Video 01: The Multimeter

To measure **resistance**, the probes should be connected to **both ends of the resistor**. Measuring the resistance of a resistor **in a built circuit** may show **less than its actual resistance**, because **current always takes** A multimeter is used for measuring **voltage**, **current**, and **resistance**. It has two probes: a **black probe** and a **red probe**. The **black probe is always connected to the common (COM) socket**, while the **red probe is connected to different sockets** depending on whether you're measuring **current**, or **voltage and resistancethe path of least resistance**.

To measure **voltage**, the probes should be connected **in parallel** with the load. For the two types of voltage — **AC and DC** — the signs used are **'⎓' for DC** and **'~' for AC**. Thus, we can measure both AC and DC voltages.

To measure **current**, first connect the *red probe into the current socket*, then connect the multimeter **in series with the load**. Always start with the **high-current socket**. For more accurate readings, switch to the **low-current socket** if needed. Like voltage, we can measure **both AC and DC current**.

When the **probes of a multimeter touch each other and it beeps**, it means you're in **continuity mode**. In this mode, the multimeter checks whether **electricity can flow between two points**. If you touch both ends of a wire and it **beeps**, the wire is **not broken**. If it **doesn't beep**, the wire is **broken**.

Video 2: Dimming all kind of LED’s

The video explains about the PWM (Pulse With Modulation) which is a technique to control the brightness of a LED. Lowering voltage to dim LEDs is easy but impractical in fixed-voltage. Using a potentiometer as a dimmer has drawbacks like energy waste and component strain, especially with high-power LEDs. Instead, PWM manipulates the LED’s brightness by rapidly switching the power on and off at a specific frequency, altering the duty cycle to simulate varying voltage levels without overheating components. The duty cycle—the ratio of the on-time to the total period—controls the perceived brightness, with 100% meaning fully on and lower percentages dimming the LED accordingly. Arduino’s analogWrite function easily generates PWM signals. Less duty cycle makes light more dimmer.

Video 3: Programming an Attiny+Homemade Arduino Shield

This video explains the creator tackles the challenge of controlling a WS2801 LED strip with a separate microcontroller that cycles through different animations by a push button.

Cost-efficiency and Compactness of ATtiny85: The ATtiny85 microcontroller offers a very economical and compact alternative compared to the Arduino Uno’s ATmega328, costing about a quarter and significantly reducing size while still providing enough I/O pins and memory for simple LED animation projects. This approach is ideal for embedded or productized electronics where cost and footprint matter.

Arduino Uno as an ISP Programmer: Using an Arduino Uno to program the ATtiny85 via Arduino ISP is a creative and accessible solution that leverages existing hardware without the need for dedicated programmers. This makes the ATtiny85 more approachable for beginners.

Video 04: Arduino+Bluetooth+Android=Awesome

This video demonstrates how to control an RGB LED using an Arduino Nano and an HC-05 Bluetooth module, with commands sent from an Android smartphone.

Introduction: The video starts by highlighting the utility of Bluetooth for controlling gadgets with a smartphone, even referencing a "Big Bang Theory" clip.

HC-05 Bluetooth Module: It introduces a small HC-05 (or similar JY-MCU) Bluetooth module with 4 main pins (VCC, GND, TXD, RXD). It shows an example of using it to control LED strips in a living room.

Core Project Setup:

Components: Arduino Nano, HC-05 Bluetooth module, an RGB LED (common anode), three current-limiting resistors for the LED cathodes, and two resistors (2kΩ and 4.7kΩ) for a voltage divider.

Voltage Level Issue: A key point is that while the HC-05 module can be powered by 5V (like the Arduino), its logic levels (for TX/RX data lines) are 3.3V. The Arduino Nano operates at 5V logic levels.

Solution (Voltage Divider):

The Bluetooth module's TX (transmit) pin can connect directly to the Arduino's RX (receive) pin because 3.3V is high enough for the Arduino to register as a 'high' signal.

However, the Arduino's TX pin (outputting 5V) should not be connected directly to the Bluetooth module's RX pin (expecting 3.3V) as this could damage the module. A voltage divider (using a 2kΩ and 4.7kΩ resistor) is used to step down the Arduino's 5V TX signal to approximately 3.4V before it reaches the Bluetooth module's RX pin.

Wiring:

The RGB LED's common anode is connected to 5V. Its red, green, and blue cathodes are connected via their respective current-limiting resistors to Arduino digital pins (8, 9, and 10 in the video).

The Bluetooth module's VCC is connected to 5V, GND to GND.

Bluetooth TX connects to Arduino RX.

Arduino TX connects to the input of the voltage divider, and the output of the voltage divider (between the 2k and 4.7k resistors) connects to the Bluetooth RX.

Android App: The video uses the "S2 Terminal for Bluetooth" app (available on the Google Play Store) to send ASCII string commands (e.g., "red", "green", "blue") to the Arduino via Bluetooth.

It uses strcmp (string compare) to check if the received string matches predefined commands like "red", "green", or "blue".

Based on the command, it uses digitalWrite to turn the corresponding LED color pin LOW (since it's a common anode LED, LOW turns it ON) and the others HIGH (OFF).

It also sends a confirmation message back to the phone (e.g., "Red ON").

Demonstration: The video shows typing "red", "green", and "blue" into the S2 Terminal app, and the RGB LED on the breadboard changes color accordingly, with the phone displaying the confirmation messages.

Video 5: How to Multiplex

Problem:

The video starts by highlighting the issue of controlling a large number of LEDs (e.g., a 4x4x4 RGB LED cube with 192 "effective" LEDs, or a 10x5 matrix with 50 LEDs) when microcontrollers (like Arduino Nano, Uno, or even Mega with 54 I/Os) don't have enough individual output pins.

Solution Principle: Multiplexing (using a 10x5 LED Matrix as an example)

The core idea is to use multiplexing, specifically by controlling rows and columns.

Matrix Wiring:

LEDs are arranged in a grid (10 columns, 5 rows).

All cathodes (negative leads) of LEDs in a single column are connected together.

All anodes (positive leads) of LEDs in a single row are connected together.

This results in 10 common column connections and 5 common row connections.

Scanning Technique:

To light an individual LED (e.g., LED at row 'a', column 'A'), power is applied to row 'a' (anode high) and the corresponding column 'A' is pulled low (cathode low).

The problem with trying to light multiple, non-adjacent LEDs simultaneously this way is "ghosting" (unintended LEDs light up).

The solution is to rapidly turn on one row at a time, set the states of its columns (which LEDs in that row should be on/off), then turn that row off and move to the next.

If this scanning is done fast enough (e.g., >50-60 Hz for the entire matrix), persistence of vision makes the human eye perceive a stable, complete image.

Components Used for the 10x5 LED Matrix Demonstration:

Arduino Nano: The microcontroller to run the code and control other components.

1 x TLC5940 LED Driver IC: A 16-channel constant current LED driver. 10 of its outputs are used to control the 10 cathode columns of the LED matrix (sinking current).

5 x F9540N P-channel MOSFETs: Used as high-side switches to control the 5 anode rows of the LED matrix (sourcing current). MOSFETs are used because a full row of 10 LEDs can draw significant current (e.g., 10 LEDs * 20mA/LED = 200mA), which is too much for a single Arduino pin.

1 x 2kΩ Resistor: Connected to the IREF (current reference) pin of the TLC5940. This resistor sets the maximum current for each LED driven by the TLC5940 (in this case, to about 20mA).

5 x 1kΩ Resistors: Used as pull-up resistors for the gates of the P-channel MOSFETs, connecting the gate to 5V. This ensures the MOSFETs are turned OFF when the Arduino pin controlling them is HIGH or floating. An Arduino pin pulls the gate LOW to turn the P-channel MOSFET ON.

Circuit Setup (Simplified):

Rows (Anodes): Each of the 5 anode rows of the LED matrix is connected to the Drain of a P-channel MOSFET. The Source of each MOSFET is connected to 5V. The Gate of each MOSFET is controlled by a digital output pin on the Arduino Nano (via a 1kΩ pull-up resistor to 5V).

Columns (Cathodes): Each of the 10 cathode columns of the LED matrix is connected to one of the 10 outputs of the TLC5940 IC.

TLC5940 Control: The TLC5940 is controlled by the Arduino Nano using its serial interface pins (SCLK, SIN, XLAT, BLANK, GSCLK).

Arduino Power: The Arduino Nano is powered via USB.

Software (Arduino Code):

The code uses the Tlc5940 library (which can be downloaded from the Arduino Playground).

Setup: Pins are defined, and the TLC5940 is initialized.

Loop (Multiplexing Logic):

Iterate through each row (0 to 4).

For the current row:

Turn OFF all other rows (by setting their corresponding Arduino control pins for the MOSFETs to HIGH).

Turn ON the current row (by setting its Arduino control pin for the MOSFET to LOW).

Use Tlc.set(channel, brightness) to set the brightness (or on/off state) for each of the 10 LEDs (columns) in the currently active row. channel corresponds to the TLC5940 output/matrix column (0-9). brightness is a value from 0 (off) to 4095 (full brightness).

Call Tlc.update() to send the new LED states to the TLC5940.

A small delay (RowDuration in microseconds) is added to keep the row lit before moving to the next.

The video demonstrates simple animations like a sine wave, the static word "HOW," and a moving letter "E" by changing the Tlc.set() values for each frame of the animation within the multiplexing loop.

The RowDuration variable in the code controls how long each row is active. If too long, flicker becomes visible.

Video 6: Standalone Arduino Circuit

Goal: Run an Arduino project with just the ATmega328P chip.

1. Core Parts:

ATmega328P (with bootloader)

16 MHz Crystal

2x 22pF Capacitors

1x 10kΩ Resistor

5V Power

2. Basic Wiring:

Power: ATmega Pins 7 & 20 to 5V; Pins 8 & 22 to GND.

Reset: 10kΩ resistor from ATmega Pin 1 (RESET) to 5V.

Clock: Crystal across ATmega Pins 9 & 10. Each crystal pin to GND via a 22pF capacitor.

3. Pinout:

Use an ATmega328P pinout diagram to map Arduino pins (e.g., D9) to the chip's physical pins (e.g., Pin 15).

4. Programming (if chip has bootloader):

Easiest (Temporary): Put ATmega328P in an Arduino Uno socket, program, then move back.

Common (Standalone): Use an Arduino Uno (chip removed/disabled) or an FTDI programmer to connect:

Programmer 5V/GND → Breadboard 5V/GND

Programmer TX → ATmega RXD (Pin 2)

Programmer RX → ATmega TXD (Pin 3)

Programmer Reset/DTR → ATmega RESET (Pin 1) (often via 0.1uF cap)

Upload from Arduino IDE.

5. Key Differences from Arduino Board:

Needs external 5V regulated power.

No built-in USB.

No onboard protections or reset button.

This setup is for embedding projects compactly.

Video 7: 7 Segment Display

This video serves as a tutorial on using 7-segment displays, both with and without a microcontroller like an Arduino.

First, the video explains the basics of 7-segment displays, showing single and dual-digit common anode types (e.g., LTS-546AG). It stresses the importance of consulting datasheets to understand pinouts, segment labeling (A-G and DP for decimal point), and electrical characteristics like forward voltage.

The tutorial then demonstrates how to control a single 7-segment display without a microcontroller. This is achieved using two ICs:

An SN74LS247 (BCD to 7-segment decoder/driver), which takes a 4-bit BCD (Binary Coded Decimal) input and outputs the correct signals to light up the segments for a given digit.

An SN74LS290 (4-bit binary counter with BCD count sequence capability), which generates the 4-bit BCD input for the SN74LS247.

A simple circuit is built on a breadboard where a push button increments the counter, and the corresponding digit (0-9) is displayed.

Next, the video addresses controlling multiple 7-segment displays, which is often necessary for projects like clocks. It explains that directly driving all segments of multiple displays would consume too many Arduino pins. While briefly mentioning multiplexing as a solution (referencing a previous video on the topic), the main focus shifts to using a specialized IC:

The SAA1064, an I^2C (I2C) bus controlled 4-digit LED-driver. This IC can drive up to four 7-segment displays (multiplexing two pairs of digits internally) and communicates with the Arduino using only two wires (SDA and SCL for I^2C).

A breadboard circuit is assembled using an Arduino Nano, the SAA1064, and two dual-digit 7-segment displays. The video highlights the "cool-SAA1064-lib" Arduino library (found on GitHub) which simplifies the I^2C communication and control of the SAA1064. An example sketch from this library is uploaded to the Arduino, demonstrating the display of various numbers and characters across the four digits, showcasing the ease of use provided by the IC and library.

Video 8: Everything about LEDs and current limiting resistors

This video by GreatScott! is a tutorial on how to correctly power and use LEDs, aimed at both beginners and more advanced users.

For Beginners:

Understanding LED Specs: LEDs typically have a specified "forward voltage" (Vf) and an ideal "forward current" (If, usually around 20mA for common LEDs).

Why Resistors are Needed: Directly connecting an LED to a voltage source higher than its Vf will cause too much current to flow, burning out the LED. A resistor is used to limit this current.

Calculating Resistor Value:

Use Kirchhoff's Voltage Law: Source Voltage = LED_Vf + Resistor_Voltage_Drop. So, Resistor_Voltage_Drop = Source_Voltage - LED_Vf.

Use Ohm's Law: Resistance (R) = Resistor_Voltage_Drop / Desired_LED_Current (If).

Resistor Power Rating: Calculate the power dissipated by the resistor (P = V_resistor * I_LED) to ensure it doesn't overheat (e.g., use a 1/4W resistor if power is below 0.25W).

Multiple LEDs in Series: Connecting LEDs in series is more power-efficient. The total Vf is the sum of individual Vf's. Recalculate the resistor based on this new total Vf.

For Advanced Users:

Vf Variation: The actual forward voltage of an LED can vary slightly from the manufacturer's specification and even between LEDs from the same batch. It's best to measure it.

Problem with No Resistor (or very small one): If the source voltage is very close to the LED's Vf, the LED's V-I curve is very steep. Small changes in source voltage can lead to large, potentially damaging changes in current.

LEDs in Parallel (Common Mistake): Connecting multiple LEDs in parallel with a single current-limiting resistor is generally bad practice. Due to Vf variations, one LED might draw more current than others, leading to uneven brightness and potentially burning out that LED, which can then cause a cascade failure for the others.

Constant Current Drivers: The best way to drive LEDs is with a constant current source, which ensures the LED gets the correct current regardless of minor Vf variations or supply voltage fluctuations. Examples shown are a simple LM317 circuit and the TLC5940 (used in his LED cube).

Video 9: Diodes & Bridge Rectifiers

Diodes are electronic components that act like one-way valves for electricity, allowing current to flow in only one direction (from anode to cathode).

Key uses shown:

DC Reverse Polarity Protection: If you connect power backwards to a DC circuit, the diode blocks the current, preventing damage.

AC to DC Conversion (Rectification):

Half-Wave: A single diode can convert AC to DC by only letting the positive (or negative, depending on orientation) half of the AC wave pass, resulting in a bumpy DC.

Full-Wave (Bridge Rectifier): Using four diodes in a "bridge" configuration, both the positive and negative halves of the AC wave are converted into a more continuous (though still bumpy) DC output.

Smoothing: A capacitor is often used after rectification to smooth out the "bumps" in the DC, making it more stable.

Diodes cause a small voltage drop when conducting, which means a little power is lost as heat. Different sized diodes are used depending on the amount of current they need to handle.

Video 10: Digital to Analog Converter (DAC)]

This video explains how to convert digital signals (like those from an Arduino) into analog signals (like sound). It shows:

The difference between digital and analog signals: Digital signals are discrete (e.g., on/off, high/low, 0/1), while analog signals are continuous and can have many voltage levels.

Building an R-2R Resistor Ladder DAC: A simple method to create a Digital-to-Analog Converter using two resistor values (R and 2R) connected to multiple digital pins of an Arduino. The number of pins (bits) determines the resolution (number of analog voltage steps). An 8-bit DAC provides 256 voltage levels.

Demonstrating the R-2R DAC: The Arduino is programmed to output different digital patterns to the R-2R ladder, creating various analog waveforms (ramp, triangle, sine) visible on an oscilloscope.

Driving a Load (Speaker): An op-amp (Operational Amplifier) configured as a voltage follower is used to buffer the DAC's output, allowing it to drive a small speaker without collapsing the signal.

Arduino's analogWrite (PWM) vs. True DAC: The video clarifies that Arduino's analogWrite function produces a Pulse Width Modulation (PWM) signal, not a true analog voltage.

Filtering PWM to Analog: An LC (Inductor-Capacitor) low-pass filter is demonstrated to smooth out the PWM signal, effectively converting it into a more analog-like voltage.

Using I^2C DAC Modules: Simpler, dedicated DAC ICs (like the PCF8591 and MCP4725) that communicate with the Arduino via the I^2C protocol are presented as easier alternatives for generating true analog outputs.

In short, the video explores various methods, from DIY resistor ladders to dedicated ICs, for generating analog signals from a digital source like an Arduino.

Video 11: Sending SMS with Arduino || TC 35 GSM Module

This video demonstrates how to use a TC35 GSM module (often on a blue breakout board) with an Arduino Uno to send SMS messages.

The process involves these main stages:

Hardware Setup:

SIM Card: A SIM card is inserted into the module, but its PIN lock must be disabled beforehand using a regular phone.

Power: The module is powered (typically 5V to its breakout board).

Network Login: A button on the module (or an Arduino-controlled pin) is used to trigger the module to connect to the mobile network.

Serial Connection: The module's TX/RX pins are connected to the Arduino's serial pins (often SoftwareSerial) for communication, along with a common ground.

Arduino Control:

The Arduino is programmed to send AT commands to the GSM module.

These commands tell the module to:

Enter text mode for SMS.

Specify the recipient's phone number (in international format).

Accept the message content.

Send the SMS (often by sending a Ctrl+Z character).

Sending the SMS:

The Arduino code manages this sequence. The video shows how to input the message via the Arduino Serial Monitor.

Once the commands are correctly sent, the GSM module transmits the SMS, which is then received by the target smartphone.

Essentially, it's about physically connecting the components, ensuring the SIM is ready and the module is on the network, and then using the Arduino to "talk" to the module via AT commands to compose and send your text message.

Video 12: Coils / Inductors (Part 1)

The video explains that coils (inductors) are fundamental electronic components, basically just loops of wire.

How they work: When electricity flows through a coil, it creates a magnetic field. More loops or an iron core (like in an electromagnet) make this field stronger.

Key Behavior: Coils resist sudden changes in electrical current.

When you apply voltage, the current doesn't build up instantly; the coil slows it down.

When you try to stop the current, the coil's collapsing magnetic field tries to keep it flowing, which can create a high voltage spike.

Uses:

This energy storage and resistance to change is used in power supplies (like phone chargers to convert voltage), motors, relays (electrical switches), and transformers.

Important Consideration: The voltage spike can damage other components, so often a flyback diode is used with coils to provide a safe path for this energy.

In short, coils use magnetism to store energy, smooth out current, and are crucial for many electronic devices.

Video 13: Coils / Inductors (Part 2) || Reactance

Inductors (coils) create a special kind of resistance to alternating current (AC), called inductive reactance (XL). This isn't like the normal resistance of a wire that just creates heat.

Instead, as AC tries to flow through an inductor:

A magnetic field builds up and collapses repeatedly.

This process "fights" against changes in the current.

The higher the frequency of the AC, or the larger the inductance of the coil, the more reactance it has, and the more it limits the AC current.

This also causes the current to lag behind the voltage (they're "out of phase"). Because their reactance changes with frequency, inductors are great for making filters that block or pass certain frequencies.

Video 14: Capacitors

capacitors are electronic components that store energy in an electric field.

Here's a breakdown:

What they are: Fundamentally, capacitors are two conductive plates separated by an insulating material (called a dielectric). The video demonstrates this by making a simple one with aluminum sheets.

How to increase capacitance:

Increase the surface area of the plates.

Decrease the distance between the plates.

Use a better dielectric material (like distilled water or the paper/plastic film inside commercial capacitors).

How they work in DC circuits:

They charge up, storing energy. The voltage across them can't change instantly.

Once charged, they block DC current.

Useful for smoothing power supplies (like in phone chargers), decoupling ICs, and in timing circuits (like with a 555 timer).

How they work in AC circuits:

They pass AC current, but offer a type of resistance called "capacitive reactance," which decreases as frequency or capacitance increases.

This makes them useful for filtering (e.g., RC filters to block low or high frequencies).

They can also be used for "power factor correction" with inductive loads (like motors) by counteracting the phase shift caused by the inductor.

Important ratings:

Capacitance (Farads): How much charge it can store.

Voltage Rating: The maximum voltage it can safely handle. Exceeding this can destroy it.

Polarity: Some capacitors (like electrolytic ones) must be connected the right way around (+ to +, - to -) or they can explode.

Essentially, the video shows capacitors are versatile components for storing energy, filtering signals, and timing, with their behavior changing based on whether the circuit is DC or AC.

video 15: Temperature Measurement (Part 1) || NTC, PT100, Wheatstone Bridge

This video explains different ways to measure temperature, particularly for electronics and DIY projects, using 3D printing as a key example where precise temperature control is crucial.

It covers:

NTC Thermistors: These sensors change their resistance significantly with temperature (resistance goes down as temperature goes up). They are cheap but non-linear, making precise measurements across a wide range more complex.

PT100 RTDs (Resistance Temperature Detectors): These are more accurate and linear than NTCs (resistance goes up with temperature). However, their resistance change is small, so they require more sensitive and often complex circuitry (like Wheatstone bridges or differential operational amplifiers with precise resistors) to amplify the signal and compensate for offsets.

The video demonstrates that building these measurement circuits from scratch can be challenging due to the need for precision. It then presents easier solutions:

Pre-made PT100 Transmitter Modules: These modules take a PT100 sensor input and convert its reading into a standardized, easy-to-use output (like a 4-20mA current loop, which can then be converted to a voltage). The video shows how to build a thermometer using one of these with a microcontroller (like an Arduino) and an LCD screen.

Integrated Temperature Sensor ICs (like LM35 or DS18B20): These chips handle much of the complexity internally and output a voltage or digital data directly proportional to the temperature, simplifying the design.

Essentially, the video shows the journey from basic, somewhat inaccurate temperature sensing to more precise methods, highlighting the trade-offs between complexity, cost, and accuracy, and ultimately demonstrating a practical way to build a fairly accurate thermometer using a pre-made module.

Video-16

Resistors: Basics and Uses Functions

Resistors play an important role in electronics and are used primarily to restrict current flow and manage voltage. A conventional illustration is an LED; without a suitable resistor, LEDs can be destroyed due to excessive current. Ohm's Law tells us the value of the resistor needed to be kept within the current limits, considering the resistor's power rating so it doesn't burn out.

Resistors serve multiple purposes other than restricting the current:

Voltage dividers lower the amplitude of a signal by splitting signal voltage across two resistors.

Potentiometers (increase/decrease resistors) enjoy manual setting turning such as in volume knobs or for calibrating with sensors.

Resistors used to stabilize the digital inputs to control the erratic behavior of microcontroller (normally 10kΩ) are called pull-up/pull-down.

Resistors with small resistance values measure circuit current by monitoring the voltage drop.

Resistors assuming extreme positions become fusible resistors, losing their integrity to isolate other circuitry parts.

At higher frequencies, parasitic inductance and capacitance cause non-resistive behavior, which is more dependent on frequency. This should be the case with RF and high-speed circuits, where idealness is not an option.

Video-17

Oscillators: The heartbeat of Electronics

Essential circuits known as oscillators produce periodic voltage signals that serve as the timing foundation for measurement instruments, communication systems, and microcontrollers. Modern electronics just wouldn't work reliably without stable oscillations.

Oscillator Types:

Oscillators for relaxation (Astable Multivibrator)

creates square waves using transistors and RC networks.

Capacitors alternately charge and discharge, causing visible LED flashing.

R and C values allow you to adjust the frequency, which is excellent for learning the fundamentals.

IC 555 Timer

A multipurpose, user-friendly chip featuring flip-flops and internal comparators.

charges and discharges a capacitor between the supply voltage of ⅓ and ⅔ to produce steady rectangular waves.

Ideal for novices, but restricted at extremely high frequencies.

Circuits for LC Tanks

At resonance, the energy exchanged by the inductor (L) and capacitor (C) produces sine waves.

naturally deteriorates because of parasitic resistance; a transistor or other amplifier is needed to sustain oscillations.

Also used in RF applications but sensitive to breadboard parasitics.

Video-18

Brushless DC Motors and Electronic Speed Controllers (ESCs)

Brushless DC (BLDC) motors are a development of the traditional brushed motors because of improved efficiency, reliability, and performance. BLDC motors operate on the principle of an ingenious reversal of components: as opposed to coils being on the rotor as with the brushed motors, BLDC motors possess permanent magnets on the rotor while the coils are fixed on the stator. This fundamental conversion eliminates the wear-prone physical commutators and brushes found in traditional motors. Rotation within the motor is rather controlled electronically by an Electronic Speed Controller (ESC), which precisely sequences power to different stator coils to create a rotating magnetic field that pulls magnets along with the rotor.

The ESC is the system's intelligent controller, using sophisticated pulse-width modulation (PWM) and MOSFET switches to time the energizing of each coil to a high degree of accuracy. This electronic commutation provides motor speed and torque control with an accuracy many times that of mechanical commutation. Interestingly, the motor speed is more a function of how quickly the ESC switches between coils (frequency) than of merely the voltage input, though voltage does significantly affect peak possible speed through the KV rating of the motor (RPM per volt).

BLDC motors come in several forms, outrunner forms being of particular interest. In this instance, the external casing itself is rotated by the magnets placed upon it, and additional poles can be included, and hence more torque can be developed at slower speeds - perfect for application in electric skateboards or drones where a lot of high-torque, regulated power is needed. The KV rating scale helps users choose motors to suit their needs, where higher KV motors spin faster but produce less torque at any voltage, and lower KV motors provide more rotational force but slower speeds.

BLDC systems are so wonderful in that they provide ruggedness with capability. Without brush wear, these motors can be run for thousands of hours with minimal upkeep. The ESC's precision power delivery capability also renders these motors more energy efficient than their brushed equivalents. For anyone who deals with next-generation electric drive systems, from hobbyist quadcopter constructors to electric vehicle designers, knowledge about BLDC motors and ESCs is crucial information that bridges the gap between fundamental electrical theory and state-of-the-art motor control technology.

Video-19

The video introduces the I2C (Inter-Integrated Circuit) protocol, a popular two-wire communication method used to connect multiple devices (up to 112 slaves) to a single or multiple masters, such as microcontrollers like Arduino. I2C uses two lines: SDA (data) and SCL (clock), and requires pull-up resistors because of its open-collector design.

GreatScott! demonstrates a practical example by connecting an Arduino to a TEA5767 FM radio IC. He creates a breakout board for the IC, connects the necessary pins (including power, ground, antenna, and audio output), and adds 10kΩ pull-up resistors to the SDA and SCL lines. The video explains the importance of consulting the device's datasheet to understand the required data sequences for controlling the IC, such as setting the frequency.

The Arduino Wire library simplifies I2C communication by handling start conditions, addressing, read/write operations, and acknowledgments. The video shows how to calculate the correct values to send for tuning a specific FM frequency, convert them to hexadecimal, and send them from the Arduino. After uploading the code, GreatScott! successfully tunes the radio and amplifies the audio output.

He also demonstrates how to use an oscilloscope to visualize the I2C signals and discusses how to read data back from the IC. The video concludes by highlighting the flexibility of I2C for controlling multiple devices and encourages viewers to consult datasheets and explore other communication protocols like SPI and One-Wire.

Video-20

The video introduces the thyristor, a semiconductor device that acts as a controllable diode. Unlike a regular diode (which has two layers and two terminals), a thyristor has four layers and an additional gate terminal. When a positive voltage is applied to the gate, the thyristor switches on and stays conductive even after the gate signal is removed, as long as the current stays above a certain holding threshold. The only way to turn it off is to interrupt the current flow.

GreatScott! demonstrates this behavior using LEDs and a small load, showing the importance of reaching the latching current for the thyristor to stay on. He also discusses the turn-off time and the need for a heatsink when controlling larger loads due to power loss across the device.

The video then moves to AC applications, where the thyristor turns off automatically at every zero-crossing of the AC waveform, acting like a half-wave rectifier. To control both halves of the AC cycle, a triac (essentially two thyristors in inverse parallel) is used. GreatScott! builds a practical AC phase angle control circuit using a triac, optocouplers, an Arduino Nano, and a potentiometer. The Arduino detects the AC zero-crossing and, after a user-set delay, triggers the triac to control the power delivered to a load (like a light bulb).

This method is commonly used for dimming lights or controlling heating elements and motor speeds. The video also notes a drawback: such phase control circuits can decrease the power factor due to non-sinusoidal current. The episode ends with encouragement to experiment safely and further explore power electronics.

Video-21

The video introduces operational amplifiers (op-amps), explaining their symbol, packaging, and basic function in analog circuits. GreatScott! focuses on the three golden rules for understanding and designing op-amp circuits:

1. **The output will do whatever is necessary to keep the voltage difference between the inverting and non-inverting inputs at zero (for closed-loop configurations).**
2. **The input draws no current (ideal op-amp assumption).**
3. **Without feedback, the op-amp acts as a comparator, driving the output to its maximum or minimum voltage depending on which input is higher.**

He demonstrates these rules using practical circuits, including non-inverting and inverting amplifiers. For the non-inverting amplifier, he explains how the gain is set by resistor values and shows an example amplifying a sensor signal. He also discusses the limitations of real op-amps, such as limited output swing (not reaching the exact supply voltage), and introduces rail-to-rail op-amps for improved performance.

GreatScott! then covers the inverting amplifier configuration, showing how it can be used to amplify AC signals (like from a microphone) without also amplifying a DC offset, by properly biasing the inputs. He notes that op-amps have limited output current, so they can't directly drive speakers.

Finally, the video explains the comparator mode, where the op-amp, without feedback, switches its output fully high or low depending on the input voltages. This is useful for detecting voltage thresholds. The episode concludes by highlighting the versatility of op-amps for building various analog circuits, such as integrators, differentiators, and voltage followers, encouraging viewers to experiment and learn more.

Video-22

The video explains how to use Bipolar Junction Transistors (BJTs), specifically NPN and PNP types, as electronic switches to control loads in circuits—such as turning on a high-power LED or lamp with a low-power control signal like from an Arduino.

GreatScott! starts by introducing the basic structure and operation of NPN and PNP transistors, highlighting the importance of identifying the correct pinout (emitter, collector, base) using datasheets. He demonstrates a simple switch circuit using an NPN transistor to control a 1W LED, explaining that the transistor allows current to flow from collector to emitter when a sufficient base current is applied. He emphasizes the need for a current-limiting resistor at the base to prevent excessive current and potential damage, and shows how to calculate the appropriate resistor value using the transistor's current gain (β or hFE).

The video discusses the concept of saturation, where the transistor is fully on, but still has a small voltage drop across collector-emitter, affecting efficiency. For switching larger loads, GreatScott! explains that higher current BJTs are needed, but they suffer from more power loss and heat generation, making them less efficient for high-power applications.

He then introduces Darlington transistors, which combine two BJTs in one package to achieve much higher current gain. This allows high-current loads to be switched with very small base currents—suitable for direct control by microcontrollers like Arduino. However, Darlingtons have higher voltage drops and slightly more power loss.

Finally, he demonstrates using a PNP transistor for switching loads connected to ground, explaining the reversed polarity requirements compared to NPN types. The video concludes by summarizing the advantages and limitations of BJTs as switches, and encourages safe experimentation.

Video-23

The video explains how to use MOSFETs (both N-channel and P-channel types) as electronic switches, which are more efficient than BJTs for controlling larger loads. GreatScott! demonstrates that while BJTs can be used for switching, they become inefficient and heat up with high currents due to energy loss across the collector-emitter path. In contrast, MOSFETs can achieve much higher efficiency—up to 97%—with minimal voltage drop and power loss.

He starts by showing how to connect an N-channel MOSFET to an Arduino to switch a load like a high-power LED. The MOSFET is controlled by applying a voltage to its gate (no gate current is required, just voltage above the threshold). He stresses the importance of adding a pulldown resistor (like 10kΩ) between the gate and source to prevent accidental turn-on from static electricity.

GreatScott! then demonstrates using the Arduino's PWM output and a potentiometer to create an LED dimmer circuit. He explains that for loads connected to the positive rail, a P-channel MOSFET is more suitable, but it requires a pull-up resistor and opposite logic (0V turns it on, 5V turns it off).

The video also addresses practical challenges, such as voltage spikes and oscillations when switching large loads quickly. These are caused by the MOSFET's parasitic capacitances and can be reduced by adding a gate resistor (e.g., 470Ω), which slows down the switching and reduces peak currents.

At higher switching frequencies, gate charge losses become significant, so specialized MOSFET driver ICs are recommended for fast, efficient switching. The video concludes by highlighting the advantages of MOSFETs for switching high-current loads in projects like motor drivers and LED controllers, and encourages viewers to experiment safely and consult datasheets for proper MOSFET selection.

Video-24

The video explains how modern hybrid synchronous stepper motors work and why they are widely used in precise positioning applications like 3D printers. GreatScott! disassembles a stepper motor to show its internal structure, which includes a rotor with permanent magnets and toothed pole shoes, and a stator with multiple coils. By energizing these coils in a specific sequence, the rotor moves in precise steps—typically 200 steps per full rotation (1.8° per step).

He demonstrates manually energizing the coils to move the rotor and explains the concept of "holding torque," which keeps the rotor in position even when not moving. The video discusses different driving methods:

- **Wave driving:** Only one coil is energized at a time (lowest torque).
- **Full-step driving:** Two coils are energized, increasing torque.
- **Half-step and microstepping:** Combines coil states to increase resolution and smoothness, allowing up to 1/16 or more steps per rotation.

GreatScott! builds a simple H-bridge driver circuit with MOSFETs and an Arduino to automate the stepping sequence. He then introduces the A4988 stepper driver IC, which simplifies microstepping and current control, making the motor run smoother and quieter. The video demonstrates how to connect and control the A4988 using a 555 timer or Arduino, and shows how microstepping produces smoother motion and less noise.

The video concludes by highlighting the advantages of stepper motors—precise control, holding torque, and smooth motion with microstepping—making them ideal for applications like 3D printers and CNC machines.

Video-25

The video provides a detailed explanation of servo motors, a popular motor-driven positioning system combining motors with control electronics for precise angular control. It begins by describing the physical characteristics of common servos, highlighting the three wires: ground, VCC, and a PWM control signal that determines the shaft angle between -90° and +90° based on signal pulse duration. Internally, servos utilize a DC motor with a gear train to reduce speed and increase torque, and a potentiometer that provides a feedback signal for accurate positioning. The servo's control IC compares the potentiometer's voltage with the PWM input and drives the motor to minimize the difference, achieving the desired shaft angle.

Enhancements like metal gears and stronger motors are available for increased torque and durability, as seen in upgraded servos like the MG996R. The video also explains how servos can be controlled using an Arduino microcontroller via servo libraries generating PWM signals, and how those signals can also be created using simpler electronics like a 555 timer IC without the need for microcontrollers.

Additionally, the video covers an interesting hack to convert a standard servo into a continuous rotation servo, essentially turning it into a variable speed motor. This involves removing mechanical end stops and the feedback potentiometer and replacing the potentiometer with fixed resistors to simulate constant feedback, allowing the motor to spin indefinitely in either direction depending on the PWM signal. The video closes by encouraging viewers to explore servo use in their projects, understanding the underlying mechanisms and control methods to unlock their full potential.

Video-26

The 555 timer IC remains one of the most popular and versatile integrated circuits in electronics, widely used for timing, pulse generation, and oscillator applications. This essay delves into the internal structure of the 555 timer, its working principles, and the fundamental circuit configurations that exploit its functionality: monostable, bistable, and astable multivibrators. Detailed explanations focus on how the internal comparators, flip-flop, resistors, and transistors collaborate to produce stable and adjustable output signals. Furthermore, the limitations of the classic bipolar 555 and the improvements provided by CMOS versions such as the TLC555 are highlighted, particularly in PWM generation and power efficiency. The analysis concludes by showcasing how external components such as resistors, capacitors, diodes, and potentiometers manipulate the IC to tailor timing and frequency characteristics, illustrating the 555's indispensable role in electronic circuit design.

#Pin Functions

- **Pin 1 (Ground)**: Connects to the circuit ground.

- **Pin 2 (Trigger)**: A falling voltage at this pin below one-third of the supply voltage sets the flip-flop, initiating output changes.

- **Pin 3 (Output)**: The pulse output that can drive external loads.

- **Pin 4 (Reset)**: Resets the flip-flop when pulled to ground, overriding other inputs.

- **Pin 5 (Control Voltage)**: Alters the reference voltage levels, allowing modulation of timing intervals. Typically stabilized with a 10 nF capacitor.

- **Pin 6 (Threshold)**: When voltage at this pin exceeds two-thirds of the supply voltage, it resets the flip-flop.

- **Pin 7 (Discharge)**: Connected to the transistor that discharges the timing capacitor to ground.

- **Pin 8 (VCC)**: Powers the IC.

This internal configuration allows the 555 IC to generate timed pulses, stable output levels, or oscillating waveforms depending on the external circuitry.

### Monostable Multivibrator Configuration

In the monostable mode, the 555 timer acts as a one-shot pulse generator, producing a single output pulse of a predetermined duration in response to an external trigger. Initially, the timing capacitor is discharged via pin 7, causing a zero-volt potential at the threshold and discharge pins. When the trigger at pin 2 momentarily falls below the threshold (one-third of supply voltage), it sets the flip-flop, turning the output pin high and disabling the discharge transistor. Consequently, the capacitor begins to charge through an external resistor.

Video-26

The 555 timer IC remains one of the most popular and versatile integrated circuits in electronics, widely used for timing, pulse generation, and oscillator applications. This essay delves into the internal structure of the 555 timer, its working principles, and the fundamental circuit configurations that exploit its functionality: monostable, bistable, and astable multivibrators. Detailed explanations focus on how the internal comparators, flip-flop, resistors, and transistors collaborate to produce stable and adjustable output signals. Furthermore, the limitations of the classic bipolar 555 and the improvements provided by CMOS versions such as the TLC555 are highlighted, particularly in PWM generation and power efficiency. The analysis concludes by showcasing how external components such as resistors, capacitors, diodes, and potentiometers manipulate the IC to tailor timing and frequency characteristics, illustrating the 555's indispensable role in electronic circuit design.

#Pin Functions

- **Pin 1 (Ground)**: Connects to the circuit ground.

- **Pin 2 (Trigger)**: A falling voltage at this pin below one-third of the supply voltage sets the flip-flop, initiating output changes.

- **Pin 3 (Output)**: The pulse output that can drive external loads.

- **Pin 4 (Reset)**: Resets the flip-flop when pulled to ground, overriding other inputs.

- **Pin 5 (Control Voltage)**: Alters the reference voltage levels, allowing modulation of timing intervals. Typically stabilized with a 10 nF capacitor.

- **Pin 6 (Threshold)**: When voltage at this pin exceeds two-thirds of the supply voltage, it resets the flip-flop.

- **Pin 7 (Discharge)**: Connected to the transistor that discharges the timing capacitor to ground.

- **Pin 8 (VCC)**: Powers the IC.

This internal configuration allows the 555 IC to generate timed pulses, stable output levels, or oscillating waveforms depending on the external circuitry.

### Monostable Multivibrator Configuration

In the monostable mode, the 555 timer acts as a one-shot pulse generator, producing a single output pulse of a predetermined duration in response to an external trigger. Initially, the timing capacitor is discharged via pin 7, causing a zero-volt potential at the threshold and discharge pins. When the trigger at pin 2 momentarily falls below the threshold (one-third of supply voltage), it sets the flip-flop, turning the output pin high and disabling the discharge transistor. Consequently, the capacitor begins to charge through an external resistor.

### Bistable Multivibrator Configuration

The bistable mode enables the 555 timer to function as a basic flip-flop with two stable output states: high and low. This mode requires no timing capacitor, relying solely on the trigger and reset pins to toggle the output. Applying a ground potential to pin 2 sets the flip-flop, turning the output high, while grounding pin 4 resets it to low. Due to this arrangement, the output holds its state indefinitely until explicitly changed by applying a trigger or reset pulse. Bistable operation is ideal for simple on/off control tasks, such as manually toggling an LED with push buttons.

### Astable Multivibrator Configuration

Operating the 555 timer in astable mode results in a continuous oscillating output — essentially, a square waveform whose frequency and duty cycle depend on the values of resistors and capacitors connected externally. The timing capacitor charges through two resistors during the ON period and discharges through one resistor during the OFF period.

### Limitations of the Bipolar 555 and CMOS Alternatives

A critical limitation of the standard bipolar 555 timer arises due to the nature of its output stage. It employs bipolar transistors, which cannot drive the output to the full supply voltage; typically, the high output voltage reaches about 4.6 volts on a 5-volt supply line, creating a voltage offset that leads to unsymmetrical charge and discharge cycles. This prevents generating precise 50% duty cycle square waves and limits performance at higher frequencies.

Video-27

This video explores the fundamentals of Analog-to-Digital Converters (ADCs), focusing on their functionality, specifications, and practical implementation, particularly with Arduino microcontrollers. The discussion begins by demonstrating how to use Arduino's analog input pins to convert an analog voltage into a 10-bit digital signal, highlighting the importance of understanding how the ADC interprets voltage levels into digital values. The video then explains crucial ADC specifications, such as sampling rate and resolution, through practical examples like sampling a 10 kHz sine wave. It delves into the Nyquist-Shannon sampling theorem to justify using higher sampling rates and demonstrates how Arduino's default ADC sampling rate can be adjusted to improve signal reconstruction.

The core working principle of the successive approximation ADC is thoroughly explained, using a simplified 4-bit example to show the stepwise comparison and conversion of an analog voltage into a digital code via components like the sample-and-hold circuit, comparator, DAC (Digital-to-Analog Converter), and SAR (Successive Approximation Register). The impact of resolution on voltage measurement accuracy is emphasized, illustrating how a higher bit count reduces voltage step size, enhancing ADC precision.

Additionally, the video presents a practical approach for using a dedicated 12-bit ADC IC (ADS7816) with an Arduino, outlining the importance of reviewing datasheets and following recommended connection and programming protocols. Finally, it introduces the flash ADC—a type of ADC known for speed but with trade-offs in resolution and complexity, making it more suitable for lower-bit implementations or specialized applications. This DIY-friendly flash ADC example demonstrates how multiple comparators and a resistor network can quickly output a binary value. The video closes with encouragement to experiment with ADCs while reiterating key ADC concepts.

Video-28

The video provides an in-depth comparison and explanation of Insulated Gate Bipolar Transistors (IGBTs) and Metal-Oxide-Semiconductor Field-Effect Transistors (MOSFETs), particularly in the context of powering solid-state Tesla coils. The presenter starts by outlining the typical inverter topologies used for Tesla coils, such as half-bridge and full-bridge circuits, and discusses the practical use of n-channel MOSFETs and IGBTs in these setups. The focus then shifts to the fundamental differences between MOSFETs and IGBTs, breaking down their internal structures, operating principles, and electrical behaviors.

An IGBT is explained as a hybrid device combining an n-channel MOSFET and a bipolar junction transistor (BJT), offering high-voltage and high-current capabilities with some trade-offs in switching speed and power loss compared to MOSFETs. The video uses practical circuits, such as a simple light bulb switch, to illustrate how IGBTs function, how the gate behaves like a capacitor, and why a pull-down resistor or dedicated driver ICs are essential for proper operation, especially at higher frequencies.

Data comparisons are shown between the switching times and power losses of MOSFETs and IGBTs, demonstrating that MOSFETs have faster switching capabilities and lower conduction losses at low currents, while IGBTs perform better in handling high voltage and current applications albeit with greater switching losses. The recommendation from this analysis is that MOSFETs are ideal for applications operating above 200 kHz, whereas IGBTs are better suited for frequencies below this threshold.

Finally, the video concludes that IGBTs, thanks to their high breakdown voltage and current capacity, are practical for medium-speed, high-power applications like solid-state Tesla coils, while MOSFETs excel in scenarios requiring fast switching and lower conduction losses. Viewers are encouraged to refer to additional foundational videos on MOSFETs and BJTs for a deeper understanding of these components.

Video-29

The video provides an in-depth explanation of solar panels, focusing on how they generate power, their internal wiring, and how to maximize power output for practical applications such as charging batteries. It begins by describing the basic composition of solar panels, which are made up of multiple solar cells connected in series to increase voltage output. The presenter explains that each individual solar cell produces about 0.5 volts when illuminated, but since a single cell's voltage is low, connecting many in series boosts the overall voltage to usable levels (e.g., a 100 Watt panel with 36 cells produces about 14.3 volts open circuit voltage).

The video also discusses the drawbacks of series connections, particularly how shading on even a part of the solar panel can drastically reduce overall power output due to increased resistance in the affected cells. To mitigate this, bypass diodes are introduced inside the panel's junction box, allowing current to flow around shaded cells to maintain power output. Additionally, blocking diodes prevent reverse current flow when panels are connected in parallel, protecting the system from damage.

The presenter emphasizes that the rated power (e.g., 100 watts) is measured under Standard Test Conditions (STC), which are idealized conditions rarely met in real life, such as perfect sunlight intensity and cell temperature. Actual power output is often less, depending on environmental factors and the type of electrical load connected.

Using a practical example with small solar panels and LEDs, the video shows how output voltage and power can vary substantially depending on the load. The equivalent circuit of a solar cell is explained, describing components such as the constant current source, diode, parallel resistance (losses from defects), and series resistance (losses from wiring and contacts). This model helps understand why different loads affect voltage and current, and how to find the Maximum Power Point (MPP)—the best voltage and current combination that yields the highest power output.

To identify the MPP, the presenter uses a DIY power logger with variable load resistances to measure voltage and current, plotting the data in Excel to visualize open circuit voltage, short circuit current, and the MPP. The demonstration confirms that MPP lies at a voltage and current below the open circuit voltage and short circuit current.

The most efficient way to extract maximum power is using Maximum Power Point Tracking (MPPT) charge controllers, which use switching converts to mimic the ideal load for the solar panel and optimize battery charging. In contrast, simpler PWM controllers do not track MPP and can reduce efficiency by up to 40%.

Video-30

This video provides an in-depth exploration of timers in microcontrollers, specifically focusing on the Atmega328P used in Arduino Uno boards. It addresses the necessity of precisely timed events for complex microcontroller projects like an alarm clock, LED matrix control, and PWM (Pulse Width Modulation) signal generation. The tutorial begins by illustrating the shortcomings of the simple delay() function, such as delayed response to inputs and timing drift over long periods. It then introduces the fundamental concept of hardware timers as autonomous peripherals that run independently from the main program loop, thereby solving these timing issues.

The video dives into configuring Timer1, a 16-bit timer, in its normal counting mode, demonstrating how the timer increments with each clock cycle and overflows after reaching its maximum count of 65,535. Overflow interrupts are explained as a mechanism for triggering actions at precise time intervals without blocking the microcontroller's main loop. To extend the timing intervals, prescalers are introduced, which effectively slow down the timer's counting speed, enabling longer wait times between overflows.

Further, the tutorial discusses how to establish precise time intervals by initializing the timer's count register to a specific start value. For more complex requirements, like multiple interrupts at different intervals, the video demonstrates the use of CTC (Clear Timer on Compare Match) mode, which compares the timer value against predefined thresholds stored in compare registers and triggers interrupts upon matches.

Moving beyond simple timing, the video explains how to use Timer1 to generate PWM signals. By setting specific bits, the timer can toggle output pins at rates and duty cycles determined by the compare registers, allowing flexible control of outputs such as LED brightness or motor speeds. The fast PWM mode is highlighted, showing how it can reach frequencies up to 62.5 kHz and even extend to 8 MHz by adjusting the top value dynamically through input capture registers (ICR1). The use of potentiometers and the Arduino map() function is shown to manually vary duty cycle and frequency in real time.

In conclusion, the video equips viewers with a foundational understanding of microcontroller timers, their configuration, and practical applications, encouraging further study of the extensive timer sections in datasheets for deeper knowledge