Lab 4- Introduction to Electrical Design

A solid grasp of electrical design is the glue that makes embedded-system ideas real. When you can read a schematic, those squiggly lines and cryptic symbols turn into a clear map of how processors, sensors and power rails actually talk to one another—unlocking faster debugging and safer builds. And when you can sketch even a minimalist schematic of your own, you communicate intent unambiguously: teammates, instructors and future-you can reproduce, critique and iterate on the circuit without guessing what jumper went where. In project reports, those diagrams document design choices as rigorously as code comments do for firmware, making your work transferable and maintainable. In short, schematic literacy—and the habit of documenting hardware with the same discipline you apply to software—elevates a prototype from a one-off tangle of wires to a professional, sharable embedded solution.

SparkFun's **"How to Read a Schematic"** tutorial walks beginners through every core symbol and convention—nets, reference designators, power rails and bus labels—so that by the last page you can trace (or sketch) an embedded-system circuit with real confidence. The step-by-step format, peppered with clear drawings and plain-language tips, makes it an ideal first stop before you dive into board-level debugging or PCB design work. learn.sparkfun.com

What makes this guide doubly valuable is the culture that produced it. SparkFun has, since 2003, built its business on **open-source hardware**: none of its boards or reference designs are patented, encouraging users to study, remix and share them. That ethos extends to education: SparkFun's in-house Department of Education curates hundreds of freely accessible tutorials, videos and GitHub libraries, explicitly "documenting everything" so students and professors can integrate the material into their courses. Their philosophy is simple—open documentation lowers barriers, accelerates innovation and makes the road from idea to finished embedded project shorter for everyone.

1. How to Read a Schematic. (By sparkfun)

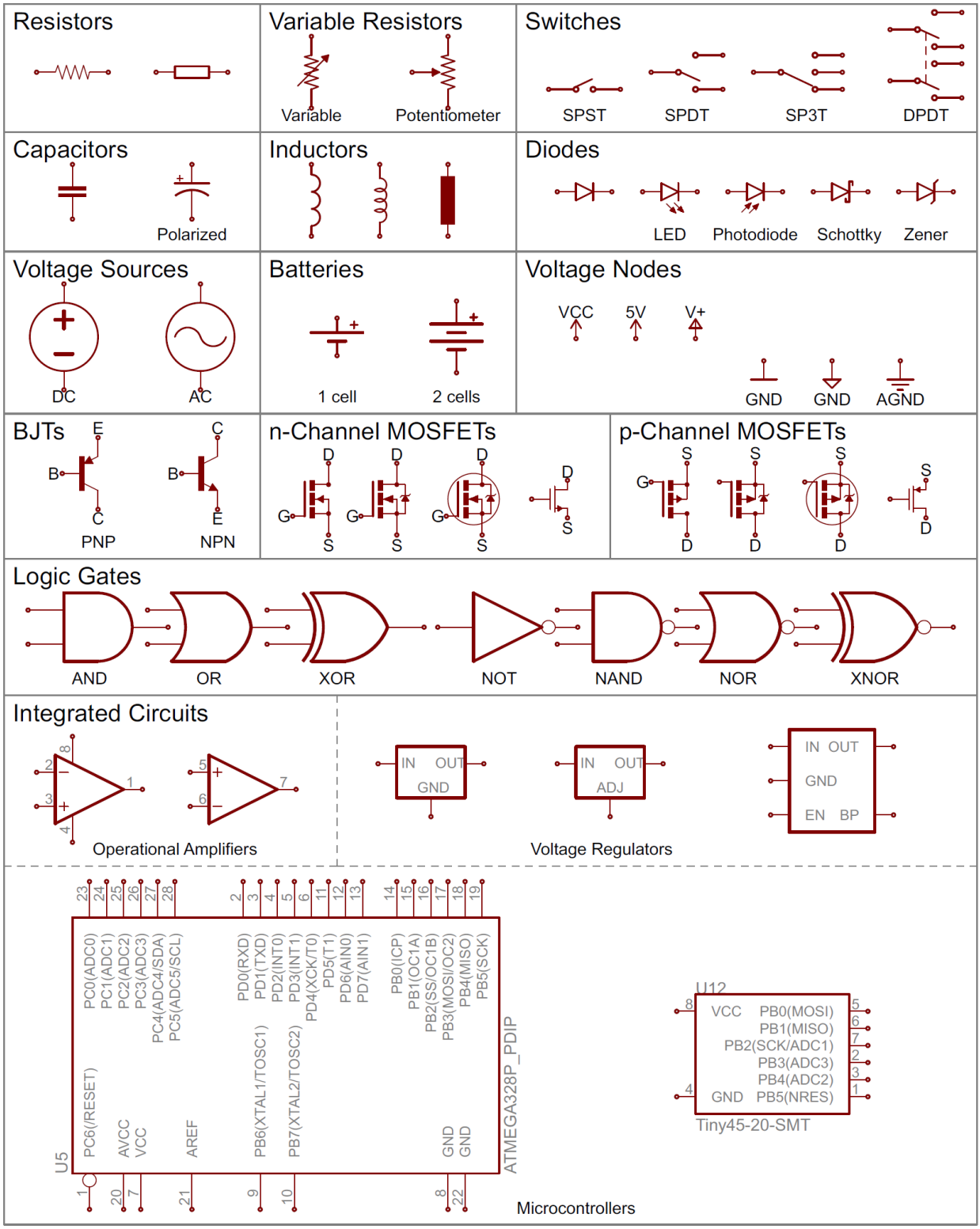

Figure 1: Fundamental symbols: https://learn.sparkfun.com/tutorials/how-to-read-a-schematic/all

2. Embedded-Systems Fundamentals & Protocols

Binary logic sits at the very heart of every embedded system: those 1 s and 0 s define the voltage "highs" and "lows" that your microcontroller ultimately turns into sensor readings, PWM-driven motors, or packets on an I²C bus. The SparkFun *Basics* and *Protocols* series below layers that fundamental idea upward—from raw logic levels through analog/digital conversion and on to the serial links that tie whole boards together.

Category	SparkFun tutorial	What you'll learn (key take-aways)
Basics	Binary	Bits, nibbles, bytes; converting between decimal ↔ binary; why two's-complement and bit-wise operations matter in firmware.
	Logic Levels	Voltage thresholds for TTL vs CMOS, 5 V vs 3 V3 systems, and level-shifting strategies.
	Analog-to-Digital Conversion	How ADCs sample voltages, resolution & reference choices, plus Arduino code examples.
	Pulse-Width Modulation	Duty-cycle control for LED dimming, motor speed, and pseudo-analog audio signals.
	Analog vs Digital	Side-by-side comparison of continuous vs discrete signals and when to use each.
Protocols	I²C	Two-wire addressable bus, pull-ups, multi-master setups, speed grades to 1 MHz+.
	SPI	Four-wire synchronous bus, chip-select strategy, full-duplex transfers at tens of MHz.
	Serial Communication	UART framing, baud rates, TTL vs RS-232 levels, optional flow-control signals.

Keep this cheat-sheet handy: spotting labels like **MOSI**, **SDA**, or a 0 b prefix in code becomes second nature once you've skimmed each primer—making schematics clearer and firmware bring-up far smoother.

3. Designing Your Own Circuits

Drafting a schematic is the first real act of engineering: it turns loose ideas and napkin-sketch pinouts into an unambiguous document that teammates, instructors, and future-you can all read the same way. By committing each power rail, microcontroller pin, and sensor bus to a proper diagram, you catch wiring errors before they happen, speed up code bring-up, and create a portable reference that slots neatly into reports or GitHub repos. Mastering this step also unlocks the next big perk on your hardware journey: once the schematic is sound, you're only a board-layout session away from producing a custom PCB—transforming a prototype tangle of jumpers into a professional, plug-and-play embedded solution.

Below is a quick-reference roster of the EDA tools you're most likely to meet, grouped by licence model. Follow the links to the vendor sites for downloads or details.

Software	Licence / Cost Tier	Highlights	Official Site
KiCad	**Open-source (GPL v3), free**	Full-featured schematic, PCB, 3-D viewer; cross-platform.	kicad.org
LibrePCB	**Open-source (GPL v3), free**	Modern UI, built-in library manager, no online lock-in.	librepcb.org
gEDA / Lepton EDA	**Open-source (GPL), free**	Classic Unix tool-chain (gschem, PCB, ngspice, etc.).	geda-project.org
Fritzing	**Open-source; pay-what-you-want download (≥ €8)**	Breadboard + schematic + PCB views; great for teaching.	fritzing.org
EasyEDA (STD / Pro)	**Free cloud tier, Pro tier free for education / paid for enterprise**	Browser-based capture, PCB, spice; integrates JLCPCB fab & parts.	easyeda.com
Upverter	**Free, cloud-based**	Drag-and-drop modular design, real-time collaboration.	upverter.com
DesignSpark PCB	**Freeware**	Unlimited board size/layers, backed by RS Components.	designspark.com/pcb-software

Software	Licence / Cost Tier	Highlights	Official Site
Altium CircuitMaker	**Free (community/hobby use)**	Altium engine with community project sharing.	altium.com/circuitmaker
Autodesk Fusion 360 w/ EAGLE	**Free personal / education; paid subscription for commercial**	Integrated MCAD-ECAD workflow; hobby tier capped at 2 layers/80 cm^2.	autodesk.com/fusion-personal
Altium Designer	**Commercial (subscription); *free student licence***	Industry heavyweight; Altium 365 cloud collaboration.	altium.com
Cadence OrCAD X	**Commercial; *free academic suite* + 30-day trial**	Capture, PSpice, PCB Designer; same tools industry uses.	cadence.com/orcad-academic
Siemens PADS Professional	**Commercial; *free 1-year student edition***	Xpedition-based flow, cloud collaboration, supply-chain data.	eda.sw.siemens.com/pads
DipTrace	**Tiered licences (Lite → Full, $75 – $ 995); time-limited demo**	Clean UI, strong 3-D, translators from EAGLE/Altium.	diptrace.com
Proteus Design Suite	**Commercial; ~20 % academic discount**	Combines schematic, PCB and real-time MCU simulation.	labcenter.com

Lab tip – two quick-start options

- **EasyEDA (browser-based)** – No install needed; just sign-in and start drawing in any modern browser. SparkFun parts libraries are already hosted, and you can export Gerbers straight to JLCPCB.
  Getting-started link: EasyEDA's official "Introduction" and quick tutorials https://docs.easyeda.com/en/Introduction/Introduction-to-EasyEDA/
- **KiCad (pre-installed in the lab)** – A fully open-source toolchain ready on the lab PCs (and free for your own laptop). Use it when you want local files, unlimited board size, or tight Git integration.
  Getting-started link: KiCad's "Getting Started in KiCad" guide docs.kicad.org

For the Lab report:

1. Check-Your-Understanding Questions

a. **Schematic Goals** In one sentence each, list the two main reasons we insist every project include a formal schematic before any breadboarding begins.

b. **Symbol Recognition** The symbols  and  appear throughout a sample schematic. State what each represents and give one practical design rule that applies to that net or bus.

c. **Tool Selection** You have a Chromebook at home and a Windows workstation in the lab.
a. Which of the two recommended tools—EasyEDA or KiCad—would you choose for off-campus work, and why?
b. Name one feature of the *other* tool that might convince you to switch for a large team project.

d. **Level-Shifting Challenge** A 5 V microcontroller must talk SPI to a 3.3 V accelerometer. Sketch or describe a safe interface and list two drawbacks of using simple resistor dividers on the SPI lines at 8 MHz.

e. **From Schematic to PCB** Outline the four high-level steps that convert a finished schematic into a manufactured PCB, and name the file set delivered to the fab house at the end of the process.

f. **Documentation Best-Practices** Give two concrete examples of information that should *not* be embedded only in the PCB layout but should also be called out directly on the schematic.

g. **Error Hunting** During ERC/DRC you receive a "power-pin no driver" warning on the VDD net. Explain what this means and describe the quickest way to investigate and resolve it inside your chosen EDA tool.

h. **Future-Proofing** Suppose your schematic includes a proprietary sensor module that often goes out of stock. Describe one design decision you can take *while drawing the schematic* to make future component substitution easier.

2. Post-Lab Reflection Tasks

a. Experience Summary (≤ 150 words)
Briefly describe what went smoothly, what surprised you, and one skill you still need to strengthen when translating an idea into a formal schematic.

b. Home-Experiment Schematic
Choose **one** setup from the Arduino Sidekick kit (e.g., potentiometer + LED dimmer, mini-servo sweep, or push-button counter) and create a clean schematic in your preferred EDA tool. Export as PDF or PNG and include it in the report. Highlight (with callouts) at least one power rail, one ground symbol, and any communication lines used.

c. Tool Reflection & Screenshot
Provide a single annotated screenshot of your schematic editor. Label two features that sped up your work (e.g., ERC checks, symbol search, net labels) and explain why they mattered in ≤ 80 words.

d. Error-Catch Log
List up to **three** schematic-capture warnings or errors you encountered (e.g., "power-pin no driver" on VDD). For each, note what caused it and how you fixed it.

e. **Level-Shift Justification**
   If your circuit mixes 5 V and 3.3 V logic, attach a snippet (or sketch) of your chosen level-shifting method and justify the choice in ≤ 60 words.
f. **Next-Step PCB Sketch (optional for bonus)**
   Convert the schematics to a top-level placement sketch showing where you would place each component on a 5 cm × 5 cm board. One paragraph (≤ 100 words) on your placement rationale earns up to 5 bonus points.