers the necessary components, wiring, and programming methods, including using an FTDI chip for USB to serial conversion. The video also discusses the advantages and disadvantages of using a standalone circuit compared to a traditional Arduino board.

Prerequisites

One 16-megahertz clock crystal, two 22 pico-farad capacitors to generate external clock signals, one 10k Ω resistor to connect to the reset pin of Atmega328P and 5V

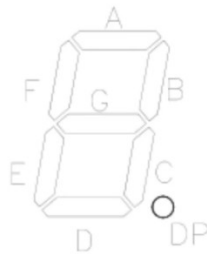
Electronic Basics #7

Link: [Electronic Basics #7: 7 Segment Display - YouTube](#)

This tutorial shows how to use 7-segment displays with and without Arduino. It talks about the different types of displays, how to connect them, and how to control them using BCD drivers and counters. It also explains multiplexing, which helps control more digits at once. The video focuses on real-life uses and suggests trying things out with the circuits and libraries provided.

There are different types of seven-segment displays, like single-digit and double-digit ones. Each type is useful for different kinds of projects.

There are 8 individual LEDs which have a single common anode configuration. 7 cathodes are used for the bars and other two are used for the decimal. Look through the specification sheet to see how to connect them.



From the specification sheet the bars are labeled from A – G and the decimal is labeled as DP.

To control the 7 segments display without a microcontroller we need a BCD. Using a BCD to 7-segment display driver makes it easier to control number displays. The SN74LS247 chip helps with this by controlling each segment (cathode) the right way. These BCD take 4 bits input and automatically turn on correct segments.

A 4-bit binary counter like the SN74LS290 is effective for controlling inputs without an Arduino. It has a BCD count sequence. So, it is easy to combine with the BCD.

For more than one segments display we need to use multiplexing as it reduces power consumption and requires fewer IOs to control the displays. For this IC-SLA1064 is used.

Electrical Basics #8

Link: [Electronic Basics #8: Everything about LEDs and current limiting resistors](#)

The tutorial explains how to use LEDs properly, emphasizing the importance of current limiting resistors. It covers essential parameters like forward voltage and current, calculations for resistor values, and the impact of voltage variations. The video also discusses series and parallel configurations and the best practices for driving LEDs with constant current sources.

Using resistors properly is crucial for LED circuits to prevent overheating and ensure efficient power usage. Series connections can reduce wasted power while providing adequate brightness for multiple LEDs. Use resistors to keep the current steady,

especially when the voltage might change. This helps protect the LEDs from getting damaged if too much current suddenly flows through them.

It's important to use the right voltage and current for LEDs, because if these are not correct, the current might not flow evenly, and the LEDs can stop working too soon. Using a constant current setup is usually better than just using constant voltage. Different LEDs have varying forward voltages, complicating the use of a single resistor which can lead to unequal current distribution. This can cause some LEDs to fail sooner than others.

Electrical Basics #9

Link: [Electronic Basics #9: Diodes & Bridge Rectifiers](#)

Discusses the function and importance of diodes, particularly in power supplies, and how they are used for **AC to DC conversion**.

Protecting Circuits from Reverse Polarity

One big reason we use diodes is to stop damage when power is connected the wrong way. In DC circuits, if you accidentally switch the positive and negative wires, your circuit might break. But if you place a diode with its **anode** connected to the positive voltage and its **cathode** to the circuit input, current will only flow in the right direction. If you connect it backward, the diode blocks the current, so nothing gets damaged.

Voltage Drop and Power Loss in Diode

Diodes aren't perfect, though. When current flows through them, there's always a small **voltage drop**, the input voltage is a bit higher than what the circuit actually receives. This drop depends on how much current is being used. That means some power gets wasted as heat. If a circuit uses a lot of current, the diode might get hot, and you might need a bigger or stronger diode.

Turning AC into DC with Diodes

Diodes are really useful for turning **AC (alternating current)** into **DC (direct current)** . That's because diodes only let current pass in one direction. The video shows that if you use just one diode with a transformer, the result is a really rough DC signal—it only uses the positive parts of the wave and throws away the negative parts. Adding a capacitor can smooth things out a little, but if the circuit draws current, the voltage still bounces up and down, because the capacitor charges and discharges with each wave.

What a Bridge Rectifier Does

To fix the problem of losing the negative parts of the AC wave, the video shows how to use a **bridge rectifier** , which is a setup using four diodes. This setup is clever because it makes sure that no matter which direction the AC current is flowing, the output still goes in the same (correct) direction. Basically, it flips the negative part of the wave to make it positive. This gives you a much more stable DC signal, which is better for powering circuits.

Electronic Basics #10

Link: [Electronic Basics #10: Digital to Analog Converter \(DAC\)](#)

This video talks about how digital signals can be turned into analog signals using a DAC (Digital to Analog Converter). It explains why this is useful and how it works. Digital devices like microcontrollers work with simple on/off signals usually called high/low or 1 and 0. But the real world uses analog signals, like sound waves or light brightness, which can have many levels. A DAC helps us turn those digital signals into smooth analog voltages so we can do things like create sound, adjust light smoothly, or generate waveforms.

The R-2R resistor ladder method

There are many ways to make a DAC, and the video focuses on one simple method called the R-2R resistor ladder. This method uses resistors with two specific values (like $10\text{k}\Omega$ and $20\text{k}\Omega$) arranged in a pattern. Arduino's digital pins are used to send binary values (high or low), and the resistor network turns that into a voltage. The more bits you use, the more precise the output. For example, an 8-bit DAC can create 256 different

voltage levels. The output voltage depends on how accurate your resistors are. If the resistors aren't exactly the right value, the output might not be as smooth or correct.

Different analog signals like – ramp, triangle wave, sine wave can be made with this setup.

A speaker directly to the resistor ladder, the voltage can change or drop. This happens because the load draws current and messes up the signal. To fix this, you can use a **voltage follower** (usually an op-amp). This doesn't boost the signal—it just keeps it steady even when something is connected to it.

Smoothing out PWM with an RC filter

To make the PWM signal more like a real analog output, you can use an **RC filter**—that's just a resistor and a capacitor

Dedicated **DAC chip**, like the **DAC0800** can be used for converting digital signals to analog and give more precise output.

Electronic Basics #11

Link: [Electronic Basics #11: Sending SMS with Arduino || TC 35 GSM Module](#)

This video shows how to use a TC35 GSM module with an Arduino UNO to send SMS messages.

Connect 5V to VCC. Although 12V can work, it's risky unless MAX232 chip is removed. The board only needs around 6mA in standby, and you'll need to press a button on the module to get it to log into the mobile network. Once it connects, a small LED will blink every few seconds. You can also control this button from the Arduino using a digital pin (like pin 10).

Communication with the module happens using AT commands over serial. It works at 9,600 baud and accepts 5V logic, so it's safe with the Arduino. You can connect to it through TX, RX, and GND pins.

The video walks through an example Arduino sketch that sends an SMS. You write your message in the Serial Monitor, end it with a **dot (.)**, and the Arduino handles sending it to a predefined phone number.

Electronic Basics #12

Link: [Electronic Basics #12: Coils / Inductors \(Part 1\)](#)

This video introduces coils, also called inductors, which are really important basic components in electronics.

When electricity flows through a wire, it creates a magnetic field around it. If you wind the wire into a coil, the field becomes stronger. And if you add something like an iron core in the middle, it gets even more powerful. That's how things like electromagnets work. Devices like relays use this effect to switch things on and off. The coil's ability to create a magnetic field is measured as inductance, and the unit for that is Henry.

Lenz's Law: As the current starts to flow and creates a magnetic field, the coil pushes back with its own voltage, trying to resist the change. It also does this when the current is turned off, trying to keep the current flowing for a bit longer.

In a DC circuit, coils don't behave like normal resistors. When you turn on the voltage, the current doesn't jump to the final value right away, it slowly builds up. The more inductance a coil has, the longer this takes. This happens because of **Lenz's Law**.

In boost converters, the energy stored in the coil helps increase the voltage. For example, it can turn 3.7V into 5V by quickly switching the coil on and off.

Coils are also used in step-down converters to store energy and help keep the output voltage steady.

Electronic Basics #13

Link: [Electronic Basics #13: Coils / Inductors \(Part 2\) || Reactance - YouTube](#)

This video continues from the previous part and explains how coils (inductors) behave in AC circuits. In DC circuits, inductors resist sudden changes in current, but in AC circuits, they behave a bit differently.

When you put a coil in an AC circuit, it shows something called **reactance**. This is kind of like resistance, but instead of turning energy into heat, the coil stores energy in a magnetic field and then gives it back. This also leads to **reactive power**, where power oscillates between the voltage source and the load, which can strain power grids.

Key factors that affect the inductive reactance –

- Decreasing the inductance of the coil leads to less reactance and consequently more current flow, following Ohm's Law principles
- The reactance is directly dependent on the **frequency** of the AC signal. Higher frequencies result in more reactance, causing less current to flow

The formula of reactance is –

$$\text{Reactance, } X_{\{L\}} = 2\pi fL$$

Because coils behave differently at different frequencies, they're super useful for filtering signals:

- coil with a **resistor**, can make a **high-pass filter**, which blocks low-frequency signals and lets high ones through.
- low-pass filters can be made, which block the high frequencies and let the lows pass.

In resistor-only circuits, voltage and current rise and fall together. But with inductors, the current lags behind the voltage—they get out of sync. This phase shift can go up to 90 degrees in a perfect coil. You can change the amount of phase shift by adjusting the coil's inductance.

Electronic Basics #14

Link: [Electronic Basics #14: Capacitors](#)

This video explains the importance and common use of capacitors

A capacitor is made of two conductive plates placed close together, separated by a small gap. When you connect a DC voltage across the plates, a brief current flows—just long enough to charge the plates. This creates an electrostatic field between them, which stores energy. Unlike resistors, which turn energy into heat, capacitors store it. Capacitors don't store as much energy as batteries, but they release it much faster, making them super useful in electronic circuits.

Capacitance depends on –

- The surface area of the plates. The more area, the more capacitance
- The distance between the plates. The closer, the more capacitance
- The material between the plates.

Beyond capacitor's voltage rating, it can be destroyed

In DC circuits, capacitor's voltage can't change instantly, but current can. When you apply voltage, current flows at first, then slowly decreases as the capacitor charges. This makes them perfect for smoothing voltage at power supply outputs, decoupling ICs to prevent noise. Creating time delays with resistors

In AC circuits, capacitor resist current flow with capacitive reactance. The higher the frequency of the signal, the easier it is for current to flow.

$$\text{capacitive reactance, } X_C = \frac{1}{2\pi fC}$$

this frequency-dependent behavior makes capacitors incredibly useful in filters:

- RC low-pass filters block high frequencies.
- RC high-pass filters block low frequencies.
- These are simpler and cheaper than filters using coils (RL filters).

Just like coils cause current to lag voltage, capacitors do the opposite - they cause current to lead voltage.

This phase shift is useful for:

- Balancing out inductive loads (like motors) in power systems
- Reducing reactive power that burdens the electrical grid

Electronics Basics #15

Link: [Electronic Basics #15: Temperature Measurement \(Part 1\) || NTC, PT100, Wheatstone Bridge](#)

The video explains how we measure temperature in electronics, a super common need in things like 3D printers, power supplies, or anything industrial. It turns out there are several ways to do it, each with its own pros and cons.

NTC Thermistors are cheap, simple sensors whose resistance drops as temperature rises. You can get them in 1K, 10K, and 100K versions (measured at 25°C). They're great for high resolution, meaning they can detect tiny changes well, but they're not very accurate, and the relationship between temperature and resistance isn't linear. Still, they're handy up to around 150°C.

PT100s sensors are more precise and used in professional setups. They start at 100 ohms at 0°C, and resistance increases almost linearly with temperature. They're good up to around 850°C and much more stable than NTCs.

We can use a **constant current** and measure voltage, but small voltages are hard for microcontrollers to read. We can add a **reference voltage** and subtract it with a differential op-amp. Finally, we can use a **Wheatstone bridge** —a classic setup that converts resistance changes into a measurable voltage difference.

In all cases, you'll usually need to **amplify** the signal before feeding it to a microcontroller, and precision resistors are key

We can also use ready-made transmitter ICs. These cost around \$5, and they output a standard current signal. Connect it to a resistor, read the voltage, and you're done. The video shows this in action with an LCD and a microcontroller.

Electronic Basics #16

Link: [Electronic Basics #16: Resistors - YouTube](#)

This video covers about resistors.

Resistors are one of the core building blocks of electronics. Their main job is to limit current. For example, if you connect an LED directly to a power supply, you'll probably burn it out. But throw a resistor in series, and it keeps the current at a safe level by converting excess energy into heat. You can calculate how much resistance you need using Ohm's Law.

Not all resistors are created equal. A standard ¼-watt resistor can't handle much heat. For anything pulling more power, you need a power resistor, which is physically larger and designed to safely dissipate more energy.

Resistors also let you divide voltage. This is super useful, like when converting a signal from 5V down to 3.3V for a microcontroller. You just chain two resistors together in series.

A potentiometer is basically a variable resistor. Turn the knob, and the resistance changes. These are handy for things like volume control, screen brightness, or as input knobs for microcontroller projects.

When you have a digital input pin that's not connected to anything, it can "float" unpredictably. That's where pull-up or pull-down resistors come in. They keep things stable:

- Pull-down connects the pin to ground, keeping it low.

- Pull-up connects it to VCC, keeping it high.

Current Sensing

Put a very small resistor (called a current sense resistor) in line with your load, measure the voltage across it, and you can calculate how much current is flowing using Ohm's Law. This is great for building constant current sources, or just for monitoring.

Sacrificial Resistor

Resistors can even act as cheap fuses—if you purposely push too much power through them, they'll burn out, protecting other components.

In AC Circuits

Unlike capacitors or inductors, resistors don't cause a phase shift in AC circuits—they behave the same as they do with DC. However, at high frequencies, real-world resistors start to act a bit weird due to parasitic inductance and capacitance. So in high-speed or RF circuits, you have to watch out for these non-ideal behaviors.

Electronic Basics #17

Link: [Electronic Basics #17: Oscillators || RC, LC, Crystal](#)

This video teaches about oscillators, the circuits that make repeating signals like square, sine, or triangle waves.

Oscillators are circuits that generate repeating AC voltage signals. These waveforms are everywhere in electronics:

- Clocks in microcontrollers, computers, and multimeters.
- Carrier signals in radios or data communication
- Generating tones, signals, or time bases in all kinds of systems

Relaxation Oscillators

These are simple, often used in beginner projects. One common example is the astable multivibrator, which creates a square wave. Here's how it works –

- Two capacitors charge and discharge alternately through resistors.
- When one reaches a certain voltage, it triggers a transistor to discharge it.
- This back-and-forth charging makes the voltage flip up and down like a blinking LED.

The 555 Timer IC

A classic chip that uses this idea. Add a couple of resistors and a capacitor, and you get a nice, stable square wave output. Internally, it switches at about $\frac{1}{3}$ and $\frac{2}{3}$ of the supply voltage. You can also tweak the output frequency by changing the component values or using a potentiometer.

LC Oscillators

Used for high-frequency sine waves.

What happens in this oscillator –

- The capacitor charges up.
- It dumps energy into the inductor, building a magnetic field.
- That field collapses and recharges the capacitor in the opposite direction
- This resonant “**ping-pong**” continues as long as energy is available.

This back-and-forth creates smooth sine waves. But in real life, some energy is lost as heat because of resistance. To keep the oscillation going, you need to feed energy back into the system, usually using an **amplifier** (like a transistor) that's tuned to boost the signal just right.

The **resonance frequency** (where it oscillates best) depends on the values of L and C. Tweak those, and you get different frequencies.

Crystal Oscillator

Used for rock-solid frequency stability. A quartz crystal naturally vibrates at a specific frequency when you apply voltage sort of like a tuning fork for electricity. These vibrations are incredibly consistent and are used in:

- Microcontrollers (to set exact clock speeds like 16 MHz)
- Watches and real-time clocks
- Radios and precision instruments

Electronic Basics #18

Link: [Electronic Basics #18: DC & Brushless DC Motor + ESC](#)

This video is about DC motors, Brushless DC motors, and Electronic Speed Controllers.

DC Motors

DC motors are simple and have been around for ages. They work by pushing current through coils inside the motor, which generates magnetic fields that interact with permanent magnets to make the motor spin.

- Inside, you've got a rotor (the part that spins) with coils, and a stator (stationary) with permanent magnets.
- The coils on the rotor are connected to a commutator, and carbon brushes press against this to feed in electricity.
- As the motor spins, the commutator and brushes automatically reverse the current direction in the coils at just the right time, keeping the rotation continuous.
- In short, it's a mechanical solution to generating a rotating magnetic field.

Brushless DC Motors

In brushless DC, instead of a rotating coil and stationary magnets, the coils stay still, and the magnets spin.

- The rotor holds permanent magnets (usually alternating N/S poles), while the stator has the coils.

- No brushes, no commutator, so no physical contact, which means more efficiency, and quieter operation.

Electronic Speed Controllers

ESCs are the brains behind brushless motors.

- They take a control signal (usually a voltage) and use it to decide how fast to switch the coil currents on and off.
- These switching steps create a rotating magnetic field that spins the rotor.
- The faster these steps happen, the faster the motor spins

Internally, ESCs use MOSFETs (both N- and P-channel types) to switch the voltage on and off cleanly and efficiently. The more MOSFETs in parallel, the more current the ESC can handle, which means you can drive more powerful motors (but at a higher cost).

PWM (Pulse Width Modulation) is also used to modulate the power output, especially to control torque or efficiency.

Electronics Basics #19

I2C, or Inter-Integrated Circuit, is a two-wire, synchronous serial communication protocol. It's designed to let a master device (like an Arduino Nano) talk to multiple slave devices using just two wires: SDA (Serial Data Line) and SCL (Serial Clock Line).

Master Slave Setup

- There's usually one master, though multiple masters are allowed.
- The master initiates communication and controls the clock.
- You can connect up to 112 slaves, each with a unique 7-bit address.

Physical Connection

- SDA (A4) and SCL (A5) on Arduino Nano.
- Both lines require 10k Ω pull-up resistors to work correctly. Because I2C devices use open-drain/open-collector outputs, which can only pull the line low. The pull-

up resistors ensure the line goes high when no device is pulling it down, allowing proper high/low voltage states (needed for binary 1s and 0s).

The Communication Process

1. Start Condition

Triggered when SDA goes low while SCL is high. This signals the start of communication. handled by libraries like `Wire.begin()`.

2. Sending the Address

The master sends the 7-bit address of the target device. Followed by a Read/Write bit: '0' = write, '1' = Read.

3. Acknowledge (ACK) Bit

After receiving the address and R/W bit, the slave replies with an ACK—a signal saying, “Got it, I’m ready.”

4. Data Transfer

Data moves in 8-bit chunks (bytes). Whether sending or receiving, each byte is followed by an ACK. Datasheets are crucial: they tell you what bytes to send to get the slave device to do something useful.

5. Stop Condition

Communication ends when SDA goes high while SCL is high.

Electronic Basics #20

Link: [Electronic Basics #20: Thyristor, Triac || Phase Angle Control](#)

This video explains the function of Thyristors, Triacs, and how they’re used in phase angle control. It introduces Thyristors and Triacs as key components in AC power control, especially in phase angle control applications such as light dimming, motor speed regulation, and heating control.

Triacs: A Triac is essentially two Thyristors in inverse parallel, allowing control of both halves of the AC waveform. It can conduct current in both directions and is commonly used for AC phase control applications.

Phase Angle Control Circuit:

The source describes a microcontroller-based phase angle control circuit using a Triac to adjust AC power delivery.

- **Zero-Cross Detection**

A full-bridge rectifier + optocoupler detects the zero-crossings of the AC waveform. This detection generates a **short pulse** at each zero crossing.

- **Microcontroller Logic**

The zero-crossing pulse is fed to an interrupt pin on the Arduino. A potentiometer (connected to an analog pin) sets the delay (0–10 ms) after each zero crossing. After this delay, the Arduino triggers a second optocoupler, which activates the Triac, turning on the load for the remainder of the half-cycle.