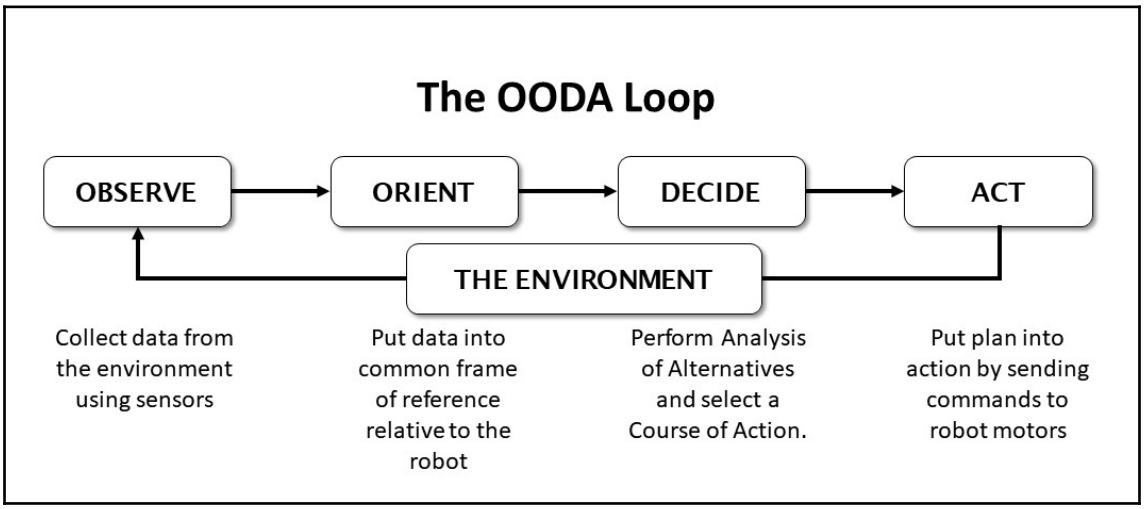
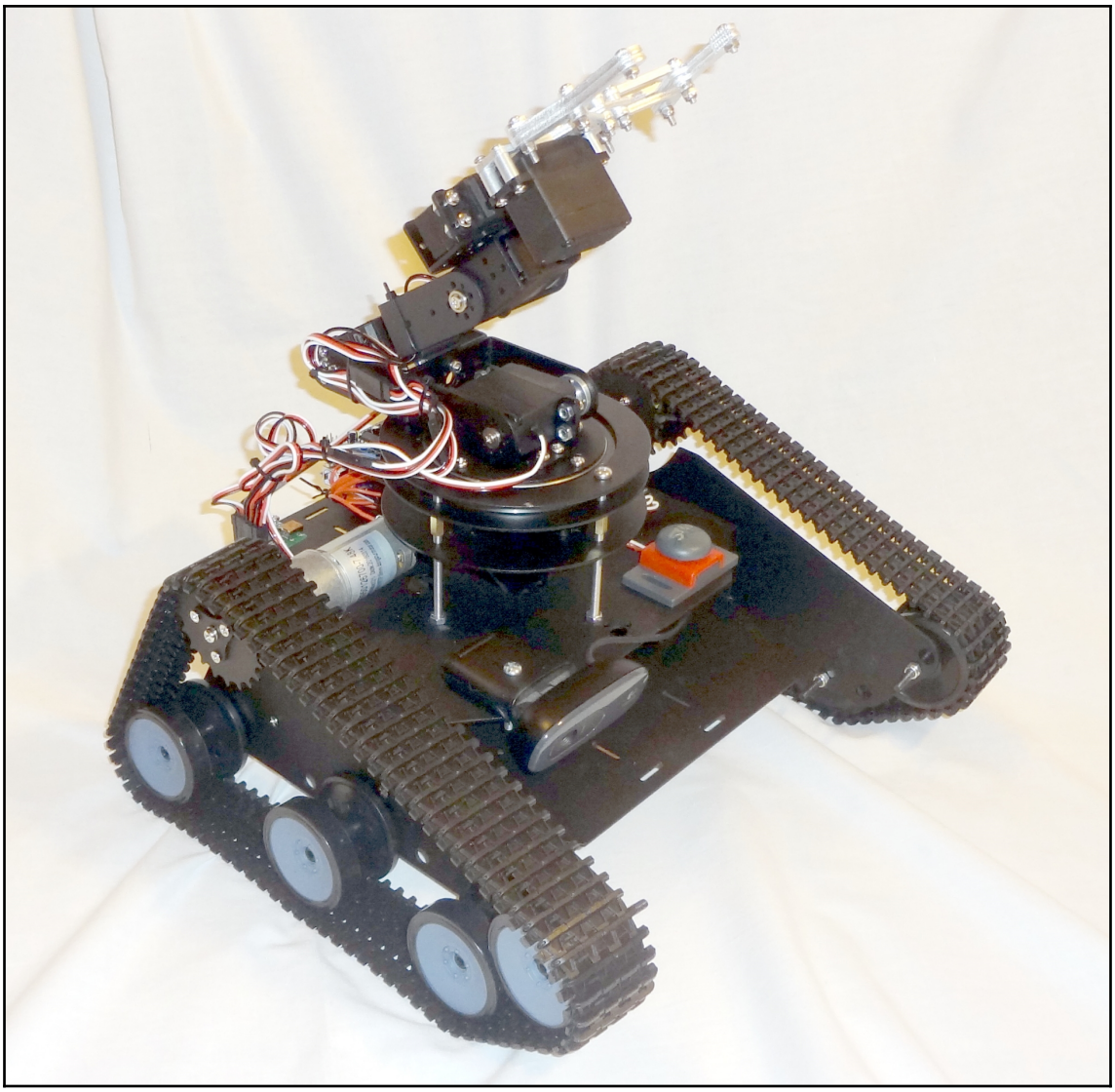
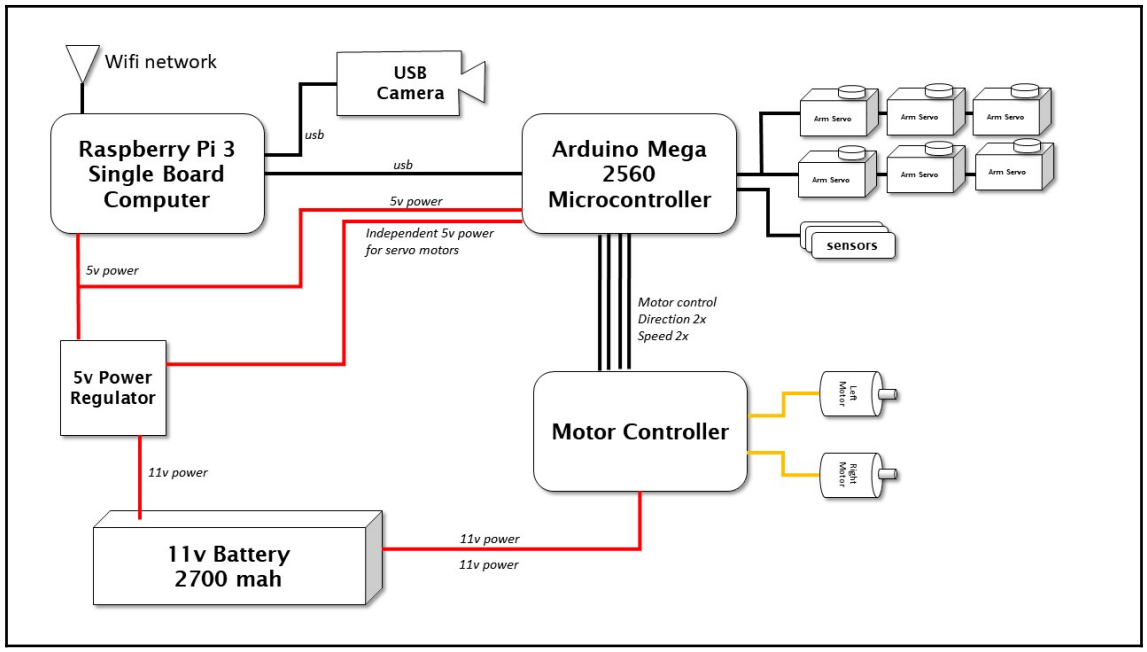
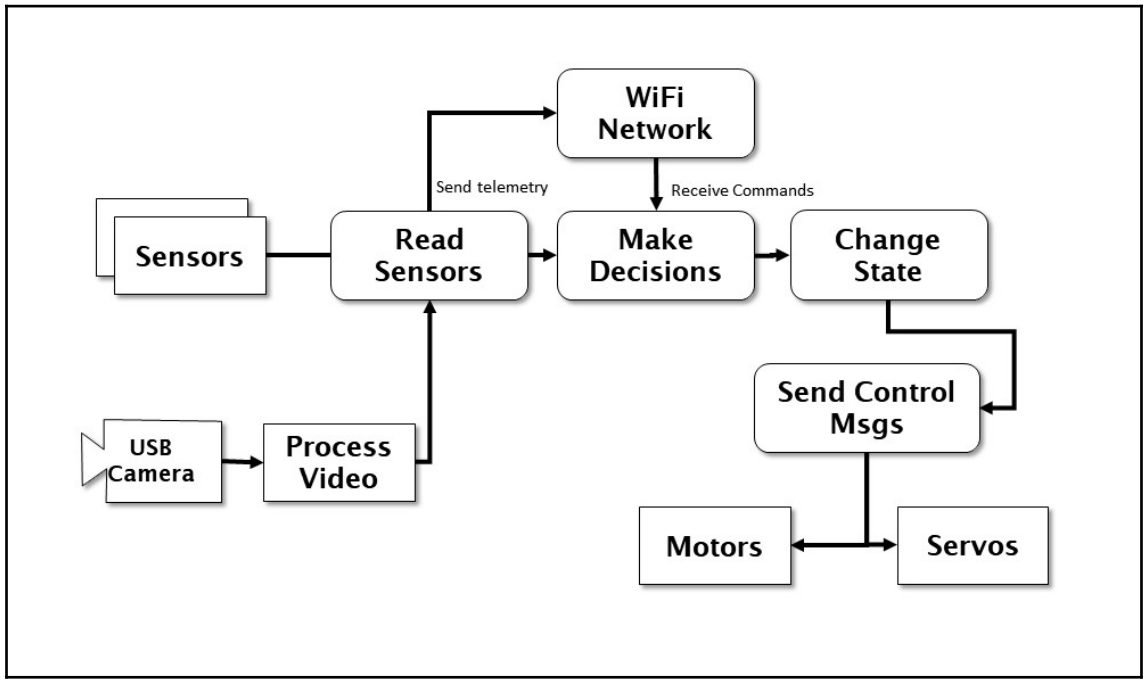
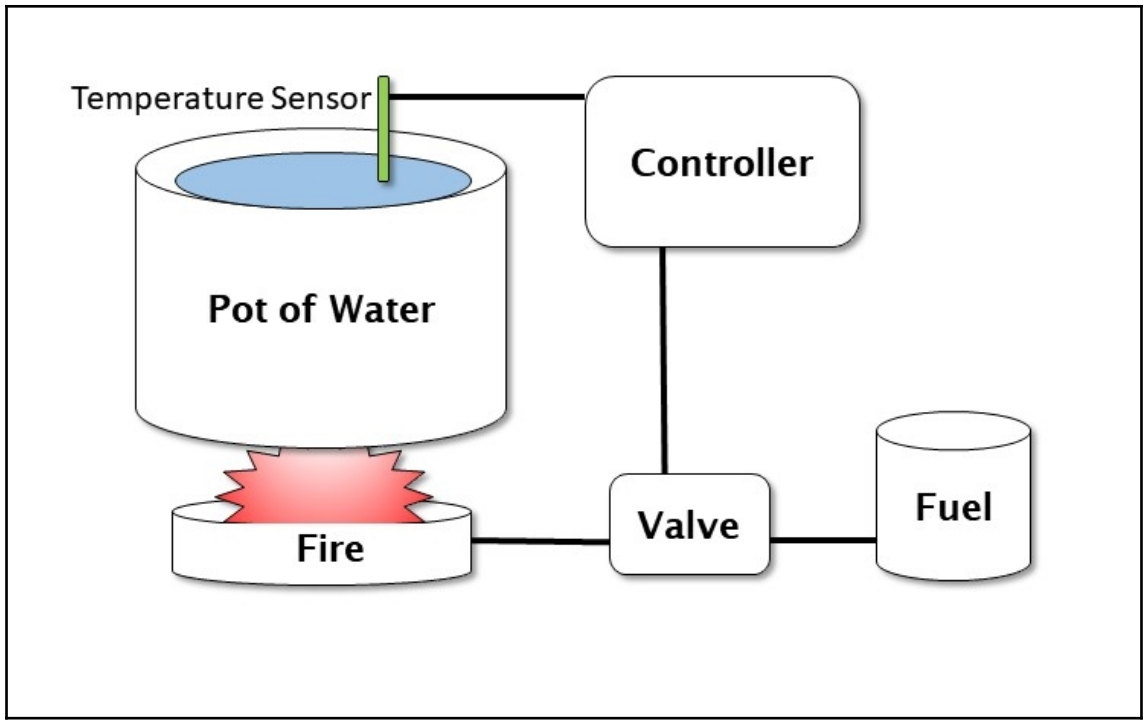
Chapter 1: Foundation for Advanced Robotics and AI





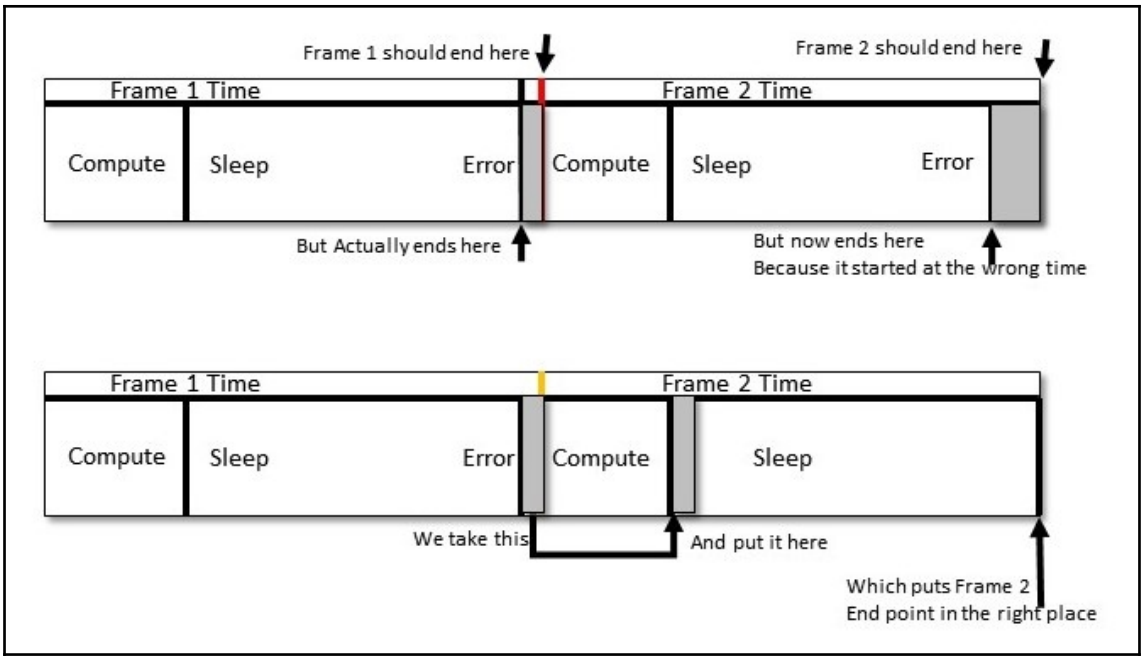

Wifi network
USB Camera
Raspberry Pi 3 Single Board Computer
usb
usb
Arduino Mega 2560 Microcontroller
Arm Servo
Arm Servo
Arm Servo
Arm Servo
Arm Servo
Arm Servo
sensors
5v power
Independent 5v power for servo motors
5v power
Motor control
Direction 2x
Speed 2x
5v Power Regulator
Motor Controller
Left Motor
Right Motor
11v power
11v Battery 2700 mah
11v power
11v power

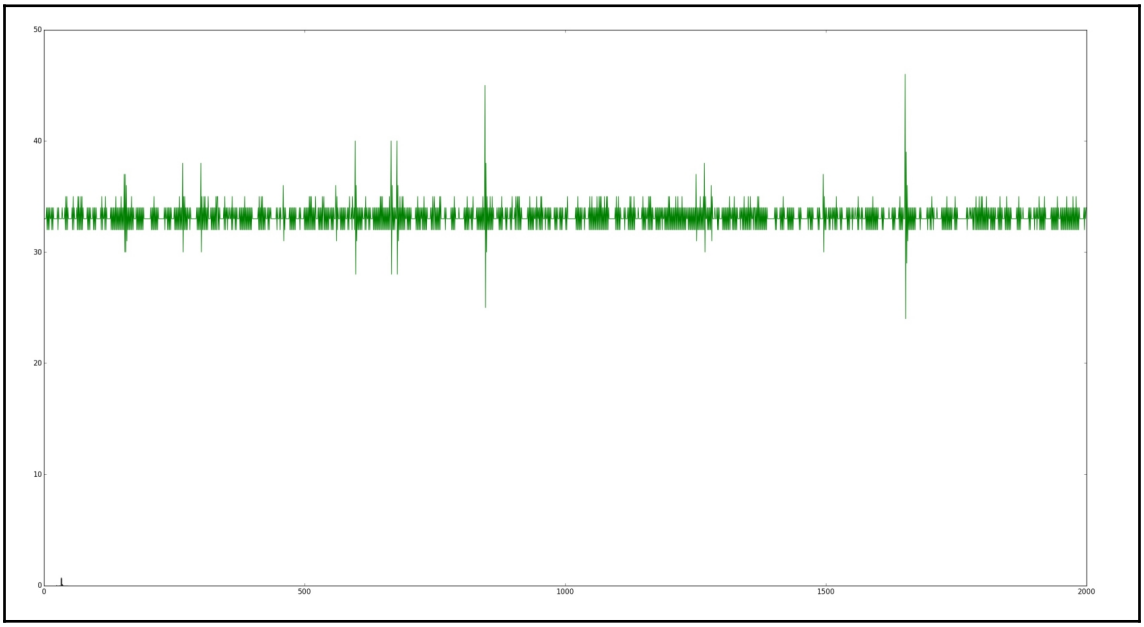

WiFi Network
Send telemetry
Receive Commands
Sensors
Read Sensors
Make Decisions
Change State
Send Control Msgs
USB Camera
Process Video
Motors
Servos



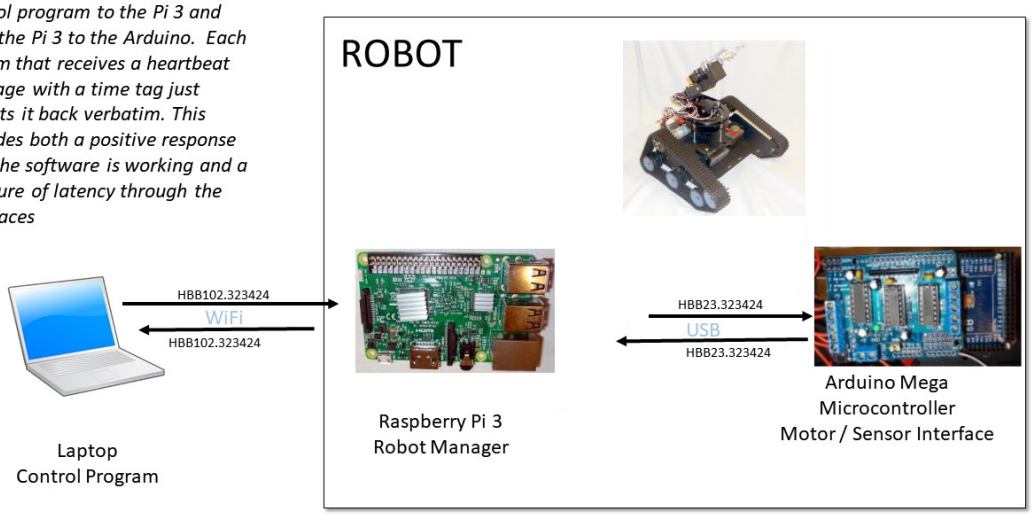
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
A		A		A		A		A		A		A		A		A		A		A		A		A		A		A	
B				B				B				B				B				B				B				B	
C			C			C			C			C			C			C			C			C			C		
D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
A		A		A		A		A		A		A		A		A		A		A		A		A		A		A	
	B				B				B				B				B				B				B				B
C			C			C			C			C			C			C			C			C			C		
D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D

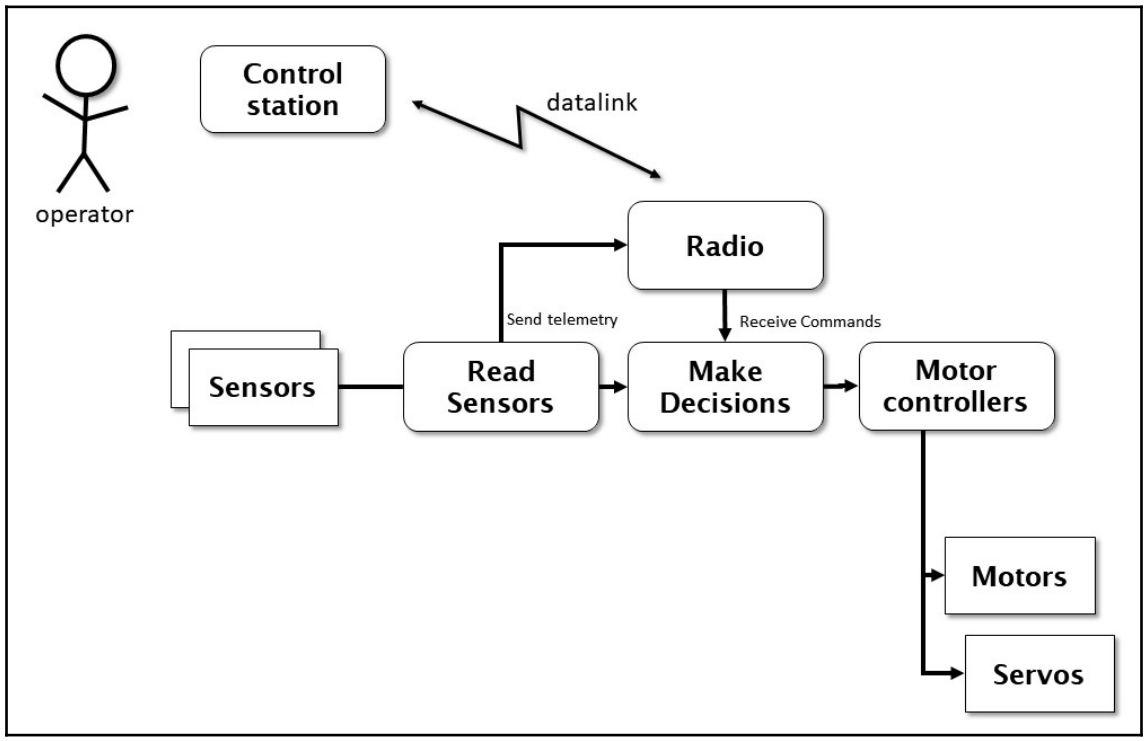

Frame 1 should end here
Frame 2 should end here
Frame 1 Time
Frame 2 Time
Compute
Sleep
Error
Compute
Sleep
Error
But Actually ends here
But now ends here
Because it started at the wrong time
Frame 1 Time
Frame 2 Time
Compute
Sleep
Error
Compute
Sleep
We take this
And put it here
Which puts Frame 2
End point in the right place

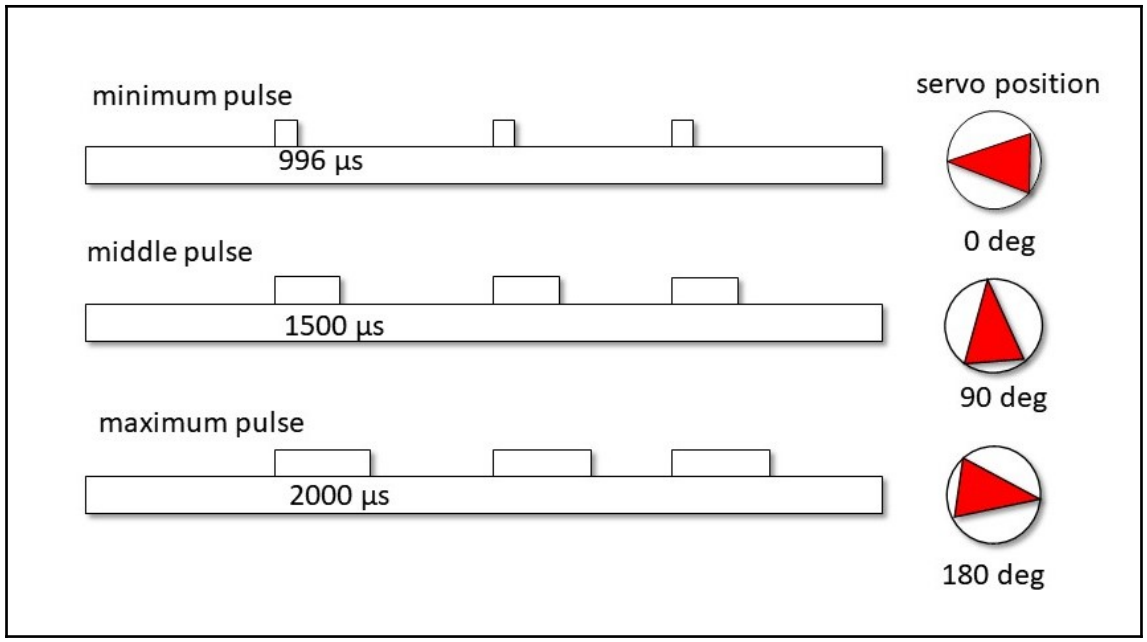

50
40
30
20
10
0
0
500
1000
1500
2000

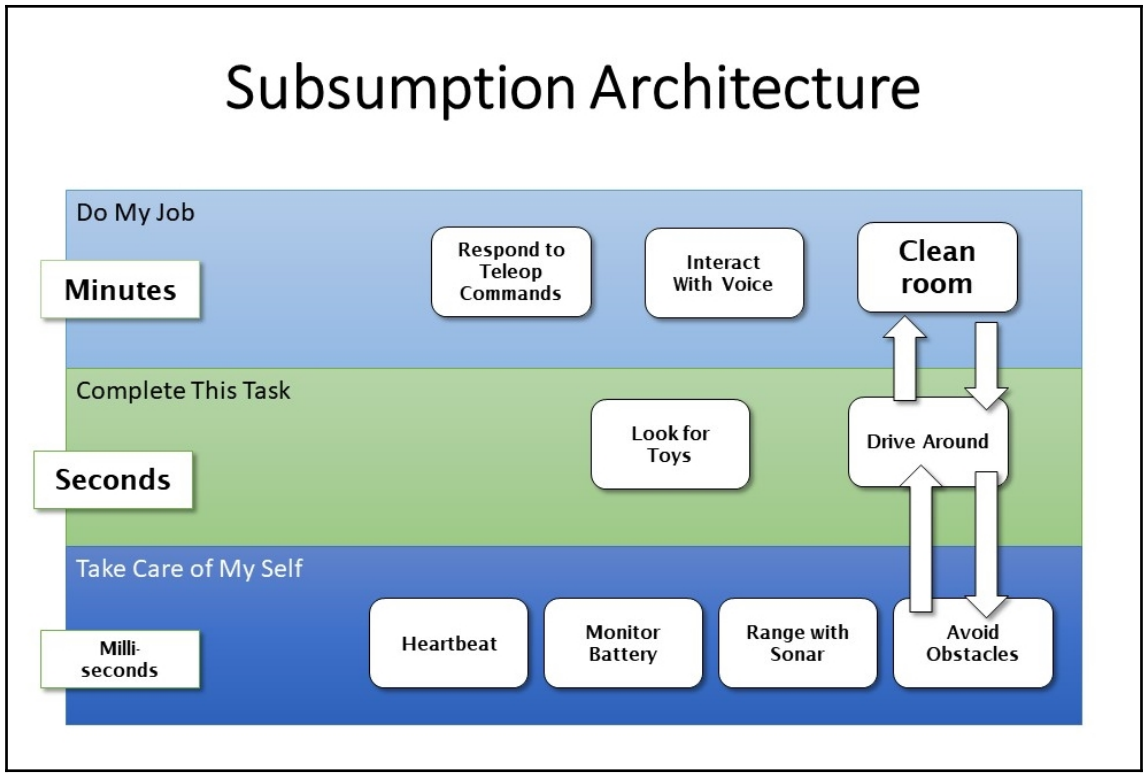
Heartbeat messages go from the control program to the Pi 3 and from the Pi 3 to the Arduino. Each system that receives a heartbeat message with a time tag just repeats it back verbatim. This provides both a positive response that the software is working and a measure of latency through the interfaces

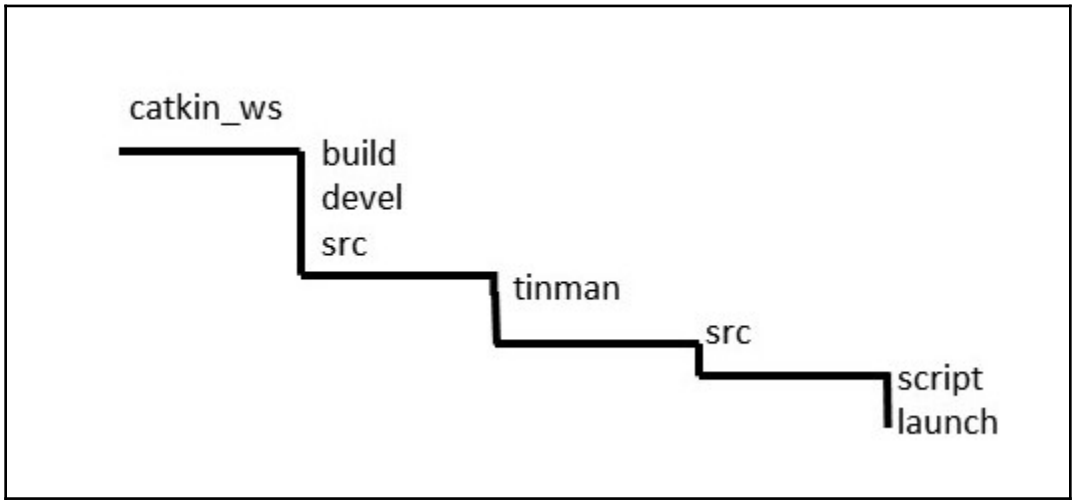


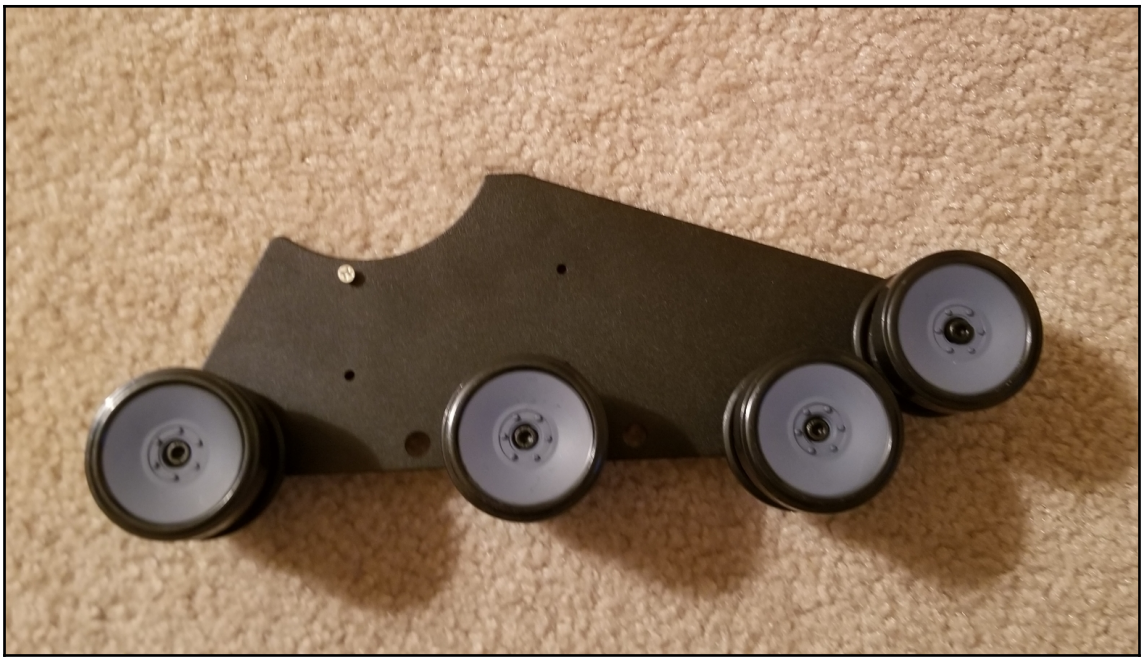
Chapter 2: Setting Up Your Robot

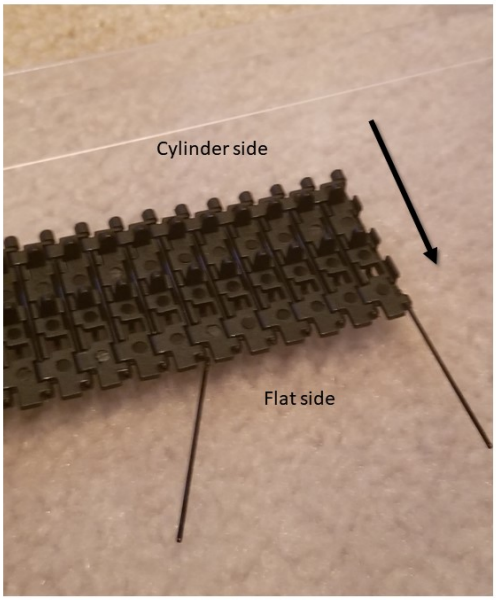


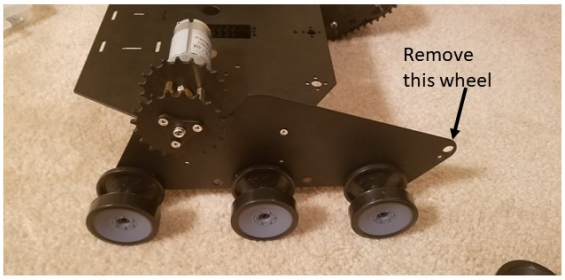

minimum pulse
996 µs
middle pulse
1500 µs
maximum pulse
2000 µs
servo position
0 deg
90 deg
180 deg

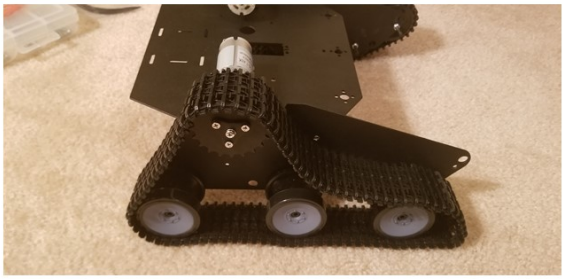

Subsumption Architecture
Do My Job
Minutes
Respond to Teleop Commands
Interact With Voice
Clean room
Complete This Task
Seconds
Look for Toys
Drive Around
Take Care of My Self
Milli-seconds
Heartbeat
Monitor Battery
Range with Sonar
Avoid Obstacles

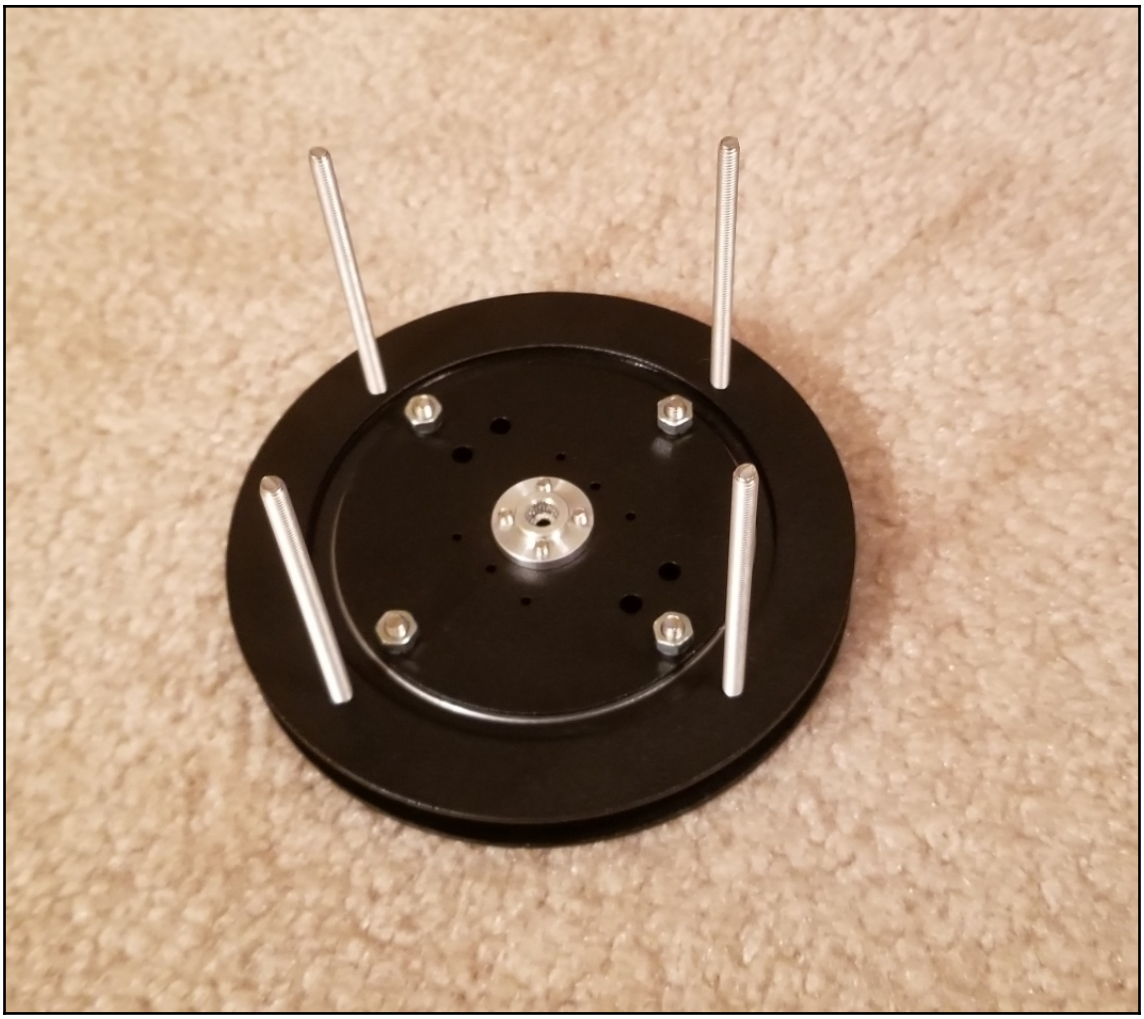

catkin_ws
build
devel
src
tinman
src
script
launch

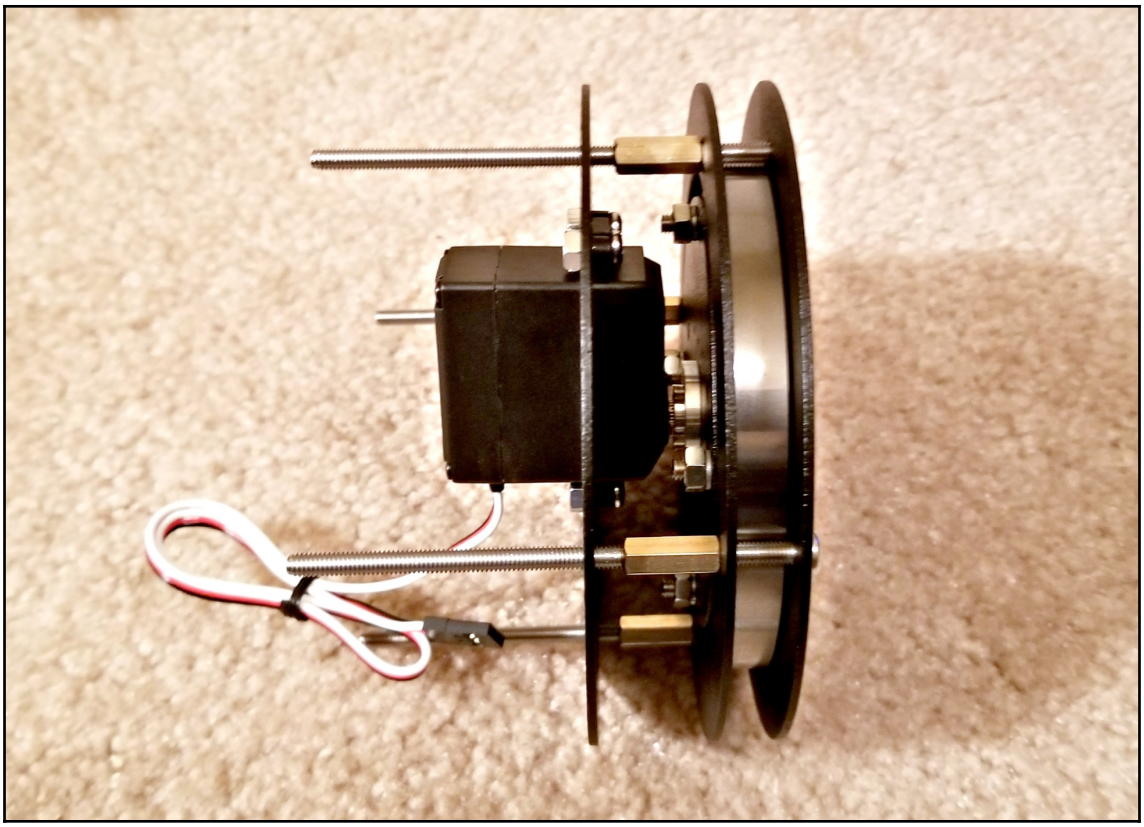


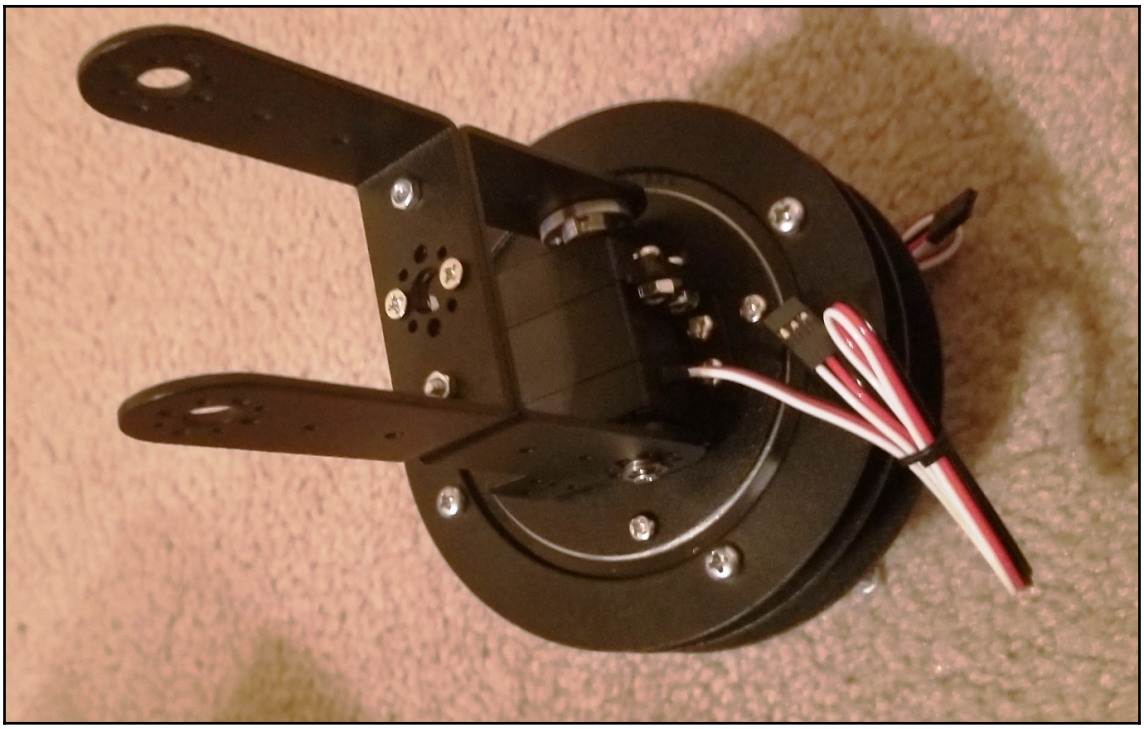

Cylinder side
Flat side

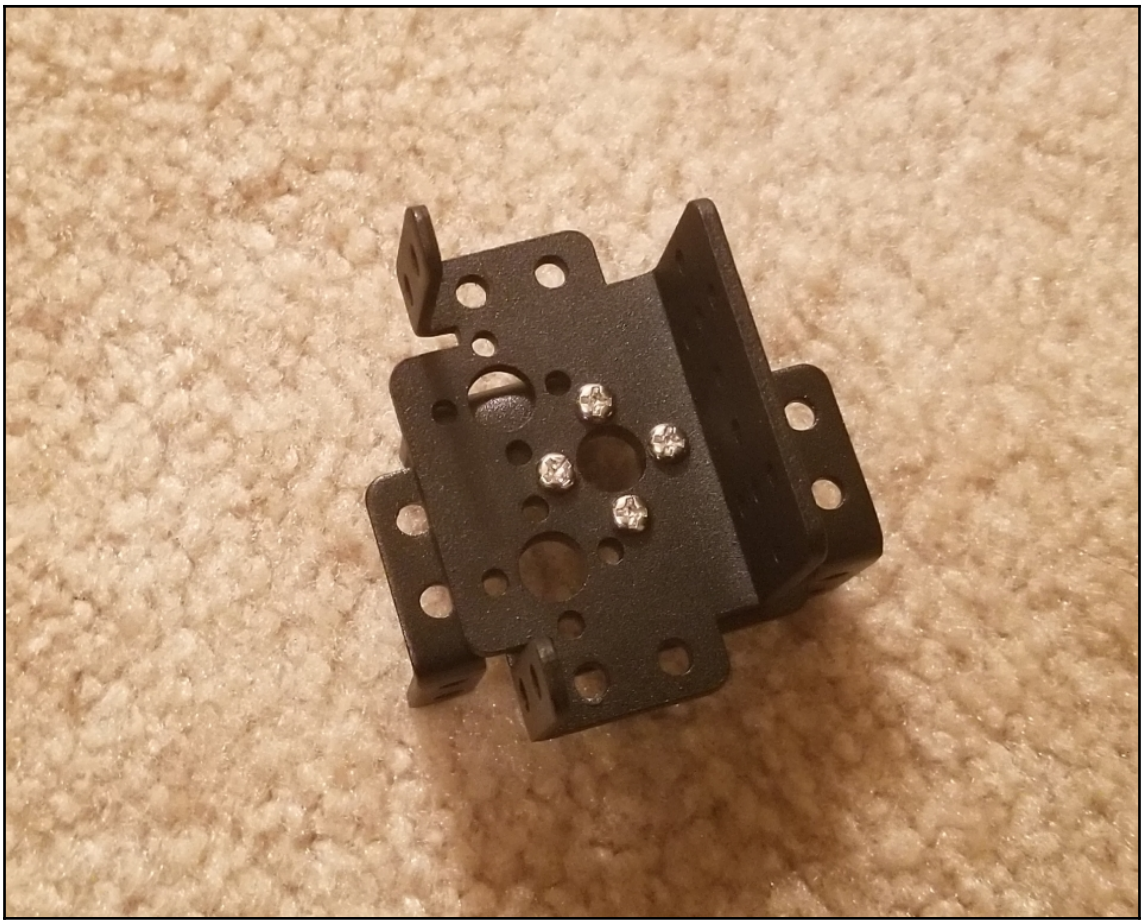

Remove
this wheel

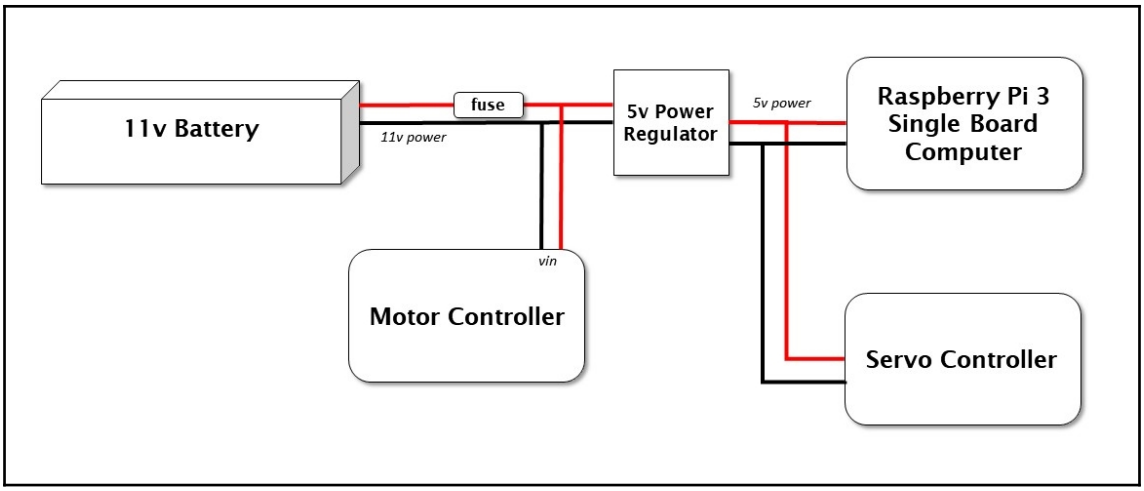




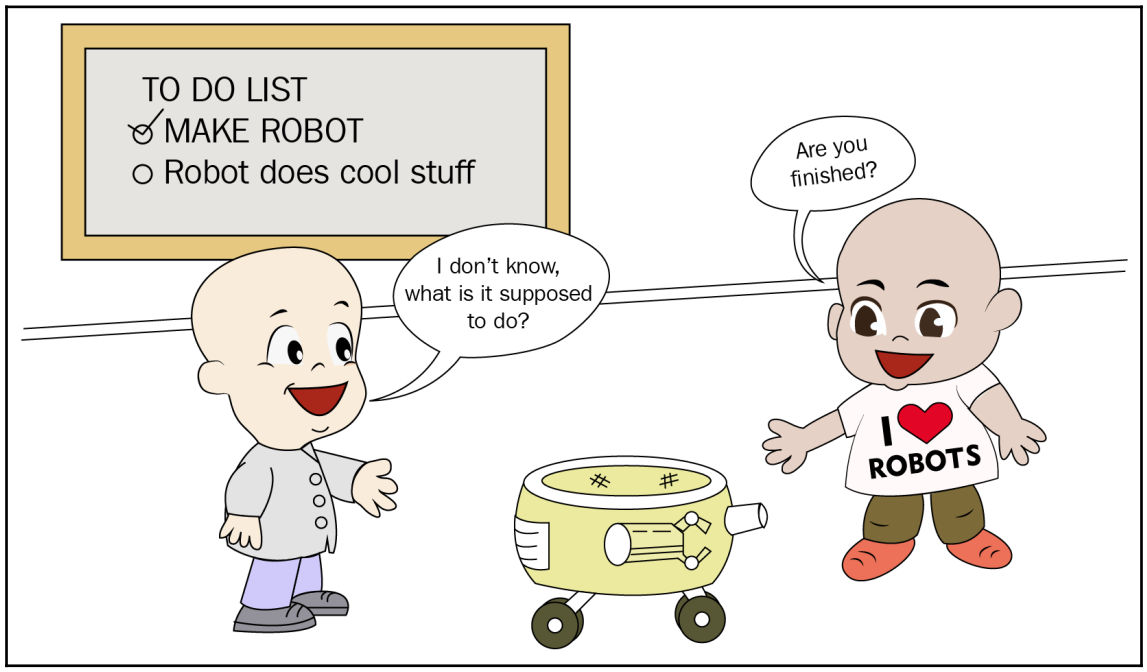


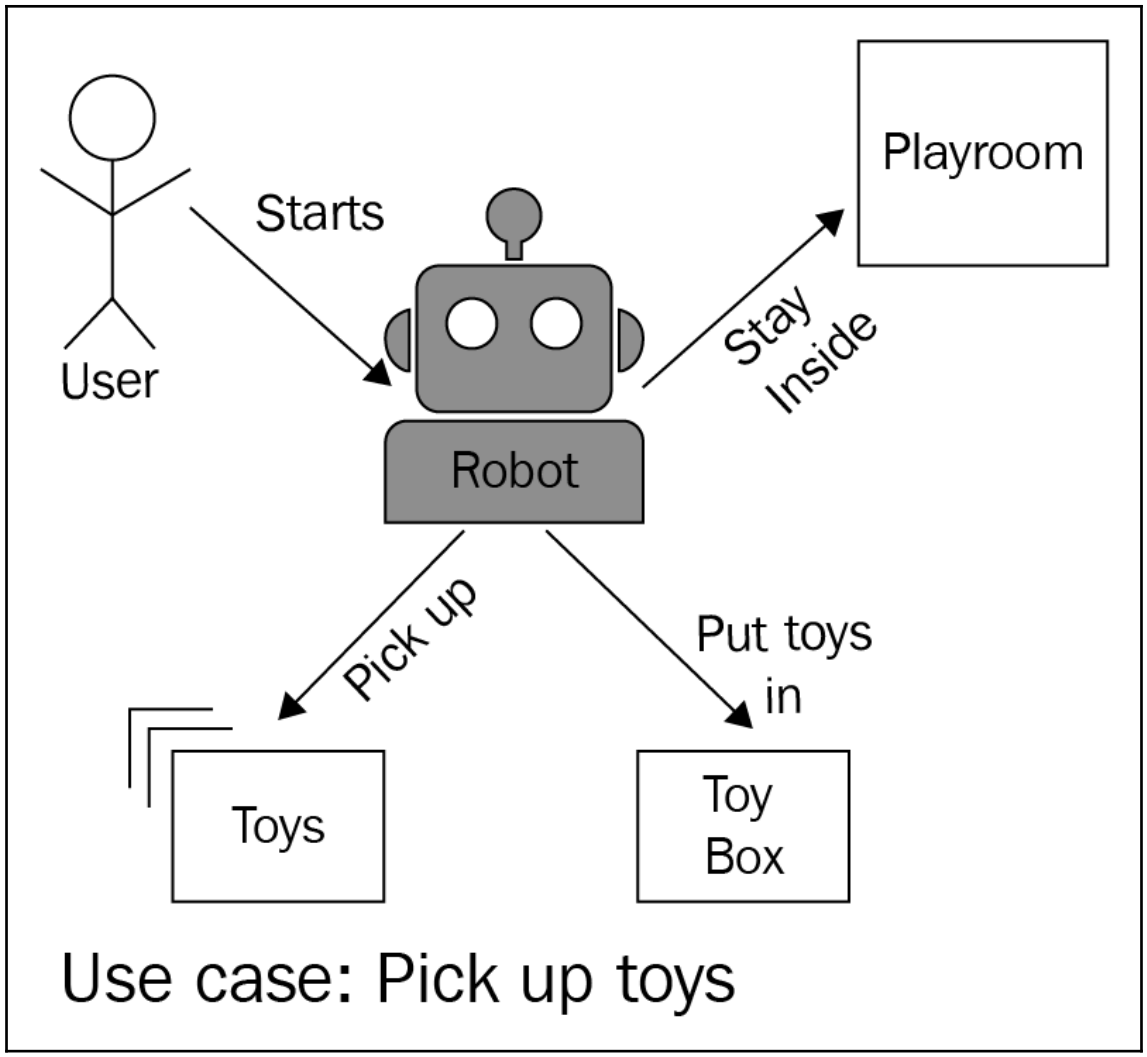


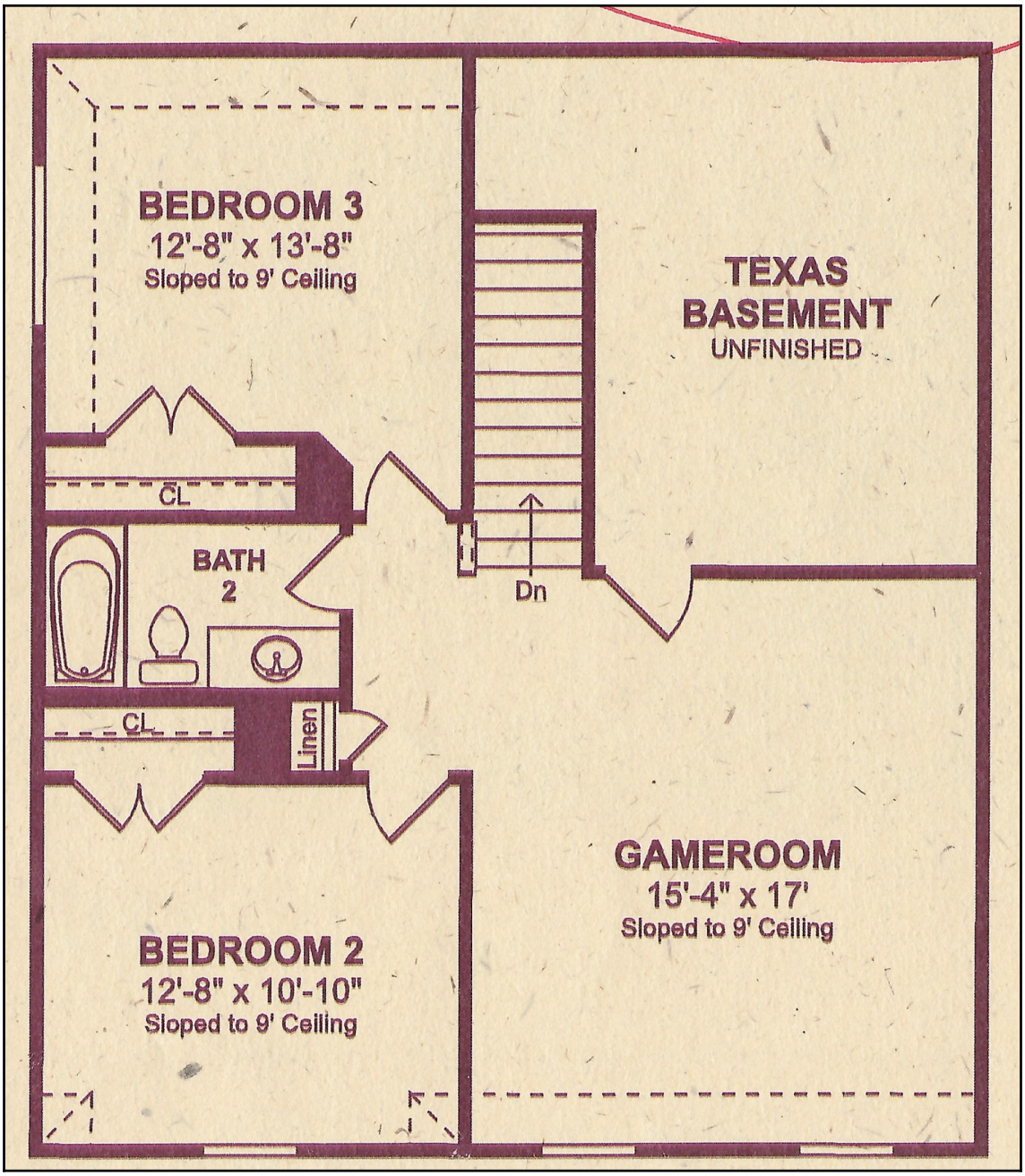


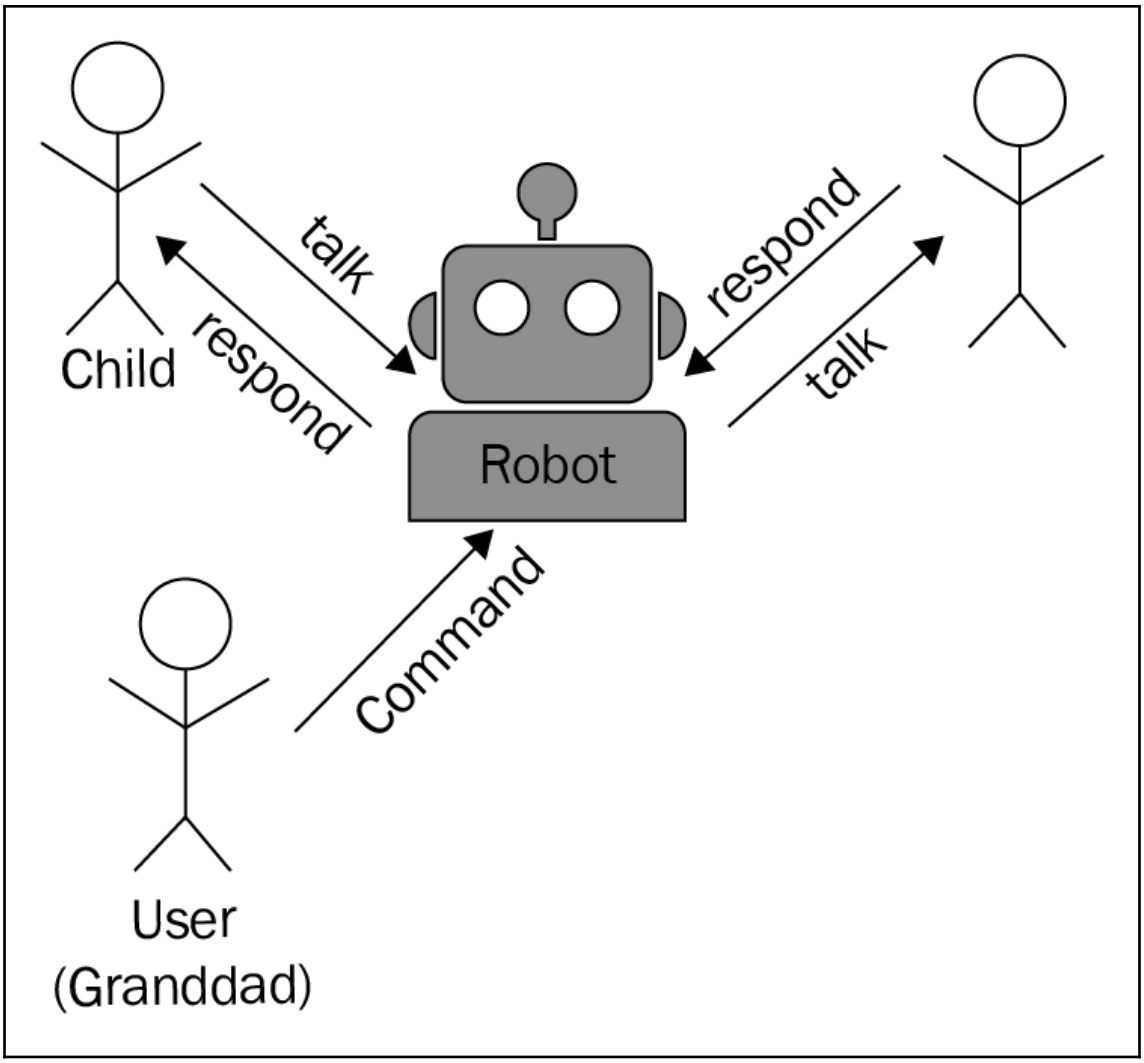

11v Battery
fuse
11v power
5v Power Regulator
5v power
Raspberry Pi 3 Single Board Computer
vin
Motor Controller
Servo Controller

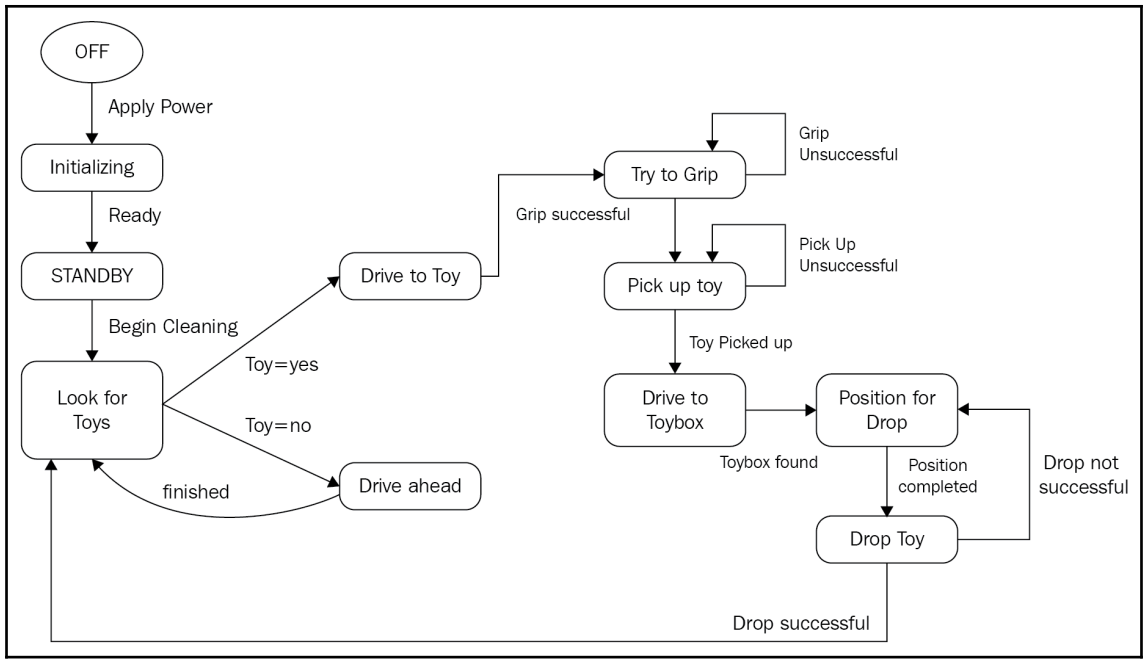
Chapter 3: A Concept for a Practical Robot Design Process

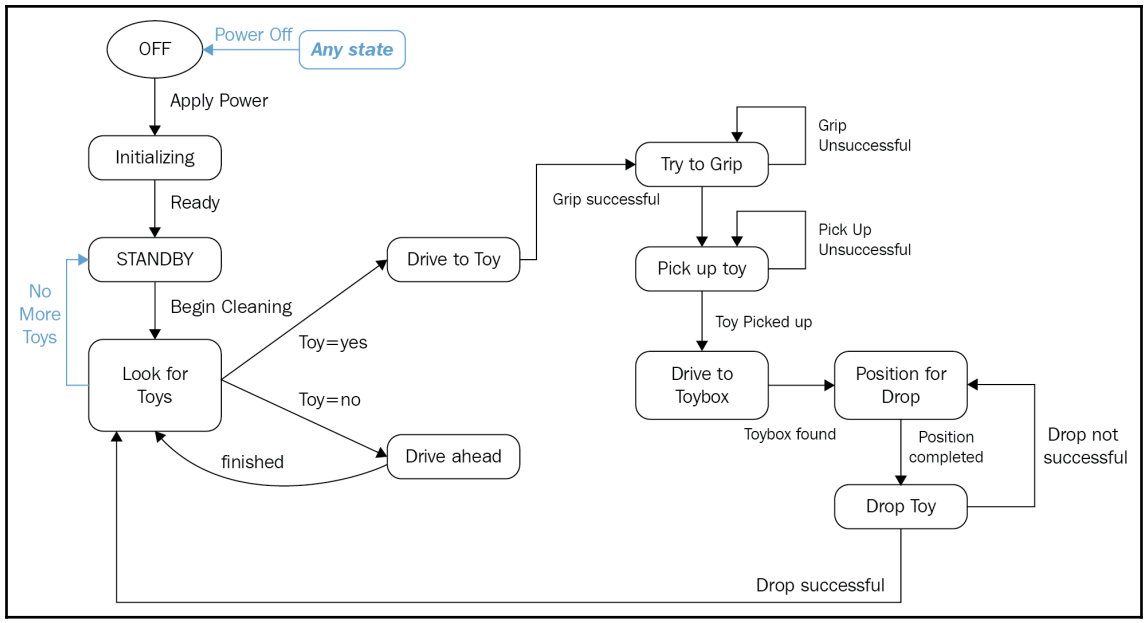


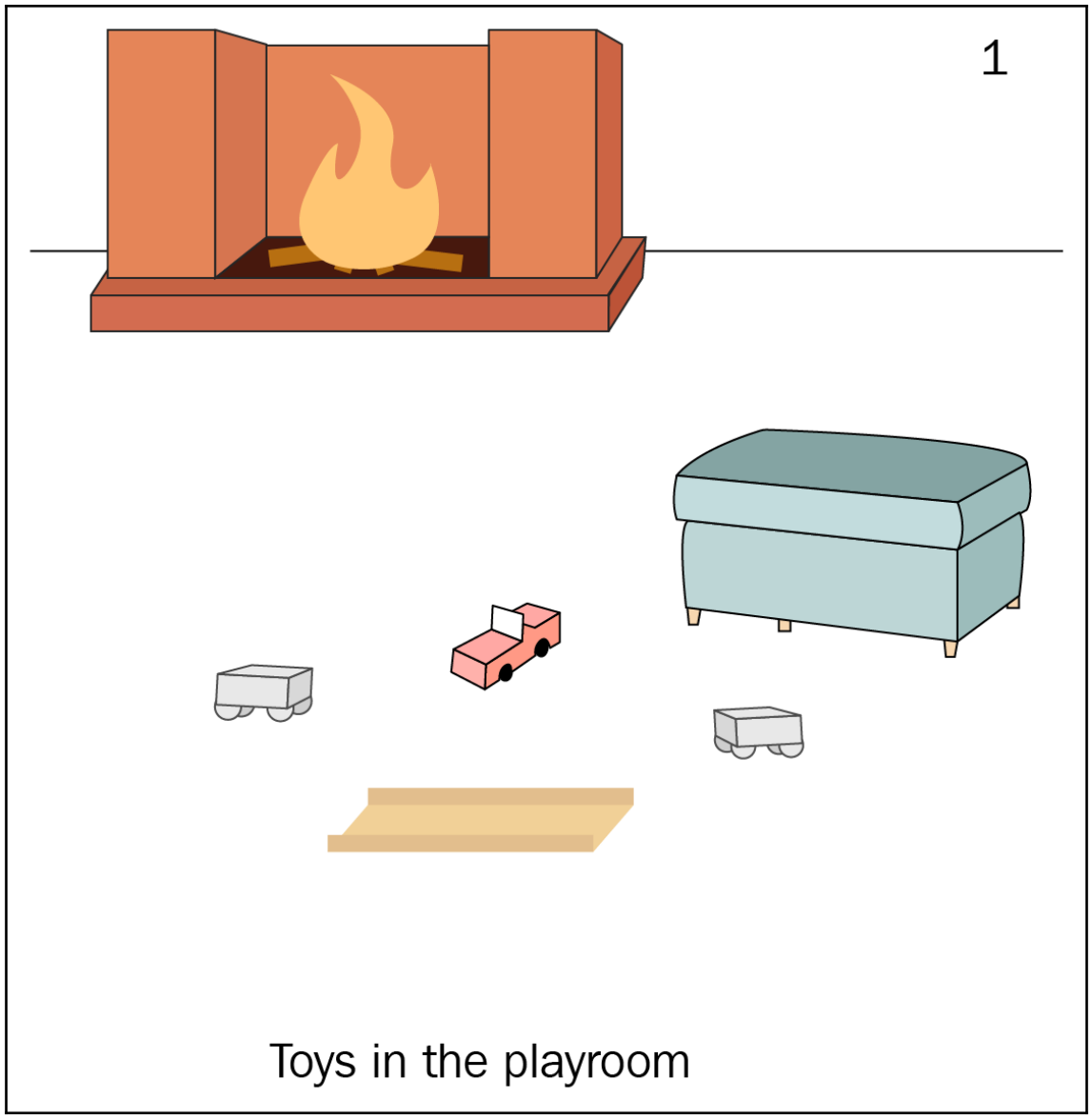

Playroom
Starts
User
Stay
Inside
Robot
Pick up
Put toys
in
Toys
Toy
Box
Use case: Pick up toys


BEDROOM 3
12'-8" x 13'-8"
Sloped to 9' Ceiling
TEXAS
BASEMENT
UNFINISHED
CL
BATH
2
Dn
CL
Linen
GAMEROOM
15'-4" x 17'
Sloped to 9' Ceiling
BEDROOM 2
12'-8" x 10'-10"
Sloped to 9' Ceiling

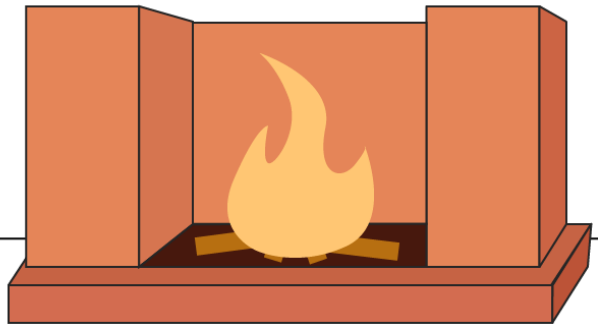

talk
respond
Child
respond
talk
Robot
Command
User
(Granddad)


OFF
Apply Power
Initializing
Ready
STANDBY
Begin Cleaning
Look for Toys
Toy=yes
Toy=no
finished
Drive to Toy
Drive ahead
Grip successful
Try to Grip
Grip Unsuccessful
Pick up toy
Pick Up Unsuccessful
Toy Picked up
Drive to Toybox
Toybox found
Position for Drop
Position completed
Drop not successful
Drop Toy
Drop successful

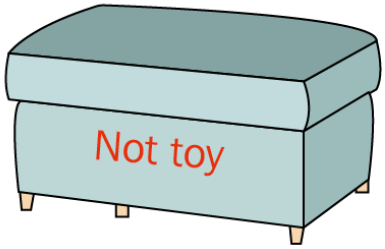

OFF
Power Off
Any state
Apply Power
Initializing
Ready
STANDBY
No More Toys
Begin Cleaning
Look for Toys
Toy=yes
Toy=no
finished
Drive to Toy
Drive ahead
Grip successful
Try to Grip
Grip Unsuccessful
Pick up toy
Pick Up Unsuccessful
Toy Picked up
Drive to Toybox
Toybox found
Position for Drop
Position completed
Drop not successful
Drop Toy
Drop successful


1
Toys in the playroom

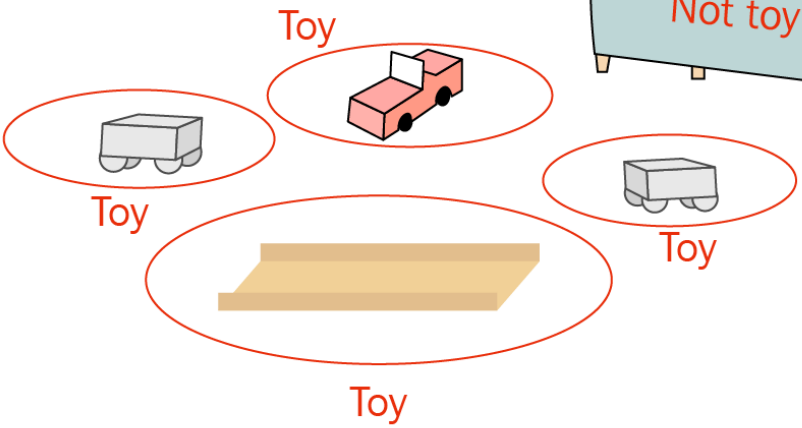
2



Not toy



Not toy



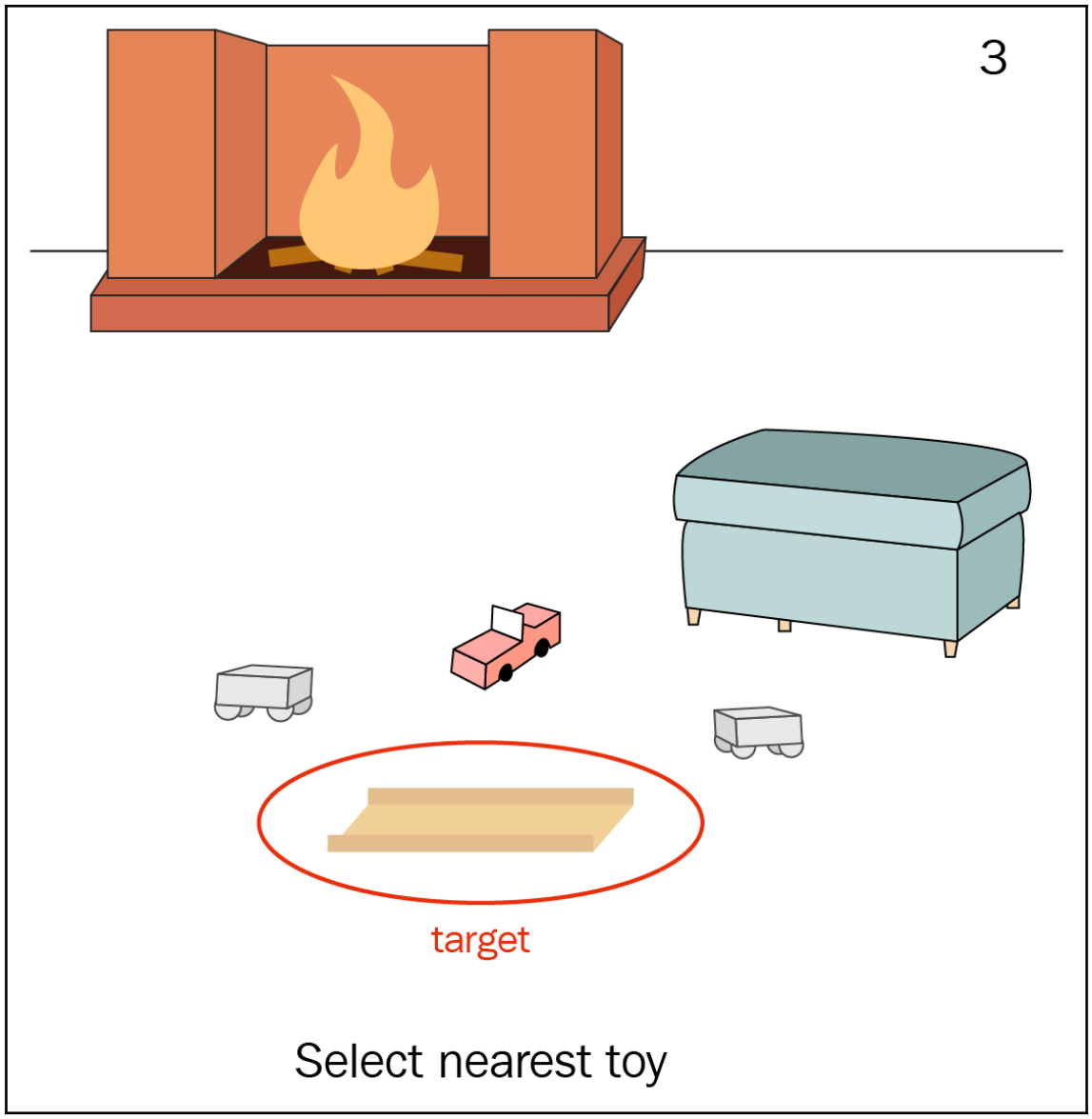
Toy

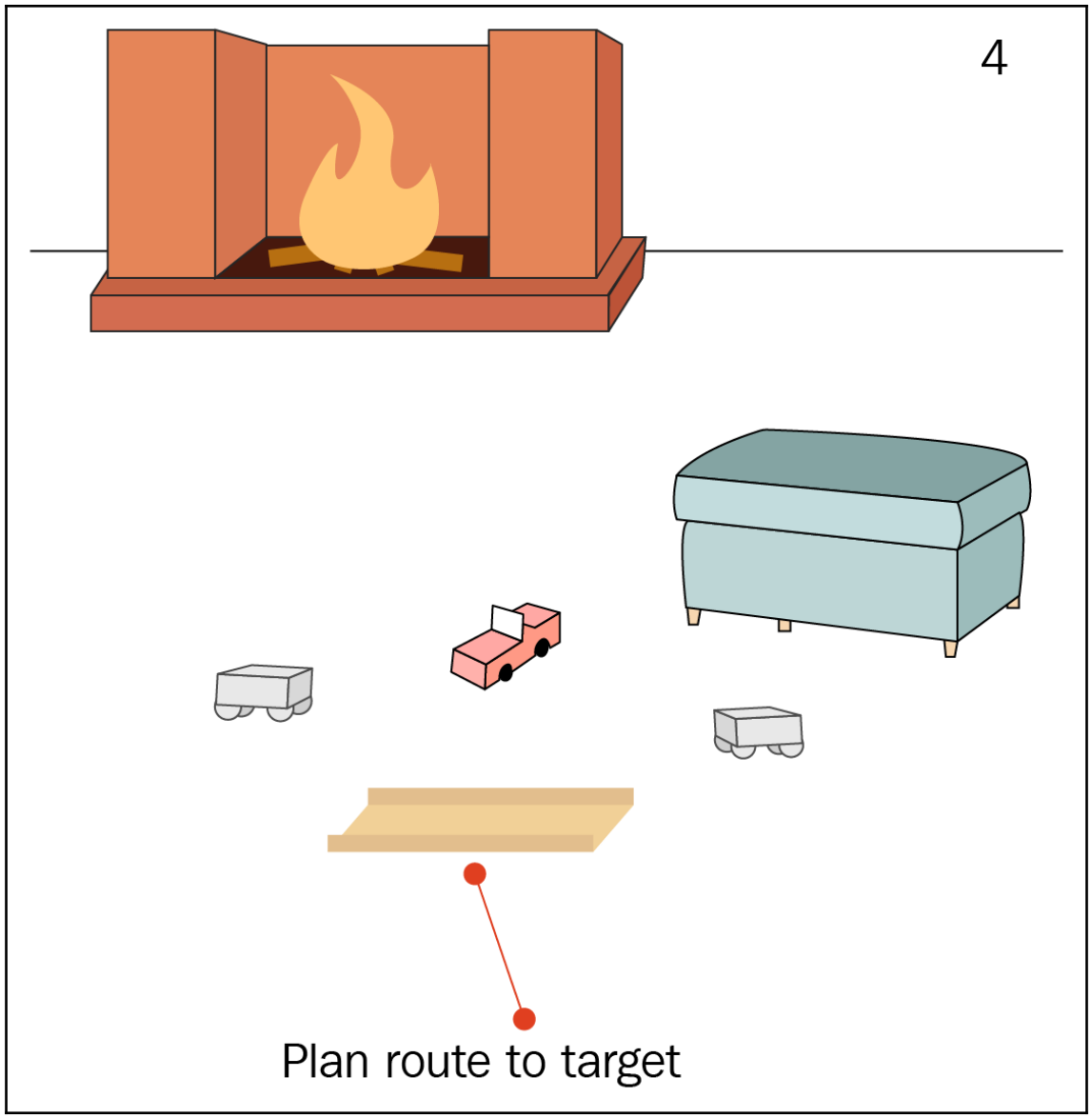
Toy

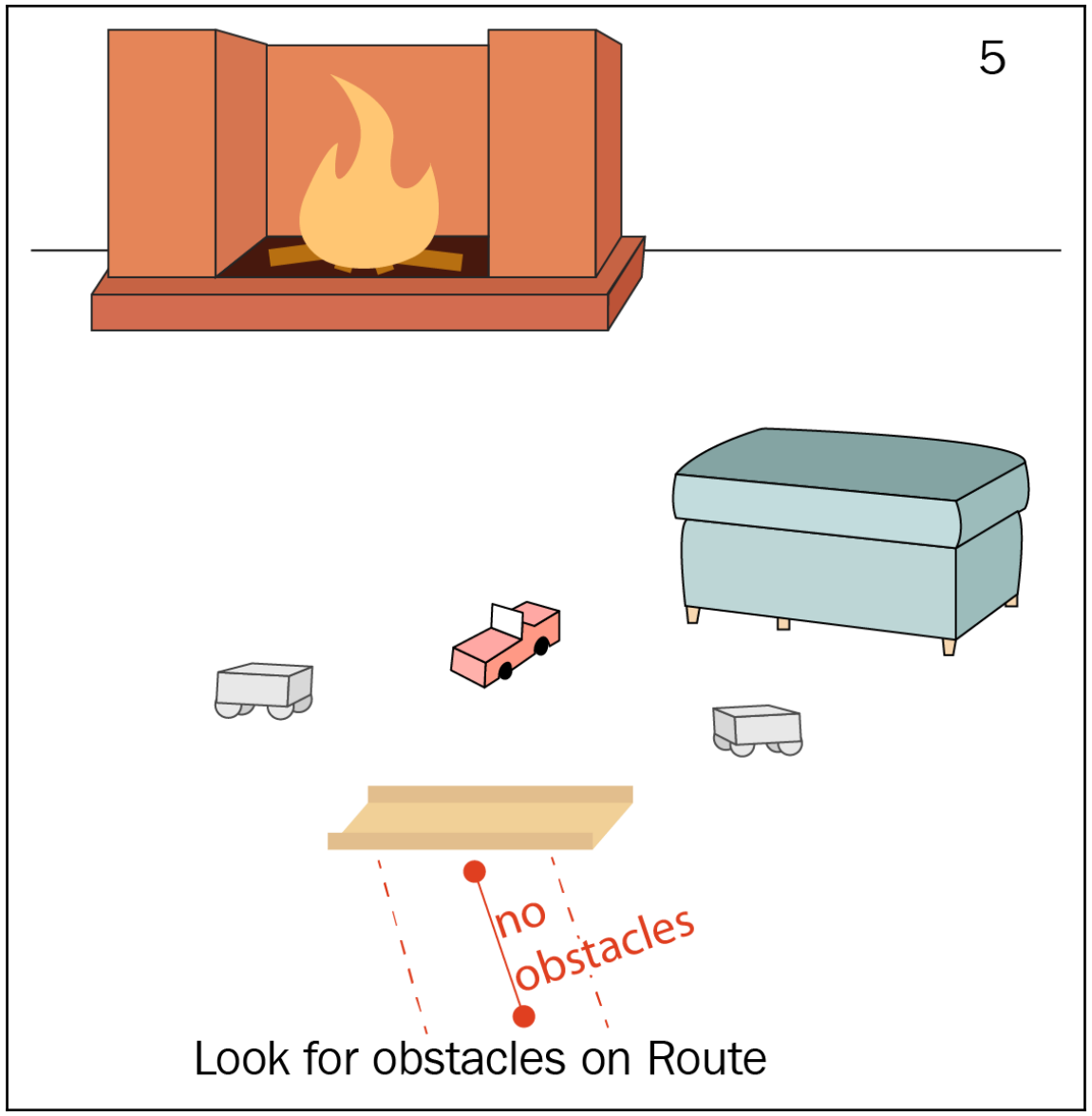
Toy

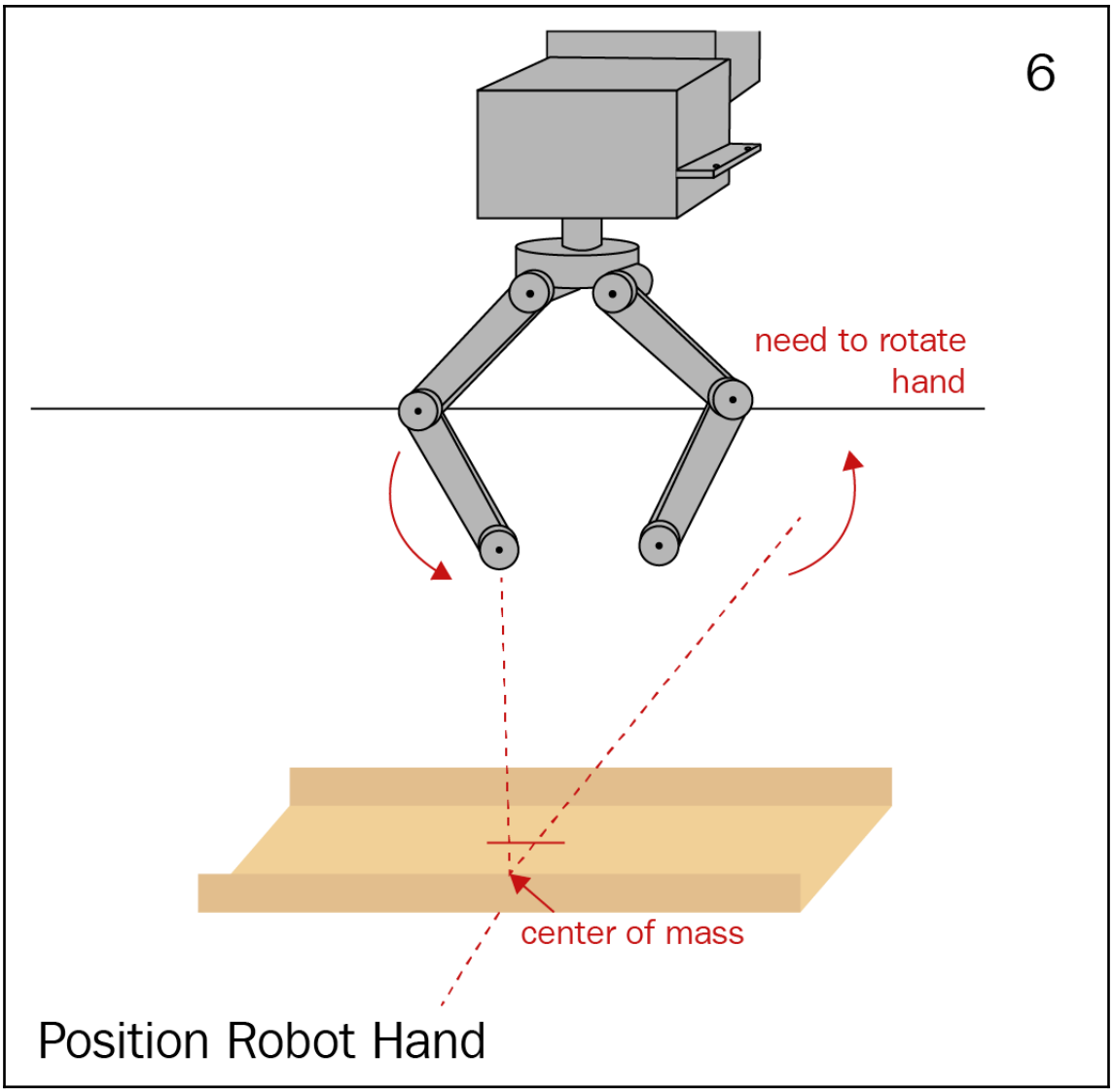
Toy

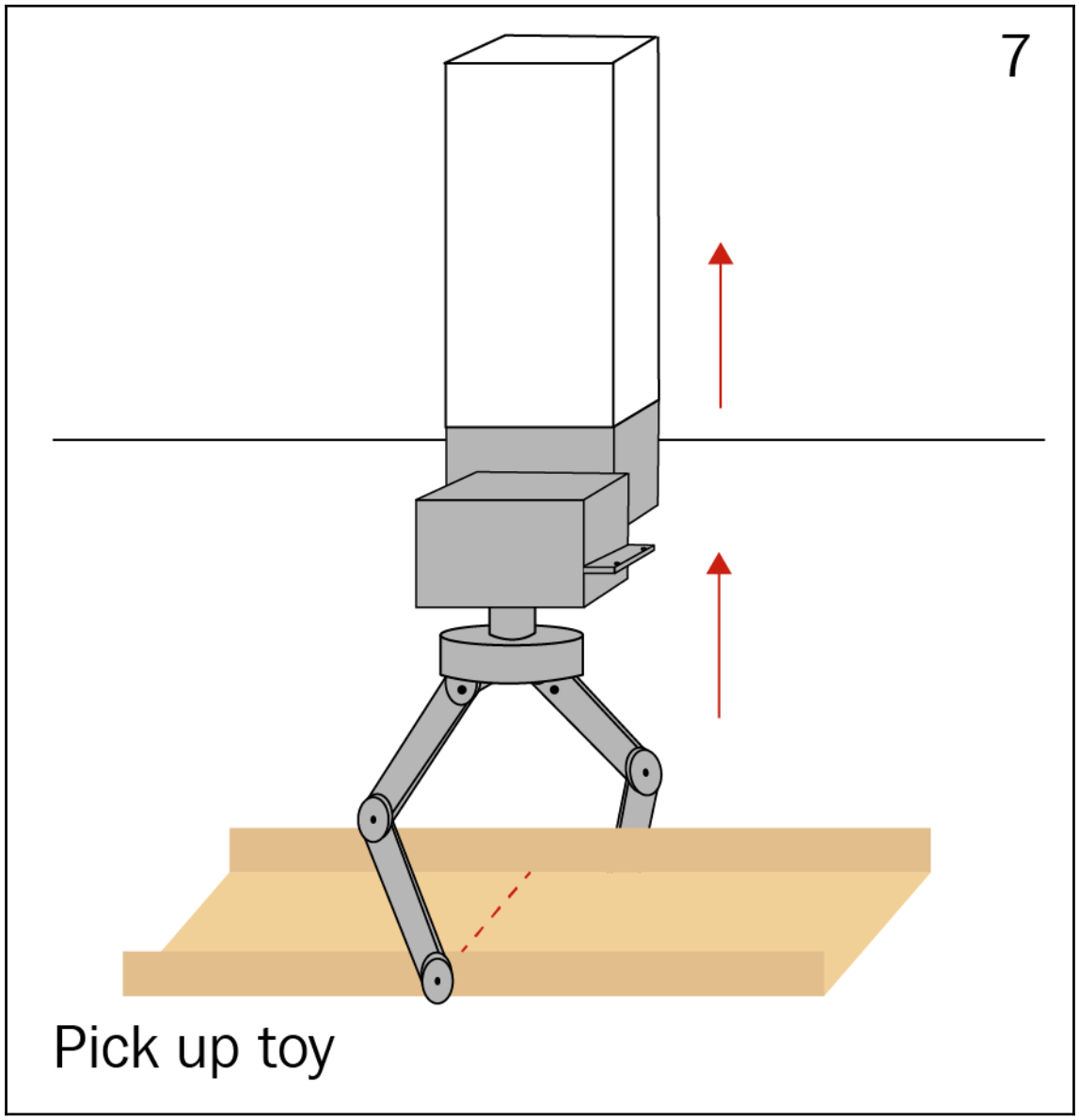
Look for toys

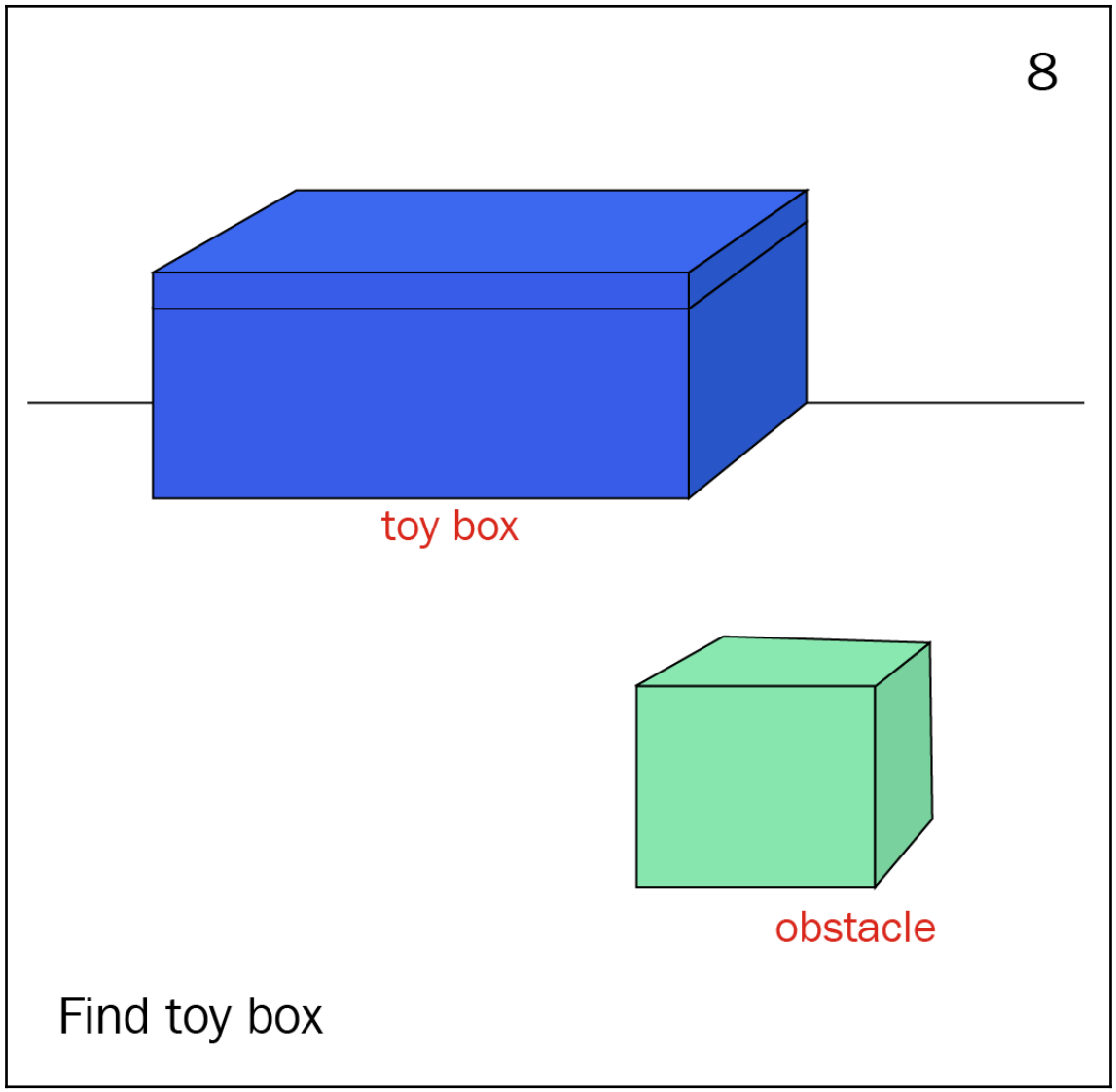

3
target
Select nearest toy

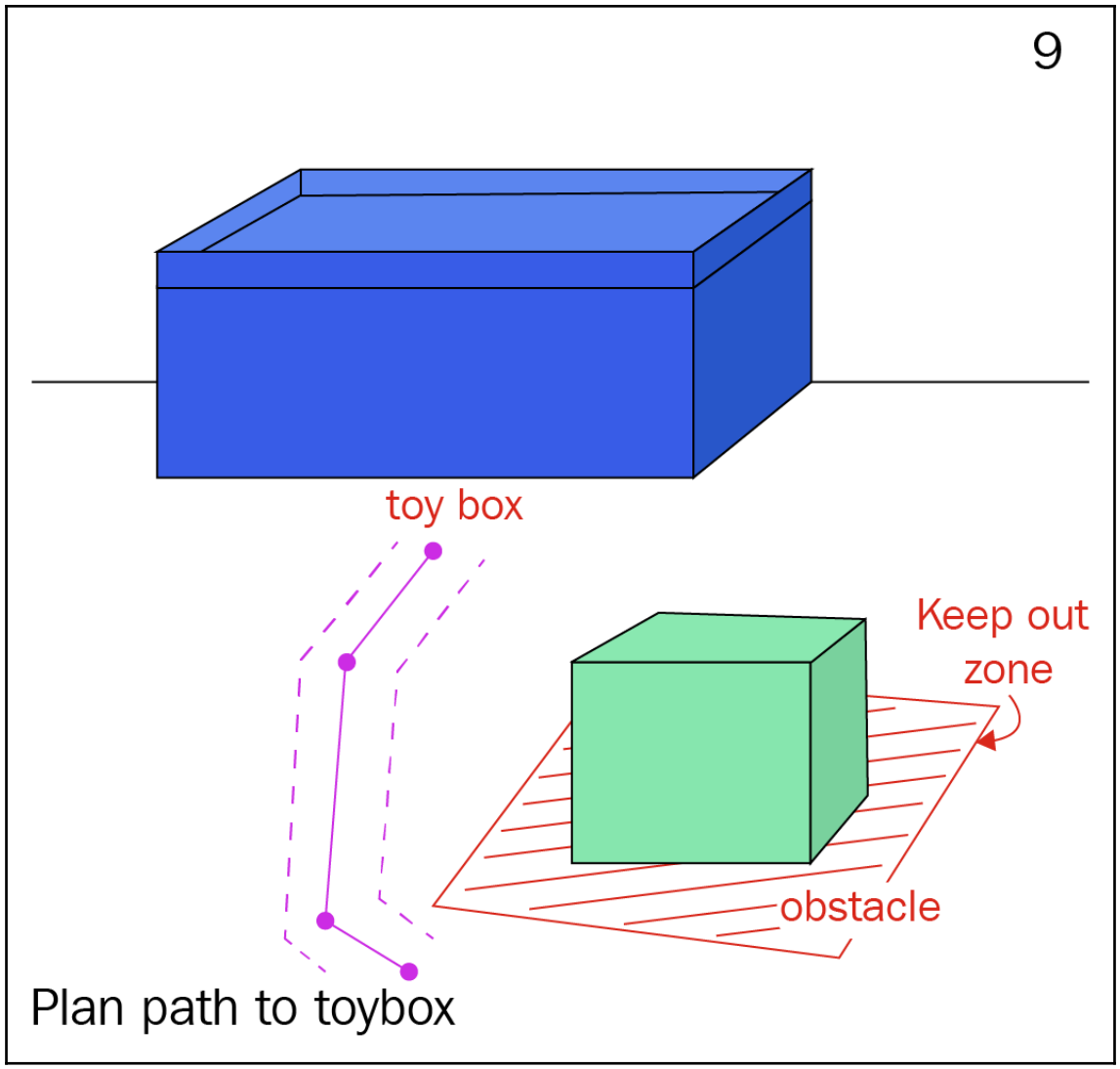

4
Plan route to target

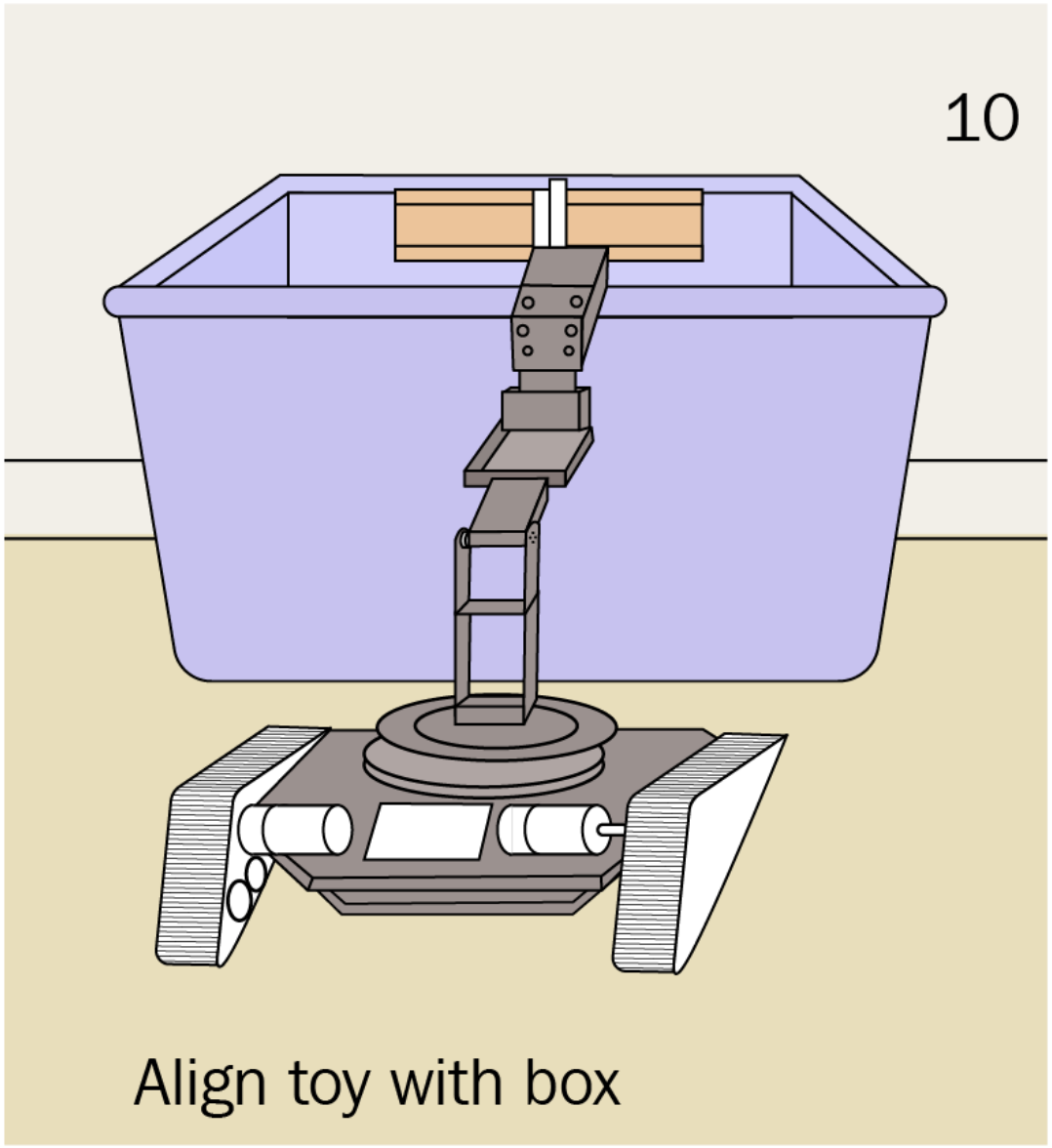

5
no
obstacles
Look for obstacles on Route

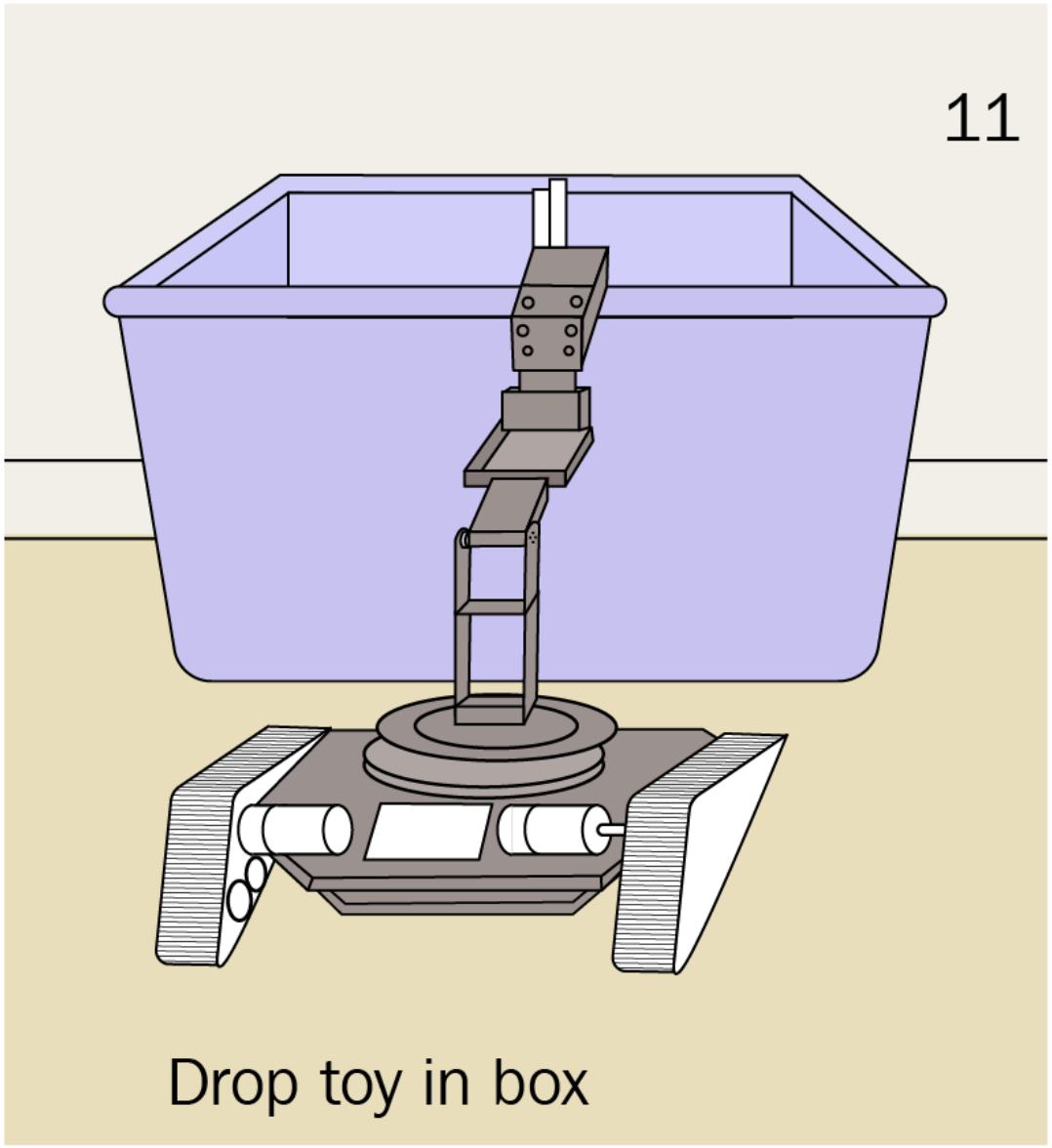

6
need to rotate
hand
center of mass
Position Robot Hand


7
Pick up toy

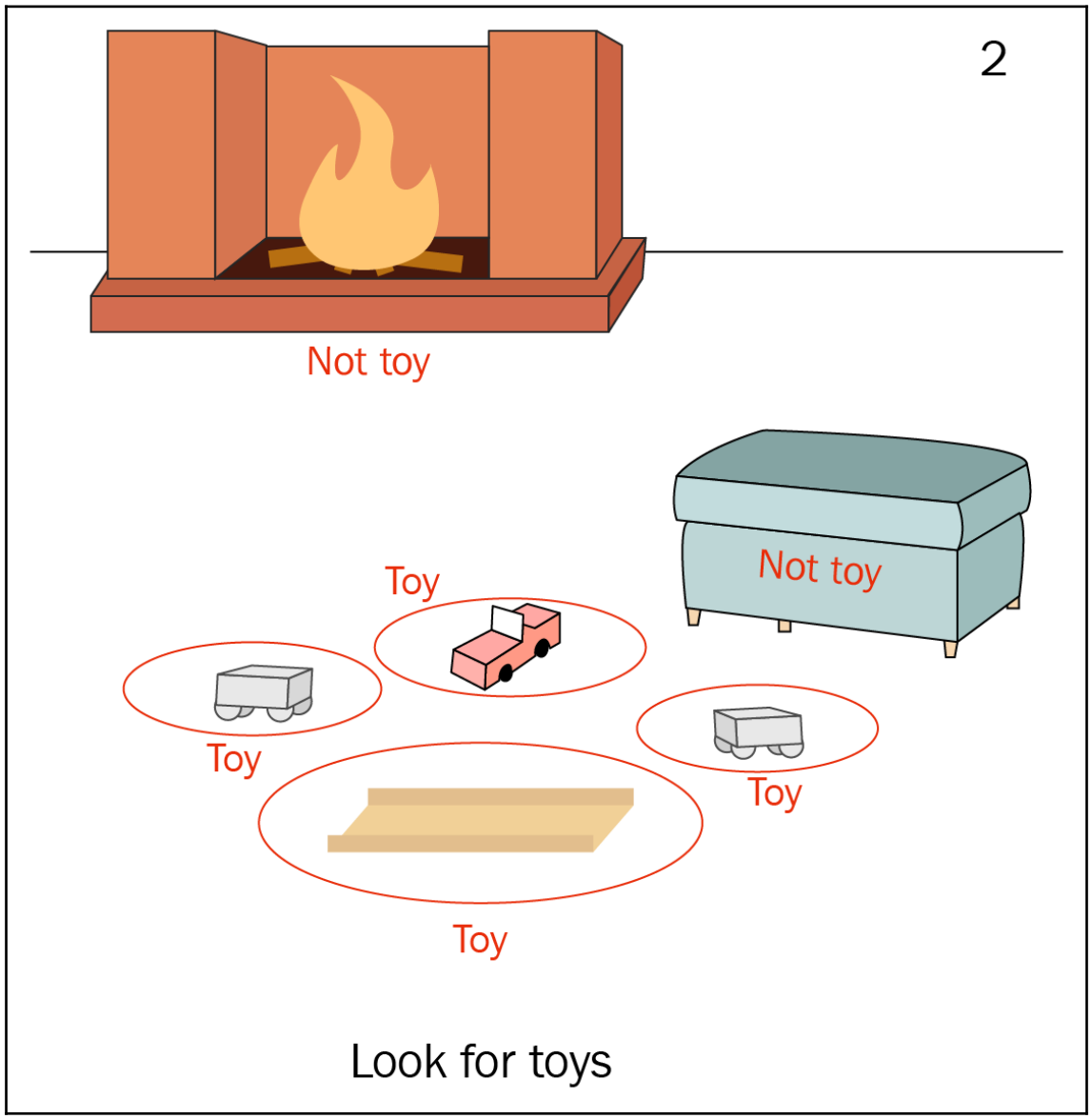

8
toy box
obstacle
Find toy box


9
toy box
Keep out
zone
obstacle
Plan path to toybox

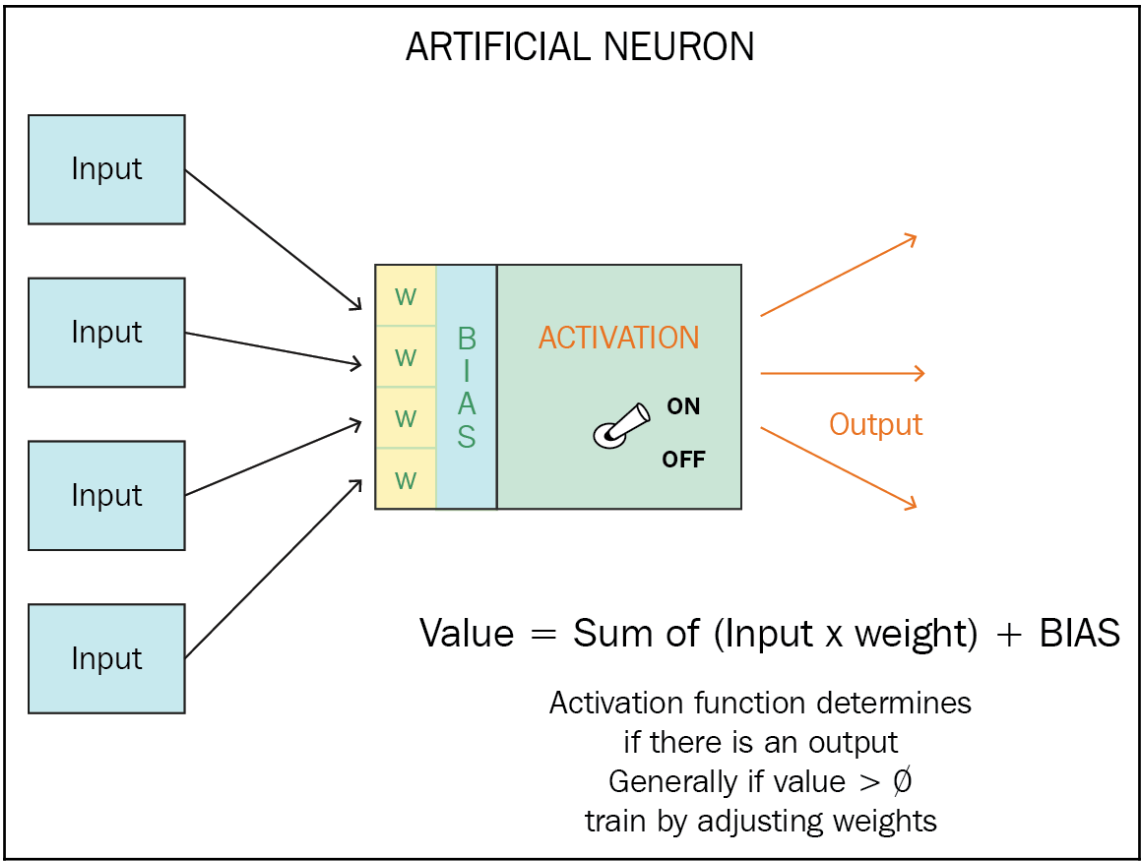

10
Align toy with box

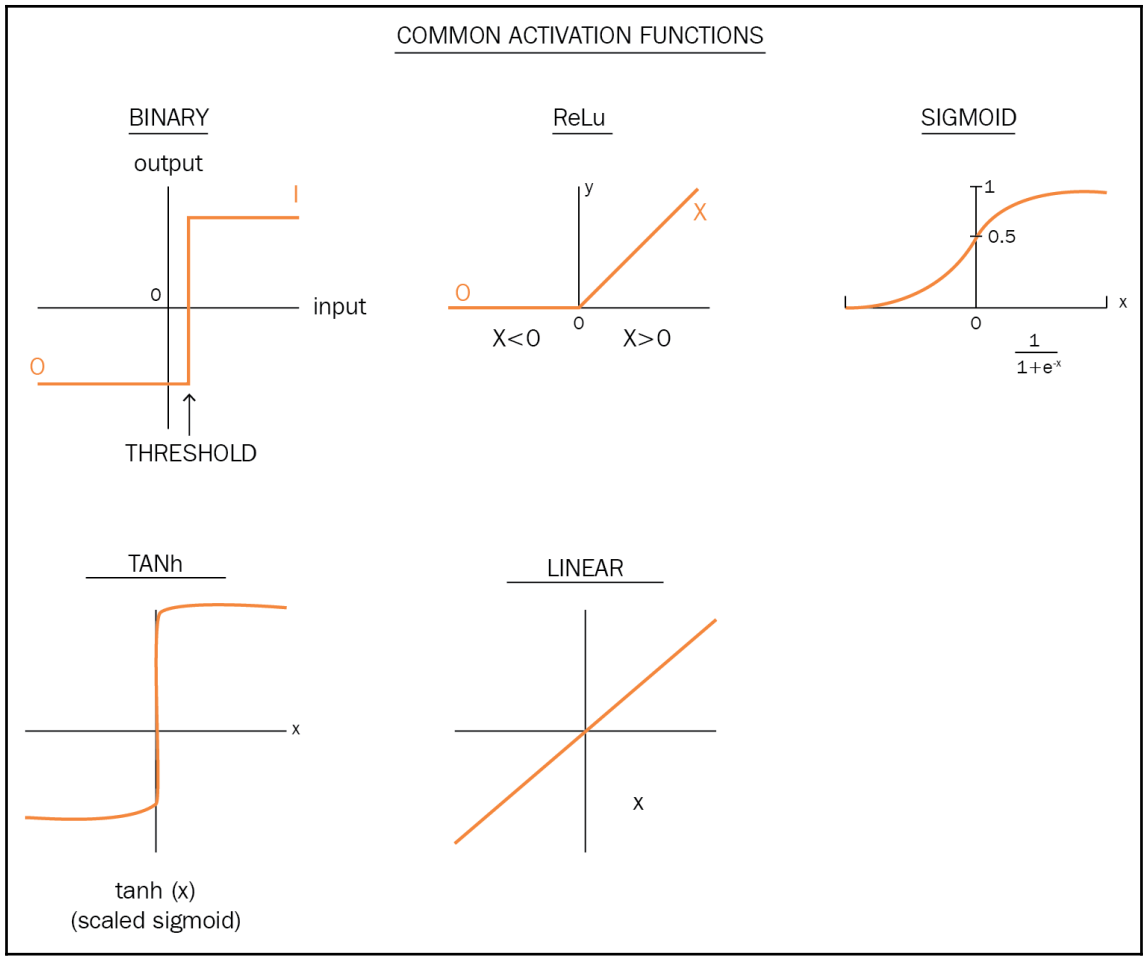

11
Drop toy in box

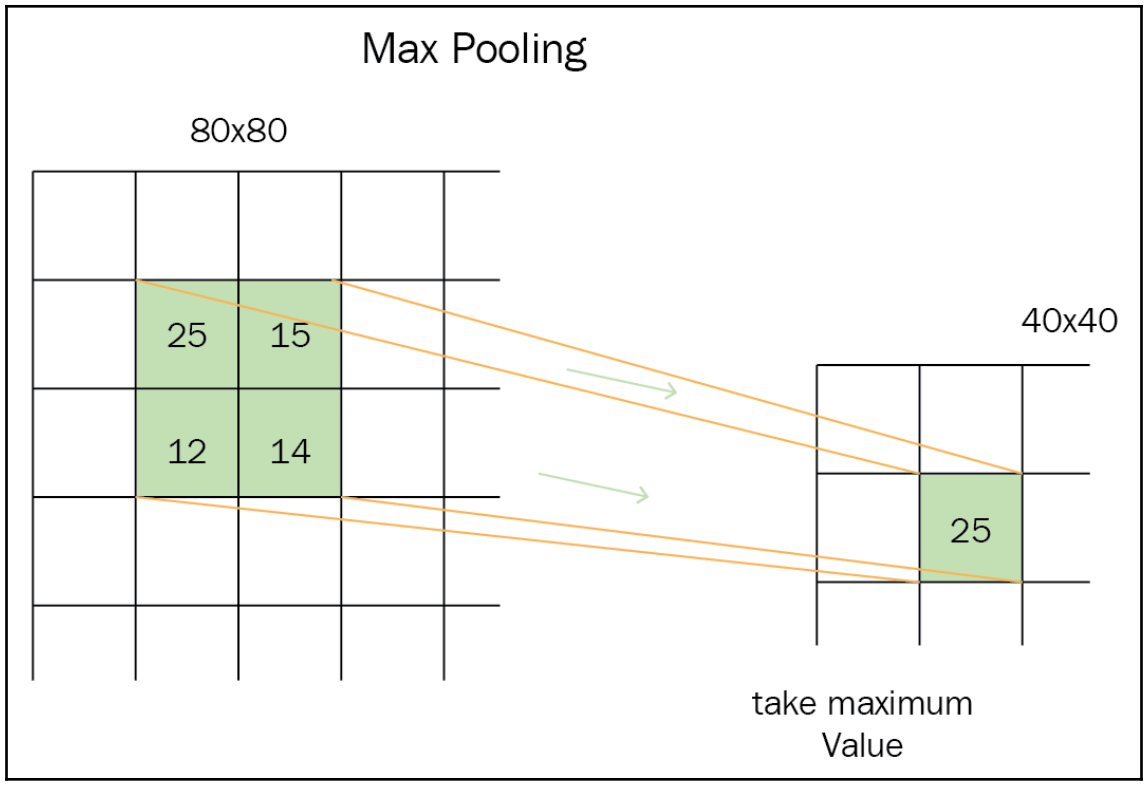
Chapter 4: Object Recognition Using Neural Networks and Supervised Learning

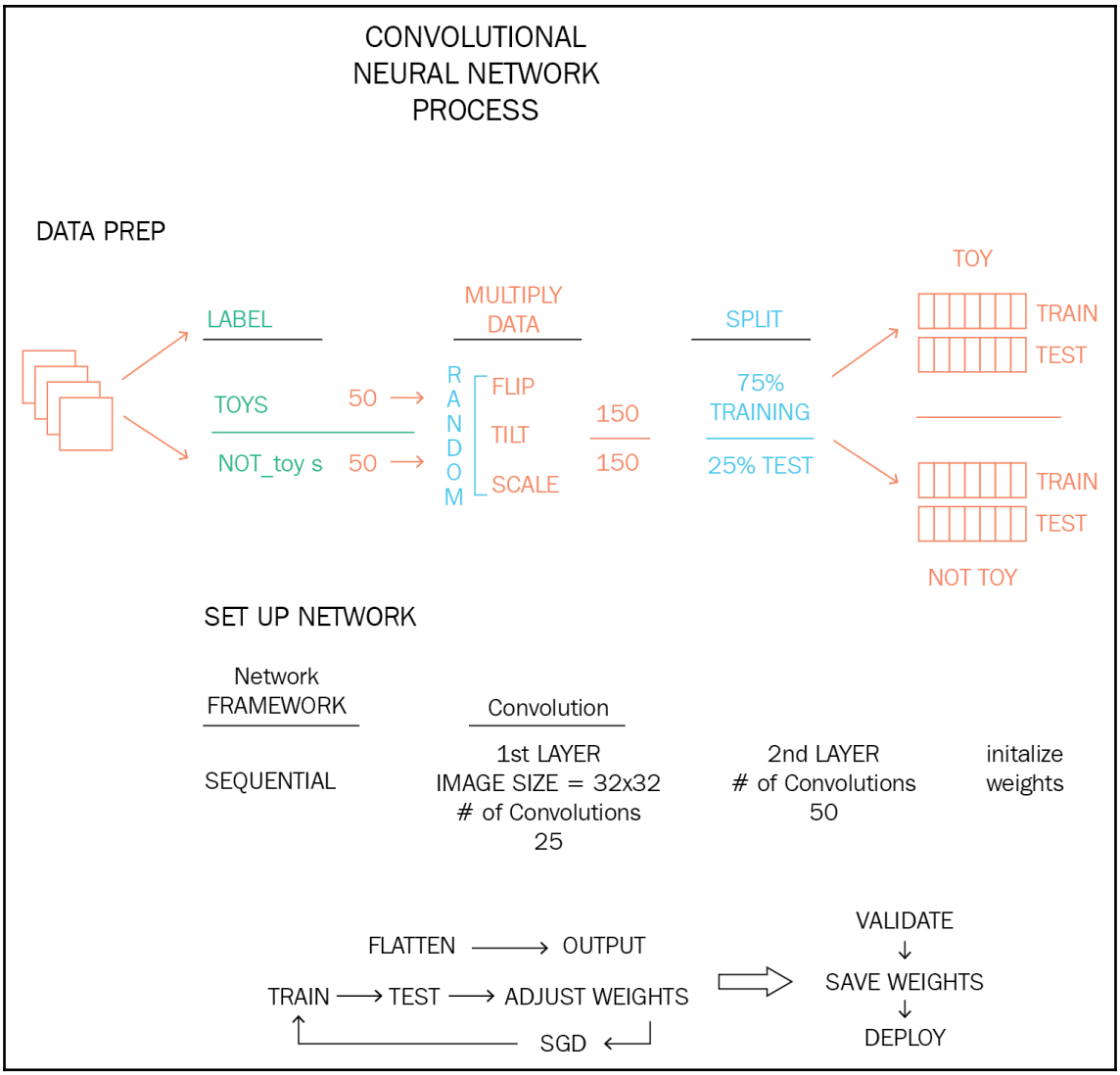


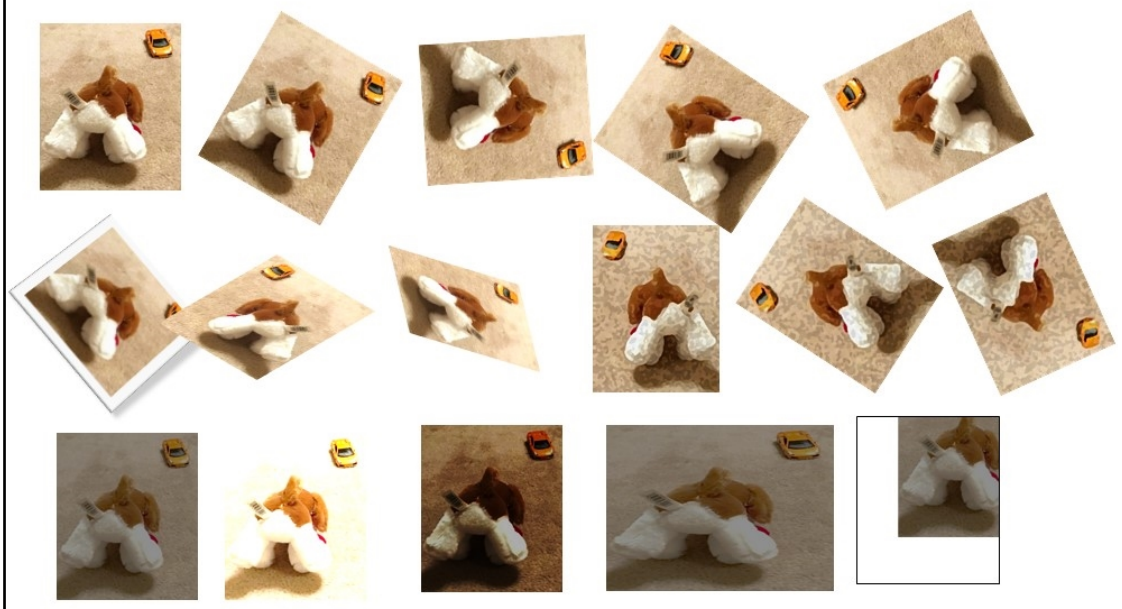

Result of Sobel Edge Detection Convolution


ARTIFICIAL NEURON
Input
Input
Input
Input
w
w
w
w
B
I
A
S
ACTIVATION
ON
OFF
Output
Value = Sum of (Input x weight) + BIAS
Activation function determines
if there is an output
Generally if value > Ø
train by adjusting weights

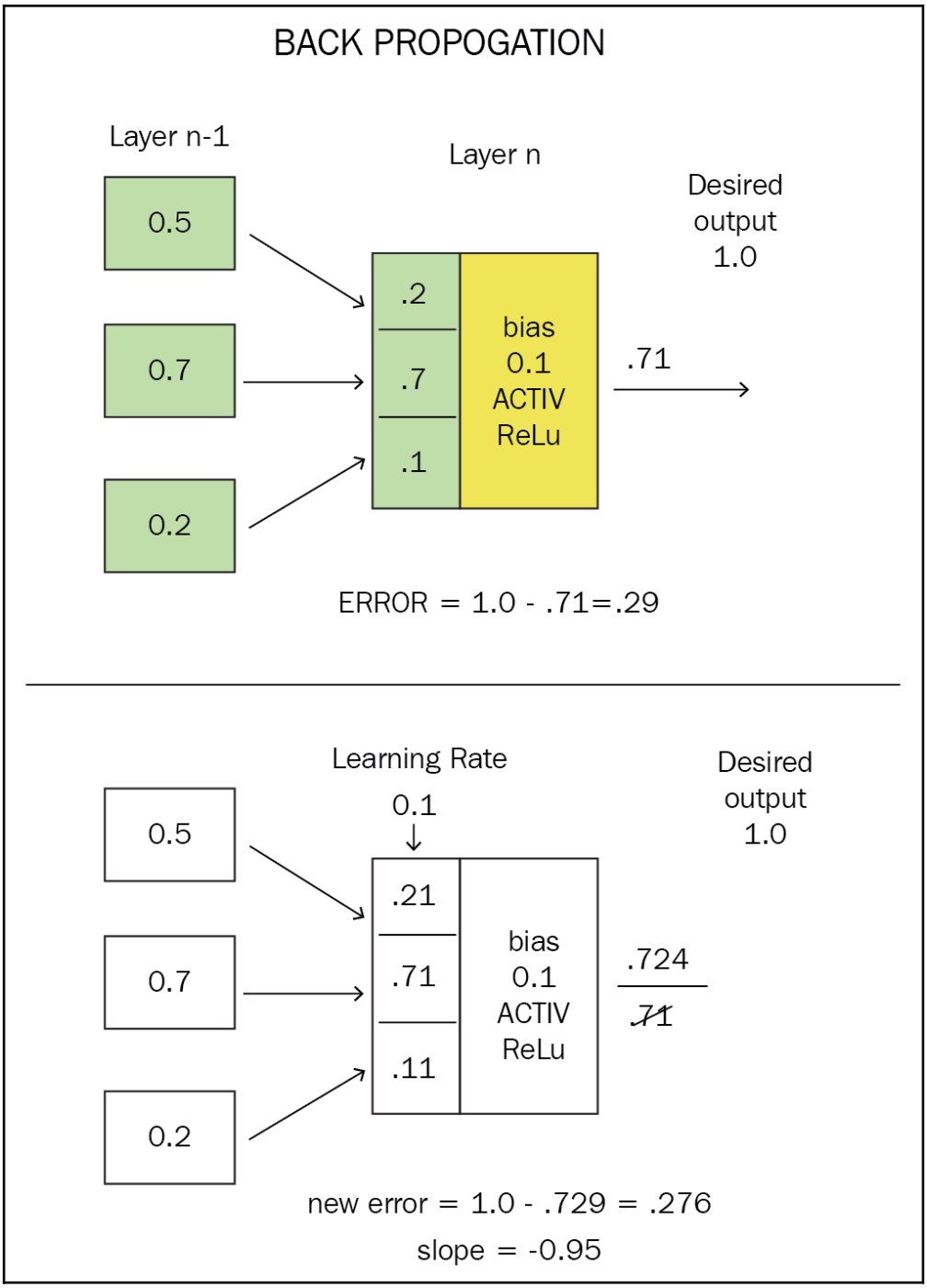

COMMON ACTIVATION FUNCTIONS
BINARY
output
I
0
input
O
THRESHOLD
ReLu
y
X
O
0
X<0
X>0
SIGMOID
1
0.5
0
x
1
1+e^x
TANh
x
tanh (x)
(scaled sigmoid)
LINEAR
x

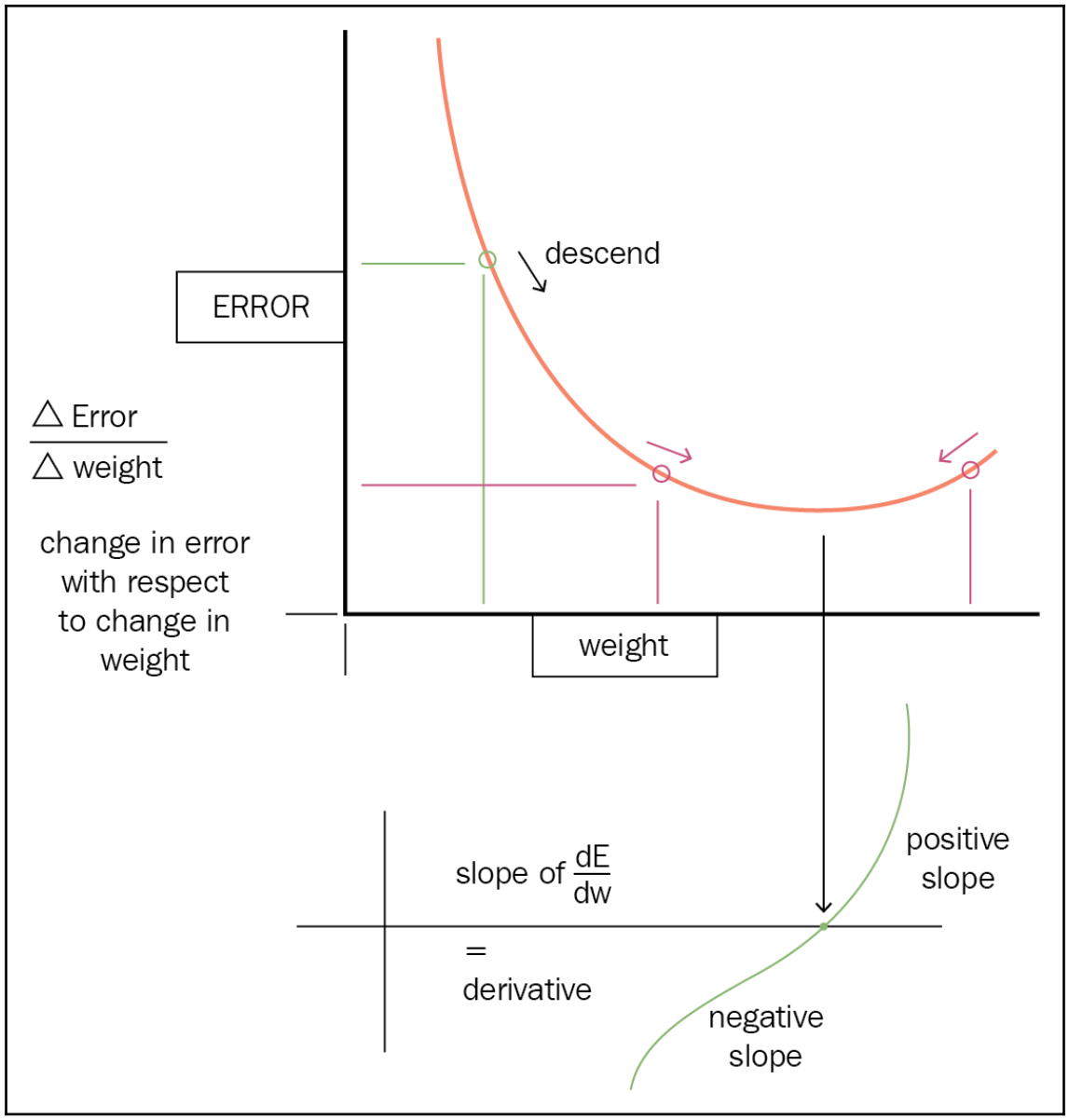

Max Pooling
80x80
25
15
12
14
40x40
25
take maximum
Value

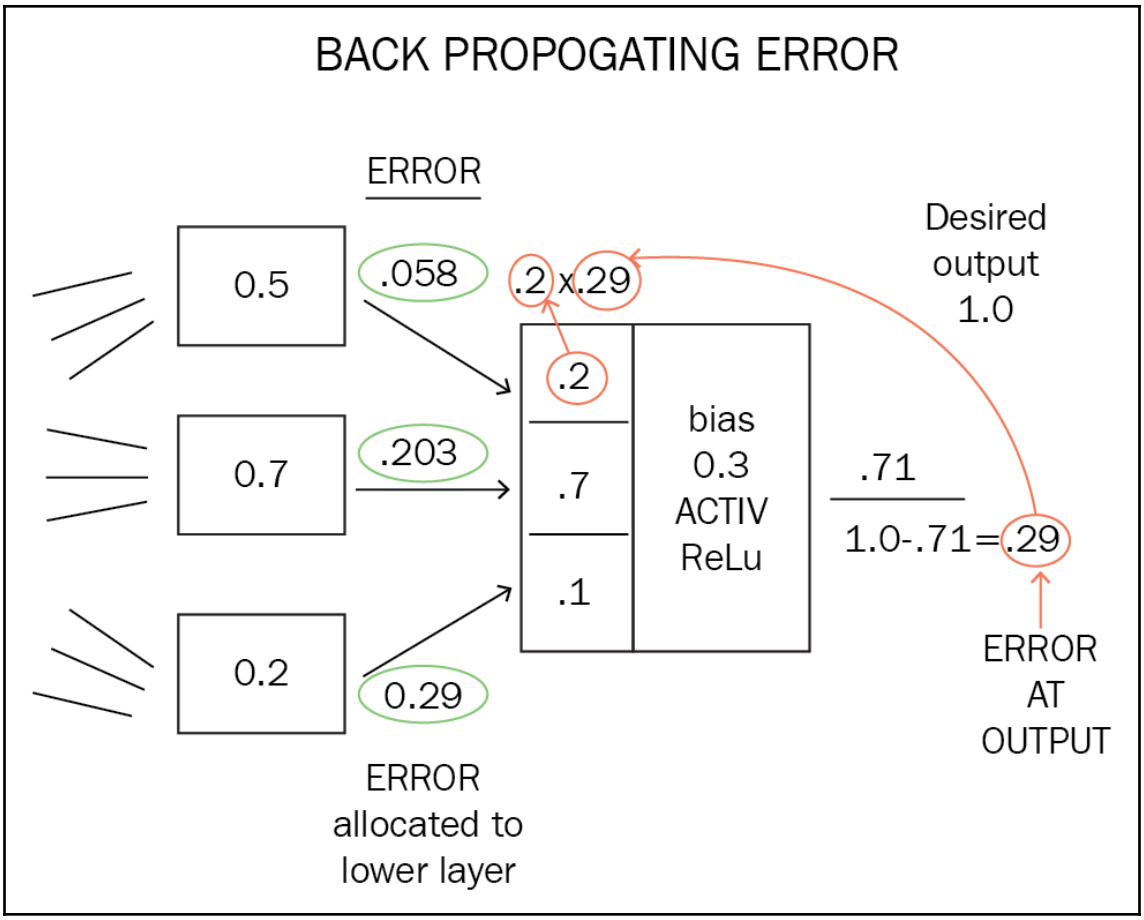

CONVOLUTIONAL
NEURAL NETWORK
PROCESS
DATA PREP
LABEL
TOYS
50
NOT_toy s
50
MULTIPLY
DATA
RANDOM
FLIP
TILT
SCALE
150
150
SPLIT
75%
TRAINING
25% TEST
TOY
TRAIN
TEST
TRAIN
TEST
NOT TOY
SET UP NETWORK
Network
FRAMEWORK
SEQUENTIAL
Convolution
1st LAYER
IMAGE SIZE = 32x32
# of Convolutions
25
2nd LAYER
# of Convolutions
50
initalize
weights
FLATTEN
OUTPUT
TRAIN
TEST
ADJUST WEIGHTS
SGD
VALIDATE
SAVE WEIGHTS
DEPLOY

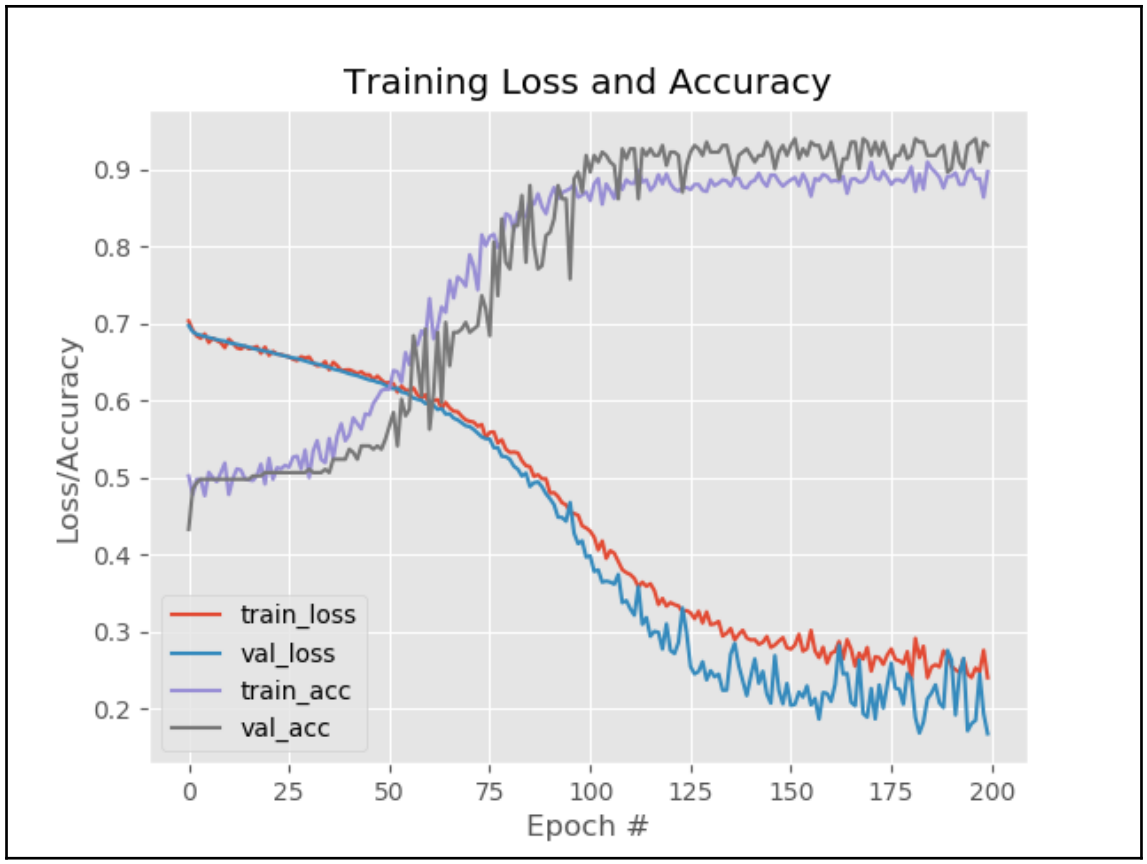


Data Augmentation works by random rotations, flips, mirrors, noise, shears, and lighting


BACK PROPOGATION
Layer n-1
Layer n
Desired output 1.0
0.5
0.7
0.2
.2
.7
.1
bias 0.1 ACTIV ReLu
.71
ERROR = 1.0 - .71=.29
Learning Rate
0.1
Desired output 1.0
0.5
0.7
0.2
.21
.71
.11
bias 0.1 ACTIV ReLu
.724
.71
new error = 1.0 - .729 = .276
slope = -0.95

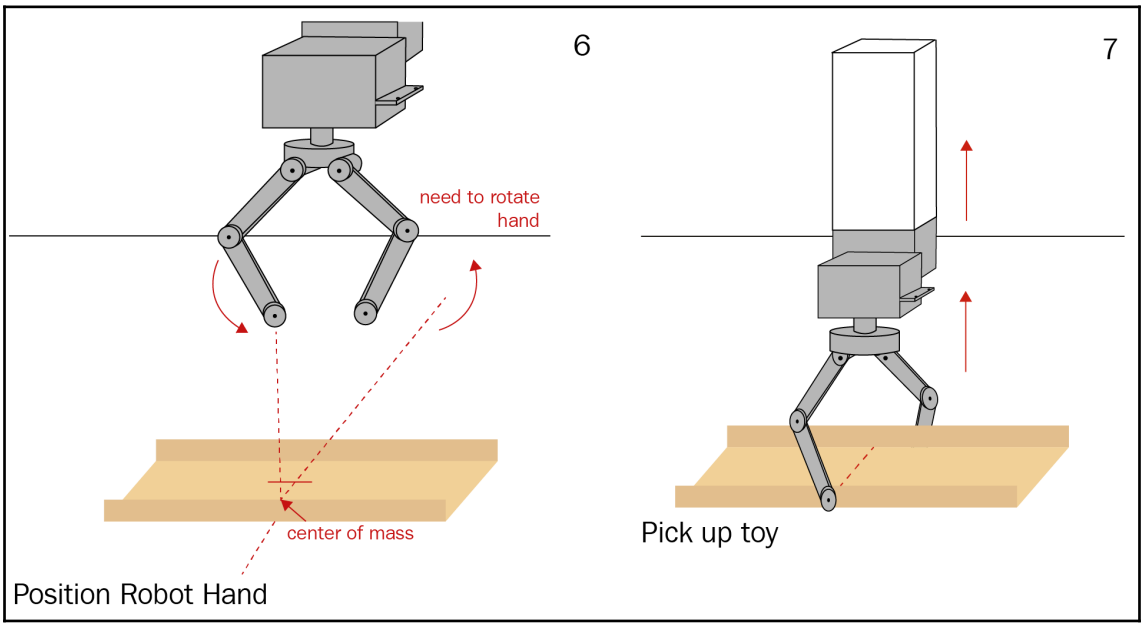

ERROR
descend
△ Error
△ weight
change in error
with respect
to change in
weight
weight
slope of dE/dw
=
derivative
positive
slope
negative
slope

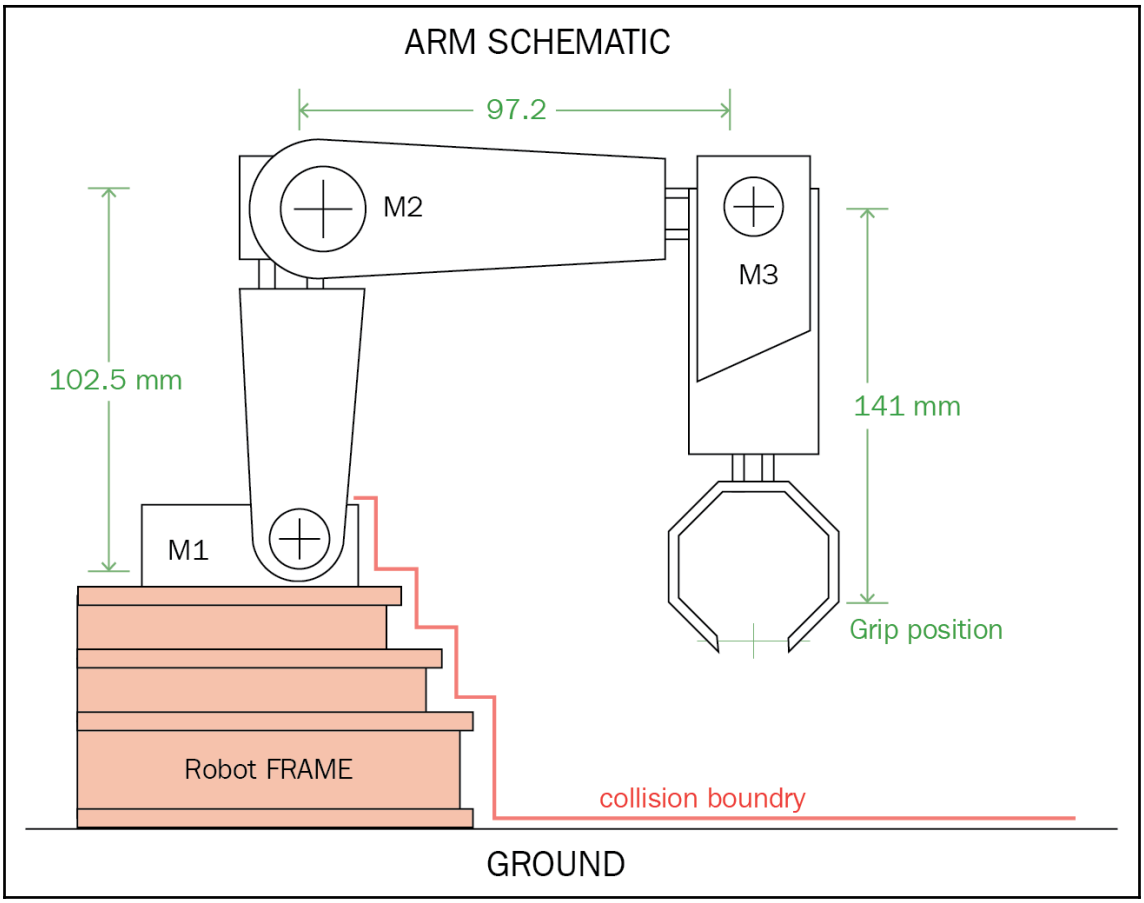

BACK PROPOGATING ERROR
ERROR
0.5
.058
.2 x.29
Desired
output
1.0
.2
0.7
.203
.7
bias
0.3
ACTIV
ReLu
.71
1.0-.71=.29
.1
0.2
0.29
ERROR
AT
OUTPUT
ERROR
allocated to
lower layer

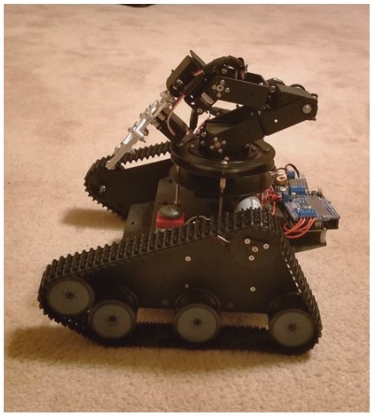

Training Loss and Accuracy
Loss/Accuracy
Epoch #
train_loss
val_loss
train_acc
val_acc
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
25
50
75
100
125
150
175
200


Not toy: 98.54%
toy: 99.83%

Chapter 5: Picking up the Toys



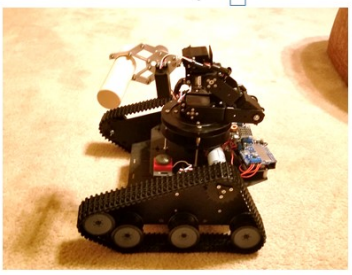

ARM SCHEMATIC
97.2
M2
M3
102.5 mm
141 mm
M1
Grip position
Robot FRAME
collision boundry
GROUND



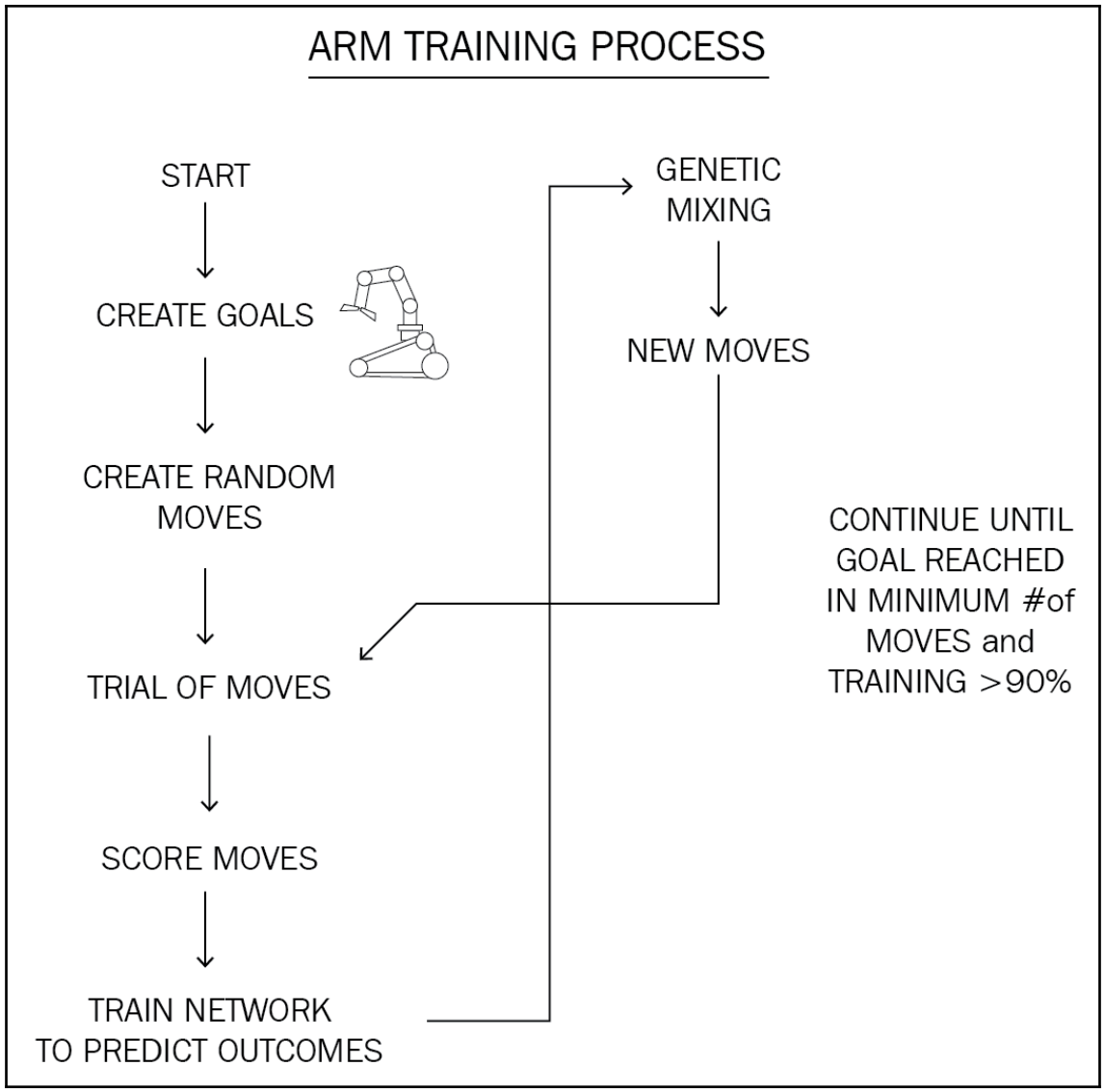
Neutral Carry – normal arm position

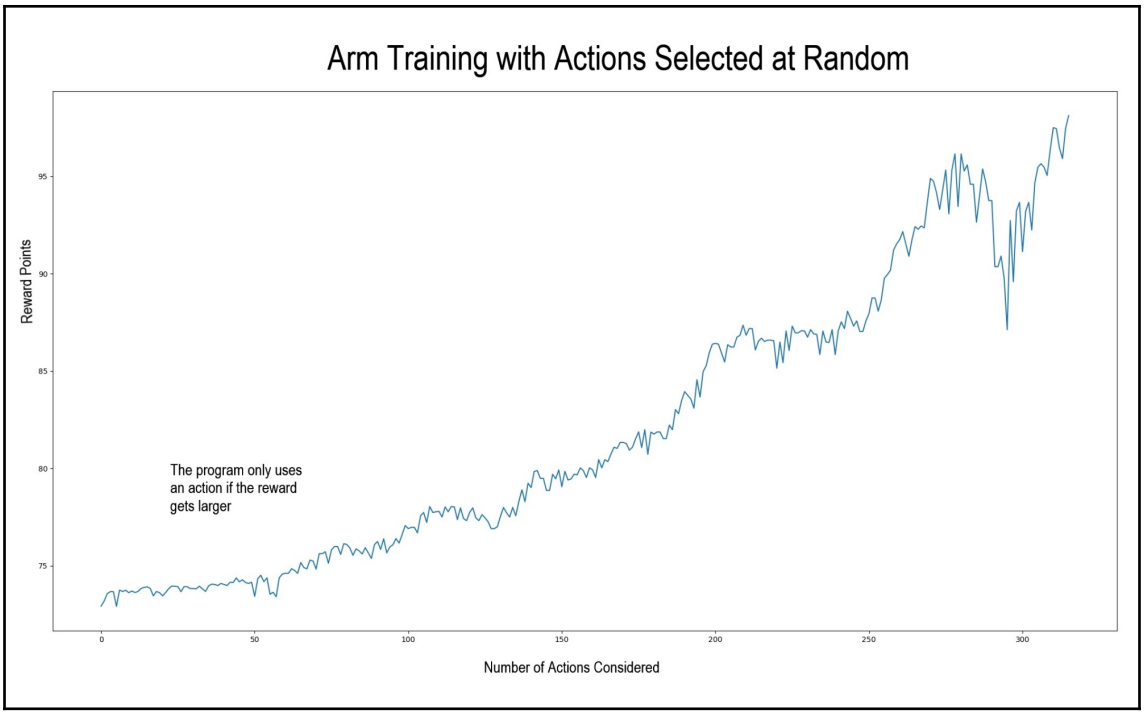
High Carry ⇨

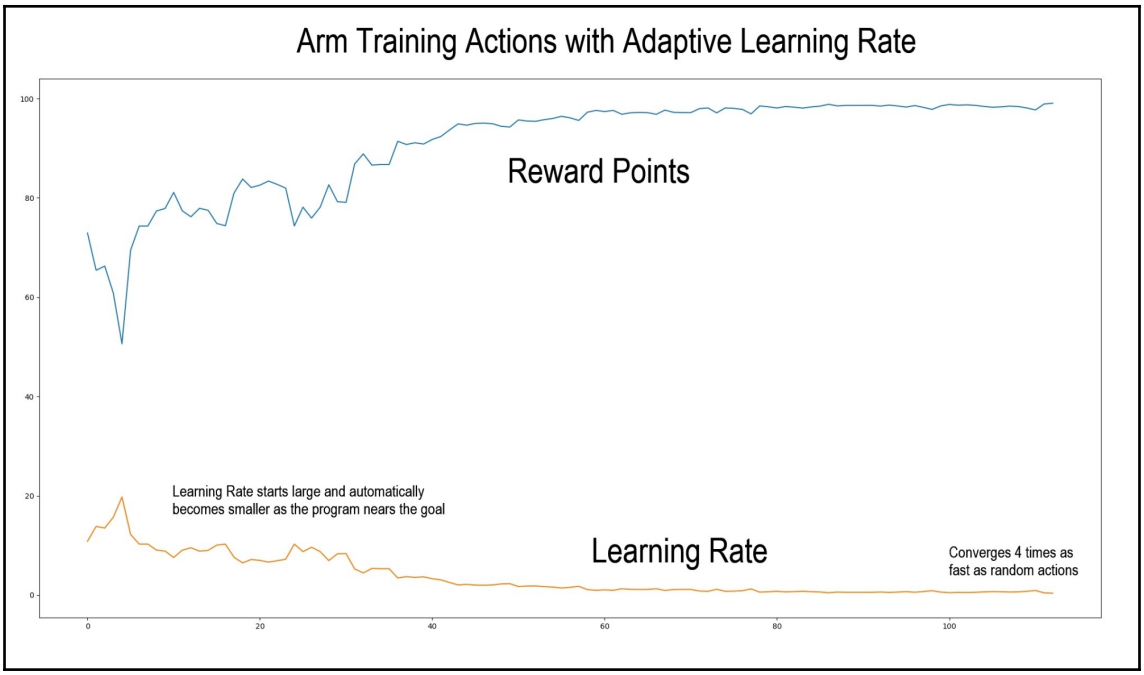
Pick Up ⇧

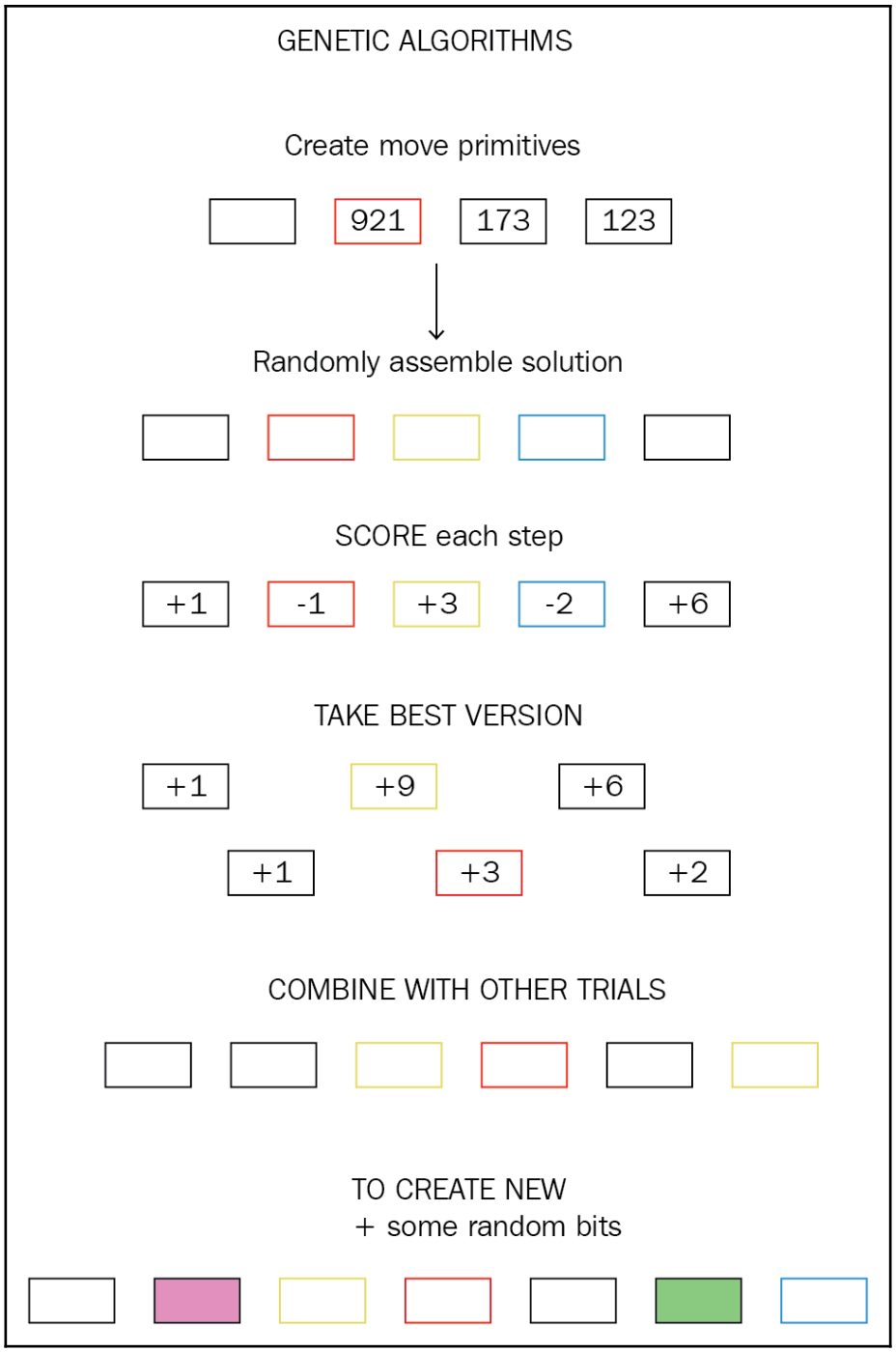


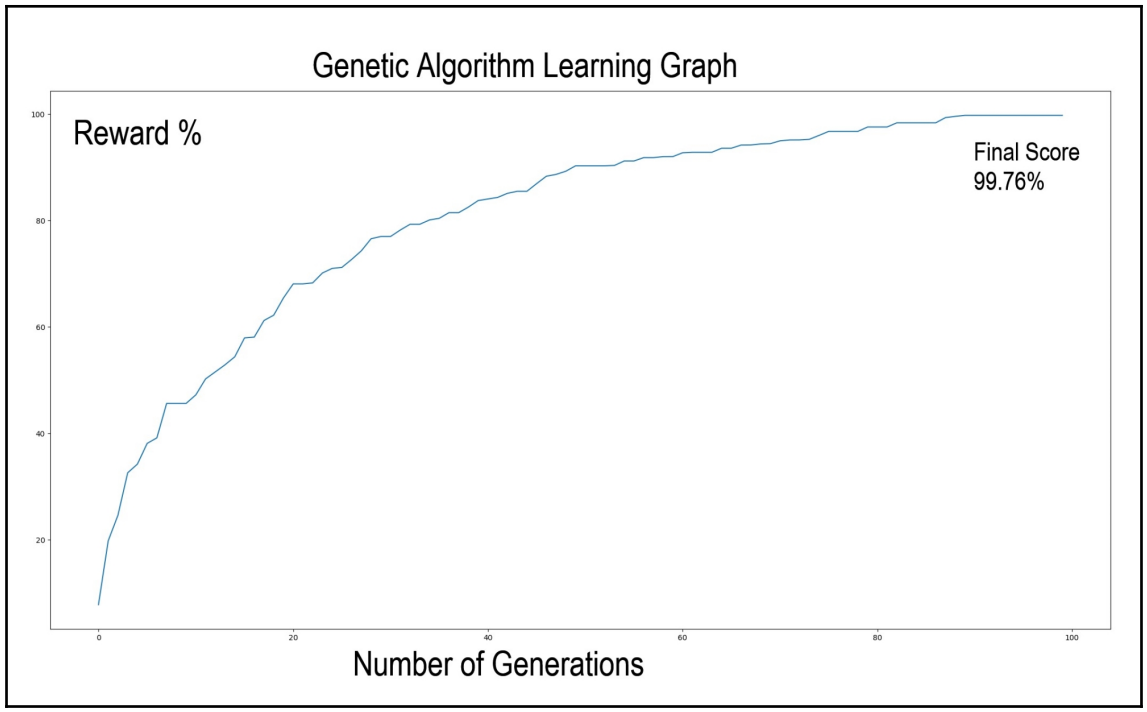
Drop Off
At toy box


ARM TRAINING PROCESS
START
CREATE GOALS
CREATE RANDOM MOVES
TRIAL OF MOVES
SCORE MOVES
TRAIN NETWORK TO PREDICT OUTCOMES
GENETIC MIXING
NEW MOVES
CONTINUE UNTIL GOAL REACHED IN MINIMUM #of MOVES and TRAINING >90%

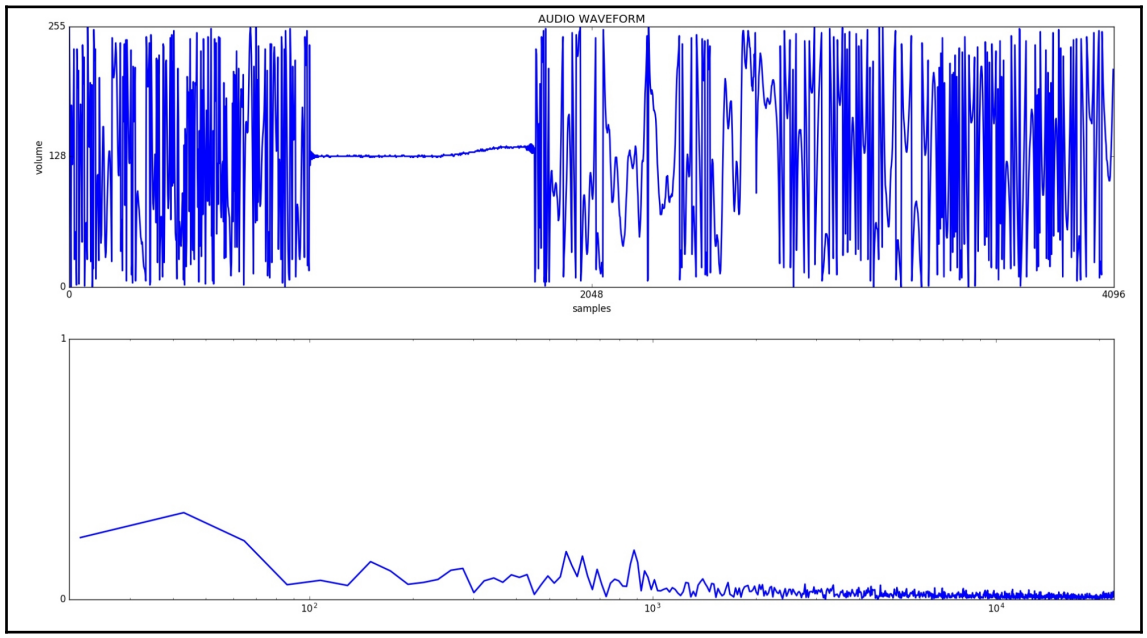

Arm Training with Actions Selected at Random
Reward Points
95
90
85
80
75
The program only uses
an action if the reward
gets larger
0
50
100
150
200
250
300
Number of Actions Considered

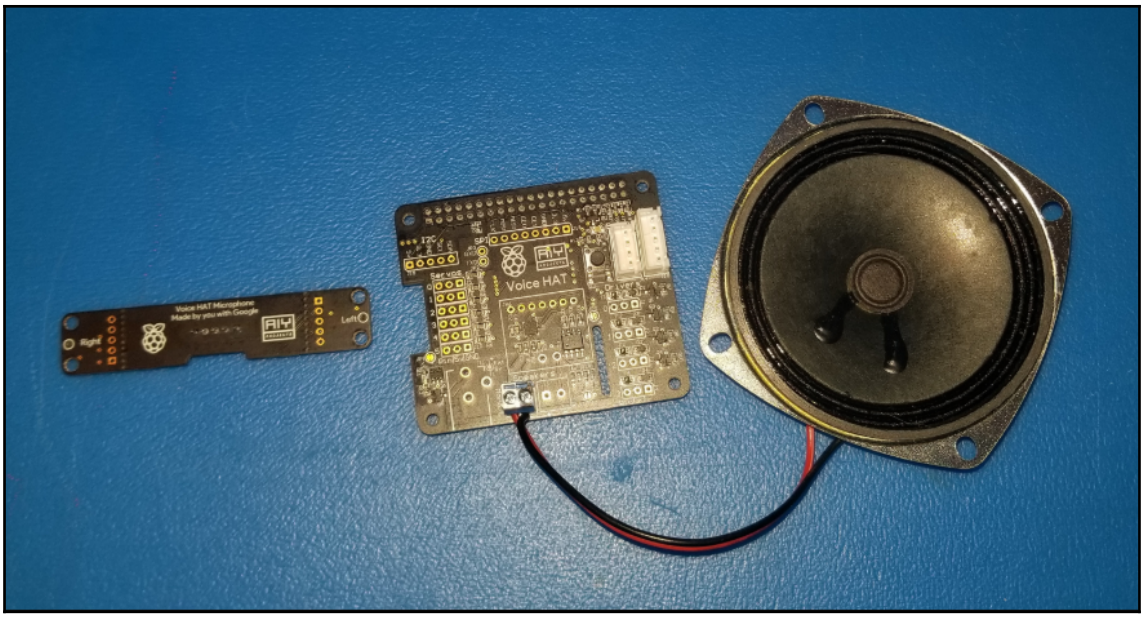

Arm Training Actions with Adaptive Learning Rate
Reward Points
Learning Rate starts large and automatically
becomes smaller as the program nears the goal
Learning Rate
Converges 4 times as
fast as random actions
100
80
60
40
20
0
0
20
40
60
80
100


GENETIC ALGORITHMS
Create move primitives
921
173
123
Randomly assemble solution
SCORE each step
+1
-1
+3
-2
+6
TAKE BEST VERSION
+1
+9
+6
+1
+3
+2
COMBINE WITH OTHER TRIALS
TO CREATE NEW
+ some random bits

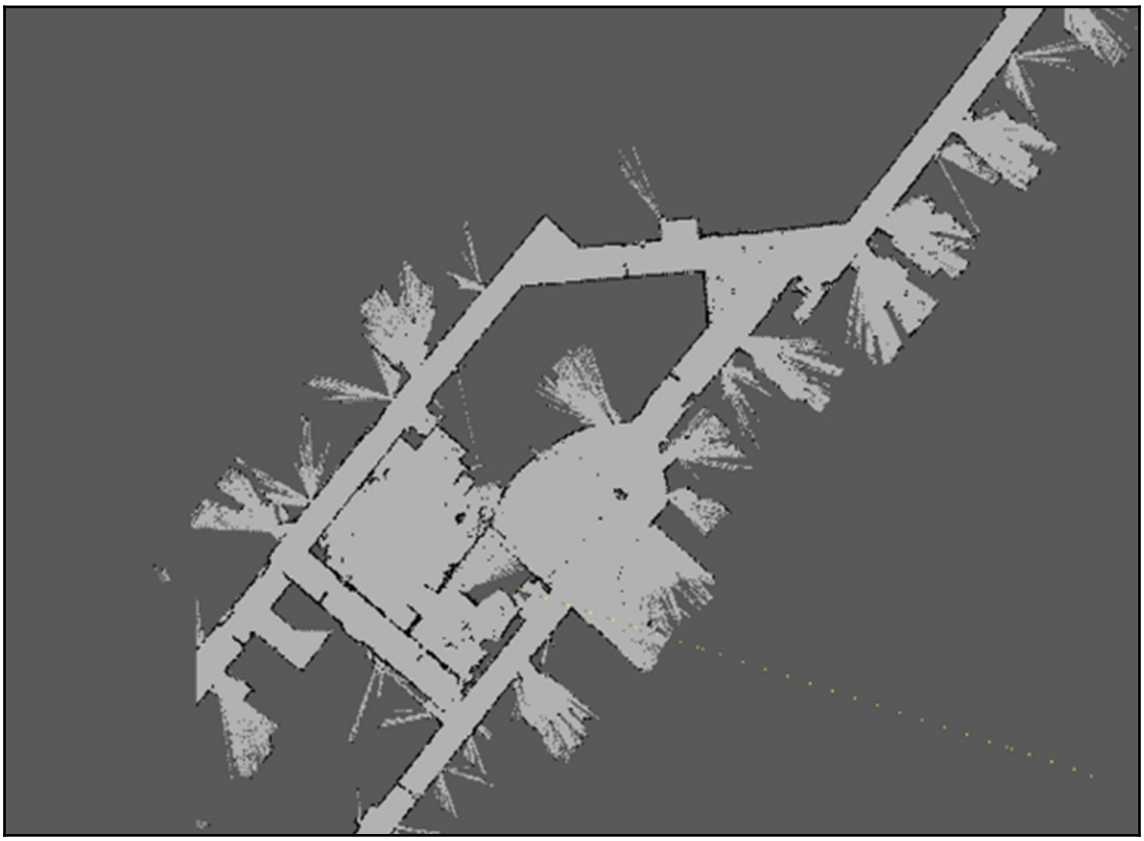

Genetic Algorithm Learning Graph
Reward %
Final Score
99.76%
100
80
60
40
20
0
20
40
60
80
100
Number of Generations

Chapter 6: Teaching a Robot to Listen

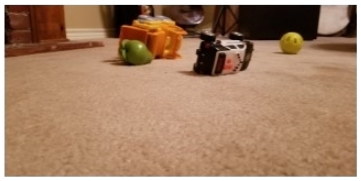



Voice HAT Microphone
Made by you with Google
Right
Left
I2C
SPI
Ser. vos
Voice HAT

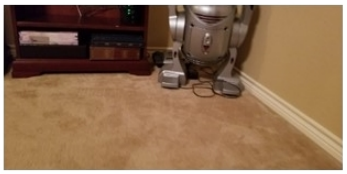
Chapter 7: Avoiding the Stairs



Selection by Different Attributes:



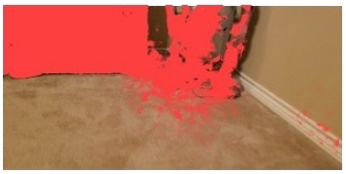
Select by Similar Saturation



Second Image

Select by Similar Hue

Select by Similar Value (luminosity)



Select by Similar Hue

Floor Finder Concepts

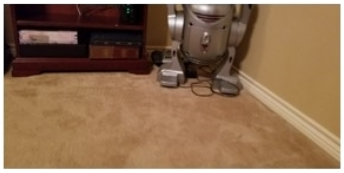
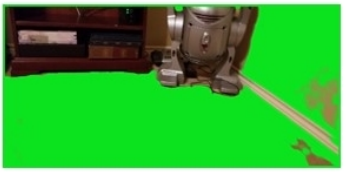


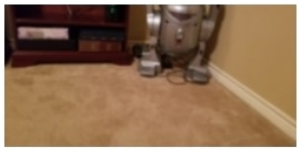
Image from Camera

Select Region that represents the floor (red triangle)



Adding an "erode" function reduces noise

Down sampled to 320x240



Gaussian Blur x 3

Select all the values in the red triangle and find matching values in the image. Turn matching pixels GREEN and non-matching RED

Now we can project lines from the center bottom of the image to the place where they encounter a red pixel. That is our safe driving zone

Floor Finder With Toys

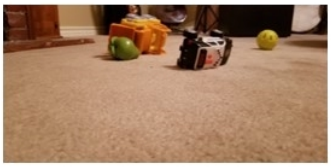
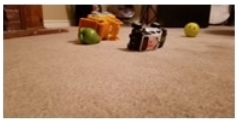
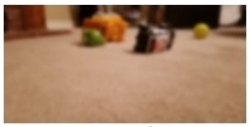


Image from Camera

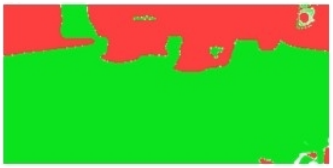


Downsampled to 320x240



Gaussian Blur x 3

Select Region that represents the floor (red triangle)



Select all the values in the red triangle and find matching values in the image. Turn matching pixels GREEN and non-matching RED

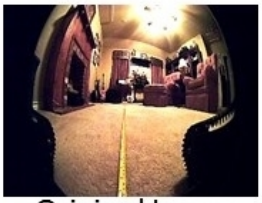
Now we can project lines from the center bottom of the image to the place where they encounter a red pixel. That is our safe driving zone

Since our camera geometry is fixed, we can estimate the distance to obstacles by counting the number of pixels along the projected lines

Only Processing the Top Half of the Image

We have to convert the rectangular image to a square for the neural network. This makes no difference in the results

Image Processing for Neural Network

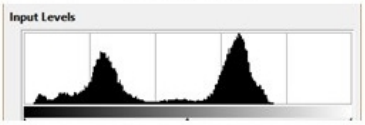


Original Image
800x600

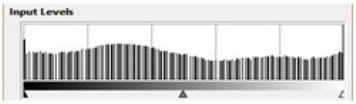
Take only Top Half
320 x 120

Convert to square
matrix
244x244

Grayscale



Values Before Equalize



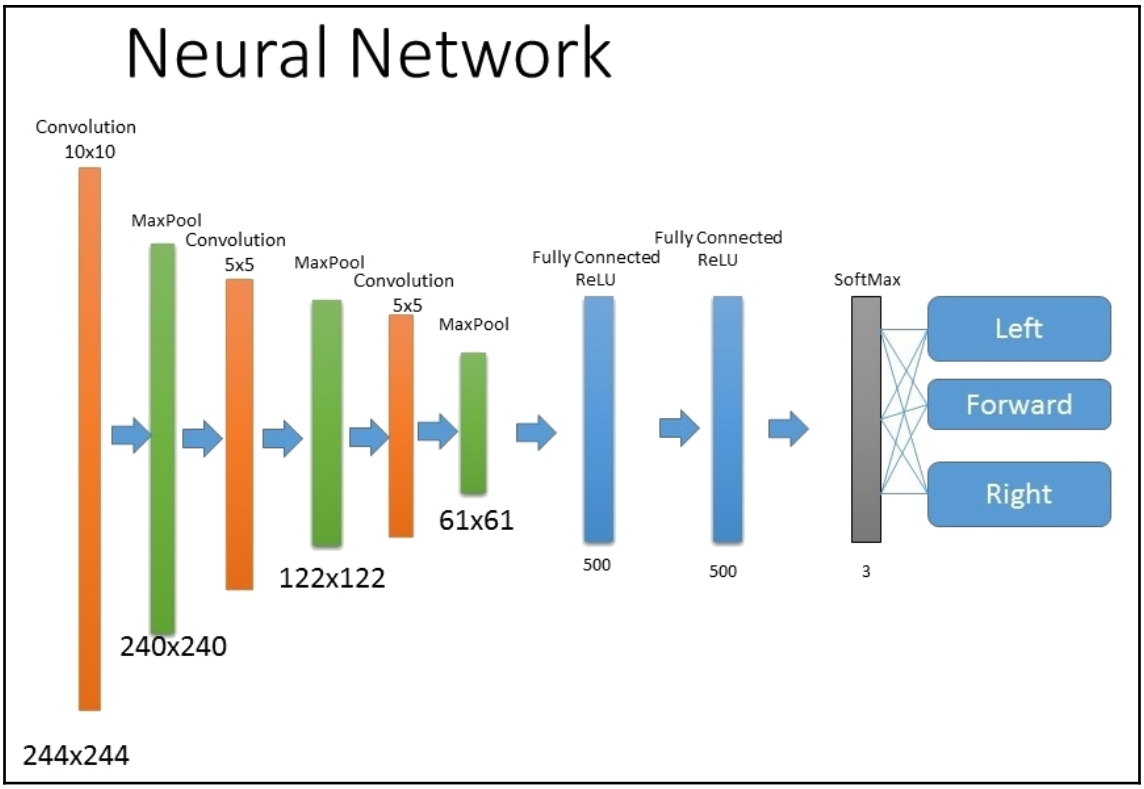
Values after Equalize

Gaussian Blur 5x5

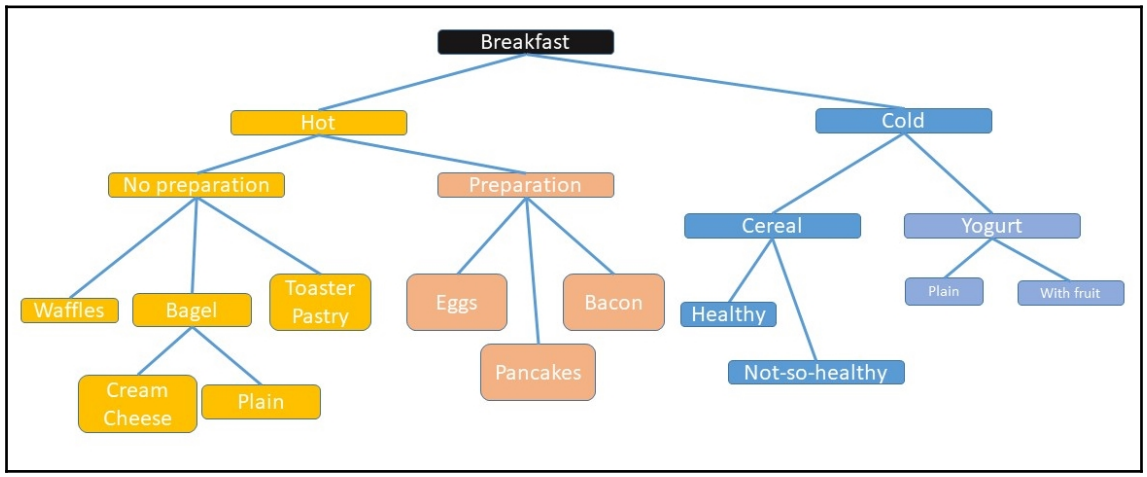
Drive Straight

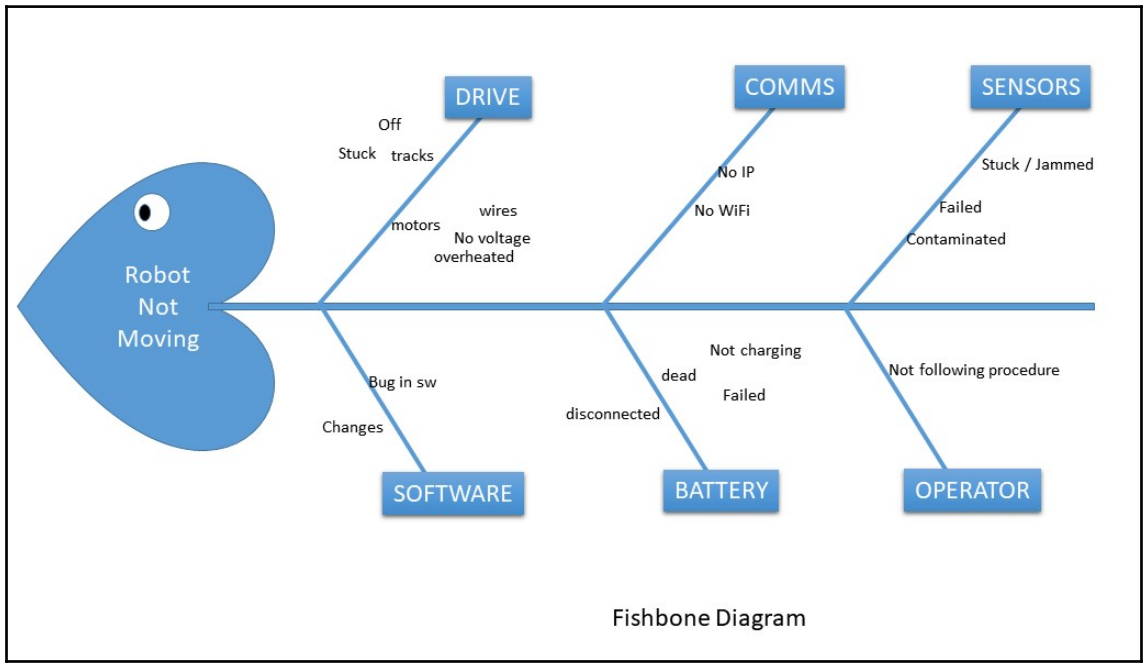
Label the Image

Present to Neural Network

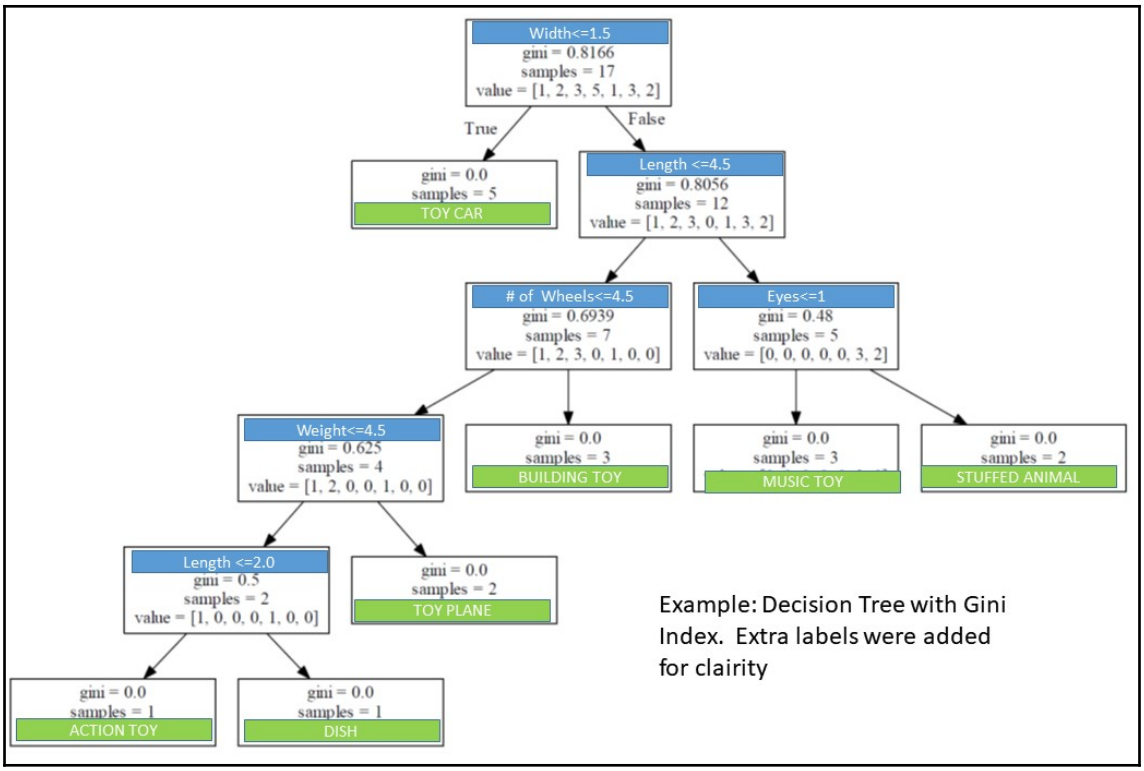

Neural Network
Convolution
10x10
MaxPool
Convolution
5x5
MaxPool
Convolution
5x5
MaxPool
Fully Connected
ReLU
Fully Connected
ReLU
SoftMax
Left
Forward
Right
240x240
122x122
61x61
500
500
3
244x244

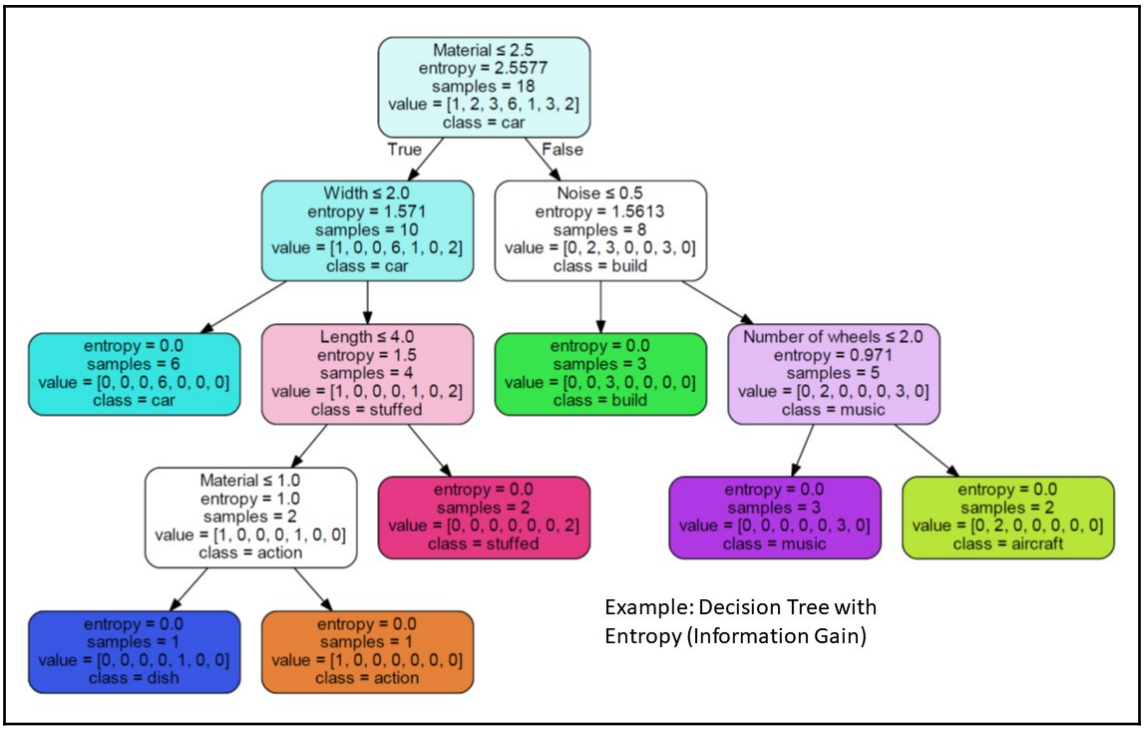
Chapter 8: Putting Things Away



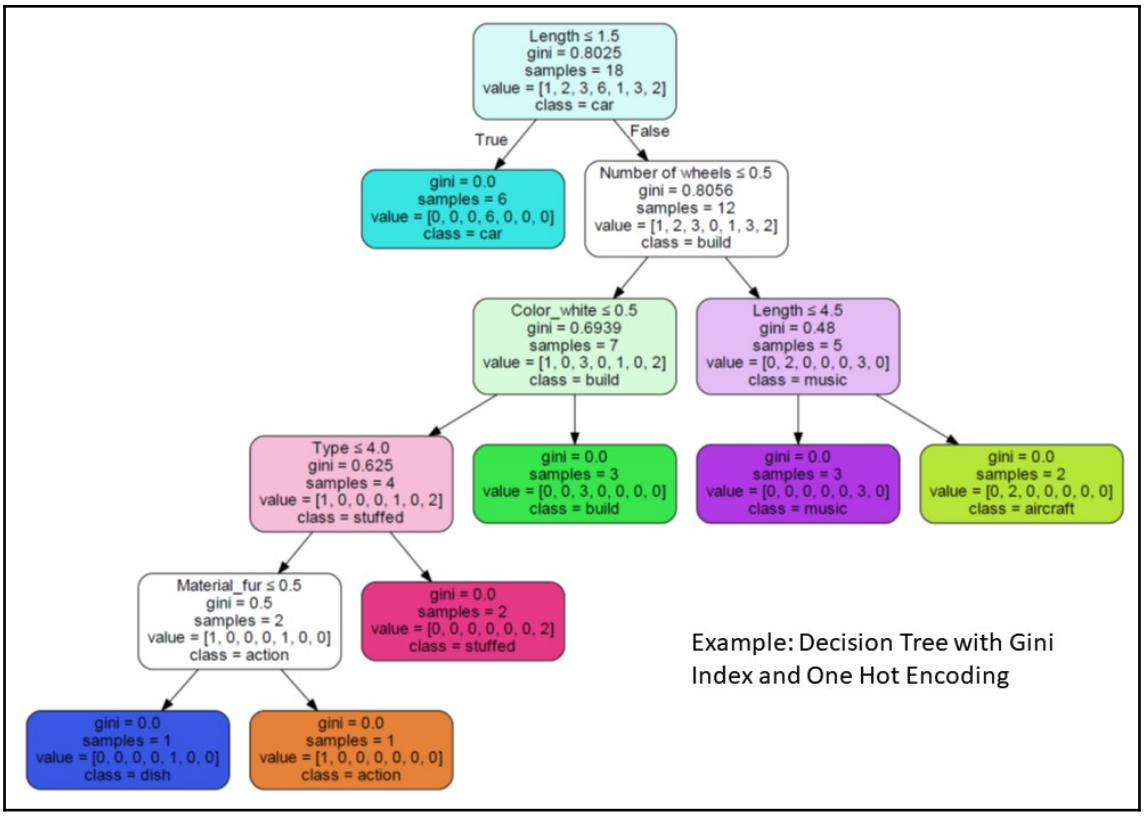


Fishbone Diagram

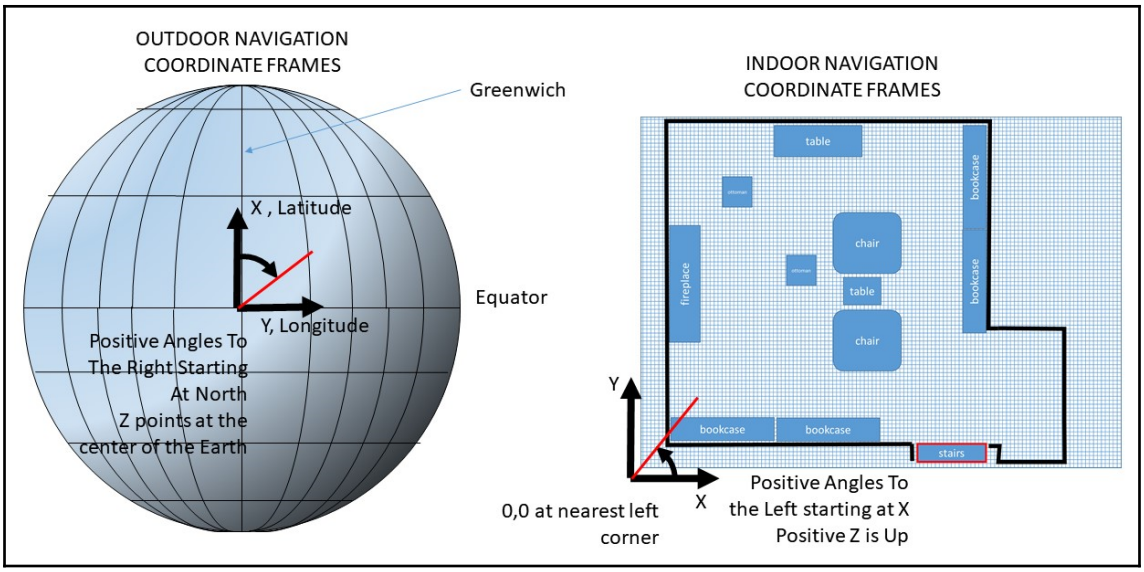

Width<=1.5
gini = 0.8166
samples = 17
value = [1, 2, 3, 5, 1, 3, 2]
True
False
gini = 0.0
samples = 5
TOY CAR
Length <=4.5
gini = 0.8056
samples = 12
value = [1, 2, 3, 0, 1, 3, 2]
# of Wheels<=4.5
gini = 0.6939
samples = 7
value = [1, 2, 3, 0, 1, 0, 0]
Eyes<=1
gini = 0.48
samples = 5
value = [0, 0, 0, 0, 0, 3, 2]
Weight<=4.5
gini = 0.625
samples = 4
value = [1, 2, 0, 0, 1, 0, 0]
gini = 0.0
samples = 3
BUILDING TOY
gini = 0.0
samples = 3
MUSIC TOY
gini = 0.0
samples = 2
STUFFED ANIMAL
Length <=2.0
gini = 0.5
samples = 2
value = [1, 0, 0, 0, 1, 0, 0]
gini = 0.0
samples = 2
TOY PLANE
gini = 0.0
samples = 1
ACTION TOY
gini = 0.0
samples = 1
DISH
Example: Decision Tree with Gini Index. Extra labels were added for clairity



Example: Decision Tree with Entropy (Information Gain)



Example: Decision Tree with Gini Index and One Hot Encoding


OUTDOOR NAVIGATION
COORDINATE FRAMES
Greenwich
X , Latitude
Y, Longitude
Equator
Positive Angles To
The Right Starting
At North
Z points at the
center of the Earth
INDOOR NAVIGATION
COORDINATE FRAMES
table
bookcase
ottoman
chair
fireplace
ottoman
table
bookcase
chair
Y
bookcase
bookcase
stairs
X
0,0 at nearest left
corner
Positive Angles To
the Left starting at X
Positive Z is Up

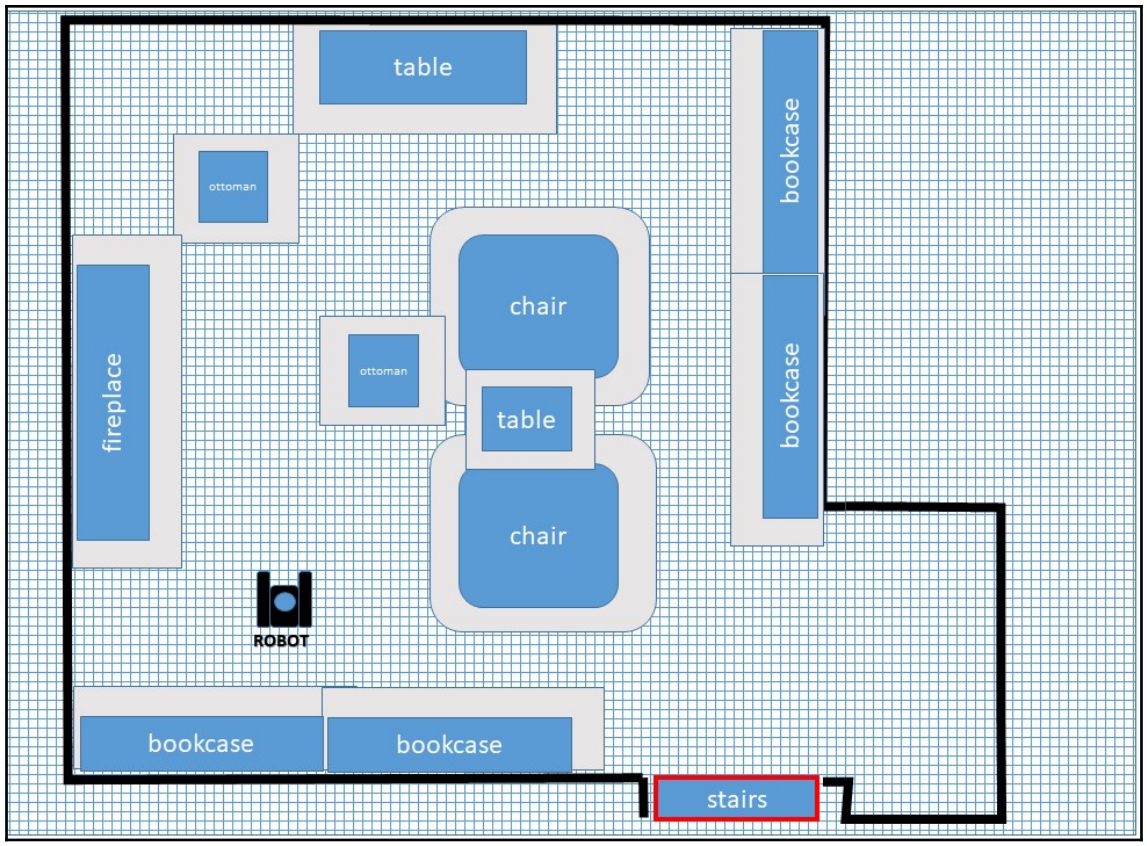
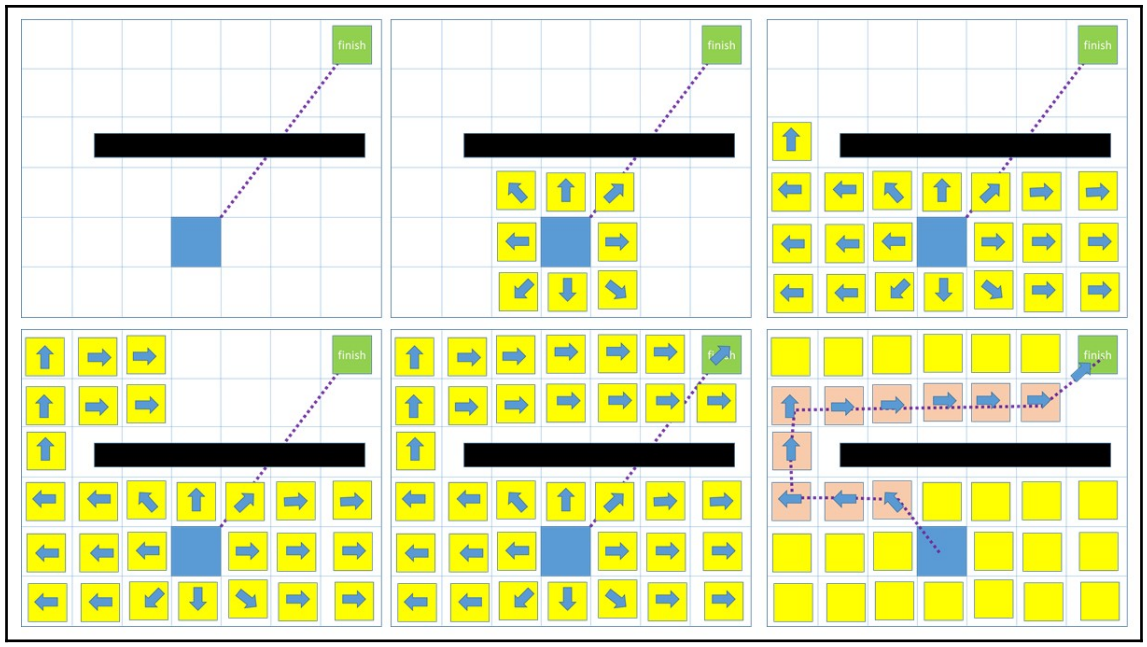
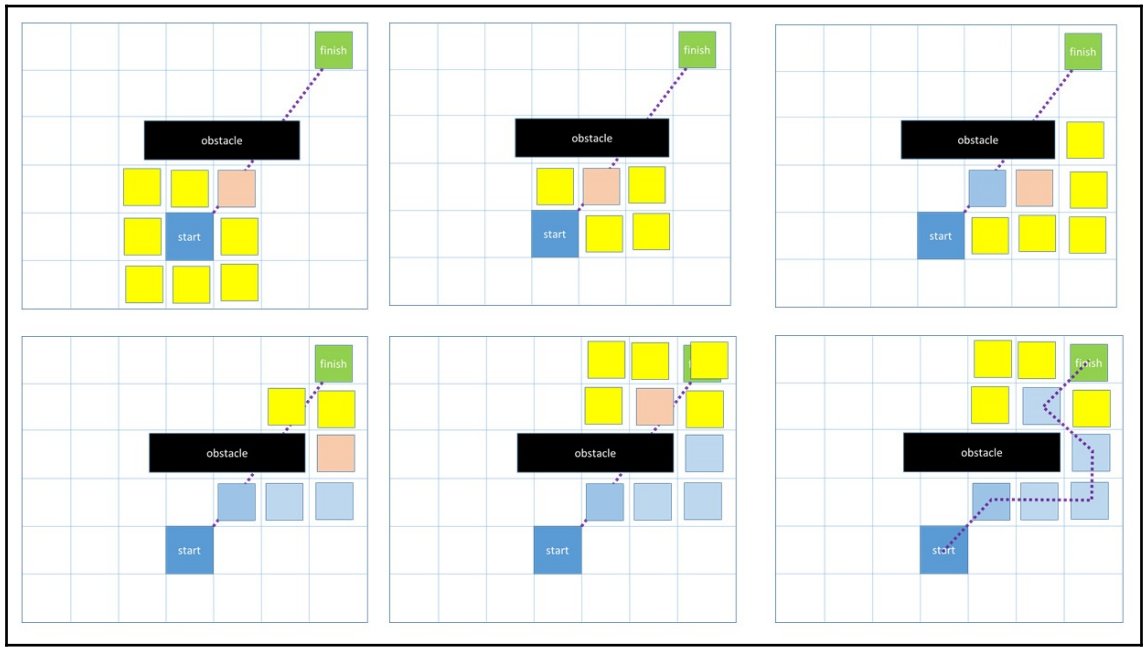
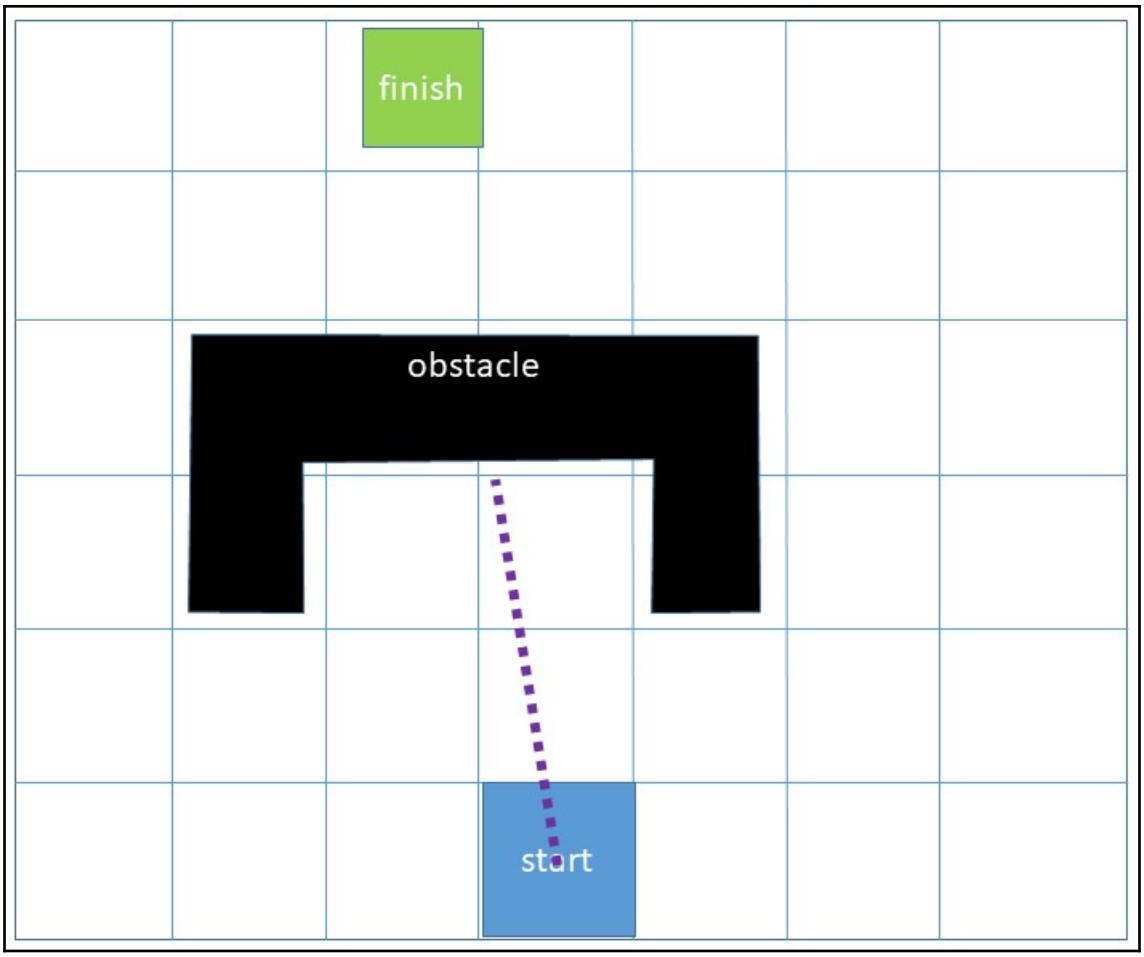
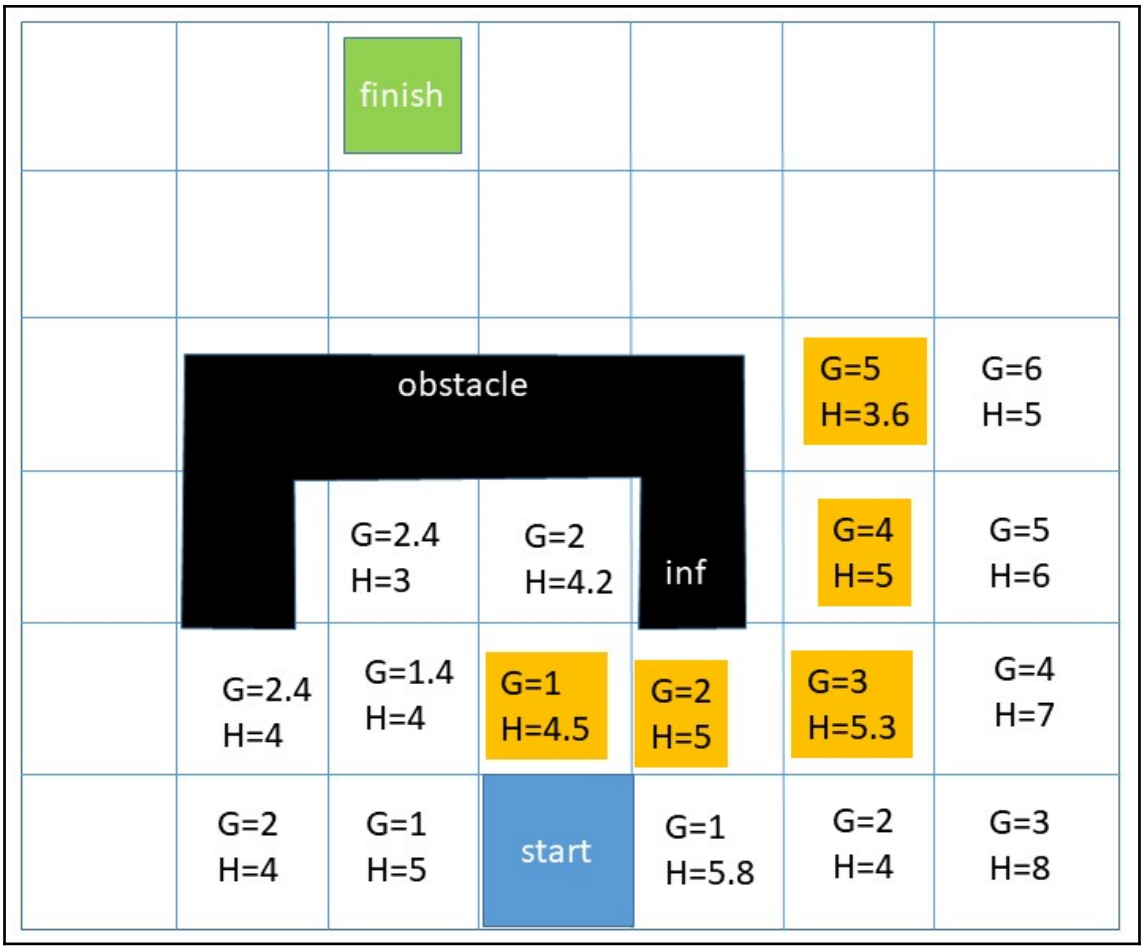


table
ottoman
bookcase
chair
fireplace
ottoman
table
bookcase
chair
ROBOT
bookcase
bookcase
stairs


finish
finish
finish
finish
finish
finish


finish
obstacle
start
finish
obstacle
start
finish
obstacle
start
finish
obstacle
start
finish
obstacle
start
finish
obstacle
start

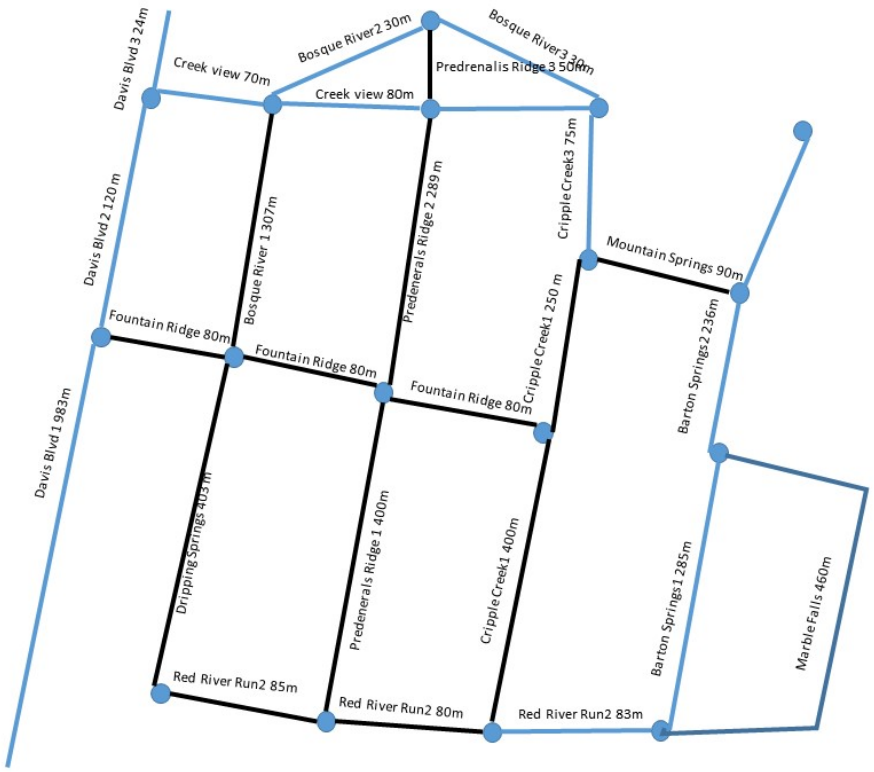

finish
obstacle
start


finish
obstacle
G=5
H=3.6
G=6
H=5
G=2.4
H=3
G=2
H=4.2
inf
G=4
H=5
G=5
H=6
G=2.4
H=4
G=1.4
H=4
G=1
H=4.5
G=2
H=5
G=3
H=5.3
G=4
H=7
G=2
H=4
G=1
H=5
start
G=1
H=5.8
G=2
H=4
G=3
H=8

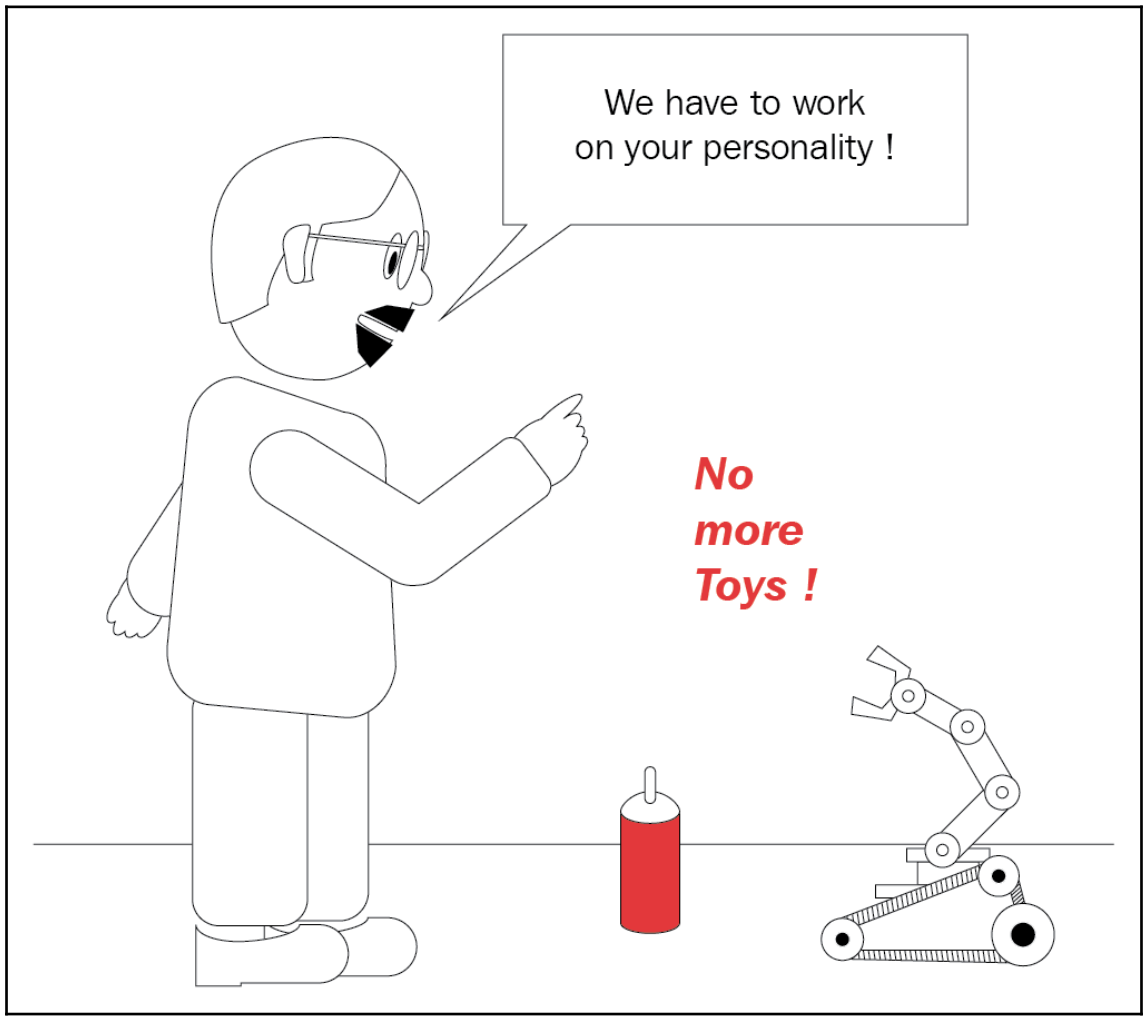
Road Network Database has a list of nodes (circles) and links (lines).

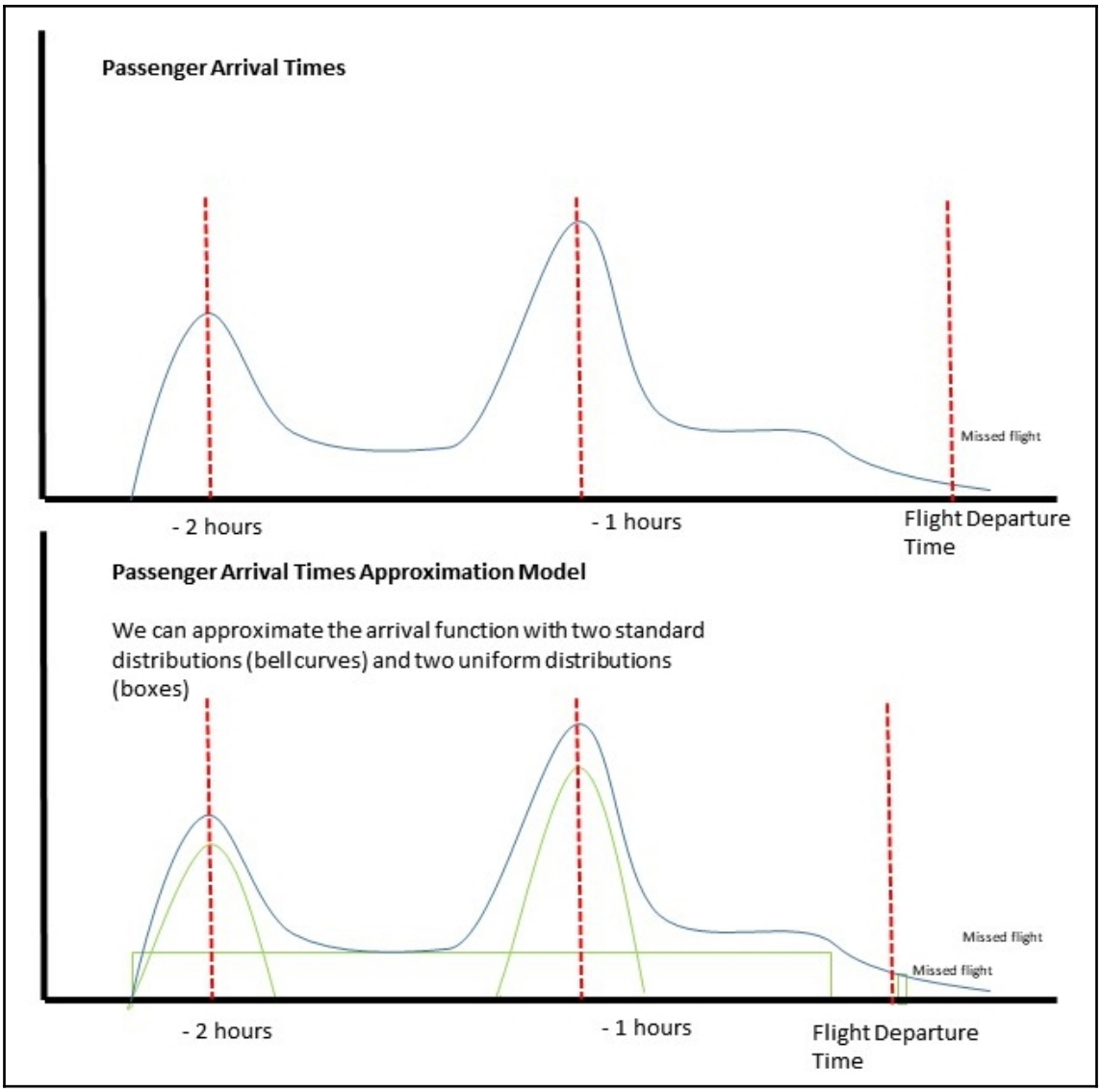
Each node has a list of which links intersect at the node

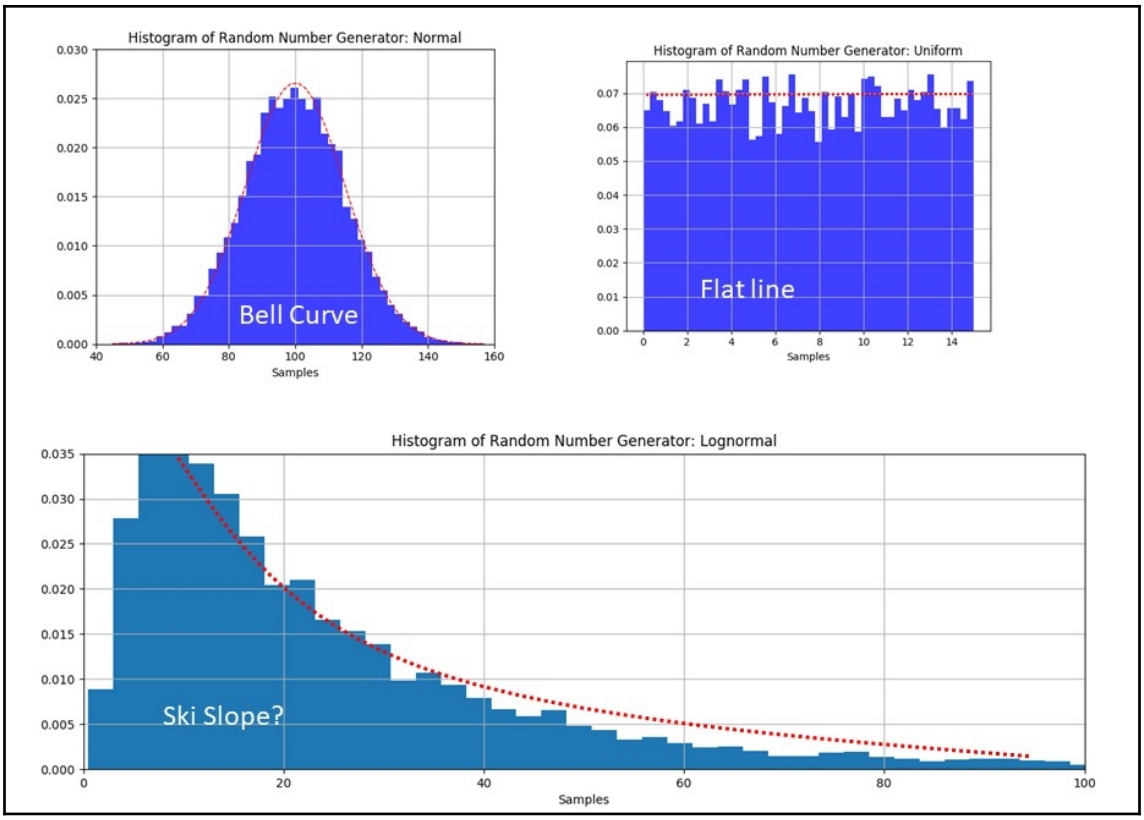
Each link has a length and a cost (type of road), as well as a pointer to the node at each end

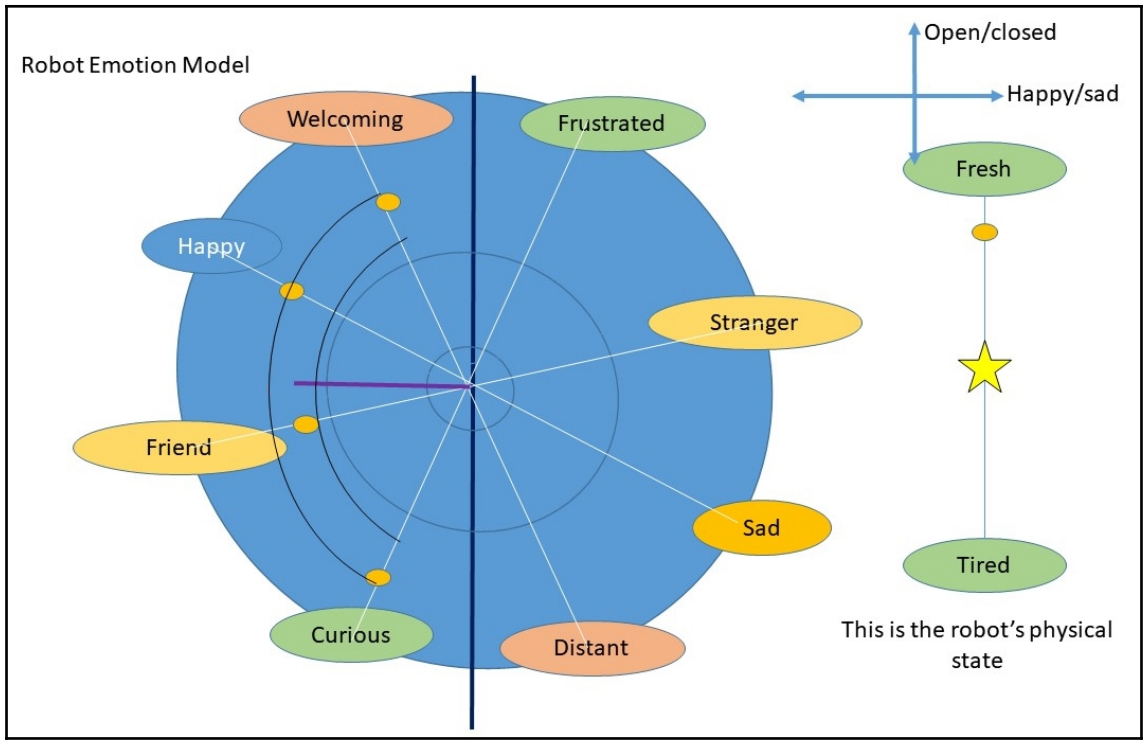


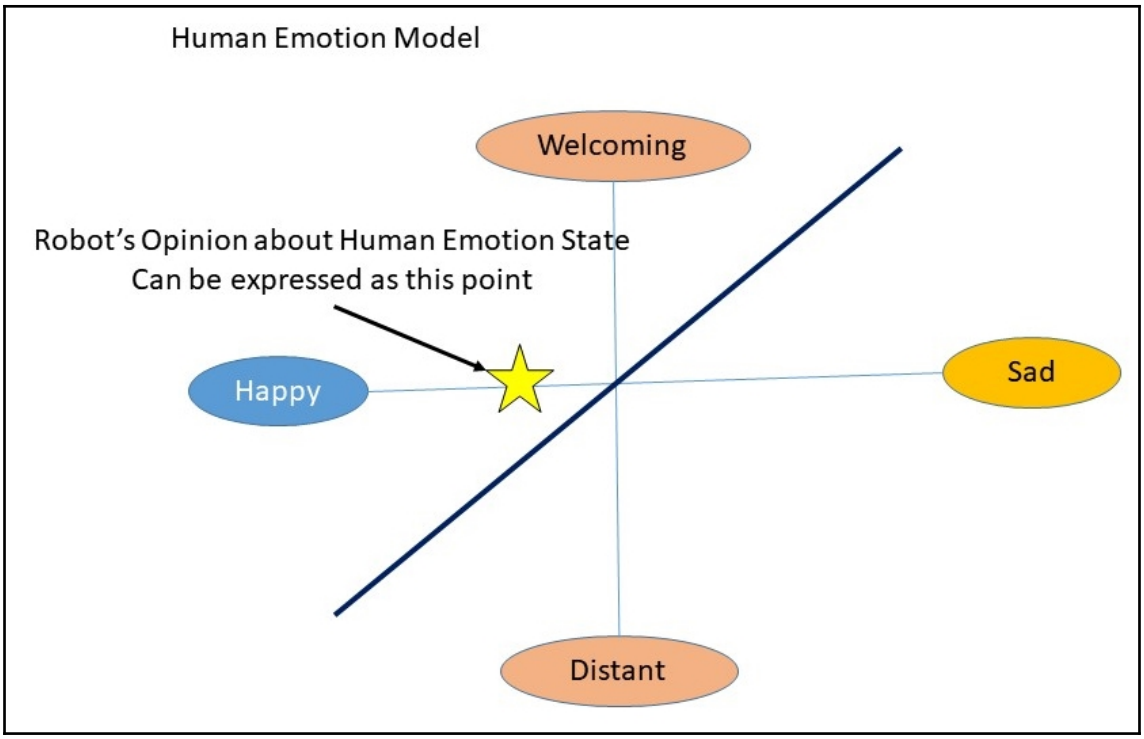
Chapter 9: Giving the Robot an Artificial Personality



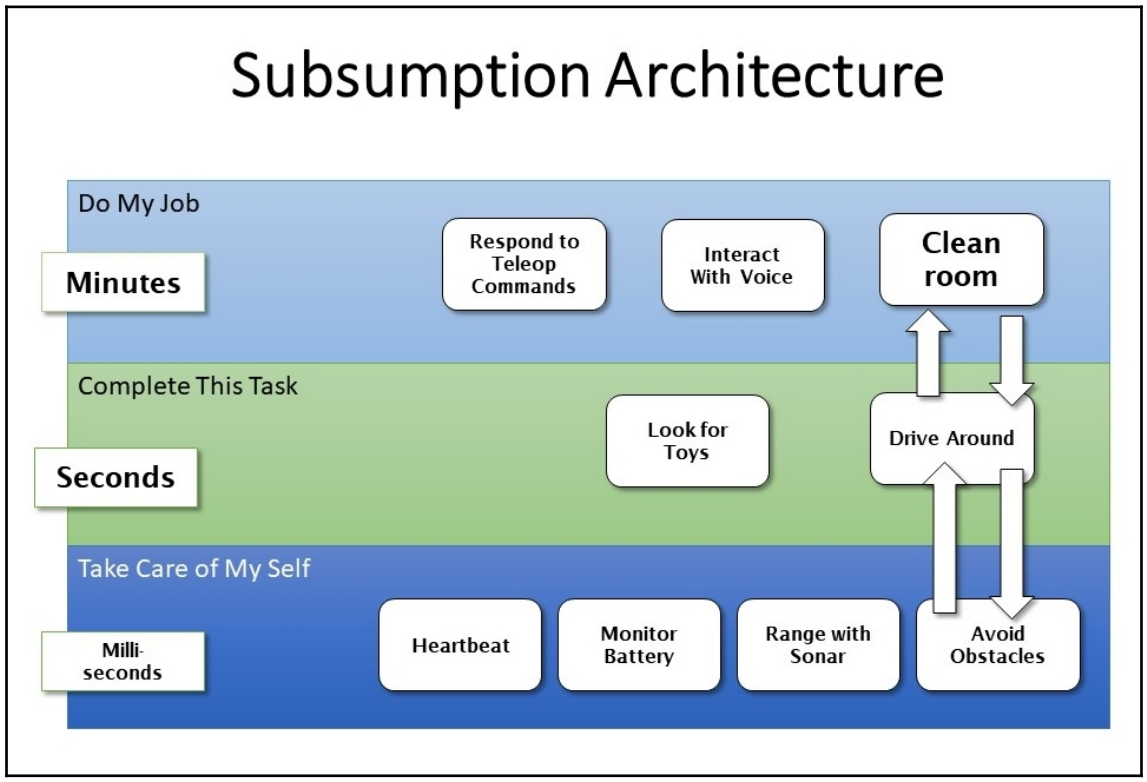

Passenger Arrival Times
Missed flight
- 2 hours
- 1 hours
Flight Departure Time
Passenger Arrival Times Approximation Model
We can approximate the arrival function with two standard distributions (bell curves) and two uniform distributions (boxes)
Missed flight
Missed flight
- 2 hours
- 1 hours
Flight Departure Time

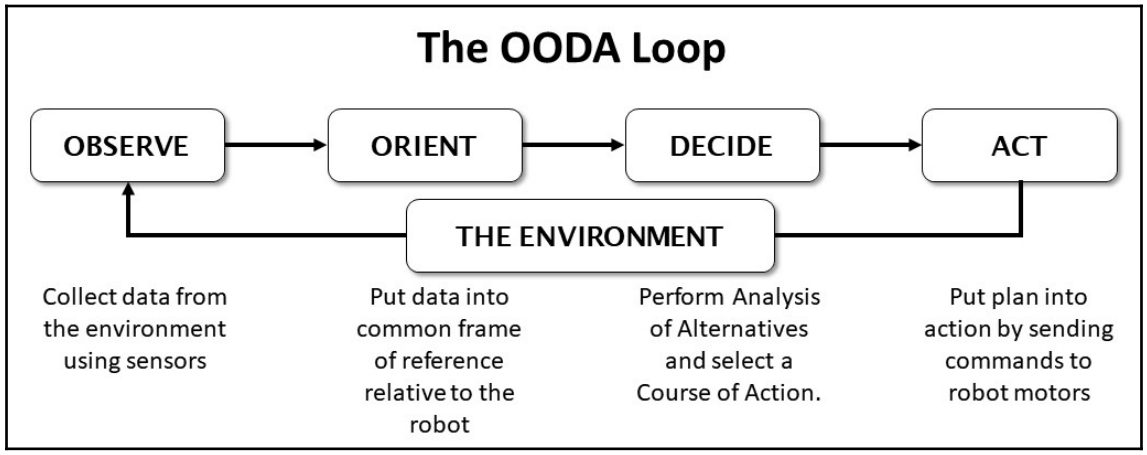

Histogram of Random Number Generator: Normal
Bell Curve
Samples
Histogram of Random Number Generator: Uniform
Flat line
Samples
Histogram of Random Number Generator: Lognormal
Ski Slope?
Samples

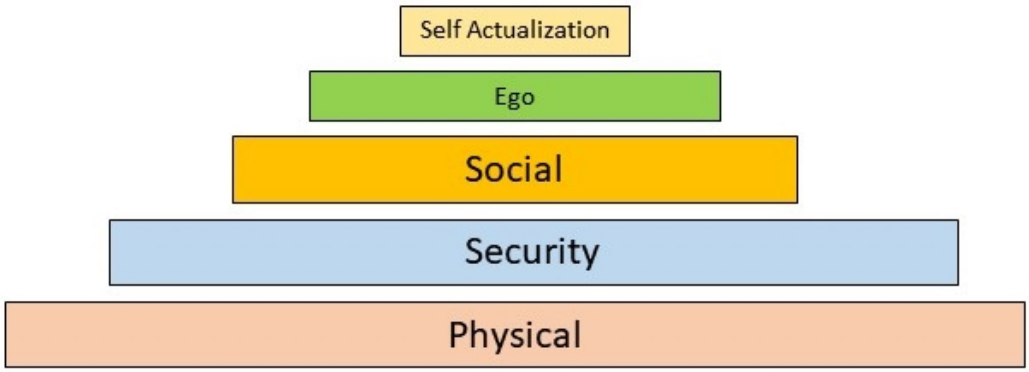

Robot Emotion Model
Welcoming
Frustrated
Happy
Stranger
Friend
Sad
Curious
Distant
Open/closed
Happy/sad
Fresh
Tired
This is the robot's physical state


Human Emotion Model
Welcoming
Robot's Opinion about Human Emotion State
Can be expressed as this point
Happy
Sad
Distant

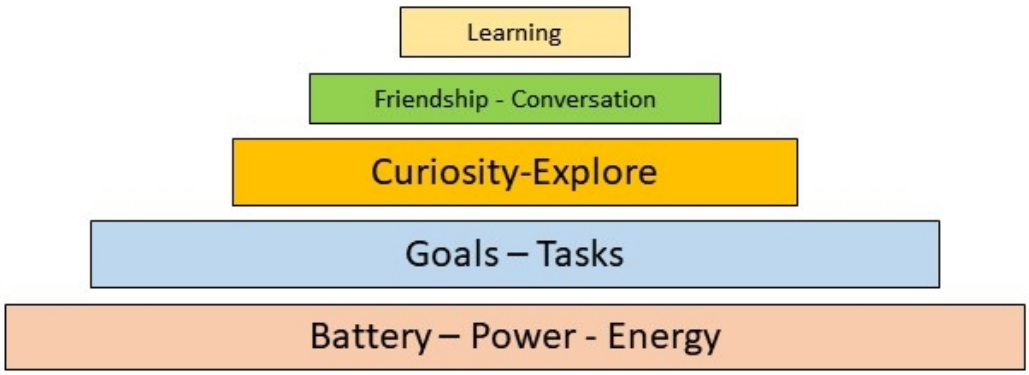
Chapter 10: Conclusions and Reflections







Human Needs (Source: Mazlow)



Theoretical Robot Needs?