Teo asked about callback/publisher dilemma

3 solutions -

1) create publisher inside call back
   - this causes inefficiency b/c publisher is then created and destroyed each time call back is called.

2) create global Publisher variable during setup, and refer to variable in call back
   - this "poisons the namespace" ie is a "gross" solution but works well

3) Use a class variable
   - similar problem as (2) and also works

Moral of the story: "do the thing that works" Zucker.
The goal is to get the robot to do something. Use good programming practices in-so-far as the code is maintainable, but fanciness is not always the best solution.

Subsumption Architecture

47 reactive robotic architecture \rightarrow behavior-based robotics

(sensor) \rightarrow \text{behaviors} \rightarrow \text{motor commands}

\text{Inhibition} \rightarrow \text{Subsumption}

✓ developed @ MIT by Rodney Brooks
✓ good for simple architectures
✗ bad for integrating representations/planning
✗ bad if behaviors don't have natural priority
Example of Subsumption Architecture: WALL-E

Modules
- Top of pile
- Gripper sensor
- Debris detector

State Machines
- Each machine is in only one state at a time (current state)
- State machines can do most things that subsumption arch can

Example of state machine: WALL-E

Problems:
- There can become a problem with scalability in both approaches
- Robot could get stuck in state if a case occurs that is unexpected
- This makes debugging difficult
```python
def main_loop(self):
    while True:
        if self.state == 'wander':
            self.wander()
        elif self.state == 'pickup':
            ...
    def wander(self):
        # emit random wheel commands
        self.rand_wheels()
        if self.see_debris():
            self.state = 'pick up'
```

Going from state machine diagram to code is straightforward.

**Sensors**
- The onset of sensors getting way better and cheaper has fast forwarded the robotics field.
- Every sensor this class will use has a transducer. This converts mechanical or radiant energy to an electrical signal.

**Types of sensors:**
- Passive → senses energy from the world
- Active → senses how world transforms self-produced energy
Examples of sensors
- Camera (passive sensor)
  → Camera w/ flash (active sensor)
- Xbox connect (active sensor)
- Photo-interrupter (active sensor)

- Rotary encoder (active)

- Detects absence/presence of an object by means of a light transmitter

- Converts angular position to digital code.
- Can measure angular displacement (rotational)
- However, the devise cannot sense direction w/ one sensor → to fix use 2 sensors

Cameras
- Great! Cameras can do complicated and interesting things
  → recognize things (faces, colors, etc...)
  → Visual odometry / servoing
    - perception / vision control
    - Computer vision
- Terrible! Computer vision is really hard.
  → Cameras take in too much data
example of cameras providing too much data
- approximate total data at turtle bot sensors
  - 2 encoders at 100 Hz → 25 bytes/sec
  - bump sensor + cliff sensor → 50 bytes/sec
  - gyro @ 30 Hz → 40 bytes/sec
- total: b/w 100 and 200 Hz
- total data of connect 2D camera
  - 640 x 480 video @ 30 Hz → 3.2 Mbytes/sec

Conclusion: cameras use way way more data

sensing internal state example:
- in humans (my stomach hurts feeling)
- in robots (indication that battery is low)

classnotes: 04/18/14

Analog vs Digital Sensor
- digital → 1's and 0's
- Analog → continuous voltage/current resistance

To convert between, we use ADC (analog to digital converter)

Qx) 3-bit ADC

input: 0 - 5 V
output: 000 - 111

- there exists accuracy and speed problems w/ A/D converters

maps voltage to binary 3-bit output
Example of digital camera as ADC

Sensor Array

Lens

→ sunlight hits flower, lens captures the light and the sensor converts analog information to digital pixels.

→ the more bits a camera sensor can represent, the more colors an image can have.

2-bits can represent 4 levels of granularity

What affects the intensity of a pixel?

→ the brightness of external light
→ the width of the aperture
→ the wider the aperture the less depth of field
→ the reflectivity of the object
→ the exposure time

What affects the color of a pixel?

→ sensor array uses wavelength sensitive material

<table>
<thead>
<tr>
<th>R</th>
<th>G</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>R</td>
<td>G</td>
</tr>
<tr>
<td>G</td>
<td>B</td>
<td>R</td>
</tr>
<tr>
<td>R</td>
<td>G</td>
<td>B</td>
</tr>
</tbody>
</table>

→ this is Bayer's method in sensor array of splitting up wavelength sensitivity
→ de-bayering is the process of assigning an R and B value to a G pixel - or any other combination. It looks and surrounding pixel values.

**Organization of Pixels in Computer memory**

<table>
<thead>
<tr>
<th>R_0</th>
<th>B_0</th>
<th>G_0</th>
<th>R_1</th>
<th>B_1</th>
<th>G_1</th>
<th>...</th>
<th>R_0</th>
<th>B_0</th>
<th>G_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

→ Row-major - Pixels in some row adjacent in memory

**Data Processing**

→ pipeline from incoming light to bits in memory

incoming light → aperture → exposure → A/D → white gain balance

red-eye removal → image stabilization → filters → ... → bytes in memory

→ this is a long pipeline w/ lots of knobs to tweak
Motors

Take away (important) notes:

1) Motors are easy to model electrically
2) Linear relationship b/w torque and speed
3) Quadratic relationship b/w torque and power
4) Gears are very helpful

Electric Model:

\[ V = IR + E \]
\[ = IR + \omega \cdot Ke \]

KVL: \[ V = IR + E \]
\[ = IR + \omega \cdot Ke \]

Torque:

\[ \tau = r \cdot F \]
\[ \text{torque (N.m)} \]
\[ \text{distance (m)} \]

SI UNITS

<table>
<thead>
<tr>
<th>Base</th>
<th>Derived</th>
</tr>
</thead>
<tbody>
<tr>
<td>length m</td>
<td>force N = kg m/s²</td>
</tr>
<tr>
<td>time s</td>
<td>mass kg</td>
</tr>
<tr>
<td>J N m = kg m²/s²</td>
<td>current A = W J/s</td>
</tr>
<tr>
<td></td>
<td>Power</td>
</tr>
</tbody>
</table>

Electrical Power

\[ P_e = (I \text{amps}) \cdot (V \text{volts}) \]
\[ \text{in watts = J/s} \]

\[ P_e \neq P_m \text{ b/c of energy loss during conversion} \]

\[ P_m = \eta \cdot P_e \text{ where } \eta < 1 \] (\eta \text{ is called efficiency})

Mechanical Power

\[ P_m = F \cdot r \cdot \omega \]
\[ = (\tau \text{ N.m}) \cdot (\omega \text{ rad/sec}) \]
\[ \text{in watts = J/s} \]