Introduction

This workshop is intended to introduce you to the concepts behind "Reverse Engineering". Reverse Engineering is basically figuring out how something works and what it's made of.

Reverse Engineering can apply to complete products or systems, or only to parts of these products. I can involve figuring out how software works, or how physical interfces between components work.

This tutorial is designed to introduce you to the practice of examining physical interfaces - connections between different parts of a device, or between a device and the outside world. This type of work typically requires at least some basic knowledge of electronics and interface protocols, and examining and using the information exchanged over those interfaces.

We will be using the Revenge board set. This board set is designed to present a functional 'product' which is not too complex, and allows easy access to all electrical signals for probing.

Why Reverse Engineer?

Reverse engineering is undertaken for many reasons. Some of these are:

Competitive Analysis

Competitive analysis is done by companies to examine their competitor's products, and by nations to determine their enemies' capabilities.

If you can figure out exactly how much it costs your competition to manufacture something, you can determine which competitive pricing balances your gain against their losses. You may gain insight into possible future plans, enhancements, or expansions.

You might identify certain operations which are not as efficient as the ones your product performs, and use this to craft benchmarks showing your product to be superior. Or you can provide your sales force with knowledge about the deficiencies of the competition's offerings.

Compatible Product Design

Reverse Engineering is also done in order to be able to market products which operate with or connect to other products.

Say for example you wanted to sell an alarm clock with a docking port for an iPhone or Android device, and you wanted the device alarm to trigger the alarm clock. You'd need to understand which interfaces are available on the device, and how they work.

Product Enhancement

It's also done to add new functionality to products or extend their usefulness, even when the original product may be long out of production. An example of this is the work that I did figuring out how to connect with Brother knitting machines.

This has enabled many new uses for these machines, connected directly to computers.

More Information About Reverse Engineering

- **Wikipedia: Reverse Engineering**

Legality of Reverse Engineering

There are some very firm boundaries and many very soft ones when it comes to reverse engineering products. The following view is largely U.S.-centric.

I am not a lawyer, and an umfamiliar with most international law. Nothing I say should be taken as legal advice. Laws governing use and reverse enginnering of products have been proliferating in recent years, and it's difficult to keep up.

Resources:

- The Electronic Frontier Foundation
- Chilling Effects FAQ
- Yale Law Journal

Example: Technically Protected Digital Content

Provo Craft is the maker of the "Cricut" line of paper and vinyl cutters. They also sell cartridges containing design libraries, software which communicates with their products, and hardware accessories which plug into the cutters.

Cricut is a closed, proprietary system, and Provo Craft has filed lawsuits to force makers of compatible equipment to cease allowing compatibility with their products.

The Cricut cutters are similar in many ways to knitting machines, and could support a large ecosystem of independent developers of applications and patterns, but this is not the model that Provo Craft has chosen to support. Their business model is to own the entire supply chain, including content and tools for editing or creating any user content.

Some people have reverse engineered the interface to Cricut machines, but they are unable to publish software using this interface because the data exchange with the cutter is encrypted. The Digital Millenium Copyright Act (DMCA) makes it illegal to break the encryption on a device (even if you own it and even if the encryption is trivial) in order to gain access to the underlying functionality of that device.

Unfortunately, the DMCA makes it very easy for companies to shut down anyone who they claim infringes on their

products.Courts and juries generally have a poor understanding of the technological issues involved, and the average hobbyist cannot afford a defense against any court action. It's becoming common for companies to make legal threats, whether or not the alleged infringer has actually violated the law. It is so expensive to defend against these threats that most hobbyists and small businesses simply concede.

If you feel that you are being targeted for action even though you are not infringing any laws, there are resources available to help you. However, these organizations generally operate on a shoestring, so don't expect them to be able to fight a team of corporate lawyers on your behalf.

About the approach of this book and the target audience

This material is written to be used with a set of Revenge (reverse engineering, get it?) boards, and takes a hardware oriented approach from the beginning. This is because a lot of the reverse engineering that people find useful is done to allow consumer devices to be used for extended purposes not supported by the original producers of that equipment, and generally this requires an approach of focusing on the interfaces (more on this later).

Let's Begin

We're going to review some basic knowledge.

- Voltage and current
- Standard Logic Voltage Levels (TTL, CMOS, 3V3, 5V, 12V)
- Endianness
- Embedded Magazine: Serial Protocols
- Wikipedia: Serial Protocols
- Parallel Busses
-
-

Reverse engineering by exmining functionality

If you want to know how much it costs to make something, figuring it out is almost as simple as taking it apart and adding up the costs of all the parts. There's more, of course. You need to figure out the manufacturing costs, and costs of custom components (Think of the metal case for an iPhoen - how would you figure out how much Apple pays for each of those?).

But if you want to know how to connect to something, or use it in ways it wasn't designed for, then you need to know how it works, both at a physical level and at a higher level of protocols and interfaces. We're going to focus on figuring out how something works. This is often tedious work, but it can be a lot of fun and personally rewarding. It's also a really good way to learn a lot about electronics, system design, and product design.

In order to figure out 'how something works', you have to begin by figuring out where you can insert yourself into the operation of the system to begin gaining knowledge. In most cases, this is done by identifying a standardized physical interface in the product, and looking at what's going on. In many cases, the interfaces between various components of a system use one of the serial buses we discussed earlier.

If you can open up the device, then learn as much as you can about these connections by observation. See if you can determine the type(s) of microcontroller used. Look up the data sheets. You don't need to understand everything about the controller, but you can guess a lot by looking.

If the interface uses two wires and they are connected to the 'RX' and 'TX' pins, then it's probably a simple asynchronous serial connection. If it has two wires and they're connected to pins the data sheet labels 'SDA' and 'SCL', then the interface is probably I2C. If there are threefour wires, it may be SPI, etc.

Observing the signals with an oscilloscope, logic analyzer, or bus pirate can help identify these protocols, and even decode them into readable numerical form

Synchronous protocols like SPI and I2C have clock information as one of the signals, which means that decoding of the signal requires no knowledge about how long each 'bit' is held on the signal lines. Asynchronous signals require

some knowledge of the bit rate or baud rate used

Asynchronous serial signals may be very easily monitored using a standard transceiver interface, such as the Adafruit FTDI Friend. This allows monitoring and decosing the signal as it is in use. However, you will not get valid decoding unless you have selected the correct bit/baud rate.

Examining the Revenge Board

The revenge board set is two boards - One board has a joystick and accelerometer (tilt sensor). The other board has LED displays. Sensor data is collected on one board and displayed on the other. Each board has an arduino controller, which allows access to all signals used on the board.

Source code for the revenge boards hardware ans software can be found here: Revenge Git Repo