การออกแบบตัวเครื่องคำนวณปรับค่าพารามิเตอร์พีไอดีอัตโนมัติบน iOS

DESIGN OF AUTO-TUNING PID PARAMETER CALCULATOR BASED ON iOS

วิภาวรรณ คำสวนจิก

ศศิพงศ์ สมบูรณ์ใจ

ชวณัฏฐ์ ศรีบุญญา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2555

การออกแบบตัวเครื่องคำนวณปรับค่าพารามิเตอร์พีไอดีอัตโนมัติบน iOS

DESIGN OF AUTO-TUNING PID PARAMETER CALCULATOR BASED ON iOS

วิภาวรรณ คำสวนจิก

ศศิพงค์ สมบูรณ์ใจ

ชวณัฏฐ์ ศรีบุญญา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2555

DESIGN OF AUTO-TUNING PID PARAMETER CALCULATOR BASED ON iOS

WIPAWAN KHAMSUANJIK

SASIPONG SOMBOONJAI

CHAWANAT SRIBUNYA

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2012

ปริญญานิพนธ์ปีการศึกษา 2555

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

หัวข้อปริญญานิพนธ์	การออกแบบตัวเครื่องคำนวณปรับค่าพารามิเตอร์พีไอดีอัตโนมัติบน iOS DESIGN OF AUTO-TUNING PID PARAMETER CALCULATOR BASED ON iOS	
นักศึกษาผู้จัดทำ	นางสาววิภาวรรณ คำสวนจิก	**รหัสนักศึกษา** 50011475
	นายศศิพงค์ สมบูรณ์ใจ	**รหัสนักศึกษา** 51011288
	นายชวณัฏฐ์ ศรีบุญญา	**รหัสนักศึกษา** 52010230
ปริญญา	วิศวกรรมศาสตรบัณฑิต	
สาขาวิชา	วิศวกรรมการวัดคุม	
ปีการศึกษา	2555	

อาจารย์ผู้ควบคุมปริญญานิพนธ์	ลายมือชื่อ
รศ.อาจินต์ น่วมสำราญ รศ.ดร.วิทยา ทิพสุวรรณพร	

หัวข้อปริญญานิพนธ์	การออกแบบตัวคำนวณปรับค่าพารามิเตอร์พีไอดีอัตโนมัติบน iOS DESIGN OF AUTO-TUNING PID PARAMETER CALCULATOR BASED ON iOS	
นักศึกษาผู้จัดทำ	นางสาววิภาวรรณ คำสวนจิก	**รหัสนักศึกษา** 50011475
	นายศศิพงศ์ สมบูรณ์ใจ	**รหัสนักศึกษา** 51011288
	นายชวณัฏฐ์ ศรีบุญญา	**รหัสนักศึกษา** 52010230
อาจารย์ที่ปรึกษา	รศ.อาจินต์ น่วมสำราญ	
	รศ.ดร.วิทยา ทิพย์สุวรรณพร	
ปีการศึกษา	2555	

บทคัดย่อ

การออกแบบตัวเครื่องคำนวณปรับค่าพารามิเตอร์พีไอดีอัตโนมัติบน iOS เป็นโครงงานวิจัยเพื่อทำการศึกษาและออกแบบตัวปรับค่าพีไอดีพารามิเตอร์พีไอดีอัตโนมัติบนอุปกรณ์ iOS ได้แก่ iPhone, iPod และ iPad โดยจัดทำการทดลองขึ้นด้วยกัน 2 แบบ คือ 1.ทำการออกแบบตัวปรับค่าพีไอดีพารามิเตอร์อัตโนมัติ โดยใช้โปรแกรม TouchOSC ในการสร้าง Layout เพื่อเชื่อมต่อกับแบบจำลองโมเดลอ้างอิงของกระบวนการควบคุมแบบพีไอดีพารามิเตอร์บนโปรแกรม LabVIEW ผ่านทางเครือข่าย wi-fi ซึ่งโปรแกรม TouchOSC ทำหน้าที่เป็นตัว MIDI Controller บนอุปกรณ์ iOS ซึ่งสามารถสั่งงานผ่านทางเครือข่าย wi-fi ได้ และ 2. ทำการสร้าง Application บนอุปกรณ์ iOS ด้วยโปรแกรม Xcode บนคอมพิวเตอร์ที่ติดตั้งระบบปฏิบัติการ MAC OS X เพื่อที่จะสามารถทำการควบคุมหรือปรับค่าพีไอดีพารามิเตอร์อัตโนมัติได้จากระยะไกลและแสดงผลบนหน้าจอของอุปกรณ์ iOS เช่น iPhone, iPod และ iPad

Thesis Title	DESIGN OF AUTO-TUNING PID PARAMETER CALCULATOR BASED ON iOS
Authors	Ms. Wipawan Khamsuanjik **Student ID.** 50011475
	Mr. Sasipong Somboonjai **Student ID.** 51011288
	Mr. Chawanat Sribunya **Student ID.** 52010230
Thesis Advisor	Assoc.Prof. Arjin Numsomran
	Assoc.Prof.Dr. Vittaya Tipsuwanporn
Year	2012

ABSTRACT

The Design of auto-tuning PID parameter calculator based on iOS is the project to study and design the auto-tuning PID parameter based on iOS devices such as iPhone, iPod and iPad. By we test experiments with two different models. First is Creating The Design of auto-tuning PID parameter layout by TouchOSC program for linking with the reference model of PID parameter controlling process on LabVIEW program via wi-fi. Second, creating the iOS application by using Xcode program on computer that use Mac OS X Operating System to controlling and monitoring the PID parameter value or adapt the PID parameter. The result will be shown on iOS devices display such as iPhone, iPod and iPad.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จเป็นอย่างดีด้วยคำแนะนำ คำปรึกษาและอนุเคราะห์อุปกรณ์เครื่องมือในการทดลองจาก รศ. อาจินต์ น่วมสำราญ และ รศ.ดร. วิทยา ทิพย์สุวรรณพร อาจารย์ผู้ควบคุมปริญญานิพนธ์ และเป็นผู้ปรับวิธีคิดและทัศนคติเกี่ยวกับการทดลองและการดำเนินชีวิตให้กับคณะผู้จัดทำด้วยดีเสมอมา ข้าพเจ้ารู้สึกซาบซึ้งใจในความอนุเคราะห์จากท่าน และขอบพระคุณเป็นอย่างสูง

ขอกราบขอบพระคุณคณาจารย์สาขาวิชาวิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุกๆท่านที่ได้ประสิทธิ์ประสาทวิชาให้กับคณะผู้จัดทำ

ขอขอบพระคุณ บิดา มารดา ของคณะผู้จัดทำทุกท่าน ที่ท่านได้ให้ความอนุเคราะห์ ในด้านค่าใช้จ่าย และคอยให้กำลังใจกับคณะผู้จัดทำด้วยความรักและความหวังดีเสมอมา ทำให้คณะผู้จัดทำสามารถทำวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี

ขอขอบพระคุณ คุณสราวุธ พลเสน เป็นอย่างยิ่ง สำหรับคำแนะนำและแนวทางที่ดีเสมอมา

ขอขอบคุณเพื่อนๆ พี่ๆ น้องๆ สาขาวิชาวิศวกรรมการวัดคุมที่เป็นกำลังใจ รวมทั้งให้ความช่วยเหลือในเรื่องต่างๆที่เกี่ยวข้องกับการศึกษาตลอดมา

คุณค่าและประโยชน์อันพึงมาจากปริญญานิพนธ์ฉบับนี้คณะผู้จัดทำขอมอบแด่ผู้มีพระคุณทุกท่าน

คณะผู้จัดทำ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญภาพ

ภาพที่ หน้า

สารบัญภาพ(ต่อ)

ภาพที่ **หน้า**

สารบัญภาพ(ต่อ)

ภาพที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความสำคัญของปริญญานิพนธ์

เนื่องจากในสมัยปัจจุบันนี้ มีความก้าวหน้าทางเทคโนโลยีไปมาก ซึ่งในระบบอุตสาหกรรมนั้นจำเป็นที่จะต้องการความรวดเร็วและแม่นยำ เนื่องจากต้องทำการแข่งขันกับเวลา ซึ่งถ้าเกิดความล่าช้าหรือความผิดพลาดขึ้น นั่นอาจจะหมายถึงความเสียหายในทางธุรกิจ โดยปริญญานิพนธ์เล่มนี้จะทำการนำเสนอการออกแบบตัวคำนวณปรับค่าพีไอดีพารามิเตอร์อัตโนมัติบนอุปกรณ์ iOS เช่น iPhone, iPad, iPod ซึ่งเป็นอุปกรณ์ Smart devices ที่มีความสามารถในการทำงานที่หลากหลาย เพื่อความสะดวกสบายและรวดเร็วในการปรับค่าพารามิเตอร์ ซึ่งทำให้สามารถปรับค่าพารามิเตอร์ได้จากระยะไกล โดยไม่จำเป็นต้องเข้าไปทำการปรับค่าพารามิเตอร์ที่หน้างานโดยตรง ซึ่งอาจจะมีความลำบากและยุ่งยากในการเข้าไปทำการปรับค่าพารามิเตอร์ด้วยตนเอง

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1. ศึกษาและออกแบบเครื่องคำนวณปรับค่าพารามิเตอร์ตัวควบคุมพีไอดี
2. ศึกษาและเขียนโปรแกรมเครื่องคำนวณปรับค่าพารามิเตอร์ของตัวควบคุมพีไอดีบน iOS
3. ศึกษาวิธีวิเคราะห์ผลตอบสนองของกระบวนการในการนำไปประยุกต์ใช้ในการหาค่าพารามิเตอร์ของระบบ

1.3 ขอบเขตของปริญญานิพนธ์

1. จัดทำโปรแกรมเครื่องคำนวณปรับค่าพารามิเตอร์พีไอดีบนiOS
2. จัดทำวีดีโอและคู่มือการใช้งานโปรแกรม
3. ประยุกต์ใช้กับกระบวนการ Level Control, Temp Control หรือ Position Control

1.4 ขั้นตอนการศึกษา

1. ศึกษาการใช้งานโปรแกรม LabVIEW และ TouchOSC
2. ศึกษาวิธีการสร้าง Layout บนโปรแกรม TouchOSC Editor
3. ศึกษาทฤษฎีการควบคุมพีไอดีพารามิเตอร์แบบอัตโนมัติ โดยวิธี Step test และวิธี Ultimate Gain
4. ศึกษาวิธีการเขียนภาษา Objective-C บนโปรแกรม Xcode

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. รู้และเข้าใจรูปแบบการเขียนภาษา G (Graphic) ฟังก์ชัน วิธีการใช้งาน รวมทั้งคุณสมบัติของการทำงานต่างๆ บนโปรแกรม LabVIEW TouchOSC และวิธีการสร้าง Layout โดยใช้โปรแกรม TouchOSC Editor
2. เข้าใจทฤษฎีการควบคุมค่าพารามิเตอร์พีไอดีแบบอัตโนมัติ โดยใช้วิธีการของ Ziegler-Nichols ทั้งการทดลองแบบวงเปิด (Step test) และแบบวงปิด (Ultimate Gain)
3. สามารถทำการเขียนโปรแกรมด้วยภาษา Objective-C ซึ่งเป็นภาษาโปรแกรมเชิงวัตถุได้
4. สามารถนำหลักการทั้งการควบคุมค่าพีไอดีพารามิเตอร์ การเขียนภาษา G และการเขียนภาษา Objective-C เข้ามาประยุกต์ใช้ร่วมกัน เพื่อทำการสร้างตัวควบคุมค่าพีไอดีพารามิเตอร์บนอุปกรณ์ iOS

บทที่ 2
ทฤษฎีและหลักการที่เกี่ยวข้อง

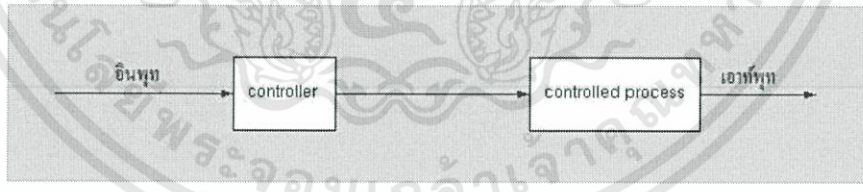
2.1 ทฤษฎีการควบคุม

การควบคุมนั้นเป็นส่วนที่สำคัญของระบบควบคุมแบบอัตโนมัติ โดยตัวควบคุม (Controller) ทำหน้าที่เปรียบเทียบและทำการปรับแต่งผลตอบสนองของระบบให้เป็นไปตามที่ต้องการ ซึ่งสัญญาณที่ส่งออกมาควบคุมกระบวนการนั้น จะเป็นไปตามรูปแบบและกฎที่ผู้ทำการควบคุมได้ทำการเลือกไว้ โดยผลการควบคุมนั้นจะถูกวัด และป้อนกลับไปยังตัวควบคุมเพื่อทำการเปรียบเทียบกับค่าที่ได้กำหนดไว้ และจะกระทำซ้ำๆเช่นนี้ไปเรื่อยๆ

โดยการควบคุมนั้นเราสามารถแบ่งออกได้เป็น 2 ประเภท ได้แก่

1. **ระบบควบคุมแบบวงเปิด (Open-loop Control System)**

   ระบบควบคุมแบบวงเปิดนั้นก็คือระบบควบคุมที่ไม่มีการป้อนกลับของค่าเอาท์พุท โดยค่าเอาท์พุทที่ได้มาจะไม่มีผลต่อการควบคุมของระบบ ซึ่งจะไม่มีการนำค่าเอาท์พุทที่ได้มาเปรียบเทียบกับค่าอินพุทที่ป้อนเข้าไปยังระบบควบคุม ดังรูปที่ 2.1 ดังนี้

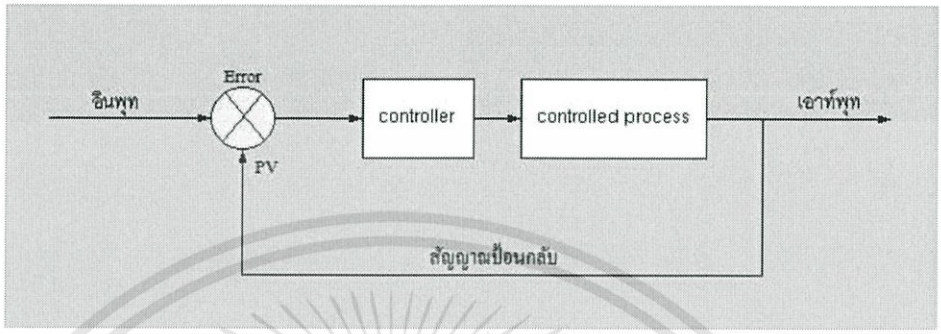


รูปที่ 2.1 แสดงบล็อกไดอะแกรมของการควบคุมแบบวงเปิด

2. **ระบบควบคุมแบบวงปิด (Closed-loop Control System)**

   ระบบควบคุมแบบวงปิดเป็นระบบที่มีการปรับปรุงและเพิ่มประสิทธิภาพของการควบคุมให้เพิ่มขึ้นจากระบบควบคุมแบบวงปิด โดยระบบควบคุมแบบวงปิดจะมีการนำสัญญาณเอาท์พุทที่ได้จากระบบนั้นมาป้อนกลับ เพื่อทำการเปรียบเทียบกับสัญญาณอินพุทที่ป้อนเข้ามาภายในระบบ ซึ่งความแตกต่างระหว่างสัญญาณทั้งสองที่นำมาเปรียบเทียบกันนั้นก็คือ สัญญาณค่าผิดพลาด (Error) เพื่อที่จะใช้เป็นสัญญาณป้อนเข้าไปยังตัว

ควบคุม (Controller) เพื่อให้ตัวควบคุมนำไปสร้างสัญญาณควบคุมขึ้นมาใหม่เพื่อที่จะทำการลดในส่วนของความผิดพลาดที่เกิดขึ้นภายในระบบและทำให้เอาท์พุทของระบบเข้าสู่ค่าที่ได้ปรับตั้งเอาไว้ (Set Point) **ดังรูปที่ 2.2 ดังนี้**



รูปที่ 2.2 แสดงบล็อกไดอะแกรมของการควบคุมแบบวงปิด

2.1.1 ทฤษฎีการควบคุมแบบวงปิด โดยตัวควบคุมพีไอดี

ระบบควบคุมแบบวงปิด สัญญาณค่าความคลาดเคลื่อนซึ่งเป็นค่าสัญญาณแตกต่างระหว่างสัญญาณทางด้านอินพุตกับสัญญาณป้อนกลับจะถูกป้อนกลับไปยังตัวควบคุม เพื่อที่จะลดค่าความคลาดเคลื่อนของระบบ และทำให้เอาต์พุตของระบบมีค่าตามที่ต้องการ สัญญาณป้อนกลับนี้อาจเป็นสัญญาณเอาต์พุต หรือเป็นสัญญาณที่เป็นฟังก์ชั่นของสัญญาณเอาต์พุต โดยปกติแล้ว ระบบควบคุมแบบวงปิด จะประกอบไปด้วยส่วนประกอบต่างๆ ดังต่อไปนี้

1. ตัวควบคุม หมายถึง เครื่องมือหรือ อุปกรณ์ที่ใช้ในการสร้างสัญญาณที่ต้องการควบคุมให้ได้ผลตอบสนองตามต้องการ เช่นตัวควบคุมอาจเป็นแบบเปิด/ปิด หรือแบบ PID เป็นต้น
2. อุปกรณ์วัด หมายถึง อุปกรณ์ที่ใช้สำหรับการวัดค่าต่างๆหรือใช้ในการแปลงสัญญาณ (Converter) ที่มีเอาต์พุตตามสัญญาณมาตรฐาน โดยอุปกรณ์แบบนี้ ได้แก่ เซนเซอร์ หรือทรานสดิวเซอร์เป็นต้น
3. อุปกรณ์ปรับกระบวนการ หมายถึง อุปกรณ์ที่ทำหน้าที่ปรับสภาวะการควบคุมของกระบวนการด้วยการเปลี่ยนแปลงสัญญาณตามค่าสัญญาณควบคุม อุปกรณ์ในประเภทนี้ได้แก่ Control Valve เป็นต้น

4. กระบวนการ หมายถึง กระบวนการทางฟิสิกส์ที่เราต้องการควบคุม ให้มีสภาวะตามต้องการ ขณะที่สภาวะการทำงาน หรือสภาพแวดล้อมอาจจะหยุดนิ่งหรือว่า มีการเปลี่ยนแปลงอยู่ตลอดเวลา เช่น กระบวนการวัดอุณหภูมิ, ความดัน, อัตราการไหล เป็นต้น
5. สัญญาณมาตรฐาน (Standard Signal) ในการที่ต้องการเชื่อมอุปกรณ์ในระบบควบคุมอัตโนมัติ ให้ทำงานได้ตามความต้องการ เช่น สัญญาณลม จะต้องมีค่าสัญญาณมาตรฐาน คือ 0.2-1.0 Kg/cm^2 (3-15 psi) สัญญาณไฟฟ้า มีค่าสัญญาณมาตรฐานในรูปของแรงดันไฟฟ้า คือ 1-5 VDC หรือ 4-20 mA ในรูปของกระแสไฟฟ้า เป็นต้น

การควบคุมกระบวนการทางอุตสาหกรรมในสมัยปัจจุบันนิยมใช้ตัวควบคุมแบบพีไอดี เนื่องจากมีโครงสร้างการทำงานที่ไม่ซับซ้อนและสามารถเข้าใจได้ง่าย โดยการใช้งานตัวควบคุมพีไอดีเพื่อการควบคุมนี้ขึ้นอยู่กับการปรับค่าพารามิเตอร์ของตัวควบคุมพีไอดีให้มีความเหมาะสมเพื่อให้ได้ผลตอบสนองของระบบให้เป็นไปตามที่ต้องการ โดยจะทำการอธิบายถึงการควบคุมแบบพีไอดีในหัวข้อต่อไป

2.1.2 กริยาการควบคุมแบบป้อนกลับโดยตัวตบคุมพีไอดี

2.1.2.1 กริยาการควบคุมแบบเปิด-ปิด (ON-OFF)

กริยาการควบคุมแบบเปิด-ปิด เป็นกริยาการควบคุมที่ง่ายที่สุดและนิยมใช้ในการควบคุมกระบวนการที่ไม่ต้องการความเที่ยงตรงสูง โดยการควบคุมจะทำงานเพียง 2 สถานะ คือ เปิด (100%) และปิด (0%)

2.1.2.2 กริยาการควบคุมแบบสัดส่วน (P หรือ Proportional)

กริยาการควบคุมแบบสัดส่วนนั้น หมายความว่า ค่าเอาท์พุทของตัวควบคุมจะแปรผันตรงกับค่าความคลาดเคลื่อน กล่าวคือ ถ้าค่าความคลาดเคลื่อนมีค่ามากขึ้น ค่าเอาท์พุทของตัวควบคุมก็จะมีค่ามากขึ้นตาม และถ้าค่าความคลาดเคลื่อนมีค่าน้อยลง ค่าเอาท์พุทของตัวควบคุมก็จะมีค่าน้อยลงตามกริยาการควบคุมแบบสัดส่วน โดยกริยาการควบคุมแบบสัดส่วนสามารถเขียนได้ดังสมการต่อไปนี้

$$mp(t) = Kpe(t) + \overline{m} \qquad (2.1)$$

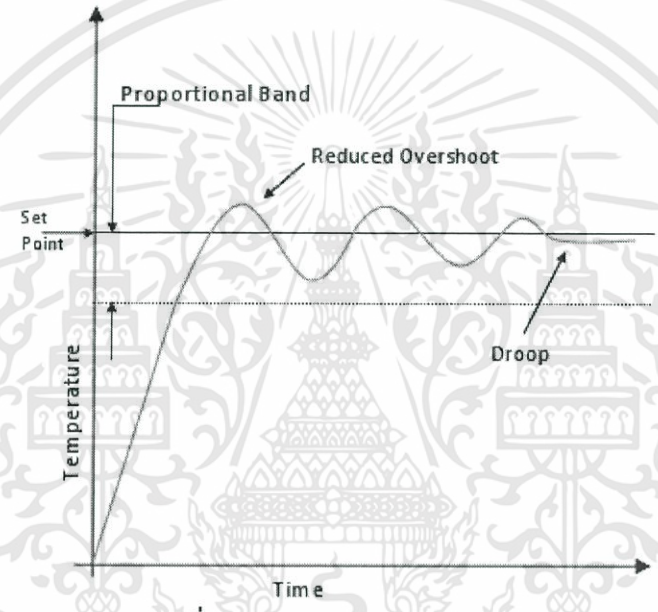
โดย

$mp(t)$ = ค่าเอาท์พุทของตัวควบคุมแบบ Proportional

Kp = อัตราขยายของตัวควบคุมแบบ Proportional

$\overline{m}$ = ค่าเอาท์พุทของตัวควบคุมที่ค่าความคลาดเคลื่อนเท่ากับศูนย์

ซึ่งข้อเสียของกริยาการควบคุมแบบสัดส่วนนั้นก็คือ ไม่สามารถกำจัดค่าออฟเซ็ตได้ และคุณสมบัติของกริยาการควบคุมแบบสัดส่วน จะแสดงดังรูปที่ 2.3 ต่อไปนี้



รูปที่ 2.3 แสดงคุณสมบัติกริยาการควบคุมแบบ P

2.1.2.3 กริยาการควบคุมแบบ I (Integral)

กริยาการควบคุมแบบ I หรือเรียกอีกอย่างหนึ่งว่า การควบคุมแบบรีเซ็ต (Reset Control) โดยค่าเอาท์พุทของตัวควบคุม จะสามารถหาได้จากค่าของพื้นที่ทั้งหมดใต้กราฟของค่าความคลาดเคลื่อนต่อเวลา นำมาคูณกับค่าคงที่ ที่เรียกว่า อัตราขยายของตัวควบคุมแบบ I (Integral Gain) ดังสมการต่อไปนี้

$$m_I(t) = K_I \int_0^t e(t)dt + \bar{m}_I(0) \qquad (2.2)$$

โดย

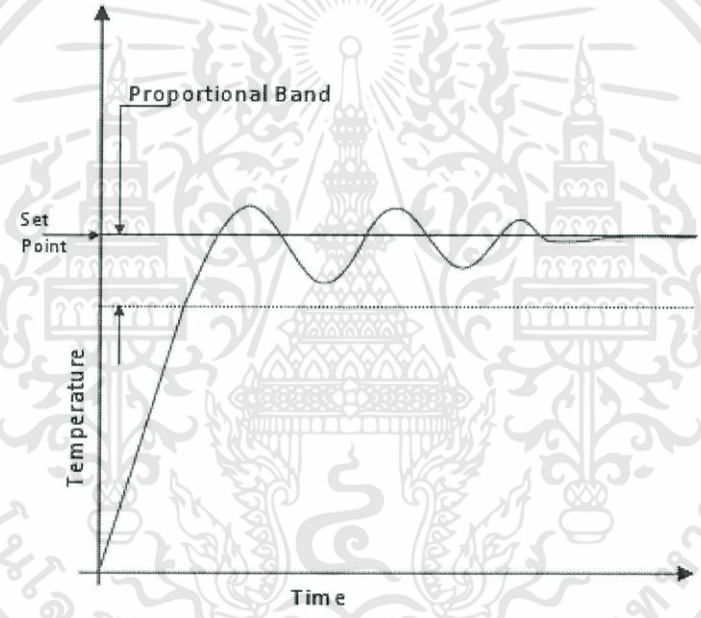
$m_I(t)$ = ค่าเอาท์พุทของตัวควบคุมแบบ Integral

K_I = อัตราขยายของตัวควบคุมแบบ Integral

$\int_0^t e(t)dt$ = พื้นที่ทั้งหมดของค่าความคลาดเคลื่อน

$\overline{m}_{I(0)}$ = ค่าเอาท์พุทของตัวควบคุมที่เวลา t เท่ากับศูนย์

ผลของกริยาการควบคุมแบบ I นี้จะทำให้ไม่เกิดออฟเซ็ทขึ้นในระบบ และลดค่าพุ่งเกิน (Overshoot) ของระบบลงได้ แต่ถ้ากริยาการควบคุมมีค่าสูงเกินไป จะทำให้ผลตอบสนองของกระบวนการช้าลง ผลตอบสนองของกริยาการควบคุมแบบ I จะแสดงดังรูปที่ 2.4 ต่อไปนี้



รูปที่ 2.4 แสดงคุณสมบัติกริยาการควบคุมแบบ I

2.1.2.4 กริยาการควบคุมแบบ D (Derivative)

กริยาการควบคุมแบบ D หรืออีกชื่อหนึ่งว่า การควบคุมแบบอัตราส่วน (Rate Action) โดยสัญญาณเอาท์พุทของตัวควบคุมจะขึ้นอยู่กับอัตราการเปลี่ยนแปลงของค่าความคลาดเคลื่อนต่อเวลา จะเห็นว่าค่าความคลาดเคลื่อนนี้มีโอกาสเป็นศูนย์ได้ และค่าเอาท์พุทก็สามารถเปลี่ยนแปลงให้มีค่าสูงขึ้น เมื่อความคลาดเคลื่อนเปลี่ยนแปลง ซึ่งเราจะเรียกการกระทำดังกล่าวว่า อัตราการกระทำ (Rate Action) ดังสมการต่อไปนี้

$$m_D(t) = K_D \frac{de(t)}{dt} \quad (2.3)$$

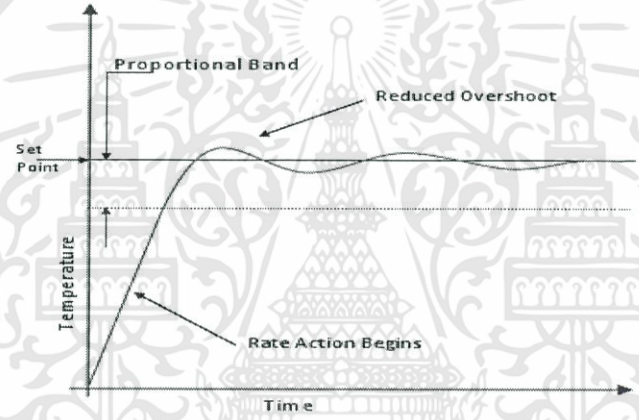
โดย

$m_D(t)$ คือ ค่าเอาท์พุทของตัวควบคุมแบบ D

K_D คือ อัตราขยายของตัวควบคุมแบบ D

$\frac{de(t)}{dt}$ คือ ค่าความคลาดเคลื่อนของเวลา

โดยคุณสมบัติของกริยาการควบคุมแบบ D จะแสดงดังรูปที่ 2.5 ต่อไปนี้



รูปที่ 2.5 แสดงคุณสมบัติกริยาการควบคุมแบบ D

2.1.2.5 กริยาการควบคุมแบบ PI (Proportional-Integral)

ตามที่กล่าวมาแล้วว่ากริยาควบคุมแบบสัดส่วนนั้น มีข้อเสียตรงที่จะมีค่าออฟเซ็ทเกิดขึ้น ซึ่งเราสามารถกำจัดค่าออฟเซ็ทที่เกิดขึ้นได้โดยการเพิ่มกริยาควบคุมแบบปริพันธ์ (I) เข้าไป โดยจะสามารถแสดงในรูปของสมการได้ดังนี้

$$m_{PI}(t) = \overline{m} + K_p e(t) + K_p K_I \int_0^t e(t)dt \quad (2.4)$$

หรือ

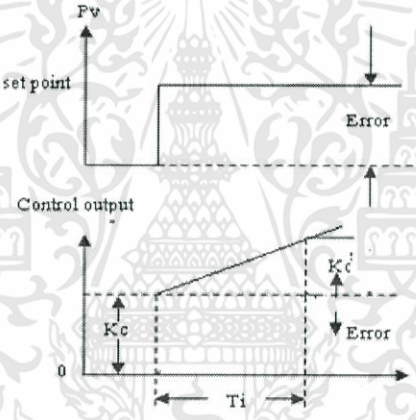
$$m_{PI}(t) = \overline{m} + K_p e(t) + \frac{K_c}{T_i}\int_o^t e(t)dt \qquad (2.5)$$

โดย

$K_c = K_p$ และ $K_I = \frac{1}{T_i}$

T_i คือ ค่าเวลา Integral

โดยคุณสมบัติของกริยาการควบคุมแบบ PI จะแสดงดังรูปที่ 2.6 ต่อไปนี้



รูปที่ 2.6 แสดงคุณสมบัติกริยาการควบคุมแบบ PI ชนิดกระทำตรง (Direct Action PI)

2.1.2.6 กริยาการควบคุมแบบ PD (Proportional-Derivative)

การประยุกต์ใช้กริยาการควบคุมแบบสัดส่วน ร่วมกับกริยาการควบคุมแบบ D นั้นมีจุดประสงค์เพื่อทำให้ผลตอบสนองของระบบรวดเร็วขึ้น แต่จะไม่มีผลโดยตรงต่อผลตอบสนองของระบบที่สภาวะคงที่ ซึ่งกริยาการควบคุมแบบ PD นั้น ทางด้านเอาท์พุทจะสามารถแสดงในรูปของสมการได้ดังนี้

$$m_{PD}(t) = \overline{m} + K_P e(t) + K_P K_D \frac{de(t)}{dt} \qquad (2.6)$$

หรือ

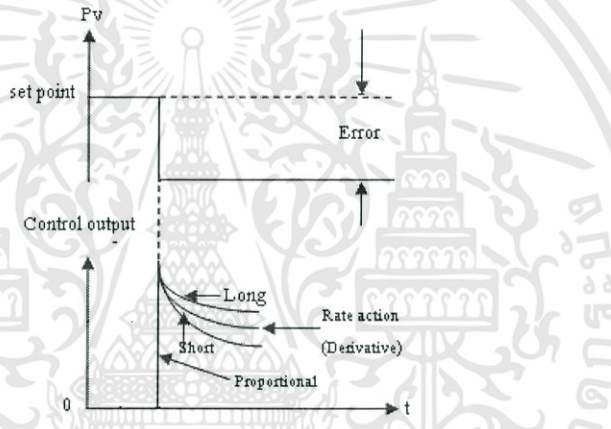
$$m_{PD}(t) = \overline{m} + K_P e(t) + K_c T_d \frac{de(t)}{dt} \quad (2.7)$$

โดย

$K_D = T_d$ และ

T_d คือ ค่าเวลา Derivative

โดยคุณสมบัติของกริยาการควบคุมแบบ PD จะแสดงดังรูปที่ 2.7 ต่อไปนี้



รูปที่ 2.7 แสดงคุณสมบัติกริยาการควบคุมแบบ PD

2.1.2.7 กริยาการควบคุมแบบ PID (Proportional-Integral-Derivative)

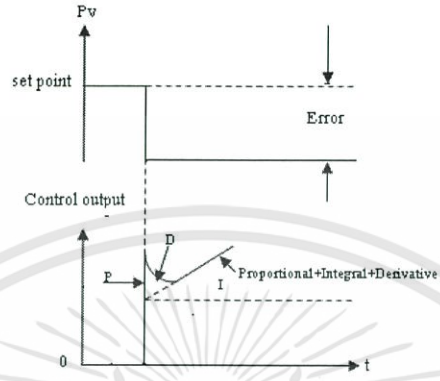
การควบคุมแบบ PID นั้น มีจุดประสงค์เพื่อให้ผลตอบสนองของระบบควบคุมมีสมรรถนะเป็นไปตามต้องการ โดยการที่จะทำให้ผลตอบสนองเป็นไปตามต้องการจึงต้องมีการใช้กริยาการควบคุมทั้ง 3 แบบ ซึ่งก็คือ กริยาการควบคุมแบบ P, I และแบบ D ร่วมกัน ซึ่งจะทำให้ได้กริยาการควบคุมแบบ PID ซึ่งจะสามารถแสดงในรูปของสมการได้ดังนี้

$$m_{PID}(t) = \overline{m} + Kpe(t) + KpK_I \int_0^t e(t)dt + KpK_D \frac{de(t)}{dt} \quad (2.8)$$

หรือ

$$m_{PID}(t) = \overline{m} + K_c e(t) + \frac{K_c}{T_i}\int_0^t e(t)dt + K_c T_d \frac{de(t)}{dt} \qquad (2.9)$$

โดยคุณสมบัติของกริยาการควบคุมแบบ PID จะแสดงดังรูปที่ 2.8 ต่อไปนี้

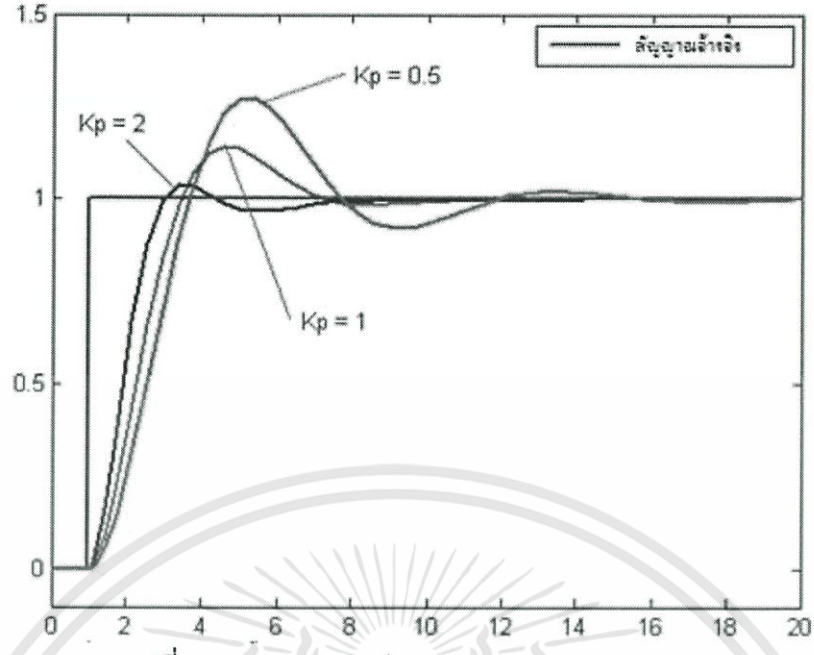


รูปที่ 2.8 แสดงคุณสมบัติกริยาการควบคุมแบบ PID ชนิดกระทำตรง (Direct Action PID)

2.1.3 ผลของพารามิเตอร์ของตัวควบคุมแบบ PID กับกระบวนการ

2.1.3.1 ผลของ PB (Proportional Band) กับกระบวนการ

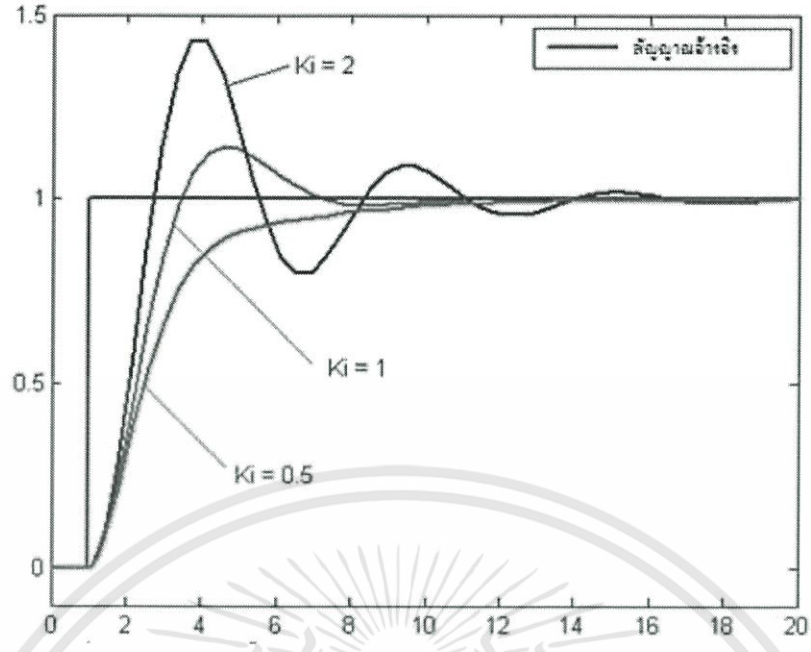
ผลของ PB (Proportional Band) กับกระบวนการจะเป็นสัดส่วนผกผันกับอัตราขยาย (Gain) ของกระบวนการ โดยที่ ความสัมพันธ์ระหว่างค่าอัตราขยายกับค่า PB สามารถแสดงได้ด้วยสมการทางคณิตศาสตร์ดังนี้ Gain = 100% / PB สำหรับผลของ P โดยระบบควบคุมแบบ P นั้นค่า PB จะมีผลกับขนาดของผลตอบสนองรวมไปถึงเรื่องความไว (Sensitivity) และออฟเซ็ท (Offset) หรือค่าผิดพลาดในสภาวะคงตัว (Steady state) ผลของ PB ที่มากเกินไปจะมีผลทำให้เกิดค่าผิดพลาด (Error) ในสภาวะคงตัวที่มากขึ้นและจะทำให้ความไวของผลตอบสนองที่ลดลง ในทางกลับกัน ถ้าค่าของ PB ที่น้อยเกินไปจะทำให้กระบวนการเกิดการแกว่งขึ้นหรือเข้าใกล้จุดไม่เสถียร (Unstable) มากขึ้น รูปที่ 2.9 จะแสดงถึงผลของการเพิ่มหรือลดค่าของ PB ต่อกระบวนการกำลังสอง ที่มีค่าของ Integral Time และ Derivative Time ที่เท่ากัน



รูปที่ 2.9 ผลของ PB ต่อระบบการควบคุมแบบ PID

2.1.3.2 **ผลของ Integral Time กับกระบวนการ**

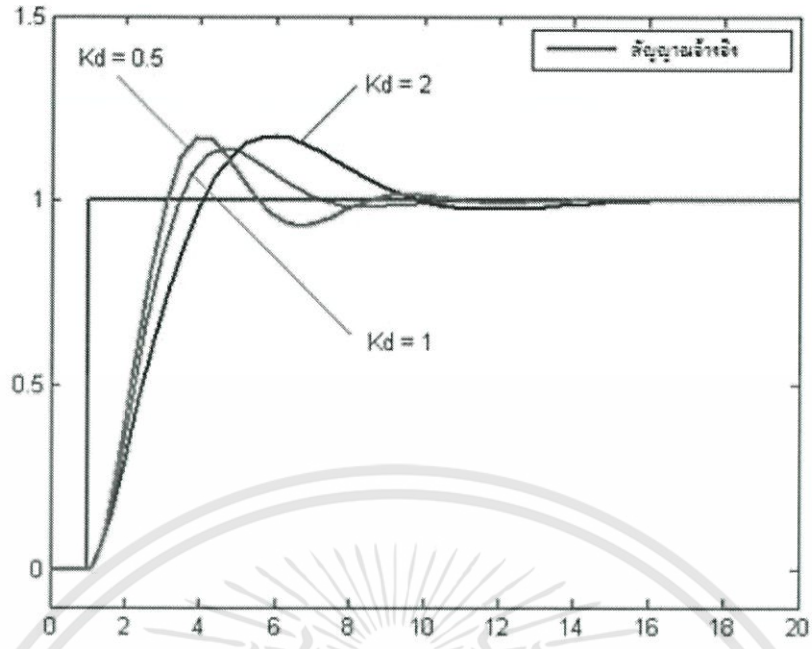
สำหรับ Integral Action นั้นจะมีผลตอบสนองของค่าสะสมแต่จะไม่มีผลกับผลตอบสนองในลักษณะทันทีทันใด การลดค่า Integral Time จะมีผลทำให้ ออฟเซ็ทหรือค่าผิดพลาดในสภาวะคงตัวของกระบวนการมีค่าลดลงเนื่องจาก Integral Action จะเป็นสัดส่วนโดยตรงกับขนาดของความผิดพลาดสะสมของผลตอบสนอง แต่ถ้าค่า Integral Time ที่ตั้งให้กระบวนการมีค่าน้อยเกินไป (Integral Action มากเกินไป) จะทำให้ผลตอบสนองของกระบวนการเกิดการแกว่งขึ้น (ค่าพุ่งเกินมากเกินไป) เวลาที่ใช้ในการเข้าสู่สภาวะคงตัวก็จะนานขึ้น ในทำนองกลับกันนั้นการเพิ่มค่า Integral Time จะมีผลทำให้เวลาที่ใช้ในการเข้าสู่สภาวะคงตัวเร็วขึ้น คือจะเพิ่มค่าผิดพลาดในสถานะคงตัวของกระบวนการ ถ้าผลตอบสนองของกระบวนการนั้นเป็นผลตอบสนองที่มีค่าผิดพลาดในสถานะคงตัวอยู่ ในขณะเดียวกันนั้น ก็จะเป็นการลดการแกว่งหรือค่าพุ่งเกิน (Overshoot) ของการตอบสนองของระบบในกรณีที่กระบวนการนั้นๆมีค่า Integral Time เดิมน้อยเกินไป โดยจะสามารถแสดงผลของการเพิ่มหรือลดค่า Integral Time ในการควบคุมแบบ PID ที่มีค่าของ Proportional Band และ Derivative Time ที่เท่ากัน ดังรูปที่ 2.10



รูปที่ 2.10 ผลของ Integral Time ต่อระบบการควบคุมแบบ PID

2.1.3.3 **ผลของ Derivative Time กับผลตอบสนองต่อระบบ**

ผลของ Derivative Action จะเป็นสัดส่วนโดยตรงกับค่าอัตราการเปลี่ยนแปลงของอินพุท ซึ่งมีผลทำให้ระบบตอบสนองการเปลี่ยนแปลงได้เร็วขึ้น สำหรับตัวควบคุมแบบ PID นั้น ค่าของ Derivative Action จะมีผลกับการเปลี่ยนแปลงของตัวแปรที่เกี่ยวกับระบบ โดยจะทำให้ระบบมีการตอบสนองต่อการเปลี่ยนแปลงที่เร็วขึ้นแต่ Derivative Action จะไม่มีผลต่อการเปลี่ยนแปลงของ Set Point แต่อย่างใด แต่อย่างไรก็ตาม Derivative Action จะมีผลเสียในการควบคุมระบบที่มีสัญญาณรบกวนมากเนื่องจาก Derivative Action จะไวต่อการเปลี่ยนแปลงสัญญาณที่เข้ามาอย่างมาก ดังนั้น Derivative Control อาจจะมีผลทำให้ระบบที่มีสัญญาณรบกวนมากนั้นไม่มีเสถียรภาพ (Unstable) โดยจะสามารถแสดงผลของการเพิ่มหรือลดค่า Integral Time ในการควบคุมแบบ PID ที่มีค่าของ Proportional Band และ Integral Time ที่เท่ากัน ดังรูปที่ 2.10



รูปที่ 2.11 ผลของ Derivative Time ต่อระบบการควบคุมแบบ PID

2.1.4 ทฤษฎี Auto-tuning PID

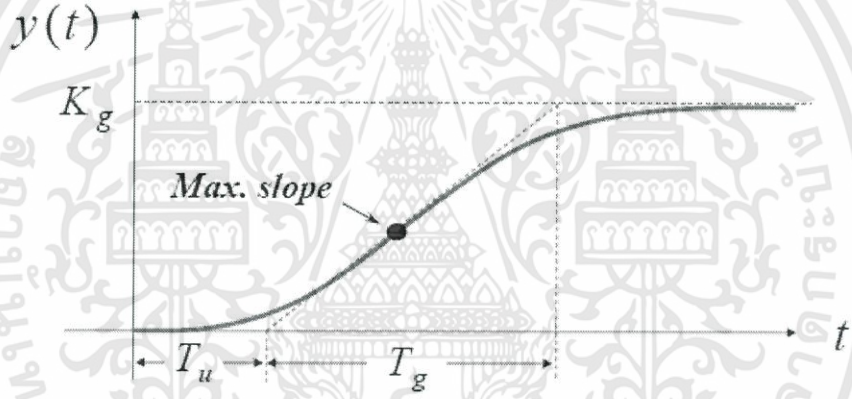
เนื่องจากในปัจจุบันนี้ ตัวควบคุมแบบ PID ได้ถูกใช้งานกันอย่างกว้างขวาง แต่ยังคงมีปัญหาในเรื่องของความยุ่งยากในการปรับค่าพารามิเตอร์ ซึ่งส่วนใหญ่ยังต้องใช้วิธีการลองผิดลองถูก (Trial and Error) อยู่ ทำให้เกิดความยุ่งยากและเสียเวลาในการปรับแต่งค่าพารามิเตอร์ โดยเฉพาะอย่างยิ่งถ้าผู้ควบคุม ไม่มีความชำนาญและประสบการณ์ในการปรับแต่งค่าพารามิเตอร์ที่ไม่มากพอ โดยการใช้ Auto-tuning PID ก็คือการใช้โปรแกรม เพื่อทำการหาค่าพารามิเตอร์ของตัวควบคุมโดยอัตโนมัติ ซึ่งสามารถลดเวลาที่ต้องใช้ในการปรับแต่งค่าพารามิเตอร์ลงได้ในระดับหนึ่ง ซึ่งผู้ใช้งานหรือผู้ควบคุมเพียงแค่ทำการป้อนค่าเป้าหมาย (Set Point) เข้าไป แล้วทำการสั่งการเพื่อให้ระบบ Auto-tuning ทำงาน

2.1.4.1 การหาค่าพารามิเตอร์อัตโนมัติของตัวควบคุมโดยวิธี Ziegler-Nichols

การหาค่าพีไอดีพารามิเตอร์แบบอัตโนมัติโดยใช้วิธีของ Ziegler-Nichols จะมีจุดมุ่งหมายที่จะทำให้ผลตอบสนองเวลาของระบบต่ออินพุตแบบ Unit Step มีค่าพุ่งเกินสูงสุดไม่เกิน 25% ซึ่งวิธีการหาค่าแบบ Ziegler-Nichols จะมีอยู่ 2 วิธี ได้แก่

1. วิธีการ Open Loop Tuning (Step Test)

วิธีนี้จะทำการหาค่าพารามิเตอร์ของตัวควบคุมพีไอดี จากผลตอบสนองของกระบวนการที่ถูกควบคุมแบบ closed-loop ต่ออินพุตแบบ unit step โดยที่กระบวนการในกรณีนี้จะไม่มีโพลที่จุดออริจินและไม่มีตัวควบคุมต่อร่วมอยู่กับกระบวนการ ซึ่งวิธีนี้จะมีความคล้ายคลึงกับวิธีการควบคุมแบบวงเปิดโดยวิธี Ziegler-Nichols โดยผลตอบสนองที่ได้จะต้องมีลักษณะเป็นรูปตัว S เท่านั้น แต่ถ้าผลตอบสนองไม่เป็นรูปตัว S แสดงว่าไม่สามารถใช้การได้ เนื่องจากผลการตอบสนองรูปตัว S จะถูกอธิบายคุณลักษณะของกระบวนการด้วยพารามิเตอร์ 3 ค่า คือ T_d (Delay Time), T (Time Constant) และ K (Process Gain) ดังจะแสดงในรูปที่ 2.12 และตารางที่ 2.1



รูปที่ 2.12 แสดงผลตอบสนองรูปตัว S เมื่อใช้วิธีแบบ Open Loop

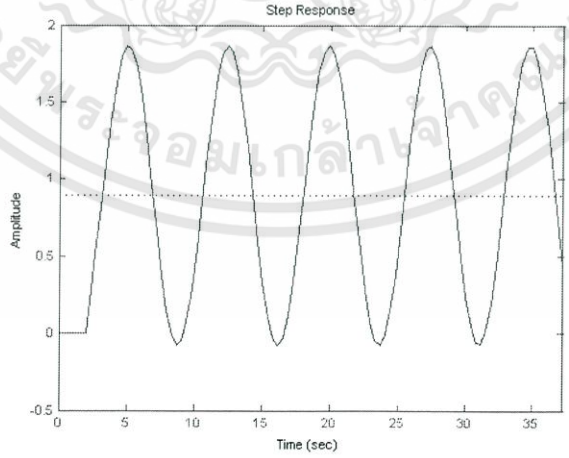
ตารางที่ 2.1 แสดงสูตรที่ใช้ในการหาค่าพารามิเตอร์ของตัวควบคุม PID โดยใช้วิธี Ziegler-Nichols แบบ Open Loop

Controller	K_p	K_i	K_d
P	1/a	0	0
PI	0.9/a	3L	0
PID	1.2/a	2L	L/2

โดยโมเดลของทั้งสามพารามิเตอร์รวมกันนี้จะเรียกว่า “Easy-Tune algorithm” โดยเราสามารถรับรู้เกี่ยวกับพฤติกรรมของกระบวนการ ได้โดยการยอมให้วิธีการนี้ทำการคาดการณ์ปฏิกิริยาของกระบวนการในความพยายามที่จะแก้ไขสิ่งใดๆก็ตาม ที่ไม่ใช่เพียงแค่ Step input และทำการคำนวณค่าพารามิเตอร์ออกมาเพื่อให้ตัวควบคุมสามารถทำงานสัมพันธ์กับกระบวนการได้

2. วิธีการ Closed Loop Tuning (Ultimate Gain)

วิธีนี้ถูกคิดค้นขึ้นใน ค.ศ. 1984 วิธีนี้จะหาค่าพารามิเตอร์ของตัวควบคุม PID จากผลตอบสนองเวลาของระบบหรือกระบวนการที่ถูกควบคุมด้วยตัวควบคุมแบบ P ต่ออินพุตแบบ Unit Step โดยวิธีนี้เป็นการปรับปรุงและต่อยอดมาจากวิธีการปรับค่าพีไอดีพารามิเตอร์แบบวงปิดของ Ziegler-Nichols ซึ่งวิธีการนี้จำเป็นที่จะต้องมีตัวควบคุมแบบ P ต่อร่วมอยู่เพื่อให้สามารถปรับค่าอัตราขยายได้ และการหา ค่าพารามิเตอร์โดยวิธีนี้ไม่จำเป็นต้องทราบค่าฟังก์ชันถ่ายโอน (Transfer Function) ของกระบวนการ โดยในการหาค่าพารามิเตอร์ของกระบวนการจะสามารถทำได้โดยการปรับค่าของ K_p ไปเรื่อยๆ จนผลตอบสนองของกระบวนการเกิดการแกว่งอย่างต่อเนื่อง และเรียกอัตราขยายที่ทำให้เกิดการแกว่งนี้ว่า Ultimate Gain (K_u) และเรียกค่าคาบของการแกว่งว่า Ultimate Period (T_u) จากนั้นนำค่าตัวแปรดังกล่าวไปคำนวณหาค่าพารามิเตอร์ของตัวควบคุมดังรูปที่ 2.13 และตารางที่ 2.2



รูปที่ 2.13 แสดงผลตอบสนองที่เกิดการแกว่งอย่างต่อเนื่อง

ตารางที่ 2.2 แสดงสูตรที่ใช้ในการหาค่าพารามิเตอร์ของตัวควบคุม PID โดยใช้วิธี Ziegler-Nichols แบบ Closed-Loop (Ultimate Gain)

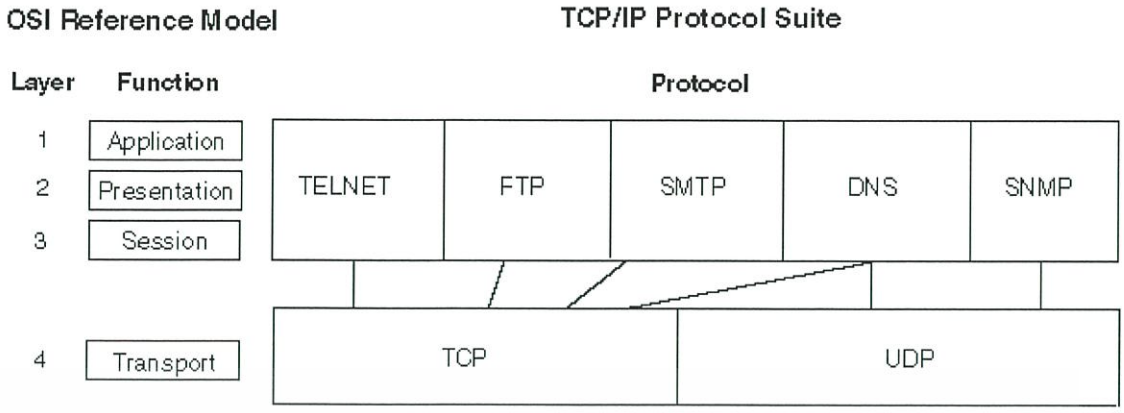
Controller	K_p	K_i	K_d
P	$0.5\ K_u$	0	0
PI	$0.4\ K_u$	$0.8\ T_u$	0
PID	$0.6\ K_u$	$0.5\ T_u$	$0.125\ T_u$

2.2 ทฤษฎีการสื่อสาร

2.2.1 ทฤษฎีการสื่อสารแบบ UDP

การสื่อสารแบบ UDP ย่อมาจาก User Datagram Protocol เป็น Protocol ที่มีลักษณะการเชื่อมต่อแบบ connectionless คือ ข้อมูลจะถูกแบ่งเป็นส่วนๆ ตามที่อยู่ปลายทาง แล้วทำการส่งผ่านตัวกลางไปยังปลายทาง อาจจะใช้เส้นทางคนละเส้นทางกันก็ได้ รวมทั้งข้อมูลแต่ละชิ้นอาจจะถึงก่อนหลังแตกต่างกันไปได้ด้วย ทำให้การเริ่มต้นส่งทำได้รวดเร็ว ทำให้ไม่ต้องเสียเวลาสร้าง Connection โดยการส่งข้อมูลแบบนี้จะไม่มีการรับประกันความถูกต้องของข้อมูลเนื่องจากโปรโตคอลยูดีพีไม่มีสัญญาณตรวจทานข้อมูล (acknowledgement) ในการส่งข้อมูลแต่ละครั้งและไม่มีการส่งข้อมูลใหม่อีกในกรณีที่เกิดความผิดพลาดของการส่งข้อมูล เมื่อเป็นเช่นนี้แอพพลิเคชันหรือโปรเซสใดที่ต้องอาศัยโปรโตคอล UDP ในการส่งผ่านข้อมูลก็อาจจะต้องสร้างกระบวนการตรวจสอบข้อมูลขึ้นมาเอง โดยการสื่อสารแบบนี้มักใช้กับข้อมูลประเภทมัลติมีเดีย โดย packet ของ UDP จะเรียกว่า User Datagram โดยจะมีขนาดของ Header 8 bytes และประกอบไปด้วยฟิลด์ต่างๆ โดยแต่ละฟิลด์ต่างๆจะมีขนาด 16 บิต ดังนี้

1) Source port number
2) Destination port number
3) Length
4) Checksum



รูปที่ 2.14 แสดงถึงการเปรียบเทียบระหว่างโปรโตคอล UDP กับ TCP

ขั้นตอนกลไกการทำงานโดยใช้โปรโตคอล UDP

ในขั้นตอนของ Process Layer เมื่อโปรแกรมควบคุมอุปกรณ์เครือข่ายเช่น โปรแกรม Network management ต้องการส่งข้อมูลไปยังอุปกรณ์ที่ต้องการ แอพพลิเคชันนั้นจะติดต่อผ่าน SNMP Protocol ในชั้น Process Layer

โดย SNMP Protocol จะมีหน้าที่ติดต่อกับโปรโคคอล UDP ในชั้นถัดไป เพื่อขอติดต่อผ่านพอร์ตที่กำหนดหลังจากนั้น จะทำหน้าที่เตรียมข้อมูลที่จะส่ง รวมทั้งที่อยู่ปลายทางและทำการส่งผ่านข้อมูลโปรโตคอล UDP ที่อยู่ใน Transport Layer

ส่วนโปรโตคอล UDP ทำหน้าที่ผนึกข้อมูลหรือ Datagram นั้น ไปกับโปรโตคอลไอพีในชั้นถัดไป เพื่อส่งข้อมูลออกจากเครื่อง

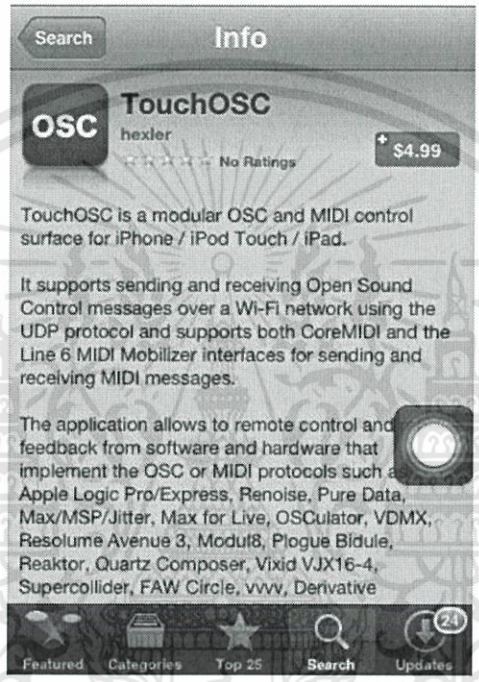
2.2.2 ทฤษฎีการสื่อสารแบบ TCP

โปรโตคอล TCP (Transmission Control Protocol) เป็นโปรโตคอลที่มีการรับส่งข้อมูลแบบ Stream oriented protocol หมายความว่า การรับส่งข้อมูลจะไม่คำนึงถึงปริมาณข้อมูลที่จะส่งไป แต่จะทำการแบ่งข้อมูลออกเป็นส่วนย่อยๆ ก่อน แล้วจึงจะส่งไปยังปลายทางอย่างต่อเนื่องเป็นลำดับข้อมูลหรือเป็น Stream นั่นเอง ในกรณีที่ข้อมูลส่วนใดส่วนหนึ่งสูญหายไป ก็จะส่งข้อมูลส่วนนั้นใหม่อีกครั้ง สำหรับปลายทางก็จะทำหน้าที่จัดเรียงส่วนของข้อมูล Datagram ใหม่ให้ต่อเนื่องกันและประกอบกับเป็นข้อมูลทั้งหมดได้ ซึ่งจะแยกข้อมูลที่ไม่ถูกต้องออก ดังนั้น Application หรือ Process ใดๆที่อาศัยการส่งผ่านข้อมูลด้วยโปรโตคอลแบบ TCP จะต้องใช้ขนาดของหน่วยความจำและ Bandwidth มากกว่าแบบ UDP

โดยการติดต่อระหว่างกันของโปรโตคอล TCP จะต้องเป็นแบบ Connection-oriented คือต้องการสร้างการติดต่อระหว่างทั้งสองด้านเสียก่อน แล้วจึงจะสามารถรับส่งข้อมูลไปได้พร้อมกัน (Full duplex) เมื่อ Application ต้องการส่งผ่านข้อมูลจะใช้ Protocol ที่เหมาะสมในชั้น Process Layer ติดต่อไปและมีการสร้างช่องส่งผ่านข้อมูลตาม port ที่ได้กำหนดไว้เพื่อทำการส่งผ่านข้อมูลไปยัง TCP Protocol ซึ่งในระหว่างการส่งข้อมูลนี้ TCP Protocol จะเพิ่มกระบวนการตรวจทานข้อมูลเพื่อตรวจสอบความถูกต้องของข้อมูล โดยการส่งเฟรมตรวจทานข้อมูล (Acknowledgement Frame) และจะทำการส่งข้อมูลให้ใหม่อีกครั้ง ถ้าเกิดความผิดพลาดขึ้นระหว่างการส่ง

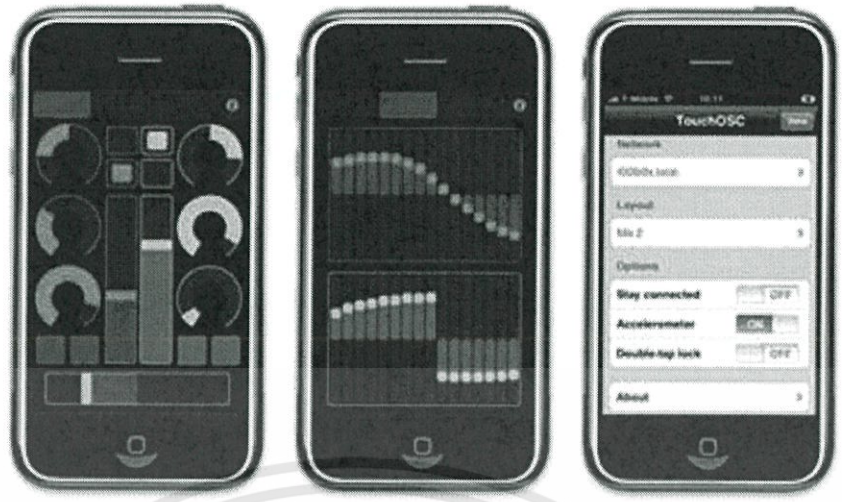
2.3 โปรแกรม TouchOSC

โปรแกรม TouchOSC (Touch Open Sound Control) เป็นโปรแกรมบนอุปกรณ์ iOS ที่ทำตัวเป็น MIDI Controller สามารถสั่งงานผ่านทาง wifi ให้โปรแกรมที่รับข้อมูล MIDI CC ควบคุมการทำงานได้



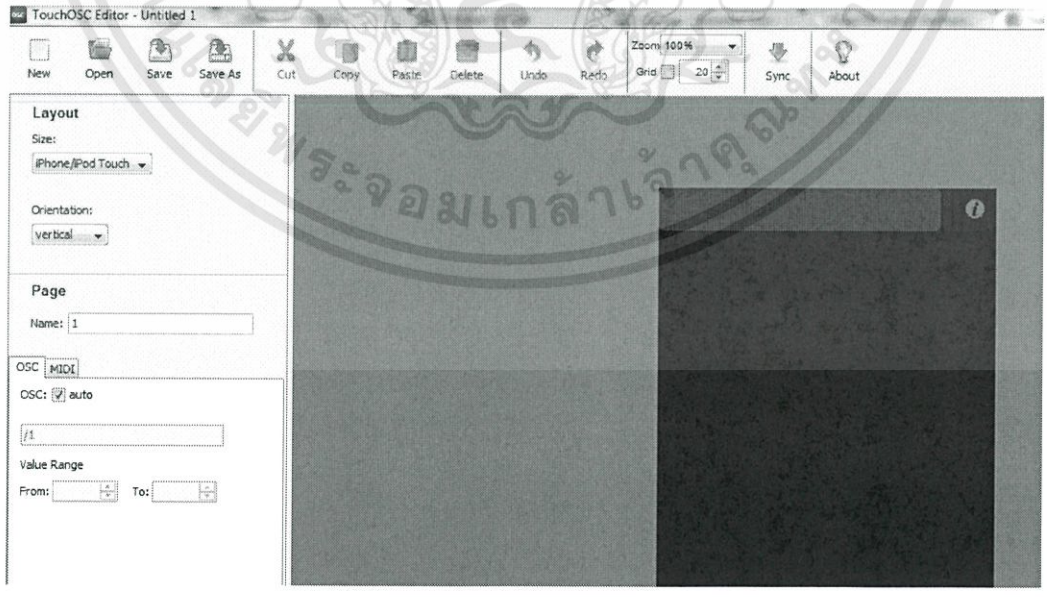
รูปที่ 2.15 แสดงถึงหน้าดาวน์โหลดโปรแกรม TouchOSC ซึ่งสามารถดาวน์โหลดได้จาก App Store

โดยการรับ-ส่งข้อมูลผ่านโปรแกรม TouchOSC นั้นจะใช้โปรโตคอลการรับ-ส่งข้อมูลแบบ UDP (User Datagram Protocol) ซึ่งไม่จำเป็นต้องทำการสร้างการติดต่อระหว่างกันก่อน โดยในโครงงานนี้ ได้มีการทำการเชื่อมต่อระหว่างโปรแกรม TouchOSC กับโปรแกรม LabVIEW ซึ่งจะได้อธิบายรายละเอียดเกี่ยวกับโปรแกรม LabVIEW ในหัวข้อต่อไป



รูปที่ 2.16 แสดงถึงหน้าตาของโปรแกรม TouchOSC

การประยุกต์ใช้จะช่วยให้สามารถทำการควบคุมอุปกรณ์จากระยะไกลได้ โดยซอฟต์แวร์และฮาร์ดแวร์ที่สามารถใช้ OSC หรือโปรโตคอล MIDI มีหลากหลายประเภท ไม่ว่าจะเป็น Apple Logic Pro/Express, Renoise, Pure Data, Max/MSP/Jitter, Max for Live, OSCulator, VDMX, Resolume Avenue 3, Modul8, Plogue Bidule, Reaktor, Quartz Composer, The Missing Link, Vixid VJX16-4, Supercollider, FAW Circle, vvvv, Derivative TouchDesigner, Isadora รวมทั้ง Traktor และ Ableton Live เป็นต้น

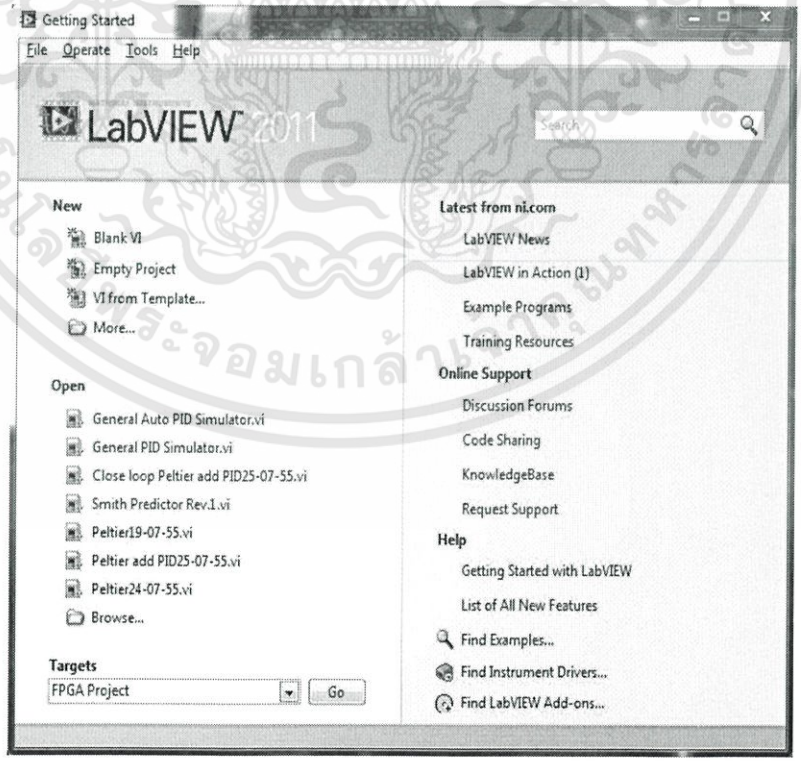


รูปที่ 2.17 แสดงหน้าตาของโปรแกรม TouchOSC Editor ที่ใช้ในการสร้าง Layout

ซึ่งเราสามารถทำการเลือกชนิดของอุปกรณ์ iOS ที่ต้องการสร้าง Layout ได้ ไม่ว่าจะเป็น iPhone/iPod หรือว่า iPad โดยเราสามารถเลือกรูปแบบของ Layout ได้ตามต้องการ ไม่ว่าจะเป็นการใส่ switch, fader หรือแม้กระทั่งหลอด led จำลองลงบนตัวโปรแกรมได้ เป็นต้น

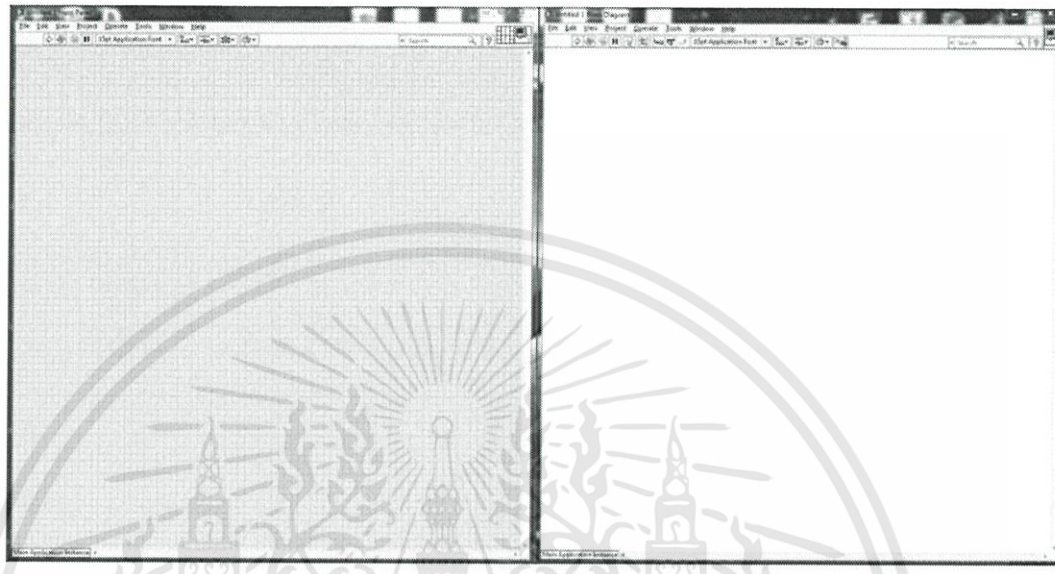
2.4 โปรแกรม LabVIEW

โปรแกรม LabVIEW (Laboratory Virtual Instrument Engineering Workbench) เป็นโปรแกรมที่ถูกสร้างขึ้นมาเพื่อนำมาใช้ในงานทางด้านการวัดและสร้าง เครื่องมือวัดเสมือนจริงในห้องปฏิบัติการทางวิศวกรรม ดังนั้นจุดประสงค์หลักของการทำงานของโปรแกรมนี้ก็คือการจัดการในด้านการวัดและเครื่องมือวัด อย่างมีประสิทธิภาพ และในตัวของโปรแกรมจะประกอบไปด้วยฟังก์ชันที่ใช้ช่วยในการวัดมากมายและจะมีประโยชน์อย่างสูงเมื่อใช้ร่วมกับเครื่องมือวัดทางวิศวกรรมต่างๆ โดยโปรแกรมนี้จะใช้ภาษา G (Graphic) ซึ่งเป็นภาษารูปภาพแทนที่จะเป็นการเขียนโปรแกรมแบบทั่วไปที่เราจะต้องทำการพิมพ์โค้ดข้อมูลต่างๆลงไป เพื่อทำการเขียนโปรแกรมขึ้นมา



รูปที่ 2.18 แสดงถึง Startup Screen ของโปรแกรม LabVIEW

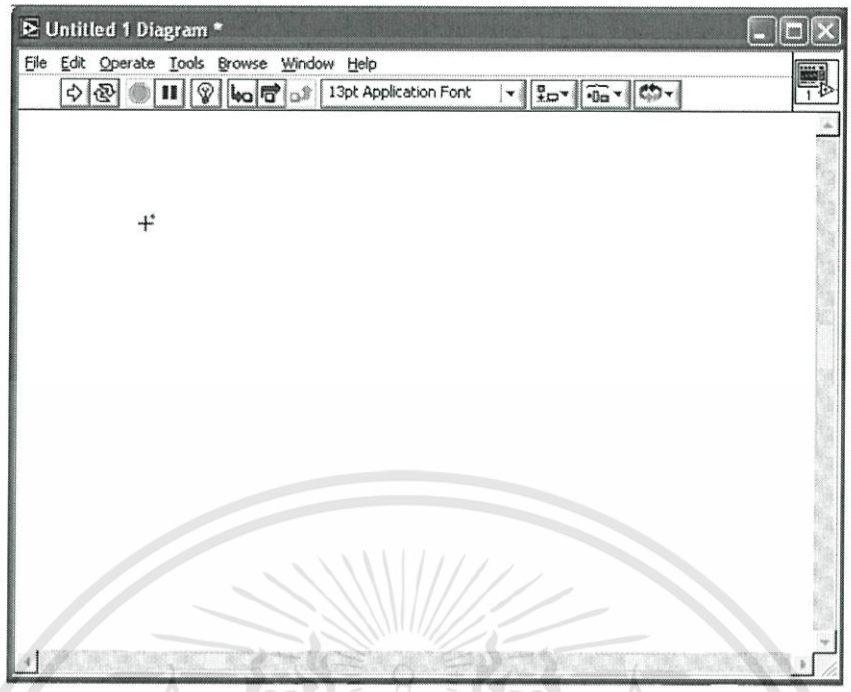
โดยในการเริ่มสร้างโปรแกรมบนโปรแกรม LabVIEW จะเริ่มต้นด้วยการสร้างจากไฟล์เปล่า โดยการคลิก Blank.vi เมื่อทำการคลิกเข้าไปแล้วจะพบกับหน้าต่าง ดังรูปที่ 2.19



รูปที่ 2.19 แสดงหน้าต่างของโปรแกรม LabVIEW ในภาพรวม

2.4.1 บล็อกไดอะแกรม

ในหน้าต่างบล็อกไดอะแกรม (Block Diagram) จะประกอบไปด้วยอุปกรณ์ที่ใช้ในการสร้างโปรแกรม โดยเปรียบเสมือนว่าเป็น Source code ของการเขียนโปรแกรมในภาษาอื่นๆ โดยการเชื่อมโยงอุปกรณ์ต่างๆ ภายใน Block Diagram นั้นสามารถทำได้ด้วยการลากสายสัญญาณ (Wire) จาก Terminal ของอุปกรณ์แต่ละตัวให้เชื่อมโยงกัน และกำหนดค่าหรือคุณสมบัติของอุปกรณ์แต่ละตัวก็จะทำให้โปรแกรมทำงานตามที่ผู้เขียนต้องการ



รูปที่ 2.20 แสดงถึงหน้าต่างของ Block Diagram ภายในโปรแกรม LabVIEW

โดยภายในบล็อกไดอะแกรม จะประกอบไปด้วยส่วนที่สำคัญ 3 ส่วน ได้แก่ Node, Terminal และ Wires ซึ่งจะมีการอธิบายหน้าที่และความหมาย ดังนี้

1. Node

เป็นตัวกระทำให้โปรแกรมทำงานตามที่ผู้เขียนต้องการ ถ้าเปรียบเทียบกับโปรแกรมทั่วไป Nodes ก็เปรียบเสมือน statement, functions, และ subroutines โหนดในโปรแกรม LabVIEW มี 3 ชนิด คือ Function, Sub VI โหนด และ Structure โหนดแบบ Function จะสร้างขึ้นมาเพื่อให้สามารถทำงานขั้นพื้นฐานได้ เช่น การบวกเลข การคูณเลข เป็นต้น

2. Terminal

เป็นทางผ่านของข้อมูลระหว่างบล็อกไดอะแกรมกับ Front Panel และระหว่าง node แต่ละ node ในบล็อกไดอะแกรม โดยตัว Terminal เปรียบได้กับค่าพารามิเตอร์และค่าคงที่ในโปรแกรมภาษาอื่น Terminal แต่ละชนิดก็มีความแตกต่างกันซึ่งในโปรแกรม LabVIEW มี Terminal อยู่ 4 ชนิดคือ Control and Indicator Terminals, node Terminal, Constants, และ Terminal พิเศษ ซึ่ง Terminal แต่ละตัวสามารถลากสายสัญญาณเพื่อเป็นทางผ่านของข้อมูล สำหรับตัวอย่างใน

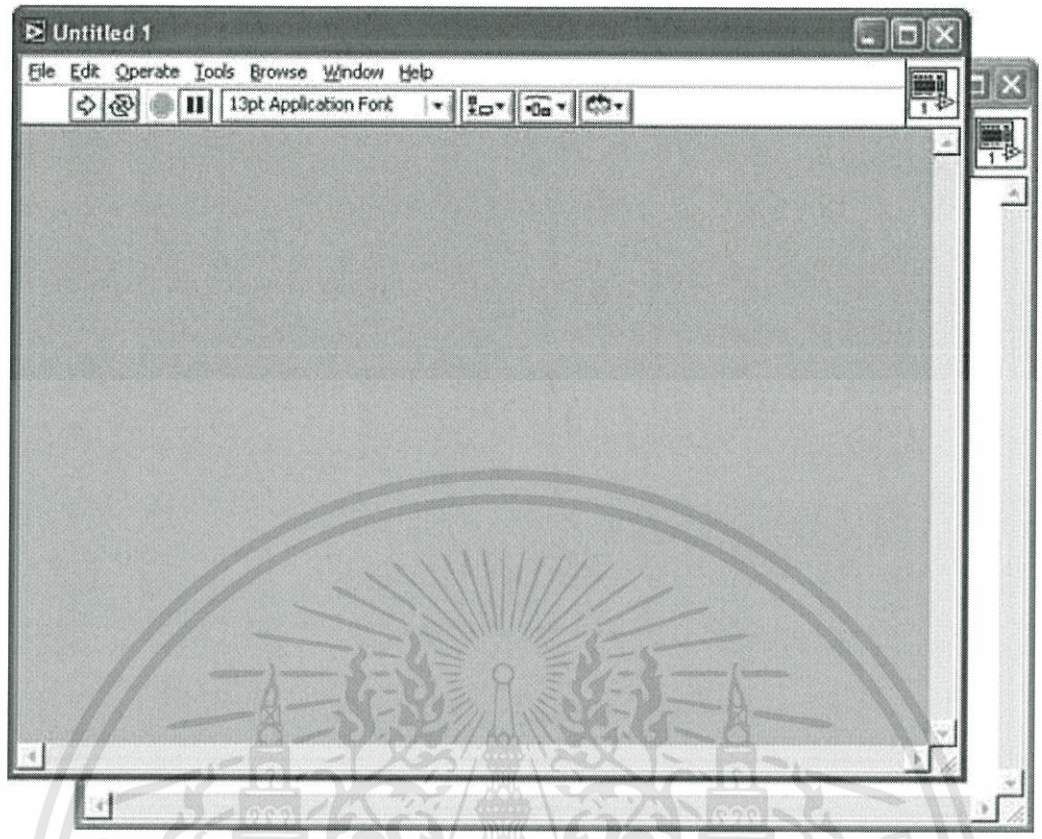
กรณีของ Control and Indicator Terminals เมื่อมีข้อมูลแบบตัวเลขผ่านเข้ายังบล็อกไดอะแกรม โดยผ่านช่องทาง Control Terminals แล้วทำการประมวลผลในหน้าต่างบล็อกไดอะแกรมเมื่อโปรแกรมทำการประมวลผลเสร็จเรียบร้อย ก็จะส่งข้อมูลจากหน้าต่างบล็อกไดอะแกรมไปยังหน้าต่าง Front Panel โดยผ่านช่องทาง Indicator Terminal

3. Wires

เป็นทางผ่านของข้อมูลระหว่าง Terminal โดย Wires นั้นเปรียบเสมือนการลากสายสัญญาณเชื่อมต่อระหว่าง อุปกรณ์ โดยการเชื่อมต่อ Wires ภายในโปรแกรม LabVIEW นั้นสามารถทำได้โดยการเลือกเครื่องมือที่ชื่อว่า connect wire ที่หน้าต่าง Tool Palette ที่มีลักษณะคล้ายกับหลอดด้าย เมาส์จะเปลี่ยนเป็นรูปหลอดด้ายจากนั้นลากไปวางตรงจุดต่อของ Terminal แรกที่ต้องการลากคลิกเมาส์หนึ่งครั้งแล้วลากไปยังจุดต่อของ Terminal ปลายทางที่ต้องการคลิกเมาส์อีกหนึ่งครั้งก็เสร็จการเชื่อมต่อสัญญาณ

2.4.2 Front Panel

หน้าต่าง Front Panel ภายในโปรแกรม LabVIEW นั้นจะเป็นส่วนที่ใช้สำหรับติดต่อกับผู้ใช้งาน ซึ่งในหน้าต่างของ front Panel จะประกอบไปด้วยอุปกรณ์อยู่ 2 ส่วน ได้แก่ อุปกรณ์สำหรับการควบคุม (Control) ยกตัวอย่างเช่น สวิทช์หรือ Knob เป็นต้น และอุปกรณ์อีกอย่างหนึ่งก็คืออุปกรณ์สำหรับแสดงผล (Indicator) เช่น LED, Graph และ Chart เป็นต้น และในส่วนของอุปกรณ์อื่นๆ ที่ไม่ใช่อุปกรณ์ควบคุมและแสดงผลจะแสดงไอคอนอยู่ในหน้าต่างบล็อกไดอะแกรม สำหรับอุปกรณ์ควบคุมในภาษา G จะหมายถึง “Input” และอุปกรณ์แสดงผลจะหมายถึง “Output” การวางอุปกรณ์ควบคุมและอุปกรณ์แสดงผลลงไปบนหน้าต่างฟร้อนท์พาเนล ทำได้โดยการเลือกอุปกรณ์ที่ต้องการบนหน้าต่าง Control Palette แล้วใช้เมาส์ลากไปวางบนหน้าต่างพาเนล ซึ่งการเรียกหน้าต่าง Control Palette ทำได้โดยการคลิ๊กเมาส์ทางด้านขวาบริเวณพื้นที่ว่างของหน้าต่างพาเนลก็จะปรากฏหน้าต่าง Control Palette ที่ประกอบไปด้วยอุปกรณ์ควบคุมและอุปกรณ์แสดงผลขึ้นมา การเรียกใช้งานหน้าต่าง Control Palette อีกวิธีหนึ่งก็คือ เลือกที่เมนู Window ของแท็บเมนูบาร์ แล้วเลือก Show Control Palette จะ ปรากฏหน้าต่างของ Control Palette ขึ้นมาให้เลือกใช้งาน เช่นเดียวกัน



รูปที่ 2.21 แสดงถึงส่วนของ Front Panel ภายในโปรแกรม LabVIEW

2.4.3 วิธีการสร้างไฟล์สกุล .vi

ในหัวข้อนี้จะกล่าวถึงขั้นตอนและวิธีการสร้างไฟล์สกุล .vi ซึ่งเป็นไฟล์ของโปรแกรม LabVIEW โดยในการสร้างไฟล์สกุล .vi นั้นเมื่อเราลากอุปกรณ์ควบคุมและแสดงผลมาวางบนหน้าต่างของ Front Panel ในเวลาเดียวกันนั้นก็จะปรากฏ Terminal ของอุปกรณ์นั้นๆ ขึ้นที่หน้าต่าง Block Diagram โดย Terminal ที่แสดงผลขึ้นมานั้นจะทำหน้าที่เป็นตัวที่เชื่อมต่อ wires หรือสายสัญญาณเข้าด้วยกัน เพื่อเชื่อมต่อให้ตัวโปรแกรมนั้นทำงานเป็นไปตามเงื่อนไขที่ผู้สร้างไฟล์ในโปรแกรม LabVIEW ต้องการ

หลังจากที่เราวางอุปกรณ์ควบคุมและแสดงผลบน Front Panel เรียบร้อยแล้ว ต่อไปก็จะทำการพิจารณาใน Block Diagram ดังที่กล่าวข้างต้นว่าอุปกรณ์ควบคุมและแสดงผลที่ เราวางลงไปบน Front Panel นั้นทำให้เกิด Terminal ของอุปกรณ์เหล่านั้นที่ Block Diagram ด้วย แต่เนื่องจากบน Front Panel นั้นจะไม่ทำการแสดงอุปกรณ์ทุกตัว โดยอุปกรณ์บางชนิดนี้จะปรากฏเพียงแค่ในส่วนของ Block Diagram เท่านั้น

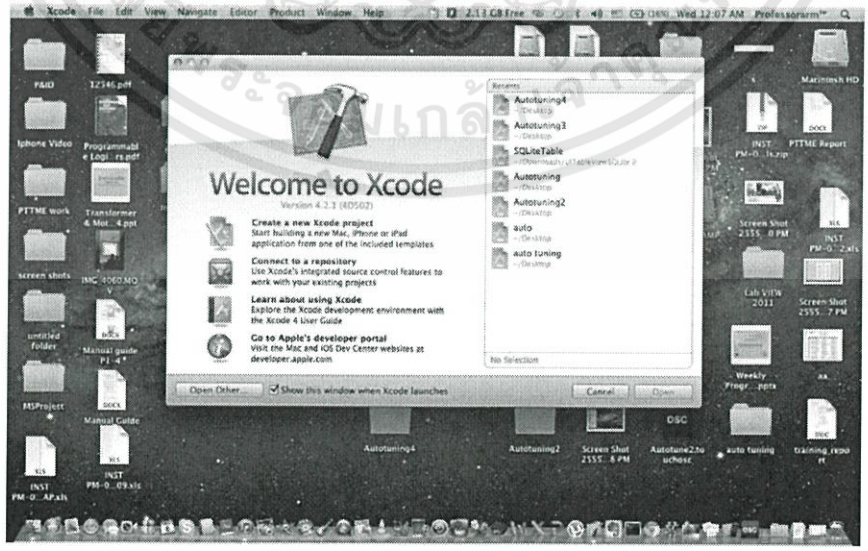
2.5 โปรแกรม Xcode

โปรแกรม Xcode เป็น Tools ที่ได้รับการพัฒนาโดย Apple Inc. ซึ่งใช้สำหรับสร้างซอฟท์แวร์และเขียนแอพพลิเคชันของอุปกรณ์ iOS และอุปกรณ์ที่ใช้ Mac OS X โดยจะเรียกว่า Xcode IDE (Integrated Development Environment) ซึ่งจะใช้ภาษา Objective-C ในการเขียนโปรแกรมเพื่อสร้างแอพพลิเคชันขึ้นมา โดยที่ตัวโปรแกรมนั้นสามารถทำการ Simulate อุปกรณ์ iOS ขึ้นมาเพื่อทำการทดสอบความถูกต้องของตัวโปรแกรมที่เขียนขึ้นมาก่อนได้

โดยภาษาที่ใช้ในการเขียนโปรแกรมเพื่อพัฒนาแอพพลิเคชันลงบนอุปกรณ์ iOS นั้น จะใช้ภาษาที่ เรียกว่า ภาษา Objective-C ซึ่งเป็นภาษาโปรแกรมเชิงวัตถุ (Object Oriented Programming) โดยมีโครงสร้างโปรแกรมที่คล้ายกับภาษา C++ แต่ในส่วนของภาษา Objective-C จะได้รับการพัฒนาโดยการเพิ่มเติมรูปแบบการเขียนโปรแกรม เพื่อรองรับการออกแบบโปรแกรมเชิงวัตถุ โดยจะใช้การส่ง message ไปยัง object ต่างๆ โดยจะไม่มีการเรียก method อย่างเช่นในภาษา C++

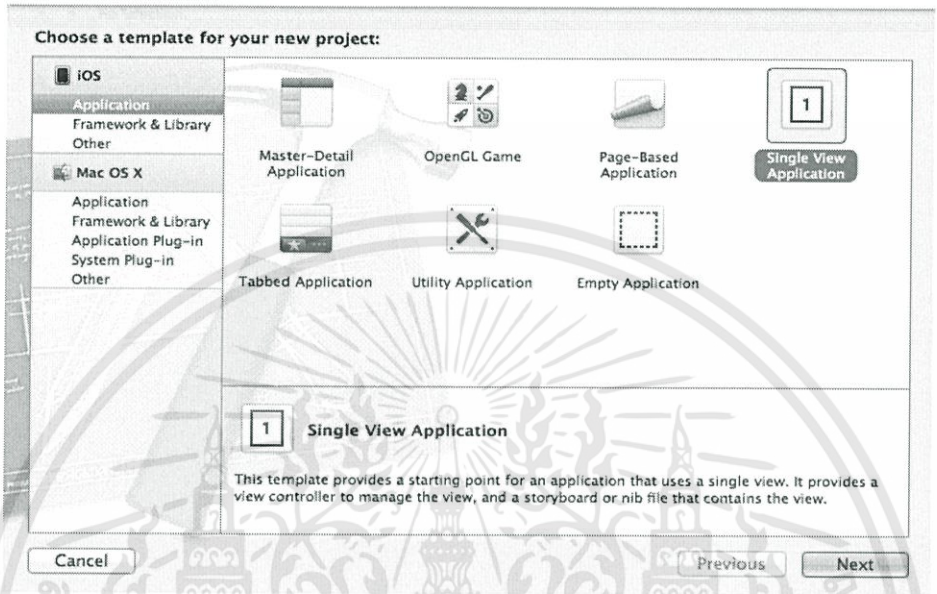
โดยการเริ่มสร้างโปรเจคใหม่ มีขั้นตอนในการสร้าง ดังนี้

1. ทำการเปิดโปรแกรม Xcode บน เครื่อง Mac แล้วทำการกดไปที่ Create a new Xcode Project เพื่อทำการสร้างโปรเจคใหม่ โดยโปรแกรม Xcode ที่ทำการใช้งานนั้นจะเป็นเวอร์ชัน 4.2.1



รูปที่ 2.22 แสดงรูป Start-up Window ของโปรแกรม Xcode เวอร์ชัน 4.2.1

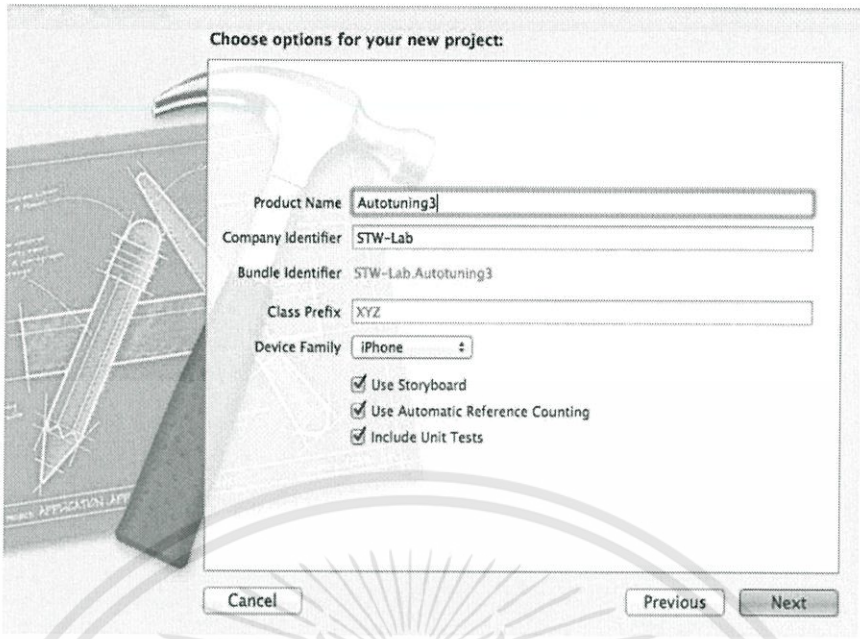
2. ทำการเลือกที่ Single View Application ซึ่งเป็นรูปแบบที่จะใช้ในการสร้าง View-based Application บน iOS ขึ้นมา



รูปที่ 2.23 แสดงขั้นตอนการเลือก Single View Application

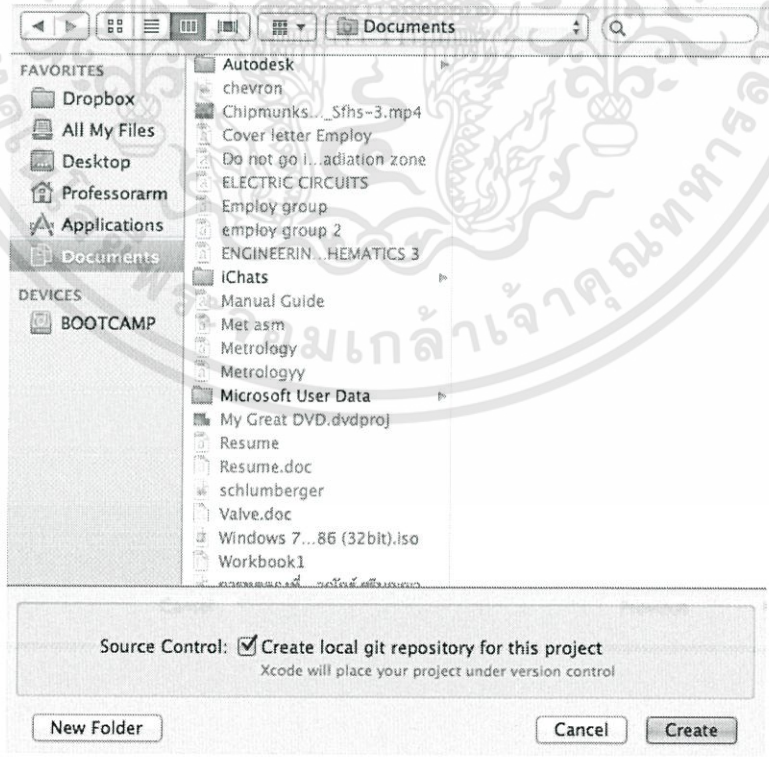
3. ใส่คำต่างๆลงในช่องต่อไปนี้
   - Product Name : ชื่อโปรเจคที่เราต้องการ
   - Company Identifier : สำหรับระบุชื่อความเป็นเจ้าของ Application
   - Bundle Identifier : ชื่อระบุ Bundle ของ Application นี้
   - Device Family : คือการเลือกว่าต้องการเขียน Application บนอุปกรณ์ iOS ประเภทไหน ได้แก่ iPhone, iPhone(Retina) หรือ iPad โดยในที่นี้จะทำการเลือก iPhone

เมื่อทำการเลือกเสร็จแล้ว ทำการกด Next เพื่อเข้าสู่ขั้นตอนถัดไป



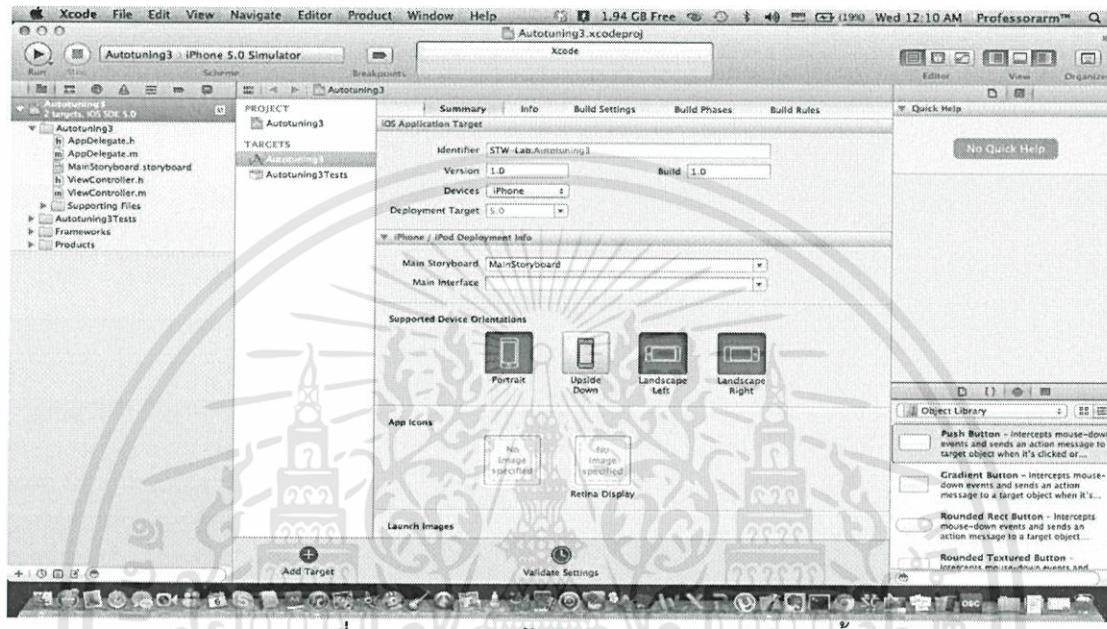
รูปที่ 2.24 แสดงหน้าต่างสำหรับใส่ข้อมูลต่างๆเกี่ยวกับโปรแกรม

4. ทำการเลือก Directory ที่เราต้องการจะทำการบันทึกไฟล์ เมื่อเลือกแล้วก็ทำการกดปุ่ม Create



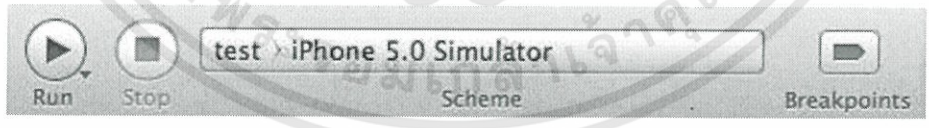
รูปที่ 2.25 แสดงหน้าต่างสำหรับเลือกตำแหน่งที่จะทำการบันทึกโปรเจค

5. เมื่อเลือก Directory ที่ต้องการบันทึกโปรเจคและกด Create แล้ว ตัวโปรแกรม Xcode จะทำการเข้ามายังหน้า Application พร้อมให้ทำการสร้างและพัฒนา Application บน iOS ได้ และสามารถทำการตรวจสอบระบบได้ โดนการกดปุ่ม Run



รูปที่ 2.26 แสดงหน้าต่างสำหรับปรับแต่งข้อมูลพื้นฐาน

โดยบริเวณด้านซ้ายบนของหน้าจอ เป็นส่วนของการ Run Project ใช้สำหรับทดลองการทำงานของโปรแกรมที่ทำการเขียนลงไป และปุ่ม Stop สำหรับหยุดการทดลอง



รูปที่ 2.27 แสดงหน้าต่างของการ Run Project

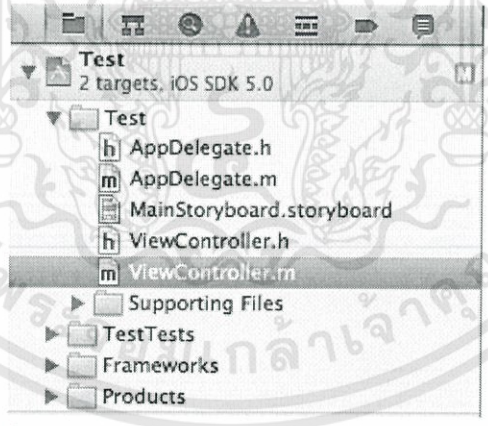
บริเวณด้านบนของหน้าจอ เป็นส่วนของ Jump Bar ใช้สำหรับแสดงว่ากำลังทำงานอยู่ในตำแหน่งใด

รูปที่ 2.28 แสดงหน้าต่าง Jump Bar

บริเวณด้านขวาบนของหน้าจอ จะเป็นในส่วนของการปรับแต่งหน้าจอทั้งหมดว่าเลือกเปิดหรือปิด หน้าต่างในส่วนใด

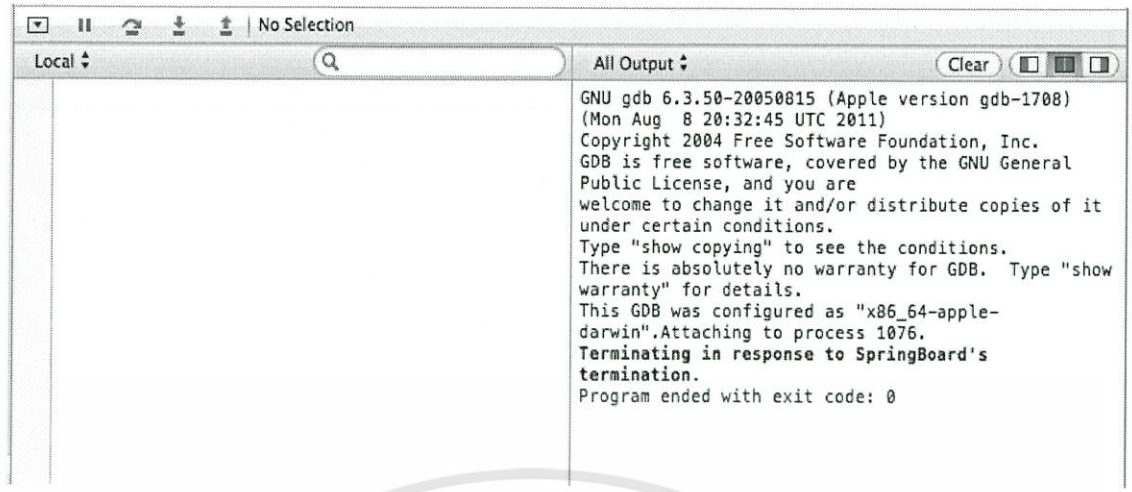
รูปที่ 2.29 แสดงหน้าต่างการปรับแต่งหน้าจอ

บริเวณด้านซ้ายของหน้าจอ เป็นส่วนของ Navigators ซึ่งแบ่ง Files ต่างๆ ออกเป็นหมวดหมู่ต่างๆกันไป



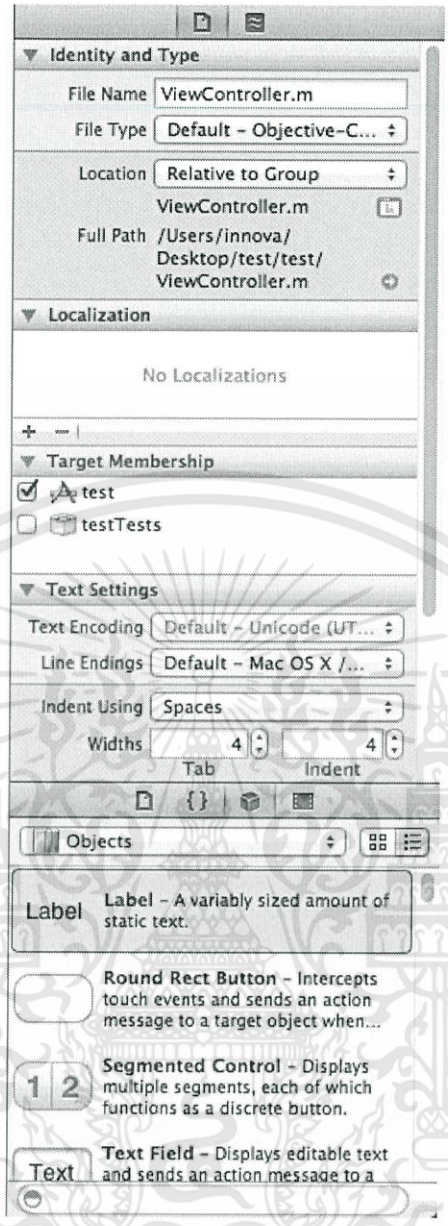
รูปที่ 2.30 แสดงหน้าต่างในส่วนของ Navigators

บริเวณด้านล่างของโปรแกรม จะเป็นส่วนของ Local และ Output สามารถปรับแต่งด้านล่างได้อีกครั้งว่าจะแสดง หน้าต่างไหนหรือทั้งสองหน้าต่าง



รูปที่ 2.31 แสดงส่วนของ Local และ Output ในโปรแกรม Xcode

บริเวณด้านขวาของโปรแกรม จะเป็นส่วนหน้าต่างของ Property ของ File, Text setting และ Object ต่างๆ



รูปที่ 2.32 แสดงหน้าต่างของ Property

6. เมื่อทำการกดปุ่ม Run แล้ว ถ้าไม่เกิดปัญหาอะไร ตัวโปรแกรม Xcode ก็จะทำการแสดงตัว Simulator ขึ้นมา เพื่อที่จะทำการแสดงผลการทดสอบ Application บนคอมพิวเตอร์ได้ ก่อนที่จะมีการใส่ Application ลงในตัวอุปกรณ์ iOS จริง

(ก)

(ข)

รูปที่ 2.33 (ก) แสดงถึงตัว Simulator ของ iPhone บนโปรแกรม Xcode

(ข) แสดงถึงตัว Simulator ของ iPad บนโปรแกรม Xcode

บทที่ 3

วิธีการดำเนินงาน

ในปริญญานิพนธ์เล่มนี้ ได้มีการทำตัวปรับค่าพีไอดีพารามิเตอร์อัตโนมัติขึ้นมา 2 แบบ ได้แก่

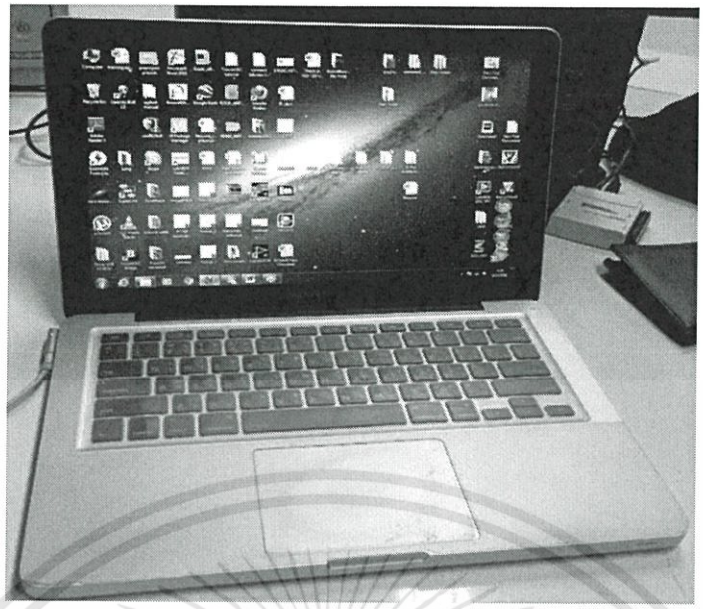
1. การสร้าง Layout โดยใช้โปรแกรม TouchOSC ร่วมกับโปรแกรม LabVIEW
2. การออกแบบ Layout ในการเขียน Application บนโปรแกรม Xcode

ซึ่งรายละเอียดของการออกแบบการทดลองทั้งสองนี้จะได้ทำการอธิบายอยู่ในหัวข้อที่ 3.2 และ 3.3 ตามลำดับ

3.1 อุปกรณ์ที่ใช้ในการออกแบบตัวปรับค่าพีไอดีพารามิเตอร์อัตโนมัติ

3.1.1 คอมพิวเตอร์ (Windows + Mac OS X)

คอมพิวเตอร์ในฝั่งระบบปฏิบัติการ Windows มีไว้สำหรับทำการสร้าง Layout เชื่อมต่อในโปรแกรม TouchOSC Editor รวมทั้งใช้สำหรับติดตั้งโปรแกรม LabVIEW เพื่อใช้ในการรับ – ส่งข้อมูลไปยังอุปกรณ์ iOS และคอมพิวเตอร์ในส่วนของระบบปฏิบัติการ Mac OS X มีหน้าที่สำหรับใช้เขียนโปรแกรม Xcode เพื่อทำการสร้าง Application ลงบนอุปกรณ์ iOS



รูปที่ 3.1 แสดงรูปคอมพิวเตอร์ที่ใช้ในการทดลอง

3.1.2 อุปกรณ์ iOS

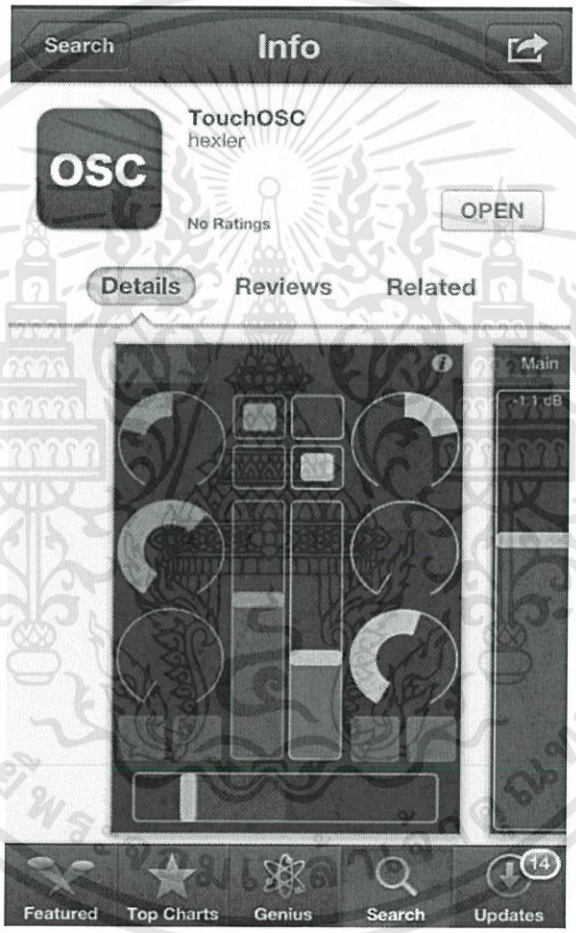
อุปกรณ์ iOS มีหน้าที่เปรียบเสมือนรีโมทควบคุมและปรับแต่งค่าพีไอดีพารามิเตอร์ โดยจะมีหน้าที่สำหรับติดตั้งโปรแกรม TouchOSC สำหรับการทดลองที่ใช้รูปแบบของการสร้าง Layout และเชื่อมต่อกับโปรแกรม LabVIEW และใช้เป็นที่ติดตั้งของ Application ที่ทำการเขียนบนคอมพิวเตอร์ สำหรับการทดลองที่ใช้รูปแบบของการใช้โปรแกรม Xcode

รูปที่ 3.2 แสดงอุปกรณ์ iOS ที่ใช้ในการทดลอง

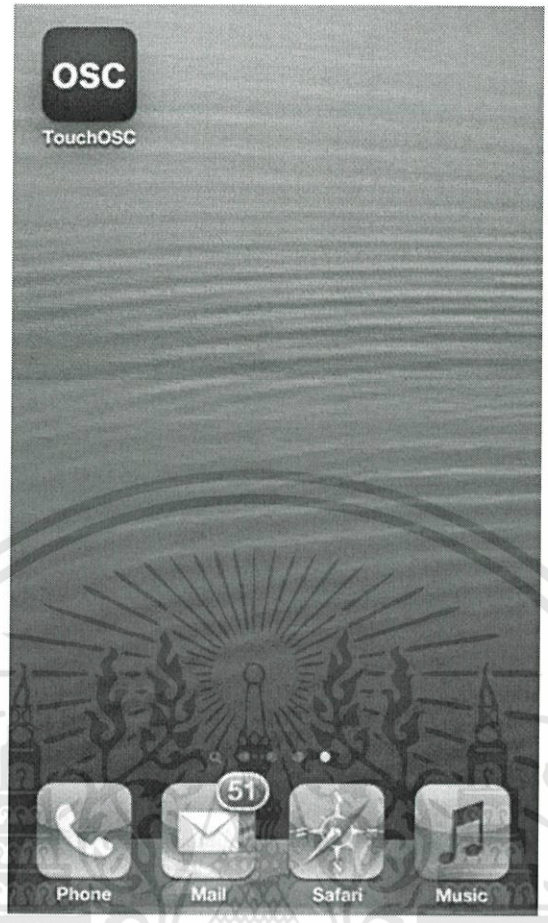
3.2 การสร้าง Layout และการเชื่อมต่อโดยใช้โปรแกรม TouchOSC + LabVIEW

3.2.1 การสร้าง Layout

ในการสร้าง Layout สำหรับปรับค่าพีไอดีพารามิเตอร์ลงบนอุปกรณ์ iOS นั้น จะต้องทำการใช้โปรแกรม TouchOSC สำหรับติดตั้งลงบนอุปกรณ์ iOS ซึ่งสามารถทำการดาวน์โหลดได้จาก App Store

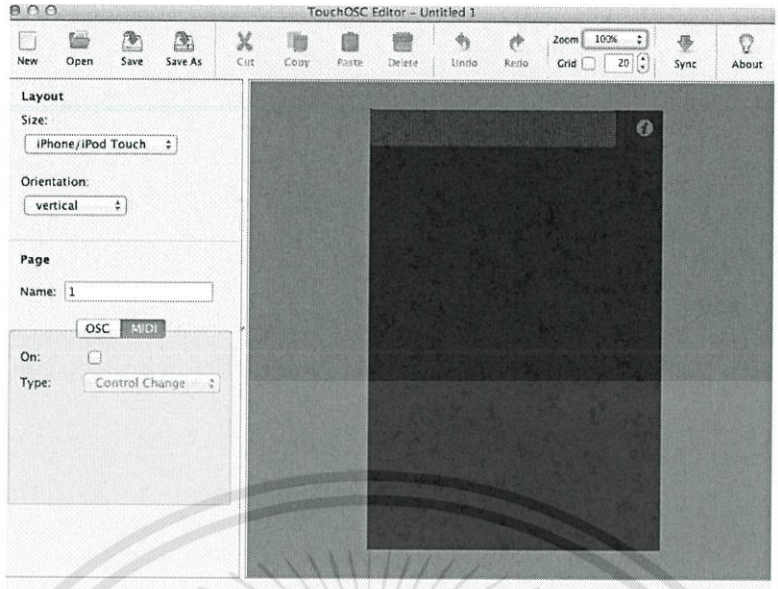


รูปที่ 3.3 แสดงหน้าดาวน์โหลด Application ที่มีชื่อว่า TouchOSC



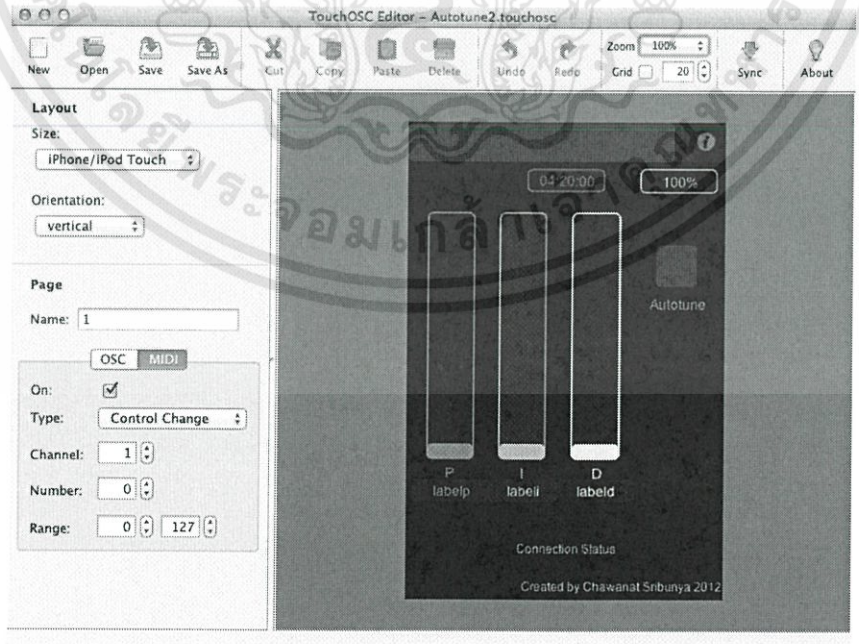
รูปที่ 3.4 แสดงตัว Icon ของโปรแกรม TouchOSC บน iOS

โดยภายในโปรแกรม TouchOSC นั้นจะมี Layout มาตรฐานอยู่ภายในโปรแกรมอยู่แล้ว แต่เนื่องจากการออกแบบ Layout สำหรับปรับค่าพีไอดีพารามิเตอร์อัตโนมัตินั้นจำเป็นที่จะต้องทำการออกแบบ Layout เสียใหม่ ซึ่งการที่เราจะออกแบบ Layout ใหม่นั้น จะต้องทำการติดตั้งโปรแกรม TouchOSC Editor ลงบนคอมพิวเตอร์เสียก่อน ซึ่งหน้าตาของโปรแกรม TouchOSC Editor จะแสดงดังรูปที่ 3.5



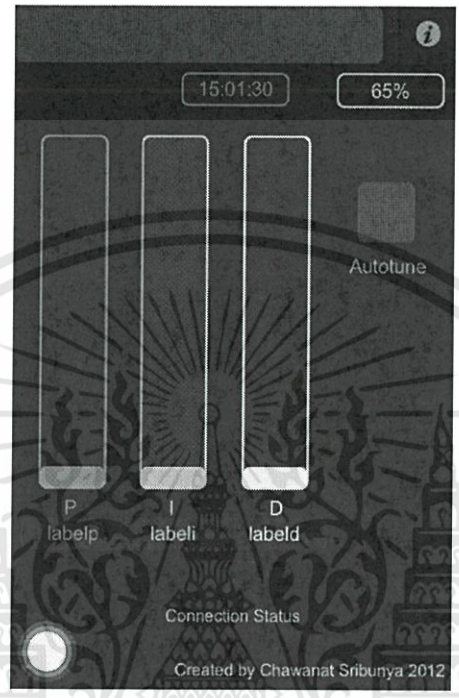
รูปที่ 3.5 แสดงหน้าตาของโปรแกรม TouchOSC Editor

โดยเราสามารถทำการปรับแต่ง Layout ของโปรแกรมได้ตามต้องการ ซึ่งเราสามารถเลือกรูปแบบของการจัดวาง รวมทั้งประเภทของอุปกรณ์ iOS ที่ต้องการจะใช้ได้ ไม่ว่าจะเป็น iPhone, iPod หรือ iPad โดยจะทำการแสดงในส่วนของการปรับแต่ง Layout ในภาพรวมดังรูปที่ 3.6 ต่อไปนี้



รูปที่ 3.6 แสดงถึงส่วนการปรับแต่ง Layout ในโปรแกรม TouchOSC Editor

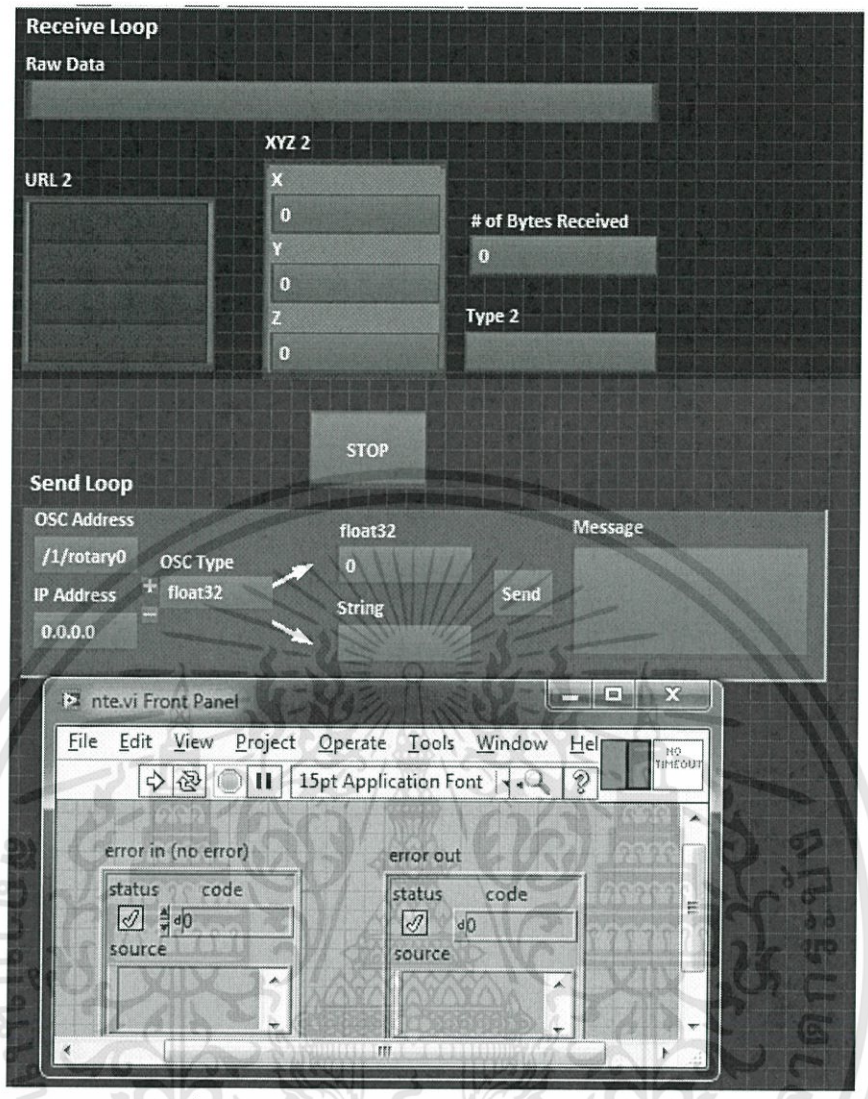
เมื่อเราทำการปรับแต่ง Layout เรียบร้อยแล้ว เราสามารถทำการ Sync ในส่วนของ Layout เพื่อทำการส่ง Layout ที่สร้างขึ้นในโปรแกรม TouchOSC Editor ไปยังตัวอุปกรณ์ iOS ที่ทำการติดตั้งโปรแกรม TouchOSC เรียบร้อยแล้ว โดยจะแสดงดังรูปที่ 3.7



รูปที่ 3.7 แสดง Layout ที่ได้ทำการสร้างบนโปรแกรม TouchOSC Editor

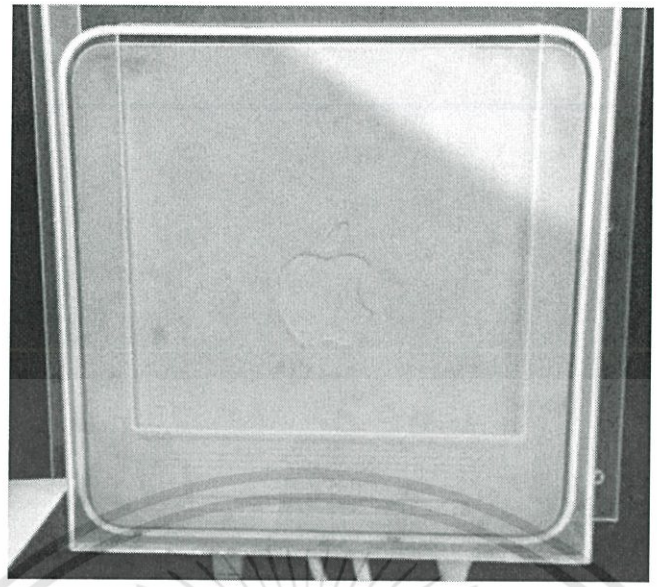
3.2.2 การเชื่อมต่อโปรแกรม TouchOSC เข้ากับโปรแกรม LabVIEW

เนื่องจากโปรแกรม TouchOSC แต่เดิมนั้นมีจุดประสงค์ในการพัฒนาเพื่อใช้เป็น MIDI Controller เป็นหลัก แต่ตัวโปรแกรมนั้นสามารถทำการประยุกต์เพื่อใช้ในการควบคุมพีไอดีพารามิเตอร์ได้โดยการเชื่อมต่อกับโปรแกรม LabVIEW โดยทำการใช้ไฟล์สกุล .vi ที่มีชื่อว่า PID.vi และ nte.vi



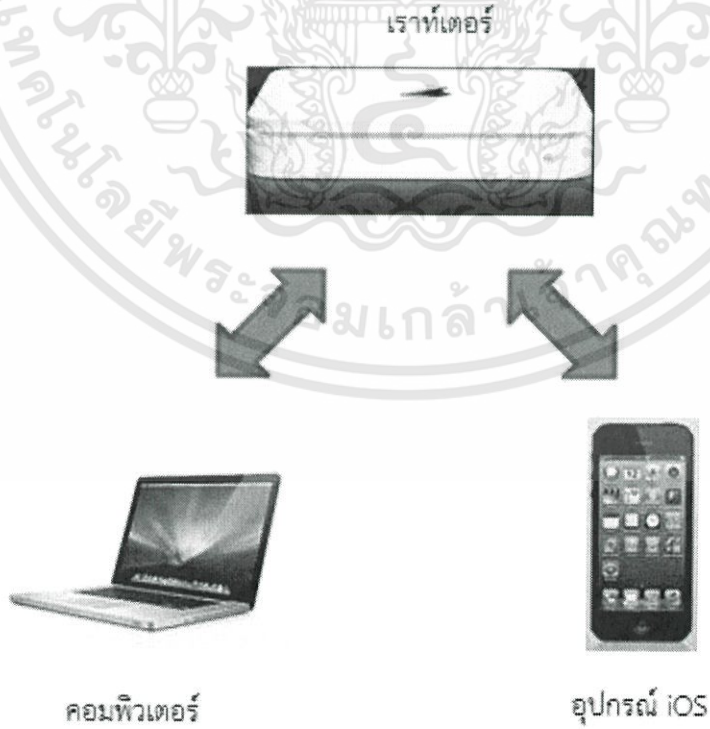
รูปที่ 3.8 แสดงไฟล์สกุล.vi ในโปรแกรม LabVIEW

ในส่วนของการเชื่อมต่อโปรแกรม TouchOSC เข้ากับโปรแกรม LabVIEW นั้น จะอาศัยการเชื่อมต่อผ่านเครือข่ายแลนแบบไร้สาย (Wireless LAN) โดยใช้เราท์เตอร์เป็นตัวกลางในการเชื่อมต่อเพื่อทำการรับ – ส่งข้อมูลระหว่างทั้งสองโปรแกรมเข้าด้วยกัน



รูปที่ 3.9 แสดงรูปของเราท์เตอร์ที่ใช้ในการเชื่อมต่อ

ซึ่งรูปแบบของการสื่อสารที่จะใช้นั้นจะเป็นแบบ UDP (User Datagram Protocol) ที่มีความสามารถในการรับ-ส่งข้อมูลที่รวดเร็ว โดยไม่จำเป็นที่จะต้องทำการสร้างการติดต่อขึ้นมาก่อน โดยจะสามารถแสดงแผนผังของการเชื่อมต่อได้ดังรูปที่ 3.9



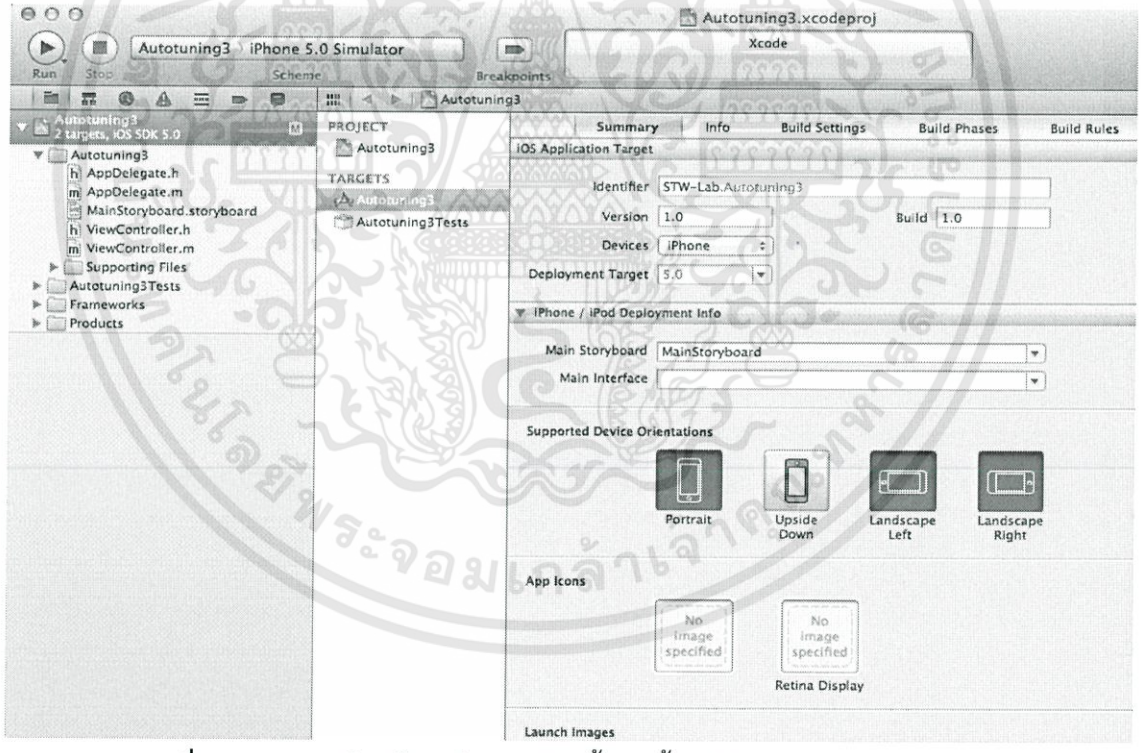
รูปที่ 3.10 แสดงแผนผังการเชื่อมต่อระหว่างอุปกรณ์ในภาพรวม

3.3 การออกแบบ Layout ในการเขียน Application บนโปรแกรม Xcode

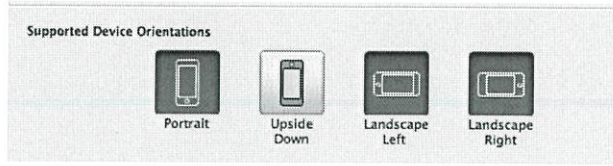
3.3.1 การปรับแต่งข้อมูลพื้นฐานในการเขียนโปรแกรม Xcode

ภายในโปรแกรม Xcode นั้นเราสามารถทำการปรับแต่งรูปแบบของโปรแกรมได้ตามต้องการ โดยในส่วนของข้อมูลพื้นฐานนี้ เราสามารถปรับแต่งและเลือกอุปกรณ์ที่เราต้องการจะทำการเขียน Application ลงไป รวมทั้งรูปแบบของการแสดงผล ว่าให้สามารถเอียงตามรูปแบบการถือของผู้ใช้ ไอคอนของ Application และในส่วนของ Launch Images ได้

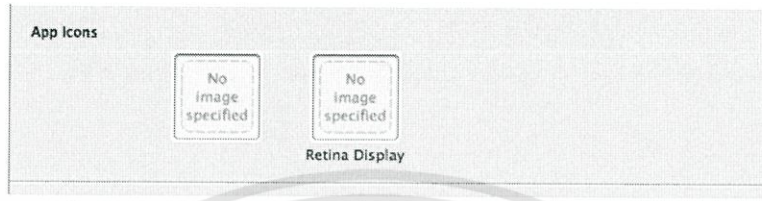
รูปที่ 3.11 แสดงถึงหน้าต่างสำหรับเลือกประเภทของอุปกรณ์ iOS ที่ต้องการเขียน



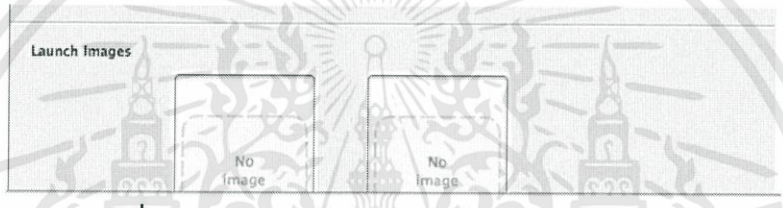
รูปที่ 3.12 แสดงถึงหน้าต่างในการปรับตั้งค่าเบื้องต้นในโปรแกรม Xcode



รูปที่ 3.13 แสดงหน้าต่างสำหรับเลือกรูปแบบการแสดงผล



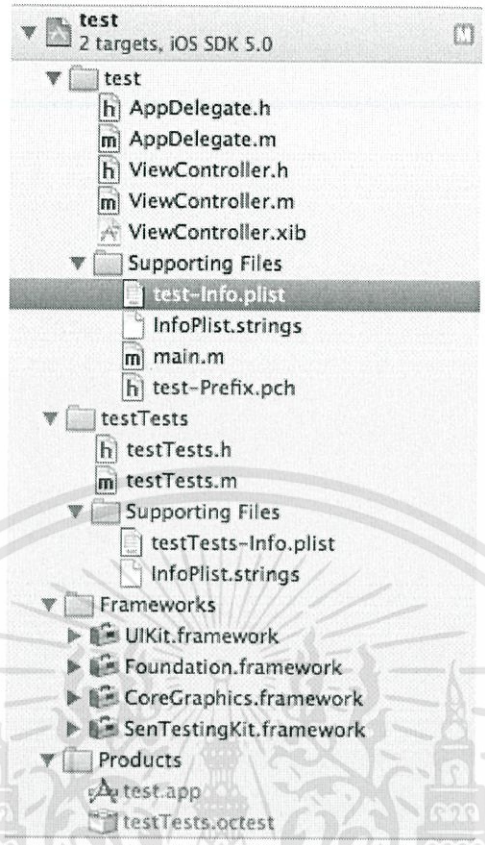
รูปที่ 3.14 แสดงหน้าต่างสำหรับเลือกไอคอนของ App



รูปที่ 3.15 แสดงหน้าต่างสำหรับใส่ Launch Images

3.3.2 รูปแบบของไฟล์ต่างๆในโปรแกรม Xcode

แต่ละโปรแกรมที่เราทำการสร้างขึ้นมานั้น จะมีรูปแบบของไฟล์ต่างๆกันไป ซึ่งแต่ละไฟล์จะมีหน้าที่แตกต่างกันไป โดยไฟล์และโฟลเดอร์ที่ถูกสร้างขึ้นมาในแต่ละ 1 โปรแกรม มีดังนี้



รูปที่ 3.16 แสดงไฟล์และโฟลเดอร์ต่างๆที่ถูกสร้างขึ้นใน 1 โปรแกรม

- Main Folder จะเป็นโฟลเดอร์ที่มีชื่อเดียวกับโปรแกรม (โฟลเดอร์ชื่อ test จากรูป) มีหน้าที่เก็บไฟล์สำหรับใช้ในการเขียนโปรแกรม ประกอบไปด้วยไฟล์ต่างๆ ดังต่อไปนี้
- ไฟล์สกุล .h และ .m คือส่วนของ Header กับ Source โดย .h จะเป็น Header เอาไว้ ประกาศ Class ส่วน .m จะเป็นส่วนของ Source files เป็นที่ไว้สำหรับเขียน Objective-C (เหมือนกับสกุล .c ในการเขียนภาษาซี)
- ไฟล์สกุล .xib คือตัวสร้าง Interface (Interface Builder) สำหรับการทำโปรแกรม
- Supporting Files Folder จะเป็นโฟลเดอร์สำหรับเก็บไฟล์ต่างๆที่เป็นคุณสมบัติของโปรเจค (ตัว Application) ที่เราทำการสร้างขึ้น ประกอบไปด้วยไฟล์ต่างๆ ดังนี้
- ไฟล์สกุล .plist (property list) มีหนเก็บค่าคุณสมบัติต่างๆ ของ Application
- ไฟล์สกุล .strings เป็น text file ที่เอาไว้เก็บ Version ของโปรแกรม (ใช้ Note ปกติ)

- ไฟล์สกุล .pch (pre compile header) จะรวมรายชื่อ header ของ framework อื่นๆ ที่ต้องใช้ในโปรเจคและจะมีการทำการคอมไพล์ล่วงหน้าก่อน เพื่อลดเวลาในการคอมไพล์ ของโปรเจคหรือตัว Application ลง
- Framework Folder
- ไฟล์สกุล .framework ทำหน้าที่เสมือน library ในภาษาซี มีหน้าที่ช่วยให้ผู้ใช้ทำการ เขียนโปรแกรมได้ง่ายขึ้น
- Product Folder
- ไฟล์สกุล .app ก็คือ Application ของ Project ซึ่งเป็นผลลัพธ์ของโปรเจคทั้งหมด

3.3.3 การประกาศ Class และ Method ในภาษา Objective-C

การประกาศ Class ในภาษา Objective-C สามารถทำได้ 2 แบบ ได้แก่แบบ Strong type และ Weak type โดยแต่ละแบบจะมีวิธีการประกาศ Class ที่แตกต่างกันไป ดังนี้

1. แบบ Strong type จะทำการประกาศ Class โดยการพิมพ์คำว่า Class แล้วตามด้วยเครื่องหมาย * โดยเครื่องหมาย * จะทำหน้าที่เป็นตัวชี้หรือ Pointer เช่นคำว่า Class *Autotuning; เป็นต้น
2. แบบ Weak type จะทำการประกาศ Class โดยการพิมพ์คำว่า id แล้วตามด้วยชื่อ ซึ่งคำว่า id นั้นทำหน้าที่เป็น Pointer จึงไม่จำเป็นที่จะต้องใช้เครื่องหมาย * เหมือนกับแบบ Strong type เช่นคำว่า id Autotuning; เป็นต้น

การประกาศ Method ในภาษา Objective-C สามารถทำได้ 2 แบบเช่นกัน ได้แก่ แบบที่มีเครื่องหมายบวกนำหน้า (+) และแบบที่ขึ้นต้นด้วยเครื่องหมายลบ (-) โดยแต่ละแบบนั้นมีจุดประสงค์ในการใช้งานที่แตกต่างกัน ดังนี้

1. Method ที่ขึ้นต้นด้วยเครื่องหมายบวก หมายความว่าเป็น Method ที่ไม่จำเป็นต้องมีการสร้าง Object ขึ้นมาก่อน แต่ใน Method แบบนี้จะมีการสร้าง Object ขึ้นมาหลังจากที่ Method นี้ถูกเรียกใช้
2. Method ที่ขึ้นต้นด้วยเครื่องหมายลบ หมายความว่าเป็น Method ทั่วไป ที่เอาไว้กำหนดงานต่างๆของ Class จึงจะสามารถเรียก Method นี้มาใช้งานได้

และ Method แบบนี้ เราจะต้องทำการสร้าง Object ของ Class นั้นๆขึ้นมาก่อน

โดยในส่วนของวิธีการประกาศ Method นั้น จะสามารถทำได้โดยการพิมพ์ไว้ในเครื่องหมายก้ามปู ([ ])

บทที่ 4

การทดลองและผลการทดลอง

4.1 กล่าวนำ

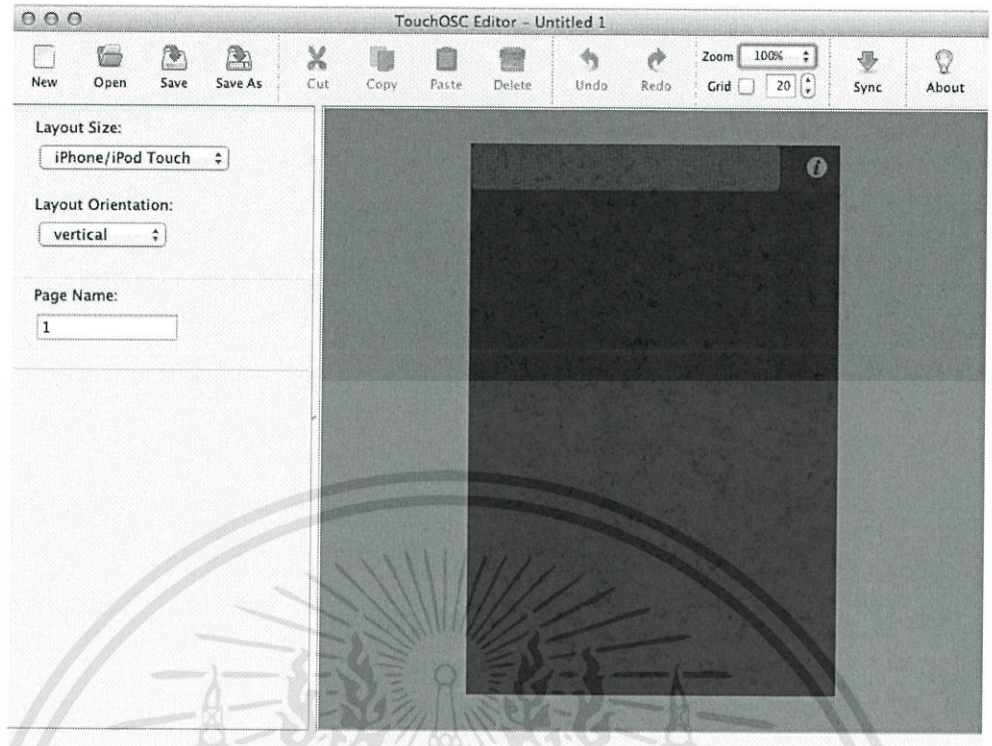
หลังจากในส่วนของบทที่แล้วได้กล่าวถึงอุปกรณ์ และการออกแบบการเชื่อมต่อ ทั้งในรูปแบบของการออกแบบ Layout ในโปรแกรม TouchOSC เพื่อทำการควบคุมค่าพารามิเตอร์ในโปรแกรม LabVIEW และในโปรแกรม Xcode แล้ว ในส่วนของบทนี้ก็จะทำการกล่าวถึงในส่วนของการสร้าง Layout ในโปรแกรม TouchOSC และการเขียน Application ด้วยโปรแกรม Xcode ลงบนอุปกรณ์ iOS

4.2 การทดลองบนโปรแกรม TouchOSC + LabVIEW

ในส่วนของหัวข้อนี้ จะทำการกล่าวถึงการสร้าง Layout ขึ้นเพื่อใช้เป็น Layout สำหรับทำการปรับค่าและทำการควบคุมพีไอดีพารามิเตอร์แบบอัตโนมัติบนอุปกรณ์ iOS โดยใช้โปรแกรม TouchOSC ไปยังตัวควบคุมพีไอดีพารามิเตอร์ที่สร้างขึ้นบนโปรแกรม LabVIEW ผ่านทาง wi-fi ซึ่งอุปกรณ์ iOS ที่ทำการใช้ปรับค่านั้นมีหน้าที่เปรียบเสมือนกับรีโมทคอนโทรล ซึ่งเราสามารถสั่งการได้จากอุปกรณ์โดยตรง โดยมีขั้นตอนต่างๆดังต่อไปนี้

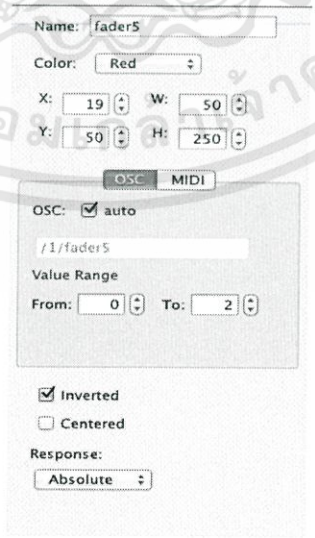
4.2.1 การปรับแต่งและการกำหนดสเกลของ Layout บนโปรแกรม TouchOSC Editor

ในการปรับแต่ง Layout ที่จะนำไปแสดงผลในโปรแกรม TouchOSC บนอุปกรณ์ iOS นั้น สามารถทำการปรับแต่งได้โดยใช้โปรแกรม TouchOSC Editor ซึ่งมีหน้าตาของโปรแกรมดังรูปที่ 4.1



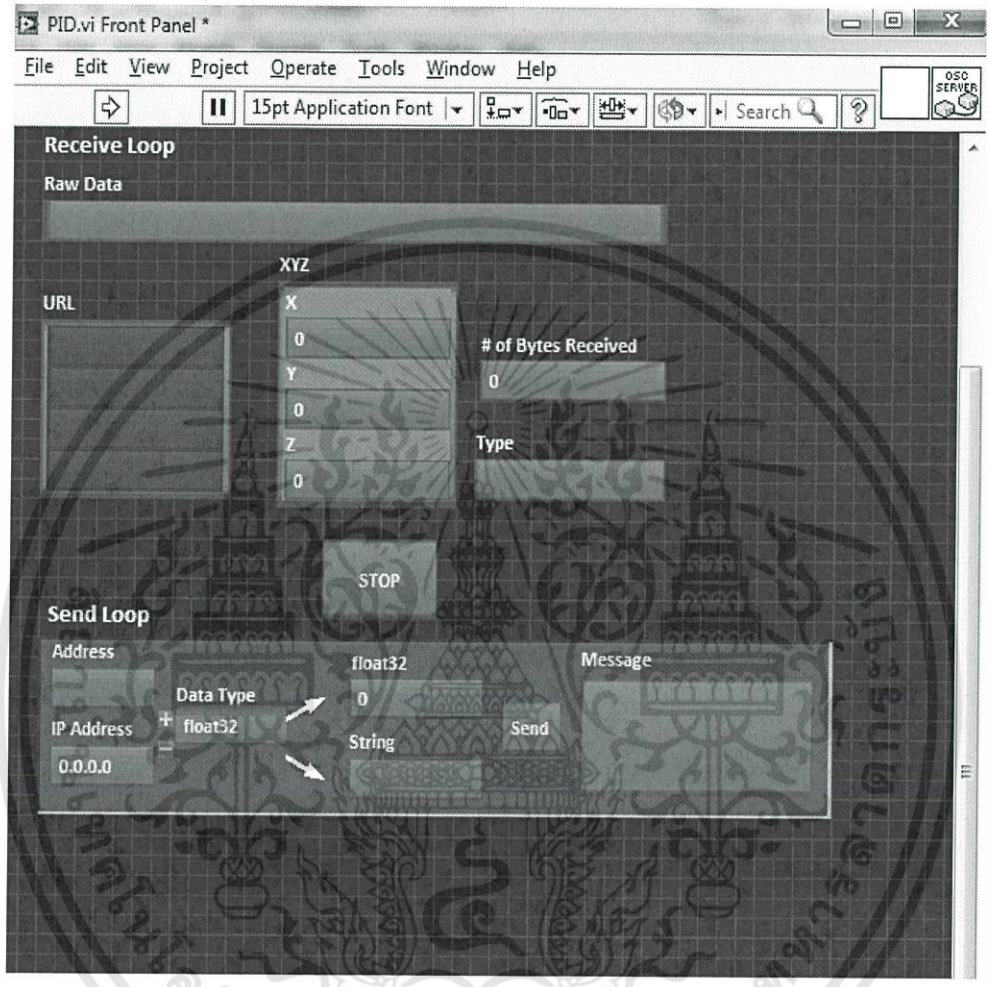
รูปที่ 4.1 แสดงโปรแกรม TouchOSC Editor

โดยสามารถเลือกประเภทของเครื่องมือที่จะนำมาใส่ได้ โดยการคลิกขวาที่ตัวไฟล์ที่ทำการสร้างขึ้น แล้วทำการเลือกประเภทของอุปกรณ์ตามต้องการ และอุปกรณ์แต่ละตัวสามารถกำหนดค่าได้ โดยจะแสดงตัวอย่างการกำหนดค่า, สเกล และเรนจ์ (Range) ของ Fader ซึ่งเป็นตัวปรับค่าแบบสไลด์ขึ้น-ลง ดังรูปที่ 4.2

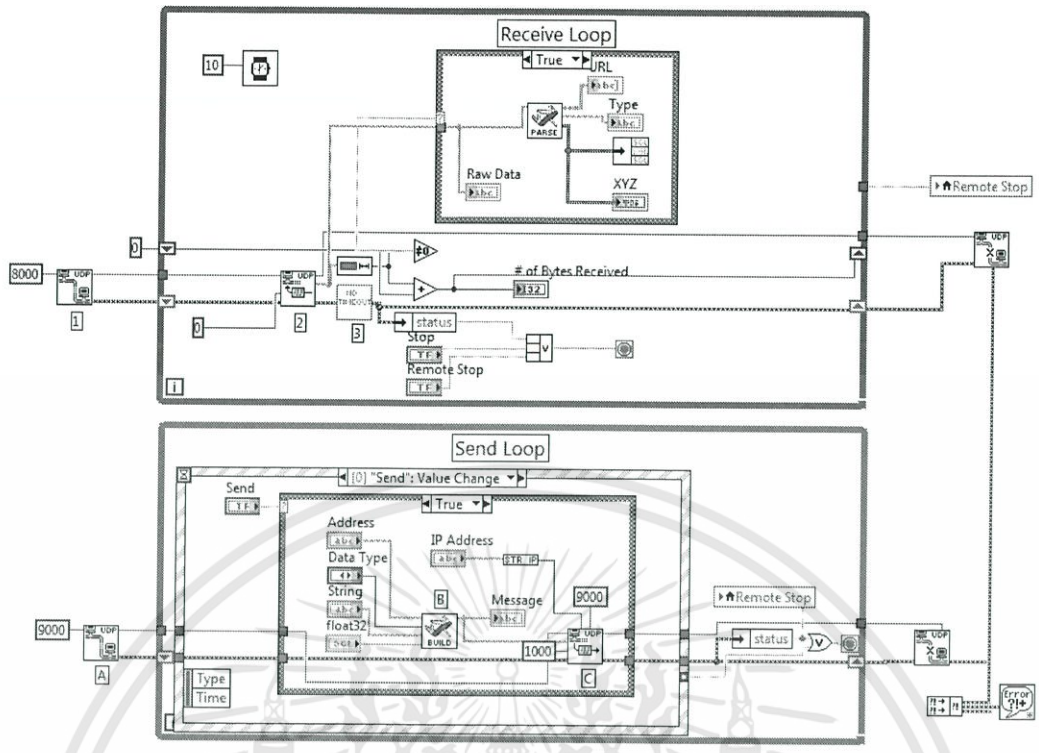


รูปที่ 4.2 แสดงตัวอย่างการปรับตั้งค่าของตัว Fader ซึ่งมี OSC Address คือ /1/fader5

และในส่วนของโปรแกรม LabVIEW นั้นจะทำการใช้ไฟล์สกุล.vi ที่ทำการสร้างและดัดแปลง ซึ่งมี Front Panel และ Block Diagram ดังรูปที่ 4.3 และ 4.4 ซึ่งมีชื่อว่า PID.vi

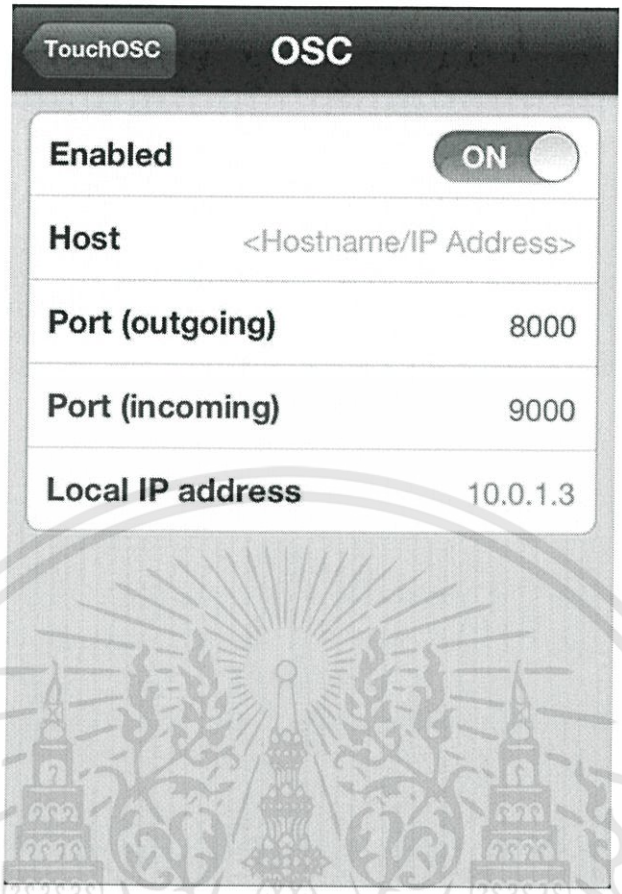


รูปที่ 4.3 แสดง Front Panel ของไฟล์ PID.vi



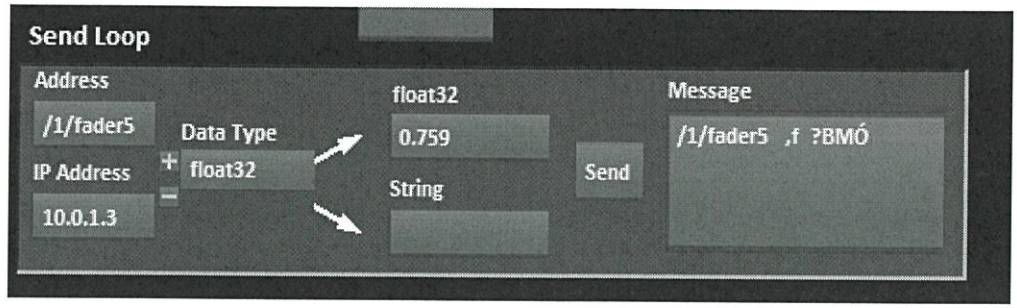
รูปที่ 4.4 แสดง Block Diagram ของไฟล์ PID.vi

มาถึงในส่วนของการปรับตั้งค่าบนโปรแกรม TouchOSC เพื่อที่จะทำการส่งค่าพีไอดีพารามิเตอร์ ผ่านทางเครือข่าย wi-fi ซึ่งเราจะต้องทำการกำหนดค่าความเร็วของ Port ทั้งด้านรับและด้านส่ง ซึ่งควรจะใช้ Port Number ที่แตกต่างกัน โดย Port ส่งของโปรแกรม TouchOSC ก็จะกลายเป็น Port รับของโปรแกรม LabVIEW และในทางตรงกันข้าม Port รับของโปรแกรม TouchOSC ก็จะกลายเป็น Port ส่งของโปรแกรม LabVIEW เช่นกัน โดยในการทดลองนี้ได้ทำการตั้ง Port number ในการส่งข้อมูลออกจากโปรแกรม TouchOSC ไว้ที่ 8000 และความเร็วในการรับข้อมูลเข้ามายังโปรแกรม TouchOSC ไว้ที่ 9000 และเช่น การกำหนด Local IP address ไว้ที่ 10.0.1.3 เหมือนกันกับในโปรแกรม LabVIEW



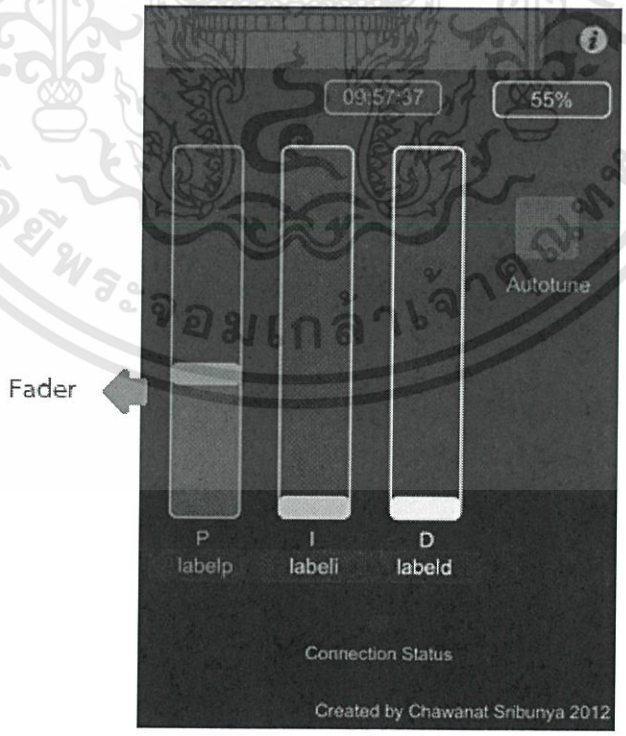
รูปที่ 4.5 แสดงการปรับตั้งค่าต่างๆบนโปรแกรม TouchOSC

เมื่อทำการตั้งค่าในส่วนของโปรแกรม TouchOSC แล้ว ในส่วนของโปรแกรม LabVIEW ก็จะต้องมีการปรับตั้งค่าเช่นกัน ซึ่งในส่วนของความเร็วในการรับ-ส่งข้อมูลนั้นได้มีการกำหนดค่าไว้ที่ตัวโปรแกรมเองเรียบร้อยแล้ว และถ้าเป็นกรณีที่ต้องการจะให้ตัวควบคุมพีไอดีพารามิเตอร์บนโปรแกรม LabVIEW ทำการส่งค่าไปยังอุปกรณ์ iOS สามารถทำได้โดยการใส่ OSC Address ลงไปในช่อง Address ของ Send Loop เช่น /1/fader5 เป็นต้น ทำการกำหนด IP address ให้ตรงกับภายในโปรแกรม TouchOSC และทำการเลือกประเภทของข้อมูลที่จะทำการส่งข้อมูลว่าจะส่งในรูปของข้อความ (string) หรือเป็นตัวเลข (float32) แล้วทำการกดปุ่ม Run โดยการตั้งค่า โดยขั้นตอนข้างต้นจะสาธิตดังรูปที่ 4.6



รูปที่ 4.6 แสดงการตั้งค่าบนโปรแกรม LabVIEW ในกรณีที่ต้องการจะส่งข้อมูลไปยังอุปกรณ์ iOS (ในกรณีนี้จะแสดงการส่งในรูปแบบของ float32)

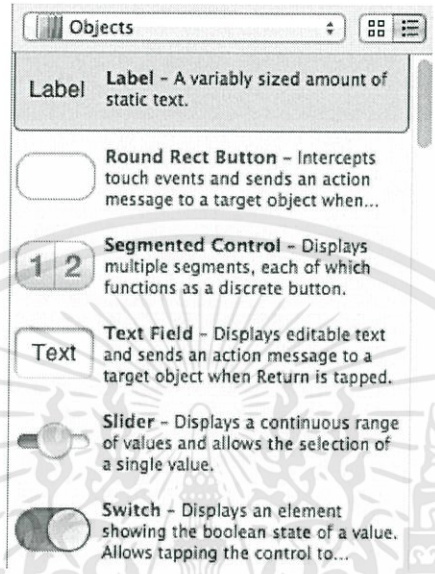
เมื่อทำการกดปุ่ม Send เพื่อทำการส่งข้อมูลในรูปของ float32 ออกไปยังอุปกรณ์ iOS จะได้ผลของการรับค่า โดย Address ที่ชื่อว่า /1/fader5 หมายถึงตัว Fader สีแดง ซึ่งแทนค่าของ K_p และ Fader สีเขียว ซึ่งแทนค่า Ki จะมี Address คือ /1/fader7 และ Fader สีเหลือง ซึ่งทำหน้าที่แทนค่าของ K_d จะมี Address คือ /1/fader6 ซึ่งในการทดลองนี้ได้มีการกำหนดเรนจ์ (Range) ของ Fader ทั้งสามตัว ซึ่งก็คือ Range ของ K_p, K_i และ K_d ไว้ที่ 0 ถึง 2 เมื่อทำการส่งค่า float32 ที่ 0.759 ออกไป ผลลัพธ์ที่ได้ก็จะแสดงดังรูปที่ 4.7



รูปที่ 4.7 แสดงผลลัพธ์ที่ได้บนอุปกรณ์ iOS เมื่อทำการรับค่าที่ส่งมาจากโปรแกรม LabVIEW แล้ว

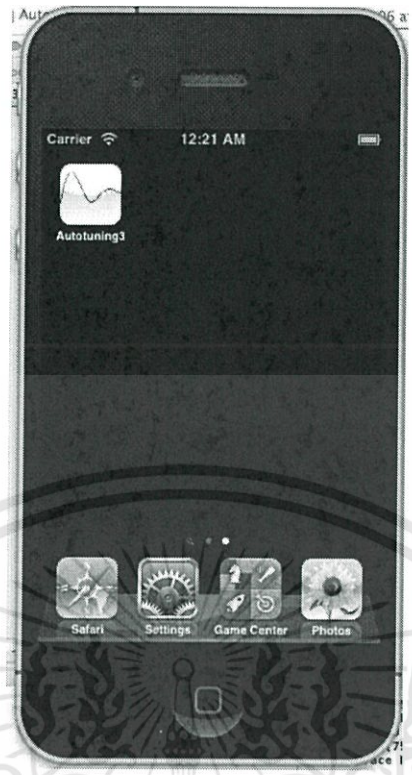
4.3 การทดลองบนโปรแกรม Xcode

ในส่วนของหัวข้อนี้ จะกล่าวถึงการเขียน Application บนโปรแกรม Xcode เพื่อทำการสร้าง Application สำหรับทำการควบคุมและปรับค่าพีไอดีพารามิเตอร์อัตโนมัติ



รูปที่ 4.8 แสดงปุ่มต่างๆซึ่งเป็นเครื่องมือในการสร้างตัว Application

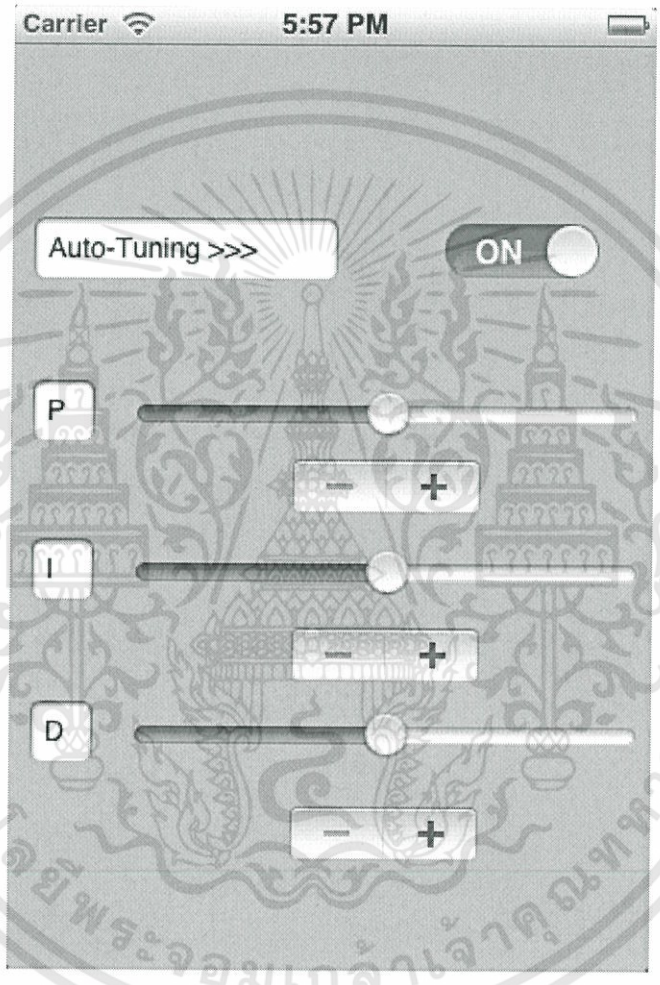
เมื่อทำการเขียนโปรแกรมเพื่อทำการสร้าง Application แล้ว เราสามารถทำการจำลองการทำงานของโปรแกรมก่อนได้ โดยใช้โปรแกรม iOS Simulator เพื่อความสะดวกในการตรวจสอบความถูกต้องของโปรแกรม ซึ่งไม่จำเป็นที่จะต้องนำอุปกรณ์ iOS ไปทดลองใช้โดยตรง เพื่อลดปัญหาของความผิดพลาดของโปรแกรมที่อาจจะเกิดขึ้นได้ โดยตัวอย่างของ iOS Simulator แสดงดังรูปที่ 4.9 ถึง 4.10



รูปที่ 4.9 แสดงการจำลองบน iOS Simulator โดย Application ที่ทำการสร้างขึ้นมามีชื่อว่า Autotuning3 ปรากฏเป็นไอคอนอยู่บริเวณมุมซ้ายบนของตัว Simulator

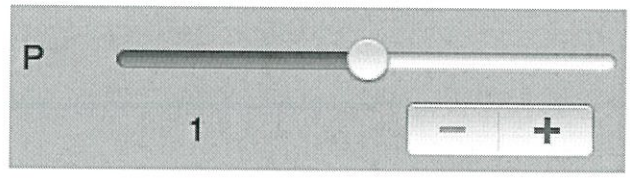
รูปที่ 4.10 แสดงถึงตัว Application ที่ได้ทำการสร้างในขั้นแรก

เนื่องจากได้มีการปรับปรุงตัว Application อยู่เป็นระยะๆ เพื่อให้ตัวโปรแกรมมีความสมบูรณ์และครอบคลุมมากขึ้น จึงได้มีการพัฒนาและปรับปรุงรูปแบบในส่วนของตัว Application มีหลายรูปแบบ โดยในการทดลองนี้ ได้มีการกำหนด Range ของตัว Slider ทั้งสามแท่งไว้ที่ระยะ 0-2 เช่นเดียวกับการทดลองในหัวข้อที่ 4.1

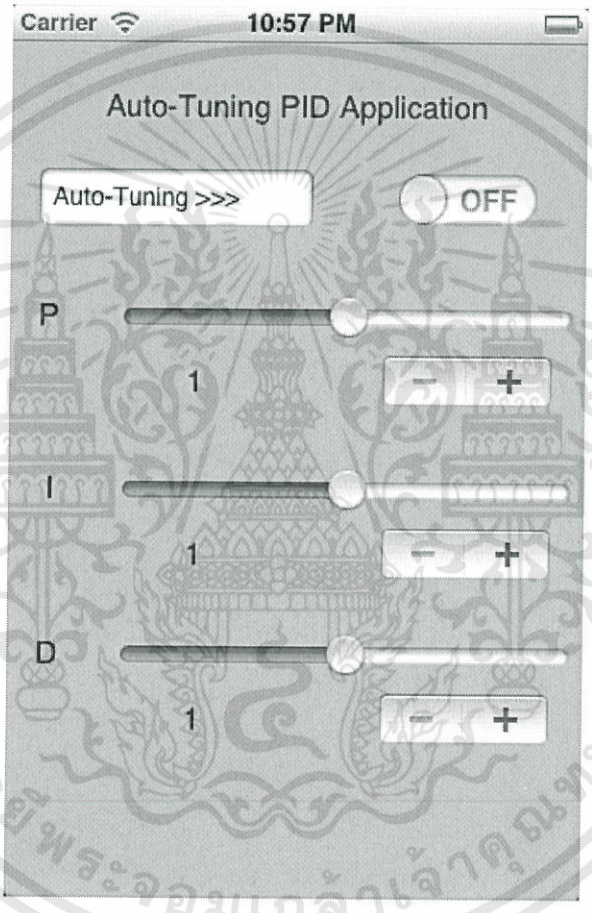


รูปที่ 4.11 แสดงถึงตัว Application ที่ได้ทำการปรับปรุงต่อมา

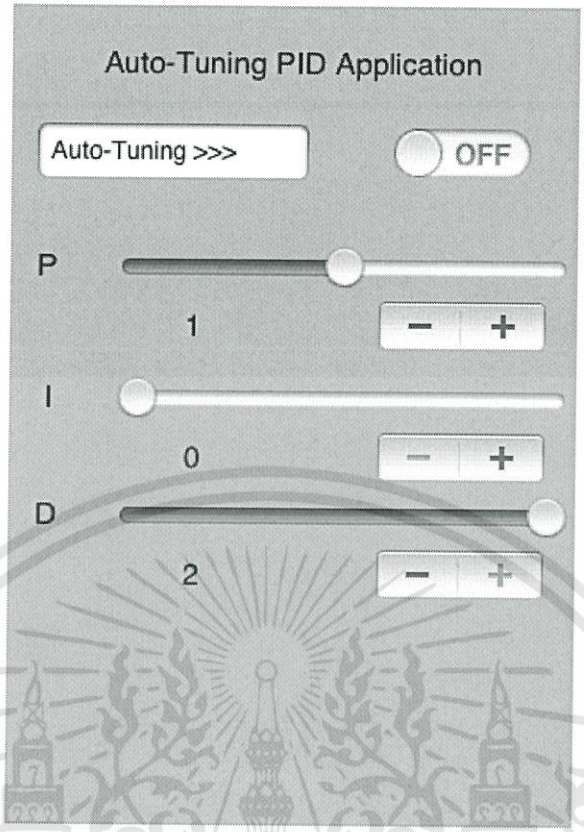
โดยในการปรับปรุงรอบล่าสุด ได้มีการทำการเพิ่มเติมในส่วนของปุ่มปรับเพิ่ม/ลดค่า (ปุ่มที่เป็นเครื่องหมายบวกและลบ) รวมทั้งมีการแสดงค่าพารามิเตอร์ในรูปของตัวเลข เพื่อความสะดวกสบายและเพิ่มความแม่นยำในการอ่านค่าพารามิเตอร์มากขึ้น



รูปที่ 4.12 แสดงสัญลักษณ์ต่างๆที่ใช้ในการเขียน Application โดย แถบสีฟ้า-ขาว คือ Slider สำหรับปรับค่าขึ้น-ลง โดยการเลื่อนไปทางซ้ายและขวา



รูปที่ 4.13 แสดงถึงตัว Application ที่ได้รับการปรับปรุงล่าสุด



รูปที่ 4.14 แสดงถึงการปรับค่าของ K_p, K_i, K_dขึ้น-ลง โดยการเลื่อนไปทางซ้ายเพื่อทำการลดค่าลง และทำการเลื่อนไปทางขวาเพื่อเพิ่มค่าพารามิเตอร์ให้มากขึ้น

บทที่ 5

สรุปผลการทดลองและข้อเสนอแนะ

5.1 สรุปผลการทดลอง

หลังจากที่ได้ทำการออกแบบ และทำการเขียนโปรแกรมต่างๆเพื่อที่จะใช้ในการสร้างตัวปรับค่าพีไอดีพารามิเตอร์อัตโนมัติบนอุปกรณ์ iOS แล้ว ผลการทดลองที่ได้ นั่นคือ สามารถทำการออกแบบและสามารถปรับค่าพีไอดีพารามิเตอร์ได้จากทางอุปกรณ์ iOS โดยทำการปรับค่าพารามิเตอร์ผ่านทางเครือข่าย wi-fi เพื่อให้สามารถทำการควบคุมค่าพารามิเตอร์จากระยะไกลโดยผ่านทางเครือข่าย wi-fi ได้

5.2 ปัญหาที่พบในการวิจัยและแนวทางในการแก้ปัญหา

ปัญหาที่พบในการทดลองนั้นพบว่ามีปัญหาอยู่บ้าง ไม่ว่าจะเป็นการปรับตั้งค่าเพื่อให้สามารถรับ-ส่งโปรแกรมได้อย่างสมบูรณ์ เนื่องจากการเชื่อมต่อนั้นจะต้องทำการต่อ wi-fi ในวงเดียวกัน และต้องทำการเซ็ทค่า IP address ให้ตรงกันทั้งตัวคอมพิวเตอร์และอุปกรณ์ iOS และกราฟของการทดลองพบว่าค่า Process Variable ยังไม่สามารถเข้าสู่ค่า Setpoint ได้อย่างรวดเร็วเท่าที่ควร ซึ่งเป็นผลมาจากการเขียนโปรแกรมที่ทำการใส่ค่า Range ของการปรับพีไอดีพารามิเตอร์ที่น้อยเกินไป รวมทั้งการศึกษาการเขียนโปรแกรมด้วยภาษา Objective-C ซึ่งแม้ว่าถ้ามองผิวเผินจะมีความคล้ายคลึงกับภาษา C++ แต่ในเรื่องของการเขียนจริงนั้นพบว่า มีความแตกต่างกันมาก เนื่องจากการเขียนโปรแกรมภาษา Objective-C เป็นการเขียนในรูปแบบของการเขียนโปรแกรมเชิงวัตถุ (Object Oriented Programming) ซึ่งต่างจากภาษา C++ ซึ่งปัญหาเหล่านี้สามารถแก้ไขได้โดยการฝึกฝนและเรียนรู้หาเทคนิคการเขียน Application อย่างสม่ำเสมอ

5.3 ข้อเสนอแนะและแนวทางในการพัฒนา

จากการออกแบบและทดสอบตัวปรับค่าพีไอดีพารามิเตอร์อัตโนมัติบน iOS นั้น พบว่าสามารถนำแนวทางเหล่านี้ไปทำการพัฒนาได้อีก ไม่ว่าจะเป็นการสร้าง Application ขึ้นมาเพื่อใช้สำหรับการปรับค่าพีไอดีพารามิเตอร์ในสมาร์ทโฟนประเภท Android และ Windows Phone รวมทั้งอาจจะทำการเพิ่มเติมในส่วนของการทำให้แสดงผลผ่านเว็บไซต์ ซึ่งมีระยะของการควบคุมที่ไกลกว่าการใช้ wi-fi เป็นต้น

บรรณานุกรม

กิจไพบูลย์ ชีวพันธุศรี. 2554. **LabVIEW ซอฟต์แวร์เพื่อการพัฒนาระบบการวัดและควบคุม.** กรุงเทพฯ: วี.พรินท์,

ขจรศักดิ์ สังข์เจริญ. 2555. **การเขียน iPhone Apps สำหรับผู้เริ่มต้น.** พิมพ์ครั้งที่ 1 กรุงเทพมหานคร. นายอินทร์.

รวิทัต ภู่หลำ . 2554. **คู่มือเขียน iPhone Apps.** พิมพ์ครั้งที่ 1 กรุงเทพ ฯ โปรวิชั่น,บจก

C.J.Savant, Jr.,Ph.D. " **Control System Design**". McGraw-Hill, Inc. 1964

M. Zhuang, and D.P.Athenon, "**PID controller design for a TITO system**", Iee Proc.-Control Theory Appl., 1994

Erica Sadun . 2552. **The iPhone Developer's Cookbook Building Application with the iPhone SDK**. USA : Addison – Wesley

http://www.arjin.info