ระบบควบคุมพิทช์ชั่น Pitch bend และ Vibrato สำหรับเครื่องดนตรีประเภท Keyboard synthesizer โดยใช้เซ็นเซอร์ตรวจจับการสัมผัสของนิ้วมือ

Pitch bend and Vibrato controller for Keyboard synthesizer by using a touch sensor

นพพัชร์ อริยพาษฎ์

NOPPHASIT ARIYAPHASIT

นัทธพงศ์ วัฒนศิริ

NATTAPONG WATTANASIRI

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2557

ระบบควบคุมฟังก์ชัน Pitch bend และ Vibrato สำหรับเครื่องดนตรีประเภท Keyboard synthesizer โดยใช้เซ็นเซอร์ตรวจจับการสัมผัสของนิ้วมือ

Pitch bend and Vibrato controller for Keyboard synthesizer by using a touch sensor

นพสิทธิ์ อริยพสิษฐ์

NOPPHASIT ARIYAPHASIT

นัทธพงศ์ วัฒนศิริ

NATTAPONG WATTANASIRI

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2557

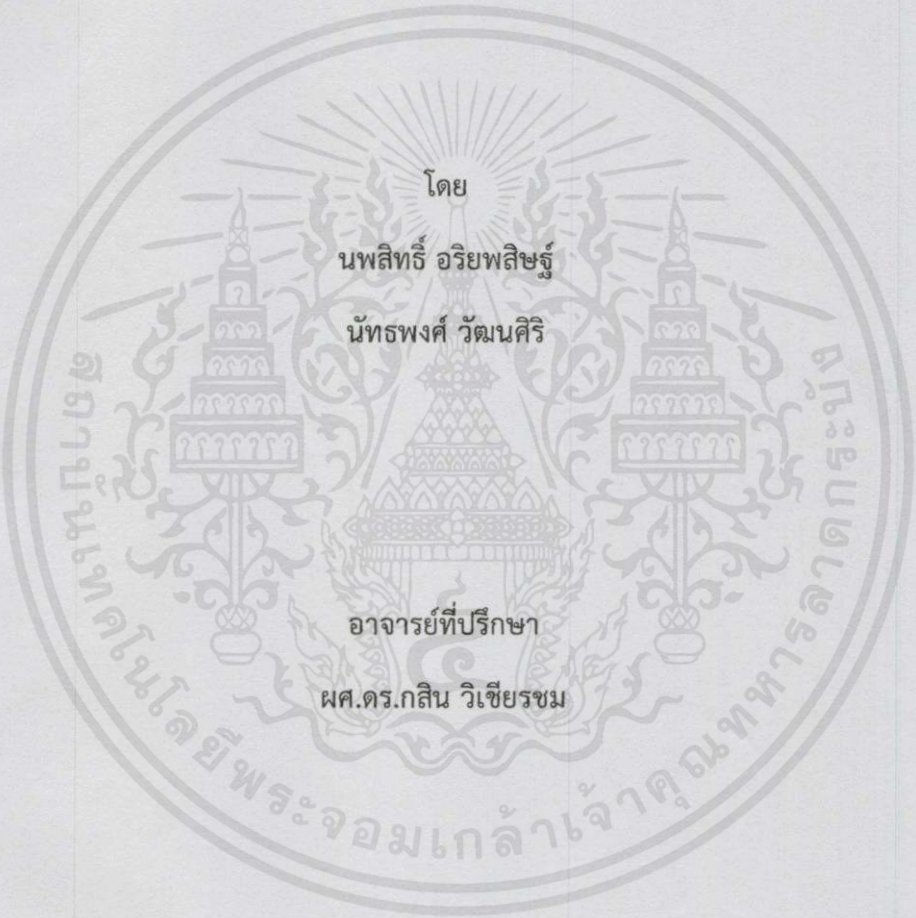
ระบบควบคุมฟังก์ชัน Pitch bend และ Vibrato สำหรับเครื่องดนตรีประเภท Keyboard synthesizer โดยใช้เซ็นเซอร์ตรวจจับการสัมผัสของนิ้วมือ

Pitch bend and Vibrato controller for Keyboard synthesizer by using a touch sensor

โดย

นพสิทธิ์ อริยพสิษฐ์

นัทธพงศ์ วัฒนศิริ



อาจารย์ที่ปรึกษา

ผศ.ดร.กสิน วิเชียรชม

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2557

ปริญญานิพนธ์ปีการศึกษา 2557

สาขาวิชา วิศวกรรมอิเล็กทรอนิกส์

คณะ วิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบควบคุมฟังก์ชัน Pitch bend และ Vibrato สำหรับเครื่องดนตรีประเภท Keyboard synthesizer โดยใช้เซ็นเซอร์ตรวจจับการสัมผัสของนิ้วมือ

Pitch bend and Vibrato controller for Keyboard synthesizer by using a touch sensor

ผู้จัดทำ นายนพสิทธิ์ อริยพสิษฐ์ รหัสประจำตัว 54010664

นายนัทธพงศ์ วัฒนศิริ รหัสประจำตัว 54010693

ปริญญานิพนธ์ฉบับนี้ผ่านการตรวจสอบโดยอาจารย์ที่ปรึกษาแล้ว

(ผศ.ดร.กสิน วิเชียรชม)
อาจารย์ที่ปรึกษา

หัวข้อวิทยานิพนธ์	ระบบควบคุมฟังก์ชัน Pitch bend และ Vibrato สำหรับเครื่องดนตรีประเภท Keyboard synthesizer โดยใช้เซนเซอร์ตรวจจับการสัมผัสของนิ้วมือ	
นักศึกษา	นาย นพสิทธิ์ อริยพสิษฐ์	รหัสประจำตัว 54010664
	นาย นัทธพงศ์ วัฒนศิริ	รหัสประจำตัว 54010693
ปริญญา	วิศวกรรมศาสตรบัณฑิต	
สาขาวิชา	วิศวกรรมอิเล็กทรอนิกส์	
ปีการศึกษา	2557	
อาจารย์ที่ปรึกษาวิทยานิพนธ์	ผศ.ดร. กสิน วิเชียรชม	

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้นำเสนอระบบควบคุมลักษณะทางเสียงของ Pitch Bend และ Vibrato ของเครื่องดนตรีอิเล็กทรอนิกส์ประเภทคีย์บอร์ดซินธิไซเซอร์ (Keyboard Synthesizer) โดยใช้พร็อกซิมิตี้เซ็นเซอร์ชนิดตรวจจับการสัมผัสด้วยการวัดความจุไฟฟ้า (Capacitive Proximity Touch Sensor) ในการตรวจจับตำแหน่งการสัมผัสของนิ้วมือบนพื้นผิวของแผ่นเซ็นเซอร์ เพื่อนำไปสร้างชุดข้อมูล MIDI ที่ควบคุมลักษณะทางเสียงของ Pitch Bend และ Vibrato โดยปกตินั้นคีย์บอร์ดซินธิไซเซอร์จะใช้ล้อคันโยก (Wheel) หรือจอยสติ๊ก (Joystick) ซึ่งสร้างบนพื้นฐานของโพเทนชิโอมิเตอร์ (Potentiometer) ในการควบคุมลักษณะทางเสียงของ Pitch Bend และ Vibrato เมื่อนักดนตรีต้องการจะควบคุมลักษณะเสียงดังกล่าวนั้น นักดนตรีต้องใช้มือหนึ่งในการโยกล้อคันโยกซึ่งจะทำให้นักดนตรีสามารถเล่นคีย์บอร์ดซินธิไซเซอร์ขณะนั้นได้เพียงมือข้างเดียว การใช้ระบบควบคุมลักษณะทางเสียงของวิทยานิพนธ์นี้จะทำให้นักคนตรีสามารถเล่นโน้ตดนตรีและควบคุมลักษณะทางเสียงพร้อมกันได้ด้วยทั้งสองมือเป็นการเพิ่มเทคนิคการเล่นดนตรีให้มีความหลากหลายทางเสียงมากขึ้น

Thesis Title	Pitch bend and Vibrato controller for Keyboard synthesizer by using a touch sensor
Students	Mr. Nopphasit Ariyaphasit Student ID 54010664
	Mr. Natthapong Wattanasiri Student ID 54010693
Graduate Level	Bachelor Degree of Engineering
Department	Electronics Engineering
Academic Year	2014
Advisor	Assist. Prof. Dr. Kasin Vichienchom

Abstract

This thesis presents the system of Pitch bend and Vibrato controller for a keyboard synthesizer using the capacitive proximity touch sensor. The touch sensor is used to measure the position and contact area of fingers on touch sensor's surface which is used to generate the MIDI data format. The MIDI data format will be sent to the digital music instrument to control the Pitch bend and Vibrato effect that corresponds to the movement of the fingers on the touch sensor. For the pitch bend and vibrato controller on the keyboard synthesizers, particularly those based on potentiometer as the wheel and joystick, the keyboardist has to use one hand to control the joystick when using the Pitch bend or Vibrato. As a result, it prevents the keyboardist to play the music freely with both hands. Consequently, the system in this report will make the keyboardist be able to use the Pitch bend and Vibrato without losing one hand. This will helps the musicians to create new ideas and techniques for playing the keyboard synthesizer and to produce new sounds for their musical work.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้และงานวิจัยทั้งหมดสำเร็จลุล่วงได้ด้วยความกรุณาจากอาจารย์ที่ปรึกษา ดร. กสิน วิเชียรชม ที่ให้โอกาศและคำแนะนำในการแก้ปัญหาข้อผิดพลาดและอุปสรรคต่างๆที่เกิดขึ้นระหว่างการดำเนินการจัดทำปริญญานิพนธ์ ตลอดจนเอื้อเฟื้อสถานที่ในการทำวิทยานิพนธ์ จนวิทยานิพนธ์สามารถสำเร็จลุล่วงได้ตามเป้าหมายที่วางไว้ คณะผู้จัดทำขอขอบพระคุณเป็นอย่างสูง

ขอขอบคุณคณะวิศวกรรมศาสตร์ สาขาวิชาอิเล็กทรอนิกส์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังที่ให้ทุนสนับสนุนและเอื้อเฟื้อสถานที่ในการทำวิทยานิพนธ์ ทำให้วิทยานิพนธ์สำเร็จลุล่วงไปได้ด้วยดี

ขอขอบคุณชุมนุมหุ่นยนต์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ได้เอื้อเฟื้ออุปกรณ์และสถานที่ในการทำงาน

สุดท้ายนี้คณะผู้จัดทำขอขอบพระคุณบิดามารดาของคณะผู้จัดทำ ท่านทั้งสองเป็นผู้มีอุปการคุณและเป็นผู้ให้กำลังใจ วิทยานิพนธ์ฉบับนี้จึงสามารถสำเร็จลุล่วงด้วยดี

นาย นพสิทธิ์ อริยพสิษฐ์
นาย นัทธพงศ์ วัฒนศิริ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญรูป

รูปที่ หน้า

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

บทที่ 1

บทนำ

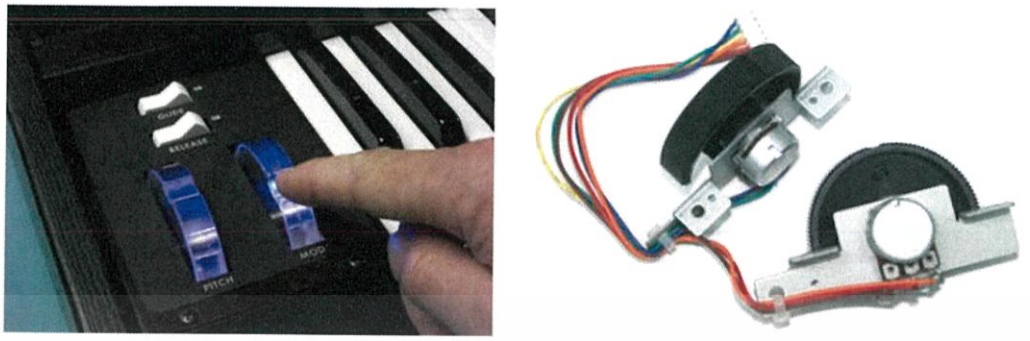
1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบัน เครื่องดนตรีอิเล็กทรอนิกส์ (Electronic music instrument) นั้นได้รับความนิยมมากขึ้น ตัวอย่างการใช้งานเช่น ซินธิไซเซอร์ (Synthesizer) ในการแสดงดนตรีสด เหตุผลเนื่องจากฟังก์ชั่นการทำงานที่หลากหลาย สามารถสร้างรูปแบบของเสียงและเทคนิคการเล่นใหม่ๆซึ่งแตกต่างจากเครื่องดนตรีดั้งเดิม (Acoustic music instrument) ถึงแม้ว่าคุณภาพของเสียงและความไพเราะจากเครื่องดนตรีแบบดั้งเดิมนั้นดีกว่า แต่ปัจจุบันเทคโนโลยีทางด้านอิเล็กทรอนิกส์ได้พัฒนาให้เครื่องดนตรีอิเล็กทรอนิกส์สามารถสร้างเสียงที่มีคุณภาพใกล้เคียงกับเครื่องดนตรีแบบดั้งเดิมมาก

รูปที่ 1.1 ตัวอย่าง Synthesizer ประเภท finger board (Roland V-combo VR-09, Live performance keyboard)

ซินธิไซเซอร์ เป็นเครื่องดนตรีอิเล็กทรอนิกส์ที่ออกแบบมาเพื่อสร้างเสียงจำลองต่างๆในงานดนตรี ซึ่งจะมีส่วนการควบคุม (Control interface) แตกต่างกันออกไป โดยรูปแบบที่เราสนใจในโครงงานนี้เป็นรูปแบบ Finger board แสดงดังรูปภาพที่ 1.1 ซึ่งมีลักษณะเดียวกันที่ใช้บนเปียโนมาตรฐาน สามารถเรียกชื่อเครื่องดนตรีว่า “Keyboard synthesizer”

เสียงโน้ตที่เกิดขึ้นในคีย์บอดซินธิไซเซอร์ นั้นมาจากการประมวลผลของวงจรอิเล็กทรอนิกส์ซึ่งจะรับค่าการกดของลิ่มและเซนเซอร์ต่างๆบนตัวเครื่องดนตรี มาใช้ในการสร้างลักษณะรูปแบบของสัญญาณเสียงตามที่ต้องการ เช่น การเสียงรูปแบบของโน้ตลักษณะ Pitch bend หรือ Vibrato ที่ควบคุมโดยใช้ล้อคันโยก (Wheel) ทางด้านซ้ายมือของตัว Keyboard synthesizer ดังรูปภาพที่ 1.2



รูปที่ 1.2 Pitch wheel และ Mod wheel ที่ใช้ใน Keyboard synthesizer

สำหรับการใช้งานล้อคันโยก Pitch wheel และ Mod wheel นั้น นักดนตรีจำเป็นจะต้องใช้มือหนึ่งข้างซึ่งโดยทั่วไปจะใช้มือซ้ายในการควบคุม ทำให้เกิดข้อจำกัดกรณีที่ ไม่สามารถใช้ฟังก์ชั่น Pitch bend และ Vibrato ด้วยล้อคันโยกได้หากต้องการเล่น Keyboard synthesizer ด้วยสองมือพร้อมกันได้ แสดงดังรูปภาพที่ 1.3

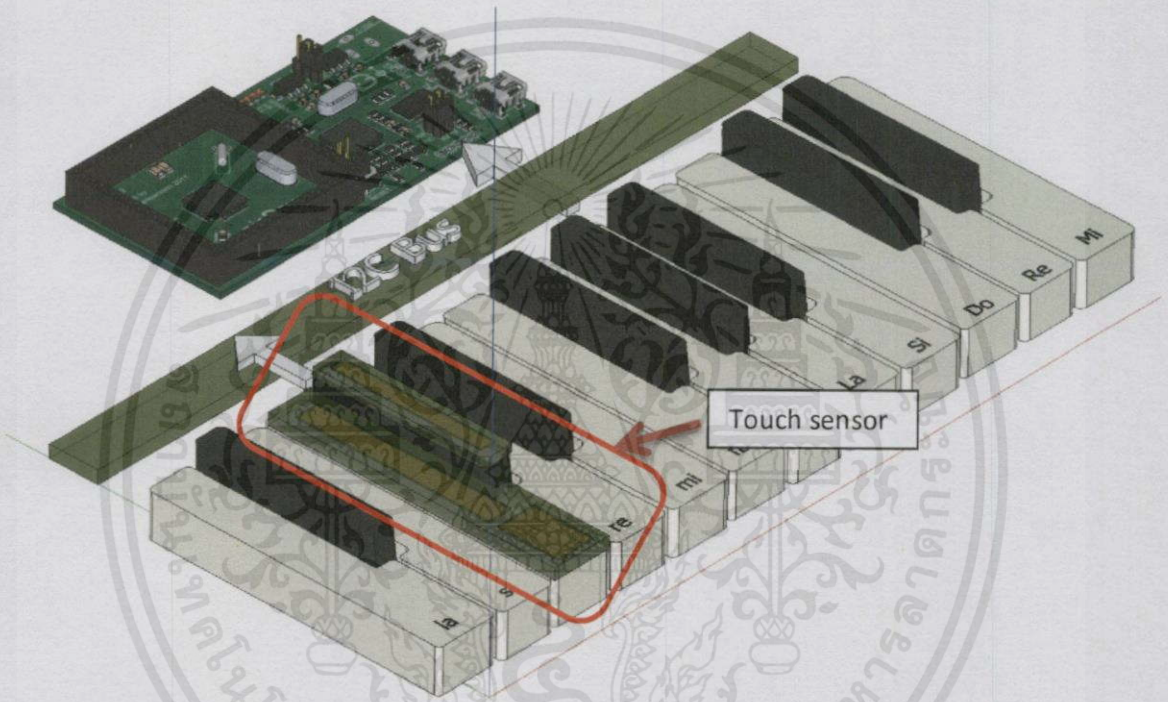
รูปที่ 1.3 ภาพแสดงการใช้ Wheel บน Synthesizer ซึ่งนักดนตรีจำเป็นต้องควบคุมด้วยมือซ้าย

ดังนั้น จึงเกิดแนวคิดที่จะสร้างเซ็นเซอร์ติดตั้งบนคีย์ของเครื่องดนตรีโดยตรง ซึ่งทำหน้าที่แทนการใช้ล้อคันโยกบนซินธิไซเซอร์ โดยการควบคุมจะใช้เทคนิคการสไลด์ของนิ้วบนคีย์ ทำให้สามารถใช้ฟังก์ชั่น Pitch bend ,Vibrato โดยที่มือทั้งสองยังสามารถเล่นโน้ตดนตรีอยู่ได้ สามารถเพิ่มเทคนิคการเล่นซินธิไซเซอร์แก่นักดนตรี ทำให้เกิดความหลากหลายของแนวคิดที่สร้างสรรค์ในการเล่นดนตรีมากขึ้น

1.2 จุดมุ่งหมายและวัตถุประสงค์ของโครงงาน

โครงงานนี้มีวัตถุประสงค์เพื่อนำเสนอแนวคิดการสร้างต้นแบบระบบจำลองควบคุมฟังก์ชั่น Pitch bend และ Vibrato บนเครื่องดนตรีประเภทซินธิไซเซอร์ ด้วยการรูดปลายนิ้วมือบนลิ่มคีย์บอร์ดแทนการใช้ล้อคันโยกที่มีอยู่บนซินธิไซเซอร์ ซึ่งจะทำให้นักดนตรีสามารถใช้งาน Pitch bend และ Vibrato ขณะที่มือทั้งสองยังสามารถเล่นโน้ตอยู่ในเวลาเดียวกันได้

โดยระบบนี้จะเสมือนเป็นเครื่องดนตรีอิเล็กทรอนิกส์ประเภทคีย์บอดซินธิไซเซอร์ ที่ใช้แผ่นเซ็นเซอร์ตรวจจับตำแหน่งการสัมผัสของปลายนิ้ว (Touch sensor) เป็นตัวแทนของคีย์ในแต่ละโน้ต ซึ่งสามารถรับข้อมูลลักษณะการรูดของปลายนิ้วบนลิ่มคีย์ได้และนำมาประมวลผลเพื่อสร้างเสียงของโน้ตที่ถูกเล่นให้เกิดฟังก์ชั่น Pitch bend และ vibrato ให้สอดคล้องตามลักษณะของการรูดนิ้ว ส่วนรูปทรงและขนาดของเซ็นเซอร์ จุดมุ่งหมายสูงสุดคือให้เซ็นเซอร์มีรูปทรงและขนาดที่ใกล้เคียงกับลิ่มคีย์ของเปียโนมาตรฐานมากที่สุด เพื่อให้สามารถติดตั้งบนลิ่มคีย์ของ Keyboard synthesizer ได้ แสดงดังรูปที่ 1.4



รูปที่ 1.4 ภาพแสดงรูปร่างและขนาดที่เหมาะสมที่สุดของ Touch sensor ในระบบ เพื่อให้สามารถติดตั้งบนลิ่มคีย์ของ Synthesizer ได้

1.3 ทฤษฎีและแนวคิดที่ใช้ในการพัฒนา

การควบคุมฟังก์ชั่น Pitch bend และ Vibrato ของระบบนั้น จะใช้พร็อกซิมิตี้เซนเซอร์ชนิดวัดการเปลี่ยนแปลงความจุไฟฟ้า (Capacitance proximity sensor) แทนการใช้ wheel บน keyboard synthesizer แบบดั้งเดิมซึ่งเป็นลักษณะของตัวต้านทางปรับค่าได้ (Potentiometer) การตรวจจับตำแหน่งของปลายนิ้วบนคีย์ซึ่งจะมีไมโครคอนโทรเลอร์ใช้คำนวณตำแหน่งของนิ้วบนเซ็นเซอร์และแปลงเป็นข้อมูลดิจิตอลเพื่อส่งไปยังบอร์ดประมวลผลหลักผ่าน I2C BUS ส่วนบอร์ดประมวลผลหลักจะใช้ในการประมวลผล ทำหน้าที่รับข้อมูลจากเซ็นเซอร์ทุกๆคีย์แปลงให้เป็นรหัส MIDI เพื่อส่งไปยังคอมพิวเตอร์

ที่รองรับการสื่อสาร MIDI ให้แสดงเสียงที่สอดคล้องตามการเล่น Keyboard ของผู้ใช้งาน โดยลักษณะเสียงจะถูกการใช้งาน Vibrato, Pitch bend ตามตำแหน่งและรูปแบบการเคลื่อนที่ของปลายนิ้ว

1.4 ขอบเขตของโครงงาน

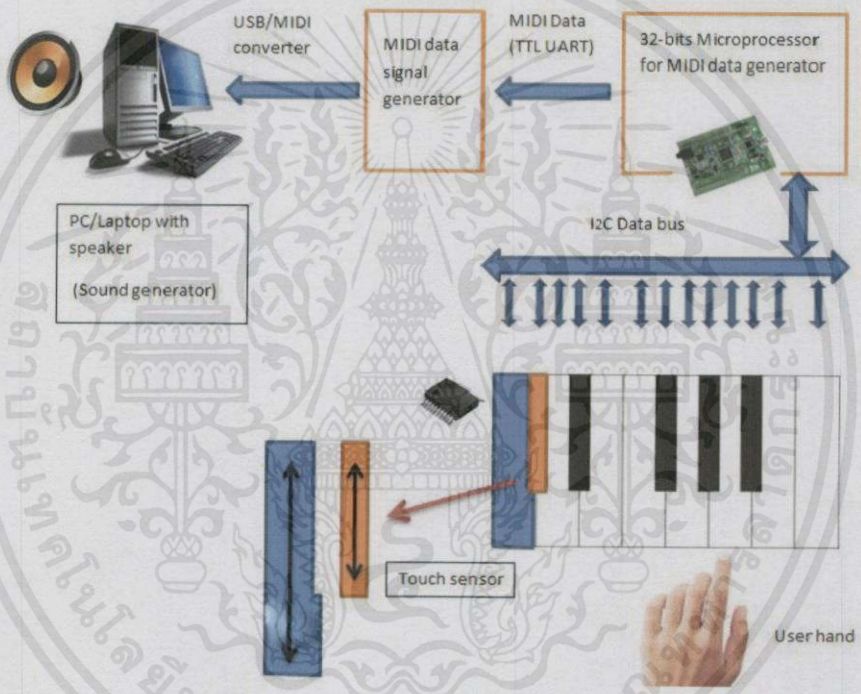
ระบบควบคุมฟังก์ชั่น Pitch bend และ Vibrato บนเครื่องดนตรีประเภท Keyboard synthesizer ในโครงงานนี้จะประกอบรายละเอียดต่างๆดังนี้

- ประกอบด้วยพร็อกซิมิตี้เซ็นเซอร์ตรวจจับตำแหน่งการสัมผัสของปลายนิ้ว(Touch sensor) ที่ใช้แทนตำแหน่งคีย์ในแต่ละโน้ต จำนวน 13 ชิ้น เรียงกันด้วยความยาวช่วงเสียงโน้ต 1 Octave โดยขนาดของเซ็นเซอร์จะออกแบบให้มีขนาดเท่ากับคีย์ขาว (White key) เท่านั้น
- เทคนิคที่ใช้ในเซนเซอร์จะใช้หลักการตรวจจับความจุไฟฟ้า(Capacitive sensing) ของนิ้วมือ โดยถูกออกแบบในรูปแบบแถบรูดนิ้ว (Slider) สามารถระบุตำแหน่งของนิ้วได้ 1 มิติตามแนวยาว ความละเอียดสูงสุดอยู่ที่ 32 ระดับต่อความยาวของแถบรูด
- ระบบจะไม่สามารถวัดความเร็วของการกดของโน้ต(Touching Velocity) ที่เป็นตัวกำหนดความดังเบาของเสียงโน้ตที่ถูกเล่น ซึ่งไม่ได้อยู่ในขอบเขตของการศึกษา
- ข้อมูลตำแหน่งการสัมผัสของปลายนิ้วของทุกเซนเซอร์จะรวบรวมเข้าสู่บอร์ดประมวลผลหลักผ่านการเชื่อมต่อแบบ BUS เพื่อสร้างเสียงโน้ตที่ถูกเล่นให้มีฟังก์ชั่น Pitch bend และ Vibrato ที่สอดคล้องกับรูดของนิ้วมือบนเซนเซอร์
- การแสดงเสียงโน้ตที่ถูกเล่นจะใช้การเชื่อมต่อระหว่างระบบที่สร้างในโครงงานกับอุปกรณ์ทางออกมาตรฐาน MIDI (Musical Instrument Digital Interface) ผ่านสาย MIDI โดยตัวอย่างของอุปกรณ์ทางออกมาตรฐาน MIDI เช่น อิเล็กทรอนิกส์คีย์บอร์ด, คอมพิวเตอร์ เป็นต้น

บทที่ 2

ทฤษฎีและหลักการ

2.1 Block diagram แสดงการทำงานของระบบ



รูปที่ 2.1 Block diagram ของระบบเซ็นเซอร์ควบคุมฟังก์ชัน Pitch bend, Vibrato สำหรับเครื่องดนตรีประเภท Keyboard synthesizer

สำหรับคีย์บอร์ดซินธิไซเซอร์ (Keyboard synthesizer) ทั่วไปแล้วจะรับสัญญาณจากล้อคันโยกเพื่อใช้ในการสร้างฟังก์ชั่น Pitch bend และ Vibrato แต่สำหรับระบบของโครงงานนี้จะใช้ข้อมูลตำแหน่งนิ้วการรูดบนบอร์ดเซ็นเซอร์มาใช้ในการสร้างฟังก์ชั่น Pitch bend แทน

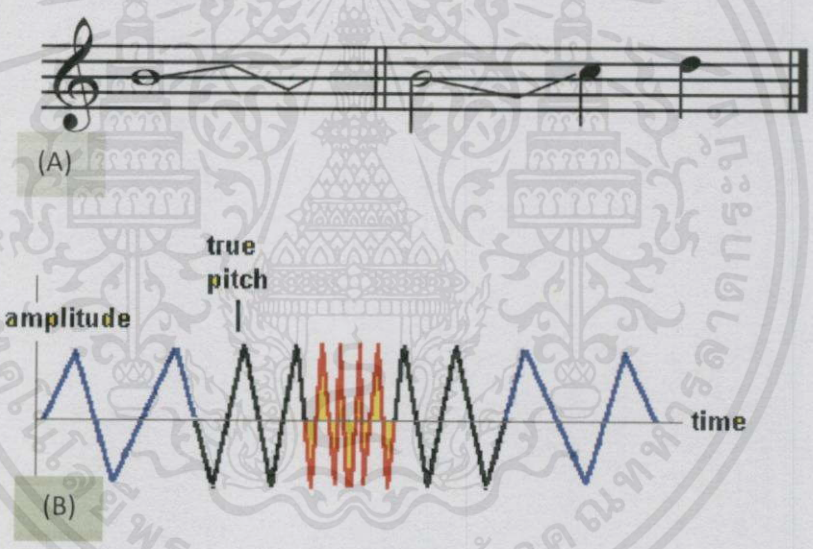
ซึ่งการตรวจจับตำแหน่งของนิ้วสามารถทำได้โดยใช้ Touch sensor ซึ่งใช้หลักการของพร็อกซิมิตี้เซ็นเซอร์ชนิดวัดความจุไฟฟ้า (Capacitive proximity sensor) ซึ่งออกแบบในลักษณะสไลเดอร์โดยใช้ไมโครคอนโทรลเลอร์ในการระบุตำแหน่งนิ้วที่สัมผัสบน Touch sensor และแปลงตำแหน่งของนิ้วเป็นข้อมูลดิจิตอลเพื่อส่งไปยังบอร์ดประมวลผลหลักผ่านบัส I2C ส่วนบอร์ดประมวลผลหลักที่ทำหน้าที่สร้าง

ชุดข้อมูล MIDI โดยเรียกบอร์ดนี้ว่า MIDI data generator ซึ่งจะใช้ไมโครคอนโทรเลอร์ที่มีความสามารถในการคำนวณสูงเพื่อรับข้อมูลจากแต่ละ Touch sensor และทำการแปลงข้อมูลที่ได้รับให้กลายเป็น MIDI data ระดับแรงดัน TTL และส่งผ่านข้อมูลไปยังบอร์ด MIDI data signal generator เพื่อแปลง MIDI data (TTL level) ให้เป็น MIDI ที่สัญญาณไฟฟ้าเป็นลักษณะ current loop ซึ่งสามารถนำไปเชื่อมต่อกับเครื่องดนตรีอื่นๆหรืออุปกรณ์กำเนิดเสียงที่ใช้ระบบการสื่อสารแบบ MIDI

2.2 Pitch bend และ Vibrato

2.2.1 Pitch bend

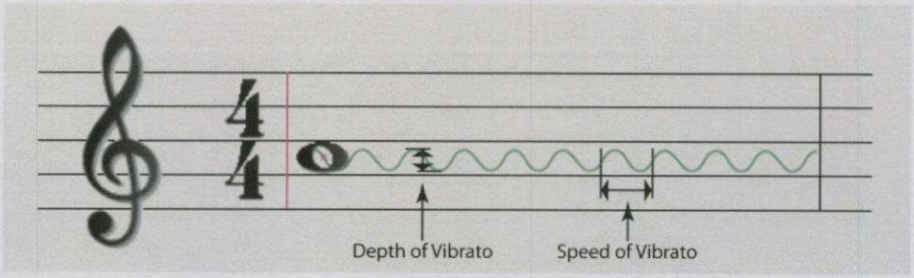
Pitch bend คือ ฟังก์ชันที่เปลี่ยนความถี่ของตัวโน้ตที่กำลังถูกเล่นอยู่ให้มีค่าเพิ่มขึ้นหรือลดลงไม่เกินหนึ่งเสียงเต็มของโน้ต โดยความถี่ของเสียงโน้ตจะเปลี่ยนไปอย่างต่อเนื่องจากความถี่โน้ตเริ่มต้น สามารถอธิบายจากสัญลักษณ์ทางดนตรีสากลดังรูปที่ 2.2



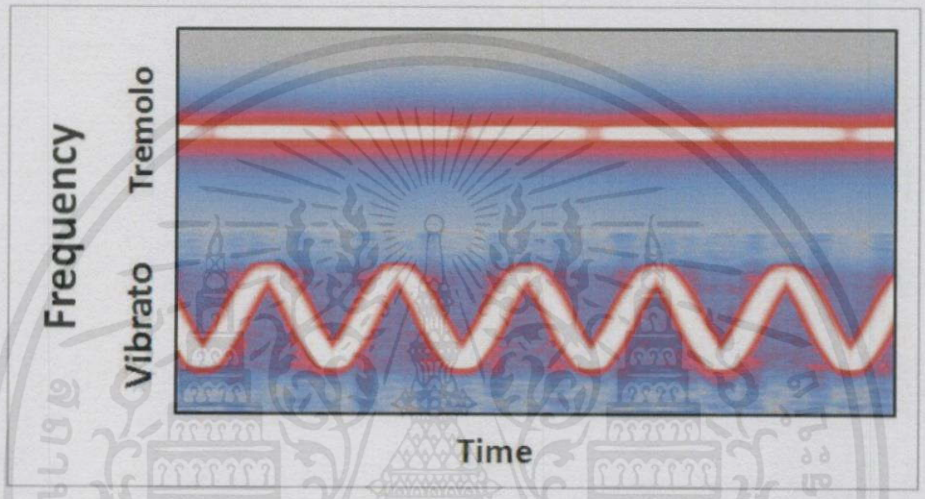
รูปที่ 2.2 (A) ภาพแสดงการอธิบายลักษณะของเสียงโน้ต Pitch bend ด้วยบรรทัด 5 เส้นของดนตรีสากล , (B) ภาพแสดงการอธิบายลักษณะของเสียงโน้ต Pitch bend ด้วย Time domain- waveform ของเสียงโน้ต

2.2.2 Vibrato

Vibrato คือ ฟังก์ชันที่ทำให้ความถี่ของโน้ตที่เล่นอยู่เปลี่ยนแปลงขึ้นลงเล็กน้อยอย่างเป็นคาบโดยการควบคุมฟังก์ชั่น จะใช้วิธีปรับคาบในการสั่นของความถี่โน้ต เพื่อให้ได้ลักษณะการสั่นของเสียงโน้ตที่เปลี่ยนไป แสดงการทำงานของฟังก์ชั่น Vibrato ดังรูปที่ 2.3



(A)



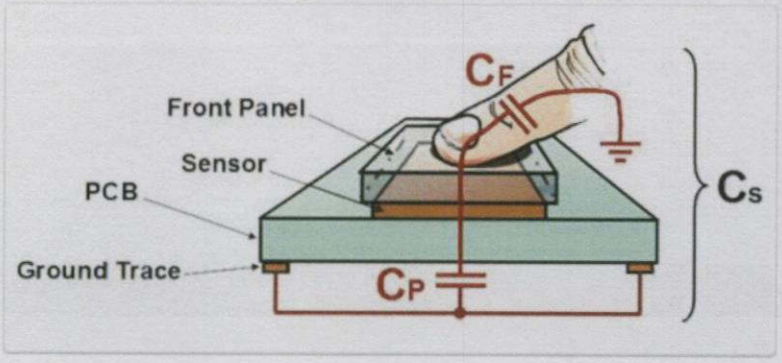
(B)

รูปที่ 2.3 (A) ภาพแสดงการอธิบายลักษณะของเสียงโน้ต Vibrato ด้วยบรรทัด 5 เส้นของดนตรีสากล, (B) ภาพแสดงการอธิบายลักษณะของเสียงโน้ต Vibrato ด้วยกราฟ Spectrogram

2.3 พร็อกซิมิตี้เซ็นเซอร์ชนิดความจุไฟฟ้า

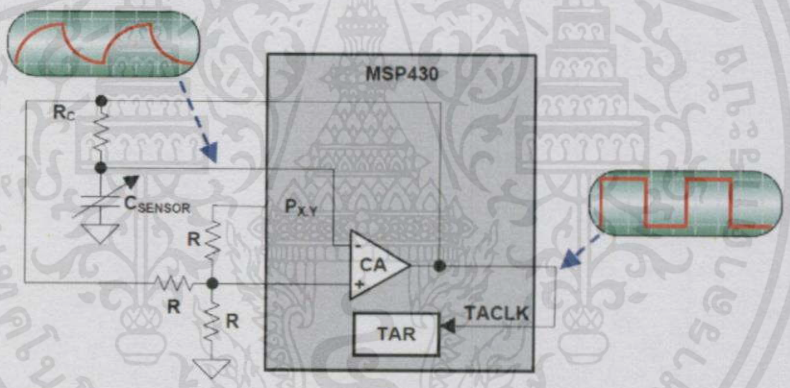
พร็อกซิมิตี้เซนเซอร์ชนิดความจุไฟฟ้า ทำงานโดยอาศัยหลักการเปลี่ยนแปลงความจุไฟฟ้าแฝง จากรูปที่ 2.4 แสดงแบบจำลองของพร็อกซิมิตี้เซนเซอร์ชนิดความจุไฟฟ้า เมื่อนิ้วมือเคลื่อนที่เข้ามาใกล้สนามไฟฟ้าที่กำเนิดโดยเซ็นเซอร์และแผ่นกราวด์จะถูกรบกวนและก่อให้เกิดค่าความจุไฟฟ้าแฝงขนานกับค่าความจุไฟฟ้าของเซ็นเซอร์ ส่งผลให้ค่าความจุไฟฟ้ารวมของเซ็นเซอร์มีค่าเพิ่มขึ้นมีค่าดังสมการที่ 2.1

$$C_{Sensor} = C_{Parasitic} + C_{Finger} \tag{2.1}$$



รูปที่ 2.4 พร็อกซิมิตี้เซ็นเซอร์ชนิดความจุไฟฟ้า

หลักการที่ใช้วัดค่าความจุไฟฟ้าที่ใช้งานสำหรับพร็อกซิมิตี้เซ็นเซอร์นั้นมีหลายวิธี ตัวอย่างเช่น การใช้วงจร Relaxation oscillator ในการตรวจจับการเปลี่ยนแปลงของความจุไฟฟ้า การใช้เทคนิคตรวจจับสัญญาณ Ramp จากการชาร์จประจุด้วยแหล่งจ่ายกระแสคงที่ หรือเทคนิคการถ่ายโอนประจุ (Charge transfer) เป็นต้น



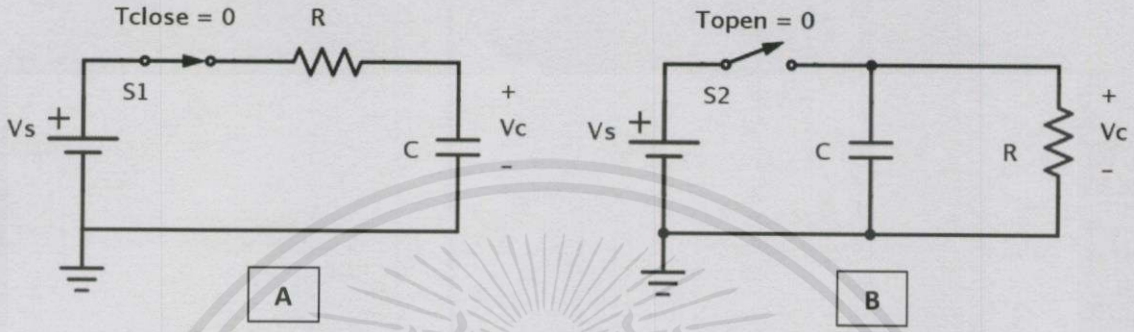
รูปที่ 2.5 ตัวอย่างการใช้วงจร Relaxation oscillator ในการตรวจจับการเปลี่ยนแปลงของความจุไฟฟ้าสำหรับสำหรับพร็อกซิมิตี้เซ็นเซอร์ตรวจจับสัมผัส

แต่เนื่องจากวงจรอิเล็กทรอนิกส์จากวิธีเหล่านี้ มีจำนวนอุปกรณ์อิเล็กทรอนิกส์ต่อการวัดพร็อกซิมิตี้เซนเซอร์ 1 ชิ้นค่อนข้างมาก จึงไม่เหมาะสมสำหรับการนำไปใช้งานบนเซ็นเซอร์ตรวจจับสัมผัสที่มีขนาดเล็กเท่ากับลิ่มของเปียโนได้ ดังนั้นสำหรับโครงงานนี้จะใช้หลักการตรวจจับการคายประจุของวงจร RC (Resistor-base Capacitive measurement) ซึ่งจะอธิบายการทำงานของวงจรในหัวข้อที่ 2.3.1

2.3.1 การชาร์จและคายประจุของวงจร RC

RC time constant มีอีกชื่อหนึ่งก็คือ “tau” เป็นช่วงเวลาที่วงจร RC ชาร์จประจุ (Charge) เข้าตัวเก็บประจุ C ผ่านตัวต้านทาน R จนความต่างศักย์ของตัวเก็บประจุมีค่าอยู่ที่ประมาณ 63.2 เปอร์เซ็นต์

ของแหล่งจ่ายหรือช่วงเวลาที่วงจร RC คายประจุ (Discharge) ที่ตัวเก็บประจุ C จนความต่างศักย์คร่อมตัวเก็บประจุมีค่าประมาณ 36.8 เปอร์เซ็นต์ของแหล่งจ่าย แสดงวงจร RC ที่ใช้อธิบายดังกล่าวในรูปที่ 2.6 และแสดงกราฟการชาร์จและคายประจุของวงจร RC ในรูที่ 2.7



รูปที่ 2.6 (A) วงจร RC แบบชาร์จประจุ, (B) วงจร RC แบบคายประจุ

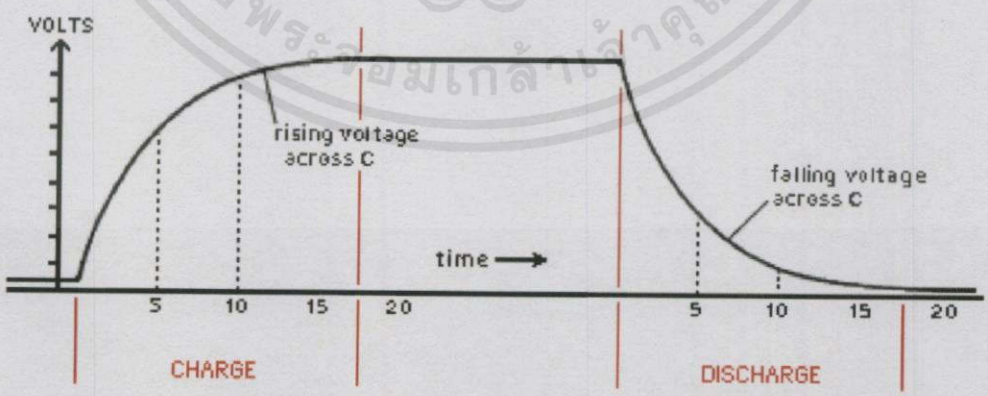
$$\tau = RC \tag{2.2}$$

สมการคายประจุ

$$Vc(t) = V_s(1 - e^{-t/\tau}) \tag{2.3}$$

สมการชาร์จประจุ

$$Vc(t) = V_s(e^{-t/\tau}) \tag{2.4}$$



รูปที่ 2.7 กราฟการชาร์จและคายประจุของวงจร RC

ดังนั้นหากสามารถตรวจจับการเปลี่ยนแปลงของเส้นกราฟการชาร์จและคายประจุขณะที่ค่าของแหล่งจ่ายแรงดัน Vs และตัวต้านทาน R ไม่เปลี่ยนแปลง ก็จะสามารถตรวจจับการเปลี่ยนแปลงของค่าความจุไฟฟ้าแฝงของพร็อกซิมิตี้เซนเซอร์ได้

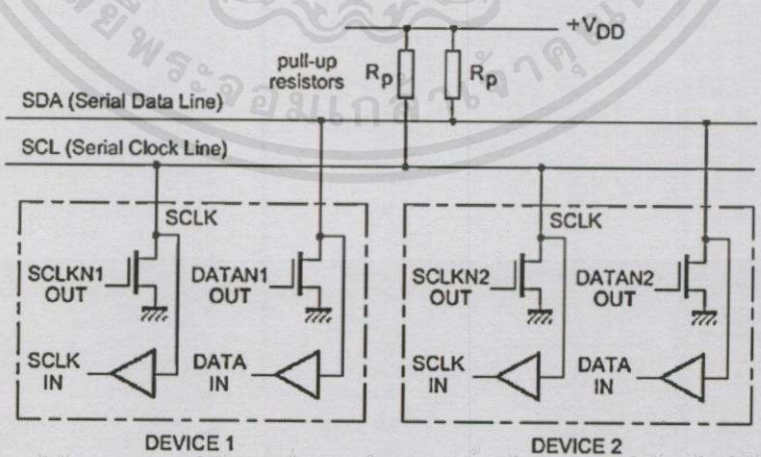
2.4 I2C

ย่อมาจาก Inter Integrate Circuit Bus (IIC) นิยมเรียกสั้นๆว่า I^2C (ไอ-แสควร์-ซี) หรือ I2C (ไอ-ทู-ซี) เป็นมาตราฐานการสื่อสารอนุกรมแบบซิงโครนัส (Synchronous) รูปแบบของบัสโดยใช้สายข้อมูลเพียง 2 เส้น (TWI : Two-Wire-Interface) ถูกพัฒนาขึ้นโดยบริษัท Philips Semiconductors สามารถใช้ติดต่อสื่อสารข้อมูลระหว่างอุปกรณ์ที่รองรับของ I2C กันเอง เช่น ระหว่างไมโครคอนโทรลเลอร์ (MCU) กับชิพที่ใช้การเชื่อมต่อแบบ I2C สายสัญญาณประกอบด้วย serial data (SDA) และ serial clock (SCL) ความเร็วของการสื่อสารอยู่ที่ 100 Kbit/s ในโหมดปกติ และ 400 Kbit/s ในโหมดความเร็วสูง

เหตุผลที่เลือกใช้บัส I2C ในการรับ-ส่งข้อมูลระหว่าง Touch sensor กับบอร์ดประมวลผลหลัก เพราะเป็นการสื่อสารแบบอนุกรมที่เชื่อมต่อโดยใช้จำนวนสายข้อมูลเพียง 2 สาย คือ SDA กับ SDL ในขณะที่สามารถเชื่อมต่ออุปกรณ์ได้จำนวนมาก (สูงสุด 128 อุปกรณ์ใน 1 บัส) ซึ่งเหมาะสมกับการเชื่อมต่อเข้ากับจำนวนบอร์ดเซ็นเซอร์ในระบบของโครงงานที่มีจำนวนมากถึง 13 บอร์ดเซนเซอร์

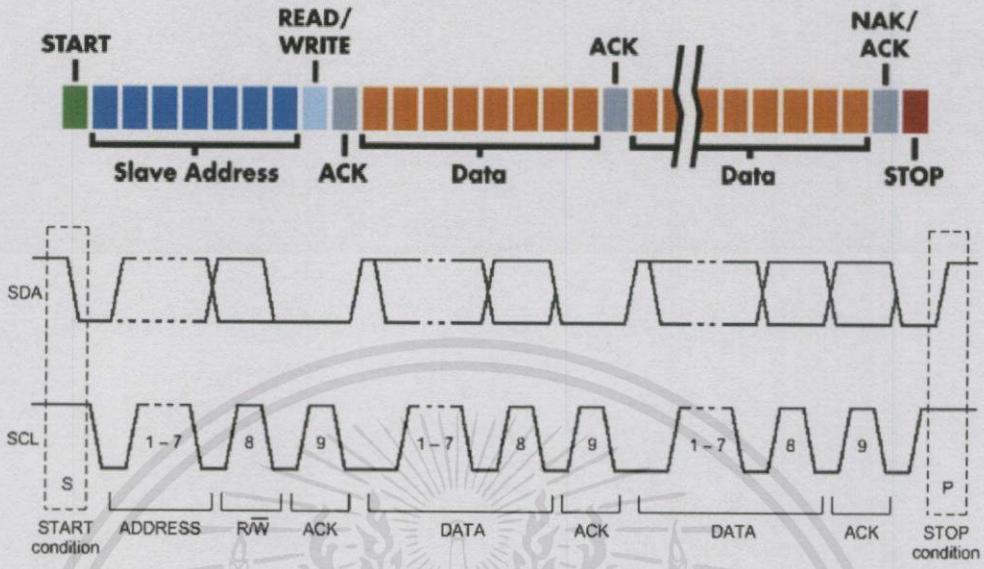
2.4.1 การเชื่อมต่อทางฮาร์ดแวร์ของอุปกรณ์แบบบัส I2C

บัส I2C ใช้สายสัญญาณ 2 เส้น คือ SCL, SDA สำหรับติดกับอุปกรณ์แบบ 2 ทิศทาง โดยที่ขาสัญญาณทั้ง 2 จะต้องต่อกับตัวต้านทาน Rp ลักษณะแบบ pull up ดังรูปที่ 2.8 ค่า Rp ประมาณ 10KOhm เนื่องจากเอาต์พุตมีลักษณะเป็น แบบ Open Drain หรือเป็นแบบ Open Collector การเชื่อมต่ออุปกรณ์หลายๆตัวเข้ากับบัสต้องไม่ทำให้ค่าความจุรวมของบัสเกิน 400 pF เพื่อมิให้แบนวิธ (Bandwidth) ของบัสต่ำลงจนไม่สามารถสื่อสารที่ความเร็ว 100Kbit/s ได้



รูปที่ 2.8 วงจรภายในอุปกรณ์บนบัส I2C และการเชื่อมต่อระหว่างอุปกรณ์บนบัส

2.4.2 โปรโตคอล (Protocol) ของการสื่อสารแบบ I2C



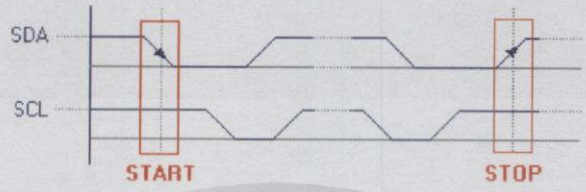
รูปที่ 2.9 ภาพแสดง Timing diagram ของ 1 ชุดข้อความที่ใช้ในโปรโตคอล I2C

อุปกรณ์ที่เชื่อมต่อบนบัส I2C ถูกแบ่งออกเป็นสองลักษณะ คือ Master และ Slave บัสนั้นสามารถมี Master และ Slave ได้มากกว่าหนึ่งซึ่งเป็นลักษณะที่เรียกว่า "Multi-master, Multi-slave" ทั้ง Master และ Slave จะสามารถเป็นได้ทั้งอุปกรณ์ส่งข้อมูล (Transmitter) หรืออุปกรณ์รับข้อมูล (Receiver) การสื่อสารบนบัสถูกควบคุมโดย Master ซึ่งจะควบคุมจังหวะในการรับ-ส่งข้อมูลระหว่าง Master กับ Slave ความเร็วของบัสจะถูกกำหนดโดย Master

เมื่อบัสอยู่ในสถานะว่าง จึงจะสามารถเริ่มการสื่อสารได้ โดยลำดับขั้นตอนจะอธิบายได้ดังนี้

1. Master จะส่งสถานะเริ่มต้น (START Conditions) เพื่อแสดงการขอใช้บัส
2. จากนั้น Master จะส่งรหัสควบคุม (Control Byte) ซึ่งประกอบด้วย รหัสประจำตัวของ Slave ที่ต้องการติดต่อ (Slave Address) และโหมดการใช้งานบัส (Mode) ซึ่งมีด้วยกัน 2 แบบคือ การเขียนข้อมูล(Write) ลงบน Slave หรืออ่านข้อมูล (Read) จาก Slave
3. เมื่อ Slave รับทราบว่า Master ต้องการที่จะติดต่อด้วย ก็จะส่งสถานะรับรู้ (Acknowledge) แจ้งให้ Master ทราบว่าข้อมูลที่ได้ส่งไปมีการรับรู้และ Slave ที่รับรู้นั้นเตรียมความพร้อมสำหรับการสื่อสารช่วงต่อไปแล้ว
4. หากบัสทำงานในโหมดเขียน Master จะส่งไบต์ข้อมูล (Data) ที่ใช้เขียนลงบนบัสเพื่อให้ Slave ทำการอ่านและบันทึก หากบัสทำงานในโหมดอ่าน Slave จะส่งไบต์ข้อมูลที่ Master ต้องการลงบนบัสเพื่อให้ Master ทำการอ่านและบันทึก จำนวนไบต์ของชุดข้อมูลนั้นขึ้นอยู่กับการตั้งค่าบัส ซึ่งสามารถมีจำนวนได้มากกว่า 1 ไบต์ และเมื่อสิ้นสุดการส่งข้อมูลทุกครั้งในแต่ละไบต์ จะต้องมีการ Acknowledge จากอุปกรณ์ที่เป็นอุปกรณ์รับข้อมูล
5. และเมื่อสิ้นสุดการส่งข้อมูล Master จะต้องส่งสถานะสิ้นสุด (STOP Conditions) เพื่อบอกกับอุปกรณ์ในบัสว่าสิ้นสุดการใช้บัสแล้ว

สถานะบัสว่าง คือสถานะที่บัสไม่ได้ถูกใช้งานทั้ง SCL และ SDA จะเป็น logic 1ทั้งคู่ การกำหนดสถานะเริ่มต้นและสถานะสิ้นสุดของ **I2C BUS** (START and STOP Conditions) ซึ่งอธิภาพด้วยภาพดังรูปที่ 2.10



รูปที่ 2.10 ลักษณะของสัญญาณ START and STOP Conditions บน I2C

สถานะเริ่มต้น (START Conditions) คือให้ SDA เปลี่ยนจาก logic 1 มาเป็น logic 0 ในขณะที่ SCL มีค่าเป็น logic 1

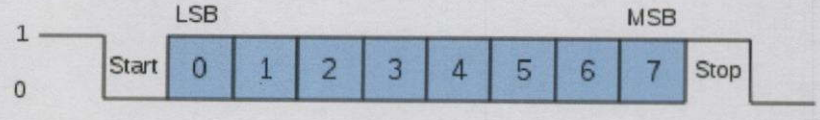
สถานะสิ้นสุด (STOP Conditions) คือให้ SDA เปลี่ยนจาก logic 0 มาเป็น logic 1 ในขณะที่ SCL มีค่าเป็น logic 1

2.5 UART

ย่อมาจาก Universal Asynchronous Receiver Transmitter (UART) เป็นมาตรฐานการสื่อสารอนุกรมแบบ Asynchronous โดยการส่งข้อมูลที่ไม่ต้องใช้สัญญาณนาฬิกามาเป็นตัวกำหนดจังหวะการรับส่งข้อมูลแต่ ใช้วิธีกำหนดรูปแบบการรับส่งข้อมูลขึ้นให้เหมือนกันทั้งอุปกรณ์รับและอุปกรณ์ส่งแทน คือ อาศัยการกำหนดความเร็วของบิตข้อมูล(Bit rate)หรือความเร็วของสัญญาณข้อมูล(Baud rate) ของการรับส่งให้เท่ากันทั้งฝั่งรับและฝั่งส่ง

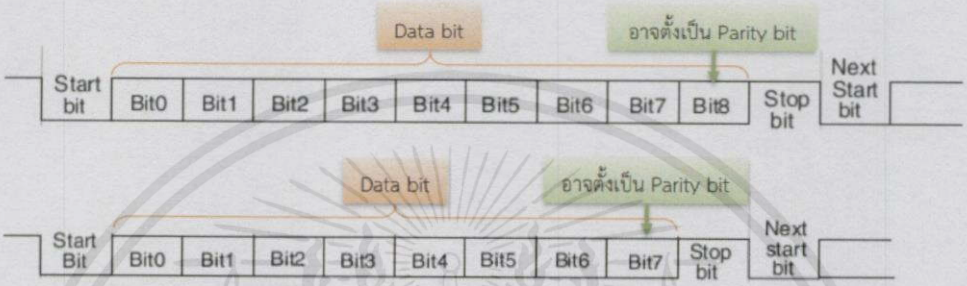
ข้อดีของการใช้ Asynchronous คือสามารถสื่อสารแบบ Full Duplex รับ และ ส่งได้ในเวลาเดียวกัน แต่ Asynchronous มีโอกาสที่ข้อมูลจะสูญหายขณะรับส่งข้อมูล หรือ รับส่งข้อมูลผิดพลาดได้มากกว่าแบบ Synchronous

สรุปกล่าวคือ UART (Universal Asynchronous Receiver Transmitter) หมายถึง รูปแบบการส่งข้อมูล ที่ถูกกำหนดขึ้นมาเพื่อใช้รับส่งข้อมูลแบบ Asynchronous โดยมีรูปแบบดังรูปที่ 2.11



รูปที่ 2.11 ภาพแสดง Timing diagram ของ 1 ชุดข้อความที่ใช้ในโปรโตคอล UART (ไม่มี Parity bit)

การสื่อสารจะเริ่มต้นจากอุปกรณ์ที่เป็นตัวส่งข้อมูลจะให้สัญญาณ Start Bit เป็น Logic 0 จากนั้นจะตามด้วยชุดข้อมูลที่ต้องการส่งซึ่งจำนวนของบิตขึ้นอยู่กับการเลือก มีตั้งแต่ 8-9 บิต แต่โดยทั่วไปจะสื่อสารด้วยข้อมูลขนาด 1 ไบต์ จากนั้นจะส่งบิตตรวจสอบความผิด (Parity bit) ดังรูปที่ 2.12 แต่หากใช้งานในโหมดไม่ใช้บิตตรวจสอบความผิดพลาดเมื่อส่งข้อมูลครบแล้วถูกปิดด้วย STOP Bit เป็น Logic 1 ได้เลย



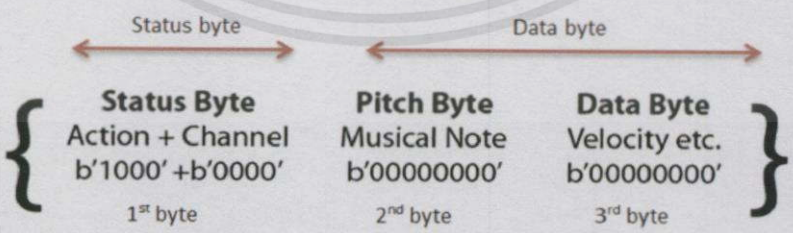
รูปที่ 2.12 ภาพแสดง Timing diagram ของโปรโตคอล UART (มี Parity bit)

2.6 MIDI

MIDI มาจากคำว่า Musical Instrument Digital Interface เป็น Protocol สำหรับการสื่อสารข้อมูลดิจิตอลของเครื่องดนตรีอิเล็กทรอนิกส์ โดยชุดข้อมูลนั้นจะประกอบด้วยสถานะต่างๆของโน้ตที่กำลังเล่นอยู่ เช่น สถานะของการเล่น (Event) ตำแหน่งของโน้ต (Pitch) ความเร็วของการเล่นโน้ต (Velocity) เป็นต้น

2.6.1 โปรโตคอลสำหรับการสื่อสารของ MIDI

MIDI โดยทั่วไปจะใช้การสื่อสารอนุกรมแบบ asynchronous mode (UART 8 bits, no parity bit, 1stop bit) ด้วยความเร็วของการส่งข้อมูล 31250 bps โดย 1 ชุดคำสั่งของ MIDI ที่ใช้งานทั่วไปจะมีความยาว 3 bytes ประกอบด้วย 1 Status byte และ 2 Data byte



รูปที่ 2.13 รูปแบบชุดคำสั่งของ MIDI

Status byte ประกอบด้วยข้อมูล 2 ส่วน ส่วนละ 4 บิต โดย 4 บิตแรกนั้นระบุถึงช่องการเชื่อมต่อของอุปกรณ์ MIDI (MIDI channel) ซึ่งสามารถรองรับการเชื่อมต่อได้สูงสุด 16 ช่อง ส่วน 4 บิตหลังเป็นคำสั่งควบคุม เช่น ให้เสียงโน้ตดัง(Note on), หยุดดัง(Note off) หรือคำสั่ง Pitch wheel เป็นต้น

Data byte ประกอบด้วยข้อมูล 2 ส่วน ส่วนละ1 ไบต์ คือ Pitch byte และ Data byte, Pitch byte นั้นใช้ระบุตำแหน่งของโน้ต เช่น C3, F#4, Bb3 เปรียบดังการเปลี่ยนความถี่ของสัญญาณเสียง ส่วน Data byte จะระบุขนาดของค่าที่ใช้ในคำสั่งจาก Status byte เช่น ความเร็วของการกดคีย์ (Velocity) ซึ่งใช้กำหนดความดังของโน้ตที่ถูกเล่น เปรียบดังการเปลี่ยนแปลงขนาดของสัญญาณเสียง เป็นต้น

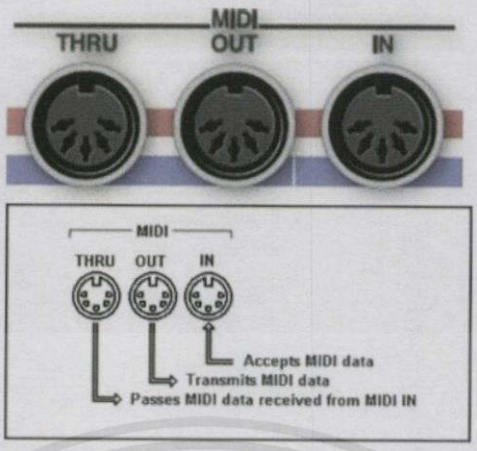
STATUS	2nd Byte	3rd Byte	Description
1000cccc	0nnnnnnn	0vvvvvvv	Note off, 'cccc' is channel, 'nnnnnnn' note value''vvvvvvv' is velocity value
1001cccc	0nnnnnnn	0vvvvvvv	Note on, 'cccc' is channel, 'nnnnnnn' note value''vvvvvvv' is velocity value
1011cccc	0nnnnnnn	0vvvvvvv	Control Change, 'cccc' is channel,'nnnnnnn' is Controller Number,'vvvvvvv' is value
1110cccc	0LLLLLLL	0mmmmmmm	Pitch Wheel Change,'cccc' is channel, 'LLLLLLL' is LSB of value 'mmmmmmm' is MSB of valuecenter (no pitch change) is 0x2000
11111000	none	none	MIDI Clock pulse, 24 pulses per quartemote, 96 to the bar (assuming 4/4 time)
11111010	none	none	Start,sent at start of sequence, followed by *MIDI clock pulses*
11111011	none	none	Continue,sequence carries on from where it was stopped
11111100	none	none	Stop,sent when sequence is stopped
11111110	none	none	Active sensing

รูปที่ 2.14 ตารางแสดงตัวอย่างชุดคำสั่งของ MIDI

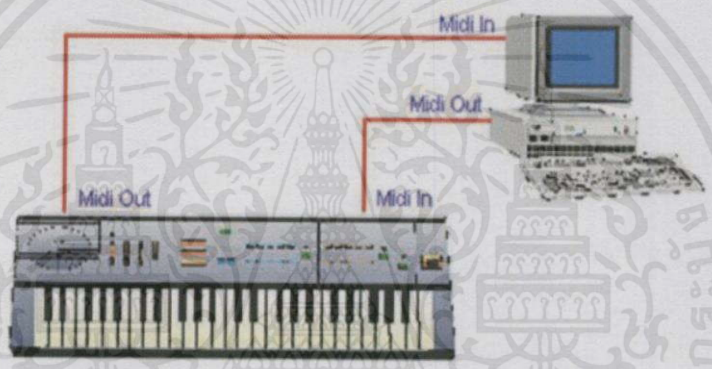
กรอบสีแดงในรูปที่ 2.14 คือชุดคำสั่งที่ใช้งานฟังก์ชัน Pitch Bend โดยสำหรับคีย์บอร์ดซินธิไซเซอร์ทั่วไปแล้วจะรับสัญญาณจากล้อคันโยก แต่สำหรับระบบของโครงงานนี้ บอร์ดประมวลผลหลักจะนำข้อมูลตำแหน่งนิ้วที่ได้จากบอร์ดเซ็นเซอร์มาใช้ในการสร้างฟังก์ชั่น Pitch Bend แทน

2.6.2 การเชื่อมต่อทางฮาร์ดแวร์ของอุปกรณ์ MIDI

การเชื่อมต่อระหว่างอุปกรณ์ MIDI จะใช้พอร์ตซึ่งจะมีประกอบด้วย 3 พอร์ต ได้แก่ MIDI IN, MIDI OUT, MIDI Thru ซึ่งแสดงไว้ดังรูปที่ 2.15



รูปที่ 2.15 พอร์ต MIDI IN, MIDI OUT, MIDI Thru

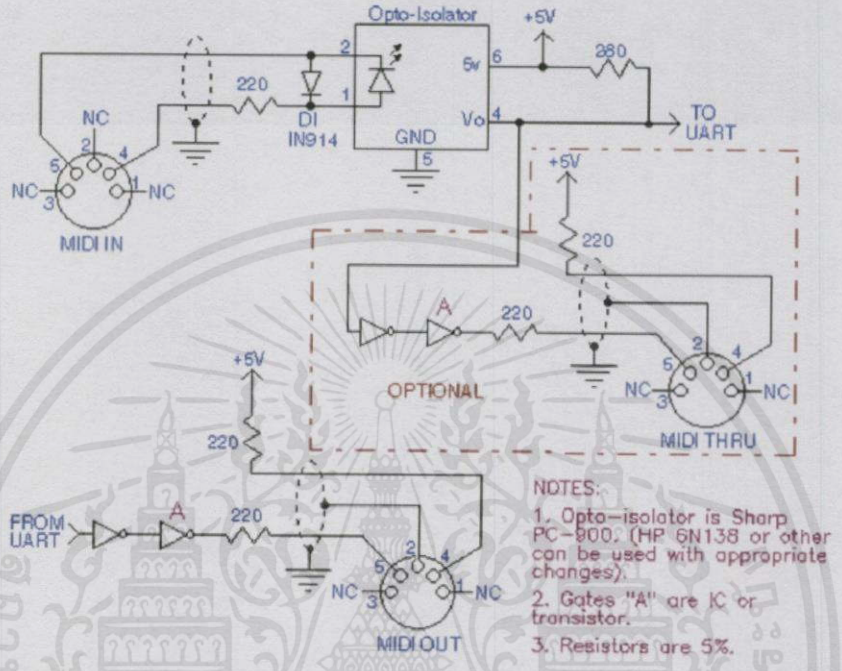


รูปที่ 2.16 ตัวอย่างการเชื่อมต่ออุปกรณ์ MIDI ระหว่างอุปกรณ์ 2 ชิ้น

MIDI จะสื่อสารข้อมูลลักษณะเป็นแบบ Full-duplex เนื่องจากอยู่บนหลักพื้นฐานของการสื่อสารแบบ UART ซึ่งแต่ละพอร์ตจะถึงกำหนดทิศทางของการสื่อสารแล้ว โดย MIDI IN เป็นพอร์ตที่ใช้สำหรับรับข้อมูลจากอุปกรณ์ MIDI อื่นที่เชื่อมต่อกัน และ MIDI OUT ใช้สำหรับส่งข้อมูลออกไปยังอุปกรณ์ MIDI อื่นที่เชื่อมต่อกัน หากมีแค่อุปกรณ์ MIDI ที่เชื่อมต่อกันเช่นระหว่าง Keyboard และ PC ในรูปที่ 2.16 สามารถเชื่อมต่อโดยใช้เพียง MIDI IN หรือ MIDI OUT ได้ ส่วน MIDI Thru นั้น จะใช้ในกรณีที่อุปกรณ์ตัวส่งเชื่อมต่อกับอุปกรณ์ตัวรับหลายๆตัว พอร์ต MIDI Thru จะเป็นเพียงทางผ่านของข้อมูลที่ได้รับโดยพอร์ต MIDI IN ของอุปกรณ์ตัวรับเท่านั้น

รูปที่ 2.17 ตัวอย่างการเชื่อมต่ออุปกรณ์ MIDI ที่มากกว่า 2 ชิ้น โดยใช้ MIDI Thru

การเชื่อมต่อทางไฟฟ้าของสายสัญญาณ วงจรอิเล็กทรอนิกส์ที่ใช้งานของ MIDI Port นั้นจะเป็นลักษณะของ Current loop ทั้งใน MIDI IN และ MIDI OUT โดยวงจรตัวอย่างการเชื่อมต่อของพอร์ตแสดงในรูปที่ 2.18



รูปที่ 2.18 ตัวอย่างวงจรการเชื่อมต่อที่ใช้ภายในของพอร์ต MIDI [4]

Current loop ของอุปกรณ์ MIDI นั้นจะทำการแยกโดดทางไฟฟ้าโดยใช้ Opto-isolator เป็นตัวส่งผ่านข้อมูล เหตุผลอันเนื่องมากจากการป้องกันและกำจัดผลที่เกิดจาก Ground loop ระหว่างอุปกรณ์ MIDI ทั้งสองเมื่อวงจรไม่ทำการแยกโดดซึ่งจะเหนี่ยวนำคลื่นแม่เหล็กไฟฟ้าภายนอก เช่น Hum 50 Hz จากระบบสายส่ง ทำให้เกิดสัญญาณรบกวนเข้าไปในระบบ

กระแสที่ใช้ใน Current loop ของ MIDI โดยทั่วเป็นจะประมาณ 5mA ซึ่งจะให้ Logic 0 ความเร็วของการเปลี่ยนแปลงระดับ Logic จะต้องมีช่วงเวลาขอบขาขึ้น (Rise time) และช่วงเวลาขอบขาลง (Fall time) ไม่ต่ำกว่า 2 us เพื่อรองรับความเร็วของ MIDI ที่ 31250 bps ได้โดยไม่เกิดความผิดพลาดของข้อมูล

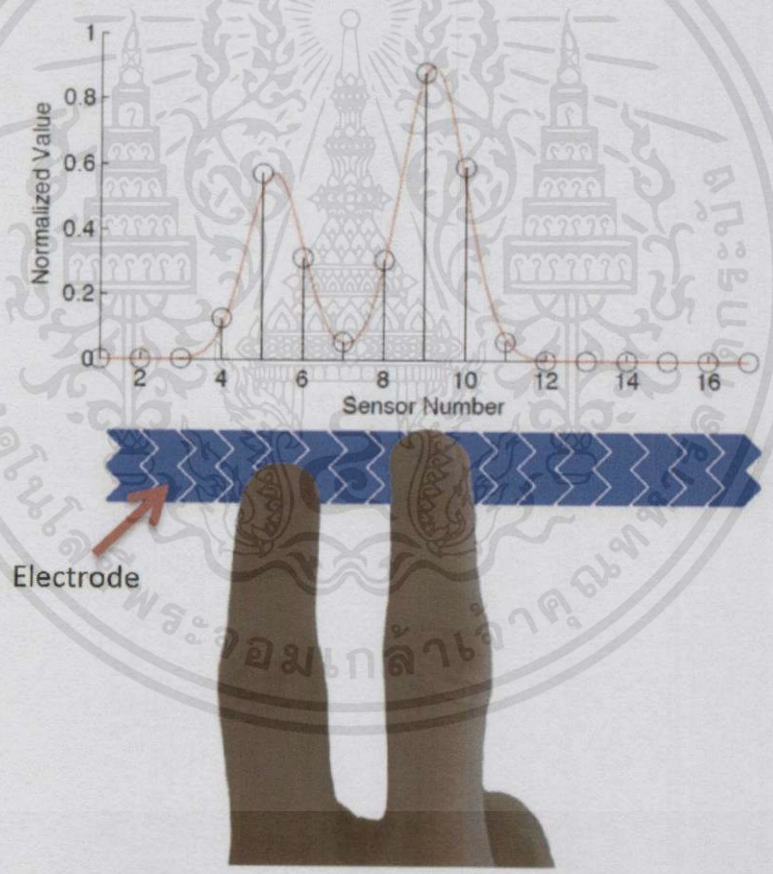
บทที่ 3

การออกแบบและพัฒนา

3.1 การออกแบบ Keyboard touch sensor

3.1.1 โครงสร้างและหลักการทำงาน

Keyboard touch sensor ถูกออกแบบให้มีโครงสร้างเป็นลักษณะของแผ่น เซ็นเซอร์ตรวจจับสัมผัสโดยใช้หลักการวัดความจุไฟฟ้า (Capacitive touch sensor) โดยขั้วสัมผัสถูกออกแบบในลักษณะของแถบสไลเดอร์ (Slider) เพื่อใช้ระบุตำแหน่งของนิ้วมือจากการรุด ซึ่งแถบสไสเดอร์จะประกอบด้วยแผ่นตรวจจับสัมผัสย่อยๆเรียกว่า อิเล็กโทรด (electrode) แสดงดังรูปที่ 3.1

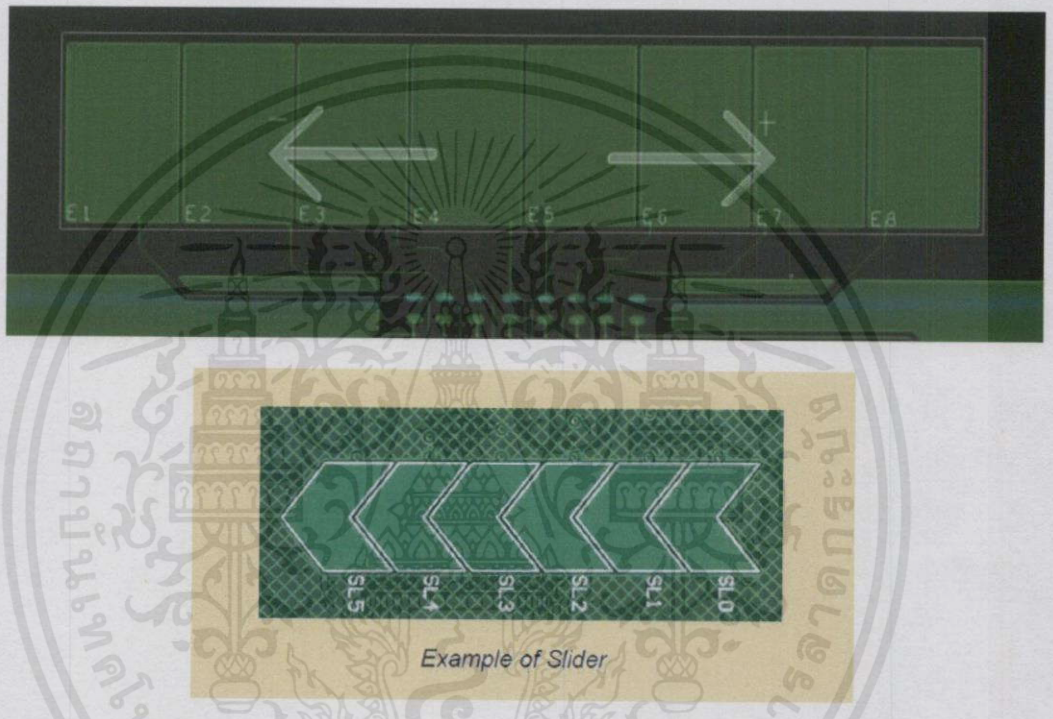


รูปที่ 3.1 แสดงลักษณะของแถบสไลเดอร์ที่ใช้บนแผ่นเซ็นเซอร์ตรวจจับสัมผัส [1]

การระบุตำแหน่งของปลายนิ้วบนแถบสไลเดอร์สามารถอธิบายได้จากรูปที่ 3.1 จากกราฟนั้น ค่าในแกนตั้งคือค่า Normalized Value บ่งบอกถึงความแรงของการสัมผัสของแต่ละอิเล็กโทรด (electrode) บนแถบสไลเดอร์ ส่วนตำแหน่งของอิเล็กโทรดจะระบุด้วยแกนนอนของ

กราฟคือค่า Sensor Number หากมีการสัมผัส ณ ตำแหน่งอิเล็กโทรดใดๆ ค่าความแรงในการสัมผัสในตำแหน่งนั้นก็จะสูงกว่าบริเวณอื่นที่ไม่ถูกการสัมผัส ดังนั้นตำแหน่งของนิ้วสัมผัสนั้นสามารถคำนวณได้จากค่าความแรงการสัมผัสของทุกๆอิเล็กโทรด

วัสดุที่ใช้ทำแผ่นเซ็นเซอร์ตรวจจับสัมผัสนั้นทำจากแผ่น PCB ชนิด 2-layer ขนาดความหนาอยู่ที่ 1.6 mm โดยออกแบบให้ลายทองแดงเป็นพื้นที่ในการสัมผัสของนิ้วมีลักษณะคล้ายกับแถบสไลเดอร์ตัวอย่างในแบบต่างๆ ดังรูปที่ 3.2



รูปที่ 3.2 แถบสไลเดอร์อย่างง่ายจากการออกแบบลายบน PCB [2]

แถบสไลด์จำเป็นที่จะต้องปิดคลุมด้วยฉนวนบางๆ เช่น แผ่นสติ๊กเกอร์บางๆ หรือใช้ Solder Mask จากกระบวนการสร้าง PCB เพื่อมิให้เกิดการเชื่อมต่อทางไฟฟ้าโดยตรงระหว่างแผ่นอิเล็กโทรดกับนิ้วของผู้ใช้งาน

สำหรับตัวอย่างข้อมูลแนะนำหลักการออกแบบ PCB layout สำหรับการออกแบบแถบสไลด์ของ Keyboard touch sensor นั้น สามารถศึกษาค้นคว้าได้จาก

- Freescale Semiconductor Application Note - Designing Touch Sensing Electrodes (AN3863)
- Atmel Application Note – Button, Slider, Wheel Sensor Guide Design (QTAN0079)
- Texas Instruments Design Guide SLAA576 – Capacitive Touch Hardware Design Guide

- Texas Instruments Application Report SLAA379 – Capacitive Single-Touch Sensor Design Guide

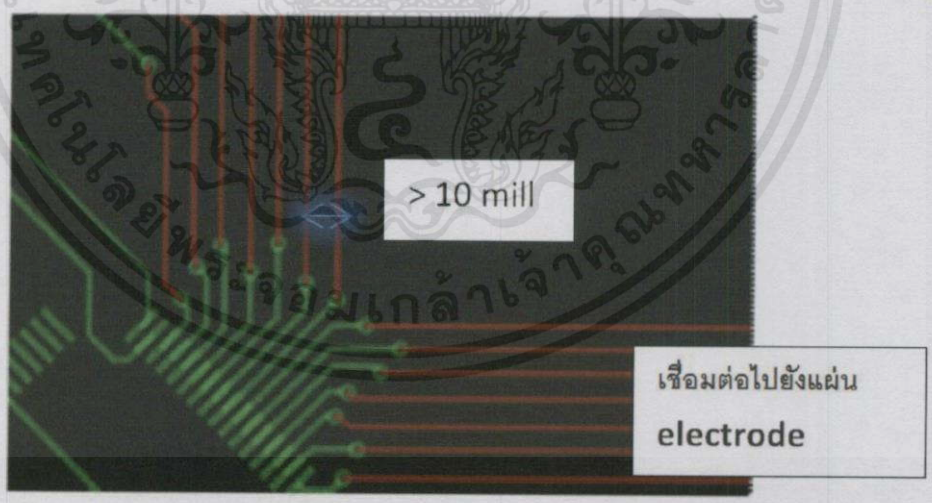
<u>โดยมีข้อสรุปหลักเกณฑ์ที่ใช้ในการออกแบบดังนี้</u>

- รูปร่างและขนาดพื้นที่ของอิเล็กโทรดนั้นมีผลต่อความไวต่อการสัมผัส (Sensitivity) อิเล็กโทรดที่มีขนาดใหญ่จะมีความไวต่อการสัมผัสมากเนื่องจากผลของการเพิ่มพื้นที่ผิว แต่จะเพิ่มความไวต่อการรบกวนของวัตถุรอบข้างด้วย

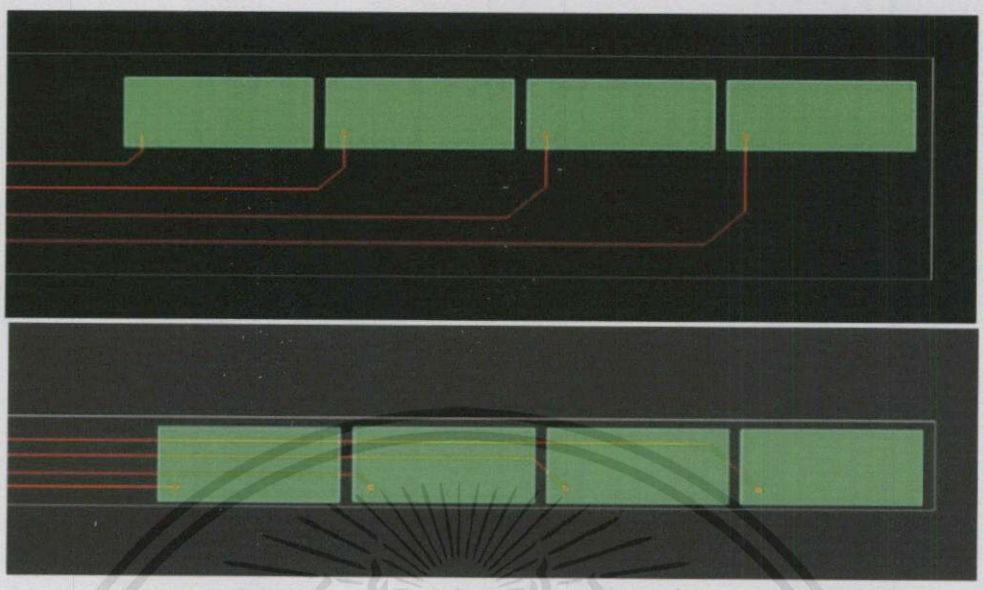
- ขนาดของ Ground plate ของวงจรควรให้มีขนาดใหญ่กว่าอิเล็กโทรดมากๆ เพื่อลดผลความไวต่อการสัมผัสของ Ground plate เพราะระบบต้องการให้อุปกรณ์ที่มีความไวต่อการสัมผัสมีเฉพาะอิเล็กโทรดเท่านั้น

- การลากลายเชื่อมต่อบน PCB ควรเว้นระยะห่างระหว่าง trace (ระยะ Clearance) ที่เชื่อมต่อกับอิเล็กโทรดให้พอสมควร ปกติไม่ควรต่ำกว่า 10 mill เพื่อลดผลไม่ให้เกิดผลการซ้อนทับทางสนามไฟฟ้าระหว่างอิเล็กโทรดกับอิเล็กโทรดที่อยู่ใกล้เคียง

- บริเวณพื้นที่ของส่วนอิเล็กโทรดจะต้องไม่มี trace ใดๆหรืออุปกรณ์อิเล็กทรอนิกส์ใดๆอยู่ข้างใต้หรืออยู่ใกล้กับอิเล็กโทรด เป็นไปได้มากที่สุดคือมีได้เฉพาะ trace ที่เชื่อมต่อกับแผ่นอิเล็กโทรดที่สามารถอยู่ด้านใต้ของอิเล็กโทรดได้เท่านั้น และ trace นั้นจะต้องมีขนาดเล็กมากเมื่อเทียบกับขนาดพื้นที่ของอิเล็กโทรด



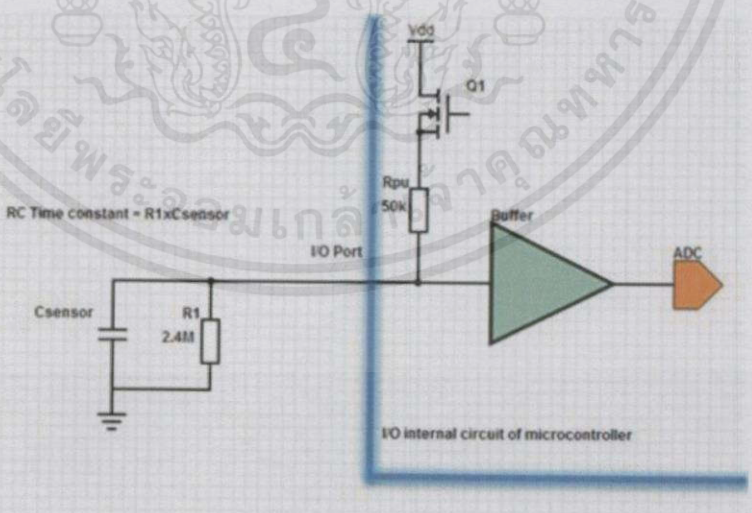
รูปที่ 3.3 ตัวอย่างการเว้นระยะห่างระหว่าง trace ในการออกแบบ Capacitive touch sensor [2]



รูปที่ 3.4 ตัวอย่างการเดินลาย trace เพื่อเชื่อมต่อกับ electrode ในการออกแบบ [2]

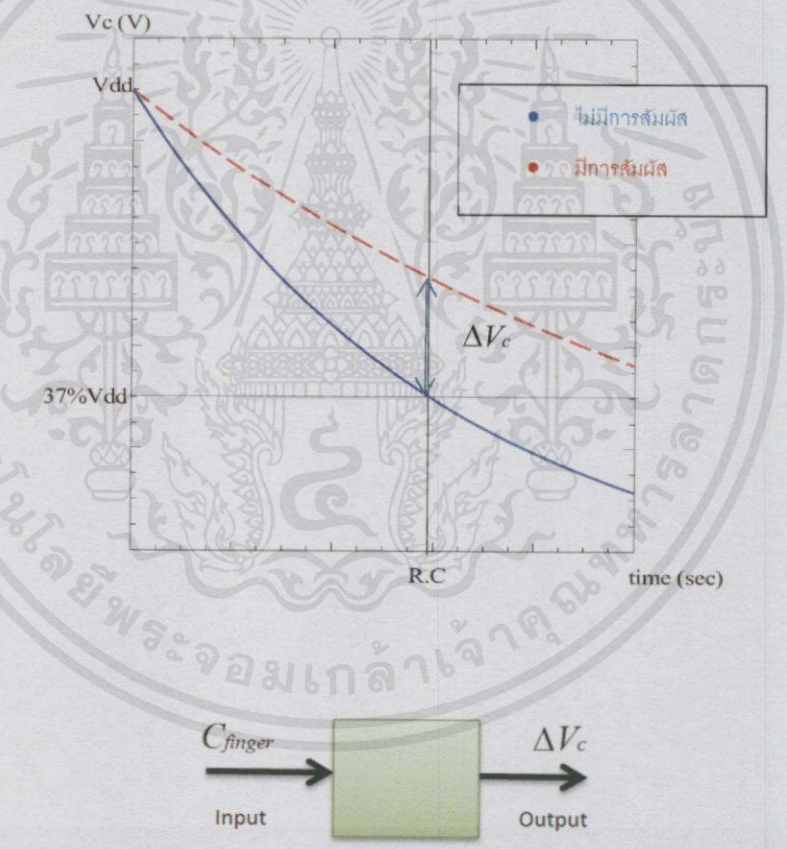
3.1.2 วงจรอิเล็กทรอนิกส์สำหรับ Keyboard touch sensor

วิธีตรวจจับการสัมผัสของนิ้วบนแถบสไลเดอร์ของ Keyboard touch sensor นั้นจะใช้หลักการวัดการเปลี่ยนแปลงกราฟการคายประจุของวงจร RC โดยมีไมโครคอนโทรเลอร์เพื่อควบคุมและวัด ซึ่งได้อธิบายหลักการทำงานดังรายละเอียดหัวข้อที่ 2.3 แล้ว ซึ่งจะเปรียบแผ่นอิเล็กโทรดเป็นตัวเก็บประจุที่มีค่าความจุแฝงจากโครงสร้างของอิเล็กโทรด โดยวงจร RC ที่ใช้นั้นเป็นดังรูปที่ 3.5



รูปที่ 3.5 วงจร RC ที่ใช้สำหรับวัดการเปลี่ยนแปลงค่า RC time constant

ไมโครคอนโทรเลอร์ที่เลือกใช้คือเบอร์ ATmega168 8-bit MCU ตระกูล AVR การทำงานของวงจรในรูปที่ 3.5 นั้น เริ่มแรกไมโครคอนโทรเลอร์จะสั่งให้ Q1 (อยู่ภายไมโครคอนโทรเลอร์ซึ่งไว้ใช้สำหรับการ Pull-up สำหรับอินพุต) ทำงาน ทำให้วงจร RC ที่เกิดจากตัวต้านทาน R1 และค่าความจุแฝงของอิเล็กโทรด Csensor ถูกชาร์จประจุจนเต็มผ่านตัวต้านทาน Rpu จากนั้นจะปิดการทำงานของ Q1 เพื่อเริ่มช่วงการคายประจุ และจะทำการวัดแรงดันตกคร่อมวงจร RC หลังจากวงจร RC คายประจุไปแล้วเป็นระยะเวลาที่กำหนดไว้โดยใช้วงจร A/D converter ภายในไมโครคอนโทรเลอร์ หากมีการสัมผัสของนิ้วบนอิเล็กโทรด ค่าความจุไฟฟ้าแฝง Csensor จะเปลี่ยนแปลง ส่งผลให้ค่าแรงดันตกคร่อมวงจร RC (V_c) ที่วัด ณ ระยะเวลาการคายประจุเดิมนั้นเปลี่ยนแปลงไป ดังนั้นการตรวจจับการสัมผัส สามารถทำได้โดยการวัดแรงดันตกคร่อมวงจร RC หลังจากวงจรคายประจุ ณ ตำแหน่งเวลาที่เหมาะสม โดยสามารถอธิบายด้วยกราฟแรงดันตกคร่อมวงจร RC ขณะทำการคายประจุและแสดงบล็อกฟังก์ชั่นระบบได้ดังรูปที่ 3.6

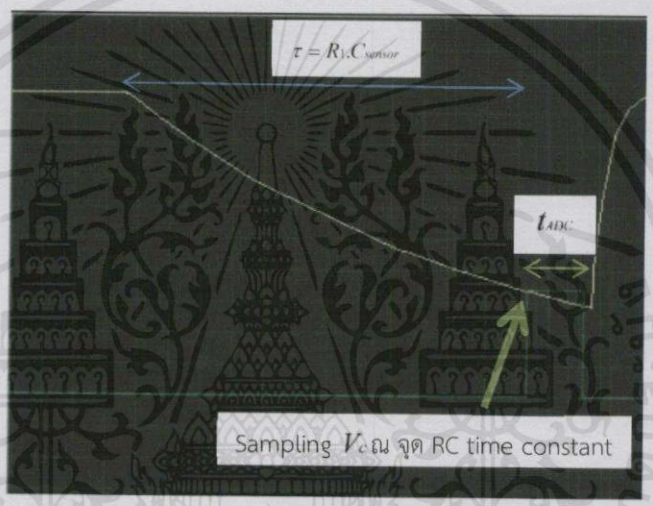


รูปที่ 3.6 (บน) กราฟแรงดันตกคร่อมวงจร RC ขณะคายประจุ, (ล่าง) บล็อกฟังก์ชั่นของระบบตรวจจับการสัมผัสเทคนิคการวัดการคายประจุของวงจร RC

จากกราฟแรงดันตกคร่อมวงจร RC ในรูปที่ 3.6 แสดงความแตกต่างของกราฟระหว่างมีนิ้วสัมผัสกับไม่มีนิ้วสัมผัส เมื่อนิ้วสัมผัสบนอิเล็กโทรด ค่าความจุแฝงของอิเล็กโทรดจะถูกเพิ่มขึ้นด้วยความจุไฟฟ้าแฝงของนิ้วซึ่งรายละเอียดนั้นได้อธิบายไว้ในหัวข้อที่ 2.3 ทำให้การคายประจุช้าลงตาม

กราฟสีแดงในรูปที่ 3.6 แรงดันตกคร่อมวงจร RC (V_c) ที่วัดได้เมื่อวงจรคายประจุด้วยระยะเวลา $\tau = R.C$ จึงมีค่าสูงกว่ากรณีที่ไม่มีสัมผัสซึ่งแสดงการลดลงของแรงดันตกคร่อมวงจร RC ด้วยกราฟสีน้ำเงิน ดังนั้น วงจรจึงสามารถตรวจจับการสัมผัสได้โดยการวัดค่าของ ΔV_c ซึ่งเป็นเอาต์พุตที่ได้

การเลือกค่า R1 ที่เหมาะสมสำหรับวงจร RC นั้นเกี่ยวข้องกับความเร็วในการคายประจุ หากการคายประจุนั้นเร็วเกินไป อาจทำให้วงจร A/D converter ไม่สามารถสุ่มวัดค่าแรงดันตกคร่อมวงจร RC ได้ทันเวลา ดังนั้น จึงต้องออกแบบให้ช่วงเวลาการคายประจุของวงจร RC นั้นมากกว่าช่วงเวลาการสุ่มค่าแรงดันของวงจร A/D converter โดยมีหลักการวิเคราะห์ดังนี้



รูปที่ 3.7 กราฟจำลองการทำงานของวงจร RC ขณะคายประจุเพื่อวิเคราะห์ผลของการสุ่มค่าแรงดันของวงจร A/D converter

จากรูปที่ 3.7 ได้แสดงผลของความเร็วในการคายประจุของวงจร RC และช่วงเวลาการสุ่มค่าแรงดัน V_c ของวงจร A/D converter หากกำหนดให้เวลาที่ใช้ในการคายประจุเท่ากับค่า RC time constant : τ ซึ่งคำนวณได้จากสมการที่ 3.1 และกำหนดให้ช่วงเวลาการสุ่มค่าแรงดันที่วงจร A/D converter ใช้คือ t_{ADC} ซึ่งให้เป็นอัตราส่วนกับค่า RC time constant เท่ากับ n ดังสมการที่ 3.2 จะสามารถคำนวณค่า R1 ที่เหมาะสมได้จากสมการที่ 3.3

$$\tau = R_1.C_{sensor} \tag{3.1}$$

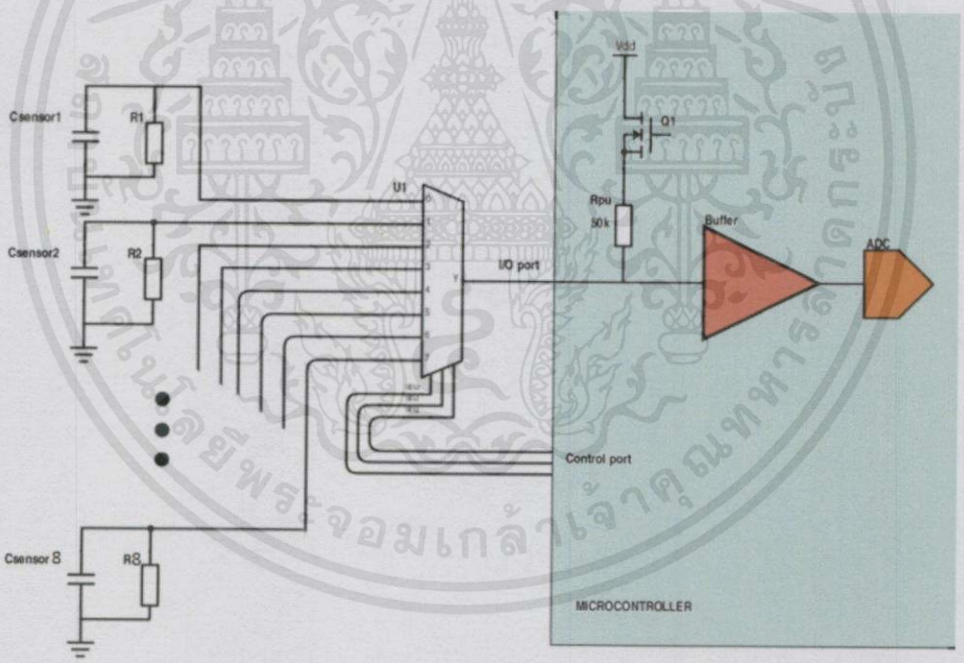
$$\tau / t_{ADC} = n \tag{3.2}$$

$$R_1 = \frac{n.t_{ADC}}{C_{sensor}} \tag{3.3}$$

โดยแนะนำให้ n ควรมีค่ามากๆ เพื่อทำให้ผลของการเปลี่ยนแปลงของค่า V_c ระหว่างการสุ่มค่าแรงดันของวงจร A/D converter น้อยลง การสุ่มค่าจึงเกิดความผิดพลาดน้อยลง

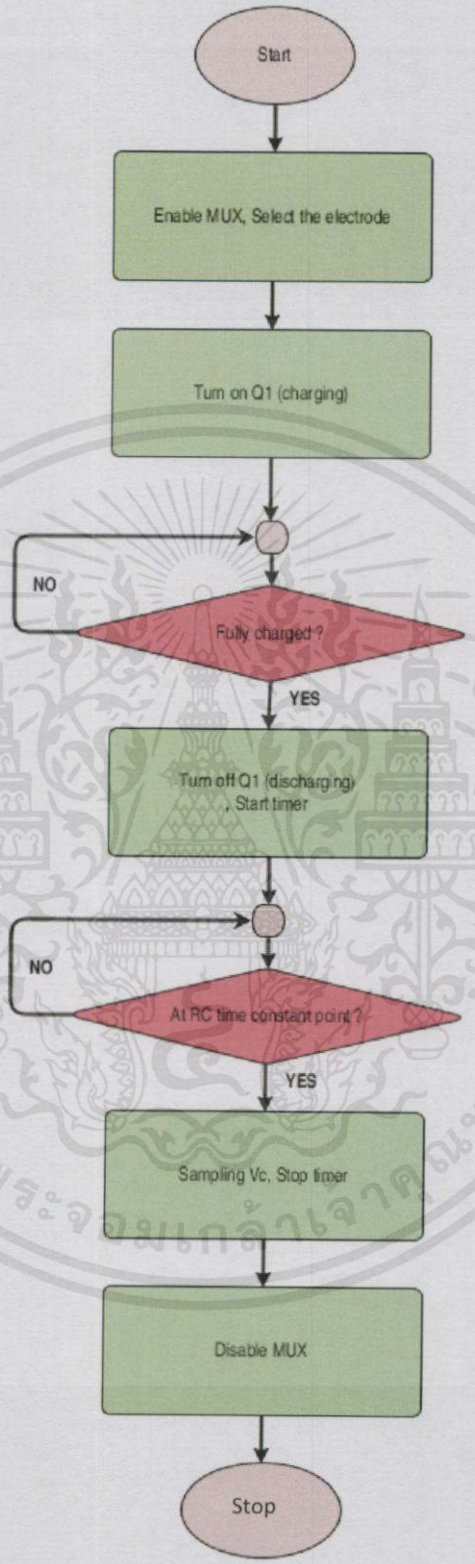
กรณีที่แถบสไลเดอร์บน Keyboard touch sensor ประกอบด้วยอิเล็กโทรดจำนวนมากๆ ซึ่งหมายถึง เซ็นเซอร์นั้นจะมีความละเอียดสูงในการตรวจจับตำแหน่งของนิ้ว จำเป็นที่จะต้องใช้พอร์ต I/O ของไมโครคอนโทรเลอร์ที่เชื่อมต่อกับ A/D converter ภายในจำนวนมาก เนื่องจากไมโครคอนโทรเลอร์ที่ได้เลือกใช้คือ ATmega168 ซึ่งมีพอร์ต A/D converter จำนวนเพียง 6 ช่องเท่านั้นที่สามารถใช้ได้ (6 จากทั้งหมด 8, อีก 2 ขานั้นใช้เป็น I2C แทนเนื่องจากมีการใช้ I/O ร่วมกัน)

การแก้ไขสามารถทำได้โดยใช้วงจรมัลติเพล็กซ์เซอร์ (Multiplexer : MUX) เพื่อการขยายจำนวนการเชื่อมต่อวงจร RC เข้ากับขา I/O ของไมโครคอนโทรเลอร์ให้มากขึ้น การทำงานของวงจรมัลติเพล็กเซอร์ทำหน้าที่เลือกช่องสัญญาณที่มีข้อมูลช่องหนึ่งจากหลายๆช่องสัญญาณมาเป็นอินพุตและต่อช่องสัญญาณที่มีข้อมูลนั้นเข้าเป็นสัญญาณเอาต์พุตเพียงเอาต์พุตเดียว และเมื่อนำมัลติเพล็กเซอร์มาปรับแก้ในวงจรรูปที่ 3.5 แล้ว จะได้วงจรที่ใช้งานจริงดังรูปที่ 3.8



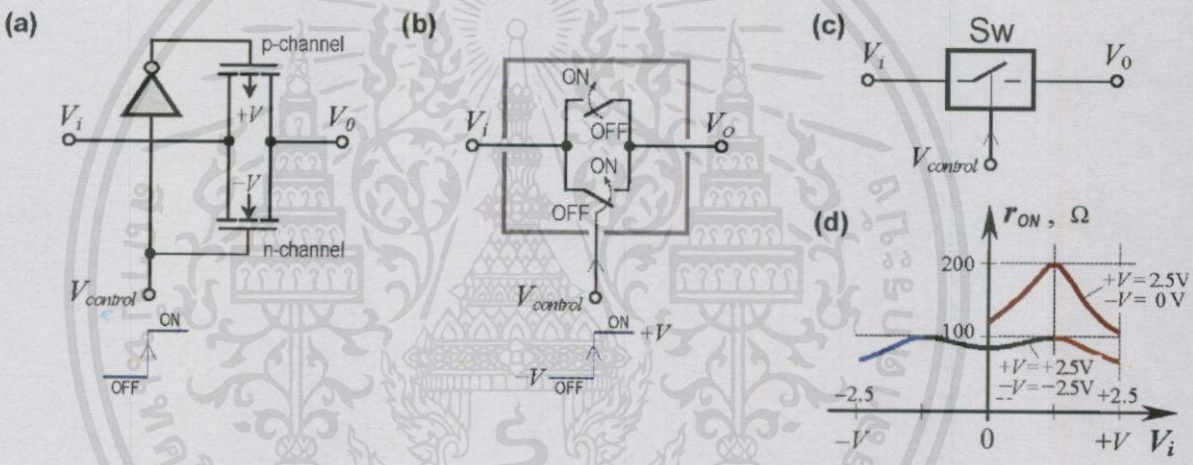
รูปที่ 3.8 วงจร RC ที่ใช้สำหรับวัดการเปลี่ยนแปลงค่า RC time constant (เพิ่มส่วนของ MUX)

จากวงจรในรูปที่ 3.8 การควบคุมการชาร์จประจุและอ่านค่าแรงดันช่วงคายประจุ ณ วงจร RC ของอิเล็กโทรดใดๆ สามารถทำได้โดยใช้สัญญาณควบคุมมัลติเพล็กเซอร์ (Control signal) เพื่อเลือกวงจร RC ของอิเล็กโทรดที่ต้องการเชื่อมต่อกับ I/O port ของไมโครคอนโทรเลอร์ โดยขั้นตอนการควบคุมการชาร์จประจุและอ่านค่าแรงดันวงจร RC ด้วยไมโครคอนโทรเลอร์ อธิบายด้วย Flow chart ในรูปที่ 3.9



รูปที่ 3.9 ขั้นตอนการควบคุมการชาร์จประจุและอ่านค่าแรงดันวงจร RC ด้วยไมโครคอนโทรเลอร์

ประเภทของมัลติเพล็กเซอร์ที่เลือกใช้ในวงจรรูปที่ 3.8 คือ 8-channel analog MUX เบอร์ CD4051 เหมาะสำหรับ A/D conversion โดยการเชื่อมต่อภายในระหว่างอินพุตของช่องสัญญาณที่ถูกเลือกกับเอาต์พุตจะเปรียบกับการใช้สวิตช์เชื่อมต่อซึ่งมีความต้านค่าหนึ่ง เรียกว่า "ON Resistance : Ron" โดยสวิตช์นี้จะสร้างจากวงจร CMOS ในรูปแบบของ Bi-directional switch ซึ่งเกิดจากการขนานกันระหว่าง NMOS ทรานซิสเตอร์และ PMOS ทรานซิสเตอร์และมีวงจรควบคุมการนำกระแสของทรานซิสเตอร์ทั้งสองดังรูปที่ 3.10 ค่า Ron ของ CD4051 จะมีค่าขึ้นอยู่กับแรงดันไฟเลี้ยง (Supply voltage) ของมัลติเพล็กเซอร์และระดับของแรงดันอินพุตโดยที่ Ron สูงสุดจะอยู่ที่ประมาณ 500 Ohms ซึ่งไม่มีผลต่อการชาร์จ RC ในวงจรรูปที่ 3.5 เนื่องจากค่าของ Rpu สูงกว่าค่า Ron มากๆ ทำให้เส้นทางการชาร์จประจุมีผลเพียงค่า Rpu เท่านั้น



รูปที่ 3.10 (a) วงจรภายในของสวิตช์แบบ CMOS Bi-direction ที่ใช้ใน Analog MUX, (b) วงจรสมมูลของ CMOS Bi-direction, (c) สัญลักษณ์วงจรไฟฟ้าของสวิตช์, (d) แสดงกราฟความสัมพันธ์ระหว่าง ON Resistance กับขนาดแรงดันอินพุต [10]

3.1.3 การเลือกขนาดและความละเอียดของ Keyboard touch sensor

ระบบเซ็นเซอร์ควบคุมฟังก์ชัน Pitch bend และ Vibrato โดยใช้ Keyboard touch sensor นั้นจะต้องมีความสามารถไม่น้อยไปกว่าการควบคุมด้วยล้อคันโยก (Wheel) หรือจอยสติก (Joystick) ที่ถูกใช้ในระบบเก่าของคีย์บอร์ดซินธิไซเซอร์ (Keyboard synthesizer) ดังรูปที่ 1.2 ในบทที่ 1 โดยความสามารถที่กล่าวนั้นคือ ความละเอียดของฟังก์ชัน Pitch bend และ Vibrato กับการตอบสนองต่อความเร็วของมือหรือนิ้วที่ใช้ในการควบคุมฟังก์ชัน Pitch bend และ Vibrato

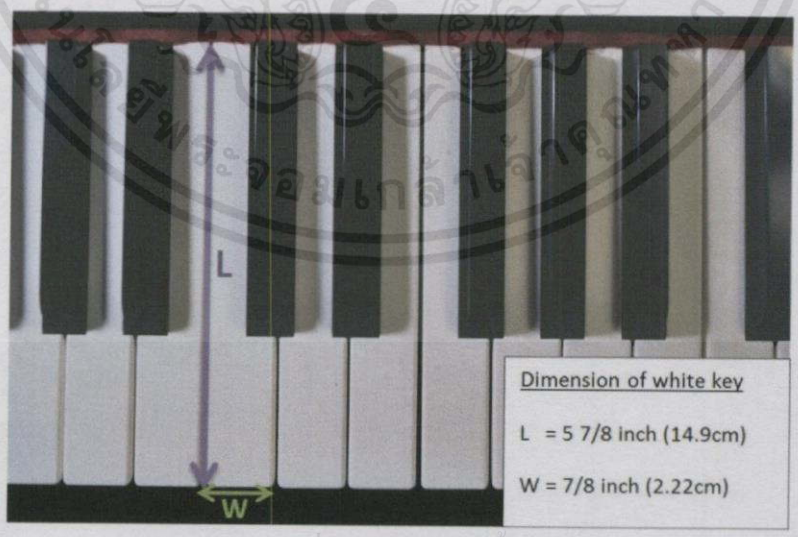
ปัญหาหลักของโครงงานที่ได้มองเห็น ซึ่งอาจเกิดจากการออกแบบที่ไม่เหมาะสม สามารถสรุปเป็นหัวข้อได้ดังนี้

- ระบบการควบคุมฟังก์ชัน Pitch bend และ Vibrato ด้วย Keyboard touch sensor ไม่สามารถตอบสนองต่อความเร็วการรูดของนิ้วในการควบคุมได้ทัน ทำให้การเปลี่ยนแปลงเสียงโน้ตเกิดความไม่ต่อเนื่องและไม่เป็นธรรมชาติ
- เกิดการหน่วงของเวลา (Delay time) ของการเปลี่ยนแปลงเสียงโน้ตหลังจากที่ระบบได้รับอินพุตของนิ้วแล้ว ซึ่งเกิดจากเวลาที่ต้องใช้ในการประมวลผลข้อมูลอินพุตเพื่อสร้างเอาต์พุตให้เป็นลักษณะของเสียงโน้ตตามลักษณะการรูดของนิ้ว

สำหรับขนาดและความละเอียดของ Keyboard touch sensor มีผลต่อความสามารถการตอบสนองต่อความเร็วการรูดของนิ้วและความละเอียดของระดับฟังก์ชัน Pitch bend และ Vibrato โดยหลักการออกแบบขนาดและความละเอียดของ Keyboard touch sensor เพื่อลดผลของปัญหาที่กล่าวมาให้ได้มากที่สุดนั้น มีดังนี้

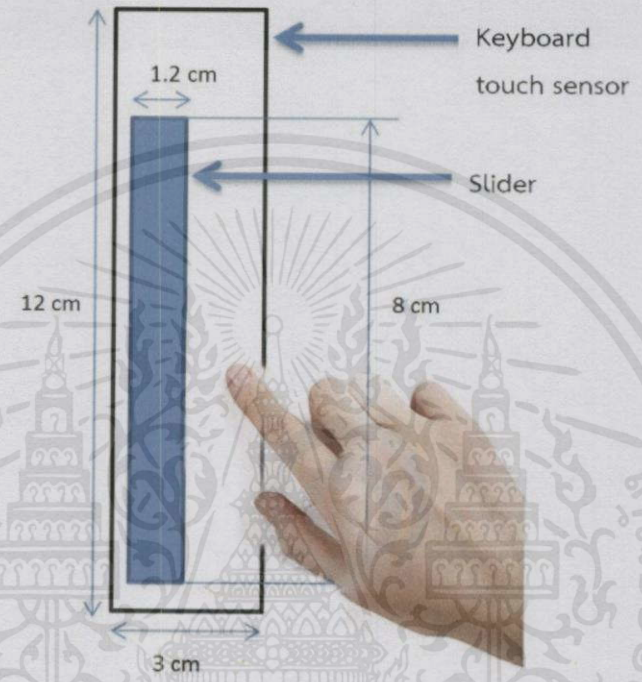
1) ขนาดของแถบสไลเดอร์ และขนาดของ Keyboard touch sensor

ตามวัตถุประสงค์ของโครงงานนั้น Keyboard touch sensor จะต้องมีขนาดเดียวกับตัวลิ่มคีย์สีขาว (White key) ของคีย์บอร์ดซินธีไซเซอร์ เพื่อที่จะสามารถติดตั้งเซนเซอร์นี้บนคีย์บอร์ดซินธีไซเซอร์จริงๆได้ โดยขนาดมาตรฐานของลิ่มคีย์ของคีย์บอร์ดซินธีไซเซอร์จะเป็นมาตรฐานเดียวกับลิ่มคีย์เปียโน ซึ่งจะประกอบด้วย คีย์สีดำ (Black key) และ คีย์สีขาว แสดงดังรูปที่ 3.11



รูปที่ 3.11 ภาพแสดงขนาดมาตรฐานความกว้างและยาวของคีย์เปียโนสีขาว (White key)

ขนาดของคีย์ดังรูปที่ 3.11 จะเป็นตัวกำหนดขอบเขตขนาดความยาวของแถบสไลเดอร์และความกว้างของอิเล็กโทรดในแถบสไลเดอร์ของ Keyboard touch sensor โดยแถบสไลเดอร์จะต้องครอบคลุมพื้นที่ตั้งแต่ด้านล่างสุดของคีย์และยาวขึ้นไปถึงด้านบนของคีย์ ความยาวและความกว้างของแถบสไลเดอร์และขนาดของแผ่น Keyboard touch sensor ที่จะนำมาสร้างเป็นเซนเซอร์ต้นแบบนั้นมีรายละเอียดแสดงในรูปที่ 3.12



รูปที่ 3.12 แสดงรายละเอียดขนาดของ Keyboard touch sensor ต้นแบบแรก

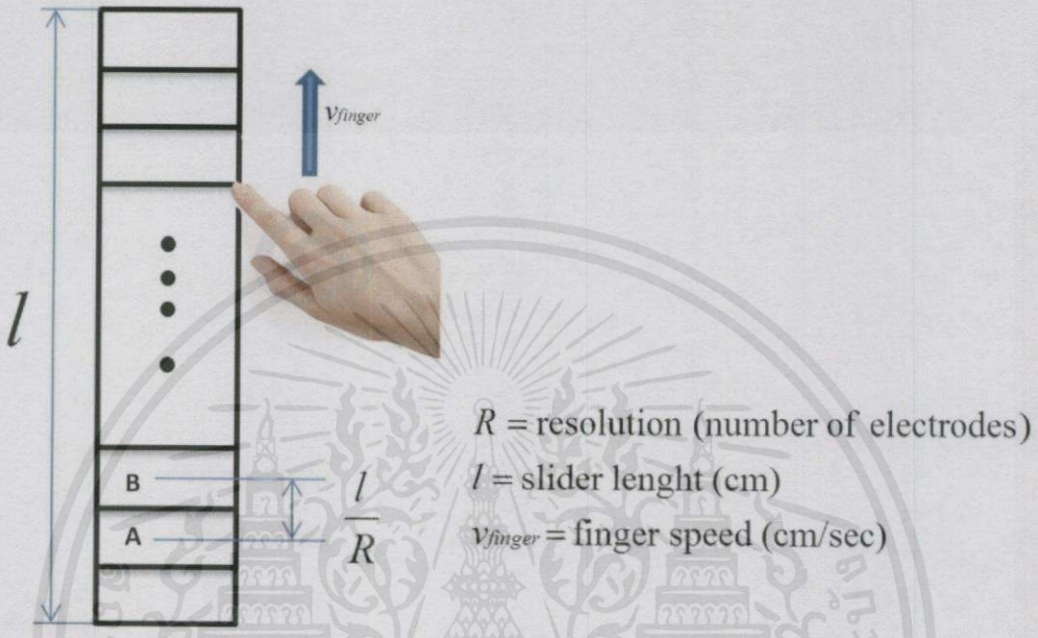
จากรูปที่ 3.12 เซ็นเซอร์ต้นแบบนั้นมีความกว้างของแถบสไลเดอร์คือ 1.2 เซนติเมตร โดยเลือกให้มีขนาดเหมาะสมกับพื้นที่ของปลายนิ้วมือ ความยาวของแถบสไลเดอร์คือ 8 เซนติเมตร ส่วนขนาดของเซ็นเซอร์คือ 12 x 3 ตร.ซม. ซึ่งมีขนาดใหญ่กว่าคีย์เปียโนมาตรฐาน สาเหตุเนื่องจากข้อจำกัดที่ด้านพื้นที่ในการออกแบบลาย PCB

2) การเลือกความละเอียด (Resolution) ของ Keyboard touch sensor

ความละเอียดของเซ็นเซอร์นั้นมาจากจำนวนอิเล็กโทรดที่อยู่ในแถบสไลเดอร์ หากแถบสไลเดอร์มีจำนวนอิเล็กโทรดอยู่มาก ความละเอียดในการระบุตำแหน่งของนิ้วจะสูง ซึ่งมีผลทำให้ค่าตำแหน่งของนิ้วขณะรูดนั้นมีความต่อเนื่อง เสียงโน้ตที่เกิดขึ้นจากฟังชั่น Pitch bend และ Vibrato ก็จะมีความต่อเนื่อง ไพเราะ เป็นธรรมชาติ

แต่หากความละเอียดมีมากเกินไป ค่าเวลาการสุ่มวัดค่า (Sampling time) ตำแหน่งของนิ้วด้วยไมโครคอนโทรเลอร์บนเซ็นเซอร์นั้นจะเพิ่มขึ้น เนื่องจากจะต้องวัดการเปลี่ยนแปลงความจุไฟฟ้าของทุกๆอิเล็กโทรดซึ่งมีจำนวนเพิ่มขึ้นตามความละเอียดที่เพิ่มขึ้น มีผลทำให้เซ็นเซอร์ไม่สามารถตอบสนองต่อการรูดของนิ้วด้วยความเร็วสูงๆได้ ดังนั้นจึงมีความจำเป็นที่จะต้องวิเคราะห์

เพื่อหาความละเอียดที่เหมาะสมที่สุดที่ทำให้ Keyboard touch sensor สามารถตอบสนองต่อความเร็วการรูดนิ้วได้ทัน และความต่อเนื่องของเสียงโน้ต โดยแนวคิดในการวิเคราะห์นั้นมีดังนี้



รูปที่ 3.13 ภาพแบบจำลองการรูดนิ้วบนเซ็นเซอร์เพื่อวิเคราะห์หาความละเอียดที่เหมาะสม

Keyboard touch sensor จะตอบสนองต่อการรูดนิ้วด้วยความเร็วสูงได้ก็ต่อเมื่อ เซ็นเซอร์สามารถสุ่มวัดค่าตำแหน่งการสัมผัสได้ทันเวลา ซึ่งทำให้ตำแหน่งของการสัมผัสที่วัดได้มีความต่อเนื่อง ดังนั้นจึงหมายถึงเซ็นเซอร์สามารถตรวจจับการสัมผัสตำแหน่งอิเล็กโทรด A และ B ได้ทันเวลาเมื่อนิ้วเคลื่อนที่ผ่านอิเล็กโทรดทั้งสองด้วยความเร็วนิ้ว (finger speed : v_{finger}) ดังแสดงในรูปที่ 3.13

จากภาพจำลองในรูปที่ 3.13 จะพิจารณาให้ความเร็วของนิ้วคงที่ตลอดช่วงการรูดเท่ากับ v_{finger} ความละเอียดของเซ็นเซอร์หมายถึงจำนวนอิเล็กโทรดที่มีอยู่ในแถบสไลเดอร์คือ R และระยะห่างตามแนวยาวจากอิเล็กโทรด A ถึง B เท่ากับ l/R ดังนั้น ความเร็วสูงสุดที่เซ็นเซอร์สามารถตอบสนองได้ทันนั้นจะมีค่าเท่ากับ

$$v_{finger} = \frac{l}{R.t_{sr}} \tag{3.4}$$

โดย t_{sr} คือเวลาที่ใช้ในการสุ่มวัดค่า (Sampling time) ตำแหน่งของนิ้วบนเซ็นเซอร์ที่มีจำนวนอิเล็กโทรดเท่ากับ R จากการพิจารณาการทำงานของวงจรเซ็นเซอร์ในหัวข้อที่ 3.1.2 จะเห็นได้ว่า หากแถบสไลเดอร์ของเซ็นเซอร์มีจำนวนอิเล็กโทรดมากๆ เซ็นเซอร์จะต้องเสียเวลามาก

ขึ้นในการวัดการเปลี่ยนแปลงค่าความจุไฟฟ้าของทุกอิเล็กโทรด โดยสมมติให้ว่าเวลาการทำงานของเซนเซอร์ส่วนใหญ่คือการสุ่มวัดค่าของอิเล็กโทรด จากการวิเคราะห์ทำให้สามารถเขียนความสัมพันธ์ระหว่างค่าเวลาการสุ่มวัดค่าตำแหน่ง t_{sr} กับจำนวนอิเล็กโทรดบนแถบสไลด์ R ได้เป็น

$$t_{sr} = R.t_s \tag{3.5}$$

โดย t_s คือเวลาที่ใช้ในการสุ่มวัดค่า (Sampling time) ตำแหน่งของนิ้วบนเซ็นเซอร์ที่มีจำนวนอิเล็กโทรดเพียง 1 อิเล็กโทรดและใช้อัลกอริทึม (Algorithm) การวัดตำแหน่งของนิ้วเดียวกับที่ใช้ในเซ็นเซอร์ที่มีจำนวนอิเล็กโทรดเท่ากับ R และเมื่อแทน t_{sr} ด้วยสมการที่ 3.5 ลงในสมการที่ 3.4 จะได้

$$v_{finger} = \frac{l}{R^2.t_s} \tag{3.6}$$

ดังนั้นจะสามารถคำนวณหาจำนวนอิเล็กโทรดสูงสุดสำหรับเซ็นเซอร์ที่ยังทำให้ Keyboard touch sensor สามารถตอบสนองต่อการรูดนิ้วด้วยความเร็ว v_{finger} ด้วยสมการนี้

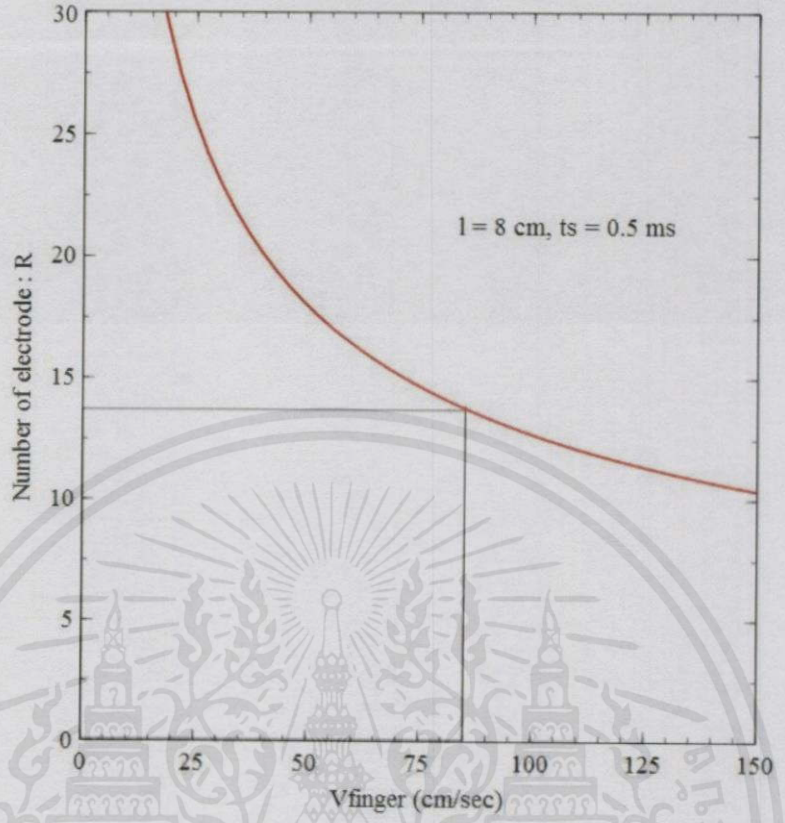
$$R = \sqrt{\frac{l}{t_s.v_{finger}}} \tag{3.7}$$

โดยที่ R = จำนวนอิเล็กโทรดบนสไลเดอร์ที่มีความยาวเท่ากับ l

l = ความยาวของแถบสไลเดอร์ (cm)

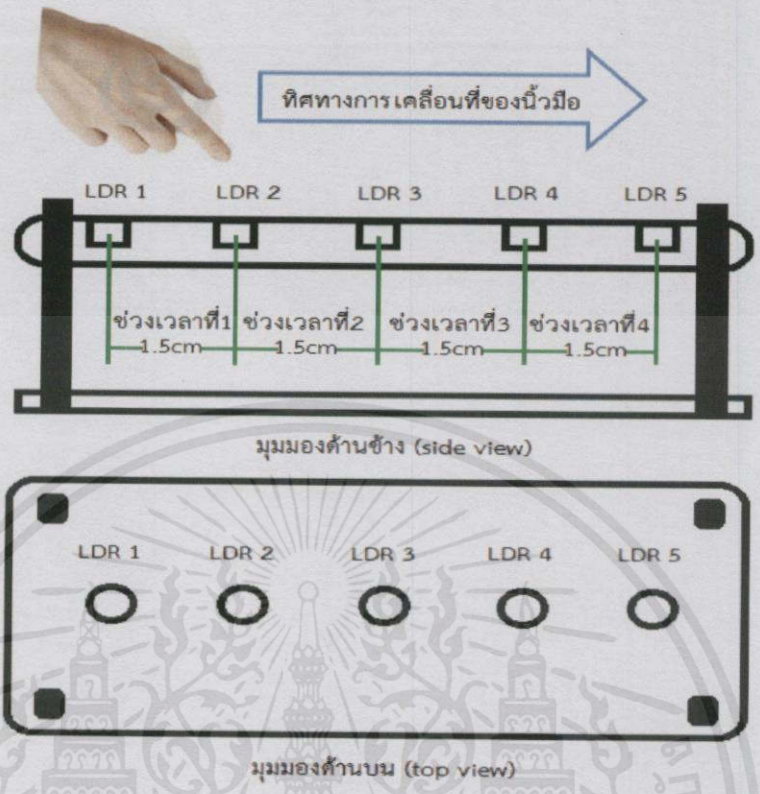
v_{finger} = ความเร็วเฉลี่ยของการรูดนิ้ว (cm/s)

t_s = เวลาที่ใช้ในการสุ่มวัดค่าตำแหน่งของนิ้วบนเซ็นเซอร์ที่มีจำนวนอิเล็กโทรด1ชิ้น



รูปที่ 3.14 กราฟความสัมพันธ์ระหว่างจำนวนอิเล็กโทรด (R) ของสไลด์กับความเร็วเฉลี่ยสูงสุดของ
นิ้วที่ Keyboard touch sensor สามารถตอบสนองต่อการรูดทัน ($Vfinger$)

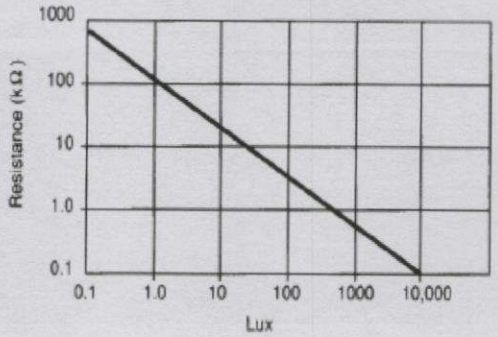
จากสมการที่ 3.7 การคำนวณหาจำนวนอิเล็กโทรดที่เหมาะสม จำเป็นที่จะต้องทราบความเร็วการรูดนิ้ว $Vfinger$ สูงสุดที่เป็นไปได้ โดยการทดลองหาความเร็วการรูดของนิ้วมือบนเซนเซอร์นั้น จะใช้เซ็นเซอร์แสงชนิดตัวต้านทานชนิดที่ไวต่อแสง (Photo-cell resistor : LDR) วัดความเร็วโดยใช้หลักการตรวจจับของช่วงเวลาที่นิ้วมือเคลื่อนจากเซ็นเซอร์แสงจุดหนึ่งไปยังตัวเซ็นเซอร์แสงอีกจุดหนึ่ง โดยกำหนดระยะห่างระหว่างเซ็นเซอร์แสงไว้คงที่ค่าหนึ่ง จากนั้นจึงนำช่วงเวลาดังกล่าวไปคำนวณหาค่าความเร็วของนิ้วมือเพื่อนำค่าความเร็วที่ได้ไปคำนวณหาจำนวนอิเล็กโทรดในแทบสไลเดอร์ของพร็อกซิมิตี้เซนเซอร์ที่เหมาะสม



รูปที่ 3.15 แบบจำลองของอุปกรณ์วัดความเร็วการรูดของนิ้วมือโดยใช้เซ็นเซอร์แสงชนิดตัวต้านทาน LDR

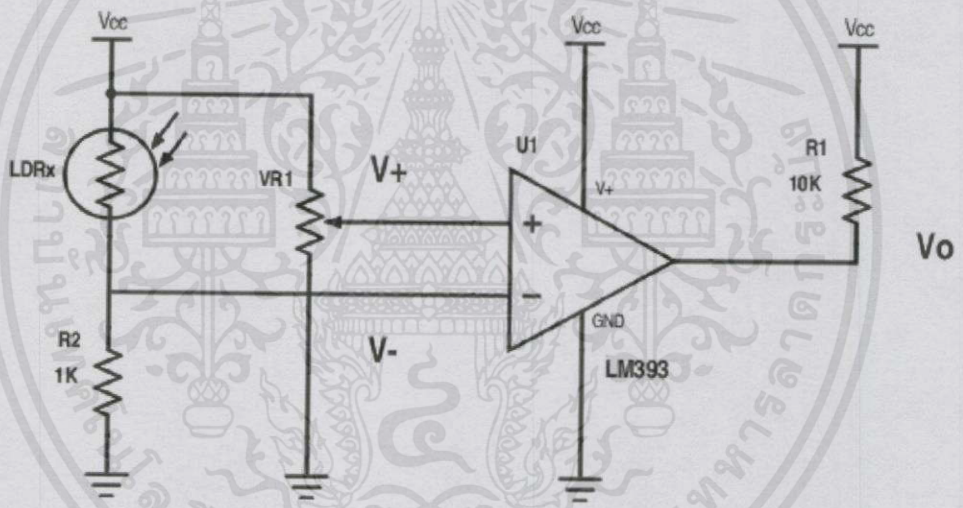
เหตุผลที่เลือกใช้ตัวต้านทานชนิดที่ไวต่อแสงในการทดลองเพราะใช้งานง่าย ราคาถูกและสามารถหาซื้อได้ง่ายแต่มีข้อเสียคือมีเวลาช่วงเวลาการเปลี่ยนแปลงของค่าความต้านทานเมื่อมีการเปลี่ยนแปลงของแสง ทั้งขาขึ้น (Rise time) และช่วงเวลาขาลง (Fall time) ค่อนข้างนาน ซึ่งทำให้ตัวต้านทานชนิดที่ไวต่อแสงเทิร์นออนและเทิร์นออฟช้าลงเกิดช่วงเวลาดีเลย์ (Delay Time) แต่ไม่เป็นปัญหาในการใช้งานเพราะช่วงเวลาดีเลย์ของตัวต้านทานชนิดที่ไวต่อแสงแต่ละตัวมีค่าเท่าๆกัน ทำให้การวัดช่วงเวลาซึ่งเกิดจากผลต่างของเวลา 2 จุด เกิดการหักล้างค่าเวลาดีเลย์

ตัวต้านทานชนิดที่ไวต่อแสงจะมีความต้านทานต่ำมากเมื่อได้รับแสงจำนวนมาก แต่มีความต้านทานสูงเมื่อได้รับแสงจำนวนน้อย ดังแสดงในรูปที่ 3.16



รูปที่ 3.16 กราฟแสดงความสัมผัสระหว่างค่าความต้านทานของ LDR กับค่าความเข้มแสงที่LDR ได้รับ [11]

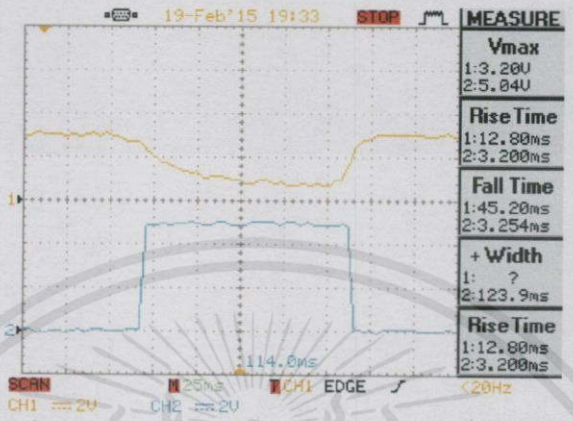
วงจรที่ใช้ตรวจจับการรูดผ่านของนิ้วมือนั้น ใช้พื้นฐานของวงจรเปรียบเทียบแรงดัน (Voltage comparator) โดยสัญญาณอินพุตนั้นได้วงจรแบ่งแรงดัน (Voltage divider) ที่มี LDR ซึ่งแสดงวงจรที่ใช้งานดังรูปที่ 3.17



รูปที่ 3.17 วงจรเปรียบเทียบแรงดันที่ใช้สำหรับการวัดความเร็วการรูดนิ้วโดยใช้ LDR

การวัดความเร็วของนิ้วเริ่มจากการรูดนิ้วจากจุดเริ่มต้นผ่าน LDR1 เมื่อนิ้วมือบดบังแสง LDR1ความต้านทานของ LDR1 จะมีค่าสูงขึ้น ทำให้แรงดันไฟฟ้าตกคร่อมตัวต้านทาน R2 น้อยลง ซึ่งเป็นผลมาจากวงจรแบ่งแรงดัน ทั้งนี้ต้องเลือกใช้ค่าของ R2 ให้มีค่าต่ำกว่าค่าความต้านทาน LDR ขณะแสงมืดอยู่ประมาณ 10 เท่า ค่าแรงดันไฟฟ้าตกคร่อม R2 จะส่งไปเข้าอินพุทลบ (Vi-) ของวงจรเปรียบเทียบ U1 โดย U1 คือ LM393 เอาต์พุตเป็นลักษณะ Open-collector ในขณะที่อินพุทบวก (Vi+) ของวงจรเปรียบเทียบนั้นคือค่าแรงดันอ้างอิงซึ่งได้จากการหมุนปรับค่าของตัวต้านทานปรับค่าได้ (VR1) เมื่อนิ้วมือเคลื่อนที่ผ่าน LDR1 วงจรเปรียบเทียบแรงดันจะให้แรงดันเอาต์พุต (Vo) เป็น HIGH เท่ากับ Vcc เพื่อส่งสัญญาณอินเตอร์รัพท์ (Interrupt) ไปที่ไมโครคอนโทรลเลอร์เพื่อให้ไทม์เมอร์ (Timer) เริ่มจับเวลาและไทม์เมอร์จะหยุดนับเมื่อไมโครคอนโทรลเลอร์ได้รับสัญญาณอินเตอร์รัพท์จากวงจรเปรียบเทียบชุดที่ 2 ซึ่งใช้ LDR2 เป็น

เซ็นเซอร์แสง โดยกราฟของสัญญาณอินเตอร์รัพท์ Vo ที่ได้จากเอาต์พุตของวงจรเปรียบเทียบนั้น แสดงได้ดังรูปที่ 3.18

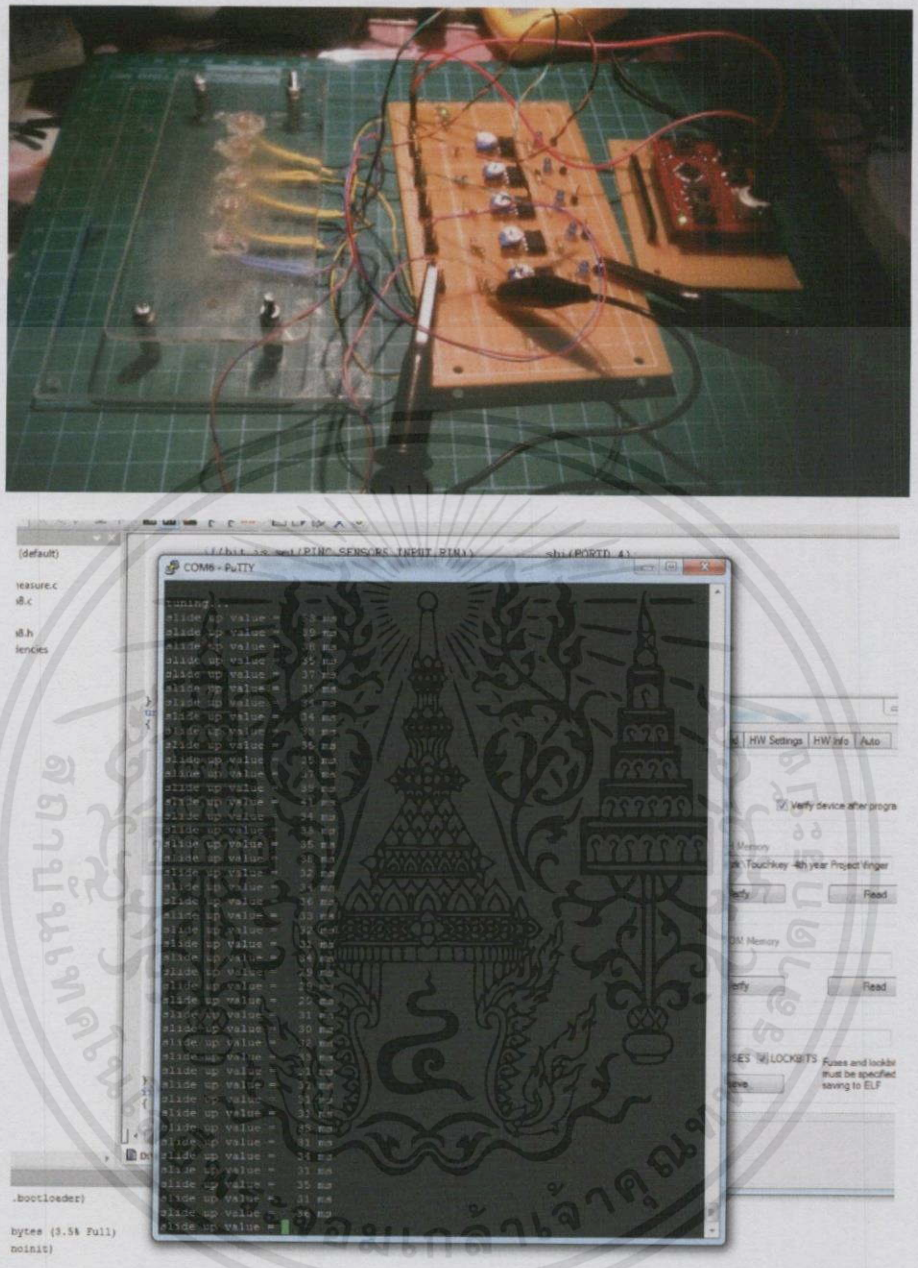


รูปที่ 3.18 กราฟสัญญาณอินเตอร์รัพท์ Vo ที่ได้จากเอาต์พุตของวงจรเปรียบเทียบ (สีฟ้า) เมื่อได้รับสัญญาณ Vi- (สีส้ม) ขณะที่นิ้วรูดผ่าน LDR

ค่าเวลาที่ได้จะถูกบันทึกไว้และทำการรีเซ็ทค่าของไทม์เมอร์เพื่อทำการจับเวลาการเคลื่อนที่ของนิ้วมือระหว่าง LDR2 กับ LDR3 และระหว่าง LDR3 กับ LDR4 และระหว่าง LDR4 กับ LDR5 โดยใช้หลักการเดียวกัน เมื่อนิ้วรูดจนผ่าน LDR5 ซึ่งเป็นจุดสิ้นสุดการรูดจะได้ช่วงเวลาด้วยกัน 4 ค่าคือ t_1, t_2, t_3, t_4 จากนั้นจะนำช่วงเวลาที่ได้ทั้งหมดไปคำนวณเพื่อหาความเร็วเฉลี่ยของนิ้วมือ ซึ่งได้กำหนดระยะทางระหว่างตัวต้านทานชนิดที่ไวต่อแสงแต่ละตัวไว้เท่ากับ 1.5 เซนติเมตร ดังนั้นความเร็วเฉลี่ยของการรูดนิ้วมือมีค่าเท่ากับ

$$\overline{V}_{finger} = \frac{1}{\mathrm{N}} \sum_{i=1}^{N} \left(\frac{1.5}{t_i} \right) \tag{3.8}$$

โดยที่ N = 4 และ $\overline{V}_{finger}$ คือความเร็วเฉลี่ยของการรูด หน่วยคือ *cm/s*

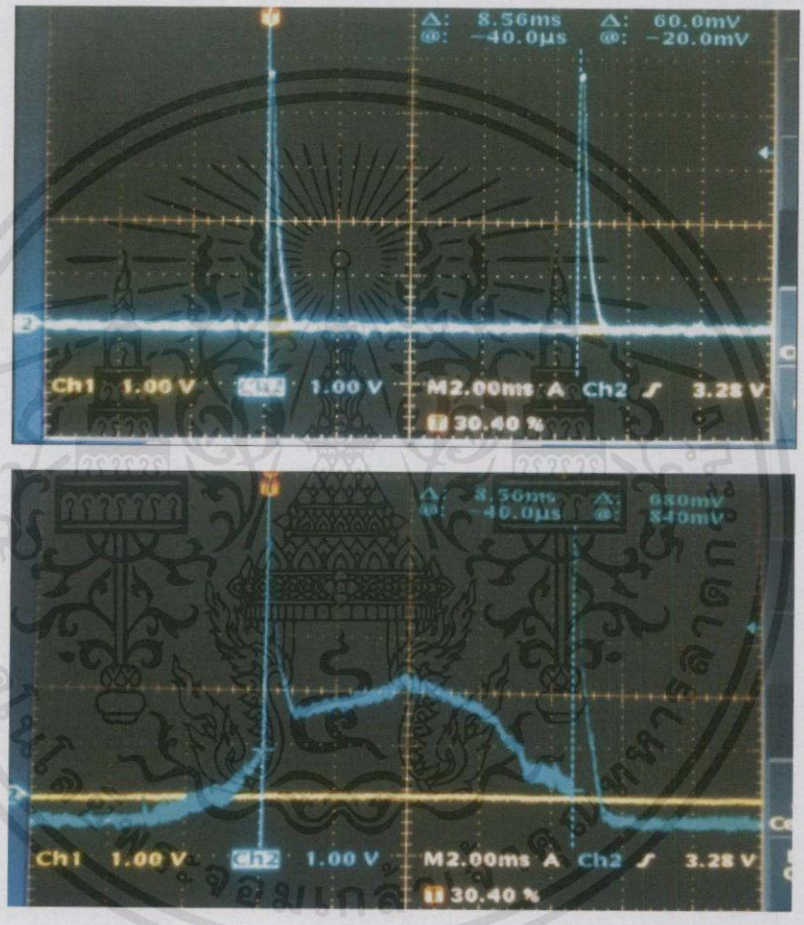


รูปที่ 3.19 (บน) ชุดอุปกรณ์ทดลองวัดความเร็วการรูดของนิ้วมือโดยใช้เซ็นเซอร์แสงชนิด LDR, (ล่าง) แสดงผลการจับเวลาการรูดนิ้วจากจุด LDR2 ถึง LDR4 ใช้เวลาเฉลี่ยประมาณ 32 ms

3.1.4 เทคนิคสำหรับการป้องกันผลจากสัญญาณรบกวน

เนื่องจากวงจรที่ใช้สำหรับการตรวจจับการสัมผัสด้วยความจุไฟฟ้า เป็นวงจรในลักษณะ “High-Impedance” ซึ่งอุปกรณ์ในวงจรมีความต้านทานสูงมากและใช้กำลังต่ำ ข้อเสียของวงจร

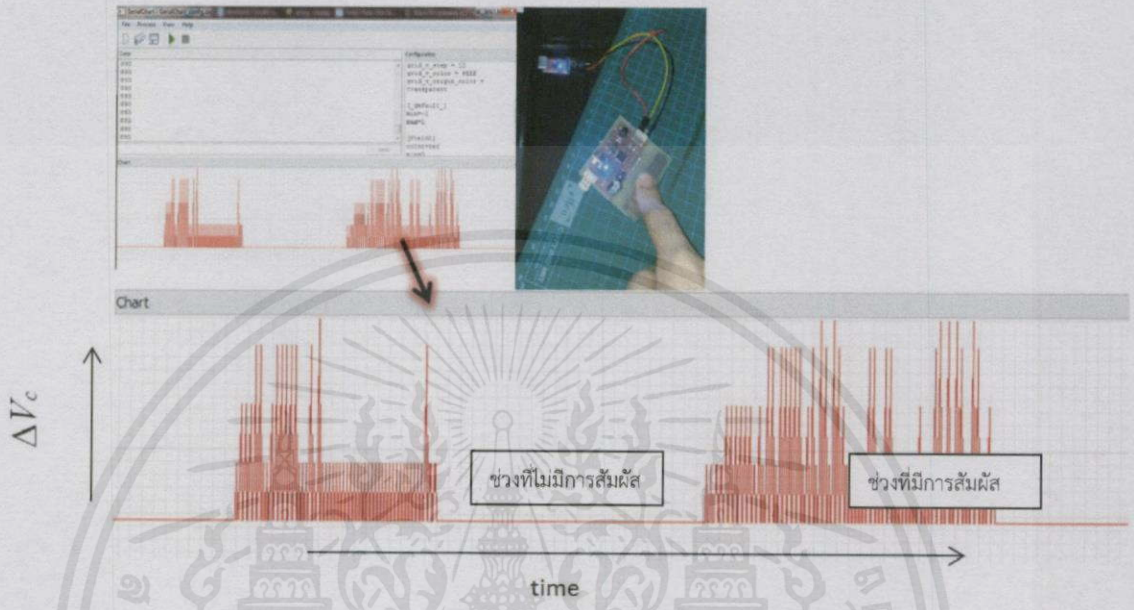
ในลักษณะนี้คือไม่สามารถทนต่อการเหนี่ยวนำสัญญาณรบกวนที่มีพลังงานสูงกว่าเข้ามาได้ง่าย ตัวอย่างเช่น การเหนี่ยวนำคลื่นแม่เหล็กไฟฟ้าจากระบบส่งไฟฟ้า (Power line) 50Hz ของร่างกาย จากการทดลองใช้งาน Keyboard touch sensor ต้นแบบที่ออกแบบโดยใช้วงจรดังรูปที่ 3.8 แล้วทำการวัดแรงดันตกคร่อมวงจร RC ผลที่เกิดขึ้นคือ ขณะที่นิ้วสัมผัสที่อิเล็กโทรดของเซ็นเซอร์แล้ววงจรเกิดการเหนี่ยวนำสัญญาณรบกวน 50Hz เข้าไปด้วย โดยแสดงผลที่เกิดขึ้นดังรูปที่ 3.20



รูปที่ 3.20 (บน)สัญญาณแรงดันตกคร่อมวงจร RC ขณะที่ไม่มีการแตะของนิ้ว, (ล่าง)สัญญาณแรงดันตกคร่อมวงจร RC ขณะที่มีการแตะของนิ้ว

จากรูปที่ 3.20 ผลของสัญญาณรบกวนที่เกิดขึ้นทำให้ขนาดของแรงดันตกคร่อมวงจร RC ขณะที่นิ้วสัมผัสถูกยกระดับแรงดัน (Offset voltage) ขึ้นด้วยสัญญาณ Sinusoid ความถี่ 50Hz เมื่อทำการสุ่มค่า (Sampling) แรงดันตกคร่อมวงจร RC ค่าจากการสุ่มที่ได้จึงมีการเปลี่ยนแปลงตลอดด้วยความถี่ 50 Hz ทำให้การประมวลเพื่อระบุการสัมผัสของนิ้วบนเซนเซอร์โดยใช้ชุดข้อมูล

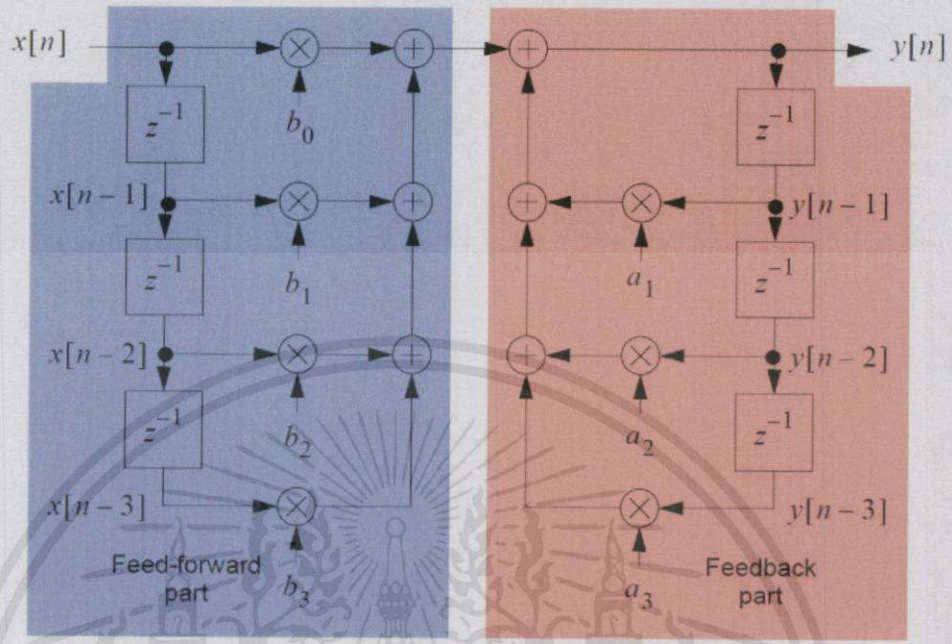
เหล่านี้เป็นไปได้ยาก โดยจะแสดงการพล็อตกราฟค่า ΔV_c ที่ได้จากการสุ่มค่าแรงดันและเปรียบเทียบกรณีที่มีนิ้วสัมผัสกับไม่มีนิ้วสัมผัสผ่านโปรแกรม Serial chart ดังรูปที่ 3.21



รูปที่ 3.21 กราฟค่า ΔV_c ที่ได้จากการสุ่มค่าแรงดันตกคร่อมวงจร RC ของ Keyboard touch sensor ต้นแบบ

จากรูปที่ 3.21 การตัดสินใจว่าอิเล็กโทรดนั้นถูกนิ้วสัมผัสหรือไม่ในช่วงที่มีการสัมผัสนั้นเป็นไปได้ยาก เนื่องจากมีการเปลี่ยนแปลงอย่างมากของค่า ΔV_c การแก้ไขนั้นไม่สามารถทำได้โดยใช้วงจรกรองอนาล็อก (Analog filter) ได้ เนื่องจากการเชื่อมต่อวงจรกรองเข้ากับวงจร RC จำเป็นต้องใช้วงจร Active filter มีความต้านทานขาเข้า (Input Impedance) สูงมากๆ เพื่อมิให้วงจรกรองเปลี่ยนแปลงรูปร่างสัญญาณของแรงดันดิสชาร์จวงจร RC แต่วงจร Active filter นั้นต้องใช้อุปกรณ์อิเล็กทรอนิกส์ค่อนข้างมากซึ่งไม่เหมาะสำหรับการออกแบบ PCB ของ Keyboard touch sensor ที่มีพื้นที่น้อยสำหรับการวางอุปกรณ์อิเล็กทรอนิกส์ ดังนั้นจึงเลือกวิธีกำจัดสัญญาณรบกวนทางซอฟแวร์โดยใช้ตัวกรองดิจิตอล (Digital filter)

ตัวกรองดิจิตอลที่ถูกนำมาใช้งานคือ IIR Low-pass filter โดย IIR ย่อมาจาก “Infinite Impulse Response” ตัวกรองนี้อยู่ในลักษณะของ Recursive filter ซึ่งค่าเอาท์พุตของตัวกรองจะเกิดจากผลรวมแบบเชิงเส้น (Linear combination) ของอินพุตที่รับเข้ามาปัจจุบัน อดีต และอดีตของเอาท์พุตด้วย เหมือนลักษณะของการป้อนกลับ (Feedback) ค่าทางอดีต โดยโครงสร้างตัวอย่างของ IIR filter นั้นจะแสดงไว้ดังรูปที่ 3.22



รูปที่ 3.22 Block diagram โครงสร้างของ 3rd Order IIR filter [3]

Transfer function ของ IIR filter โดยทั่วไปแล้วสามารถแสดงอยู่ในรูปของสมการดังนี้

$$H_{(z)} = \frac{Y_{(z)}}{X_{(z)}} = \frac{\sum_{k=0}^{N} b_{[k]} z^{-k}}{1 + \sum_{k=1}^{N} a_{[k]} z^{-k}} \qquad (3.9)$$

โดย N คือ จำนวน Order ของตัวกรอง IIR filter

b_k คือ ค่า Coeffecient ของส่วน Feed-forward

a_k คือ ค่า Coeffecient ของส่วน Feedback

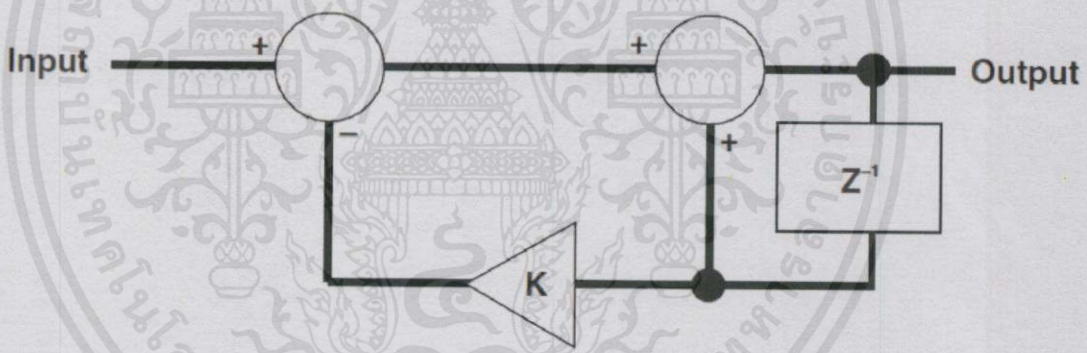
เหตุผลที่เลือกใช้งานตัวกรองประเภท IIR filter เนื่องจาก การออกแบบนั้นมีพื้นฐานมาจาก Analog filter ซึ่งสามารถออกแบบให้มีการตอบสนองความถี่ในรูปแบบต่างๆได้ เช่น Low-pass filter, High-pass filter, Band pass filter, Band stop filter เป็นต้น ข้อดีที่สำคัญที่สุดสำหรับเหตุผลการเลือก คือ IIR filter มีการตอบสนองความถี่ที่ดีและแม่นยำกว่า FIR filter (Finite Impulse Response) กรณีที่จำนวน Order ของวงจรกรองเท่ากันทั้งสอง ซึ่งหมายถึง IIR filter จะใช้พารามิเตอร์ต่างๆในการคำนวณเอาท์พุตน้อยกว่าแบบ FIR filter ทำให้ใช้เวลาในการทำงาน

น้อยกว่ามากและมีประสิทธิภาพ ซึ่งเหมาะสมสำหรับ Keyboard touch sensor ที่ต้องการความเร็วการทำงานสูงๆ ในการตอบสนองการรูดของนิ้วอย่างรวดเร็ว

การใช้งานตัวกรองความถี่เพื่อกำจัดสัญญาณรบกวน 50 Hz ออกไปนั้น ชุดข้อมูลจากการสุ่มค่าที่จะนำไปผ่านตัวกรองจะต้ององค์ประกอบของสัญญาณรบกวนอยู่ด้วย ซึ่งจะต้องใช้การสุ่มค่าด้วยความถี่การสุ่มค่าอย่างน้อยเป็น 2 เท่าของความถี่สัญญาณรบกวนที่ต้องการกำจัดออก ตามทฤษฎีการสุ่มค่าของ Nyquist-Shannon

$$f_s \geq 2.f_{noise} \tag{3.10}$$

Notch filter นั้นเหมาะสมที่สุดสำหรับการกำจัดสัญญาณรบกวน ณ ความถี่เดียวที่ต้องการเท่านั้น ทำให้การลดทอนคุณภาพของสัญญาณอินพุตน้อยมาก แต่สำหรับการตรวจจับการสัมผัสอิเล็กโทรด ความถี่สัญญาณอินพุตหลักที่ได้จากการสัมผัสนั้นมีความถี่ที่ค่อนข้างต่ำ ดังนั้นการใช้ 1st Order IIR Low-pass filter แทนซึ่งมีความซับซ้อนของฟังก์ชั่นโอนถ่ายน้อยกว่าแบบ Notch filter น่าจะเพียงพอที่จะแยกสัญญาณรบกวนความถี่ 50 Hz ออกไปได้ โดย Block diagram ของ IIR Low-pass filter ที่นำมาใช้งานนั้นแสดงดังรูปที่ 3.23



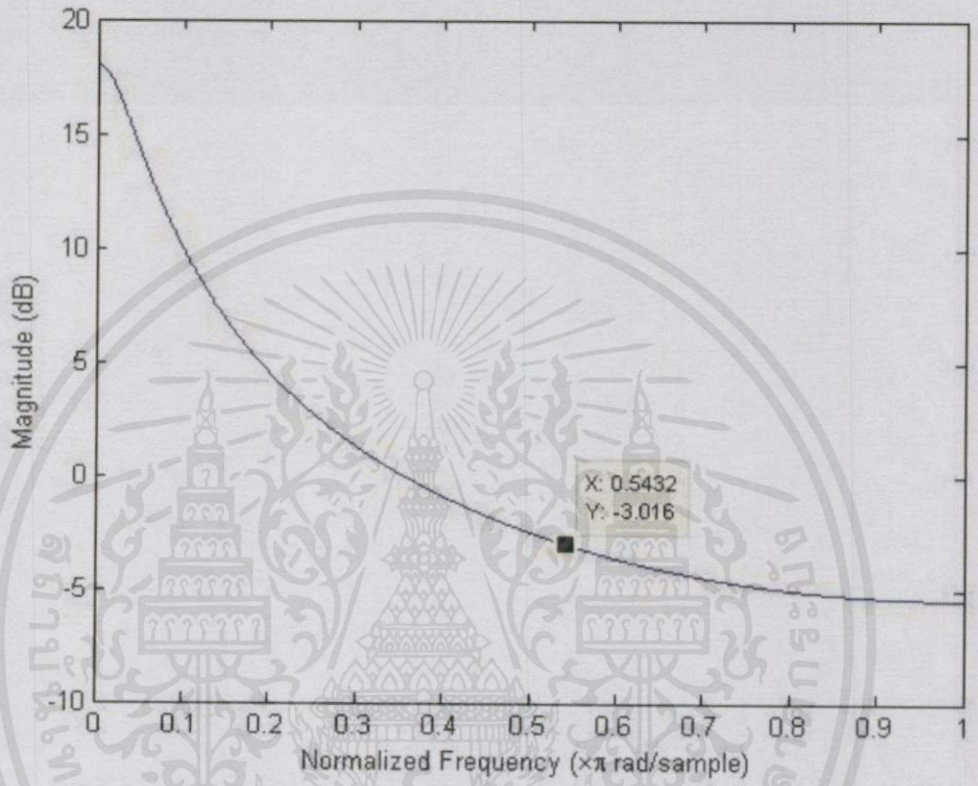
รูปที่ 3.23 Block diagram ของ IIR Low-pass filter ที่นำมาใช้งานใน Keyboard touch sensor

จาก Block diagram ของ IIR Low-pass filter ในรูปที่ 3.23 นำมาหาฟังก์ชั่นโอนถ่าย [8] (Transfer function) ของระบบได้เป็น

$$H_{(z)} = \frac{1}{1-(1-K)z^{-1}} \tag{3.11}$$

โดยที่ $1-K$ คือ ค่า Coeffecient ส่วน Feedback ของ IIR filter ที่ใช้งาน

วิธีพล็อตกราฟการตอบสนองทางความถี่ (Frequency Response) ของฟังก์ชั่นโอนถ่ายจากสมการที่ 3.11 นั้น จะใช้โปรแกรม MATLAB ในการแสดงการตอบสนองทางความถี่ โดยมีกราฟเป็นดังนี้



รูปที่ 3.24 กราฟการตอบสนองทางความถี่ของ IIR Low-pass filter โดยที่ K = 0.125

จากกราฟรูปที่ 3.24 สังเกตได้ว่า IIR Low-pass filter นี้จะมีอัตราขยายประกอบอยู่ซึ่งจะช่วยเพิ่มความไว (Sensitivity) ต่อการสัมผัสของเซ็นเซอร์และเมื่อนำค่า Normalized frequency ณ ตำแหน่ง -3dB มาคำนวณหา Cutoff frequency ของตัวกรอง เมื่อกำหนดให้ความถี่การสุ่มค่าของเซ็นเซอร์อยู่ที่ 110Hz และค่า K = 0.125

การเลือกค่า Coeffecient ที่ใช้ใน IIR Low-pass filter นั้นจะใช้วิธีการสังเกตการณ์ตอบสนองทางเวลา (Transient response) ของเอาต์พุตของตัวกรองแล้วทำการปรับค่า K จนได้การตอบสนองที่พอเหมาะ ซึ่งจะต้องให้เอาต์พุตที่มีการเปลี่ยนแปลงน้อยและมี Settling time ที่ต่ำ โดยที่ K จะต้องไม่เกิน 1 วิธีการสังเกตการณ์ตอบสนองทางเวลานั้นจะใช้วิธีการพล็อตกราฟผ่านโปรแกรม Serial Chart โดยแสดงผลลัพธ์ความแตกต่างของการใช้ IIR Low-pass filter กับการไม่ใช้ ดังรูปที่ 3.25

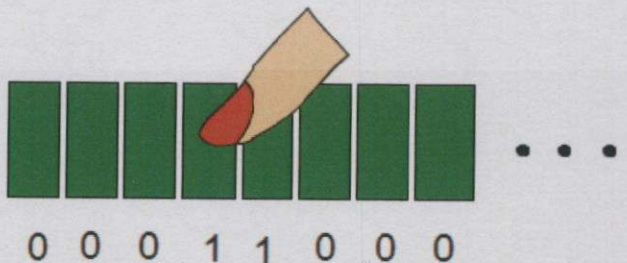
รูปที่ 3.25 กราฟแสดงความแตกต่างระหว่างข้อมูลการวัดที่ไม่ผ่าน IIR Low-pass filter กับผ่าน IIR Low-pass filter

3.1.5 การเข้ารหัสการระบุตำแหน่งของนิ้วบนแถบสไลเดอร์ของ Keyboard touch sensor

การแสดงตำแหน่งของการสัมผัสบน Keyboard touch sensor สามารถระบุได้โดยการเข้ารหัสดังรูปที่ 3.27 สถานะของแต่ละอิเล็กโทรดบนแถบสไลเดอร์จะมีเพียง 2 สถานะ คือ ถูกแตะกับไม่ถูกแตะ ซึ่งถูกแทนด้วยตัวเลข 1 และ 0 ตามลำดับ

รูปที่ 3.26 กราฟแสดงการพิจารณาว่าอิเล็กโทรดถูกสัมผัสหรือไม่

นำสัญญาณจากอิเล็กโทรดที่ผ่านการกรองมาพิจารณาโดยการกำหนดค่าขีดเริ่ม (Threshold) ถ้าระดับของสัญญาณสูงเกินค่าแรงดันขีดเริ่มแสดงว่าอิเล็กโทรดมีการสัมผัสและให้ผลลัพธ์เป็น 1 แต่ถ้าระดับของสัญญาณต่ำกว่าค่าขีดเริ่มแสดงว่าอิเล็กโทรดไม่มีการสัมผัสและให้ผลลัพธ์เป็น 0



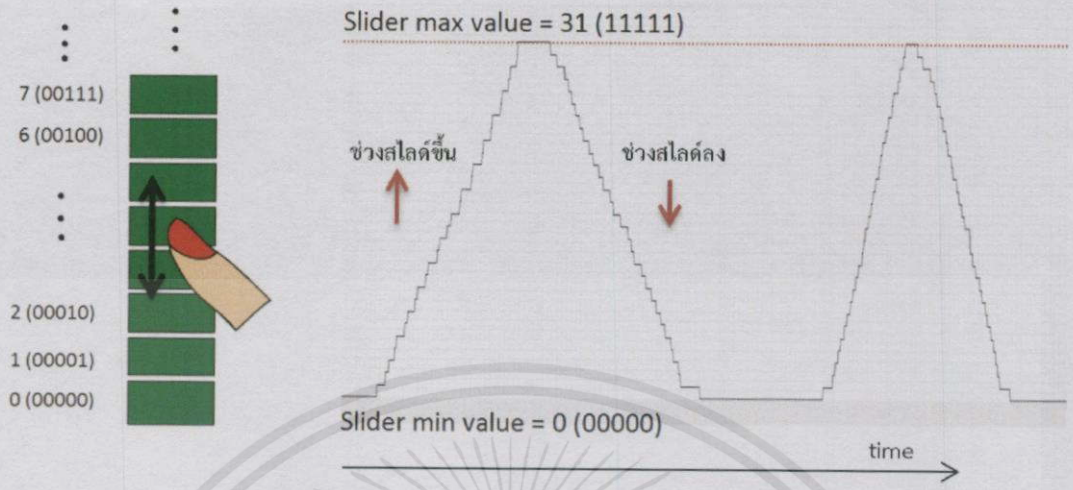
รูปที่ 3.27 ภาพแสดงสถานการณ์สัมผัสของอิเล็กโทรดบนแถบสไลเดอร์ [8]

จากรูปที่ 3.27 ตำแหน่งการสัมผัสที่ละเอียดที่สุดที่แถบสไลเดอร์สามารถทำได้ คือ ตำแหน่งการสัมผัสระหว่างอิเล็กโทรด 2 ชิ้น ดังนั้น หากแถบสไลเดอร์ประกอบด้วยอิเล็กโทรดจำนวน 16 ชิ้น ก็จะสามารถระบุตำแหน่งการสัมผัสได้ถึง 32 ระดับ โดยสถานะของแถบสไลเดอร์จะต้องเก็บอยู่ในลักษณะของ Thermometer code (หรือเรียกว่า Unary code) ขนาด 32 บิตซึ่งหากนำไปใช้ในการสื่อสาร I2C จะต้องส่งข้อมูลถึง 4 ไบต์หรือมากกว่าในการระบุตำแหน่งการสัมผัส อาจทำให้การสื่อสารระหว่าง Keyboard touch sensor กับบอร์ดประมวลผลหลักที่ใช้สำหรับสร้างชุดข้อมูล MIDI ช้าลงได้ ตัวอย่างลักษณะของการเข้ารหัสแบบ Thermometer code แสดงดังตารางรูปที่ 3.28

	Binary			Thermometer Code						
Decimal	b_1	b_2	b_3	d_1	d_2	d_3	d_4	d_5	d_6	d_7
0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	1
2	0	1	0	0	0	0	0	0	1	1
3	0	1	1	0	0	0	0	1	1	1
4	1	0	0	0	0	0	1	1	1	1
5	1	0	1	0	0	1	1	1	1	1
6	1	1	0	0	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1

รูปที่ 3.28 แสดงตัวอย่างลักษณะของ Thermometer code ขนาด 3-bit binary

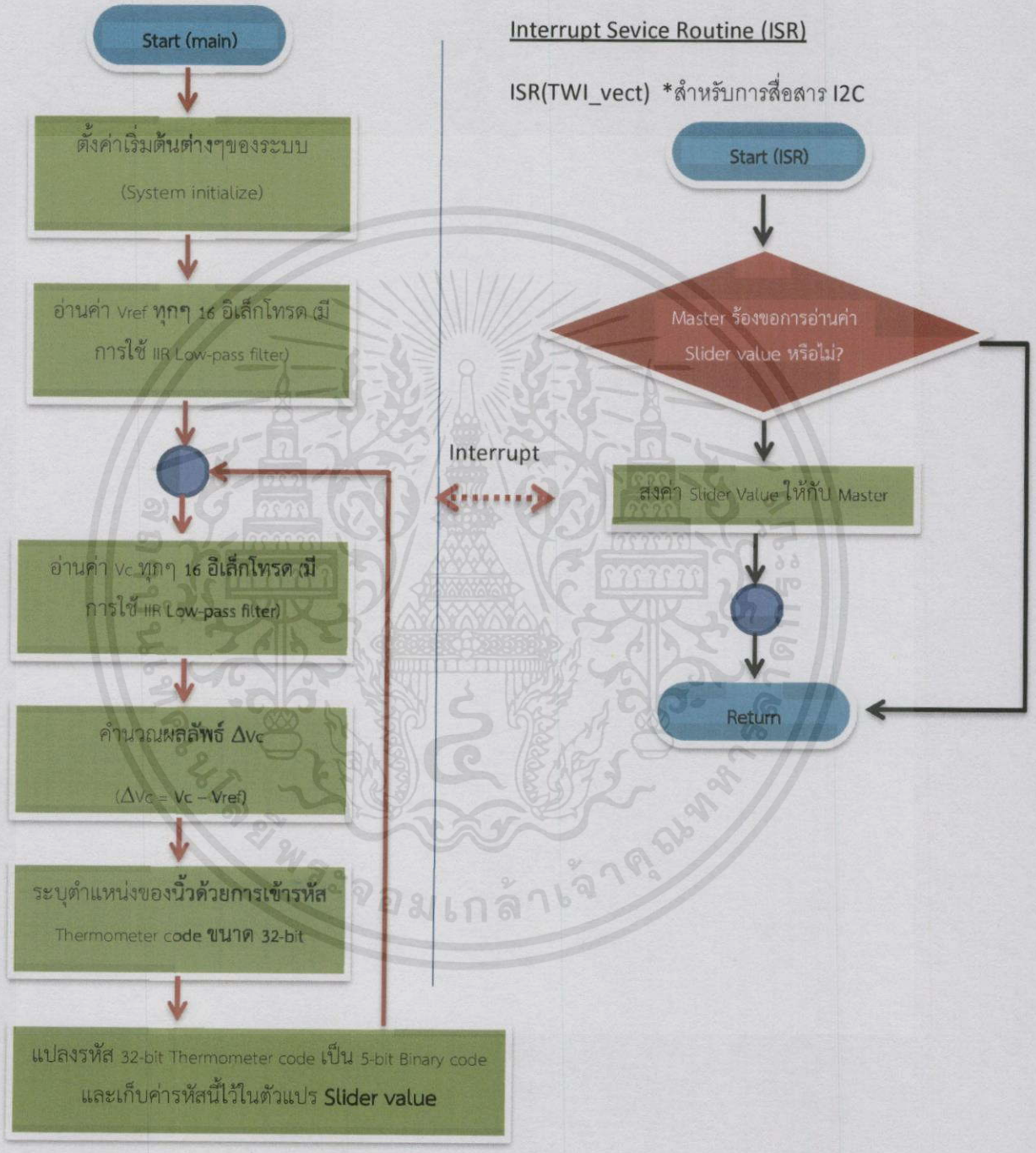
ดังนั้นมีความจำเป็นที่จะต้องถอดรหัสเพื่อให้จำนวนบิตที่ใช้ในการสื่อสารน้อยลง โดยทำการแปลงชุดข้อมูลของ Thermometer code ให้เป็นชุดข้อมูล Binary code ดังในตารางรูปที่ 3.27 ดังนั้นจาก Thermometer code ของแถบสไลเดอร์ที่ต้องใช้ 32 บิตในการระบุตำแหน่ง เมื่อเปลี่ยนเป็นชุดข้อมูลแบบ Binary แล้ว สามารถลดขนาดบิตของข้อมูลเหลือเพียง 5 บิตเท่านั้น จากนั้นนำข้อมูล Binary ที่ได้ไปสร้างฟังก์ชั่นการระบุตำแหน่งนิ้วบนสไลเดอร์ในลักษณะของกราฟขั้นบันไดดังรูปที่ 3.29



รูปที่ 3.29 แสดงการสร้างฟังก์ชั่นการระบุตำแหน่งของนิ้วบนสไลเดอร์แบบขั้นบันไดขณะที่มีการรูดนิ้วขึ้น-ลงบนแถบสไลเดอร์ [8]

3.1.6 โปรแกรมของ Keyboard touch sensor

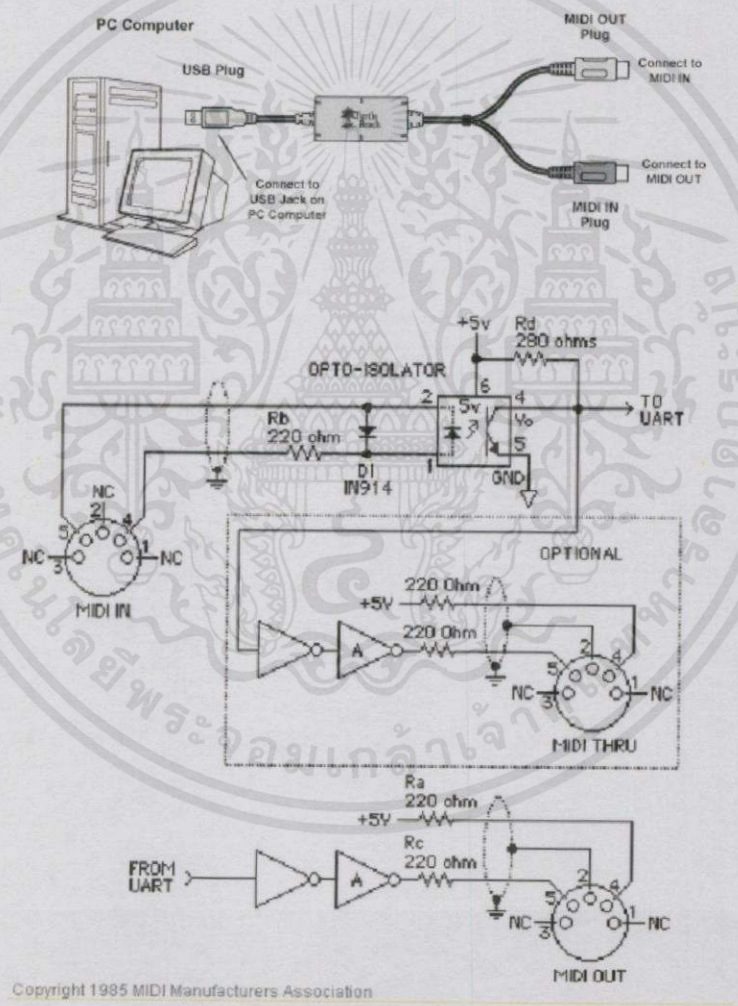
การทำงานของ Keyboard touch sensor สามารถอธิบายด้วยแผนผัง Flow chart ดังนี้



รูปที่ 3.30 Flow chart แสดงการทำงานโปรแกรมของ Keyboard touch sensor

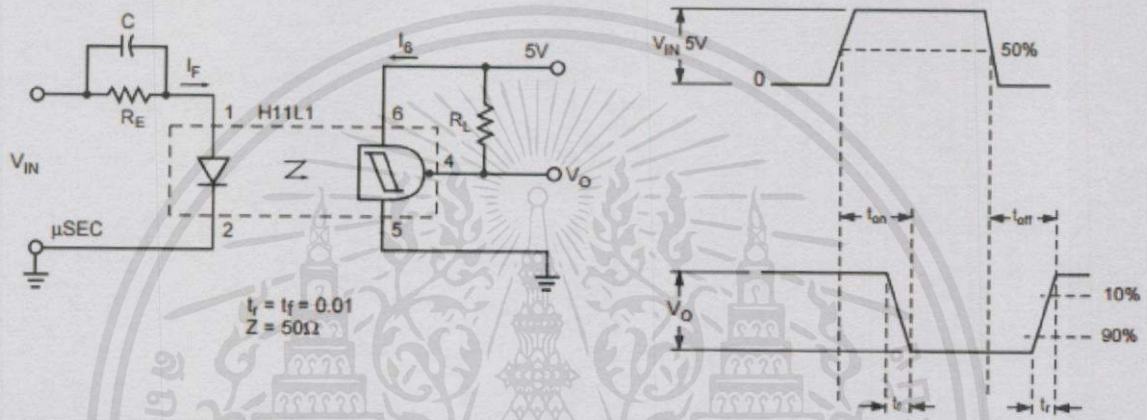
3.2 วงจรแปลงสัญญาณจาก TTL level เป็น MIDI (MIDI data signal generator)

สัญญาณ MIDI ที่ได้จากเครื่องดนตรีสามารถนำไปใช้งานได้หลายอย่าง อาทิเช่น ใช้เชื่อมต่อและควบคุมกับเครื่องดนตรีดิจิตอลเครื่องอื่นหรือนำไปต่อกับอุปกรณ์ที่รองรับการเชื่อมต่อ MIDI การ์ดเสียง (Sound card) ของคอมพิวเตอร์นั้นมีความสามารถต่อการรองรับการเชื่อมต่อ MIDI ได้ แต่ไม่มีพอร์ตสำหรับการเชื่อมต่อสัญญาณ MIDI โดยตรง การรับส่งสัญญาณไฟฟ้าที่มาจากพอร์ต MIDI จะอยู่ในรูปแบบของ Current loop ดังนั้นจึงต้องมีวงจรที่แปลงสัญญาณจากอุปกรณ์ MIDI (Current loop) กับสัญญาณระดับ TTL เพื่อเชื่อมต่ออุปกรณ์ MIDI ที่ใช้ในการสังเคราะห์เสียงโน้ต หรือเชื่อมต่อกับคอมพิวเตอร์ผ่าน MIDI/USB converter เข้ากับบอร์ดประมวลผลหลัก MIDI data generator



รูปที่ 3.31 (บน) การเชื่อมต่อ MIDI/USB converter, (ล่าง) วงจรภาคอินพุตและเอาต์พุตมาตรฐานของอุปกรณ์ MIDI [4]

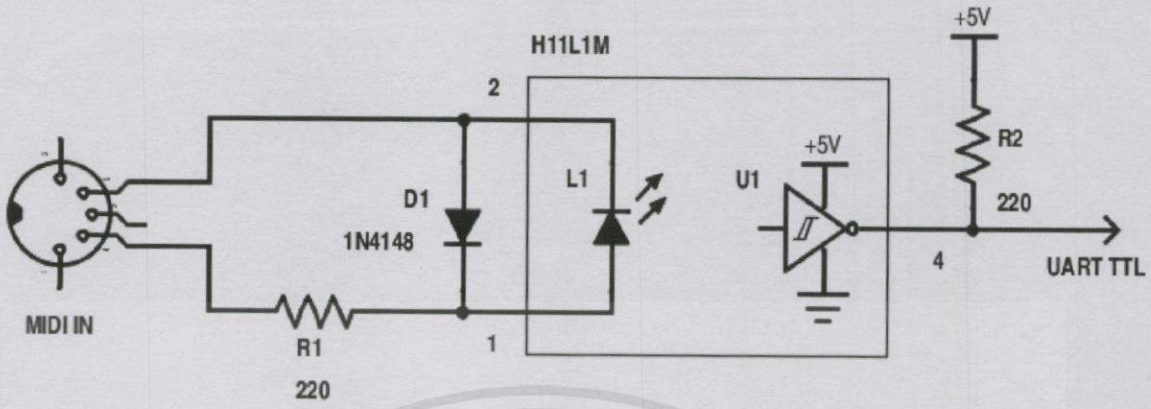
วงจรแปลงสัญญาณ MIDI เป็นสัญญาณ TTL level นั้นจำเป็นต้องใช้งาน Opto-isolator ซึ่งเหตุผลได้อธิบายไว้ในหัวข้อที่ 2.5.2 โดยจะต้องเลือก Opto-isolators ที่สามารถตอบสนองต่ออัตราการส่งข้อมูลของสัญญาณ MIDI ซึ่งมีค่าเท่ากับ 32 ไมโครวินาทีต่อบิตได้ทัน (31250 bps) โดย Opto-isolators เบอร์ H11L1M นั้นมีเวลาในการสวิตช์ตามรูปที่ 3.32 ช่วงเวลาเทิร์นออนนั้นมีค่าเท่ากับ 1 ไมโครวินาทีและช่วงเวลาเทิร์นออฟนั้นมีค่าเท่ากับ 2 ไมโครวินาที เมื่อนำมาเปรียบเทียบกับอัตราการส่งข้อมูลของสัญญาณMIDIแล้วพบว่าช่วงเวลาเทิร์นออนและเทิร์นออฟนั้นมีความเร็วการตอบสนองเพียงพอกับความเร็วของสัญญาณ MIDI และภาคเอาท์พุทของ H11L1M นั้นเป็นวงจร Schmitt trigger อยู่ภายในซึ่งมีความสามารถในการทนทานต่อสัญญาณรบกวนได้ดี



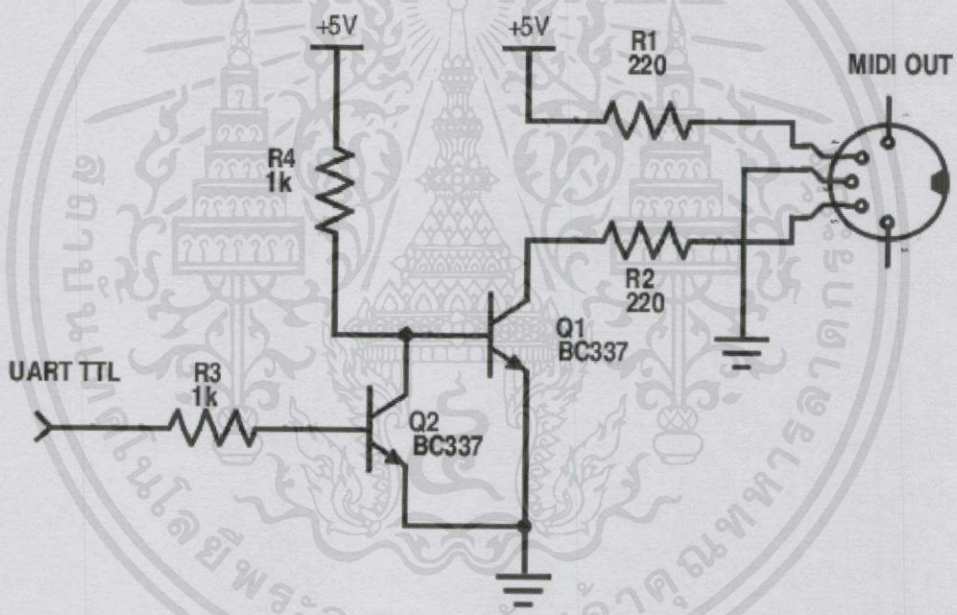
TRANSFER CHARACTERISTICS

AC Characteristics		Test Conditions	Min	Typ	Max	Units
SWITCHING SPEED						
Turn-On time	t_{on}	R_L=270Ω, V_{CC}=5V, I_F=$I_{F(on)}$, T_A=25°C		1.0		μs
				0.65		
					4	
Fall Time	t_f	R_L=270Ω, V_{CC}=5V, I_F=$I_{F(on)}$, T_A=25°C		0.1		μs
				05		
				0.1		
Turn-Off Time	t_{off}	R_L=270Ω, V_{CC}=5V, I_F=$I_{F(on)}$, T_A=25°C		2.0		μs
				1.2		
					4	
Rise time	t_r	R_L=270Ω, V_{CC}=5V, I_F=$I_{F(on)}$, T_A=25°C		0.1		μs
				0.07		
				0.1		
Data Rate				1.0		MHz

รูปที่ 3.32 วงจรภายในและคุณสมบัติการสวิทช์ของ IC Opto-isolators เบอร์ H11L1



รูปที่ 3.33 วงจร MIDI/TTL converter

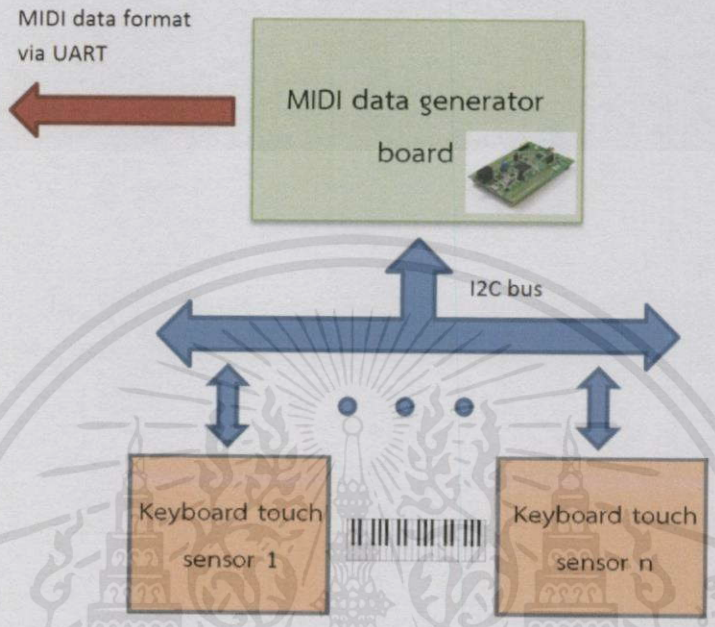


รูปที่ 3.34 วงจร TTL/MIDI converter

การสื่อสารระหว่างบอร์ดประมวลผลหลักสำหรับสร้างรหัส MIDI กับอุปกรณ์เอาต์พุต MIDI สำหรับสร้างเสียงโน้ตนั้นจะใช้งานวงจร TTL/MIDI converter จากรูปที่ 3.34 โดยการใช้ทรานซิสเตอร์ Q1 และ Q2 นั้นเป็นการแทนที่ Inverter ของวงจรมาตรฐาน ในรูปที่ 3.33 ส่วนวงจร MIDI/TTL converter นั้นถูกนำมาใช้สำหรับการ Debug ผ่านการสื่อสารรูปแบบ MIDI เพื่อตรวจสอบความถูกต้องของการแปลงข้อมูลต่างๆเท่านั้น

3.3 บอร์ดประมวลผลหลักสำหรับสร้างชุดข้อมูล MIDI (MIDI data generator)

3.3.1 โครงสร้างและหลักการทำงาน

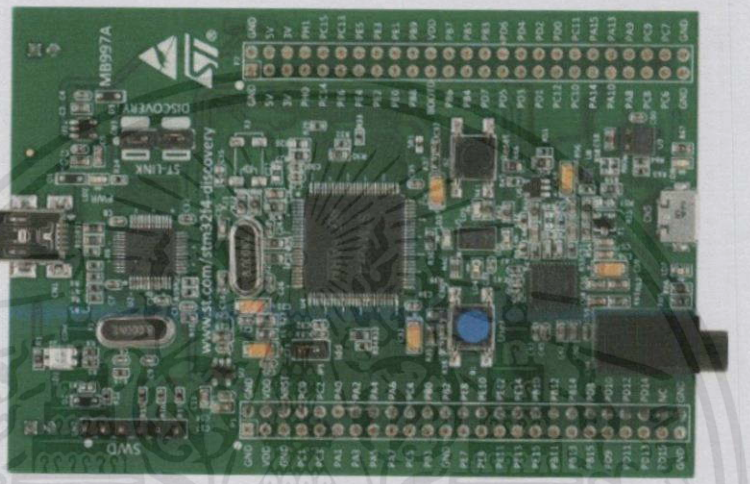


รูปที่ 3.35 Block diagram แสดงการเชื่อมต่อของบอร์ด MIDI data generator

บอร์ดประมวลหลักสำหรับสร้างชุดข้อมูล MIDI หรือสำหรับโครงงานนี้จะเรียกว่าบอร์ด MIDI data generator จะทำหน้าที่ในการส่งชุดข้อมูล MIDI โดยใช้การสื่อสารอนุกรมแบบ UART ไปยังอุปกรณ์ MIDI เอาต์พุตเพื่อใช้ขับเสียงโน้ต โดยชุดข้อมูล MIDI นั้นจะประกอบด้วยคุณลักษณะของเสียงโน้ต (เช่น Pitch bend, Vibrato เป็นต้น) ที่สอดคล้องกับลักษณะการรูดนิ้วบน Keyboard touch sensor โดยสามารถวิเคราะห์ลักษณะการรูดนิ้วได้จากการอ่านค่าตำแหน่งนิ้วจาก Keyboard touch sensor ทุกๆตำแหน่งโน้ตผ่านบัส I2C และการทำงานโดยรวมของบอร์ด MIDI data generator จะต้องเร็วเพียงพอที่ไม่ทำให้เกิดการดีเลย์ของเวลา (Delay time) การสร้างเสียงโน้ตเอาต์พุตมากเกินไปจนทำให้ผู้เล่นดนตรีรู้สึกถึงการหน่วงได้ ซึ่งจะทำการเล่นดนตรีเป็นไปได้ยาก ดังนั้นไมโครคอนโทรเลอร์ที่ใช้งานกับบอร์ด MIDI data generator จะต้องมีความเร็วในการประมวลผลสูงเพื่อที่จะจัดการกับข้อมูลของตำแหน่งนิ้วที่ได้รับมาและสร้างชุดข้อมูล MIDI เพื่อกำเนิดเสียงโน้ตให้เสร็จทันเวลาของคาบการสุ่มค่าตำแหน่งนิ้วของ Keyboard touch sensor

นอกจากการเลือกใช้ไมโครคอนโทรเลอร์ที่มีความเร็วในการประมวลผลสูงเพื่อลดการเกิดดีเลย์แล้ว ส่วนโปรแกรมของบอร์ด MIDI data generator จะต้องมีเทคนิคการจัดการทรัพยากร (Resource) ที่มีอยู่ให้ระบบสามารถตอบสนองต่อผู้ใช้ได้ทันเวลาและไม่เกิดการล้มเหลวของระบบ

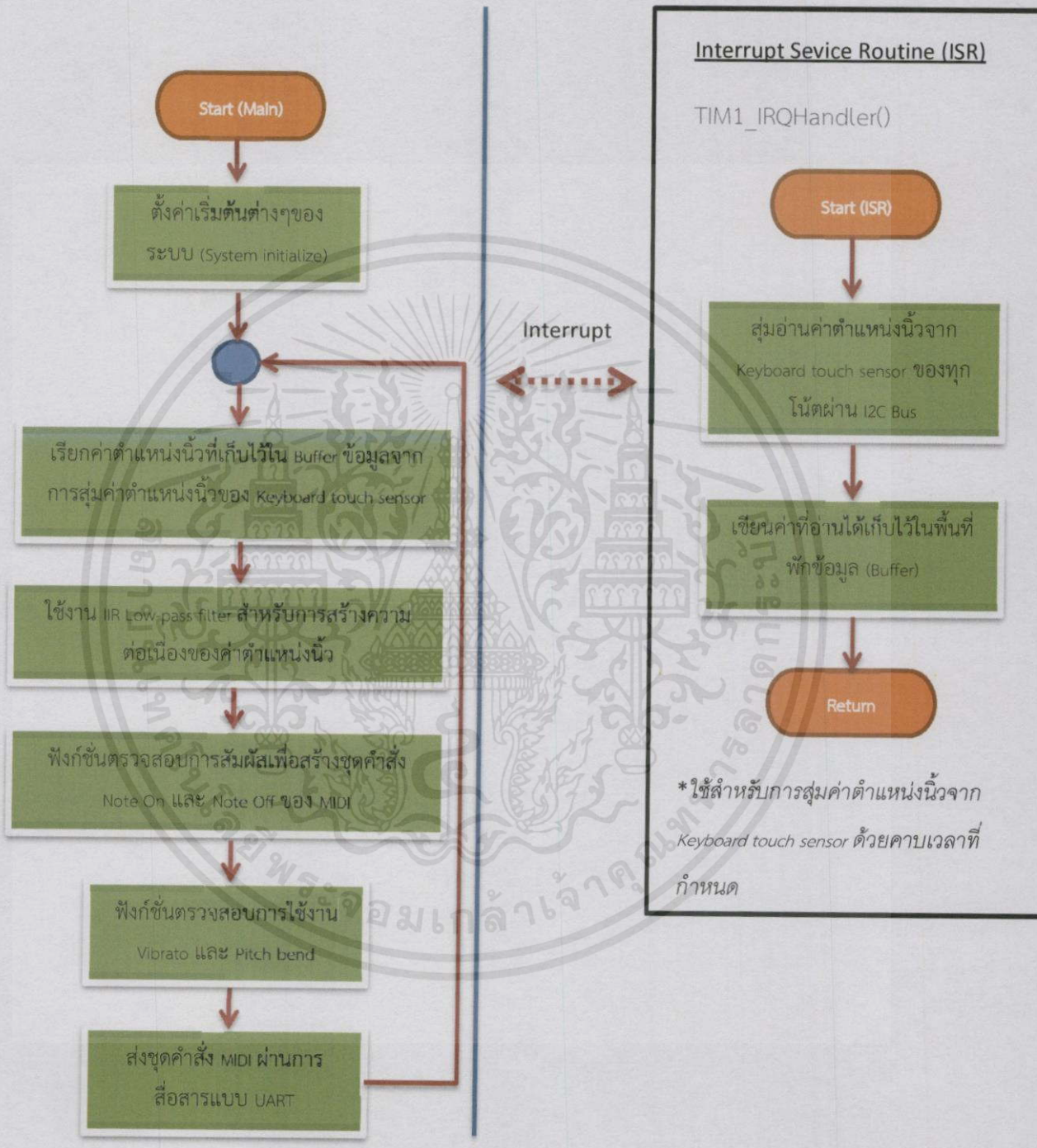
ไมโครคอนโทรเลอร์ที่เลือกใช้กับบอร์ด MIDI data generator คือ STM32F407VGT6 จากบริษัท STmicroelectronic โครงสร้างสถาปัตกรรมคือ ARM Cortex-M4 32-bit ซึ่งมีความเร็วสูงสุดอยู่ที่ 168MHz ต่อ 210 DMIPS การติดตั้งเพื่อนำไปใช้งานนั้น ได้เลือกใช้เป็นลักษณะของบอร์ด Evaluation แทนการใช้งานแบบชิฟเพื่อความสะดวกในการพัฒนา โดยบอร์ดที่เลือกคือ STM32F4DISCOVERY รูปร่างลักษณะและส่วนประกอบที่มีอยู่บนบอร์ดได้แสดงไว้ดังรูปที่ 3.36



- STM32F407VGT6 microcontroller featuring 1 MB of Flash memory, 192 KB of RAM in an LQFP100 package
- On-board ST-LINK/V2 with selection mode switch to use the kit as a standalone ST-LINK/V2 (with SWD connector for programming and debugging)
- Board power supply: through USB bus or from an external 5V supply voltage
- External application power supply: 3V and 5V
- LIS302DL or LIS3DSH, ST MEMS motion sensor, 3-axis digital output accelerometer
- MP45DT02, ST MEMS audio sensor, omnidirectional digital microphone
- CS43L22, audio DAC with integrated class D speaker driver
- Eight LEDs:
  - LD1 (red/green) for USB communication
  - LD2 (red) for 3.3V power on
  - Four user LEDs, LD3 (orange), LD4 (green), LD5 (red) and LD6 (blue)
  - 2 USB OTG LEDs LD7 (green) VBus and LD8 (red) over-current
- Two pushbuttons (user and reset)
- USB OTG with micro-AB connector
- Extension header for LQFP100 I/Os for quick connection to prototyping board and easy probing

รูปที่ 3.36 ลักษณะและส่วนประกอบของบอร์ด STM32F4DISCOVERY

โปรแกรมการทำงานโดยรวมของบอร์ด MIDI data generator สามารถอธิบายด้วยแผนผัง Flow chart ดังนี้



รูปที่ 3.37 Flow chart การทำงานโดยรวมของโปรแกรมบนบอร์ด MIDI data generator

จาก Flow chart ในรูปที่ 3.37 ส่วนของโปรแกรมหลัก (Main) นั้นจะทำงานในลักษณะ Polling โดยมีการวนลูปตรวจสอบการเรียกใช้งานฟังก์ชั่นต่างๆ เช่น การตรวจสอบการสัมผัส, การตรวจสอบการใช้งาน Vibrato และ Pitch bend เป็นต้น และจบลูปด้วยการส่งชุดคำสั่ง MIDI ผ่าน UART หากมีการเรียกใช้ฟังก์ชั่นดังกล่าวเกิดขึ้น สำหรับการสุ่มค่าตำแหน่งของนิ้วจาก Keyboard touch sensor ผ่านบัส I2C นั้น จะทำการสุ่มค่าด้วยคาบเวลาการสุ่มคงที่โดยจะใช้การอินเตอร์รัพท์ (Interrupt) ของวงจร TIMER ใน STM32F407VGT ซึ่งสามารถกำหนดคาบเวลาในการสุ่มค่าได้ตามต้องการ ความถี่ที่ใช้ในการสุ่มค่าของบอร์ดหลักต้องมีค่ามากกว่าคาบการสุ่มค่าตำแหน่งนิ้วของ Keyboard touch sensor

$$f_{S(\text{MIDI data generator})} \geq f_{S(\text{Keyboard touch sensor})} \qquad (3.12)$$

3.3.2 การควบคุมคุณลักษณะทางเสียงของอุปกรณ์ MIDI

เสียงโน้ตที่ถูกเล่นจากเครื่องดนตรีหรืออุปกรณ์ MIDI นั้นสามารถควบคุมได้ผ่านการส่งชุดคำสั่งตามโปรโตคอล (Protocol) มาตรฐานของ MIDI 1.0 ในลักษณะของ UART ซึ่งได้อธิบายไว้ในหัวข้อที่ 2.5 โดยชุดคำสั่งที่ใช้หลักๆในระบบของโครงงานนี้ได้แก่

1) Note On และ Note Off

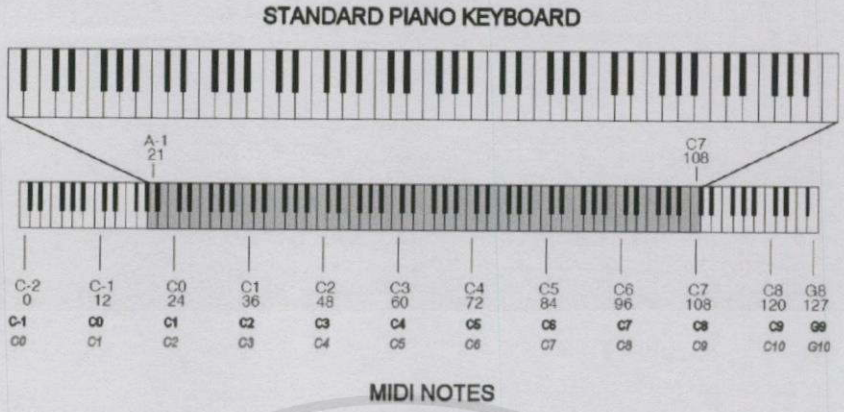
เป็นชุดคำสั่งแสดงการเล่นโน้ตและการหยุดเล่นโน้ต ณ ตำแหน่งโน้ตใดๆ ซึ่งโดยปกติแล้วคำสั่งเหล่านี้จะปรากฏเมื่อเปลี่ยนแปลงสถานะของการกดคีย์เท่านั้น (หมายถึง การเริ่มกดหรือเริ่มปล่อยคีย์) การส่งไบต์ในชุดคำสั่งจะต้องส่ง Status byte และ Data byte เรียงตามลำดับ โดยมีรายละเอียดของแต่ละไบต์ดังนี้

Status byte : 1001 CCCC
Data byte 1 : 0PPP PPPP — คำสั่ง Note On
Data byte 2 : 0VVV VVVV

Status byte : 1000 CCCC
Data byte 1 : 0PPP PPPP — คำสั่ง Note Off
Data byte 2 : 0VVV VVVV

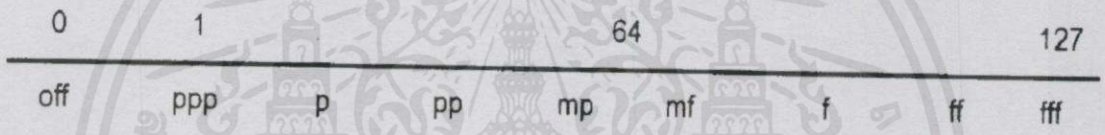
"CCCC" คือ MIDI channel มีค่าอยู่ในช่วง 0-15

"PPP PPPP" คือ ตำแหน่งของโน้ต (Pitch) โน้ตทุกตำแหน่งนั้นจะถูกระบุด้วยค่าตัวเลข มีค่าอยู่ในช่วง 0-127 โดยโน้ต Middle C จะมีค่าเท่ากับ 60 ตามภาพแสดงตำแหน่งของโน้ตของเปียโนดังรูปที่ xxx



รูปที่ 3.38 ภาพแสดงเปรียบเทียบค่า Pitch ของโน้ตใน MIDI เทียบกับตำแหน่งของโน้ตบนเปียโน

"VVV VVV" คือ ค่าความแรง ความดังในการเล่นโน้ต (Velocity) มีค่าอยู่ในช่วง 0-127 โดยมีการเปรียบเทียบค่าของ Velocity เทียบกับความดังเบาทางดนตรีสากลดังรูปที่ xxx



รูปที่ 3.39 แผนภาพเปรียบเทียบค่าของ Velocity เทียบกับความดังของโน้ตทางดนตรีสากล

```
void Note_On(unsigned char channel, unsigned char pitch, unsigned char velocity)
{
    if(velocity > 127)          velocity = 127;

    USART3_SendData(0x90|channel);
    USART3_SendData(pitch);
    USART3_SendData(velocity);
}
void Note_Off(unsigned char channel, unsigned char pitch)
{
    USART3_SendData(0x80|channel);
    USART3_SendData(pitch);
    USART3_SendData(0x00);
}
```

รูปที่ 3.40 ฟังก์ชั่นภาษา C ที่ใช้สำหรับคำสั่ง Note On และ Note Off

2) Pitch bend

เมื่อใช้งานชุดคำสั่งนี้ จะทำการเลื่อนความถี่ของโน้ตที่กำลังถูกเล่นอยู่ให้สูงขึ้นหรือต่ำลงไม่เกิน 1 ช่วงเสียงเต็ม (Tone หรือ Whole step) โดยมีความละเอียดสูงสุดอยู่ที่ 14 บิต ซึ่งจะมีช่วงของค่าอยู่ที่ 0x2000 ถึง 0x3FFF และเกิดจาก Data byte 2 ชุดรวมกัน ค่าของ Pitch bend ในสถานะเริ่มต้นที่ยังไม่มีการเลื่อนความถี่ของโน้ต (Center pitch) คือ

0x2000 ชุดคำสั่งประกอบด้วย 3 ไบต์ การส่งไบต์ในชุดคำสั่งจะต้องส่ง Status byte และ Data byte เรียงตามลำดับ โดยมีรายละเอียดของแต่ละไบต์ดังนี้

Status byte : 1110 CCCC
Data byte 1 : 0LLL LLLL
Data byte 2 : 0MMM MMMM
} คำสั่ง Pitch bend change

"CCCC" คือ MIDI channel มีค่าอยู่ในช่วง 0-15

"0LLL LLLL" และ "0MMM MMMM" คือ Least significant 7 บิต และ Most-significant 7 บิต ของค่า Pitch bend ขนาด 14 บิต ตามลำดับ

```
void Pitch_Bend(unsigned char channel,int bend_level)   //middle value at 0, max,min vaule at +8191,-8191
{   int pitch_value;
    int LSB;
    int MSB;

    pitch_value = 8192 + bend_level;
    if(pitch_value > 16383) pitch_value = 16383;    //overflow protection

    LSB = (pitch_value & 0b01111111);
    MSB = ((pitch_value >> 7) & 0b01111111);

    USART3_SendData(0xE0|channel);
    USART3_SendData((unsigned char)(LSB));
    USART3_SendData((unsigned char)(MSB));
}
```

รูปที่ 3.41 ฟังก์ชั่นภาษา C ที่ใช้สำหรับคำสั่ง Pitch bend

3) Vibrato

เมื่อใช้งานชุดคำสั่งนี้ จะทำการกวัดแกว่ง (Oscillation) ของความถี่โน้ตที่กำลังถูกเล่นอยู่ให้สูงขึ้นและต่ำลงเพียงเล็กน้อยไม่เกิน 1% ของความถี่กลางโดยที่สามารถปรับความถี่ของการกวัดแกว่งของความถี่โน้ตได้ด้วยความละเอียดอยู่ที่ 7 บิต (0-127) ความถี่การกวัดแกว่งของความถี่โน้ตของฟังก์ชั่น Vibrato ที่เหมาะสมสำหรับดนตรีจะอยู่ที่ 5-8 Hz ชื่อชุดคำสั่งการสร้างเสียง Vibrato ในมาตรฐาน MIDI นั้นจะเรียกว่าคำสั่ง Modulation wheel : Mod wheel สำหรับการส่งไบต์ในชุดคำสั่งจะต้องส่ง Status byte และ Data byte เรียงตามลำดับ โดยมีรายละเอียดของแต่ละไบต์ดังนี้

Status byte : 1011 CCCC
Data byte 1 : 0000 0001
Data byte 2 : 0mmm mmmm
} คำสั่ง Mod wheel

"CCCC" คือ MIDI channel มีค่าอยู่ในช่วง 0-15

"0mmm mmmm" และ "0mmm mmmm" คือ ค่า Mod wheel ความละเอียด 7 บิต

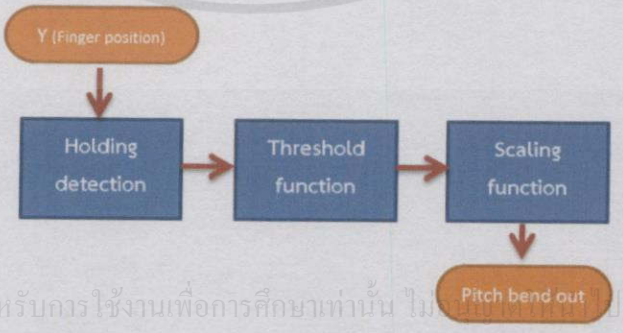
```
void Vibrato_Control(unsigned channel,int vibrato_level) //coarse mode with value from 0-127
{
    USART3_SendData(0xB0|channel);
    USART3_SendData(0x01);
    if(vibrato_level > 127)     vibrato_level = 127;
    USART3_SendData(vibrato_level);
}
```

รูปที่ 3.42 ฟังก์ชั่นภาษา C ที่ใช้สำหรับคำสั่งควบคุม Vibrato

3.3.3 การตรวจจับการใช้งาน Pitch bend และ Vibrato จากลักษณะการรูดนิ้ว

การตรวจจับการใช้งาน Pitch bend

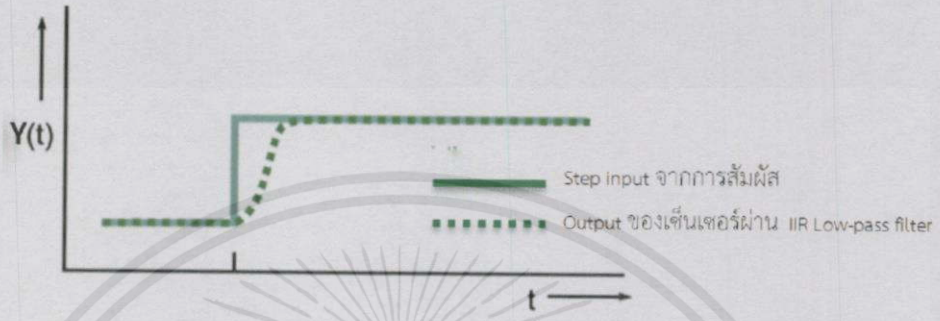
รูปแบบการควบคุม Pitch bend โดยใช้ค่าตำแหน่งสัมพันธ์ของนิ้วบน Keyboard touch sensor ที่มีลักษณะดังรูปที่ 3.28 ตามเป้าหมายของการออกแบบนั้น ต้องการให้มีลักษณะการควบคุม Pitch bend เหมือนที่ใช้ในการเล่นบน Violin ซึ่งสำหรับระบบนี้จะใช้ค่าการเปลี่ยนแปลงตำแหน่งนิ้วบนแถบสไลเดอร์จากการรูดเทียบกับตำแหน่งอ้างอิงจากการแตะคีย์จุดแรก ควบคุมขนาดการเบนของความถี่เสียงให้สูงขึ้นหรือต่ำลง โดยการกระตุ้นการทำงานของฟังก์ชั่น Pitch bend มีขั้นตอนดังต่อไปนี้



รูปที่ 3.43 Block diagram ของการตรวจจับการใช้งาน Pitch bend

1) **การตรวจจับการแตะค้างบนคีย์ (Holding detection)**

เนื่องจากการตอบสนองของค่าตำแหน่งนิ้วที่ได้จากการแตะนั้นค่อนข้างช้าอันเนื่องมาจากผลของ IIR low pass filter เมื่อทดสอบ Step response ของการแตะ ค่าตำแหน่งของนิ้วที่ได้จะค่อยๆเพิ่มจนถึงตำแหน่งที่มีการแตะโดยมีลักษณะดังรูปที่ 3.44

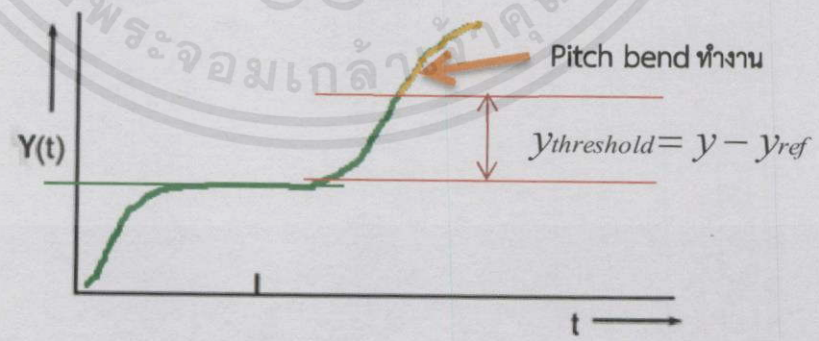


รูปที่ 3.44 กราฟการตอบสนอง Step input จากการแตะนิ้ว โดย $y(t)$ คือค่าตำแหน่งสัมพันธ์ของนิ้วบน Keyboard touch sensor

จากรูปที่ 3.44 การอ่านค่าตำแหน่งการแตะอ้างอิงนั้น จะต้องให้นิ้วแตะบนเซ็นเซอร์ค้างเป็นระยะเวลาหนึ่งก่อน และทำการอ่านค่า ณ จุดยอดของกราฟซึ่งมีอัตราการเปลี่ยนแปลงของตำแหน่งนิ้วเท่ากับ 0 หรือ $\frac{\partial y}{\partial t} = 0$

2) **Threshold function**

หลังจากที่สามารถวัดจุดอ้างอิงแล้ว การรูดขึ้นหรือลงเพื่อเลื่อนตำแหน่งของนิ้วจะต้องมากกว่าระยะขีดเริ่ม (Threshold) จึงจะเปิดการทำงานของ Pitch bend เพื่อป้องกันการเลื่อนตำแหน่งของนิ้วเพียงเล็กน้อยโดยที่ผู้เล่นดนตรีไม่ได้ตั้งใจ

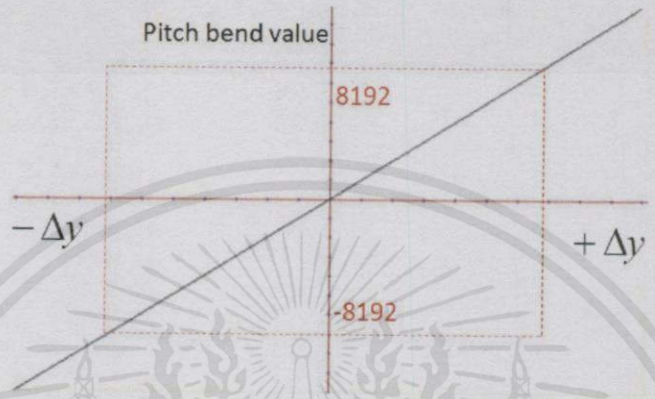


รูปที่ 3.45 กราฟตำแหน่งของนิ้วสำหรับอธิบายการทำงานของ Threshold detection

ค่าที่นำมาใช้ควบคุม Pitch bend ดังรูปที่ 3.45 คือ $\Delta y = y - y_{ref} \pm y_{threshold}$ โดยที่จะต้องอยู่ในเงื่อนไข $|y - y_{ref}| \geq y_{threshold}$

3) Scaling function

ฟังก์ชั่น Pitch bend ในชุดคำสั่ง MIDI นั้น มีช่วงอินพุตสำหรับการควบคุมอยู่ที่ -8192 ถึง 8192 ดังนั้นจะต้องมีการปรับช่วงค่าผลต่างของตำแหน่งนิ้วที่จากขั้นตอน Threshold detection ให้เหมาะสมกับช่วงอินพุตของฟังก์ชั่น Pitch bend



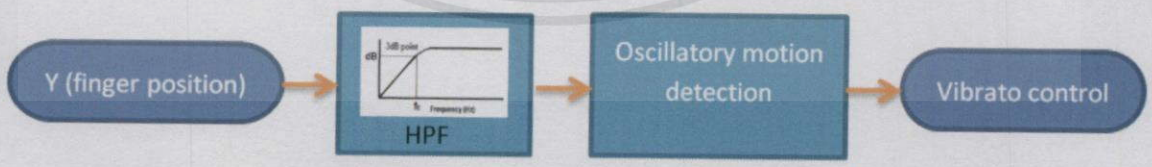
รูปที่ 3.46 กราฟความสัมพันธ์ระหว่างผลต่างของตำแหน่งนิ้ว (Δy) กับค่าอินพุตของฟังก์ชั่น Pitch bend

4) การส่งชุดคำสั่ง Pitch bend

นำค่าผลต่างของตำแหน่งนิ้วที่ได้จาก Scaling function มาใช้เป็นอินพุตสำหรับฟังก์ชั่น Pitch bend ของ MIDI ดังในหัวข้อที่ 3.3.2 เรื่องชุดคำสั่ง Pitch bend

<u>การตรวจจับการใช้งาน Vibrato</u>

รูปแบบการควบคุม Vibrato นั้น จะใช้เทคนิคการขยี้ปลายนิ้วกวัดแกว่งไปมาตามแนวยาวของแถบสไลเดอร์รอบจุดศูนย์กลางของโซนพื้นที่ด้านล่างของแถบสไลเดอร์ โดยการควบคุมจะมีลักษณะคล้ายกับการใช้ Vibrato บน Violin โดยโครงสร้างการทำงานของระบบมีดังนี้

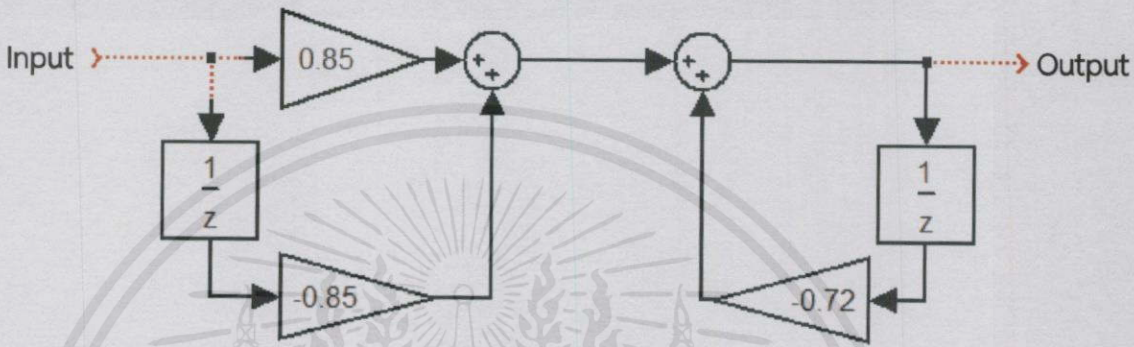


รูปที่ 3.47 Block diagram ของการตรวจจับการใช้งาน Vibrato

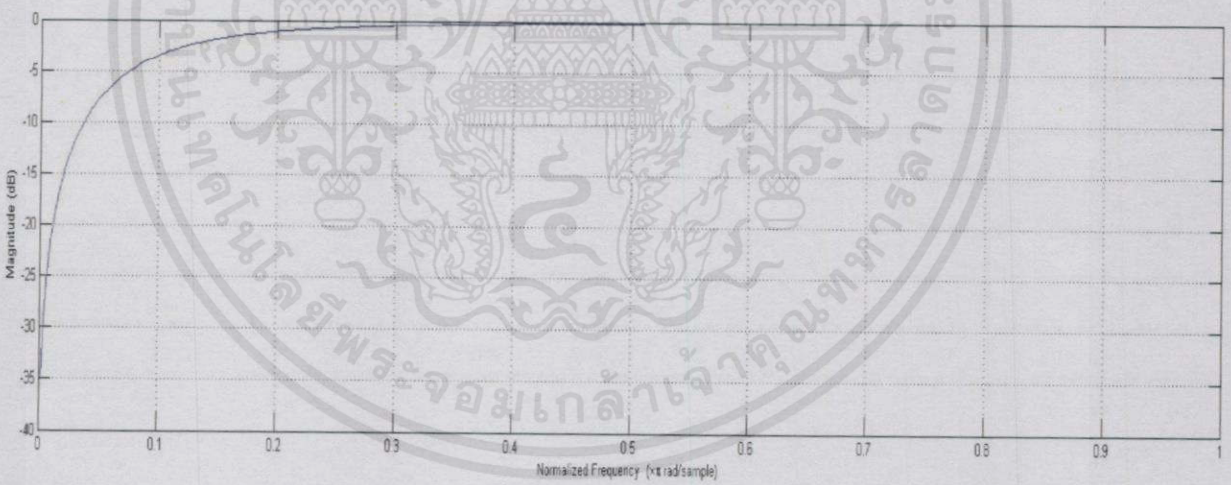
พื้นที่ของสไลเดอร์ที่ใช้การกระตุ้นการทำงานของ Vibrato นั้นจำอยู่ด้านล่างของแถบสไลเดอร์ซึ่งประกอบด้วยอิเล็กโทรด 6 ชิ้น ส่วนประกอบของขั้นตอนการตรวจจับการใช้งาน Vibrato นั้นประกอบด้วย 2 ส่วนดังนี้

1) High pass filter

การเลื่อนตำแหน่งของนิ้วขณะแตะค้างอยู่เพียงเล็กน้อย ก็สามารถกระตุ้นการทำงาน Vibrato โดยไม่ได้ตั้งใจได้ วิธีที่ใช้ป้องกันผลดังกล่าวคือการใส่ 1st Order IIR High-pass Filter ที่มี Cut-off frequency ต่ำๆสำหรับกรองความถี่ตำแหน่งที่มากจากการเลื่อนตำแหน่งเล็กน้อยของนิ้ว โดยเลือกความถี่อยู่ที่ 6 Hz



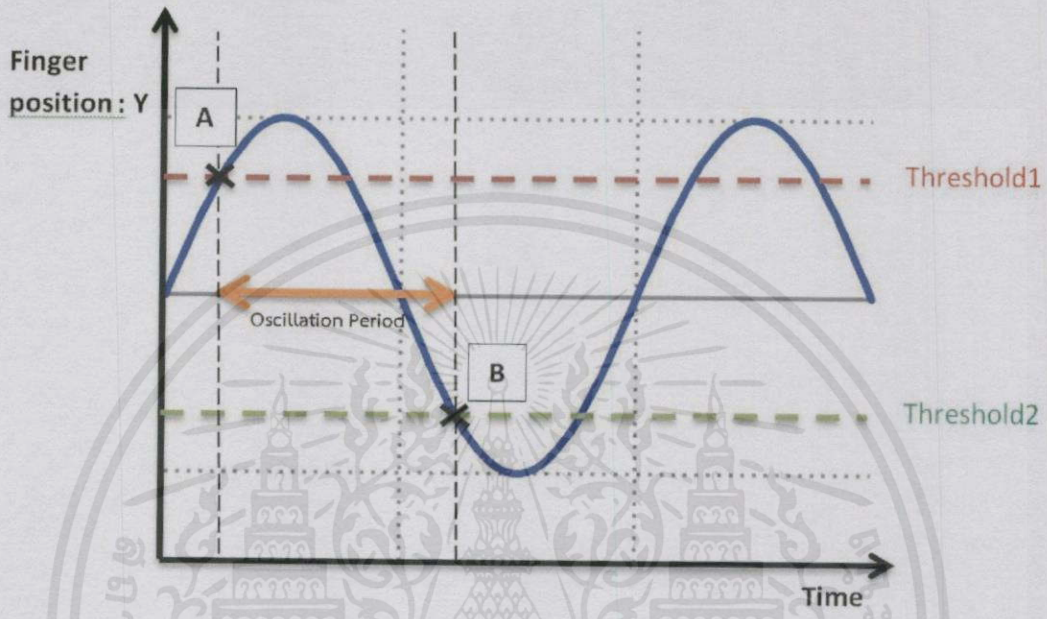
รูปที่ 3.48 Block diagramของ IIR High pass filter ที่ใช้สำหรับตรวจจับการใช้งาน Vibrato



รูปที่ 3.49 กราฟแสดง Magnitude response ของ high pass filter

2) การตรวจจับการเคลื่อนที่แบบกวัดแกว่ง (Oscillatory motion detection)

เมื่อมีการขยี้นิ้วแบบกวัดแกว่งบนแถบสไลเดอร์ ค่าตำแหน่งนิ้วที่ได้จะมีการกวัดแกว่งตามในลักษณะดังรูปที่ 3.50



รูปที่ 3.50 Waveform ของตำแหน่งนิ้วเมื่อมีการขยี้เพื่อใช้งาน Vibrato

หลักการที่ใช้สำหรับการตรวจจับการเคลื่อนที่แบบกวัดแกว่งของนิ้วนั้น จะใช้เทคนิคการตรวจจับการตัดผ่านจุดอ้างอิงศูนย์ (Zero-crossing) ซึ่งจะอธิบายด้วยรูปที่ 3.50 โดยมีหัวข้อที่สำคัญดังนี้

- Threshold :

  หลังจากค่าตำแหน่งนิ้วที่ถูกกรองด้วย High pass filter เคลื่อนที่ผ่านจุดศูนย์กลางของพื้นที่ด้านล่างที่ประกอบด้วยอิเล็กโทรด 6 ชิ้นแล้ว การกวัดแกว่งจะเกิดขึ้นเมื่อนิ้วเคลื่อนที่ต่อไปในทิศทางเดิมและผ่านตำแหน่งขีดเริ่ม (Threshold) และเมื่อเคลื่อนที่จนถึงตำแหน่งสูงสุดแล้ว นิ้วมีการเคลื่อนที่กลับผ่านตำแหน่งศูนย์กลางอีกขั้นและผ่านตำแหน่งขีดเริ่มในทิศทางตรงกันข้ามกับตอนแรก จึงนับว่ามีการกวัดแกว่งเกิดขึ้น

- ความถี่ของนิ้วที่ต่ำที่สุด (Minimum detectable frequency) :

  หลังจากตรงจับการกวัดแกว่งได้แล้ว จะทำการบันทึกเวลาของนิ้วที่ใช้สำหรับการเคลื่อนผ่านจุดขีดเริ่มหนึ่งไปยังอีกจุดหนึ่งเพื่อนำไปคำนวณหาความถี่ของการกวัดแกว่ง หากมีค่าเกินค่า Time-out ที่ได้ตั้งไว้ ก็จะพิจารณาว่าไม่มีการกวัดแกว่งและออกจากฟังก์ชันนี้ ความถี่ต่ำสุดที่ใช้สำหรับการตั้งค่า Time-out จะอยู่ประมาณ 2 Hz (Time-out มีค่าเท่ากับ 500ms)

บทที่ 4
ผลการทดสอบคุณสมบัติและความสามารถในการใช้งาน

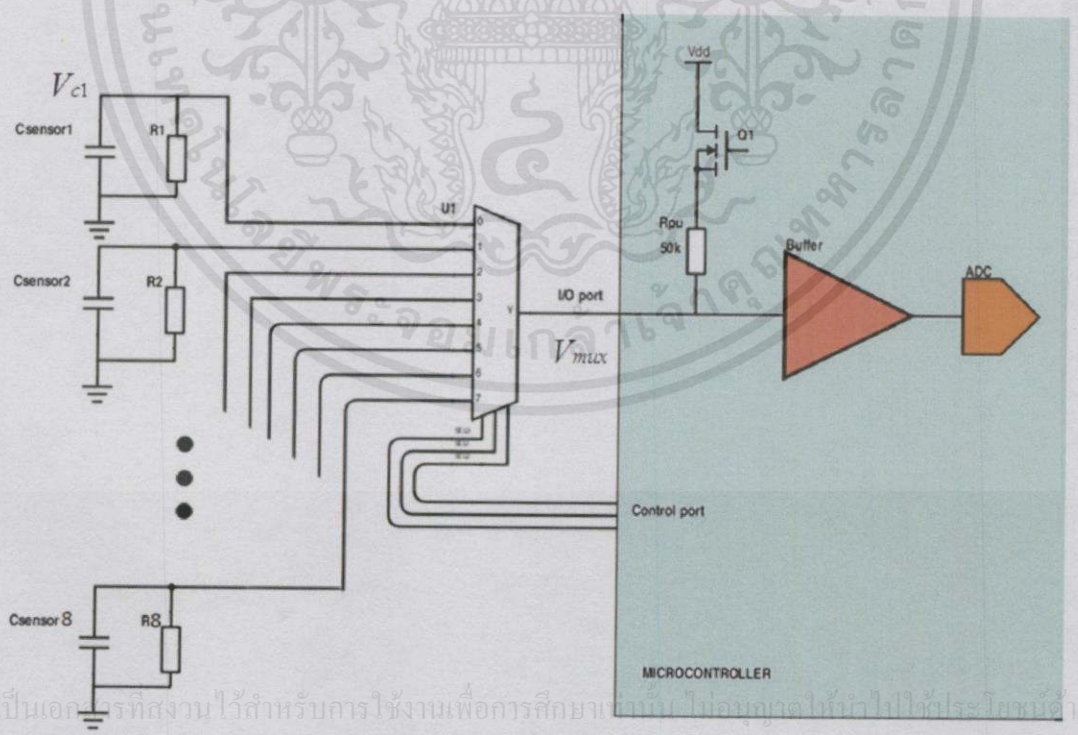
เนื้อหาภายในบทนี้ จะกล่าวถึงการวัดคุณสมบัติของอุปกรณ์ต่างๆของระบบจากผลของการออกแบบในบทที่ 3 และการทดสอบการใช้งานของระบบควบคุมฟังก์ชัน Pitch bend และ Vibrato สำหรับเครื่องดนตรีประเภท Keyboard synthesizer โดยใช้เซนเซอร์ตรวจจับการสัมผัส โดยมีรายละเอียดดังต่อไปนี้

4.1 ผลการทดลองคุณสมบัติต่างๆของอุปกรณ์จากการออกแบบ

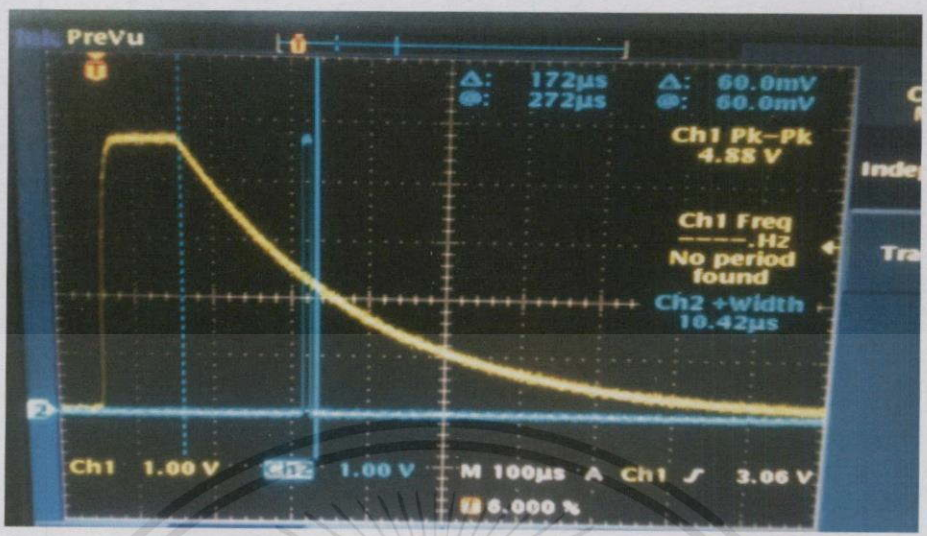
4.1.1 Keyboard touch sensor

1) จากการทดลองในหัวข้อที่ 3.1.3 ความเร็วสูงสุดของการรูดนิ้วบนแถบสไลเดอร์สำหรับการใช้งาน Pitch bend นั้นอยู่ที่ประมาณ 80 cm/s และความถี่การขยี้ของนิ้วบนแถบสไลเดอร์สำหรับการใช้งาน Vibrato นั้นอยู่ที่ประมาณ 5 Hz

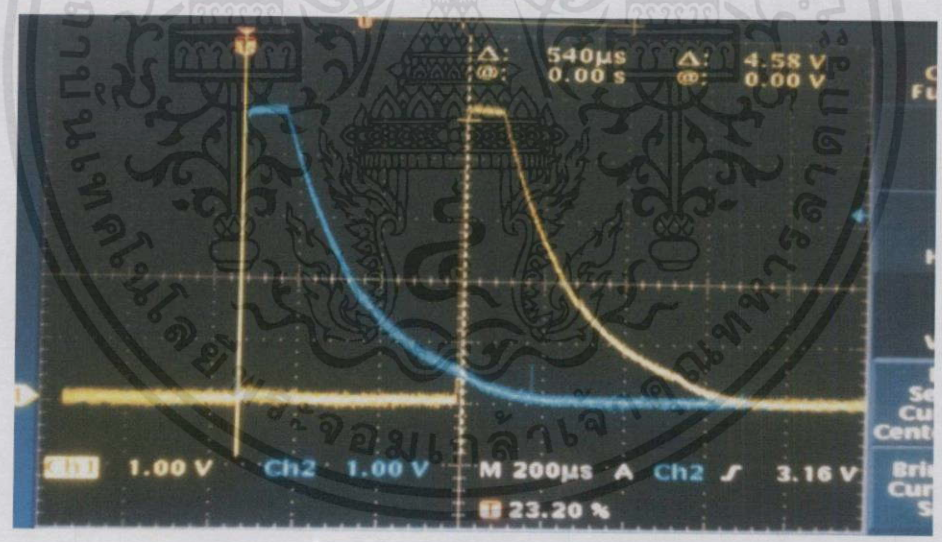
2) ผลการทำงานของวงจรตรวจจับการเปลี่ยนแปลงความจุไฟฟ้าจากหัวข้อที่ 3.1.2 นั้น จะอธิบายด้วยผลการวัดสัญญาณ ณ จุดต่างๆของวงจร โดยมีรายละเอียดดังต่อไปนี้



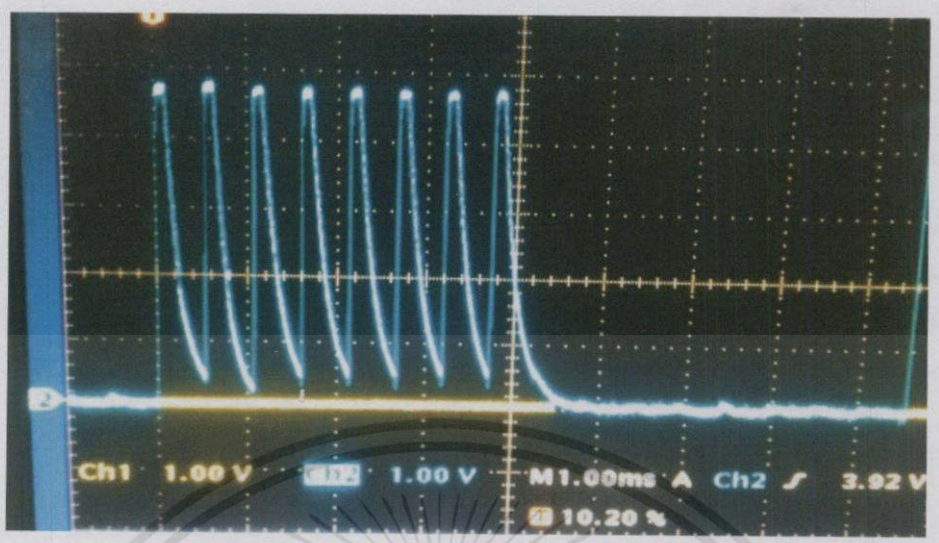
รูปที่ 4.1 วงจรตรวจจับการเปลี่ยนแปลงความจุไฟฟ้าจากหัวข้อที่ 3.1.2



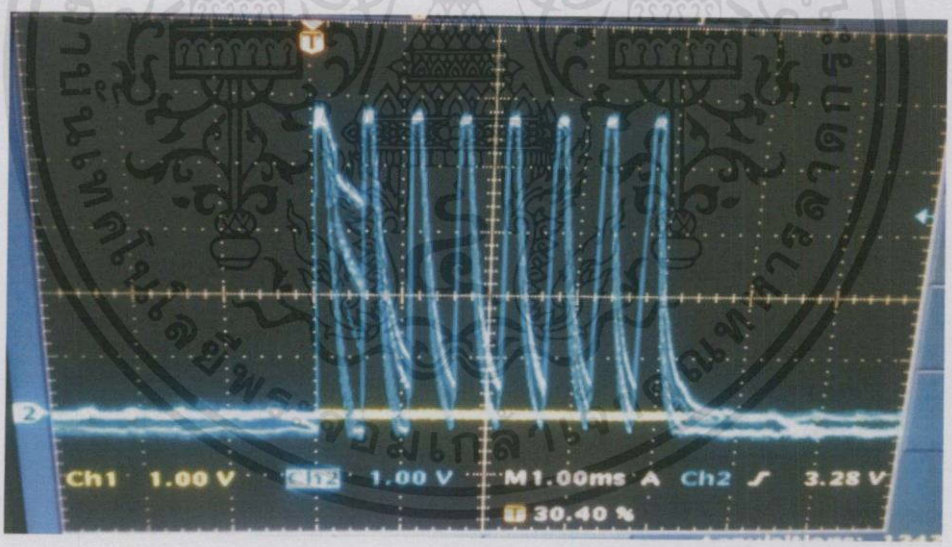
รูปที่ 4.2 (สีเหลือง) กราฟสัญญาณแรงดันตกคร่อมวงจร RC (V_c) ขณะคายประจุ, (สีฟ้า) กราฟสัญญาณแสดงจุดที่วงจร A/D converter เริ่มและสิ้นสุดการสุ่มค่าแรงดัน ณ ตำแหน่ง RC time constant



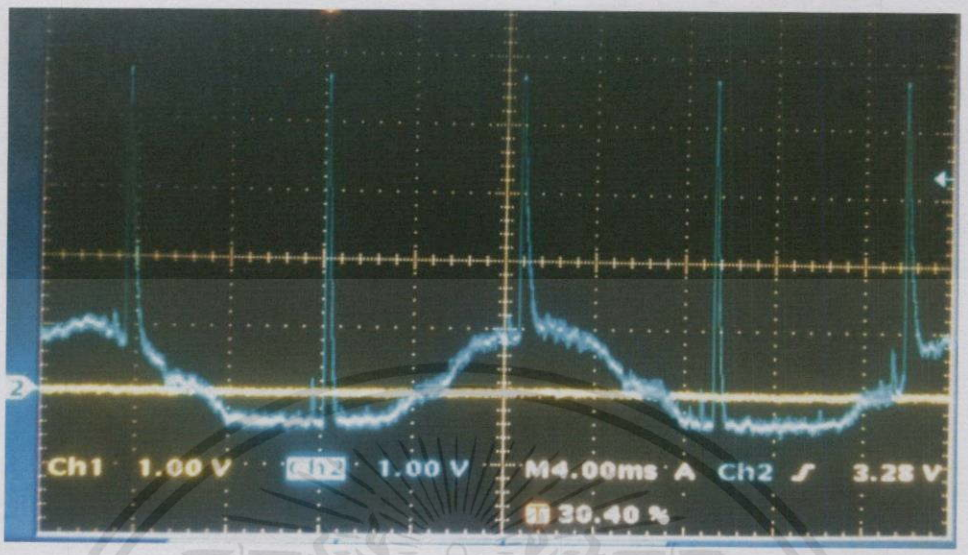
รูปที่ 4.3 ภาพแสดงความต่อเนื่องของการสับเปลี่ยนการของวงจร RC ด้วยมัลติเพล็กเซอร์, (สีฟ้า) กราฟสัญญาณแรงดันตกคร่อมวงจร RC ชุดที่ 1 (V_{c1}), (สีเหลือง) กราฟสัญญาณแรงดันตกคร่อมวงจร RC ชุดที่ 2 (V_{c2}), ระยะห่างของเวลาการทำงานนั้นมีค่าเท่ากับ 540 us



รูปที่ 4.4 กราฟสัญญาณแรงดันเอาต์พุตของมัลติเพล็กเซอร์ (V_{mux}) ขณะที่ทำการสับเปลี่ยนการทำงานของวงจร RC ด้วยกัน 8 ชุด โดยที่ไม่มีการสัมผัสของนิ้วบนอิเล็กโทรดใดๆ

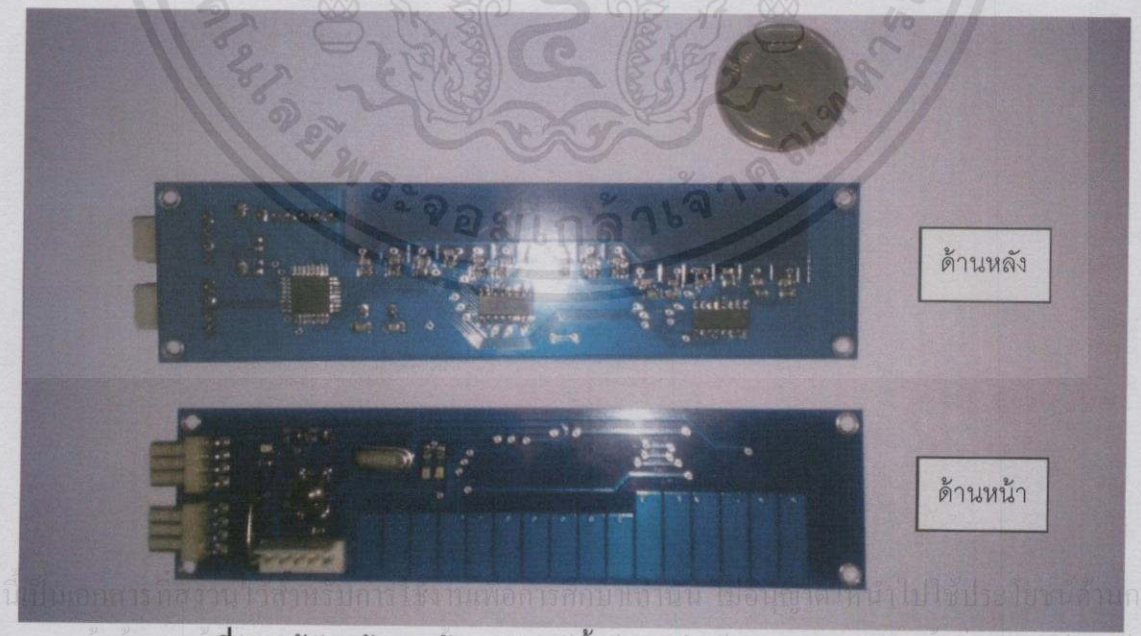


รูปที่ 4.5 กราฟสัญญาณแรงดันเอาต์พุตของมัลติเพล็กเซอร์ (V_{mux}) ขณะที่ทำการสับเปลี่ยนการทำงานของวงจร RC ด้วยกัน 8 ชุด โดยที่มีการสัมผัสของนิ้วบนอิเล็กโทรดและเกิดสัญญาณรบกวน 50 Hz ขึ้น

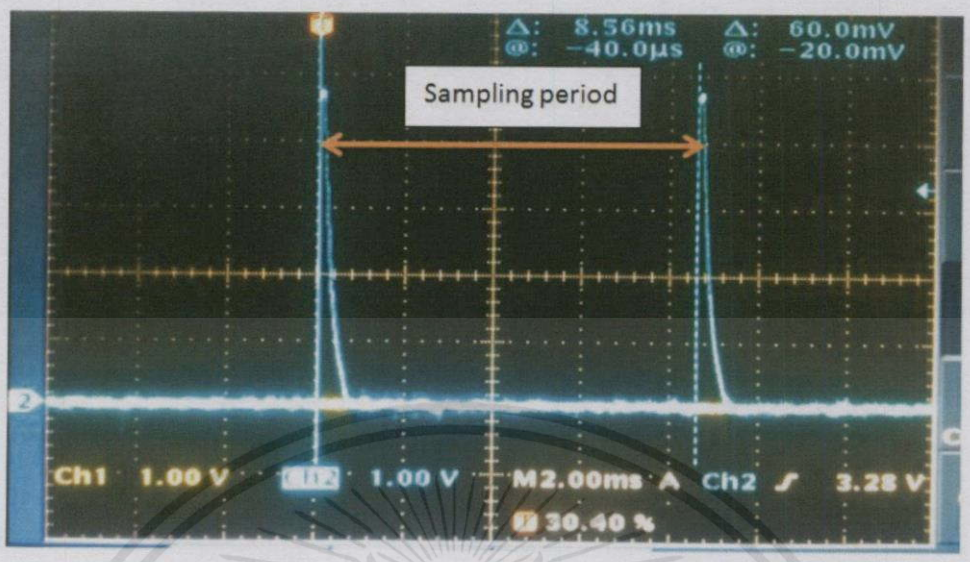


รูปที่ 4.6 แสดงผลของสัญญาณรบกวน 50 Hz ที่เกิดขึ้นจากการสัมผัสของนิ้วบนเซ็นเซอร์ ทำให้กราฟสัญญาณแรงดันของวงจร RC (V_c) มีรูปร่างที่เปลี่ยนไป

3) ชิ้นงาน Keyboard touch sensor ของการออกแบบครั้งสุดท้ายที่ใช้งานจริงนั้นมีลักษณะดังรูปที่ 4.7 โดยมีขนาดความกว้างยาวของแถบสไลเดอร์อยู่ที่ 8 cm x 1.2 cm และเลือกใช้จำนวนอิเล็กโทรดในแถบสไลเดอร์เท่ากับ 16 ชิ้น ความละเอียด (Resolution) ของเซ็นเซอร์มีค่าเท่ากับ 32 ระดับ และอัตราการสุ่มตำแหน่งนิ้วของเซ็นเซอร์จะอยู่ที่ประมาณ 120 Hz

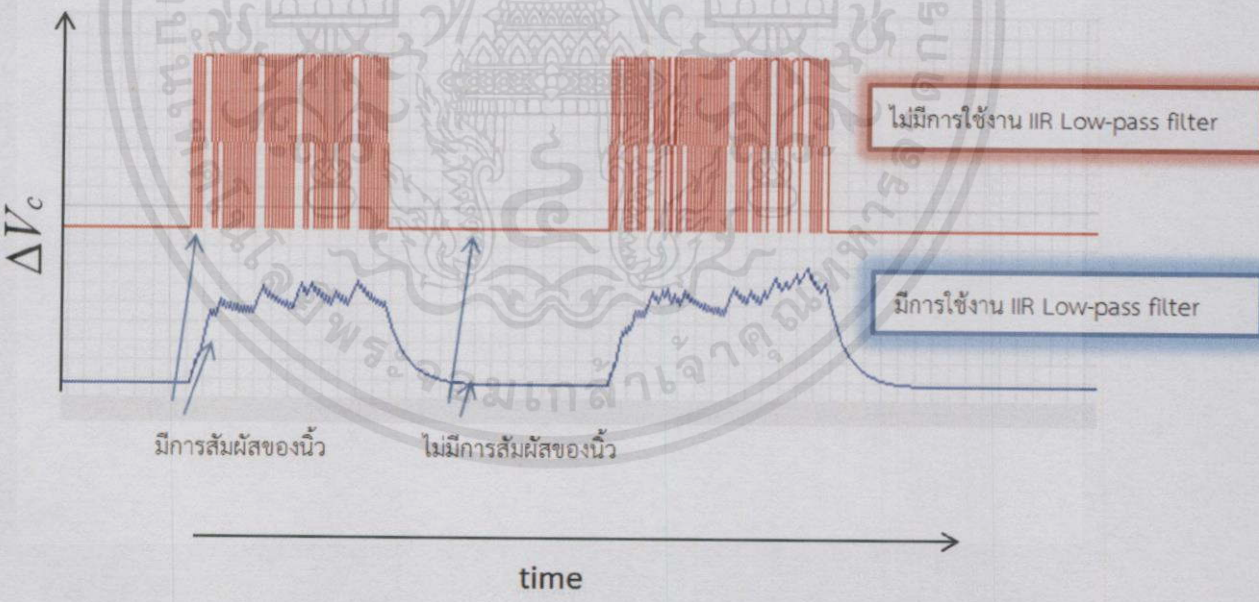


รูปที่ 4.7 ด้านหน้าและด้านหลังของชิ้นงาน Keyboard touch sensor



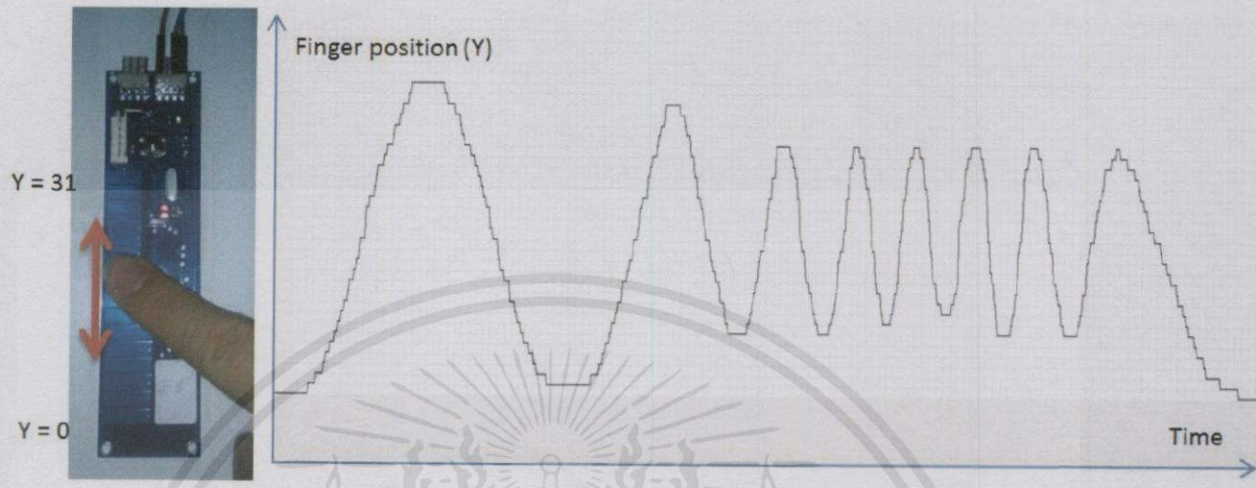
รูปที่ 4.8 กราฟสัญญาณแรงดันตกคร่อมวงจร RC (V_c) ชุดใดๆบนเซ็นเซอร์ ซึ่งจะมีคาบการทำงานเท่ากับคาบการสุ่มตำแหน่งนิ้วของเซ็นเซอร์ มีค่าเท่ากับ 8.5ms

4) ผลของการใช้ตัวกรอง IIR low-pass filter กำจัดสัญญาณรบกวน 50 Hz



รูปที่ 4.9 ภาพแสดงผลของการใช้ตัวกรอง IIR low-pass filter ที่มีต่อกราฟข้อมูลของค่า ΔV_c ที่ได้จากการทำงานของวงจรตรวจจับการสัมผัส ซึ่งสามารถลดการเปลี่ยนแปลงอย่างรวดเร็วของสัญญาณที่เกิดของสัญญาณรบกวน 50 Hz ได้

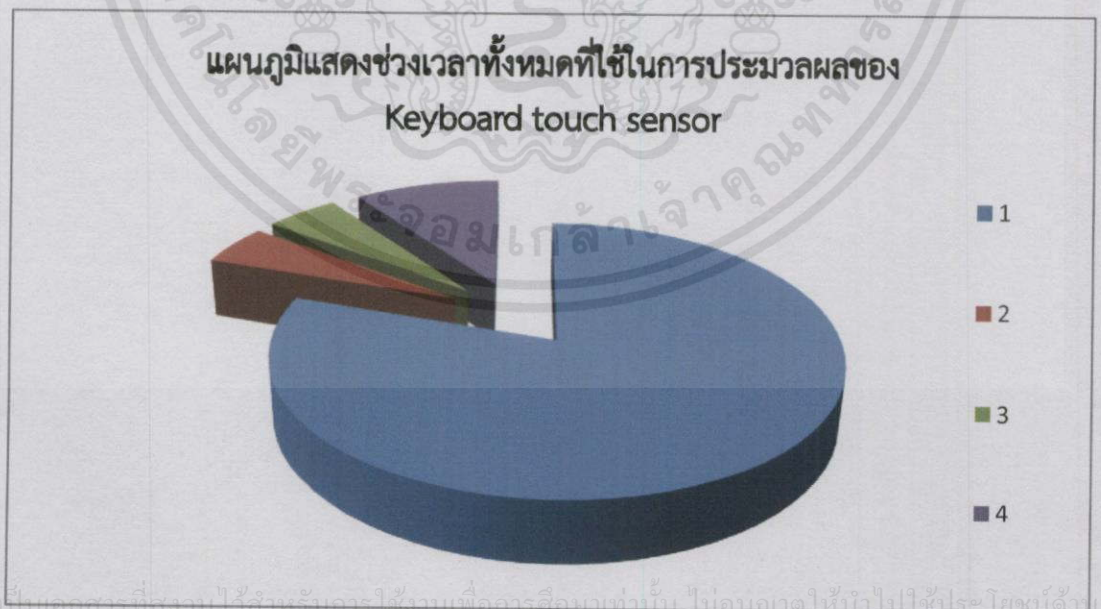
5) ผลการทดสอบการเข้ารหัสการระบุตำแหน่งของนิ้วด้วยความละเอียด 5-bit ของ Keyboard touch sensor



รูปที่ 4.10 แสดงการสร้างฟังก์ชั่นการระบุตำแหน่งของนิ้วแบบขั้นบันไดขณะที่มีการรูดนิ้วขึ้น-ลงบน Keyboard touch sensor

6) คุณสมบัติทางไฟฟ้าของ Keyboard touch sensor (Electrical characteristic) กระแสที่ใช้งานคือ 20 mA แรงดันการทำงานอยู่ที่ 5V

7) ช่วงเวลาทั้งหมดที่ใช้ในการประมวลผลของ Keyboard touch sensor แสดงเป็นแผนภูมิในรูปที่ 4.11



รูปที่ 4.11 แผนภูมิแสดงช่วงเวลาทั้งหมดที่ใช้ในการประมวลผลของ Keyboard touch sensor

1 (สีน้ำเงิน) คือ ช่วงเวลาที่ใช้ในการวัดค่าจาก 16 อิเล็กโทรดเพื่อตรวจจับการสัมผัส และช่วงเวลาสำหรับการทำงานของ IIR low pass filter คิดเป็น 80 เปอร์เซ็นต์

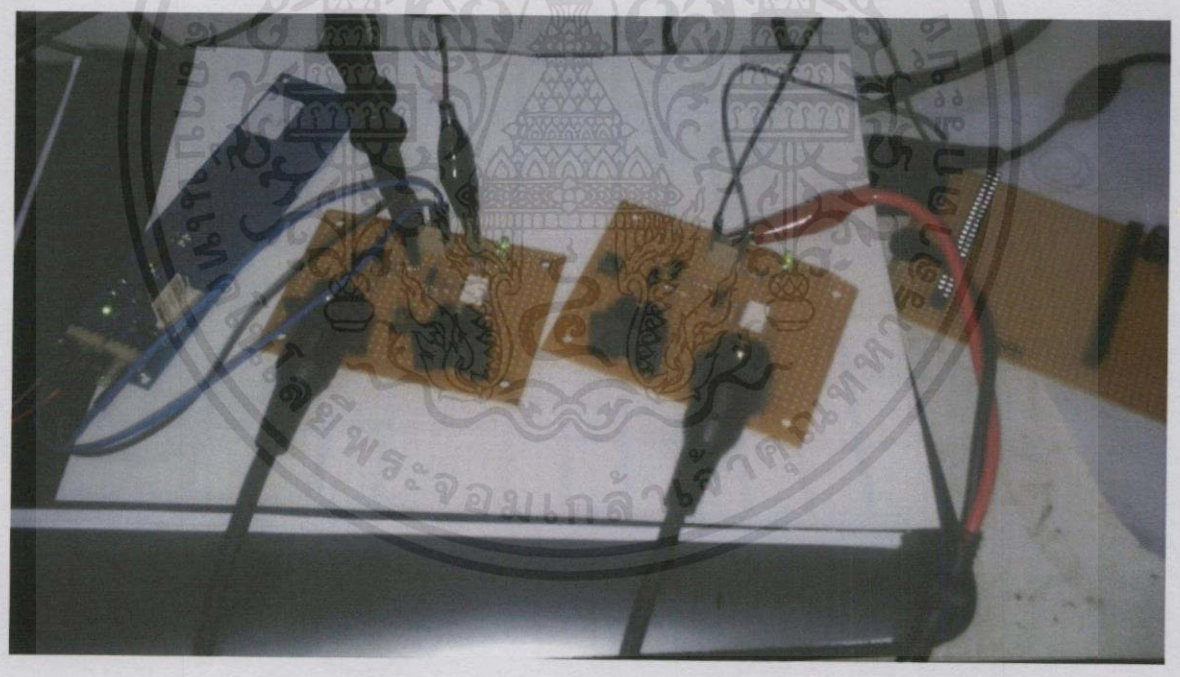
2 (สีแดง) คือ ช่วงเวลาที่ใช้ในการเข้ารหัส Thermometer code 32 bit สำหรับการระบุตำแหน่งนิ้ว คิดเป็น 5 เปอร์เซ็นต์

3 (สีเขียว) คือ ช่วงเวลาที่ใช้ในการถอดรหัส Thermometer code 32 bit ให้เป็น 5 bit binary code เพื่อใช้ระบุตำแหน่งนิ้วแบบชั้นบันได คิดเป็น 5 เปอร์เซ็นต์

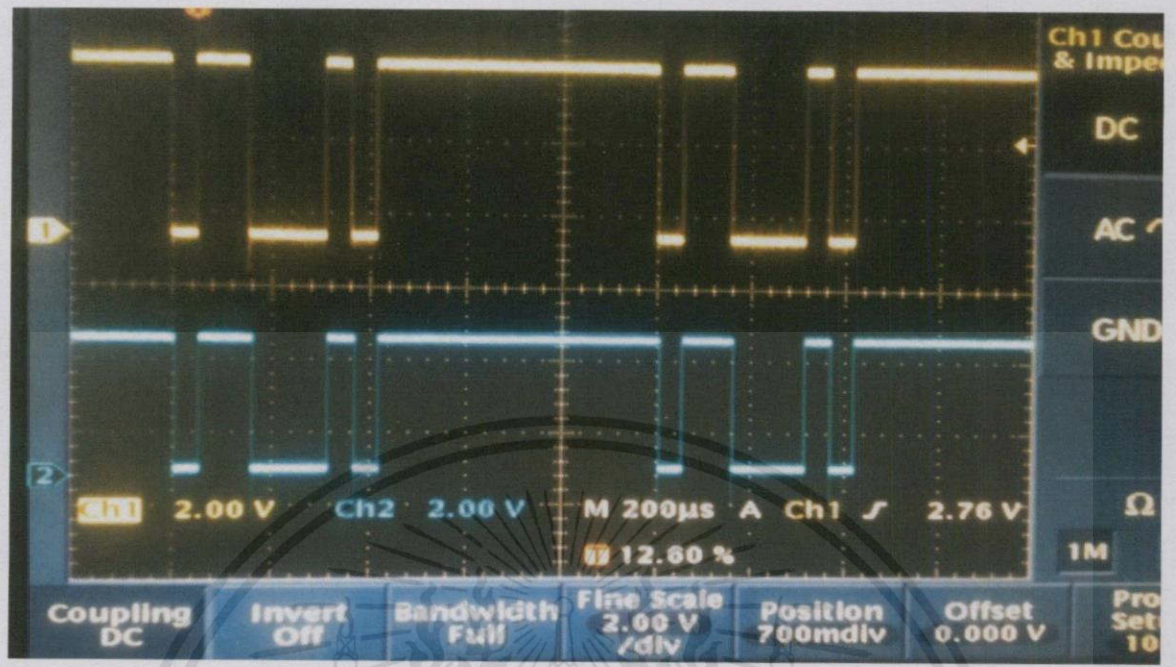
4 (สีม่วง) คือ ช่วงเวลาที่ใช้ในการสื่อสารข้อมูลผ่านบัส I2C คิดเป็น 10 เปอร์เซ็นต์

4.1.2 วงจรแปลงสัญญาณจาก TTL level เป็นสัญญาณ MIDI (MIDI data signal generator)

การทดลองทำโดยการส่งชุดข้อมูลMIDIที่อยู่ในรูปแบบสัญญาณ TTL level ไปยังวงจร TTL/MIDI converter เพื่อแปลงให้สัญญาณอยู่ในรูปแบบ Current loop จากนั้นนำสัญญาณที่ได้ส่งไปยังวงจร MIDI/TTL converter เพื่อทำการแปลงสัญญาณให้กลับไปอยู่ในรูปแบบสัญญาณ TTL level เพื่อเป็นการตรวจสอบวงจร TTL/MIDI converter ว่าสามารถแปลงข้อมูลได้อย่างถูกต้อง



รูปที่ 4.12 การทดลองความถูกต้องของวงจร TTL/MIDI converter



รูปที่ 4.13 ผลที่ได้จากการทดลองวงจร TTL/MIDI converter

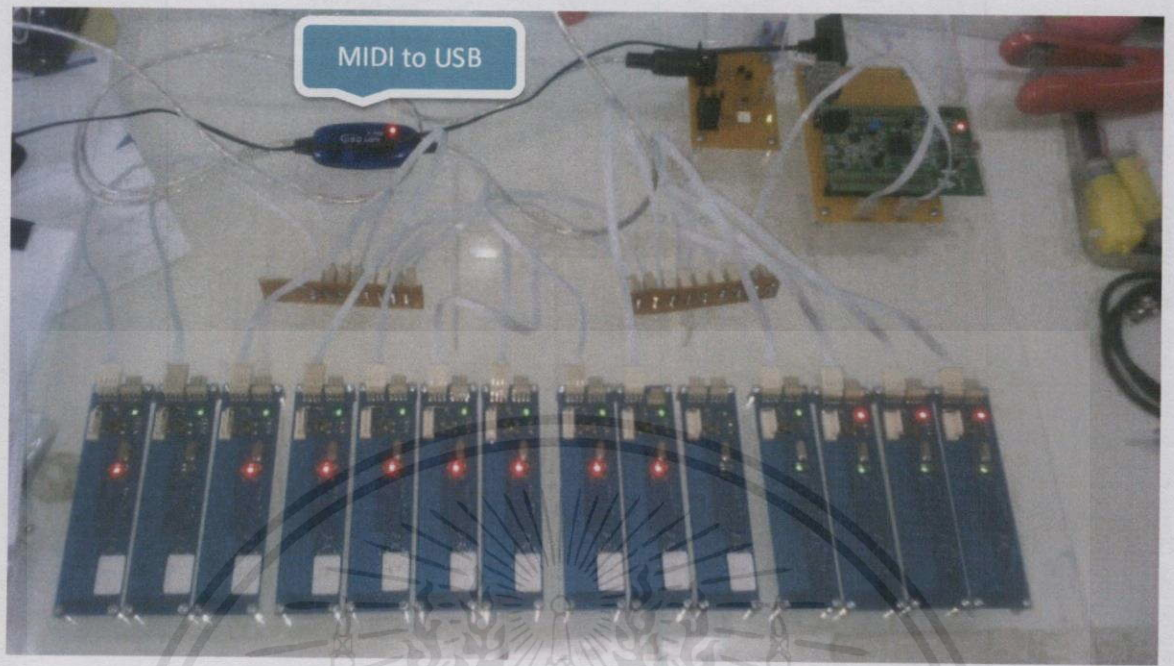
เส้นกราฟสีเหลืองคือข้อมูลMIDIที่อยู่ในรูปแบบของสัญญาณ TTL level อินพุท

เส้นกราฟสีฟ้าคือข้อมูลMIDIที่อยู่ในรูปแบบของสัญญาณ TTL level หลังจากผ่านวงจร TTL/MIDI converterและวงจร MIDI/TTL converter แล้ว

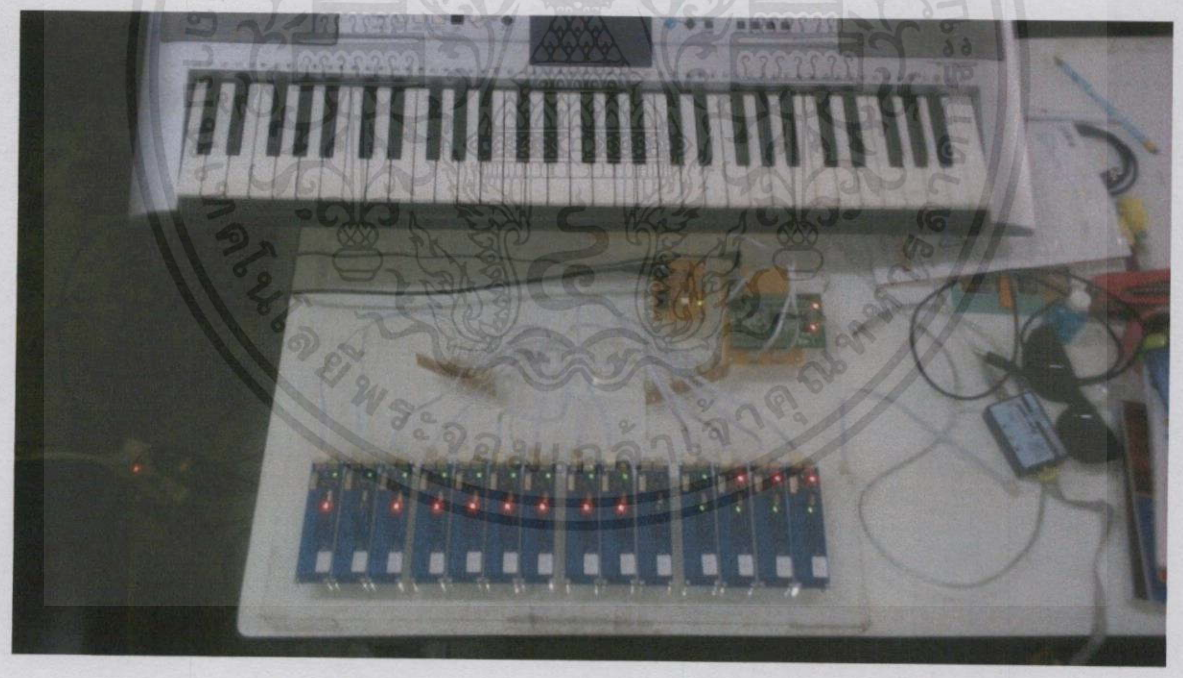
จากการทดลองพบว่าวงจร TTL/MIDI converter นั้นสามารถแปลงสัญญาณ TTL level เป็นสัญญาณ MIDI ได้อย่างถูกต้อง ไม่เกิดความผิดเพี้ยนขึ้น

4.2 ผลการทดสอบการทำงานของPitch bendและVibrato

ทำการทดลองโดยนำระบบ Keyboard touch sensor มาทดลองใช้งานฟังก์ชั่น Pitch bend และ Vibrato โดยได้ทำการทดลองให้ระบบเชื่อมต่อกับคอมพิวเตอร์ผ่าน MIDI to USB เพื่อใช้คอมพิวเตอร์ในการกำเนิดเสียงและเก็บค่าที่ได้จากการทดลอง ดังรูปที่ 4.14 และนำระบบเชื่อมต่อกับ keyboard synthesizer ที่มีอุปกรณ์กำเนิดเสียงในตัวเพื่อเป็นการทดสอบการเชื่อมต่อกับอุปกรณ์อื่นผ่านการสื่อสารแบบMIDI ดังรูปที่ 4.15



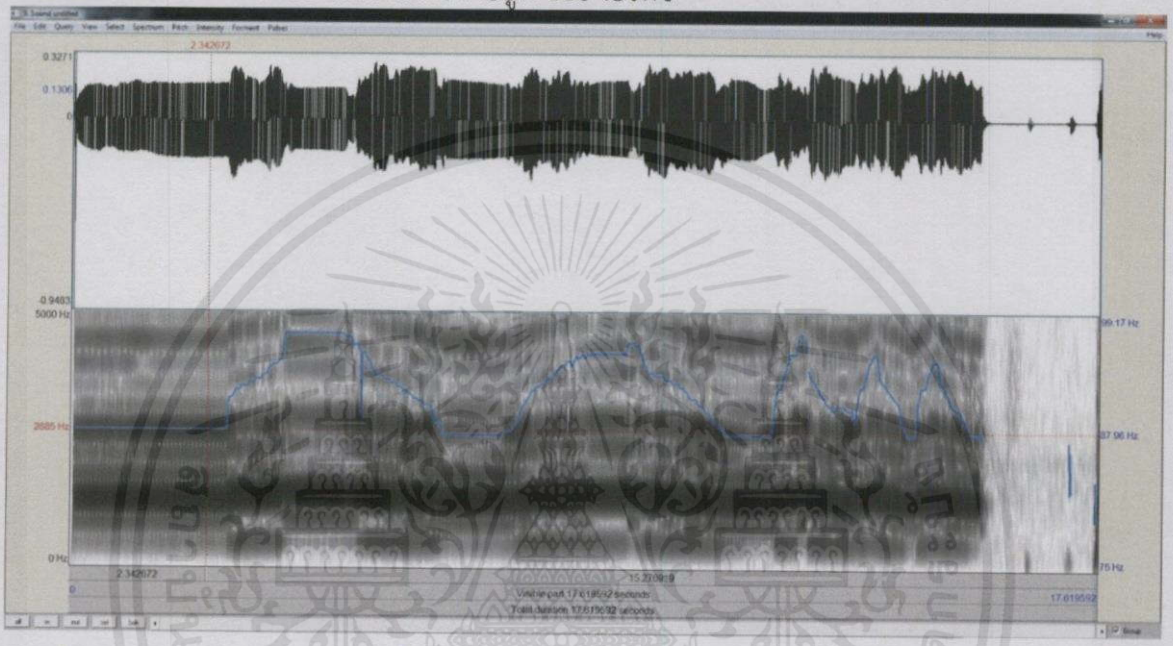
รูปที่ 4.14 ระบบเชื่อมต่อกับคอมพิวเตอร์ผ่าน MIDI to USB



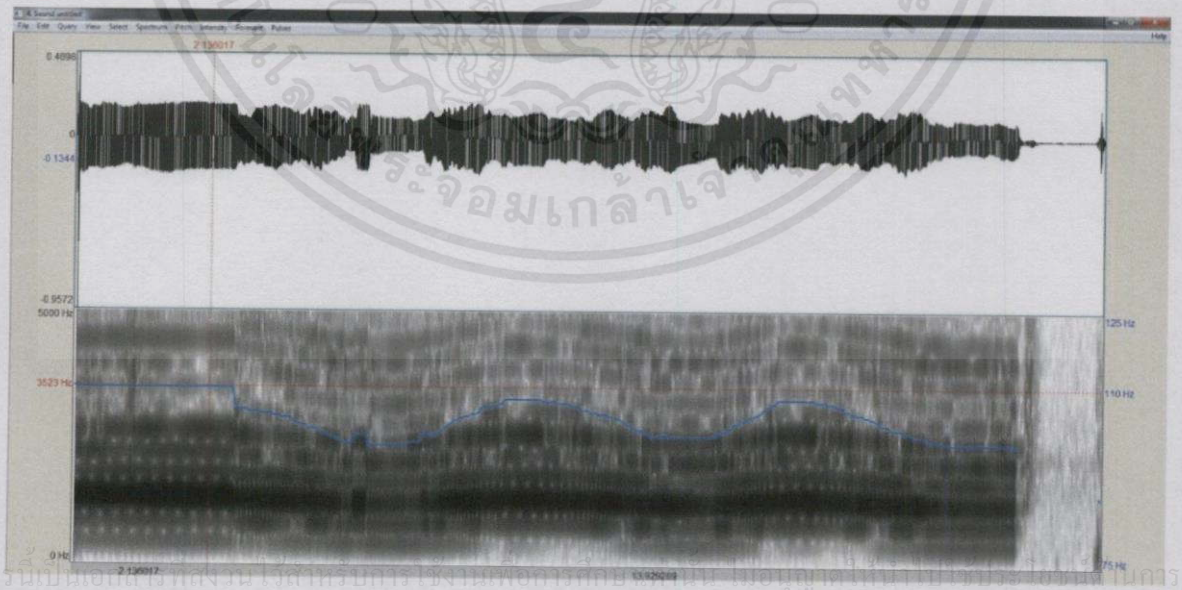
รูปที่ 4.15 ระบบเชื่อมต่อกับ keyboard synthesizer ที่มีอุปกรณ์กำเนิดเสียงในตัว

4.2.1 การทดสอบฟังก์ชั่น Pitch bend

ทดสอบโดยนำระบบเชื่อมต่อกับคอมพิวเตอร์ผ่าน MIDI to USB เพื่อใช้คอมพิวเตอร์ในการกำเนิดเสียง จากนั้นใช้โปรแกรมคอมพิวเตอร์ที่มีความสามารถในการพล็อตเสปคโตรแกรม (Spectrogram) ของเสียงโน้ตได้ เพื่อดูคุณสมบัติของฟังก์ชั่น Pitch bend ที่สร้างจากระบบ keyboard touch sensor โดยทดสอบด้วยตัวโน้ต ลา ณ ตำแหน่งออคเตป (Octave) ที่ 4 (A4) ของมาตรฐานเปียโน ดังรูปที่ 4.16 ความถี่ของตัวโน้ตดังกล่าวจะมีความถี่กลางอยู่ที่ 110 เฮิร์ทซ์



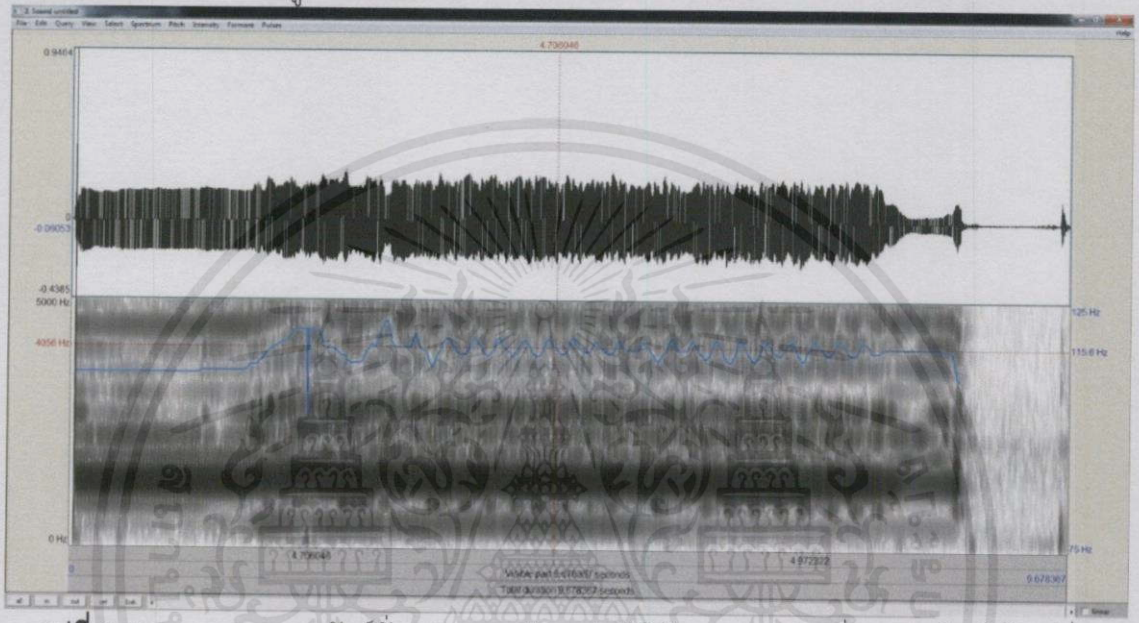
รูปที่ 4.16 ผลการทดสอบฟังก์ชั่น Pitch bend ตำแหน่งตัวโน้ต ลา ออคเตปที่ 4 (A4) แบบเพิ่มความถี่



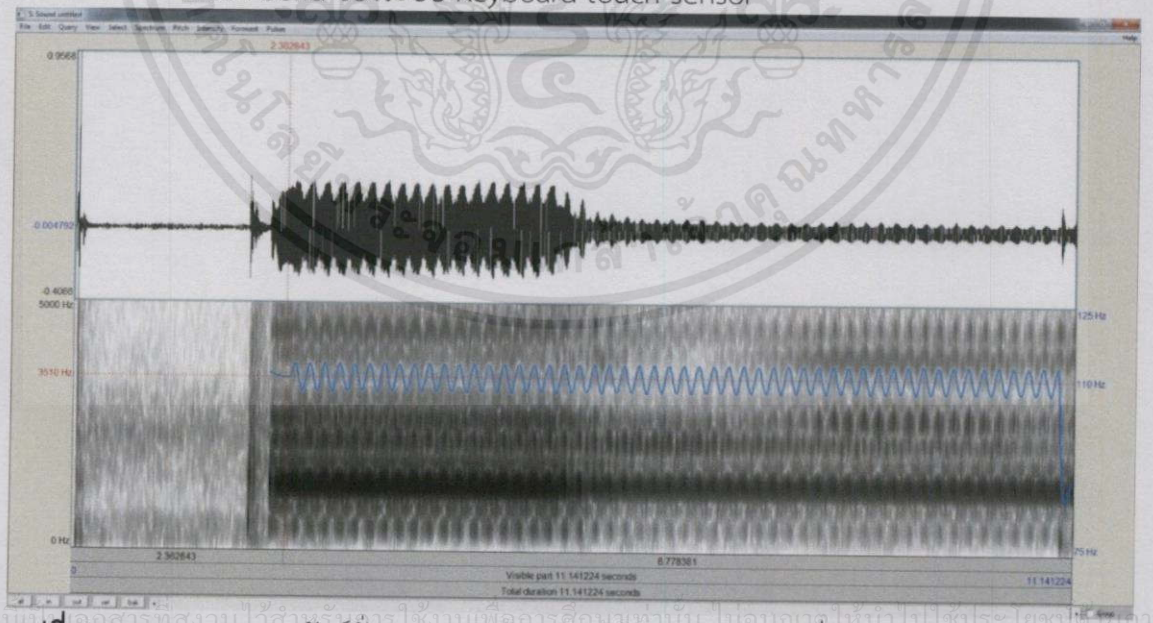
รูปที่ 4.17 ผลการทดสอบฟังก์ชั่น Pitch bend ตำแหน่งตัวโน้ต ลา ออคเตปที่ 4 (A4) แบบลดความถี่

4.2.2 การทดสอบฟังก์ชั่น Vibrato

ทดสอบโดยนำระบบเชื่อมต่อกับคอมพิวเตอร์ผ่าน MIDI to USB เพื่อใช้คอมพิวเตอร์ในการกำเนิดเสียง จากนั้นใช้โปรแกรมคอมพิวเตอร์ที่มีความสามารถในการพล็อตเสปคโตรแกรม (Spectrogram) ของเสียงโน้ตได้ เพื่อดูคุณสมบัติของฟังก์ชั่นVibratoที่สร้างจากระบบ Keyboard touch sensor โดยทดสอบด้วยตัวโน้ต ลา ณ ตำแหน่งออคเตป (Octave) ที่ 4 (A4) ของมาตรฐานเปียโน ความถี่ของตัวโน้ตดังกล่าวจะมีความถี่กลางอยู่ที่ 110 เฮิร์ทซ์



รูปที่ 4.18 ผลการทดสอบฟังก์ชั่น Vibrato ตำแหน่งตัวโน้ต ลา ออคเตปที่ 4 (A4) โดยใช้ฟังก์ชั่น Pitch bend ของระบบ Keyboard touch sensor



รูปที่ 4.19 ผลการทดสอบฟังก์ชั่น Vibrato ตำแหน่งตัวโน้ต ลา ออคเตปที่ 4 (A4) โดยการใช้ชุดคำสั่ง Modulation wheel ของ MIDI สร้างฟังก์ชั่น Vibrato

สังเกตว่าการใช้ชุดคำสั่ง Modulation wheel ของ MIDI สร้างฟังก์ชั่น Vibrato นั้นมีการเปลี่ยนแปลงของความถี่ที่ค่อนข้างเสถียรและเป็นคาบมากกว่าการใช้ฟังก์ชั่น Pitch bend ของระบบ Keyboard touch sensor สร้างฟังก์ชั่น Vibrato ดังนั้นการใช้ชุดคำสั่ง Modulation wheel ของ MIDI สร้างฟังก์ชั่น Vibrato

บทที่ 5
สรุปผลของโครงงาน

5.1 บทสรุป

โครงงานฉบับนี้นำเสนอระบบควบคุมลักษณะทางเสียงของ Pitch Bend และ Vibrato ของเครื่องดนตรีอิเล็กทรอนิกส์ประเภทคีย์บอร์ดซินธีไซเซอร์ (Keyboard Synthesizer) โดยใช้พร็อกซิมิตี้เซ็นเซอร์ชนิดตรวจจับการสัมผัสด้วยการวัดความจุไฟฟ้า ซึ่งทำหน้าที่แทนการใช้ล้อคันโยกหรือจอยสติ๊กดั่งเดิมบนซินธีไซเซอร์ทั่วไป ข้อจำกัดของการควบคุมแบบดั่งเดิมคือ นักดนตรีจำเป็นจะต้องใช้มือหนึ่งข้างซึ่งโดยทั่วไปจะใช้มือซ้ายในการควบคุม ทำไม่สามารถใช้ฟังก์ชั่น Pitch bend และ Vibrato ด้วยล้อคันโยกได้หากต้องการเล่น Keyboard synthesizer ด้วยสองมือพร้อมกันได้ แต่ระบบใหม่นี้จะควบคุมฟังก์ชั่น Pitch bend และ Vibrato ด้วยการรูดปลายนิ้วมือบนลิ่มคีย์บอร์ด ซึ่งจะทำให้นักดนตรีสามารถใช้งาน Pitch bend และ Vibrato ขณะที่มือทั้งสองยังสามารถเล่นโน้ตอยู่ในเวลาเดียวกันได้และทำให้เกิดความหลากหลายของแนวคิดที่สร้างสรรค์ในการเล่นดนตรีมากขึ้น โดยเป้าหมายของโครงงานนี้จะต้องออกแบบให้เซ็นเซอร์ของระบบนั้นสามารถนำไปติดตั้งบนลิ่มคีย์ของคีย์บอร์ดซินธีไซเซอร์ได้

5.2 ปัญหาและแนวทางแก้ไข

ปัญหาที่พบคือเซ็นเซอร์ที่ใช้สำหรับตรวจจับการสัมผัสนั้นมีขนาดใหญ่กว่าขนาดมาตรฐานของลิ่มคีย์บอร์ดเพราะข้อจำกัดขนาด ดังนั้นหากต้องการออกแบบให้เซ็นเซอร์สามารถติดตั้งบนลิ่มคีย์ได้จะต้องใช้แผ่น PCB มากกว่า 2 ชั้น (เช่น 4-layer) ขึ้นไปในการออกแบบเพื่อให้สามารถลดความซับซ้อนของลวดลาย PCB ลงได้ และการเพิ่มความไวต่อการสัมผัสสามารถทำได้โดยการใช้ PCB ที่มีความบางมากกว่านี้

ปัญหาต่อมาคือระบบยังไม่มีประสิทธิภาพมากพอที่จะตอบสนองการเล่นด้วยความรวดเร็ว สามารถแก้ไขได้โดยการเปลี่ยนวิธีการวัดความจุไฟฟ้า หรือเปลี่ยนความเร็วของ microprocessor บนแผ่นเซ็นเซอร์ให้มีความเร็วมากขึ้น และเพิ่มความเร็วของบัส I2C จาก100kbit/s เป็น 400kbit/s สุดท้ายคือสามารถเพิ่มความเร็วได้โดยการใช้ RTOS (Real Time Operating - System) เป็นพื้นฐานของการพัฒนาโปรแกรมบนบอร์ด STM32F4 เพื่อเพิ่มประสิธิภาพการจัดการของหน่วยประมวลผล ทำให้ระบบสามารถตอบสนองต่อการกดที่รวดเร็วได้ทันเวลา

เอกสารอ้างอิง

[1] Andrew McPherson, Touchkeys: Capaitive Multi-Touch Sensing on a Physical Keyboard.

[2] Oscar Camacho and Eduardo Viramontes, Designing Touch Sensing Electrodes.

[3] Dr.Mark Wockert, ECE 2610 Signals and Systems. Lesson 8 IIR Filters

[4] The MIDI Manufactuurers Association, MIDI 1.0 Detailed Specification.

[5] Andrew P. McPherson and Adrian Gierakowski and AdamM.Stark, The space Between the Notes: Adding Expressive Pitch Control to the Piano Keyboard

[6] Tim Wescott, Sampling: What Nyquist Didn't Say, and What to Do About It

[7] Zack Albus, PCB-Based Capacative Touch Sensing With MSP430

[8] Vincent Chan, Steve Underwood, MSP430 Capacitive Single-Touch Sensor Design Guide

[9] Andrew McPherson and Youngmoo Kim, Design and applications of a multi-touch musical keyboard

[10] Pual Maddox, MIDI AND THE AVR, 2002

[11] www.elin.ttu.ee/mesel/Study/subjects/0070BME/Content/CirDesg/muxes.htm

[12] kennarar.vma.is/thor/v2011/vgr402/ldr.pdf

ภาคผนวก

The Space Between the Notes: Adding Expressive Pitch Control to the Piano Keyboard

Andrew P. McPherson
Queen Mary University of London
Mile End Road, London E1 4NS, United Kingdom
andrewm@eecs.qmul.ac.uk

Adrian Gierakowski
Queen Mary University of London
Mile End Road, London E1 4NS, United Kingdom
adriang@eecs.qmul.ac.uk

Adam M. Stark
Queen Mary University of London
Mile End Road, London E1 4NS, United Kingdom
adam.stark@eecs.qmul.ac.uk

ABSTRACT
This paper addresses the question of how to extend the capabilities of a well-established interface in a way that respects users' existing expertise. The piano-style keyboard is among the most widely used and versatile of digital musical interfaces. However, it lacks the ability to alter the pitch of a note after it has been played, a limitation which prevents the performer from executing common expressive techniques including vibrato and pitch bending. We present a system for controlling pitch from the keyboard surface using capacitive touch sensors to measure the locations of the player's fingers on the keys. The large community of trained pianists makes the keyboard a compelling target for augmentation, but it also poses a challenge: how can a musical interface be extended while making use of the existing techniques performers have spent thousands of hours learning? In this paper, user studies with conservatory pianists explore the constraints of traditional keyboard technique and evaluate the usability of the continuous pitch control system. The paper also discusses implications for the extension of other established interfaces in musical and non-musical contexts.

Author Keywords
Piano keyboard; musical interfaces; capacitive touch sensing; performance technique; expressivity; digital arts

ACM Classification Keywords
H.5.5 Sound and Music Computing: Systems, Modeling

General Terms
Design; Human Factors; Experimentation

INTRODUCTION
Electronic piano-style keyboards are ubiquitous in many styles of musical performance. Beyond emulating the acoustic piano, the keyboard is frequently used to control sampled string, wind or synthesiser sounds. However, the keyboard lacks an important feature found on nearly all string and wind instruments: once a note is played, the performer has no ability to further control its pitch. This limitation prevents the keyboardist from emulating common expressive techniques including vibrato, pitch bends and slides between notes. Some keyboards feature a "pitch wheel" to the side of the keys for fine pitch adjustment, but this arrangement has several drawbacks: it requires the dedicated use of one hand; it lacks the ability to independently control the pitch of multiple notes; and it requires two unrelated gestures to play, one to select the key, another to fine-tune its pitch.

We seek to integrate expressive pitch control into the keyboard itself. There are millions of trained keyboardists who have each spent thousands of hours to become proficient, and an interface that builds on this wealth of training has significantly greater potential for user uptake than an unfamiliar design. However, existing training is also the chief obstacle for any new keyboard interface. Ideally, an extended keyboard should require minimal re-learning by the performer, and in particular, the addition of new capabilities should not interfere with traditional technique. Broadly framed, then, this paper addresses the question of how a familiar interface with a substantial body of user expertise can be extended while respecting users' existing knowledge.

BACKGROUND

Previous Enhanced Keyboard Work
Designers and musicians have long been seeking ways to add new dimensions of expressive control to the keyboard. Yang and Essl [30] use a 3D depth camera to capture gestures a performer makes above the keyboard; Brent [4] tracks pianist arm motion using an IR camera mounted above the keyboard. Both systems are used to control sonic parameters of the performance. Pianist Sarah Nicolls [18] worked with composers to create performances using light sensors, accelerometers and bio-sensors in addition to data from the keyboard.

Over the past century, several designs have sought to integrate additional control into the keyboard mechanism itself. Aftertouch (key pressure sensing) is commonly available on commercial keyboards, though it is rarely used to control pitch. The Ondioline (invented 1941) and the experimental Yamaha GX-1 (1973) both let the performer add vibrato to notes by shifting the keyboard mechanism from side to side [21, 25]. In the 1980s, Robert Moog and composer John Eaton developed a keyboard measuring the position of the performer's

CHI 2013, April 27–May 2, 2013, Paris, France.

fingers on the key surfaces [17], allowing continuous control of pitch and other sonic parameters of each note. The authors, in previous work [15], also developed a keyboard capable of continuous finger position measurements, and this work forms the hardware basis of this paper. The commercial EVO keyboard (`http://endeavour.de`) senses finger position within a 1x4cm region of each key, with an adjusted geometry to provide a wider playing surface on the black keys.

Why have none of these advances been integrated into mainstream keyboard design? The reason may have as much to do with human factors as the technology itself. In a newspaper interview, composer Eaton said of the Moog keyboard, "It's very difficult to play. But an instrument should be difficult to play. That's the only way to master musical materials, by overcoming these difficulties" [19]. Though using the location of finger-key contact as a control dimension is conceptually straightforward, it presents substantial practical difficulties in performance. Of course, many traditional instruments are also difficult to play, but beyond a certain complexity, the designer will encounter limitations of human motor control and cognitive bandwidth [23]. Jordà defines "musical instrument efficiency" as the ratio between musical output complexity and control complexity [11]. If the constraints of keyboard technique are not accounted for, enhanced keyboards run the risk of greatly increased control complexity in exchange for only modest gains in musical output.

Constraints of Keyboard Technique

Piano technique rests on two mechanical assumptions: that the velocity of the initial press, rather than actions thereafter, determines the sound of a note; and that a key can be pressed anywhere along its surface with similar results. Where the fingers touch the keyboard is thus primarily determined by the physical constraints of playing multiple notes at once, either simultaneously or in sequence.

From the interface designer's perspective, finger location is a "soft" constraint in that the performer could in theory place the fingers nearly anywhere while playing a passage. In practice, though, few performers will spend the time needed to master an instrument that demands drastic changes to their technique. The question thus becomes how to find the "space between the notes": the aspects of performer-keyboard interaction which are *not* constrained by existing technique and which can therefore be repurposed for new musical effects.

Analysis of existing performance technique is important to precisely identify its constraints. Motion analysis of piano playing has a venerable history extending at least as far back as Ortmann (1929) [20]. Recent analyses include video motion capture of piano performance [7, 13], examinations of piano "touch" as profiles of continuous key motion [2, 16] and movement analysis of the arm using accelerometers [8].

Beyond Music: Challenges of Training and Expertise

Extending the keyboard raises a broader issue of expertise, both as an enabler and a constraint in interface design. Players of an extended instrument, even one with a high ceiling of expressivity, may experience a temporary dip in ability due to unfamiliarity. Scarr et al. [26], examining expertise in a broader HCI context, observe that this dip can deter users from adopting a higher-performance interface and propose methods of smoothing the transition from novice to expert.

Analogous situations be found in efforts to extend other familiar interfaces, including on-screen QWERTY keyboards [6] and pen-based input [10]. In the latter case, understanding typical pen motion profiles used in writing can guide the creation of more natural interfaces [29]. Touchscreen input is a frequent target for augmentation with new sensor modalities, including pressure [27], finger angle [28], device tilt [24] and shear (sideways) force [9]. Cognitive bandwidth constraints also appear in other domains, including interfaces used while driving [5]. In all cases, new input methods are added to a familiar activity, and non-interference with the original task becomes important.

When new sensors are added to an existing interface, questions may arise of when a user *intends* to engage with a new modality, versus when the sensor data merely reflects a byproduct of familiar actions. For example, when eye-tracking is used as an input device, separating intention from involuntary eye movement is a challenge [22]. Similarly, most screen touches will exert a measurable amount of pressure, but pressure will not always reflect a deliberate decision by the user. Requiring consistent regulation of finger pressure at all times would make a touchscreen substantially harder to use, and consequently a pressure-enhanced screen might make use of the new data only in selected situations. For other interactive systems, including those employing gesture recognition in free space [3], every motion by the user is necessarily an input, making it critical to separate intentional actions from non-meaningful movement [12].

Paper Overview

The remainder of this paper presents our sensor system extending the keyboard, followed by an initial investigation of the constraints of existing keyboard technique. Techniques are presented for adding expressive pitch control to the keyboard which are evaluated in user studies with expert pianists. The conclusion examines implications for both keyboard performance and interfaces outside the musical domain.

We recently developed a capacitive sensing system for measuring the location of the player's fingers on the key surfaces [15]. Thin printed circuit board overlays adhere to the surface of an existing keyboard. Figure 1 shows the sensors attached to a weighted-key electronic piano. The shapes of the sensors reflect measurements of several acoustic and electronic instruments; experimentally, we have found very little variation in key dimensions among instruments.

Figure 2 shows the principle of operation. On each black key, the capacitance values of 17 discrete pads are measured (25 on each white key). The presence of the player's finger increases the capacitance, and by interpolating between pads, a spatial resolution of 1024 or more points in the lengthwise axis is achieved. Both black and white keys measure in the lengthwise (Y) axis; the front of the white keys also measure horizontal (X) finger position with 256-point spatial resolution. Further detail on sensor operation can be found in [14].

Figure 1. Capacitive sensors installed on a Yamaha Clavinova CLP-150.

SENSOR SYSTEM

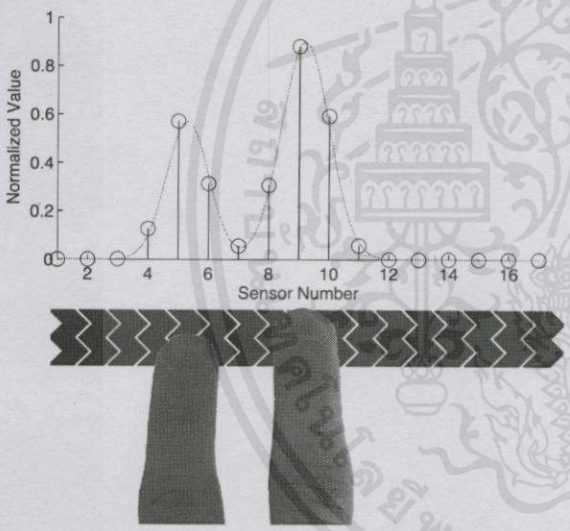


Figure 2. Simulation of touch location measurement by interpolating between discrete sensor pads.

Data Processing

Finger-key contact locations are scanned at 200 frames per second and transmitted to a computer by USB. Each incoming frame is compared to the previous frame for the same key to identify touches that have been added, removed, or changed in location. This data is combined with MIDI[1] from the underlying keyboard to determine finger location for each played note. In our analyses, we are primarily interested in the motion of the finger while a key is pressed, though the sensors can also determine finger contacts on unpressed keys.

Hardware Improvements

The sensor system used in this paper includes several improvements to the design in [15]. User feedback highlighted the square corners of the black keys as a source of discomfort; they are rounded in the current version (Figure 1). The white key sensors have been made 0.5mm narrower to reduce the likelihood of performers catching their fingers on the sides. The white and black colour of the current system preserves the familiar look of the keyboard. After considerable testing of various overlay materials, we found that the raw solder-mask coating of the printed circuit boards received the most positive response from performers.

The current sensor system also incorporates optical reflectance sensors into the back of each key (Figure 1 bottom). When a fixed reflective object (e.g. a piece of white plastic cut to the vertical contours of the keys) is placed above the back of the keyboard, these sensors provide a measurement of continuous key angle. Each optical sensor is sampled at 1000Hz. To handle this increased sensor bandwidth, the original design's 8-bit microcontroller is replaced with a design based on a 72MHz ARM Cortex-M3.[2] The current paper does not use the optical sensors, however, instead restricting its focus to MIDI and capacitive touch sensor data.

PRE-STUDY: FINGER MOTION IN PERFORMANCE

We conducted informal testing with several performers examining finger motion in traditional piano performance, including recording one author's [AG's] performance of 8 short pieces from J.S. Bach's *Notebook for Anna Magdalena Bach* (1722-25). These investigations identified two constraints:

Constraint 1: Raw finger location cannot be used as a dimension of expressive control. Finger-key contact location is highly dependent on the fingering used, which is often determined by the sequence of notes in a passage. As Figure 3 demonstrates, the different lengths of the fingers (particularly the shorter thumb and pinky) manifest in different touch locations. Furthermore, to reach the black keys, the hand must be positioned higher on the keyboard.

Clearly, these hand positions do not represent free expressive decisions, though a certain amount of variation is possible. Though raw finger position may be usable for parameters whose setting has only a secondary impact on the sound (e.g. pluck location on a string [15]), raw position cannot be used for expressive pitch control.

Constraint 2: Finger motion on the keys is common, and the system must separate intentional from unintentional actions. Though for most notes the finger remains in place on the key while it is pressed, the touch location can drift for several reasons, including rolling the finger from the pad to the tip and shifting the hand to prepare for the following notes. These factors are further investigated in a study with 8 professional-level pianists, described later in this paper ("Examining Traditional Technique"; Figure 11).

One straightforward approach to adding expressive control to the keyboard surface is to consider the *relative* position of a touch as it differs from the point of onset. This investigation highlights the need for, at minimum, a threshold which separates smaller movements that result from standard technique

[1]Musical Instrument Digital Interface; `http://www.midi.org`

[2]STMicroelectronics STM32F1; `http://www.st.com`

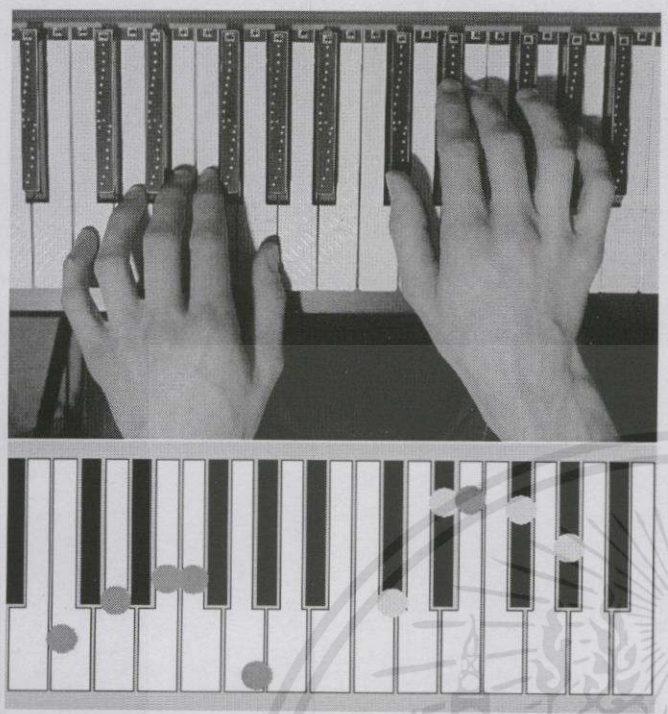


Figure 3. Relationship of hand position to touch data. Finger length and use of black keys strongly influence touch location.

from larger deliberate motions on the key surfaces [15]. In the next section we present techniques for continuous pitch control which further improve on the ability to separate deliberate expressive actions from byproducts of standard performance.

EXPRESSIVE PITCH CONTROL

We developed techniques for performing vibrato, pitch bends and slides between notes from the keyboard surface. Based on earlier feedback from performers, we chose gestures inspired by string instrument playing to control these techniques. While on violin family instruments, vibrato and pitch bends are performed on the same physical axis (lengthwise on the fingerboard), we chose to separate vibratos and pitch bends into orthogonal dimensions: vibrato on the horizontal (X) axis and pitch bends on the vertical (Y) axis (Figure 2). This was done to match the geometry of the keys (long and narrow) with the requirements of the gesture (large motion for pitch bends, small periodic motion for vibrato).

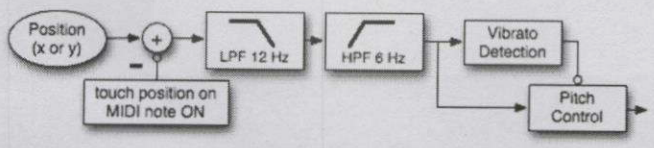


Figure 4. Data processing system for detecting and controlling vibrato based on finger motion. Arrows signify the flow of position data from the keyboard; connections with circles signify other data and triggers (e.g. changing parameters of a section).

Vibrato

To control vibrato (periodic oscillation in frequency), we had two primary goals:

1. Robust detection: vibrato should be easily activated, but with a minimum of inadvertent triggers
2. Accuracy of the centre pitch while performing the vibrato

By analogy to the violin, we chose a side-to-side rocking gesture to control vibrato, as we found sideways motion easier to control than front-to-back motion (confirmed in user studies below). However, since X-axis sensing is only available on the front of the white keys, rocking along the lengthwise (Y) axis was used for the remaining parts of the keyboard.

A simple approach might map relative finger position to pitch bend value, perhaps with a minimum threshold to activate. However, this fails to adequately address either objective: thresholds small enough to make the vibrato usable result in unacceptable numbers of false triggers, and the centre pitch is easily detuned as the finger rocks back and forth. Instead, we designed and fine-tuned a system (Figure 4) which filters out unidirectional movements and slow finger drift from its original position. Only when an oscillatory motion in a specific frequency range is detected does the pitch bending engage. A high-pass filter on the input position data ensures that the pitch always remains centred.

Position Data Filtering

The vibrato system consists of two stages: filtering and oscillatory motion detection. In the first stage, the finger position relative to note onset is low-pass filtered at 12Hz (1st-order) to smooth out irregularities in the sensor signal. A 1st-order high-pass filter at 6Hz eliminates drift from the finger moving from its original location. The filter frequencies were determined empirically, and their slow roll-off allows oscillatory motion below 6Hz to be detected while blocking near-constant input.

Oscillatory Motion Detection

Figure 5 shows the features of the filtered signal that are used to detect oscillatory motion of the finger. This section of the system has three parameters:

- **Threshold:** after the filtered finger position crosses zero, it must reach this value (in either direction) to initiate vibrato detection. When the threshold is reached, the algorithm stores the maximum deviation of the filtered finger position. This parameter is expressed as a fraction of the key width and can be set independently for X and Y axes. Default values: $x = 0.05$, $y = 0.05$.
- **Ratio:** after the filtered position crosses the threshold and the maximum value is stored, the finger position must then cross zero in the *other* direction (i.e. move right, then left or vice-versa). If X represents the maximum deviation in the first direction, the position must exceed $X * ratio$ in the other direction for vibrato to begin. Default values: $x = 0.3$, $y = 0.5$.
- **Minimum Detectable Frequency:** over a long enough time, the finger is likely to move in both directions even when vibrato is not desired. This parameter calculates a timeout (orange in Figure 5) between the first threshold cross and the second (ratio) cross in the opposite direction. Default value: 1.25Hz.

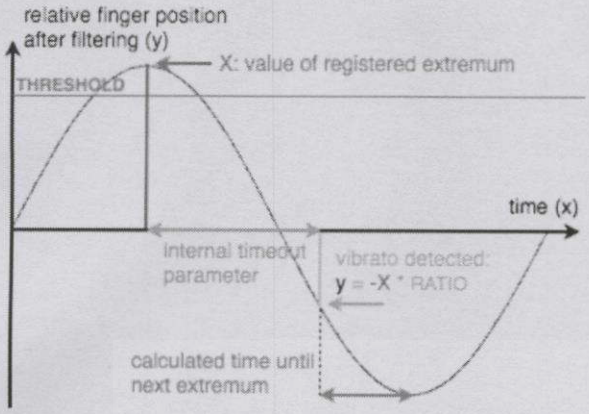


Figure 5. Parameters of the oscillatory motion detection algorithm. The waveform represents filtered finger position relative to point of onset.

Pitch Control and Vibrato Exit

Once vibrato is initiated, the pitch bend value is scaled to the filtered relative finger position. The total pitch bend range is a user-adjustable parameter, defaulting to 3 semitones. When the interval between zero-crossings of the filtered signal exceeds 1 / (minimum detectable frequency), vibrato is discontinued and the pitch bend returns gradually to 0.

Pitch Bends and Slides

We implemented a system for the performer to deliberately bend the pitch of a note, which can be used to make *portamento* (continuous-pitch) transitions between notes. The design goals were:

1. Robust detection of bend motions
2. Controllable intonation: the ability to play in tune easily while retaining control over nuances of pitch

Pitch bends can potentially span wide intervals, so it is appropriate to use the longer Y dimension of the key to control them. Since the physical motions are larger, the threshold of motion required for detection can also be larger, helping separate deliberate gestures from unintentional position variations. The total pitch range of the key is a user-adjustable parameter, with typical values ranging from 3 to 12 semitones.

Of the two goals, the second (intonation) proved more challenging: unlike string players, keyboardists are not accustomed to continuous pitch control, nor to the feel of a fretless surface on which the right pitch must be located. However we aim to allow a player unfamiliar with the system the ability to play in tune with only minutes of training. Therefore we developed a *pitch snapping* algorithm which causes the pitch to gravitate toward the steps of the chromatic scale.

Threshold Detection

Figure 6 shows the operation of the pitch bend system. Initially, the finger position is calculated relative to MIDI note onset. When the relative position crosses a pre-defined threshold, the bend algorithm engages. Our initial approach was to define the threshold as a fraction of a semitone, however for large total bend ranges (e.g. 12 semitones for the length of the key), this produces an unacceptably small

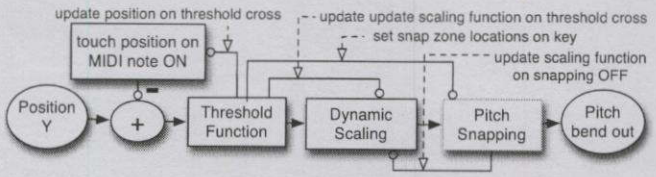


Figure 6. System for detecting and controlling pitch bends from finger motion in Y axis.

threshold and interferes with later parts of the algorithm. Thus we use two thresholds, one based on total pitch bend distances (fixed at 0.4 semitones in either direction) and another based on a fraction of the key length (user-adjustable; see Figure 7).

Dynamic Scaling

While the finger is within the threshold zone, no pitch bend occurs. Once it exits this zone, the pitch bend should engage gradually without an abrupt jump. Therefore the centre-point (zero bend) is recalculated to the edge of the zone. But to maintain consistency on the total pitch bend range of the key, the resulting map between finger location and pitch needs to be warped (Figure 7). This ensures that larger finger motions, which are less likely than fine adjustments to be executed by ear, have predictable points of arrival near the key edges.

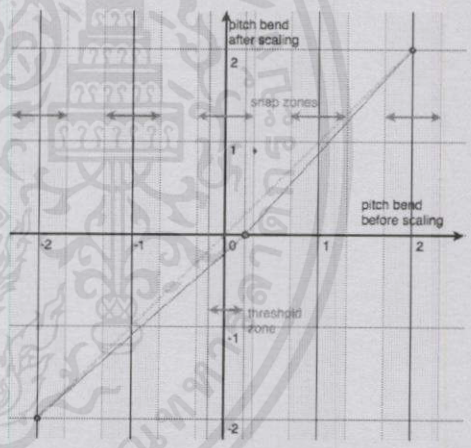


Figure 7. Threshold for pitch bend detection and resulting dynamic scaling of pitch.

Pitch Snapping

Figure 8 shows the algorithm for guiding pitch bends to the nearest semitone (right) and its conditions for activation (left). When a key is pressed, the algorithm calculates the positions along the Y axis which correspond to the steps of the chromatic scale. The *snap zones* are defined around these points, with a parameter (*snap zone size*) defining their width in semitones (Figure 9). For example, with a snap zone size of 0.3, any touch within 0.3 semitones of a chromatic pitch would engage the algorithm. The zone locations generally remain static throughout a note, even though the *dynamic scaling* function might slightly adjust the actual points corresponding to each chromatic pitch.

Snapping is engaged when a finger enters a snap zone and its speed of motion falls below a certain value. In this case, the note's pitch gets "pulled" toward the exact chromatic pitch. If

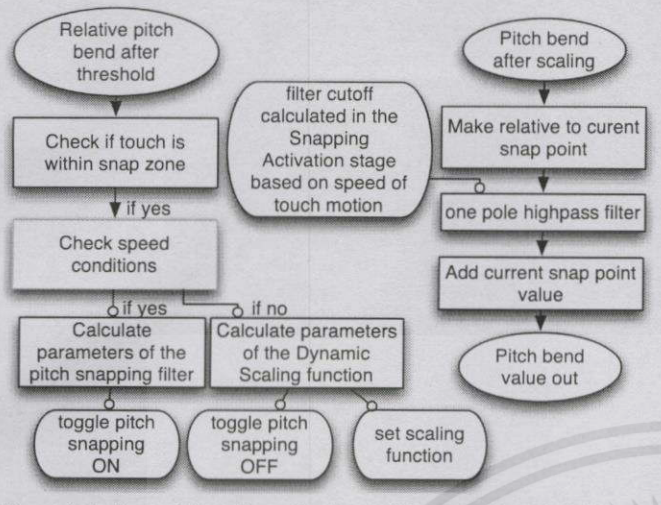


Figure 8. Left: conditions for activation and de-activation of pitch snapping, based on touch location and speed. Right: data flow in the pitch snapping algorithm, which pulls the pitch to the nearest semitone.

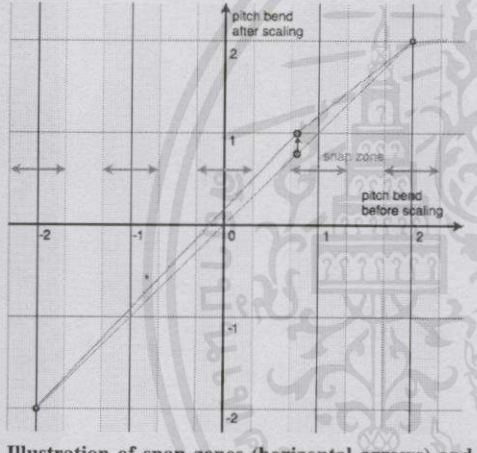


Figure 9. Illustration of snap zones (horizontal arrows) and dynamic scaling (comparison of scaled pink line to unscaled blue line).

the snap zone size is set to 0.5 semitones, then every point on the key surface is part of a snap zone, and it becomes nearly impossible to play out of tune: wherever the finger stops, the pitch is always pulled toward a chromatic step. Snap zone size is a user-adjustable parameter, and its effect is examined in the user studies below.

Parameters

In summary, the primary user-adjustable parameters are the *threshold* (expressed as a fraction of key length), the *snap zone size* (expressed as a fraction of a semitone) and the total *pitch bend range* for the whole key (expressed in semitones). Default values are 0.1, 0.4 and 7, respectively.

STUDY 1: EXAMINING TRADITIONAL TECHNIQUE

To better understand the constraints of traditional technique and to evaluate our expressive pitch control system, we conducted a study with 8 pianists (5 female, mean age 23.9, range 18-28). 7 were currently enrolled in or recently graduated from conservatory; 1 had recently graduated from (non-conservatory) university. The pianists had been playing piano for an average of 17.5 years (range 12-23).

Each session began with performances of three Inventions by J.S. Bach (BWV 772-786) and the Preludes in G major and B minor by Frédéric Chopin (op. 28 nos. 3 and 6). Each pianist was given a different selection of Inventions (out of 15 total); all played the same Preludes. The purpose was to capture the physical motions of the fingers during traditional piano performance; sensor data was logged, but no new techniques were added. In addition to musicological implications to be explored in future work, the study sought to verify or refute earlier informal findings about the aspects of finger motion that could be safely repurposed for expressive pitch control.

Mechanical and Tactile Considerations

Participants played on the instrument in Figure 1. After playing the Bach Inventions, each participant was asked to comment on how the feel of the instrument differed from a traditional piano. 4 of 8 players focused not on the sensors but the mechanical action of the electronic piano, which was felt to be less sensitive than an acoustic grand piano. 2 participants indicated that the action was realistic for an electronic keyboard, but none felt it was identical to a grand piano.

The most commonly identified drawback to the sensor system was the edges of the black keys (mentioned by 5 participants initially and 1 more on follow-up questioning), particularly the tendency for the fingers to catch on the sides when dragging the hand across the keyboard. The texture of the sensor surface was generally well-received. 5 participants found nothing at all unusual about it. 3 more commented on the texture, of which 1 found it objectionable ("too metallic"). None felt that the keys were significantly too sticky or too slippery.

Finger Motion

Detailed analysis of each performance is beyond the scope of this paper; however, the aggregate data revealed some interesting results. In addition to keys pressed, pianists frequently rested the hands on keys that were not played: in fact, fewer than 50% key touches were associated with a MIDI note. The location of touches that do not correspond to a MIDI note may be useful for locating the position of the hands or anticipating the player's next move, factors which will be explored in future work. This result also suggests that using touches on non-pressed keys as an expressive control dimension would create problems for traditional technique.

Both vibrato and pitch bend algorithms rely on finger motion relative to its starting location. Figure 10 shows histograms of the amount of finger motion per note on the X and Y axes, expressed as a fraction of the key length/width. Table 1 presents details of finger motion broken down by participants and pieces. The greater amount of motion in the X axis is to be expected given the key is much longer than it is wide. Overall, we found that motion of the fingers toward the player's body was more common ("pulling" motion; 63%) than motion away ("pushing"). Despite the different musical styles, we found little difference in the patterns of motion between Bach and Chopin, but more variation among players.

After our pre-studies, we settled on 0.1 as a default threshold for the pitch bend algorithm: finger motion within this range will not trigger a change in pitch. In our recordings of Bach

	X Mean	$X > 0.05$	$X > 0.1$	$X > 0.05$ (both dir.)	Y Mean	$Y > 0.05$	$Y > 0.1$	$Y > 0.05$ (both dir.)
All	0.153	75%	54%	8.9%	0.044	26%	11%	0.30%
Participant 1	0.124	66%	43%	8.4%	0.035	20%	7.9%	0.18%
Participant 2	0.171	80%	60%	8.9%	0.039	23%	8.7%	0.20%
Participant 3	0.154	81%	57%	8.3%	0.059	33%	17%	0.83%
Participant 4	0.126	66%	44%	6.2%	0.039	23%	9.0%	0.14%
Participant 5	0.171	78%	61%	13.6%	0.058	36%	18%	0.15%
Participant 6	0.185	82%	64%	9.2%	0.044	25%	11%	0.33%
Participant 7	0.149	76%	54%	10.4%	0.045	29%	10.3%	0.53%
Participant 8	0.144	73%	52%	8.2%	0.035	22%	7.1%	0.11%
All Bach	0.141	75%	52%	9.4%	0.044	27%	11%	0.16%
All Chopin	0.164	76%	57%	8.4%	0.045	25%	11%	0.47%

Table 1. Amount of finger motion on key surface over the course of a note, measured as the maximum deviation of the finger from its starting location. Table indicates the percent of notes deviating more than a specified value from onset; "both dir." indicates notes that deviate more than 0.05 in both directions from onset, likely to trigger a vibrato gesture.

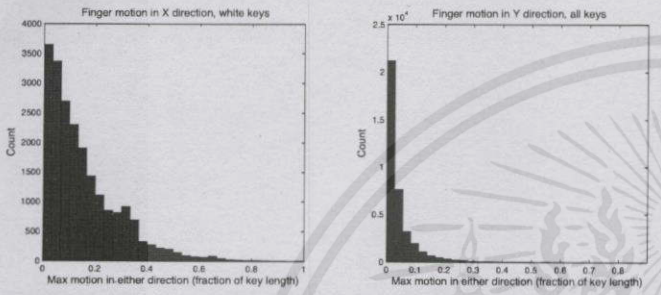


Figure 10. Histogram of maximum touch motion on a pressed note, X and Y axes.

and Chopin, 89% of notes fall below this motion threshold. 97% fall below a higher (but still playable) pitch-bend threshold of 0.2. This implies that with no change whatsoever to traditional technique, notes within the threshold would sound normally without inadvertent pitch bends. It is not necessarily a ceiling on performer accuracy: with an awareness of pitch bends and acoustic feedback from the instrument, it is likely that finger motion would be more precisely controlled.

Horizontal motion on the white keys is quite common, and accordingly, the vibrato algorithm requires evidence of a periodic motion to engage. We use motion in both directions of greater than 0.05 as a proxy for vibrato, though bidirectional motion will only trigger the vibrato if it happens in a short period of time. By this metric, approximately 9% of X-axis touches and virtually no Y-axis touches have the possibility of activating the vibrato function.

Example

Figure 11 shows an example case which involved large touch motions on the key surface. This example comes from the left hand of the B minor Prelude. The Y-axis motion reflects a combination of a deliberate technique of moving the hands toward and away from the keyboard and the mechanical constraints of shifting the hand across octaves. Future work will analyse these performances in more detail to identify scenarios in which the fingers are expected to move significantly.

STUDY 2: NEW TECHNIQUE EVALUATION

After the study of traditional technique, we evaluated our vibrato and pitch snapping systems with the same 8 participants. These tests were conducted on the previous generation interface [15] as the algorithms had been optimised in pre-studies for this sensor configuration.

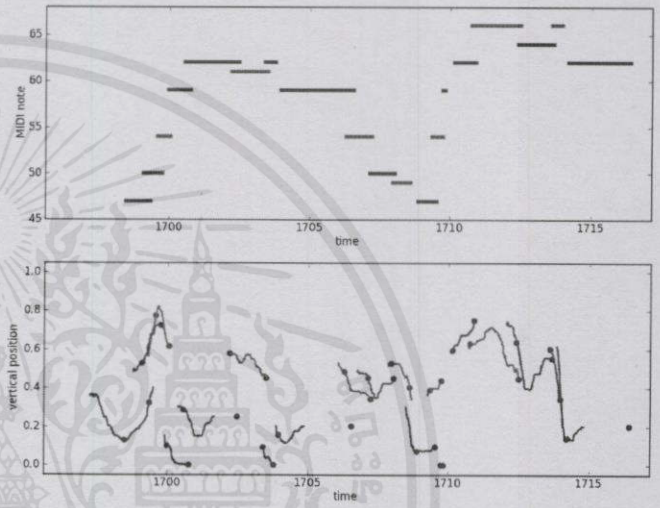


Figure 11. Example finger motion during a performance of the Chopin B minor Prelude, op. 28 no. 6. Top: MIDI notes over time; bottom: Y location of finger. Colours indicate black and white keys.

Vibrato

Each participant was initially asked to play vibratos on the white keys using side-to-side hand motion (X axis). After they became familiar with this technique, they were asked to play vibrato on the black keys with a front-to-back gesture (Y axis). Participants unanimously felt that the white key motion was natural and intuitive, but that the black key motion was not. One participant observed that the rolling wrist motion used on the white key vibrato is similar to wrist technique when playing rapidly alternating octaves. By contrast, several participants observed that the larger arm muscles were required to move the hand forwards and backwards; two participants attempted to play vibrato on the black keys by turning the forearm at a right angle.

Once the basic gestures were familiar, participants were asked to explore a range of parameter values for *sensitivity* and *pitch bend range*. Data was lost on one participant's choices, but on average, the remaining 7 chose a threshold of 0.027 and a pitch bend range of 2.2 semitones (edge-to-edge of key), a more sensitive but narrower range than default. Most participants felt that narrower vibrato was more musically appropriate, and that higher sensitivity allowed the vibrato to engage more easily while still offering reasonable robustness to inadvertent triggers.

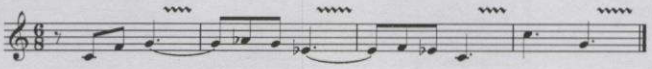


Figure 12. Test melody for selectively applying vibrato to certain notes.

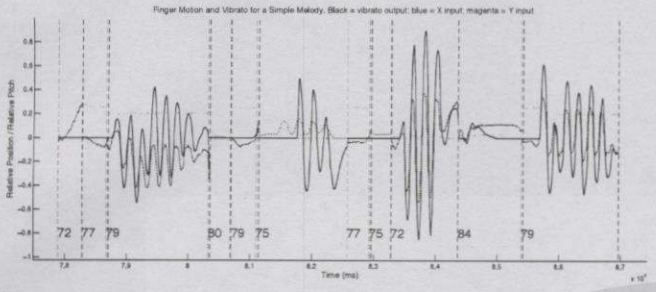


Figure 13. Example performance of a short melody with vibrato. Black line indicates output pitch; blue and magenta lines indicate X and Y relative position input. Numbers are MIDI notes.

Results in Melodic Playing

After setting their preferred parameters, each participant was asked to practice and perform a simple melody (Figure 12). Notes marked with a wavy line were to be played with vibrato, and the other notes without. Of the notes marked with vibrato, 100% had vibrato added (32/32), and 9% (5/56) of notes not marked vibrato nonetheless caused the algorithm to engage. 4 out of 5 incorrect vibrato triggers took place on the high C (m. 4), where they were generally triggered by hand motion preparing for the following G.

An example performance is shown in Figure 13. Note that the pitch stays centred even as the relative touch location drifts, fulfilling one of our primary goals.

Pitch Bends

To evaluate the accuracy and usability of the pitch bending interface, each pianist was asked to perform 24 single-note bends. For the first set of 12 bends, the total range of the key (i.e. the largest possible bend) was 7 semitones (*s.t.*); for second set of 12 bends, it was 5 s.t. Each set of 12 consisted of three intervals (2 s.t. up, 3 s.t. down, 5 s.t. up; 1 s.t. down; 3 s.t. up; 4 s.t. down) with 4 different settings for the *snap zone* parameter (0, 0.15, 0.3 and 0.4 s.t.). Presentation order within each set was randomised.

Participants were given 15 seconds to practice each bend, at which point they were asked to execute it a final time for evaluation. Each gesture was performed on a single key, and participants were free to choose any key on which to play it. The existence of the pitch-snapping algorithm (and its changing parameters) was not revealed to participants during the test.

Following the individual note test, the algorithm was explained and participants were invited to explore the *threshold* and *snap zone* parameters, choosing values that felt most natural. Each participant was then given a melody (Figure 14) to play which incorporated pitch bends. Questions that this evaluation sought to answer included:

- To what extent can pianists accurately control the tuning of bent notes?

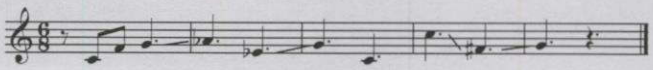


Figure 14. Test melody including pitch bends.

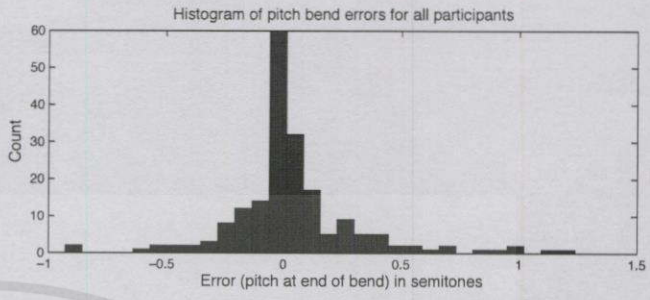


Figure 15. Pitch bend tuning errors aggregated across participants (absolute deviation between target and actual arrival pitch).

- Does the snapping algorithm improve tuning accuracy?
- Can pitch bends be triggered selectively only when desired, while other notes are left unaffected?

Analysis of the data began with manual annotation of the beginning of the bend (region of changing pitch), the end of the bend, and the end of the note. Since some participants exhibited slight pitch variations even upon arrival at the final note, the pitch was taken as a mean value between the end of the bend and just before the end of the note. The last few samples prior to note release were discarded to eliminate bias from the finger moving as it left the key. Error was calculated as the difference between the target pitch and the actual pitch, measured in semitones.

Results: Tuning Test

Figure 15 shows a histogram of tuning errors in the individual note test. The mean absolute error across all tests was 0.17 semitones, or 1.0% deviation in frequency. Accuracy varied widely by participant, from 0.067 semitone mean error (0.39% frequency error) to 0.28 semitones (1.6% frequency). Based on observation of the tests, it appeared that falling short of the target bend was more common than overshooting it; however, the final analysis did not show a significant effect one way or the other (46% of errors greater than 0.05 semitones overshot the target pitch).

Table 2 shows a breakdown of the tuning errors according to *snap zone size* along with pairwise t-test comparisons of each setting. The two larger snap zones (0.3 and 0.4 semitones) show significantly better pitch accuracy than the two smaller ones ($p < 0.007$). From this sample we do not observe a significant effect between no snapping at all (size 0) and the smallest snap (0.15), nor do we observe a significant difference between the two larger zones. We conclude that the pitch snapping algorithm achieves its goal of improving tuning, though we leave the fine-tuning of the zone size for future work (or perhaps to the performer).

Examining the results by target bend size, we observed the highest accuracy on the 1-semitone bend (Table 3) on the lowest accuracy on the 4- and 5-semitone bends, suggesting (but

Snap zone (s.t.)	0	0.15	0.3	0.4
Error (s.t.)	0.223	0.207	0.129	0.122
p-values for t-tests of zone size; bold = significant				
Snap zone (s.t.)	0	0.15	0.3	0.4
versus 0	-	0.74	**0.045**	**0.041**
versus 0.15	-	-	0.078	0.071
versus 0.3	-	-	-	0.88

Table 2. Pitch bend performance, tuning error according to snap zone parameter value.

Key range (s.t)	7	7	7	5	5	5
Interval (s.t)	2	-3	5	-1	3	-4
Error (s.t.)	0.17	0.14	0.24	0.067	0.17	0.24

Table 3. Pitch bend tuning error according to target interval size.

not proving) that small bends may be easier to execute. We did not observe a statistically significant difference in accuracy between upward bends (0.19 s.t. error) and downward bends (0.15 s.t. error; $p = 0.17$). We expected that we might observe an improvement in accuracy from beginning to end of the task as participants became accustomed to the gesture; however, this was not supported by the data. Comparing bends 2-6 against bends 20-24 (excluding the widely varying first attempt), the difference in accuracy (0.175 s.t. versus 0.123 s.t.) was not significant ($p = 0.23$). A larger sample size and longer training period would be needed to establish the extent to which pianists improve in accuracy over time.

Results: Melody

Prior to playing the melody, each participant set the *snap zone size* and *threshold* parameters to comfortable values. The mean choices were 0.24 semitones and 0.15 semitones, respectively (a larger threshold with smaller snap zone than default). However, it was unclear whether participants were able to fully internalise the effect of the *snap zone size* parameter within the limited period of exploration.

The results for the pitch bend melody were less consistent than for the melody with vibrato. One of the stronger performances is shown in Figure 16. While all notes marked with a bend had a bend applied, participants were often unhappy with their performance and would pause to replay a missed note, making a straightforward analysis of all performances difficult. In particular, participants found the leap to the high C followed by the large bend to F# (m. 4) awkward, as the finger had to be placed near the top of the upper C to allow a bend of 6 semitones down.

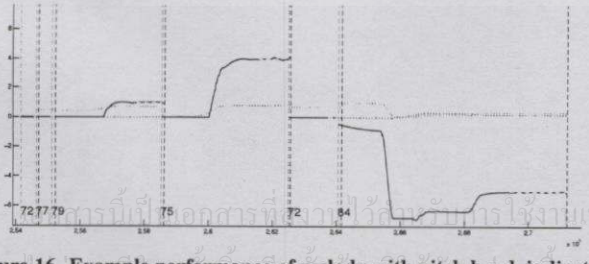


Figure 16. Example performance of melody with pitch bend, indicating relative pitch in semitones (black), raw X and Y input (dotted lines) and MIDI note numbers.

Discussion

The results of the vibrato and pitch bend tests indicate that these techniques are accurately playable by trained pianists. The melodic tests show that vibrato can be reliably applied to only selected notes, making it easily deployable in larger pieces of music.

Participants found the pitch bending in particular to be challenging, such that it would mainly find use in less technically complex passages. Since continuous pitch control is not a feature of traditional keyboard playing, several participants indicated that aural skills (hearing the right interval) was as significant a challenge as the physical motion. Nonetheless, the techniques were seen to be playable and useful. One participant, professionally active in theatre ensembles, indicated the techniques would be useful for playing other instrument sounds from the keyboard as theatre players are often asked to do. Two others mentioned the possibility of new music being composed for extended keyboard.

CONCLUSION: ENGAGING WITH EXPERTISE

Pianists spend thousands of hours mastering their instrument, in the process developing a highly specialised technique. It is not reasonable for a new musical interface to receive a similar amount of training, so existing technique should be considered a constraint just as much as basic mechanical principles. We have demonstrated a system which adds expressive pitch control to the keyboard which aims to be minimally intrusive to existing technique. Some adaptation will always be required for any new instrument (as any pianist-harpsichordist knows), but our results indicate that the new techniques are usable by trained pianists.

We offer two suggestions for designers (within or beyond the musical domain) seeking to add new sensor modalities to interfaces with pre-existing user expertise. First, before any behaviours are attached to new sensors, passive logging of the sensor data is useful to establish traditional patterns of interaction, with a focus on finding patterns that are *not* part of traditional use. In Benford's taxonomy of *expected*, *sensed* and *desired* interaction [1], passive observation can find patterns that are sensed and possibly desired, but not previously expected. Our vibrato example shows that useful patterns are often found from the temporal profile of sensor data rather than single values: every key press has a touch location, and these locations often change with time, but the particular case of an oscillating motion was not part of traditional technique.

Second, we suggest that a major problem with extending an existing interface is the potential for new techniques to be engaged unintentionally. Users may eventually adapt their technique to minimise unintentional engagement, but it could take several months of practice for this adaptation to fully develop. Sensor data from traditional use can provide a rapid and useful proxy for how much a new technique interferes with familiar use. We found that (algorithm setting dependent) fewer than 10% of notes in traditional performance would have triggered vibrato or pitch bends. This does not indicate a user's eventual false-positive rate, but shows a worst-case scenario where the user is unaware of the technique and receives no

feedback from it. We hypothesise that, in general, lower trigger rates in this passive case will translate to a better eventual separation between traditional and extended techniques.

Future Directions

The sensor hardware in this paper represented a refinement over earlier versions, but user testing indicates that further changes would be useful. In particular, the ability to sense horizontal motion on the black keys would substantially improve the user experience of playing vibrato on these notes. Future investigations will also examine existing piano technique more closely, particularly the relationship between expressive intentions and physical motion on the keyboard.

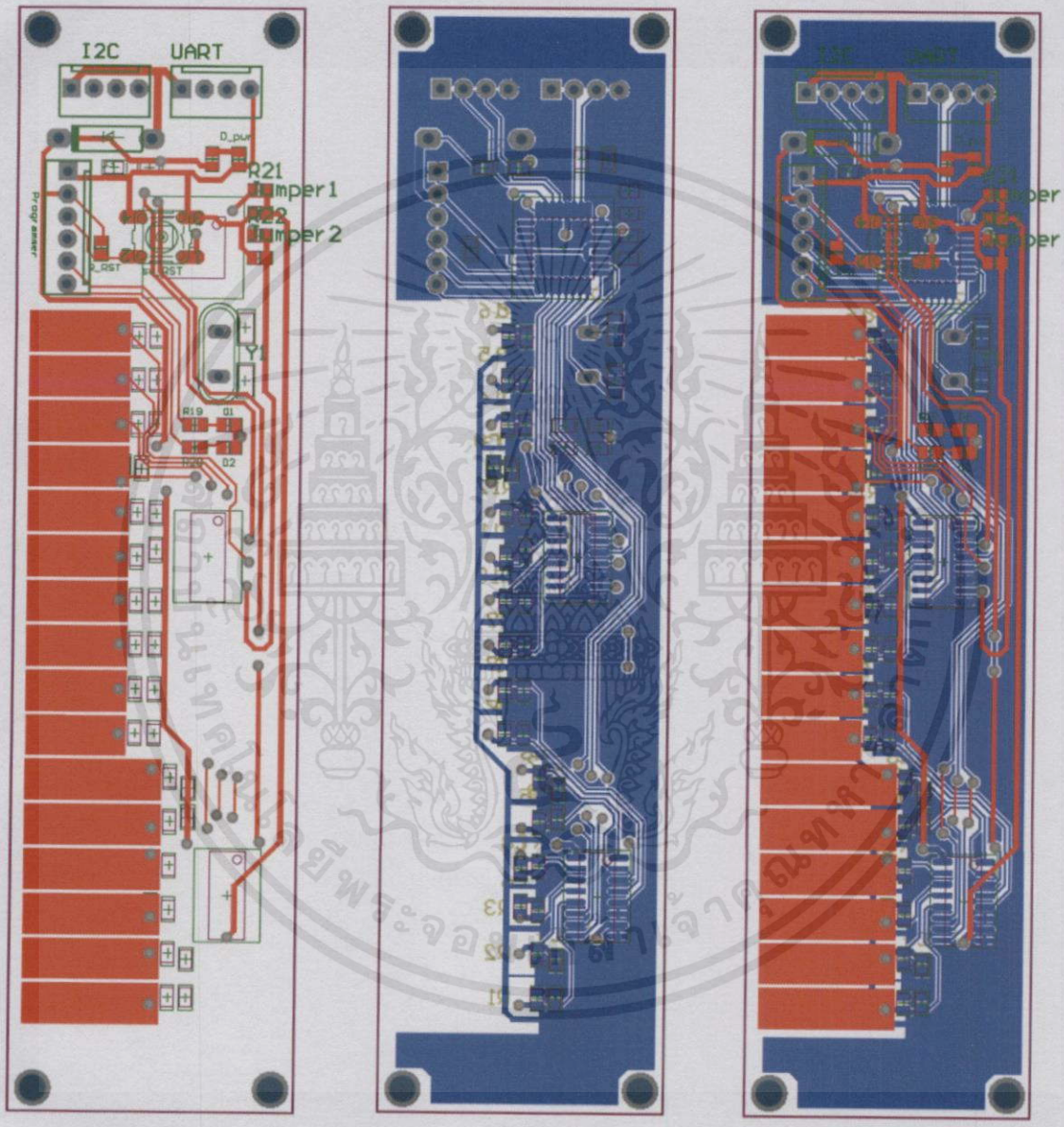
ACKNOWLEDGMENTS

This work was supported by the UK Arts and Humanities Research Council through a "Digital Transformations in Arts and Humanities" grant (AH/J013145/1).

REFERENCES

1. Benford et al., S. Expected, sensed, and desired: A framework for designing sensing-based interaction. *ACM Transactions on Computer-Human Interaction 12* (2005), 3–30.
2. Bernays, M., and Traube, C. Piano touch analysis: A MATLAB toolbox for extracting performance descriptors from high-resolution keyboard and pedalling data. In *Proc. JIM* (2012).
3. Bevilacqua, F., Zamborlin, B., Sypniewski, A., Schnell, N., Guédy, F., and Rasamimanana, N. Continuous realtime gesture following and recognition. In *Gesture in Embodied Communication and Human-Computer Interaction*, vol. 5934 of *Lecture Notes in Computer Science*. Springer, 2010, 73–84.
4. Brent, W. The gesturally-extended piano. In *Proc. NIME* (2012).
5. Burnett, G. E. On-the-move and in your car: An overview of HCI issues for in-car computing. *Intl. J. Mobile HCI 1*, 1 (2009), 60–78.
6. Findlater, L., Lee, B. Q., and Wobbrock, J. O. Beyond QWERTY: Augmenting touch-screen keyboards with multi-touch gestures for non-alphanumeric input. In *Proc. CHI* (2012).
7. Goebl, W., and Palmer, C. Tactile feedback and timing accuracy in piano performance. *Experimental Brain Research 186*, 3 (April 2008), 471–479.
8. Hadjakos, A., Aitenbichler, E., and Mühlhäuser, M. Potential use of inertial measurement sensors for piano teaching systems: Motion analysis of piano playing patterns. In *Proc. i-Maestro Workshop on Technology-Enhanced Music Education* (2008).
9. Harrison, C., and Hudson, S. E. Using shear as a supplemental two-dimensional input channel for rich touchscreen interaction. In *Proc. CHI* (2012).
10. Hasan, K., Yang, X. D., Bunt, A., and Irani, P. A-Coord input: Coordinating auxiliary input streams for augmenting contextual pen-based interactions. In *Proc. CHI* (2012).
11. Jordà, S. Instruments and players: Some thoughts on digital lutherie. *J. New Music Research 33* (2004).
12. Lee, H., and Kim, J. An HMM-based threshold model approach for gesture recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence 21* (1999), 961–973.
13. MacRitchie, J., Buck, B., and Bailey, N. J. Visualising musical structure through performance gesture. In *Proc. ISMIR* (2009).
14. McPherson, A. TouchKeys: Capacitive multi-touch sensing on a physical keyboard. In *Proc. NIME* (2012).
15. McPherson, A., and Kim, Y. Design and applications of a multi-touch musical keyboard. In *Proc. SMC* (2011).
16. McPherson, A., and Kim, Y. Multidimensional gesture sensing at the piano keyboard. In *Proceedings of the 29th ACM Conference on Human Factors in Computing Systems (CHI)* (Vancouver, Canada, 2011).
17. Moog, R. A., and Rhea, T. L. Evolution of the keyboard interface: The Bösendorfer 290 SE recording piano and the Moog multiply-touch-sensitive keyboards. *Computer Music Journal 14*, 2 (Summer 1990), 52–60.
18. Nicolls, S. Twenty-first century piano. In *Proc. NIME* (2009).
19. NJ Star-Ledger. Waiting to be heard, Wanted: Someone to bring the successor to the Moog synthesizer to life.
20. Ortmann, O. *The Physiological Mechanics of Piano Technique*. Kegan Paul, Trenc, Trubner & Co., 1929.
21. Paradiso, J. A. Electronic music: new ways to play. *IEEE Spectrum 34*, 12 (1997).
22. Poole, A., and Ball, L. Eye tracking in HCI and usability research. In *Encyclopedia of Human-Computer Interaction*, C. Ghaoui, Ed. 2006.
23. Pressing, J. Cybernetic issues in interactive performance systems. *Computer Music J. 14*, 1 (1990).
24. Rahman, M., Gustafson, S., Irani, P., and Subramanian, S. Tilt techniques: investigating the dexterity of wrist-based input. In *Proc. CHI* (2009).
25. Rothstein, J., and Metlay, M. P. Products of interest: The musician-machine interface to MIDI. *Computer Music Journal 14*, 2 (1990), 73–83.
26. Scarr, J., Cockburn, A., Gutwin, C., and Quinn, P. Dips and ceilings: Understanding and supporting transitions to expertise in user interfaces. In *Proc. CHI* (2011).
27. Stewart, C., Rohs, M., Kratz, S., and Essl, G. Characteristics of pressure-based input for mobile devices. In *Proc. CHI* (2010).
28. Wang, F., and Ren, X. Empirical evaluation for finger input properties in multi-touch interaction. In *Proc. CHI* (2009).
29. Xin, Y., Bi, X., and Ren, X. Natural use profiles for the pen: An empirical exploration of pressure, tilt, and azimuth. In *Proc. CHI* (2012).
30. Yang, Q., and Essl, G. Augmented piano performance using a depth camera. In *Proc. NIME* (2012).

PCB layout ของบอร์ด Keyboard touch sensor (2-layer)



Top layer

Bottom layer

โปรแกรมส่วนของ main ที่ใช้สำหรับ Keyboard touch sensor (พัฒนาบน AVR studio 4 ด้วยภาษา C)

```
#include<avr/io.h>
#include<util/delay.h>
#include<avr/interrupt.h>
#include<compat/deprecated.h>
#include <compat/twi.h>
#include "adc_megax8.h"
#include "i2c_slave.h"
#include "touchkey.h"
#include "uart_megaX8.h"
#define         F_CPU 16000000UL
volatile unsigned char slave_address=0x50;
volatile unsigned char data_address;
volatile unsigned char data[0xFF];
////////////////////////////////////////////////////////////////////////////////
#define THRESHOLD_FILTER_OUT_0          200
#define THRESHOLD_FILTER_OUT_1          200
#define THRESHOLD_FILTER_OUT_2          200
#define THRESHOLD_FILTER_OUT_3          200
#define THRESHOLD_FILTER_OUT_4          200
#define THRESHOLD_FILTER_OUT_5          200
#define THRESHOLD_FILTER_OUT_6          200
#define THRESHOLD_FILTER_OUT_7          200
#define THRESHOLD_FILTER_OUT_8          200
#define THRESHOLD_FILTER_OUT_9          200
#define THRESHOLD_FILTER_OUT_10         200
#define THRESHOLD_FILTER_OUT_11         200
#define THRESHOLD_FILTER_OUT_12         200
#define THRESHOLD_FILTER_OUT_13         200
#define THRESHOLD_FILTER_OUT_14         200
#define THRESHOLD_FILTER_OUT_15         200
////////////////////////////////////////////////////////////////////////////////
```

```
int voltage[16];
int filter_voltage[16];
int ref_filter_voltage[16];
int delay_voltage[16];
int diff_filter_voltage[16];
int delay_diff_filter[16];
float k1 = 0.125;
float k2 = 0.125;
float k3 = 0.250;
//for thermal code algorithm
unsigned char count = 0;
int electrode_value[16];
int     slider_value = 0;
//for graph linking algorithm
unsigned char link_count=0;
int     slope;
int link_slider_value;
int delay_link_slider_value;
//for IIR filter stage 2
int filter_slider_value;
int delay_filter_slider_value;

////////////////////////////////////////////////////////////////////////////////

void led_control(unsigned char led, unsigned char state)    //state = 1 -> led on
{       if(led == 1)
        {       if(state == 1)  PORTB |= 0b00001000;
                else                    PORTB &= ~(0b00001000);
        }
        else if(led == 2)
        {       if(state == 1)  PORTB |= 0b00010000;
                else                    PORTB &= ~(0b00010000);
        }
}
```

```
int main()
{
	touchkey_setup();
	Uart_Init(19200,1,0,0,0);
	i2c_slave_initi(slave_address);
	cli();
	//read steady state voltage -> ref filter voltage
	sbi(PORTB,3);
	sbi(PORTB,4);
	for(int loop=1;loop<=100;loop++)
	{	for(int i=0;i<=15;i++)
		{	voltage[i] = RC_discharge_measure(i);
			ref_filter_voltage[i] = voltage[i] + (int)(delay_voltage[i]*(1-k1));
			delay_voltage[i] = ref_filter_voltage[i];
		}
	}
	cbi(PORTB,3);
	cbi(PORTB,4);
	sei();
	while(1)
	{
		led_control(1, 1);
		//////////////////////Scaning the Electrodes with IIR filter (1st stage)//////////////////
		for(int i=0;i<=15;i++)
		{
			voltage[i] = RC_discharge_measure(i);
			filter_voltage[i] = voltage[i] + (int)(delay_voltage[i]*(1-k1));
			delay_voltage[i] = filter_voltage[i];

			diff_filter_voltage[i] = filter_voltage[i] - ref_filter_voltage[i];

		}
		////////////////////////////////////////////////////////////////////////////////////////
```

```
//////////////////////thermometercode algorithm////////////////////////////////
    if(diff_filter_voltage[0] > THRESHOLD_FILTER_OUT_0)       electrode_value[0] = 1;
    else if(diff_filter_voltage[0] <= THRESHOLD_FILTER_OUT_0) electrode_value[0] = 0;
    if(diff_filter_voltage[1] > THRESHOLD_FILTER_OUT_1)       electrode_value[1] = 1;
    else if(diff_filter_voltage[1] <= THRESHOLD_FILTER_OUT_1) electrode_value[1] = 0;
    if(diff_filter_voltage[2] > THRESHOLD_FILTER_OUT_2)       electrode_value[2] = 1;
    else if(diff_filter_voltage[2] <= THRESHOLD_FILTER_OUT_2) electrode_value[2] = 0;
    if(diff_filter_voltage[3] > THRESHOLD_FILTER_OUT_3)       electrode_value[3] = 1;
    else if(diff_filter_voltage[3] <= THRESHOLD_FILTER_OUT_3) electrode_value[3] = 0;
    if(diff_filter_voltage[4] > THRESHOLD_FILTER_OUT_4)       electrode_value[4] = 1;
    else if(diff_filter_voltage[4] <= THRESHOLD_FILTER_OUT_4) electrode_value[4] = 0;
    if(diff_filter_voltage[5] > THRESHOLD_FILTER_OUT_5)       electrode_value[5] = 1;
    else if(diff_filter_voltage[5] <= THRESHOLD_FILTER_OUT_5) electrode_value[5] = 0;
    if(diff_filter_voltage[6] > THRESHOLD_FILTER_OUT_6)       electrode_value[6] = 1;
    else if(diff_filter_voltage[6] <= THRESHOLD_FILTER_OUT_6) electrode_value[6] = 0;
    if(diff_filter_voltage[7] > THRESHOLD_FILTER_OUT_7)       electrode_value[7] = 1;
    else if(diff_filter_voltage[7] <= THRESHOLD_FILTER_OUT_7) electrode_value[7] = 0;
    if(diff_filter_voltage[8] > THRESHOLD_FILTER_OUT_8)       electrode_value[8] = 1;
    else if(diff_filter_voltage[8] <= THRESHOLD_FILTER_OUT_8) electrode_value[8] = 0;
    if(diff_filter_voltage[9] > THRESHOLD_FILTER_OUT_9)       electrode_value[9] = 1;
    else if(diff_filter_voltage[9] <= THRESHOLD_FILTER_OUT_9) electrode_value[9] = 0;
    if(diff_filter_voltage[10] > THRESHOLD_FILTER_OUT_10)     electrode_value[10] = 1;
    else if(diff_filter_voltage[10] <= THRESHOLD_FILTER_OUT_10) electrode_value[10] = 0;
    if(diff_filter_voltage[11] > THRESHOLD_FILTER_OUT_11)     electrode_value[11] = 1;
    else if(diff_filter_voltage[11] <= THRESHOLD_FILTER_OUT_11) electrode_value[11] = 0;
    if(diff_filter_voltage[12] > THRESHOLD_FILTER_OUT_12)     electrode_value[12] = 1;
    else if(diff_filter_voltage[12] <= THRESHOLD_FILTER_OUT_12) electrode_value[12] = 0;
    if(diff_filter_voltage[13] > THRESHOLD_FILTER_OUT_13)     electrode_value[13] = 1;
    else if(diff_filter_voltage[13] <= THRESHOLD_FILTER_OUT_13) electrode_value[13] = 0;
    if(diff_filter_voltage[14] > THRESHOLD_FILTER_OUT_14)     electrode_value[14] = 1;
    else if(diff_filter_voltage[14] <= THRESHOLD_FILTER_OUT_14) electrode_value[14] = 0;
    if(diff_filter_voltage[15] > THRESHOLD_FILTER_OUT_15)     electrode_value[15] = 1;
    else if(diff_filter_voltage[15] <= THRESHOLD_FILTER_OUT_15) electrode_value[15] = 0;
```

```
            slider_value = 0;       //reset slider value for calc a new value
            count = 0;
            for(int i=0;i<=15;i++)
            {
                  if(electrode_value[i] == 1)
                  {       slider_value = slider_value + (2*i) + 1;
                        count++;
                  }
            }
            if(count != 0)          slider_value = slider_value/count;
            //else                        slider_value = -10;     //no touch status
            ////////////////////////////////////////////////////////////////////////////////////
            ////////////////////////////////graph linking algorithm/////////////////////////////////
            if(slider_value == 0)
            {       if((delay_link_slider_value > 0)&&(link_count < 4))
                  {       link_slider_value = delay_link_slider_value;
                        link_count++;
                  }
                  else
                  {       link_slider_value = slider_value;
                        link_count = 0;
                  }
            }
            else
            {       slope = slider_value - delay_link_slider_value;
                  link_slider_value = slider_value;
                  link_count = 0;
            }
            delay_link_slider_value = link_slider_value;
            ////////////////////////////////////////////////////////////////////////////////////
            data[0x01] =  link_slider_value;
            led_control(2, 0);
      }
      return 0;
}
```

```
ISR(TWI_vect)
{
  static unsigned char i2c_state;
  unsigned char twi_status;
  // Disable Global Interrupt
  cli();
  // Get TWI Status Register, mask the prescaler bits (TWPS1,TWPS0)
  twi_status=TWSR & 0xF8;

  switch(twi_status) {
   case TW_SR_SLA_ACK:      // 0x60: SLA+W received, ACK returned
    i2c_state=0;          // Start I2C State for Register Address required

    TWCR |= (1<<TWINT);   // Clear TWINT Flag
    break;
   case TW_SR_DATA_ACK:     // 0x80: data received, ACK returned
    if (i2c_state == 0)
        {
                  data_address = TWDR;                  // Save data to the register address
                  i2c_state = 1;
    }
        else
        {
                  data[data_address] = TWDR;      // Save to the register data
                  i2c_state = 2;
    }

    TWCR |= (1<<TWINT);   // Clear TWINT Flag
    break;
   case TW_SR_STOP:         // 0xA0: stop or repeated start condition received while selected
    if (i2c_state == 2)
        {
                  i2c_state = 0;        // Reset I2C State
    }
```

```
        TWCR |= (1<<TWINT);    // Clear TWINT Flag
        break;

      case TW_ST_SLA_ACK:      // 0xA8: SLA+R received, ACK returned
      case TW_ST_DATA_ACK:     // 0xB8: data transmitted, ACK received
        if (i2c_state == 1)
            {
                        TWDR = data[data_address];      // Store data in TWDR register
                        i2c_state = 0;        // Reset I2C State
        }

        TWCR |= (1<<TWINT);    // Clear TWINT Flag
        break;
      case TW_ST_DATA_NACK:    // 0xC0: data transmitted, NACK received
      case TW_ST_LAST_DATA:    // 0xC8: last data byte transmitted, ACK received
      case TW_BUS_ERROR:       // 0x00: illegal start or stop condition
      default:
        TWCR |= (1<<TWINT);    // Clear TWINT Flag
        i2c_state = 0;         // Back to the Begining State
            data_address = 0;
    }
    // Enable Global Interrupt
    sei();
}
```