Deep Learning Robot Demo - ROS and Robotic Software

Makespace, Cambridge UK

22nd February 2016

About Me

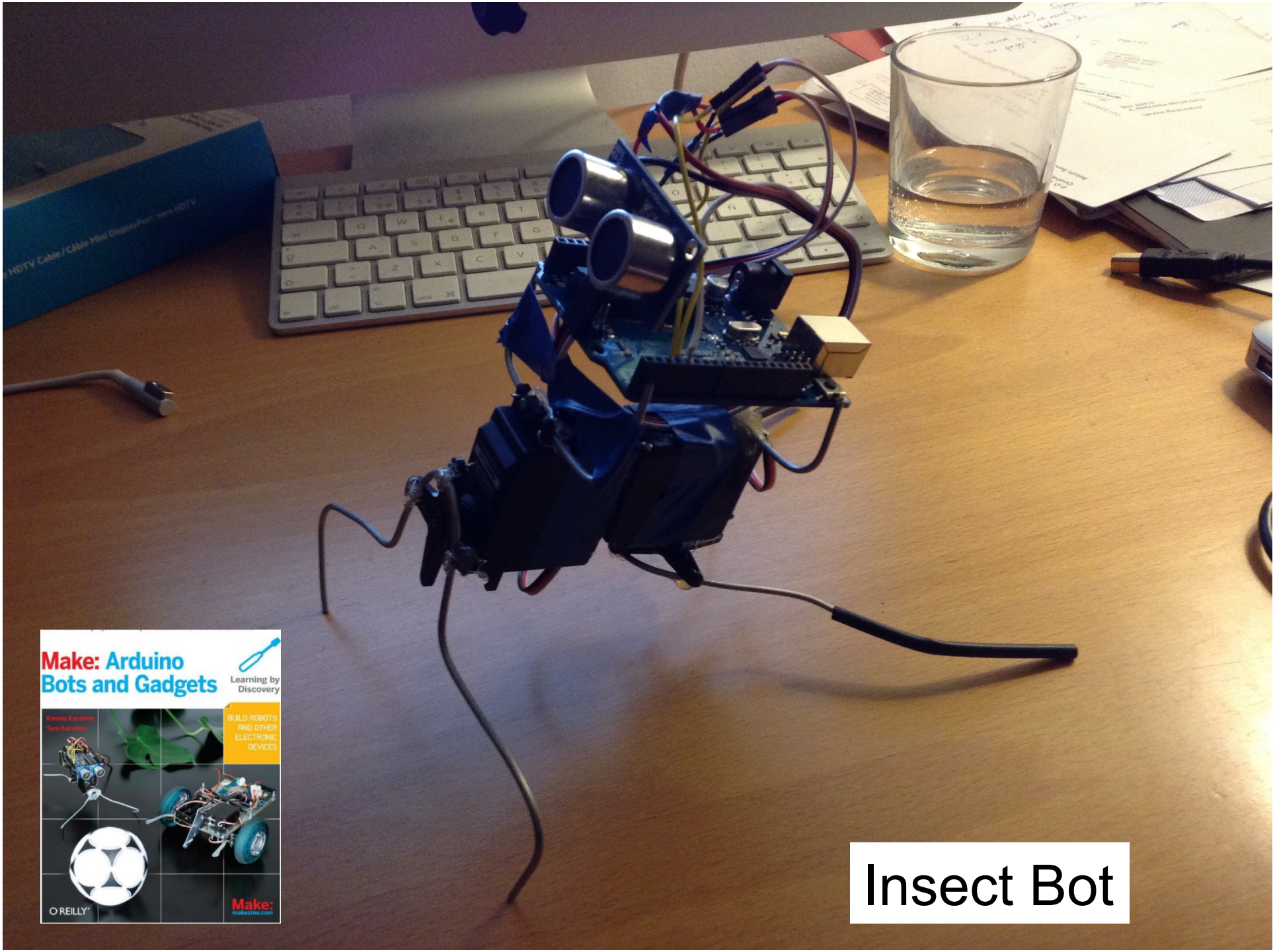
- Games
- VR
- Webisodes / Entertainment
- Software development
- Startups

siliconartists

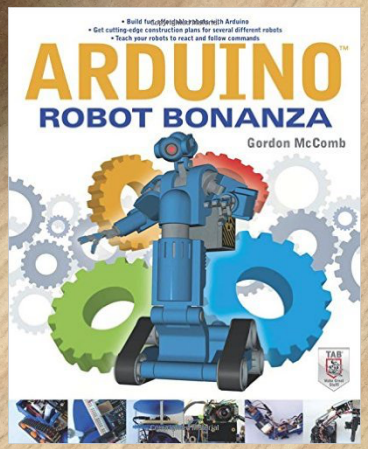
Euro-Profile
In-Depth European IT Profiles
Sold ✔

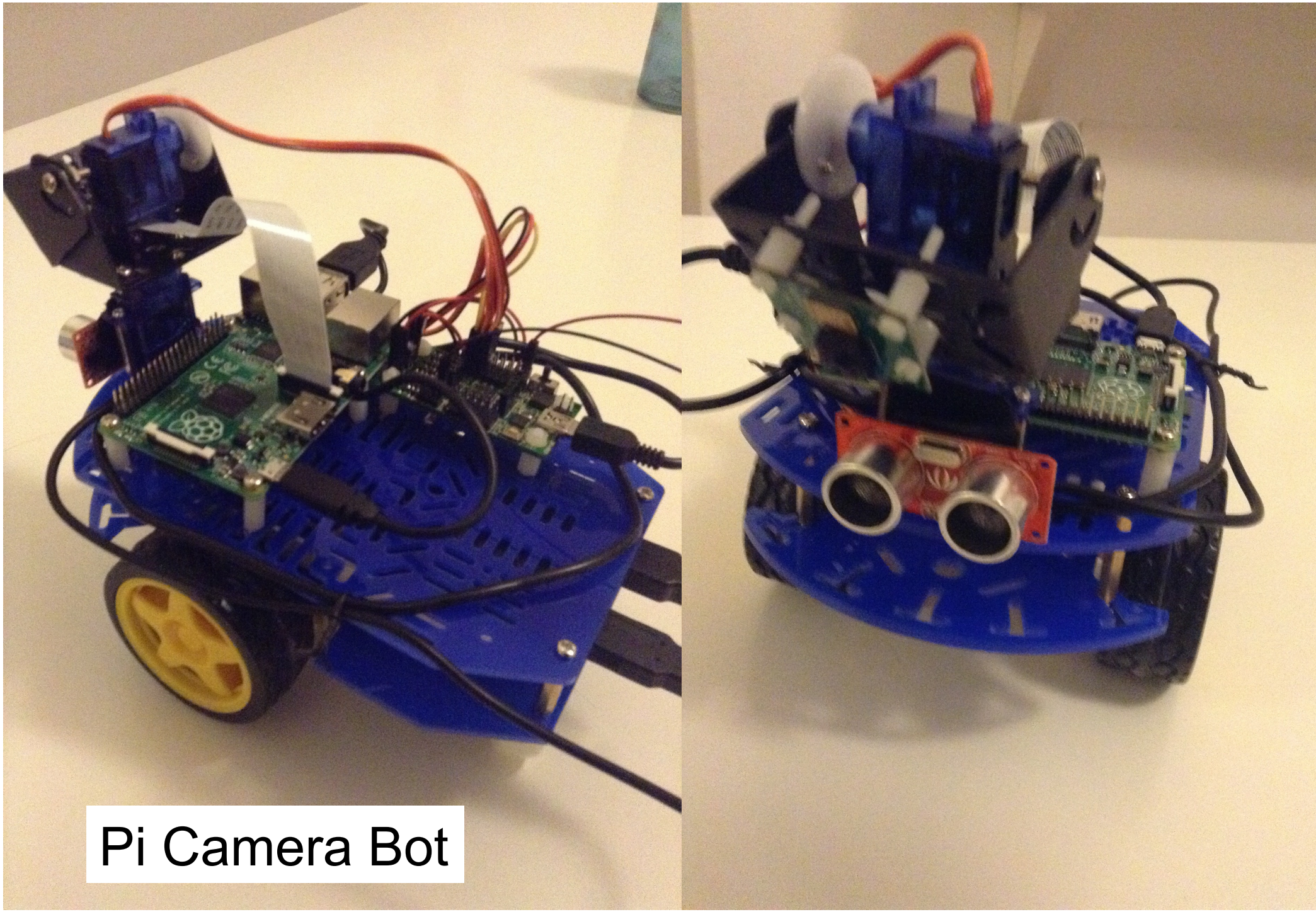
VEMM
Sold ✔

simon@robotlux.com
@eurodemanding


Make: Arduino Bots and Gadgets
Learning by Discovery
Kimmo Karvinen
Tero Karvinen
BUILD ROBOTS AND OTHER ELECTRONIC DEVICES
O'REILLY
Make:
makezine.com
Insect Bot

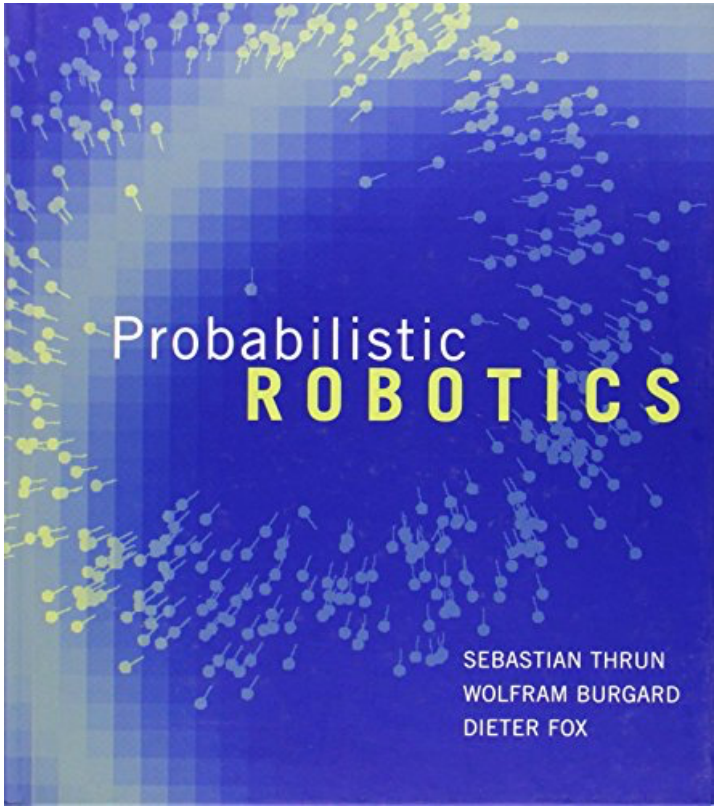
Teach Bot




Pi Camera Bot

SLAM

- Simultaneous Localization and Mapping
- Localization: How does a robot know where it is in a world of untrustworthy sensors?
- Mapping: How can it make a map when it doesn't know where it is?



The basis for the EKF-SLAM method is to describe the vehicle motion in the form

$$P(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{u}_k) \Longleftrightarrow \mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_k, \quad (6)$$

where $\mathbf{f}(\cdot)$ models vehicle kinematics and where $\mathbf{w}_k$ are additive, zero mean uncorrelated Gaussian motion disturbances with covariance $\mathbf{Q}_k$. The observation model is described in the form

$$P(\mathbf{z}_k \mid \mathbf{x}_k, \mathbf{m}) \Longleftrightarrow \mathbf{z}(k) = \mathbf{h}(\mathbf{x}_k, \mathbf{m}) + \mathbf{v}_k, \quad (7)$$

where $\mathbf{h}(\cdot)$ describes the geometry of the observation and where $\mathbf{v}_k$ are additive, zero mean uncorrelated Gaussian observation errors with covariance $\mathbf{R}_k$.

Life's too short.
What can we **steal**?

With these definitions the standard EKF method [31], [14] can be applied to compute the mean

$$\begin{bmatrix} \hat{\mathbf{x}}_{k|k} \\ \hat{\mathbf{m}}_k \end{bmatrix} = \mathrm{E}\begin{bmatrix} \mathbf{x}_k \\ \mathbf{m} \end{bmatrix} \mid \mathbf{Z}_{0:k} \Big],$$

and covariance

$$\begin{aligned} \mathbf{P}_{k|k} &= \begin{bmatrix} \mathbf{P}_{xx} & \mathbf{P}_{xm} \\ \mathbf{P}_{xm}^T & \mathbf{P}_{mm} \end{bmatrix}_{k|k} \\ &= \mathrm{E}\left[\begin{pmatrix} \mathbf{x}_k - \hat{\mathbf{x}}_k \\ \mathbf{m} - \hat{\mathbf{m}}_k \end{pmatrix} \begin{pmatrix} \mathbf{x}_k - \hat{\mathbf{x}}_k \\ \mathbf{m} - \hat{\mathbf{m}}_k \end{pmatrix}^T \mid \mathbf{Z}_{0:k} \right] \end{aligned}$$

of the joint posterior distribution $P(\mathbf{x}_k, \mathbf{m} \mid \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0)$ from:

Time-update

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k) \quad (8)$$

$$\mathbf{P}_{xx,k|k-1} = \nabla\mathbf{f}\, \mathbf{P}_{xx,k-1|k-1} \nabla\mathbf{f}^T + \mathbf{Q}_k \quad (9)$$

where $\nabla\mathbf{f}$ is the Jacobian of $\mathbf{f}$ evaluated at the estimate $\hat{\mathbf{x}}_{k-1|k-1}$. There is generally no need to perform a time-update for stationary landmarks[3].

Observation-update

$$\begin{bmatrix} \hat{\mathbf{x}}_{k|k} \\ \hat{\mathbf{m}}_k \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{x}}_{k|k-1} \\ \hat{\mathbf{m}}_{k-1} \end{bmatrix} + \mathbf{W}_k \left[\mathbf{z}(k) - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}, \hat{\mathbf{m}}_{k-1}) \right] \quad (10)$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{W}_k \mathbf{S}_k \mathbf{W}_k^T \quad (11)$$

where

$$\mathbf{S}_k = \nabla\mathbf{h}\mathbf{P}_{k|k-1}\nabla\mathbf{h}^T + \mathbf{R}_k$$

$$\mathbf{W}_k = \mathbf{P}_{k|k-1}\nabla\mathbf{h}^T\mathbf{S}_k^{-1}$$

and where $\nabla\mathbf{h}$ is the Jacobian of $\mathbf{h}$ evaluated at $\hat{\mathbf{x}}_{k|k-1}$ and $\hat{\mathbf{m}}_{k-1}$.

ROS

Robot Operating System

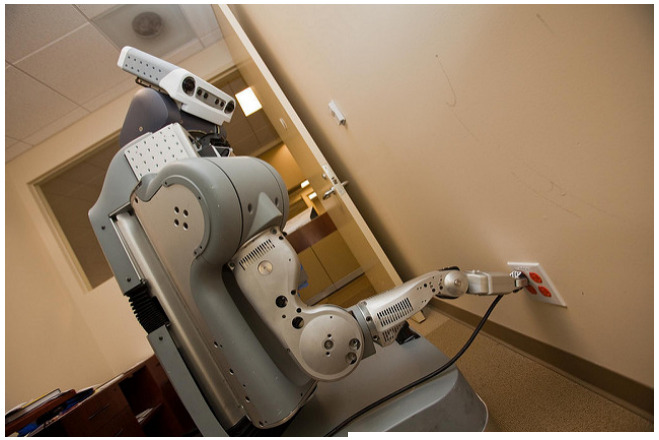
- Not just for robots
- Not an operating system

ROS = An open source framework and a collection of packages that are useful in robotics

ROS Packages

- Navigation: SLAM, autonomous navigation…
- Robot Arm: Kinematics, inverse kinematics…
- Hardware Drivers: LIDARs, sound, motors, vision…
- Interfaces: OpenCV, Caffe, Speech to text…
- …

The History of ROS

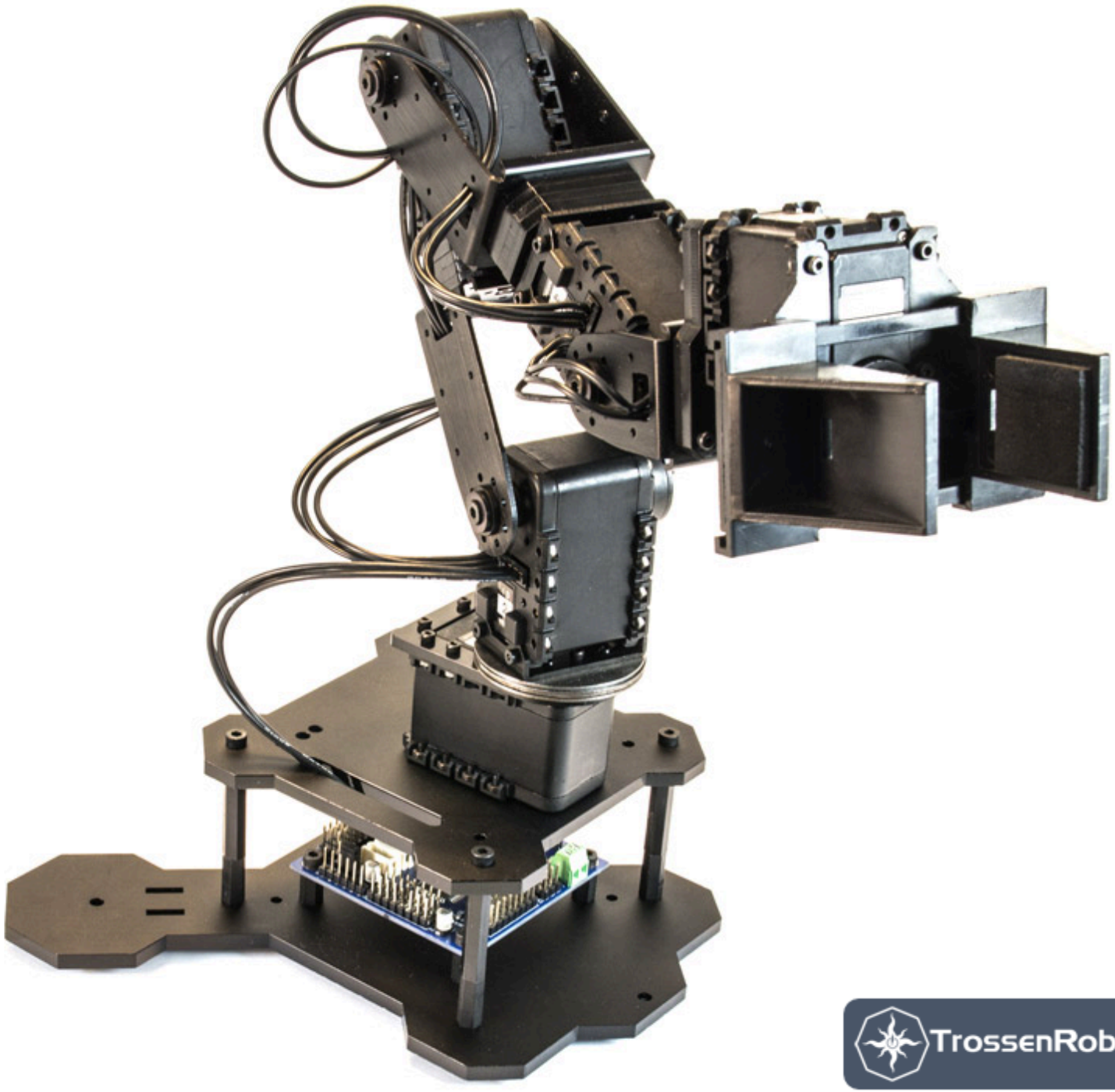


Open Source
Robotics Foundation

TurtleBot™

ROS

open hardware


TrossenRobotics.com

Parrot AR.Drone 2.0 Elite with ROS drivers

Erle Robotics

Driverless Development Vehicle with ROS Interface

Choose either the Lincoln MKZ or Ford Fusion as a development vehicle.

Full control of

- throttle
- brakes
- steering
- shifting
- turn signals

Read production sensor data such as

- gyros
- accelerometers
- gps
- wheel speeds
- tire pressures

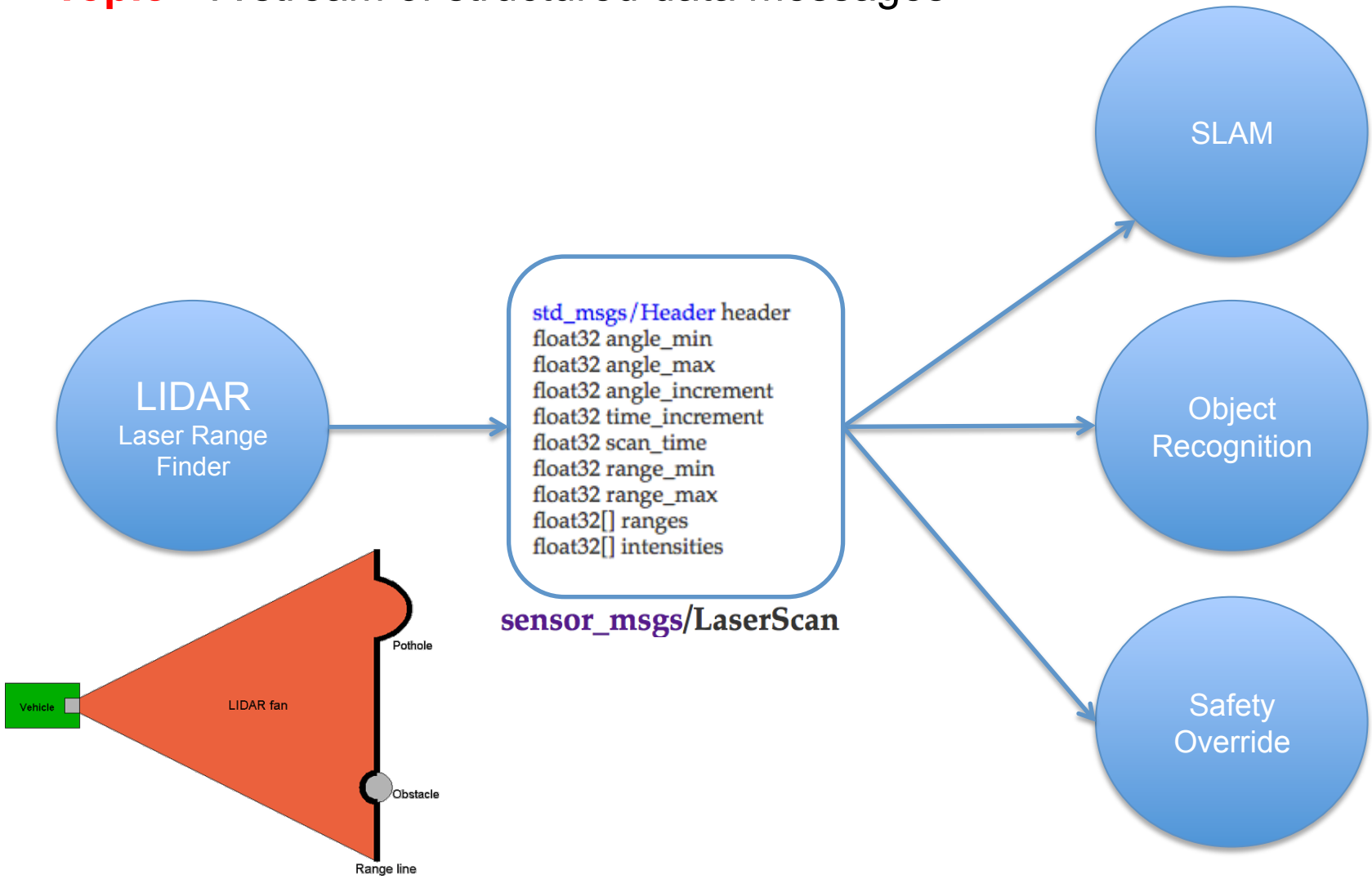
DATASPEED Inc.

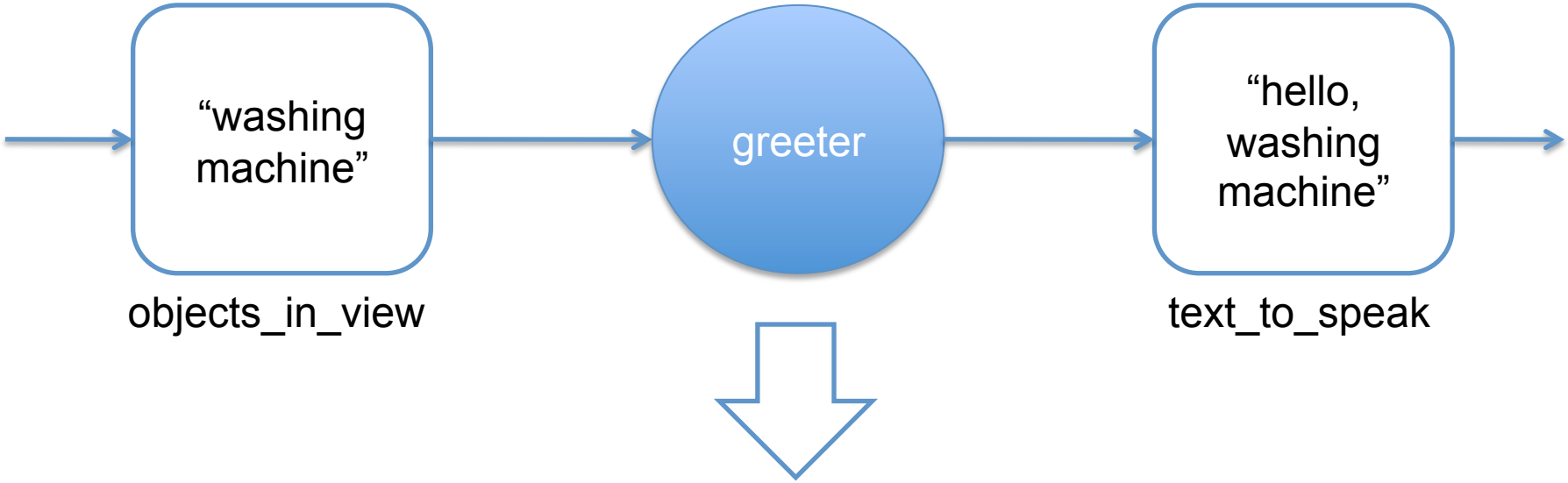
There are no visual indications that the production vehicle has been modified. All electronics and wiring are hidden.

ROS Architecture: Nodes and Topics

Node – independent software process that publishes and subscribes to Topics

Topic – A stream of structured data messages





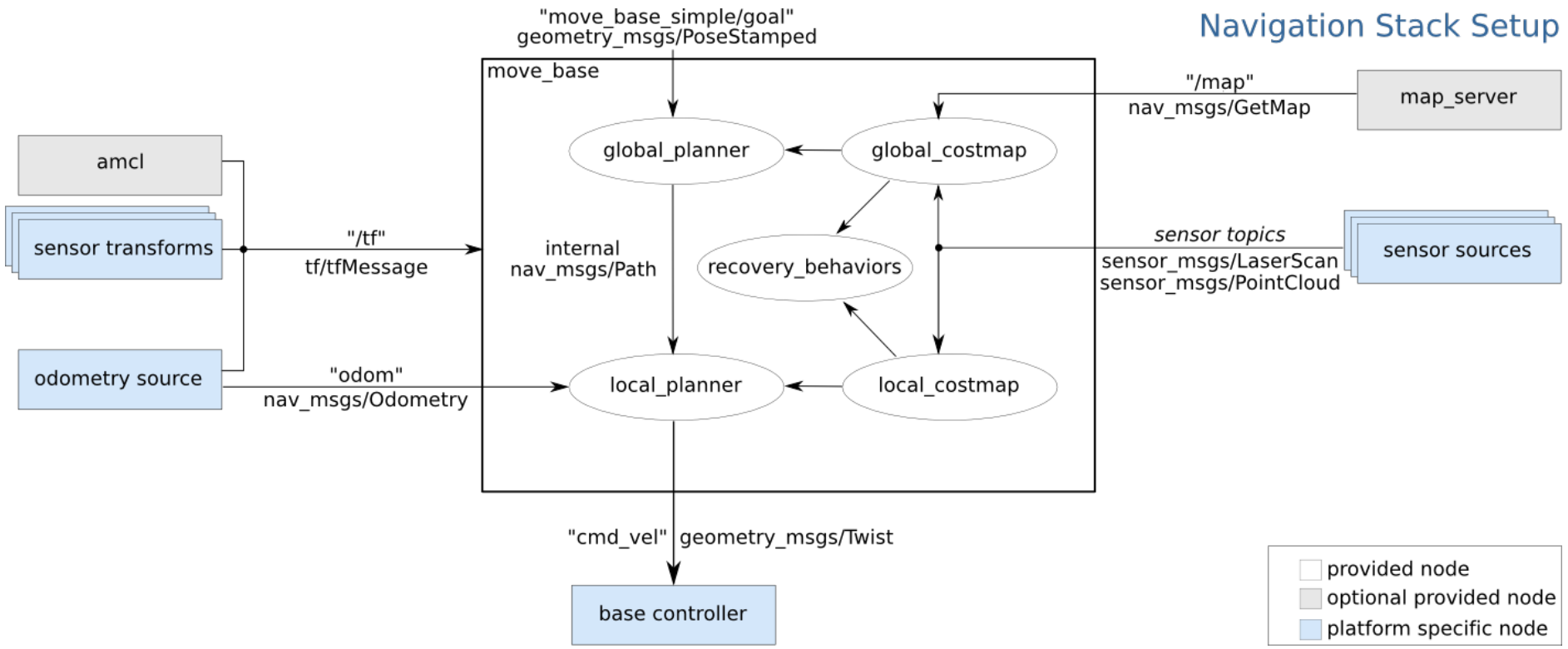

```python
import rospy
from std_msgs.msg import String
Pub = 0

if __name__ == '__main__':
    rospy.init_node('greeter', anonymous=True)
    global Pub
    Pub = rospy.Publisher('/text_to_speak', String, queue_size=10)
    rospy.Subscriber("/objects_in_view", String, message_received)
    rospy.spin()

def message_received(string_message):
    global Pub
    Pub.publish("Hello, " + string_message)
```

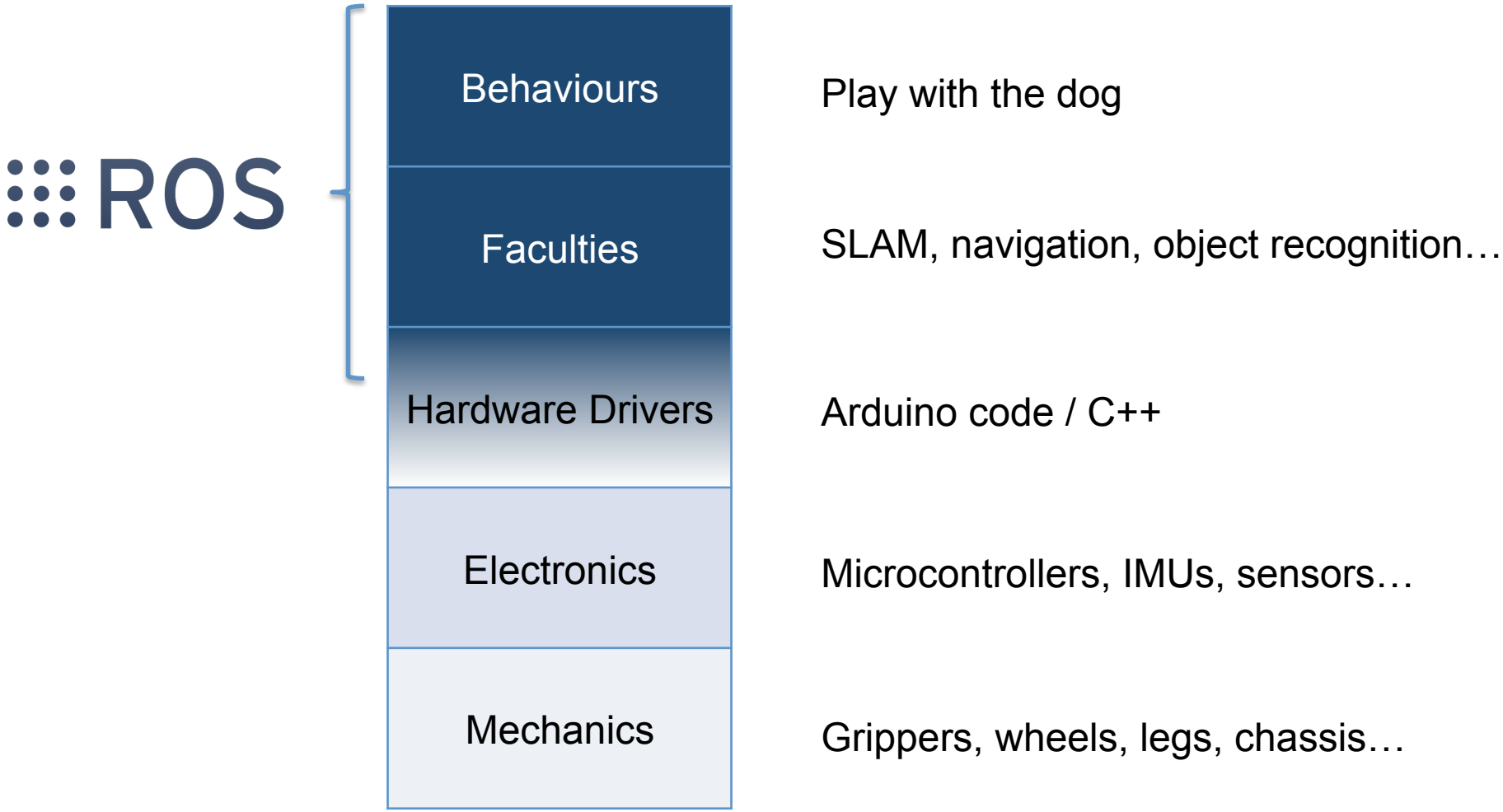
greeter.py

ROS Navigation Stack

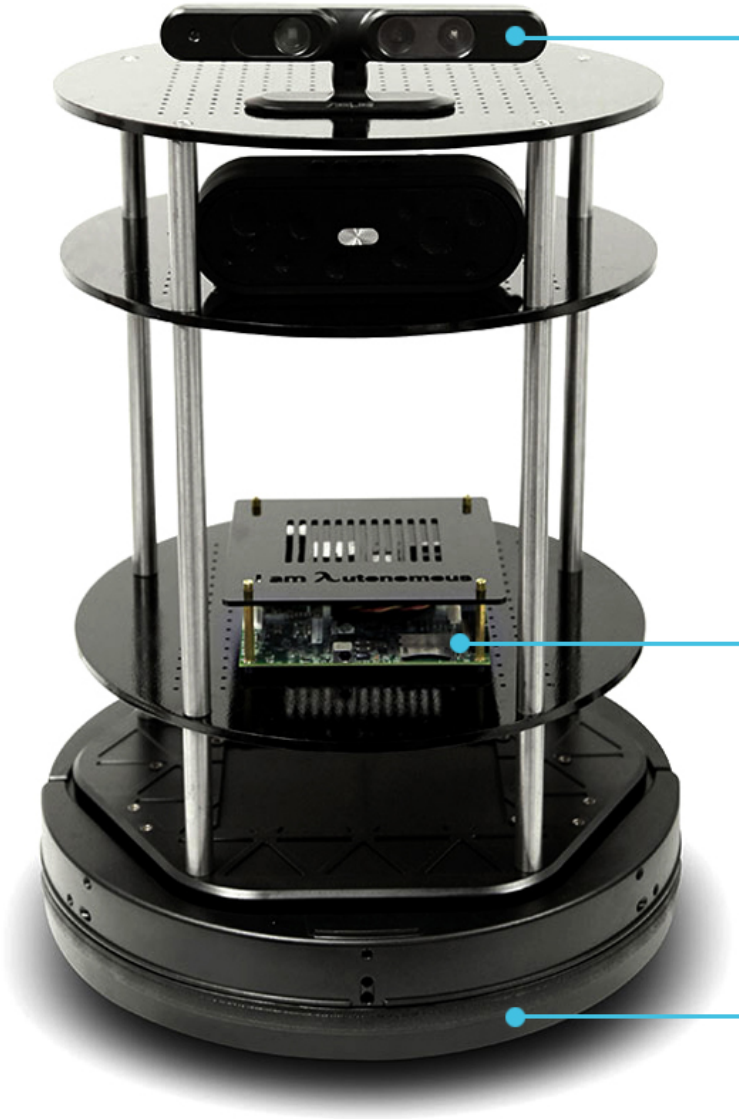


Enough to do SLAM and autonomous navigation

Where in the stack do you want to experiment?



The Deep Learning Robot



3D Depth Camera

Google TensorFlow
Robot Operating System (ROS)
Torch
Theano
Caffe
CUDA + cuDNN
Tegra K1
Wifi + Bluetooth

Mobile Base

www.autonomous.ai
$1000 = GBP 700

Kobuki Mobile Base

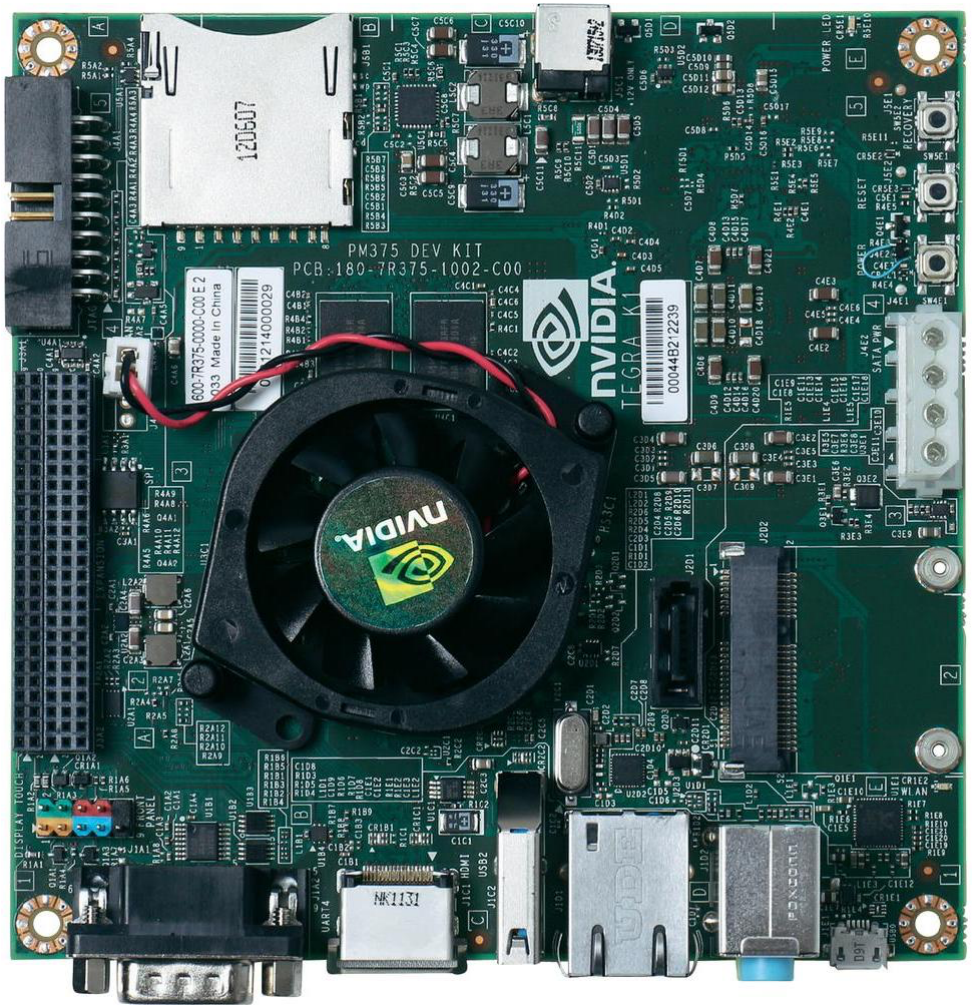
- 2 wheel, differential drive
- Wheel encoders
- 3 bump sensors
- 1 cliff sensor
- Wheeldrop sensor
- Gyroscope
- IR-based docking
- USB communication with robot motherboard

nVidia Jetson TK1

Robot motherboard:

- ARM CPU
- 2 Gb RAM
- 16Gb Flash
- nVidia GPU with 192 CUDA cores
- Wifi & Bluetooth

Principal value add is CUDA acceleration of deep learning tools



Asus Xtion Pro Live

- Camera with RGBD (RGB + depth output)
- Uses infrared to rangefind
- Microphone
- USB communication with Robot motherboard

- Primesense, succesor to Kinect
- Intel RealSense3D is like succesor to this

Demo

Great, free, introductory course on the maths of SLAM, autonomous navigation

Artificial Intelligence for Robotics
UDACITY

https://www.udacity.com/course/artificial-intelligence-for-robotics--cs373

Deep Learning

Neural Networks

Machine Learning

If X has features a, b, c, d… then what is Y ?

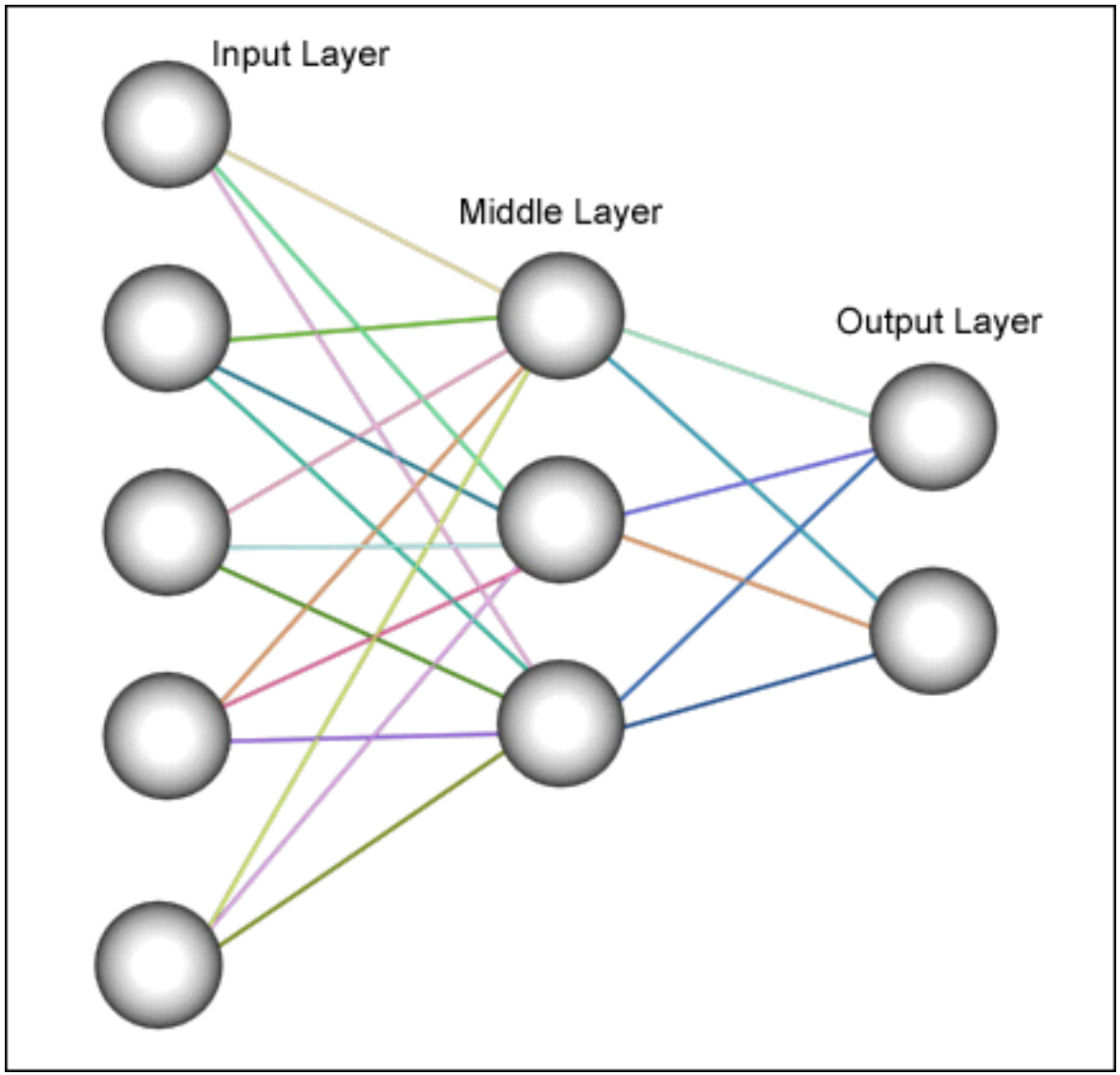
If X is age 42 then what is their net worth ?

If X is a house with 3 BDR, centre of Cambridge and in lousy condition then what is the price ?

If X is an email with words “viagra”, “cheap”… then is it spam ?

If X is an image with pixels (1, 2, 3…10,000) then is it showing my grandmother?

Neural Networks



Neural Networks

- Retro *and* futuristic

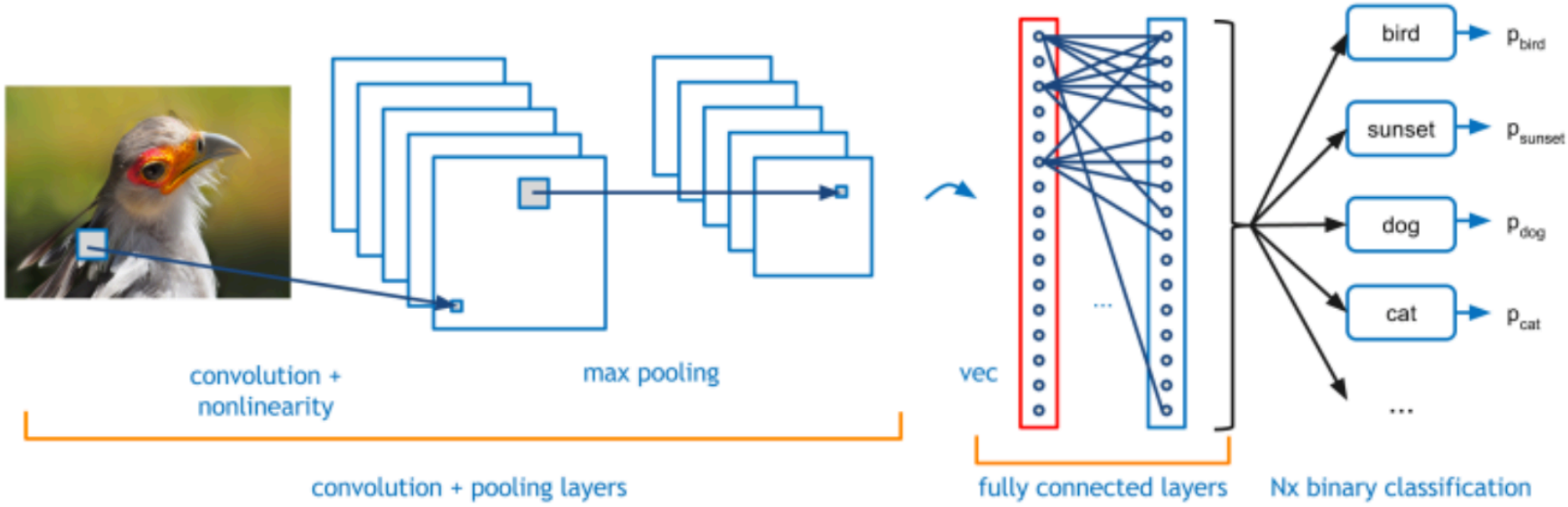
- They work now (but didn't in the 80s) because of
  - Fast CPUs
  - Fast GPUs (all thanks to gamers)
  - Large datasets

- Deep Learning
- CNN: Convolutional Neural Networks
- RNN: Recurrent Neural Networks
- ….

- Is back propagation the fundamental computational building block of the human brain?

Caffe

- Tool for designing, training and testing neural networks, especially related to vision
- CUDA accelerated
- Widely used in research
- Pre-installed on the robot (along with similar Google TensorFlow, Theano etc.)



Demo

Survey

What next?

Richard Branson (left) tries out Simon Birrell's award-winning game.

Norfolk boy's video game proves a winner

Norfolk computer whizz


PONG
ATARI

10 print “piss off”
20 goto 10

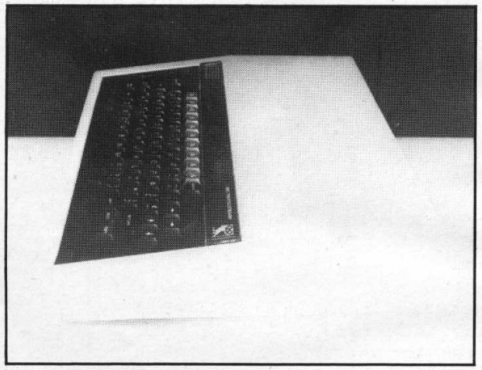
Listing 1. One-dimensional life

```
   10 REM 1D Life
   20 REM by Susan Stepney
   30 REM for B/B+/M/C/E/A
   40 REM (c) Acorn User November 1988
   50 :
   60 ON ERROR MODE 7:PROCerror
   70 MODE 7
   80 PROCsetup
   90 MODE mode%
  100 VDU 23,1,0;0;0;0;
  110 PROCscreen
  120 WAIT=GET
  130 END
  140 :
  150 DEF PROCsetup
  160 PROCparams
  170 mode%=2
  180 M%=160*2^(2-(mode% MOD 3))
  190 line%=256
  200 xres%=1280/M%
  210 yres%=1024/line%
  220 DIM c1% M%
  230 DIM c2% M%
  240 IF rnd PROCinitrnd ELSE PROCinitse
ed
  250 ENDPROC
  260 :
  270 DEF PROCscreen
  280 LOCAL A%,B%,X%,Y%
  290 Y%=1023
  300 FOR J%=0 TO line%-1
  310 IF J% MOD 2=0 THEN A%=c1%:B%=c2% E
LSE A%=c2%:B%=c1%
  320 FOR I%=0 TO N%-1
  330 GCOL 0,?(A%+I%)
  340 PLOT 69,I%*xres%,Y%
  350 GCOL 0,?(A%+M%-1-I%)
  360 PLOT 69,(M%-1-I%)*xres%,Y%
  370 index%=?(A%+(I%-N%+M%) MOD M%)
  380 index2%=?(A%+(M%-1-I%-N%) MOD M%)
  390 FOR K%=-N%+1 TO N%
  400 index%=index%+?(A%+(I%+K%+M%)MOD M
%)
  410 index2%=index2%+?(A%+ (M%-1-I%+K%)
MOD M%)
  420 NEXT
  430 ?(B%+I%)=rule%(index%)
  440 ?(B%+M%-1+I%)=rule%(index2%)
  450 NEXT I%
  460 X%=N%*xres%
  470 FOR I%=N% TO M%-1-N%
  480 GCOL 0,?(A%+I%)
  490 PLOT 69,X%,Y%
  500 index%=?(A%+I%-N%)
  510 FOR K%=-N%+1 TO N%
  520 index%=index%+?(A%+I%+K%)
  530 NEXT
  540 ?(B%+I%)=rule%(index%)
  550 X%=X%+xres%
  560 NEXT I%
  570 Y%=Y%-yres%
  580 NEXT J%
  590 ENDPROC
  600 :
  610 DEF PROCerror
  620 VDU 23,1,1;0;0;0;
  630 REPORT:PRINT " on line ";ERL
  640 END
  650 ENDPROC
  660 :
  670 DEF PROCparams
  680 INPUT "neighbourhood      N : "N%
  690 INPUT "states             S : "S%
  700 dim%=(S%-1)*(2*N%+1)+1
  710 INPUT "rule               R : "rul
e$
  720 IF LEN(rule$)<dim% rule$=STRING$(d
im%-LEN(rule$),"0")+rule$
  730 DIM rule%(dim%)
  740 FOR I%=0 TO dim%-1
  750 rule%(I%)=EVAL(MID$(rule$,dim%-I%,
1))
  760 IF rule%(I%)>=S% VDU7:PRINT"invali
d state ";rule%(I%);" in rule ";rule$:EN
D
  770 NEXT I%
  780 PRINT "seed pattern"'"  RETURN for
 random,"'"  or string of numbers in ran
ge 0-";S%-1;"  : ":INPUT""seed$
  790 IF seed$="" THEN rnd=TRUE ELSE rnd
 = FALSE
  800 ENDPROC
  810 :
  820 DEF PROCinitrnd
  830 FOR I%=0 TO M%
  840 ?(c1%+I%)=RND(S%)-1
  850 NEXT
  860 ENDPROC
  870 :
  880 DEF PROCinitseed
  890 FOR I%=0 TO M%
  900 ?(c1%+I%)=0
  910 NEXT
  920 len%=LEN(seed$)
  930 l2%=len%/2
  940 start%=M%/2-l2%
  950 end%=M%/2+l2%
  960 IF len% MOD 2=0 THEN end%=end%-1
  970 FOR I%=start% TO end%
  980 ?(c1%+I%)=EVAL(MID$(seed$,I%-start
%+1,1))
  990 IF ?(c1%+I%)>=S% VDU7:PRINT"invali
d state ";?(c1%+I%);" in seed pattern ";
seed$:END
 1000 NEXT
 1010 ENDPROC
```

Machine Code Made Easy

Steve Willis

We begin our series on Machine Code with an invaluable introduction to 6502 microprocessor circuitry.



PROGRAMMING BY NUMBERS – A GUIDE TO 6502 MACHINE CODE

Right, so you have now learnt BASIC but your friends keep mentioning a thing called MACHINE CODE, do they mean the Serial No. of the computer? Well, over the next few articles I hope to show you what MACHINE CODE is all about with particular reference to the BBC micro.

During this series I will make reference to the BBC USER Guide (BUG) where the information given is relevant and can save me being repetitive. Another book that you may find useful, if you seriously intend using machine code programs (mad fools!) is PROGRAMMING THE 6502 by ZAKS (published by SYBEX & being Volume 1 of 4).

Okay, first we had better get an idea of how a microprocessor works in relation to it's circuitry. The diagram, Figure 1, shows a block diagram of a microprocessor and a minimal circuit with which it operates. We will build upon this diagram in future articles. Figure 2 shows the main components of the microprocessor itself.

6502 PROCESSOR (HEXADECIMAL)

The 6502 processor is an 8-bit processor, meaning that data is transferred around the system by 8 individual parallel lines simultaneously. Each of these lines is digitally switched to either +5volts or 0 volts (i.e. 'ON' or 'OFF'); this is the basis of a BINARY system. As there are eight lines, each of which may be 'ON' or 'OFF', it can be calculated that there are 256 different combinations for one BYTE (the term given to one transfer of the 8 parellel BITS.

The usual way of writing down an 8-bit number is to use '1' for a line that is 'ON' and '0' for one that is off. The 8-bit number is then given in the form 11001010 etc., where the first digit is known as 'D7' and the last as 'DO' (with D6, D5 . . in the middle). However this BINARY representation is not very easy to decode into a decimal number — the above being the binary equivalent of 202. The conversion from binary to decimal is achieved as follows:—

BINARY — 11001010

1	×	128	=	128
1	×	64	=	64
0	×	32	=	0
0	×	16	=	0
1	×	8	=	8
0	×	4	=	0
1	×	2	=	2
0	×	1	=	0
		DECIMAL	=	202

As a matter of interest, as the 8 bits are designated D7-DO the conversion is X7 * 2 7) + (X6 * 2 6) + . . . where X is the binary bit value (0 or 1) and the power of 2 is the D value.

For most ordinary mortals the conversion from 8-bit binary to decimal proves a headache to say the least — unless you have hours to waste. Thus the hexadecimal system is used, this being not only easier to use but also, we will see later, more logical for 8-bit use. In this system the 8-bit binary number is split into two 4-bit groups (D7-D4 & D3-D0) and each is then given a hexadecimal code.

Given a 4-bit group (e.g. '1010') there are 16 possible combinations from '0000' to '1111'. Now in decimal we would require two digits to represent any of these combinations above '1001' (9) but we wish to have only one digit to represent our 4-bit group. Hexadecimal achieves this by representing 10-15 as A-F, so the equivalent decimal/hexadecimal groups are as shown on page 71 of BUG.

Just as a check I give a few examples of conversions below:—
The conversion to decimal from hexadecimal is much easier than from binary — 1st Hex * 16 + 2nd Hex = Decimal.

BINARY	HEXADECIMAL	DECIMAL
0001 0000	1 0	16
0101 0111	5 7	87
0110 1000	6 8	104
0010 1010	2 A	42
0100 1011	4 B	75
1000 1100	8 C	140
1001 1101	9 D	157
1010 1110	A E	174
0011 1111	3 F	63

THE DATA BUS

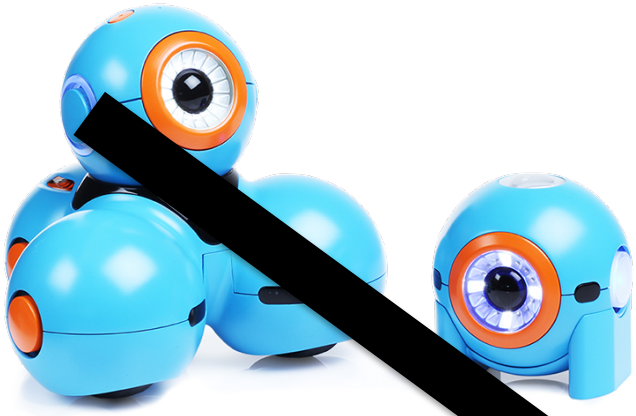
Now we have looked at an 8-bit number let us now look at the main highway from the microprocessor which gives it the name '8-bit processor' — the DATA bus. This 'highway' is the channel by which the microprocessor collects and distributes data to all the 'data responsive' devices in the computer. some of these devices have permanent values stored in them which can only be read and not replaced, in Figure 1 the ROM devices are of this type. Others can both be read and written to — in our minimal system these are the RAM devices.

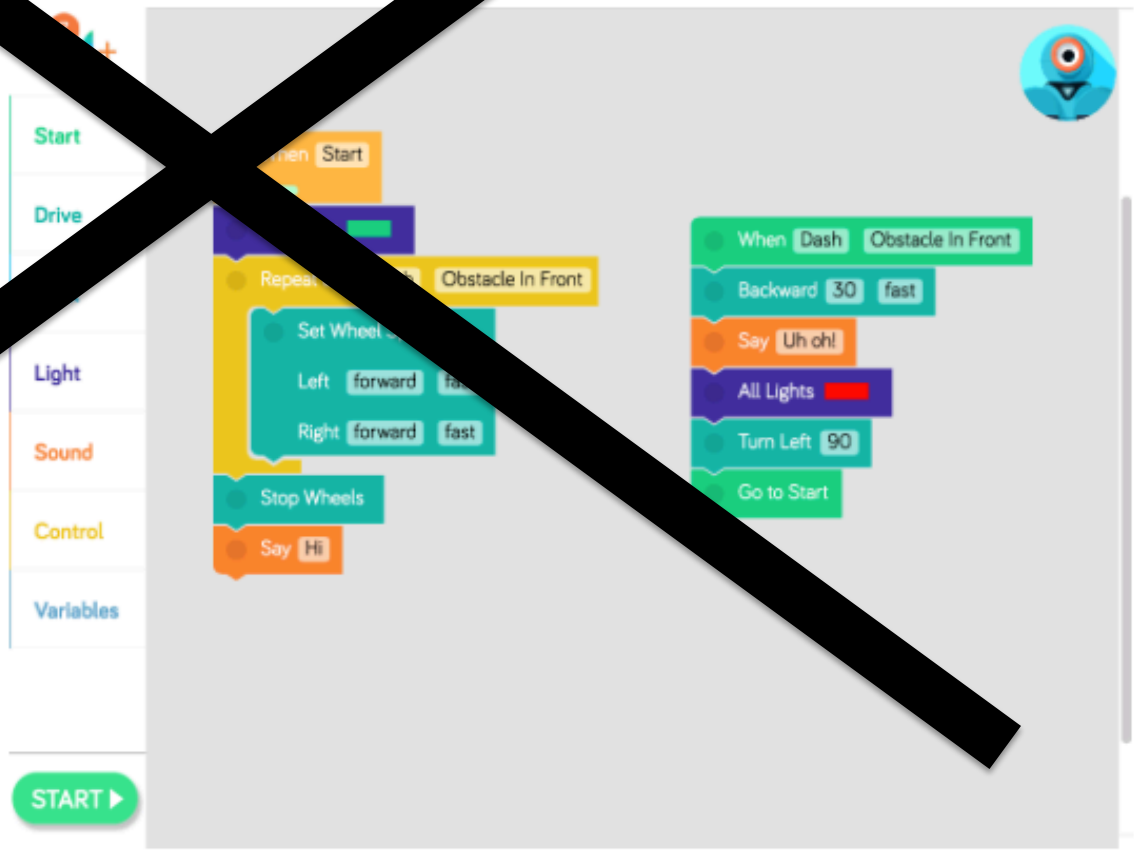
The name ROM stands for 'READ ONLY MEMORY' and these devices generally hold the operating system programs and language programs — yes, they are programs. RAM stands for 'RANDOM ACCESS MEMORY' and these devices are available for storing temporary information. This temporary information may be one of your programs or data required by the operating system or language programs.

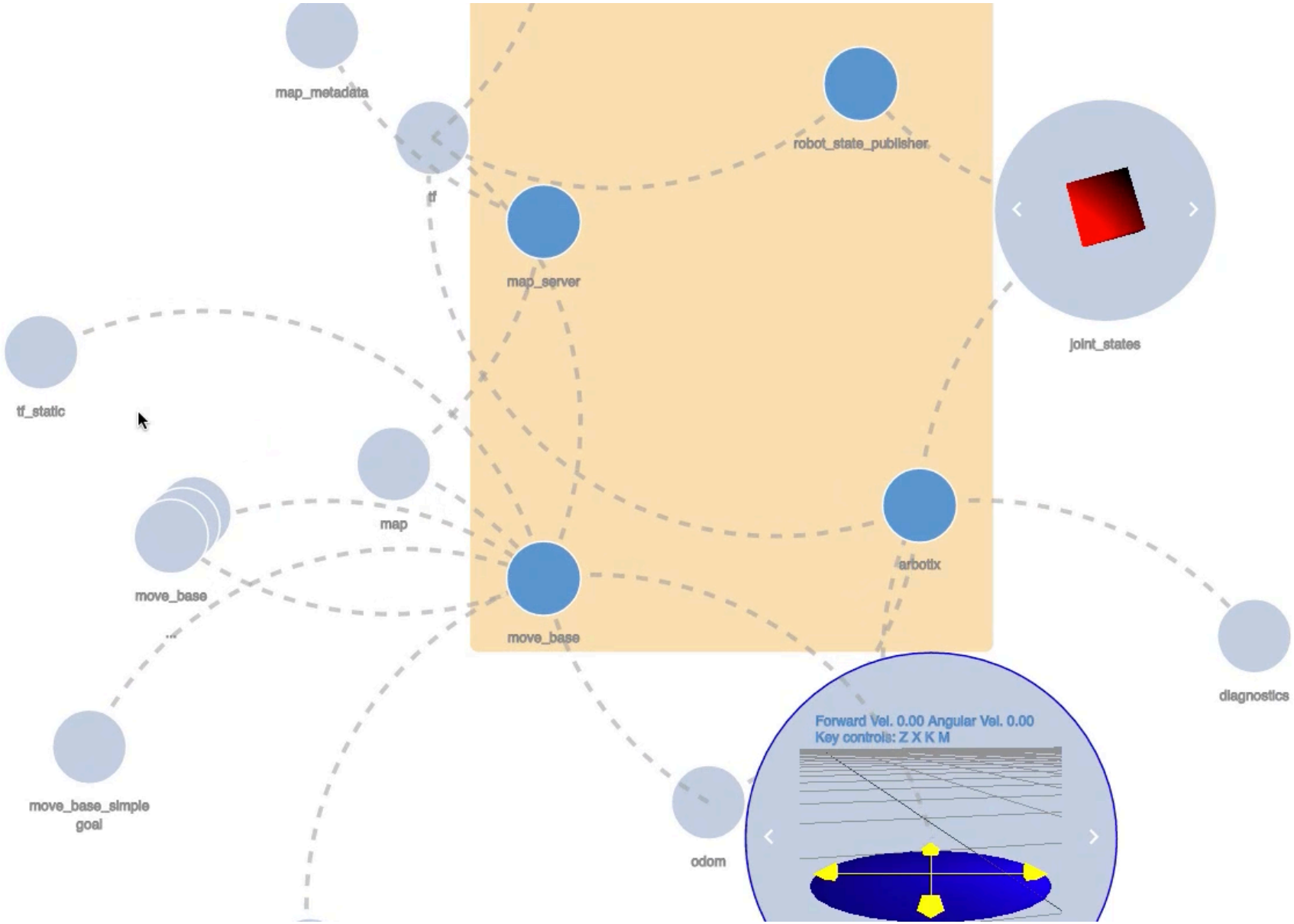
However, it's all very well having data flying up and down the data bus but how does the processor manage to get or store data at the right device. Well the whole system can be equated to a single line model railway, where a data BYTE train can only run in one direction at a time and only one train is allowed on the track at any time. The train is guided to the correct device by points and allowed out by signals, thus a predetermined route can be set up at the start of the run, allowing data to leave one device on the system and travel to another.

So, what sets the points and signals?

10 print “piss off”
20 goto 10




Start
Drive
Light
Sound
Control
Variables
START
Start
Obstacle In Front
Left forward
Right forward fast
Stop Wheels
Say Hi
When Dash Obstacle In Front
Backward 30 fast
Say Uh oh!
All Lights
Turn Left 90
Go to Start


map_metadata
tf
robot_state_publisher
map_server
joint_states
tf_static
map
move_base
...
move_base
arbotix
diagnostics
move_base_simple
goal
odom
Forward Vel. 0.00 Angular Vel. 0.00
Key controls: Z X K M

Thanks

simon@eurodemand.com
www.artificialhumancompanions.com
@eurodemanding