Beach Cities Robotics

FIRST Team 294

Robotics Instruction Course

Dale Hall, Bob Hosken, and Rick Wagner

Version 1.29, July 2, 2007, first created June 6, 2003

This document is written to accompany a course of instruction in robotics fundamentals for high school students participating in a competitive robotics team. Our intent in preparing the course content is to keep the material accessible to motivated high school freshmen yet allow it to be stimulating to seniors. The course covers topics essential to an understanding of robotics for a contributing competitive team member.

Robotics is a discipline at the intersection of mechanics, electronics, and computer science. As an area of research, the hard problems of robotics are algorithmic, so robotics is properly placed in the computer science departments of most universities. Robotic algorithms are founded on the physical world in which robots find themselves, so the mechanical and electrical topics are treated here first.

Part 1 is an introduction to mechanics that reviews analytic geometry and vectors before describing Newtonian statics. Part 2 is an introduction to strength of materials and structures. Part 3 is an introduction to mechanical design. Part 4 introduces electricity and DC circuits. Part 5 is an overview of motive power including batteries, motors, and control circuits. Part 6 is an introduction to computer programming. Part 7 introduces robot algorithms.

This document is maintained in Microsoft Word and exported to HTML format for Web publication. As such, it may be best viewed with Microsoft's Internet Explorer. We have found that some versions of Netscape do not properly display the Greek letters used in some of the equations here.

This instruction course material continues to be a work in progress and is updated from time to time (see the version history section). New sections under consideration include programming for the FIRST controller and teamwork and strategy.

Table of Contents

1 Introduction to Mechanics 8
1.1 Review of Cartesian Geometry 8
1.1.1 Points in the Plane 8
1.1.2 Points in Space 9
1.1.3 Lines 9
1.1.4 Surfaces 10
1.1.5 Volumes 10
1.2 Vectors and Forces in the Plane and in 3D 11
1.2.1 Forces and their Representations 11
1.2.2 Point Forces 11
1.2.3 Magnitude and Direction 12
1.2.4 Components of Forces 12
1.2.5 Vectors: Direction, Velocity, Force 12
1.2.6 2D Force Vectors 12
1.2.7 3D Force Vectors 12
1.2.8 Newton's Law of Transmissibility 12
1.3 Resultant Forces and Moments 12
1.3.1 Vector Operators: Sum, Difference, Scalar Multiplication 12
1.3.2 Moments, Torques, Couples, and Wrenches 13
1.3.3 Newton's Laws of Motion: Linear and Rotational 13
1.4 Statics: Reactions and Free Body Analysis 13
1.4.1 Newton: Action and Reaction 14
1.4.2 Statics: Sum of Forces = 0 = Sum of Moments 14
1.4.3 Solving Static Equations 14
1.4.4 Static Equilibrium and Indeterminism 16
2 Strength and Structures 17
2.1 Stress and Strain 17
2.1.1 Tensile Stress and Strain 17
2.1.2 Compression Stress and Strain 17
2.1.3 Shear Stress and Strain 18

2.1.4 Bulk Stress and Strain 18
2.2 Tension-Compression, Bending, and Torsion 18
2.2.1 Tensile (Compressive) Stress 18
2.2.2 Bending Stress 18
2.2.3 Torsional Stress 20
2.3 Material Properties, Test, and Analysis 21
2.3.1 Strength 21
2.3.2 Stiffness 21
2.3.3 Other Material Properties 21
2.3.4 Materials Testing 22
2.3.5 Analyzing Designs 22
2.4 Structures: Beams, Columns, Frames, Struts 22
2.4.1 Beams 22
2.4.2 Columns 23
2.4.3 Moment Bearing Joints 23
2.4.4 Frame and Panel 23
2.4.5 Struts and Trusses 23
2.5 Deflection Analysis 23
2.5.1 Axial Deflection 23
2.5.2 Beam Deflection 24
2.5.3 Torsional Deflection 24
3 Designing Structures and Mechanisms 25
3.1 Fabrication of Parts 25
3.2 Assemblies, Bolted Joints, Welding 25
3.2.1 Threaded Fasteners 25
3.3 Friction 26
3.4 Mechanical Advantage: Levers and Linkages 26
3.5 Rotary Mechanisms: Gears, Chains, Belts, and Pulleys 27
4 Electricity 28
4.1 Basic Terms 28
4.1.1 Voltage: Force that causes Charge to move 28
4.1.2 Battery: Device with constant voltage, E 28

4.1.3 Charge: Property of quantity of ions or electrons 28
4.1.4 Current: Speed of charge motion 28
4.1.5 Resistor: Electrical device that slows flow of charge 28
4.1.6 Power: Rate of electrical energy flow 28
4.2 Schematic Symbols 28
4.3 Example Circuit 29
4.4 Ohm’s Law 29
4.5 Measurements 30
4.5.1 Put the Leads in proper socket 30
4.5.2 Put the pointer on appropriate measurement scale 30
4.5.3 Caution 30
4.6 Examples 32
4.7 Expanded Example 32
4.8 Voltage Divider 33
4.9 Electrical Power 34
5 Motive Power 35
5.1 Mass and Weight 35
5.2 Newton's Laws of Motion 35
5.3 Work, Heat, Energy, Current, Voltage, and Power 35
5.4 Inductance and Capacitance 36
5.5 Energy Defined 37
5.6 Motors and Generators 37
5.7 Principles of Operation 39
5.7.1 DC Motor Operation 41
5.7.2 DC Motor Control 42
5.8 Torque and Speed 46
5.9 Motive Power Guidelines 48
5.10 Drive Trains 50
5.10.1 Mechanical Advantage 50
5.10.2 Gears and Transmissions 50
5.10.3 Chains and Sprockets 52
6 Introduction to Programming 54

6.1 Algorithms and Computation 54
6.2 Variables and Operators 55
6.3 Decisions 56
6.4 Data Structures 57
6.4.1 Primitive Types 57
6.4.2 Arrays 57
6.4.3 Objects 57
6.5 Program Design 58
6.6 Computer Learning Resources 59
7 Robot Algorithms 60
7.1 Processes in the Physical World 60
7.1.1 Robot Attributes: Autonomy, Mobility, Manipulation 60
7.1.2 Space Robotics 61
7.2 Planning 63
7.3 Sensing and Control 63
7.4 Navigation 64
7.5 Advanced Robotics Example: Robo-Butler 64
7.5.1 Robo-Butler Requirements 64
7.5.2 Robo-Butler Specification 65
7.5.3 Object Oriented versus Structured Paradigm 71
7.5.4 Object Oriented Analysis: Robo-Butler Example 72
7.5.5 Robo-Butler Design 76
8 Programming the FIRST Microcontroller for 2004 76
9 Acknowledgements 76
10 References 76
11 Version History 76

Table of Figures

Figure 1: Cartesian coordinate system showing a point labeled "A" with its coordinates. 8

Figure 2: Two equal and opposite forces, F1 and F2, acting along the same line cancel each other out. 13

Figure 3: If the two forces are separated by some distance (d) they form a "couple." 13

Figure 4: End view of a wall with weight W, wind force F, and reactions. At some level of wind force, R3 will go to zero and the wall will blow over. 14

Figure 5: A convenient coordinate system is chosen and dimensions are assigned. 15

Figure 6: Tensile force on a section of wire. 17

Figure 7: A cantilever beam loaded at the end with a vertical force and its corresponding shear and moment loading diagrams. 19

Figure 8: A simply supported beam with a center load. 20

Figure 9: A shaft loaded in torsion. 20

Figure 10: A hex-head of a bolt. The three radial dashes indicate that this is an SAE grade 5 bolt with a proof strength of 85 ksi. 25

Figure 11: Schematic symbols. 28

Figure 12: Example circuit. 29

Figure 13: Ohm's law. 30

Figure 14: Resistance measurement. 31

Figure 15: Voltage measurement. 31

Figure 16: Current measurement. 32

Figure 17: Expanded example schematic. 33

Figure 18: Voltage divider schematic. 34

Figure 19: A typical permanent magnet DC electric motor, disassembled. 38

Figure 20: Schematic of a simple permanent magnet DC motor 39

Figure 21: Faraday's induction law. 40

Figure 22: DC motor operation. 42

Figure 23: Battery performance. 43

Figure 24: 2001 National Champ's PWM oscilloscope photo. 44

Figure 25: Victor 884 and Spike H-bridge internal four switch operation; two diagonal switch closures determine motor direction. 44

Figure 26: Victor 884 and Spike H-bridge internal four switch operation; joystick to neutral, effect of jumper: coast and brake. 45

Figure 27: Using 30” length #6 AWG to measure robot current method: DMM + Ohms law I = V/R (Rshunt = high current low ohm shunt). ... 45

Figure 28: Using 60A circuit breaker to measure robot current method: DMM + Ohms law I = V/R (Rshunt = high current low ohm shunt). ... 46

Figure 29: Lorentz’ law. ... 47

Figure 30: DC motor torque vs. speed curve. ... 48

Figure 31: Double reduction hand winch. ... 50

Figure 32: Spur gears. The smaller gear is called a pinion. ... 51

Figure 33: Worm drive. The worm drives the specially cut spur gear. ... 52

Figure 34: Roller chain and sprockets. ... 53

Figure 35: Roller chain terminology. ... 53

Figure 36: Mars exploration rover. ... 60

Figure 37: Artist’s conception of Jupiter’s Icy Moons Orbiter (JIMO) nuclear powered robotic spacecraft. ... 62

Figure 38: First refinement of the data flow diagram. ... 67

Figure 39: Second refinement of the data flow diagram. ... 68

Figure 40: Entities in the Robo-Butler scenarios. ... 68

Figure 41: Entity-relationship diagram. ... 69

Figure 42: STD for the Robo-Butler. ... 70

Figure 43: Petri net for a portion of the Robo-Butler software specification. ... 71

Figure 44: The entity-relationship diagram becomes the class model ... 72

Figure 45: Dynamic model for the Robo-Butler class ... 73

Figure 46: The refined class model adds inheritance to distinguish guests from the owner and the various types of possession objects. ... 74

Figure 47: Robo-Butler functional model. ... 75

Figure 48: Robo-Butler project PERT chart. The numbers on the arrows are estimated time spans in days for each of the pointed-to activites. ... 75

1 Introduction to Mechanics

1.1 Review of Cartesian Geometry

In the 17th century, philosopher and mathematician René Descartes initiated the use of a spatial coordinate system so that geometric problems could be attacked using algebraic techniques. This system is called the Cartesian coordinate geometry.

Scientists, engineers, and mathematicians use the technique of abstraction to simplify problems to make them more easily solvable. Abstraction is disregarding irrelevant detail so that a solution becomes more accessible.

One very powerful and common technique of abstraction is to render three dimensional (3D) problems into two dimensional (2D) equivalent problems. We can often find a way to represent the essential parts of a problem in 2D. For example, even though a road in the real world goes up and down hills, the essential problem is to figure out where to turn where roads intersect each other. Thus, a road map for navigating is usually rendered on 2D paper.

1.1.1 Points in the Plane

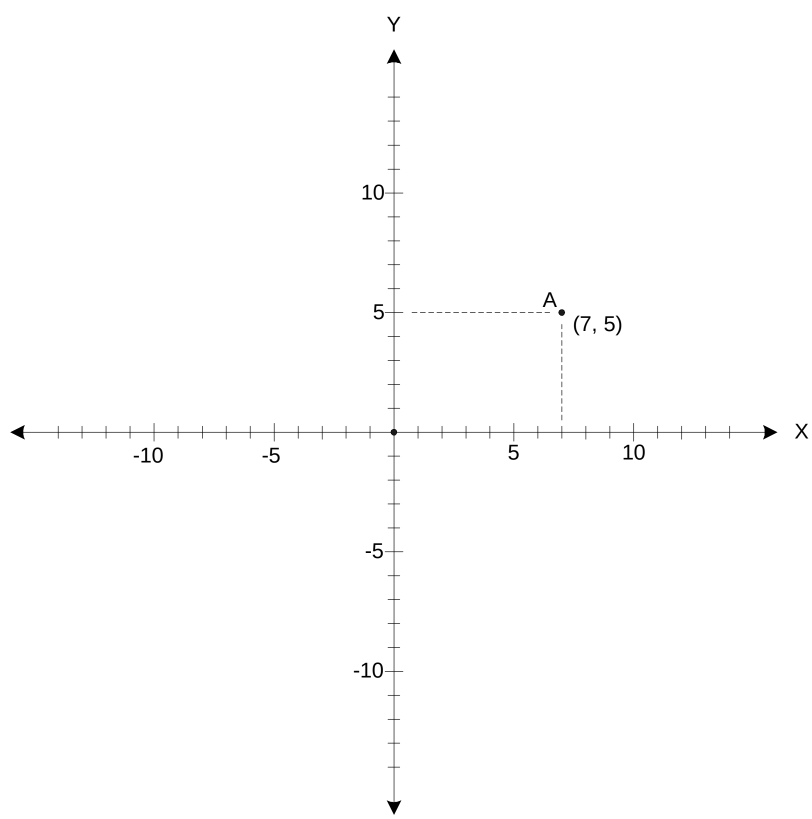


Figure 1: Cartesian coordinate system showing a point labeled “A” with its coordinates.

Geometers use the concept of a "point" as an abstraction of the idea of location. A point is the simplest geometric concept.

1.1.1.1 X and Y axes

In the plane, two lines intersecting at right angles can be arbitrarily designated as location references for all points in the plane.

1.1.1.2 Ordered pair notation

Every point in the plane has a distance from the Y axis (its X coordinate) and a distance from the X axis (its Y coordinate). These two coordinates are written in the order X, then Y, an ordered pair notation in parentheses like this: (x, y).

1.1.1.3 The Theorem of Pythagoras

Pythagoras was an ancient Greek philosopher who founded a school devoted to mathematics. Pythagoras is best known for his proof of the theorem that bears his name. See the Web page (http://teamster.usc.edu/~dteam/images/robotics/Pythagoras.html) for his proof of the Pythagorean theorem. The Pythagorean theorem is invoked in computing the distance between two points.

1.1.1.4 Computing the Distance Between Two Points

If we regard a line drawn between any two points in the plane as the hypotenuse of a right triangle, with the other two sides of that triangle being formed by horizontal and vertical line segments, then the horizontal side of that triangle is the difference in the X coordinates of the two points, and the vertical side of that triangle is the difference in the Y coordinates. Applying the Pythagorean theorem to those two horizontal and vertical sides allows us to compute the hypotenuse, which is the distance between the two points:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

1.1.2 Points in Space

Some problems are inherently three-dimensional and cannot be collapsed into 2D without losing some vital information.

1.1.2.1 The Z Axis

To assign a Z coordinate to points in three-dimensional (3D) space we establish a third axis (Z) that is normal (at right angles) to both the X and Y axes.

1.1.2.2 Ordered Triple Notation

3D points have an ordered triple notation, with the Z coordinate coming after the X and Y coordinates, like this: (x, y, z).

1.1.3 Lines

A piece of string pulled tight forms a straight line, which is the shortest distance between two points.

1.1.3.1 Lines and Segments

In Euclidean geometry a “line” is of infinite extent, an idealization that doesn’t exist in our everyday experience. A “segment” is a portion of a line that is bounded by two end points.

1.1.3.2 Slope-Intercept Equation

Any non-vertical line can be represented by the equation

y = mx + b

where m is the slope of the line (ratio of rise to run) and b is the Y-axis intersection of the line.

1.1.3.3 Two-point equation

Given two points on a line, the line can be represented by the equation

$$y - y_1 = (y_2 - y_1)(x - x_1) \div (x_2 - x_1)$$

1.1.3.4 Lines in Space

A line in space can be conveniently represented by six numbers: three numbers to define a point in space that is somewhere on the line, and three numbers to represent a direction vector along the line.

1.1.4 Surfaces

The boundary of a solid object is called a surface. In the plane, lines can form boundaries of 2D regions.

1.1.4.1 Polygons

A 2D region bounded by line segments is called a polygon (many sides). The simplest polygon is a triangle.

1.1.4.2 Rectangles

A rectangle is a four-sided polygon with each side at right angles to its neighboring side.

1.1.4.3 Circles: Radial and Tangential Lines

A circle is a locus of points equidistant from a center point. Lines radial to a circle are incidental to the center of the circle. Tangential lines touch the circle at only one point.

1.1.4.4 General 2D Regions in Space

A finite general 2D contiguous region is bounded by a curve of finite length. Points in the plane are partitioned into three proper subsets by a general region: points in the region, points outside the region, and points on the boundary of the region. If the 2D region has holes in it then the boundary is disjoint.

1.1.5 Volumes

Solid objects occupy space. A bounded space is called a volume.

1.1.5.1 Polyhedra and Boundaries of Volumes

A polyhedron is a solid with polygonal faces, that is, the boundary of the polyhedral volume is a set of polygons.

1.1.5.2 The Platonic Solids

The five Platonic solids are the regular polyhedra that each have only regular polygons for faces. Plato who was already famous as a wrestler became famous as a mathematician when he proved that there were only five regular polyhedra: tetrahedron, cube, octahedron, dodecahedron, and icosahedron (with triangles, squares, triangles, pentagons, and triangles, as faces, respectively.

1.1.5.3 Rectangular Prisms

A right rectangular prism is a box shape and has many uses in design. The rooms of houses and other buildings are usually in this shape. Boards, slabs, and other raw materials often come in this shape.

1.1.5.4 Spheres

The boundary of a sphere is the 3D counterpart of a circle.

1.1.5.5 Volumes of Revolution

A 2D shape when rotated out-of-plane (about some axis in the plane) describes a *volume of revolution*. Such shapes are produced by “turning” as on a lathe. A potter also produces such a shape in clay with his wheel.

1.1.5.6 Swept Volumes

A 2D shape when driven along a line (straight or curved) in space describes a “swept volume.” The handle of a coffee cup might be an example of a swept volume.

1.2 Vectors and Forces in the Plane and in 3D

1.2.1 Forces and their Representations

A “force” is a push or a pull. Forces can cause things to move, bend, or break. The weight of an object is a force downward toward the center of the Earth caused by the acceleration of gravity. Impact forces are caused by the sudden deceleration of a thrown or moving object. Centrifugal forces are caused by the acceleration of a moving and turning object. Force is measured in units of Newtons (N) in the international system (SI) and in units of pounds (lb_f) in the English and United States (USCS) systems.

Force distributed over area is pressure, and conversely, pressure on an area results in a force.

1.2.2 Point Forces

The concept of a point force is a useful abstraction because a point is the simplest geometric object and many mechanical problems can be satisfactorily solved using point forces. In practice, most forces originate as pressures exerted over areas. For example, if you push on a door to open it the area of contact of your hand distributes the force of the push into a pressure over the area of contact. The dynamics of pushing the door open are simplified if we consider the push as acting on a single point within the area of contact.

1.2.3 Magnitude and Direction

Point forces are applied at some point location in space, but for analysis purposes, only the magnitude of the force and its direction are important. There is a line of action of the force: any point of application on that line is equivalent to any other. The magnitude of the force tells us how much force is applied. The direction is the particular line of action of the force.

1.2.4 Components of Forces

Cartesian coordinate representation allows us to break up forces into components aligned with the coordinate system axes. This lets us solve 2D and 3D problems as single dimensional problems separately.

1.2.5 Vectors: Direction, Velocity, Force

Something that has magnitude and direction we call a *vector*. Point forces have magnitude and direction so we commonly represent forces as vectors. If we want to refer to direction only, we can use a vector with a unit magnitude, called a *unit vector*. Another property that is commonly represented as a vector is *velocity*.

1.2.6 2D Force Vectors

A force vector in 2D (some plane) has an X-direction component and a Y-direction component.

1.2.7 3D Force Vectors

A force vector in 3D is a 2D vector with a Z-direction component added to it.

1.2.8 Newton's Law of Transmissibility

A force vector at a point in space is equivalent to a similar force vector at any point along the line defined by the vector and the point. This is why a spear works well as a weapon: a person holding the spear can have the force of his hands transmitted along the shaft of the spear to its point. This is called Newton's law of the transmissibility of forces, named after Sir Isaac Newton who first expressed the idea formally.

1.3 Resultant Forces and Moments

1.3.1 Vector Operators: Sum, Difference, Scalar Multiplication

Expressing forces as vectors in a Cartesian coordinate system makes it easy to perform operations on them. To add two forces, add up their x, y, and z components separately. Similarly for subtractions. To multiply a force by a scalar (a dimensionless numerical value), multiply all the components by the value.

1.3.2 Moments, Torques, Couples, and Wrenches

Two equal and opposite forces passing through the same point (on the same line of action) will cancel each other out. However, if the point of action of one of the forces is moved off the line of action, the result is a torque. This torque is called a couple or moment. All three terms, moment, torque, and couple, are equivalent in meaning, but there are language conventions for their use. Torque usually refers to moment with its axis aligned with that of a shaft. The term “couple” usually refers to moment due to equal and opposite forces separated by a distance.

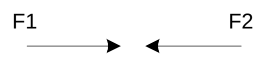


Figure 2: Two equal and opposite forces, F1 and F2, acting along the same line cancel each other out.

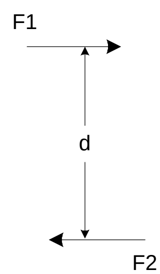


Figure 3: If the two forces are separated by some distance (d) they form a “couple.”

1.3.3 Newton’s Laws of Motion: Linear and Rotational

Sir Isaac Newton formulated the relationships among force, mass, and acceleration on one hand, and among moment, moment of inertia, and angular acceleration on the other.

$\mathbf{F} = m\mathbf{a}$

$\mathbf{M} = I\alpha$

1.4 Statics: Reactions and Free Body Analysis

Solid objects deform under load, but usually in successful engineering structures, these deformations are small enough that we can ignore them for structural analysis purposes. Neglecting the deformations is a simplifying assumption, another form of the principle of abstraction. First we assume the object under study does not deform at all, that it is completely rigid. Then we analyze the loads on the body (object), and if the loads are sufficiently high we may decide we need to do a deformation analysis. To simplify a system, we take each of the elements of the system and consider them separately. These separate elements are called free bodies.

1.4.1 Newton: Action and Reaction

Sir Isaac Newton observed that every action is accompanied by an equal and opposite reaction. If a body is unconstrained, the reaction is inertial and the body accelerates. If the body is constrained (nailed down, for example), the constraints react the applied load.

1.4.2 Statics: Sum of Forces = 0 = Sum of Moments

Consider the act of holding a block of metal in a vise. We turn the screw handle to move the vice jaws together until they clamp down on the block. If we increase the force that is holding the block, the moving jaw presses on the block and the fixed jaw of the vice *reacts* the load. The load and its reaction are equal and opposite. As such, they completely balance each other: they cancel out. This situation of balanced loads is true in every *static* situation.

1.4.3 Solving Static Equations

In static structures we often want to find internal loads or reactions, with applied loads being given. First we draw a diagram of the structure with the applied loads and the constraints. Then we solve for the reactions by setting up the equations of static equilibrium:

$\Sigma \mathbf{F} = 0$

$\Sigma \mathbf{M} = 0$

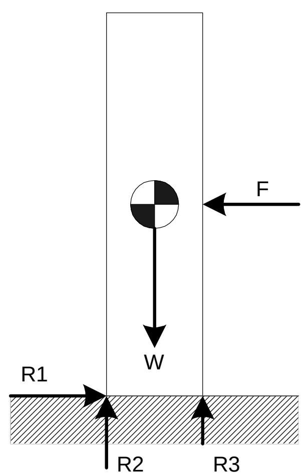


Figure 4: End view of a wall with weight W, wind force F, and reactions. At some level of wind force, R3 will go to zero and the wall will blow over.

To find the wind force for which the wall blows over, we first choose a convenient coordinate system and assign dimensions for the height (h) and thickness (t) of the wall as shown below in Figure 5.

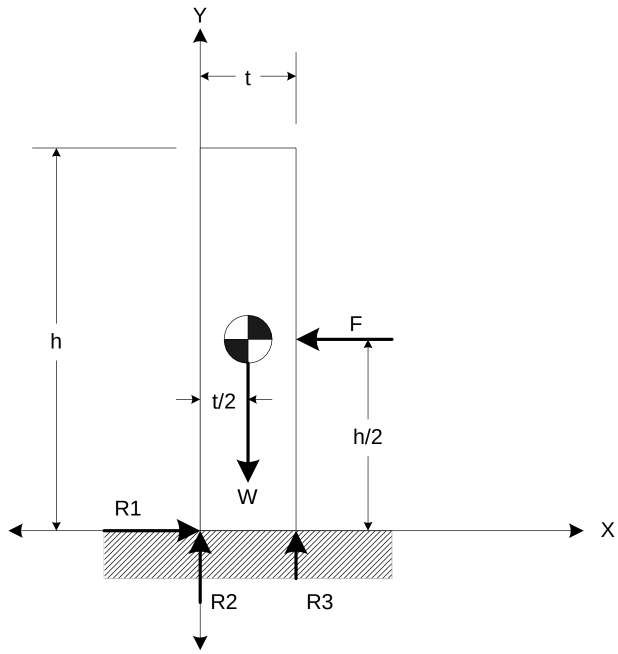


Figure 5: A convenient coordinate system is chosen and dimensions are assigned.

We then set up the equations of static equilibrium:

$\Sigma F_x = 0 = -F + R_1$ (1)

$\Sigma F_y = 0 = -W + R_2 + R_3$ (2)

$\Sigma M_z = 0 = Fh/2 - Wt/2 + R_3 t$ (3)

These three linear equations are true simultaneously.

Suppose now we are given that the height of the wall (h) is 100 inches, the thickness (t) of the wall is 20 inches, the weight for a section of the wall (W) is 6,000 pounds, and the wind force on that section of the wall (F) is 280 pounds:

$h = 100$ (4)

$t = 20$ (5)

$W = 6000$ (6)

$F = 280$ (7)

Problem: does the wall blow over? There are two ways to approach this problem. The first is to solve for R3. If it is negative, the wall blows over. The second approach is to set R3 to zero and solve for F, which would be the force required to blow the wall down. If F is less than 280 lb, the wall blows down.

With our convenient choice of coordinate system, equations (1) and (2) play no role in the solution approaches. Using the first approach, we solve equation (3) for R_3:

$R_3 = W/2 - (F * h) / 2t$ (8)

Substituting the values for the variables into equation (8) we have:

$$R_3 = 6000 / 2 - (280 \times 100) / (2 \times 20) = 2300 \quad (9)$$

A positive reaction, so the wall is stable. Using the second approach, we set R_3 equal to zero in equation (3):

$$Fh/2 - Wt/2 + 0 = 0 \quad (10)$$

Solving equation (10) for F we have:

$$F = Wt/h = (6000 \times 20) / 100 = 1200 \quad (11)$$

It takes 1200 lb to blow the wall over, so the wall is stable.

1.4.4 Static Equilibrium and Indeterminism

If the forces and reactions on a body are balanced as given by the equations of static equilibrium, then the body is said to be in static equilibrium.

Sometimes a rigid body is constrained in such a way that there is no solution for the equations of static equilibrium. Such a case is called statically indeterminate. When this happens we can make some assumptions to either bound the reactions, or develop a solution using deflection analysis, which is computationally more difficult.

2 Strength and Structures

A robot requires a physical body for making its way around in the world. The robot body must be strong enough to withstand the forces it will be subjected to in the course of its operation. These forces can be internally generated such as the motor torque reaction at a motor mount, or externally generated such as impact loads experienced in a collision.

2.1 Stress and Strain

Solid materials resist deformation under tension, compression, shear, and bulk stresses. The amount that solid materials deform under stress is called strain. Homogeneous materials, such as metals, resist tension, compression, and shear about equally well.

2.1.1 Tensile Stress and Strain

Consider pulling on a piece of wire. The stress in the wire is the tensile force in the wire divided by its cross-sectional area, in units of pounds per square inch (psi). If it is pulled hard enough the wire will break. The stress experienced by the wire when it breaks is called the ultimate tensile stress for the material the wire is made of. At stress levels below ultimate, the wire will increase in length. The tensile strain is the increase in length divided by the length of the unstressed wire. The units of strain are inches per inch (dimensionless).

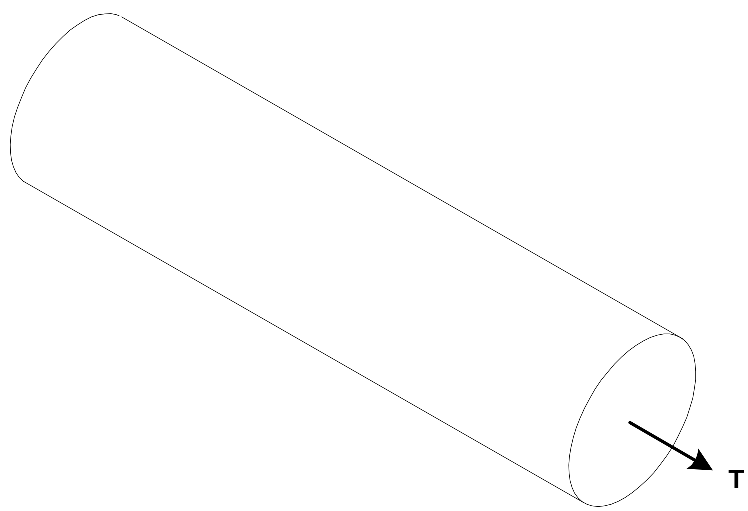


Figure 6: Tensile force on a section of wire.

2.1.2 Compression Stress and Strain

Compression stress and strain are very similar to those for tension, except they are in the reverse direction. A thin wire will bend easily so compression is better illustrated by the case of a stack of bricks. The bottom brick in the stack will experience the most compression stress and strain. As the bricks are stacked up, the bottom brick actually gets a bit thinner. At some ultimate load, the brick will be crushed.

2.1.3 Shear Stress and Strain

Shear stress and strain are due to shearing force applied to the material. When brittle materials (like fired clay bricks) fail under compression loading, they actually fail due to shear stress.

2.1.4 Bulk Stress and Strain

Solids and liquids both resist bulk deformation. Gasses have no deformation resistance at all. If we fill a balloon with air and take it under water, it's volume gets smaller by about half at 30 feet down. A balloon filled with water doesn't change in volume significantly if we submerge it. A block of metal under water is undergoing bulk stress: the pressure is the same in all directions.

2.2 Tension-Compression, Bending, and Torsion

Tensile (compressive), bending, and torsion are loads that we need to consider when designing a mechanical element. These three types of loads can be applied simultaneously to a part. Fortunately, for most engineering parts, deflections will be linear and negligible so we can use the principle of linear superposition to help us solve for the maximal stress on the part. Linear superposition means that the loads don't interact synergistically with each other, we can just calculate them separately and add them linearly.

2.2.1 Tensile (Compressive) Stress

Tension and compression calculations are similar, only the sign of the stress is reversed. Remember this mnemonic: tension is positive. If you feel tense before an exam, that's a good thing.

Suppose we have a square rod or bar, one inch on a side. The cross-sectional area of the bar is one square inch (1.0 in^2). If we pull on the bar with a tension of 100 lb, the tensile stress in the bar is 100 pounds per square inch (100 psi, or 100 lb/in^2). Similarly if we push on the bar, but since compression is negative, the stress is –100 psi.

The formula for tensile (compressive) stress is

$$\sigma = F / A$$

where the Greek letter sigma (s) is stress, F is the load, and A is the cross-sectional area of the member. A positive force is a tensile load.

2.2.2 Bending Stress

If we hold one end of the square bar in a vice and apply a pure moment (couple) to the end of the bar (with the axis of the moment transverse to the bar's long axis), the bar will bend (slightly) and assume a circular shape. This is because under the bending load, one side of the bar gets longer and one side of the bar gets shorter. The region in the middle of the bar does not change in length. The regions of maximal stress are at the surfaces of the bar.

The formula for (normal) bending stress is

σ = Mc / I

where M is the moment, c is the distance from the neutral axis, and I is the area moment of inertia of the cross section of the member.

Bending stress is also caused by force loads. For example, the cantilever beam shown below in Figure 7 has a moment reaction at its supporting wall:

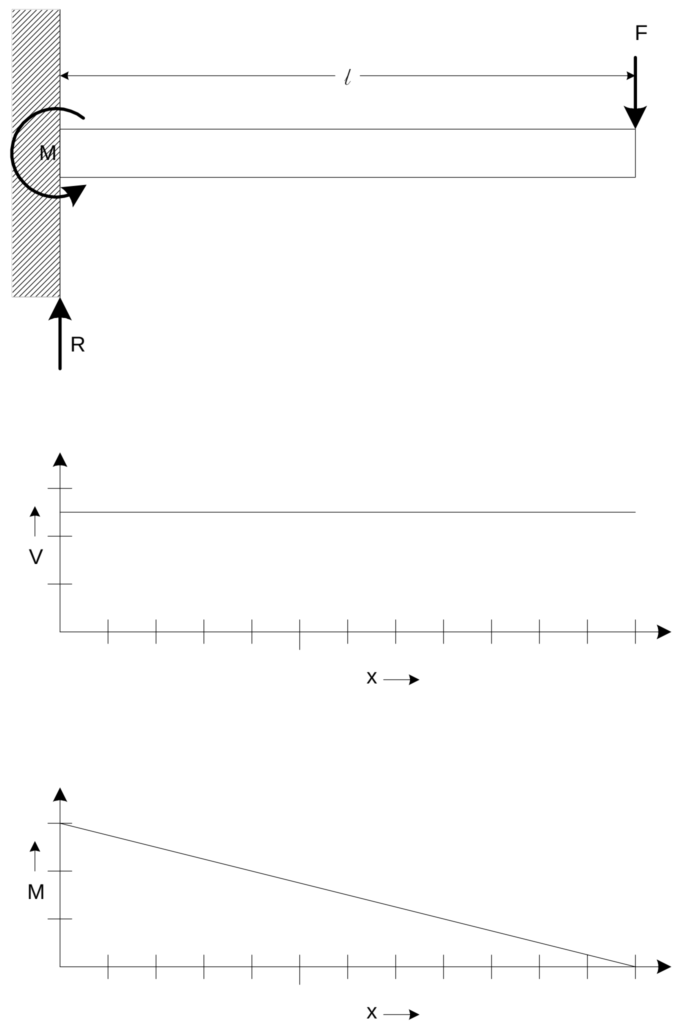


Figure 7: A cantilever beam loaded at the end with a vertical force and its corresponding shear and moment loading diagrams.

The shear in this case is constant throughout the beam but the moment increases linearly to a maximum at its wall reaction point. The beam shown below in Figure 8 is called simply supported and it has a point load in its center. The maximum bending moment in the simply supported case will occur at the point load. The two simple supports will not react moments, so the moments are zero at those points.

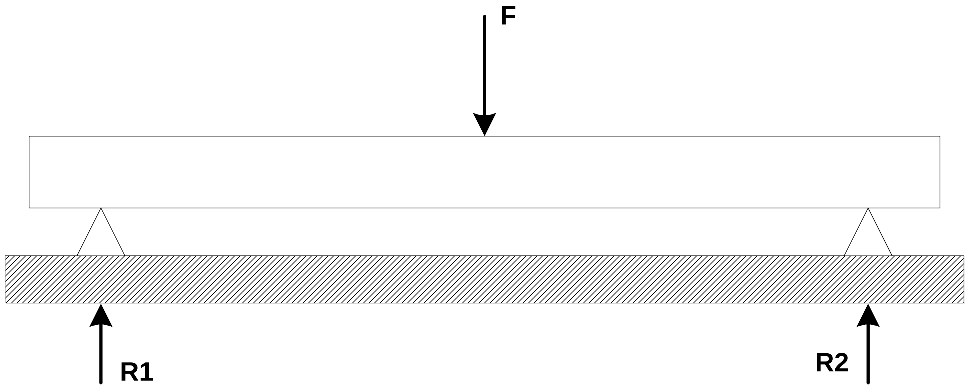


Figure 8: A simply supported beam with a center load.

For a rectangular cross section, $I = bh^3/12$, where b is the base width, h is the height of the section.

There are also shear stresses involved in bending, and these stresses combine non-linearly with normal stress, but normal stresses usually dominate and with reasonable safety margins we can usually neglect performing an exact stress computation.

2.2.3 Torsional Stress

For a round bar, a moment aligned with the long axis of the bar is called a torsion load. Torsional stress is a kind of shear stress and is also maximal at the surface of the bar.

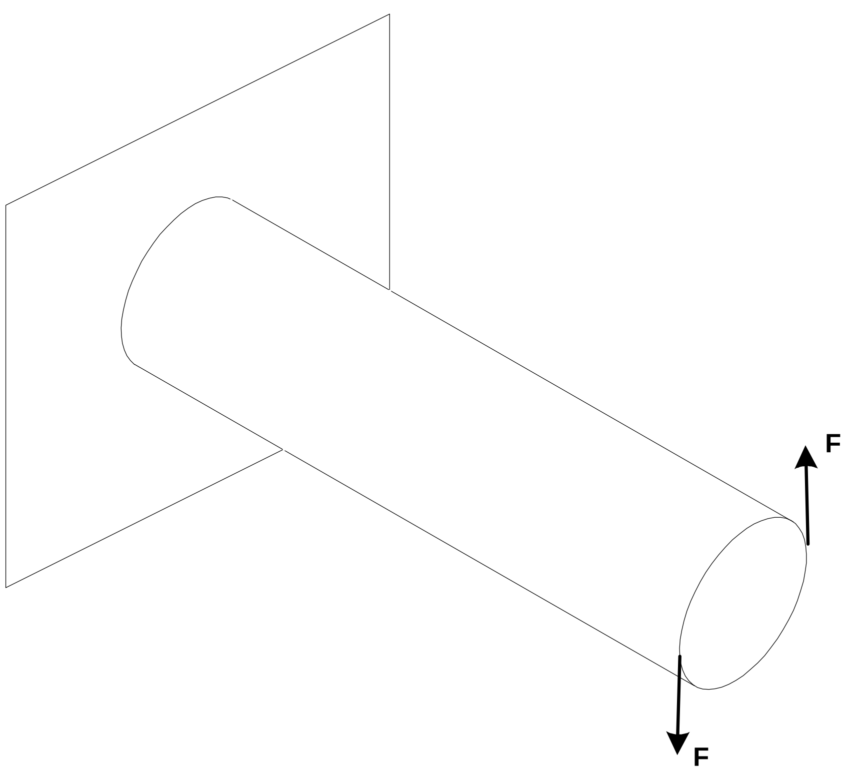


Figure 9: A shaft loaded in torsion.

The formula for torsional stress is

$\tau = T\rho/J$

where the Greek letter tau (t) is the shear stress, T is the torque applied to the member, the Greek letter rho (r) is the radius of the stress location, and J is the polar moment of inertia of the shaft.

For a circular cross section, $J = \pi d^4/32$.

2.3 Material Properties, Test, and Analysis

Different solid materials vary in their structural properties, including strength, stiffness, and toughness. These properties of materials are well defined. In addition to structural property differences, materials vary in other properties such as coefficient of thermal expansion (CTE), thermal conductance, resistance to heat, resistance to corrosion, resistance to moisture, compatibility with adhesives, etc.

2.3.1 Strength

Strength is the ability of a material to take a load without fracturing or yielding excessively. Strengths to consider include tensile, compressive, and shear. The strength of a material can depend on the processes used to prepare the material. For example, heating of metals can soften them by annealing and cold working can harden them. Strength and hardness are closely related. The harder a material, generally, the stronger it is.

2.3.2 Stiffness

The stiffer a material is the less it elastically deforms under load. The modulus of elasticity (Young's modulus) is a key material property for computing deflection. For most structures in robotics, stiffer is better. Usually (but not always) a stiffer design is also stronger.

2.3.3 Other Material Properties

Toughness is a measure of how much mechanical (impact) energy a material can absorb without fracturing. Brittle materials such as glass, tool steel, and cast iron, even though they may have considerable static strength, have very little toughness. Mild steel has excellent toughness.

Corrosion resistance of metals should be considered for outdoor or marine applications. Generally, copper and aluminum alloys are good choices for mildly corrosive environments. Plain or carbon steel can be used in lubricated and sealed mechanisms out of doors. Corrosion resistant steel (CRES) has chromium added. Steels with 30% or more chromium are non-magnetic and are called “stainless.” However, even stainless steel will rust to some extent when left unprotected in a marine environment.

Electronic components and motors generate significant heat that needs to be removed. Conducting the heat away with structural or heat sink elements is often a good option.

The free electrons that make metals good electrical conductors also make them good conductors of heat.

Coefficients of thermal expansion in structural elements subjected to temperature variation sometimes need to be considered.

Thermal insulators can be used to isolate components sensitive to temperature changes.

2.3.4 Materials Testing

Standard methods for measuring material properties have been developed. The two most important test methods are tensile testing and hardness testing.

2.3.4.1 Tensile Testing

Tensile test machines are common features in materials laboratories. Necked-down material samples are prepared and installed in the upper and lower parts of the machine. Deflection is imposed on the material while measuring the resulting load. Concrete has rather poor tensile properties, so concrete samples are tested in compression.

2.3.4.2 Hardness Testing

Hardness can be tested by measuring how much a standard point can be pushed into the material with a given static force. Two common types of hardness measures are called Brinnel hardness and Rockwell hardness.

2.3.5 Analyzing Designs

To verify that a design is sufficient for its intended purpose, analysis is performed. First we determine the worst-case loads that might be reasonably expected to be applied to the structure. Then the reactions are calculated and the internal loads determined. From the internal loads, stress at critical points is computed. If the maximum stress is less than the material capability (ultimate stress), then "safety margin" can be said to exist. Safety margin is the capability minus the expected load or stress. *Safety factor* is the ratio of the capability to the expected load or stress. For applications where the failure of a system or component could result in the loss of human life, a safety factor of at least five (5.0) should be used. For general robotics applications, a safety factor between 1.5 and 3.0 should be sufficient.

2.4 Structures: Beams, Columns, Frames, Struts

Structural systems are built of elements such as beams, columns, and struts.

2.4.1 Beams

Mud bricks are one of the oldest building materials, dating back to the dawn of civilization over 10 thousand years ago. In order to put doors and windows in mud brick houses, wooden beams were used to support the weight of the bricks above the openings.

Beams in buildings support vertical load due to weight on a floor or roof. Beams can be loaded with point loads or distributed loads, and can be simply supported, cantilevered, or supported in some combination of ways.

2.4.2 Columns

Columns are the vertical posts that hold up the beams in buildings. The monumental temples of ancient Egypt and Greece had stone columns to support stone beams. The vertical load in a column is a compressive load. A slender column will fail by *buckling*. Buckling failure of a column is rapid and catastrophic, so special attention must be paid to the analysis of slender columns. Leonard Euler was the first to analyze buckling in columns, and his classic formulas are in use today. The formula for critical buckling load is

$$P_{cr} / A = \pi^2 E/(l/k)^2$$

for a rounded end column where k is the radius of gyration of the cross section and A is the area of the cross section.

2.4.3 Moment Bearing Joints

As a rule, moment bearing joints should be avoided in designs where weight is critical. Where necessary, moment bearing joints can be formed by bolting, riveting, welding, and bonding.

2.4.4 Frame and Panel

Frame and panel construction in wood was invented by the ancient Egyptians. Frame and panel construction is very efficient and is commonly used in spacecraft where high strength and stiffness with light weight are design drivers.

2.4.5 Struts and Trusses

Trusses made from struts are among the most efficient of structures. Trusses are also commonly used in spacecraft design. Trusses also have the advantage that they are easy to analyze.

2.5 Deflection Analysis

In static analysis we treat structures as rigid bodies. A “rigid body” is an idealization that doesn’t exist in reality. Rigid body analysis is an example of abstraction to make analysis easier. However, in the real world, everything deflects under load. Saying that there is no absolutely rigid body is the same as saying that nothing is infinitely stiff.

Different materials have different stiffnesses. Young’s modulus (E).

2.5.1 Axial Deflection

When a truss is loaded, the internal loads are tension and compression in the struts. These axial loads cause axial deflections: the struts change length under load. These length changes accumulate throughout the truss structure causing overall deflection of the loaded interfaces.

2.5.2 Beam Deflection

Beams deflect primarily by bending. A straight beam will curve slightly under bending load. The amount of curvature is proportional to the moment load. For a cantilever beam under moment load, the curvature is constant throughout the beam and the formula for the deflection is:

$y = Mx^2/2EI$

For a cantilever beam with a point load at the end, the formula for deflection is:

$y = Fx2(x - 3l)/6EI$

The maximum deflection is:

$y_{max} = Fl^3/3EI$

2.5.3 Torsional Deflection

A torsionally loaded member deflects by twisting. Its deflection is angular. The formula for torsional deflection is:

$\theta = Tl/GJ$

Where G is the bulk modulus of the material.

3 Designing Structures and Mechanisms

Design for manufacturing (DFM) means considering fabrication and assembly techniques when making design decisions. Mechanisms contain moving parts and usually require maintaining close tolerances so that the mechanisms will function properly. Internal loads generated within mechanisms also need to be considered.

3.1 Fabrication of Parts

Mass production techniques and the economy of scale must be considered when designing for the consumer market. For special applications like robotics where one-of-a-kind systems are built, mass production techniques such as metal casting and injection molding of plastics can be ignored. Machining (material removal) and sheet metal cutting and bending are the two major fabrication techniques that apply to competitive robotics design.

Fabrication techniques should be kept in mind when selecting materials. Plain (mild) steel can be easily flame cut and welded, but machining of steel is much more difficult than machining of aluminum. Plain steel has poor corrosion resistance, but for indoor use, that’s usually not a problem. Aluminum and steel have about the same strength-to-weight ratio, but aluminum generally has higher stiffness-to-weight, due to the use of thicker sections for similar weight.

3.2 Assemblies, Bolted Joints, Welding

For mass produced items, assemblies are sometimes designed to snap together. Snap assemblies can be difficult to repair, however. Light duty assemblies are sometimes bonded. Specialized applications of graphite and other composites used in aircraft and space structures are usually bonded. Welding and bolting are two assembly methods suitable for competitive robotics.

3.2.1 Threaded Fasteners

A *screw* is intended to be tightened by applying torque to its head. A bolt is intended to be tightened by applying torque to a *nut*. A *stud* resembles a threaded rod. One end is inserted in a tapped hole, the other end receives a nut. A high preload in bolts improves both fatigue resistance and the locking effect. If a properly installed bolt does not fail during tightening, it never will fail.

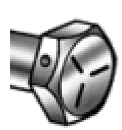


Figure 10: A hex-head of a bolt. The three radial dashes indicate that this is an SAE grade 5 bolt with a proof strength of 85 ksi.

3.3 Friction

Friction is a force that resists sliding motion and is always in a direction opposite to the sliding motion or applied force. The amount of friction is proportional to the force holding two surfaces in contact and depends on the materials of the two surface. This is called Coulomb friction, after the scientist who developed this mathematical relationship. Coulomb friction does not depend on the area of contact, only the force pressing the two surfaces together. For example, metal sliding on ice has a very low coefficient of friction (μ), and rubber on asphalt has a fairly high coefficient of friction. If you are driving a car on ice, getting wider tires won't help you. You need to change the properties of the tires such as putting metal studs in them or adding chains.

The expression for frictional force, F, is:

$F = \mu P$

where P is the force of compression between the two materials and μ is the coefficient of friction.

The coefficient of friction has two regimes, static and dynamic. Generally, the coefficient of static friction (μ_s) is higher than the coefficient of dynamic (or sliding) friction (μ_d). That's why a car will stop faster if the brakes are controlled so that the tires don't slip.

Here are some typical coefficients of friction

Material	μ_s	μ_d
Rubber on asphalt (dry)	1.0	0.9
Rubber on asphalt (wet)	0.6	0.5
Steel on aluminum (dry)	0.3	0.2
Steel on aluminum (oily)	0.15	0.1

Antifriction bearings (ball or roller bearings) have equivalent coefficients of friction in the range of 0.002.

3.4 Mechanical Advantage: Levers and Linkages

The ancient Greek engineer Archimedes said "give me a fulcrum and a lever long enough and I will move the Earth." Mechanical advantage can be either an advantage in speed (sacrificing force) or an advantage in force (sacrificing speed). Mechanical advantage is analyzed using the same techniques used in static structural analysis.

A "linkage" is a combination of levers connected with links. Linkages generally provide non-linear kinematics and are difficult to analyze manually, but a number of computer applications are available to help with design and analysis of linkages.

3.5 Rotary Mechanisms: Gears, Chains, Belts, and Pulleys

James Watt built the first steam engine that converted linear motion (of a piston) to rotary motion (of a fly wheel) using a linkage (connecting rod between the piston and fly wheel). Rotary motion provides convenient power transmission through shafts, gears, belts, pulleys, etc. Watt's invention revolutionized manufacturing because a single large engine could provide power throughout a factory using those transmission techniques. Previously, factories had to be situated on rivers to take advantage of power from a water wheel.

Gears provide the most efficient power transmission with the greatest power density (power to weight ratio). Most automotive internal combustion engines utilize gears for the transmission of power from the engine to the wheels. However, gears require precision manufacturing to maintain critical dimensions of gear mesh. Adjustable gear mesh can be provided, but in practice, gear adjustment is not generally used.

Roller chains and sprockets provide light duty flexible power transmission. As chains wear, however, they change in length, so some adjustment mechanism is usually provided.

Drive belts come in many types. Flat belts are still used in many industrial applications for factory power transmission. Plain V-belts are common in automotive applications for driving auxiliary equipment such as air conditioning pumps and alternators. Notched V-belts are sometimes used for driving cam shafts. V-belts require sufficient tension to provide friction so the belts don't slip. This tension imposes normal (radial) load on bearings that are higher than for gears or chains.

4 Electricity

4.1 Basic Terms

4.1.1 Voltage: Force that causes Charge to move

- Count Alessandro Volta – Italian physicist
- Studied batteries and static electricity in late 1700s

4.1.2 Battery: Device with constant voltage, E

4.1.3 Charge: Property of quantity of ions or electrons

- Unit: Coulomb = 6.24 * 10^19 electrons
- Charles August Coulomb - French physicist

4.1.4 Current: Speed of charge motion

- Andre Marie Ampere – French physicist
- One ampere current = one coulomb per second

4.1.5 Resistor: Electrical device that slows flow of charge

- George Simon Ohm – German physicist

4.1.6 Power: Rate of electrical energy flow

- James Watt – Scottish Engineer

4.2 Schematic Symbols

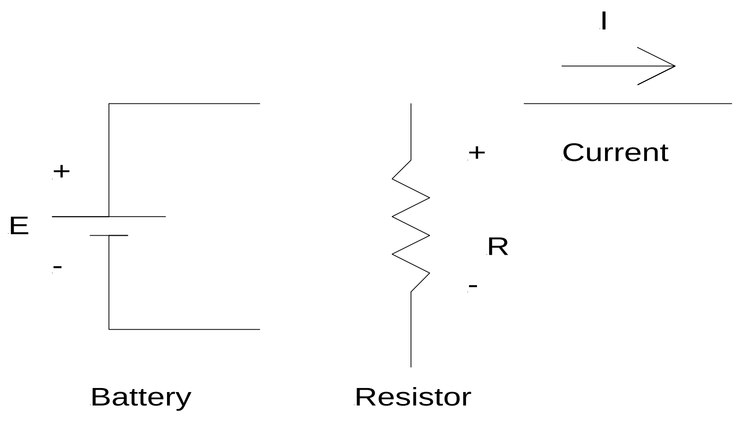


Figure 11: Schematic symbols.

4.3 Example Circuit

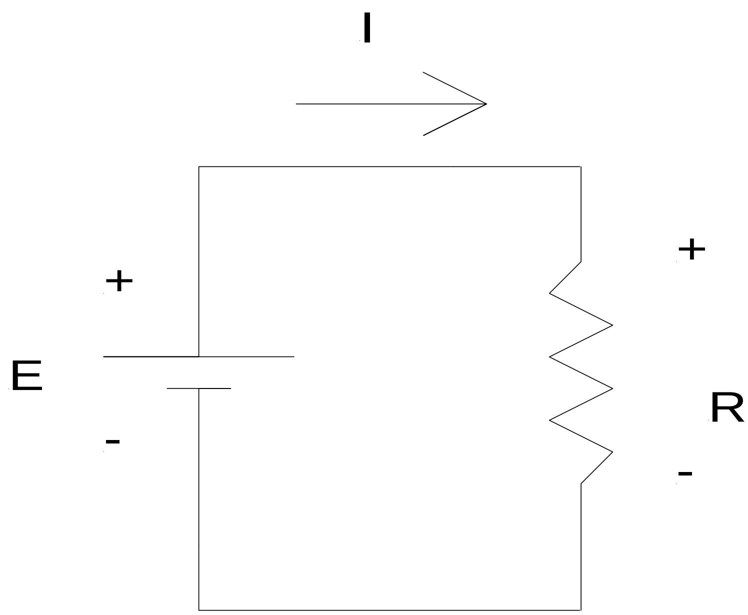


Figure 12: Example circuit.

4.4 Ohm's Law

Ohm's law states that the voltage, V, across a resistor is equal to the current, I, times the resistance, R.

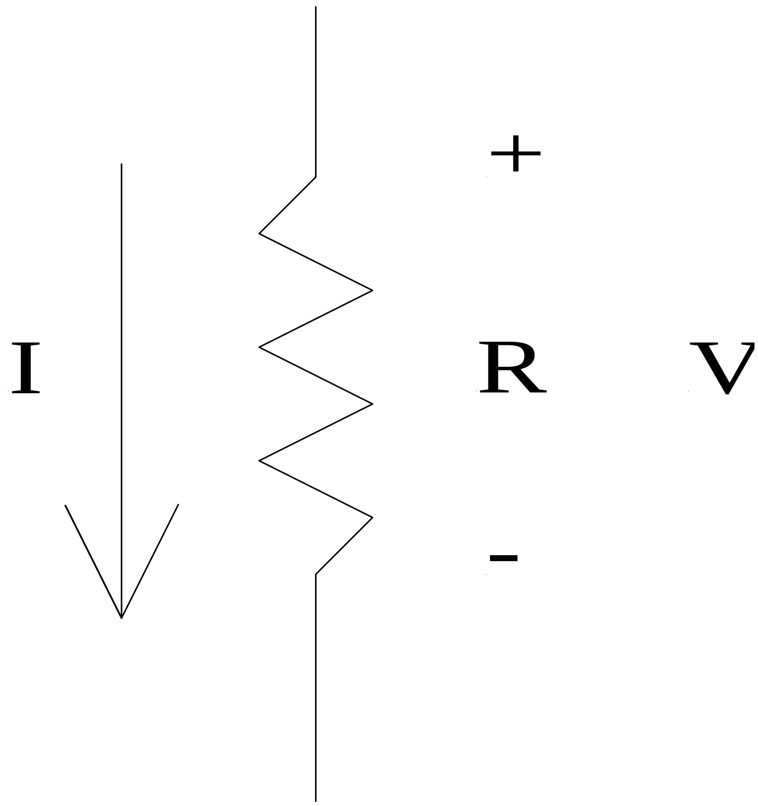


Figure 13: Ohm's law.

4.5 Measurements

4.5.1 Put the Leads in proper socket

- Use A or mA for current measurement
- Use Vdc for direct current (DC) voltage measurements

4.5.2 Put the pointer on appropriate measurement scale

- Principles of operation
- The current flows through the meter during current measurement
- The meter is approximately an open circuit during voltage measurement

4.5.3 Caution

- Never put the leads across a voltage during a resistance measurement

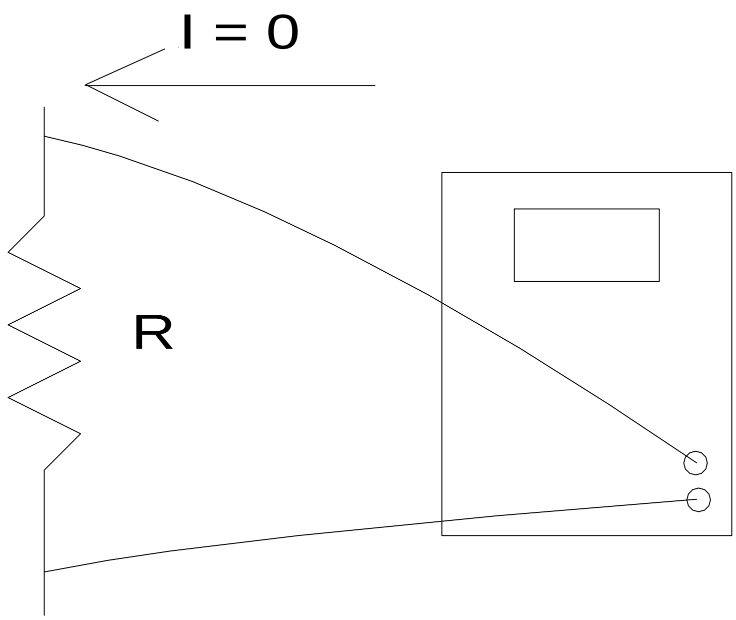


Figure 14: Resistance measurement.

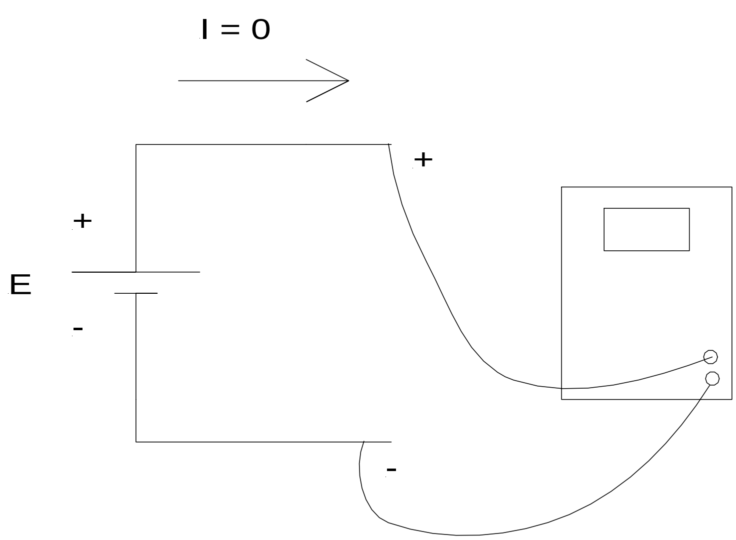


Figure 15: Voltage measurement.

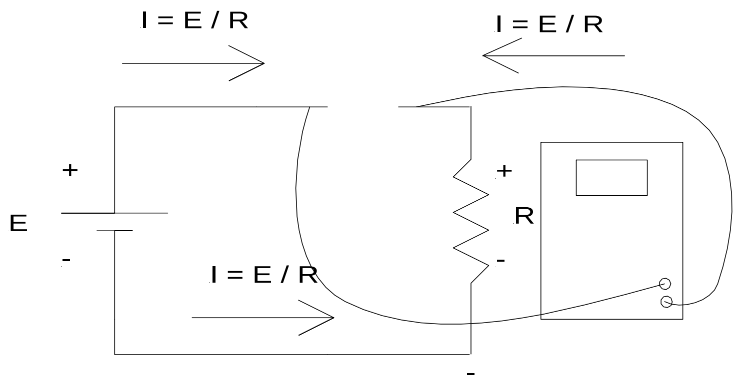


Figure 16: Current measurement.

4.6 Examples

- Let battery voltage equal 12 volts
- Let resistance of resistor equal 30 ohms
    - Current = E / R = 12 / 30 = 0.40 amperes = 400 ma
- Let resistance of resistor equal 16 ohms
    - Current = E / R = 12 / 16 = 0.75 amperes = 750 ma
- Note that with less resistance there is more current

4.7 Expanded Example

- Put two resistors in series
    - The same current, I, flows through each resistor
- Voltage across first resistor V1 = I * R1
- Voltage across second resistor V2 = I * R2
- Total voltage I * R1 + I * R2 = I * (R1 + R2)
- Therefore: Battery voltage, E = I * (R1 + R2)
- And, I = E / (R1 + R2)
- Resistors in series add together.

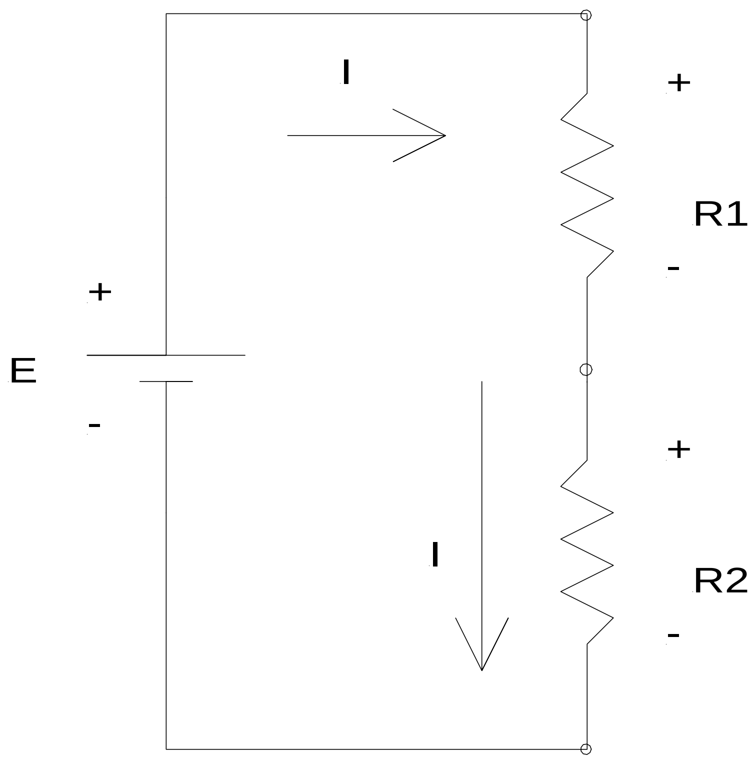


Figure 17: Expanded example schematic.

4.8 Voltage Divider

Voltage is “divided” into several voltage “drops” over a series of resistors. The voltage drop for a resistor is proportional to the resistance of the resistor divided by the total series resistance.

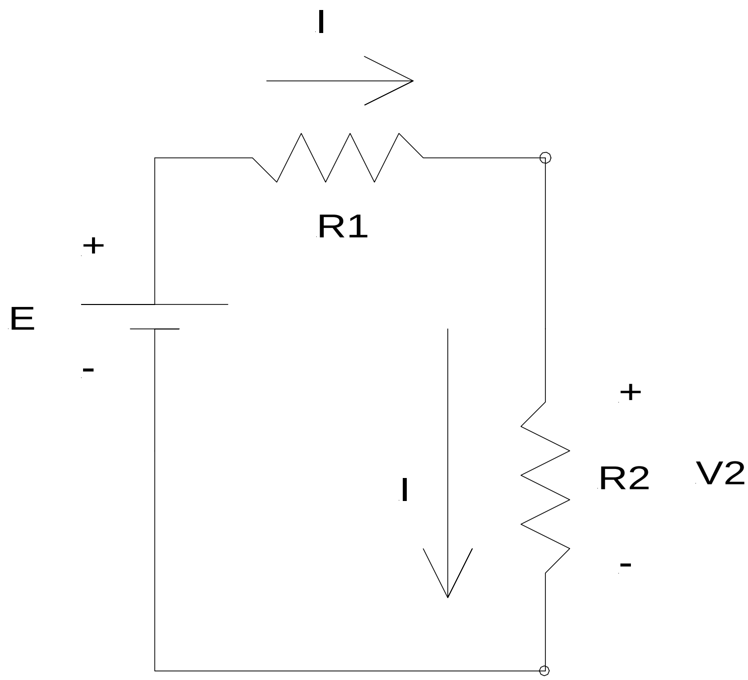


Figure 18: Voltage divider schematic.

- The voltage divider is a common circuit.
- The output voltage of this circuit is V2.
- I = E / (R1 + R2)
- V2 = I * R2
- V2 = (R2 / (R1 + R2)) * E

4.9 Electrical Power

Power is the rate (in time) at which energy is transmitted or expended.

- DC power is equal to the voltage times the current
- The unit of power is the watt
- P = V * I
- The power delivered to a resistor is converted to heat

5 Motive Power

Mobility is an important feature of a robot. Without the ability to move quickly, the tasks a robot can perform in a given time are limited. An agile robot is a competitive robot. “Power” is the time rate at which work is done or energy emitted or transferred.

5.1 Mass and Weight

Mass is defined as the measure of how much matter an object or body contains; i.e. the total number of subatomic particles (electrons, protons, neutrons) in the object. Weight is the mass times the pull of gravity (attractive force between bodies possessing mass).

5.2 Newton's Laws of Motion

Sir Isaac Newton (1642-1727) formulated the laws of motion and gravitation. The ancients believed that being at rest was the “natural state” of objects and that if all forces were removed from an object, it would come to rest. Galileo, however, observed that if friction forces were reduced to zero, an object would continue in motion. Newton codified this observation in his famous equation of classical mechanics:

$\mathbf{F} = m\mathbf{a}$

That is, force equals mass times acceleration. The bold letters in the above equation indicate that they represent vectors; that the equation applies in three dimensions. If zero force is applied to a finite mass, its acceleration is also zero: an object in motion will remain in motion.

5.3 Work, Heat, Energy, Current, Voltage, and Power

Work is a directly usable form of energy. Heat is a waste form of energy. Some machines can take heat and turn it into work. Such a machine is called a *heat engine*. Examples of heat engines are gas fired steam power plants and aircraft propulsion turbofans. Any heat engine requires both a heat source and a cold sink. As long as a sink is available to reject heat, work and power can be extracted from heat. Dean Kamien’s latest project is a sterling cycle engine driven water purification system that produces rotary motion to generate electricity.

Energy is measured in terms of force and distance. Pushing through a distance requires energy. In the USCS (United States Customary System), energy is measured in foot-pounds. In the SI (System International), energy is measured in Newton-meters, also called joules.

Power is the rate of doing work. One horsepower is 550 ft-lb / sec. One joule per second equals one watt.

James Prescott Joule established the law of conservation of energy in 1837, quantifying experimentally the relation between work and heat, and determining that one calorie of heat is equivalent to 4.18 Joules. German Physicist Herman Von Helmholtz produced the mathematical proof for conservation of energy derived from the Newtonian laws in 1847.

The First Law of Thermodynamics: A change of internal energy results from heat absorbed by a system plus work done on a system.

Energy is a measure of how long power is sustained, i.e. how much work can be done. (see Energy Defined, below)

Power is a measure of how quickly work can be done. P=dW/dt, where dW is the change in work and dt is the change in time (time interval).

Electrical energy is a measure of average electrical power during a period of time:

Joule = Watt * Sec

The simplest case is a resistor with a DC voltage source applied, where P = VI (= I^2*R = E^2/R) watts. A resistor stores no energy. The resistor function is to impede electrical flow which enables control in, and of, electrical circuits.

Power in watts = Volts * Amperes, = Amps squared * Resistance in ohms, Volts squared/ Resistance in ohms.

Energy in joules is obtained by multiplying average Power by time in seconds resulting in watt*seconds = Joules.

5.4 Inductance and Capacitance

More complex are the two reactive components; the inductor (i.e. a coil of wire such as in a motor) and the capacitor.

This complexity is due to the fact that these components store energy. The inductor stores energy in a transitory electromagnetic field. Once an inductor is charged it must immediately be returned (discharged) when the source of energy is removed. The applied energy is transitory (pulse voltage = current ramp); otherwise a condition known as saturation occurs which replaces the normally efficient inductive reactance charge mechanism (energy is recoverable) with the inefficient heat generation (energy is essentially non-recoverable i.e. lost as heat) in the DC resistance of the wire An increase in trapped heat increases wire resistance, compounding losses.

Inductive reactance is a special type of “resistance” characterized by inductor voltage and current being out of phase (non-zero phase). In an ideal inductor (L), the voltage leads the current by 90 degrees. Energy in an inductor = (LI^2)/2 joules (stored briefly in an electromagnetic field such as in a motor).

Energy in an inductor = (electromagnetic inductance in Henrys * amperes squared) / 2 Joules.

Capacitive reactance is a special type of “resistance” characterized by capacitor voltage and current being out of phase. (non-zero phase). In an ideal capacitor (C), the voltage lags the current by 90 degrees. Energy in an capacitor = (CV^2)/2 joules (stored indefinitely in an electrostatic field, i.e. two conductors separated by an insulator)

Energy in an capacitor = (electrostatic capacitance in Farads * volts squared) / 2 Joules.

Energy stored in an electrostatic field is complementary (directly opposite) to that stored electromagnetically in an inductor.

In an alternating current (AC) excited circuit composed of LC there is one frequency called *resonance* where XL=Xc. they cancel, circuit reactance = 0.

No energy is dissipated/consumed/wasted in reactance of electromagnetic inductance-coils or electrostatic capacitance-capacitors.

Energy is dissipated *only* in the DC resistance associated with producing a reactance (i.e. DC contact and wire resistance of the coil or imperfection of the insulator of a capacitance and contact (series) resistance to its plates).

5.5 Energy Defined

(This section is reproduced from an on-line encyclopedia. Fair use for educational purposes.)

Energy is the capacity of matter to perform work as the result of its motion or its position in relation to forces acting on it. Energy associated with motion is known as kinetic energy (1/2 mass * velocity^2), and energy related to position is called potential energy (Force * distance). Thus, a swinging pendulum has maximum potential energy at the terminal points; at all intermediate positions it has both kinetic and potential energy in varying proportions (except at its nadir, where the potential energy is zero). Energy exists in various forms, including mechanical, thermal, chemical, electrical, radiant, and atomic.

All forms of energy are interconvertible by appropriate processes. In the process of transformation either kinetic or potential energy may be lost or gained, but the sum total of the two remains always the same. A weight suspended from a cord has potential energy due to its position, inasmuch as it can perform work in the process of falling. An electric battery has potential energy in chemical form. A piece of magnesium has potential energy stored in chemical form that is expended in the form of heat and light if the magnesium is ignited. If a gun is fired, the potential energy of the gunpowder is transformed into the kinetic energy of the moving projectile. The kinetic mechanical energy of the moving rotor of a dynamo (motor) is changed into kinetic electrical energy by electromagnetic induction.

All forms of energy tend to be transformed into heat, which is the most transient form of energy. In mechanical devices energy not expended in useful work is dissipated in frictional heat, and losses in electrical circuits are largely heat losses. Empirical observation in the 19th century led to the conclusion that although energy can be transformed, it cannot be created or destroyed. This concept, known as the conservation of energy, constitutes one of the basic principles of classical mechanics. The principle, along with the parallel principle of conservation of matter, holds true only for phenomena involving velocities that are small compared with the velocity of light. At higher velocities close to that of light, as in nuclear reactions, energy and matter are interconvertible. In modern physics the two concepts, the conservation of energy and of mass, are thus unified.

5.6 Motors and Generators

Electric motors and generators are devices used to convert energy forms. Generators convert mechanical energy into electrical energy. Motors convert electrical energy into mechanical, specifically rotary, which is an especially useful form, implementing rotational power on demand with infinite (continuous) control. All permanent magnet (PM) motors (including those in the FIRST robotics kit) are both motors and generators. In operation the robot drive motor functions as a generator when the robot is pushed from an outside power source faster than the speed control set-point, or the inertia (kinetic energy) of the robot after the operator removes power. This becomes an important electrical design issue for the motor controller (V884) and for understanding the dynamics of the electric motor controlled robot drive.

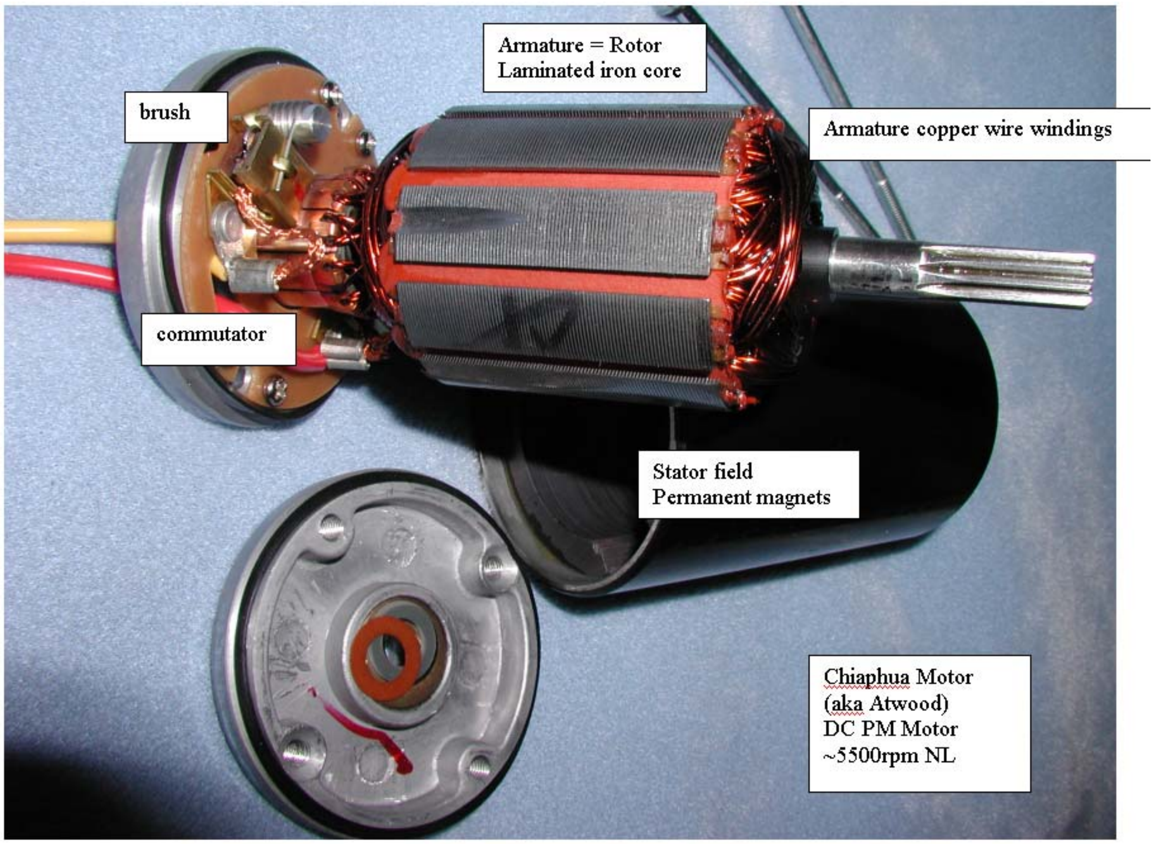


Figure 19: A typical permanent magnet DC electric motor, disassembled.

In electric motors and generators, heat is waste and needs to be removed to avoid performance degradation and damage to the armature or other parts. Electric motors run at their optimal design levels at a constant speed are about 90% efficient. For every 100 watts delivered to a motor, about 10 watts are turned into heat and 90 watts go into useful work. In competitive robot applications, motors are run at variable speeds infrequently passing through the optimal range. In such cases, efficiency can drop well below 50%, such as in frequent accelerating from a standing start, (drawing up to 300 amperes to start movement depending on the load resistance = weight/friction, accelerate rapidly or push or be pushed by, another robot), full throttle forward to reverse and vice-versa, which happens frequently in competitive robotics.

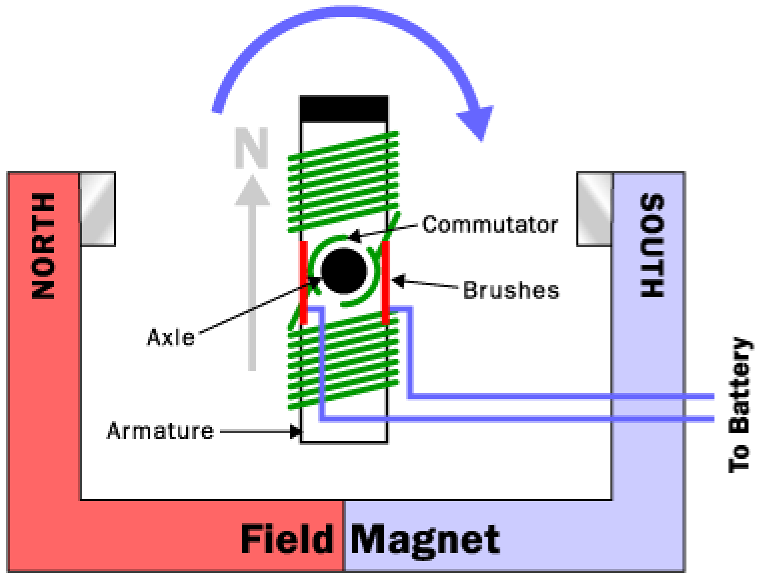


Figure 20: Schematic of a simple permanent magnet DC motor

5.7 Principles of Operation

Magnetic poles: likes attract, opposites repel; convention: one end we call the *north* pole the opposite end the *south* pole

Electromagnetic induction was discovered by British scientist Michael Faraday in 1831. If an electrical conductor is moved through a magnetic field (or a stationary conductor is in the proximity of a conductor of varying field (also known as alternating current, or AC), a current is induced in that conductor. The converse of this principle is that of electromagnetic reaction first observed by French physicist Andre Marie Ampere in 1820. If a current is passed though a conductor in a magnetic field, a mechanical force is produced.

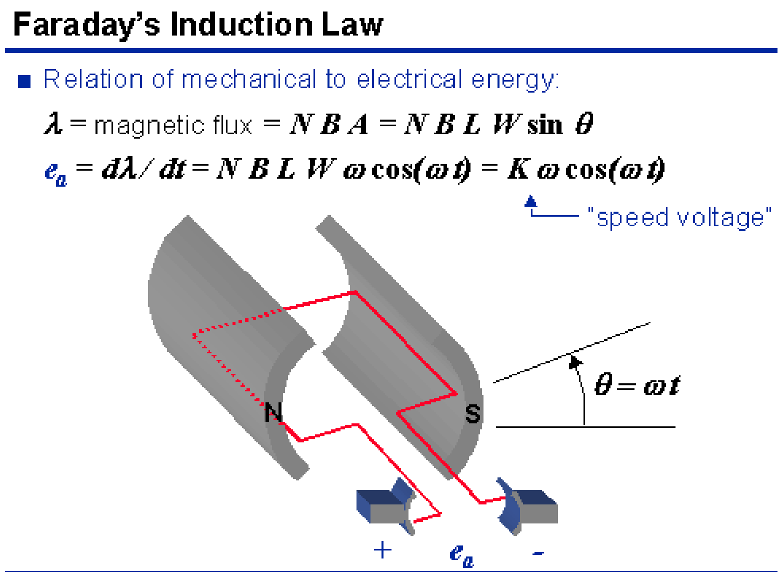


Figure 21: Faraday's induction law.

Both motors and generators consist of two basic entities: the field magnet called a stator (non-moving) and the armature called the rotor. Magnetic material (such as iron) is used in the rotor and the stator concentrates magnetic field flux density. Field laminations (thin insulated layers of iron) decrease undesirable (heat producing) eddy current losses (from same induction that causes the desirable rotation).

Motor function: the polarity of the rotor relative to the field must constantly change else rotational motion is not maintained. Motor operation is inherently an alternating current (AC) function so special consideration is required for DC operation. The *commutator* provides a dynamic alternating switching action using hemispherical copper conductors connected to the rotor coil.

Stationary, often spring loaded, brushes complete the external circuit. (usually composed of a carbon copper mixture to which the battery is connected). The commutator changes DC to AC to effect rotation in a motor. In a generator the commutator changes native rotor AC to DC.

If AC output power is desired slip rings are substituted consisting of two side by side insulated rings with one brush on each. AC is the natural result of a vector tracing out a 360 degree circle at a fixed velocity yielding two smoothly alternating half cycle amplitudes, positive and negative. The frequency of the sine wave thus produced is the reciprocal of its period (time for one full cycle). Generators may rotate the permanent magnet and extract power from the stator to avoid slip ring I^2R loss.

A DC brushless motor design avoids physical commutation by using electronic Hall effect sensing and field effect transistor (FET) drivers to increase efficiency (because of less friction and potentially less voltage drop). Brushless motors are commonly used in fans with high torque requirements that complicate brushless designs.

A two-pole motor is conceptually simple; however, it suffers from start-up problems due to the geometry of having only two opposing states. Practical motors employ at least three poles to assure self starting and smooth rotation.

5.7.1 DC Motor Operation

The spinning of the armature within a magnetic field induces a voltage in the armature windings. This induced voltage is opposite in direction to the external voltage applied to the armature, and hence is called back voltage or counter electromotive force (CEMF). As the motor rotates more rapidly, the back voltage rises until it is almost equal to the applied voltage. The current is then small, and the speed of the motor will remain constant as long as the motor is not under load and is performing no mechanical work except that required to turn the armature.

Under load the armature turns more slowly, reducing the back voltage and permitting a larger current to flow in the armature. The motor is thus able to receive more electric power from the source supplying it and to do more mechanical work.

Because the speed of rotation controls the flow of current in the armature, special devices must be used for starting DC motors. When the armature is at rest, it has virtually no resistance, and if the normal working voltage is applied, a large current will flow, which may damage the commutator or the armature windings. The usual means of preventing such damage is the use of a starting resistance in series with the armature to lower the current until the motor begins to develop an adequate CEMF. As the motor picks up speed, the resistance is gradually reduced, either manually or automatically.

Alternatively, the FIRST controller uses PWM (pulse width modulation, i.e. varying duty cycle of a fixed frequency square wave) to gradually increase and decrease average current to the armature winding via joystick control to the Victor V884 switched H-bridge motor driver.

The speed at which a DC motor operates depends on the strength of the magnetic field acting on the armature, as well as current provided to the armature. The stronger the field, the slower is the rate of rotation needed to generate a back voltage large enough to counteract the applied voltage. For this reason the speed of DC motors may also be controlled by varying the field winding current assuming the field is an electromagnet All current FIRST “power” motors use PM fields (and consequently have constant field strength).

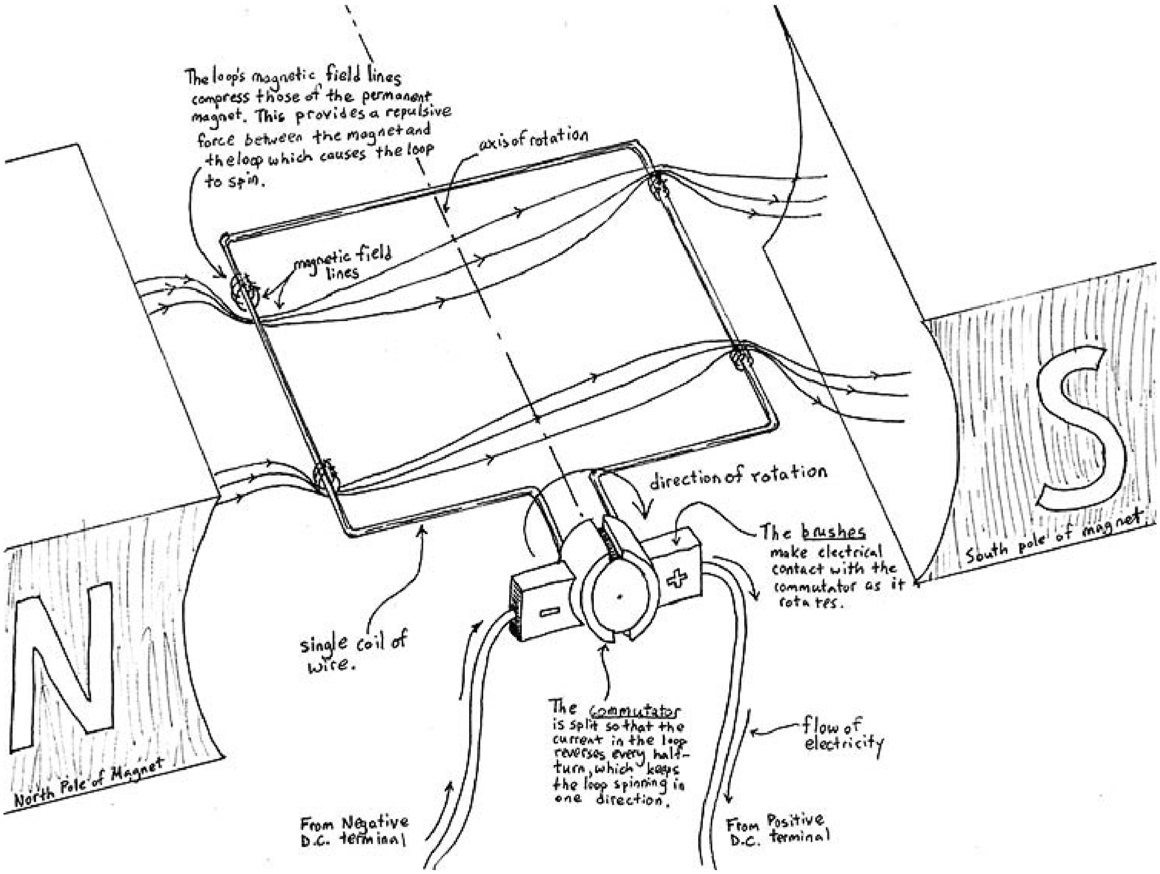


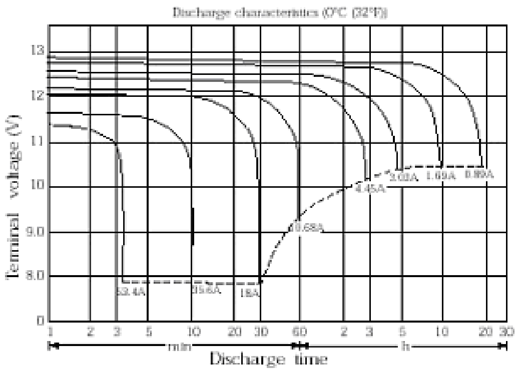
Figure 22: DC motor operation.

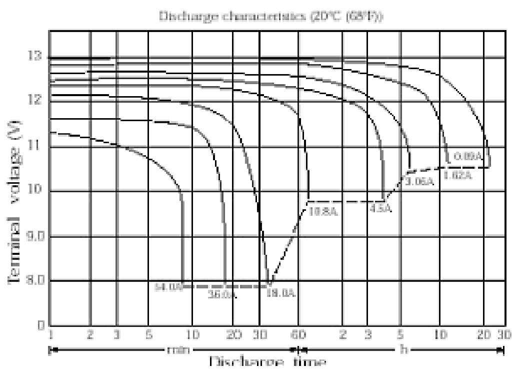
5.7.2 DC Motor Control

Motor power comes from a lead-acid sealed battery. Specifications for an 18 ampere-hour battery are shown below in Figure 23.

LEAD-ACID BATTERY

ES18-12
12Volt 18AH





TYPE		ES18-12
NOMINAL VOLTAGE		12 Volt
NOMINAL CAPACITY		18 AH / 20 HR
DIMENSIONS	HEIGHT	167 mm
	LENGTH	181 mm
	WIDTH	76 mm
WEIGHT		APPROX. 6.1 KGS
CAPACITY	20HR 0.90A	18.0 AH
	5HR 3.06A	15.3 AH
	1HR 10.80A	10.8 AH
	1C 18.0A	9.0 AH
INTERNAL RESISTANCE		APPROX. 15mΩ
MAX. DISCHARGE CURRENT		230 A (5 SEC.)
CHARGING VOLTAGE 20°C (68°F)	STANDBY USE	13.5-13.8V (-20mV/°C)
	CYCLE USE	14.4-15.0V (-30mV/°C)
MAX. CHARGE CURRENT		5.34 A

Figure 23: Battery performance.

When a motor starts from a standing start, there is a high spike in motor current, as shown in the oscilloscope photo shown below in Figure 24.

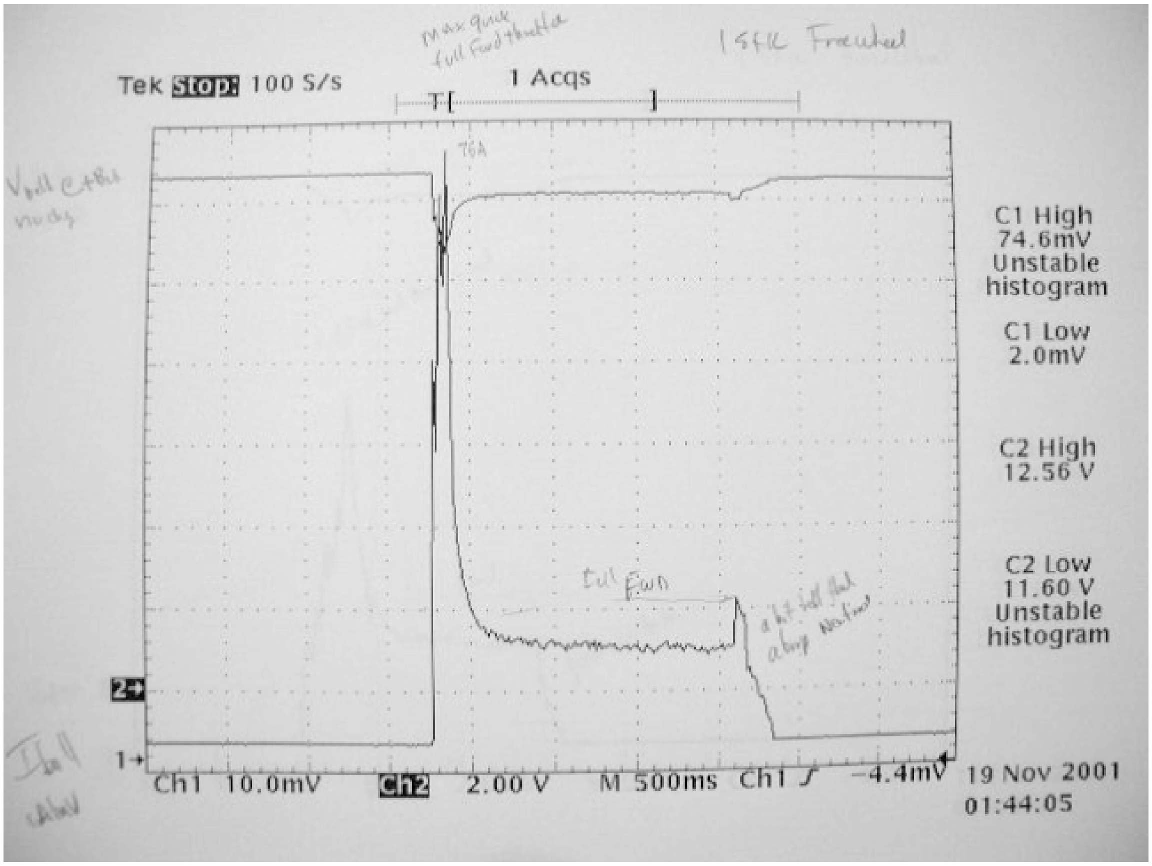


Figure 24: 2001 National Champ's PWM oscilloscope photo.

The direction the motor turns is controlled by the Victor 884 "H" bridge as shown in the schematic diagrams below in Figure 25. The switches shown are notional, and in the actual Victor 884 electronic device are implemented with power transistors.

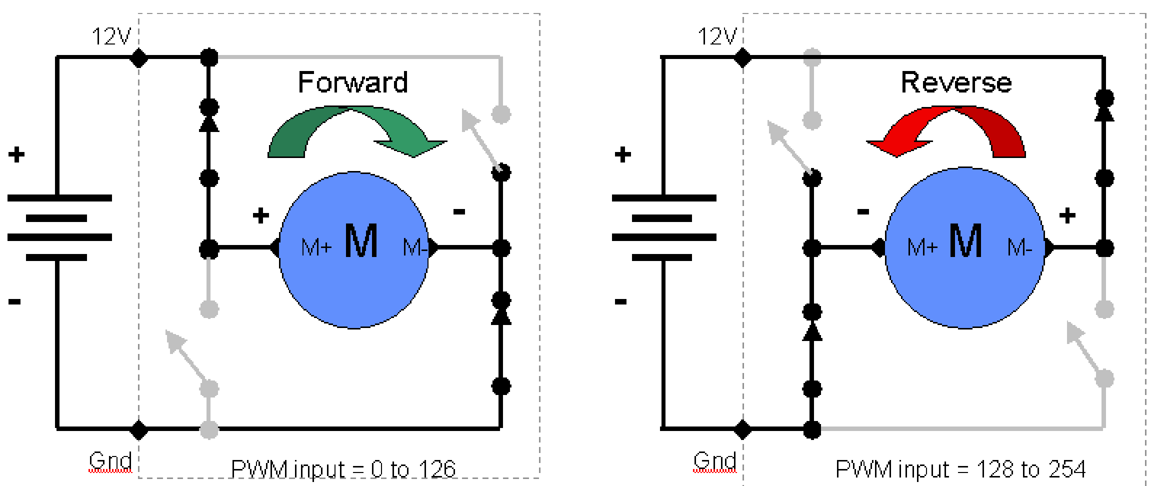


Figure 25: Victor 884 and Spike H-bridge internal four switch operation; two diagonal switch closures determine motor direction.

Coasting and braking are effected as shown below in Figure 26. For coasting, the motor circuit is opened completely. For motor braking, a short circuit is made, leading to counter EMF to produce braking torque.

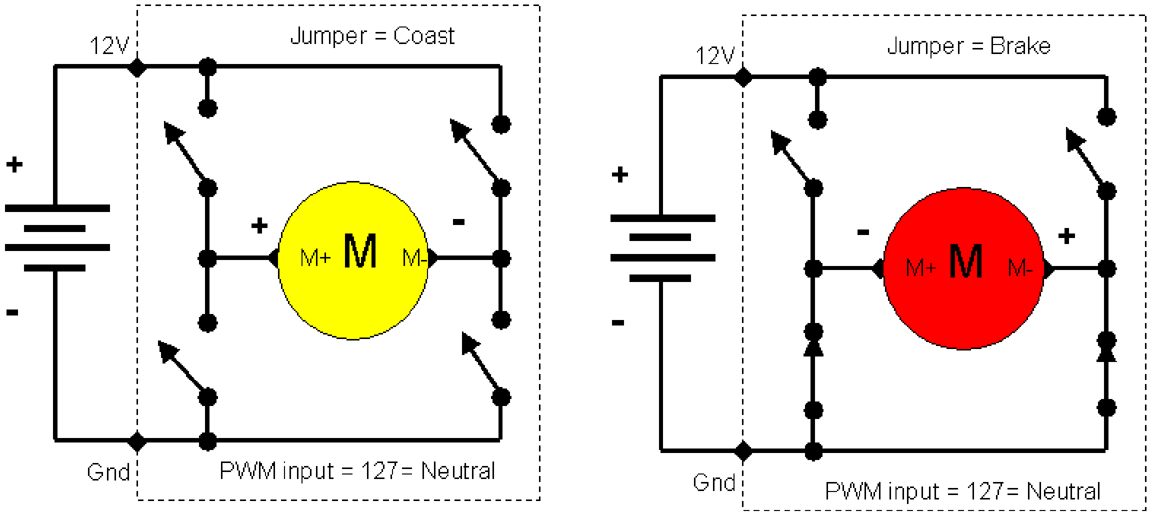


Figure 26: Victor 884 and Spike H-bridge internal four switch operation; joystick to neutral, effect of jumper: coast and brake.

To measure motor current, the voltage drop across a piece of wire can be measured, as shown below in Figure 27.

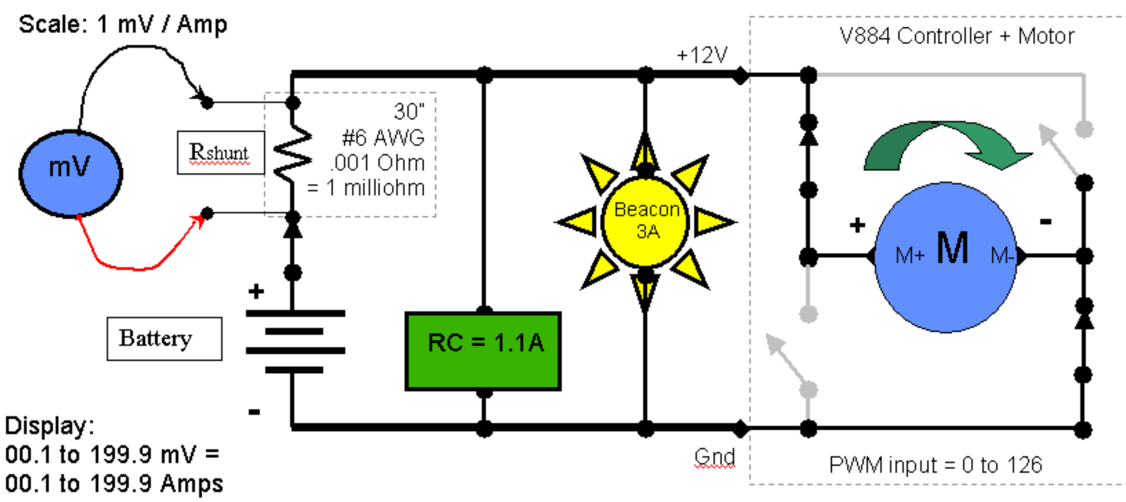


Figure 27: Using 30" length #6 AWG to measure robot current method: DMM + Ohms law I = V/R (Rshunt = high current low ohm shunt).

Alternatively, the circuit breaker voltage drop can be measured, as shown below in Figure 28:

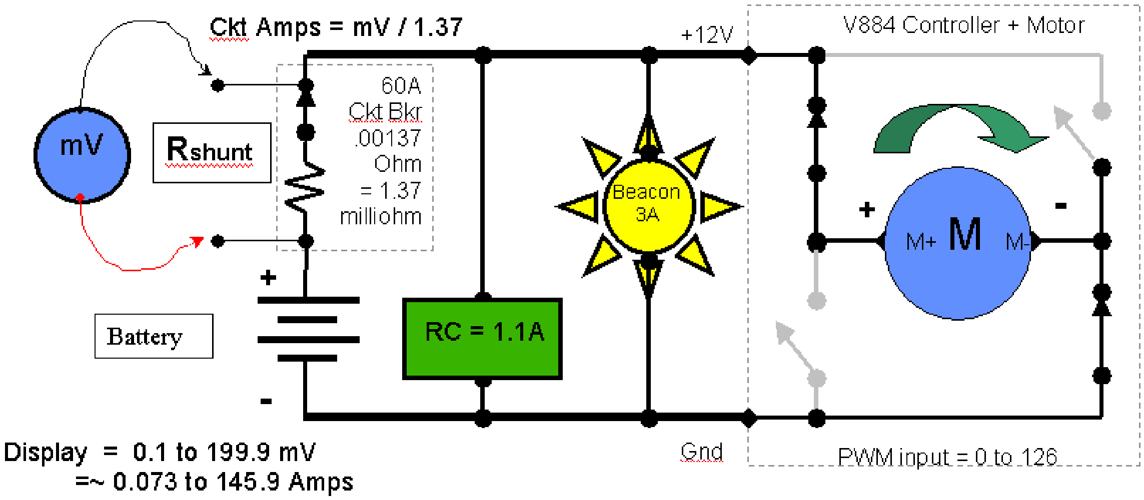


Figure 28: Using 60A circuit breaker to measure robot current method: DMM + Ohms law I = V/R (Rshunt = high current low ohm shunt).

Motor current can be controlled with the on-board controller by using the voltage drop in a sensing algorithm.

5.8 Torque and Speed

Motor Characteristics:

- Stall torque (shaft pinned from motion while briefly applying power)
- Stall current (shaft pinned from motion while briefly applying power)
- Free speed (no load RPM while applying power)
- Free current (no load RPM while applying power)

All the above are proportional to voltage.

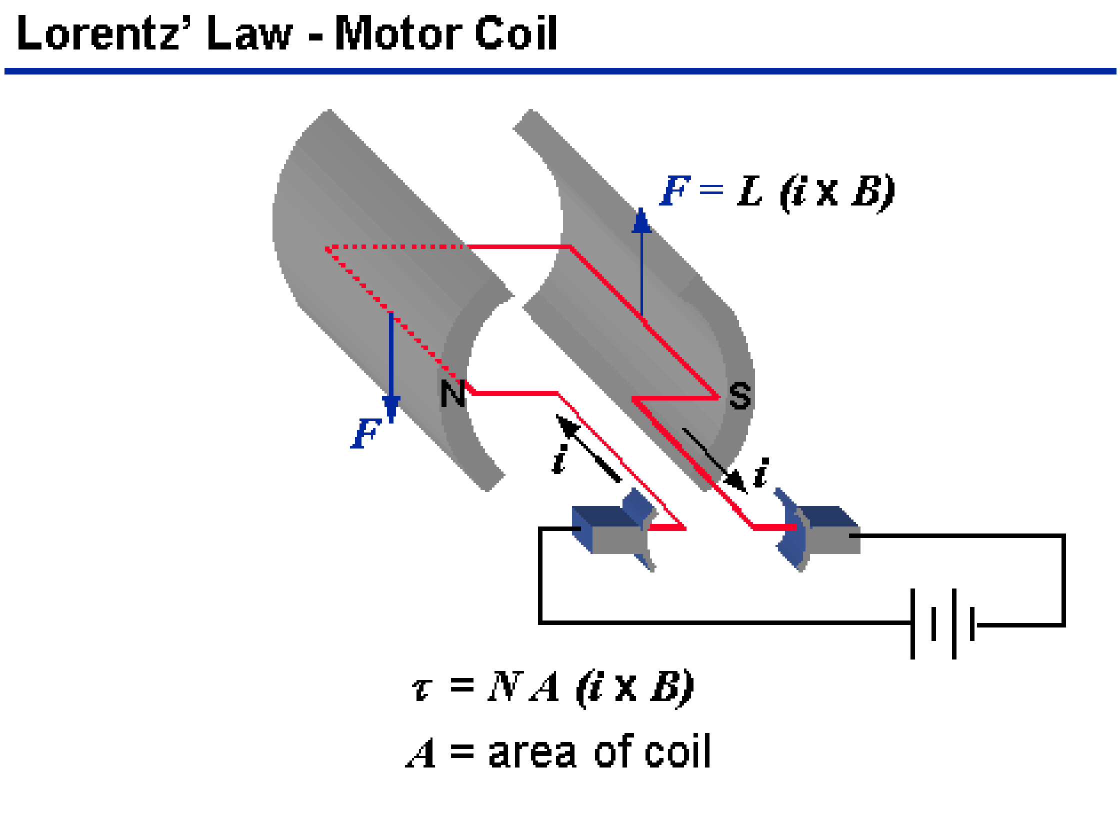


Figure 29: Lorentz' law.

DC motor torque and speed characteristics are shown below in Figure 30:

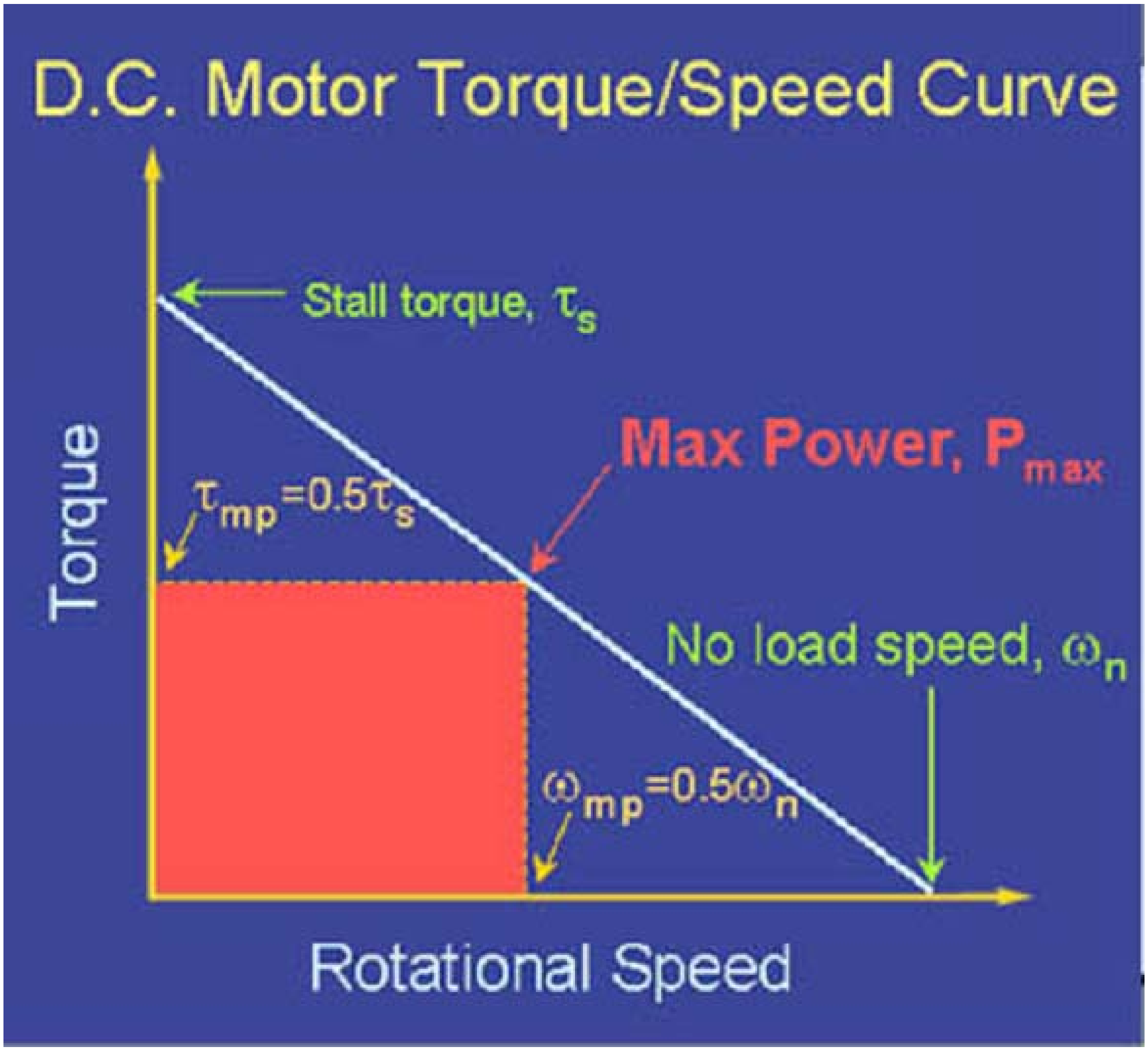


Figure 30: DC motor torque vs. speed curve.

5.9 Motive Power Guidelines

These guidelines were developed by Dale Hall based on observations of FIRST competitions since 1999:

- Require reliable mechanical hardware, electrical connections, and strategy
- The team relies on the robot. If it breaks down the team is unhappy, other teams notice, and you go home early.
- Loose structure or loose crimp connectors mean the motor might fail with 40 seconds remaining, or worse, a completely dead robot at the start of the match.
- Drivers grow and learn from errors and continually improve, know the rules and make good quick decisions.
- All wiring, RC PWM and relay pins and electronics should be protected from inadvertent pinches and impacts.

- Use simple one-piece protective cover with 4-point attachment for easy and quick access and repair of the robot.
- KISS: keep it supremely simple. Design complexity causes schedule slips.
- Successful teams do a thorough job on a few things (opposed to tackling lots of things).
- Keep the center of gravity (CG) Low. Secure the battery well, it’s the heaviest part and the robot dies without it.
- Use default the switch or keep downward compatibility: drive motors, basic function.
- Use great care around RC PWM, relay, and switch jumper pins: they break easily, defeat compatibility.
- Drive: use powerful motors and gearboxes and chain drive to all wheels. Repeat: develop reliable all-wheel chain drive with powerful motors: This is the single most important design on the robot.

5.10 Drive Trains

The drive train transmits motive power from the DC motors to the wheels or treads. The torque available from the motors must be increased, usually by a spur gear transmission, and then must be applied to the wheels, usually by a roller chain and sprocket system.

5.10.1 Mechanical Advantage

Levers and ramps provide mechanical advantage, which works by trading force for distance or speed. Gears apply leverage. Worms and screws apply the ramp principle. Figure 31, below, shows a double reduction hand winch. The crank turns a pinion gear that drives a spur gear. On the other side of the driven spur is another pinion which drives the winch spur (red) that turns the winch drum, winding a cable. Very high cable loads can be achieved with a double reduction hand winch (thousands of pounds), but the speed is very slow (up to a hundred crank turns per drum turn).

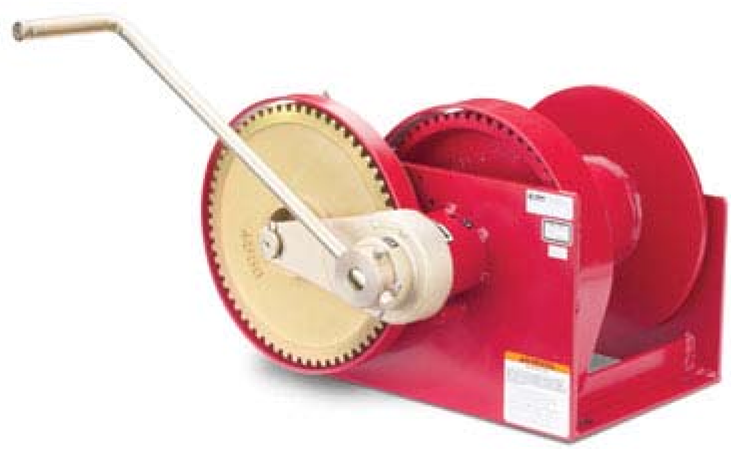


Figure 31: Double reduction hand winch.

Winches, powered by electric motors, are common design features in FRC robots. Speed of wind-up (inches per second) and winding force (pounds) are the design trade-offs in selecting the drive ratio.

5.10.2 Gears and Transmissions

The robot wheels turn much slower (and have much higher torque) than the motors that power them. Therefore, speed reduction and torque increase are obtained from transmission mechanisms.

Figure 32: Spur gears. The smaller gear is called a pinion.

Spur gears are among the most common transmission elements and are the most efficient, losing only approximately 2% to 5% of the energy per gear stage. For double reduction, the pinion drives the spur which drives another pinion that drives the final spur. Triple reduction can be used in some applications, but is not likely to be encountered in FRC robots. Most transmissions are used as speed reducers. Speed increaser applications are rare (but often spectacular, as in the case of the trebuchet).

Worm drives give a high speed reduction with attendant torque increase in a single stage. Worm drives necessarily use a cross-axis configuration.

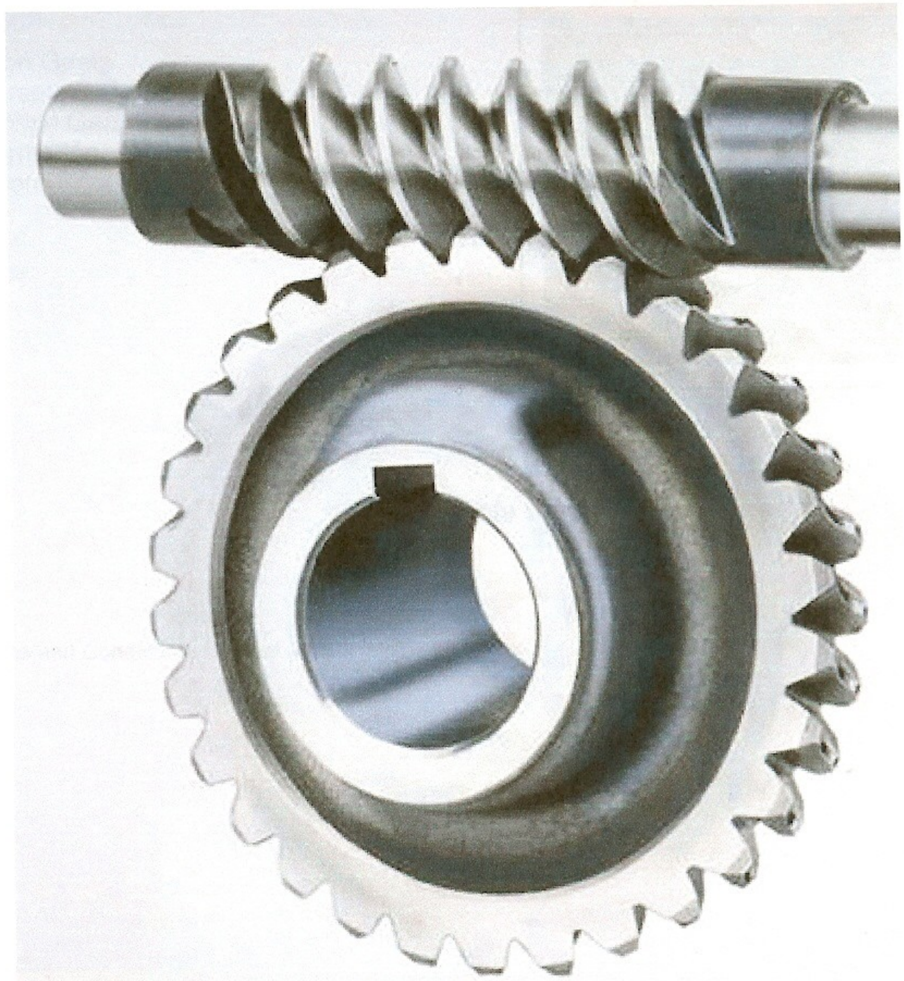


Figure 33: Worm drive. The worm drives the specially cut spur gear.

Worm drives can be back driven if the pitch of the worm and friction coefficient allow it. Many worm drives are designed to preclude back driving.

5.10.3 Chains and Sprockets

Additional reduction can be achieved with chain drive, which is often used for the final stage to the wheels or arm joint.

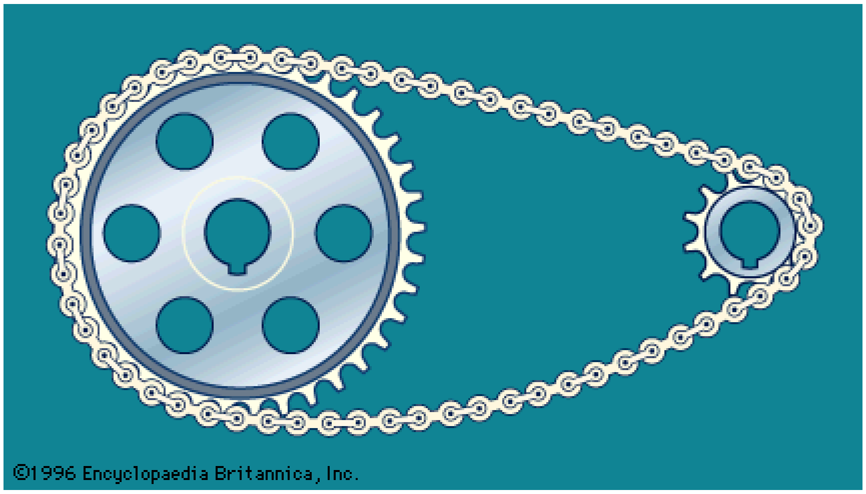


Figure 34: Roller chain and sprockets.

Chain drives are light weight and flexible, avoiding the precision alignment that gears require. An idler sprocket, either adjustable or on a spring tensioner arm, is often used to keep the chain properly tensioned. In use, a chain should be loose enough to avoid binding of the roller pins, but tight enough to prevent tooth skipping.

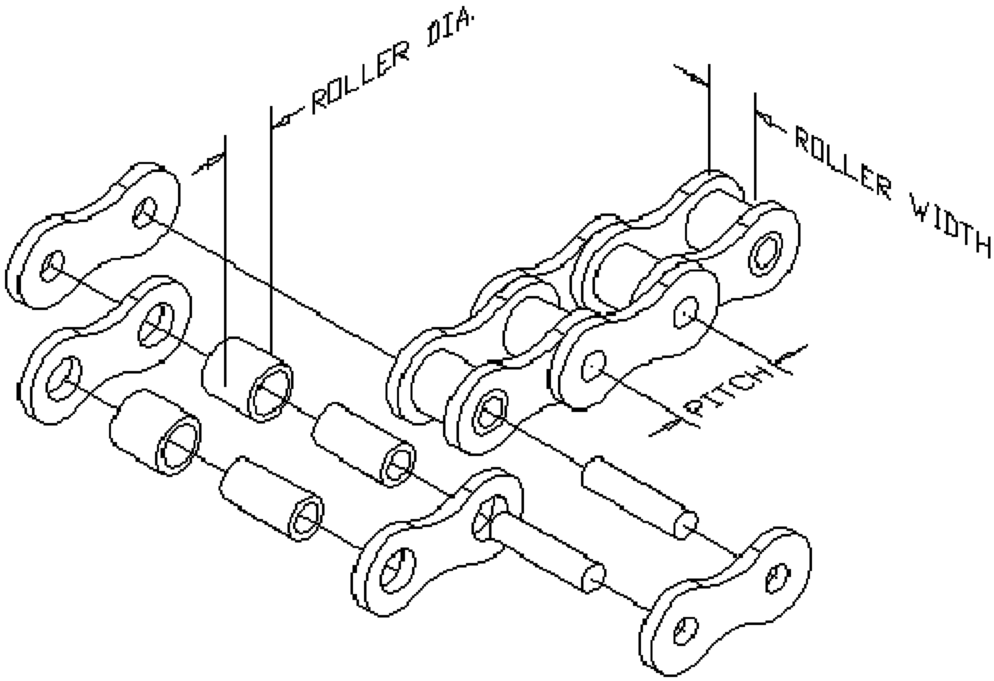


Figure 35: Roller chain terminology.

Roller chain should be lubricated with a heavy oil and then wiped clean. Unlubricated chain can wear out and break.

6 Introduction to Programming

This section describes computers and programs so that the beginning student will understand how to design his own programs to solve various problems.

6.1 Algorithms and Computation

It is occasionally amusing to see a circular definition of computation in a computer science textbook. For example, I have seen computation defined as what computing machines do and computers defined as machines for doing computation. A computer is a machine for executing algorithms. Computation is indeed what computers do: the execution of algorithms.

Definition: An algorithm is a set (often, but not necessarily, a sequence) of steps ordered to accomplish a goal.

A key to understanding this definition is that the algorithm, when properly executed, must accomplish the goal. Another term for algorithm is "effective procedure." It must work (accomplish its goal in a finite number of steps) in order to be an algorithm. Sets of steps that don't (aren't guaranteed to) accomplish a goal might not be meaningless, but they don't constitute an algorithm.

For example, suppose the task is to sort a deck of cards, ace through king for each suit, with the suits in order clubs, diamonds, hearts, and spades. As one can imagine, there are various effective ways to sort the deck. One of them is to take the unsorted deck, take the top card and put it face up on the table. Then take the next card. If it belongs to the right of the face-up card, put it face up to the right of it, otherwise, put it face up to the left of it. Now there are two face-up cards on the table. Take the next card and put it in its place on the table, either to the left of the two cards, insert it between them, or to the right of the two cards. Take the next card and put it in its proper place in the face-up sequence. Repeat until all the cards have been placed. The deck is now sorted.

The above procedure is called an *insertion sort* because the next card is inserted into the sequence in its proper place. The insertion sort is guaranteed to end with a sorted deck after 2652 operations ($n * (n – 1)$ where $n = 52$, and examining the sequence on the table takes m steps where m is the number of cards on the table to the left of the inserted card's place). Now think about this procedure for sorting the cards:

Take the deck of cards and throw it high in the air so that the cards scatter and fall to the floor. Pick up all the cards and stack them into a deck. Check to see if the deck is sorted. If it's sorted, stop, otherwise repeat. I call this procedure the *bogo sort* (bogo for bogus).

Would you ever consider using the bogo sort? You could get lucky and have a sorted deck on the first try, but there is no guarantee that the procedure would ever work. For that reason, the bogo sort is not an algorithm.

However, that doesn't mean that the bogo sort is meaningless. It might be fun to program a robot that way and watch it throw cards and pick them up all day. The meaning in this case is defined by the robot programmer.

Before we get into programming robots we are going to learn some simpler programs. A program is an algorithm in computer code (or computer language).

Computers are used for a variety of purposes, including mathematical analysis, databases, word processing, document publishing, computer aided design, graphics, games, and industrial process control. When electronic computers were first invented they were intended for and first used for mathematical computation. As computers became more capable they were harnessed for other tasks. In learning programming, it's best to start learning the fundamentals with some simple mathematical algorithms. Later we will learn some other functions such as text manipulation and interfacing with the user.

In instructional writing such as this it is useful to give programming examples, which requires the use of a programming language. The examples here use the language Java for which an on-line reference is available at http://java.sun.com/. There you can also download the Java 2 software environment (J2SE) free and install it on your own computer. With the online tutorials and application programming interface (API) reference, the student has access to a complete programming instruction course.

6.2 Variables and Operators

To solve algebra problems we use mathematical notation such as

x = 2 + 5

In this case, x is a variable and the plus sign is an operator signifying the addition operation. In Java, the symbols for variables are strings of characters without embedded spaces. For example, we could use the strings "ex", "eks" "distanceX" in place of the "x" in the example above. Java code is *case sensitive*: capital letters have meaning in distinguishing symbolic strings. The "2" and the "5" are called *numeric literals*.

Commonly used arithmetic operators in Java include the following:

`+`	addition
`-`	subtraction
`*`	multiplication
`/`	division

Suppose we wanted to write some Java code to compute the area (a) of a rectangle of height h and width w. We could write:

```
float a = 0;
float h = 3.5;
float w = 2;

a = h * w;
```

The first three lines declare the three variables used in the program, three floating point (continuously variable) numbers. They are initialized to their starting values. There is a blank line after the variable declarations to show that we are entering a different section of the program. The last line is an *assignment statement*. It sets the value of A equal to H times W, the area of the rectangle.

In addition to the floating point variable type, Java has integer, character, String (actually an object type) and double precision floating point types (and a few others not frequently used).

When a programmer wants a number that never changes, such as a constant like pi, the number is declared as constant with the Java keyword "final." Of course, pi or some other constant could be declared as a variable, but then there is the possibility that a programming error could change the value of the constant during the course of program execution, leading to a difficult-to-find bug.

6.3 Decisions

Decisions are essential to algorithms that simulate intelligent behavior. The simplest decision is selecting one alternative from two choices. For example, the question "is six bigger than seven" is answered by deciding "no." There are only two possible answers to that question. The question "I am thinking of a whole number from three to five inclusive. What is it?" has three possible answers, three, four, or five, so it's not a simple decision.

Simple decisions in Java are represented by *if-else* syntax. For example, in Java we could write the first decision example by:

```
if (6 > 7)
{
  // This can't happen.
}
else
{
  // Code here will execute.
}
```

The `>` (greater than) symbol is a comparison operator. Commonly used comparison operators in Java include:

`>`	greater than
`>=`	greater than or equal to
`<`	less than
`<=`	less than or equal to
`==`	exactly equal to (be careful with this one with floating point values)
`!=`	not equal to

The Java compiler will ignore any text on a line after the two slashes. These lines beginning with two slashes are *comments*. Comments in code are good. Generally, almost every statement line in a program should have a comment to tell the human reader what's going on in the program. Ordinarily, two numeric literals would not be used in an if decision because the decision could never change. This example is merely to illustrate the syntax.

6.4 Data Structures

There are two fundamental data structures in Java, arrays and objects, but it's not quite that simple. In Java, an array can be regarded as an object, and one can have an array of objects. In an object-oriented language like Java, these distinctions can be blurred because everything can be treated as an object.

6.4.1 Primitive Types

The "primitive data types" are the building blocks for more complex data structures. These primitive types are common to most programming languages, and generally include integers, floating point (rational) numbers, and characters, like the letter "a" or the digit "6" or punctuation marks.

The primitive types are founded on "words" (the fundamental units) of computer memory. Most personal computers use a 32-bit word. In general, programs are unable to access fewer bits of memory at a time. For example, Java has a "true" or "false" Boolean (one bit) type, but the variable is stored in a whole word of computer memory. C programs use "1" or "0" to represent true or false, which is a 32-bit integer representation, again taking a whole word of memory. Likewise, an 8-bit character type takes a full word of storage.

If efficient memory use should be an issue a programmer can resort to clever schemes to pack more information into a single word. For example, up to 32 Boolean values can be stored in a single integer, but the parsing scheme required is generally not worth the programming effort. The integer is constructed by taking each of the Boolean values needed and adding the appropriate power of two if the value is true.

6.4.2 Arrays

An array is a sequence of the same type of variable. For example, a point in space can be represented by an array of three floating point values.

```
float sfPointA[];
sfPointA = new int[3]

sfPointA[0] = 7;
sfPointA[1] = 5;
sfPointA[2] = 0;
```

6.4.3 Objects

A better way to represent a point in space is as an object.

```
class Point3D
{
  float sfX = 0;           // Default value.
  float sfY = 0;           // Default value.
```

```
  float sfZ = 0;           // Default value.
}
```

Then it would be easy to establish an array of points, representing, say, a trajectory (path for a robot to follow).

```
int i = 0;
int iNumPoints = 500;
Point3D pTrajectory[];
pTrajectory = new Point3d[iNumPoints];  // Variable array
                                        // declaration.
for (i = 0; i < iNumPoints; i++)        // Run a loop.
{
  pTrajectory[i].sfX = i;
  pTrajectory[i].sfY = i * 2;
  pTrajectory[i].sfZ = 0;
}
```

In addition to incorporating attribute data, objects can also have functions or "behaviors," making an object oriented (OO) approach to robotic algorithms quite useful.

6.5 Program Design

There are a number of ways to approach the design of programs, but they generally fall into two major categories, procedural design and object oriented design. Procedural design focuses on actions that the program should perform. Object oriented design applies attention equally to the things that act and are acted on (objects) and to the actions that objects perform.

For example, in the traffic simulation (http://teamster.usc.edu/~dteam/images/traffic/), each of the cars is an object that has a driving behavior. The cars want to accelerate to the speed limit and to avoid crashes by braking when necessary.

When driving on the freeway, if the number of cars on the road increases above some value, a "stop wave" behavior emerges. There will be a place in the line of cars where movement comes to a halt. When a car arrives at this place, it will have to stop to avoid hitting the stopped car in front of it, and then go when the car in front of it starts to move. Cars generally can stop much more quickly than they can accelerate forward. These stop waves tend to propagate backwards in the flow of cars. I wanted to see if I could replicate this behavior in a simulation. The traffic simulation makes many simplifying assumptions. There is only one lane in a continuous circuit. All the cars want to travel at the same top speed. All the cars have the same acceleration and braking properties. Even with these simplifications, the stop wave behavior emerges as the number of cars is increased.

Another program example, this one with more user input, is the lunar lander (http://teamster.usc.edu/~dteam/images/lander/), which has an autopilot mode, and can serve as an introduction to robot algorithms.

6.6 Computer Learning Resources

Some excellent computer and programming learning resources are available on the Web, including the excellent tutorial, Online Interactive Modules for Teaching Computer Science at http://courses.cs.vt.edu/~csonline/index.html.

7 Robot Algorithms

Robotics and artificial intelligence (AI) are closely related. In fact, some text books treat robotics as a subtopic of artificial intelligence, while others handle it the other way around. A robot is a physical device. Because every robot should be smart, every robot needs a brain, so every robot needs artificial intelligence in addition to motive power and sensing. Therefore, AI is most properly, in my opinion, a sub-discipline of robotics. That is not to say, however, that there is no need for the study of pure AI independent of robotics.

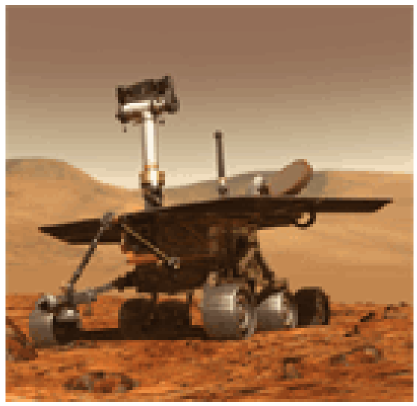


Figure 36: Mars exploration rover.

Pure AI deals with topics such as natural language communication and understanding, game playing and military strategy, expert systems (such as medical diagnostics), manufacturing planning, and so on. An example of a pure AI program is the tic-tac-toe applet (http://teamster.usc.edu/~dteam/images/tictac/) with source code. What makes robot AI special is it must allow the robot to get along in the physical world.

7.1 Processes in the Physical World

Understanding the physical world requires knowledge of physics which involves three spatial dimensions and time. Geometry and dynamics, therefore, play important parts in most robot algorithms.

7.1.1 Robot Attributes: Autonomy, Mobility, Manipulation

Robots are distinguished from other artifices by the three key attributes of autonomy, mobility, and dexterity, or the ability to manipulate its surroundings. Not all robots have all three attributes, but any device that has all three is definitely a robot.

7.1.1.1 Autonomy

Autonomy is defined as the ability of a system to operate successfully for extended periods without human intervention.

7.1.1.2 Mobility

The easiest way to obtain mobility is by putting wheels on a robot. Tracks are sometimes used. Legged locomotion provides some advantages on rough terrain but is technically challenging to implement.

7.1.1.3 Manipulation

Manipulation can be as simple as using gas jets to blow dust and dirt (as was done in the first robot on the Web, the Mercury Project, by Prof. Ken Goldberg, see http://www.usc.edu/dept/raiders/), and as complex as replicating a human hand for grasping and holding. Pushing boxes around on the floor is a simple form of manipulation.

7.1.2 Space Robotics

Space robotics is concerned with the use of robots or robotic algorithms in space. Generally, there are two kinds of space robotics. First is the "traditional" autonomous guidance and control systems used on spacecraft, and second is the planetary explorer rover, usually a wheeled vehicle.

Planetary rovers have sensing and control requirements very similar to those for FIRST competition robots. Because other planets are so far away, the communication time for robot remote control is usually an hour or more, so a remote controlled vehicle could get in a lot of trouble very quickly. Therefore, NASA, ESA, and other space agencies use autonomous (true) robots for planetary exploration.

NASA's Jet Propulsion Laboratory (JPL) in Pasadena, California is developing a nuclear fission powered spacecraft to investigate the three icy moons of Jupiter, Callisto, Ganymede and Europs. JPL has specified significant autonomy for the Jupiter Icy Moons Orbiter (JIMO) spacecraft (see Figure 37, below).

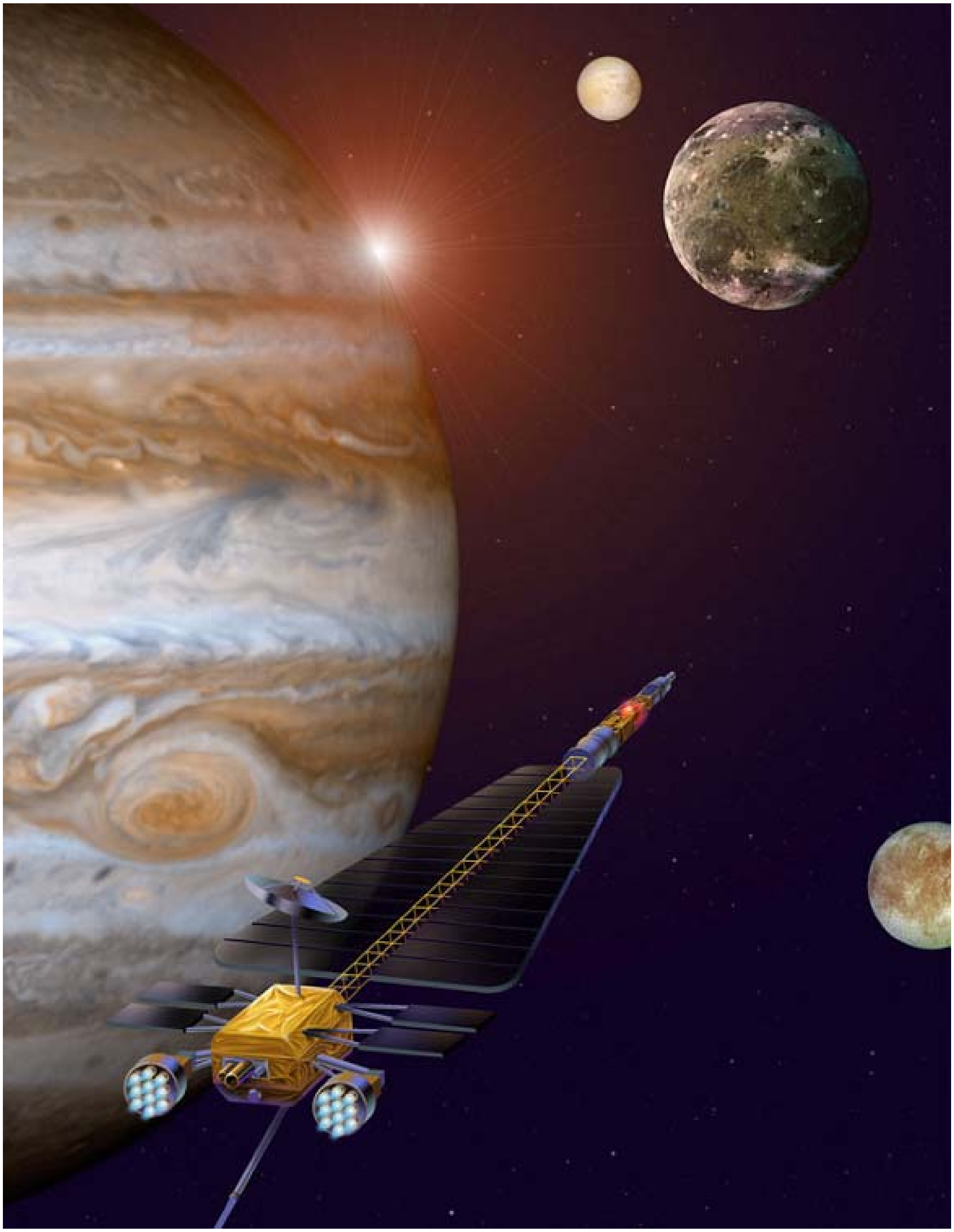


Figure 37: Artist's conception of Jupiter's Icy Moons Orbiter (JIMO) nuclear powered robotic spacecraft.

The need for autonomy on JIMO is apparent when one considers that the communication time is on the order of hours due to the great distance JIMO will travel from Earth. Although the formal requirements for JIMO autonomy are expressed in terms of fault management, if one applies the literal definition of autonomy to the superseding requirement for mission success, one can divide the need for autonomous operation into two categories: scheduled autonomous operations and unscheduled autonomous operations.

Unscheduled autonomous operations are clearly those autonomous actions necessitated by unplanned events such as fault occurrence or other unexpected experience. The need to prepare for scheduled autonomous operation becomes clear when we consider what orbit insertion requires in the way of system control without human supervision.

Successful navigation in the moons of Jupiter requires precision attitude and thrust control and that requires realtime updates of vehicle and planetary orbital data based on sensor inputs. While in theory such precision insertions can be safely performed by executing stored procedures, autonomous capability for evaluating sensor data and updating plans in realtime gives a level of system robustness not obtainable in any other way. Consider, for example, what might happen if previously unknown electromagnetic or gravitational phenomena should exert a small but significant effect on an insertion operation with a narrow thrust margin. Space vehicle autonomy can also reduce mission operation costs by reducing the complexity of mission control and therefore reducing the number of people required in the ground control station. Increasing the level of autonomy in the ground station can also have a cost reduction effect. [1]

Therefore, when inferring requirements for autonomy, we must also consider mission success, risk reduction, and robustness criteria, and not regard autonomy as merely a "fix it in case something goes wrong" feature, but as a full partner in mission operations, designed in from the beginning.

7.1.2.1 Space Robotics References

1. http://www.nasa.gov/
2. http://robotics.jpl.nasa.gov/
3. http://teamster.usc.edu/~dteam/Space/JWSTOverivew_files/frame.htm
4. http://teamster.usc.edu/~dteam/Space/JWST02_DEP.AVI

7.2 Planning

Robot motion planning is the AI field of planning applied to robot motion. For example, suppose a robot is capable of stacking and unstacking boxes, and the task is to find a red colored box in a stack and move it to a special location without knocking any stack over. A robot could be programmed to find the red box in a stack, unstack boxes on top of it until the red box is the top box, then retrieve the red box. The robot's program would need to allow it to create a plan to unstack the boxes before it could act in an intelligent way.

7.3 Sensing and Control

Robotic action usually involves a sensing and control loop. The robot senses the environment around it, updates its plan accordingly, then acts on the plan, and repeats the process: sense, plan, act. The sensor data is used to update the robot's understanding, or map of the world it operates in. This "state space" is then fed into the planning algorithm, and the plan of action to alter the state space is the output of planning.

7.4 Navigation

Navigation is finding the robot's way around in the environment. Mobility is one of the major characteristics of robots, and action plans usually involve changing the robot's location. This means that the robot's state space includes a layout map of the robot's world, and the robot's place in that map.

In FIRST 2003 Stack Attack the autonomy period focused mainly on getting the robot to successfully change its location from its starting position to somewhere over the ramp into enemy territory. The scope of that autonomous task was fairly limited, but it's a safe bet that next year's autonomy challenge will be significantly more difficult.

7.5 Advanced Robotics Example: Robo-Butler

An advanced robotic application requires a software engineering approach to software development. That means that the development process is a team effort and should be managed to avoid chaos. A simple model of a software engineering project is a waterfall (sequence) of the phases *requirements*, *specification*, *design*, *code*, and *test*. Normally there will be some iteration between phases as the results of one phase are used to check the previous phase's validity. This example uses a hypothetical product, a robotic butler for the affluent home owner. The example was developed for a software engineering course taught at the University of Southern California (USC) in the previous century.

This example first uses an informal specification. Suppose you are a software engineer working for Rick's Universal Robots (RUR) company. Further suppose that RUR wants to introduce a new domestic robot, the Robo-Butler. And suppose that the mechanical design of the robot has been fairly well established and that it is up to you to produce the software for the robot.

The target customer group for Robo-Butler is a typical upper-middle-class house-holder with two Infinitis in his driveway and fifty thousand dollars burning a hole in his pocket. He wants a household robot with which to impress his friends.

You decide to hold some sensing sessions with typical members of the target customer group in order to establish requirements. Here are the results of the sensing sessions:

7.5.1 Robo-Butler Requirements

- The primary requirement is to impress the customer's friends when they are at the customer's house. That is, the Robo-Butler should (1) perform efficiently as a butler, (1.1) recognizing and (1.2) greeting the guests by name at the door, (1.3) offering to take their coats, and (1.4) escorting them to the host, at a party for example.

- The Robo-Butler should perform other duties at social functions, such as (1.5) fetching drinks for the guests, (1.6) collecting empty glasses and taking them to the kitchen, and (1.7) emptying ashtrays as necessary.
- (2) The Robo-Butler should not pose a threat or injure any person at any time. (2.1) We want the guests to be impressed by the robot, not afraid of it.
- (3) The Robo-Butler should not let anyone into the party who is not on the invitaion list. (3.1) In the case of an unrecognized guest, the Robo-Butler should politely ask the guest to wait (outside at the doorstep) and then summon the host (customer-owner)
- Many customers asked if the robot could wash dishes. At this time, the answer is "no" because that task is too difficult. (4) Perhaps in the next version, (4.1) a new software module can be added that will have that capability. (4.2) The same goes for vacuuming the floor

7.5.2 Robo-Butler Specification

A number has been assigned to each general and specific requirement identified above. The list of specifications is structured accordingly.

1 Butler Functionality

The primary functionality of butler behavior will require a speech recognition module, a vision module with face recognition, a voice module with speech synthesis, and appropriate locomotion and manipulation modules.

1.1 Guest Recognition

The vision system shall utilize an appropriate face recognition technology and shall allow the customer to easily load images of his friends for training. The recognition shall be 98 percent accurate under typical operating conditions under typical exterior or interior lighting, day or night (except in the case of guests arriving for a Halloween party).

1.2 Guest Greeting

The Robo-Butler, upon recognizing a guest or group of guests who are all on the invitation or general access list, shall open the door for them and politely greet each by name with an appropriate welcome (such as "How very pleasant to see you again, Ms. Calvin, won't you please come in?).

1.3 Guest Possession Assistance

As the guests enter the house, the Robo-Butler shall assist the guests with their coats, hats, and other possessions which they wish the Robo-Butler to store for them. Robo-Butler shall extend his hand, palm up, to each guest in turn and say "May I take your coat (and hat)?" as appropriate. If a guest should say, for example "Yes, please, and will you also take my cane?" then the Robo-Butler should accommodate the request and say "Yes, certainly."

The robot should place the collected guest possessions in the hall closet in an appropriate manner, planning the best process for doing so (there are a lot of implicit specifications in this one, and they need to be further broken out).

1.4 Guest Escort

The Robo-Butler should say "please come with me" or other expression as appropriate and lead the way to the host (customer-owner). If there should be some event, such as a guest saying "just a minute while I adjust my hair in the hall mirror," for example, the robot should wait appropriately and then continue escorting the guest(s).

1.5 Drink Fetching

Robo-Butler should not offer drinks to the guests, but should respond to requests such as "will you please bring me a glass of white wine?" This requires knowledge of various drinks and how to assemble them, and how to carry them without spilling. A tray will be be used to carry drinks as appropriate. The robot should also respond appropriately to requests by the host for drinks for his guests.

1.6 Glass and Plate Bussing

The Robo-Butler, when not answering the door or performing drink fetching duties, should look around and be on the alert for empty glasses not being held by guests and empty or abandoned-looking plates not being used by guests. When these items are identified, Robo-Butler should take them to the kitchen, using a tray as necessary, and put them on empty counter space in an efficient way.

1.7 Ashtray Emptying

When not otherwise occupied, Robo-Butler shall keep an eye out for ashtrays more than one third full (of cigarette butts), and shall empty them into an appropriate fire-proof receptical.

2 Safety

All behavior modules shall be proven correct to preclude unexpected behavior. The Robo-Butler shall be incapable of exerting more than 40 pounds force within two feet of a human, and shall be incabable of moving any component at a greater speed than two feet per second within two feet of a human.

2.1 Benign Behavior

The Robo-Butler shall have a distinctly "robotic" motion so as not to startle humans with potential false recognition, yet the robot shall have a butler-like demeanor so as to immediately convey his benign function.

3 Security

The Robot-Butler shall preclude the entrance of uninvited guests.

3.1 Unrecognized Person

Upon the non-recognition of a person at the door, the robot shall not unlock or open the door, but shall immediately summon or go to the customer-owner.

4 Extensibility

The software shall allow for the "easy" extensibility as defined below:

4.1 Dishwashing

Future dishwashing capability shall be allowed for. The vision and manipulation modules shall be replaceable, or otherwise upgradable or adaptable.

4.2 Vacuuming

Future vacuum cleaning capability shall be allowed for. The vision and manipulation modules shall be replaceable, or otherwise upgradable or adaptable.

After reviewing the informal specification to see if it meets the requirements, the software engineering process further develops it into a semi-formal specification using some graphical techniques.

7.5.2.1 Structured Systems Analysis

We first create a high-level data flow diagram (DFD):

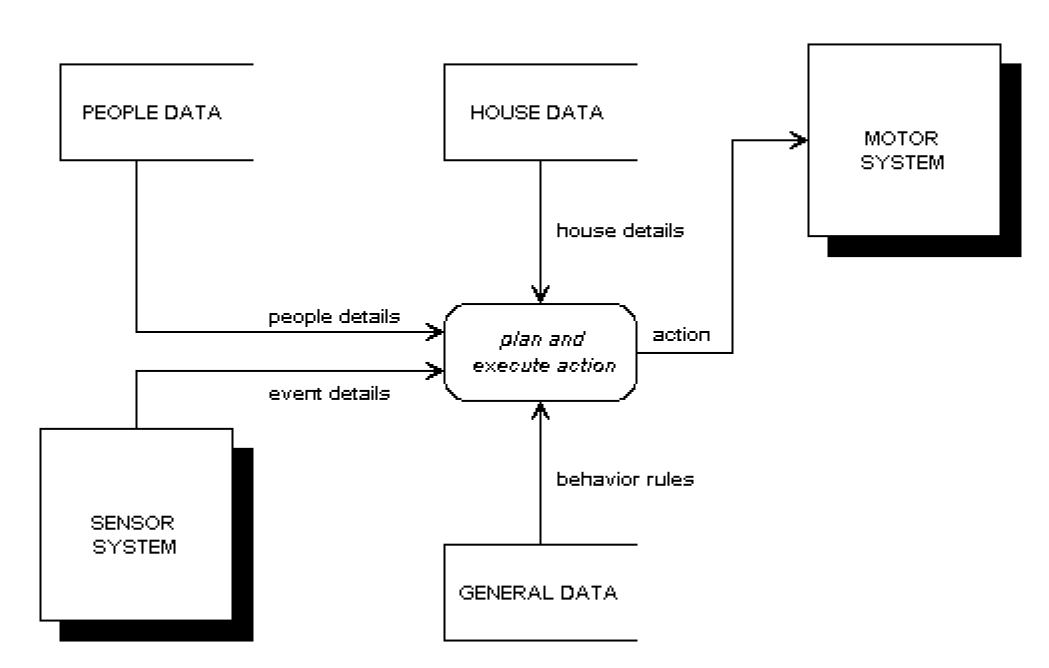


Figure 38: First refinement of the data flow diagram.

Next we produce a second refinement of the DFD:

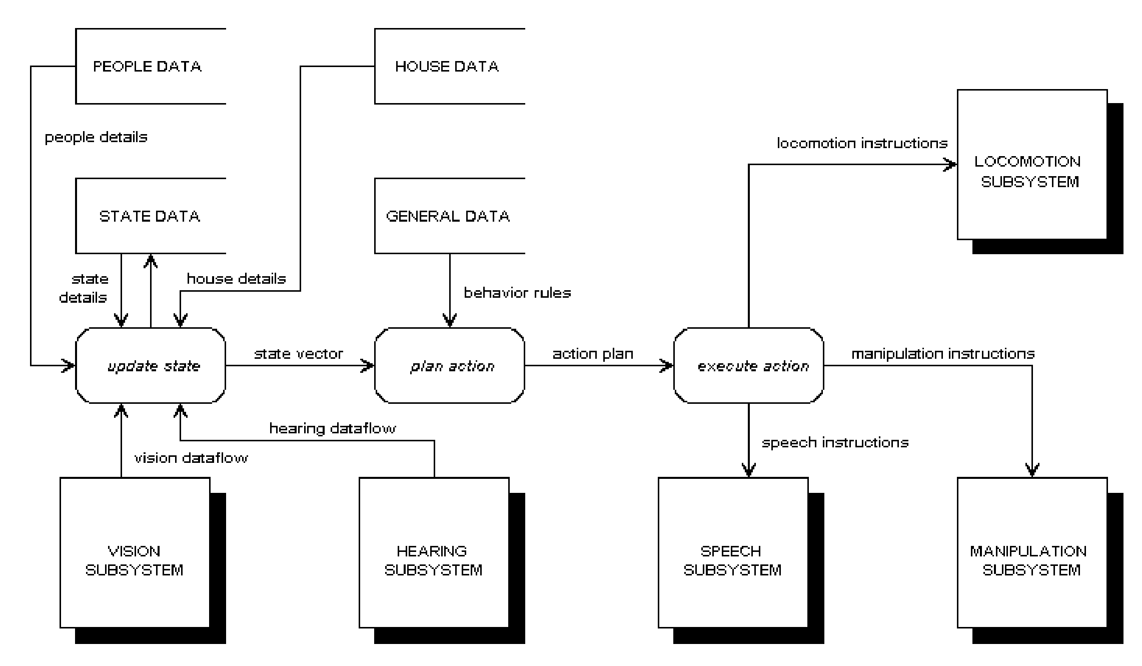


Figure 39: Second refinement of the data flow diagram.

The structured systems analysis will require several more iterations of DFD refinement. As you can see, the DFD will become quite large, so it is not shown here.

The next steps are to define the logic of the processes, define the data stores, define the physical resources, determine the I/O specifications, perform sizing of the system, and determine hardware requirements.

7.5.2.2 Entity-Relationship Modeling

Entity-relationship modeling was developed for database applications, but is also useful as the starting point for object oriented analysis, which is probably the best technique available to apply to the Robo-Butler problem.

We start by identifying the entities in the Robo-Butler scenarios:

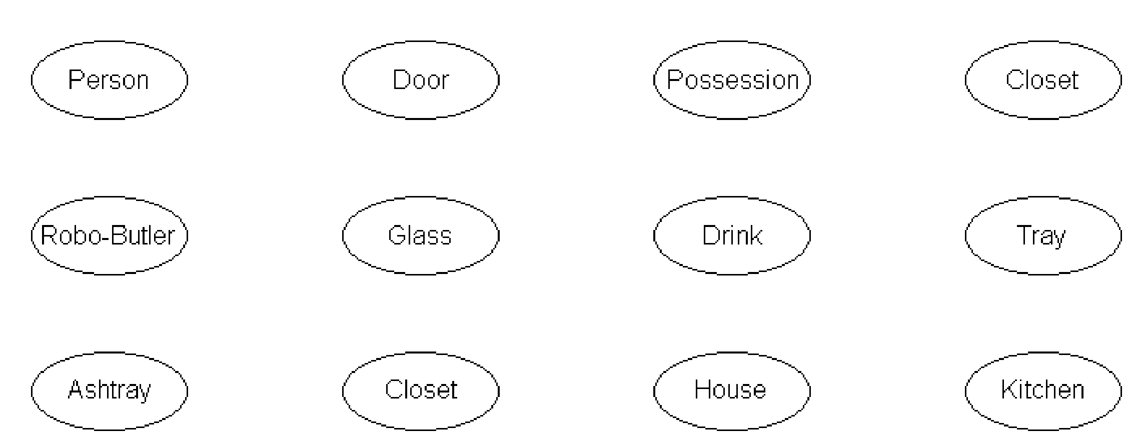


Figure 40: Entities in the Robo-Butler scenarios.

Then we identify the relationships among the entities:

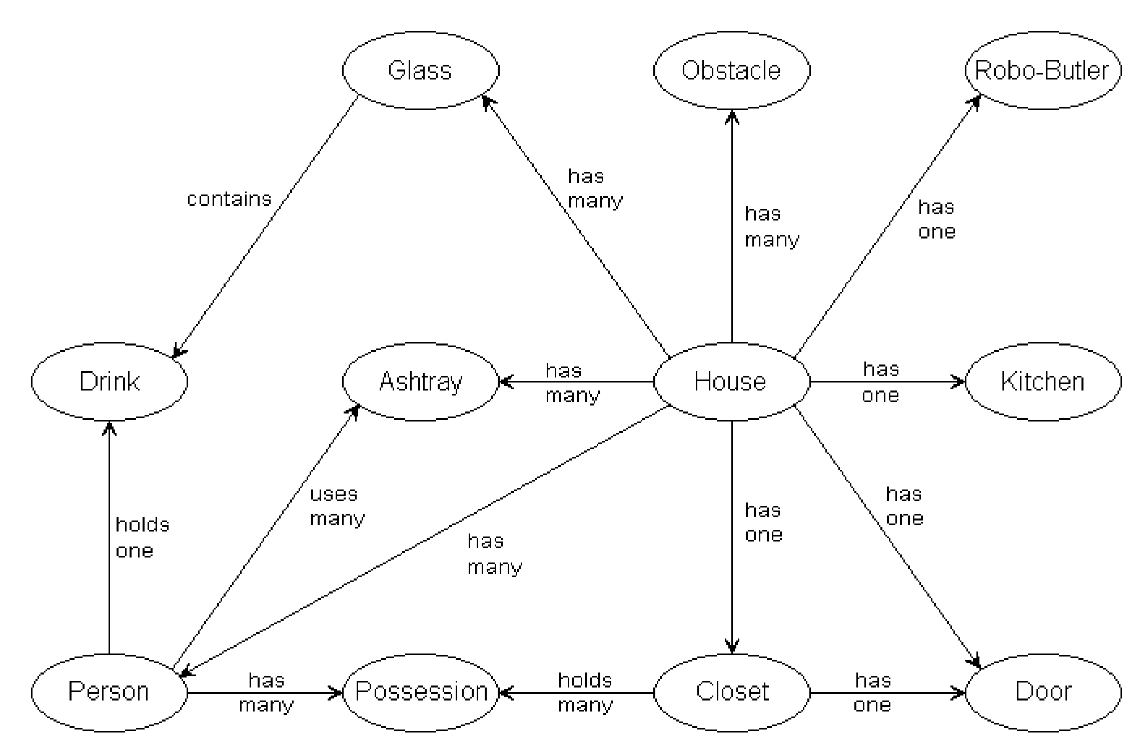


Figure 41: Entity-relationship diagram.

Next, we identify the attributes of the entities. A person, for example has a name, a location (X and Y coordinates), a direction he is facing (angle), an attitude (standing, walking, sitting, or lying down), and a reference image.

The entity-relationship model will be further developed into an object model by adding inheritance and methods (object oriented analysis).

7.5.2.3 Extended Finite State Machine

The Robo-Butler software will now be specified using an extended finite state machine. Note that all of the states defined below can be divided into substates.

Let *Scanning(glass, ashtray)* be the state where Robo-Butler is otherwise unoccupied, but is alert for abandoned drink containers and ashtrays in need of emptying.

Let *g* be a subset of the set of invited guests *G* and let *p* be a set of people outside the door of the house, a subset of the set of all people *P*. Note that *G* is also a subset of *P*.

Let *AtDoor(p)* denote the state in which a group of people has arrived at the door. Let *AtDoor(g)* denote the state in which Robo-Butler has determined that the group is made up entirely of invited guests, and let *AtDoor(p, g)* denote the state in which one or more of the people at the door is not recognized as an invited guest.

Let *Door(open)* denote the state in which the front door is open, and let *InHouse(g)* denote the state in which the front door is closed with the newly-arrived guests inside.

Let *Store(possessions)* denote the state in which Robo-Butler is storing the guests' possessions, and let *Escort(g)* denote the state in which Robo-Butler is escorting the guests to the owner.

Let *Fetch(drink)* denote the state in which Robo-Butler is fetching drinks for the guests, let *Bus(glasses)* denote the state in which Robo-Butler is carrying glasses to the kitchen, and let *Empty(ashtray)* denote the state in which Robo-Butler is emptying an ashtray.

Let *Summon(owner)* denote the state in which the Robo-Butler is summoning the owner. Then the high level state-transition diagram for the Robo-Butler software is:

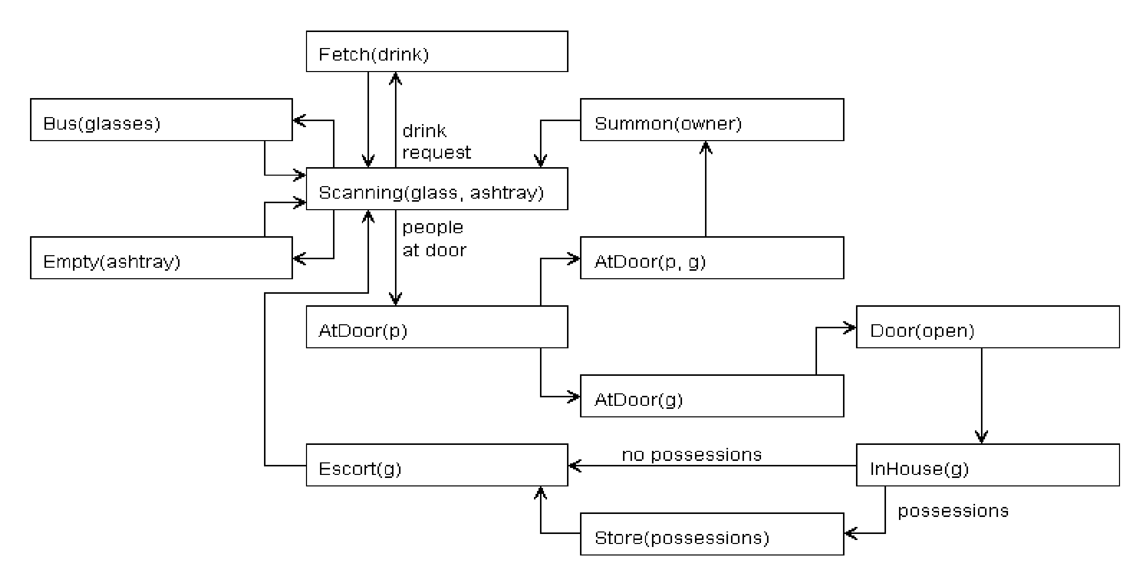
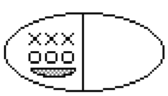


Figure 42: STD for the Robo-Butler.

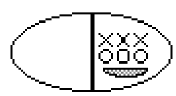
The finite state machine can also be used to solve the cannibals and missionaries problem, stated:

> Three missionaries are traveling with three cannibals. They come to a river where there is a boat that holds only two people. If the number of cannibals is ever greater than the number of missionaries on either shore, the missionaries get eaten. How do they cross the river without the missionaries getting eaten?

To solve this problem, draw the start and goal states:



Start



Goal

The three Xs in the elipses symbolize the cannibals, the three Os symbolize the missionaries, the vertical line symbolizes the river, and the boat symbol symbolizes the boat. The positions of the symbols (left or right of the river) denote the states.

Then starting with the start state, draw allowed states and connect them with arrows to denote the crossings of the river in the boat as transitions. Continue until the goal state is reached.

7.5.2.4 Petri Net

A Petri net is not fully suitable for specifying the Robo-Butler software because we need an initial higher level representation, but a portion of the Robo-Butler problem can be specified for illustrative purposes. Two of the functions of the Robo-Butler, ashtray emptying and drink fetching, have been used for the Petri net below:

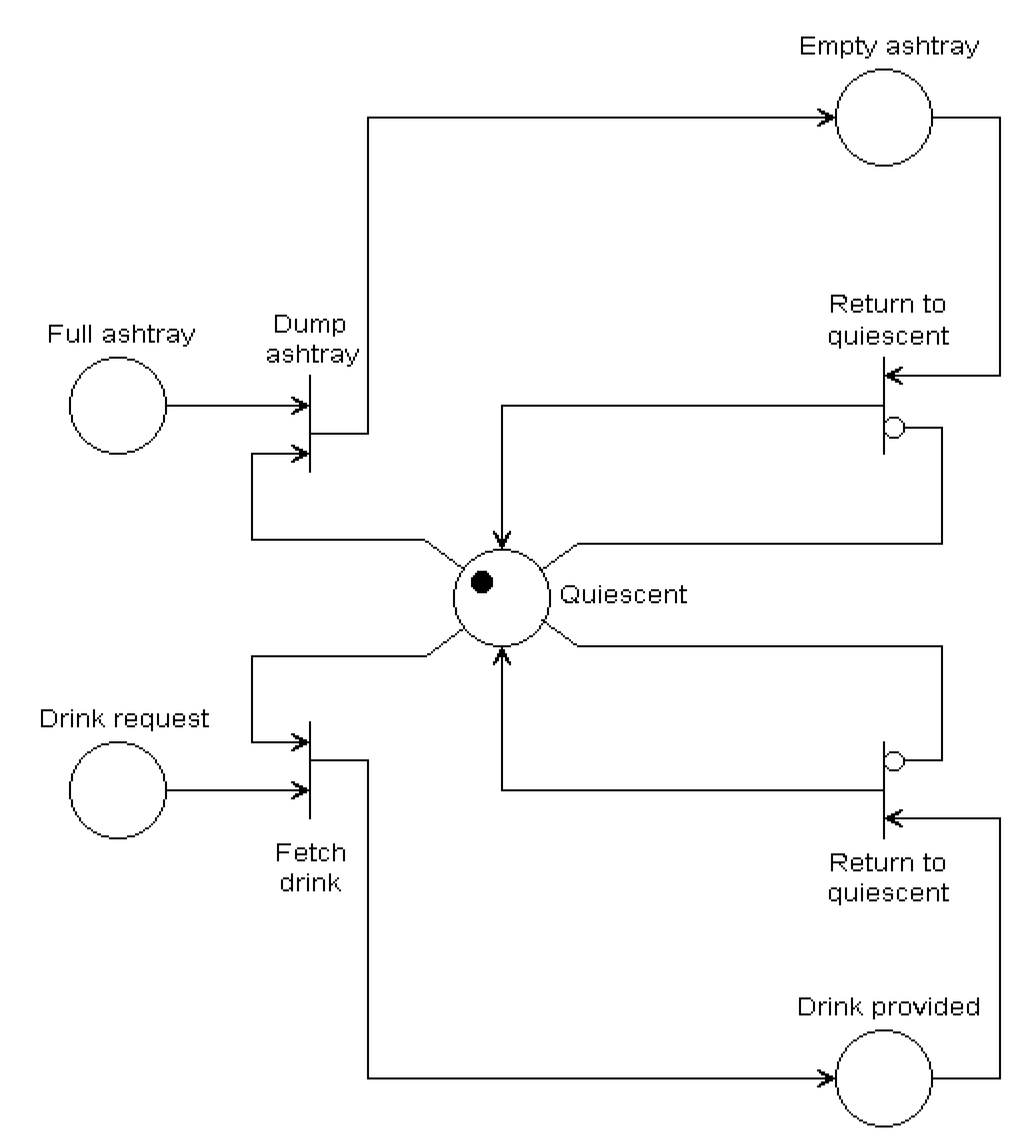


Figure 43: Petri net for a portion of the Robo-Butler software specification.

7.5.3 Object Oriented versus Structured Paradigm

The structured paradigm, although contributing to many successes in software development and maintenance, has been less successful with large projects. The object oriented (OO) paradigm offers the best chance for new improvements in software engineering productivity.

Data and actions are two sides of the same coin: data can't change without action, and actions without data are meaningless.

While a class defines both data and actions (member data and member functions), they cannot be considered simultaneously with existing techniques in the OO paradigm. Hence, we need to alternate our consideration from one to the other.

A well-designed object (has high cohesion and low coupling) models all aspects of one physical entity. Hence, it is easily and safely maintainable: we minimize the chances of introducing a regression fault. The reusability of objects is enhanced with inheritance.

7.5.4 Object Oriented Analysis: Robo-Butler Example

7.5.4.1 Class Modeling

We begin with the entity-relationship model developed previously in the structured systems analysis:

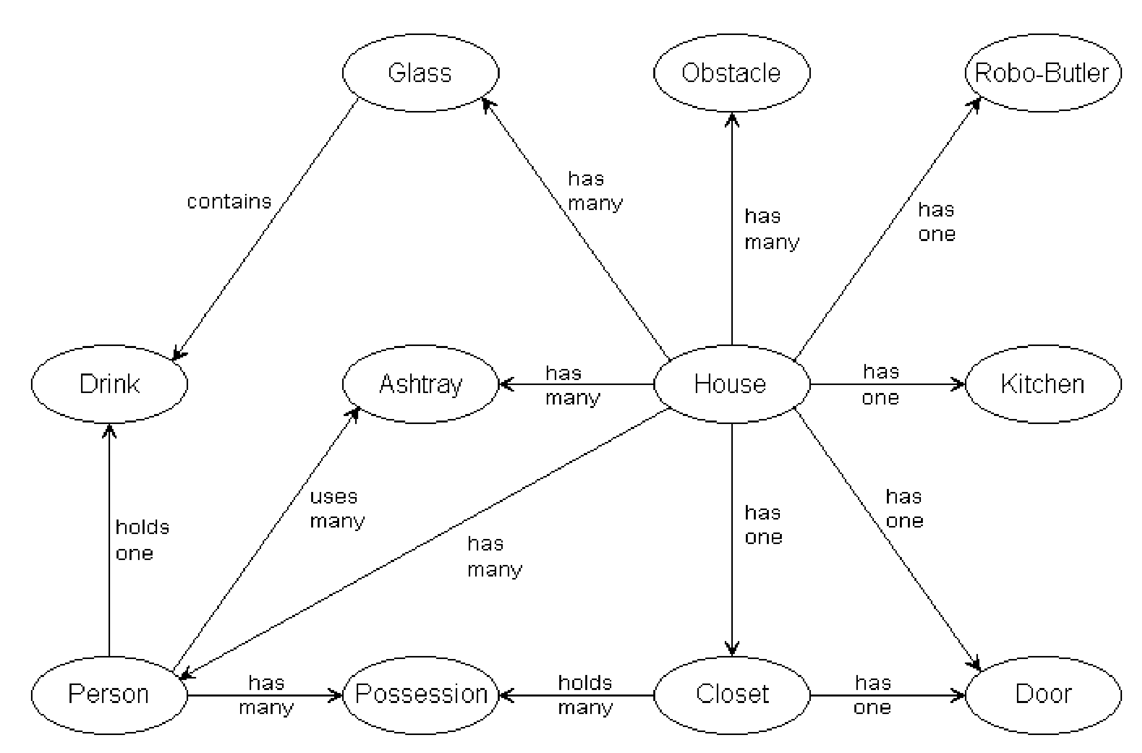


Figure 44: The entity-relationship diagram becomes the class model

No inheritance is used at this stage of refinement, and methods will be developed later as well. What were called "entities" in the structured paradigm are called "classes" in the object oriented paradigm.

7.5.4.2 Dynamic Modeling

Next we create a dynamic model for the Robo-Butler class:

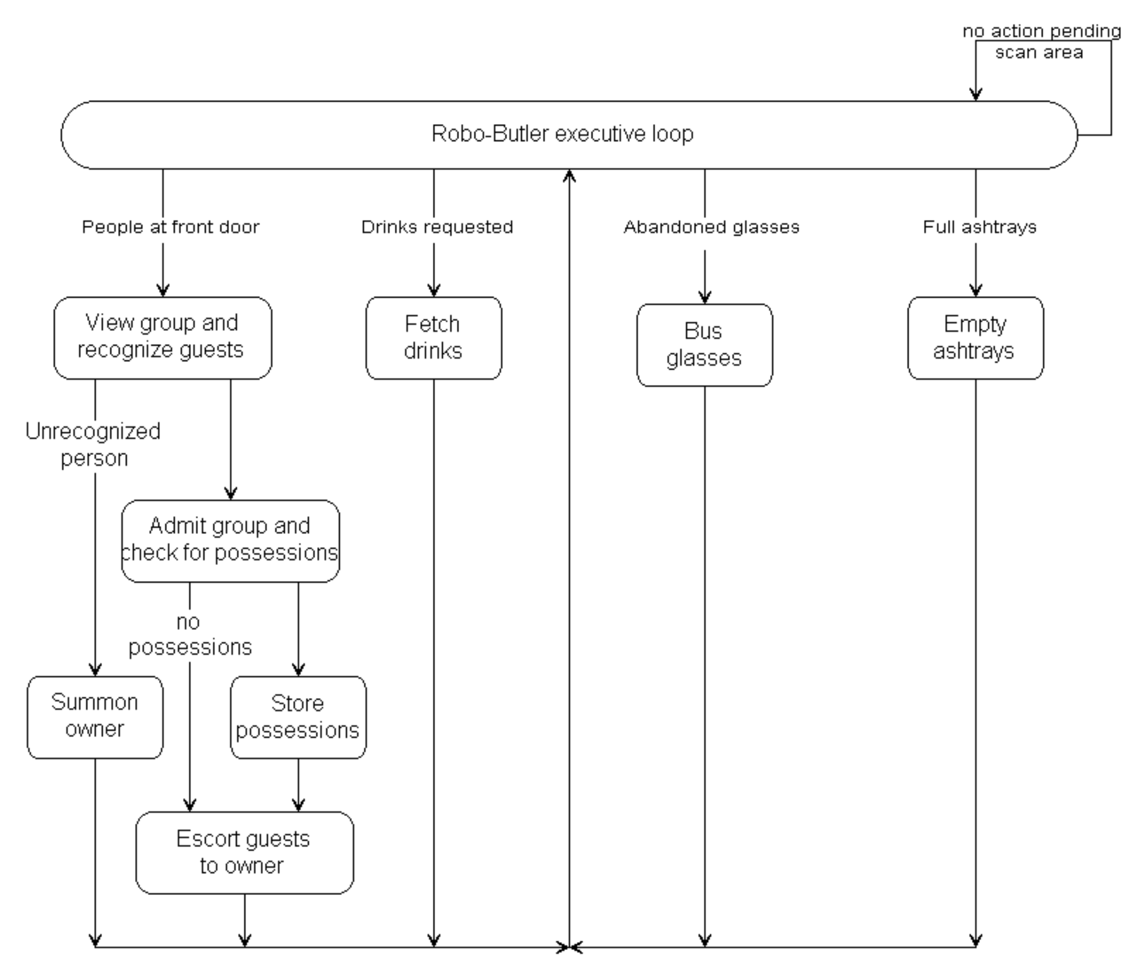


Figure 45: Dynamic model for the Robo-Butler class

7.5.4.3 Functional Modeling

Before creating the functional model, we refine the class model by adding inheritance:

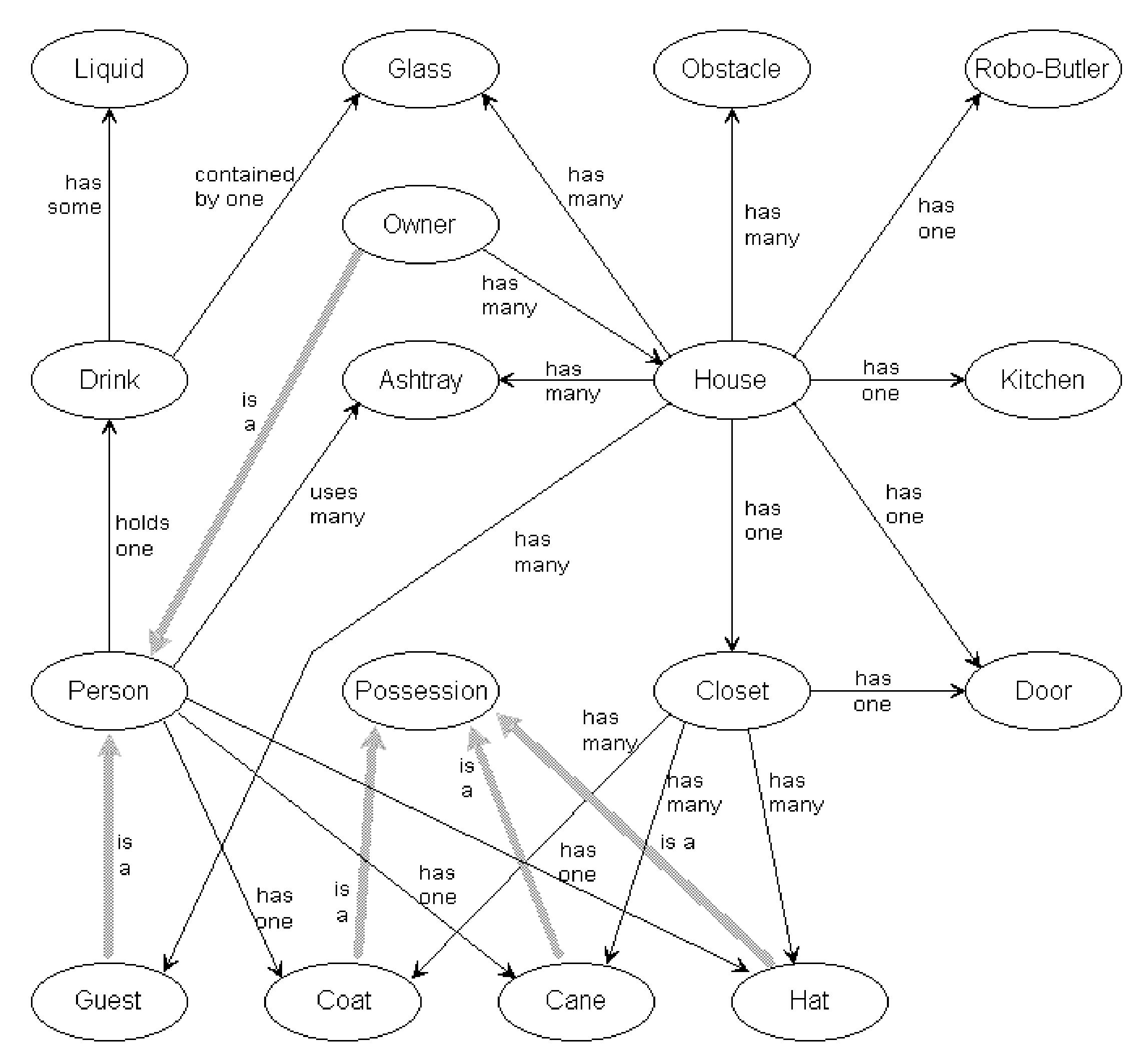


Figure 46: The refined class model adds inheritance to distinguish guests from the owner and the various types of possession objects.

We need to distinguish the owner as a special person. The robot also needs to distinguish among the types of guest possessions. Now we're ready to produce the functional model:

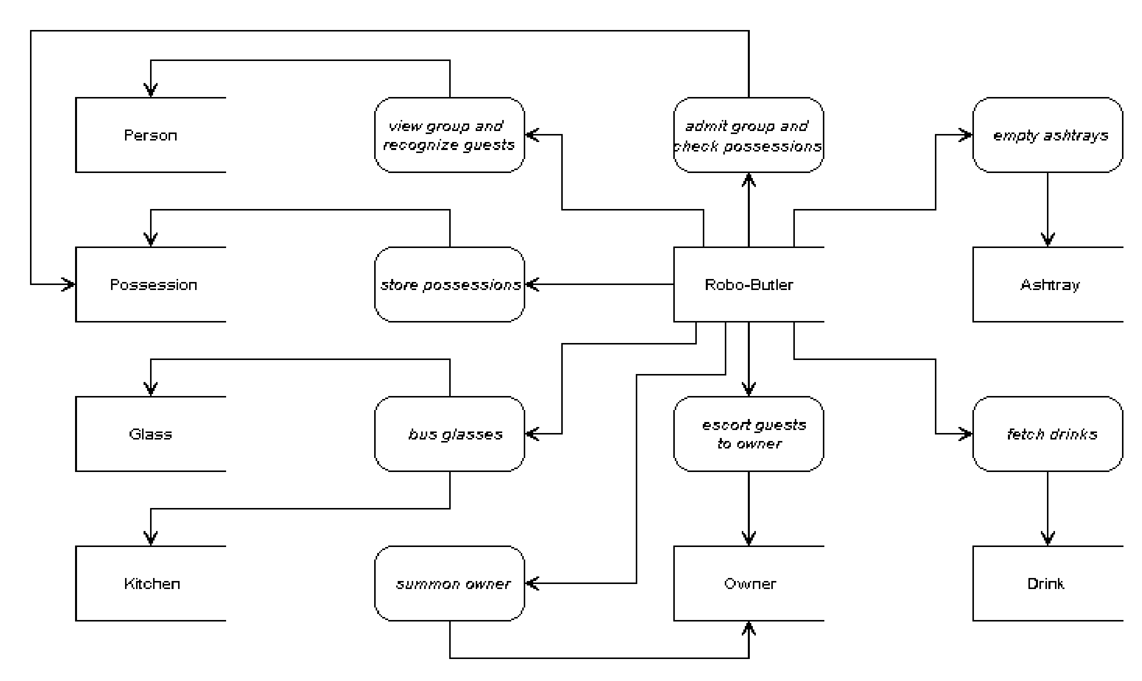


Figure 47: Robo-Butler functional model.

This completes the semi-formal specification example for the Robo-Butler. Now that we know *what* the specification is (what the robot should do), we need to decide *how* to do it (software design). Before starting design, it's a good idea to use the specification to refine the plan for producing the robot. One way to do that is to produce a PERT (project evaluation and review techniques) chart:

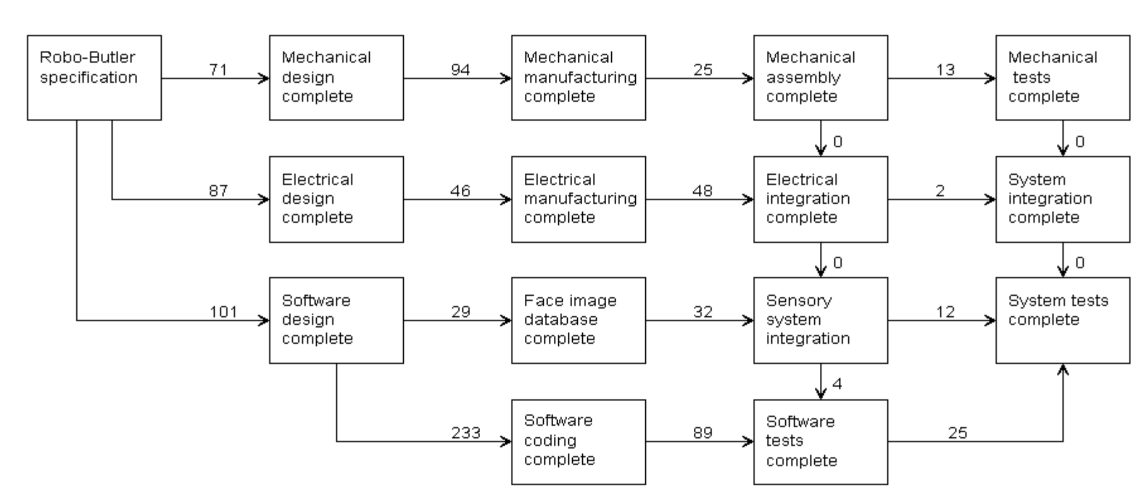


Figure 48: Robo-Butler project PERT chart. The numbers on the arrows are estimated time spans in days for each of the pointed-to activites.

7.5.5 Robo-Butler Design

It is not until the design phase that the "how" of the implementation is decided. This includes things like deciding what programming language to use, and what kind of hardware to run it on. Performance specifications often drive those decisions. In FIRST competitions, of course, those decisions are already made at the start.

8 Programming the FIRST Microcontroller for 2004

To be supplied by Mark Miller.

9 Acknowledgements

We wish to thank Harald Schifferl (http://hfs-design.at) for providing Web hosting services for this document.

10 References

[1] Sami Asmar *et. al*, "Reducing Cost with Autonomous Operations of the Deep Space Radio Science Receiver," JPL, Cal. Tech., Undated Web PDF document.

11 Version History

Version 1.0, June 6, 2003, contained a course outline and the the introduction to mechanics section, first published on the Web.

Version 1.24, October 4, 2003, first complete course published on the Web.