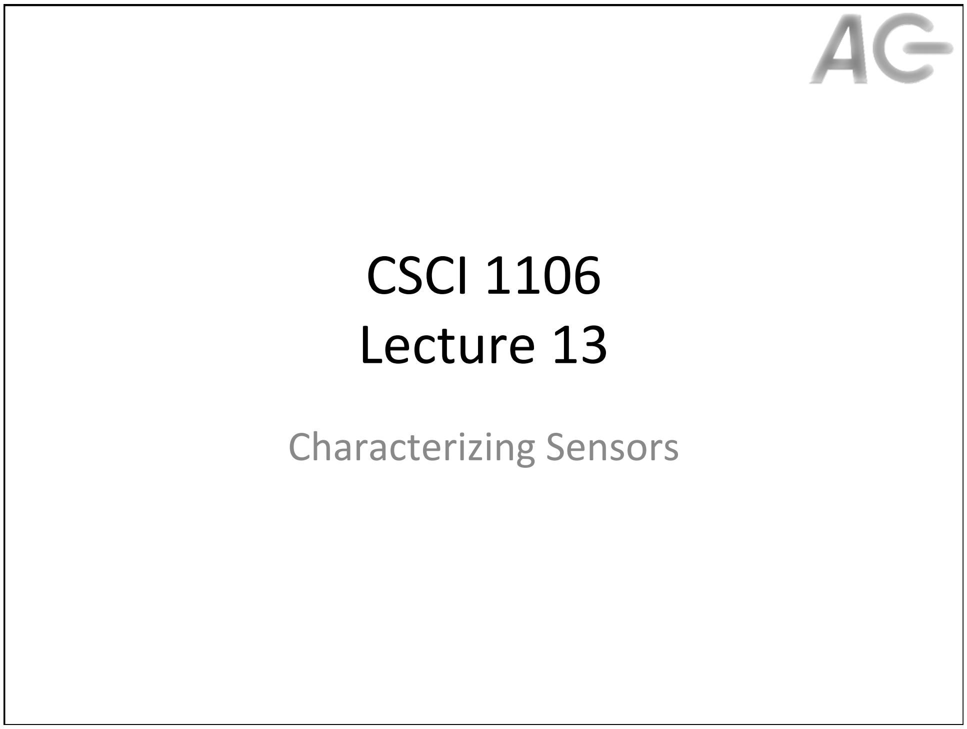
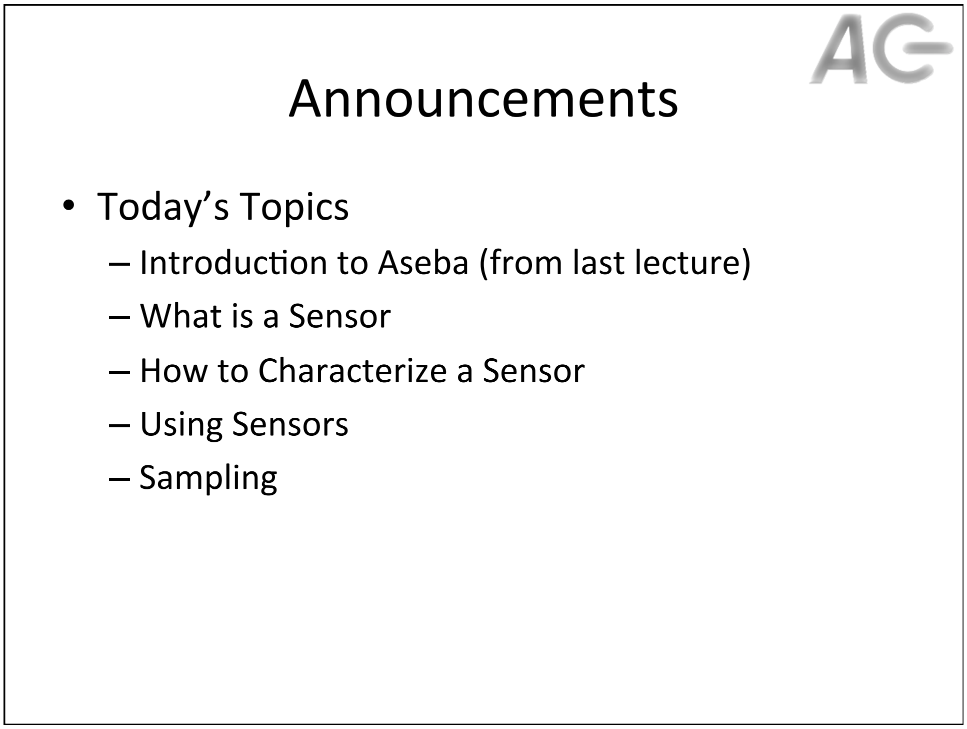


CSCI 1106 Lecture 13

Characterizing Sensors

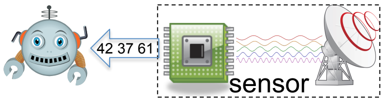


Announcements

- Today’s Topics
  - – Introduction to Aseba (from last lecture)
  - – What is a Sensor
  - – How to Characterize a Sensor
  - – Using Sensors
  - – Sampling

What is a Sensor?

- A *sensor* senses a property in its environment
  - Sound
  - Light
  - Acceleration
  - Temperature



- The input to the sensor is *analog* (*continuous*)
- A sensor converts the analog input to *digital* (*discrete*) values
- Sensors are described by a variety of characteristics

Sensor Characteristics

- *Sensitivity* : the minimum change of input that will result in change in output
- *Range* : the minimum and maximum inputs that a sensor can handle
- *Response Time* : how quickly the sensor can change state as a result of a change of input
- *Precision* : the degree of reproducibility of the measurement
- *Accuracy* : the maximum difference between the true and measured value
- *Bias* : the systemic error of the sensor
- *Variability* : the random deviation from the true value

A Simple Exercise

- For each the four objects (A, B, C, D)
  - – Estimate the object's length
  - – Aggregate results from multiple estimates
  - – Measure the object
  - – Compare the estimates to measured value
- What conclusions can we reach?

Sensors are Imperfect

- Sensors have two kinds of errors
  - – *Bias*: a systemic deviation from the true value
    - E.g., a clock that runs fast, or
    - A thermostat that thinks its warmer than it is.
  - – *Variability*: random deviation from the true value
    - E.g., static on the radio and
    - Flickering low-oil sensor
- Key Ideas:
  - – No matter how good a sensor is, it is imperfect
  - – Imperfect sensors introduce *uncertainty*
  - – Need to quantify the uncertainty
  - – Need to quantify a sensor's characteristics

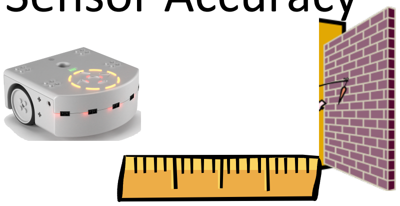
How to Characterize a Sensor

1. Identify the sensor we want to characterize
2. Identify the sensor characteristic we want to measure
3. Identify the possible variables of the characteristic
4. Fix all but one of the variables
5. Create a sequence of known ``actual'' inputs where the
   - One variable is varied and
   - All other variables are fixed
6. Perform a sequence of measurements (*multiple times)* on the inputs
7. Tabulate the results and compute means
8. Plot the results
9. Repeat steps 4 – 8, allowing a different variable to vary each time
10. Analyze the plot(s) to model the sensor

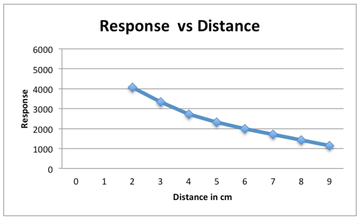
Questions:

1. How do we get the "measured" values?
2. How do we get the "actual" values?
3. How do we ensure all other variables are fixed?

Example: Proximity Sensor Accuracy



1. Sensor: Horizontal Proximity Sensor
2. Characteristic: Accuracy
3. Variables to consider:
   - Distance to target
   - Target size
   - Target material
   - Target shape
4. Fix all variables except "Distance to Target"
5. Create a sequence of known inputs
6. Perform a sequence of measurements for each input
7. Tabulate the results and compute means
8. Plot the results
9. Repeat steps 4 - 8
10. Analyze the plot to derive the sensor's characteristics

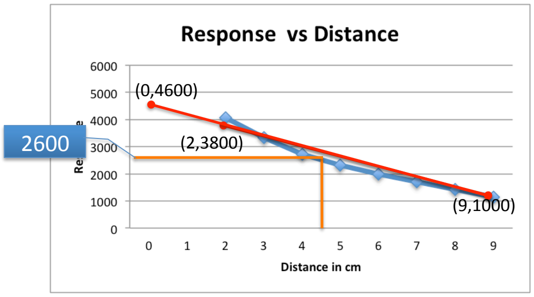


Inputl (cm)	2	3	4	5	6	7	8	9
Response 1	4051	3343	2735	2311	1973	1708	1421	1145
Response 2	4056	3340	2734	2320	1983	1697	1426	1152
Response 3	4062	3347	2721	2307	1981	1702	1408	1138
Average	**4056**	**3343**	**2730**	**2313**	**1979**	**1702**	**1418**	**1145**

Making Use of the Results

- General observation(s)
  - Response decreases as distance increases
  - Useful for visual interpolation
- Create a linear model
  - Draw a linear approximation
  - Compute slope (*m*) and intercept (*b*) of the line
  - Plug into equation of a line
- Then what?

$$m = \frac{rise}{run} = \frac{y_2 - y_1}{x_2 - x_1} = \frac{1000 - 3800}{9 - 2} \cong -400$$

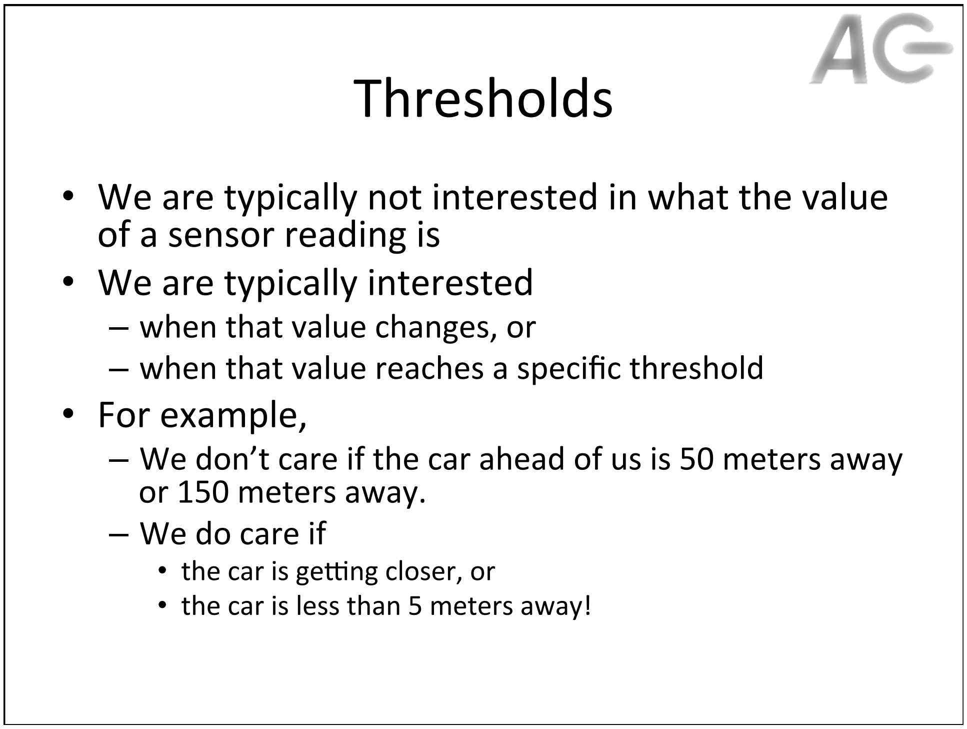


$$b = 4600$$

$$y = mx + b \Rightarrow y = -400x + 4600$$

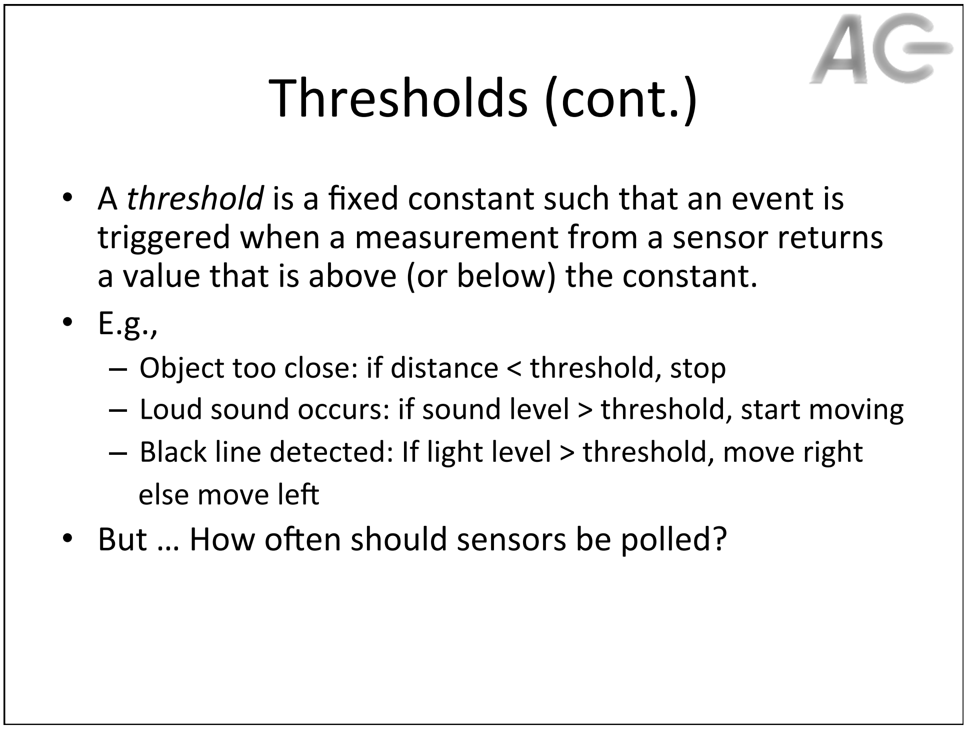
Using Sensors

- Perform sensor readings when events occur
  - Checking a sensor's reading is called *polling* the sensor
- It is the program's responsibility to *interpret* the sensor reading, i.e.,
  - Translate the value returned by the sensor into a meaningful decision
- A simple way to assign meaning is to use *thresholds*



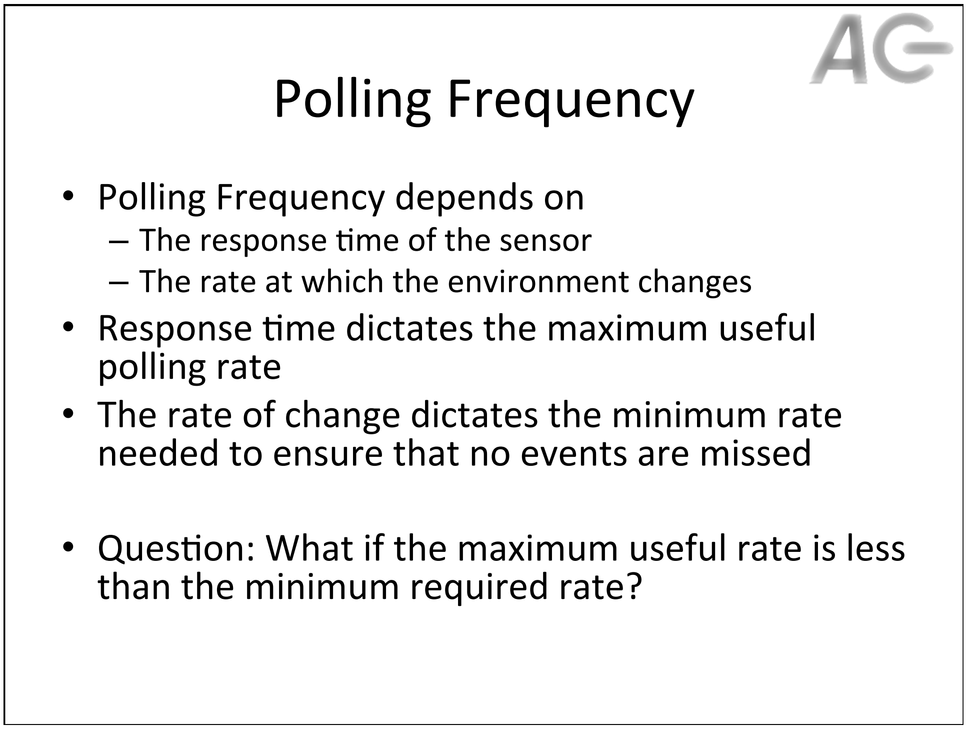
Thresholds

- We are typically not interested in what the value of a sensor reading is
- We are typically interested
  - when that value changes, or
  - when that value reaches a specific threshold
- For example,
  - We don’t care if the car ahead of us is 50 meters away or 150 meters away.
  - We do care if
    - the car is getting closer, or
    - the car is less than 5 meters away!



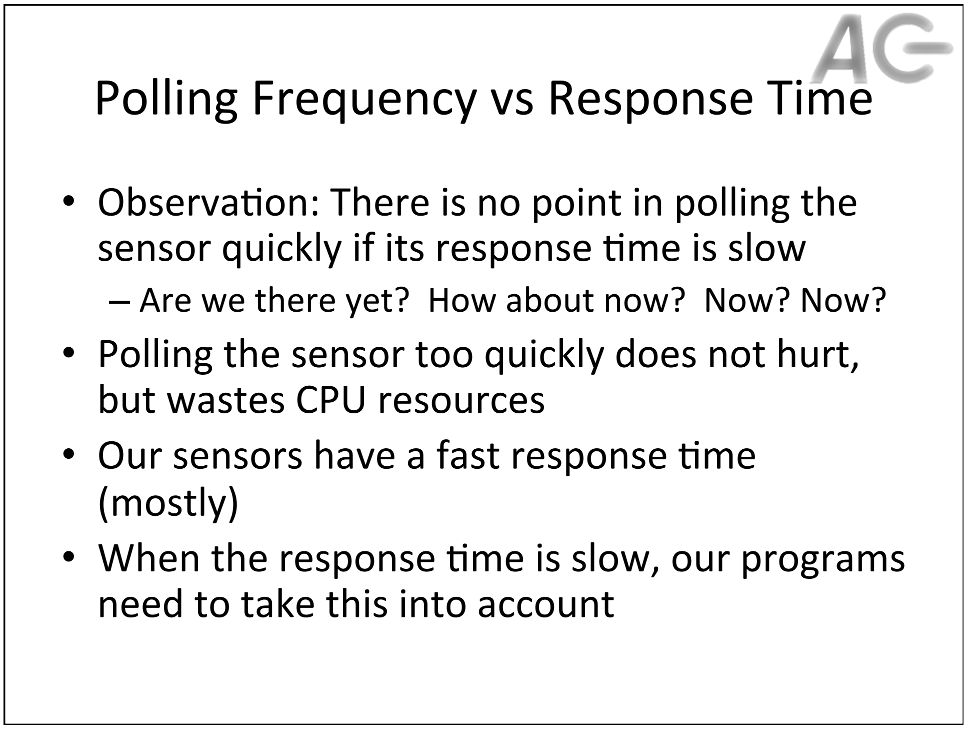
Thresholds (cont.)

- A *threshold* is a fixed constant such that an event is triggered when a measurement from a sensor returns a value that is above (or below) the constant.
- E.g.,
  - Object too close: if distance < threshold, stop
  - Loud sound occurs: if sound level > threshold, start moving
  - Black line detected: If light level > threshold, move right else move left
- But … How often should sensors be polled?



Polling Frequency

- Polling Frequency depends on
  - The response time of the sensor
  - The rate at which the environment changes
- Response time dictates the maximum useful polling rate
- The rate of change dictates the minimum rate needed to ensure that no events are missed
- Question: What if the maximum useful rate is less than the minimum required rate?

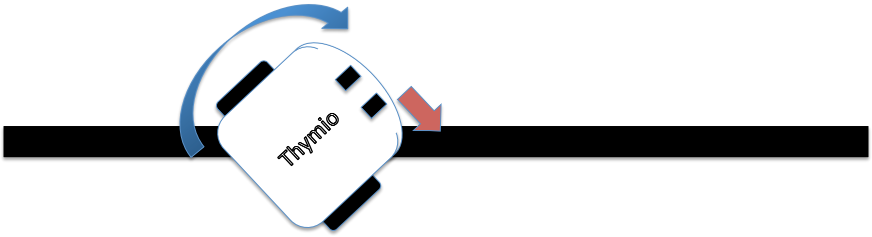


Polling Frequency vs Response Time

- Observation: There is no point in polling the sensor quickly if its response time is slow
  - Are we there yet? How about now? Now? Now?
- Polling the sensor too quickly does not hurt, but wastes CPU resources
- Our sensors have a fast response time (mostly)
- When the response time is slow, our programs need to take this into account

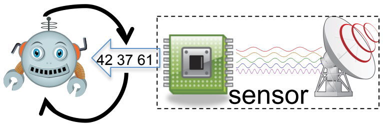
When Response Time Matters

- In Follow-The-Line
  - The angular velocity of the light sensor is quite fast
  - This could cause the sensor to move over the black line too quickly to pick it up
  - This would result in the robot losing the line
- How do we ensure that the robot does not miss the line?

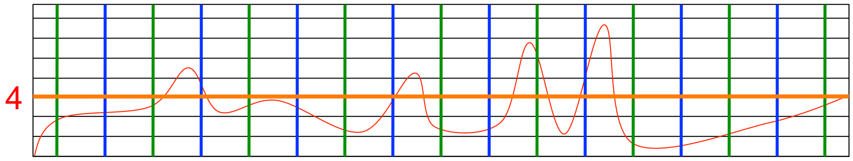


Sampling

- Sensors must be polled (sampled) for values
- The *sampling rate* is the frequency of the polls
- A higher rate means we are
  - Less likely to miss a change in inputs
  - Using more CPU time to poll the sensor
- If the rate is too high, there is no time to do anything else



Example: When is the Signal above 4?



S1 2 2.5 2.5 1.25 1.25 $\boxed{6}$ 0.5 1 2.5

S2 2.25 $\boxed{5}$ 2.5 $\boxed{4}$ 1.25 $\boxed{5}$ 0.5 2

S* 2 2.25 2.5 $\boxed{5}$ 2.5 2.5 1.25 $\boxed{4}$ 1.25 1.25 $\boxed{6}$ $\boxed{5}$ 0.5 0.5 1 2 2.5

What If We Do Detect a Change?

- Suppose the sensor returns a different value.
- Does this mean that the environment has changed?
- Are you sure?