การพัฒนาระบบสื่อสารไร้สายต้นแบบในห้องเรียนสำหรับนักเรียนที่บกพร่องทางการได้ยินด้วยเทคโนโลยีบลูทูธ

นายอาภา สุวรรณรัตน์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์มหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2555

DEVELOPMENT OF A WIRELESS COMMUNICATION SYSTEM PROTOTYPE FOR STUDENTS WITH HEARING IMPAIRMENT USING BLUETOOTH TECHNOLOGY

Mr.Arpa Suwannarat

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Engineering Program in Computer Engineering

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2012

หัวข้อวิทยานิพนธ์	การพัฒนาระบบสื่อสารไร้สายต้นแบบในห้องเรียนสำหรับนักเรียนที่บกพร่องทางการได้ยินด้วยเทคโนโลยีบลูทูธ
โดย	นายอาภา สุวรรณรัตน์
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	ผู้ช่วยศาสตราจารย์ ดร. เศรษฐา ปานงาม
อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม	ดร. พศิน อิศรเสนา ณ อยุธยา

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

…….. คณบดีคณะวิศวกรรมศาสตร์
(รองศาสตราจารย์ ดร. บุญสม เลิศหิรัญวงศ์)

คณะกรรมการสอบวิทยานิพนธ์

…… ประธานกรรมการ
(ศาสตราจารย์ ดร. ประภาส จงสถิตย์วัฒนา)

…….. อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(ผู้ช่วยศาสตราจารย์ ดร. เศรษฐา ปานงาม)

…….. อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม
(ดร. พศิน อิศรเสนา ณ อยุธยา)

…… กรรมการภายนอกมหาวิทยาลัย
(ดร. อภิชาติ อินทรพานิชย์)

อาภา สุวรรณรัตน์ : การพัฒนาระบบสื่อสารไร้สายต้นแบบในห้องเรียนสำหรับนักเรียนที่บกพร่องทางการได้ยินด้วยเทคโนโลยีบลูทูธ. (DEVELOPMENT OF A WIRELESS COMMUNICATION SYSTEM PROTOTYPE FOR STUDENTS WITH HEARING IMPAIRMENT USING BLUETOOTH TECHNOLOGY) อ. ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ.ดร. เศรษฐา ปานงาม, อ. ที่ปรึกษาวิทยานิพนธ์ร่วม : ดร. พศิน อิศรเสนา ณ อยุธยา, 106 หน้า.

สำหรับนักเรียนที่บกพร่องทางการได้ยินแล้ว ระบบสื่อสารไร้สายในห้องเรียนเป็นอุปกรณ์ที่มีความสำคัญเป็นอย่างยิ่ง เนื่องจากเป็นอุปกรณ์ที่ช่วยการเรียนการสอนในห้องเรียนเป็นไปอย่างมีประสิทธิภาพ ระบบเอฟเอ็มเป็นระบบการสื่อสารไร้สายระบบหนึ่งซึ่งถูกนำมาใช้ในการเรียนการสอนในห้องเรียนของนักเรียนที่บกพร่องทางการได้ยิน แต่ระบบเอฟเอ็มก็มีข้อเสียสำคัญ คือไม่ทนทานต่อสัญญาณรบกวน ประกอบกับคุณสมบัติสำคัญ 3 ประการของเทคโนโลยีบลูทูธ ได้แก่ ทนทานต่อสัญญาณรบกวน ใช้พลังงานต่ำ และราคาถูก รวมถึงความแพร่หลายในการใช้งานเทคโนโลยีบลูทูธในปัจจุบัน งานวิจัยนี้จึงได้นำเสนอการนำออกแบบและพัฒนาระบบสื่อสารไร้สายในห้องเรียนต้นแบบด้วยเทคโนโลยีบลูทูธ โดยระบบต้นแบบถูกพัฒนาให้เป็นเรียลไทม์แอพพลิเคชันที่ใช้ช่องสัญญาณ ACL ในการส่งข้อมูลเสียงผ่านทางชั้นการทำงาน L2CAP จากการทดลองพบว่า เทคโนโลยีบลูทูธสามารถนำมาประยุกต์ใช้ในการพัฒนาระบบสื่อสารไร้สายในห้องเรียนสำหรับนักเรียนที่บกพร่องทางการได้ยินได้ สำหรับจำนวนอุปกรณ์ลูกข่ายในพิโคเน็ต (piconet) ซึ่งเป็นข้อจำกัดของเทคโนโลยีบลูทูธนั้น สามารถแก้ไขได้โดยการเพิ่มจำนวนอุปกรณ์แม่ข่ายให้สอดคล้องกับจำนวนนักเรียนในห้องเรียน ซึ่งพิโคเน็ตใช้เวลาในการส่งข้อมูลเสียงต่ำกว่าการเพิ่มจำนวนอุปกรณ์ลูกข่ายด้วยการเชื่อมต่อแบบ Scatternet และสามารถรองรับการทำงานที่ระยะสูงสุด 12 เมตร

ภาควิชา : .....วิศวกรรมคอมพิวเตอร์.... ลายมือชื่อนิสิต .....
สาขาวิชา : .....วิศวกรรมคอมพิวเตอร์... ลายมือชื่อ อ. ที่ปรึกษาวิทยานิพนธ์หลัก .....
ปีการศึกษา : .....2555..... ลายมือชื่อ อ. ที่ปรึกษาวิทยานิพนธ์ร่วม .....

5270580721 : MAJOR COMPUTER ENINEERING
KEYWORDS : STUDENTS WITH HEARING IMPAIRMENT / ASSISTIVE LISTENING SYSTEMS (ALSs) / BLUETOOTH TECHNOLOGY

ARPA SUWANNARAT : DEVELOPMENT OF A WIRELESS COMMUNICATION SYSTEM PROTOTYPE FOR STUDENTS WITH HEARING IMPAIRMENT USING BLUETOOTH TECHNOLOGY. ADVISOR : ASST.PROF. SETHA PAN-NGUM, Ph.D., CO-ADVISOR : PASIN ISRASENA, Ph.D., 106 pp.

For Students with hearing impairment, Assistive Listening Systems (ALSs) is an important system used to make dialogue accessible in classroom environment. FM system is one of the widely used ALS which is named by the method of signal transmission. The major disadvantage of FM system is that it is subject to interference from other sources. The key features of Bluetooth wireless technology are robustness, low power and low cost. Also with the popularity of Bluetooth, this research aims to design and develop an ALS prototype for students with hearing impairment using Bluetooth technology. The prototype is a real-time application that uses ACL channel to send audio data via L2CAP layer. The result shows that it is possible to develop Bluetooth ALS which produces audible sound from one teacher to seven students. The limitation of number of Bluetooth connections can be overcome by using more than one transmitter forming a piconet according to the number of students. Transmission by a piconet gives lower latency than by a scatternet. In addition, the prototype can support the maximum transmission range of 12 meters.

Department : …Computer Engineering… Student's Signature : .....
Field of Study : ..Computer Engineering... Advisor's Signature : .....
Academic Year : .....2012..... Co-advisor's Signature : .....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยความอนุเคราะห์อย่างยิ่งของ ผศ.ดร.เศรษฐา ปานงาม และ ดร.พศิน อิศรเสนา ณ อยุธยา ซึ่งท่านได้ให้ความรู้ แนะนำแนวทางการวิจัย ตรวจสอบให้คำแนะนำ และสนับสนุนเป็นอย่างดี จนทำให้การวิจัยในครั้งนี้สำเร็จออกมาด้วยดี

ขอขอบคุณครอบครัวที่คอยติดตามให้กำลังใจและสนับสนุน ขอบคุณวศิน เที่ยงคุณากฤต ที่ให้คำแนะนำและความช่วยเหลือเป็นอย่างดี รวมถึงท่านอื่น ๆ ที่มิได้กล่าวชื่อไว้ ณ ที่นี้ ที่มีส่วนช่วยให้วิทยานิพนธ์สำเร็จได้ด้วยดี

สารบัญ

หน้า

หน้า

สารบัญตาราง

หน้า

สารบัญรูป

หน้า

หน้า

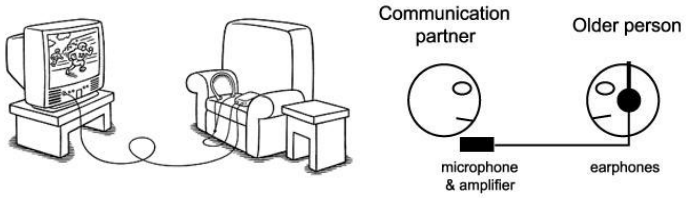
หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

งานวิจัยนี้สนใจเกี่ยวกับเด็กที่บกพร่องทางการได้ยินเป็นพิเศษ เด็กที่บกพร่องทางการได้ยินนั้นอาจเป็นได้ทั้งเด็กหูตึงและหูหนวก ในกรณีที่หูหนวกนั้นจะหมายถึง เด็กที่สูญเสียการได้ยิน 90 เดซิเบลขึ้นไป สำหรับกรณีที่หูตึงจะหมายถึงเด็กที่สูญเสียการได้ยินตั้งแต่ 26-89 เดซิเบล ซึ่งเด็กหูตึงสามารถได้ยินได้โดยอาศัยเครื่องช่วยฟัง (Hearing Aid) ดังนั้นเด็กที่บกพร่องทางการได้ยินจึงสามารถทำการเรียนการสอนได้โดยใช้วิธีสอนพูด (Oral Method) การทำการเรียนการสอนโดยวิธีสอนพูดนั้น ครูจะไม่ใช้ภาษามือในการสื่อสารกับนักเรียน แต่จะใช้การพูด โดยนักเรียนแต่ละคนจะมีเครื่องช่วยฟัง ซึ่งจะทำหน้าที่ช่วยขยายเสียงที่ครูพูด ทำให้นักเรียนที่มีความบกพร่องทางการได้ยินสามารถใช้ชีวิตประจำวันได้เหมือนกับคนปกติทั่วไป การเรียนการสอนวิธีนี้ ครูต้องทำการเตรียมเครื่องเสียงให้มีเสียงเหมาะกับขนาดห้องเรียน จำนวนนักเรียน และสภาพแวดล้อมของห้องเรียน ปัญหาที่พบมากคือ นักเรียนที่บกพร่องทางการได้ยินอาจจะฟังเสียงครูไม่ชัดเจน เนื่องจากสภาพแวดล้อมของห้องเรียน ระยะห่างจากลำโพง รวมถึงคุณภาพของลำโพง ปัญหาที่เกิดขึ้นเหล่านี้ทำให้เกิดความยากลำบากในการเรียนของนักเรียน แต่หากในชั้นเรียนมีอุปกรณ์สื่อสารไร้สายที่ช่วยให้สามารถตัดปัญหาต่างๆดังกล่าวออกไปได้ก็จะเป็นประโยชน์กับการเรียนการสอนเป็นอย่างยิ่ง

สำหรับระบบช่วยการได้ยิน (ALSs: Assistive Listening Systems) [1] จะมีอุปกรณ์ช่วยในการได้ยิน (ALDs: Assistive Listening Devices) เป็นอุปกรณ์เพื่อช่วยในการฟังในสถานการณ์ที่ยากลำบาก โดยไม่จำเป็นว่าผู้ที่สวมใส่อุปกรณ์นั้นจะเป็นผู้ที่บกพร่องทางการได้ยินหรือไม่ ซึ่งอุปกรณ์ช่วยการได้ยินอาจจะให้ผลการได้ยินที่ดีกว่าการได้ยินของผู้ที่ไม่บกพร่องทางการได้ยินที่ไม่ได้ใส่อุปกรณ์นี้ อุปกรณ์ช่วยการได้ยินสามารถช่วยให้ผู้ฟังมีสมาธิจดจ่อกับเสียงที่ต้องการฟังได้ในกรณีที่ผู้ฟังและผู้พูดอยู่ห่างกัน เช่น เสียงของครูในห้องเรียน ห้องทำงาน หรือห้องประชุม และช่วยลดเสียงรบกวนจากสภาพแวดล้อม เช่น เสียงพูดคุย เสียงยานพาหนะ เป็นต้น ทำให้การสื่อสารชัดเจนยิ่งขึ้น



รูปที่ 1 การใช้งานอุปกรณ์ช่วยฟัง [1]

ชนิดและระบบช่วยการได้ยินสามารถแบ่งได้เป็น 3 ระบบ ได้แก่ ระบบไอแอล (IL: Induction Loop) ระบบคลื่นสัญญาเอฟเอ็ม (FM: Frequency Modulation) และระบบไออาร์ (IR: Infrared) ซึ่งทั้งสามระบบดังกล่าวมีข้อดีและข้อด้อยแตกต่างกัน ทั้งค่าใช้จ่าย ความยุ่งยากซับซ้อนในการพัฒนา และการใช้งานเป็นต้น

สำหรับเทคโนโลยีบลูทูธนั้น เป็นเทคโนโลยีสื่อสารไร้สายที่ถูกพัฒนาขึ้นเพื่อทดแทนการใช้งานอุปกรณ์ใช้สายต่างๆ ไม่ว่าจะเป็นการเชื่อมต่อคอมพิวเตอร์เข้ากับอุปกรณ์พกพาต่างๆ ได้แก่ โทรศัพท์มือถือ PDA หรือ Pocket PC หรือจะเป็นการเชื่อมต่อโทรศัพท์มือถือเข้ากับชุดหูฟัง เป็นต้น นอกจากนี้เทคโนโลยีบลูทูธยังมีคุณสมบัติ 3 ประการ ได้แก่ ทนทานต่อสัญญาณรบกวน ใช้พลังงานต่ำ และราคาถูก [2] ประกอบกับเป็นเทคโนโลยีที่กำลังเป็นที่สนใจแลใช้งานอย่างแพร่หลายในปัจจุบัน จึงมีแนวคิดในการพัฒนาอุปกรณ์ไร้สายต้นแบบซึ่งใช้เทคโนโลยีบลูทูธเพื่อช่วยในการเรียนการสอนสำหรับนักเรียนที่บกพร่องทางการได้ยินที่มีประสิทธิภาพดีและราคาถูก เพื่อให้สามารถดำเนินการเรียนการสอนได้อย่างมีประสิทธิภาพ

1.2 วัตถุประสงค์ของการวิจัย

เพื่อพัฒนาระบบสื่อสารไร้สายต้นแบบในห้องเรียนสำหรับนักเรียนที่บกพร่องทางการได้ยินด้วยเทคโนโลยีบลูทูธ

1.3 ขอบเขตของการวิจัย

- ระบบมีการสื่อสารทางเดียวแบบแพร่สัญญาณ (one-to-many) จากครูถึงนักเรียนมากกว่า 7 คน
- ระบบมีระยะการทำงาน 10 เมตรโดยไม่มีสิ่งกีดขวาง ตามสภาพการใช้งานในห้องเรียนทั่วไป
- ระบบสามารถทำการเชื่อมต่ออัตโนมัติระหว่างอุปกรณ์ที่ตัวครูและนักเรียน

1.4 ประโยชน์ที่คาดว่าจะได้รับ

ได้อุปกรณ์ต้นแบบในการช่วยฟังในห้องเรียนสำหรับนักเรียนที่บกพร่องทางการได้ยิน โดยใช้ระบบสื่อสารไร้สายด้วยเทคโนโลยีบลูทูธ

1.5 วิธีดำเนินการทำวิจัย

- ศึกษาข้อมูลเทคโนโลยีบลูทูธ
- ศึกษาข้อมูลเอกสารและงานวิจัยที่เกี่ยวข้องกับหัวข้องานวิจัย
- ออกแบบลักษณะการทำงาน
- พัฒนาโปรแกรมเพื่อสามารถทำงานแบบแพร่สัญญาณ (Broadcast) ได้
- พัฒนาประสิทธิภาพและคุณภาพของข้อมูลเสียง
- ทดสอบและตรวจสอบประสิทธิภาพของระบบสื่อสารไร้สายต้นแบบ
- ปรับปรุงประสิทธิภาพของระบบสื่อสารไร้สายต้นแบบ
- สรุปผลการวิจัยและจัดทำวิทยานิพนธ์

1.6 ผลงานที่ตีพิมพ์จากวิทยานิพนธ์

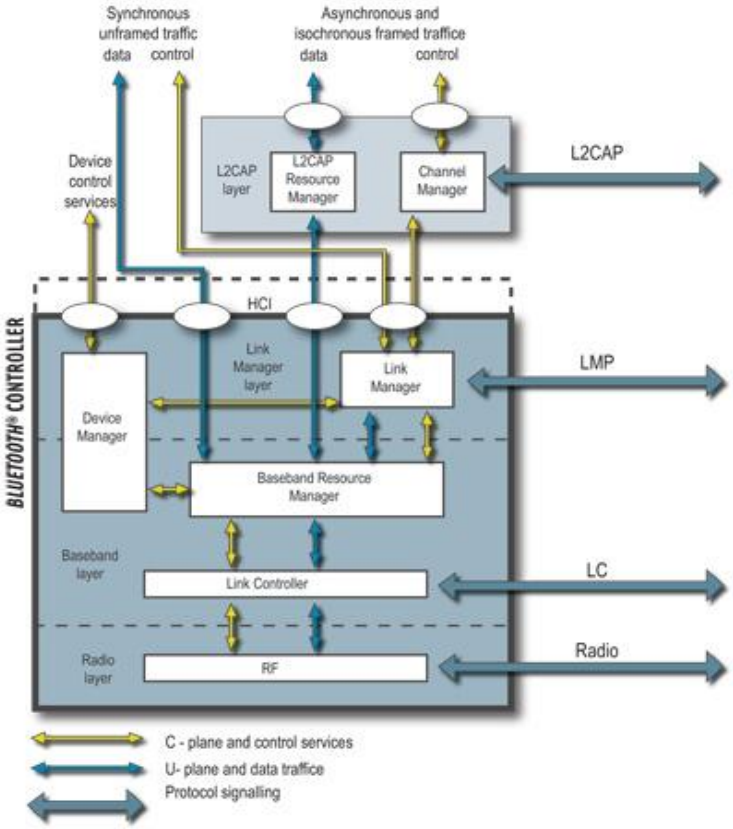
ส่วนหนึ่งของวิทยานิพนธ์นี้ได้รับการตอบรับให้ตีพิมพ์เป็นบทความทางวิชาการในหัวข้อเรื่อง Evaluation of Bluetooth Based Assistive Listening System [3] โดย นายอาภา สุวรรณรัตน์ ผศ.ดร. เศรษฐา ปานงาม และ ดร. พศิน อิศรเสนา ณ อยุธยา ในงานประชุมวิชาการ ICICTES – The International Conference on Information and Communication Technology for Embedded Systems (ICICTES 2013) ณ บ้านอัมพวารีสอร์ท จังหวัดสมุทรสงคราม ประเทศไทย วันที่ 24-26 มกราคม พ.ศ. 2556

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 สถาปัตกรรมของเทคโนโลยีบลูทูธ (Bluetooth Architecture)

บลูทูธเป็นเทคโนโลยีสื่อสารไร้สายที่ถูกพัฒนาขึ้นเพื่อทดแทนการใช้งานอุปกรณ์ใช้สายต่างๆ ทั้งอุปกรณ์ในรถยนต์ โทรศัพท์เคลื่อนที่ จนถึงอุปกรณ์ทางการแพทย์และคอมพิวเตอร์ โดยมีคุณสมบัติหลัก 3 ประการได้แก่ ทนทานต่อสัญญาณรบกวน ใช้พลังงานต่ำและราคาถูก



รูปที่ 2 สถาปัตยกรรมของเทคโนโลยีบลูทูธ [2]

จากรูปที่ 2 แสดงสถาปัตยกรรมของระบบบลูทูธ ซึ่งประกอบด้วย 4 ชั้นการทำงานหลักที่ถูกกำหนดไว้ใน Bluetooth Specification

2.1.1.1 คลื่นวิทยุ (Radio)

เทคโนโลยีบลูทูธทำงานบนคลื่นความถี่สาธารณะสำหรับอุตสาหกรรม วิทยาศาสตร์ และการแพทย์ (Unlicensed ISM) ที่ย่านความถี่ 2.4 GHz โดยอาศัยเครื่อง

รับส่งสัญญาณที่เป็น Frequency Hop เพื่อจัดการกับการรบกวนและความอ่อนแอลงของสัญญาณ

โหมดของการกล้ำสัญญาณ (Modulation Mode) ได้แก่ โหมดการทำงานปกติเรียกว่า Basic Rate ที่ใช้เทคนิค Shaped, Binary FM Modulation เพื่อลดความซับซ้อนของเครื่องรับส่งสัญญาณ และโหมดเสริม เรียกกว่า Enhanced Data Rate ซึ่งใช้เทคนิคการกล้ำสัญญาณแบบพีเอสเค (PSK Modulation) มีค่า Variant 2 ค่า คือ **π**/4-DQPSK และ 8DPSK โดยซิมโบลเรตสำหรับทุกการกล้ำสัญญาณคือ 1 Ms/s และบิตเรตที่ 1 Mbps สำหรับ Basic Rate 2 Mbps สำหรับ Enhanced Data Rate ที่ใช้ **π**/4-DQPSK และ 3 Mbps สำหรับ Enhanced Data Rare ที่ใช้ 8DPSK และทั้ง Basic Rate และ Enhanced Data Rate เป็นการสื่อสารแบบ Full Duplex ที่ใช้เทคนิค Time Division Duplex (TDD)

ระยะการทำงานของอุปกรณ์บลทูธสามารถแบ่งได้เป็น 3 Class ดังนี้

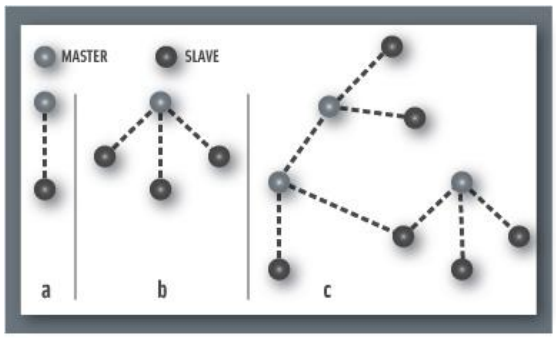
Class 3 ระยะการทำงาน 1 เมตรหรือ 3 ฟุต

Class 2 ระยะการทำงาน 10 เมตรหรือ 33 ฟุต

Class 1 ระยะการทำงาน 100 เมตรหรือ 300 ฟุต

2.1.1.2 Baseband

Baseband กำหนดขั้นตอนการใช้งานชั้นการทำงาน Radio ระหว่างอุปกรณ์บลูทูธ ซึ่งระบบบลูทูธให้บริการการเชื่อมต่อแบบ point-to-point และ point-to-multipoint ดังรูป



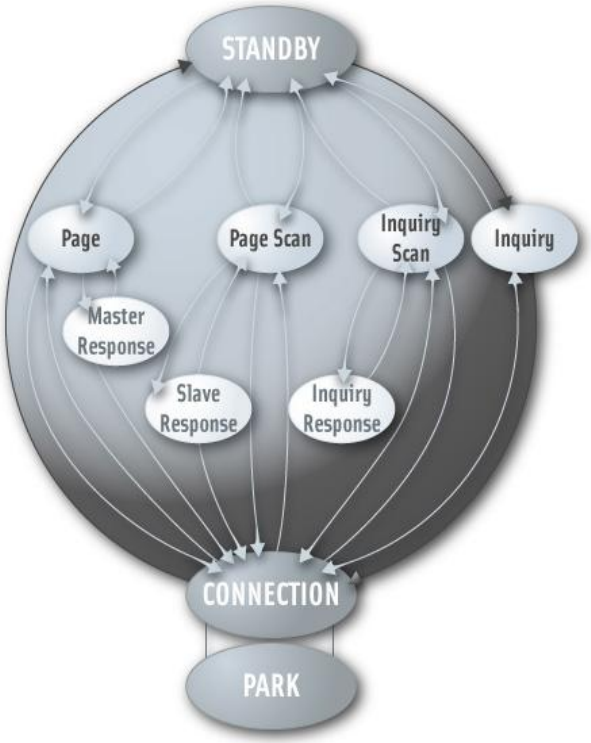
รูปที่ 3 รูปแบบการเชื่อมต่อของเทคโนโลยีบลูทูธ [2]

(a) **พิโคเน็ตที่มีอุปกรณ์ลูกข่ายตัวเดียว** (b) **พิโคเน็ตที่มีอุปกรณ์ลูกข่ายหลายตัว** (c) **การเชื่อมต่อแบบ** Scatternet

พิโคเน็ต (Piconet) เป็นรูปแบบการเชื่อมต่อของอุปกรณ์บลูทูธตั้งแต่ 2-7 ตัวซึ่งใช้ Physical Channel ร่วมกัน โดยมีอุปกรณ์ตัวหนึ่งทำหน้าที่เป็นแม่ข่าย (Master) และ

อุปกรณ์ที่เหลือทำหน้าที่เป็นลูกข่าย (Slave) สำหรับ Scatternet เป็นรูปแบบการเชื่อมต่อซึ่งอุปกรณ์ตัวหนึ่งทำหน้าที่เป็นแม่ข่ายในพิโคเน็ตหนึ่งและเป็นลูกข่ายในอีกพิโคเน็ตหนึ่ง

2.1.1.2.1 Link Controller



รูปที่ 4 แผนภาพการทำงานของ Link Controller [2]

รูปที่ 4 แสดงแผนภาพการทำงานของ Link Controller ซึ่งประกอบด้วย 3 สถานะหลัก คือ STANDBY, CONNECTION และ PARK ประกอบกับ 7 สถานะย่อย ได้แก่ Page, Page Scan, Inquiry, Inquiry Scan, Master response, Slave Response และ Inquiry Response ซึ่งสถานะย่อยเหล่านี้จะทำงานระหว่างสถานะหลักเพื่อสร้างการเชื่อมต่อและค้นหาอุปกรณ์อื่น โดยในการเปลี่ยนจากสถานะหลักหรือสถานะย่อยไปยังอีกสถานะหนึ่งนั้น ต้องใช้คำสั่งจาก Link Manager หรือสัญญาณภายในของ Link Controller เช่น Trigger หรือ Timeout

2.1.1.3 Link Manager Protocol (LMP)

LMP ทำหน้าที่ควบคุมและต่อรองการทำงานระหว่างอุปกรณ์ ประกอบด้วยการสร้างและควบคุม Logical Transports, Logical Links และควบคุม Physical Links โดย

อาศัยการติดต่อกันระหว่าง Link Manager (LM) ระหว่างอุปกรณ์ด้วย ACL Logical Transport

2.1.1.4 Logical Link Control and Adaption Protocol (L2CAP)

L2CAP รองรับการทำงานของชั้นการทำงานด้านบนโดยทำหน้าที่ Multiplexing, แบ่งและรวมข้อมูล และส่งข้อมูลตามข้อกำหนดด้านคุณภาพ (QoS) ซึ่งสามารถแบ่งการทำงานได้เป็น 3 โหมด ได้แก่

Basic L2CAP Mode

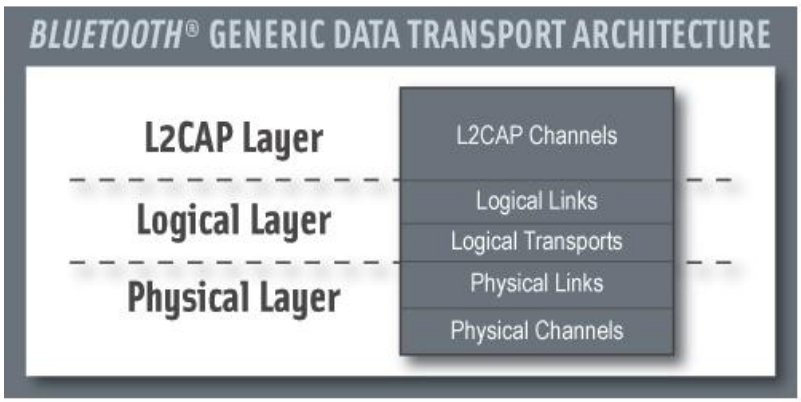
Flow Control Mode

Retransmission Mode

2.1.2 ระบบการส่งข้อมูลของเทคโนโลยีบลูทูธ (Data Transport)

2.1.2.1 ระบบการส่งข้อมูลของเทคโนโลยีบลูทูธ (Data Transport)

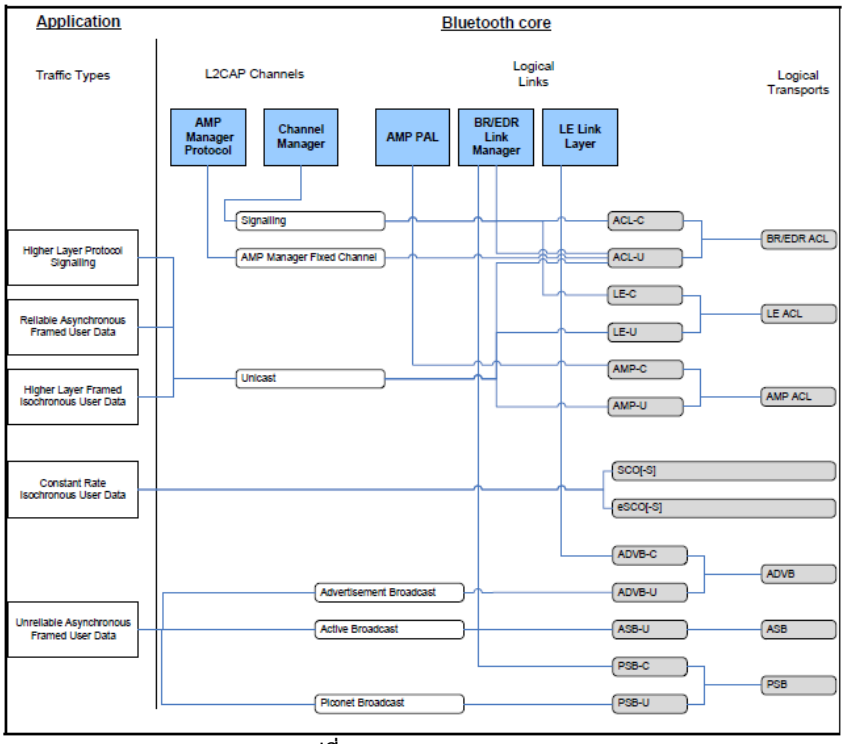
ระบบการส่งข้อมูลของระบบบลูทูธทำงานตามสถาปัตยกรรมชั้นการทำงาน ซึ่งแบ่งการทำงานออกเป็นระดับชั้นต่างๆ ซึ่งสามารถอธิบายได้ดังรูป



รูปที่ 5 สถาปัตยกรรมการส่งข้อมูลของเทคโนโลยีบลูทูธ [2]

สถาปัตยกรรมการส่งข้อมูลของบลูทูธซึ่งแบ่ง Logical Layer ออกเป็นสองชั้นการทำงานย่อย ได้แก่ Logical Links และ Logical Transports ซึ่ง Logical Links จะทำหน้าที่ขนส่งข้อมูลระหว่างอุปกรณ์ ส่วน Logical Transports จะหน้าที่ให้บริการ Logical Links ชนิดต่างๆ

2.1.2.1.1 Core Traffic Bearers



รูปที่ 6 Core Traffic Bearers [2]

บลูทูธประกอบด้วย Traffic Bearers มาตราฐานสำหรับโปรโตคอลและข้อมูลต่างๆ

Logical Links จะถูกตั้งชื่อตามชนิดของ Logical Transport ที่เกี่ยวข้องและต่อท้ายด้วยชนิดของข้อมูลที่ถูกส่ง เช่น C สำหรับการส่ง LMP Messages, U สำหรับการส่งข้อมูลผู้ใช้ (L2CAP PDUs) และ S สำหรับการส่งข้อมูลที่เป็น Stream ไม่ว่าจะเป็นชนิด Synchronous หรือ Isochronous ซึ่งส่วนต่อท้ายเหล่านี้สามารถละได้หากเข้าใจว่ากำลัง Logical Link ใดกำลังทำงานหรือถูกพูดถึงอยู่ เช่น ACL Logical Link จะหมายถึง ACL-C Logical Link ในกรณีที่กำลังส่ง LMP Messages หรือจะหมายถึง ACL-U Logical Link ในกรณีที่กำลังส่งข้อมูลของชั้นการทำงาน L2CAP

สำหรับการทำงานของแต่ละแอพพลิเคชันนั้น จะใช้ Traffic Bearers ชนิดใดก็ขึ้นกับคุณลักษณะของการส่งข้อมูลของแอพพลิเคชันนั้น เช่น ในพิโคเน็ตที่มีลูกข่ายหนึ่งตัว เครื่องแม่ข่าย (Master) อาจเลือกส่งข้อมูลบน ACL-U Logical Link แทนการส่งข้อมูลบน ASB-U หรือ PSB-U Logical Links ซึ่งจะให้ประสิทธิภาพในการส่งข้อมูลที่ดีกว่า

2.1.2.1.1.1. Framed Data Traffic

ชั้นการทำงาน L2CAP ให้บริการ Frame-Oriented Transport สำหรับข้อมูลผู้ใช้ชนิด Asynchronous และ Isochronous โดยข้อมูลที่รองรับโดยบริการนี้จะเป็นข้อมูลชนิด Variable-Sized Frames ซึ่งขึ้นกับการต่อรองสำหรับแต่ละช่องสัญญาณ

Connection-Oriented L2CAP Channels อาจถูกสร้างสำหรับการส่งข้อมูลแบบ Unicast (point-to-point) หรือการส่งข้อมูลจากเครื่องแม่ข่ายไปยังเครื่องลูกข่ายรายเดียวซึ่งสามารถเป็นได้ทั้งการส่งข้อมูลทางเดียวหรือการส่งข้อมูลสองทาง สำหรับ Connectionless L2CAP Channels จะใช้ส่งข้อมูลชนิด Broadcast และจะเป็นการส่งข้อมูลชนิดทางเดียวเท่านั้น นอกจากนี้ L2CAP Channels ยังมีข้อกำหนด QoS ซึ่งเป็นข้อกำหนดต่างๆ ในการส่งข้อมูล เช่น การกำหนดช่วงเวลาสำหรับการส่งข้อมูล เป็นต้น

สำหรับ L2CAP Channel Manager มีหน้าที่ในการจัดเรียงข้อมูลให้เหมาะสมกับ Logical Link รวมถึงการ Multiplex ข้อมูลของชั้นการทำงาน L2CAP

2.1.2.1.1.2. Unframed Data Traffic

สำหรับแอพพลิเคชันที่ส่งข้อมูลเป็น Stream หรือไม่ต้องการส่งข้อมูลในลักษณะเฟรมข้อมูล อาจมีการส่งข้อมูลโดยตรงไปยัง Baseband Logical Link โดยไม่ต้องใช้ L2CAP Channels ซึ่งระบบบลูทูธรองรับการส่งข้อมูลลักษณะนี้โดยมีชนิดข้อมูลเป็น Isochronous ที่มีความเร็วในการส่งข้อมูลคงที่โดยใช้ SCO-S หรือ eSCO-S Logical Link ซึ่ง Logical Links เหล่านี้จะทำการจอง Bandwidth ของ Physical Channel และให้บริการความเร็วในการส่งข้อมูลที่คงที่ตามสัญญาณนาฬิกาของพิโคเน็ต และข้อมูลจะถูกส่งในขนาดแพกเก็ตที่กำหนดในช่วงเวลาที่กำหนดซึ่งพารามิเตอร์ทั้งสองนี้จะถูกกำหนดขึ้นขณะทำการสร้างช่องสัญญาณ สำหรับ eSCO Links จะให้การส่งข้อมูลที่ความเร็วและความน่าเชื่อถือในการส่งข้อมูลสูงกว่าโดยอาศัยการส่งซ้ำ และการทำงานในโหมด Enhance Data Rate จะถูกรองรับโดย eSCO เท่านั้น นอกจากนี้ทั้ง SCO และ eSCO จะไม่รองรับการทำ Multiplexing ข้อมูลบน Logical Links หรือชั้นการทำงานใดๆทั้งสิ้น

ในการส่งข้อมูลนั้น แอพพลิเคชันจะเลือกชนิดของ Logical Links ที่เหมาะสมที่สุดที่ Baseband รองรับ แล้วจึงทำการสร้าง กำหนดคุณลักษณะต่างๆในการส่งข้อมูลและยกเลิกเมื่อการส่งข้อมูลเสร็จสมบูรณ์ ซึ่งโดยปกติแล้วแอพพลิเคชันจะใช้ Framed L2CAP Unicast Channel เพื่อจะส่งข้อมูลควบคุม (C-Plane Information)

2.1.2.1.1.3. Reliability of Traffic Bearers

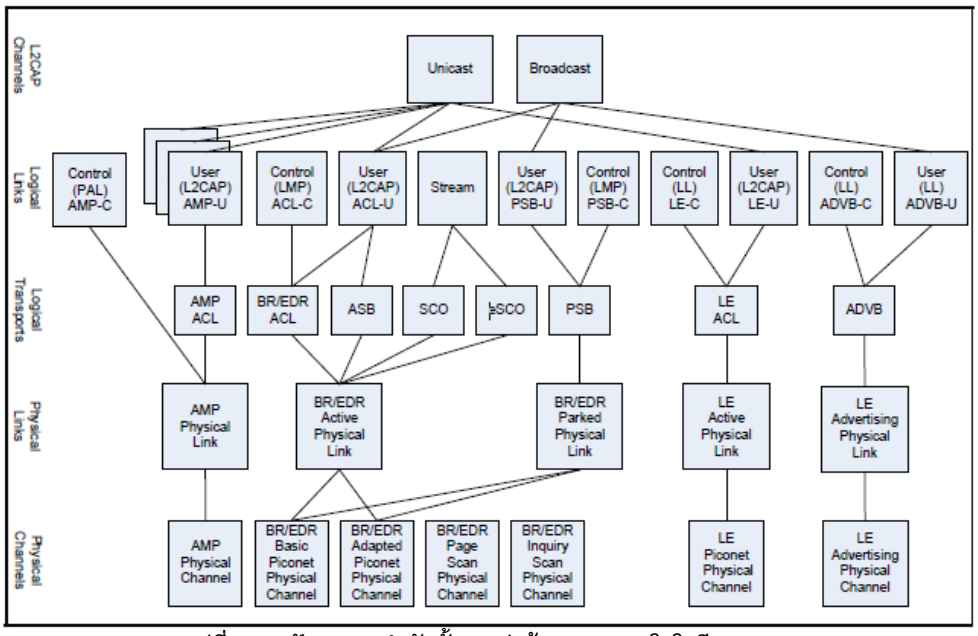
บลูทูธมีวิธีการจัดการกับความน่าเชื่อถือในการส่งข้อมูลในแต่ละชั้นการทำงาน โดยที่ Header ของ Baseband แพกเก็ตจะใช้ Forward Error Correction (FEC) ซึ่งเป็นการตรวจสอบความผิดพลาดของข้อมูลที่ฝั่งผู้รับและใช้ Header Error Check (HEC) เพื่อตรวจหาข้อผิดพลาดที่ยังเหลืออยู่หลังการแก้ไขข้อผิดพลาดดังกล่าว

สำหรับ ACL Logical Transports จะใช้โปรโตคอล ARQ ซึ่งเพิ่มความน่าเชื่อถือในการส่งข้อมูลโดยการส่งแพกเก็ตซ้ำ และ eSCO จะปรับปรุงการใช้งานโปรโตคอลนี้โดยการกำหนดจำนวนครั้งของการส่งซ้ำ ซึ่งมักจะเกิดปัญหาเมื่อพบกับข้อมูลที่มีขนาดยาวและจำนวนมาก

สำหรับชั้นการทำงาน L2CAP จะให้บริการการควบคุมความผิดของข้อมูลซึ่งถูกออกแบบให้ตรวจสอบความผิดพลาดของข้อมูลที่ไม่สามารถตรวจสอบได้บนชั้นการทำงาน Baseband และร้องขอให้มีการส่งซ้ำสำหรับข้อมูลดังกล่าว

สำหรับ Broadcast Links จะใช้การส่งข้อมูลซ้ำเพื่อให้มีความน่าเชื่อถือในการส่งข้อมูล เนื่องจากไม่สามารถใช้งาน Feedback และโปรโตคอล ARQ ได้

2.1.2.2 Transport Architecture Entities



รูปที่ 7 สถาปัตยกรรมลำดับชั้นการส่งข้อมูลของเทคโนโลยีบลูทูธ [2]

สถาปัตยกรรมการส่งข้อมูลของระบบบลูทูธเป็นลำดับชั้นตั้งแต่ชั้นการทำงานด้านล่าง คือ Physical Channels ไปจนถึง L2CAP Channels

2.1.2.3 Physical Channels

Physical Channel เป็นชั้นการทำงานล่างสุดในระบบบลูทูธ ซึ่งคุณสมบัติต่างๆ ของ Physical Channel จะถูกกำหนดโดยความถี่คลื่นวิทยุร่วมกับพารามิเตอร์ต่างๆ โดยมี Frequency Hopping เพื่อผลของการรบกวนของสัญญาณ ซึ่งอุปกรณ์แต่ละตัวจะติดต่อกันโดยใช้ Physical Channel ร่วมกัน และใช้คลื่นความถี่เดียวกัน รวมถึงอยู่ในระยะการทำงานของกันและกัน และด้วยเหตุผลดังกล่าวจึงมีความเป็นไปได้ที่อุปกรณ์หลายๆตัวจะใช้คลื่นวิทยุความถี่เดียวกัน เป็นผลให้เกิดการชนกันของคลื่นความถี่ ดังนั้นเพื่อลดผลของการชนกันดังกล่าวจึงมีการกำหนด Access Code บน Physical Channel เพื่อแยกแยะอุปกรณ์แต่ละตัว โดย Access Code นี้จะเป็นคุณสมบัติของแต่ละ Physical Channel และจะเป็นส่วนแรกของแพกเก็ตข้อมูลเสมอ โดยPhysical Channel สามารถแบ่งได้เป็นหลายชนิดตามจุดประสงค์ของการใช้

2.1.2.3.1 Basic Piconet Channel

ถูกใช้เพื่อการสื่อสารกันระหว่างอุปกรณ์ที่เชื่อมต่อกันแล้วในการทำงานปกติ ซึ่ง Basic Piconet Channel จะถูกกำหนดคุณสมบัติโดย Pseudo-

Random Sequence Hopping ซึ่งจะเป็นลักษณะเฉพาะของแต่ละพิโคเน็ตและถูกกำหนดโดย Device Address ของอุปกรณ์แม่ข่าย (Master) โดยสัญญาณนาฬิกาของอุปกรณ์แม่ข่ายจะกำหนดเฟสของ Hopping Sequence ดังกล่าว และมีการแบ่งช่องสัญญาณออกเป็นช่วงเวลา (Time Slot) ที่สอดคล้องกับ Hop Frequency ของคลื่นวิทยุ ซึ่งช่วงเวลาดังกล่าวจะถูกกำหนดเลขตามสัญญาณนาฬิกาของอุปกรณ์แม่ข่าย และแพกเก็ตจะถูกส่งตามช่วงเวลาเหล่านั้น โดยอุปกรณ์แม่ข่ายจะเริ่มการส่งข้อมูลในช่วงเวลาเลขคู่เท่านั้น

ลักษณะการส่งข้อมูลของ Basic Piconet Channel เป็นการใช้ช่องสัญญาณร่วมกันระหว่างอุปกรณ์ โดยอุปกรณ์ตัวหนึ่งทำหน้าที่เป็นแม่ข่ายและอุปกรณ์ที่เหลือทำหน้าที่เป็นลูกข่าย การสื่อสารกันระหว่างอุปกรณ์ในพิโคเน็ตเป็นการสื่อสารระหว่างแม่ข่ายและลูกข่ายเท่านั้น ไม่มีการสื่อสารโดยตรงระหว่างลูกข่ายด้วยกัน

Basic Piconet Channel รองรับการทำงานของ Physical Links, Logical Transports, Logical Links และ L2CAP Channels สำหรับการติดต่อสื่อสารทั่วไป

2.1.2.3.2 Adapted Piconet Channel

Adapted Piconet Channel ต่างจาก Basic Piconet Channel สองข้อ ได้แก่ ข้อแรกคืออุปกรณ์ลูกข่ายจะใช้ความถี่เดียวกับความถี่ที่อุปกรณ์แม่ข่ายใช้ก่อนหน้า นั่นคือ ไม่มีการคำนวณความถี่ใหม่ระหว่างอุปกรณ์แม่ข่ายและอุปกรณ์ลูกข่าย ข้อแตกต่างข้อสอง คือ จำนวนคลื่นความถี่ที่ใช้จะมีจำนวนน้อยกว่า 79 ความถี่

นอกจากที่กล่าวมาแล้ว Adapted Piconet Channel มีคุณสมบัติอื่นๆ เหมือนกับ Basic Piconet Channel ทั้งหมด

2.1.2.3.3 Inquiry Scan Channel

Inquiry Scan Channel ถูกใช้เพื่อให้ค้นหาอุปกรณ์อื่นเพื่อสร้างพิโคเน็ต โดยอุปกรณ์ที่เป็น Discoverable รอรับการร้องขอ (requests) และส่งการตอบรับ (responses) ตอบกลับการร้องขอเหล่านั้น

Inquiring Scan Channel ใช้ Hopping Pattern ที่ช้ากว่า Basic Piconet Channel และ Adapted Piconet Channel โดยใช้สัญญาณนาฬิกาของอุปกรณ์ Inquiring ในการคำนวณ Hopping Pattern

อุปกรณ์จะเริ่มต้นกระบวนการ Inquiring เมื่ออุปกรณ์ Discoverable ได้รับการร้องขอจากอุปกรณ์ Inquiring เมื่อกระบวนการเสร็จสมบูรณ์อุปกรณ์ทั้งสองจะเข้าสู่ กระบวนการ Paging ต่อไป

ในกระบวนการค้นหาอุปกรณ์ (Inquiring Procedure) อุปกรณ์ทั้ง Inquiring และ Discoverable จะแลกเปลี่ยนข้อมูลกันในลักษณะ point-to-point ระหว่างการส่งแพกเก็ตระหว่างอุปกรณ์ Inquiring และ Discoverable สามารถพิจารณาได้ว่ามี Physical Link ชั่วคราวเกิดขึ้นระหว่างอุปกรณ์เหล่านั้น

2.1.2.3.4 Page Scan Channel

Page Scan Channel ถูกใช้เพื่อสร้างพิโคเน็ต โดยอุปกรณ์เรียกว่า Connectable จะรอรับ Page Request จากอุปกรณ์และแลกเปลี่ยนข้อมูลระหว่างกันหลังจากได้รับ Page Request แล้ว

Page Scan Channel ใช้ Hopping Pattern ที่ช้ากว่า Basic Piconet Channel และ Adapted Piconet Channel โดยใช้สัญญาณนาฬิกาของอุปกรณ์ Paging ในการคำนวณ Hopping Pattern

อุปกรณ์จะเริ่มต้นกระบวนการ Paging เมื่ออุปกรณ์ Connectable ได้รับการร้องขอจากอุปกรณ์ Paging เมื่อกระบวนการเสร็จสมบูรณ์อุปกรณ์ทั้งสองจะเข้าสู่ Basic Piconet Channel โดยอุปกรณ์ Paging ทำหน้าที่เป็นแม่ข่ายของพิโคเน็ต

ในกระบวนการ Paging (Paging Procedure) อุปกรณ์ทั้ง Paging และ Connectable จะแลกเปลี่ยนข้อมูลกันในลักษณะ point-to-point ในระหว่างการส่งแพกเก็ตระหว่างอุปกรณ์ Paging และ Connectable สามารถพิจารณาได้ว่ามี Physical Link ชั่วคราวเกิดขึ้นระหว่างอุปกรณ์เหล่านั้น

2.1.2.4 Physical Links

Physical Links คือการเชื่อมต่อกันในชั้นการทำงาน Baseband ระหว่างอุปกรณ์บลูทูธ ซึ่งทำงานร่วมกับ Physical Channel เดียว

จริงๆแล้ว Physical Link ไม่มีโครงสร้างจริงในระบบ ฟิลด์ต่างๆในแพกเก็ตไม่ว่าจะเป็น Access Code รวมถึง สัญญาณนาฬิกาและ Address ของอุปกรณ์แม่ข่าย ถูกใช้เพื่อระบุถึง Physical Channel นั่นคือไม่มีส่วนใดในแพกเก็ตเลยที่ระบุถึง Physical Link แต่จะสามารถที่จะอ้างอิงถึง Physical Link ได้โดยอาศัย Logical Transport

2.1.2.5 Logical Links and Logical Transports

Logical Links ทำงานร่วมกับ Logical Transports เพื่อรองรับการทำงานของแอพพลิเคชันต่างๆ โดยมีคุณสมบัติต่างๆ ได้แก่ Flow Control, กระบวนการ Acknowledgement หรือ Repeat, การกำหนดลำดับและการกำหนดตารางการทำงาน

Logical Transport	**Logical Link**	**Physical Channel**	**Overview**
Asynchronous Connection-Oriented (ACL)	Control (LMP) ACL-C User (L2CAP) ACL-U	Active Physical Link Basic or Adapted Physical Channel	- มีความน่าเชื่อถือในการส่งข้อมูล - สื่อสารแบบสองทิศทาง - ติดต่อกันแบบ Point-to-point
Synchronous Connection-Oriented (SCO)	Stream (unframed) SCO-S	Active Physical Link Basic or Adapted Physical Channel	- สื่อสารแบบสองทิศทาง - ติดต่อกันแบบ Point-to-point - ใช้สำหรับส่งข้อมูลคงที่ความเร็ว 64 kbps
Extended Synchronous Connection-Oriented (eSCO)	Stream (unframed) eSCO-S	Active Physical Link Basic or Adapted Physical Channel	- สื่อสารแบบสองทิศทาง - ติดต่อกันแบบ Point-to-point - กำหนดจำนวนการส่งข้อมูลซ้ำ - ใช้สำหรับส่งข้อมูลความเร็วคงที่โดยอ้างอิงกับสัญญาณ

			นาฬิกาของแม่ข่าย
Active Slave Broadcast (ASB)	User (L2CAP) ASB-U	Active Physical Link Basic or Adapted Physical Channel	- ไม่มีความน่าเชื่อมถือในการส่งข้อมูล - สื่อสารแบบทิศทางเดียว - กระจายข้อมูลไปยังกลุ่มผู้ใช้ L2CAP
Parked Slave Broadcast (PSB)	Control (LMP) PSB-C User (L2CAP) PSB-U	Parked Physical Link Basic or Adapted Physical Channel	- ไม่มีความน่าเชื่อมถือในการส่งข้อมูล - สื่อสารแบบทิศทางเดียว - ใช้สำหรับส่ง LMP และ L2CAP ไปยังอุปกรณ์ที่อยู่ในโหมด Parked

ตารางที่ 1 ชนิดของ Logical Transport [2]

ตารางที่ 1 แสดงถึงชนิดของ Logical Transport, ชนิดของ Logical Link และ Physical Channel ที่รองรับและจุดประสงค์ของ Logical Transport ชนิดต่างๆ

2.1.2.5.1 Logical Transports

2.1.2.5.1.1. Asynchronous Connection-Oriented (ACL)

ACL Logical Transport จะถูกใช้สำหรับส่งสัญญาณควบคุม LM และ L2CAP และส่งข้อมูลของผู้ใช้ โดยใช้ 1-bit ARQN/SEQN เพื่อเพิ่มความน่าเชื่อถือในการส่งข้อมูลไปยังลูกข่ายที่ Active ในพิโคเน็ต ซึ่งแต่ละพิโคเน็ตสมาชิกในพิโคเน็ตจะมี ACL Logical Transport ไปยังแม่ข่ายเรียกว่า ACL เริ่มต้น (Default ACL)

ACL เริ่มต้นจะถูกสร้างระหว่างแม่ข่ายและลูกข่ายเมื่ออุปกรณ์เข้ามาเป็นสมาชิกของพิโคเน็ต ซึ่งเชื่อมต่อกันด้วย Basic Piconet Physical Channel โดย ACL เริ่มต้นจะกำหนดค่า Logical Transport Address (LT_ADDR) โดยแม่ข่ายของพิโคเน็ต ซึ่งค่า LT_ADDR นี้จะถูกใช้เพื่อระบุถึง Active Physical Link

เนื่องจากเหตุผลด้านความสอดคล้องของ Bluetooth Specification ที่สามารถใช้ ACL เริ่มต้นซ้ำเพื่อส่งข้อมูลของ SCO Logical Transport ได้ จึงต้องใช้ชนิดของแพกเก็ตร่วมกับ LT_ADDR เพื่อระบุถึง ACL Logical Transport เพราะ Logical Transport ทั้งสองชนิดใช้แพกเก็ตต่างชนิดกัน

ACL เริ่มต้นสามารถถูกใช้เพื่อส่งข้อมูลชนิด Isochronous ได้ โดยกำหนดให้ทำการ Flush แพกเก็ตที่หมดอายุทั้งหมด

ถ้าหาก ACL เริ่มต้นถูกกำจัดออกไปจาก Active Physical Link แล้ว Logical Transport อื่นๆที่เกิดขึ้นระหว่างแม่ข่ายและลูกข่ายจะถูกกำจัดออกไปด้วย และในกรณีที่เกิดการสูญเสียการเชื่อมโยง Physical Channel ของพิโคเน็ต Physical Link, Logical Links และ Logical Transports จะถูกยกเลิกไปทันที

การยกเลิก ACL เริ่มต้นโดยที่ยังคงเชื่อมโยงสัญญาณกับพิโคเน็ตอยู่สามารถทำได้โดยกระบวนการที่เรียกว่า “Parking”

เมื่ออุปกรณ์เข้าสู่สถานะ Parked ACL Logical Link ที่ทำงานอยู่บน ACL Logical Transport เริ่มต้นจะถูกระงับการทำงาน เมื่ออุปกรณ์กลับเข้าสู่สถานะที่ Active ACL Logical Transport ใหม่จะถูกสร้างขึ้นและ ACL Logical Link ที่ถูกระงับการทำงานไว้ก็จะถูกรวมเข้ากับ ACL Logical Transport อีกครั้ง

2.1.2.5.1.2. Synchronous Connection-Oriented (SCO)

SCO เป็น Logical Transport ที่มีลักษณะเป็นแบบสมมาตรและมีการส่งข้อมูลแบบ point-to-point ระหว่างแม่ข่ายกับลูกข่าย SCO Logical Transport ส่งข้อมูลด้วยความเร็ว 64 kbps ที่สอดคล้องกับสัญญาณนาฬิกาของพิโคเน็ต ซึ่งโดยปกติแล้วข้อมูลเหล่านี้จะเป็นข้อมูลเสียงที่ผ่านการเข้ารหัสแล้ว และ SCO รองรับทนทานของการส่งข้อมูล ดีเลย์และการใช้ความเร็วในการส่งข้อมูล โดยสามารถมี SCO ได้พร้อมกัน 3 SCO

สำหรับแต่ละ SCO-S Logical Link จะทำงานบน SCO Logical Transport หนึ่งซึ่งกำหนดค่า LT_ADDR เดียวกันกับค่า LT_ADDR ของ ACL Logical Transport เริ่มต้นระหว่างอุปกรณ์ ดังนั้น LT_ADDR จึงไม่เพียงพอในการระบุเป้าหมายของการส่งข้อมูล ทำให้ต้องใช้ LT_ADDR

ร่วมกับ Slot Number ซึ่งเป็นคุณสมบัติของ Physical Channel ร่วมกันชนิดของแพกเก็ตเพื่อระบุการส่งข้อมูลบน SCO Link

2.1.2.5.1.3. Extended Synchronous Connection-Oriented (eSCO)

eSCO เป็น Logical Transport ที่มีลักษณะเป็นแบบไม่สมมาตรและมีการส่งข้อมูลแบบ point-to-point ระหว่างแม่ข่ายกับลูกข่าย eSCO นำเสนอความสามารถที่เพิ่มเติมจาก SCO มาตรฐาน

สำหรับ eSCO-S Logical Link แต่ละอัน จะทำงานบน eSCO Logical Transport เดียวโดยกำหนด LT_ADDR เฉพาะสำหรับพิโคเน็ต โดยจะถูกสร้างโดยใช้สัญญาณ LM และทำตามการจัดสรรตารางเวลาเช่นเดียวกับ SCO-S

2.1.2.5.1.4. Active Slave Broadcast (ASB)

ASB เป็น Logical Transport ที่ถูกใช้สำหรับการส่งข้อมูล L2CAP ของผู้ใช้ไปยังสมาชิกทุกตัวในพิโคเน็ตซึ่งใช้ Physical Channel ร่วมกัน ASB Logical Transport ไม่มีโปรโตคอล Acknowledgement และมีการสื่อสารแบบทางเดียวจากแม่ข่ายไปยังลูกข่าย และเนื่องจากไม่มีการใช้งานโปรโตคอล Acknowledgement จึงเป็นการสื่อสารที่ไม่น่าเชื่อถือ ทำให้มีการใช้การส่งข้อมูลซ้ำเพื่อเพิ่มความน่าเชื่อถือในการส่งข้อมูล

การใช้งาน ASB Logical Transport ทำได้โดยการกำหนดค่า LT_ADDR เช่นเดียวกับ PSB Logical Transport นอกจากนี้ ASB จะไม่ถูกใช้เพื่อส่ง LMP หรือสัญญาณควบคุมของ L2CAP

2.1.2.5.1.5. Parked Slave Broadcast (PSB)

PSB เป็น Logical Transport ที่ถูกใช้สำหรับการสื่อสารระหว่างแม่ข่ายและลูกข่ายในโหมด Park ซึ่งหมายถึงโหมดที่ยกเลิกการทำงานของ ACL Logical Transport เริ่มต้น

PSB เป็น Logical Transport ที่ซับซ้อนกว่า Logical Transport อื่นๆ เพราะประกอบด้วยเฟสการทำงานต่างๆที่มีจุดประสงค์ต่างๆกัน โดยเฟสการทำงานเหล่านั้นได้แก่ เฟสข้อมูลควบคุม (Control Information Phase) ซึ่งใช้ส่งข้อมูล LMP Logical Link, เฟสข้อมูลผู้ใช้ (User

Information Phase) ซึ่งใช้ส่งข้อมูล L2CAP Logical Link และเฟสการเข้าถึง (Access Phase) ซึ่งใช้ส่งสัญญาณของชั้นการทำงาน Baseband โดยในช่วงเวลาหนึ่งจะมีเฟสเดียวเท่านั้นที่สามารถทำงานได้

PSB Logical Transport จะถูกใช้งานโดยกำหนดค่า LT_ADDR เป็น 0 เช่นเดียวกับ ASB Logical Transport โดยที่ลูกข่ายจะไม่สับสนกับการใช้งาน LT_ADDR เพราะอุปกรณ์ลูกข่ายเหล่านั้นจะอยู่ในพิโคเน็ตซึ่งใช้งาน PSB Logical Transport อยู่

2.1.2.5.2 Logical Links

เนื่องจาก Logical Transport สามารถรองรับการทำงานของ Logical Links ชนิดต่างๆพร้อมกันได้ Logical Link จึงมี Logical Link Identifier (LLID) ใน Header ของข้อมูลของ Baseband แพกเก็ต

2.1.2.5.2.1. ACL-C

ACL Control Logical Link (ACL-C) ถูกใช้เพื่อส่งสัญญาณ LMP (LMP Signaling) ระหว่างอุปกรณ์ในพิโคเน็ต ซึ่ง ACL-C Logical Link จะมีความสำคัญมากกว่า ACL-U Logical Link เสมอ

2.1.2.5.2.2. ACL-U

User Asynchronous/Isochronous Logical Link (ACL-U) ถูกใช้เพื่อส่งเฟรมข้อมูลชนิด Asynchronous และ Isochronous สำหรับแพกเก็ตของ ACL-U Link จะมี LLID สองค่า โดยค่าหนึ่งบอกจุดเริ่มต้นของเฟรมข้อมูลและอีกค่าหนึ่งบอกความต่อเนื่องจากเฟรมก่อนหน้า ซึ่งค่า LLID นี้จะช่วยในด้านความถูกต้องของการรวมข้อมูล

2.1.2.5.2.3. SCO-S/eSCO-S

Synchronous (SCO-S) และ Extended Synchronous (eSCO-S) Logical Links ถูกใช้สำหรับข้อมูลชนิด Isochronous ซึ่งเป็นการส่งข้อมูลแบบ Stream ซึ่ง Logical Links เหล่านี้จะทำงานร่วมกับ Logical Transport หนึ่งที่ไม่มี LLID บนแพกเก็ตและกำหนดขนาดและความเร็วในการส่งข้อมูลที่คงที่ รวมถึงทำงานรวมกับ Logical Link หนึ่งซึ่งกำหนดขนาดของแพกเก็ตและช่วงเวลาในการส่งข้อมูลเช่นเดียวกัน

สำหรับข้อมูลชนิด Isochronous ชนิด Variable Rate ซึ่งไม่สามารถถูกส่งบน SCO-S หรือ eSCO-S Logical Links ได้ จะถูกส่งบน ACL-U Logical Links แทน

2.1.2.6 L2CAP Channels

L2CAP ให้บริการการ Multiplexing เพื่อให้แอพพลิชันหลายชนิดใช้งาน ACL-U Logical Link ร่วมกัน โดยแอพพลิเคชันและโปรโตคอลต่างๆจะติดต่อกับ L2CAP โดยใช้การติดต่อชนิด Channel-Oriented เพื่อสร้างการเชื่อมต่อกับชั้นการทำงานเดียวกันที่อุปกรณ์อื่น โดยมี Channel Identifier (CID) ระบุถึงอุปกรณ์ที่จะทำการติดต่อด้วย ซึ่ง CID นี้จะถูกกำหนดโดย L2CAP และอุปกรณ์แต่ละตัวก็จะมี CID ต่างกัน สำหรับข้อกำหนด QoS จะถูกกำหนดตามชนิดของแอพพลิเคชัน

นอกจากหน้าที่หลักในการทำ Multiplexing ข้อมูลแล้ว L2CAP ยังทำหน้าที่ควบคุมการส่งข้อมูลในชั้นการทำงาน L2CAP ระหว่างอุปกรณ์ ซึ่งจะถูกกำหนดในการสร้างช่องสัญญาณเพื่อความน่าเชื่อถือในการส่งข้อมูลโดยการตรวจสอบความผิดพลาดของข้อมูลและการส่งข้อมูลซ้ำ นอกจากนี้ ในกรณีที่มีการทำงานร่วมกับ HCI L2CAP จะทำการแบ่งข้อมูล (L2CAP SDUs) เพื่อให้เหมาะกับขนาดบัฟเฟอร์ของชั้นการทำงาน Baseband และเพื่อทำตามกระบวนการทำงานของ HCI ด้วย

2.1.3 **รูปแบบการสื่อสารของ** (Communication Topology)

2.1.3.1 Piconet Topology

พิโคเน็ตเป็นการสร้างรูปแบบการสื่อสารไร้สายของเทคโนโลยีบลูทูธ ประกอบด้วยอุปกรณ์ตั้งแต่ 2 ตัวขึ้นไปที่ใช้ Physical Channel เดียวกัน ซึ่งหมายถึงการใช้สัญญาณนาฬิกาและ Hopping Sequence ร่วมกัน โดยสัญญาณนาฬิกานั้นคือสัญญาณนาฬิกาของอุปกรณ์แม่ข่าย สำหรับ Hopping Sequence นั้นได้รับมาจากสัญญาณนาฬิกา Address ของอุปกรณ์แม่ข่าย

ในเวลาเดียวกันอุปกรณ์บลูทูธสามารถเป็นสมาชิกของพิโคเน็ตได้มากกว่าหนึ่งพิโคเน็ต เพราะทำงานด้วยหลักการ Time Division Multiplexing แต่อุปกรณ์หนึ่งตัวไม่สามารถเป็นแม่ข่าย (master) ได้มากกว่าหนึ่งพิโคเน็ต เพราะ อุปกรณ์ในพิโคเน็ตจะอ้างอิงสัญญาณนาฬิกาของอุปกรณ์แม่ข่าย (master) จึงเป็นไปไม่ได้ที่จะสามารถเป็น

อุปกรณ์แม่ข่ายได้มากกว่าหนึ่งพิโคเน็ต แต่อุปกรณ์หนึ่งตัวสามารถเป็นลูกข่ายหรือเป็นสมาชิกของหลายพิโคเน็ตได้ โดยเหตุการณ์ดังกล่าวเรียกว่า “Scatternet”

2.1.3.2 Operation Procedures and Modes

ระบบบลูทูธมีโหมดการทำงานต่างๆสำหรับใช้ในการส่งข้อมูลกันระหว่างอุปกรณ์บลูทูธในพิโคเน็ต และเนื่องจากกระบวนการและโหมดการทำงานต่างๆเกิดขึ้นในชั้นการทำงานต่างๆของสถาปัตยกรรมของบลูทูธ อุปกรณ์จึงสามารถอยู่ในโหมดหรือกระบวนการทำงานต่างๆเหล่านี้ได้พร้อมกันในเวลาเดียวกัน

2.1.3.2.1 Inquiring (Discovering) Procedure

กระบวนการ Inquiry เป็นกระบวนการที่ทำให้อุปกรณ์บลูทูธค้นพบอุปกรณ์บลูทูธอื่น หรือสามารถถูกค้นพบได้โดยอุปกรณ์อื่นในบริเวณนั้น สำหรับกระบวน Inquiry นี้เป็นกระบวนการแบบไม่สมมาตร (Asymmetrical)

อุปกรณ์ที่ทำหน้าที่ค้นหาอุปกรณ์อื่นเรียกว่า “Inquiring Device” โดยจะส่ง “Inquiry Requests” ไปยังอุปกรณ์ที่สามารถถูกค้นพบได้โดยอุปกรณ์อื่น “Discoverable Device” ซึ่งจะทำหน้าที่คอยรับ “Inquiry Requests” จาก “Inquiring Device” และส่ง “Responds” กลับไป ซึ่งการรับส่ง “Inquiry Requests” และ “Responses” ดังกล่าว จะใช้ Physical Channel พิเศษสำหรับการรับส่งดังกล่าวโดยเฉพาะ

ในกรณีที่อุปกรณ์ทั้ง “Inquiring Device” และ “Discoverable Device” เชื่อมต่อกับอุปกรณ์อื่นในพิโคเน็ตอยู่แล้ว “Inquiry Physical Channel” จะทำหน้าที่จัดการควบคุมการใช้ช่องสัญญาณดังกล่าวโดยไม่กระทบกับคุณภาพการส่งข้อมูลของ Logical Transports ที่เกิดขึ้นก่อนหน้า

2.1.3.2.2 Paging (Connecting) Procedure

กระบวนการนี้เป็นการสร้างการเชื่อมต่อระหว่างอุปกรณ์ เป็นกระบวนการแบบไม่สมมาตร (Asymmetrical) โดยอุปกรณ์หนึ่งเรียกว่า “Paging Device” ทำงานในกระบวนการ “Page Procedure” คือการส่งแพกเก็ตไปยังอุปกรณ์อีกตัว เรียกว่า “Connectable Device” จะทำงานในกระบวนกวนการ “Page Scanning” ซึ่งใช้งาน Physical Channel พิเศษที่เหมาะสำหรับการทำงานสร้างการเชื่อมต่อระหว่างอุปกรณ์

2.1.3.2.3 Connected Mode

หลังจากเสร็จสิ้นกระบวนการต่างๆ อุปกรณ์จะแป็นสมาชิกของพิโคเน็ต นั่นคือมีการสร้าง Physical Channel สำหรับการสื่อสาร มี Physical Link ระหว่างอุปกรณ์ และมี ACL-C และ ACL-U Logical Link เริ่มต้น

2.1.3.2.4 Hold Mode

เมื่ออยู่ในโหมดนี้บลูทูธจะสงวนการทำงานไว้ให้กับ Synchronous Links ได้แก่ SCO และ eSCO และไม่รองรับการทำงานของ Asynchronous Links เลย โดยอุปกรณ์จะเข้าสู่โหมดการทำงานนี้เมื่อมีการร้องขอ และเมื่อการทำงานเสร็จสมบูรณ์อุปกรณ์จะกลับเข้าสู่โหมดการทำงานเดิม

2.1.3.2.5 Sniff Mode

โหมด Sniff จะรองรับการทำงานของอุปกรณ์เพื่อลดการใช้พลังงานเข้าร่วมกับ Physical Channel อื่น โดยกำหนด Duty Cycle ของ ACL Logical Transport โดยไม่เกี่ยวข้องกับ SCO หรือ eSCO Logical Trasport

2.1.3.2.6 Parked State

สถานะ Parked เป็นสถานะที่อุปกรณ์ต่างๆยังคงเป็นสมาชิกของพิโคเน็ต แต่ไม่รองรับการทำงานของ Logical Links ต่างๆ ยกเว้น PSB-C และ PSB-U ซึ่งถูกใช้เพื่อสื่อสารกันระหว่างอุปกรณ์แม่ข่าย (master) และอุปกรณ์ลูกข่าย (slave) นั่นหมายถึงมีข้อจำกัดในการส่งข้อมูลซึ่งกำหนดโดยค่าพารามิเตอร์ต่างๆ ของ PSB Logical Transport

2.1.3.2.7 Role Switch Procedure

กระบวนการนี้เป็นกระบวรการสำหรับสลับหน้าที่ของอุปกรณ์ในพิโคเน็ต โดยจะเป็นการใช้ Physical Channel ที่อ้างอิงจากอุปกรณ์แม่ข่าย (master) ตัวใหม่แทน Physical Channel ที่อ้างอิงจากอุปกรณ์แม่ข่าย (master) ตัวเดิม หลังจากกระบวนการเสร็จสิ้นอาจถูกยกเลิกไป หรือยังคงอยู่หากยังมีอุปกรณ์ลูกข่าย (slaves) ยังคงเชื่อมต่อกับ Physical Channel ดังกล่าวอยู่

นอกจากนี้ กระบวนการนี้ยังทำงานยกเลิก Logical Transports ต่างๆ แต่ยังคงเก็บ ACL Logical Links เพื่อรองรับการทำงานของ Physical Channel

ใหม่ สำหรับโหมดการทำงานต่างๆ เช่น โหมด Sniff จะต้องทำการติดต่อเพื่อสร้างการทำงานใหม่

2.1.3.2.8 Enhance Data Rate

Enhanced Data Rate (EDR) เป็นวิธีการเพิ่มความเร็วในการส่งข้อมูล รวมถึงพัฒนาความสามารถในการรองรับการติดต่อกับอุปกรณ์จำนวนมาก และลดการใช้พลังงานลง โดยเปลี่ยนแปลงความจุและชนิดของแพกเก็ตโดยไม่เปลี่ยนแปลงส่วนที่เหลือของสถาปัตยกรรม

EDR อาจถูกเลือกเป็นโหมดการทำงานหนึ่ง ซึ่งทำงานอิสระบน Logical Transports ต่างๆ ได้แก่ ACL-U และ eSCO-S Logical Transports แต่จะไม่สามารถทำงานได้บน ACL-C, SCO-S และ Broadcast Logical Transport

2.1.4 **บลูทูธโปรไฟล์** (Profile Overview)

บลูทูธโปรไฟล์เป็นตัวกำหนดแอพพลิเคชันและความสามารถต่างๆที่อุปกรณ์บลูทูธสามารถใช้ในการสื่อสารกับอุปกรณ์บลูทูธอื่น ซึ่งบลูทูธโปรไฟล์มีหลากหลายตามชนิดของแอพพลิเคชันสำหรับแต่ละอุปกรณ์ ซึ่งทำให้ผู้พัฒนาสามารถพัฒนาแอพพลิเคชันที่ต้องการเพื่อทำงานกับอุปกรณ์บลูทูธได้โดยอ้างอิงจากโปรไฟล์เหล่านี้

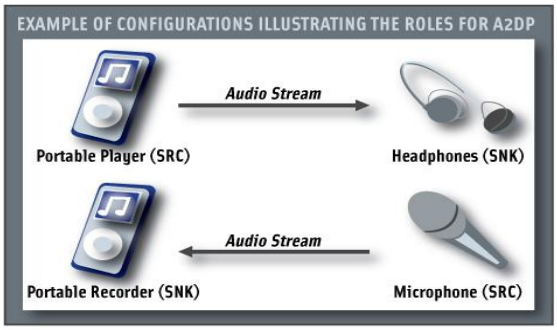
บลูทูธโปรไฟล์มีคุณสมบัติหรือข้อมูลพื้นฐาน ดังนี้

1. ทำงานร่วมกับโปรไฟล์อื่นๆ
2. แนะนำรูปแบบอินเตอร์เฟสสำหรับผู้ใช้
3. การทำงานของโปรไฟล์บนบลูทูธโปรโตคอล จะอาศัยการทำงานเฉพาะซึ่งทำงานให้กับโปรไฟล์นั้น โดยส่วนการทำงานดังกล่าวมีออปชั่นและพารามิเตอร์สำหรับแต่ละชั้นการทำงาน

สำหรับบลูทูธโปรไฟล์ที่จะนำเสนอในเอกสารฉบับนี้ ประกอบด้วยโปรไฟล์ต่างๆที่เกี่ยวข้องกับการส่งข้อมูลเสียงผ่านอุปกรณ์บลูทูธซึ่งมีการใช้งานอยู่ในปัจจุบัน

2.1.4.1 A2DP (Advance Audio Distribution Profile)

A2DP เป็นบลูทูธโปรไฟล์ซึ่งรองรับการทำงานส่งข้อมูลเสียงคุณภาพสูงระดับสเตอริโอ (stereo-quality audio) ผ่านอุปกรณ์บลูทูธ



รูปที่ 8 รูปแบบการใช้งานโปรไฟล์ A2DP [4]

รูปที่ 8 แสดงการทำงานของเครื่องเล่นเพลงบลูทูธ ซึ่งได้แก่ MP3 player, Walkman และสเตอริโอ เป็นต้น ซึ่งเครื่องเล่นทำหน้าที่เป็นแหล่งกำเนิดเสียง (audio source) และ wireless headset หรือ wireless stereo speakers ทำหน้าที่เป็นตัวรับเสียง (audio sink)

ตัวอย่างผลิตภัณฑ์หรืออุปกรณ์บลูทูธซึ่งใช้โปรไฟล์ A2DP ได้แก่

Stereo Headphones

Stereo Speakers

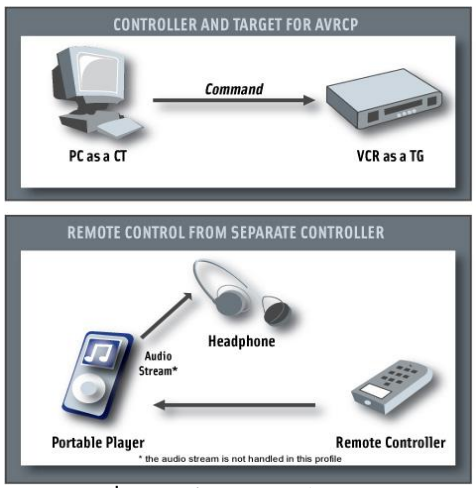
MP3 Player

Music Phones

Stereo Adapters

2.1.4.2 AVRCP (A/V Remote Control Profile)

AVRCP ถูกออกแบบมาให้รองรับอินเตอร์เฟสมาตราฐานเพื่อควบคุมการทำงานของโทรทัศน์, อุปกรณ์ hi-fi หรืออุปกรณ์อื่นซึ่งมีการควบคุมโดยรีโมตคอนโทรล และสามารถทำงานร่วมกับกับโปรไฟล์ A2DP หรือ VDP ได้



รูปที่ 9 การใช้งานโปรไฟล์ AVRCP [4]

โดยพื้นฐานของการใช้งานรีโมตคอนโทรลแล้ว ผู้ใช้สามารถปรับเปลี่ยนฟังก์ชันต่างๆตามเมนูซึ่งได้แก่ การปรับความเข้มแสงของโทรทัศน์ หรือตั้งเวลา รวมถึงฟังก์ชันเกี่ยวกับเสียงเช่น การเพิ่มลดระดับเสียง การสั่งเล่น การสั่งหยุด เป็นต้น

ตัวอย่างผลิตภัณฑ์หรืออุปกรณ์ที่ใช้โปรไฟล์ AVRCP ได้แก่

อุปกรณ์ควบคุม

Personal Computer

PDA

โทรศัพท์เคลื่อนที่

Remote Controller

อุปกรณ์เกี่ยวกับเสียงหรือวิดีโอ ได้แก่ Headphones, Player, Recorder

อุปกรณ์เป้าหมาย

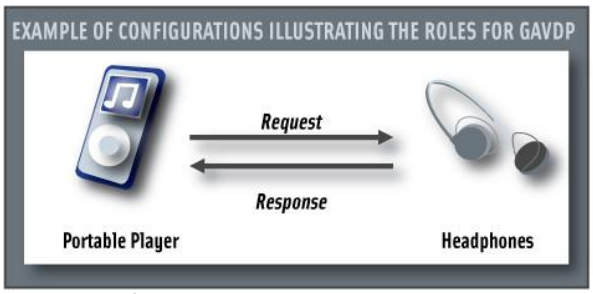
เครื่องเล่นวิดีโอ

โทรทัศน์

Tuner

2.1.4.3 GAVDP (Generic A/V Distribution Profile)

GAVDP รองรับการทำงานพื้นฐานสำหรับโปรไฟล์ A2DP และ VDP ซึ่งเป็นการทำงานพื้นฐานของระบบซึ่งถูกออกแบบมาเพื่อการส่งข้อมูลวิดีโอหรือเสียงด้วยเทคโนโลยีบลูทูธ



รูปที่ 10 **รูปแบบการใช้งานโปรไฟล์** GAVDP [4]

โดยปกติแล้วโปรไฟล์ GAVDP มักใช้งานกับอุปกรณ์ประเภท wireless stereo headphones และเครื่องเล่นต่างๆ เช่น MP3 player หรือ walkman ซึ่งเครื่องเล่นจะส่งข้อความไปยัง headphones เพื่อสร้างการเชื่อมต่อหรือแม้แต่การขอปรับการรับส่งข้อมูลเสียงโดย headphones เอง

ตัวอย่างผลิตภัณฑ์หรืออุปกรณ์ที่ใช้โปรไฟล์ GAVDP

Music Player

Stereo Headphones

Stereo Speakers

Laptop

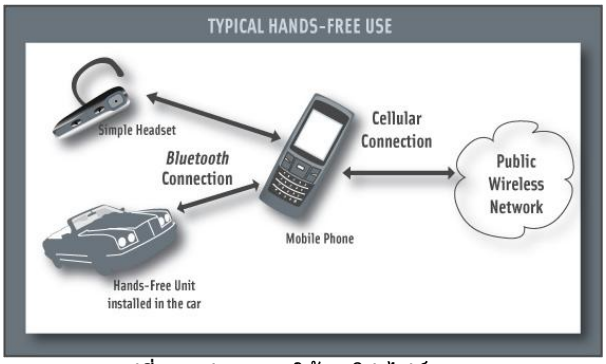
คอมพิวเตอร์ตั้งโต๊ะ

โทรศัพท์เคลื่อนที่

PDA

2.1.4.4 HFP (Hand-Free Profile)

HFP อธิบายถึงการทำงานของอุปกรณ์ที่ทำหน้าที่เป็น Gateway ซึ่งถูกใช้วางและรับสายจากอุปกรณ์แฮนด์ฟรี (Hand-Free Device)



รูปที่ 11 **รูปแบบการใช้งานโปรไฟล์** HFP [4]

ตัวอย่างเหตุการณ์นี้ คือ การใช้อุปกรณ์เสริมต่างๆภายในรถยนต์ ซึ่งอุปกรณ์เหล่านั้นจะติดต่อสื่อสารกับโทรศัพท์เคลื่อนที่ของผู้ใช้ และถูกใช้เพื่อรับและวางสายโทรศัพท์ ได้แก่ Headset ชนิดไร้สาย เป็นต้น

ตัวอย่างผลิตภัณฑ์หรืออุปกรณ์ที่ใช้โปรไฟล์ HFP

รถยนต์

อุปกรณ์เสริมภายในรถยนต์

ระบบ GPS

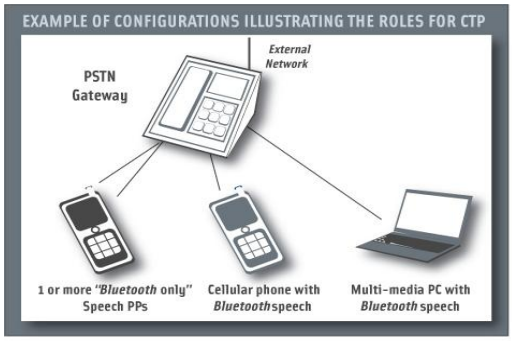
Headset

โทรศัพท์เคลื่อนที่

PDA

2.1.4.5 CTP (Cordless Telephony Profile)

CTP อธิบายถึงการทำงานของโทรศัพท์ไร้สายที่ใช้เทคโนโลยีบลูทูธ



รูปที่ 12 **รูปแบบการใช้งานโปรไฟล์** CTP [4]

สามารถใช้งานได้กับทั้งโทรศัพท์ไร้สายและโทรศัพท์เคลื่อนที่ซึ่งทำหน้าที่เป็นโทรศัพท์ไร้สายที่สามารถเชื่อมโยงกับ Base Station ที่มีการใช้งาน CTP โดยมีอุปกรณ์ซึ่งทำหน้าที่เป็น CTP Gateway เชื่อมโยงโทรศัพท์บ้านและระบบเครือข่ายโทรศัพท์ภายในบ้าน

ตัวอย่างผลิตภัณฑ์หรืออุปกรณ์ที่ใช้งานโปรไฟล์ CTP

Laptop

คอมพิวเตอร์ตั้งโต๊ะ

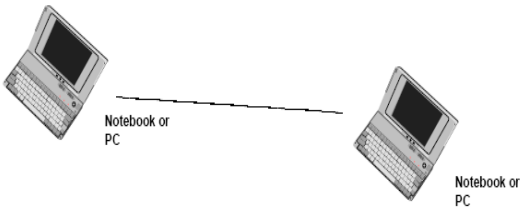
โทรศัพท์ไร้สาย

โทรศัพท์เคลื่อนที่

PDA

2.1.4.6 SPP (Serial Port Profile)

SPP เชื่อมต่ออุปกรณ์บลูทูธด้วยการสร้างและใช้งาน Serial Port เสมือน



รูปที่ 13 **รูปแบบการใช้งานโปรไฟล์** SPP [4]

ตัวอย่างผลิตภัณฑ์หรืออุปกรณ์ที่ใช้งานโปรไฟล์ CTP

Laptop

เครื่องคอมพิวเตอร์ตั้งโต๊ะ

2.1.5 **การเปรียบเทียบเชิงเทคนิค** (Technical Comparison)

ตารางเปรียบเทียบข้อมูลเชิงเทคนิคระหว่างเทคโนโลยีบลูทูธและเทคโนโลยีสื่อสารไร้สายต่างๆ ซึ่งจะเห็นได้ว่า เทคโนโลยีบลูทูธมีความเร็วมากพอที่จะส่งข้อมูลเสียง คือ 1-3 Mbps และมีราคาถูกคือ $3 เมื่อเทียบกับเทคโนโลยีอื่นๆ

		ZigBee	**Bluetooth**	**802.11b**	**802.11g**	**802.11a**	**802.11n**	**UWB**
Throughput	**Mbps.**	0.03	1-3	11	54	54	200	200
Max Range	**Ft.**	75	30	200	200	150	150	30
Sweet-pot	**Mbps-ft**	0.03@75	1-3@10	2@200	2@200	36@100	100@100	200@10
Service	**Bps-ft2**	530	314M	251G	251G	1.13T	3.14T	62G
Power	**mW**	30	100	750	1000	1500	2000	400
BW	**mHz**	0.6	1	22	20	20	40	500
Spectral Efficiency	**b/Hz**	0.05	1	0.5	2.7	2.7	5	0.4
Power Efficiency 1	**mW/Mbps**	1000	100	68	19	27	10	2
Power Efficiency 2	**mAh/GB**	2211	67	46	12	18	7	1.3
TTGB	**Time**	3.1 day	2.2 hr	12 min	2.5 min	2.5 min	40 sec	40 sec
Price	**US$**	$2	$3	$5	$9	$12	$20	$7

ตารางที่ 2 ตารางเปรียบเทียบเชิงเทคนิค [4]

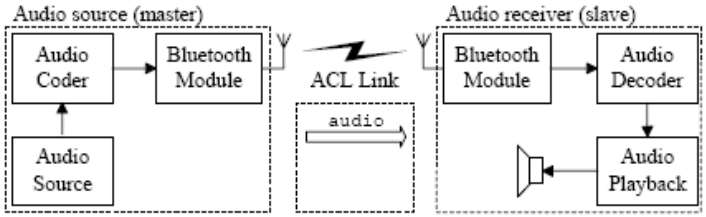
2.2 งานวิจัยที่เกี่ยวข้อง

งานวิจัย [6] ได้นำเสนอข้อสรุปเกี่ยวกับการกระจายข้อมูลด้วยเทคโนโลยีบลูทูธจากการพัฒนาระบบ Bluecasting ชื่อ Baloo รวมถึงวิเคราะห์และนำเสนอวิธีแก้ปัญหาที่เกิดขึ้นกับ Baloo ซึ่งมีหลักการทำงานในลักษณะของ Bluecasting Server กระจายข่าวสารและโฆษณาไปยังอุปกรณ์บลูทูธ

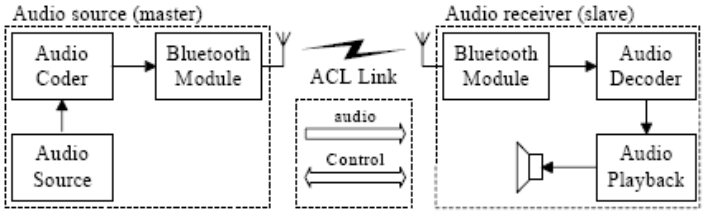
ข้อสรุปของงานวิจัยเป็นการตอบคำถามของงานวิจัย ดังนี้

1. เทคโนโลยีบลูทูธไม่ได้ถูกออกแบบมาสำหรับการกระจายข้อมูล ทำให้มีข้อจำกัดในแง่ของความเร็วในการส่งข้อมูล เพราะความเร็วในการส่งข้อมูลจะลดลงเมื่อมีจำนวนอุปกรณ์ลูกข่ายในพิโคเน็ตมากขึ้น นอกจากนี้ยังมีข้อจำกัดในเรื่องของจำนวนเครื่องลูกข่ายในพิโคเน็ตซึ่งมีจำนวนลูกข่ายสูงสุดได้เพียง 7 ตัวเท่านั้น เวลาในการค้นหาอุปกรณ์เพื่อสร้างพิโคเน็ต และขนาดของไฟล์ที่จะทำการส่ง แต่ข้อจำกัดดังกล่าวอาจมีปัญหาน้อยลง เนื่องด้วยเวอร์ชันใหม่ของเทคโนโลยีบลูทูธซึ่งจะมีความเร็วในการส่งข้อมูลมากขึ้น และใช้เวลาในการค้นหาอุปกรณ์เพื่อสร้างพิโคเน็ตที่น้อยลง
2. ด้วยเทคโนโลยีบลูทูธแล้ว ชนิดของข้อมูลไม่เป็นปัญหาสำหรับการส่งข้อมูล ปัญหานี้จะเป็นข้อจำกัดของอุปกรณ์ที่รับข้อมูลดังกล่าว ว่ามีแอพพลิเคชันที่สามารถแสดงผลข้อมูลที่รับมาได้หรือไม่
3. ในการทำงานของระบบ Baloo เกิดปัญหาที่เรียกว่า Bluespam นั่นคือ ผู้ใช้ได้รับข่าวสารหรือโฆษณาที่ไม่ต้องการซึ่งเป็นลักษณะเดียวกับ E-mail spam ในการงานอินเตอร์เน็ต
4. ลักษณะการทำงานที่เหมาะสำหรับการกระจายข้อมูลด้วยเทคโนโลยีบลูทูธเป็น 3 ลักษณะ ได้แก่ การแชร์ข้อมูล การส่งข้อมูลที่มีขนาดจำกัด และการกระจายข้อมูลไปยังกลุ่มผู้รับ แต่ลักษณะการทำงานต่างๆดังกล่าวจะถูกจำกัดด้วยข้อจำกัดด้านเทคโนโลยี นั่นคือ กระจายข้อมูลไปยังจำนวนผู้รับที่จำกัด ในระยะที่จำกัด และข้อมูลขนาดเล็ก

งานวิจัย [7] ศึกษาผลกระทบของพารามิเตอร์ต่างๆ เช่น ระยะทาง ชนิดของแพกเก็ต และความยาวของข้อมูล ที่มีต่อการส่งข้อมูลเสียงชนิด MP3 บน การสื่อสารไร้สายด้วยเทคโนโลยีบลูทูธ โดยได้พัฒนาระบบเพื่อการเล่นข้อมูลเสียงทั้งระบบเสียงโมโนและระบบเสียงสเตอริโอ ซึ่งระบบดังกล่าวถูกแบ่งการพัฒนาออกเป็น 2 ชนิด ได้แก่ Bluetooth point-to-point Application และ Bluetooth point-to-multipoint Application

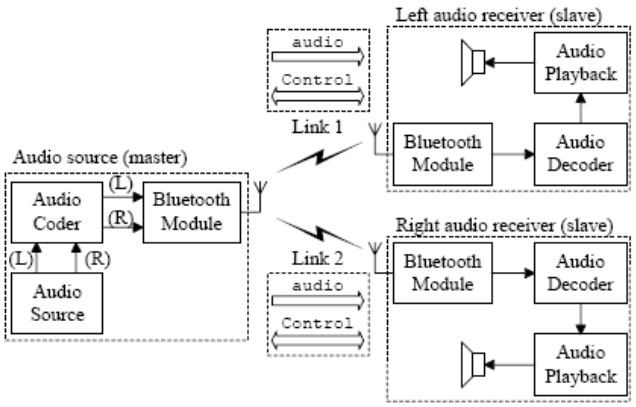


รูปที่ 14 Point-to-point application [7]

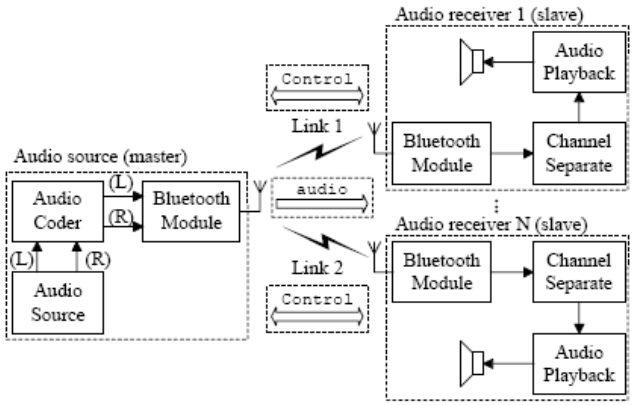


รูปที่ 15 Point-to-point application with control data [7]

รูปที่ 14 และรูปที่ 15 เป็นการส่งข้อมูลแบบ point-to-point จากอุปกรณ์แม่ข่ายไปยังอุปกรณ์ลูกข่ายโดยอาศัยการติดต่อชนิด ACL ซึ่งรูปทั้งสองต่างกันที่รูปที่ มีการส่งข้อมูลควบคุมบางอย่าง เช่น สถานะของข้อมูลไปยังเครื่องลูกข่ายด้วย



รูปที่ 16 Point-to-multipoint application [7]



รูปที่ 17 Point-to-multipoint application (Broadcast) [7]

รูปที่ 16 และรูปที่ 17 เป็นการส่งข้อมูลแบบ point-to-multipoint จากเครื่องลูกข่ายไปเครื่องลูกข่ายซึ่งเป็นระบบเสียงแบบสเตอริโอ นั่นคือ ประกอบด้วยช่องสัญญาณซ้ายและขวา

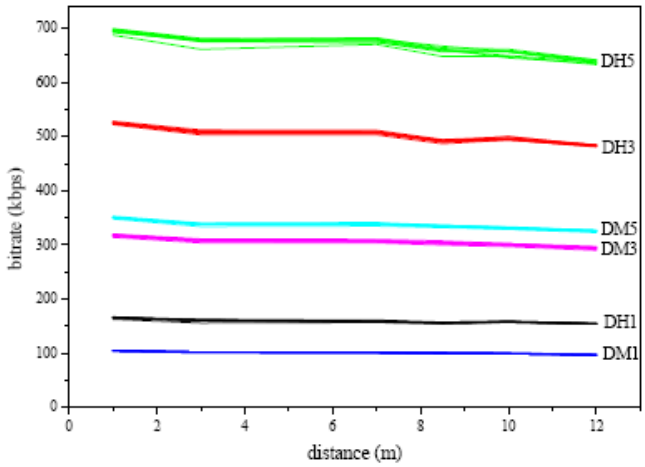
ซึ่งทั้งสองรูปมีความแตกต่างกันที่รูป เป็นการส่งข้อมูลเสียงพร้อมกับข้อมูลควบคุมไปช่องสัญญาณซ้ายและขวาแยกจากกัน ส่วนรูป เป็นการส่งข้อมูลเสียงและข้อมูลควบคุมแบบ Broadcast ไปยังช่องสัญญาณทั้งซ้ายและขวาพร้อมกัน

สำหรับเงื่อนไขและพารามิเตอร์สำหรับทำการทดสอบได้แก่

Distance d (m)	ACL packet	L_p (bytes)
1-20	DM1, DM3, DM5 DH1, DH3, DH5	1013 – 55x1013 (1k – 55k)

ตารางที่ 3 เงื่อนไขและพารามิเตอร์ [7]

โดยผลการวิจัยเป็นดังนี้

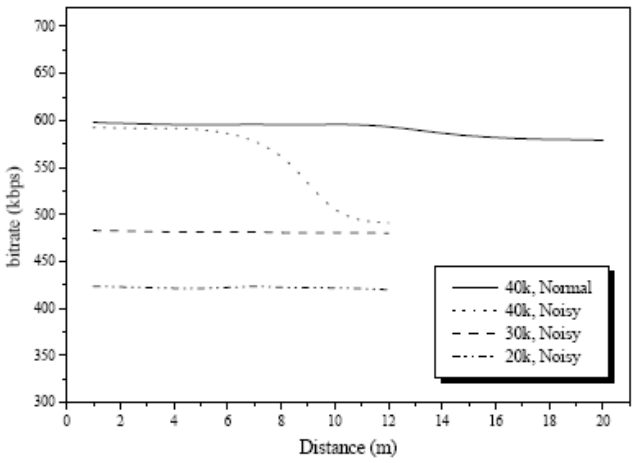


รูปที่ 18 ผลการวิจัยสำหรับการส่งข้อมูลแบบ point-to-point [7]

รูปที่ 18 เป็นผลการวิจัยสำหรับการส่งข้อมูลแบบ point-to-point ตามเงื่อนไขต่างข้างต้น โดยไม่มีสัญญาณรบกวนพบว่า

ACL แพกเก็ตชนิด DH5 ให้ความเร็วในการส่งข้อมูลสูงสุดนั่นคือ 723.2 kbps ตามรายละเอียดที่กำหนดไว้ใน [3] ระยะห่างระหว่าง Bluebox Server และ Bluebox Client ไม่มีผลชัดเจนต่อความเร็วในการส่งข้อมูล

ขนาดข้อมูล 1-55 kB ไม่มีผลต่อความเร็วในการส่งข้อมูล



รูปที่ 19 **ผลการวิจัยสำหรับการส่งข้อมูลแบบ** point-to-point **ด้วยแพกเก็ตชนิด** DH5 [7]

รูปที่ 19 แสดงผลการวิจัยสำหรับการส่งข้อมูลแบบ point-to-point โดยกำหนดชนิดแพกเก็ตเป็น DH5 ขนาดของความยาวข้อมูลและระยะทางในการส่งข้อมูล รวมทั้งส่งข้อมูลควบคุมไปพร้อมกับข้อมูลเสียงด้วย ทั้งในสภาพแวดล้อมที่มีและไม่มีสัญญาณรบกวน พบว่า

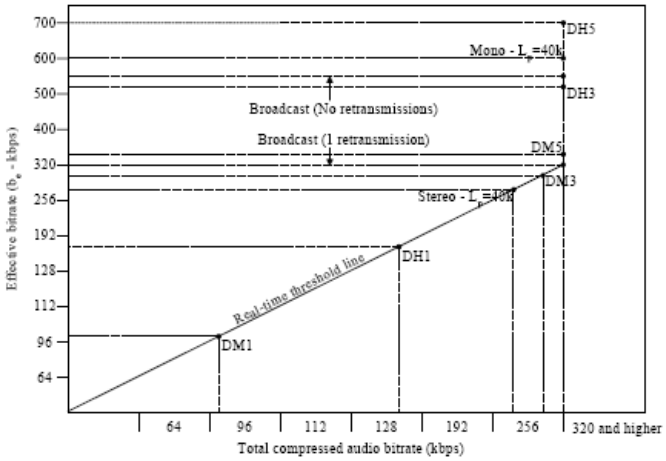
ความเร็วในการส่งข้อมูลลดลง เนื่องจากต้องส่งข้อมูลเพิ่มขึ้น นั่นคือ ข้อมูลควบคุมนั่นเอง ความเร็วในการส่งข้อมูลขึ้นกับความยาวของข้อมูล โดยข้อมูลขนาด 40k มีผลทำให้เกิด delay ในสภาพแวดล้อมที่มีสัญญาณรบกวน

ในสภาพแวดล้อมที่มีสัญญาณรบกวน ไม่สามารถส่งข้อมูลที่ระยะเกินกว่า 12 ม.ได้แต่ในสภาพแวดล้อมที่ไม่มีสัญญาณรบกวนสามารถส่งข้อมูลได้ที่ระยะเกินกว่า 20 ม.

สำหรับการส่งข้อมูลแบบ point-to-multipoint ทำการทดลองโดยกำหนดชนิดแพกเก็ตเป็น DH5 และระยะทางในการส่งข้อมูล 1 ม. ซึ่งแสดงถึงความเร็วในการส่งข้อมูลที่ลดลงและอัตราการสูญหายของข้อมูล ตามจำนวนการส่งข้อมูลซ้ำ

N	b_e (kbps)	Audio data losses
0	551	7%-9%
1	325	2%-3%

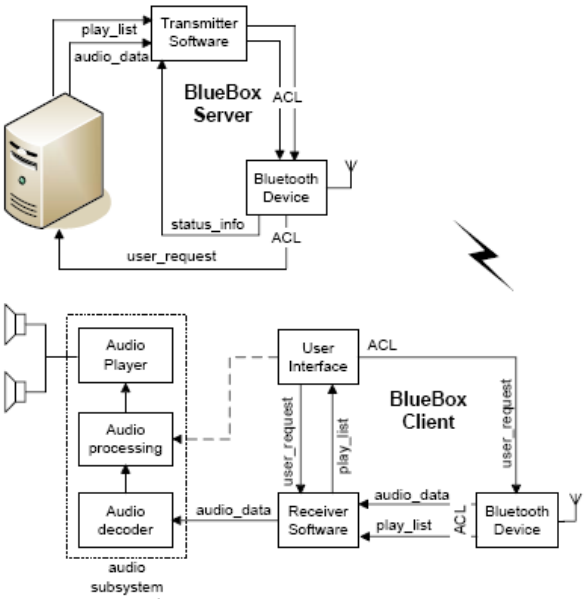
ตารางที่ 4 **ความเร็วในการส่งข้อมูลและอัตราการสูญหายของข้อมูลเมื่อเทียบกับจำนวนการส่งข้อมูลซ้ำ สำหรับการส่งข้อมูลแบบ** point-to-multipoint [7]



รูปที่ 20 **สรุปผลการวิจัย** [7]

จากผลการวิจัยข้างต้น สามารถสรุปได้ว่าเทคโนโลยีบลูทูธมีความเร็วเพียงพอในการส่งข้อมูลเสียงทั้งในระบบโมโนและระบบสเตอริโอ แต่สำหรับการส่งข้อมูลในระบบสเตอริโอหรือ point-to-multipoint จะเกิดปัญหา Phase Distortion ระหว่างช่องสัญญาณทั้งซ้ายและขวา ซึ่งมีผลกระทบต่อคุณภาพเสียง โดยรูปที่ 20 แสดงความสัมพันธ์ระหว่างความเร็วในการส่งข้อมูลและความละเอียดของข้อมูลที่ระยะ 1 ม.

งานวิจัย [8] พัฒนา BlueBox ทำงานในลักษณะ Client-Server ใช้เทคโนโลยีบลูทูธในการส่งข้อมูลเสียงคุณภาพสูง ในที่นี่คือ MP3 (128 Kbps) จากอุปกรณ์ซึ่งทำหน้าที่เป็นตู้เพลงไปยังอุปกรณ์พกพาต่างๆของ



รูปที่ 21 **สถาปัตยกรรมของ** BlueBox [8]

รูปที่ 21 แสดงสถาปัตยกรรมของ BlueBox ประกอบด้วยส่วนต่างๆ ดังนี้

BlueBox Server ทำหน้าที่เก็บข้อมูลรายการเพลงและส่งข้อมูลเพลงที่ได้รับการร้องขอจากผู้ใช้ โดยต้องสามารถรองรับการทำงานของผู้ใช้ได้หลายคนในเวลาเดียวกัน

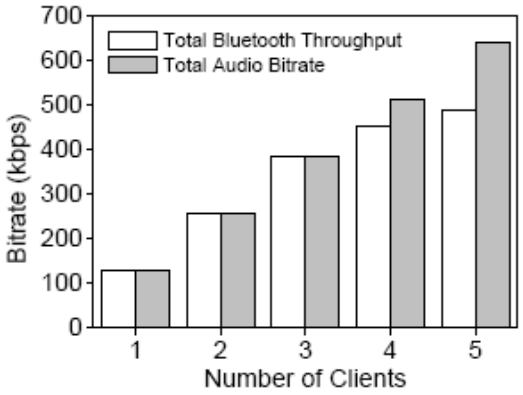
BlueBox Client ทำหน้าที่ในส่วนของผู้ใช้ โดยร้องขอรายการเพลงไปยัง BlueBox Server ประกอบด้วยการทำงาน 3 ส่วน คือ

Reciever Software ทำหน้าที่รับส่งข้อมูลกับ BlueBox Server

User Interface ทำหน้าที่เก็บรายการเพลงและเริ่มต้นการใช้งานต่างๆของผู้ใช้

Audio Sub-system ทำหน้าที่ถอดรหัสและเล่นข้อมูลเสียงออกทางการ์ดเสียง

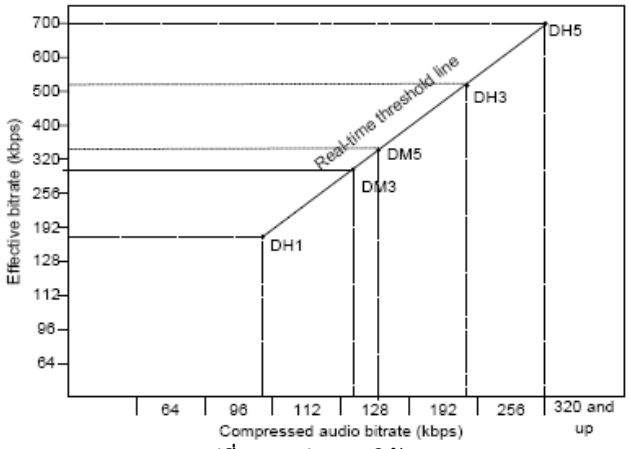
ผลการวิจัยสามารถแสดงได้ดังนี้



รูปที่ 22 จำนวน BlueBox Client **กับความเร็วในการส่งข้อมูล** [8]

จากรูปที่ 22 แสดงจำนวน BlueBox Client และความเร็วในการส่งข้อมูลสูงสุดที่ Bluebox Server สามารถรองรับได้ โดยพบว่าในเวลาเดียวกัน BlueBox Server รองรับจำนวน BlueBox Client ได้สูงสุด 3 ตัว สำหรับกรณีที่ BlueBox Client มากกว่า 3 ตัวนั้น พบว่า ความเร็วในการส่งข้อมูลไม่ได้ตามที่คาดหวังไว้ (512 kbps – BlueBox Client 4 ตัว, 640 kbps – BlueBox Client 5 ตัว) ซึ่งอาจเกิดจากจำนวน BlueBox Client ที่มากขึ้นเป็นผลให้เกิดโอเวอร์เฮดระหว่างการส่งข้อมูลมากขึ้นหรือเกิดการส่งข้อมูลซ้ำมากขึ้น

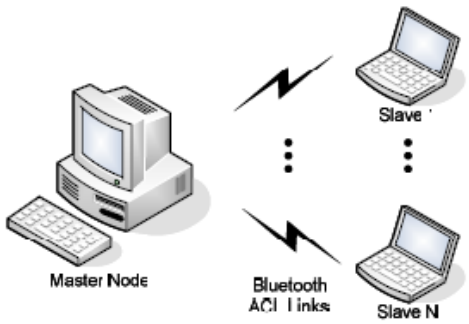
นอกจากนี้ยังสามารถสรุปความสัมพันธ์ระหว่างความเร็วในการส่งข้อมูลและความละเอียดของข้อมูลในกรณีที่มี BlueBox Client 2 ตัว โดยกราฟแสดงความเร็วและชนิดของแพกเก็ตที่สามารถรองรับการทำงานแบบ Real-time ได้



รูปที่ 23 **สรุปผลการวิจัย** [8]

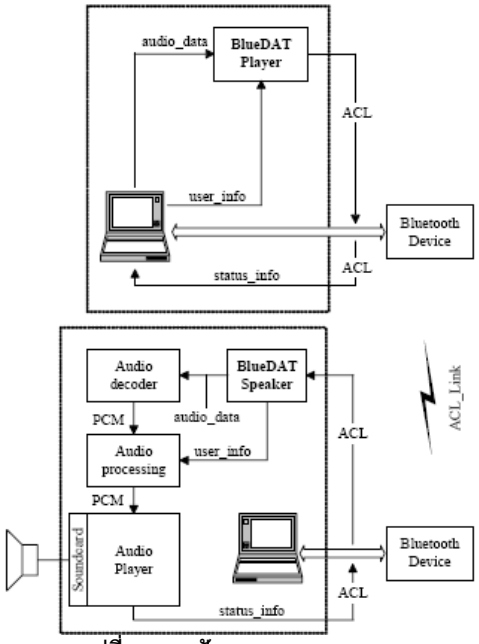
จากรูปที่ 23 พบว่าแพกเก็ตชนิด DM3 ก็เพียงพอต่อการส่งข้อมูล MP3 ซึ่งมีความละเอียด 128 kbps แบบ Real-time

งานวิจัย [9] ได้ออกแบบและพัฒนาแอพพลิเคชันสำหรับอุปกรณ์เครื่องใช้ภายในบ้าน เพื่อลดการใช้สายเชื่อมโยงกันระหว่างอุปกรณ์ ในที่นี้คือ BlueDAT เป็นแอพพลิเคชันไร้สายเพื่อเล่นส่งข้อมูลเสียงที่มีคุณภาพจากตัวส่งหนึ่งตัว ไปยังตัวรับหนึ่งหรือสองตัวด้วยเทคโนโลยีบลูทูธ ดังรูปที่ 24



รูปที่ 24 BlueDAT Application [9]

สถาปัตยกรรม BlueDAT ประกอบด้วย 2 ส่วน ได้แก่ BlueDAT Player ทำหน้าส่งข้อมูลเสียงมายัง BlueDAT Speaker ซึ่งมีได้สูงสุดจำนวน 2 ตัว บนช่องสัญญาณ ACL



รูปที่ 25 **สถาปัตยกรรม** BlueDAT [9]

ช่องสัญญาณ ACL เป็นช่องสัญญาณที่มีการส่งข้อมูลแบบ point-to-multipoint แบบสองทิศทาง ซึ่งสามารถให้ความเร็วในการส่งข้อมูลได้สูงสุด 721 kbps ต่างกับช่องสัญญาณแบบ SCO ที่เป็นการส่งข้อมูลแบบ point-to-point และให้ความเร็วในการส่งข้อมูลได้สูงสุด 192 kbps เท่านั้น

ผลการวิจัย สามารถสรุปได้ดังตาราง

MEASURED EFFECTIVE BITRATE AND MAXIMUM DISTANCES IN POINT-TO-POINT AND POINT-TO-MULTIPOINT MODES

Number of Clients	Room Type	Total Effective Bitrate (kbps)	Maximum Distance (m)
1	Normal	320	10
	Noisy	192	10
2	Normal	640	3
	Noisy	128	5

ตารางที่ 5 Bit Rate **และระยะในการส่งข้อมูล** [9]

เมื่อมี BlueDAT Speaker 2 ตัว ในสภาพแวดล้อมปกติ ระยะที่ Speaker สามารถเล่นเสียงได้โดยไม่มีการบิดเบือนอยู่ที่ระยะ 3 เมตร โดยระยะที่เกินกว่านี้แอพพลิเคชันจะทำการปรับค่าอัตราการบีบอัดข้อมูลโดยอัตโนมัติ สำหรับในสภาพแวดล้อมที่มีสัญญาณรบกวน คุณภาพของเสียงก็จะลดลงด้วย โดยที่ระยะมากกว่า 5 เมตร แอพพลิเคชันจะทำการปรับค่าอัตราการบีบอัดข้อมูลลงจนกระทั่งเหลือ 64 kbps

MEASURED EFFECTIVE BITRATE AND AUDIO DATA LOSSES IN BROADCAST MODE

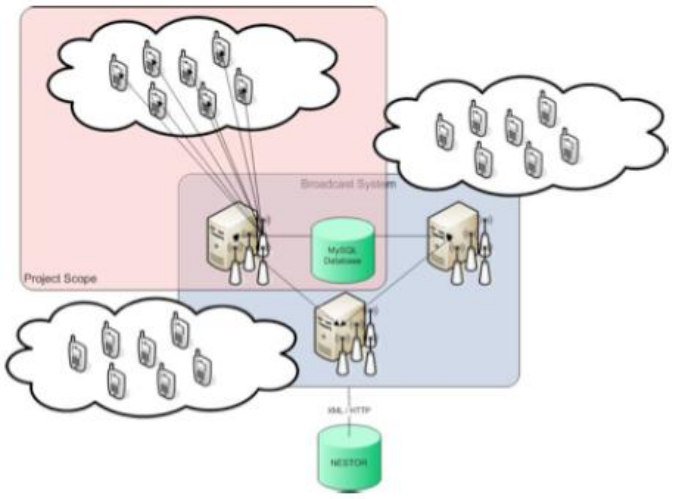
Number of Retransmissions (N)	Total Effective Bitrate (kbps)	Audio Data Losses
0	551	7%-9%
1	325	2%-3%

ตารางที่ 6 Bit Rate **และอัตราการสูญหายของข้อมูลในโหมด** Broadcast [9]

นอกจากนี้เพื่อลดอัตราการสูญหายของข้อมูล งานวิจัยนี้ยังเพิ่มจำนวนการส่งข้อมูลซ้ำ ซึ่งมีผลดังตารางที่ 6 ซึ่งการส่งข้อมูลซ้ำหนึ่งครั้งสามารถลดอัตราการสูญหายของข้อมูลลงจาก 7-9% เหลือเพียง 2-3% เท่านั้น

งานวิจัย [10] พัฒนา RuGBlue เพื่อทำการวิจัย ซึ่งผลการวิจัยแสดงถึงข้อจำกัดและความเป็นไปได้ต่างๆของเทคโนโลยีบลูทูธ โดยอาศัยการส่งข้อมูลแบบ Broadcast ไปยังอุปกรณ์ต่างๆ ในลักษณะ Proximity Marketing หรือการโฆษณาในพื้นที่เปิด ซึ่งในงานวิจัยนี้คือ ตึก Bernoulliborg ณ University of Groningen และใจกลางเมือง Groningen ประเทศฮอลแลนด์

RuGBlue ประกอบด้วย 5 ส่วน ได้แก่ RuGBlue ซอฟท์แวร์, Bluetooth Antennae, ฐานข้อมูลภายใน, ฐานข้อมูลภายนอก และผู้ใช้งานบลูทูธ



รูปที่ 26 **ระบบ** RugBlue [10]

ในการพัฒนา RuGBlue นั้นผู้พัฒนาเลือกใช้ Ubuntu 8.04 เป็นระบบปฏิบัติการร่วมกับ BlueZ Stack และ BlueCove เป็นไลบรารี่ในส่วนของซอฟท์แวร์ ใช้ Dongles ในส่วนที่เป็นฮาร์ดแวร์ และเลือก MySQL สำหรับระบบฐานข้อมูล

การทดลองถูกกำหนดเป็น 4 การทดลอง แต่การทดลองที่เกี่ยวข้องกับงานวิจัยนี้ได้แก่

1. การทดสอบความสามารถในการตอบสนองของอุปกรณ์ เป็นการทดสอบความเร็วในการรับส่งข้อมูลระหว่างอุปกรณ์ ซึ่งเป็นสิ่งจำเป็นสำหรับการส่งข้อมูล

แบบ Broadcast ผลการทดสอบพบว่า อุปกรณ์ผู้รับสามารถรับข้อความที่ส่งออกไปได้ 17 เครื่อง คิดเป็น 50.94% จากจำนวนอุปกรณ์ทั้งหมด นั่นคือ สามารถส่งข้อมูลแบบ Broadcast ไปยังอุปกรณ์มากกว่า 7 ตัวได้

2. การทดสอบการเชื่อมต่อ เป็นการทดสอบความสามารถในการรองรับการติดต่อกับอุปกรณ์มากกว่า 7 ตัว ซึ่งเป็นข้อจำกัดของเทคโนโลยีบลูทูธ

Round	# Delivery dongles	Max. possible # simultaneous connections	Max. # simultaneous connections reached
1	1	7	7
2	2	14	13
3	3	21	14
4	2	14	14

ตารางที่ 7 จำนวนการเชื่อมต่อ ณ ตึก Bernouliborg **ของ** University of Gronigen [10]

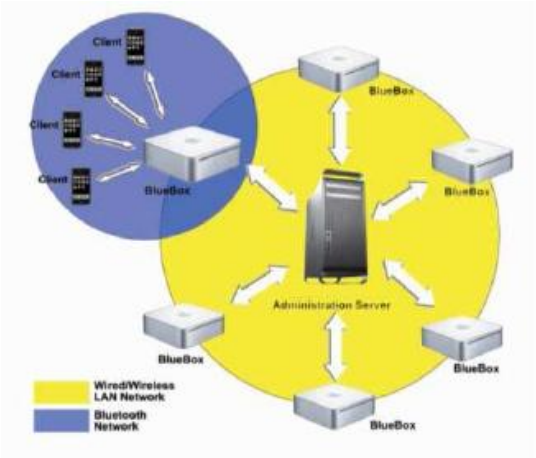
Round	# Delivery dongles	Max. possible # simultaneous connections	Max. # simultaneous connections reached
1	3	21	21
2	4	28	19
3	4	28	22

ตารางที่ 8 จำนวนการเชื่อมต่อ ณ ใจกลางเมือง Gronigen [10]

ตารางที่ 7 และตารางที่ 8 แสดงให้เห็นว่า การเพิ่มจำนวนอุปกรณ์ซึ่งทำหน้าที่เป็นตัวส่งข้อมูล สามารถเพิ่มจำนวนผู้รับในการส่งข้อมูลแบบ Broadcast ได้

จากผลการทดลองสามารถสรุปได้ว่า ถึงแม้จะมีปัจจัยมากมายในการส่งข้อมูลแบบ Broadcast ไม่ว่าจะเป็นข้อจำกัดด้านเวลา ชนิดหรือยี่ห้อของอุปกรณ์ ความพร้อมของอุปกรณ์ การรบกวนของสัญญาณต่างๆ รวมถึงระบบปฏิบัติการของอุปกรณ์ส่งข้อมูล แต่เทคโนโลยีบลูทูธก็สามารถทำการส่งข้อมูลแบบ Broadcast ไปยังอุปกรณ์ต่างๆได้ นอกจากนี้ยังสามารถเพิ่มจำนวนผู้รับข้อมูลได้โดยการเพิ่มจำนวนตัวส่งข้อมูล

งานวิจัย [11] พัฒนาระบบเรียกว่า BlueBus มีจุดประสงค์เพื่อเป็นศูนย์กลางในการกระจายข่าวสารข้อมูลไปยังอุปกรณ์ของผู้ใช้ไม่ว่าจะเป็นโทรศัพท์เคลื่อนที่ หรือ PDA เป็นต้น โดยระบบ BlueBus ติดตั้งอยู่ที่สถานีขนส่งและรถประจำทาง



รูปที่ 27 ระบบ BlueBus [11]

ระบบ BlueBus ประกอบด้วย 3 ส่วน

Administration Server ทำหน้าที่เป็นฐานข้อมูลเก็บข้อมูลข่าวสารต่างๆที่จะส่งให้ผู้ใช้

BlueBox ทำหน้าที่ติดต่อกับอุปกรณ์ของผู้ใช้ด้วยเทคโนโลยีบลูทูธบนรถประจำทาง

อุปกรณ์ของผู้ใช้ ทำหน้าที่รับข่าวสารหรือบริการจาก BlueBox

การบริการของระบบ BlueBus ประกอบด้วย การรับข่าวสาร การ Chat การดาวน์โหลดข้อมูล และการเล่นเกมส์ ซึ่งผู้ใช้จะถูกสอนและลงโปรแกรมลงบนอุปกรณ์ของผู้ใช้ก่อนการใช้งาน ซึ่งผลการทดสอบเป็นดังนี้

Cell Phone Model	Single connection	4 Simultaneous Connections
Motorola L6	5Kbps	5Kbps
Nokia 3230	41Kbps	20Kbps
Nokia 6280	104Kbps	30Kbps
Nokia N70	116Kbps	32Kbps
SonyErricson K750i	40Kbps	19Kbps
SonyErricson P910i	20Kbps	11Kbps
Samsung D800	43Kbps	21Kbps

ตารางที่ 9 ผลการวิจัย [11]

ตารางที่ 9 แสดงการใช้งานบริการต่างๆของผู้ใช้ โดยมีรายละเอียด ได้แก่ ชนิดของอุปกรณ์ของผู้ใช้ และความเร็วในการใช้งานตามจำนวนผู้ใช้งานทั้งการใช้งานคนเดียว และการใช้งานจากผู้ใช้ 4 คนพร้อมกัน ซึ่งแสดงให้เห็นว่า สามารถใช้เทคโนโลยีบลูทูธเป็นสื่อในการกระจายข่าวสารไปยังอุปกรณ์พกพาชนิดต่างๆได้

งานวิจัย [12] ศึกษาประสิทธิภาพของการส่งข้อมูลแบบ Broadcast ของระบบบลูทูธ โดยการวิเคราะห์ผลกระทบของชนิดของ ACL แพกเก็ต ซึ่งมีขนาดและวิธีการป้องกันความผิดพลาด

ของข้อมูล (Error Protection Scheme) ที่แตกต่างกันในมุมของความน่าเชื่อถือในการส่งข้อมูล (Reliability) และ Throughput บนคุณลักษณะของสื่อสัญญาณซึ่งกำหนดโดยค่า Bit Error Rate (BER) ซึ่งผลของงานวิจัยที่ได้คือ ชนิดของ ACL แพกเก็ตและจำนวนครั้งของการส่งข้อมูลซ้ำ (Retransmission) ที่เหมาะสมเพื่อให้ได้ประสิทธิภาพของการส่งข้อมูลแบบ Broadcast สูงสุด

Bit Error Rate	Optimum Mode Selection	N_{BC}	Throughput (kbps)
[BER $<7.5x10^{-7}$]	DH5	1	723.00
[$7.5x10^{-7}$, 10^{-6}]	DH3	1	585.60
[10^{-6}, 10^{-4}]	DM5	1	477.80
[10^{-4}, $5x10^{-4}$]	DM3	1	387.30
[$5x10^{-4}$, $5.5x10^{-4}$]	DM5	2	238.90
[$5.5x10^{-4}$, $2.5x10^{-3}$]	DM3	2	193.50
[$2.5x10^{-3}$, $3x10^{-3}$]	DM5	3	159.30
[$3x10^{-3}$, $5x10^{-3}$]	DM3	3	129.00
[$5x\ 10^{-3}$, $6.5x10^{-3}$]	DM3	4	96.80
[$6.5x10^{-3}$, $7x10^{-3}$]	DM3	5	77.40
[$7x10^{-3}$, $8x10^{-3}$]	DM3	6	64.50
[$8x10^{-3}$, $9.5x10^{-3}$]	DM3	7	48.40
[$9.5x10^{-3}$, 10^{-2}]	DM3	8	45.00
[10^{-2}, $1.5x10^{-2}$]	DM1	4	27.20
[$1.5x10^{-2}$, $2x10^{-2}$]	DM1	5	21.76
[$2x10^{-2}$, $2.5x10^{-2}$]	DM1	7	15.54
[$2.5x10^{-2}$, $3x10^{-2}$]	DM1	9	12.10
[$3x10^{-2}$, $3.5x10^{-2}$]	DM1	13	8.35
[$3.5x10^{-2}$, $4x10^{-2}$]	DM1	17	6.40
[$4x10^{-2}$, $4.5x10^{-2}$]	DM1	23	4.73
[$4.5x10^{-2}$, $5x10^{-2}$]	DM1	31	3.51
[$5x10^{-2}$, $5.5x10^{-2}$]	DM1	43	2.53
[$5.5x10^{-2}$, $6x10^{-2}$]	DM1	61	1.78

ตารางที่ 10 **ผลการวิจัย** [12]

ผลการวิจัยสรุปสามารถสรุปได้ดังตารางที่ 10 ซึ่งประกอบด้วย ค่า BER ที่แสดงถึงคุณลักษณะของสื่อสัญญาณ ชนิดของแพกเก็ต ที่แสดงถึงขนาดและชนิดของการป้องกันความผิดพลาดของข้อมูล จำนวนครั้งของการส่งข้อมูลซ้ำ และ Throughput ที่ได้เมื่อต้องการความน่าเชื่อถือของระบบที่ 99.9 %

งานวิจัย [13] อธิบายการส่งข้อมูลเสียงบนอินเตอร์เนตโปรโตคอล (VoIP) ด้วยเทคโนโลยีบลูทูธ โดยอธิบายถึงบลูทูธโปรไฟล์ (Bluetooth Profiles) ซึ่งทำหน้าที่เป็นเสมือนพิมพ์เขียวสำหรับการทำงานของอุปกรณ์ที่มีลักษณะการทำงานของแอพพลิเคชันแบบเดียวกัน ได้แก่ A2DP, LAN และ BNEP

นอกจากนี้ยังกล่าวถึงคุณสมบัติของช่องสัญญาณ SCO และ ACL สำหรับช่องสัญญาณแบบ SCO นั้นมีลักษณะการส่งข้อมูลระหว่างอุปกรณ์แม่ข่ายและลูกข่ายเป็นแบบ point-to-point

ส่วน ACL มีลักษณะการส่งข้อมูลแบบ point-to-multipoint โดยนอกจากลักษณะการส่งข้อมูลแล้ว ACL ยังรองรับการในเรื่องของความปลอดภัยของข้อมูลซึ่งเป็นสิ่งสำคัญสำหรับการส่งข้อมูลเสียง ทั้ง Authorization, Authentication และ Encryption

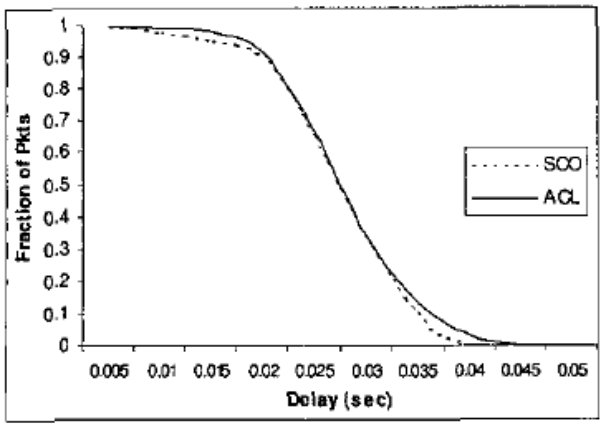
งานวิจัยนี้มีข้อสรุปว่า ประสิทธิภาพและความสำเร็จในการออกแบบและพัฒนาอุปกรณ์ซึ่งจะทำการส่งเสียงด้วยเทคโนโลยีบลูทูธนั้น ขึ้นกับการเลือกใช้บลูทูธโปรไฟล์ที่เหมาะสม ซึ่งได้แก่ A2DP เหมาะสำหรับแอพพลิเคชันที่ต้องการความเร็วในการส่งข้อมูลระหว่างแม่ข่ายไปยังลูกข่าย และลูกข่ายไปยังแม่ข่ายที่ไม่เท่ากัน สำหรับ BNEP เหมาะสำหรับแอพพลิเคชันที่ความเร็วในการส่งข้อมูลดังกล่าวเท่ากัน

งานวิจัย [14] จำลองและประเมินประสิทธิภาพของบลูทูธที่พิโคเน็ตประกอบด้วยอุปกรณ์แม่ข่ายหนึ่งตัวและอุปกรณ์ลูกข่ายมากกว่า 7 ตัว โดยมุ่งเน้นไปยัง Throughput และ Delay ของระบบเมื่อทำการส่งข้อมูลแบบ point-to-multipoint โดยใช้ BONeS เป็นเครื่องมือในการสร้างและประเมินประสิทธิภาพของแบบจำลอง

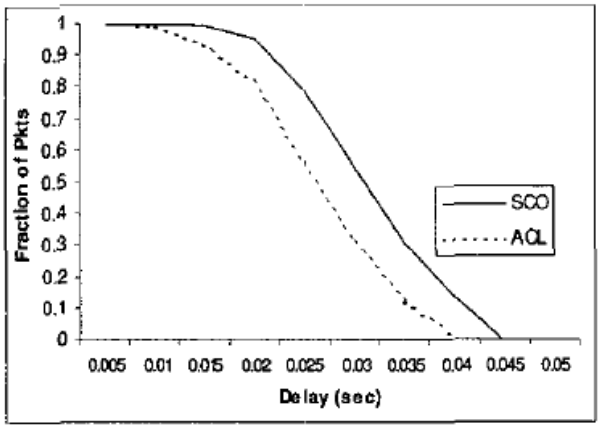
จากผลการวิจัย พบว่า ความเร็วในการส่งในการส่งข้อมูลจะน้อยกว่า 10 kbps เมื่อพิโคเน็ตมีการเชื่อมต่อ SCO และมีการเชื่อมต่อ ACL มากกว่า 6 การเชื่อมต่อ นอกจากนี้ค่า Delay สามารถควบคุมให้น้อยกว่า 10 ms ได้ในกรณีที่มีการเชื่อมต่อ ACL ไม่เกิน 4 การเชื่อมต่อ ซึ่งแสดงให้เห็นว่าชนิดและจำนวนการเชื่อมต่อมีผลต่อประสิทธิภาพของระบบบลูทูธทั้ง Throughput และ Delay

งานวิจัย [15] เปรียบเทียบข้อดีและข้อเสียในการใช้แพกเก็ต SCO และ ACL สำหรับส่งข้อมูลเสียง โดยงานวิจัยนี้อธิบายถึง คุณลักษณะของแพกเก็ตทั้ง SCO และ ACL และอธิบายถึง Codec ชนิดต่างๆที่ใช้สำหรับการส่งข้อมูลเสียง ได้แก่ Log PCM, CVSD, MPEG Encoding และ Variable bit-rate รวมถึง Lossless Compression และเนื่องจาก SCO รองรับการส่งข้อมูลที่ความเร็ว 64 kbps ซึ่งรองรับความถี่ที่ 4 MHz จึงสามารถรองรับการส่งข้อมูลเสียงพูดได้เท่านั้น ไม่เหมาะสำหรับการส่งข้อมูลประเภทดนตรีหรือข้อมูลอื่นที่ต้องการความถี่สูงกว่านี้ การใช้งาน Codec บน SCO จึงเป็น Codec ที่ไม่ซับซ้อนได้แก่ Log PCM และ CVSD

งานวิจัย [16] ใช้ NS-2 และ ROK 101007 module (Ericsson Bluetooth Application Toolkits) เป็นเครื่องมือในการจำลองการส่งข้อมูลเสียงโดยใช้การช่องสัญญาณ ACL แทนช่องสัญญาณ SCO



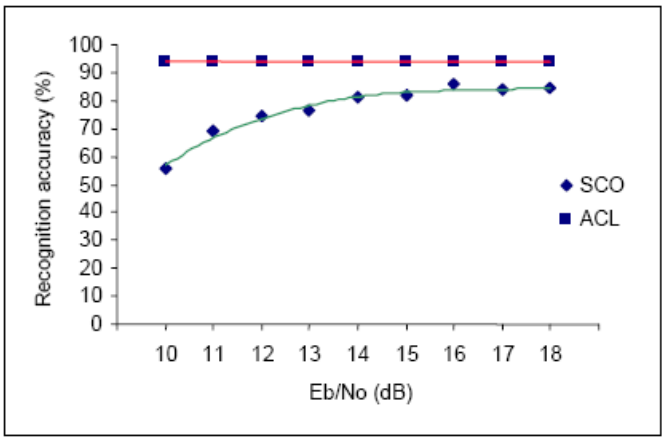
รูปที่ 28 ดีเลย์ของการส่งแพกเก็ตเสียงด้วยช่องสัญญาณ ACL **เทียบกับ** SCO **ชนิด** HV2 [16]



รูปที่ 29 ดีเลย์ของการส่งแพกเก็ตเสียงด้วยช่องสัญญาณ ACL **เทียบกับ** SCO **ชนิด** HV3 [16]

จากผลการวิจัยพบว่า การใช้ช่องสัญญาณ ACL เป็นทางเลือกที่ดีในการส่งข้อมูลเสียง ถึงแม้จะเกิดดีเลย์ในการส่งข้อมูลอยู่บ้างแต่ก็อยู่ในเกณฑ์ที่สามารถยอมรับได้ ดังรูปที่ 28 ที่ดีเลย์ของการใช้ช่องสัญญาณ ACL มากกกว่า SCO ที่ใช้แพกเก็ตชนิด HV2 และดีเลย์ที่น้อยหรือดีกว่าเมื่อเทียบกับการใช้ช่องสัญญาณ SCO ที่ใช้แพกเก็ตชนิด HV3 ดังรูปที่ 29

งานวิจัย [17] เปรียบเทียบคุณภาพการรู้จำเสียงพูดสำหรับข้อมูลเสียงพูดบนช่องสัญญาณ ACL และ SCO ที่คุณภาพของช่องสัญญาณต่างๆ โดยอาศัยผู้พูดผู้ชาย 5 คน พูดคำพูด 50 คำ ซึ่งจะถูกเข้ารหัส PCM ที่ความถี่ 8 Hz และมีขนาด 8 บิตต่อแซมเปิล กำหนดค่าสัญญาณรบกวนระหว่าง 10-18 dB (Es/No) และกำหนดให้ใช้แพกเก็ต DM1 สำหรับ ACL และ HV3 สำหรับ SCO



รูปที่ 30 ผลการวิจัย [17]

รูปที่ 30 แสดงผลการเปรียบเทียบการรู้จำเสียงพูดสำหรับข้อมูลเสียงบนเทคโนโลยีบลูทูธที่ใช้ช่องสัญญาณ ACL และ SCO ซึ่งแต่ละคำพูดจะทำการทดลองซ้ำ 15 ครั้ง พบว่า ความแม่นยำเฉลี่ยในการรู้จำเสียงพูดที่ใช้ช่องสัญญาณ SCO คือ 55% เมื่อคุณภาพของสื่อสัญญาณไม่ดี และ 84% เมื่อคุณภาพของสื่อสัญญาณดี ในขณะที่ความแม่นยำเฉลี่ยในการรู้จำเสียงพูดที่ใช้ช่องสัญญาณ ACL คือ 94% โดยไม่ขึ้นกับคุณภาพของสื่อสัญญาณ เพราะไม่มีความผิดพลาดในการส่งข้อมูล แต่การใช้ช่องสัญญาณ ACL ก็มีดีเลย์มากกว่าเล็กน้อยเมื่อเทียบกับการใช้ช่องสัญญาณ SCO แต่ไม่มีนัยสำคัญสำหรับแอพพลิเคชันประเภทรู้จำเสียงพูด

งานวิจัย [18] เปรียบเทียบความเร็วในการส่งข้อมูลและประสิทธิภาพในการส่งข้อมูลของ Overlaid Bluetooth Piconets (OBP) กับ Scatternet รวมถึงแสดงให้เห็นถึงความเป็นไปได้ในการใช้งาน OBP แทน Scatternet โดยอาศัย UCBT ns-2 เป็นเครื่องมือในการจำลองการทำงานของ OBP ผลการวิจัยพบว่า OBP ให้ค่า Throughput ที่สูงกว่า Scatternet เพราะไม่ต้องการ Routing Protocol เช่นเดียวกับ Scatternet

เนื่องจาก ASB Logical Transport ยังไม่ได้ถูกพัฒนาขึ้นมาใช้งานจริงในปัจจุบัน งานวิจัย [19] จึงพัฒนา JSR82ext ขึ้นเพื่อรองรับการทำงานดังกล่าว โดยเป็นการพัฒนาเพิ่มเติมจากบลูทูธไลบรารี่สำหรับภาษาจาวาที่มีชื่อว่า JSR-82 [20] และได้ทำการทดสอบการทำงานของ ASB Logical Transport ที่พัฒนาขึ้นโดยการวัดประสิทธิภาพของการส่งข้อมูลเสียงบนช่องสัญญาณดังกล่าว ซึ่งได้แก่ ความเร็วเฉลี่ยในการส่งข้อมูล ดีเลย์เฉลี่ยในการส่งข้อมูลเสียงแต่ละแพ็กเก็ต และค่าเบี่ยงเบนมาตราฐานสำหรับดีเลย์เฉลี่ยในการส่งข้อมูลเสียงแต่ละแพ็กเก็ต ซึ่งผลการ

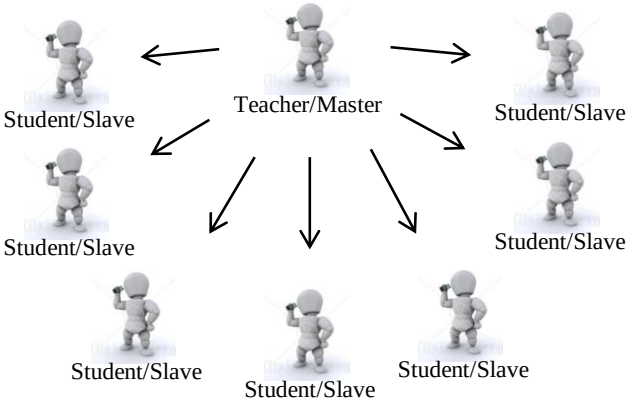
ทดลองแสดงให้เห็นว่า สำหรับการส่งข้อมูลเสียงแล้ว การใช้งานแพ็กเก็ตชนิด DH5 บนช่องสัญญาณ ACL ให้ผลการทดลองที่ดีที่สุด

งานวิจัย [20] พัฒนาโปรเจคต์ BlueEar เป็นระบบช่วยฟังต้นแบบ ซึ่งทำหน้าที่เป็นสื่อกลางระหว่างผู้ใช้งานกับอุปกรณ์ต่างๆ ไม่ว่าจะเป็นโทรศัพท์ กริ่งประตูบ้าน เป็นต้น ระบบ BlueEar ได้ถูกทำการทดสอบในสามประเทศ ซึ่งผลการทดสอบไม่เป็นที่น่าพอใจเนื่องจากสัญญาณรบกวน ถึงแม้จะมีข้อดีที่ระบบถูกออกแบบให้มีการสื่อสารแบบสองทางและรองรับอุปกรณ์ต่างๆได้หลายชนิดได้ และเนื่องจากงานวิจัยไม่ได้ระบุชัดเจนว่าการสื่อสารระหว่างผู้ใช้กับอุปกรณ์เป็นการสื่อสารแบบ Multicast แต่จากการที่ผู้ใช้จำเป็นต้องปรับค่าเริ่มต้นของระบบทุกครั้งเพื่อให้สอดคล้องกับชนิดกับอุปกรณ์ที่ผู้ใช้ต้องการติดต่อด้วย จึงทำให้เข้าใจได้ว่าระบบมีการสื่อสารระหว่างผู้ใช้และอุปกรณ์ต่างๆเป็นแบบ Unicast

บทที่ 3
วิธีดำเนินการวิจัย

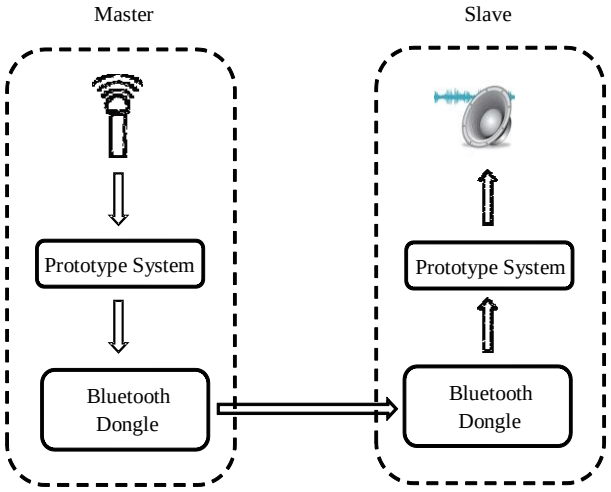
3.1 การออกแบบและพัฒนาระบบต้นแบบ

ระบบต้นแบบมีลักษณะการทำงานเป็นเรียลไทม์แอพพลิเคชัน (Real-Time Application) ที่มีการสื่อสารทางเดียว (One Way Communication) จากครูไปยังนักเรียน



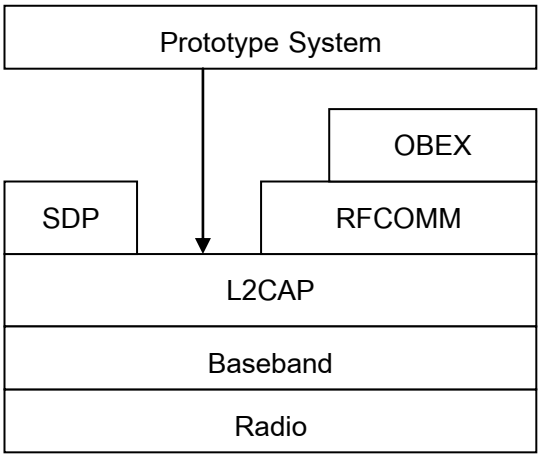
รูปที่ 31 ลักษณะการทำงานของระบบสื่อสารไร้สายต้นแบบ

ซึ่งมีรายละเอียดการทำงานดังนี้



รูปที่ 32 รายละเอียดการทำงานของระบบสื่อสารไร้สายต้นแบบ

การทำงานของระบบต้นแบบเป็นการทำงานในรูปแบบ Client-Server ซึ่งแบ่งการทำงานเป็น 2 ส่วน ได้แก่ ฝั่งส่งหรือ Master ทำหน้าที่เป็น Server ให้บริการการส่งเสียงพูดจากครูไปยังนักเรียน ฝั่งรับหรือ Slave ทำหน้าที่เป็น Client คอยรับบริการการส่งเสียงพูดจากฝั่งส่งหรือ Master



รูปที่ 33 การทำงานของระบบต้นแบบร่วมกับเทคโนโลยีบลูทูธ

รูปที่ 33 อธิบายถึงการพัฒนาระบบต้นแบบในส่วนของแอพพลิชันร่วมกับเทคโนโลยีบลูทูธ โดยระบบสื่อสารไร้สายต้นแบบทำงานอยู่เหนือชั้นการทำงาน L2CAP ของเทคโนโลยีบลูทูธ เพื่อเลือกใช้ช่องสัญญาณ ACL

3.1.1 ส่วนประกอบพื้นฐานของระบบ

ระบบสื่อสารต้นแบบประกอบด้วยส่วนประกอบพื้นฐานสำหรับการทำงานดังนี้

1. Bluetooth Dongles ที่มีจำหน่ายทั่วไปในท้องตลาด ในที่นี้คือ Bluetooth Version 4.0 Class 2 ซึ่งรองรับการทำงานระยะ 10 เมตร [3]

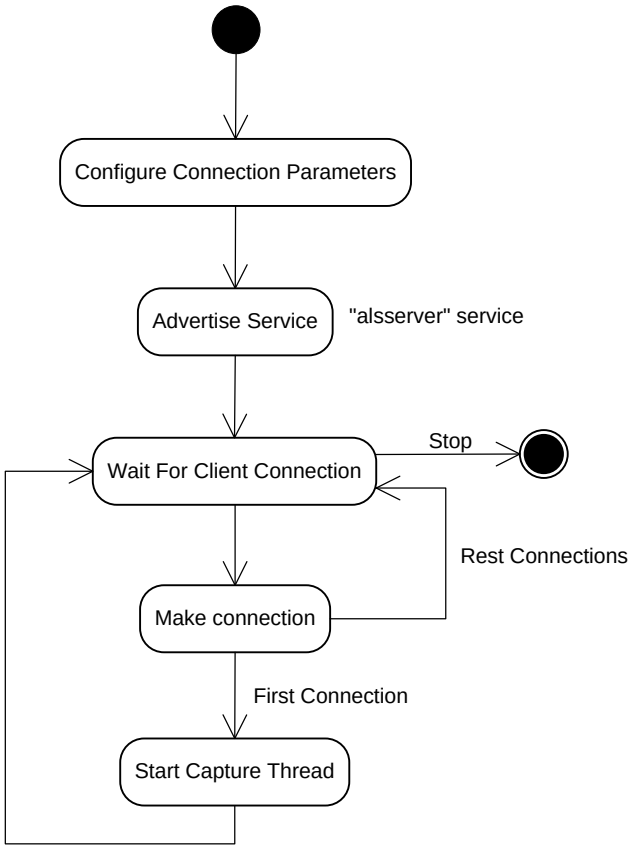
รูปที่ 34 Bluetooth Dongles

2. ไลบรารี่สำหรับการพัฒนาบลูทูธแอพพลิเคชันด้วยภาษาจาวา Bluecove (JSR-82 Implementation)
3. ระบบปฏิบัติการลินุกซ์ Ubuntu ที่มีแพ็กเก็จสำคัญสำหรับการทำงานของบลูทูธดังนี้ bluez และ libbluetooth-dev [20]
4. Java JDK สำหรับการพัฒนาแอพพลิเคชันที่พัฒนาด้วยภาษาจาวา ในที่นี้คือ จาวา เวอร์ชัน 1.6
5. Java Sound API สำหรับจัดการเสียงพูดที่รับจากไมโครโฟนและเล่นออกสู่ลำโพง [21]

3.1.2 **แผนภาพกิจกรรม** (Activitiy Diagram)

การทำงานของระบบต้นแบบสามารถอธิบายได้โดยแบ่งตามหน้าที่และรูปแบบการเชื่อมต่อ โดยรูปแบบการเชื่อมต่อแบบพิโคเน็ต ประกอบด้วยอุปกรณ์ 2 ชนิด คือ Server ซึ่งทำหน้าที่ส่งเสียงจากครูไปยังนักเรียน และ Client ซึ่งรับเสียงจากครู

1. Server ประกอบด้วยการทำงาน 2 ส่วน คือ ส่วนสร้างการเชื่อมต่อและส่วนการส่งเสียง ซึ่งอธิบายได้ดังนี้



รูปที่ 35 การเชื่อมต่อของ Server

Server เริ่มต้นการเชื่อมต่อกับ Client โดยกำหนดค่าพารามิเตอร์ต่างๆที่จำเป็นสำหรับการสื่อสารกับ Client ได้แก่

โปรโตคอลที่ใช้ในการส่งข้อมูล กำหนดให้ใช้โปรโตคอล L2CAP

ขนาดข้อมูลสูงสุดที่ใช้ในการรับส่งข้อมูล กำหนดให้เท่าเท่ากับ 40000 ไบต์

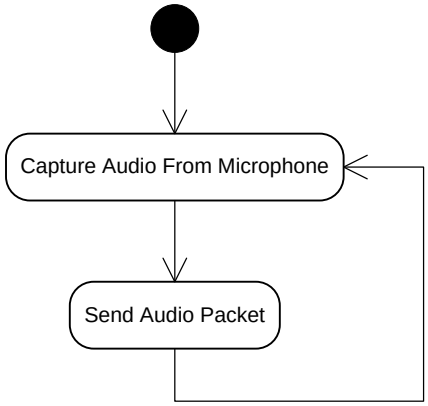
สถานะของอุปกรณ์ในพิโคเน็ต กำหนดให้มีสถานะเป็น Master

Authentication กำหนดให้ไม่มีการป้อน Pairing Pin สำหรับการเชื่อมต่อ

จากนั้นจึงประกาศบริการที่ตนเองให้บริการ ในที่นี้คือ “alsserver” แล้วจึงเข้าสู่สถานะการรอการเชื่อมต่อจาก Client

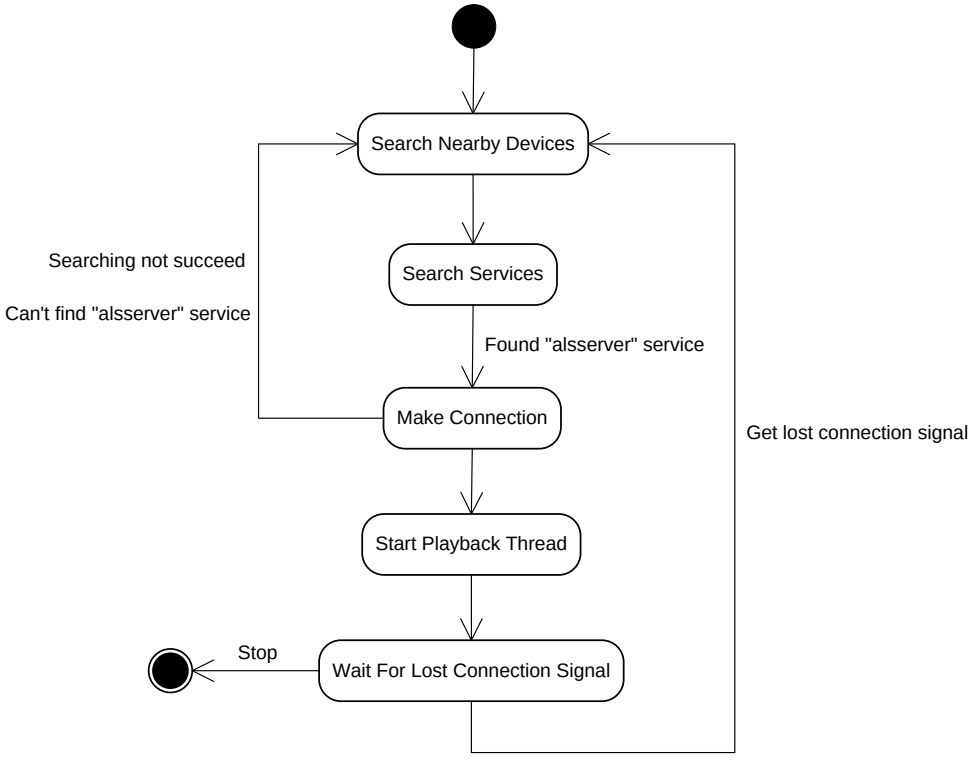
ทันทีที่การสร้างการเชื่อมต่อเสร็จสิ้น Server จะกลับเข้าสู่สถานะรอคอยเชื่อมต่อจาก Client อีกครั้ง เพื่อรอการเชื่อมต่อจาก Client ตัวถัดไป โดย Server จะเริ่มการส่งเสียงทันทีที่การเชื่อมต่อกับ Client เสร็จสิ้นซึ่งหมายถึงการสร้างพิโคเน็ตเสร็จสมบูรณ์

สำหรับส่วนการส่งเสียง Server จะรับเสียงจากไมโครโฟน แล้วจึงส่งเสียงดังกล่าวผ่านช่องสัญญาณ ACL ด้วยค่าพารามิเตอร์ที่กำหนดไว้ตอนเริ่มต้นสร้างการเชื่อมต่อ จากนั้นจึงกลับมาทำงานตามกระบวนการเดิมสำหรับเสียงชุดถัดไป



รูปที่ 36 การส่งเสียงของ Server

2. Client ประกอบด้วยการทำงาน 2 ส่วน คือ ส่วนการสร้างการเชื่อมต่อกับ Server และส่วนการเล่นเสียง ซึ่งอธิบายได้ดังนี้

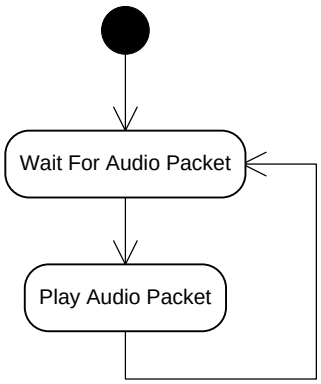


รูปที่ 37 การเชื่อมต่อของ Client

Client เริ่มต้นการทำงานโดยค้นหาอุปกรณ์ที่อยู่ในบริเวณใกล้เคียง จากนั้นจึงทำการค้นหาบริการที่ต้องการจาก Server ซึ่งในที่นี้คือ “alsserver” หากพบบริการที่ต้องการ Client เริ่มต้นสร้างการเชื่อมต่อกับ Server ต่อไป แต่หากไม่พบบริการที่ต้องการ Client จะกลับเข้าสู่สถานะค้นหาอุปกรณ์ใกล้เคียงอีกครั้ง

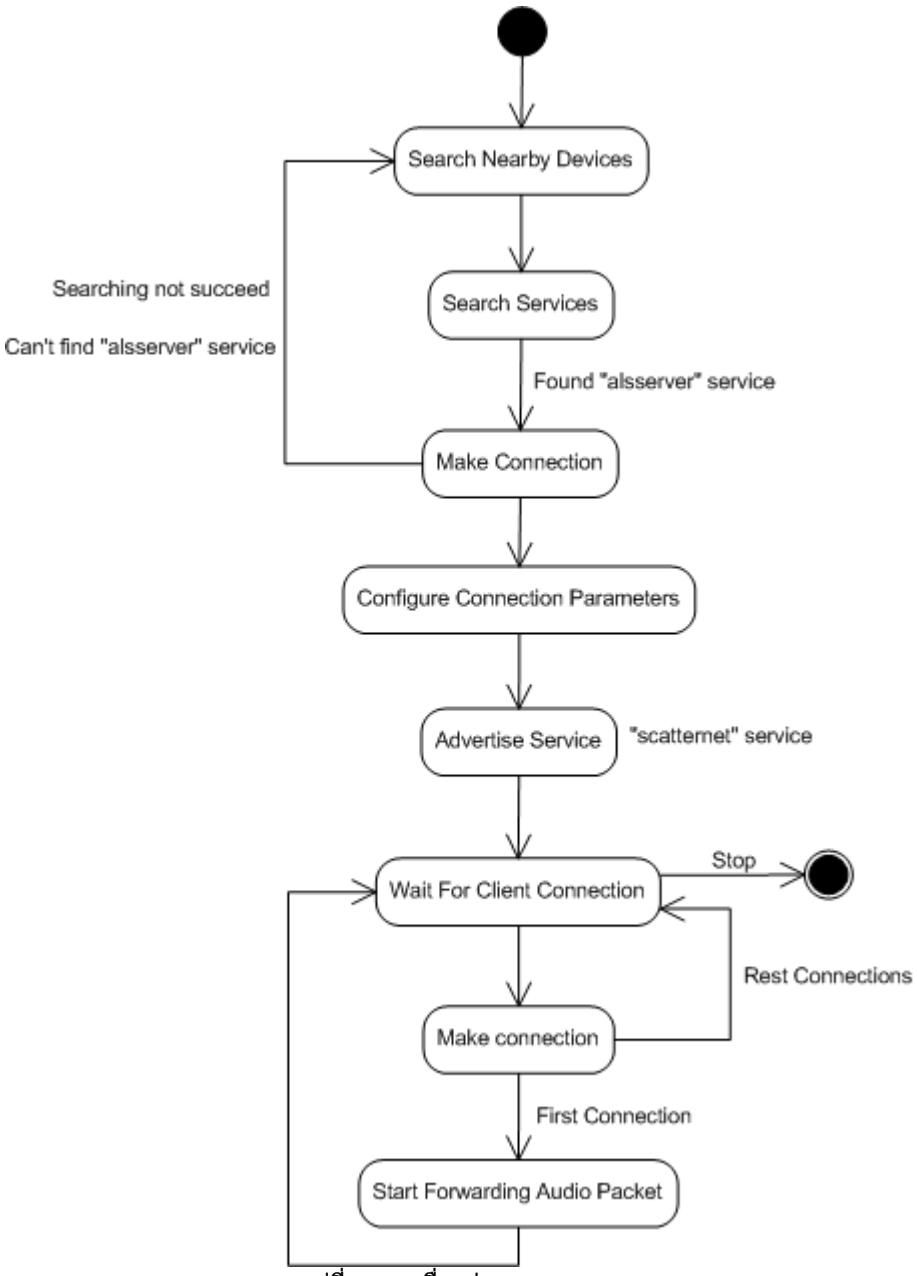
เมื่อการสร้างการเชื่อมต่อกับ Server เสร็จสิ้น ซึ่งหมายถึงการสร้างพิโคเน็ตเสร็จสมบูรณ์ Client จึงเริ่มการเล่นเสียงทันที พร้อมทั้งเข้าสู่สถานะรอคอยสัญญาณขาดการเชื่อมต่อ เพื่อให้สามารถทำการเชื่อมต่อกับ Server ได้อัตโนมัติในกรณีที่การเชื่อมต่อกับ Server ขาดหาย

สำหรับส่วนเล่นเสียง Client จะคอยรับเสียงจาก Server ผ่านช่องสัญญาณ ACL จากนั้นจึงส่งเสียงดังกล่าวเพื่อเล่นผ่านหูฟังหรือลำโพงต่อไป



รูปที่ 38 การเล่นเสียงของ Client

3. Scatternet เป็นรูปแบบการเชื่อมต่อเพื่อให้ระบบต้นแบบรองรับจำนวนนักเรียนมากกว่า 7 คน ซึ่งในรูปแบบการเชื่อมแบบ Scatternet อุปกรณ์อย่างน้อยตัวหนึ่งซึ่งเป็นสมาชิกของ Scatternet จะทำหน้าที่เป็นทั้ง Master ของพิโคเน็ตหนึ่ง และเป็น Slave ของอีกพิโคเน็ตหนึ่ง ซึ่งอุปกรณ์ดังกล่าวมีการทำงานดังนี้



รูปที่ 39 การเชื่อมต่อของ Scatternet

อุปกรณ์เริ่มต้นการทำงานในฐานะ Client ของพิโคเน็ตหนึ่งก่อน โดยค้นหาอุปกรณ์ใกล้เคียง จากนั้นจึงค้นหาบริการที่ต้องการจาก Server ในที่นี้คือ “alsserver” หากพบบริการดังกล่าวก็จะทำการสร้างการเชื่อมต่อในฐานะ Client ของพิโคเน็ตนั้นต่อไปจนเสร็จสมบูรณ์ หากไม่พบบริการดังกล่าว อุปกรณ์จะเริ่มต้นค้นหาอุปกรณ์ใกล้เคียงอีกครั้ง

เมื่อการเชื่อมต่อในฐานะ Client ของพิโคเน็ตหนึ่งเสร็จสิ้น อุปกรณ์จึงเริ่มสร้างการเชื่อมต่อในฐานะ Server ของอีกพิโคเน็ตหนึ่ง โดยเริ่มจากการกำหนดค่าพารามิเตอร์ต่างๆที่จำเป็นสำหรับการสื่อสารกับ Client ได้แก่

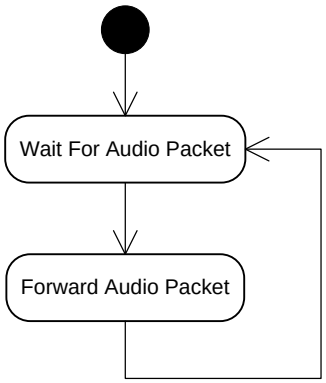
โปรโตคอลที่ใช้ในการส่งข้อมูล กำหนดเป็น L2CAP

ขนาดข้อมูลสูงสุดที่ใช้ในการรับส่งข้อมูล กำหนดให้เท่ากับ 40000 ไบต์

สถานะของอุปกรณ์ในพิโคเน็ต กำหนดให้สถานะเป็น Master

Authentication กำหนดให้ไม่มีการป้อน Pairing Pin สำหรับการเชื่อมต่อ

จากนั้นจึงประกาศบริการที่ตนเองให้บริการ ซึ่งในที่นี้คือ “scatternet” แล้วจึงกลับเข้าสู่การรอคอยการเชื่อมต่อจาก Slave ตัวถัดไป โดย Scatternet จะเริ่มส่งต่อเสียงจาก Server ในพิโคเน็ตที่หนึ่งไปยัง Client ในพิโคเน็ตที่สอง เมื่อการเชื่อมต่อระหว่าง Server และ Client ในพิโคเน็ตที่สองเสร็จสิ้น ซึ่งหมายถึงการสร้าง Scatternet เสร็จสมบูรณ์

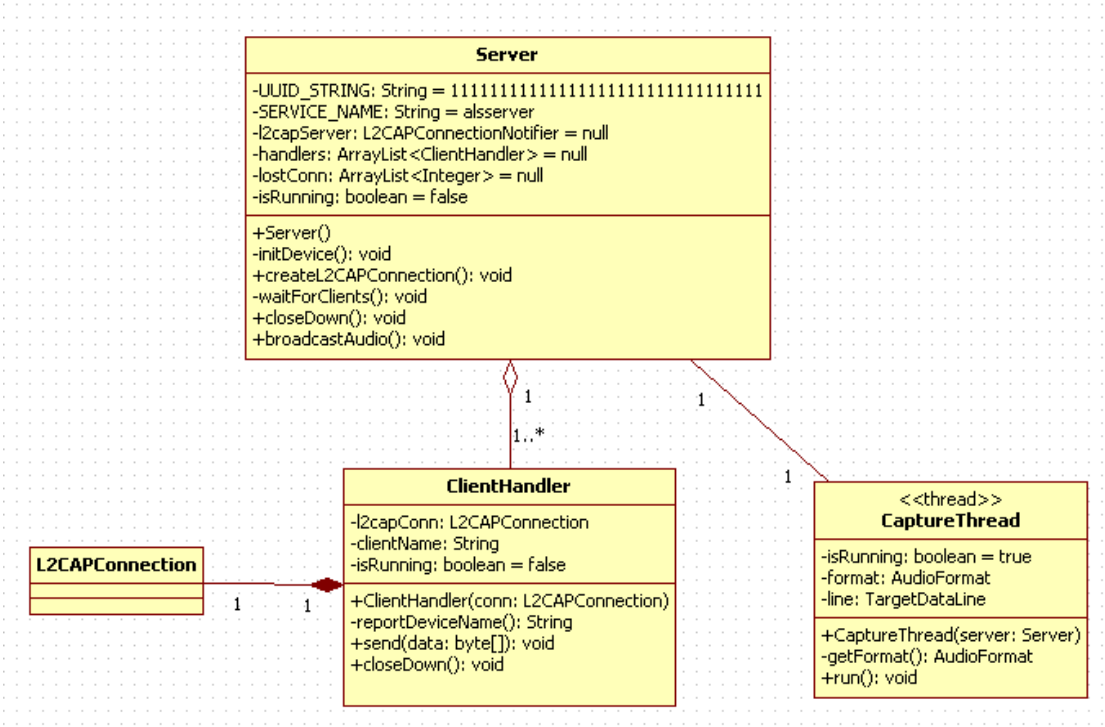


รูปที่ 40 ส่วนส่งต่อแพกเก็ตเสียง

3.1.3 **แผนภาพคลาส** (Class Diagram)

เพื่ออธิบายถึงรูปแบบความสัมพันธ์ในการทำงานร่วมกันระหว่างคลาสต่างๆ สามารถอธิบายได้ตามสถานะของอุปกรณ์ดังนี้

1. Server



รูปที่ 41 Server Class Diagram

การทำงานของคลาสต่างๆในสถานะ Server หรือ Master ของพิโคเน็ตประกอบด้วยคลาสต่างๆดังนี้

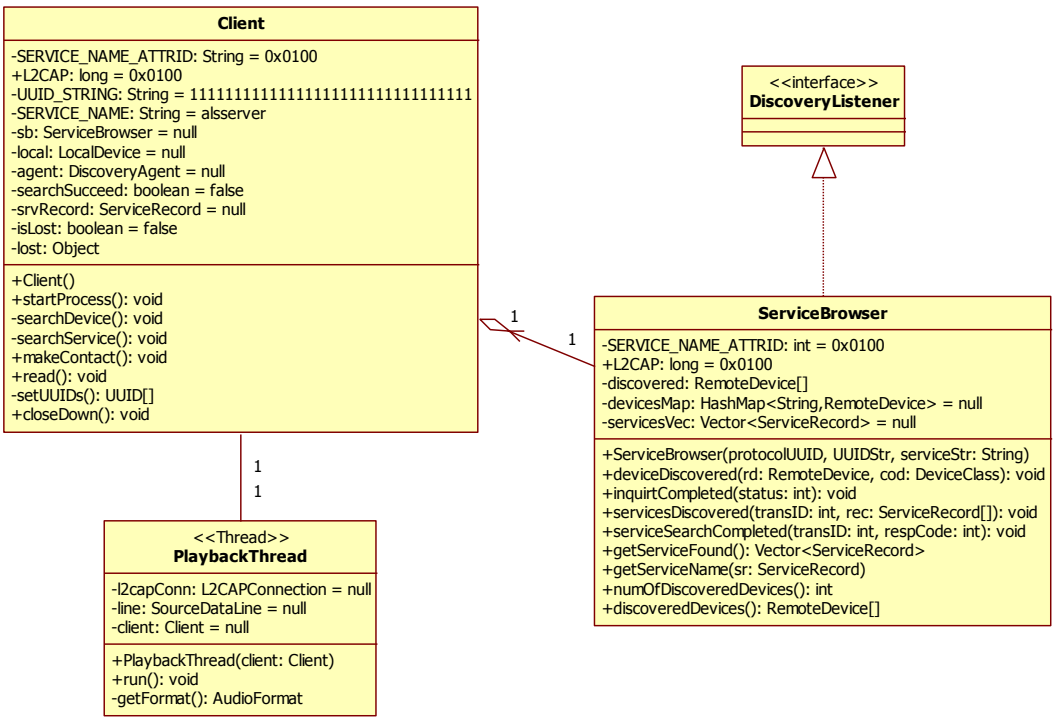
คลาส Server ทำหน้าที่เกี่ยวกับการสร้างการเชื่อมต่อกับ Client โดยเฉพาะค่าพารามิเตอร์ที่จำเป็นสำหรับการส่งข้อมูลระหว่าง Server และ Client รวมถึงการบริการที่จะให้บริการ นอกจากนี้ยังทำหน้าที่ส่งเสียงให้กับ Client ทุกตัวที่เป็นสมาชิกของพิโคเน็ตอีกด้วย

คลาส ClientHandler เป็นส่วนหนึ่งของคลาส Server โดยทำหน้าที่เก็บออบเจกต์ของ L2CAPConnection ซึ่งแทนการเชื่อมต่อระหว่าง Server และ Client แต่ละตัว ออบเจกต์ของ Server จะเก็บออบเจกต์ของ L2CAPConnection ไว้ใน ArrayList ซึ่งเป็นการเก็บข้อมูลรูปแบบหนึ่งในภาษาจาวา

คลาส CaptureThread ทำหน้าที่คอยรับเสียงจากไมโครโฟนพร้อมกับส่งเสียงดังกล่าวให้กับ Server เพื่อส่งให้กับ Client ต่อไป โดยสามารถกำหนดคุณสมบัติต่างๆ

สำหรับการรับเสียงจากไมโครโฟนได้ ไม่ว่าจะเป็น Sample Rate, Bit per Sample หรือ Channel ในที่นี้กำหนดให้เป็น PCM(Pulse Code Modulation) ซึ่งมี Sample Rate ที่ 8000 Hz 8 Bits per Sample และ Channel เป็น Mono ซึ่งให้บิตเรตที่ 64kbps

2. Client



รูปที่ 42 Client Class Diagram

การทำงานของคลาสต่างๆในสถานะ Client หรือ Slave ของพิโคเน็ตประกอบด้วยคลาสต่างๆดังนี้

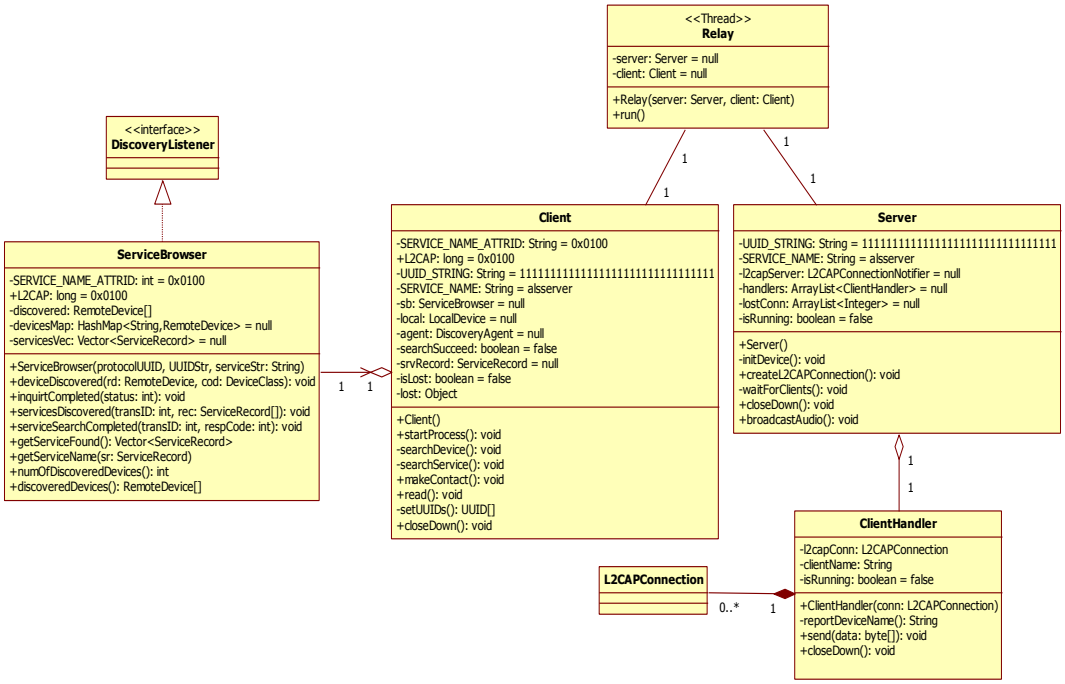
คลาส Client ทำหน้าที่เกี่ยวกับการเชื่อมต่อกับ Server โดยเฉพาะค่าพารามิเตอร์ที่จำเป็นสำหรับการส่งข้อมูลระหว่าง Server และ Client รวมถึงการบริการที่ต้องการจาก Server

คลาส ServiceBrowser เป็นส่วนหนึ่งของคลาส Client ทำหน้าที่ในการค้นหาอุปกรณ์ใกล้เคียง รวมถึงค้นหาบริการที่ Client ต้องการจาก Server โดยคลาส ServiceBrowser ต้องสืบทอดและ implement เมธอดจากอินเตอร์เฟส DiscoveryListener

คลาส PlaybackThread ทำหน้าที่ในการเล่นเสียงที่ได้รับมาจาก Server ออกสู่หูฟังหรือลำโพง และเพื่อให้สามารถเล่นเสียงที่รับมาจาก Server ได้อย่างถูกต้อง จึงต้อง

กำหนดคุณสมบัติของเสียงให้สอดคล้องกับฝั่ง Server นั่นคือ Sample Rate ที่ 8000 Hz 8 Bits per Sample และ Channel เป็น Mono ซึ่งให้บิตเรตที่ 64kbps

3. Scatternet



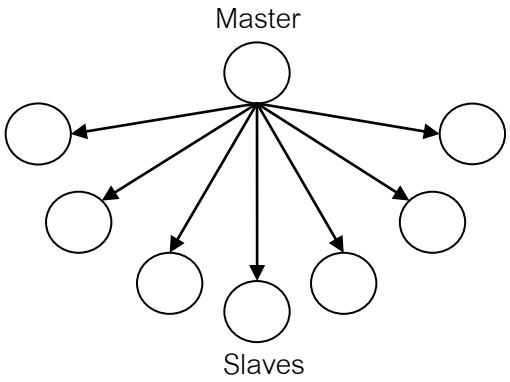
รูปที่ 43 Scatternet Class Diagram

การทำงานของคลาสต่างๆในการเชื่อมต่อแบบ Scatternet ประกอบด้วยคลาสต่างๆที่มีความสัมพันธ์กันเช่นเดียวกับการเชื่อมต่อในรูปแบบพิโคเน็ต เนื่องจาก Scatternet ประกอบด้วย 2 พิโคเน็ต ซึ่งจะมีอุปกรณ์อย่างน้อยหนึ่งตัว ทำหน้าที่เป็น Master หรือ Server ในพิโคเน็ตหนึ่ง และทำหน้าที่เป็น Slave หรือ Client ในอีกพิโคเน็ตหนึ่ง จึงมีเพียงคลาส Relay เท่านั้นที่ถูกเพิ่มเข้ามาเพื่อเชื่อมต่อการส่งข้อมูลระหว่าง 2 พิโคเน็ตดังกล่าว โดยทำหน้าที่ในการส่งต่อข้อมูลเสียงจาก Server ของพิโคเน็ตหนึ่งไปยัง Client ของอีกพิโคเน็ตหนึ่ง

3.2 การทดลองระบบสื่อสารไร้สายต้นแบบ

เนื่องจากระบบต้นแบบเป็นระบบที่มีการทำงานเป็นแอพพลิเคชันแบบเรียลไทม์ (Real-Time Application) พารามิเตอร์ที่สำคัญซึ่งแสดงถึงคุณภาพของระบบ ได้แก่ อัตราเร็วในการส่งข้อมูล (Data Rate) และระยะเวลาที่ใช้ในการส่งข้อมูล (Latency) การทดลองจึงเป็นการวัดค่าพารามิเตอร์ทั้งสองตัวดังกล่าว เพื่อบ่งชี้ถึงคุณภาพของระบบต้นแบบ โดยทำการทดลองตามรูปแบบการเชื่อมต่อของอุปกรณ์ ดังนี้

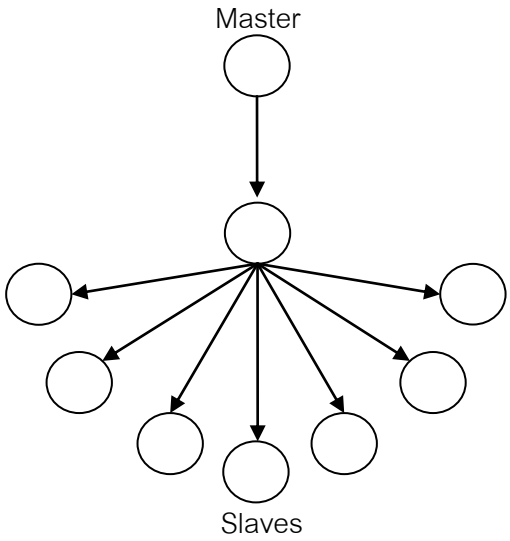
1. พิโคเน็ต (Piconet)



รูปที่ 44 รูปแบบการเชื่อมต่อแบบพิโคเน็ต

เพื่อทดสอบความเป็นไปได้ในการส่งข้อมูลแบบแพร่สัญญาณ (one-to-many) ของระบบต้นแบบ จึงกำหนดให้ระบบต้นแบบสร้างการเชื่อมต่อแบบพิโคเน็ตซึ่งมีจำนวนสมาชิกได้สูงสุด 7 ตัว [3] แล้วจึงวัดอัตราเร็วในการส่งข้อมูลและระยะเวลาที่ใช้ในการส่งข้อมูล

2. Scatternet



รูปที่ 45 รูปแบบการเชื่อมต่อแบบ Scatternet

เพื่อทดสอบการส่งข้อมูลเสียงของระบบต้นแบบในกรณีที่ต้องรองรับจำนวนนักเรียนมากกว่า 7 คน จึงกำหนดให้ระบบต้นแบบสร้างรูปแบบการเชื่อมต่อแบบ Scatternet แล้วจึงอัตราเร็วในการส่งข้อมูลและระยะเวลาที่ใช้ในการส่ง

สำหรับการวัดอัตราเร็วในการส่งข้อมูล (Data Rate) และระยะเวลาที่ใช้ในการส่งข้อมูล (Latency) มีขั้นตอนดังนี้

- อัตราเร็วในการส่งข้อมูล (Data Rate) เนื่องจากทำ Time Synchronization ระหว่าง Server และ Client ด้วย NTP Server ก็ให้ความต่างของเวลาระหว่าง Server และ Client ในระดับหลักสิบมิลลิวินาที ซึ่งมีผลทำให้การคำนวณอัตราเร็วในการส่งข้อมูลคลาดเคลื่อน จึงอาศัยการวัดโดยบันทึกเวลาที่เริ่มรับข้อมูลที่ Client และบันทึกเวลาอีกครั้งเมื่อ Client ได้รับข้อมูลเสร็จสิ้น จากนั้นจึงนำเวลาดังกล่าวมาคำนวณร่วมกับขนาดของข้อมูลเสียงที่ทำการส่งข้อมูล จึงได้อัตราเร็วในการส่งข้อมูลระหว่าง Server และ Client
- ระยะเวลาที่ใช้ในการส่งข้อมูล (Latency) อาศัย Oscilloscope และ Function Generator โดย Function Generator ทำหน้าที่สร้างสัญญาณแทนเสียงพูด และ Oscilloscope ทำหน้าที่วัดสัญญาณที่ออกจาก Function Generator เทียบกับสัญญาณที่ออกจากลำโพงของ Client โดยกำหนดสัญญาณแทนเสียงพูดเป็นคลื่นรูปซายน์ (Sine Wave) ขนาด 1 Vpp และความถี่ 50 Hz

รูปที่ 46 การวัดระยะเวลาที่ใช้ในการส่งข้อมูล (Latency)

บทที่ 4
การทดลองและการวิเคราะห์ผล

การทดลองและวิเคราะห์ผลการทดลองนี้เป็นการทดลองและวิเคราะห์ผลการทดลองตามขอบเขตของการวิจัยดังนี้

4.1 การสื่อสารจากครูถึงนักเรียนมากกว่า 7 คน

การทดลองนี้เป็นการทดลองระบบต้นแบบเพื่อรองรับจำนวนนักเรียนในระบบ รวมถึงวิเคราะห์ขนาดของข้อมูลและขนาดของบัฟเฟอร์ โดยมีอัตราเร็วในการส่งข้อมูล (Data Rate) และระยะเวลาที่ใช้ในการส่งข้อมูล (Latency) เป็นตัวชี้วัดซึ่งแสดงถึงคุณภาพของระบบ

4.1.1 การเชื่อมต่อแบบพิโคเน็ต (Piconet)

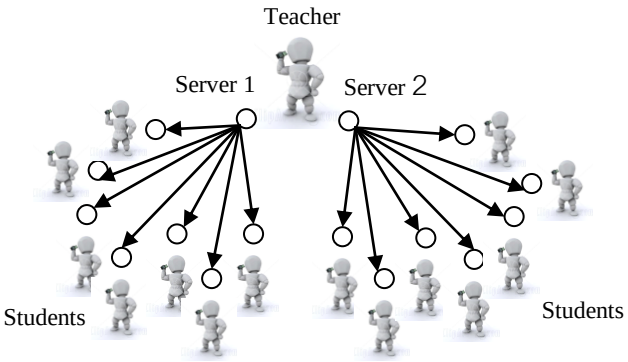
การเชื่อมต่อแบบพิโคเน็ตเป็นรูปแบบการเชื่อมต่อที่ประกอบด้วยอุปกรณ์ซึ่งทำหน้าที่เป็น Master หรือ Server ผู้ให้บริการการส่งเสียงจากครูไปยังนักเรียน 1 ตัว และอุปกรณ์ที่ทำหน้าที่เป็น Slave หรือ Client ผู้คอยรับบริการการส่งเสียงที่ตัวนักเรียนอีก 7 ตัว โดยแบ่งการทดลองเป็น 2 ส่วน ดังนี้

1. <u>จำนวนสมาชิกของพิโคเน็ต</u>

การทดลองนี้ทำการทดลองโดยการให้ Server ทำการเชื่อมต่อกับ Client ทีละตัวเพื่อหาจำนวนสมาชิกสูงสุดที่พิโคเน็ตสามารถรองรับได้ โดยทั้ง Server และ Client จะทำงานตามกิจกรรมที่ได้ออกแบบไว้ในส่วนของการออกแบบและพัฒนาระบบในหัวข้อ 3.1.2 นั่นคือ หลังจากการประกาศบริการที่ตนเองจะให้บริการ Server จะเข้าสู่สถานะรอคอยการเชื่อมต่อจาก Client จากนั้นจึงกำหนดให้ Client ตัวที่หนึ่งเริ่มสร้างการเชื่อมต่อกับ Server หลังจากพบบริการที่ตนเองต้องการ เมื่อการเชื่อมต่อระหว่าง Server และ Client ตัวที่หนึ่งเสร็จสิ้น Server จะกลับเข้าสู่สถานะรอคอยการเชื่อมต่อจาก Client ตัวถัดไป

จากการทดลองพบว่า พิโคเน็ตประกอบด้วย Server ซึ่งทำหน้าที่เป็น Master ของพิโคเน็ต 1 ตัวและ Client ซึ่งทำหน้าที่เป็น Slave ของพิโคเน็ต 7 ตัวซึ่งสอดคล้องกับรายละเอียดของเทคโนโลยีบลูทูธ [3] และหลังจากที่พิโคเน็ตมีจำนวนสมาชิกที่ทำหน้าที่เป็น Slave ของพิโคเน็ตครบ 7 ตัวแล้ว จะไม่สามารถทำการเชื่อมต่อเพื่อเพิ่มจำนวนสมาชิกในพิโคเน็ตได้อีก โดย Slave หรือ Client ลำดับถัดพบ Server อยู่ในบริเวณใกล้เคียงแต่ไม่สามารถสร้างการเชื่อมต่อกับ Server

ได้ ดังนั้นในกรณีที่ระบบต้องการรองรับจำนวนนักเรียนหรือ Client มากกว่า 7 ตัว จึงทำได้ด้วยการเพิ่มจำนวนอุปกรณ์ซึ่งทำหน้าที่เป็น Server ที่ตัวครูให้เป็นสัดส่วนสอดคล้องกับจำนวนนักเรียน



รูปที่ 47 การเชื่อมต่อแบบพิโคเน็ตในกรณีที่รองรับนักเรียน 14 คน

รูปที่ 47 แสดงรูปแบบการเชื่อมต่อแบบพิโคเน็ตเพื่อรองรับจำนวนนักเรียน 14 คน โดยเพิ่มจำนวนอุปกรณ์ซึ่งทำหน้าที่เป็น Server ที่ตัวครูเป็นจำนวน 2 ตัว

2. <u>อัตราเร็วในการส่งข้อมูล (Data Rate) และระยะเวลาที่ใช้ในการส่งข้อมูล (Latency)</u>

การทดลองนี้เป็นการทดลองหาขนาดข้อมูลและขนาดบัฟเฟอร์ที่ส่งผลให้อัตราเร็วในการส่งข้อมูล (Data Rate) มีค่าประมาณ 64 kbps ซึ่งเป็นอัตราเร็วที่เท่ากับการพูดคุยทางโทรศัพท์ [22] และใช้เวลาในการส่งข้อมูล (Latency) ไม่เกิน 100 ms ซึ่งเป็นระยะเวลาที่ยอมรับได้ในระบบที่มีลักษณะเดียวกับการประชุมทางไกล [23] ซึ่งทำการทดลองด้วยการสร้างพิโคเน็ตตามจำนวน Client ตั้งแต่ 1 ถึง 7 แล้วจึงวัดอัตราเร็วในการส่งข้อมูลและระยะเวลาที่ใช้ในการส่งข้อมูลตามวิธีการซึ่งได้อธิบายไว้ในหัวข้อ 3.2

จำนวน Clients	ผลการทดลอง			
	ขนาดข้อมูล (ไบต์)	อัตราเร็วในการส่งข้อมูล (kbps)	ระยะเวลาที่ใช้ในการส่งข้อมูลที่วัดด้วยซอฟท์แวร์ (ms)	ระยะเวลาที่ใช้ในการส่งข้อมูลจริง (ms)
1	20	63.98	4.96	60
2	40	64.28	9.94	56
3	80	64.09	19.92	60
4	120	64.12	29.97	70
5	160	64.23	39.93	66
6	240	64.04	59.77	86
7	320	64.55	79.94	114

ตารางที่ 11 ผลการวัดอัตราเร็วและเวลาที่ใช้ในการส่งข้อมูล

ตารางที่ 11 แสดงให้เห็นว่า สำหรับระบบต้นแบบซึ่ง Server ต้องให้บริการ Client สูงสุดจำนวน 7 ตัว ข้อมูลขนาด 320 ไบต์เป็นขนาดที่เหมาะสมเพราะให้อัตราเร็วในการส่งข้อมูลเท่ากับ 64.55 kbps และอัตราเร็วในการส่งข้อมูลเท่ากับ 114 ms แต่ระบบที่ Server ให้บริการ Client 6 ตัวโดยส่งข้อมูลขนาด 240 ไบต์ ซึ่งใช้เวลาในการส่งข้อมูล 86 ms ซึ่งใช้เวลาน้อยกว่า 100 ms เป็นระบบที่เหมาะสมต่อการนำไปใช้งานจริงมากกว่า

ขนาดของข้อมูลที่เปลี่ยนแปลงไปตามจำนวนสมาชิกซึ่งทำหน้าที่เป็น Client ของระบบนั้น เนื่องจากระบบต้นแบบมีการทำงานแบบ Round Robin ซึ่ง Server จะคอยวนส่งข้อมูลเสียงตั้งแต่ Client ตัวที่ 1 หรือนักเรียนคนที่ 1 ไปจน Client หรือนักเรียนคนสุดท้ายในระบบ ขนาดข้อมูลที่เล็กเกินไป ทำให้นักเรียนแต่ละคนใช้เวลารอคอยข้อมูลเสียงชุดถัดไป ส่งผลให้อัตราเร็วในการส่งข้อมูลลดลง เนื่องจากได้รับข้อมูลขนาดเดิมแต่ใช้เวลาในการรอคอยรับข้อมูลนานขึ้น และเป็นผลให้คุณภาพเสียงที่ผู้ฟังหรือนักเรียนได้ยินจะเป็นเสียงที่กระตุกและขาดตอน ซึ่งเป็นสาเหตุให้ไม่เข้าใจข้อความที่ครูพูด สำหรับขนาดของข้อมูลที่ใหญ่เกินไป ถึงแม้จะไม่ทำให้เกิดเสียงกระตุกหรือคุณภาพเสียงแย่ลง แต่ขนาดข้อมูลที่ใหญ่ขึ้นจะส่งผลให้ใช้เวลาในการส่งข้อมูลมากขึ้น นั่นคือสำหรับระบบต้นแบบที่ Server ต้องการให้บริการ Client สูงสุดจำนวน 7 ตัว ข้อมูลขนาดน้อยกว่า 320 ไบต์ทำให้อัตราเร็วในการส่งข้อมูลลดลงซึ่งทำให้ผู้ฟังหรือนักเรียนได้ยินเสียงกระตุกและขาด

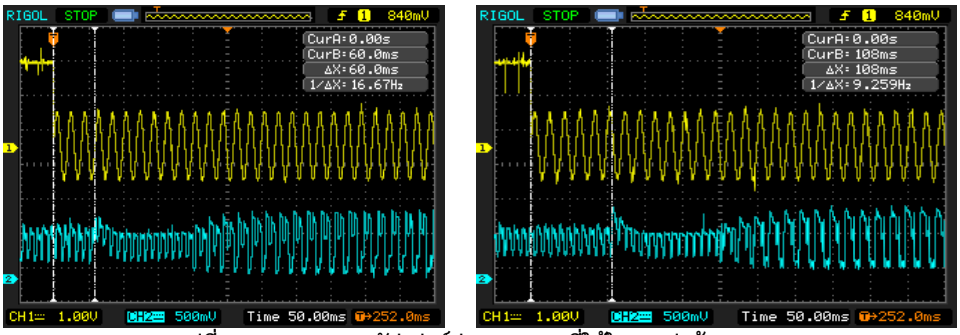
ตอน และข้อมูลขนาดมากกว่า 320 ไบต์ แม้จะไม่ทำให้อัตราเร็วในการส่งข้อมูลลดลง แต่ก็ทำให้ใช้เวลาในการส่งข้อมูลเพิ่มขึ้น ซึ่งหากมีค่าเกิน 100 ms ผู้ฟังสามารถจะรู้สึกถึงความแตกต่างของระยะเวลาของเสียงที่ได้ยินจากผู้พูดและเสียงที่ได้ยินจากหูฟังหรือลำโพง และสำหรับผู้ฟังบางคนอาจจะเริ่มรู้สึกถึงความแตกต่างดังกล่าวที่ 50 ms [23] และผู้ฟังจะไม่รู้สึกถึงความแตกต่างดังกล่าวเลยหากระยะเวลาที่ใช้ในการส่งข้อมูลน้อยกว่า 20 ms [22] ดังนั้นระยะเวลาในการส่งข้อมูลจึงควรเป็นเวลาที่น้อยที่สุดที่ไม่ทำให้คุณภาพของระบบแย่ลง

นอกจากอัตราเร็วในการส่งข้อมูลแล้ว ตารางที่ 11 ยังแสดงถึงระยะเวลาที่ใช้ในการส่งข้อมูล ได้แก่ ระยะเวลาที่ใช้ในการส่งข้อมูลจริง (Actual Latency) ซึ่งเป็นการวัดโดยใช้ Oscilloscope โดยมี Function Generator สร้างสัญญาณคลื่นรูปซายน์แทนเสียงพูด และระยะเวลาในการส่งข้อมูลที่วัดได้โดยซอฟท์แวร์ (Measured by Software Latency) ซึ่งประกอบด้วย ระยะเวลาที่ถูกดึกออกจากบัฟเฟอร์, ระยะเวลาในการส่งข้อมูลจาก Server ไปยัง Client และระยะเวลาที่ข้อมูลเสียงถูกส่งใส่ลงในบัฟเฟอร์เพื่อเล่นออกสู่ลำโพงหรือหูฟัง

จากการทดลอง การส่งข้อมูลขนาด 320 ไบต์ของระบบต้นแบบใช้ระยะเวลาในการส่งข้อมูลจริงเท่ากับ 114 ms ซึ่งประกอบระยะเวลาที่วัดได้โดยซอฟท์แวร์ด้วยเวลา 80 ms แบ่งเป็นเวลาในการอ่านข้อมูลเสียงจากบัฟเฟอร์ที่ Server ประมาณ 40 ms และเวลาในการรับข้อมูลเสียงและส่งข้อมูลเสียงดังกล่าวลงในบัฟเฟอร์เพื่อเล่นออกสู่ลำโพงหรือหูฟังที่ Client อีก 40 ms รวมกับเวลาซึ่งไม่สามารถวัดได้โดยซอฟท์แวร์อีก 34 ms ซึ่งควรประกอบด้วย ระยะเวลาในการแปลงเสียงจากอนาล็อกเป็นดิจิตัล (Analog to Digital), เวลาในการสร้างแพกเก็ตข้อมูล (Packetization), เวลาในการรอให้บัฟเฟอร์ที่ Client พร้อมที่จะเล่นเสียง และเวลาในการแปลงข้อมูลดิจิตัลกลับเป็นอนาล็อก (Digital to Analog) นอกจากนี้ผลการทดลองยังแสดงให้เห็นอีกว่า เมื่อข้อมูลขนาดเล็กกว่า 80 ไบต์ จะใช้เวลาในการส่งข้อมูลเท่ากันที่ประมาณ 60 ms นั่นคือ ระบบใช้เวลาในการประมวลผลเพื่อส่งข้อมูลอย่างน้อย 60 ms เสมอ

ในเรียลไทม์แอพพลิเคชัน นอกจากขนาดข้อมูลแล้ว ขนาดบัฟเฟอร์ก็เป็นพารามิเตอร์สำคัญ โดยเฉพาะอย่างยิ่งในแอพพลิเคชันที่ความเร็วในการส่งข้อมูลในขณะเข้าสู่ระบบและออกจากระบบไม่เท่ากัน ขนาดบัฟเฟอร์ที่เหมาะสมจะกำหนดทั้งคุณภาพของเสียงและระยะเวลาในการส่งข้อมูล ขนาดบัฟเฟอร์ที่เล็กเกินไปมีผลกับคุณภาพเสียงซึ่งจะมีเสียงกระตุกและผู้ฟังไม่สามารถเข้าใจข้อความจากผู้พูดได้ ขนาดบัฟเฟอร์ที่ใหญ่เกินไปสามารถแก้ปัญหาคุณภาพเสียงได้แต่จะใช้เวลาในการส่งข้อมูลนานขึ้นเพราะข้อมูลต้องถูกเก็บลงในบัฟเฟอร์ก่อนที่จะถูกอ่านจากบัฟเฟอร์ขึ้นมาเล่นออกสู่หูฟังหรือลำโพง

สำหรับระบบต้นแบบนี้ เนื่องจากมีอัตราเร็วในการส่งข้อมูลที่สอดคล้องกันทั้งหมดตั้งแต่การรับเสียงพูดจากไมโครโฟนด้วยการ Encode แบบ PCM ความเร็วในการส่งข้อมูลจาก Server ไปยัง Client และการเล่นเสียงออกสู่ลำโพงด้วยการ Decode แบบ PCM ซึ่งทั้งหมดให้อัตราเร็วที่ 64 kbps บัฟเฟอร์จึงไม่มีความจำเป็นในแง่ของการปรับปรุงคุณภาพเสียงเนื่องจากอัตราเร็วที่แตกต่างกันในแต่ละส่วนการทำงานของระบบ แต่เนื่องจาก Java Sound API กำหนดให้ต้องมีการใช้งานบัฟเฟอร์เสมอเพื่อเป็น Jitter Buffer ป้องกันปัญหา Jitter ที่เป็นปัญหาโดยทั่วไปของระบบประเภทเรียลไทม์แอพพลิเคชันทั้งในกรณีรับเสียงจากไมโครโฟนและเล่นเสียงออกสู่ลำโพง ขนาดของบัฟเฟอร์ที่กำหนดสำหรับระบบต้นแบบจึงมีค่าเท่ากับขนาดข้อมูล ในที่นี้คือ 320 ไบต์ การกำหนดขนาดของบัฟเฟอร์น้อยกว่าขนาดข้อมูลหรือ 320 ไบต์ จะทำให้ข้อมูลเสียงบางส่วนถูกทิ้งเพื่อให้สามารถเก็บข้อมูลเสียงลงในบัฟเฟอร์ได้ จึงส่งผลต่อคุณภาพเสียงทำให้เกิดเสียงกระตุก ในขณะที่ขนาดบัฟเฟอร์มากกว่า 320 ไบต์ ก็ไม่มีผลให้คุณภาพเสียงดีขึ้น แต่จะทำให้ระบบใช้ระยะเวลาในการส่งข้อมูลมากขึ้นเนื่องจากต้องใช้เวลาเก็บข้อมูลเสียงลงในบัฟเฟอร์ซึ่งมีขนาดใหญ่ขึ้น ซึ่งเป็นเวลาที่ไม่จำเป็นสำหรับระบบ

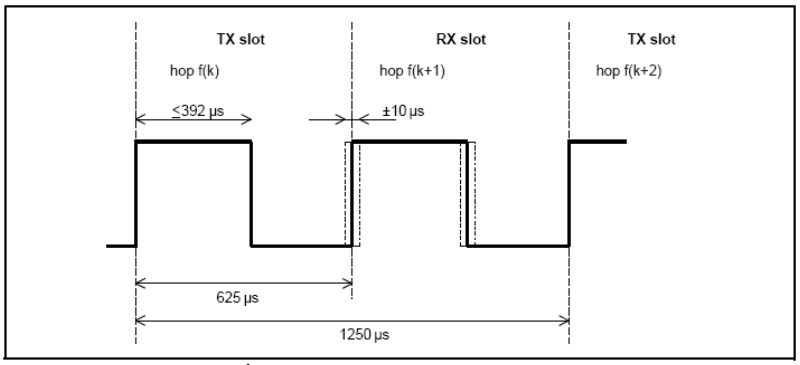


รูปที่ 48 ผลของขนาดบัฟเฟอร์ต่อระยะเวลาที่ใช้ในการส่งข้อมูล

รูปที่ 48 เปรียบเทียบผลของระยะเวลาในการส่งข้อมูลเมื่อเปลี่ยนขนาดบัฟเฟอร์ โดยกำหนดให้ใช้ขนาดข้อมูลเท่ากัน ในที่นี้คือ ขนาดข้อมูลเท่ากับ 20 ไบต์ และขนาดบัฟเฟอร์เท่ากับ 20 ไบต์และ 320 ไบต์ตามลำดับ ซึ่งแสดงให้เห็นว่า เมื่อขนาดบัฟเฟอร์เท่ากับ 20 ไบต์ ระยะเวลาในการส่งข้อมูลเท่ากับ 60 ms และบัฟเฟอร์ขนาด 320 ไบต์ระยะเวลาในการส่งข้อมูลเท่ากับ 108 ms นั่นคือ ระบบใช้เวลาในการส่งข้อมูลมากขึ้นเมื่อบัฟเฟอร์มีขนาดใหญ่ขึ้น

นอกจากนี้ ถึงแม้การส่งเสียงแบบ Round Robin จากครูไปยังนักเรียนแต่ละคนจะเป็นการส่งข้อมูลที่ไม่พร้อมกัน คือ Master ส่งข้อมูลให้กับ Slave ตัวที่หนึ่งก่อนจากนั้นจึงส่งข้อมูลให้กับ Slave ตัวถัดไปตามลำดับจนครบตามจำนวนสมาชิกในพิโคเน็ต รูปแบบการส่งข้อมูลในลักษณะนี้ก็ไม่ได้ทำให้นักเรียนรู้สึกถึงความแตกต่างของเวลาดังกล่าว เนื่องจากเทคโนโลยีบลูทูธ

ใช้รูปแบบการส่งข้อมูลแบบ TDD (Time Division Duplex) ซึ่ง Master ของพิโคเน็ตหรือในที่นี้คือ Server ที่ตัวครู จะทำการส่งข้อมูลให้กับ Client ทุกตัวในพิโคเน็ตทุกๆ 1.25 ms เป็นระยะเวลา 625 µs ดังรูปที่ 49



รูปที่ 49 รอบการรับส่งข้อมูลของ Master [3]

ในกรณีที่พิโคเน็ตประกอบสมาชิกที่เป็น Client 7 ตัว นั่นคือ หลังจาก Client หรือนักเรียนคนแรกได้ยินเสียง Client หรือนักเรียนคนสุดท้ายจะได้ยินเสียงในอีก 7.5 ms ถัดมา ซึ่งนักเรียนหรือผู้ฟังจะไม่สามารถรู้สึกถึงความต่างของเวลาดังกล่าวได้ หากเวลาดังกล่าวมีค่าน้อยกว่า 20 ms [22]

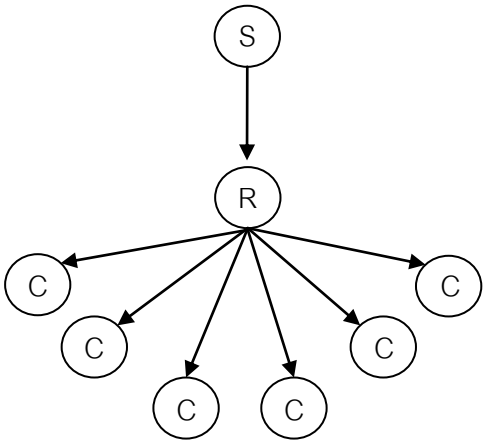
4.1.2 การเชื่อมต่อแบบ Scatternet

สำหรับการเชื่อมต่อแบบ Scatternet เป็นการสร้างการเชื่อมต่อโดยอุปกรณ์ตัวหนึ่งทำหน้าที่เป็น Master ของพิโคเน็ตหนึ่ง และเป็น Slave ของอีกพิโคเน็ตหนึ่งในเวลาเดียวกัน ซึ่งรายละเอียดของเทคโนโลยีบลูทูธ [3] ไม่ได้กำหนดคำหรือสถานะไว้เป็นพิเศษสำหรับอุปกรณ์ดังกล่าวนอกจากสถานะ Master หรือ Slave ในแต่ละพิโคเน็ต ผู้วิจัยจึงกำหนดให้ Relay (R) หมายถึงอุปกรณ์ดังกล่าวเพื่อให้ง่ายต่อการทำความเข้าใจตลอดการทดลอง

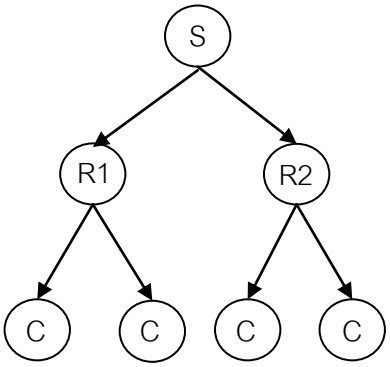
เพื่อจำลองสถานการณ์ในห้องเรียนที่ต้องการอุปกรณ์ที่ทำหน้าที่เป็น Server ที่ตัวครูหนึ่งตัวกับอุปกรณ์ที่ทำหน้าที่เป็น Client ที่ตัวนักเรียนมากกว่า 7 คน การทดลองจึงแบ่งเป็น 2 ส่วน ดังนี้

1. <u>จำนวน Clients ต่อ Server หนึ่งตัว</u>

การทดลองนี้เป็นการทดลองในลักษณะเดียวกับการเชื่อมต่อแบบพิโคเน็ต นั่นคือให้ Server สร้างการเชื่อมต่อกับ Client ทีละตัว เพื่อหาจำนวน Client สูงสุดที่ Scatternet จะสามารถรองรับได้ โดยเริ่มการเชื่อมต่อระหว่าง Master และ Relay ก่อน จากนั้นจึงเป็นการเชื่อมต่อระหว่าง Relay และ Client



รูปที่ 50 จำนวนสมาชิกสำหรับการเชื่อมต่อแบบ Scatternet



รูปที่ 51 Scatternet **ที่ประกอบด้วย 3 พิโคเน็ต**

รูปที่ 50 และรูปที่ 51 แสดงให้เห็นว่า ระบบต้นแบบสามารถทำการเชื่อมต่อแบบ Scatternet ได้ซึ่งสอดคล้องกับรายละเอียดของเทคโนโลยีบลูทูธ [3] เนื่องจากระบบต้นแบบใช้ Bluetooth Dongle version 4.0 โดยรูปที่ 50 แสดงการเชื่อมต่อแบบ Scatternet ที่ประกอบด้วยพิโคเน็ต 2 พิโคเน็ตและ Server สามารถรองรับ Client ได้สูงสุด 6 ตัว ในขณะที่รูปที่ 51 แสดงให้เห็นถึงการรูปแบบการเชื่อมต่อแบบ Scatternet ที่ประกอบด้วย 3 พิโคเน็ตที่มีสมาชิกเป็น Relay 2 ตัว และมี Client 4 ตัวเป็นสมาชิกของพิโคเน็ตของ Relay ทั้งสองตัว

จากผลการทดลองพบว่า การเชื่อมต่อแบบ Scatternet ที่ประกอบด้วยพิโคเน็ต 2 พิโคเน็ตประกอบด้วยสมาชิกเป็น Slave 1 ตัว Relay 1 ตัว และ Client ซึ่งสามารถทำการเชื่อมต่อได้สูงสุด 6 ตัวซึ่งไม่สอดคล้องกับรายละเอียดของเทคโนโลยีบลูทูธที่อธิบายไว้ว่าพิโคเน็ตสามารถมีสมาชิกได้สูงสุด 7 ตัว [3] แต่เมื่อรวมกับผลการทดลองการเชื่อมต่อพิโคเน็ตในการทดลอง 4.1.1 ซึ่งพิโคเน็ตสามารถรองรับจำนวน Client ได้สูงสุด 7 ตัวทำให้สามารถคำนวณได้ว่า สำหรับการเชื่อมต่อแบบ Scatternet Server 1 ตัวสามารถรองรับจำนวน Client ได้สูงสุด 42 ตัว นั่นคือ พิโคเน็ตของ Server ที่ตัวครูมีสมาชิกเป็น Relay ได้สูงสุด 7 ตัว และ Relay 7 ตัวดังกล่าวสามารถ

สร้างพิโคเน็ตของตนเอง โดยมีสมาชิกได้สูงสุด 6 ตัว ซึ่งเป็นจำนวนที่เพียงพอสำหรับความต้องการของระบบต้นแบบ

2. <u>อัตราเร็วในการส่งข้อมูล (Data Rate) และระยะเวลาที่ใช้ในการส่งข้อมูล (Latency)</u>

การทดลองนี้เป็นการทดลองลักษณะเดียวกับการเชื่อมต่อแบบพิโคเน็ต นั่นคือ ทดลองหาขนาดข้อมูลที่ที่ส่งผลให้อัตราเร็วในการส่งข้อมูล (Data Rate) มีค่าประมาณ 64 kbps และใช้เวลาในการส่งข้อมูล (Latency) ไม่เกิน 100 ms โดยการทดลองเป็นการสร้าง Scatternet ซึ่งประกอบด้วย Server 1 ตัว Relay 1 ตัว และ Client สูงสุด 6 ตัวดังรูปที่ 52 แล้วจึงวัดอัตราเร็วในการส่งข้อมูลและระยะเวลาที่ใช้ในการส่งข้อมูลตามวิธีการซึ่งได้อธิบายไว้ในหัวข้อ 3.2

จำนวน Clients	**ผลการทดลอง**	
	Packet Size (Byte)	Data Rate (kbps)
1	320	64.02
6	320	64.11

ตารางที่ 12 คุณภาพของระบบต้นแบบเมื่อทำการเชื่อมต่อแบบ Scatternet

ตารางที่ 12 แสดงให้เห็นว่า สำหรับรับบต้นแบบที่มีการเชื่อมต่อแบบ Scatternet ซึ่ง Server 1 ตัวรองรับจำนวน Client 6 ตัว ข้อมูลขนาด 320 ไบต์ขนาดเหมาะสมที่สุดเพราะให้อัตราเร็วในการส่งข้อมูลเท่ากับ 64.11 kbps

เนื่องจากเป้าหมายของการเชื่อมต่อแบบ Scatternet คือ การเพิ่มจำนวน Client ในการเชื่อมต่อกับ Server หนึ่งตัว และจากผลการทดลอง 4.1.1 เมื่อทำการเชื่อมต่อแบบพิโคเน็ต ข้อมูลขนาด 320 ไบต์เป็นขนาดข้อมูลที่เหมาะสมที่สุดของระบบในกรณีดังกล่าว ในการทดลองนี้จึงเริ่มทำการทดลองโดยกำหนดขนาดข้อมูลเท่ากับ 320 ไบต์ นอกจากนี้การวัดความเร็วในการส่งข้อมูลที่จำนวน Client 1 และ 6 ตัว ก็เพียงพอที่จะแสดงถึงคุณภาพของระบบต้นแบบ เพราะหากระบบต้นแบบสามารถรองรับจำนวน Client 6 ตัวได้ จำนวน Client ที่ลดลงก็ไม่เป็นปัญหาต่อการทำงานของระบบแต่อย่างใด

ในการทดลองนี้ การเชื่อมต่อแบบ Scatternet ประกอบด้วย 2 พิโคเน็ตนั่นคือ Server-Relay พิโคเน็ตหนึ่งและ Relay-Client อีกพิโคเน็ตหนึ่ง และ Relay ทำหน้าที่เป็นเพียงตัวกลางในการส่งต่อข้อมูลเสียงที่ได้รับมาจาก Server ไปยัง Client เท่านั้น ซึ่งจากผลการทดลองที่ 4.1.1 เมื่อทำการส่งข้อมูล 320 ไบต์ เวลาดังกล่าวมีค่าประมาณ 40 ms ประกอบกับระยะเวลาในการส่งข้อมูลในการเชื่อมต่อแบบพิโคเน็ตใช้เวลา 114 ms ดังนั้นในการเชื่อมต่อแบบ Scatternet จึงควรใช้ระยะเวลาในการส่งข้อมูลประมาณ 160 ms

เมื่อเปรียบเทียบผลการทดลองระหว่างการเชื่อมต่อแบบพิโคเน็ตและการเชื่อมต่อแบบ Scatternet แล้ว พบว่า หากต้องการให้ระบบต้นซึ่งประกอบด้วย Server หนึ่งตัวรองรับจำนวน Client ทั้งหมด 6 ตัว การเชื่อมต่อแบบพิโคเน็ตเป็นรูปแบบที่เหมาะสมมากกว่า เพราะใช้ระยะเวลาในการส่งข้อมูลน้อยกว่า นั่นคือ การเชื่อมต่อแบบพิโคเน็ตใช้เวลาในการส่งข้อมูลประมาณ 120 ms ในขณะที่การเชื่อมต่อแบบ Scatternet ใช้เวลาในการส่งข้อมูลประมาณ 160 ms

4.2 การสื่อสารที่ระยะ 10 เมตร

การทดลองนี้เป็นการทดลองการทำงานของระบบที่ระยะ 10 เมตร เพื่อจำลองสถานการณ์จริงในห้องเรียนของนักเรียนที่บกพร่องทางการได้ยิน โดยกำหนดให้มีรูปแบบการเชื่อมต่อเป็นพิโคเน็ตที่ประกอบด้วยสมาชิกที่ทำหน้าที่เป็น Server หนึ่งตัว และสมาชิกที่ทำหน้าที่เป็น Client อีกหนึ่งตัว ที่ระยะ 1 เมตร, 5 เมตร และ 10 เมตร ตามลำดับ และกำหนดขนาดข้อมูลเท่ากับ 320 ไบต์ตามผลการทดลองที่ได้จากการทดลองที่ 4.1 สำหรับรูปแบบการเชื่อมต่อแบบพิโคเน็ต

ระยะทาง (เมตร)	Data Rate (kbps)
1	64.18
5	64.19
10	64.25
12 (ระยะสูงสุด)	64.00

ตารางที่ 13 ผลการทดลองที่ระยะต่างๆ

จากข้อกำหนดเบื้องต้นของระบบในหัวข้อ 3.1.1 เพื่อรองรับการทำงานระยะ 10 เมตรตามสภาพแวดล้อมของห้องเรียนจริง ระบบต้นแบบจึงเลือกใช้ Bluetooth Dongle class 2 ซึ่งมีระยะการทำงาน 10 เมตร [3] ซึ่งสอดคล้องกับความต้องการของระบบ โดยได้ทำการทดลองการสร้างพิโคเน็ตและส่งข้อมูลเสียงระหว่าง Server 1 ตัวและ Client 1 ตัว ที่ระยะ 1 เมตร, 5 เมตร และ 10 เมตรตามลำดับ ซึ่งผลการทดลองตามตารางที่ 13 แสดงให้เห็นว่าที่ระยะการทำงานดังกล่าวระบบสามารถสร้างพิโคเน็ตและส่งข้อมูลได้โดยให้อัตราเร็วประมาณ 64 kbps

นอกจากนี้เพื่อค้นหาระยะสูงสุดที่ระบบสามารถทำงานได้ จึงทำการทดลองโดยการเพิ่มระยะห่างระหว่าง Server และ Client ออกไปครั้งละ 1 เมตรแล้วทดลองสร้างพิโคเน็ตและส่งข้อมูล พบว่าระบบยังคงสามารถสร้างพิโคเน็ตและส่งข้อมูลได้โดยให้อัตราเร็วประมาณ 64 kbps ที่ระยะสูงสุด 12 เมตร ระยะที่เกินกว่านั้น Client ไม่สามารถค้นหา Server ได้เนื่องจากอยู่นอกระยะการทำงาน ทำให้ระบบไม่สามารถสร้างพิโคเน็ตเพื่อทำการส่งข้อมูลได้ จึงสรุปได้ว่า Bluetooth Dongle class 2 เป็นอุปกรณ์ที่เหมาะสมกับระยะการทำงานของระบบ และรองรับระยะการ

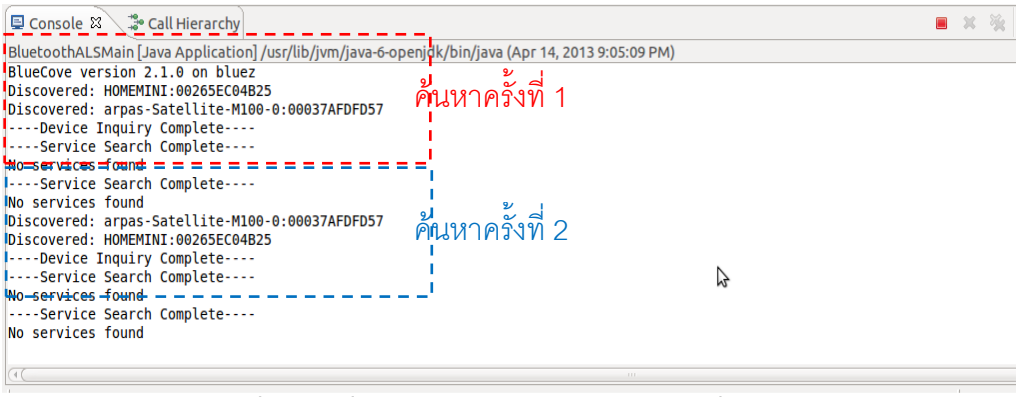
ทำงานที่สอดคล้องกับรายละเอียดของเทคโนโลยีบลูทูธคือ ระยะ 10 เมตร [3] และหากต้องการให้ระบบรองรับระยะการทำงานมากขึ้นก็สามารถเลือกใช้ Bluetooth Dongle class 1 ซึ่งรองรับระยะการทำงาน 100 เมตรแทน [3] ที่มีบริการ พบว่า Bluetooth Dongle class 2 ซึ่งมีระยะการทำงาน 10 เมตร

4.3 การเชื่อมต่ออัตโนมัติระหว่างครูและนักเรียน

การทดลองนี้เป็นการทดลองการเชื่อมต่อของระบบระหว่างครูกับนักเรียน เพื่อจำลองสถานการณ์จริงในห้องเรียนของนักเรียนที่บกพร่องทางการได้ยิน โดยแบ่งการทดลองเป็น 2 ส่วน ดังนี้

4.3.1 กรณี Client ไม่พบบริการที่ต้องการ

เป็นการทดลองเพื่อจำลองสถานการณ์จริงในห้องเรียนที่ Client ไม่พบบริการที่ต้องการจากอุปกรณ์ที่อยู่ในระยะใกล้เคียงด้วยการปิดการทำงานของ Server หรืออุปกรณ์ที่มีบริการที่ Client ต้องการ นั่นคือให้ Client เริ่มทำงานเพียงตัวเดียว

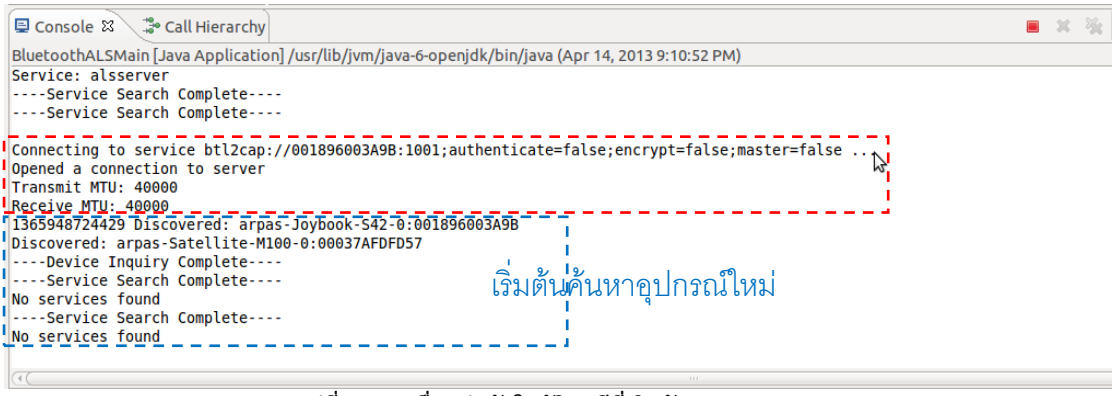


รูปที่ 52 การเชื่อมต่ออัตโนมัติกรณี Client ไม่พบบริการที่ต้องการ

รูปที่ 52 แสดงให้เห็นการทำงานของระบบในส่วน Client ในกรณีเริ่มต้นการใช้งานหรือสร้างพิโคเน็ตครั้งแรกในกรณีที่ Client ไม่พบบริการที่ต้องการจากอุปกรณ์ใกล้เคียง ซึ่งจะเห็นได้ว่า Client ทำงานตามขั้นตอนที่ได้ออกแบบและอธิบายแผนภาพกิจกรรมหัวข้อ 3.1.2 โดย Client ซึ่งเริ่มต้นการทำงานโดยการค้นหาอุปกรณ์ที่อยู่ในระยะใกล้เคียง จากนั้นจึงค้นหาบริการที่ต้องการอุปกรณ์ที่พบเหล่านั้น เมื่อไม่พบบริการที่ต้องการจึงเริ่มต้นค้นหาอุปกรณ์ใกล้เคียงใหม่อีก

4.3.2 กรณีขาดการติดต่อระหว่างครูและนักเรียน

การทดลองนี้เป็นการจำลองสถานการณ์จริงในห้องเรียนที่อาจเกิดการขาดการเชื่อมต่อระหว่างครูและนักเรียนในระหว่างการเรียนการสอน ซึ่งทำการทดลองด้วยการสร้างพิโคเน็ตระหว่าง Server 1 ตัวและ Client 1 ตัว เมื่อการสร้างพิโคเน็ตเสร็จสมบูรณ์จึงเริ่มทำการส่งข้อมูลเสียงจาก Server ไปยัง Client จากนั้นปิดการทำงานของ Server เพื่อจำลองสถานการณ์ดังกล่าว



รูปที่ 53 การเชื่อมต่ออัตโนมัติกรณีที่เกิดสัญญาณขาดหาย

รูปที่ 53 แสดงให้ของการทำงานของระบบเมื่อเกิดการขาดการติดต่อระหว่าง Server ที่ตัวครูและ Client ที่ตัวนักเรียน ไม่ว่าจะเนื่องจากระยะห่างระหว่าง Server และ Client หรือสาเหตุใดก็ตาม เมื่อการขาดการติดต่อเกิดขึ้น Client ซึ่งถูกออกแบบให้คอยรับสัญญาณ Lost Connection จะได้รับสัญญาณดังกล่าวจากการขาดการติดต่อนั้น และเริ่มต้นสร้างการเชื่อมต่อกับ Server ใหม่ทันที โดยเริ่มต้นกระบวนการจากการค้นอุปกรณ์ในระยะใกล้เคียง เมื่อพบอุปกรณ์ซึ่งมีบริการที่ต้องการจึงสร้างพิโคเน็ตและเริ่มต้นการทำงานปกติ พร้อมทั้งคอยสัญญาณ Lost Connection ในกรณีเกิดการขาดการติดต่ออีกครั้ง ซึ่งในกรณีนี้หากมี Server ที่มีบริการที่ Client ตัวดังกล่าวต้องการมากกว่า 1 ตัวอยู่ในบริเวณใกล้เคียง Client จะทำการสร้างการพิโคเน็ตกับ Server ตัวแรกที่พบ ซึ่งจากการทดลองพบว่า Server ตัวดังกล่าวจะอยู่ในระยะใกล้กว่าเสมอ

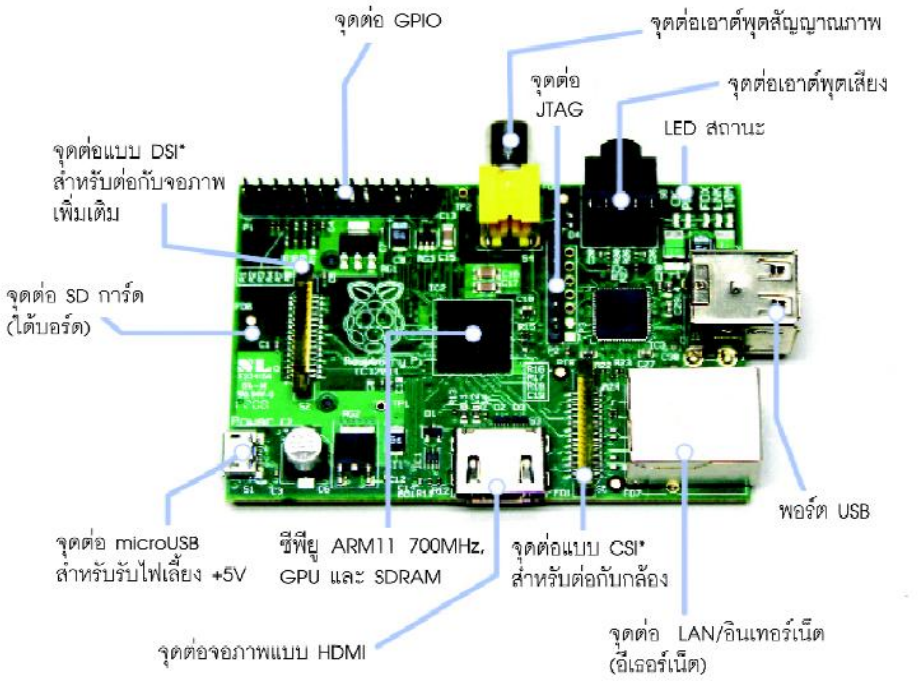
4.4 การทดลองบนอุปกรณ์ฝังตัว

การทดลองนี้เป็นการทดลองการทำงานของระบบต้นแบบ เพื่อแสดงถึงความเป็นไปได้ในการนำระบบไปพัฒนาและใช้งานจริงบนอุปกรณ์ประเภทฝังตัว (Embedded) ซึ่งมีทรัพยากรในการทำงานจำกัด ทั้งความเร็วในการประมวลผลและขนาดของหน่วยความจำ

4.4.1 Raspberry Pi

งานวิจัยนี้เลือกใช้ Raspberry Pi ซึ่งเป็นชุด Development Kit ที่ถูกออกแบบและพัฒนาโดยกลุ่มนักวิจัยในห้องวิจัยคอมพิวเตอร์ของมหาวิทยาลัยแห่งเคมบริดจ์ (University of

Cambridge Computer's Laboratory) ที่ถูกออกแบบมาเพื่อใช้สำหรับเป็นอุปกรณ์เริ่มต้นในการเรียนรู้ทักษะทางคอมพิวเตอร์และการเขียนโปรแกรมสำหรับเด็ก [24] ซึ่งมีส่วนประกอบดังรูปที่ 54 และมีคุณสมบัติทางเทคนิคตามตารางที่ 14



รูปที่ 54 ส่วนประกอบของ Raspberry Pi [25]

Chip	Broadcom BCM2835
CPU	ARM11 Core ARM1176JZF-S 700 MHz
GPU	Broadcom VideoCore IV
Memory	SDRAM 512 MB
Interface	USB 2.0 (2 ports)
	HDMI output
	Video output
	Jack 3.5 mm.
	GPIO
	Ethernet
Power	5.0 V 700 mA
Size	85.60 x 53.98 mm. (3.370 x 2.125 inches)

ตารางที่ 14 คุณสมบัติทางเทคนิคของ Raspberry Pi [25]

Raspberry Pi เป็นอุปกรณ์ฝังตัวที่มีลักษณะเป็นคอมพิวเตอร์ขนาดเล็ก (Computer on module) นั่นคือ มีทรัพยากรในการทำงานจำกัดทั้งความเร็วและขนาดของหน่วยความจำ เช่นเดียวกับอุปกรณ์ฝังตัวทั่วไป แต่ก็มีลักษณะการใช้งานแบบเครื่องคอมพิวเตอร์ตั้งโต๊ะ นั่นคือมีระบบปฏิบัติการ, User Interface และใช้ SD card แทนการทำงานของฮาร์ดดิสก์ และด้วยคุณลักษณะต่างๆเหล่านี้ของ Raspberry Pi ประกอบกับการทดลองก่อนหน้านี้เป็นการพัฒนาและทำการทดลองบนเครื่องคอมพิวเตอร์ตั้งโต๊ะ การทำการทดลองบน Raspberry Pi จึงลดเวลาในการพัฒนาระบบบนอุปกรณ์ฝังตัวซึ่งสามารถนำโค้ดของระบบมาทำการทดลองได้ทันที

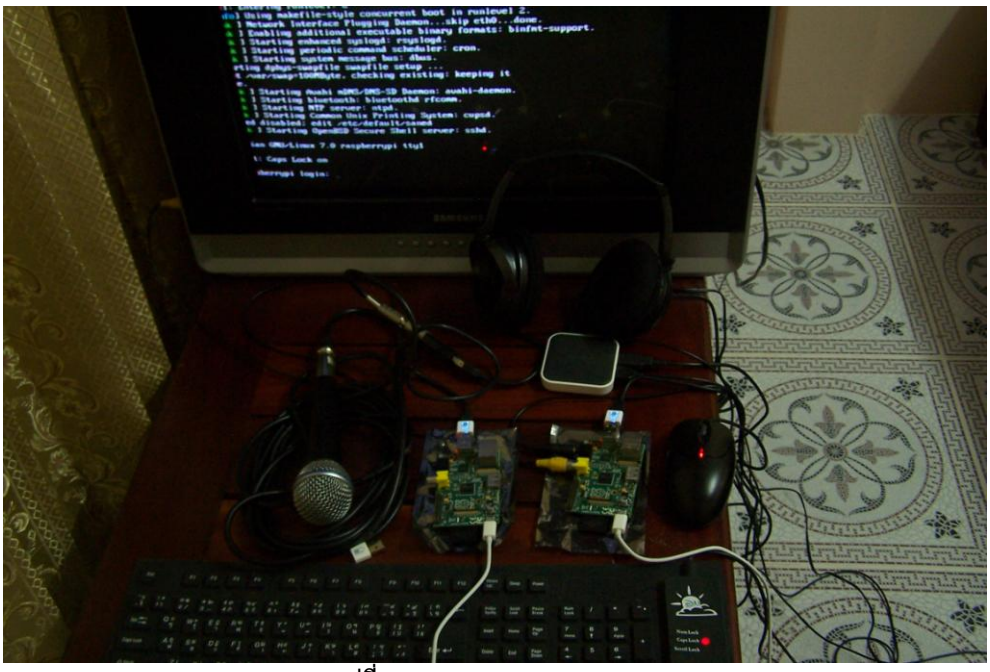
4.4.2 ส่วนประกอบพื้นฐานของระบบ

ส่วนประกอบพื้นฐานของระบบประกอบด้วยอุปกรณ์ ดังนี้

1. Raspberry Pi
2. SD card ซึ่งลงซอฟท์แวร์ต่างๆดังนี้
    a. Soft-Float Debian Wheezy เป็นระบบปฏิบัติการ
    b. Java JDK for Embedded
    c. ไลบรารี่ Bluecove สำหรับ ARM โปรเซสเซอร์ ซึ่งสามารถดาวน์โหลดได้จาก [26]
    d. โค้ดของระบบต้นแบบที่ได้พัฒนาขึ้น
3. ไมโครโฟน
4. หูฟังหรือลำโพง
5. Bluetooth Dongle Version 4.0 class 2 ตัว
6. USB Hub สำหรับเมาส์และคีย์บอร์ด

4.4.3 การทดสอบการทำงานของระบบบน Raspberry Pi

การทดลองนี้เป็นการทดสอบการทำงานของระบบบน Raspberry Pi เพื่อทดลองว่า Raspberry Pi สามารถรองรับการทำงานของระบบได้หรือไม่ ซึ่งระบบจะประกอบด้วย Raspberry Pi 2 ชุด ทำหน้าที่เป็น Server และ Client โดย Server ต่อกับไมโครโฟนเพื่อรับเสียงผู้พูดและ Client ต่อกับหูฟังเพื่อรอรับเสียงพูดจาก Server โดยจะทำการสร้างพิโคเน็ตที่ประกอบด้วย Server 1 ตัวและ Client 1 ตัว ซึ่งทั้งServer และ Client จะทำงานตามขั้นตอนที่ได้ออกแบบไว้ในหัวข้อ 3.1.2



รูปที่ 55 **การทดลองบน** Raspberry Pi

รูปที่ 55 แสดงการทดลองระบบต้นแบบบน Raspberry Pi ซึ่งกำหนดค่าพารามิเตอร์ต่างๆ เช่นเดียวกับที่ทำการทดลองบนเครื่องคอมพิวเตอร์ตั้งโต๊ะ นั่นคือ

พารามิเตอร์สำหรับ Server

โปรโตคอลที่ใช้ในการส่งข้อมูล กำหนดเป็น L2CAP

ขนาดข้อมูลสูงสุดที่ใช้ในการรับส่งข้อมูล กำหนดให้เท่ากับ 40000 ไบต์

สถานะของอุปกรณ์ในพิโคเน็ต กำหนดให้สถานะเป็น Master

Authentication กำหนดให้ไม่มีการป้อน Pairing Pin สำหรับการเชื่อมต่อ

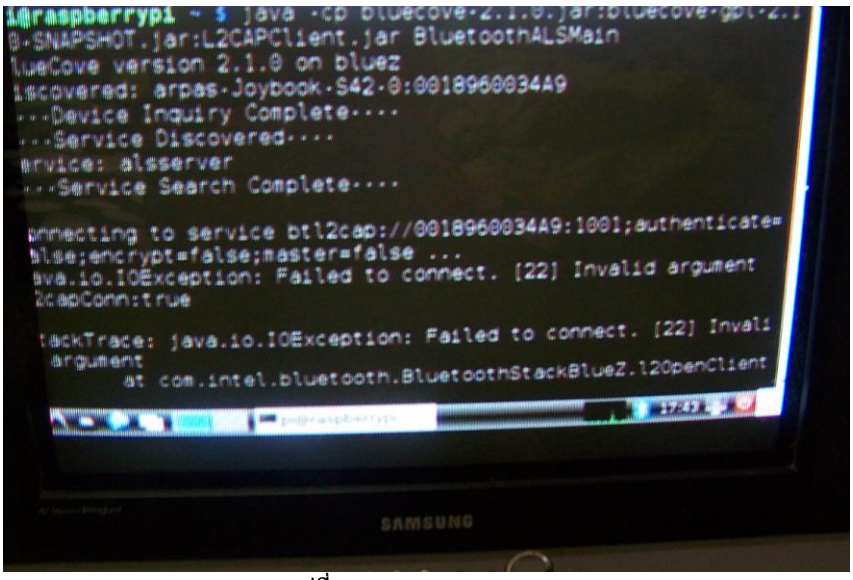
พารามิเตอร์สำหรับ Client

โปรโตคอลที่ใช้ในการส่งข้อมูล กำหนดเป็น L2CAP

ขนาดข้อมูลสูงสุดที่ใช้ในการรับส่งข้อมูล กำหนดให้เท่ากับ 40000 ไบต์

สถานะของอุปกรณ์ในพิโคเน็ต กำหนดให้สถานะเป็น Slave

Authentication กำหนดให้ไม่มีการป้อน Pairing Pin สำหรับการเชื่อมต่อ



รูปที่ 56 **ผลการทดลองบน** Raspberry Pi

รูปที่ 56 แสดงผลการทดลองบน Raspberry Pi ซึ่ง Client ไม่สามารถสร้างการเชื่อมต่อกับ Server ได้หรือไม่สามารถสร้างพิโคเน็ตได้นั่นเอง โดย Exception หรือข้อผิดพลาดที่เกิดขึ้นเป็นขั้นตอนหลังจาก Client ค้นพบ Server ซึ่งเป็นอุปกรณ์ใกล้เคียงและค้นพบบริการที่ต้องการจาก Server ลำดับถัดมาเป็นการสร้างการเชื่อมต่อระหว่าง Server และ Client ด้วยค่าพารามิเตอร์ต่างๆตามที่ได้กำหนดไว้แล้วในเบื้องต้น

Exception หรือข้อผิดพลาดที่เกิดขึ้น คือ “Failed to connect. [22] Invalid argument” ซึ่งหมายถึงระบบไม่สามารถสร้างการเชื่อมต่อระหว่าง Server และ Client ได้เนื่องจากการกำหนดค่าพารามิเตอร์เพื่อใช้ในการเชื่อมต่อนั้นไม่ถูกต้อง ทั้งที่การทดลองนี้กำหนดค่าพารามิเตอร์เหล่านี้เป็นค่าเดียวกับที่ระบบสามารถทำงานได้บนเครื่องคอมพิวเตอร์ตั้งโต๊ะในการทดลองก่อนหน้านี้ โดยการทำงานของอุปกรณ์จะตกลงค่าพารามิเตอร์ดังกล่าวด้วยรูปแบบข้อความดังนี้

“btl2cap://0018960034A9:1001;authenticat=false;encrypt=false;master=false”

btl2cap หมายถึง โปรโตคอล L2CAP

0018960034A9 หมายถึง แอดเดรสของอุปกรณ์ดังกล่าว

1001 หมายถึง รหัสสื่อถึงโปรโตคอล L2CAP

authenticate = false หมายถึง ไม่ใช้ Pairing Pin

encrypt = false หมายถึง ไม่มีการเข้ารหัสข้อความ

master = false หมายถึง กำหนดให้ทำหน้าที่เป็น Slave สำหรับ Client

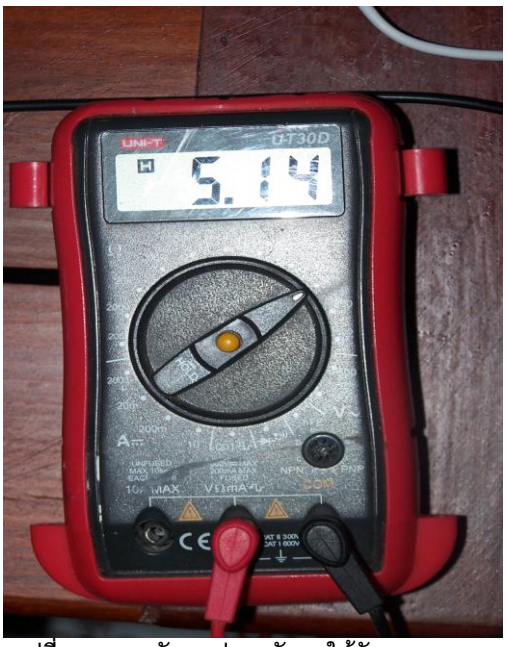
master = true หมายถึง กำหนดให้ทำหน้าที่เป็น Master สำหรับ Server

พารามิเตอร์เหล่านี้เป็นพารามิเตอร์ซึ่งถูกกำหนดด้วยสถานะของอุปกรณ์ในพิโคเน็ตและลักษณะการเชื่อมต่อที่ต้องการใช้งานอยู่แล้ว นั่นคือ หากต้องการใช้โปรโตคอล L2CAP ต้องกำหนดพารามิเตอร์เป็น “btl2cap” และ “1001” เท่านั้น การเปลี่ยนเป็นค่าอื่นไม่สามารถสร้างการเชื่อมต่อระหว่างอุปกรณ์ได้ หรือหากต้องการให้มีสถานะเป็น Master ก็ต้องกำหนดเป็น master = true นอกจากพารามิเตอร์เหล่านี้ยังมีขนาดข้อมูลสูงสุดเป็นอีกพารามิเตอร์หนึ่งที่จำเป็นในการสร้างการเชื่อมต่อระหว่างอุปกรณ์และใช้งานโปรโตคอล L2CAP ซึ่งหมายถึงขนาดของข้อมูลชนิด L2CAP ที่ใช้ในการรับส่งระหว่างอุปกรณ์มีค่าไม่เกินค่าที่กำหนด โดยสามารถกำหนดค่าได้ตั้งแต่ 40 ถึง 65535 ไบต์ ซึ่งเป็นพารามิเตอร์เดียวที่สามารถเปลี่ยนแปลงค่าได้มากมาย แต่จากการทดลองพบว่า การเปลี่ยนแปลงค่าพารามิเตอร์เหล่านี้ไม่ได้ทำให้ระบบสามารถทำงานได้บน Raspberry Pi และยังคงคืนค่า Exception กลับมาเช่นเดิม

อาจเนื่องจากการทดลองก่อนหน้านี้เป็นการทดลองการทำงานของระบบบนเครื่องคอมพิวเตอร์ตั้งโต๊ะที่มีทรัพยากรในการประมวลทั้งความเร็วหรือหน่วยความจำที่สูงกว่า Raspberry Pi มาก ซึ่งอาจเป็นสาเหตุให้ระบบต้นแบบไม่สามารถทำงานได้บน Raspberry Pi จนเป็นเหตุให้แสดงค่า Exception ดังกล่าวกลับมา จึงทำการทดลองเพิ่มเติมดังนี้

1. <u>การจ่ายพลังงานให้กับ Raspberry Pi</u>

รูปที่ 57 การวัดการจ่ายพลังงานให้กับ Raspberry Pi

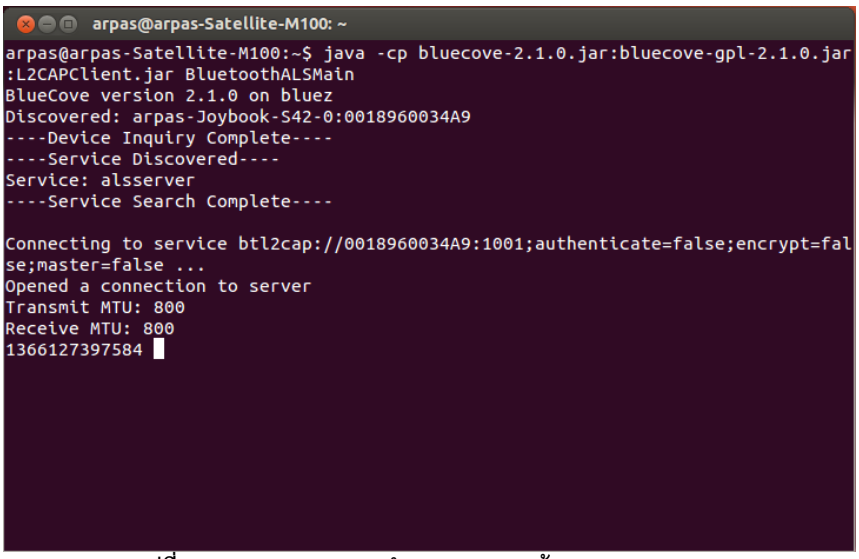


รูปที่ 58 ผลการวัดการจ่ายพลังงานให้กับ Raspberry Pi

รูปที่ 57 และรูปที่ 58 แสดงการวัดและปริมาณแรงดันไฟฟ้าที่ถูกจ่ายให้กับ Raspberry Pi เท่ากับ 5.14 โวลต์ ซึ่งสอดคล้องกับคุณสมบัติทางเทคนิคของ Raspberry Pi ซึ่งต้องการแรงดัน5 โวลต์ [25]

2. <u>OS Kernel</u>

ความแตกต่างระหว่างระบบปฏิบัติการบน Raspberry Pi และระบบปฏิบัติการบนเครื่องคอมพิวเตอร์ตั้งโต๊ะอาจเป็นสาเหตุให้ระบบไม่สามารถทำงานได้บน Raspberry Pi เนื่องจาก Soft-Float Debian Wheezy ซึ่งเป็นระบบปฏิบัติการบน Raspberry Pi เป็นระบบปฏิบัติการลินุกซ์ที่ถูกพัฒนาบน kernel เวอร์ชัน 3.2 ในขณะที่ Ubuntu 10.10 ที่ใช้งานบนเครื่องคอมพิวเตอร์ตั้งโต๊ะนั้นถูกพัฒนาบน kernel เวอร์ชัน 2.6



รูปที่ 59 **ผลการทดสอบการทำงานของระบบต้นแบบบน** Ubuntu 12.04 LTS

รูปที่ 59 แสดงผลการทดลองของระบบบน Ubuntu 12.04 LTS ซึ่งเป็น kernel เวอร์ชัน 3.2 เช่นเดียวกับ Soft-Float Debian Wheezy บน Raspberry Pi ซึ่งผลการทดลองก็แสดงให้เห็นว่าระบบสามารถทำงานได้ทั้งบนระบบปฏิบัติการลีนุกซ์ที่ใช้ kernel ทั้งสองเวอร์ชัน

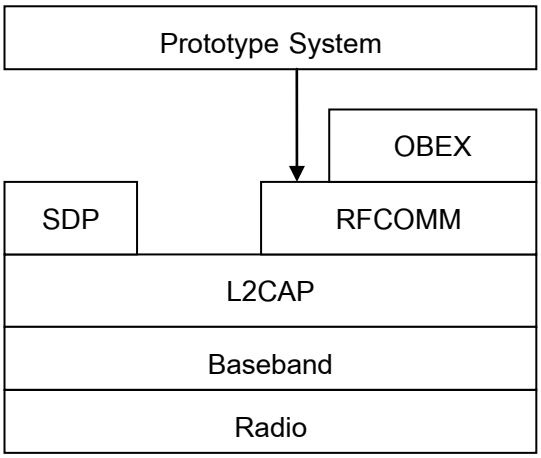
3. ไลบรารี่ Bluecove

เนื่องจาก Raspberry Pi มีสถาปัตยกรรมเป็น ARM โปรเซสเซอร์ ซึ่งเป็นสถาปัตยกรรมต่างชนิดกับคอมพิวเตอร์ตั้งโต๊ะ ดังนั้นจึงต้องใช้ไลบรารี่ Bluecove สำหรับ ARM โปรเซสเซอร์ ในการทดลองบน Raspberry Pi ครั้งแรกนั้น ระบบเรียกใช้ไลบรารี่ Bluecove สำหรับ ARM โปรเซสเซอร์ที่สามารถดาวน์โหลดได้จาก [26] ซึ่งอาจจะเป็นไลบรารี่ที่ไม่เหมาะกับสถาปัตยกรรมบน Raspberry Pi จนเป็นสาเหตุให้ระบบไม่สามารถทำงานบน Raspberry Pi

การทดลองส่วนนี้จึงเป็นการคอมไพล์ไลบรารี่ Bluecove บน Raspberry Pi เองเพื่อพิสูจน์ว่าสาเหตุของปัญหาอยู่ที่สถาปัตยกรรมของ Raspberry Pi หรือไม่ ผลการทดลองพบว่า ด้วยการใช้งานไลบรารี่ Bluecove ซึ่งทำการคอมไพล์บน Raspberry Pi ระบบก็ไม่สามารถทำงานได้ โดยให้ผลเช่นเดียวกับไลบรารี่เดิม

4. การทดลองใช้โปรโตคอล RFCOMM

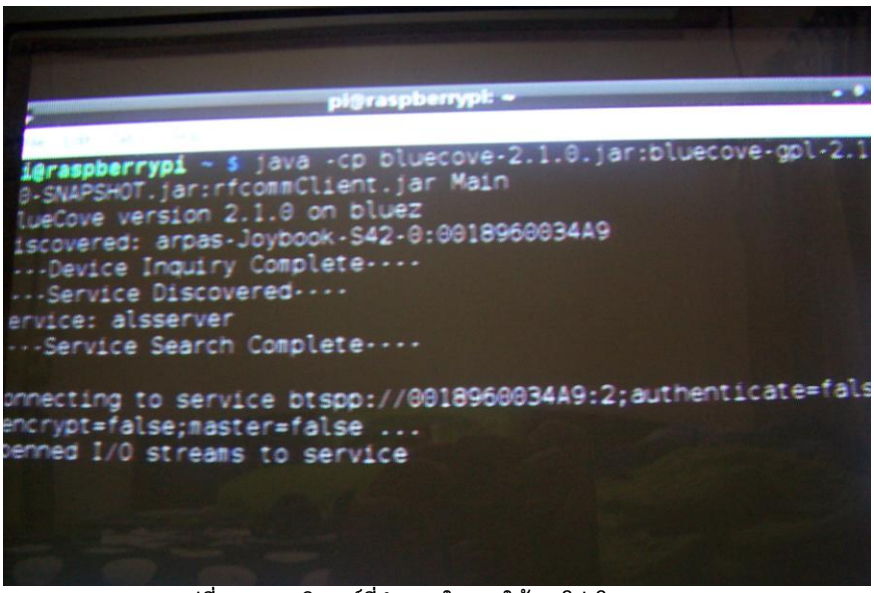
การทดลองนี้เป็นการทดลองเพื่อทดสอบว่า Raspberry Pi สามารถรองรับการทำงานของระบบต้นแบบได้หรือไม่ โดยเลือกใช้โปรโตคอล RFCOMM แทน L2CAP เดิม โดยโปรโตคอล RFCOMM เป็นชั้นการทำงานที่อยู่เหนือ L2CAP ซึ่งใช้ Serial Port Profile (SPP) และ L2CAP เพื่อใช้งานช่องสัญญาณ ACL เช่นเดียวกัน แต่มีความเร็วในการส่งข้อมูลต่ำกว่าการใช้ L2CAP โดยตรง นั่นคือมีความเร็วในการส่งข้อมูลเพียง 128 kbps [4]



รูปที่ 60 ระบบต้นแบบที่เมื่อใช้โปรโตคอล RFCOMM

รูปที่ 60 แสดงให้เห็นการทำงานของระบบเมื่อใช้งานโปรโตคอล RFCOMM โดยเป็นการใช้งานโปรโตคอล L2CAP ผ่านโปรโตคอล RFCOMM ซึ่งต่างจากระบบต้นแบบเดิมที่เป็นการใช้งานโปรโตคอล L2CAP โดยตรง

สำหรับการทดลองการใช้งานโปรโตคอล RFCOMM ก็เป็นการทดลองในลักษณะเดียวกับการใช้งานโปรโตคอล L2CAP นั่นคือเป็นการทดลองเพื่อดูว่าระบบสามารถทำการส่งข้อมูลเสียงบน Raspberry Pi ได้หรือไม่ โดยทำการสร้างพิโคเน็ตซึ่งประกอบด้วย Server 1 ตัว และ Client 1 ตัว เมื่อการสร้างพิโคเน็ตเสร็จสิ้น



รูปที่ 61 พารามิเตอร์ที่กำหนดในการใช้งานโปรโตคอล RFCOMM

รูปที่ 61 แสดงพารามิเตอร์ที่กำหนดในการใช้งานโปรโตคอล RFCOMM นั่นคือ “btspp://0018960034A9:2;authenticate=false;encrypt=false;master=false”

btspp หมายถึง การใช้ Serial Port Profile (SPP)

001890034A9 หมายถึง แอดเดรสของอุปกรณ์

2 หมายถึง พอร์ตสำหรับโปรไฟล์ SPP

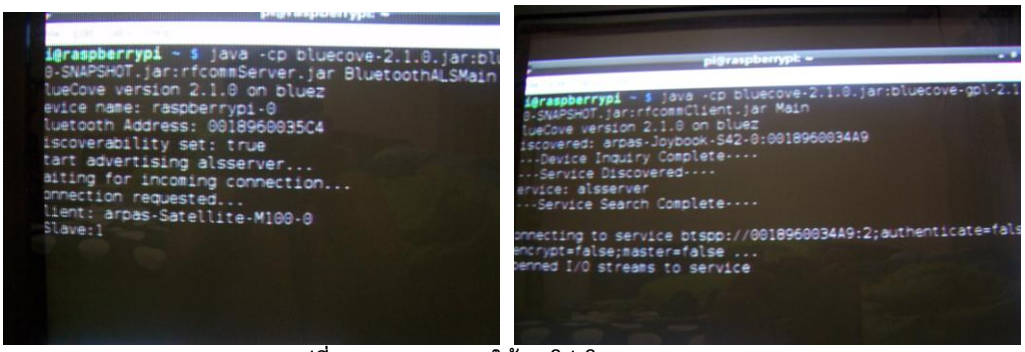
authenticate = false หมายถึง ไม่ใช้งาน Pairing Pin

encrypt = false หมายถึง ไม่มีการเข้ารหัสข้อมูล

master = false หมายถึง กำหนดให้เป็น slave สำหรับ Client

master = true หมายถึง กำหนดให้เป็น slave สำหรับ Server

สำหรับการทำงานของระบบต้นแบบด้วยโปรโตคอล L2CAP และโปรโตคอล RFCOMM นั้น ระบบต้นแบบกำหนดค่าพารามิเตอร์ต่างๆเหล่านี้เช่นเดียวกับการทดลองบนเครื่องคอมพิวเตอร์ตั้งโต๊ะ โดยความแตกต่างในการใช้งานของทั้งสองโปรโตคอลต่างกันเพียงชนิดของออบเจ็กต์ที่ใช้ในการเชื่อมต่อระหว่างอุปกรณ์ นอกจากนี้ การกำหนดขนาดข้อมูล ขนาดบัฟเฟอร์ และการกำหนดค่าพารามิเตอร์ต่างๆสำหรับโปรโตคอล RFCOMM ก็เป็นค่าพารามิเตอร์เดียวกันกับโปรโตคอล L2CAP ต่างกันเพียงโปรโตคอล RFCOMM ไม่มีการกำหนดขนาดข้อมูลสูงสุดในการรับส่งข้อมูลเท่านั้น



รูปที่ 62 ผลการทดลองใช้งานโปรโตคอล RFCOMM

รูปที่ 62 แสดงผลการทดลองใช้งานโปรโตคอล RFCOMM ของระบบต้นแบบบน Raspberry Pi ซึ่งแสดงให้เห็นว่า Raspberry Pi สามารถรองรับการทำงานของระบบต้นแบบได้ โดยเลือกใช้โปรโตคอล RFCOMM โดยรูปแรกคือ Server หรือ Master และรูปที่สองคือ Client หรือ Slave ซึ่งทั้ง Server และ Client สามารถสร้างพิโคเน็ตและส่งข้อมูลเสียงกันได้

จากการทดลองพบว่า ระบบไม่สามารถทำงานได้บน Raspberry Pi ได้เมื่อเลือกใช้โปรโตคอล L2CAP เนื่องจากค่าพารามิเตอร์ที่กำหนดให้สำหรับการรับส่งข้อมูลไม่ถูกต้องทั้งที่ค่าที่กำหนดให้กับพารามิเตอร์ต่างๆก็เป็นค่าเดียวกับการทดลองที่ทำบนเครื่องคอมพิวเตอร์ตั้งโต๊ะ จากนั้นจึงทำการทดลองย่อยเพื่อหาสาเหตุที่อาจทำให้ระบบไม่สามารถทำงานได้บน Raspberry Pi ซึ่งผลการทดลองต่างๆแสดงให้เห็นว่า Raspberry Pi ได้รับพลังงานเพียงพอตามคุณสมบัติทางเทคนิค, ระบบสามารถทำงานได้ทั้งบนระบบปฏิบัติการลีนุกซ์ที่ถูกพัฒนาบน kernel เวอร์ชัน 2.6 และ 3.2 และไลบรารี่ที่ใช้งานก็เป็นไลบรารี่ที่ตรงกับสถาปัตยกรรมของ Raspberry Pi จึงยังไม่

สามารถสรุปได้ว่า ระบบที่ใช้โปรโตคอล L2CAP ไม่สามารถทำงานบน Raspberry Pi ได้ด้วยสาเหตุใด

แต่การทดลองใช้งานโปรโตคอล RFCOMM ก็แสดงให้เห็นว่า ระบบสามารถทำงานได้บน Raspberry Pi เมื่อใช้โปรโตคอล RFCOMM แทน L2CAP ทั้งที่โปรโตคอล RFCOMM ทำงานบนโปรโตคอล L2CAP ซึ่งทำให้สรุปได้เพียงว่าโปรโตคอล RFCOMM เป็นทางเลือกหนึ่งในการพัฒนาระบบต่อไปถึงแม้จะให้ความเร็วในการส่งข้อมูลต่ำกว่า

4.5 การทดลองเวลาที่ใช้ในการสร้างพิโคเน็ต

การทดลองนี้เป็นการทดลองเพื่อวัดเวลาที่ใช้ในการสร้างพิโคเน็ตที่ประกอบด้วย Server 1 ตัวและ Client 7 ตัว โดยทำการทดลองด้วยการบันทึกเวลาที่ Server และ Client แต่ละตัวสามารถทำการเชื่อมต่อกันได้ โดยแบ่งการทดลองเป็น 2 ส่วนดังนี้

การสร้างการเชื่อมต่อระหว่าง Server และ Client ทีละตัว ทำการทดลองด้วยการบันทึกเวลาที่ Client แต่ละตัวใช้ในการสร้างการเชื่อมต่อกับ Server ตั้งแต่เริ่มต้นค้นหาอุปกรณ์จนการเชื่อมต่อเสร็จสิ้น จากการทดลองพบว่า เวลาที่ใช้ในการเชื่อมต่อระหว่าง Server และ Client แต่ละตัวใช้เวลาประมาณ 15 วินาที และใช้เวลาในการสร้างพิโคเน็ตที่มีสมาชิกเป็น Client ทั้งหมด 7 ตัว 105 วินาที

การสร้างการเชื่อมต่อระหว่าง Server และ Client พร้อมกัน 7 ตัว เป็นการให้ Client ทุกตัวเริ่มต้นค้นหา Server พร้อมกัน ซึ่งในกรณีที่ Server สามารถทำการเชื่อมต่อกับ Client ได้สำเร็จ ระบบจะใช้เวลาในการสร้างพิโคเน็ตเร็วที่สุดเพียงประมาณ 40 วินาที แต่จากการทดลองยังพบอีกว่า การที่ Client ทุกตัวทำการค้นหาและเชื่อมต่อไปยัง Server ในเวลาเดียวกัน โดยมากมักจะไม่สามารถสร้างพิโคเน็ตที่มีจำนวนสมาชิกตามที่ต้องการได้สำเร็จ ซึ่งมักจะสร้างพิโคเน็ตได้สูงสุดเพียง 2-3 ตัวเท่านั้น ในขณะที่ Client ตัวที่เหลือสามารถพบ Server ในการค้นหาอุปกรณ์ใกล้เคียงแต่ไม่สามารถสร้างพิโคเน็ตกับ Server ดังกล่าวได้ ดังนั้นจึงควรสร้างพิโคเน็ตโดยทำการเชื่อมต่อ Server กับ Client ทีละตัว ซึ่งถึงแม้จะใช้เวลาในการสร้างการเชื่อมต่อมากกว่า แต่ก็สามารถทำให้พิโคเน็ตสามารถมีจำนวนสมาชิกได้มากกว่า

บทที่ 5
บทสรุป

5.1 สรุปผลการวิจัย

จากการทดลองระบบสื่อสารไร้สายต้นแบบสำหรับนักเรียนที่บกพร่องทางการได้ยินด้วยเทคโนโลยีบลูทูธ สามารถสรุปเป็นผลวิจัย ดังนี้

- ระบบต้นแบบสามารถรองรับจำนวนนักเรียนได้มากกว่า 7 คนด้วยรูปแบบการเชื่อมต่อแบบ Scatternet ซึ่งอุปกรณ์ที่ทำหน้าที่เป็น Server ที่ตัวครู 1 ตัว สามารถสร้างพิโคเน็ตที่มีสมาชิกได้สูงสุด 7 ตัวซึ่งสอดคล้องกับรายละเอียดของเทคโนโลยีบลูทูธ [3] และสมาชิกทั้ง 7 ตัวดังกล่าวสามารถสร้างพิโคเน็ตของตนเองได้โดยมีสมาชิกสูงสุด 6 ตัว ถึงแม้จะไม่สอดคล้องกับรายละเอียดของเทคโนโลยีบลูทูธ [3] แต่ก็ทำให้อุปกรณ์ที่ทำหน้าที่เป็น Server ที่ตัวครูสามารถให้บริการอุปกรณ์ซึ่งทำหน้าที่เป็น Client ในระบบได้ถึง 42 ตัว ซึ่งเพียงพอสำหรับความต้องการของระบบต้นแบบ แต่ถึงแม้การเชื่อมต่อแบบ Scatternet จะทำให้ระบบต้นแบบสามารถรองรับจำนวนอุปกรณ์ได้เพียงพอต่อความต้องการของระบบต้นแบบ แต่การเชื่อมต่อนี้ก็ใช้ระยะเวลาในการส่งข้อมูล (Latency) มากกว่าการเชื่อมต่อแบบพิโคเน็ตเนื่องจากมีอุปกรณ์ซึ่งทำหน้าที่เป็น Relay เป็นตัวกลางในการส่งข้อมูลจาก Server ไปยัง Client ต่างจากพิโคเน็ตที่เป็นการส่งข้อมูลจาก Server ไปยัง Client โดยตรง ดังนั้นในการใช้งานจริงควรเลือกใช้การเชื่อมต่อแบบพิโคเน็ต โดยเพิ่มจำนวนอุปกรณ์ Server ที่ตัวครูให้สอดคล้องกับจำนวนนักเรียน และควรกำหนดให้ระบบต้นแบบใช้รูปแบบการเชื่อมต่อแบบพิโคเน็ตที่มีสมาชิกเป็นนักเรียนหรือ Client จำนวน 6 ตัวเพื่อให้ระยะเวลาในการส่งข้อมูลไม่เกิน 100 ms [23]
- ขนาดข้อมูลและขนาดบัฟเฟอร์เป็นพารามิเตอร์ที่สำคัญของระบบที่มีลักษณะการทำงานแบบเรียลไทม์แอพพลิเคชัน ขนาดข้อมูลและขนาดบัฟเฟอร์ที่เหมาะสมจะใช้ระยะเวลาในการส่งข้อมูลที่น้อยที่สุดที่คุณภาพเสียงไม่เสียไป และส่งผลถึงจำนวน Client ที่ระบบสามารถรองรับได้
- ระบบต้นแบบสามารถให้บริการนักเรียนได้ระยะสูงสุด 12 เมตร โดยการเลือกใช้ Bluetooth Dongle Class 2 ซึ่งรองรับการทำงานที่ระยะ 10 เมตร [3] ที่อัตราเร็วใน

การส่งข้อมูล (Data Rate) ประมาณ 64 kbps ซึ่งแสดงให้เห็นถึงคุณภาพของเสียงที่เป็นคุณภาพเดียวกับระบบโทรศัพท์ [22]

- ระบบต้นแบบสามารถทำการเชื่อมต่ออัตโนมัติระหว่างอุปกรณ์ที่ตัวครูและนักเรียนได้ทั้งในกรณีที่ Client ที่ตัวนักเรียนไม่พบบริการที่ตนเองต้องการ และในกรณีที่เกิดการขาดการติดต่อระหว่างครูและนักเรียนในระหว่างการเรียนการสอน โดยได้ออกแบบให้ Client ค้นหาอุปกรณ์และบริการที่ตนเองต้องการไปจนกระทั่งพบแล้วจึงสร้างการเชื่อมต่อกับ Server และในขณะเดียวกันก็รอคอยสัญญาณเพื่อเริ่มต้นค้นหาอุปกรณ์และบริการใหม่เพื่อสร้างการเชื่อมต่อใหม่ทันทีที่เกิดการขาดการติดต่อระหว่างครูและนักเรียน ซึ่งฟังก์ชันการทำงานลักษณะนี้เป็นส่วนสำคัญซึ่งเติมเต็มคุณภาพของระบบในการนำไปใช้งานจริง เพื่อให้ผู้ใช้สามารถใช้งานได้อย่างสะดวกสบาย ในการพัฒนาระบบจึงควรพิจารณาฟังก์ชันเหล่านี้ตั้งแต่ขั้นตอนการออกแบบการทำงานของระบบเพื่อให้ง่ายต่อการพัฒนาระบบ
- เนื่องจากระบบปฏิบัติการลินุกซ์มีการจัดการเสียงไม่ดีเท่ากับระบบปฏิบัติการวินโดวส์ คุณภาพเสียงที่ได้จากระบบซึ่งทำงานบนระบบปฏิบัติการลินุกซ์จึงไม่ดีเท่ากับระบบลักษณะเดียวกันที่ทำงานบนระบบปฏิบัติการวินโดวส์

5.2 **ข้อเสนอแนะ**

จากการพัฒนาระบบสื่อสารไร้สายต้นแบบสำหรับนักเรียนที่บกพร่องทางการได้ยินด้วยเทคโนโลยีบลูทูธ ผู้ทำวิจัยมีข้อเสนอแนะดังนี้

- เพื่อลดระยะเวลาที่ใช้ในการส่งข้อมูลของระบบลง ควรเลือก Sound API ที่ใช้เวลาในการประมวลผลน้อยกว่านี้ ซึ่งนอกจากจะทำให้ระยะเวลาในการส่งข้อมูลลดลงแล้วยังเป็นการเพิ่มจำนวนนักเรียนที่ระบบต้นแบบสามารถให้บริการได้อีกด้วย ดังผลการทดลองที่แสดงให้เห็นว่า ระบบต้นแบบใช้เวลาในการส่งข้อมูล 114 ms เมื่อมีรูปแบบการเชื่อมต่อแบบพิโคเน็ตรองรับจำนวน Client 7 ตัว หากสามารถลดเวลาดังกล่าวลงให้น้อยกว่า 100 ms ได้ ระบบก็จะสามารถให้บริการนักเรียนได้ 7 คนพร้อมกันได้
- ในการสร้างพิโคเน็ต อุปกรณ์ซึ่งทำหน้าที่เป็น Server ที่ตัวครูควรเชื่อมต่อกับ Client ที่ตัวนักเรียนทีละคน เนื่องจากการเชื่อมต่อกับ Client พร้อมกัน ระบบอาจจะไม่สามารถสร้างพิโคเน็ตที่มีจำนวนสมาชิกตามที่ต้องการได้
- RFCOMM หรือ Serial Port Profile (SPP) เป็นทางเลือกหนึ่งในการพัฒนาระบบหากไม่สามารถใช้งาน L2CAP ในการพัฒนาระบบได้ เนื่องจากเป็นโปรโตคอลที่มี

การใช้งาน L2CAP เช่นเดียวกัน แต่ให้อัตราเร็วในการส่งข้อมูล 128 kbps [4] ซึ่งหากต้องการส่งข้อมูลเสียงที่มีความละเอียด 64 kbps ระบบจะให้บริการนักเรียนได้สูงสุดเพียง 2 คนเท่านั้น

- PCM เป็นรูปแบบการ Encode และ Decode ที่เหมาะสมที่สุดของระบบ เพราะมีข้อดีคือใช้การประมวลผลน้อยที่สุด ซึ่งเหมาะกับระบบที่ทำงานบนอุปกรณ์ฝังตัว ถึงแม้จะมีข้อเสียที่ให้ Bandwidth สูง แต่การทำงานของระบบเป็นการทำงานในรูปแบบพิโคเน็ตซึ่งมีจำนวนสมาชิกสูงสุดเพียง 7 ตัวเท่านั้น การบีบอัดข้อมูลเสียงจึงไม่มีความจำเป็นมากนักและอาจใช้เวลาในเพิ่มขึ้นในการบีบอัดข้อมูลเสียงอีกด้วย ซึ่งต่างจากระบบโทรศัพท์ไร้สายที่จำเป็นต้องมีการบีบอัดข้อมูลเสียงเพราะต้องรองรับอุปกรณ์จำนวนมาก
- หากต้องการคุณภาพเสียงที่ดีขึ้น ควรเลือกใช้ Encoder ซึ่งให้ความละเอียดมากกว่า PCM ที่มี Sampling Rate เท่ากับ 8000 Hz และใช้ Bit per Sample เท่ากับ 8 บิต แต่คุณภาพเสียงที่ได้จะมีผลต่อจำนวน Client ของระบบ เพราะเทคโนโลยีบลูทูธมีอัตราเร็วในการให้บริการจำกัดที่ 721 kbps นั่นคือ หากใช้ Encoder ที่มี Sampling Rate เท่ากับ 8000 Hz และใช้ Bit per Sample เท่ากับ 16 บิต จะได้ความละเอียดที่ 128 kbps ซึ่งระบบจะรองรับจำนวน Client ได้สูงสุด 5 ตัว
- การใช้พลังงานเป็นปัจจัยสำคัญอีกข้อหนึ่งในการใช้งานจริงในห้องเรียน ซึ่งสามารถคำนวณได้ดังนี้ Raspberry Pi ใช้แรงดันไฟฟ้า 5V และกระแส 700mA คิดเป็น 3.5W หากเลือกใช้แบตเตอรี่ขนาด 1.7V ปล่อยกระแส 1700mA ต่อชั่วโมง ต้องใช้แบตเตอรี่จำนวน 3 ชุดเพื่อให้ได้แรงดันไฟฟ้า 5V ซึ่งเท่ากับ 8.5W ซึ่งสามารถใช้งานได้ประมาณ 2.4 ชั่วโมงหรือ 2 ชั่วโมง 20 นาที เพื่อให้สามารถใช้งานได้ 6-8 ชั่วโมงต่อวันจึงควรมีแบตเตอรี่สำรองในการใช้งาน นอกจากนี้ยังสามารถปรับปรุงการทำงานของระบบต้นแบบด้วยการเลือกใช้ interrupt แทนการใช้ลูปเพื่อลดภาระการทำงานของ CPU ทำให้ลดการใช้พลังงานลงอีกด้วย

5.3 การดำเนินงานในอนาคต

จากการพัฒนาระบบสื่อสารไร้สายต้นแบบสำหรับนักเรียนที่บกพร่องทางการได้ยินด้วยเทคโนโลยีบลูทูธ ผู้วิจัยมีแนวทางในการดำเนินงานในอนาคตดังนี้

- ทดสอบการทำงานของระบบบนอุปกรณ์ฝังตัวเพิ่มเติม เพื่อหาสาเหตุหรือข้อผิดพลาดในการใช้งานโปรโตคอล L2CAP

- พัฒนาระบบบนอุปกรณ์ฝังตัว เพื่อให้สามารถนำไปใช้งานได้จริงในอนาคต ซึ่งผลการทดสอบการทำงานของระบบบน Raspberry Pi แสดงให้เห็นว่าระบบต้นแบบสามารถทำงานได้บนอุปกรณ์ซึ่งมีทรัพยากรจำกัด
- ปรับปรุงการทำงานของระบบโดยใช้งาน interrupt แทนการใช้ลูปเพื่อลดภาระการทำงานของ CPU ทำให้อุปกรณ์ใช้พลังงานน้อยลง

รายการอ้างอิง

[1] Bakke, M.H., Levitt, H., Ross, M., and Erickson, F. Large Area Assistive Listening System (ALS): Review and Recommendations. Final Report to United Stated Architectural Transportation Barriers. Compliance Board. 1999.

[2] Bluetooth SIG. Bluetooth Specification v4.0 [online]. Available from : http://www.bluetooth.com [2010, April 1]

[3] Suwannarat, A., Pan-ngum, S., and Israsena. P. Evaluation of Bluetooth Based Assistive Listening System. The 2013 International Conference on Information and Communication Technology for Embedded Systems (ICICTES 2013), pp. 53-56. 2013.

[4] Bluetooth SIG. Bluetooth Basics [online]. Available from : http://www.bluetooth.com/Pages/How-it-Works.aspx [2010, April 1]

[5] Wang, A.I., and Ho, J. Is Bluetooth broadcasting practical and useful?. Collaborative Technologies and System, pp. 356-363. 2007.

[6] Floros, A., Koutroubas, M., Tatlas, N.A., and Mourjopoulos. A Study of Wireless Compressed Digital-Audio Transmission. presented at the Audio Engineering Society 112th Convention, Munich. 2002.

[7] Floros, A., Tatlas, A., and Mourjopoulos. Bluebox: A Cable-Free Digital Jukebox for Compressed-Quality Audio Delivery. IEEE Transactions on Consumer Electronics. Vol 51. No. 2. pp. 534-539. 2005.

[8] Floros, A., Tatllas, A., and Mourjopoulos. A High-Quality Digital Audio Delivery Bluetooth Platform. IEEE Transactions on Consumer Electronics. Vol. 52. No. 3. pp. 909-916. 2006.

[9] Aiello, M., De Jong, R., and De Nes, J. Bluetooth Broadcasting: How Far Can We Go? An Experimental Study. 2009 Joint Conference on Pervasive Computing (JCPC). pp. 471-476. 2009.

[10] Choong, Y.N., Lin, C., and Xiaoyu, L. Bluebus: A Scalable Solution for Localized Mobile Service in a Public Bus. Proceedings of 4th International Conference on Mobile Technology and Systems, pp. 712-715. 2007.

[11] Dogan. K., Gurel, G., Kamci, A.K., and Korpeoglu, I. A Performance Analysis of Bluetooth Broadcasting Scheme. Springer Berlin/Heidelberg. Vol. 3992. pp. 996-999. 2006.

[12] Bilan, A. Streaming Audio Over Bluetooth ACL Links. Proceedings of the International Conference on Information Technology: Coding and Computing (Computers and Communications), pp. 287-291. 2003.

[13] Holmes, A. A Comparison of SCO and ACL Packets for Audio Transmission on Bluetooth. Multimedia Systems (MMS), 2003.

[14] Lee, T.J., Jang, K., Kang, H., and Park, J. Model and Performance Evaluation of a Piconet for Point-to-Multipoint Communication in Bluetooth. Vehicular Technology Conference, Vol. 2. pp. 1144-1148. 2002.

[15] Kapoor, R., Chen, L.J., Lee, Y.Z., and Gerla, M. Bluetooth: Carrying Voice over ACL Links. The 4th International Workshop on Mobile and Wireless Communications Network, pp. 379-383. 2002.

[16] Russo, M., Begusic, D., Rozic, N., and Stella, M. Speech Recognition over Bluetooth ACL and SCO Links: A Comparison, Consumer Communications and Networking Conference, pp. 493-497. 2005.

[17] Jung, S., Chang, A., and Gerla, M. Performance Comparison of Overlaid Bluetooth Piconets (OBP) and Bluetooth Scatternet. Wireless Communications and Networking Conference, pp. 505-510. 2006.

[18] Bellavista, P., Stefanelli, C., and Tortonesi, M. QoS Management Middleware Solutions for Bluetooth Audio Distribution. Pervasive and Mobile Computing, pp. 117-138. 2008.

[19] Van Riesen, I., Vlaskamp, F., and Gelderblom, G. Bridging the Gap: Access to Audio Sources for People with Hearing Aids. Assistive Technology – Shaping the Future: AAATE 2003 Conference Proceedings, Vol. 11. No. 1. pp. 505-509. 2003.

[20] BlueCove Team. BlueCove [online]. Available from : http://bluecove.org/ [2013, April 16]

[21] Oracle. Java Sound Programmer Guide [online]. Available from : http://www.oracle.com/technetwork/java/javase/sound-dev-guide-1-159173.pdf [2013, April 16]

[22] Noll, P. Mpeg Digital Audio Coding. Signal Processing Magazine. Vol. 14. pp. 59-81. 1997.

[23] FireStone, S., Ramalingam, T., and Fry, S. Voice and Video Conferencing Fundamental. pp. 223-255. Cisco Press, 2007.

[24] Rasperry Pi. About us [online]. Available from : http://www.raspberrypi.org/about [2013, April 16]

[25] INEX. RaspberryPi Introsheet-Thai [online]. Available from : www.inex.co.th/store/manual/RaspberryPi_IntroSheet-Thai.pdf [2013, April 16]

[26] Neotos, T. Bluecove on ARM [online]. Available from : http://www.neotos.de/en/content/bluecove-arm [2013, April 16]

ภาคผนวก

ภาคผนวก ก.
โค้ดของระบบต้นแบบที่พัฒนาขึ้นด้วยภาษาจาวา

โค้ดของระบบต้นแบบถูกพัฒนาด้วยภาษาจาวา โดยทำงานร่วมกับ Java Sound API และจาวาไลบรารี่สำหรับบลูทูธแอพพลิเคชัน Bluecove (JSR-82 Implementation) โดยประกอบด้วยการทำงาน 3 ส่วน ดังนี้

1. Server ทำหน้าที่ควบคุมการทำงานของอุปกรณ์ที่ตัวครู

BluetoothALSMain.java

```
public class BluetoothALSMain {

	/**
	 * @param args
	 */

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		// Server
		Server server = new Server();
		server.startServer();

	}

}
```

ClientHandler.java

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

import javax.bluetooth.L2CAPConnection;
import javax.bluetooth.RemoteDevice;
import javax.microedition.io.StreamConnection;
import javax.print.attribute.standard.DateTimeAtCompleted;


public class ClientHandler extends Thread{

	// Client Connection
	private StreamConnection rfcommConn; // RFCOMM
	private L2CAPConnection l2capConn; //L2CAP

	private InputStream in;
	private OutputStream out;

	private volatile boolean isRunning = false;
	private String clientName;
```

```
	public ClientHandler(L2CAPConnection conn)
	{
		this.l2capConn = conn;

	    // store the name of the connected client
	    clientName = reportDeviceName();
	    System.out.println("Client: " + clientName);
	}

	private String reportDeviceName()
	/* Return the 'friendly' name of the device being
examined,
	 *         or "device ??" */
	{
		String devName;
		try {
			RemoteDevice rd;
			//if(this.l2capConn != null)
				rd =
RemoteDevice.getRemoteDevice(l2capConn);
			//else
			//    rd =
RemoteDevice.getRemoteDevice(rfcommConn);
			devName = rd.getFriendlyName(false);  // to
reduce connections
		}
		catch (IOException e){
			devName = "device ??";
		}

	    return devName;
	}  // end of reportDeviceName()

	public void send(byte[] data) throws Exception
	{
		try{
			this.l2capConn.send(data);
		}catch(Exception e){
			System.out.println(e.getMessage());
			if(e.getMessage().contains("Failed to write"))
{
				throw e;
			}
			else {
				e.printStackTrace();
			}
		}
	}

	public void closeDown()
	{
		isRunning = false;
	}
}
```

Server.java

```
import java.io.IOException;
import java.util.ArrayList;

import javax.bluetooth.BluetoothStateException;
import javax.bluetooth.DiscoveryAgent;
import javax.bluetooth.L2CAPConnection;
import javax.bluetooth.L2CAPConnectionNotifier;
import javax.bluetooth.LocalDevice;
import javax.microedition.io.Connector;
import javax.microedition.io.StreamConnectionNotifier;


public class Server {

      // UUID and name of the service
      private static final String UUID_STRING =
"11111111111111111111111111111111";

      // 32 hex digits which will become a 128 bit ID
      private static final String SERVICE_NAME = "alsserver";

      // Server
      private StreamConnectionNotifier rfcommServer;  // RFCOMM
      private L2CAPConnectionNotifier l2capServer; //L2CAP

      private ArrayList<ClientHandler> handlers;
      private volatile boolean isRunning = false;

      public Server()
      {
            handlers = new ArrayList<ClientHandler>();

            Runtime.getRuntime().addShutdownHook(new Thread() {
                  public void run()
                  {
                        closeDown();
                  }
            });

            initDevice();

            // start Console
            Console console = new Console(this);
            console.pack();
      }
```

```
private void initDevice()
      {
            try {  // make the server's device discoverable
                  LocalDevice local =
LocalDevice.getLocalDevice();
                  System.out.println("Device name: " +
local.getFriendlyName());
                  System.out.println("Bluetooth Address: " +
local.getBluetoothAddress());
                  boolean res =
local.setDiscoverable(DiscoveryAgent.GIAC);
                  System.out.println("Discoverability set: " +
res);
            }
            catch (BluetoothStateException e) {
                  System.out.println(e);
                  System.exit(1);
            }
      }

      private void createL2CAPConnection()
      {
            try {
                  System.out.println("Start advertising " +
SERVICE_NAME + "...");

                  l2capServer = (L2CAPConnectionNotifier)
Connector.open(
                              "btl2cap://localhost:" +
UUID_STRING +
                              ";name=" + SERVICE_NAME +
";authenticate=false" + ";master=true" +

      ";ReceiveMTU=8000;TransmitMTU=8000");
                  /* for most devices, with authenticate=false
there *
                   *                  should be no need for pin
pairing */
                  }
            catch (IOException e) {
                  System.out.println(e);
                  System.exit(1);
            }
      }
      public void startServer()
      {
            createL2CAPConnection();
            processClients();
      }
      public void closeServer()
      {
            closeDown();
      }
      public void broadcastAudio(byte[] audio)
      {
            for(ClientHandler hand: handlers)
            {
                  hand.send(audio);
            }
      }
```

```
      private void processClients()
      // Wait for client connections, creating a handler for
each one
      {
            isRunning = true;
            try {
                  System.out.println("Waiting for incoming
connection...");
                  while (isRunning) {

//            StreamConnection conn = server.acceptAndOpen();
              //L2CAP
              L2CAPConnection conn =
l2capServer.acceptAndOpen();
                 // wait for a client connection
                 /* acceptAndOpen() also adds the service record
to the
                    device's SDDB, making the service visible to
clients */
              System.out.println("Connection requested...");

              ClientHandler hand = new ClientHandler(conn);
                    // create client handler
              handlers.add(hand);
              hand.start();
            }
          }
          catch (IOException e)
          {  System.out.println(e); }
        }  // end of processClients()

      private void closeDown()
      /* Stop accepting any further client connections, and
close down
       *         all the handlers. */
      {
            System.out.println("Closing down server");
            if (isRunning) {
                  isRunning = false;
                  try {
                        l2capServer.close();
                        // close connection, and remove service
record from SDDB
                  }
                  catch (IOException e)
                  {
                        System.out.println(e);
                  }

                  // close down all the handlers
                  for (ClientHandler hand : handlers)
                        hand.closeDown();
                  handlers.clear();
            }
      }  // end of closeDown();
}
```

CaptureThread.java

```
import java.io.IOException;
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.DataLine;
import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.TargetDataLine;
public class CaptureThread extends Thread{

      protected boolean running;
      private ByteArrayOutputStream out;
      private byte[] audio_buffer = new byte[80000];
      private Console console;
      private boolean isStreaming = true;

      // audio format
      private AudioFormat getFormat()
      {
            float sampleRate = 8000;
          int sampleSizeInBits = 8;
          int channels = 1;
          boolean signed = true;
          boolean bigEndian = true;
          return new AudioFormat(sampleRate, sampleSizeInBits,
channels, signed, bigEndian);
      }
      public CaptureThread(Console console, boolean isStreaming)
      {
            this.console = console;
            this.isStreaming = isStreaming;
            final AudioFormat format = getFormat();
            DataLine.Info info = new
DataLine.Info(TargetDataLine.class, format);
            final TargetDataLine line = (TargetDataLine)
            AudioSystem.getLine(info);
            line.open(format, 320);
            line.start();
      }
      public void run()
      {
      try {
            byte[] buffer =  new byte[320];
            running = true;
            try {
                  while (running) {
                  int count = line.read(buffer, 0, buffer.length);
            this.console.broadcastAudio(buffer);
                        }
                        out.close();
                  } catch (IOException e) {
                        System.err.println("I/O problems: " + e);
                        System.exit(-1);
                  }
            } catch (LineUnavailableException e) {
                  System.err.println("Line unavailable: " + e);
                  System.exit(-2);
          }
      }
}
```

2. Client ทำหน้าที่ควบคุมการทำงานของอุปกรณ์ที่ตัวนักเรียน

BluetoothALSMain.java

```
public class BluetoothALSMain {

	/**
	 * @param args
	 */
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		// Client
		do {
			Client client = new Client();
			client.makeContact();
			client.read();

			synchronized(client.lost) {
				while(!client.isLost) {
					try {
						client.lost.wait();
					} catch (InterruptedException e) {
						e.printStackTrace();
					}
				}
			}
			client.isLost = false;
		} while(true);
	}
}
```

Client.java

```
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.Vector;
import java.util.concurrent.SynchronousQueue;

import javax.bluetooth.BluetoothStateException;
import javax.bluetooth.DiscoveryAgent;
import javax.bluetooth.L2CAPConnection;
import javax.bluetooth.LocalDevice;
import javax.bluetooth.RemoteDevice;
import javax.bluetooth.ServiceRecord;
import javax.microedition.io.Connector;
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.DataLine;
import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.SourceDataLine;


public class Client {

      private L2CAPConnection l2capConn;

      private static int SERVICE_NAME_ATTRID = 0x0100;
      public static final long RFCOMM = 0x0003;
      public static final long L2CAP = 0x0100;

      // UUID and name of the echo service
      private static final String UUID_STRING =
"11111111111111111111111111111111";
      private static final String SERVICE_NAME = "alsserver";
// use lowercase

      private ServiceBrowser sb;
      private LocalDevice local;
      private DiscoveryAgent agent;

      private ByteArrayOutputStream buffer;

      private boolean searchSucceed = false;
      private ServiceRecord srvRecord = null;

      private boolean isRunning = true;
      private SourceDataLine line = null;

      public SynchronousQueue<byte[]> queue = new
SynchronousQueue();
      public Boolean continueProducing = Boolean.TRUE;

      public boolean isLost = false;
      public Object lost = new Object();
```

```
public void put(byte[] data) throws InterruptedException {
            this.queue.put(data);
      }

      public byte[] get() throws InterruptedException {
            return this.queue.poll();
      }

      public Client()
      {
            startProcess();
      }

      public void startProcess() {
            searchSucceed = false;
            while(!searchSucceed) {
                  searchDevice();
                  searchService();
            }
      }

      private void searchDevice() {
            try {
                  sb = new ServiceBrowser(L2CAP, UUID_STRING,
SERVICE_NAME);
                  local = LocalDevice.getLocalDevice();
                  agent = local.getDiscoveryAgent();

                  //device inquiry
                  agent.startInquiry(DiscoveryAgent.GIAC, sb);
                  synchronized(sb){
                        try{
                              sb.wait();
                        }catch(Exception e){
                              e.printStackTrace();
                        }
                  }

            } catch (BluetoothStateException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }
      }
```

```
private void searchService() {
//          int numTries = 0;
//          searchSucceed = false;
//          while(!searchSucceed){
                  try{
//                      numTries++;
//                      System.out.println("numTries: " +
numTries);
                        int[] attrSet = new int[1];
                        attrSet[0] = SERVICE_NAME_ATTRID;
                        javax.bluetooth.UUID[] uuids =
this.setUUIDs(L2CAP, UUID_STRING);

                        for(int i=0; i<sb.num_discovered; i++)
                        {
                              RemoteDevice rd =
sb.discovered[i];
                              agent.searchServices(attrSet,
uuids, rd, sb);
                              synchronized(sb){
                                    try{
                                          sb.wait();
                                          Vector<ServiceRecord>
servicesVec = sb.getServicesFound();

                                          if ((servicesVec ==
null) || (servicesVec.size() == 0)) {

      System.out.println("No services found");
                                          }
                                          else
                                          {
                                                for
(ServiceRecord sr : servicesVec) {
                                                      String
servName = sb.getServiceName(sr);

      if(servName.contains(SERVICE_NAME))
                                                      {

      srvRecord = sr;

      searchSucceed = true;

      break;
                                                      }
                                                }
                                          }
                                    }catch(Exception e){
                                          e.printStackTrace();
                                    }
                              }
                        }
                  }catch (BluetoothStateException e) {
                  // TODO Auto-generated catch block

                        e.printStackTrace();
                  }
//          }
      }
```

```
public void makeContact() {
            // TODO Auto-generated method stub
            // get a URL for the service

          String servURL = srvRecord.getConnectionURL(

ServiceRecord.NOAUTHENTICATE_NOENCRYPT, false);
//        servURL = null;
          if (servURL == null) {
            System.out.println("No service available");
            return;
          }
          System.out.println("\nConnecting to service " +
servURL + " ...");
          try {
            l2capConn = (L2CAPConnection) Connector.open(servURL
+ ";ReceiveMTU=800;TransmitMTU=800");
            System.out.println("Opened a connection to server");
            System.out.println("Transmit MTU: " +
l2capConn.getTransmitMTU());
            System.out.println("Receive MTU: " +
l2capConn.getReceiveMTU());
            //isClosed = false;   // i.e. the connection is open
          }
          catch (Exception ex)
          {
            System.out.println(ex);
            System.out.println("l2capConn:" + (l2capConn ==
null));
            System.out.print("\nStackTrace: ");
            ex.printStackTrace();
            searchService();
//          System.exit(1);
          }
      }

      public void read()
      {
            Listener listener = new Listener(this);
            listener.start();
      }

      public L2CAPConnection getL2CAPConnection()
      {
            return this.l2capConn;
      }
```

```
private javax.bluetooth.UUID[] setUUIDs(long protocolUUID,
String UUIDStr)
      {
            javax.bluetooth.UUID[] uuids;
            if (UUIDStr != null)
                  uuids = new javax.bluetooth.UUID[2];
            else
                  uuids = new javax.bluetooth.UUID[1];

          // add the UUIDs for the protocol and service
            uuids[0] = new javax.bluetooth.UUID(protocolUUID);
            if (UUIDStr != null)
                  uuids[1] =  new javax.bluetooth.UUID(UUIDStr,
false);

          return uuids;
        }  // end of setUUIDs()

      public void closeDown() {
            try {
                  // close SourceDataLine
                  line.drain();
                  line.close();
                  line = null;

                  // set parameter to be null
                  sb.close();

                  // close connection
                  l2capConn.close();

            } catch (IOException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }
      }
}
```

Listener.java

```
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;

import javax.bluetooth.L2CAPConnection;
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.DataLine;
import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.SourceDataLine;
```

```
public class Listener extends Thread {

	private L2CAPConnection l2capConn;
	private ByteArrayOutputStream buffer;
	private SourceDataLine line;
	private AudioInputStream ais;
	private Client client;

	public Listener(Client c){
		client = c;
		l2capConn = c.getL2CAPConnection();
		buffer = c.getBuffer();

		AudioFormat format = getFormat();
		DataLine.Info info = new
DataLine.Info(SourceDataLine.class, format);
		try {
			line =
(SourceDataLine)AudioSystem.getLine(info);
			line.open(format, 320);
			line.start();
		} catch (LineUnavailableException e) {
			// TODO Auto-generated catch block
			line.drain();
			line.close();
			e.printStackTrace();
		}
	}

	public void run()
	{
		int offset = 0;
		while(true){
			byte[] data = new byte[320];
			try {

	System.out.print(System.currentTimeMillis() + " ");
				int bytes_read =
l2capConn.receive(data);
//				System.out.println(bytes_read);

	System.out.println(System.currentTimeMillis());
//				buffer.write(data, 0, bytes_read);


				try {
					line.write(data, 0, bytes_read);

				} catch (Exception e) {
					// TODO Auto-generated catch block
					line.drain();
					line.close();
					e.printStackTrace();
				}
//
	System.out.println(System.currentTimeMillis());
```

```
} catch (IOException e) {
                        // TODO Auto-generated catch block
                        if(e.getMessage().equals("Peer closed
connection") || e.getMessage().equals("Connection closed")) {

                              // clear parameters' values
                              client.closeDown();

                              // auto reconnection
                              client.startProcess();
                              client.makeContact();
                        }
                        else
                              e.printStackTrace();
                  }
            }
      }

      private AudioFormat getFormat() {
            // TODO Auto-generated method stub
            float sampleRate = 8000;
            int sampleSizeInBits = 8;
            int channels = 1;
            boolean signed = true;
            boolean bigEndian = true;
            return new AudioFormat(sampleRate, sampleSizeInBits,
channels, signed, bigEndian);
      }
}
```

ServiceBrowser.java

```
import java.io.IOException;
import java.util.HashMap;
import java.util.Vector;

import javax.bluetooth.DataElement;
import javax.bluetooth.DeviceClass;
import javax.bluetooth.DiscoveryAgent;
import javax.bluetooth.DiscoveryListener;
import javax.bluetooth.LocalDevice;
import javax.bluetooth.RemoteDevice;
import javax.bluetooth.ServiceRecord;


public class ServiceBrowser implements DiscoveryListener {

      private static int SERVICE_NAME_ATTRID = 0x0100;
      public static final long RFCOMM = 0x0003;
      public static final long L2CAP = 0x0100;
      RemoteDevice discovered[] = new RemoteDevice[255];
      int num_discovered = 0;

      // stores the remote devices found during the device
search
      private HashMap<String,RemoteDevice> devicesMap = null;

      // stores the services found during the service search
      private Vector<ServiceRecord> servicesVec = null;
```

```
// for constructor
      private long protocolUUID;
      private String UUIDStr;
      private String serviceStr;

      public ServiceBrowser(long protocolUUID, String UUIDStr,
String serviceStr)
      {
            this.protocolUUID = protocolUUID;
            this.UUIDStr = UUIDStr;
            this.serviceStr = serviceStr;
      }

      public Vector<ServiceRecord> getServicesFound()
      {
            return servicesVec;
      }

      public String getServiceName(ServiceRecord sr)
      {
            DataElement d =
sr.getAttributeValue(SERVICE_NAME_ATTRID);
            if(d != null)
            {
                  return (String)d.getValue();
            }
            else
            {
                  return "unnamed service";
            }
      }

      @Override
      public void deviceDiscovered(RemoteDevice rd, DeviceClass
cod) {
            // TODO Auto-generated method stub

            int majorDC = cod.getMajorDeviceClass();

          // restrict matching device to PC or Phone
          if ((majorDC != 0x0100) && (majorDC != 0x0200)) {
            System.out.println("Device not PC or phone, so
rejected");
            return;
          }

            String addr = rd.getBluetoothAddress();
            String name = "";
            try{
                  name = rd.getFriendlyName(true);
                  this.discovered[this.num_discovered] = rd;
                  this.num_discovered++;

                  if(devicesMap == null)
                        devicesMap = new
HashMap<String,RemoteDevice>();
                  if (devicesMap.get(name) != null)
                        System.out.println(" - updating existing
info");
                  devicesMap.put(name, rd);
```

```
System.out.println("Discovered: " + name + ":" + addr);
            }catch(IOException e){
                  e.printStackTrace();
            }
      }
      @Override
      public void inquiryCompleted(int status) {
            // TODO Auto-generated method stub
            System.out.println("----Device Inquiry Complete----
");
            synchronized(this){
                  try{
                        this.notifyAll();
                  }catch(Exception e){
                        e.printStackTrace();
                  }
            }
      }
      @Override
      public void serviceSearchCompleted(int transID, int arg1)
{
            // TODO Auto-generated method stub
            System.out.println("----Service Search Complete----
");
            synchronized(this){
                  try{
                        this.notifyAll();
                  }catch(Exception e){
                        e.printStackTrace();
                  }
            }
      }
      @Override
      public void servicesDiscovered(int transID,
ServiceRecord[] servRecords) {
            // TODO Auto-generated method stub
            System.out.println("----Service Discovered----");
            if(servicesVec == null)
                  servicesVec = new Vector<ServiceRecord>();
            if ((servRecords != null) && (servRecords.length >
0))
            {
                  for (ServiceRecord sr : servRecords)
                  {
                        if (sr != null)
                        {
                              servicesVec.add(sr);
                              System.out.println("Service: " +
getServiceName(sr));
                        }
                    else
                    {
                        System.out.println("null service");

                    }
                  }
            }
      }
```

```
public int numOfDiscoveredDevices(){
            return this.num_discovered;
      }

      public RemoteDevice[] discoveredDevices(){
            return this.discovered;
      }

      public void close() {
            num_discovered = 0;
            devicesMap = null;
            servicesVec = null;
      }
}
```

3. Relay ทำหน้าที่ควบคุมการทำงานของอุปกรณ์ซึ่งเป็นสมาชิกของ 2 พิโคเน็ต ในการเชื่อมต่อแบบ Scatternet

Main.java

```
public class Main {

      /**
       * @param args
       */
      public static void main(String[] args) {
            // TODO Auto-generated method stub

            // start slave
            Client client = new Client();
            client.makeContact();

            // start master
            Server server = new Server();
            server.startServer();

            Relay relay = new Relay(this.server, this.client);
            relay.start();
      }
}
```

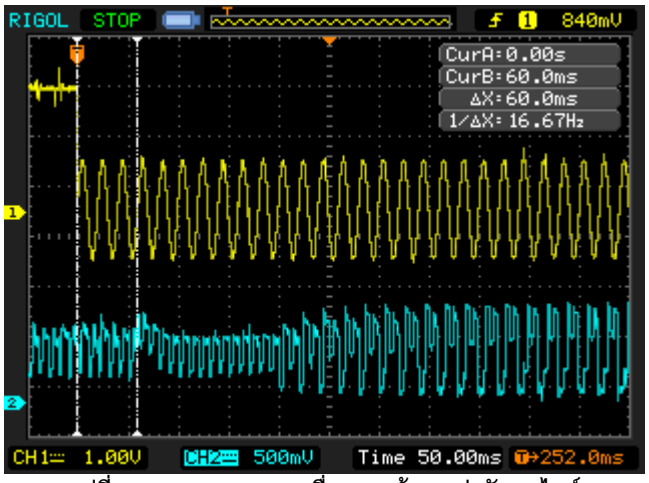
Relay.java

```
import java.io.IOException;


public class Relay extends Thread{

	private Server server;
	private Client client;
	private final int data_size = 320;

	public Relay(Server server, Client client) {
		this.server = server;
		this.client = client;
	}

	public void run() {
		byte[] data = new byte[this.data_size];
		while(true) {
			try {

	System.out.print(System.currentTimeMillis() + " ");
				int bytes_read =
this.client.getL2CAPConnection().receive(data);
//				System.out.println("read: " +
bytes_read);
				this.server.broadcastAudio(data);

	System.out.println(System.currentTimeMillis());
			} catch (IOException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
		}
	}

}
```

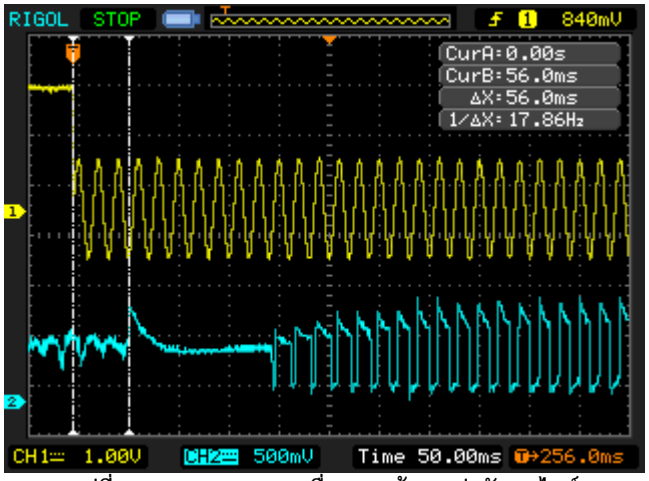
ภาคผนวก ข.

ระยะเวลาในการส่งข้อมูล (Latency) ของระบบต้นแบบเมื่อทำการเชื่อมต่อแบบพิโคเน็ต

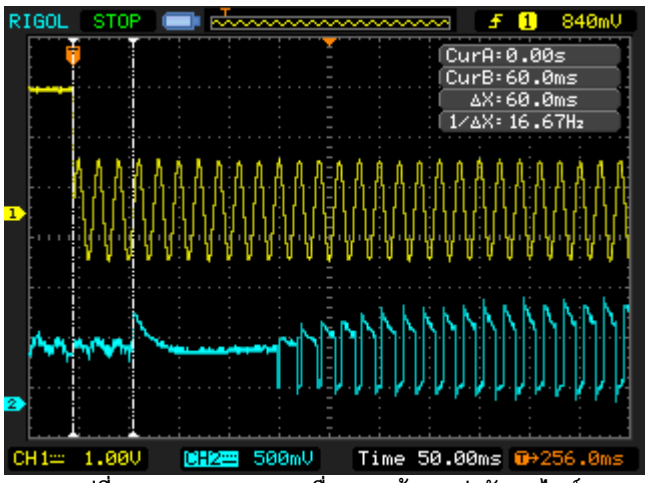
เมื่อทำการเชื่อมต่อแบบพิโคเน็ต ตามผลการทดลองดังตารางที่ 11 แสดงผลของขนาดข้อมูลและขนาดบัฟเฟอร์ ซึ่งส่งผลต่อระยะเวลาในการส่งข้อมูล (Latency) ดังนี้



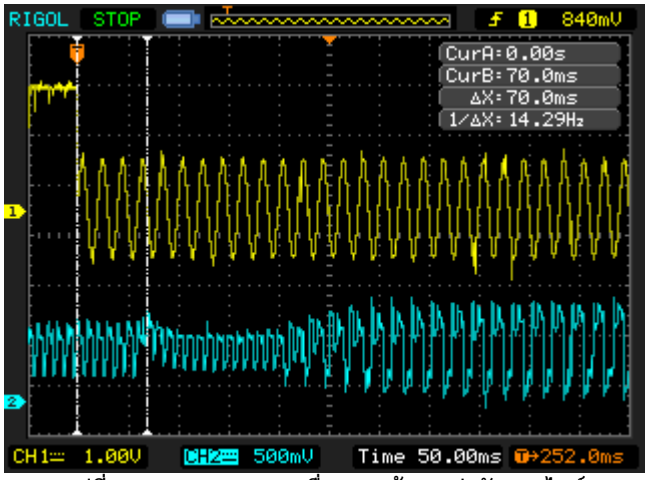
รูปที่ 63 Latency ของระบบเมื่อขนาดข้อมูลเท่ากับ 20 ไบต์



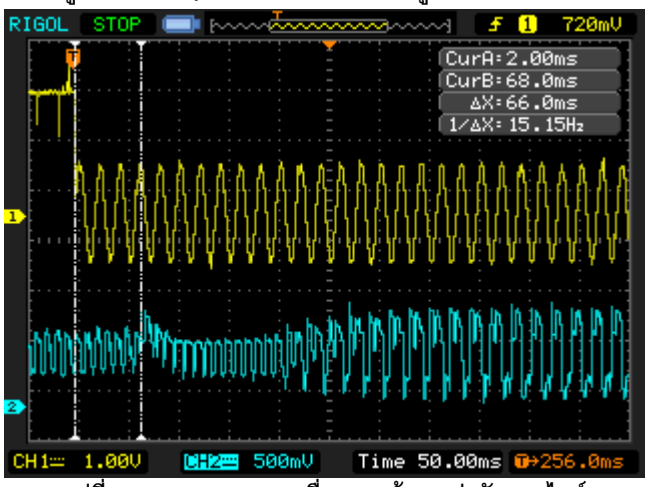
รูปที่ 64 Latency ของระบบเมื่อขนาดข้อมูลเท่ากับ 40 ไบต์



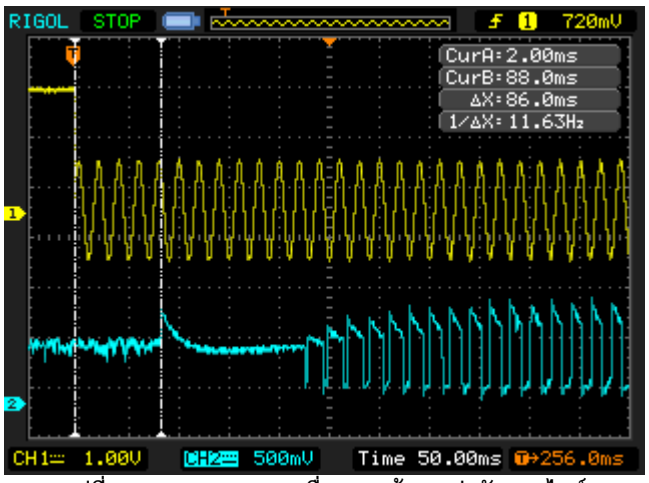
รูปที่ 65 Latency ของระบบเมื่อขนาดข้อมูลเท่ากับ 80 ไบต์



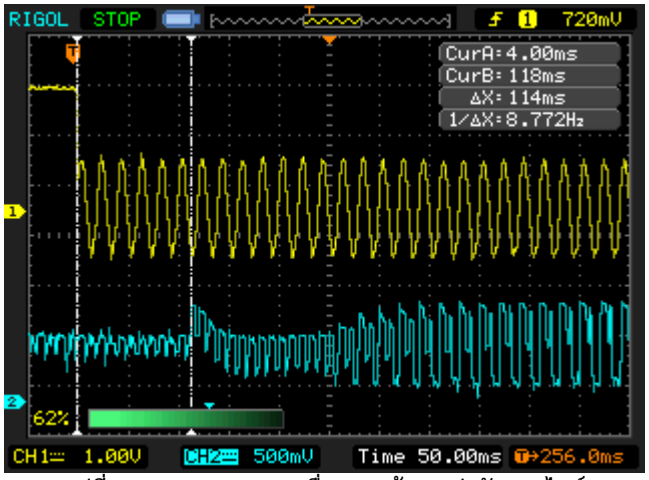
รูปที่ 66 Latency ของระบบเมื่อขนาดข้อมูลเท่ากับ 120 ไบต์



รูปที่ 67 Latency ของระบบเมื่อขนาดข้อมูลเท่ากับ 160 ไบต์



รูปที่ 68 Latency ของระบบเมื่อขนาดข้อมูลเท่ากับ 240 ไบต์



รูปที่ 69 Latency ของระบบเมื่อขนาดข้อมูลเท่ากับ 320 ไบต์

ประวัติผู้เขียนวิทยานิพนธ์

นายอาภา สุวรรณรัตน์ เกิดเมื่อวันที่ 8 พฤษภาคม พ.ศ. 2528 ที่จังหวัดสงขลา สำเร็จการศึกษาหลักสูตรวิศวกรรมศาสตร์บัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ จากภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์ ในปีการศึกษา 2550 และเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตร์มหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ ที่ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2552