

# Intro to Hardware Hacking

## Day 1

Rowan Hart

October 2, 2020

# General Announcements & Introduction

- ▶ Sign up for the bootcamp CTF at <https://play.ctf.b01lers.com>
- ▶ I will answer questions from Twitch, Youtube, and Discord.
- ▶ TODAY WILL BE SHORT!

# Outline

Hardware hacking brings together a lot of different skills from across the CTF space. We'll talk about the basic steps. We'll *also* assume that any device we're looking at is your device so I'm going to skip physical access.

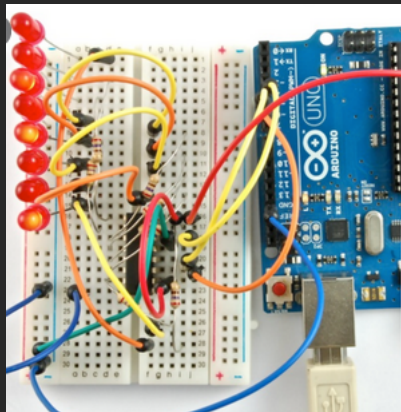
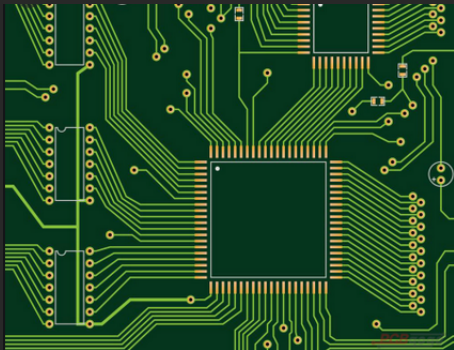
- ▶ Board components and ICs
- ▶ External interfaces
- ▶ Intercepting communications
- ▶ Obtaining Firmware
- ▶ The Hack

# Board Components

We're going to discuss PCBs mostly but all of the following components also apply to handcrafted Arduino/Raspberry Pi/Etc. Circuits.

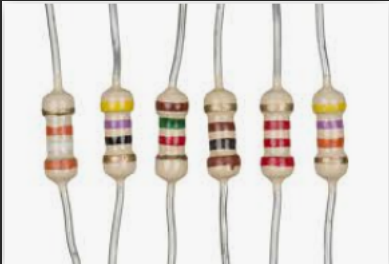
# Traces/Wires

Traces and wires do the same thing: conduct electricity from one place to another.



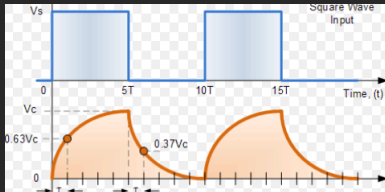
# Resistors

Resistors do what it says on the tin: resist the flow of electricity by converting electrical potential into dissipated heat. The equation  $V = I \cdot R$  (Voltage = Current times Resistance) helps us calculate voltage/current drop.



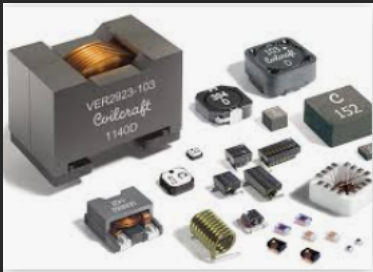
# Capacitors

Capacitors store electrical energy and then release it. Some capacitors release all the energy at once and some release it over a period of time. The types produce different waveforms (see below) and are used to denoise circuits as well as other applications.



# Inductors and Transformers

Inductors store energy in a magnetic field. They are often used to filter RF interference (a common task in CTF). Transformers step voltage up or down across a two-inductor circuit and can be used to isolate signals.

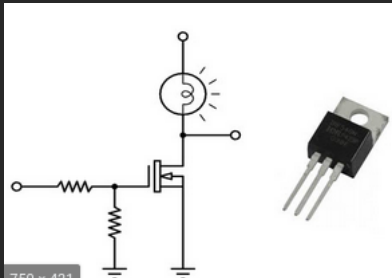




# Transistors

There are a few types of transistors but nowadays 99% of what you'll see are MOSFET transistors.

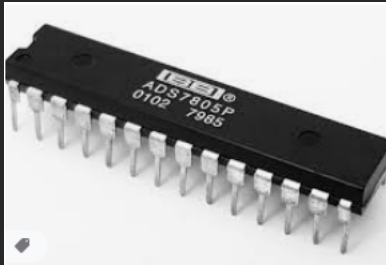
Think of transistors as a switch but instead of someone pressing a button a voltage being applied turns it on or off.



# Integrated Circuits

Integrated Circuits (ICs) handle logic. Think of them somewhat like functions in C. You provide a few wires with an input value and they provide an output. ICs can include timers, registers, XOR/NAND/OR/AND/NOT gates and much more.

Mostly in CTF you'll see them looking like below, which is called a DIP (Dual in-line package).

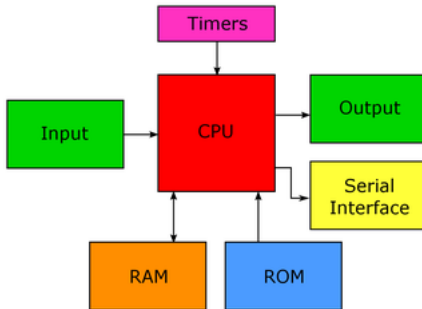


# Microprocessors and Microcontrollers

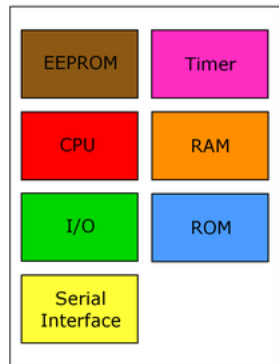
Processors are a subclass of ICs that can handle general purpose programs. For example, the ATMEGA 2560 I'll be demonstrating with later is an IC that is also a Microcontroller.

The difference between a Microprocessor and microcontroller is easier to draw:

Microprocessor: CPU  
and several supporting chips.

















































Microcontroller: CPU  
on a single chip.



# Circuit Diagrams

Circuit diagrams look pretty nasty. Here are some commonly used schematic components:

	ground		Attenuator		Mosfet		Tube Diode
	Chassis Ground		Capacitor		PNP Bipolar Transistor		Tube Triode
	Battery		Capacitor		NPN Bipolar Transistor		Tube Tetrode
	Resistor		Antenna		P-Channel Junction		Tube Pentode
	Potentiometer		Crystal		N-Channel Junction		Turn-off Diode
	Trimmer		Circuit Breaker		PNP Darlington		SPST
	Fuse		Jack		NPN Darlington		SPDT
	Transducer		Plug		Diode		DPST
	Transducer		Plug		Tunnel Diode		4-Way
	Indicator		Shielded Jack/plug		Photodiode/LED		Make Contact
	Bell				Photosensitive Diode		Break Contact
	Buzzer				Inverter		Pushbutton Make

# IC identification

Google is your friend. Luckily because of regulation and standardization it's pretty rare to come across an IC that doesn't have some sort of documentation available. Simply search the number on the IC and often you'll get a complete set of docs:

# Tools

Unlike traditional CTF, hardware hacking usually requires physical tools.

- ▶ Physical access tools:
  - ▶ Screwdrivers
  - ▶ Pliers
  - ▶ Heat gun
  - ▶ Soldering Iron
- ▶ Analysis tools:
  - ▶ Oscilloscope
  - ▶ JTAGulator / Bus Pirate / J-Link
  - ▶ Logic Analyzer / J-Trace
- ▶ Reverse engineering tools:
  - ▶ IDA Pro (first for hardware for reasons I'll explain).
  - ▶ Ghidra
  - ▶ You may need a custom tool.

# External Interfaces

Hardware systems provide a lot of places we can attach a wire and read a signal. All we have to do is determine which one we want. Common patterns we want to look out for are:

1. Serial ports
2. USB ports
3. JTAG/Debugging interface
4. I2C (Inter-Integrated Circuit) interface
5. Ethernet

After deducing the pinout, we can connect and move on to the next step.

# Firmware Recovery

Typically the first step to hacking a piece of hardware is to get the firmware. There are lots of ways to do that:

- ▶ Use debugging interface to step through the firmware and dump contents.
- ▶ Directly obtain from a flash chip.
- ▶ Many FPGAs read code from a memory chip. Read the bytestream!
- ▶ Google, wireshark. Did you know Logitech TV remotes run a web server?
- ▶ Caveat: some firmwares are encrypted (anything that someone wants to keep secure).



# Exploitation

We're not talking about hardware hacking in terms of adding a chip to your toaster to play aladdin. We want to either:

- ▶ Steal something from the hardware we aren't supposed to have:
  - ▶ Stored data
  - ▶ Encryption keys
  - ▶ Proprietary code
- ▶ Put something on the hardware that isn't supposed to be there:
  - ▶ Information collection
  - ▶ Interference

In the second case, we almost certainly need to either exploit the running firmware (best) or modify the firmware (okay for some objectives but not others). In the first case, we can just pull it off the hardware or via one of the recovery methods above, but sometimes we do need an exploit to do that too.

# The good news

The good news about exploiting embedded systems:

- ▶ Usually they have tons of bugs
- ▶ Harder to patch than a typical system
- ▶ Low level programming means less steps to gain arbitrary control once vulnerability is acquired.

The bad news:

- ▶ Embedded systems usually aren't just out on the internet (see Jack's presentation, specifically Shodan, for counterexamples).
- ▶ Often use nonstandard interfaces and protocols to talk to each other.

# The Side Channel

Side channel attacks are super popular now due to their severity and difficulty to patch (Spectre and Meltdown).

In CTF, side channel attacks are often against cryptosystems (see Evil Accountant <https://github.com/b01lers/b01lers-library>).

Caveat: side channel attacks often (but not always) require already being on the machine in some way. (Spectre/Meltdown you can be in another container, etc)

# Air-gap Jumping

To get control of these hard-to-reach targets, we have to do something called gap jumping (was going to be my demo but gnuradio is mean on my OS).

Air-gap jumping is any way you can get access to an embedded remote system via unconventional means (RF, environmental sensors, anything that will allow you to deliver your payload).

# Demo + Questions

???