เครื่องติดตามสัญญาณชีพระยะไกลต้นทุนต่ำ

LOW-COST PATIENT VITAL SIGNS REMOTE MONITORING

โดย

นางสาวสุพจนา นราทอง

นางสาวอารียา เกตุย้อย

นางสาวอารียา หงษาชาติ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2565

เครื่องติดตามสัญญาณชีพระยะไกลต้นทุนต่ำ

LOW-COST PATIENT VITAL SIGNS REMOTE MONITORING

โดย

นางสาวสุพจนา นราทอง 62010969

นางสาวอารียา เกตุย้อย 62011066

นางสาวอารียา หงษาชาติ 62011067

อาจารย์ที่ปรึกษา

รศ.ดร.พิพัฒน์ พรหมมี

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2565

ปริญญานิพนธ์ปีการศึกษา 2565

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องติดตามสัญญาณชีพระยะไกลต้นทุนต่ำ

LOW-COST PATIENT VITAL SIGNS REMOTE MONITORING

ผู้จัดทำ

1	นางสาวสุพจนา นราทอง	62010969
2	นางสาวอารียา เกตุย้อย	62011066
3	นางสาวอารียา หงษาชาติ	62011067

..... **อาจารย์ที่ปรึกษา**

(รศ.ดร.พิพัฒน์ พรหมมี)

กิตติกรรมประกาศ

การดำเนินปริญญานิพนธ์ “เครื่องติดตามสัญญาณชีพระยะไกลต้นทุนต่ำ” จะไม่สามารถสำเร็จลุล่วงไปได้ด้วยดี หากไม่ได้รับความช่วยเหลือและความอนุเคราะห์อย่างดียิ่งจากอาจารย์ที่ปรึกษาประจำกลุ่ม รศ.ดร. พิพัฒน์ พรหมมี ที่ให้คำปรึกษา แนะแนวทางในการดำเนินงาน คำสั่งสอน คอยตรวจสอบข้อผิดพลาดและแก้ไขข้อบกพร่องต่างๆที่เกิดขึ้นระหว่างดำเนินปริญญานิพนธ์นี้ รวมทั้งพี่นักศึกษาปริญญาเอก ภาควิชาโทรคมนาคมห้อง T108-A ที่กรุณา ให้คำแนะนำ คำปรึกษา และแนวทางการแก้ไขปัญหาที่เป็นประโยชน์ต่อการศึกษา ค้นคว้าวิจัยให้ปริญญานิพนธ์นี้สำเร็จสมบูรณ์ยิ่งขึ้น รวมถึงสนับสนุนสถานที่ เครื่องมือ และอุปกรณ์ต่างๆที่ใช้ระหว่างการจัดทำปริญญานิพนธ์

ขอขอบคุณคณาจารย์และเจ้าหน้าที่ประจำภาควิชาวิศวกรรมโทรคมนาคม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกท่าน ที่ได้อบรมสั่งสอน ประสิทธิ์ประสาทวิชา ความรู้ และประสบการณ์ให้แก่ผู้จัดทำ

ขอกราบขอบพระคุณ บิดา มารดา และครอบครัว ที่ให้ความรัก ความห่วงใย และเป็นกำลังใจที่สำคัญเสมอมาและที่สำคัญคือสนับสนุนทางด้านการศึกษาอันมีค่ายิ่งแก่ผู้จัดทำ

นางสาวสุพจนา นราทอง
นางสาวอารียา เกตุย้อย
นางสาวอารียา หงษาชาติ
ผู้จัดทำ

เครื่องติดตามสัญญาณชีพระยะไกลต้นทุนต่ำ

LOW-COST PATIENT VITAL SIGNS REMOTE MONITORING

โดย นางสาวสุพจนา นราทอง 62010969
นางสาวอารียา เกตุย้อย 62011066
นางสาว อารียา หงษาชาติ 62011067

อาจารย์ที่ปรึกษา รศ.ดร.พิพัฒน์ พรหมมี

บทคัดย่อ

ปริญญานิพนธ์นี้นำเสนอการออกแบบเครื่องติดตามสัญญาณชีพระยะไกลต้นทุนต่ำสำหรับผู้ป่วยที่อยู่ในภาวะวิกฤติ โดยอุปกรณ์ถูกออกแบบให้มีขนาดเล็กกะทัดรัด สามารถพกพาได้ และใช้พลังงานต่ำ ซึ่งอุปกรณ์จะประกอบด้วยตัวควบคุม ESP-WROOM-32 ที่ใช้สำหรับรวบรวมข้อมูลจากโมดูล AD8232 และโมดูล MAX30100 โดยสัญญาณชีพของผู้ป่วยหลายราย รวมถึงอัตราการเต้นของหัวใจ ปริมาณออกซิเจนในกระแสเลือด และสัญญาณ EKG จะถูกรวบรวมและแสดงในจอภาพ ของศูนย์พยาบาลผ่านระบบเครือข่าย Wi-Fi รวมทั้งระบบสามารถแจ้งเตือนและมีการส่งเสียงเตือนดังออกมาเมื่อตรวจพบความผิดปกติ นอกจากนี้ยังสามารถบันทึกประวัติย้อนหลังของผู้ป่วยได้

ABSTRACT

This presents a design of a low-cost patient vital signs remote monitoring for critical patients. The conceptual design of the devices is to be compact, have mobility, and have low energy consumption. The device contains the ESP controller for collecting data from ECG and oxygen sensors. The vital signs of multiple patients, including heart rate, oxygen, and ECG signal are collected and displayed in the nurse center monitor using a Wi-Fi network. The system can be warned, and sound alerted if some abnormal signals have been detected. In addition, the history signal can be recorded for physical investigation.

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูป

หน้า

รูปที่

สารบัญรูป (ต่อ)

หน้า

รูปที่

สารบัญรูป (ต่อ)

หน้า

รูปที่

สารบัญรูป (ต่อ)

หน้า

รูปที่

สารบัญตาราง

หน้า

ตารางที่

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เนื่องด้วยปัญหาของเครื่องติดตามสัญญาณชีพ (Patient Monitor) ที่ใช้ตรวจติดตามการทำงานของหัวใจผู้ป่วยและวัดความอิ่มตัวของออกซิเจนในกระแสเลือดของผู้ป่วยจากภายนอก ไม่สามารถดูข้อมูลจากต่างสถานที่ได้และมีมูลค่าสูงทำให้บางโรงพยาบาลมีเครื่องติดตามสัญญาณชีพที่จำกัด เราจึงได้นำความรู้และเทคโนโลยีที่มีอยู่มาพัฒนาเป็นอุปกรณ์เครื่องติดตามการทำงานของหัวใจและสัญญาณชีพต้นทุนต่ำ โดยสามารถแสดงข้อมูลย้อนหลังได้ทั้งแบบตัวเลขและรูปกราฟใน Web Application และหวังให้ผู้คนบางกลุ่มที่ไม่อาจเข้าถึง สามารถมีอุปกรณ์ที่จะสามารถมาทดแทนในส่วนของอุปกรณ์ที่มีมูลค่าสูง

1.2 วัตถุประสงค์

1) เพื่อศึกษาการออกแบบเครื่องติดตามการทำงานของหัวใจและสัญญาณชีพต้นทุนต่ำ
2) เพื่อเฝ้าระวังและติดตามสัญญาณชีพสำหรับผู้ป่วยที่อยู่ในภาวะวิกฤติ
3) เพื่อให้แพทย์ติดตามข้อมูลของผู้ป่วยในระยะทางไกลได้

1.3 ขอบเขตของปริญญานิพนธ์

1) ใช้เฝ้าติดตามสัญญาณชีพของผู้ป่วย โดยแสดงรูปคลื่นไฟฟ้าหัวใจ สามารถวัดอัตราการเต้นของหัวใจ และออกซิเจนในกระแสเลือด
2) ใช้ Wi-Fi ในการส่งสัญญาณข้อมูลในระยะทางไกล
3) ใช้แสดงข้อมูลย้อนหลังได้ทั้งแบบตัวเลขและรูปกราฟใน Web Application

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

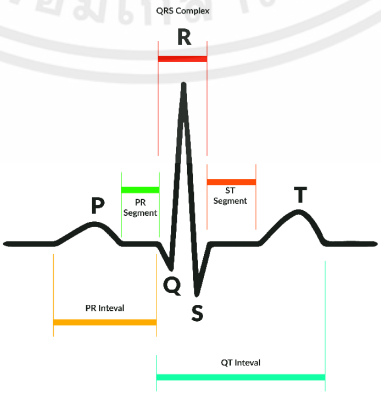
2.1 Patient Monitor

เครื่องติดตามสัญญาณชีพ ใช้ตรวจติดตามการทำงานของหัวใจผู้ป่วยชนิดข้าง เตียง โดยแสดงรูปคลื่นไฟฟ้าของหัวใจ อัตราการเต้นของหัวใจ ความอิ่มตัวของออกซิเจนในกระแสเลือดของผู้ป่วย จากภายนอก ซึ่งใช้ได้กับทารกแรกเกิด เด็กโต จนถึงผู้ใหญ่

2.1.1 การตรวจคลื่นไฟฟ้าหัวใจ

การตรวจคลื่นไฟฟ้าหัวใจ (Electrocardiography: ECG, EKG) เป็นการทดสอบที่บันทึกกระแสไฟฟ้าของหัวใจผ่านแผ่นอิเล็กโทรดขนาดเล็ก ที่ติดไว้ที่ผิวหนังของผู้ป่วย ขณะที่ไฟฟ้าผ่านกล้ามเนื้อหัวใจ กล้ามเนื้อหัวใจจะเกิดการหดตัว และคลายตัว เมื่อนำสัญญาณไฟฟ้ามาวางที่หน้าอกจะ ทำให้สามารถบันทึกคลื่นไฟฟ้าที่ออกจากหัวใจได้ ซึ่งปกติการตรวจคลื่นไฟฟ้าหัวใจ สามารถบอกได้หลายอย่าง เช่น ความผิดปกติของเยื้อหุ้มหัวใจ ณ ขณะนั้นว่ามีความผิดปกติหรือไม่ และยังเป็นการศึกษาการเต้นของหัวใจว่ามีการเต้นของหัวใจที่ผิดปกติหรือไม่ หัวใจเต้นช้าไปหรือว่าเร็วไป

การบันทึกคลื่นไฟฟ้าหัวใจดังรูปที่ 2.1 เป็นการใช้เครื่องบันทึกเป็นกราฟการเต้นของหัวใจ โดยเครื่องบันทึกจะใช้ความเร็ว ซึ่งกระดาษที่ใช้ มี 2 แบบ ได้แก่ 50 มิลลิเมตร / วินาที และ 25 มิลลิเมตร / วินาทีโดยการบันทึกคลื่นไฟฟ้าหัวใจ จะใช้เครื่องความเร็ว 25 มิลลิเมตร / วินาที ซึ่งทำให้ 1 ช่องเล็กของกระดาษบันทึกคลื่นไฟฟ้าหัวใจ มีค่าเท่ากับ 0.04 วินาที โดยความสูงของแต่ละช่องเล็กคิดเป็นความแรงของกระแสไฟฟ้า เท่ากับ 0.1 มิลลิโวลต์[1]



รูปที่ 2.1 สัญญาณคลื่นหัวใจ[2]

สัญญาณคลื่นหัวใจ จะสามารถอธิบายส่วนประกอบต่างๆได้ดังต่อไปนี้[1]

1) คลื่น P เป็นผลรวมของระยะหัวใจเมื่อถูกกระตุ้น (depolarization) ใน atrium มีทรงกลมเรียบความกว้างกว้างน้อยกว่า 0.12 วินาที ความสูงน้อยกว่า 2.5 มิลลิเมตร ซึ่งคลื่น P หนึ่งคลื่นต่อ QRS complex หนึ่งคลื่นหัวตั้งใน lead I, II, AVF, V4 V5 V6 และหัวกลับใน AVR

2) PR Interval เป็นช่วงระยะของสัญญาณไฟฟ้า จนถึงจุดเริ่มระยะ ซึ่งจะมียาวเท่ากับ 0.12 ถึง 0.20 วินาที

3) QRS complex เกิดเมื่อถูกกระตุ้น และการหดตัวของเวนตริเคิล ปกติเกิด 0-100 ครั้งต่อนาที มีจังหวะคงที่ ลักษณะคลื่นสูงแคบ 0.04 ถึง 0.10 วินาที

4) คลื่น Q เป็นคลื่นส่วนแรกของเวนตริเคิลที่ถูกกระตุ้น มีความลึกไม่เกิน 1/2 ของคลื่น QRS และคลื่น Q มีความกว้างน้อยกว่า 0.04 วินาที

5) ST segment คือส่วนที่อยู่ระหว่างจุดที่สิ้นสุดของ QRS และจุดเริ่มต้นของคลื่น T เป็นช่วงระยะเวลาเมื่อถูกกระตุ้นสิ้นสุดลงและก่อนการกลับเข้าสู่สภาพปกติจะเริ่มขึ้น

6) คลื่น T จำเป็นต้องใช้พลังงาน ซึ่ง คลื่น T สามารถเปลี่ยนแปลงได้ง่าย ยกตัวอย่างเช่น ในภาวะกล้ามเนื้อหัวใจขาดเลือด หรือมีการเปลี่ยนแปลงของอิเล็กโทรไลต์ โดยปกติคลื่น T จะสูงน้อยกว่า 0.5 V ในขั้วต่อแขนขา และน้อยกว่า 1 V ในขั้วต่อทรวงอกหัวตั้งกลมเรียบใหญ่กว่าคลื่น P

7) QT interval ระยะระหว่างจุดเริ่มต้นของ QRS complex และจุดสิ้นสุดของคลื่น T จะวัดในขณะที่เวนตริเคิลซ้ายมีการบีบตัวความยาวประมาณ 0.35 ถึง 0.40 วินาที

8) คลื่น U เป็นคลื่นบวกเล็ก ๆ เกิดหลังคลื่น T โดยปกติจะไม่ค่อยพบ แต่ในบางครั้งก็พบจะมีส่วนที่สูงมากกว่า 1 มิลลิเมตร

8) RR interval เป็นช่วงระยะเวลาระหว่าง QRS complex ถึง QRS ถัดไป

2.1.2 ค่าความอิ่มตัวของออกซิเจนในเลือด

ค่าความอิ่มตัวของระดับออกซิเจนในเลือด หรือ SpO_2 เป็นค่าหนึ่งที่บอกประสิทธิภาพการทำงานของปอดว่าปกติหรือไม่ ปกติค่าออกซิเจนในกระแสเลือดจะมีค่ามากกว่า 95 เปอร์เซ็นต์ และหากค่าออกซิเจนในกระแสเลือดมีค่าต่ำกว่า 95 เปอร์เซ็นต์ สาเหตุอาจมาจากร่างกายมีค่าออกซิเจนไม่เพียงพอ สุดท้ายถ้าระดับออกซิเจนในกระแสเลือดมีค่าต่ำกว่า 90 เปอร์เซ็นต์ ซึ่งสาเหตุอาจเกิดจากการติดเชื้อที่ระบบทางเดินหายใจ ซึ่งเป็นความผิดปกติของร่างกาย การวัดระดับออกซิเจนในเลือดเป็นการช่วยแพทย์วินิจฉัยผู้ป่วยที่มีความเสี่ยงของภาวะหยุดหายใจขณะนอนหลับ และปัญหาอื่นๆ เบื้องต้นได้[3]

2.2 หลักการ Analog to Digital Converter

ADC ย่อมาจาก Analog to Digital Converter เป็นวงจรหรืออุปกรณ์ที่ใช้แปลงสัญญาณอนาล็อกให้ เป็นสัญญาณดิจิตอล โดยสัญญาณที่อ่านได้ออกมานั้นไม่ใช่ค่าแรงดัน แต่เป็นค่าระดับขั้นของแรงดันไฟฟ้าที่วัด ได้ ซึ่ง ADC ยิ่งมีค่าระดับขั้นที่วัดได้สูงมากเท่าไร ตัวเลขค่าสูงสุดที่วัดได้ยิ่งสูงเท่านั้น โดยการแปลงค่าจาก ADC เป็นค่าแรงดันไฟฟ้าค่าที่ต้องทราบก่อนจะแปลง มีดังนี้

1) ค่าความละเอียดของ ADC (Resolution) เป็น 8 บิต 10 บิต 12 บิต หรือ 16 บิต

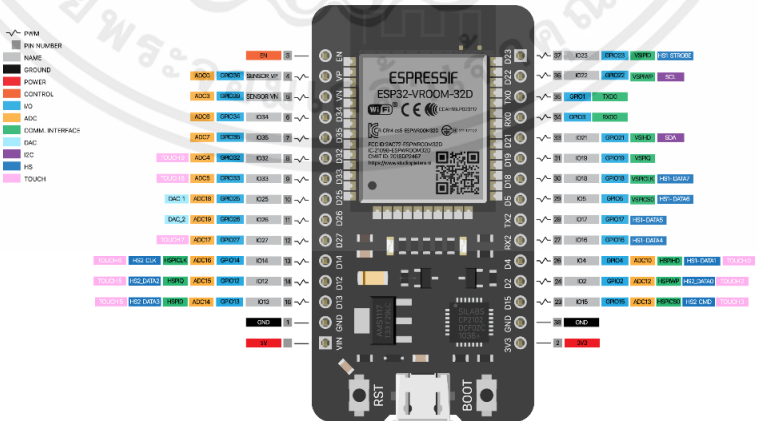
2) แรงดันอ้างอิง (Vref) แรงดันขา Vref

3) ค่าที่ได้จาก ADC

โดยหาแรงดันไฟฟ้าได้จากสูตร แรงดันไฟฟ้า = แรงดันอ้างอิง / (ค่าความละเอียด - 1) * ค่าที่ได้จาก ADC เช่น ใช้ ADC ค่าความละเอียด 12 บิต = 2 12= 4096 ใช้แรงดันอ้างอิง 3.3V และค่าที่ได้จาก ADC คือ 2047 สามารถแปลงเป็นแรงดันได้ 3.3 / (4096 - 1) * 2047 ≈ 1.65 V

2.3 ESP32

ESP32 จะมีทั้งหมด 38 ขา ดังรูปที่ 2.2 และดังตารางที่ 2.1 ซึ่งรองรับการเชื่อมต่อ Wi-Fi มาตรฐาน 802.11 b/g/n (HT40) และ Bluetooth ได้ โดยไม่ต้องซื้อโมดูลเพิ่มเติม บอร์ด ESP32 เองยังมีการทำงานที่แบ่งเป็น 2 Core และ Pin I/O เลือกการ ทำงานได้ในขาเดียวกัน เช่น การแปลง ADCโดย ESP32 มี MCU Dual Core แบบ 32 บิต ทำงานที่ ความเร็ว 160 - 240MHz SRAM 512kB Clock Speed 160MHz (Default), 240MHz โดยมีหน่วยความจำ แบบ Flash ใช้สำหรับอัปโหลดโปรแกรมขนาด 4M และมีค่าความละเอียดในการอ่านค่า ADC 12 Bit[4]



รูปที่ 2.2 ESP32 38 Pin[5]

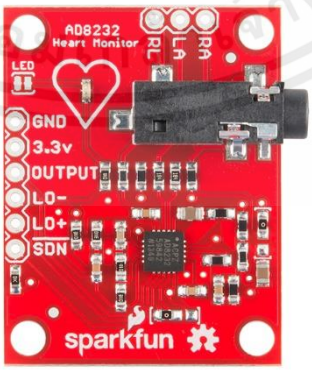
ตารางที่ 2.1 ขา GPIO ของ ESP32

ขา GPIO	จำนวน
Analog-to-Digital Converter (ADC) channels	18
SPI interfaces	3
UART interfaces	3
I2C interfaces	2
PWM output channels	16
Digital-to-Analog Converters (DAC)	2
I2S interfaces	2
Capacitive sensing GPIOs	10

2.4 Module AD8232

Module AD8232 จะเป็น Module ในรูปที่ 2.3 ใช้สำหรับวัดชีพจร การตรวจคลื่นไฟฟ้าหัวใจ เพื่อดูหัวใจเพื่อให้แพทย์ช่วยวิจัยความผิดปกติต่อ โรคหัวใจ การทำงาน Sensor จะทำการตรวจสอบสัญญาณไฟฟ้าของหัวใจ ในแต่ละจังหวะของการเต้นของหัวใจ สามารถทราบได้ว่าหัวใจมีการบีบตัวสมบูรณ์ในการส่งเลือดไปเลี้ยงส่วนต่างๆของร่างกาย โดยมีคุณสมบัติดังนี้

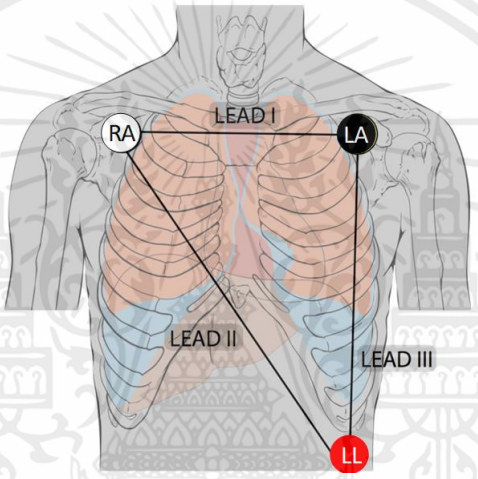
- รองรับแรงดัน DC 3.3V
- สัญญาณที่ได้เป็นแบบ Analog
- ขนาดของบอร์ดเล็ก น้ำหนักเบา



รูปที่ 2.3 AD8232[6]

โดยสัญญาณที่จะเป็นแบบ ระบบสายสื่อชนิด ๒ ขั้ว (Bipolar lead) เป็นการวัดความต่างศักย์ระหว่าง 2 ตำแหน่ง โดยให้ตำแหน่งหนึ่งเป็น ขั้วบวกและอีกตำแหน่งหนึ่งเป็นขั้วลบ ดังรูปที่ 2.4 Bipolar limb lead วัดความต่างศักย์ไฟฟ้าระหว่าง 2 จุด ของแขน/ ขา ดังนี้ [7]

1) lead I บันทึกความต่างศักย์ไฟฟ้าระหว่าง แขนขวา (-) กับแขนซ้าย (+)
2) lead II บันทึกความต่างศักย์ไฟฟ้าระหว่าง แขนขวา (-) กับขาซ้าย (+)
3) lead III บันทึกความต่างศักย์ไฟฟ้าระหว่าง แขนซ้าย (-) กับขาซ้าย (+)



รูปที่ 2.4 การวางคลื่นไฟฟ้าหัวใจ 3 Leads[8]

2.5 Module MAX30100

Module MAX30100 จะเป็น Module ในรูปที่ 2.5 คือเครื่องวัดความอิ่มตัวของออกซิเจนในกระแสเลือด และอัตราการเต้นของหัวใจ ประกอบด้วยไฟ LED สองดวง, เครื่องตรวจจับแสง, เลนส์ปรับแสงและตัวประมวลผลสัญญาณ Analog ที่มีสัญญาณรบกวนต่ำเพื่อตรวจจับออกซิเจนในกระแสเลือดและอัตราการเต้นของหัวใจ โดยปกติค่าออกซิเจนในกระแสเลือดจะมีค่ามากกว่า 95 เปอร์เซ็นต์ และหากค่าออกซิเจนในกระแสเลือดมีค่าต่ำกว่า 95 เปอร์เซ็นต์ สาเหตุอาจมาจากร่างกายมีค่าออกซิเจนไม่เพียงพอ สุดท้ายถ้าระดับออกซิเจนในกระแสเลือดมีค่าต่ำกว่า 90 เปอร์เซ็นต์ ซึ่งสาเหตุอาจเกิดจากการติดเชื้อที่ระบบทางเดินหายใจ ซึ่งเป็นความผิดปกติของร่างกาย [9]

รูปที่ 2.5 MAX30100

2.6 การส่งข้อมูลผ่าน Wi-Fi โดยใช้ ESP32

2.6.1 กระบวนการการส่งข้อมูลผ่าน Wi-Fi โดยใช้ ESP32

การทำงานกับ Wi-Fi มีอยู่ด้วยกัน 2 โหมด คือ AP (Access Point) และ STA (Station Accession Point) ในโหมด AP คือ การติดต่อโดยตรง ระหว่างกัน ในขณะที่ STA เป็น การติดต่อกับตัวปล่อยสัญญาณ สำหรับลูกข่ายอย่าง Router สำหรับ ESP32 หรือ M5Stack สามารถติดต่อได้ทั้ง AP, STA, และ AP พร้อมๆ กับ STA

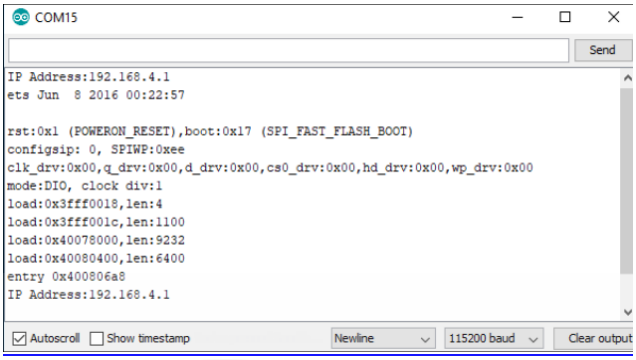
2.6.2 ไลบรารี่ที่ใช้ในการเชื่อมต่อ Wi-Fi

ไลบรารี Wi-Fi เป็นไฟล์สำคัญ ที่ใช้ในการติดต่อแบบ Wi-Fi มีฟังก์ชันที่สำคัญคือ: mode(WIFI_AP) ใช้เพื่อเลือกโหมด ใน สำหรับ AP ใช้ WIFI_AP และ softAP (Name,Password) ใช้สำหรับติดต่อในโหมด AP โดยใช้ ชื่อ และรหัส ซึ่งตัวอย่างการเชื่อมต่อ Wi-Fi และผลการเชื่อมต่อแสดงดังรูปที่ 2.6 และ 2.7 ตามลำดับ

ตัวอย่าง 1 การต่อเชื่อม WiFi ในโหมด AP

```
#include <WiFi.h>
#include <WiFiAP.h>
const char* WIFI_AP_NAME = "ESP32 AP";
const char* WIFI_AP_PASS = "12345678";
void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_AP);
  WiFi.softAP(WIFI_AP_NAME, WIFI_AP_PASS);
  Serial.print("IP Address:");
  Serial.println(WiFi.softAPIP());
}
void loop() {
  // put your main code here, to run repeatedly:
}
```

รูปที่ 2.6 ตัวอย่างการเชื่อมต่อ Wi-Fi ในโหมด AP[10]



รูปที่ 2.7 ตัวอย่างผลการเชื่อมต่อ Wi-Fi ในโหมด AP[10]

สำหรับโหมด STA มีการติดต่อผ่านการใช้ฟังก์ชัน begin(name, password) เมื่อเชื่อมต่อได้จะมีสถานะ โดยใช้ฟังชัน status( ) โดยเทียบกับ สถานะ WL_CONNECTED ซึ่งมีตัวอย่างการเชื่อมต่อและผลการเชื่อมต่อ แสดงจากรูปที่ 2.8 และ 2.9 ตามลำดับ

```
#include <WiFi.h>

const char* WIFI_STA_NAME = "OPPO A3s";
const char* WIFI_STA_PASS = "12345678";
void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_STA_NAME, WIFI_STA_PASS);
  while(WiFi.status() != WL_CONNECTED){
    delay(500);
    Serial.print(".");
  }
  Serial.println();
  Serial.println("WiFi Connected!");
  Serial.print("IP:");
  Serial.println(WiFi.localIP());//Router gives IP.
}
void loop() {
  // put your main code here, to run repeatedly:
}
```

รูปที่ 2.8 ตัวอย่างการเชื่อมต่อ Wi-Fi ในโหมด STA[10]

รูปที่ 2.9 ตัวอย่างผลการเชื่อมต่อ Wi-Fi ในโหมด STA[10]

2.7 โปรโตคอล MQTT

MQTT ย่อมาจาก Message Queuing Telemetry Transport เป็น protocol ที่ใช้ในส่งข้อมูล โดย MQTT พัฒนามาเพื่อใช้ในกับอุปกรณ์ IoT ซึ่งจะมี Server icon และ Explorer ดังรูปที่ 2.10 โดยทำงานในรูปแบบ Broker และ Clients Network ซึ่งในการใช้งาน ได้มีการออกแบบให้สามารถส่งข้อมูลแบบ Real- Time ทำให้อุปกรณ์ใช้พลังงานต่ำ MQTT ถูกพัฒนามาจากโปรโตคอล TCP/IP ที่ส่งข้อมูลแบบ 1: 1 ทำให้สิ้นเปลืองทรัพยากร จึงไม่เหมาะกับอุปกรณ์ IoT เพราะอุปกรณ์มีการส่งข้อมูลตลอดเวลา และหนึ่งอุปกรณ์อาจทำการรับหรือส่งข้อมูลไปยังหลายอุปกรณ์ หรือการส่งข้อมูลแบบ One-To-All โดยอุปกรณ์ทุกตัวที่ทำการ Subscriber ไปยัง Topic ใดๆ บน Broker จะได้รับข้อมูลที่ Publisher เป็นผู้ส่งให้ Topic นั้นๆบน Broker ทั้งหมด โดย MQTT มีเป้าหมายคือเป็นโปรโตคอลที่มีประสิทธิภาพสูง ส่งข้อมูลขนาดเล็ก ใช้พลังงานต่ำ [11]

รูปที่ 2.10 MQTT Server icon(Mosquitto) และ MQTT explorer

2.7.1 องค์ประกอบของ MQTT

2.7.1.1 Broker เป็น Server ที่เป็นตัวกลางในการรับข้อมูลจาก Publisher และส่งข้อมูลไปยัง Subscriber

2.7.1.2 Clients (Subscriber / Publisher)

1) Publisher เป็นตัวที่ส่งข้อมูลให้กับ Topic ที่อยู่ใน Broker ทำการ Publish

2) Subscriber เป็นตัวที่รับข้อมูลจาก Topic ที่อยู่ใน Broker ทำการ Subscribe

2.6.1.3 Topic เป็นหัวเรื่องที่เราเลือกในการรับส่งข้อมูลระหว่าง Publisher กับ Subscriber

2.7.2 หลักการทำงานของ MQTT

โปรโตคอล MQTT แบ่งบทบาทของอุปกรณ์ใน Network เป็น 2 ประเภท ได้แก่ Message Broker และ Client[12]

2.7.2.1 MQTT Broker เป็นเซิร์ฟเวอร์ซึ่งทำหน้าที่รับ ข้อความที่ส่งจาก Client ตัวหนึ่งและส่งไปยัง Client อีกตัวหนึ่ง โดย MQTT Broker สามารถทำงานได้บนหลาย Platform เช่น Window, MAC OS, Linux และ Ubuntu โดยที่เราสามารถสร้าง Broker ขึ้นใช้งานเองได้ เนื่องจากเป็น Open Source

2.7.2.2 MQTT Client เป็นอุปกรณ์ที่ทำหน้าที่ส่งข้อความไปยัง Broker และยังสามารถรับข้อความจาก Broker ได้ โดย Client อาจจะเป็นอุปกรณ์ประเภท เช่น Micro controller, Arduino, โทรศัพท์มือถือ รวมไปถึงคอมพิวเตอร์ด้วย ซึ่งจะมี Library สำหรับรัน MQTT Client ด้วยโมเดลการเชื่อมต่อแบบ Publish และ Subscribe ทำให้การวาง Network สำหรับ MQTT ไม่มีความจำเป็นต้องให้อุปกรณ์ทุกตัวอยู่บน Network ในวงเดียวกัน แต่ทำให้ MQTT Broker อยู่บน IP Address ที่เป็น Public เพียงตัวเดียว ส่วน Client ตัวอื่น ไม่จำเป็นต้องเป็น Public IP แต่แค่สามารถเข้าถึง Internet ได้ ดังนั้น Client 1 ก็จะสามารถส่งข้อความหา Client 2 ได้โดยที่ Client 1 ไม่ต้องเชื่อมต่อกับ Client 2 โดยตรง แต่ส่งหา Broker ที่ทำหน้าที่เป็นตัวกลาง ในการรับและส่งข้อมูลแทน

2.7.3 การทำงานของ MQTT Client

MQTT Client จะมีคำสั่ง ในการส่งไปยัง Broker ทั้งหมด 4 Command แสดงจากรูปที่ 2.11

Command	Description
Connect	เป็นการร้องขอการเชื่อมต่อไปยัง Broker ในขั้นตอนนี้ Client จะต้องระบุ Address ของ Broker, Port ที่ใช้ ซึ่งโดยปกติจะใช้ Port 1883 เป็น Default รวมถึง User/Password หรือ Certificate ในการเชื่อมต่อ (ถ้ามี)
Disconnect	เป็นการร้องขอให้ตัดการเชื่อมต่อจาก Broker
Publish	เป็นการส่งข้อความจาก Client ขึ้นไปที่ Broker โดยการกำหนดตัวแปร บน Broker จะใช้สิ่งที่เรียกว่า Topic เป็นตัวกำหนด คล้ายกับ Tag ใน PLC เช่น Topic : temperature/floor temperature/roof temperature/room ในที่นี้ temperature จะมีลักษณะคล้ายกับ categories name หรือ folder ในการเก็บ tag ต่างๆ
Subscribe	Client จะสามารถ Subscribe Topic ที่อยู่บน Broker ได้ และเมื่อมี Client อื่นทำการ publish ค่าใหม่ไปที่ Topic ที่กำลัง Subscribe อยู่ เมื่อนั้นตัว Client จะได้รับข้อความใหม่ทันทีโดยไม่ต้องร้องขอ เช่น ClientA subscribe Topic: temperature/floor อยู่ จากนั้น ClientB ทำการ Publish ค่าอุณหภูมิล่าสุดมาที่ Topic นี้ ClientA จะได้รับค่าอุณหภูมิล่าสุดทันที
Unsubscribe	คือคำสั่งยกเลิกการ Subscribe จาก Topic ใด Topic หนึ่ง โดยที่สถานะยัง Connect กับ Broker อยู่

รูปที่ 2.11 Command ในการส่งข้อมูลไปยัง Broker[12]

2.7.4 คุณภาพข้อมูล (QoS)

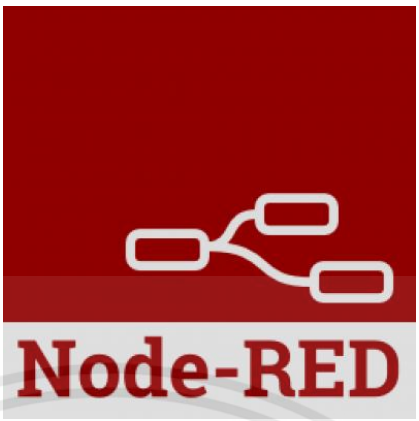
2.6.4.1 QoS0 – เป็นการส่งแบบ ส่งแล้วส่งเลยไม่มีการรอการตอบรับจากจากผู้รับ เรียกว่าส่งอย่างมาก 1 ครั้ง (at most once)

2.7.4.2 QoS1 – เป็นการส่งแบบรอการตอบรับจากผู้รับ ถ้าไม่ได้รับการตอบรับ จะรอระยะเวลาหนึ่ง และทำการส่งข้อความนั้นใหม่ เรียกว่า ส่งอย่างน้อย 1 ครั้ง (at least once)

2.7.4.3 QoS2 – เป็นการส่งแบบผู้ส่ง รอการตอบรับจากผู้รับ เรียกว่าส่งเพียง 1 ครั้ง(exactly once)

2.8 Node-RED

Node red เป็นแอปพลิเคชันตัวหนึ่งที่สามารถนำไปประยุกต์ใช้งานกับอุปกรณ์ IoT เนื่องจากเป็น Open source จึงทำให้มีผู้พัฒนาโปรแกรมมากมาย รวมถึงตัวอย่างโปรแกรมที่สามารถส่งต่อและศึกษาได้อย่างง่าย โดยมี icon ดังรูปที่ 2.12



รูปที่ 2.12 Node-Red icon[13]

Node-RED จัดการเหตุการณ์ขึ้นอยู่กับ Node.js แอปพลิเคชัน ทำงานเป็นเว็บเซิร์ฟเวอร์และผู้ใช้สามารถปรับแต่งและจัดการการเชื่อมต่อระหว่างฮาร์ดแวร์ต่างๆ และสร้างขั้นตอนการทำงานเบราว์เซอร์ของคอมพิวเตอร์ ไม่มีซอฟต์แวร์ที่มีราคาแพง แต่เป็นเรื่องง่ายและทำงานในเว็บเบราว์เซอร์ แอปพลิเคชันเซิร์ฟเวอร์นี้ยังมีความสามารถในการทำงานบนอุปกรณ์ ยกตัวอย่างเช่น Raspberry Pi, ESP32 อินเทอร์เฟซผู้ใช้ของ Node-RED ดูเรียบง่ายทำให้แทบไม่มีปัญหาในการพัฒนาโครงการ IoT ด้วย Node-RED ใน GitHub แสดงใน JSON (JavaScript Object Notation) และสามารถส่งออกไปยังคลิปบอร์ดได้อย่างง่ายดายหรือสามารถนำเข้าสู่ Node-RED หรือแชร์ทางออนไลน์

2.9 Database

Database หรือ ฐานข้อมูล เป็นการรวบรวมข้อมูล โดยมีความสัมพันธ์กัน โดยไม่ได้จำเป็นว่าข้อมูลทั้งหมดจะต้องเก็บไว้ในหมวด Files เดียวกันหรือแยกเก็บหลายๆหมวด Files ซึ่งถูกจัดเก็บอย่างเป็นระบบ โดยจะมีซอฟต์แวร์เข้ามาควบคุมกระบวนการใช้งาน การทำงาน รวมถึงการประมวลผล ทำให้ผู้ใช้สามารถจัดการข้อมูลได้อย่างมีประสิทธิภาพ โดย Database MySQL สามารถใช้งานได้กับหลายภาษา เช่น C, C++, Python, Java

โดยฐานข้อมูลที่มีประสิทธิภาพ ต้องมีคุณสมบัติ ดังต่อไปนี้

1) Database ต้องมีประสิทธิภาพการทำงานสูง เพื่อสามารถรองรับงานได้หลายรูปแบบภายในระบบเดียว

2) ความปลอดภัย ซึ่งจะสามารถปกป้องข้อมูลที่สำคัญได้ตลอดเวลา

3) มีความมั่นคงและมีเสถียรภาพ มี Downtime ที่ต่ำ ซึ่งจะลดโอกาสที่ระบบจะหยุดทำงาน และต้องมีการอัปเกรดระบบทั้งในระดับของ Software และ Hardware ตลอดเวลา

4) ต้องบริหารจัดการและบำรุงรักษาได้ง่าย เพื่อช่วยลดระยะเวลาในการทำงานลง

5) การจัดเก็บข้อมูลได้อย่างมีประสิทธิภาพ จะสามารถลดหรือเพิ่มขยายได้ทั้งในระยะสั้นและระยะยาว

2.9.1 ประโยชน์ของฐานข้อมูล

1) ลดการเก็บข้อมูลที่ซ้ำซ้อนกัน ข้อมูลบางชุดที่อยู่ในรูปของหมวด Files อาจมีแสดงอยู่หลายที่ เพราะมีผู้ใช้ข้อมูลชุดนี้หลายคน เมื่อใช้ระบบฐานข้อมูลแล้วจะช่วยให้ความซ้ำซ้อนของข้อมูลลดน้อยลง

2) รักษาความถูกต้องของข้อมูล เนื่องจากฐานข้อมูลมีเพียงฐานข้อมูลเดียว ในกรณีที่มีข้อมูลชุดเดียวกันแสดงอยู่หลายที่ในฐานข้อมูล ซึ่งข้อมูลเหล่านี้ต้องตรงกัน หากมีการแก้ไขข้อมูลนี้ทุกแห่งที่ข้อมูลแสดงอยู่ จะแก้ไขให้ถูกต้องเหมือนกันโดยอัตโนมัติด้วยระบบจัดการฐานข้อมูลที่มีประสิทธิภาพ

3) การรักษาความปลอดภัยให้กับชุดข้อมูลทำได้อย่างสะดวก ซึ่งในส่วนของการเข้าถึงข้อมูลนี้ควรให้เฉพาะผู้ที่เกี่ยวข้องเข้าถึงเท่านั้น ซึ่งจะทำให้เกิดความปลอดภัยของข้อมูลมากยิ่งขึ้นด้วย

2.9.2 ระบบฐานข้อมูล (Database System)

ระบบฐานข้อมูล คือ ระบบที่มีการรวบรวมข้อมูลต่าง ๆ ที่เกี่ยวข้องกันเข้าไว้ด้วยกันอย่างเป็นระบบมีความสัมพันธ์ระหว่างข้อมูลต่าง ๆ ที่ชัดเจน ภายในระบบฐานข้อมูลประกอบด้วยแฟ้มข้อมูลหลายแฟ้มที่มีข้อมูล ที่มีความเกี่ยวข้องกันเข้าไว้ด้วยกันอย่างเป็นระบบและมีการเปิดโอกาสให้ผู้ใช้สามารถใช้งานและดูแลรักษาป้องกันข้อมูลเหล่านี้ ได้อย่างมีประสิทธิภาพ โดยมีซอฟต์แวร์ที่เปรียบเสมือนสื่อกลางระหว่างผู้ใช้และโปรแกรมต่าง ๆ ที่เกี่ยวข้องกับการใช้ฐานข้อมูล เรียกว่า ระบบจัดการฐานข้อมูล หรือ DBMS (data base management system)มีหน้าที่ช่วยให้ผู้ใช้เข้าถึงข้อมูลได้ง่ายสะดวกและมีประสิทธิภาพ การเข้าถึงข้อมูลของผู้ใช้อาจเป็นการสร้างฐานข้อมูล การแก้ไขฐานข้อมูล หรือการตั้งคำถามเพื่อให้ได้ข้อมูลมา โดยผู้ใช้ไม่จำเป็นต้องรับรู้เกี่ยวกับรายละเอียดภายในโครงสร้างของฐานข้อมูล [14]

2.10 หลักการทำงาน MySQL

รูปที่ 2.13 โลโก้ MySQL[15]

MySQL เป็นระบบจัดการฐานข้อมูลแบบ Database Management System (DBMS) ดังรูปที่ 2.13 ซึ่งฐานข้อมูลมีโครงสร้างของการเก็บรวบรวมข้อมูล ในการที่จะเพิ่มเติมหรือประมวลผลข้อมูลที่เก็บในฐานข้อมูลจำเป็นต้องอาศัยระบบจัดการฐานข้อมูล ซึ่งจะทำหน้าที่เป็นตัวกลางในการจัดการกับข้อมูลในฐานข้อมูลทั้งสำหรับการใช้งานเฉพาะ และรองรับการทำงานของแอพลิเคชันอื่นๆที่ต้องการ การใช้งานข้อมูลในฐานข้อมูล เพื่อให้ได้รับความสะดวกในการจัดการกับข้อมูลจำนวนมาก MySQL ทำหน้าที่เป็นทั้งตัวฐานข้อมูลและระบบจัดการฐานข้อมูล

MySQL เป็นระบบจัดการฐานข้อมูลแบบ Relational Database Management System ซึ่งจะทำการเก็บข้อมูลทั้งหมดในรูปแบบของตารางแทนการเก็บข้อมูลทั้งหมดลงใน File เพียง File เดียว ทำให้สามารถทำงานได้อย่างรวดเร็วและมีความยืดหยุ่น และแต่ละตารางที่เก็บข้อมูลสามารถเชื่อมโยงเข้าหากันทำให้สามารถจัดกลุ่มข้อมูลได้ตามต้องการ

MySQL ถูกออกแบบการทำงานในลักษณะของ Client และ Server ประกอบด้วยส่วนหลัก 2 ส่วน คือส่วนของผู้ให้บริการ (Server) และส่วนของผู้ใช้บริการ (Client) โดยในแต่ละส่วนก็จะมีโปรแกรมสำหรับการทำงานตามหน้าที่ของตน[16]

2.11 Web Application

Web Application คือ การออกแบบซอฟท์แวร์ประยุกต์ที่ทำงานบนเว็บผ่านเครือข่ายอินเตอร์เน็ต

2.11.1 หลักการทำงานของ Web Application

โครงสร้าง Web Application ที่ใช้ในปัจจุบันเป็นโครงสร้างแบบ Client-Server Model ซึ่ง Client-Server Model เป็นโครงสร้างที่มีการแยกส่วนฝั่งผู้ให้บริการหรือทรัพยากร (Service

Provider or Resource Provider) ซึ่งเรียกว่าเป็น ฝั่ง Server และฝั่งผู้ขอใช้บริการหรือทรัพยากร (Service Requester or Resource Requester) ซึ่งเรียกว่าเป็น ฝั่ง Client โดยฝั่ง Client จะเป็นฝ่ายเริ่มต้นการสื่อสาร และ Server จะอยู่ในสถานะรอการเชื่อมต่อ

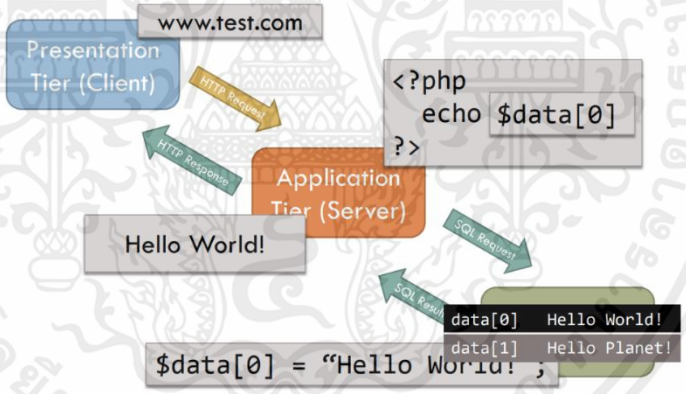
2.11.2 Client-Server Architecture

สถาปัตยกรรมแบบ Three-tier ดังรูปที่ 2.14 เป็นสถาปัตยกรรมที่มีการแยกการแสดงผล การประมวลข้อมูล และการจัดการข้อมูลออกจากกันเป็น 3 ส่วน ดังนี้

1) Presentation Tier แสดงผลลัพธ์ของการประมวลผล ซึ่งจะเกิดขึ้นที่ ฝั่ง Client

2) Application Tier ทำหน้าที่ประมวลผลข้อมูล และควบคุมการไหลของข้อมูลระหว่าง Web Browser ของ Client กับฐานข้อมูล

3) Data Storage Tier เป็น Tier ที่จัดเก็บข้อมูลและดูแลข้อมูลทั้งหมด



รูปที่ 2.14 Three-tier Diagram[17]

2.11.3 การเขียน Web Application

ในการเขียน Web Application สามารถเขียนหลายภาษาให้อยู่ในไฟล์เดียวกันได้ โดยแต่ละภาษาจะถูกประมวลผลใน Tier ที่แตกต่างกัน จึงจำเป็นต้องใช้ภาษาให้ถูกต้อง โดยภาษา HTML, JavaScript, CSS เป็นภาษาที่ถูกประมวลผลที่ Presentation Tier โดยภาษาที่กล่าวมาควบคุมการแสดงผลในหน้าเว็บ ซึ่งภาษา PHP เป็นภาษาที่ถูกประมวลผลที่ Application Tier ภาษา PHP จะควบคุมการไหลของข้อมูลระหว่าง Client และ Data Storage รวมถึงการคำนวณต่าง ๆ และ MySQL เป็นภาษาที่ประมวลผลที่ Data Storage Tier ภาษานี้จะบันทึก แก้ไข ลบ และ เลือกข้อมูลเข้าและออกจากฐานข้อมูล[17]

2.12 Xampp

รูปที่ 2.15 โลโก้ Xampp[18]

Xampp เป็นโปรแกรม Apache web server ที่ไว้ใช้จำลอง web server เพื่อทดสอบสคริปต์หรือหน้าเว็บในเครื่อง ซึ่งเป็นโปรแกรมง่ายต่อการติดตั้งและใช้งาน ไม่เสียค่าใช้จ่าย โดยในโปรแกรม Xampp มาพร้อมกับภาษา PHP ซึ่งเป็นภาษาสำหรับพัฒนาเว็บแอปพลิเคชัน, MySQL, Perl, OpenSSL , phpMyAdmin จะสนับสนุนฐานข้อมูล MySQL และ SQLite โดยโปรแกรม Xampp จะอยู่ในรูปแบบของไฟล์ Zip, tar หรือ exe โปรแกรม Xampp อยู่ภายใต้ใบอนุญาตของ GNU General Public License [19]

2.12.1 ข้อดีของ XAMPP

1) สามารถติดตั้งเว็บเซิร์ฟเวอร์ได้ง่าย อีกทั้งยังประหยัดเวลาเนื่องจากไม่จำเป็นต้องติดตั้งและตั้งค่าโปรแกรมแต่ละตัวด้วยตนเอง

2) สามารถดาวน์โหลดมาติดตั้ง และใช้งานได้ฟรี

3) รองรับการทำงานบนระบบปฏิบัติการต่างๆ ที่หลากหลาย

4) โปรแกรมทำงานได้ดี รองรับการสร้างเว็บเซิร์ฟเวอร์ได้อย่างมีประสิทธิภาพ

5) โปรแกรมได้รับความนิยมใช้งานอย่างแพร่หลาย

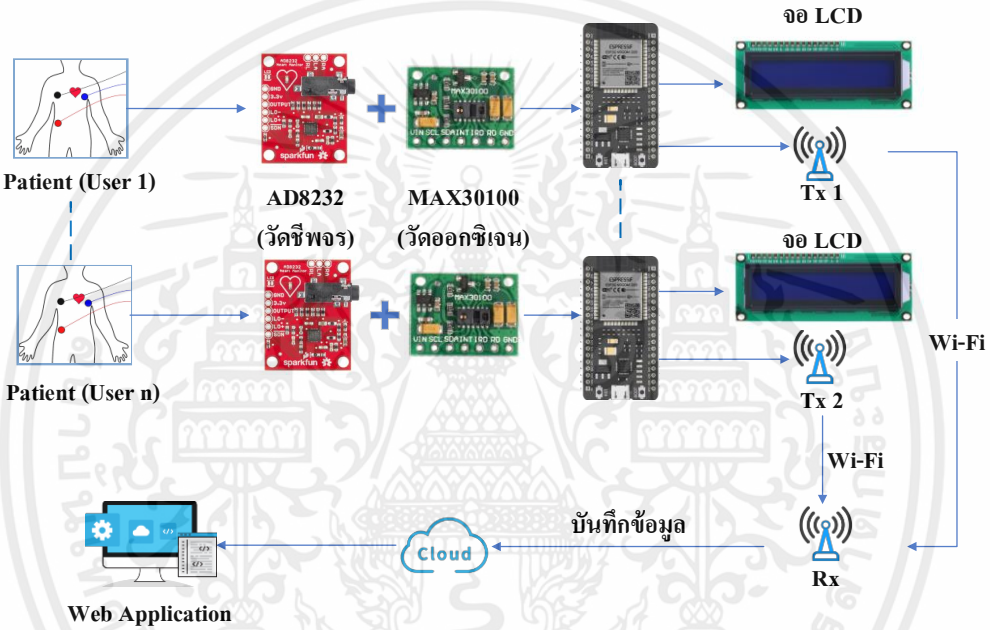
6) มีบทความ เอกสาร คู่มือ วิธีการปรับแต่ง และแก้ไขปัญหา เยอะ

7) โปรแกรมมีเวอร์ชั่นต่างๆ ให้เลือกตามความเหมาะสมในการใช้งาน

บทที่ 3
การออกแบบและการจัดทำปริญญานิพนธ์

3.1 การออกแบบ

3.1.1 การออกแบบภาพรวม



รูปที่ 3.1 บล็อกไดอะแกรม

เมื่อทำการวัดสัญญาณชีพจร(โมดูล AD8232) และวัดปริมาณออกซิเจนในเลือด (โมดูล MAX30100) จากร่างกายของผู้ป่วยมาแล้ว ค่าที่วัดได้จะถูกแปลงจากอนาล็อกเป็นดิจิตอลโดยผ่านไมโครคอนโทรลเลอร์ ESP32 Wi-Fi ซึ่งเราสามารถดูการเชื่อมต่อ Wi-Fi บนจอ LCD ได้ มีการส่งข้อมูลที่เป็นดิจิตอลผ่าน Wi-Fi เพื่อส่งไปยัง cloud เพื่อเก็บข้อมูลที่รับมาไว้ในฐานข้อมูล และถูกนำไปใช้โดย web application ซึ่งหน้าเว็บจะแสดงข้อมูลผู้ป่วย สัญญาณชีพจร ปริมาณออกซิเจนในเลือด และสามารถย้อนดูสัญญาณชีพจรหรือปริมารออกซิเจนในเลือดได้ โดยที่แพทย์สามารถติดตามอาการของผู้ป่วยได้จากทุกที่และทุกเวลาผ่าน Web application ดังรูปที่ 3.1

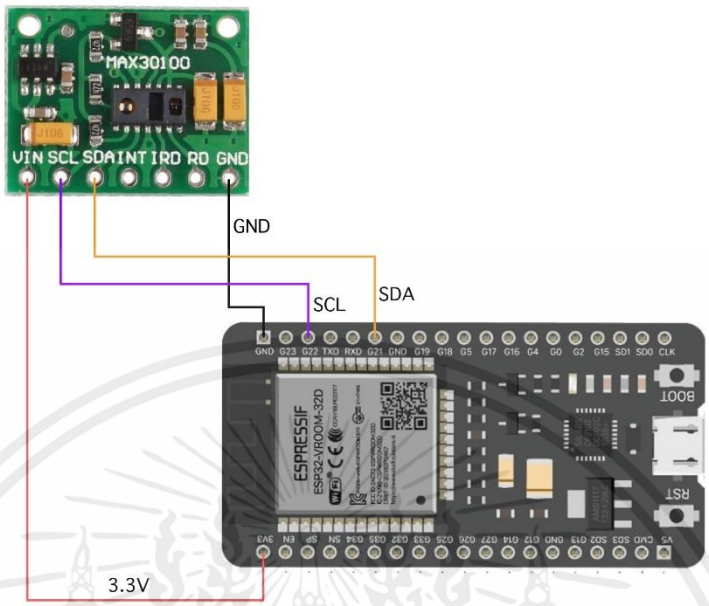
3.1.2 สร้างการเชื่อมต่อระหว่าง MAX30100 กับ ESP32

ทำการออกแบบให้ ESP32 ทำงานร่วมกับ MAX30100 เพื่อให้ตัว sensor บน MAX30100 ทำการตรวจจับค่าออกซิเจนในเลือด และอัตราการเต้นของหัวใจ (Heart Rate) แล้วดูค่าที่ได้บน serial monitor โดยจะมีการเขียนโปรแกรมดังรูปที่ 3.2 จากนั้นเราจะนำค่าที่ได้มาตรวจสอบความแม่นยำของ MAX30100 เมื่อเทียบกับผลจากเครื่อง oximeter ซึ่งมีวงจรการเชื่อมต่อดังรูปที่ 3.3 และตารางที่ 3.1

max30100new §

```
#include <Wire.h>
#include "MAX30100_PulseOximeter.h"
#define REPORTING_PERIOD_MS   2000

PulseOximeter pox;
uint32_t tsLastReport = 0;
void onBeatDetected()
{
    Serial.println("Beat!");
}
void setup()
{
    Serial.begin(115200);
    Serial.print("Initializing pulse oximeter..");
    if (!pox.begin()) {
        Serial.println("FAILED");
        for(;;);
    } else {
        Serial.println("SUCCESS");
    }
     pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);
    pox.setOnBeatDetectedCallback(onBeatDetected);
}
void loop()
{
    pox.update();
    if (millis() - tsLastReport > REPORTING_PERIOD_MS) {
        Serial.print("Heart rate:");
        Serial.print(pox.getHeartRate());
        Serial.print("bpm / SpO2:");
        Serial.print(pox.getSpO2());
        Serial.println("%");
        tsLastReport = millis();
    }
}
```

รูปที่ 3.2 code ที่ใช้ในการวัดค่าออกซิเจนในกระแสเลือด และอัตราการเต้นของหัวใจจาก MAX30100



รูปที่ 3.3 วงจรการเชื่อมต่อระหว่าง MAX30100 กับ ESP32

ตารางที่ 3.1 การเชื่อมต่อวงจร MAX30100 กับ ESP32

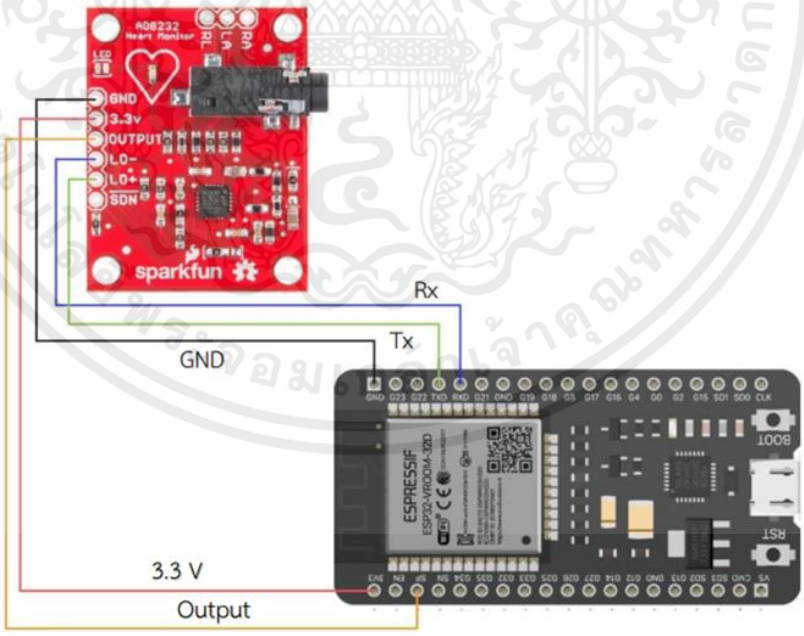
ESP32	MAX30100
GND	GND
3.3V	VIN
G21	SDA
G22	SCL

3.1.3 สร้างการเชื่อมต่อระหว่าง AD8232 กับ ESP32

ทำการออกแบบให้ ESP32 ทำงานร่วมกับ AD8232 เพื่อทำการวัดสัญญาณคลื่นหัวใจ แล้วดูสัญญาณที่ได้บน serial monitor โดยจะมีการเขียนโปรแกรมจากรูปที่ 3.4 และมีวงจรการเชื่อมต่อดังรูปที่ 3.5 และตารางที่ 3.2

```
esp32_bt_ECG§
#define LED_BUILTIN 2
float voltage_value = 0;
void setup() {

  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(9600);
  Serial.println(); // blank line in serial ...
  pinMode(41, INPUT); // Setup for leads off detection LO +
  pinMode(40, INPUT); // Setup for leads off detection LO -
}
void loop() {
  if((digitalRead(40) == 1)||(digitalRead(41) == 1)){
    Serial.println('!');
  }
  else{
    // send the value of analog input 0 to serial:

    voltage_value = (analogRead(A0)/4095.0)*3.3;
    Serial.print("Voltage ");
    Serial.println(voltage_value);
    delay(1);
}
}
```

รูปที่ 3.4 code ที่ใช้ในการวัดสัญญาณคลื่นหัวใจจาก AD8232



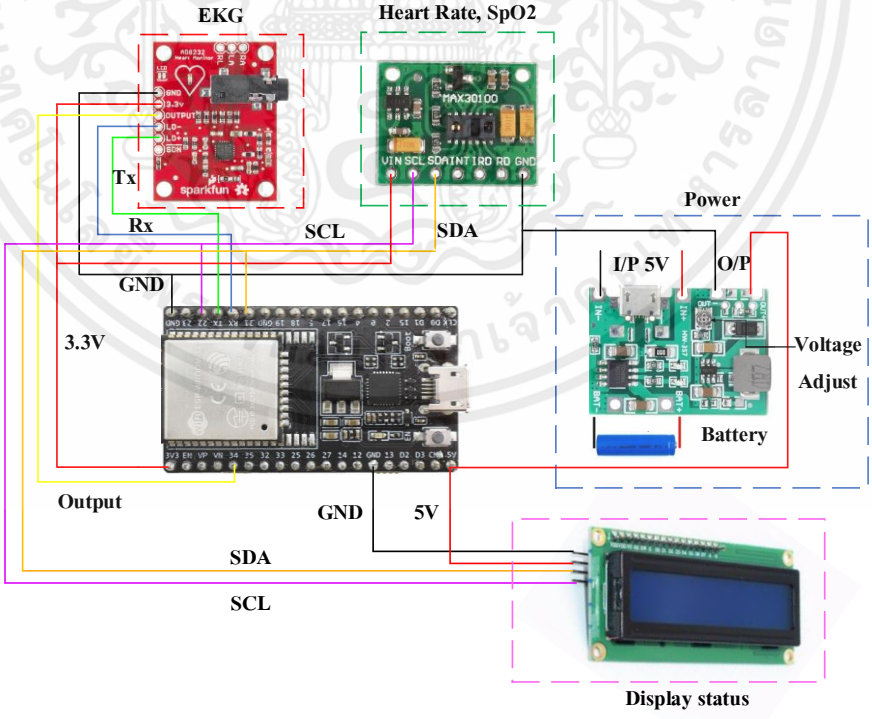
รูปที่ 3.5 วงจรการเชื่อมต่อระหว่าง AD8232 กับ ESP32

ตารางที่ 3.2 การเชื่อมต่อวงจรระหว่าง AD8232 กับ ESP32

ESP32	AD8232
GND	GND
3.3V	3.3V
SP	OUTPUT
TXD	LO-
RXD	LO+

3.1.4 ออกแบบวงจรรวมทั้งหมดของการวัดสัญญาณคลื่นหัวใจ, ค่าออกซิเจนในเลือด และค่าอัตราการเต้นของหัวใจ

ทำการออกแบบให้ ESP32 ทำงานร่วมกับ AD8232 และ MAX30100 เพื่อทำการวัดสัญญาณคลื่นหัวใจ, ค่าออกซิเจนในกระแสเลือด และค่าอัตราการเต้นของหัวใจ พร้อมกัน โดยมีการเพิ่มแบตเตอรี่(battery) ผ่านโมดูล AC01 ซึ่งสามารถจ่ายกระแสไฟให้แก่ตัว ESP32 อีกทั้งยังใช้ในการชาร์จไฟให้แก่แบตเตอรี่ได้ผ่านหัว micro ดังรูปที่ 3.6 ทำให้ ESP32 มีการทำงานอย่างเต็มประสิทธิภาพมากขึ้น



รูปที่ 3.6 วงจรรวมในการวัดสัญญาณคลื่นหัวใจ, ค่าออกซิเจนในเลือด
และค่าอัตราการเต้นของหัวใจ

3.2 เครื่องมือที่ใช้ในการทดลอง

ในปริญญานิพนธ์นี้ มีอุปกรณ์และเครื่องมือที่ใช้ในการทดลองดังนี้

3.2.1 กล่องวงจรรวมในการวัดสัญญาณคลื่นหัวใจ, ค่าออกซิเจนในเลือด และค่าอัตราการเต้นของหัวใจ

วงจรรวมในการการวัดสัญญาณคลื่นหัวใจ, ค่าออกซิเจนในกระแสเลือด และค่าอัตราการเต้นของหัวใจ เมื่อนำมาออกแบบและใส่ลงในกล่องอเนกประสงค์ จะได้วงจรภายในกล่องเป็นดังรูปที่ 3.7 นอกจากนี้ยังได้มีการจัดทำตัวอุปกรณ์เพิ่มอีก 2 ตัว ดังรูปที่ 3.8 เพื่อให้ใช้ได้กับผู้ป่วยทีละหลายๆคน

รูปที่ 3.7 วงจรรวมภายในกล่องที่จะใช้ในการวัดสัญญาณคลื่นหัวใจ, ค่าออกซิเจนในเลือด และค่าอัตราการเต้นของหัวใจ

รูปที่ 3.8 ตัวอุปกรณ์ที่ใช้งานจริงทั้ง 3 ตัว

3.2.2 เครื่องวัดออกซิเจนที่ปลายนิ้ว (Fingertip Pulse Oximeter)

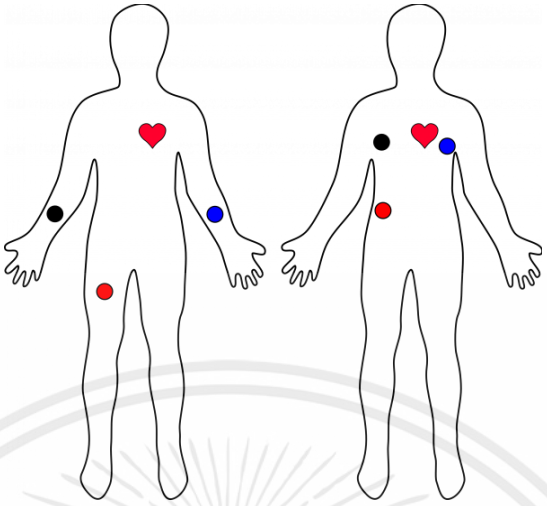
อุปกรณ์ที่ใช้วัดค่าออกซิเจนที่นิ้ว คืออุปกรณ์ที่วัดความเข้มข้นของออกซิเจนในกระแสเลือด (Oxygen Saturation) และสามารถใช้ในการวัดอัตราการเต้นของหัวใจได้ โดยที่อุปกรณ์มีลักษณะคล้ายเครื่องเย็บกระดาษดังรูปที่ 3.9 จะใช้งานโดยการหนีบที่บริเวณปลายนิ้ว จากนั้นจะตรวจระดับออกซิเจนด้วยการปล่อยคลื่นแสงจากอุปกรณ์ที่อยู่ข้างบนบริเวณเล็บ ผ่านนิ้วลงมาที่ตัวรับแสงที่อยู่อีกด้านบริเวณปลายนิ้วของผู้ใช้ เพื่อตรวจวัดฮีโมโกลบินที่จับตัวอยู่กับออกซิเจนในกระแสเลือด (Oxyhemoglobin) จากนั้นจึงนำมาคำนวณหาค่าความเข้มข้นของออกซิเจนภายในเลือดออกเป็นเปอร์เซ็นต์ แล้วแสดงผลผ่านหน้าจอของอุปกรณ์

รูปที่ 3.9 เครื่องวัดออกซิเจนปลายนิ้ว

3.3 การจัดเก็บผลการทดลอง

3.3.1 การทดสอบสัญญาณคลื่นหัวใจ

ทำการทดสอบเพื่อดูสัญญาณคลื่นหัวใจ ที่วัดได้จาก AD8232 โดยเราจะใช้สายที่เชื่อมต่อกับ AD8232 ต่อเข้ากับแผ่น electrode แล้วนำมาแปะยัง 3 ตำแหน่งบนร่างกายดังรูปที่ 3.10



รูปที่ 3.10 ตำแหน่งในการติดแผ่น electrode บนร่างกายทั้ง 3 ตำแหน่ง

3.3.2 การทดสอบค่าออกซิเจนในเลือด และค่าอัตราการเต้นของหัวใจ

ทำการทดสอบเพื่อดูค่าออกซิเจนภายในเลือด และค่าอัตราการเต้นของหัวใจ ที่วัดได้จาก MAX30100 โดยเราใส่ MAX30100 ลงในอุปกรณ์ที่สามารถหนีบนิ้วได้เหมือนกับ Fingertip Pulse Oximeter เพื่อให้ได้ค่าที่เราต้องการได้อย่างแม่นยำมากขึ้น

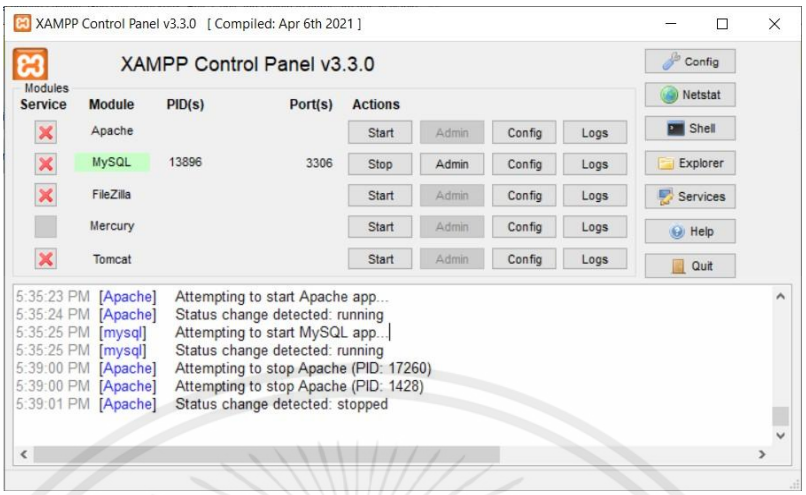
3.3.3 การทดสอบการส่งข้อมูลที่ได้ไปยัง Node-Red

ทำการทดสอบการส่งสัญญาณคลื่นหัวใจ, ค่าออกซิเจนในเลือด และค่าอัตราการเต้นของหัวใจ ที่ได้ไปยัง MQTT เพื่อใช้ในการส่งข้อมูลทั้งหมดไปยัง Node-Red

3.4 เก็บข้อมูลลงในฐานข้อมูล

3.4.1 การใช้ XAMPP เป็นตัวจำลอง web server

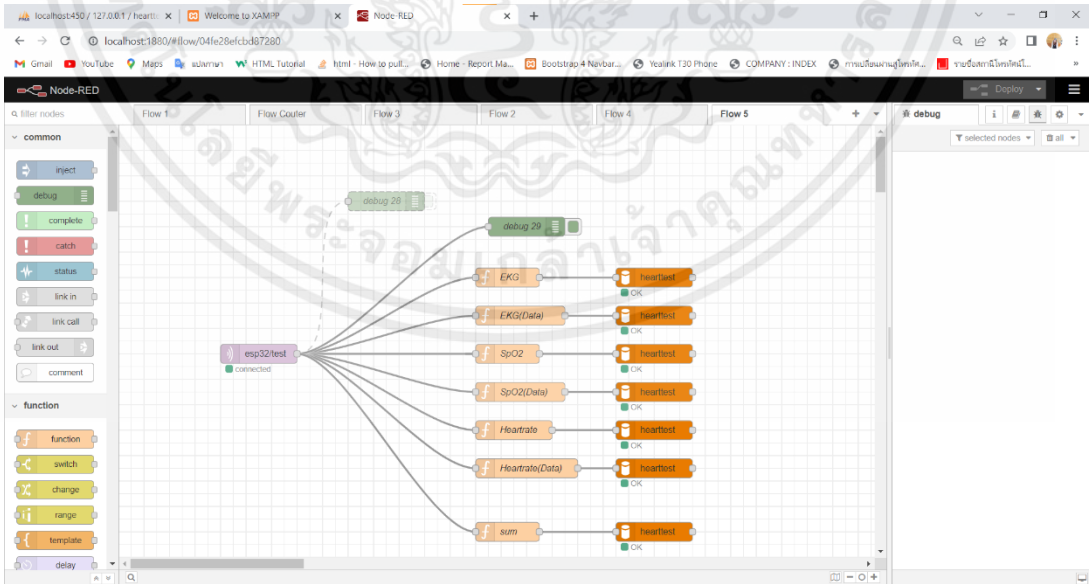
ทำการติดตั้งโปรแกรม XAMPP และตั้งค่าต่างๆ เพื่อใช้ในการจำลอง web server ซึ่ง Xampp เป็นโปรแกรม Apache web server เพื่อทดสอบ สคริปต์หรือหน้าเว็บในเครื่อง โดยในโปรแกรม Xampp มาพร้อมกับภาษา PHP ซึ่งเป็นภาษาสำหรับพัฒนาเว็บแอปพลิเคชัน, MySQL Apache จะทำหน้าที่เป็น web server ดังรูปที่ 3.11



รูปที่ 3.11 หน้าจอขณะเปิดใช้งาน XAMPP

3.4.2 การเก็บข้อมูลใน MySQL

Flow ใน Node-RED ดังรูปที่ 3.12 จะประกอบด้วยโหนด mqtt in เป็นโหนดที่รับข้อมูลจาก MQTT, โหนด function เป็นโหนดสำหรับกำหนดคำสั่ง, โหนด debug สำหรับการแสดงข้อมูลบน Node-RED และโหนด MySQL สำหรับการดึงข้อมูลไปเก็บในฐานข้อมูล

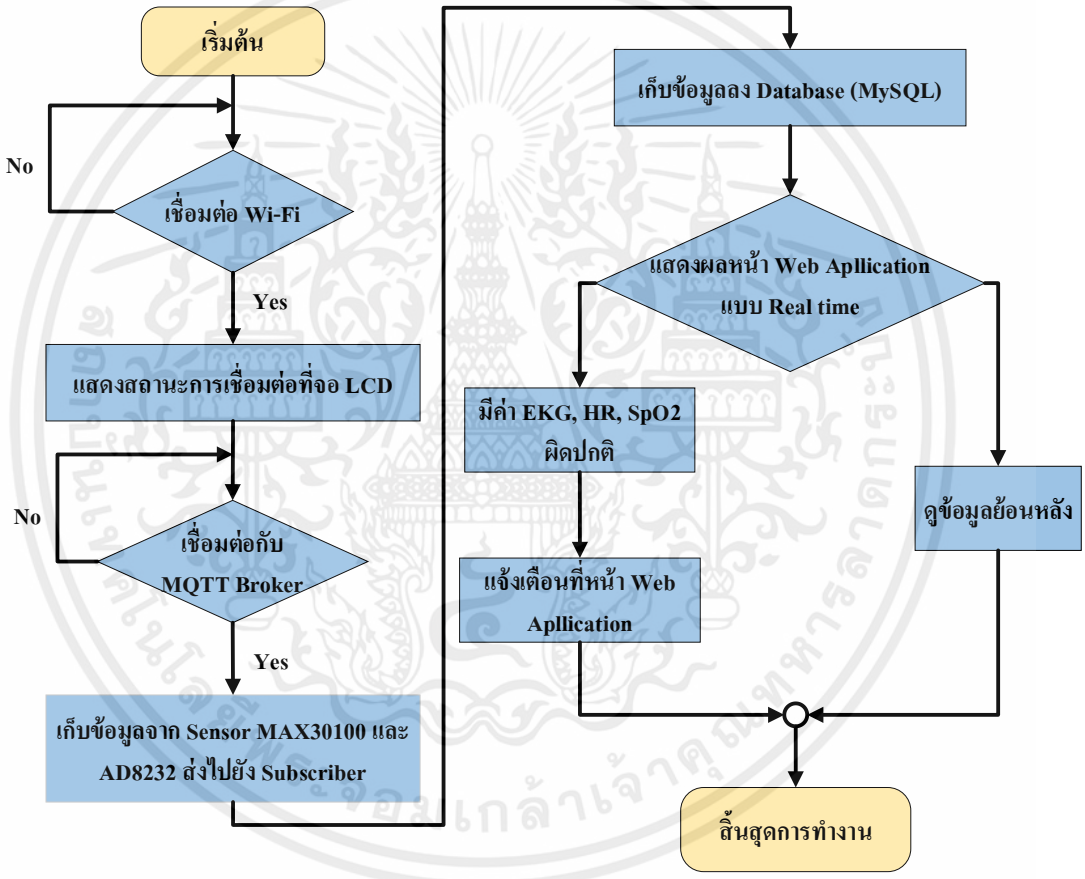


รูปที่ 3.12 Flow ที่ใช้ในการจัดเก็บข้อมูลลงในฐานข้อมูล

3.5 การออกแบบหน้า web application พร้อม Flow chart การทำงาน

3.5.1 Flow chart การทำงาน

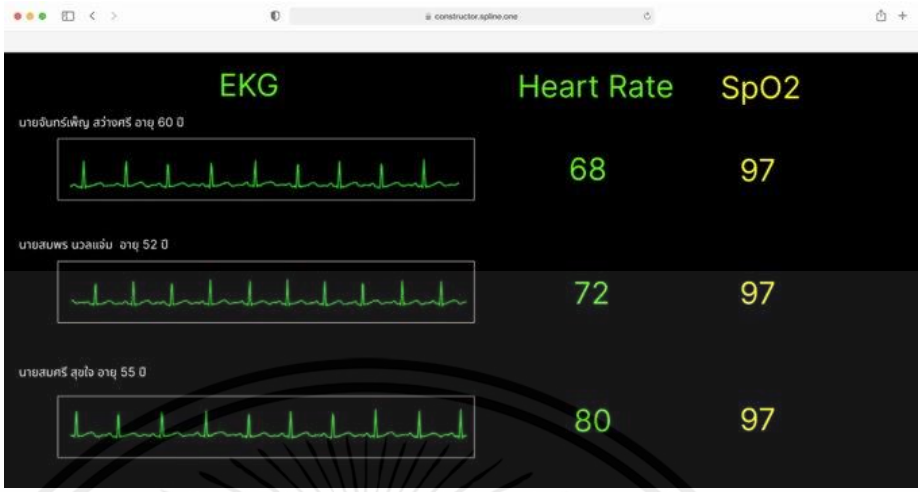
Flow chart การทำงานตั้งแต่การเริ่มใช้งานอุปกรณ์ โดยเริ่มจากการเชื่อมต่อ WIFI เชื่อมต่อกับ Broker MQTT และทำการส่งข้อมูลจาก ESP32 ไปเก็บยัง ฐานข้อมูล MySQL และนำข้อมูลที่เก็บมาทำการแสดงผลหน้า web application แบบ real time รวมถึงเงื่อนไขของการแจ้งเตือนผ่านหน้า web application ดังรูปที่ 3.13



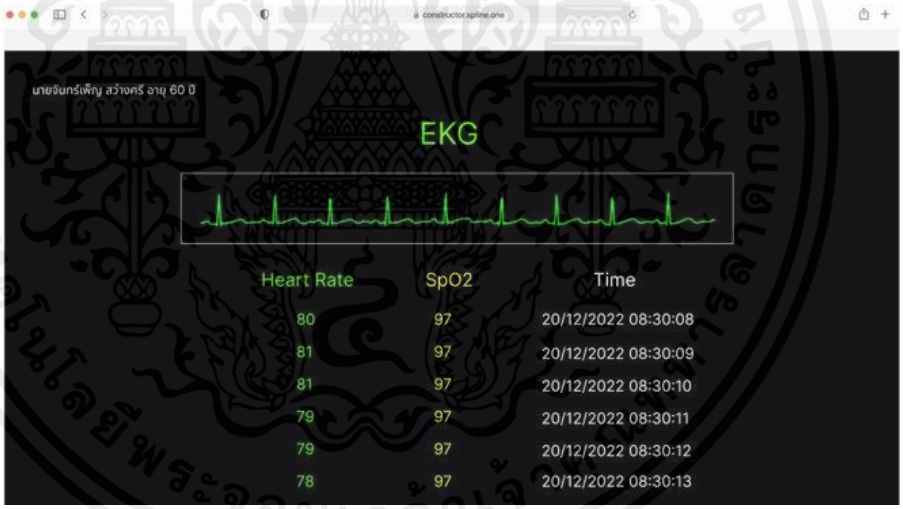
รูปที่ 3.13 Flow chart การทำงาน

3.5.2 ออกแบบหน้า web application ที่จะใช้กับผู้ป่วย

Web Application เป็นแพลตฟอร์มที่ทางผู้จัดทำได้ออกแบบและสร้างขึ้น โดยจะแสดงข้อมูลสัญญาณคลื่นหัวใจ, ค่าอัตราการเต้นหัวใจ และค่าออกซิเจนในกระแสเลือดแบบ Real time ดังรูปที่ 3.14 นอกจากนี้ยังสามารถคลิกไปที่ชื่อผู้ป่วย เพื่อดูข้อมูลย้อนหลังของผู้ป่วยรายคน ดังรูปที่ 3.15

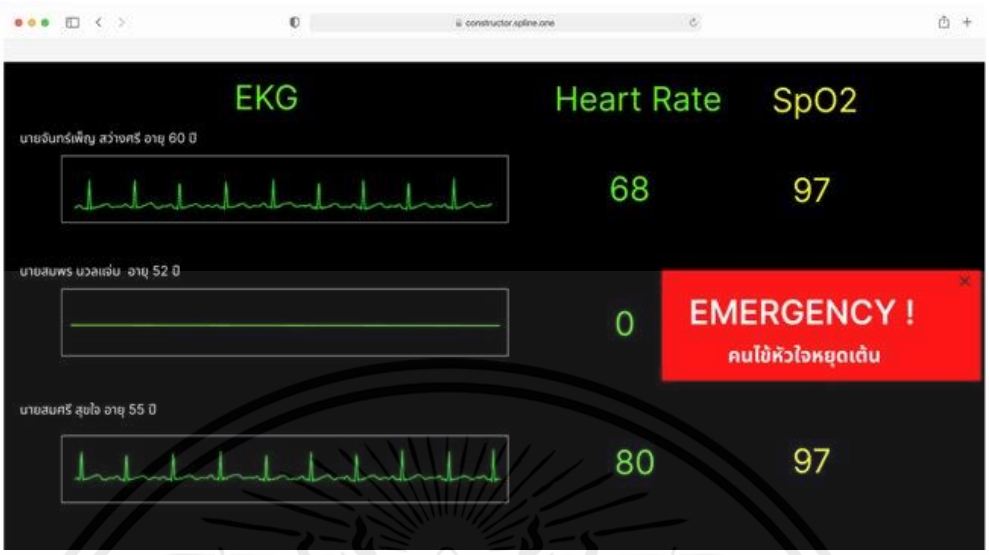


รูปที่ 3.14 หน้า Web Application แสดงสัญญาณคลื่นหัวใจ, ค่าออกซิเจนในเลือด และค่าอัตราการเต้นของหัวใจ

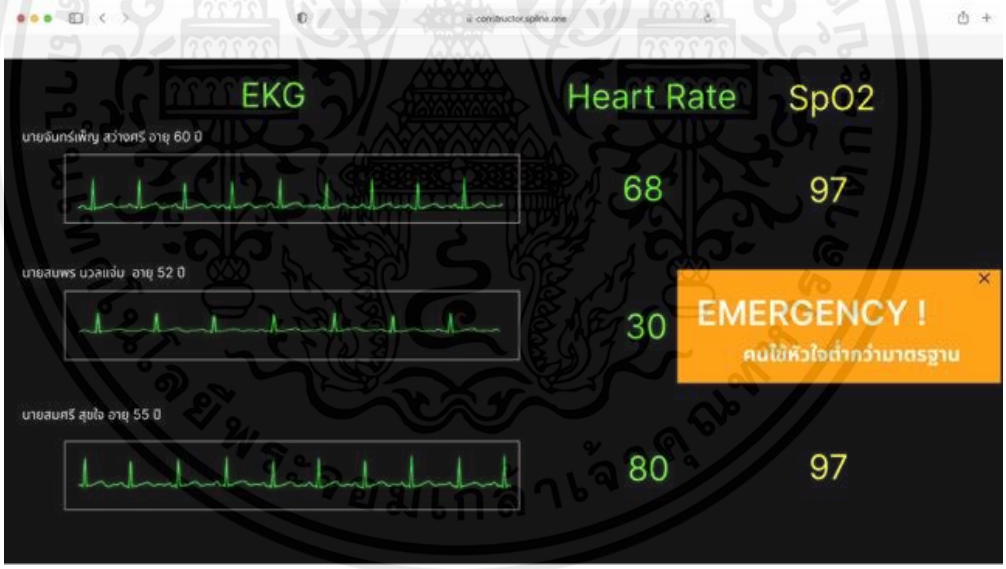


รูปที่ 3.15 หน้า Web Application แสดงสัญญาณคลื่นหัวใจ, ค่าออกซิเจนในเลือด และค่าอัตราการเต้นของหัวใจ ของผู้ป่วยรายคนย้อนหลัง

Web Application มีระบบแจ้งเตือนสถานะ หากผู้ป่วยมีภาวะผิดปกติ โดยจะแสดงสถานะเมื่ออัตราการเต้นของหัวใจผู้ป่วยหยุดทำงาน ดังรูปที่ 3.16 และแสดงสถานะเมื่อออกซิเจนในกระแสเลือดและอัตราการเต้นหัวใจผู้ป่วยต่ำกว่ามาตรฐาน ดังรูปที่ 3.17 ซึ่งการแจ้งเตือนของระบบจะช่วยให้แพทย์หรือคนดูแลใกล้ชิดสามารถรับรู้และช่วยเหลือผู้ป่วยได้ทันเวลา



รูปที่ 3.16 การแจ้งเตือนสถานะบน Web application เมื่อหัวใจหยุดเต้น



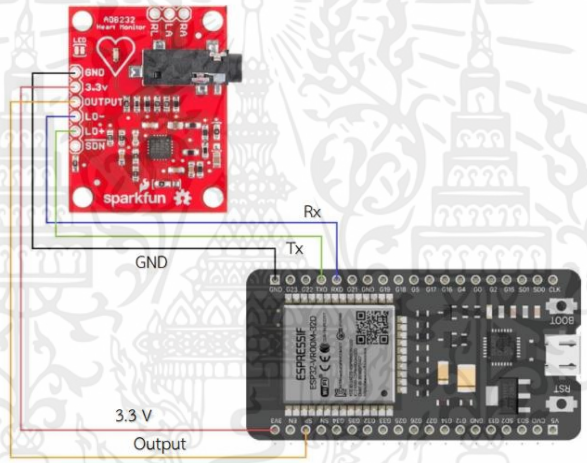
รูปที่ 3.17 การแจ้งเตือนสถานะบน Web application เมื่อหัวใจเต้นต่ำกว่ามาตรฐาน

บทที่ 4

ผลการทดลอง

4.1 ผลการวัดสัญญาณชีพจรจากโมดูล AD8232

ทำการวัดสัญญาณชีพจร จากโมดูล AD8232 โดยนำไปเชื่อมต่อกับ ESP32 ตามรูปที่ 4.1 จะได้สัญญาณเอาต์พุตเป็นแบบ Analog ดังรูปที่ 4.2 ซึ่งเป็น สัญญาณชีพจรโดยมีช่วง amplitude ตั้งแต่ 0-3.3 โวลต์

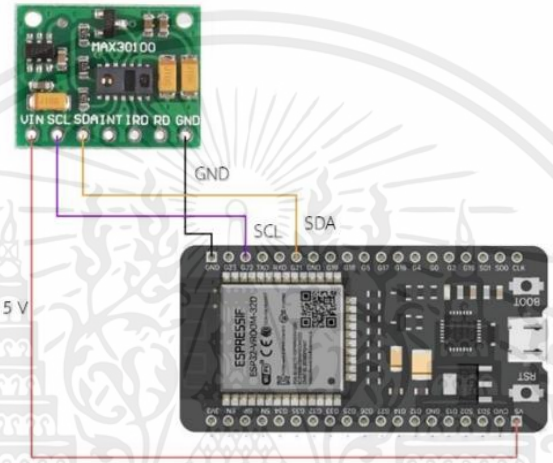


รูปที่ 4.1 โมดูล AD8232 เชื่อมต่อกับ ESP32

รูปที่ 4.2 สัญญาณหัวใจที่ได้จากโมดูล AD8232

4.2 ผลการวัดค่าออกซิเจนในเลือด จากโมดูล MAX30100

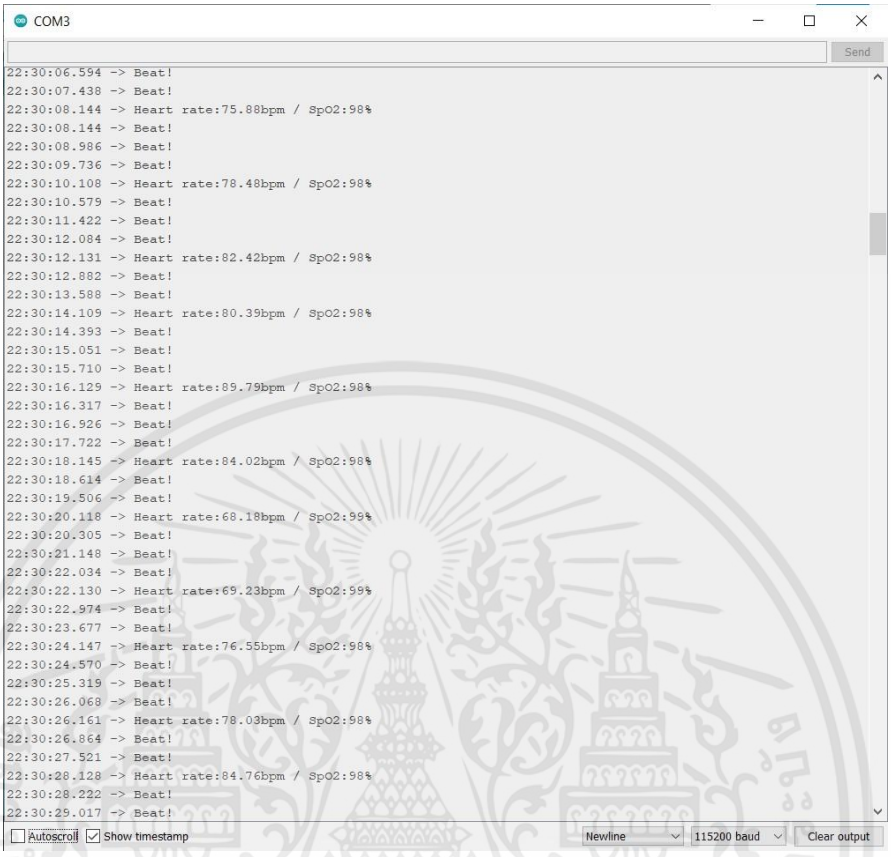
เราจะนำโมดูล MAX30100 ต่อกับ ESP32 ตามรูปที่ 4.3 แล้วทดสอบความถูกต้องและแม่นยำของ MAX30100 เมื่อเทียบกับเครื่อง Pulse Oximeter โดยเราจะทำการทดสอบ 5 รอบ ซึ่งแต่ละรอบจะวัดค่าออกซิเจนในเลือด และค่าอัตราการเต้นของหัวใจทั้งหมด 10 ครั้ง



รูปที่ 4.3 โมดูล MAX30100 เชื่อมต่อกับ ESP32

4.2.1 การทดลองครั้งที่ 1

วัดค่าออกซิเจนในกระแสเลือด และค่าอัตราการเต้นของหัวใจ จาก MAX30100 ทั้งหมด 10 ครั้งดังรูปที่ 4.4 และวัดค่าออกซิเจนในกระแสเลือด และค่าอัตราการเต้นของหัวใจ ที่ได้จากเครื่อง Pulse Oximeter ดังรูปที่ 4.5

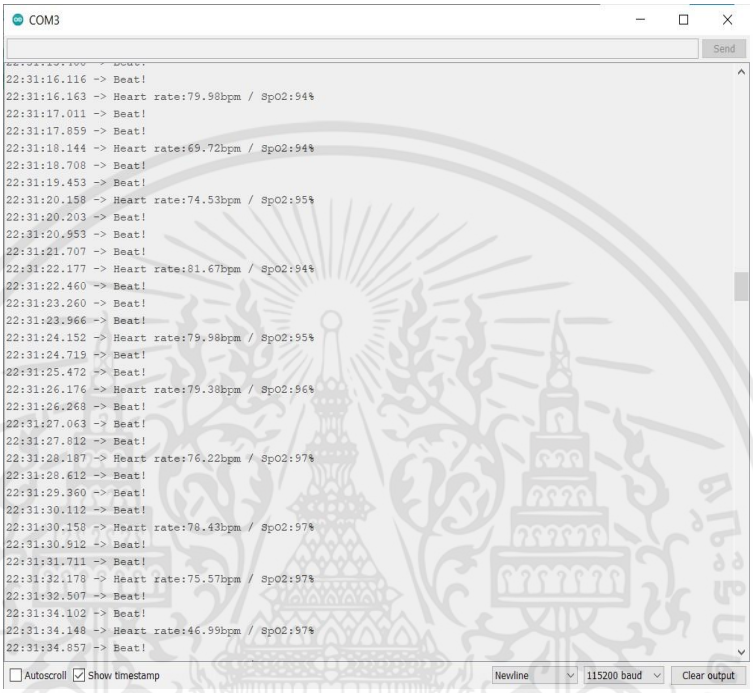


รูปที่ 4.4 ค่าออกซิเจนในเลือดและอัตราการเต้นของหัวใจได้จาก MAX30100 ทั้งหมด 10 ครั้งในรอบที่ 1

รูปที่ 4.5 ค่าออกซิเจนในเลือดและอัตราการเต้นของหัวใจที่ได้จากเครื่อง Pulse Oximeter รอบที่ 1

4.2.2 การทดลองครั้งที่ 2

วัดค่าออกซิเจนในกระแสเลือดและค่าอัตราการเต้นของหัวใจ จาก MAX30100 ทั้งหมด 10 ครั้งดังรูปที่ 4.6 และวัดค่าออกซิเจนในกระแสเลือด และค่าอัตราการเต้นของหัวใจ ที่ได้จากเครื่อง Pulse Oximeter ดังรูปที่ 4.7

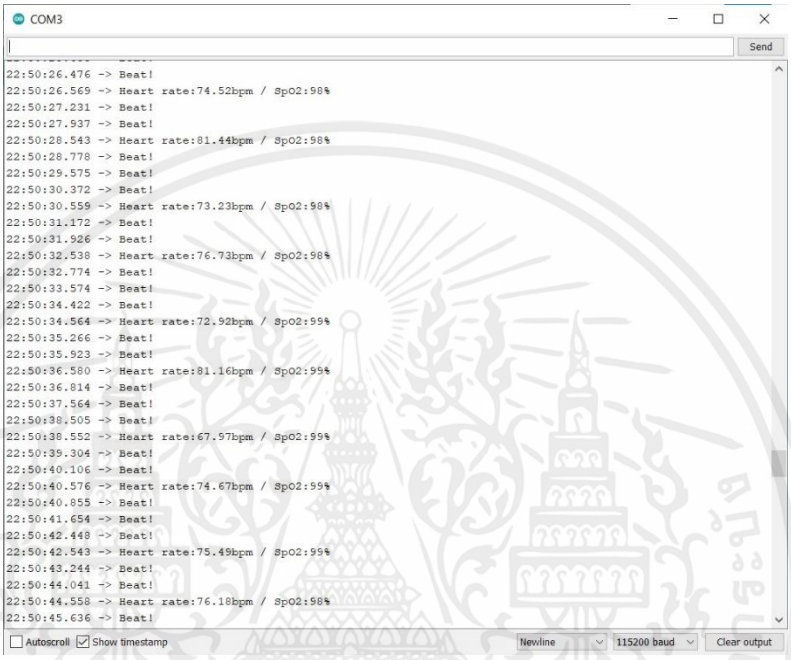


รูปที่ 4.6 ค่าออกซิเจนในเลือดและอัตราการเต้นของหัวใจที่ได้จาก MAX30100 ทั้งหมด 10 ครั้งในรอบที่ 2

รูปที่ 4.7 ค่าออกซิเจนในเลือดและอัตราการเต้นของหัวใจที่ได้จากเครื่อง Pulse Oximeter รอบที่ 2

4.2.3 การทดลองครั้งที่ 3

วัดค่าออกซิเจนในกระแสเลือด และค่าอัตราการเต้นของหัวใจ จาก MAX30100 ทั้งหมด 10 ครั้งดังรูปที่ 4.8 และวัดค่าออกซิเจนในกระแสเลือด และค่าอัตราการเต้นของหัวใจ ที่ได้จากเครื่อง Pulse Oximeter ดังรูปที่ 4.9

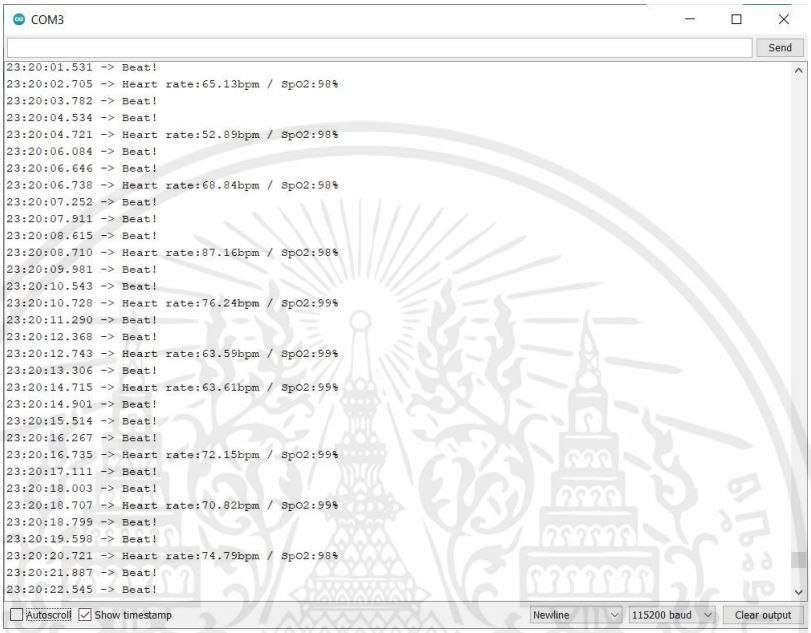


รูปที่ 4.8 ค่าออกซิเจนในเลือดและอัตราการเต้นของหัวใจที่ได้จาก MAX30100 ทั้งหมด 10 ครั้งในรอบที่ 3

รูปที่ 4.9 ค่าออกซิเจนในเลือดและอัตราการเต้นของหัวใจที่ได้จากเครื่อง Pulse Oximeter รอบที่ 3

4.2.4 การทดลองครั้งที่ 4

วัดค่าออกซิเจนในกระแสเลือด และค่าอัตราการเต้นของหัวใจ จาก MAX30100 ทั้งหมด 10 ครั้งดังรูปที่ 4.10 และวัดค่าออกซิเจนในกระแสเลือด และค่าอัตราการเต้นของหัวใจ ที่ได้จากเครื่อง Pulse Oximeter ดังรูปที่ 4.11

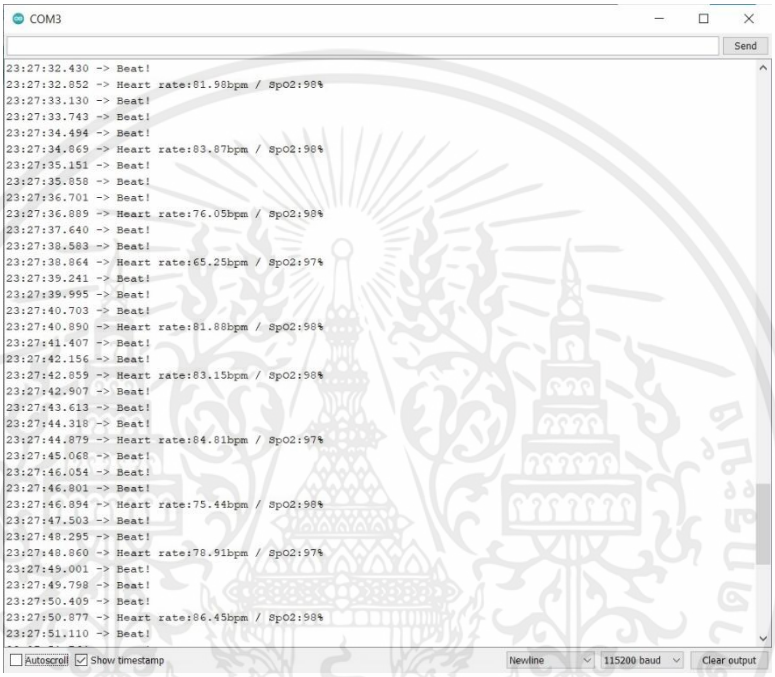


รูปที่ 4.10 ค่าออกซิเจนในเลือดและอัตราการเต้นของหัวใจที่ได้จาก MAX30100 ทั้งหมด 10 ครั้งในรอบที่ 4

รูปที่ 4.11 ค่าออกซิเจนในเลือดและอัตราการเต้นของหัวใจที่ได้จาก เครื่อง Pulse Oximeter รอบที่ 4

4.2.5 การทดลองครั้งที่ 5

วัดค่าออกซิเจนในกระแสเลือด และค่าอัตราการเต้นของหัวใจ จาก MAX30100 ทั้งหมด 10 ครั้งดังรูปที่ 4.12 และวัดค่าออกซิเจนในกระแสเลือด และค่าอัตราการเต้นของหัวใจ ที่ได้จากเครื่อง Pulse Oximeter ดังรูปที่ 4.13



รูปที่ 4.12 ค่าออกซิเจนในเลือดและอัตราการเต้นของหัวใจที่ได้จาก MAX30100 ทั้งหมด 10 ครั้งในรอบที่ 5

รูปที่ 4.13 ค่าออกซิเจนในเลือดและอัตราการเต้นของหัวใจที่ได้จาก เครื่อง Pulse Oximeter รอบที่ 5

จากผลการทดลองที่ได้ทั้งหมด 5 รอบ เราจะนำค่าออกซิเจนในกระแสเลือด และค่าอัตราการเต้นของหัวใจ ทั้ง 10 ครั้ง แต่ละรอบมาคำนวณหาค่าเฉลี่ย แล้วนำมาคิดหาเปอร์เซ็นความคลาดเคลื่อนของค่าออกซิเจนในกระแสเลือด และค่าอัตราการเต้นของหัวใจ ดังตารางที่ 4.1 เพื่อตรวจสอบความแม่นยำของ MAX30100 เมื่อเทียบกับเครื่อง Pulse Oximeter ซึ่งจากค่าความคลาดเคลื่อนที่ได้นั้น จะเห็นว่าไม่เกิน 5% ดังนั้นค่าที่ได้จาก MAX30100 จึงมีความน่าเชื่อถือและสามารถนำมาใช้จริงได้

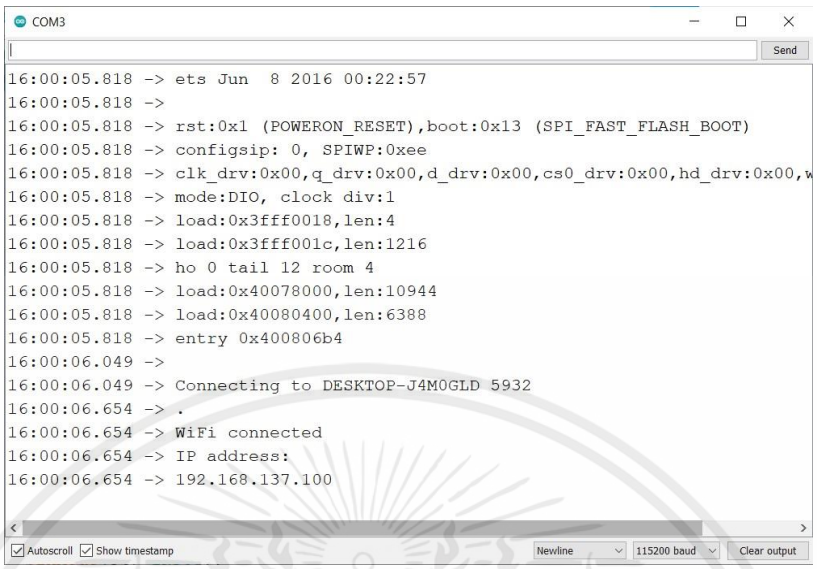
ตารางที่ 4.1 แสดงค่าความแตกต่างและค่าความคลาดเคลื่อนที่ได้จากการเปรียบเทียบระหว่าง MAX30100 และเครื่อง Pulse Oximeter

ครั้งที่	Max30100		เครื่อง pulse oximeter		ค่าความคลาดเคลื่อนของ Heart rate (%)	ค่าความคลาดเคลื่อน ของ SpO_2 (%)
	Mean Heart rate (bpm)	Mean SpO_2 (%)	Heart rate (bpm)	SpO_2 (%)		
1	78.30	98.20	81.00	99.00	3.33	0.80
2	74.25	95.60	76.00	99.00	2.30	3.40
3	75.43	98.50	76.00	99.00	0.75	0.50
4	69.52	98.50	72.00	99.00	3.44	0.50
5	79.78	97.70	79.00	99.00	0.99	1.30

4.3 ผลการเชื่อมต่อ ESP32 กับ Wi-Fi

ทำการเชื่อมต่อ Wi-Fi กับ ESP32 ซึ่งโค้ดสำหรับเชื่อมต่อแสดงดังรูปที่ 4.14 และผลการเชื่อมต่อจะแสดงดังรูปที่ 4.15

รูปที่ 4.14 โค้ดสำหรับเชื่อมต่อ Wi-Fi กับ ESP32

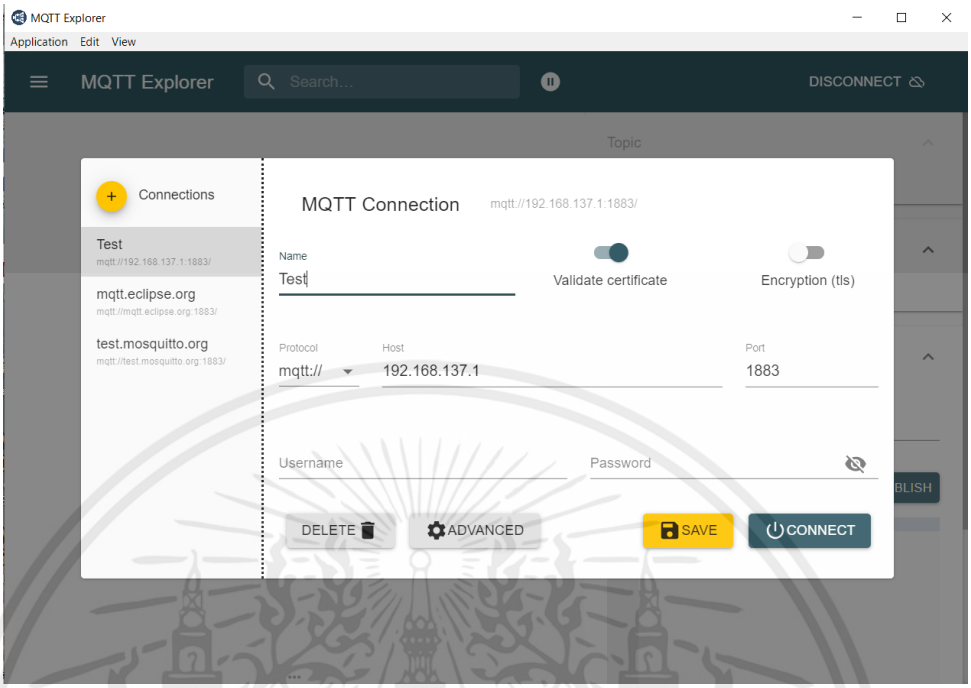


รูปที่ 4.15 ผลการเชื่อมต่อ Wi-Fi กับ ESP32 สำเร็จ

4.4 ผลการเชื่อมต่อระหว่าง ESP32 กับ MQTT เพื่อส่งข้อมูล

ทำการกำหนดค่าพารามิเตอร์ที่ใช้ในการเชื่อมต่อกับ MQTT ได้แก่ IP Address local ของ Wi-Fi ที่เชื่อมต่อ, ชื่อ topic (เส้นทางที่ต้องการส่งข้อมูล), และหมายเลข port ของ MQTT ดังรูปที่ 4.16 ให้ตรงกับหน้าแอปพลิเคชันของ MQTT explorer ดังรูปที่ 4.17

รูปที่ 4.16 กำหนดค่าพารามิเตอร์ที่ใช้ในการเชื่อมต่อกับ MQTT



รูปที่ 4.17 หน้าแอปพลิเคชันของ MQTT explorer

ถ้าส่งข้อความไปให้ MQTT ฟังก์ชัน callback จะถูกเรียกใช้ เพราะ MQTT จะตอบกลับมาที่ ESP32 เพื่อเป็นการตรวจสอบว่าข้อความที่ส่งไปส่งไปถึงหรือไม่ดังรูปที่ 4.18

```
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();

}
```

รูปที่ 4.18 โค้ดฟังก์ชันตอบกลับว่าข้อความส่งไปยัง MQTT แล้ว

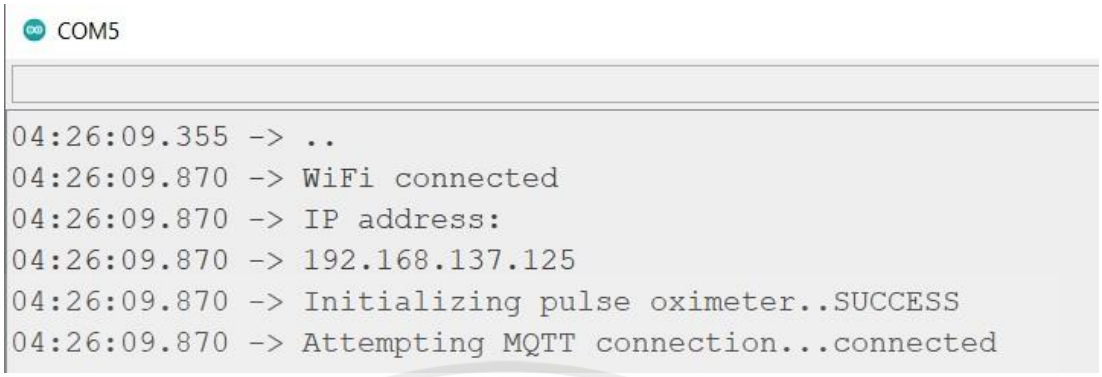
ถ้า ESP32 ไม่สามารถเชื่อมต่อกับ MQTT ได้ ฟังก์ชัน reconnect โดยใช้โค้ดดังรูปที่ 4.19 ซึ่งจะถูกเรียกใช้ โดยเป็นฟังชันก์การทำงานที่สั่งให้เริ่มเชื่อมต่อใหม่อีกครั้ง จนกว่าจะเชื่อมต่อสำเร็จ โดยใช้โค้ดดังรูปที่ 4.20 และหากเชื่อมต่อสำเร็จจะแสดงดังรูปที่ 4.21

```
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Create a random client ID
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    // Attempt to connect
    if (client.connect(clientId.c_str())) {
      Serial.println("connected");
      // Once connected, publish an announcement...
      client.publish("outTopic", "hello world");
      // ... and resubscribe
      client.subscribe("inTopic");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}

void setup() {

  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
  pinMode(34, INPUT);
}
```

รูปที่ 4.19 ฟังก์ชัน reconnect เมื่อเกิดการเชื่อมต่อไม่สำเร็จ

```
void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  ADC_VALUE = analogRead(Analog_channel_pin);

  unsigned long now = millis();
  if (now - lastMsg > 50) {
    lastMsg = now;
//    ++value;
    snprintf (msg, MSG_BUFFER_SIZE, "%f", (ADC_VALUE/4095.0)*3.3);
    delay(100);
//    Serial.print("Publish message: ");
    Serial.println(msg);
    client.publish("esp32/test", msg);

  }
}
```

รูปที่ 4.20 โค้ดเชื่อมต่อ ESP32 เชื่อมต่อกับ Wi-Fi และ MQTT

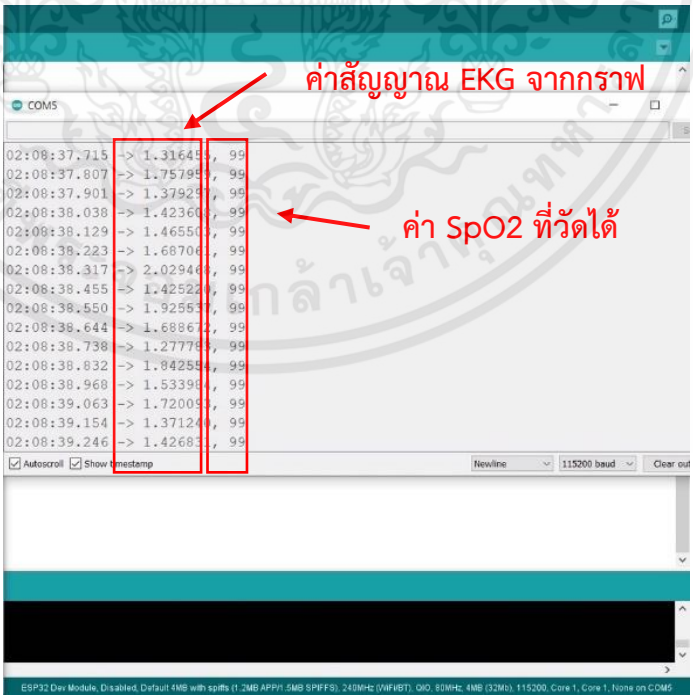


รูปที่ 4.21 ผลการเชื่อมต่อ ESP32 เชื่อมต่อกับ Wi-Fi และ MQTT สำเร็จ

4.5 ผลการเชื่อมต่อระหว่าง MQTT กับ Node-RED เพื่อส่งข้อมูล

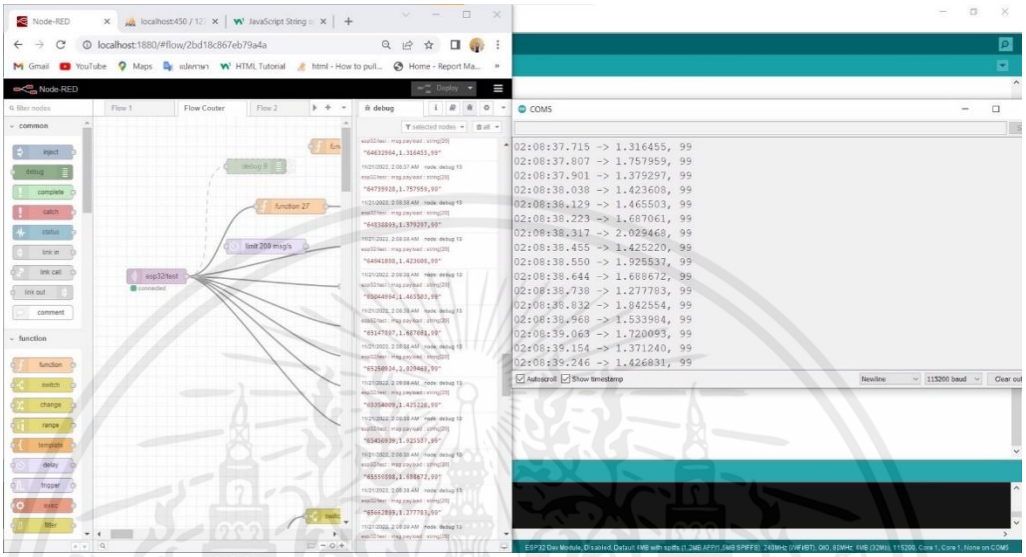
เมื่อทำการเชื่อมต่อ MQTT กับ Node-RED สำเร็จแล้ว ที่หน้าเว็บของ Node-RED ตรงโหนดของ MQTT จะขึ้นสถานะ connected และเชื่อมต่อโหนด debug เพื่อให้ Node-RED แสดงข้อมูลที่รับมาที่ช่อง debug

เมื่อทำการวัดสัญญาณชีพจรและค่าออกซิเจนในเลือด ESP32 จะทำการส่งค่าที่ได้ไปยัง MQTT และ Node-RED ต่อไปตามลำดับ ดังรูปที่ 4.22

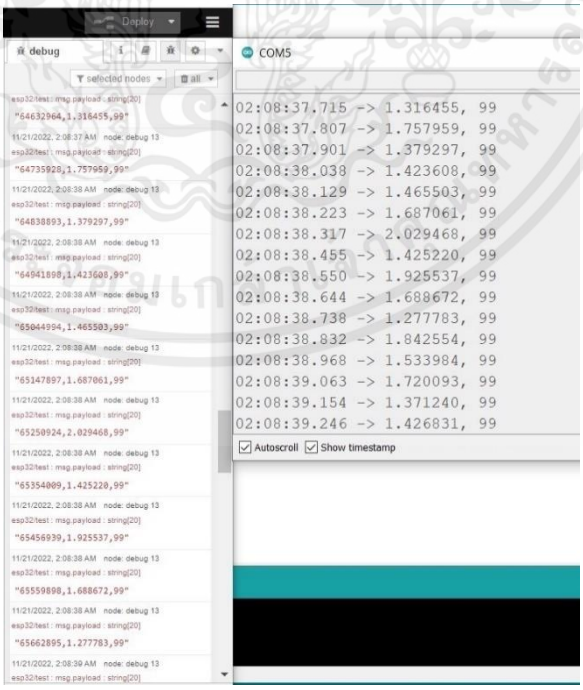


รูปที่ 4.22 ข้อมูลที่จะส่งจาก ESP32

เมื่อมีการส่งข้อมูลจาก ESP32 ผ่าน MQTT มายัง Node-RED จะแสดงผลที่หน้าต่าง debug โดยจะแสดงค่าที่ได้รับมาจาก MQTT ดังรูปที่ 4.23



รูปที่ 4.23 ข้อมูลที่ส่งจาก ESP32 ผ่าน MQTT มายัง Node-RED

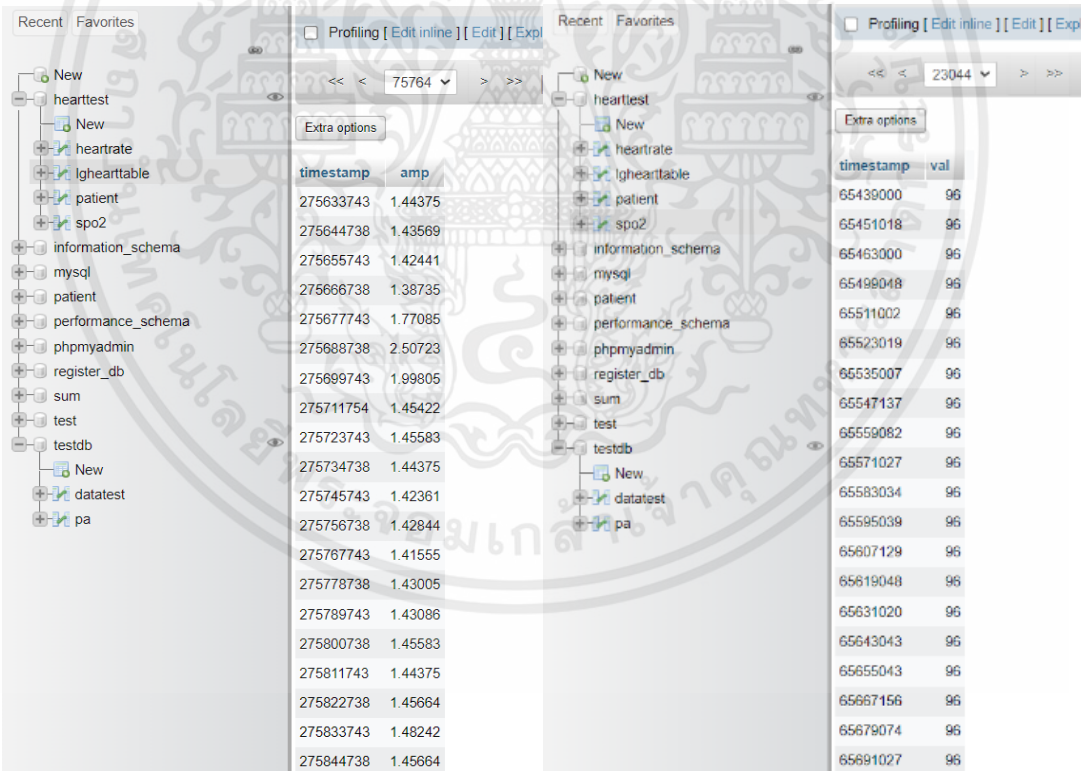


รูปที่ 4.24 เปรียบเทียบข้อมูลที่ Node-RED รับมาจาก MQTT กับ serial monitor

เมื่อ ESP32 ส่งข้อมูลผ่าน Wi-Fi ไปยัง MQTT และจาก MQTT มายัง Node-RED จะได้ผลดังรูป 4.24 โดยค่าที่แสดงเป็นค่าของสัญญาณคลื่นหัวใจ และออกซิเจนในเลือดที่วัดได้จากโมดูล MAX30100 และได้มีการตรวจสอบความแม่นยำ เมื่อเทียบกับ output จาก serial monitor พบว่าข้อมูลตรงกัน

4.6 ผลการนำข้อมูลจาก Node Red เก็บลงในฐานข้อมูล

เมื่อผู้ป่วยใช้งานตัวอุปกรณ์ จะมีการวัดอัตราการเต้นของหัวใจ หรือค่าออกซิเจนในกระแสเลือด โดยข้อมูลที่วัดได้จะส่งผ่าน Wi-Fi แล้วส่งไปยัง MQTT และนำมาเก็บข้อมูลลงใน MySQL เช่น ค่าสัญญาณคลื่นหัวใจ ซึ่งแสดงดังรูปที่ 4.25 (ก) หรือค่าออกซิเจนในกระแสเลือด ดังรูปที่ 4.26 (ข) ซึ่ง MySQL เป็นระบบจัดการฐานข้อมูล โดยจะทำการเก็บข้อมูลทั้งเวลาและค่าที่วัดได้ในรูปแบบของตาราง

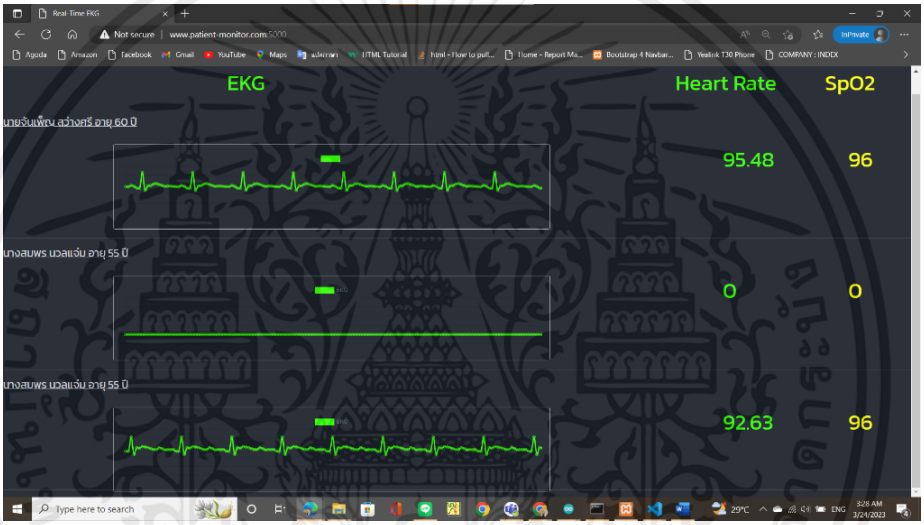


(ก) (ข)

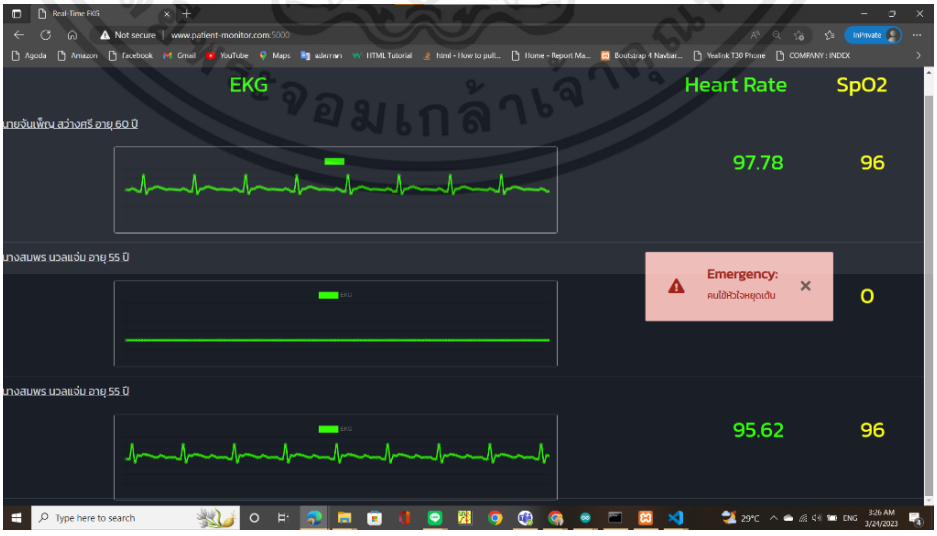
รูปที่ 4.25 ตัวอย่างการเก็บข้อมูลใน ฐานข้อมูล (ก) ข้อมูลของสัญญาณคลื่นหัวใจ (ข) ข้อมูลของออกซิเจนในเลือด

4.7 การแสดงผลบนหน้า Web application

มีการทำ Web Application ตามที่ออกแบบเอาไว้ ซึ่งสามารถใช้ได้กับสมาร์ทโฟนและคอมพิวเตอร์ โดยข้อมูลจะส่งผ่านอินเตอร์เน็ต Wi-Fi ซึ่งการทำงานจะแสดงข้อมูลที่ได้รับจากอุปกรณ์แล้วนำมาแสดงผลในรูปแบบของกราฟเพื่อสะดวกในการติดตามผล เมื่อทำการส่งข้อมูลเพื่อเก็บ ในระบบฐานข้อมูลเรียบร้อยแล้วจะนำข้อมูลมาผลแสดงบน Web Application โดยจะแสดงข้อมูลสัญญาณ คลื่นหัวใจ, ค่าออกซิเจนในกระแสเลือด และค่าอัตราการเต้นของหัวใจแบบ Real time ดังรูปที่ 4.26 โดยมีการแจ้งเตือนเมื่อพบความผิดปกติดังรูปที่ 4.27 สามารถดูย้อนหลังได้ดังรูปที่ 4.28



รูปที่ 4.26 Web Application แสดงแสดงสัญญาณคลื่นหัวใจ, อัตราการเต้นของหัวใจและออกซิเจนในเลือด



รูปที่ 4.27 Web Application แสดงการแจ้งเตือน

รูปที่ 4.28 Web Application แสดงแสดงสัญญาณคลื่นหัวใจ, อัตราการเต้นของหัวใจ และออกซิเจนในเลือดของผู้ป่วยรายคนย้อนหลัง

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

ปริญญานิพนธ์นี้มีจุดประสงค์เพื่อทำเครื่องติดตามสัญญาณชีพระยะไกลต้นทุนต่ำ (Low-cost Patient Vital signs remote monitoring) ซึ่งมีราคาที่ถูกกว่าและมีประสิทธิภาพใกล้เคียงกับ Patient Monitor ตามท้องตลาด โดยผลที่ได้จากการทำงานของอุปกรณ์สามารถแสดงคลื่นหัวใจ และสามารถวัดอัตราการเต้นของหัวใจกับค่าออกซิเจนในเลือด ได้ตรงกับ Oximeter ที่มีตามท้องตลาด

โดยมีผลการทดสอบความถูกต้องและแม่นยำของ sensor Max30100 เปรียบเทียบกับเครื่อง pulse oximeter ทั้งหมด 5 รอบ ดังนี้ ค่าเปอร์เซ็นต์ความคลาดเคลื่อนอัตราการเต้นของหัวใจเท่ากับ 3.33%, 2.30%, 0.75%, 3.44%, 0.99% และเปอร์เซ็นต์ความคลาดเคลื่อนออกซิเจนในเลือดเท่ากับ 0.80%, 3.40%, 0.50%, 0.50% และ 1.30%

5.2 ข้อเสนอแนะ

จากการทำการทดสอบเครื่องติดตามสัญญาณชีพระยะไกลต้นทุนต่ำ ในการวัดสัญญาณคลื่นหัวใจ การที่ขยับร่างกายเยอะหรือติดแผ่นอิเล็กโทรดผิดตำแหน่ง จะทำให้สัญญาณคลื่นหัวใจไม่นิ่ง ดังนั้น การวัดคลื่นหัวใจ ต้องอยู่ในท่าที่นิ่งหรืออยู่ในท่านอน เพื่อให้ได้สัญญาณที่คงที่

บรรณานุกรม

[1] Health me now. “การตรวจคลื่นไฟฟ้าหัวใจ” https://healthmenowth.com/diagnosis/special-diagnosis/electrocardiography/

[2] Michelle White. “An Overview of the Workings and Afflictions of the AV Node” https://medium.com/medical-cps/an-overview-of-the-workings-and-afflictions-of-the-av-node-9253a808659b

[3] GARMIN. “รู้ได้อย่างไรว่า SpO2 ไม่ปกติ” https://www.garmin.com/th-TH/blog/the-oxygen-level-in-the-blood/

[4] pDragon. “ESP32 ทำงานอย่างไร Pin I/O พื้นฐาน” https://v89infinity.com/esp32/

[5] StudioPieters. “ESP32”, https://www.studiopieters.nl/esp32-pinout/

[6] SparkFun. “Single Lead Heart Rate Monitor - AD8232” https://www.sparkfun.com/products/12650

[7] ทัศนีย์ ภู่สำอางค์, “การเลือก lead เพื่อติดตามคลื่น ST-segment ผู้ป่วยกล้ามเนื้อหัวใจขาดเลือด :การตดัสินใจทางคลินิกในหอผู้ป่วยวกิฤตและฉุกเฉิน Selection of Optimal EKG Lead Monitoring ST-segment in Myocardial Ischemia: Clinical judgement in Intensive and Emergency Care Unit”. คณะพยาบาลศาสตร์ มหาวิทยาลัยธรรมศาสตร์ ศูนย์รังสิต ตำบลคลองหนึ่ง อำเภอคลองหลวง จังหวัดปทุมธานี 12120

[8] F. Khateb, P.Prommee, and T.Kulej, “MIOTA-based Filters for Noise and Motion Artifact Reductions in Biosignal Acquisition,”. IEEE Access, vol. 10, 14325–14338, 2022

[9] gravitechthai. “MAX30100” https://shorturl.asia/OyT0U

บรรณานุกรม (ต่อ)

[10] theerapone. “การใช้งาน Wi-Fi และ HTTP”
http://theerapone.com/sbc/courses/m5stack/doc/M5Stack_WiFi.pdf

[11] riverplus. “What is MQTT ? โปรโตคอลเพื่อการสื่อสารของ IOT โดยเฉพาะ”
https://iiot.riverplus.com/mqtt/

[12] Sonic. “MQTT กับงาน Industrial Internet of Things (IoT)”
https://shorturl.asia/yGXd1

[13] IOTQU. “NODE-RED”
https://iotqu.blogspot.com/2019/02/sekilas-tentang-node-red.html

[14] mindphp.com, “Database คืออะไร”,
https://rb.gy/jqadld

[15] saixiii, “MySQL คืออะไร? และ ไว้ทำอะไร?”,
https://saixiii.com/what-is-mysql/

[16] ศูนย์เทคโนโลยีสารสนเทศและการสื่อสาร สป., “การใช้งาน MySQL”,
http://elearning.psru.ac.th/courses/66/SQL/CH07_BasicMySQL.pdf

[17] Thapanapong Rukkanchanunt, “WEB APPLICATION I”,
https://www2.cs.science.cmu.ac.th/courses/204202/lib/exe/fetch.php?media=lec04.pdf

[18] thaiware, “XAMPP (โปรแกรม XAMPP ติดตั้ง Web Server ด้วยตัวเอง)”,
https://software.thaiware.com/3095-XAMPP-Web-Server.html

[19] Xampp, “xampp-คืออะไร”,
https://rb.gy/mtnhff

ภาคผนวก

โค้ดที่ใช้ในปริญญานิพนธ์

โค้ดวัดชีพจร (โมดูล AD8232 เชื่อมต่อกับ ESP32)

```
const int Analog_channel_pin = 34;
int ADC_VALUE = 0;
float voltage_value = 0;
int DAC = 0;

void setup()
{
Serial.begin(115200);
pinMode(34,INPUT);
}
void loop()
{
ADC_VALUE = analogRead(Analog_channel_pin);
//Serial.print("ADC VALUE = ");
//Serial.println(ADC_VALUE);
//delay(100);


voltage_value = (ADC_VALUE/4095.0)*3.3;
Serial.print("Voltage ");
Serial.println(voltage_value);
```

โค้ดวัดค่าออกซิเจนในเลือด (โมดูล MAX30100 เชื่อมต่อกับ ESP32)

```
#include <Wire.h>
#include "MAX30100_PulseOximeter.h"

#define REPORTING_PERIOD_MS   2000

PulseOximeter pox;
uint32_t tsLastReport = 0;

void onBeatDetected()
{
   Serial.println("Beat!");
}

void setup()
{
   Serial.begin(115200);
   Serial.print("Initializing pulse oximeter..");

   if (!pox.begin()) {
      Serial.println("FAILED");
      for(;;);
   } else {
      Serial.println("SUCCESS");
   }
.
    pox.setOnBeatDetectedCallback(onBeatDetected);
    pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);
}
void loop()
{
   // Make sure to call update as fast as possible
   pox.update();
   if (millis() - tsLastReport > REPORTING_PERIOD_MS) {
      Serial.print("Heart rate:");
      Serial.print(pox.getHeartRate());
      Serial.print("bpm / SpO2:");
      Serial.print(pox.getSpO2());
      Serial.println("%");

      tsLastReport = millis();
   }
}
```

โค้ดรวมวัดชีพจรและวัดค่าออกซิเจน โดยส่งผ่าน Wi-Fi และ MQTT เพื่อส่งต่อข้อมูล

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <Wire.h>

#include "MAX30100_PulseOximeter.h"


#define REPORTING_PERIOD_MS   2000

#include <LiquidCrystal_I2C.h>

// Max30100
#define REPORTING_PERIOD_MS   2000
#define SAMPLING_RATE   MAX30100_SAMPRATE_100HZ
MAX30100 sensor;
PulseOximeter pox;
uint32_t tsLastReport = 0;

#define LED_BUILTIN 2 //pin with LED to turn on when BT connected

// WiFi
const char *ssid = "LAPTOP-SAJVGMK2 4675"; //WiFi name
String strssid = "LAPTOP-SAJVGMK2 4675";
const char *password = "12356789";

// ******Broker*********
const char *mqtt_server = "192.168.137.1";
const char *topic = "Test/Msg";
const char *mqtt_username = "";
const char *mqtt_password = "";
const int mqtt_port = 1883;


WiFiClient espClient;
PubSubClient client(espClient);

unsigned long now = 0;
unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE (50)
char msg[MSG_BUFFER_SIZE];
char msgpn[MSG_BUFFER_SIZE];
char msg1[MSG_BUFFER_SIZE];

int value = 0;
```

```
void setup_wifi() {

  delay(10);

  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Connecting to");
  lcd.setCursor(0, 1);
  lcd.print(ssid);


  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
    //delay(500);
    //lcd.clear();
    //lcd.setCursor(0, 0);
    //lcd.print("Wi-Fi Disconnect");
  }
  randomSeed(micros());
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("WiFi-connected");
  lcd.setCursor(0, 1);
  lcd.print(ssid);


}

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  //lcd.setCursor(0, 0);
  //lcd.print("Message arrived ");

  for (int i = 0; i < length; i++) {
```

```
    Serial.print((char)payload[i]);
  }
  Serial.println();
}
void reconnect() {

  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");

    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);

    if (client.connect(clientId.c_str())) {
      Serial.println("connected");
      client.publish("outTopic", "hello world");
      client.subscribe("inTopic");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");

      delay(5000);
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Disconnected");

    }
  }
}


void setup() {

  // initialize the LCD

    //lcd.setCursor(0, 0);
      //  lcd.print("HR : ");

  Serial.begin(115200);
  pinMode(34, INPUT);
  pinMode(32, INPUT);
  pinMode(12, OUTPUT);
  digitalWrite(12,1);
  delay(10);
  lcd.begin();
  setup_wifi();
  client.setServer(mqtt_server, 1883);
```

```
  client.setCallback(callback);
  Serial.print("Initializing pulse oximeter..");

    if (!pox.begin()) {
       Serial.println("FAILED");
       //lcd.clear();
       //lcd.setCursor(3, 0);
       //lcd.print("FAILED");
//       for(;;);
    } else {
       Serial.println("SUCCESS");
       //lcd.clear();
       //lcd.setCursor(3, 0);
       //lcd.print("SUCCESS");
    }

    pox.setIRLedCurrent(MAX30100_LED_CURR_46_8MA);
    sensor.setSamplingRate(SAMPLING_RATE);
    //delay(100);
    digitalWrite(12,0);

}


 void loop() {
    if (!client.connected()) {

      reconnect();

    }
    client.loop();
    pox.update();

    now = micros();
    Serial.println(digitalRead(32));
    if (digitalRead(32) == 1 ){
       lastMsg = micros();
       digitalWrite(12,1);
//       delay(10);
       lcd.begin();
       lcd.clear();
       lcd.setCursor(0, 0);
       lcd.print(strssid);
       delay(5000);
       digitalWrite(12,0);
    }
```

```
   float temp = (analogRead(34)/ 4096.0 * 3.3);
   snprintf (msg, MSG_BUFFER_SIZE, "%lu,%f,%s,%s, %d", now, temp,
String(pox.getHeartRate()), String(pox.getSpO2()), 1);
   snprintf (msgpn, MSG_BUFFER_SIZE, "%f, %s, %s", temp ,
String(pox.getHeartRate()), String(pox.getSpO2()));
     snprintf (msg1, MSG_BUFFER_SIZE, "lu,%f", now, temp);
//     Serial.print("Publish message: ");
   Serial.println(msg1); // ดูกราฟ
//Serial.println(msg);
   client.publish("esp32/test", msg);
//   digitalWrite(12,1);
   delay(100);



//     tsLastReport = millis();
   }
 // }
```