Lego NXT

Acoustic Tape Measure Application

Student:	Jade Cheng
Project:	M.S. Capstone in Computer Sciences
Date:	April 14, 2011
Supervisor:	Dr. Edoardo S. Biagioni, PhD

Introduction

One Laptop per Child (OLPC)

OLPC is a project aiming to create educational opportunities for children.

OLPC provides children rugged, low-cost, low-power, connected laptops.

XO Laptop

OLPC's focus is on the development, construction, and deployment of XO laptops.

XO laptops are regarded as a main factor that started the trend of Netbooks.

XO Laptop

Introduction

XO Activities

XOs run the Sugar Learning Platform, and their applications are called Sugar Activities.

XOs have about 160 Activities available and 14 of them are pre-installed.

Sugar Activities are diverse: education, social networking, and entertainment.

Activity Name	***Activity Description***
BROWSE	Web browser
DISTANCE	Measure distance between XOs
PIPPY	Python programming
RECORD	Video and audio capture
TERMINAL	Activity version of the Sugar terminal
WRITE	Word processor

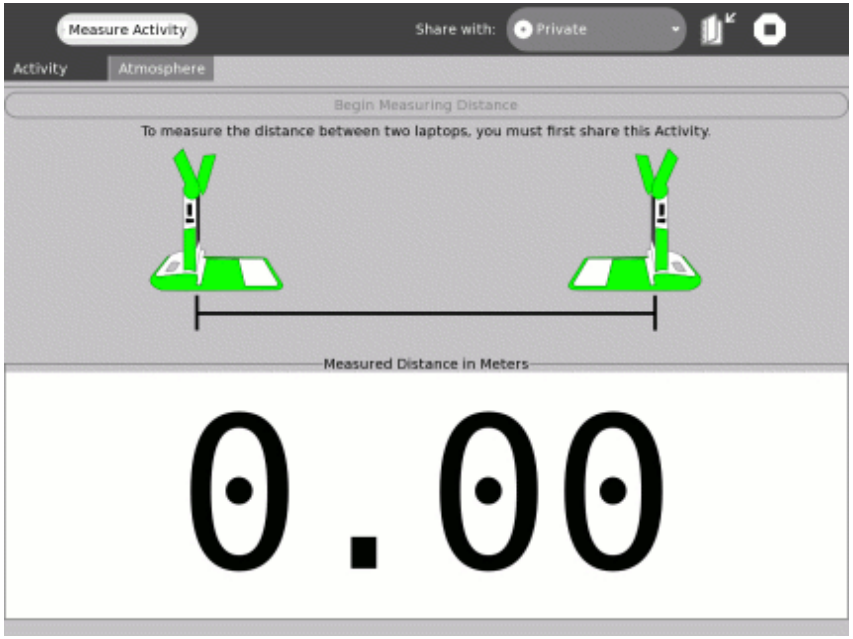
Sugar Activity Examples

Introduction

Acoustic Tape Measure (Distance) Activity

The Distance Activity is a pre-installed Sugar Activity.

The Distance Activity measures physical distances using sound pulses.



Distance Activity User Interface

Introduction

Distance Activity in Action

Both XOs must be connected to the same shared network.

When the activity is started, it sends invitations to all neighboring XOs.

Once a second XO accepts the invitation, both XOs can start measuring distances.

Distance Activity with Incorrect Placement (left) and Correct Placement (right)

Videos of the XO Distance Activity in Action

http://www2.hawaii.edu/~yucheng/projects/acoustic-tape-measure/#program-in-action

Introduction

Lego Mindstorms

Lego Mindstorms is a line of programmable robotics and construction toys.

Lego Mindstorms contains many components including cables, sensors, and the NXT.

This project used Lego Mindstorms NXT 2.0, which was released in 2009.

Lego NXT with Sensors and Motors

Introduction

Lego NXT (Brick)

The NXT is the main component of Lego Mindstorms.

The NXT controls as many as four sensors and three motors via RJ12 cables.

The NXT has a 100×64-pixel monochrome LCD display and four buttons.

The NXT's stock firmware provides a user interface of hierarchical menus.

The NXT has a speaker and can play sound files at sampling rates up to 8 kHz.

NXT Brick		***NXT Sensors***	
CPU	Pentium II	Touch	0 or 1
RAM	32 MB	Ultrasonic	0 ~ 170 (cm)
HDD	115 MB	Light	Intensity 0 ~ 100
		Color	RGB 0 ~ 255
		Sound	Intensity 0 ~ 100
		Rotation	0 ~ 360 (degree)

Lego NXT specs

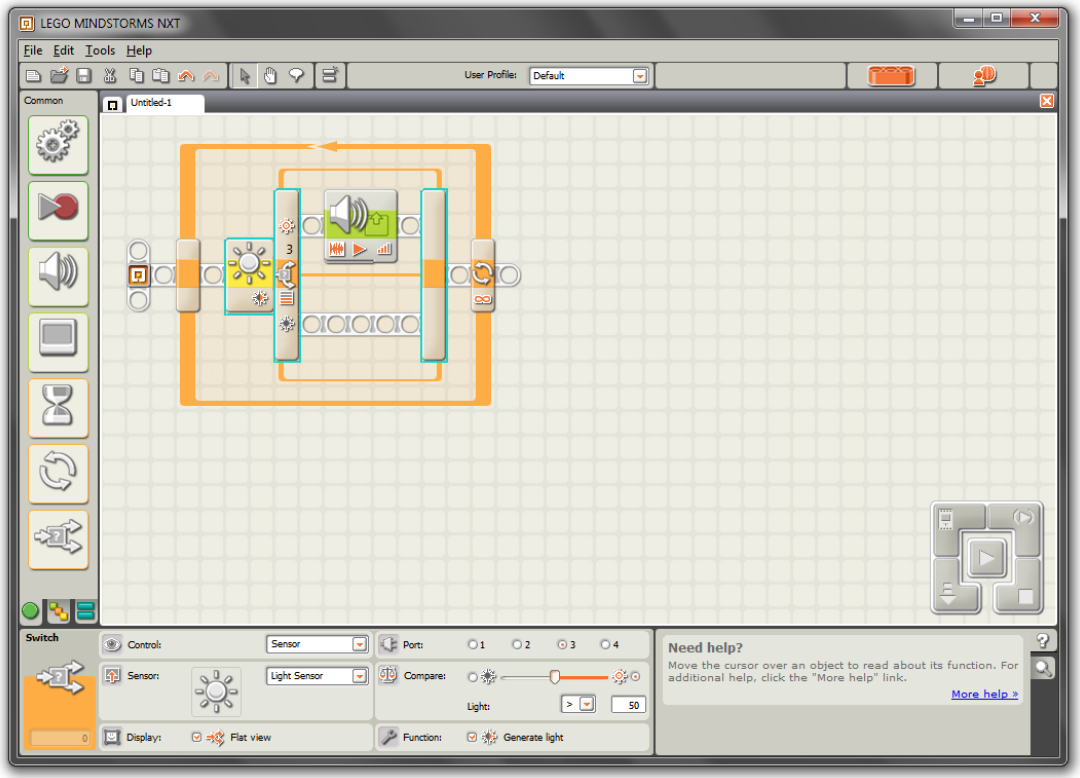
Introduction

NXT-G

NXT-G is a programming package distributed with Lego Mindstorms.

NXT-G features an interactive drag-and-drop environment.

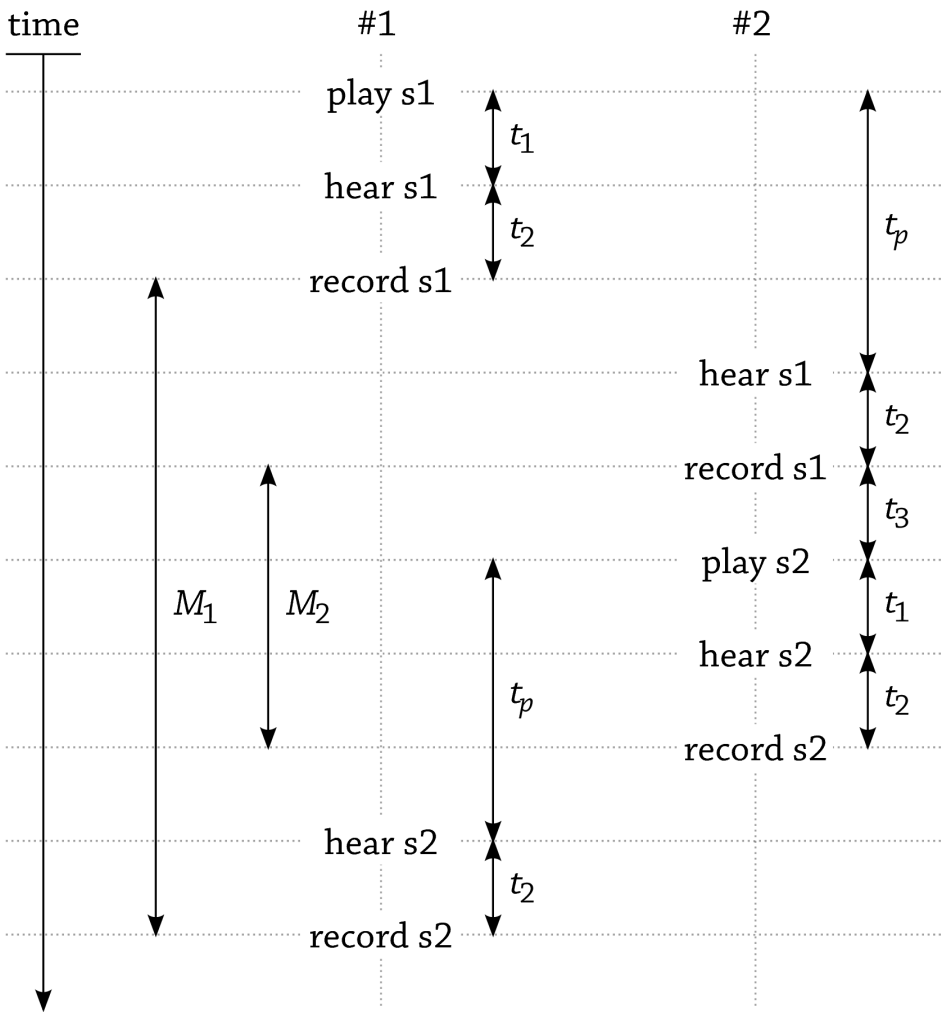
NXT-G is suitable only for simple programs.



NXT-G Example Program

Related Work – Algorithm

Distance Algorithm Overview



Visual Explanation of the Distance Algorithm

Example:

$$t_1 = 5 \text{ ns}$$

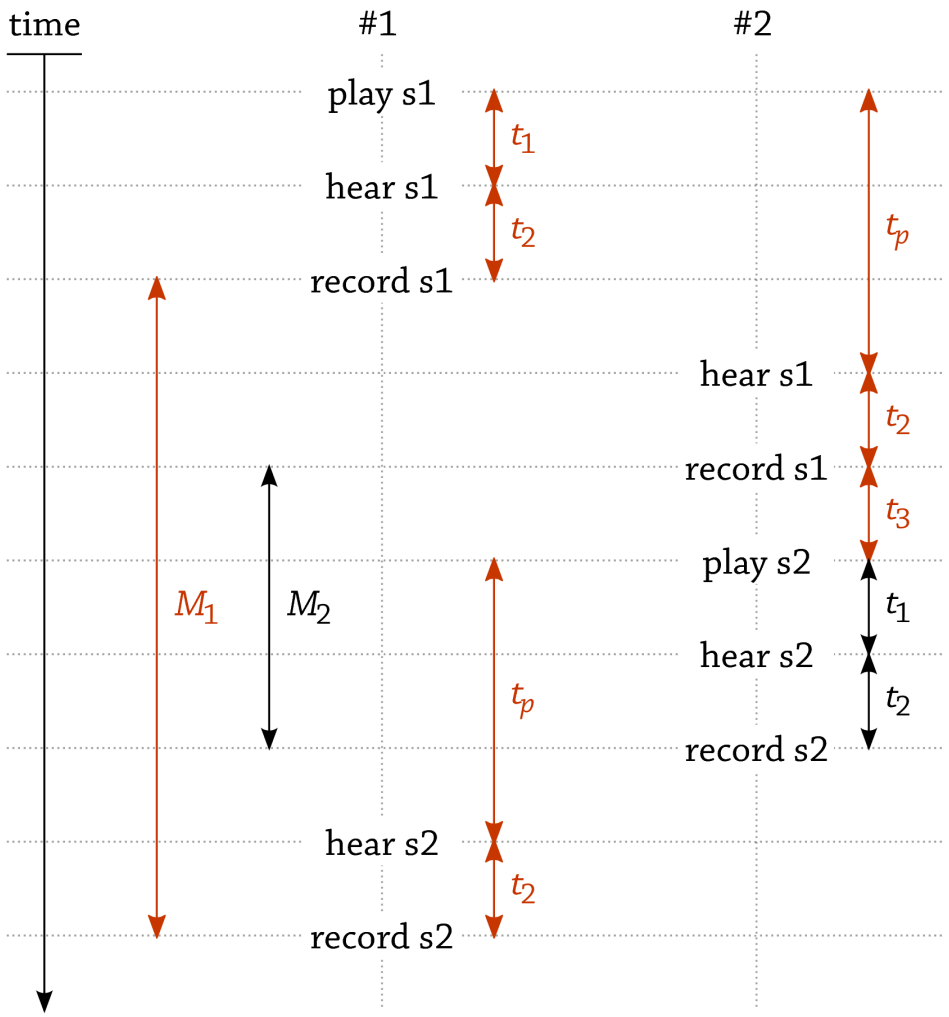
$$t_2 = 100 \text{ ns}$$

$$t_3 = 2 \times 10^9 \text{ ns}$$

$$t_p = 1 \times 10^6 \text{ ns}.$$

Related Work – Algorithm

Distance Algorithm Details



Timelines Relevant to M_1

$$M_1 = t_p - (t_1 + t_2) + t_2 + t_3 + t_p + t_2$$

$$= 2 \cdot t_p + t_3 + t_2 - t_1.$$

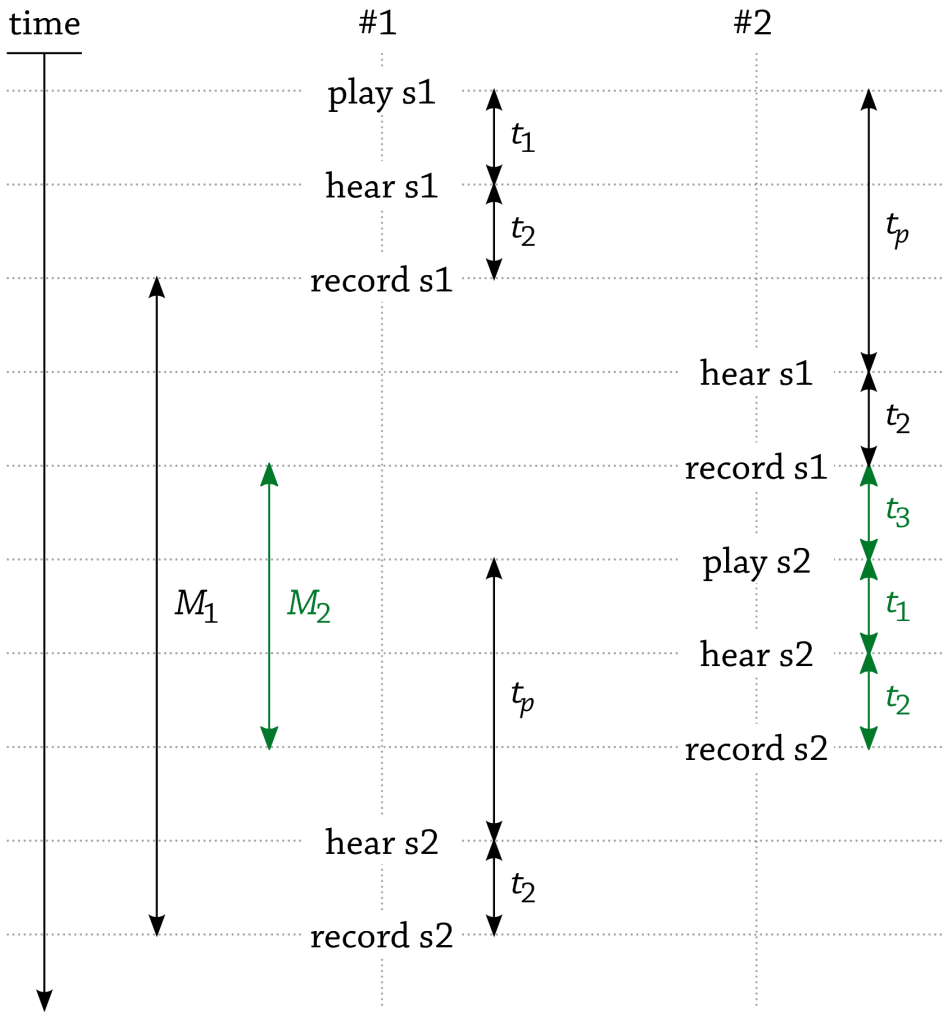
Example:

$$M_1 = 2 \cdot t_p + t_3 + t_2 - t_1$$

$$= 2 \times 10^6 + 2 \times 10^9 + 100 - 5.$$

Related Work – Algorithm

Distance Algorithm Details



Timelines Relevant to M_2

$$M_2 = t_3 + t_1 + t_2.$$

Example:

$$M_2 = t_3 + t_1 + t_2$$

$$= 2 \times 10^9 + 5 + 100.$$

Related Work – Algorithm

Distance Algorithm Details

$$M_1 = 2 \cdot t_p + t_3 + t_2 - t_1$$

$$M_2 = t_3 + t_1 + t_2$$

$$\Rightarrow M_1 - M_2 = 2 \cdot t_p - 2 \cdot t_1$$

$$d = t_p \cdot c_{\text{sound}}$$

$$\Rightarrow d = \frac{M_1 - M_2 + 2 \cdot t_1}{2} \cdot c_{\text{sound}}$$

$$t_1 \approx 0$$

$$\Rightarrow d \approx \frac{M_1 - M_2}{2} \cdot c_{\text{sound}}.$$

Example:

$$M_1 - M_2 = 2 \times 10^6 - 10$$

$$c_{\text{sound}} \approx 1.366 \cdot 10^{-5} \text{ in} / \text{ns}$$

$$d \approx \frac{M_1 - M_2}{2} \cdot c_{\text{sound}}$$

$$\approx 1 \times 10^6 \times 1.366 \cdot 10^{-5}$$

$$\approx 13.66 \text{ in.}$$

Related Work – Algorithm

Distance Algorithm Assumptions

To apply this algorithm, three environmental and hardware conditions need to hold.

$$c_{\text{sound at computer \#1}} = c_{\text{sound at computer \#2}}$$

$$t_1 \approx 0$$

$$t_{2 \text{ at computer \#1}} = t_{2 \text{ at computer \#2}}$$

We make three assumptions, respectively.

1. The environmental conditions are the same at both computers.

   e.g., temperature, air pressure, and humidity.

2. The sound sensor of a computer is sufficiently close to its speaker.

   i.e., sound travels instantaneously from the speaker to the sensor on one computer.

3. Two computers have identical hardware and instruction sets to record a sound.

   i.e., the durations between hearing and recording a sound are identical.

Related Work – Algorithm

Distance Algorithm Conclusion

$$d \approx \frac{M_1 - M_2}{2} \cdot c_{\text{sound}}.$$

Symbol	***Description***
M_1	The interval between recording s1 and s2 on computer #1.
M_2	The interval between recording s1 and s2 on computer #2.
t_p	The propagation delay due to the distance of the two computers.
t_1	The interval between playing a sound and hearing the sound on the same computer.
t_2	The interval between hearing a sound and recording the sound.
t_3	The interval between recording a sound and playing a sound on computer #2.

Related Work – leJOS

NXT Java Setup – leJOS

leJOS is a high-level, open-source language based on Java.

leJOS applications execute on the NXT using a custom firmware.

leJOS's firmware provides a subset of the typical JRE.

NXT Java Setup – Embedded Programming Environment

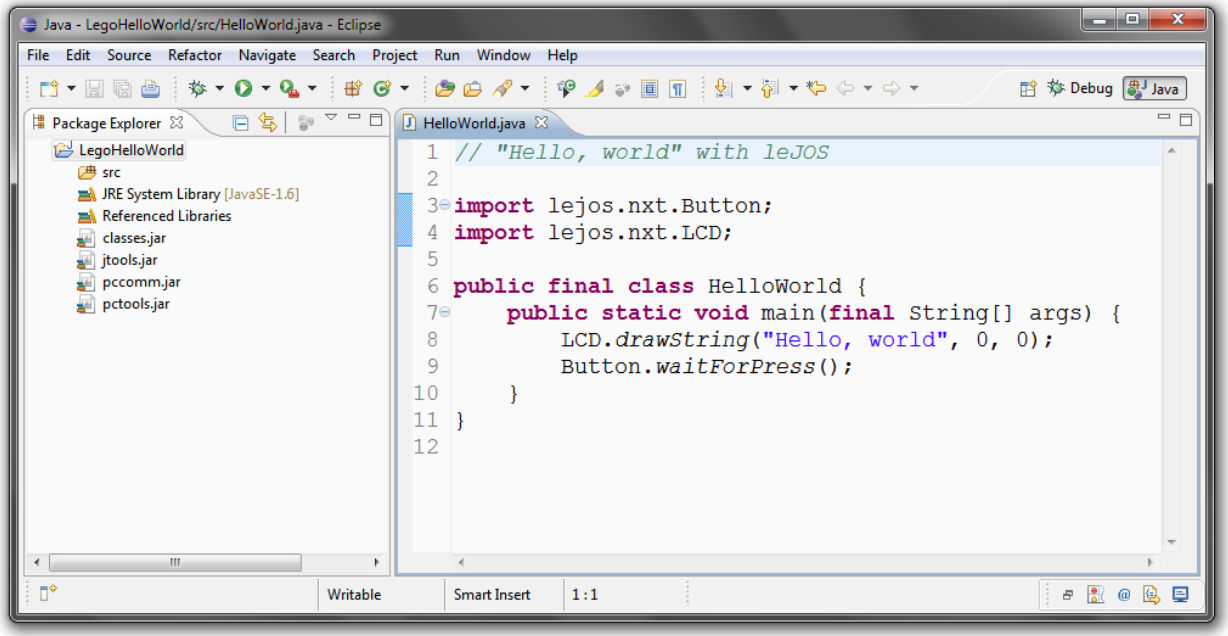
The Eclipse IDE was used for this project.

An external compiling tool was used to compile the code into a leJOS application.

An external transferring tool was used to upload the program to the NXT brick.

Related Work – leJOS

NXT Java Setup – Hello World



leJOS Hello World Program

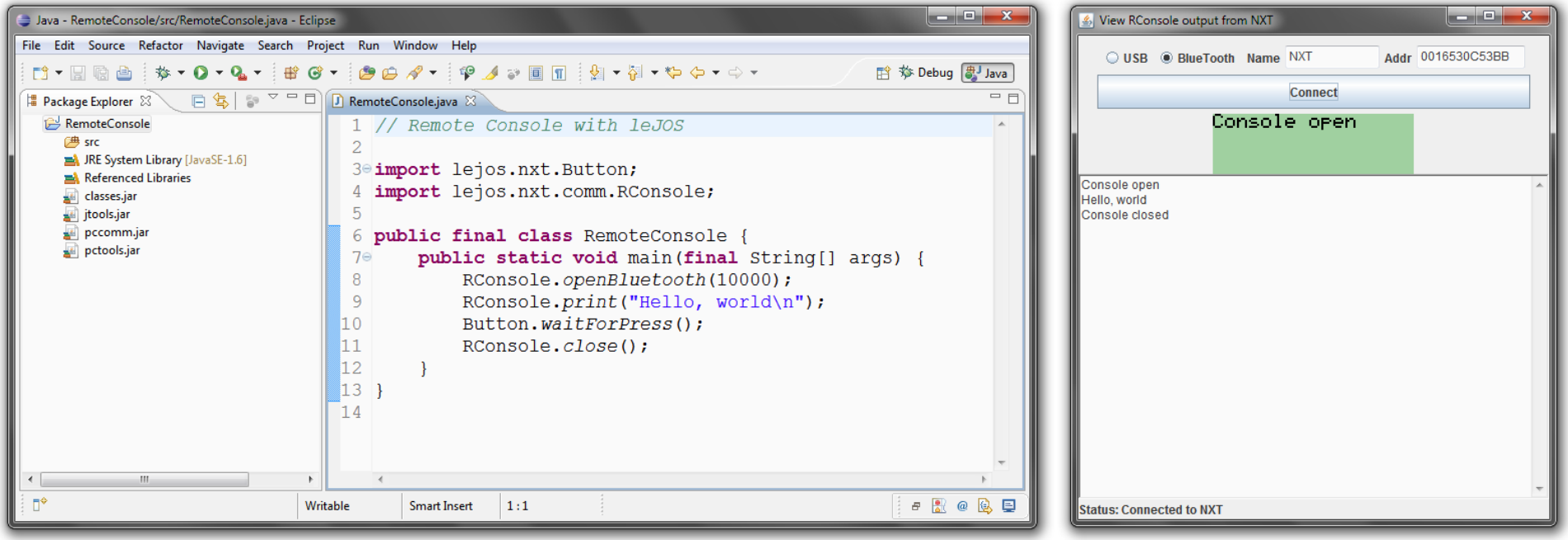
Related Work – leJOS

NXT Java Setup – Runtime Diagnostics

The NXT LCD is inadequate for diagnosing and debugging high-speed operations.

leJOS provides a remote console module.

The remote console requires a connection, either USB or Bluetooth.



leJOS Remote Console

Related Work – NXT Sound System

NXT Sound System

Proper implementation of the Distance algorithm requires some basic functionality.

Component	***Requirement***
Speakers	To play sounds louder than ambient noise
Sound Sensors	To detect sounds from an NXT speaker
System	To record and play sounds simultaneously

Basic Hardware Requirement for the Distance Algorithm

Related Work – NXT Sound System

NXT Sound System

To use the algorithm with three or more participants, such as in a robotic soccer application, and to make measurements simultaneously, the NXT would need two additional capabilities.

Component	***Requirement***
Speakers	To play distinctive sounds louder than ambient noise
Sound Sensors	To detect distinctive sounds from NXT speakers

Hardware Requirement for Multiple Participants Measuring Simultaneously

Alternatively, the participants could follow a pre-defined protocol and take turns emitting sounds.

Related Work – NXT Sound System

NXT Sound System – Speakers

The Lego NXT speakers are capable of playing sounds

at a specified frequency (Hz)

at a specified volume (% of the system sound)

for a specified duration (ms)

Speaker Experiment 1

We tested the NXT's ability to play different frequencies at a fixed volume.

Speaker Experiment 2

We tested the NXT's ability to play different volumes at a fixed frequency.

Audio Results for Experiment 1 and 2

`http://www2.hawaii.edu/~yucheng/projects/nxt-sound/#speakers`

Related Work – NXT Sound System

NXT Sound System – Sound Sensor

The Lego NXT sound sensor detects sounds in two modes.

Decibels (DB) mode

Adjusted Decibel (DBA) mode

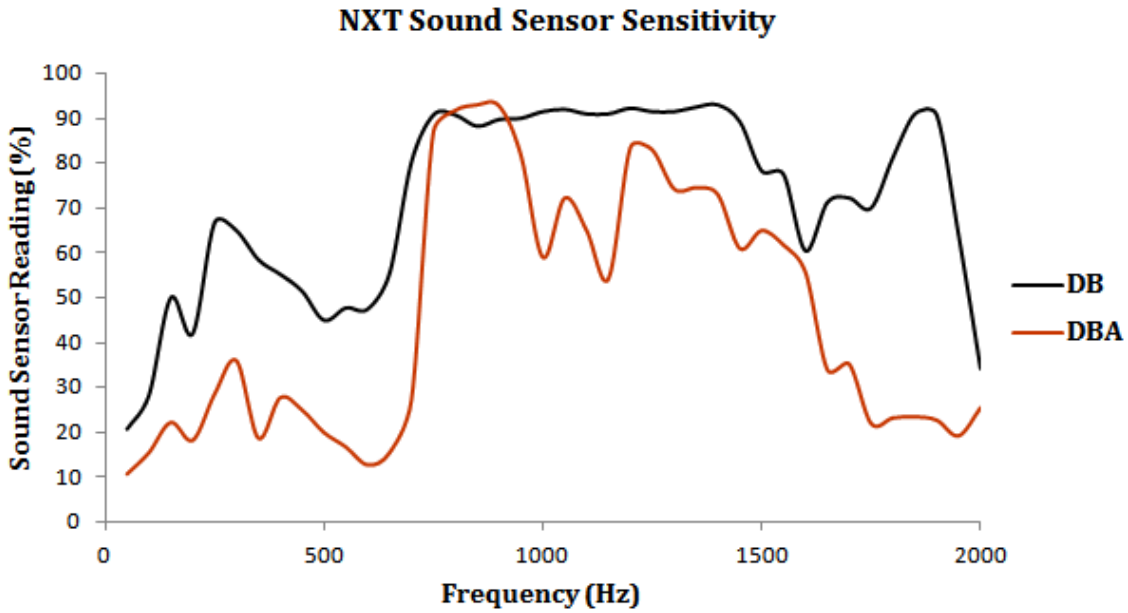
NXT Sound Sensor

Related Work – NXT Sound System

NXT Sound System – Frequency and Mode Experiment

We programmed one NXT as a sound generator that played tones at varying frequencies.

We programmed the other NXT as a sound receiver that recorded sounds.



The NXT sound sensor is able to distinguish sounds.

The largest, most stable range of distinguishable frequencies were in DB mode.

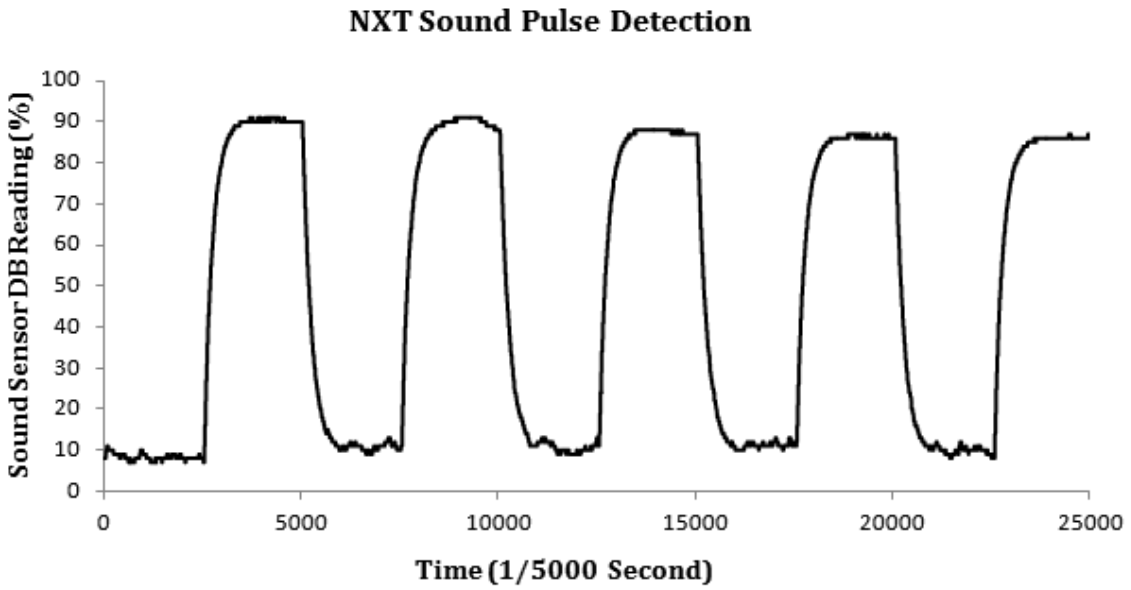
The largest, most stable range of distinguishable frequencies were 800 ~ 1400 Hz.

Related Work – NXT Sound System

NXT Sound System – Sound Pulse Detection Experiment

We programmed one NXT as a sound generator that played a fixed frequency, 1100 Hz.

We programmed the other NXT as a sound receiver that recorded sounds in real time.



The NXT sound sensor does not record sound as frequencies (oscillations).

The NXT sound sensor can capture approximately 5000 samples per second.

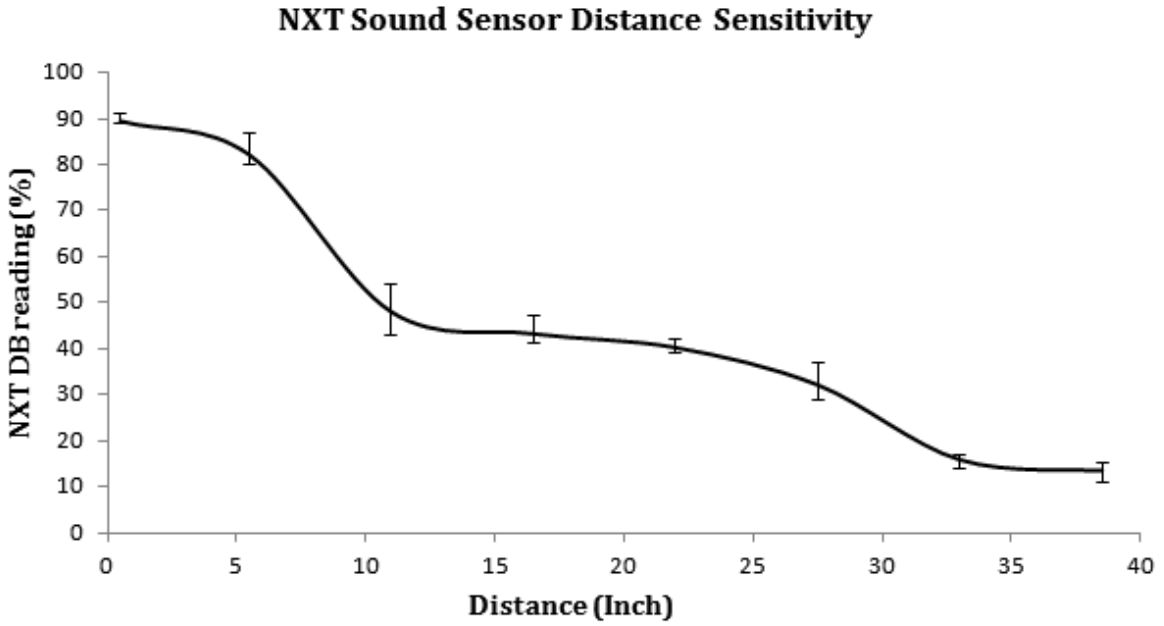
The NXT sound sensor can detect pulses when they are louder than ambient noise.

Related Work – NXT Sound System

NXT Sound System – Distance Sensitivity Experiment

We programmed one NXT as a sound generator that played a fixed frequency, 1100 Hz.

We programmed the other NXT as a sound receiver that recorded sounds.



The NXT sound sensor can distinguish sounds from a nearby NXT (~ 30 inches away).

This range is considerably less than the sound detecting capability range of the sensor.

This greatly reduces the usefulness of implementing the Distance algorithm on the NXT.

Related Work – Implementation

State Machine Design

Two state machines were designed and implemented.

The client unit initiates a connection and plays the first tone.

The server unit responds to the client's invitation and plays the second tone.

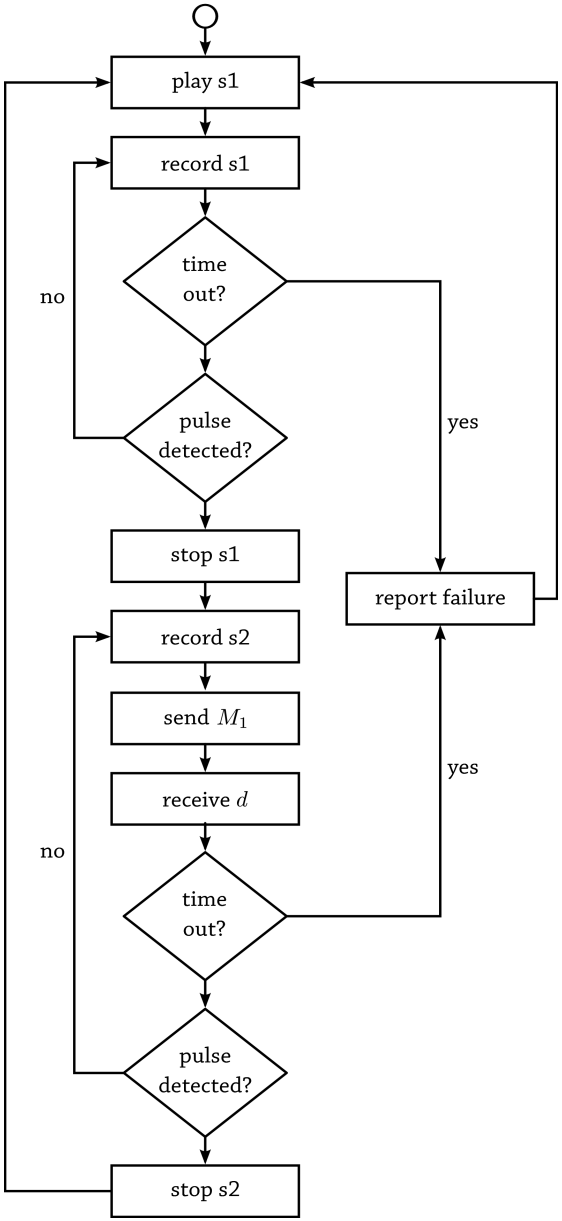
The application starts by allowing the user to specify the NXT's mode of operation.

The NXTs execute their state machines and continuously display synchronized readings.

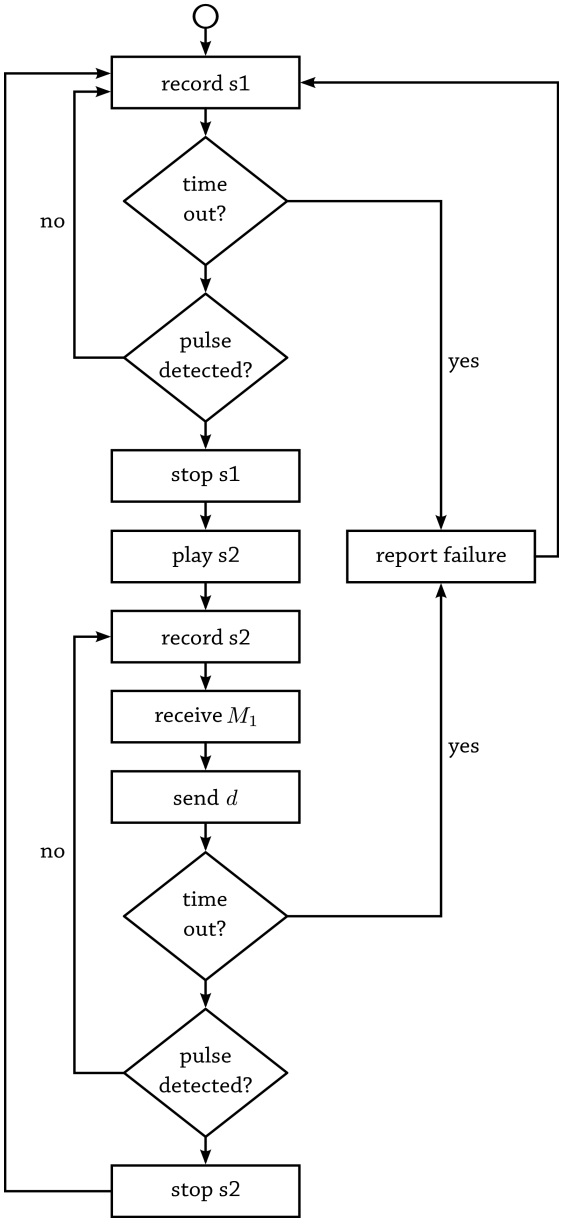
Implementation

`http://www2.hawaii.edu/~yucheng/projects/nxt-distance-implementation/source`

Related Work – Implementation



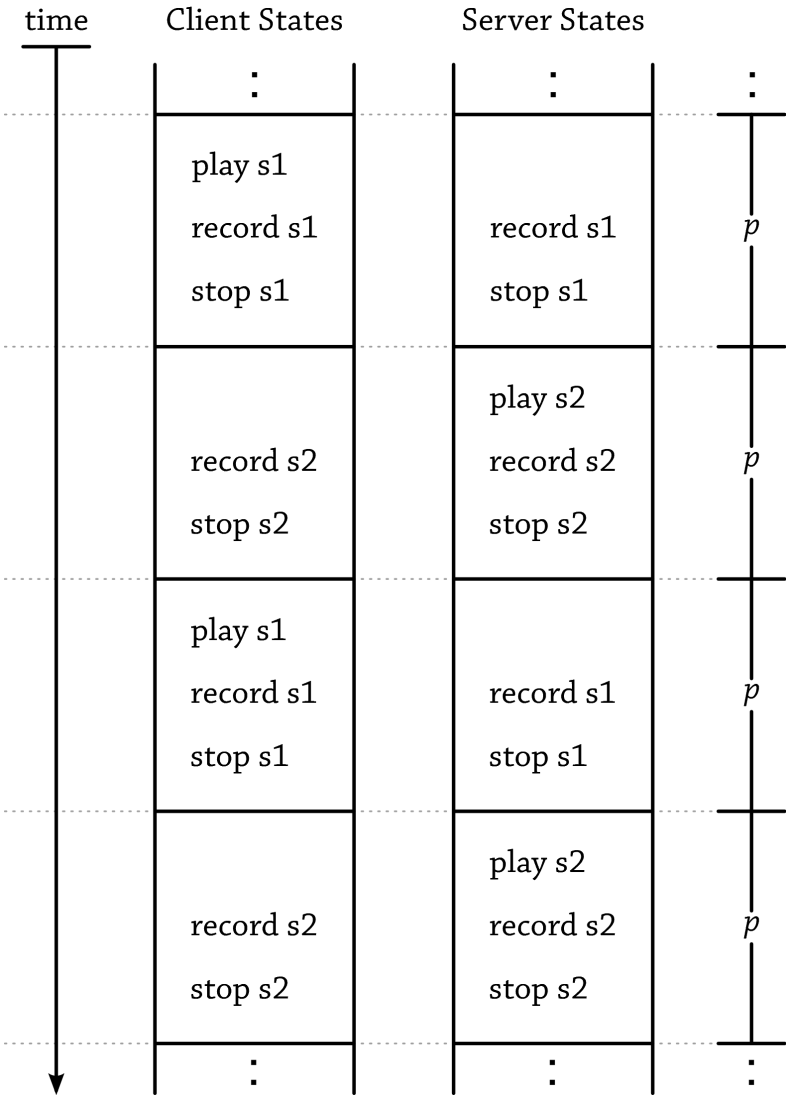
Client State Machine



Server State Machine

Related Work – Implementation

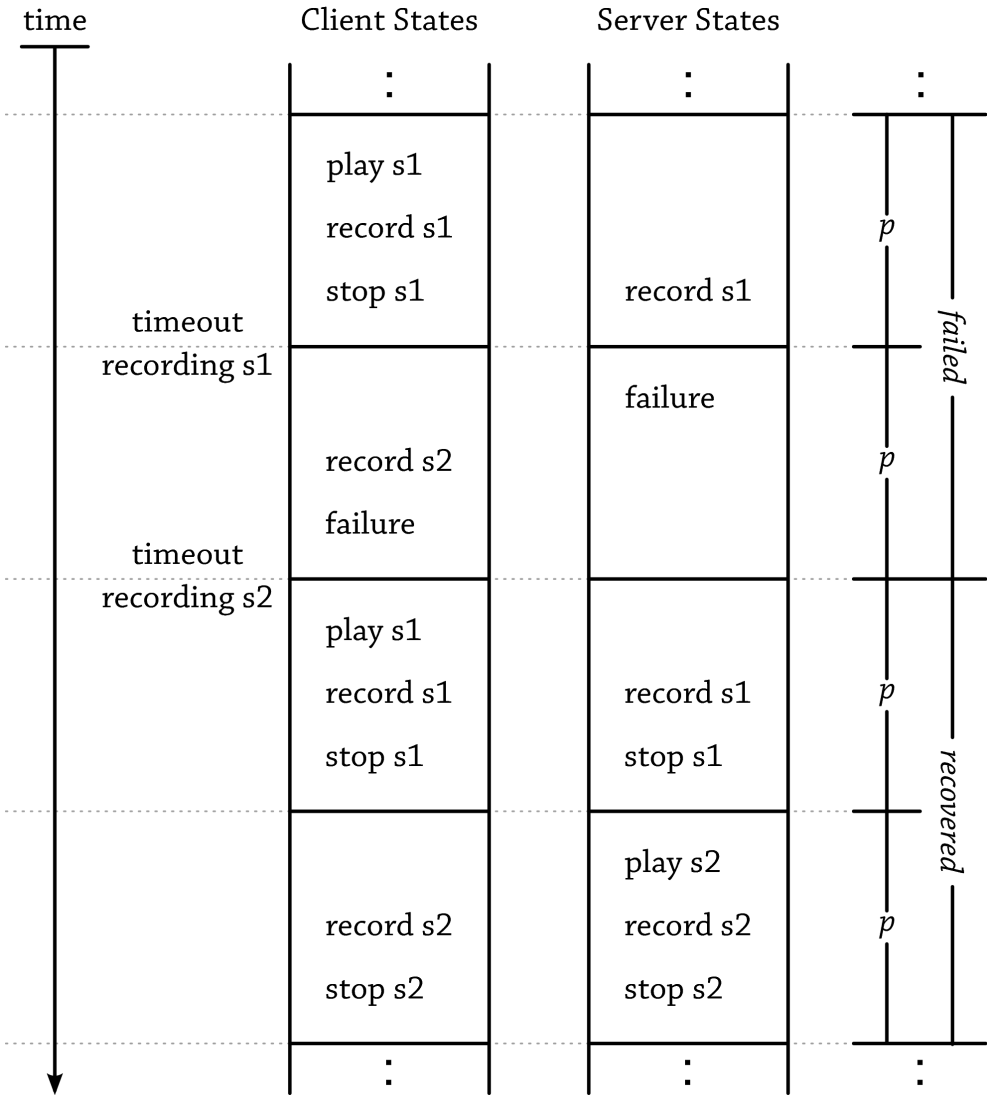
Event Timeline in an Ideal Scenario



Event Timeline

Related Work – Implementation

Event Timeline When a Failure Occurs



Event Timeline When a Failure Occurs

Related Work – Implementation

NXT Distance Application in Action

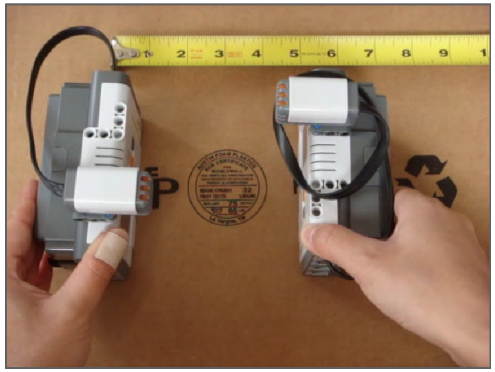
The client and server NXTs run the same version of the program.

The client and server are selected by the user when the application starts.

The client initiates the connection; the two NXTs take turns playing tones.

The client and server display synchronized measurements.

The program terminates when the user presses the escape button.



NXT Distance Application in Action

Videos of the NXT Distance Application in Action

`http://www2.hawaii.edu/~yucheng/projects/nxt-distance-implementation/#program-in-action`

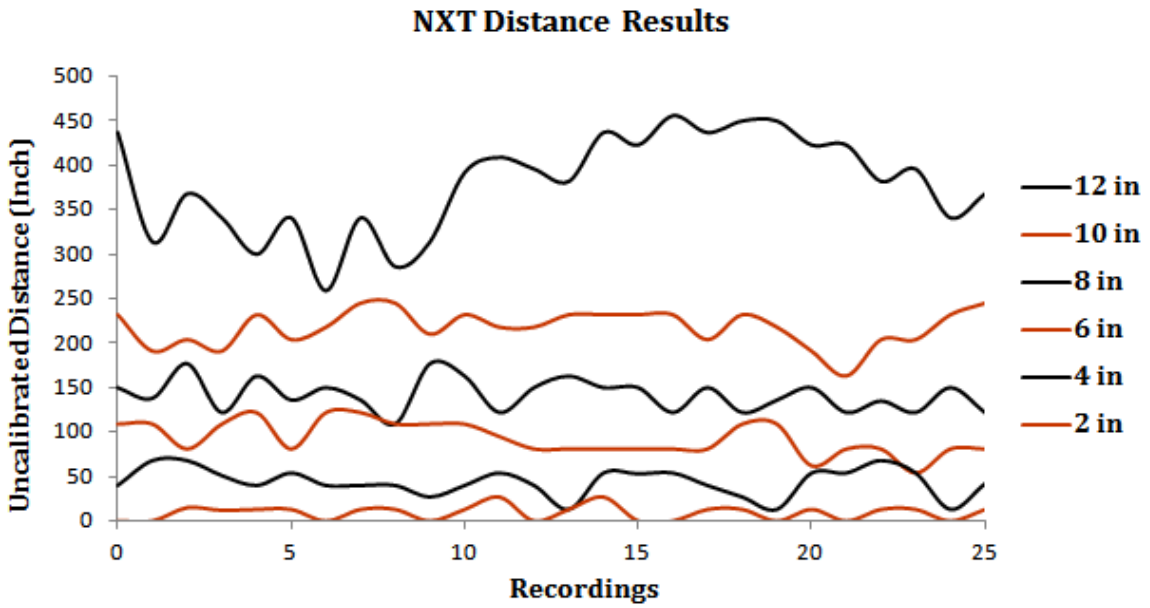
Related Work – Implementation

NXT Distance Application Results

Recordings were computed using the following formula as discussed previously.

$$d \approx \frac{M_1 - M_2}{2} \cdot c_{\text{sound}},$$

The speed of sound at 80 °F is 347 meters per second, which corresponds to 1.366×10^{-5} inches per nanosecond.

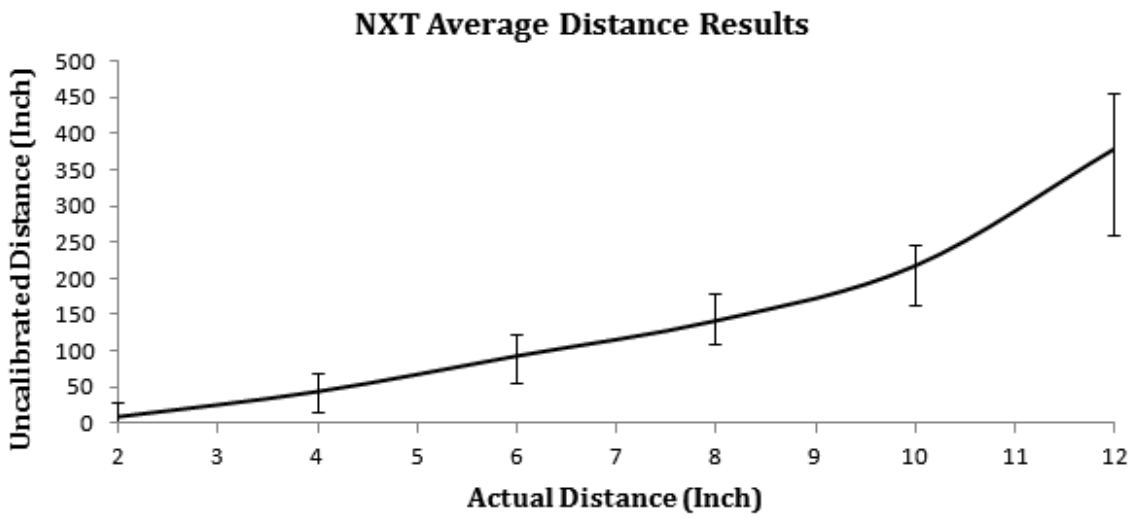


Related Work – Implementation

NXT Distance Application Results

The previous chart showed recorded versus actual distance per actual distance.

The following chart shows the average recorded distance versus actual distance.



The NXT Distance application did not calculate the correct distances.

The NXT Distance application reported consistent values that directly corresponded to the actual distances of separation of two NXTs.

Related Work – Analysis

NXT Distance Application Analysis

At this point, we didn't know why we were getting these results.

Deeper analysis and more thorough investigation were required.

Verification of Algorithm Assumptions

To apply this algorithm, three environmental and hardware conditions need to hold.

$$c_{\text{sound at computer \#1}} = c_{\text{sound at computer \#2}}$$

$$t_1 \approx 0$$

$$t_{2 \text{ at computer \#1}} = t_{2 \text{ at computer \#2}}$$

Verify these three assumptions, respectively.

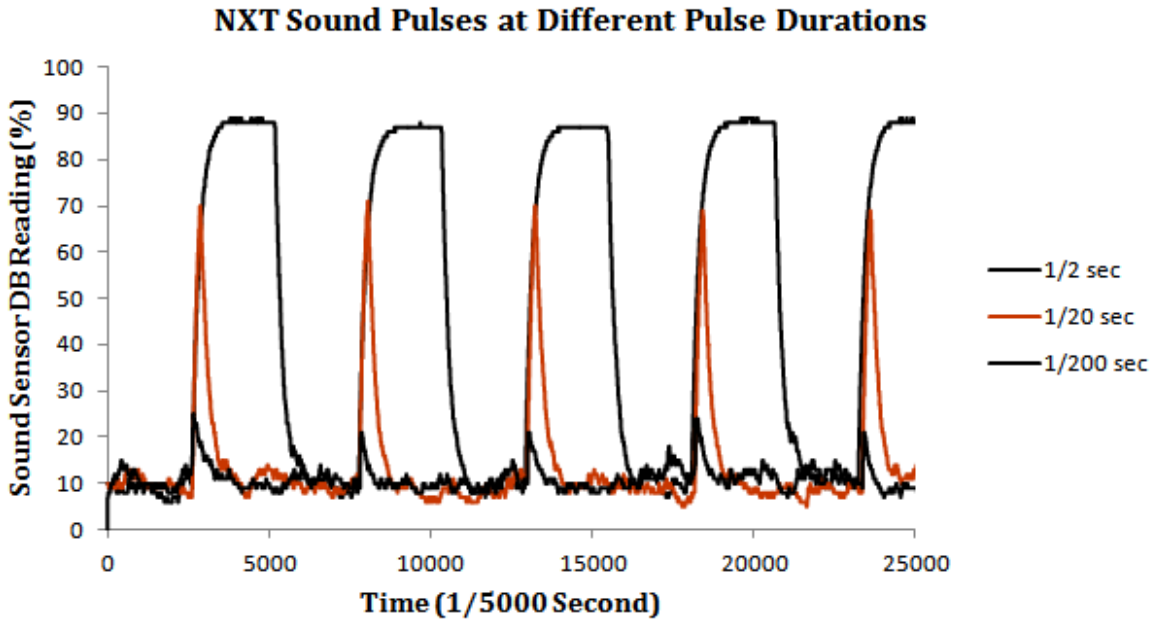
1. The environmental conditions are the same at the two NXTs.
2. The sound sensor of a NXT is sufficiently close to its speaker.
3. Two NXTs have identical hardware and instruction sets to record a sound.

Related Work – Analysis

Sound Pulse Attack Time vs. Sound Durations Experiment

We programmed one NXT to play a fixed tone at different durations.

We programmed the other NXT to record all sounds in real time.



Short sound durations would not suffice for the NXT Distance program.

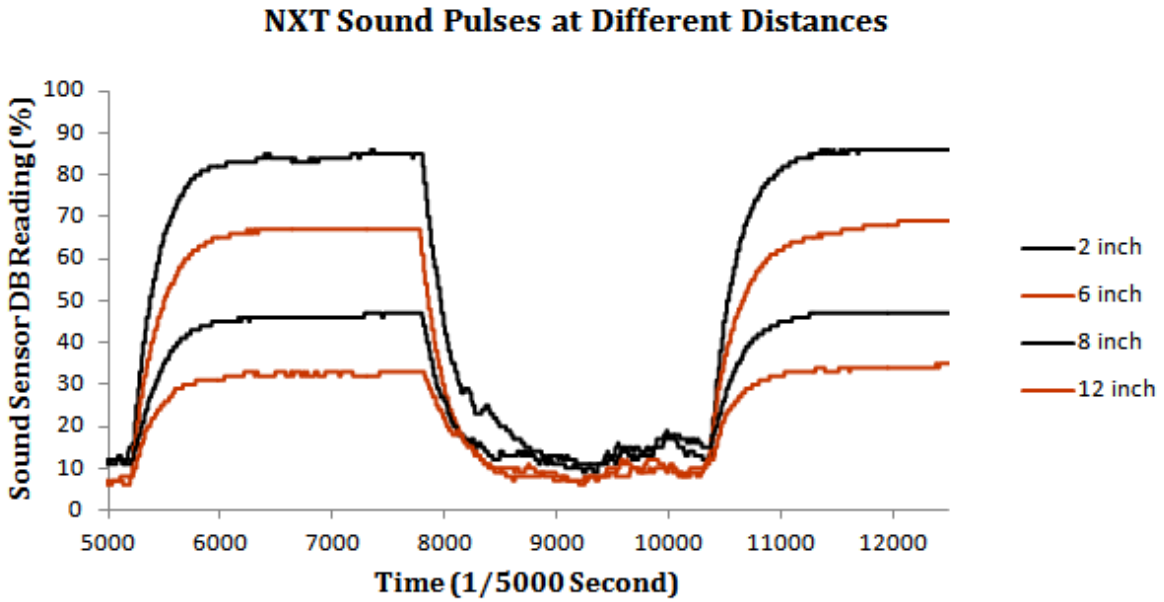
The attack time of the sound pulses is long, and a short duration is not enough time for the sensor to detect a tone generated by another NXT at its peak volume.

Related Work – Analysis

Sound Pulse Attack Time vs. Distances Experiment

We programmed one NXT to play a fixed tone at a fixed duration at different distances.

We programmed the other NXT to record all sounds in real time.



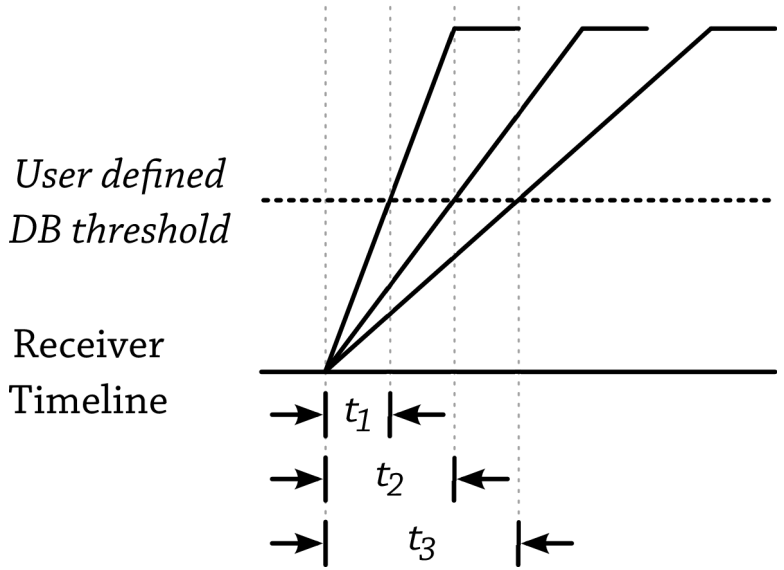
The NXT Distance application takes significantly different periods of time to reach the user-defined DB threshold.

This difference is based solely on the distance of separation between the two NXTs.

Related Work – Analysis

Sound Pulse Attack Time

The following chart illustrates the effect of distance on sound pulse attack time.

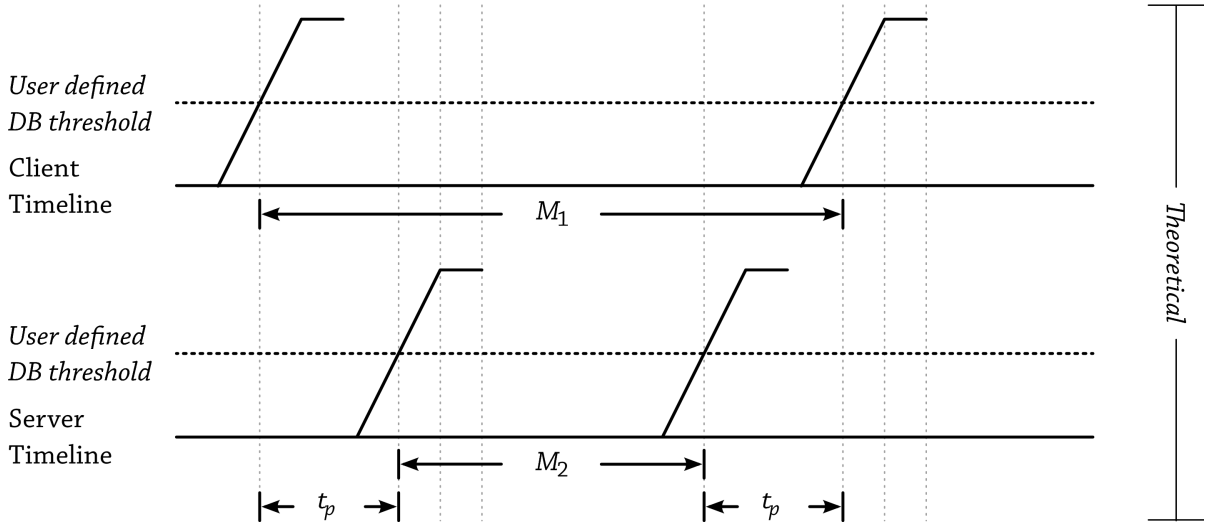


Symbol	***Description***
t_1	The amount of time to reach the DB threshold for a sound pulse generated at a close distance.
t_2	The amount of time to reach the DB threshold for a sound pulse generated at an average distance.
t_3	The amount of time to reach the DB threshold for a sound pulse generated at a far distance.

Related Work – Analysis

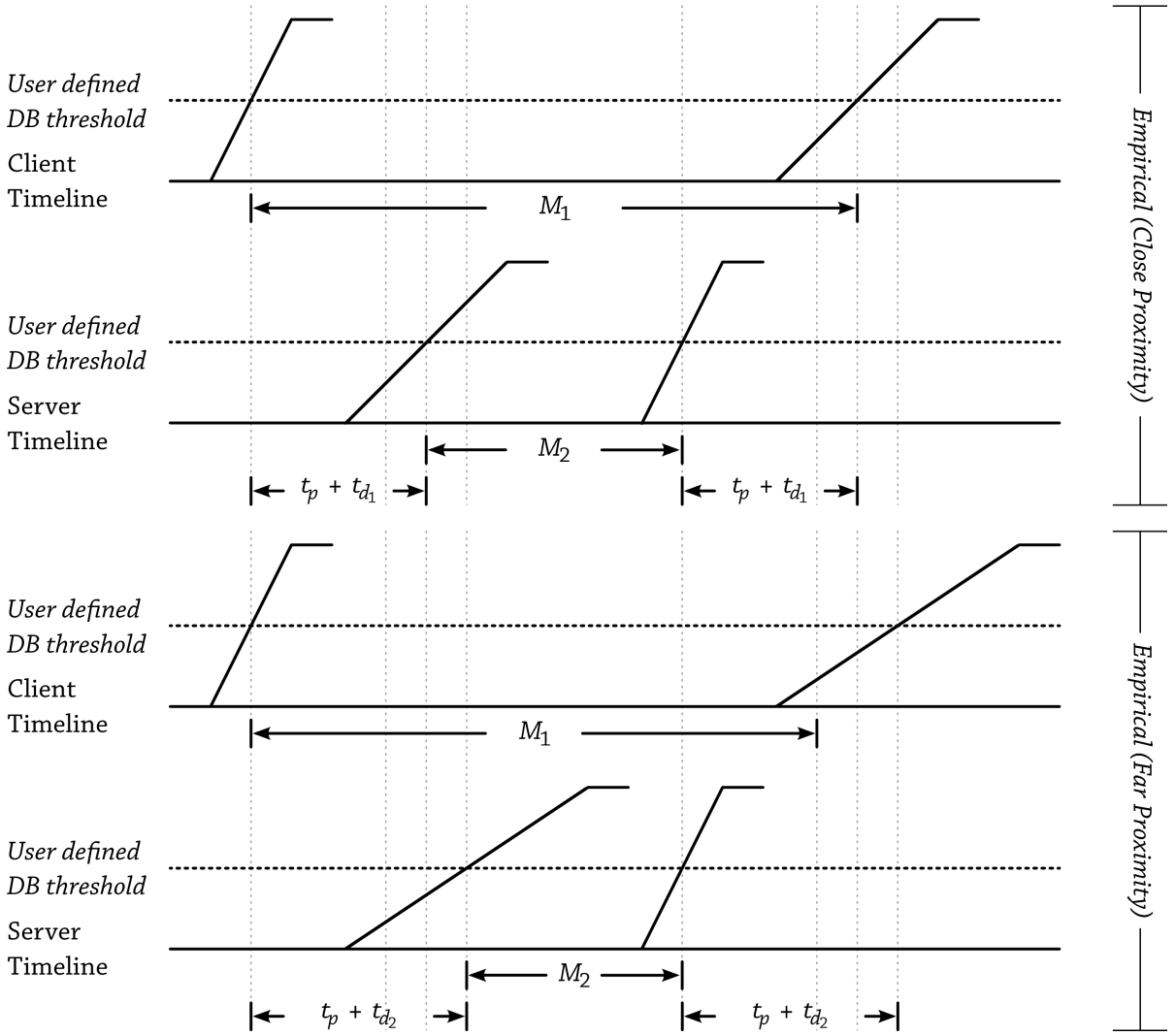
Attack Time Influence in NXT Distance Application

$$M_1 - M_2 = 2 \cdot t_p \,.$$



NXT Distance Application for Simplified Model

Related Work – Analysis



NXT Distance Application Considering Influence of Attack Time

Related Work – Analysis

Attack Time Influence in the NXT Distance Algorithm

$$M_1 - M_2 = 2 \cdot t_p + 2 \cdot t_d.$$

Symbol	***Description***
M_1	The interval between recording s1 and s2 on the client
M_2	The interval between recording s1 and s2 on server.
t_p	The propagation delay due to the distance between the two computers.
t_{d_1}	The delay caused by different attack times at distance d_1.
t_{d_2}	The delay caused by different attack times at distance d_2.

Discussion & Conclusions

Project Summary

We studied the algorithm of the Acoustic Tape Measure (Distance) program.

We explored the Java programming environment of the Lego NXT.

We performed a series of sound system experiments with test programs.

We implemented the Distance algorithm as an autonomous program running on NXTs.

We analyzed the application performance and noticed a consistent error term.

We conducted more experiments and determined the origin of this error.

Discussion & Conclusions

The NXT Distance Application

The NXT Distance application reports consistent values that directly correspond to the actual distances of separation of two NXTs.

The NXT Distance measurements within a detectable range can easily be calibrated so that the NXT reports correct distances.

The usefulness of this algorithm on the NXT is mainly restricted by the quick diminishing of sound sensitivity as the distance increases.

leJOS Java Support for NXT

The leJOS SDK utilizes a Java compiler to build images that execute on the NXT.

The JRE is limited but does support network communications, threading, and a variety of classes to control the NXT hardware.

Combined with Eclipse, leJOS and the Lego NXT could be a good candidate for introductory computer science and software engineering courses.