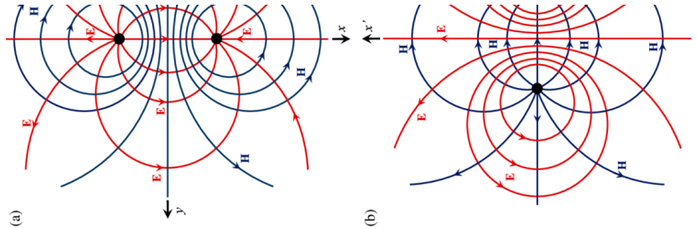


Discovery-Oriented Design using Theory and Prototyping

The Wheelhouse

It’s possible to specialize in doing things nobody knows how to do. It’s like being a mouse trained to solve unfamiliar mazes brilliantly. Some people come by this ability through experience but no program teaches it. I am formalizing it here under the name Discovery-Oriented Design.

This intensive curriculum will supercharge designers, architects, and artists with the ability to work effectively in unfamiliar or unexplored territories.

Participants will learn the hard stuff — the equations, algorithms, circuits, and systems — that are all around us but unseen until we understand them. They will apply these new ways of seeing and thinking so constantly that it becomes their comfortable native territory. Participants will also become adept at quickly designing, fabricating, and testing prototypes based on these concepts. This combination of theory and prototyping is a powerful practice for discovering and creating the very new.

Each full day will include lectures, discussions, and hours of hands-on prototyping of projects relevant to the current theory subjects.

The Problem

This curriculum is the opposite of the many programs in hacking, making, or design & technology here in New York City.

These hacking programs teach ways to use products like laser cutters and Arduinos creatively. But students typically graduate understanding little about math, physics, circuit design, computation, or systems. They may learn to use a 3D printer but not a chisel, MIG welder, or CNC mill. They can copy a schematic diagram to a breadboard but don’t understand analog circuit design.

Twenty years ago, there was a revolution at the intersection of the creative and the technical. A creative person with a little knowledge of code and circuits was a wizard. There was abundant low-hanging fruit and many rewards. But the world around that intersection has changed and the low-hanging fruit is now mostly gone. These programs in hacking and making may be preparing students for a wave that has already crested.

The horizon of exciting possibilities is broader than ever but it takes more work to get there. I propose Discovery-Oriented Design as one effective path.

Theory and Adjacency

Fundamental theoretical knowledge is more valuable than provisional job-related knowledge like how to use a specific program or tool. Because it enables one to see patterns, connections, and possibilities that are otherwise invisible.

Novel problems are more solvable when they are adjacent to subjects we understand. This curriculum’s bedrock of low-level theoretical knowledge touches a vast surface area of other subjects, bringing them within range of understanding.

Prolific Prototyping is Different

Prolific prototyping is a constant process of building and testing. It’s like taking soundings to discover the hidden landscape of impasses and opportunities in an uncharted territory. Prototypes are tests, not products. All prototypes are successful if they are informative. And they often answer additional questions we have not yet thought to ask. There is no faster way to develop an intuitive sense of a new territory.

Prolific prototyping can radically accelerate the process of discovery in design and engineering.

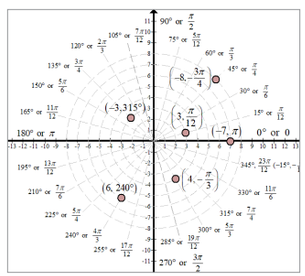
Module 1

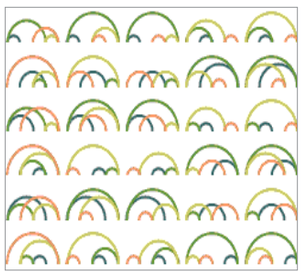
Proportions & Patterns

Systems of Numbers
the nature or numbers
natural numbers
integers
real and imaginary-
complex numbers

Algebra Review
Cartesian coordinates
polar coordinates
functions and graphs
exponents
scientific notation
solving polynomials
parametric equations

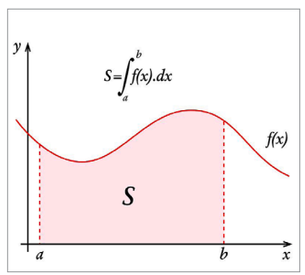
Trigonometry
solving triangles
solving circlessine
cosine
tangent
special triangles
exotic function





Combinations
permutations
basic probability
binary
octal
hexadecimal

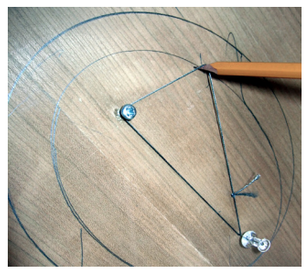
Basic calculus
derivatives
simple integrals
integrals and time

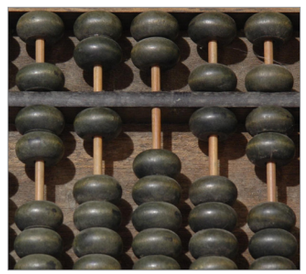


Sample Exercises

Design and create a 2-person game with wooden pieces based on permutations and/or probability.

Use pure geometry to create ellipses, parabolas, and hyperbolas with wood.





Design and create a binary abacus and hexadecimal abacus.

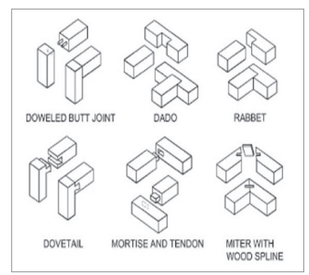
Using Hand Tools

Safety and First Aid
No exceptions
Appropriate clothing
PPW
Treating minor wounds
Treating minor burns

Using Hand Tools
measurement, precision
wrenches & pliers
saws, cutters, knives
striking & struck tools
drivers
vises, clamps, drills
sharpening tools

Palette of Materials
woods
metals
plastics
ceramics, stone
fibers, fabrics, leather
biomaterials

Working With Woods
types of woods
elasticity and grain
cutting
types of joins
finishing



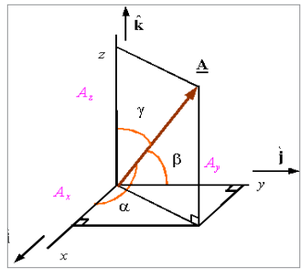
Module 2

Fields, Forces, & Waves

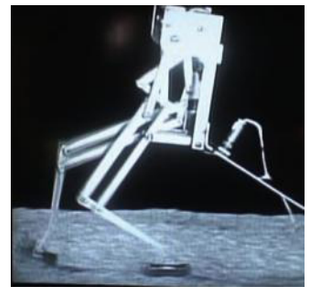
Sample Exercises

Using Machine Tools

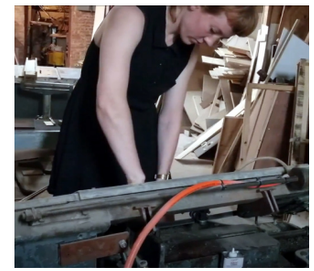
Vectors
unit vectors i,j
vector addition
vector subtraction
3D and 4D vectors

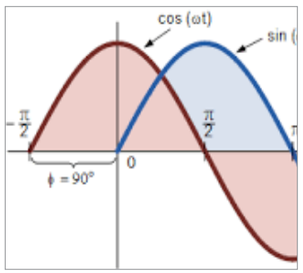


Design and fabricate a mechanism that walks

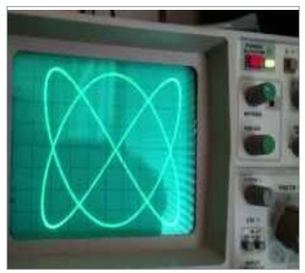


Cutting and Drilling
electric drills & drivers
drill press
table saw
band saw
chop saw

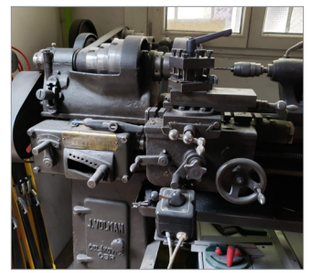




Motion and Time
displacement
velocity
acceleration
Oscillations and time
Waves
kinematics
Dynamics

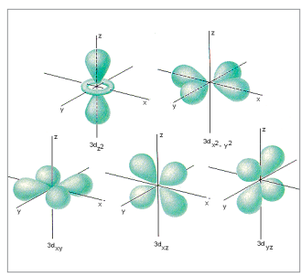


Design and build a simple electromagnetic transmitter and receiver to send pulses of energy across an air gap.



Shaping
lathe
manual mill

Atoms
Rutherford Model
forces
electron shells and ions
Periodic Table



Contest: Using field equations and scrap metals, engineer and build the most stupidly powerful capacitor you can.

Integral form

$$\oint \vec{E} \cdot d\vec{A} = \frac{q}{\varepsilon_0} = 4\pi k q$$

Differential form

$$\nabla \cdot E = \frac{\rho}{\varepsilon_0} = 4\pi k \rho$$

Working With Metals
types of metals
precision
cutting
welding
casting
finishing

Electricity
electric charge
the electric field
potential
capatacitance
current and resistance
electromotive force

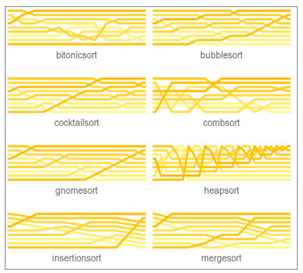
E&M
the magnetic field
inductance
oscillations
waves

Prototypes are Tests
design vs. technology
high and low precision
designing for fabrication
1-day prototypes
'failed' prototypes

Module 3

Logic & Flow

Flowcharts
conditionals
loops
data and memory
time complexity



With a Pencil
Sieve of Eratosthenes
Euclid's algorithm

blackjack single-player
blackjack card-counting
blackjack player+dealer
poker single-player
poker card-counting

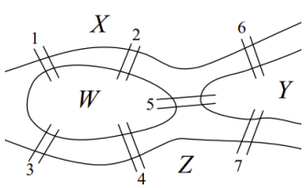
Quicksort
Binary Search
Breadth First Search
Depth First Search
Merge Sort

Dijkstra's Algorithm
Other Graph Algorithms

Lee Algorithm

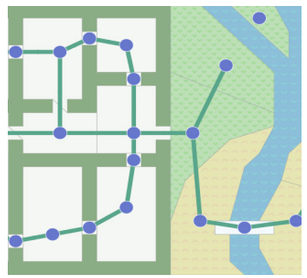
Kalman Filter
Fast Fourier Transform

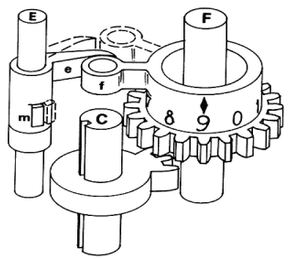
Graph Theory Basics
the world as networks
points / lines
edges / vertices
directed / undirected
ordered / unordered
connected/disconnected
applications



Sample Exercises

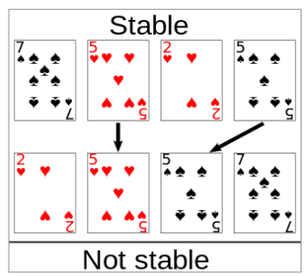
Design, create, and test a dating (or social introduction) algorithm implemented spatially across a whole building, transit network, or neighborhood using signage





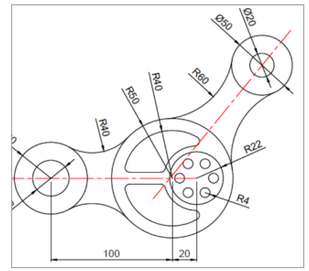
Design and create a machine that encodes base-10 to hexadecimal using gears

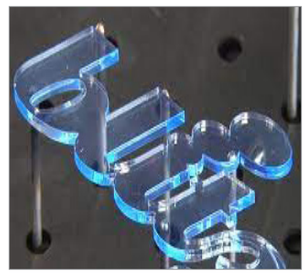
Describe various card games as algorithms using flow charts, including card-counting techniques. Test your algorithms against each other's letting the algorithms make all choices in a game.



CAD/CAM & Mechanisms

CAD & 3D Models
3D Concepts
CAD Concepts
FreeCAD or Fusion 360
Blender





Working w/ Plastics
variety of properties
cutting
joining / adhesives
casting
finishing

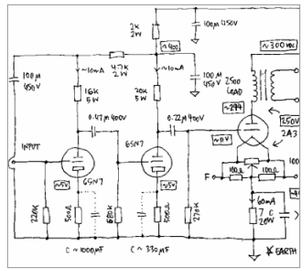
Using CNC Tools
router
mill
3D printer
laser cutter
wire bender

Power Transmission
gears
belts and pulleys
chain and sprockets
rotary motion
linear motion

Module 4

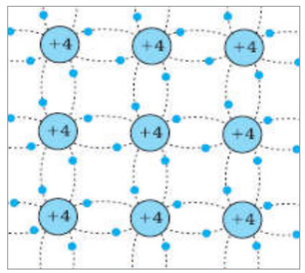
Analog & Digital

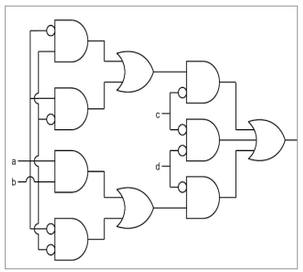
Circuit Basics
Ohm's Law
Kirchoff's Laws
series & parallel
simple elements:
cells
lamps
resistors
capacitors
inductors
relays



Analog Circuits
Dividers
Rectifiers
Voltage Regulator
Amplifiers
hi- and lo-pass filters
Oscillators
Integrator /Differentiator

Semiconductors
semiconductor physics
diodes
BJTs
field-effect transistors

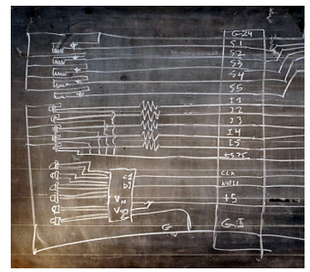


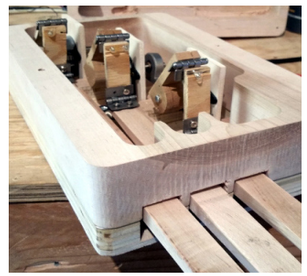


Digital Circuits
Boolean logic
AND, OR, gates
NOT, NAND, NOR,
XNOR gates
Latches, flip-flops
Counters
Timers

Sample Exercises

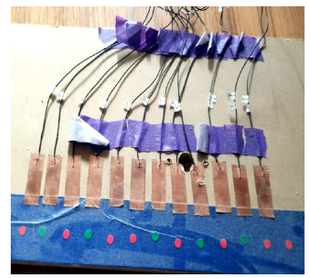
Design, CAD, and fab a circuit board for a circuit that performs a simple computation.



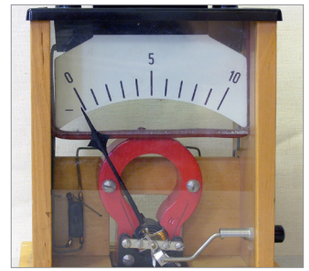


Design and create mechanical versions of all of the basic logic gates

Design and create a simple music synthesizer in any medium

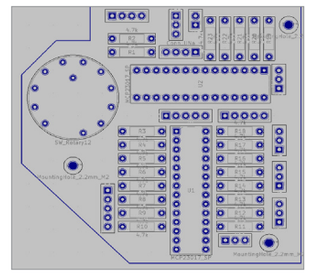


Electronic Tools & Fab



Electronics Skills
circuits
measurement
soldering & prototyping
meters
oscilloscopes

Circuit Board Design
reading data sheets
selecting components
Simulation
Circuit Board CAD

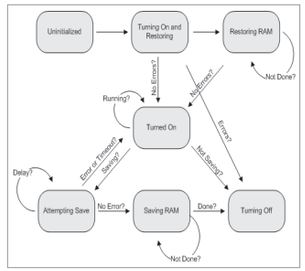


Troubleshooting
structure troubleshooting
magic bullet|Higgs boson
invisible influences
electrical connections

Module 5

Systems & Feedback

Compute Concepts
bits, bytes, ASCII
State Machines
Von Neumann machines
DIY simple computers



The Bare Metal
intro to Verilog, FPGAs
Intro to Assembly
Intro to compilation

Systems Basics
Collections and Systems
Stocks
Flows
Dynamic Equilibrium
Feedback Loops
Dominance
Delays
Oscillations
Constraints

Python 1
variables and memory
operators, conditions
control flow and loops
data structures
functions
modules

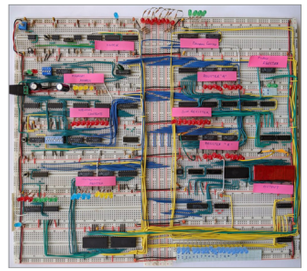
Python 2
classes & decorators
object orientation
lambda, map & filter
comprehensions
generators
threading

```
# Main handles network send/recv and can see all other classes directl
class Main(threading.Thread):
    def __init__(self, hostname):
        threading.Thread.__init__(self)
        self.network = Network(hostname, self.network_message_handler,
        self.queue = Queue.Queue()
        self.utils = Util(hostname)
        #self.gainRampThread = gainRampThread
        # default intermediate frequency
        #self.xtal_freq = 167233.6
        self.xtal_freq = 167218.6
        self.f_offset = 0           # adjust output freq
        self.status = {
            "avl-voice-1":"pass", # because this passes if it can resp
            "avl-voice-1-crystal-frequency-counter":"unset",
            "avl-voice-1-harmonic-generators":"unset",
            "avl-voice-1-harmonic-volume":"unset",
        }
        # get voice messages
        self.network.thirtybirds.subscribe_to_topic("voice_1")
        self.network.thirtybirds.subscribe_to_topic("client_monitor_re
        self.network.thirtybirds.subscribe_to_topic("mandala_device_re

    def update_device_status(self, devicename, status):
        print "update_device_status",devicename, status
        if self.status[devicename] != status:
            self.status[devicename] = status
            msg = [devicename, status]
            self.network.thirtybirds.send("mandala_device_status", msg
```

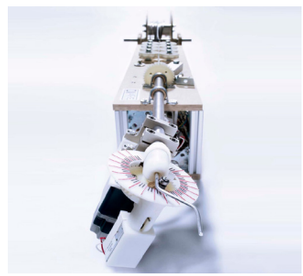
Sample Exercises

Design and fabricate a binary -> ASCII -> 14-segment-display circuit using switches and relays



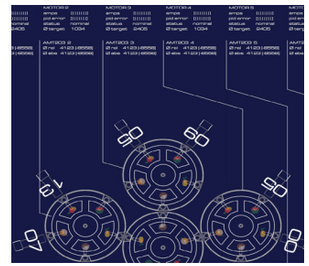
Write a Python program to offload an everyday process you usually do in your head.

Design, engineer, and fabricate a new type of computer-controlled tool.

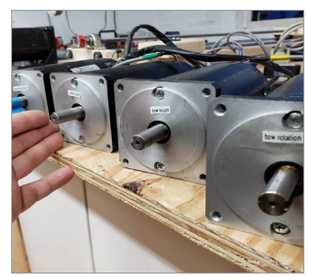


Control & Actuation

Designing Systems
Constraints
Iterations
Top-down design
Bottom-up design
Convergence diagrams
Discovery-orientation

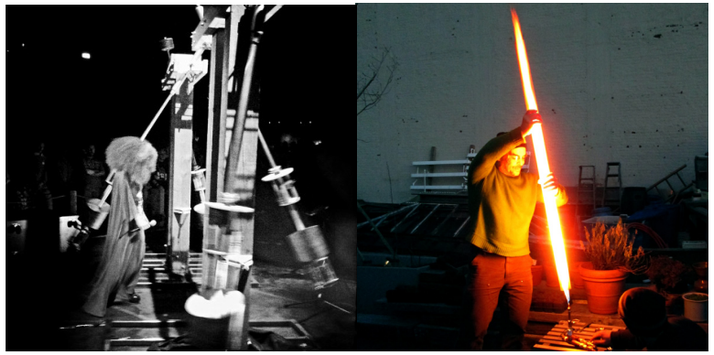


Actuation
motors
solenoids
valves for fluid & gas
illumination
indicators
shift registers
digital-analog converters



Sensors
switches
encoders
inertial sensors
prox/beam sensors
analog-digital converters

Andy Cavatorta Studio



Andy Cavatorta Studio specializes in ambitious projects unbounded by categories. Fine arts commissions, mobile robotics, opera set design, kinetic sculpture, product design, musical instruments, software platforms, game design, R&D, and whatever comes up next.

Clients include Björk, the MIT Museum, Royal Opera House (London), Oslo Opera House, MoMA, the central bank of Mexico, Barney's, Pierre Huyghe, and many more.

Andy Cavatorta is the creative and technical principal of Andy Cavatorta Studio.

Two good examples of his theory + prototyping methodology can be found here:

https://andycavatorta.com/irvine.html
https://andycavatorta.com/gravityharps.html