CSCI 1106
Lecture 14

Using Sensors and Actuators

Motion model

Announcements

- Today's Topics
  - Using Sensors
  - Sampling
  - Debouncing
  - Kinematics: Motion model

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

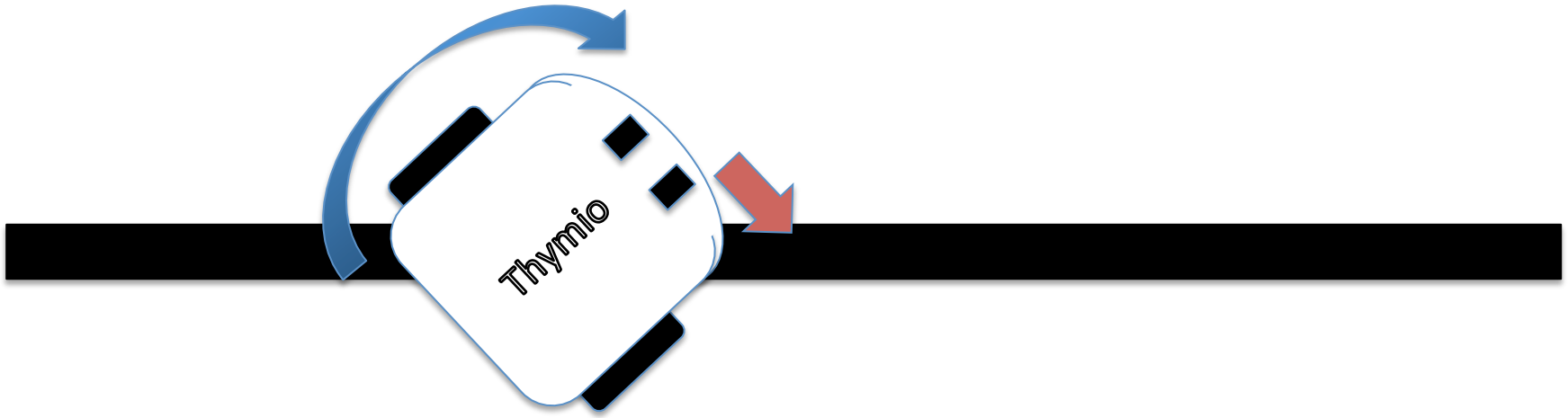
- Polling Frequency depends on
  - The response time of the sensor
  - The rate at which the environment changes
- Response time dictates the maximum useful polling rate
- The rate of change dictates the minimum rate needed to ensure that no events are missed

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

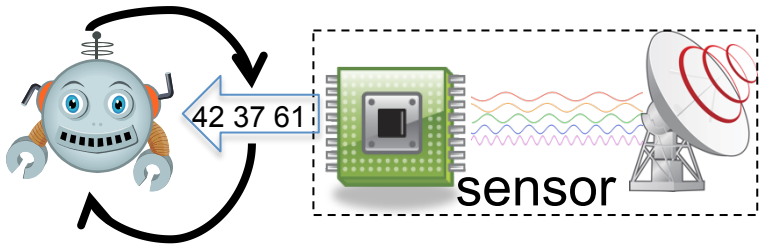
When Response Time Matters

- 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?

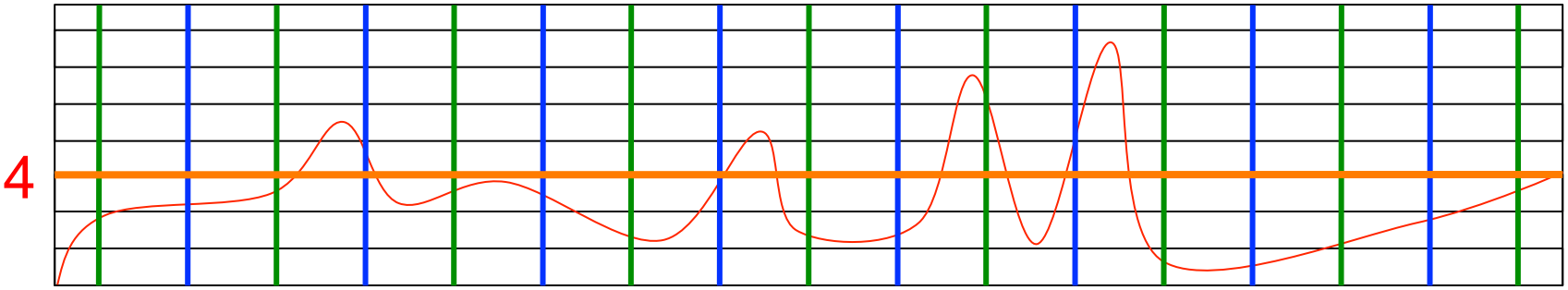


Sampling

- 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?

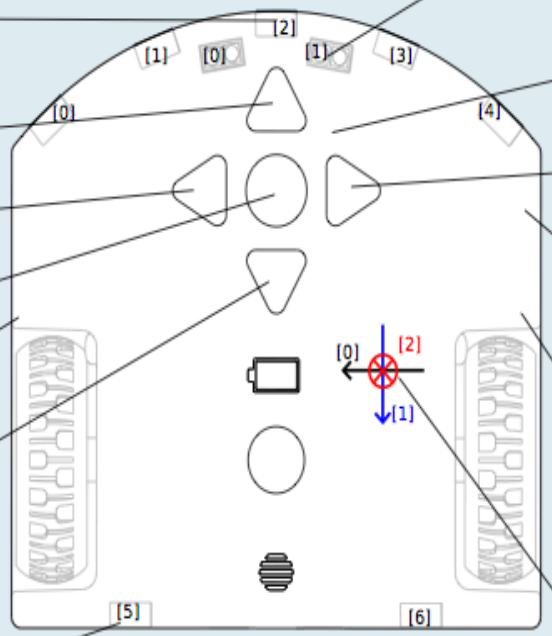


S1 2 2.5 2.5 1.25 1.25 6 0.5 1 2.5

S2 2.25 5 2.5 4 1.25 5 0.5 2

S* 2 2.25 2.5 5 2.5 2.5 1.25 4 1.25 1.25 6 5 0.5 0.5 1 2 2.5

Sensors



prox.horizontal[0-4] {0...~4300}
prox 10 Hz

button.forward {0,1}
button.forward pressed or released

button.left {0,1}
button.left pressed or released

button.center {0,1}
button.center pressed or released

temperature [1/10 °C]
temperature 1 Hz

button.backward {0,1}
button.backward pressed or released

prox.horizontal[5-6] {0...~4300}
prox 10 Hz

prox.ground.delta[0-1] =reflected-ambiant
prox.ground.reflected[0-1] {0...1023}
prox.ground.ambient[0-1] {0...1023}
prox 10 Hz

buttons 20 Hz

button.right {0,1}
button.right pressed or released

rc5.address
rc5.command
rc5 signal received

mic.threshold {0..255}
mic.intensity {0..255}
mic mic.intensity>mic.threshold
sound.record(N) N: {0...32767}, record as 'rN.wav'
N=-1, stop recording

acc[0-2] {-32...32}, 23=1g
acc 16 Hz
tap shock detected

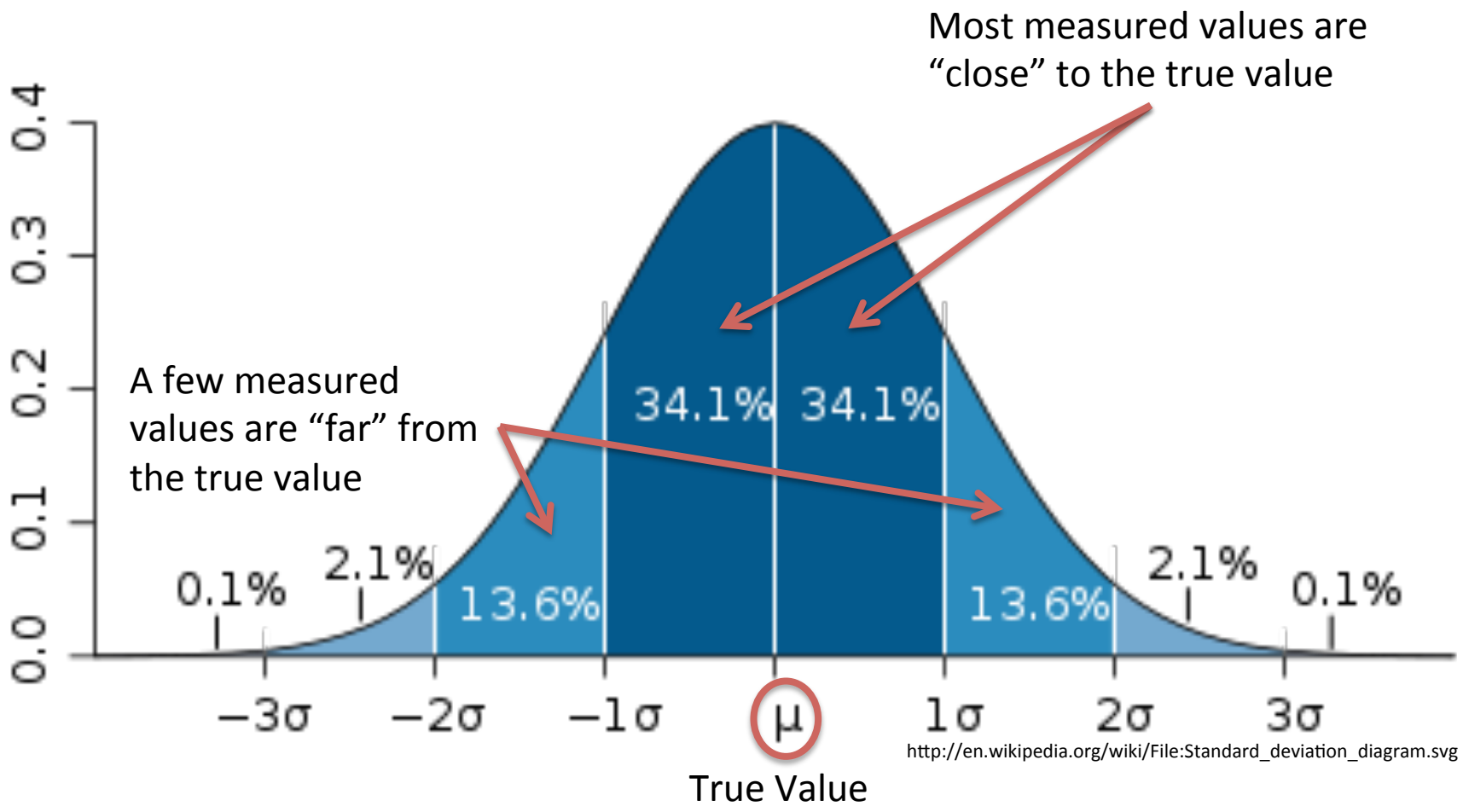
What If We Do Detect a Change?

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

Variability of Sensors

- Problem: All sensors have some variability
  - The measured value randomly deviates from the true value
- A sensor may report different values for the same true value
- The reported values will be *distributed* around the true value
  - We assume the bias is 0

Normal Distribution



Dealing with Variability

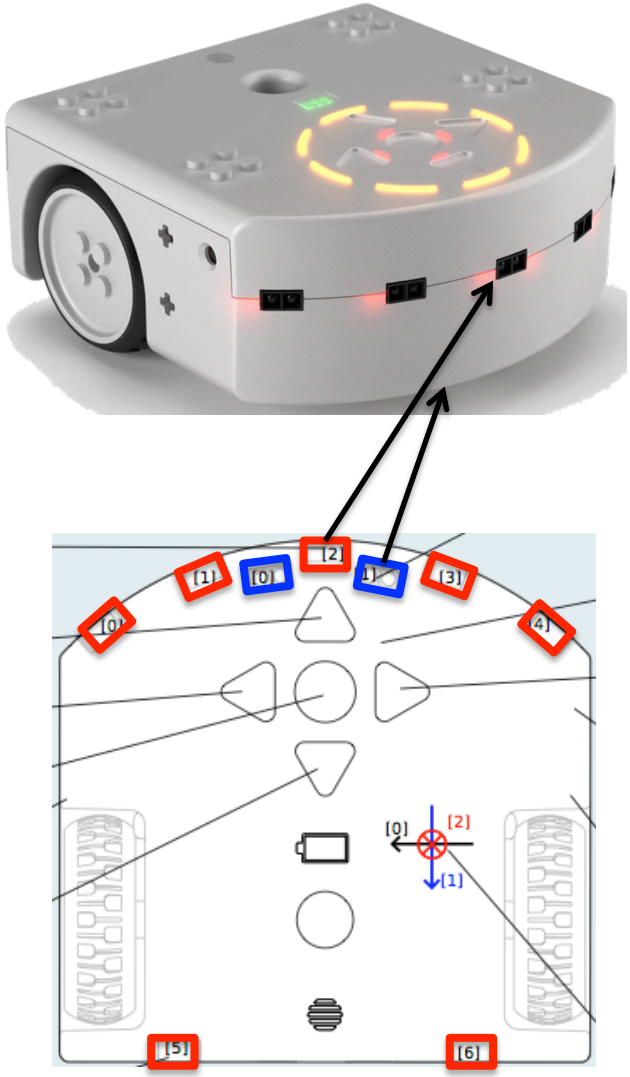
- Observations:
  - A single measurement may not report the true value or even close to the true value
  - Most reported values are “close” to the true value (assuming no bias)
  - Reported values are distributed around the true value
- Conclusions:
  - Take multiple measurements
  - Average the results

Sensor Debouncing

- Key Idea: Need to filter the data from a sensor
- Approach:
  - Take a number of samples (polls)
    - More is better
  - Combine samples for a more precise measurement
    - Average (mean)
    - Median
    - Mode
- Trade-off:
  - Get a more precise measurement
  - Costs more time to perform

Proximity (prox) Sensors

- 7 horizontal proximity sensors
  - Measures distance to objects using infra-red light
  - 5 in front and 2 in rear
  - Range: 0 (nothing) to 4000+ (object very close)
  - Values stored in `prox.horizontal[0:6]`
- 2 ground proximity sensors
  - Measures light from the ground
    - *ambient* (surrounding light)
    - *reflected* (infra-red light emitted by sensor
    - *delta* (difference between ambient and reflected)
  - 2 in front
  - Response ranges: 0 (no light) to 1023 (full light)
  - Values stored in array
    - `prox.ground.ambient[0:1]`
    - `prox.ground.reflected[0:1]`
    - `prox.ground.delta[0:1]`



Actuators

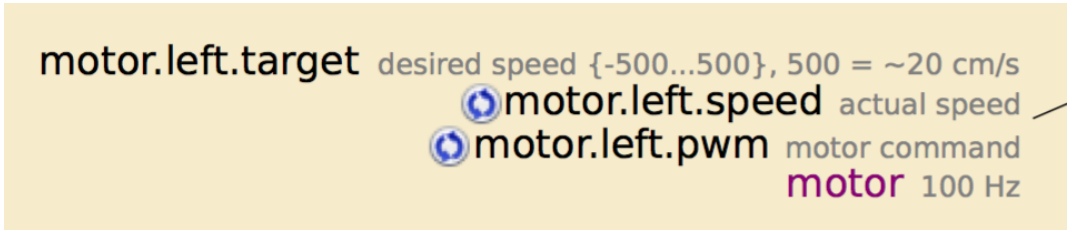
- Actuators allow the robot to affect the world
- Actuators include:
  - Motors
  - Solenoids
  - Hydraulic mechanisms
  - Lasers
- Actuators are characterized by their parameters and tolerances:
  - Torque, force, and pressure
  - Speed, power, and strength
  - Accuracy and precision
- Parameters are fed into actuators as digital values
- Actuators use the parameters to control the behaviour, e.g.,
  - Turn on the left motor at speed 200

Using Actuators

- Actuators are typically used in two ways:
  - Synchronous use
    - E.g., make 1 rotation
  - Asynchronous use
    - E.g., start rotating
- Synchronous use:
  - Start operation
  - Wait until the operation completes
  - Continue program
- Asynchronous use:
  - Start operation
  - Continue program
  - Use sensors or poll actuator to determine operation completion

- Question: Are Thymio's motors used synchronously or asynchronously?

Thymio's Motors



- Motors create rotation
- Thymio's motors have one parameter:
  - *target*: desired speed of the motor
    - -500 ... 500
    - Positive means forward
    - Negative means reverse
    - 0 means stop
    - A speed of 500 is (approximately) 20cm/s
- Thymio's motors have following sensors
  - *speed*: current speed sensed by the motor
  - *pwm*: commands sent to the motor
- The Trade-off
  - Faster speed implies more forward momentum
  - More forward momentum implies less accuracy and precision
  - The slower the robot moves, the more precise its behaviour

Kinematics: Motion model

A motion model is a mathematical description of the motion in response to a motor command, e.g.

future location = function(current location, motor command)

For example, consider a robot that we can tell to drive with velocity v [m/s] for a time t [s] along a line:

$$x(t) = x(0) + v * t$$

Differential drive robot

- Two motors that can operate independently
- The pose of a robot describes the state of the robot l=(x,y,θ)