MATLAB interface to Lego EV3

Liam O’Sullivan and Peter Corke
MATLAB and the NXT brick

- Aachen RWTH Toolbox
- Low-level primitives
- Requires downloading a custom binary into the brick
- Handles BlueTooth
- Has quite poor motor control
  - no move to position
  - synchronised motion hard

```matlab
hNXT = COM_OpenNXT();           % look for USB devices
COM_SetDefaultNXT(hNXT);        % sets global default handle

% now we don't have to use hNXT anymore!
NXT_PlayTone(440, 500);
pause(0.5);
NXT_PlayTone(440, 500);

% but never forget to clean up after your work!!!
COM_CloseNXT(hNXT);
```
Lego EV3

- Released September 2013
  - successor to the NXT brick
The new EV3

- 300MHz TI ARM 9 processor
  - 64M RAM
  - 16M flash
  - USB port
  - can be cascaded
  - Bluetooth
- Linux (Angstrom distro) 2.6.33 kernel
  - All source code available
  - Virtual machine for running Lego code
  - telnet daemon
- 4 motors!
Connectivity

• WiFi
• USB + WiFi dongle
  – can access a command shell
• USB + HID
  – appears as a HID device
  • under Mac/Linux not in /dev
  • need libhid, hidapi etc
Development

- Get the source code from github
- Customize and build an image on an SD card
- Install own programs
Virtual machine

- At Linux runlevel 5
  - the virtual machine is process: lms2012
- memory oriented architecture
- 3 operand instructions, eg. A=B+C
- Operands are:
  - constant, local, global
  - of different lengths and types (int, float, array etc)
- Supports threads
- Host can upload
  - entire programs, eg. from Lego development tools
  - short sequences
Talking to the brick

• Build a command
  – a multi-byte sequence
• Send it to the brick
• Optionally get the response

• Brick state and communications handles
  – Brick object
• An EV3 command buffer
  – Command object
function playThreeTones(brick)
% Brick.playThreeTones Play three tones on the brick
% 
% Brick.playThreeTones() plays three tones consecutively on 
% the brick with one upload command.
% 
% Example::
% 
% b.playThreeTones();
% cmd = Command();
% cmd.addHeaderDirect(42,0,0);
% cmd.opSOUND_TONE(5,440,500);
% cmd.opSOUND_READY();
% cmd.opSOUND_TONE(10,880,500);
% cmd.opSOUND_READY();
% cmd.opSOUND_TONE(15,1320,500);
% cmd.opSOUND_READY();
% cmd.addLength();
% brick.send(cmd);
end

function playTone(brick, volume, frequency, duration)
% Brick.playTone Play a tone on the brick
% 
% Brick.playTone(volume,frequency,duration) plays a 
% tone at a volume, frequency and duration.
% 
% Notes::
% - volume is the tone volume from 0 to 100.
% - frequency is the tone frequency in Hz.
% - duration is the tone duration in ms.
% 
% Example::
% 
% b.playTone(5,400,500)
% cmd = Command();
% cmd.addHeaderDirect(42,0,0);
% cmd.opSOUND_TONE(volume,frequency,duration);
% cmd.addLength();
% brick.send(cmd);
end

function opSOUND_TONE(cmd,volume,frequency,duration)
% Command.opSOUND_TONE Add a opSOUND_TONE 
% 
% Command.opSOUND_TONE(volume,frequency,duration) adds a 
% opSOUND opcode with a TONE subcode to the command object.
% 
% Notes::
% - volume is the tone volume from 0 to 100
% - frequency is the tone frequency in Hz
% - duration is the tone duration in ms
% - opSOUND,LC0(1),LC1(volume),LC2(frequency),LC2(duration)
% 
% Example::
% 
% cmd.opSOUND_TONE(5,1000,500)
% cmd.addDirectCommand(ByteCodes.Sound);
% cmd.LC0(SoundSubCodes.Tone);
% cmd.LC1(volume);
% cmd.LC2(frequency);
% cmd.LC2(duration);
end
Communications

• Create a top-level Brick object

```python
b = Brick('ioType','usb')
```

```python
b = Brick('ioType','bt','serPort','/dev/rfcomm0')
```

```python
b = Brick('ioType', 'wifi', 'wfAddr', '192.168.1.104', 'wfPort', 5555, 'wfSN', '0016533dbaf5')
```

• Or use the MATLAB Instrumentation Toolbox

```python
b = Brick('ioType', 'instrbt', 'btDevice', 'EV3', 'btChannel', 1)
```

```python
b = Brick('ioType', 'instrwifi', 'wfAddr', '192.168.1.104', 'wfPort', 5555, 'wfSN', '0016533dbaf5')
```
Sensors

• Voltage interface

\[
voltage = b.uiReadVbatt();
\]

• Generic sensor interface

\[
name = b.inputDeviceGetName(layer, Device.Port1);
\]

\[
reading = b.inputReadSI(layer, Device.Port1, Device.USDistCM);
\]
Moving the motor

- **Position feedback**
  
  ```
  b.outputGetCount(layer, Device.MotorA);
  b.outputClrCount(layer, Device.MotorA);
  ```

- **Configure motor**
  
  ```
  b.outputPower(layer, Device.MotorA, pwr);
  b.outputStepSpeed(layer, Device.MotorA, speed, up, down, brakemode);
  b.outputStart(layer, Device.MotorA);
  b.outputStop(layer, Device.MotorA, brakemode);
  b.outputStopAll();
  ```

- **Brakemode**
  
  `Device.brake, Device.coast`
https://wiki.qut.edu.au/display/cyphy/Lego+EV3