



รายงาน

เรื่อง ระบบติดตามอุณหภูมิอัจฉริยะสำหรับผู้ป่วยที่อยู่ตามลำพัง

เสนอ

ผู้ช่วยศาสตราจารย์ ดร. วิจินันท์ ตันติธรรม

จัดทำโดย สมาชิกกลุ่ม 14

น.ส. ปภัสสินทร์ กมลจริยฉัตร์ รหัสนักศึกษา 6687036

นาย สิรชีร์ กลินบัว รหัสนักศึกษา 6687102

คณะเทคโนโลยีสารสนเทศและการสื่อสาร มหาวิทยาลัยมหิดล นักศึกษาชั้นปีที่ 2

รายงานนี้เป็นส่วนหนึ่งของรายวิชา ITDS282_ Internet of Things Lab

ภาคเรียนที่ 2 ปีการศึกษา 2567

คำนำ

รายงานฉบับนี้จัดทำขึ้นเพื่อเสนอแนวคิดและแนวทางการพัฒนาเทคโนโลยี Internet of Things (IoT) ได้เปลี่ยนแปลงวิธีการดูแลสุขภาพในปัจจุบัน โดยเฉพาะสำหรับผู้ป่วยที่ต้องอยู่ตามลำพัง การติดตามอุณหภูมิร่างกายเป็นสิ่งสำคัญ เนื่องจากภาวะไข้สูงอาจนำไปสู่อันตรายถึงชีวิตหากไม่ได้รับการดูแลทันท่วงที

โครงการ "ระบบติดตามอุณหภูมิอัจฉริยะสำหรับผู้ป่วยที่อยู่ตามลำพัง" นี้พัฒนาขึ้นโดยใช้เซนเซอร์ DHT22 ร่วมกับบอร์ด ESP32 และโมดูล RTC เพื่อตรวจวัดอุณหภูมิแบบต่อเนื่อง เมื่ออุณหภูมิสูงถึง 39 องศาเซลเซียส ระบบจะส่งสัญญาณเตือนผ่าน LED เพื่อให้ผู้ที่อยู่ใกล้เคียงทราบและให้ความช่วยเหลือได้ทันที

โครงการนี้มุ่งเน้นการสร้างนวัตกรรมที่เข้าถึงได้และมีประสิทธิภาพ เพื่อยกระดับคุณภาพชีวิตของผู้ป่วยและเพิ่มความมั่นใจให้กับผู้ดูแลที่ไม่สามารถอยู่เฝ้าได้ตลอดเวลา

จัดทำโดย

น.ส. ปักสินธ์ กมลจริยะฉัตร์

นาย สิริชัย กลินบัว

สารบัญ

หัวข้อ	หน้า
บทที่ 1 บทนำ	1
1.1 ความเป็นมาและความสำคัญ	2
1.2 วัตถุประสงค์ของโครงงาน	3
1.3 ขอบเขตของโครงงาน	4
1.4 ประโยชน์ที่คาดว่าจะได้รับ	5
บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง	6
2.1 ทฤษฎีที่เกี่ยวข้อง	6
2.1.1 อินเทอร์เน็ตของสรรพสิ่ง (Internet of Things: IoT)	6
2.1.2 การประมวลผลแบบเรียลไทม์ (Real-time Processing)	6
บทที่ 3 ตัวอย่างการใช้งาน scenarios	7
3.1 ตัวอย่างการใช้งาน scenarios	7
3.2 เงื่อนไขการใช้งานระบบ	7
บทที่ 4 การทำงานของระบบ	9
4.1 Finite State Machine (FSM)	9
4.2 Fritzing	9
บทที่ 5 อุปกรณ์และเทคโนโลยีที่เกี่ยวข้อง	10
5.1 อุปกรณ์ที่เกี่ยวข้อง	10
5.1.1 บอร์ด ESP32	10
5.1.2 โมดูล RTC (Real-Time Clock)	11
5.1.3 LED	11
5.1.4 DHT22	12
5.1.5 Button	13
5.2 เทคโนโลยีที่เกี่ยวข้อง	14
5.2.1 แอปพลิเคชัน Blynk	14
5.2.2 แอปพลิเคชัน Node-RED	15

5.2.3 ไมโครคอนโทรลเลอร์ Arduino	16
บทที่ 6 ผลลัพธ์ของโครงการ	17
6.1 การต่อบอร์ดจริง	17
6.2 Code สำหรับอัปโหลดโปรแกรมไปยังบอร์ด ESP32	19
6.3 การทำงานของ Flow Node-RED	23
6.4 การทำงานของ Blynk	33
6.5 สรุปผลการศึกษา	34
อ้างอิง	35

บทที่ 1 บทนำ

บทนำ

ในยุคปัจจุบันเทคโนโลยี Internet of Things (IoT) ได้เข้ามามีบทบาทสำคัญในการพัฒนาระบบต่าง ๆ ที่ช่วยอำนวย ความสะดวกและยกระดับคุณภาพชีวิตของผู้คน โดยเฉพาะในด้านการดูแลสุขภาพ สำหรับผู้ป่วยที่ต้องอยู่ตามลำพัง การติดตาม อุณหภูมิร่างกายถือเป็นเรื่องที่มีความสำคัญ เนื่องจากการเปลี่ยนแปลงของอุณหภูมิอาจเป็นสัญญาณที่ปรับชี้สีภาวะสุขภาพที่ อันตราย เช่น การเป็นไข้สูง ซึ่งอาจนำไปสู่ภาวะหมดสติหรือ死อก

โครงการนี้จึงมีวัตถุประสงค์เพื่อพัฒนาและติดตั้งวงจร IoT ที่สามารถช่วยวัดอุณหภูมิของผู้ป่วยที่อยู่ตามลำพัง โดยใช้เซนเซอร์ DHT22 เพื่อวัดอุณหภูมิ ร่วมกับบอร์ด ESP32 ที่ทำหน้าที่ในการประมวลผลและส่งข้อมูลไปยังระบบ ตรวจสอบอุณหภูมิ ของผู้ป่วยตลอดเวลา ในกรณีที่อุณหภูมิสูง 39 องศาเซลเซียส ซึ่งเป็นอุณหภูมิที่มีความเสี่ยงต่อภาวะ死อกหรือหมดสติ ระบบจะใช้ LED แสดงสัญญาณเตือนภัยให้กับผู้ที่อยู่ใกล้เคียง และใช้โมดูล RTC ในการจับเวลาที่แม่นยำเพื่อให้การตรวจจับการเปลี่ยนแปลง อุณหภูมิเป็นไปอย่างมีประสิทธิภาพ

โครงการนี้สามารถช่วยเพิ่มความปลอดภัยให้กับผู้ป่วยที่ต้องอยู่ตามลำพัง โดยสามารถแจ้งเตือนและดำเนินการตาม ขั้นตอนที่เหมาะสมเมื่อเกิดภาวะอุณหภูมิสูงที่อันตราย ทำให้การดูแลผู้ป่วยที่อยู่ห่างไกลจากผู้ดูแลมีความปลอดภัยมากขึ้น และ เป็นการใช้เทคโนโลยีเพื่อเสริมสร้างคุณภาพชีวิตของผู้ที่ต้องการการดูแลสุขภาพอย่างต่อเนื่องในรูปแบบที่สะดวกและมี ประสิทธิภาพ

จัดทำโดย

น.ส. ปวัลสินทร์ กมลจิริยะฉัตร

นาย สิริธีร์ กลินบัว

1.1 ความเป็นมาและความสำคัญ

ในปัจจุบันเทคโนโลยี Internet of Things (IoT) ได้รับการพัฒนาอย่างรวดเร็วและมีการนำมาใช้ในหลายด้าน รวมถึง การดูแลสุขภาพ การติดตามอุณหภูมิร่างกายของผู้ป่วยเป็นสิ่งที่สำคัญ เนื่องจากการมีอุณหภูมิร่างกายที่สูงเกินไปอาจนำไปสู่ภาวะไข้สูง ซึ่งสามารถเป็นอันตรายถึงชีวิตได้ โดยเฉพาะในผู้ป่วยที่ต้องอยู่ท่ามลำพังหรือไม่มีการดูแลจากผู้ใกล้ชิด การตรวจสอบ อุณหภูมิร่างกายอย่างสม่ำเสมอจะเป็นเรื่องที่จำเป็นเพื่อป้องกันเหตุการณ์ที่อาจเกิดขึ้นในกรณีที่ผู้ป่วยไม่สามารถติดต่อหรือ ตระหนักถึงอาการของตนเองได้

การพัฒนาโครงการนี้จึงมีความสำคัญเนื่องจากสามารถช่วยลดความเสี่ยงในการเกิดภาวะไข้สูงถึงระดับที่อันตราย เช่น การหมดสติหรือภาวะซึมอก ซึ่งเป็นภาวะที่เกิดจากการที่ร่างกายไม่สามารถควบคุมอุณหภูมิได้ โครงการนี้จะช่วยให้ผู้ป่วยที่ต้องอยู่ ตามลำพังสามารถได้รับการดูแลได้ดีขึ้นโดยไม่ต้องพึ่งพาการดูแลจากบุคคลอื่นตลอดเวลา

ด้วยการใช้เทคโนโลยี IoT และเซนเซอร์ที่สามารถวัดอุณหภูมิได้อย่างแม่นยำ และระบบเตือนภัยที่สามารถแจ้งเตือน บุคคลที่เกี่ยวข้องได้ทันทีเมื่อมีการเปลี่ยนแปลงที่สำคัญของอุณหภูมิ ร่างกายของผู้ป่วย ความสำคัญของโครงการนี้คือการ เสริมสร้างความปลอดภัยและความมั่นคงในการดูแลผู้ป่วยที่ต้องการการดูแลตลอดเวลา โดยเฉพาะผู้ที่อยู่ตามลำพัง ซึ่งมักจะ ประสบปัญหาการขาดการดูแลจากผู้ช่วยหรือเจ้าหน้าที่ทางการแพทย์

1.2 วัตถุประสงค์ของโครงการ

โครงการนี้มีวัตถุประสงค์หลักในการพัฒนาและออกแบบระบบ IoT ที่ใช้ในการติดตามและวัดอุณหภูมิของผู้ป่วยที่ต้องอยู่ตามลำพัง เพื่อช่วยในการดูแลและตรวจสอบสุขภาพของผู้ป่วยในกรณีที่ไม่สามารถติดต่อหรือดูแลได้โดยตรงจากบุคคลอื่น ระบบนี้จะใช้เซนเซอร์ DHT22 ในการวัดอุณหภูมิร่างกายของผู้ป่วย โดยอาศัยบอร์ด ESP32 ในการประมวลผลข้อมูลและเชื่อมต่อระบบต่าง ๆ ซึ่งจะช่วยให้สามารถตรวจจับอุณหภูมิที่สูงผิดปกติได้ทันที และสามารถเตือนภัยเมื่ออุณหภูมิร่างกายของผู้ป่วยสูงถึง 39 องศาเซลเซียส ซึ่งเป็นสัญญาณที่อาจนำไปสู่ภาวะช็อกหรือหมดสติ

อีกทั้งยังมีการใช้โมดูล RTC (Real-Time Clock) เพื่อช่วยในการจับเวลาที่แม่นยำและกำหนดเวลาการตรวจจับอุณหภูมิที่เหมาะสม เพื่อให้ระบบทำงานได้อย่างมีประสิทธิภาพและทันเวลา รวมถึงมีปุ่มที่สามารถกดหยุดจับเวลาได้ทุกเมื่อ ในกรณีที่พบว่าอุณหภูมิร่างกายของผู้ป่วยเกินค่าที่กำหนด ระบบจะส่งสัญญาณเตือนผ่าน LED ที่สามารถมองเห็นได้จากบุคคลรอบข้าง ซึ่งจะเป็นการเตือนภัยเบื้องต้นและสามารถช่วยให้ผู้ป่วยได้รับการดูแลหรือช่วยเหลือได้ทันที

วัตถุประสงค์ของโครงการนี้จึงไม่เพียงแต่เพื่อติดตามสุขภาพของผู้ป่วยที่ต้องอยู่ตามลำพังเท่านั้น แต่ยังมุ่งหวังที่จะช่วยเพิ่มความปลอดภัยให้กับผู้ป่วยในกรณีฉุกเฉิน โดยสามารถจับการเปลี่ยนแปลงของอุณหภูมิได้อย่างรวดเร็วและมีการเตือนภัยที่ชัดเจน ซึ่งจะทำให้ผู้ป่วยได้รับการดูแลอย่างทันท่วงที และลดความเสี่ยงจากการเกิดภาวะช็อกหรือหมดสติจากอุณหภูมิที่สูงเกินไปได้อย่างมีประสิทธิภาพ

1.3 ขอบเขตของโครงงาน

โครงงานนี้มีขอบเขตในการพัฒนาระบบ IoT เพื่อวัดอุณหภูมิของผู้ป่วยที่ต้องอยู่ตามลำพัง โดยมีการใช้เทคโนโลยีเซ็นเซอร์และบอร์ด ESP32 เพื่อให้ระบบสามารถตรวจสอบอุณหภูมิของผู้ป่วยและแสดงผลผ่านการเตือนภัยอัตโนมัติ เมื่ออุณหภูมิร่างกายของผู้ป่วยสูงถึงค่าที่กำหนด ซึ่งระบบจะทำงานในลักษณะดังนี้:

1. การวัดอุณหภูมิ: ใช้เซ็นเซอร์ DHT22 ในการวัดอุณหภูมิร่างกายของผู้ป่วย ซึ่งเซ็นเซอร์นี้สามารถวัดอุณหภูมิอย่างแม่นยำ
2. การประมวลผลข้อมูล: ใช้บอร์ด ESP32 เพื่อประมวลผลข้อมูลที่ได้รับจากเซ็นเซอร์ DHT22 และตรวจสอบว่าอุณหภูมิร่างกายของผู้ป่วยมีค่ามากกว่าหรือเท่ากับ 39 องศาเซลเซียส ซึ่งถือว่าเป็นอุณหภูมิที่อาจเสี่ยงต่อภาวะช็อกหรือหมดสติ
3. การแสดงสัญญาณเตือนภัย: เมื่ออุณหภูมิของผู้ป่วยเกินค่าที่กำหนด ระบบจะใช้ LED ในการแสดงสัญญาณเตือนภัยให้บุคคลใกล้เคียงสามารถสังเกตเห็นได้ง่าย
4. การใช้เม็ดเวลา RTC: ระบบจะใช้เม็ดเวลา RTC (Real-Time Clock) เพื่อช่วยในการจับเวลาและควบคุมการทำงานของระบบให้สามารถตรวจจับอุณหภูมิในเวลาที่เหมาะสมและแม่นยำ
5. การแจ้งเตือน: ระบบจะทำการแจ้งเตือนเมื่ออุณหภูมิของผู้ป่วยถึงค่าที่เสี่ยง ซึ่งสามารถให้ข้อมูลเกี่ยวกับความจำเป็นในการให้การดูแลหรือการรักษาทันที
6. การหยุดจับเวลา: ผู้ใช้สามารถกดปุ่มเพื่อยุดการจับเวลา เมื่อส่งผู้ป่วยถึงมือแพทย์

ขอบเขตที่ไม่รวมอยู่ในโครงงาน:

- ระบบจะไม่เชื่อมต่อกับระบบเครือข่ายภายนอกหรือแอปพลิเคชันบนมือถือ เนื่องจากไม่ได้รวมฟังก์ชัน การแจ้งเตือนผ่านโทรศัพท์มือถือในโครงงานนี้
- ระบบไม่รวมถึงการใช้เซ็นเซอร์หรืออุปกรณ์อื่น ๆ ที่เกี่ยวข้องกับการตรวจสอบสุขภาพประเภทอื่น เช่น การวัดความดันโลหิต หรือการตรวจวัดอัตราการเต้นของหัวใจ
- โครงงานไม่ได้มุ่งเน้นไปที่การพัฒนาระบบที่ใช้ในการตรวจจับอุณหภูมิในผู้ป่วยหลายคนพร้อมกัน แต่ จะใช้กับผู้ป่วยหนึ่งคนในแต่ละครั้ง

1.4 ประโยชน์ที่คาดว่าจะได้รับ

โครงการนี้มีประโยชน์ที่คาดว่าจะได้รับหลากหลายด้าน ทั้งในเรื่องของการพัฒนาเทคโนโลยีและการดูแลสุขภาพ รวมถึงการเพิ่มความปลอดภัยให้กับผู้ป่วยที่ต้องอยู่ตามลำพัง ดังนี้:

1. การเพิ่มความปลอดภัยให้กับผู้ป่วย: ระบบ IoT ที่พัฒนาในโครงการนี้จะช่วยให้สามารถติดตามและตรวจสอบอุณหภูมิร่างกายของผู้ป่วยที่ต้องอยู่ตามลำพังได้อย่างต่อเนื่อง โดยเฉพาะในกรณีที่อุณหภูมิของผู้ป่วยสูงเกิน 39 องศาเซลเซียส ซึ่งสื่อไปถึงต่อภาวะซึมเศร้าหรือหมดสติ ระบบจะเตือนภัยทันที ทำให้ผู้ป่วยได้รับการดูแลและช่วยเหลือได้ทันท่วงที่
2. ช่วยในการติดตามสุขภาพของผู้ป่วยที่ห่างไกล: สำหรับผู้ป่วยที่ไม่สามารถติดต่อหรือมีผู้ดูแลได้โดยตรง การใช้เทคโนโลยี IoT ทำให้สามารถติดตามสุขภาพของผู้ป่วยได้จากระยะไกล โดยไม่ต้องมีการเข้าถึงตัวผู้ป่วยทุกครั้ง ระบบนี้จะช่วยลดความเสี่ยงจากการไม่สามารถรับรู้ภาวะอันตรายในเวลาที่เหมาะสม
3. การลดความเสี่ยงจากการลักพาตัว: การตรวจสอบอุณหภูมิของผู้ป่วยอย่างสม่ำเสมอและการแจ้งเตือนเมื่ออุณหภูมิสูงเกินเกณฑ์ที่กำหนดจะช่วยป้องกันไม่ให้เกิดภาวะรุนแรงที่อาจเกิดขึ้นจากอุณหภูมิร่างกายที่สูงผิดปกติ เช่น ภาวะซึมเศร้าหรือหมดสติ ซึ่งจะช่วยลดความเสี่ยงในการเกิดอันตรายจากภาวะเหล่านี้
4. การประยุกต์ใช้เทคโนโลยี IoT ใน การดูแลสุขภาพ: โครงการนี้เป็นตัวอย่างของการนำเทคโนโลยี IoT มาใช้ในด้านการดูแลสุขภาพ ซึ่งมีแนวโน้มที่จะได้รับความนิยมและมีความสำคัญเพิ่มขึ้นในอนาคต การพัฒนาเทคโนโลยีในด้านนี้จะช่วยให้ผู้ป่วยสามารถได้รับการดูแลจากระยะไกลและช่วยให้การดูแลสุขภาพมีประสิทธิภาพมากขึ้น
5. ความสะดวกในการใช้งาน: ระบบที่พัฒนาขึ้นมาจะใช้งานได้ง่าย มีการแสดงผลการเตือนภัยที่ชัดเจน ทำให้บุคลากรรอบข้างสามารถเห็นสัญญาณเตือนและสามารถดำเนินการช่วยเหลือผู้ป่วยได้อย่างรวดเร็วและมีประสิทธิภาพ
6. การลดภาระของผู้ดูแล: สำหรับผู้ดูแลผู้ป่วยที่มีภาระมาก การใช้ระบบนี้จะช่วยลดภาระในการตรวจสอบสุขภาพผู้ป่วยทุกวัน เนื่องจากระบบจะทำงานอัตโนมัติในการติดตามอุณหภูมิและแจ้งเตือนเมื่อมีภาวะผิดปกติ ซึ่งช่วยให้ผู้ดูแลสามารถให้ความสนใจในกรณีที่สำคัญได้ทันท่วงที่
7. โอกาสในการขยายผลในอนาคต: ระบบนี้สามารถนำไปพัฒนาและขยายให้รองรับผู้ป่วยหลายคนได้ในอนาคต และสามารถเชื่อมต่อกับระบบอื่น ๆ เช่น ระบบการเตือนภัยผ่านแอปพลิเคชันมือถือหรือระบบคลาวด์ ซึ่งจะเพิ่มประสิทธิภาพในการดูแลผู้ป่วยและทำให้การตรวจสอบสุขภาพเป็นไปอย่างมีประสิทธิภาพและทั่วถึงมากขึ้น

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 IoT หรือ Internet of Things คือแนวคิดที่ทำให้อุปกรณ์ต่าง ๆ (Things) สามารถเชื่อมต่ออินเทอร์เน็ตได้ และสามารถสื่อสาร แลกเปลี่ยนข้อมูลกันได้ โดยไม่ต้องพึ่งพามนุษย์โดยตรง ประกอบด้วยองค์ประกอบหลัก 4 ส่วน ได้แก่

1. อุปกรณ์ตรวจจับ (Sensors) - อุปกรณ์ที่ใช้ในการเก็บรวบรวมข้อมูลจากสภาพแวดล้อม เช่น เซ็นเซอร์อุณหภูมิ เซ็นเซอร์ความชื้น เซ็นเซอร์แสง และในกรณีของโครงงานนี้คือเซ็นเซอร์อัลตราโซนิกที่ใช้วัดระยะห่าง
2. การเชื่อมต่อ (Connectivity) - เทคโนโลยีที่ใช้ในการส่งข้อมูลระหว่างอุปกรณ์ต่างๆ และระบบคลาวด์ เช่น Wi-Fi, Bluetooth, ZigBee, หรือ Cellular (3G/4G/5G)
3. การประมวลผลข้อมูล (Data Processing) - การวิเคราะห์และประมวลผลข้อมูลที่ได้จากอุปกรณ์ตรวจจับ เพื่อนำไปใช้ในการควบคุมหรือตัดสินใจ
4. ส่วนติดต่อผู้ใช้งาน (User Interface) - ส่วนที่ใช้ในการแสดงผลข้อมูลและควบคุมอุปกรณ์ IoT เช่น แอปพลิเคชันบนสมาร์ทโฟน หรือเว็บแอปพลิเคชัน

ในโครงงานนี้ นำแนวคิด IoT มาประยุกต์ใช้ในการพัฒนาระบบตรวจสอบอุณหภูมิร่างกายและแจ้งเตือน ใช้ RTC ในการจับเวลา และใช้การเชื่อมต่อ Wi-Fi ผ่านบอร์ด ESP32 เพื่อส่งข้อมูลไปยังแอปพลิเคชัน Blynk ซึ่งทำหน้าที่เป็นส่วนติดต่อผู้ใช้งาน

2.1.2 การประมวลผลแบบเรียลไทม์ (Real-time Processing)

กระบวนการประมวลผลข้อมูลที่ระบบสามารถรับข้อมูลและตอบสนองกลับมาได้ภายในเวลาที่กำหนด โดยข้อมูลที่ได้รับจะถูกประมวลผลในทันทีเพื่อให้ได้ผลลัพธ์ที่สอดคล้องกับข้อกำหนดด้านเวลา (Time Constraint) ของระบบนั้น ๆ

ระบบที่ใช้การประมวลผลแบบเรียลไทม์จำเป็นต้องมีความแม่นยำและรวดเร็วในการตอบสนองต่อเหตุการณ์ต่างๆ เพื่อให้มั่นใจว่าการทำงานยังคงถูกต้องแม่นในสถานการณ์ที่มีการเปลี่ยนแปลงอย่างรวดเร็ว

การประมวลผลแบบเรียลไทม์ในโครงงานนี้ประกอบด้วย:

1. การประมวลผลและตัดสินใจ - บอร์ด ESP32 นำข้อมูลที่ตรวจวัดได้มาประมวลผลและตัดสินใจว่าควรให้ LED ดับ หรือ สว่าง
2. การแสดงผลและส่งข้อมูล - ข้อมูลระยะห่างและค่าอุณหภูมิจะถูกส่งไปยังแอปพลิเคชัน Blynk แบบเรียลไทม์ ทำให้ผู้ใช้งานสามารถตรวจสอบสถานะได้ตลอดเวลา

บทที่ 3

ตัวอย่างการใช้งาน scenarios

3.1 ตัวอย่างการใช้งาน scenarios

Scenario : การแจ้งเตือนอัตโนมัติเมื่อผู้ป่วยมีอุณหภูมิร่างกายสูงเกินค่าที่กำหนด

การทำงานของระบบ

ระบบนี้ถูกออกแบบมาเพื่อติดตามอาการผู้ป่วยที่อยู่บ้านคนเดียว โดยเฉพาะในกรณีที่เสี่ยงภาวะซึ่งก่อจากไข้สูง เช่น ในช่วงการระบาดของโรค COVID-19 หรือภาวะไข้จากการติดเชื้ออื่น ๆ

เซ็นเซอร์ DHT22 จะทำหน้าที่วัดอุณหภูมิและความชื้นร่างกายแบบต่อเนื่อง หากตรวจพบว่าอุณหภูมิสูงตั้งแต่ 39.0°C ขึ้นไป ระบบจะเข้าสู่โหมดเฝ้าระวังโดยอัตโนมัติ ซึ่งประกอบด้วย:

- แจ้งเตือนผ่าน LED: ไฟ LED สีแดงจะติดขึ้นทันที เพื่อแสดงว่าเกิดภาวะเสี่ยง
- เริ่มจับเวลาอัตโนมัติ: ใช้โมดูล RTC (Real Time Clock) ในการจับเวลา เพื่อรับ��ว่าการไข้สูงดำเนินมากวินาทีหรือกี่นาทีแล้ว
- ส่งข้อมูลขึ้น Blynk Cloud: ข้อมูลอุณหภูมิ ความชื้น และระยะเวลาจะถูกส่งแบบเรียลไทม์ไปยังแอป Blynk บนมือถือ ทำให้ญาติหรือแพทย์สามารถติดตามสถานะได้ตลอดเวลา
- สามารถกดปุ่มหยุดจับเวลา: เมื่อผู้ป่วยได้รับการช่วยเหลือแล้ว เช่น ถูกส่งตัวไปถึงโรงพยาบาล ญาติสามารถกดปุ่มบน อุปกรณ์เพื่อยุดจับเวลาและปิดการแจ้งเตือน

3.2 เงื่อนไขการใช้งานระบบ

1. การใช้ปุ่มกด (Push Button) เพื่อควบคุมการหยุดเวลา

ปุ่มกดเชื่อมต่อกับ ESP32 ที่ GPIO18 ทำหน้าที่สั่งหยุดการจับเวลาโดยการเปลี่ยนสถานะตัวแปรควบคุมในโปรแกรม การออกแบบนี้ช่วยให้สามารถจัดการสถานการณ์ฉุกเฉินได้ง่าย ไม่ต้องรีเซ็ตระบบทั้งหมด

2. การใช้ RTC (Real Time Clock) สำหรับจับเวลาแม่นยำ

ระบบใช้โมดูล RTC (DS3231) เพื่อจับเวลาว่าผู้ป่วยมีไข้สูงอยู่นานแค่ไหน โดยไม่ต้องพึ่ง Delay() ในโปรแกรม ทำให้สามารถตรวจวัดและส่งค่าทุกอย่างขึ้น Blynk ได้ตลอดเวลาแบบไม่มีการหน่วง

3. การใช้ BlynkTimer แทน delay()

เพื่อให้ระบบไม่ค้างและทำงานหลายอย่างพร้อมกัน (Multitask) เช่น ตรวจสอบอุณหภูมิ ตรวจจับการกดปุ่ม และอัปเดตข้อมูล Blynk ได้แบบเรียลไทม์ จึงเลือกใช้ BlynkTimer ซึ่งเป็นการทำงานแบบ Non-blocking

4. WiFi ต้องเสถียรตลอดเวลาเมื่อต้องการใช้ร่วมกับ Blynk

การเชื่อมต่อ Blynk จำเป็นต้องใช้ WiFi ที่มีความเสถียร หากอินเทอร์เน็ตขาดหาย ระบบจะไม่สามารถอัปโหลดข้อมูลหรือแสดงผลผ่าน Dashboard ได้ แม้ว่าอุปกรณ์ภายใน (DHT, RTC, บุ่ม) จะยังทำงานต่อได้ก็ตาม ดังนั้นการใช้งานจริงควรเลือก WiFi ที่มั่นคง หรือพัฒนาโดยตรง Offline เพิ่มเติมหากจำเป็น

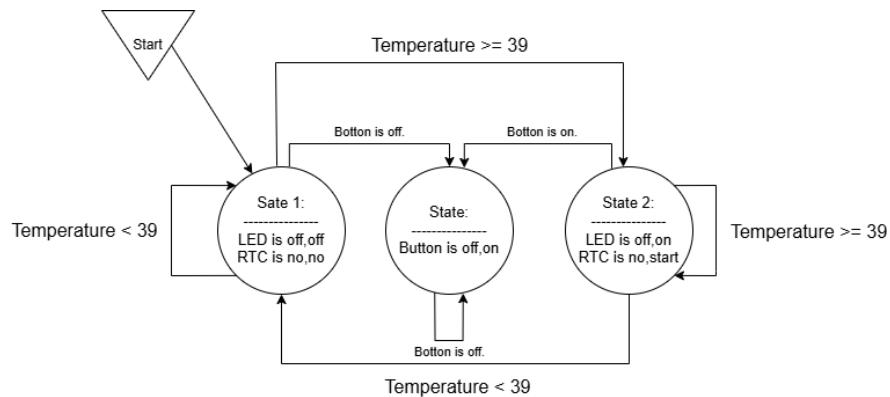
5. ข้อควรระวังด้านความปลอดภัย

ควรตรวจสอบการเชื่อมสายไฟของ DHT22, RTC และบุ่มให้แน่นหนา ไม่หลวม หรือขาดช่วง เพราะอาจทำให้ค่าที่วัดได้ผิดพลาด หรือระบบทำงานไม่สมบูรณ์

บทที่ 4

การทำงานของระบบ

4.1 Finite State Machine (FSM)

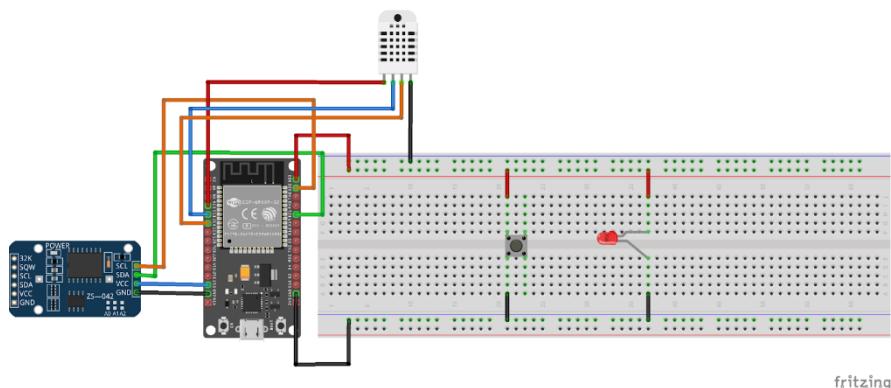


การทำงานของระบบ

เมื่อระบบเริ่มทำงาน อันดับแรกจะตรวจสอบอุณหภูมิหากอุณหภูมิสูงถึง 39 องศา LED แต่เดิมดับจะสว่าง และ RTC จะเริ่มจับเวลา หากต้องการหยุดจับเวลาสามารถกดปุ่มได้ แต่ถ้าหากอุณหภูมิปกติ คือน้อยกว่า 39 องศา LED และ RTC จะไม่มี

ปฏิกิริยาใดๆ LED จะยังคงดับอยู่ และ RTC จะไม่จับเวลา

4.2 Fritzing



ภาพแสดงการเชื่อมต่อวงจรอิเล็กทรอนิกส์บนโปรแกรม Fritzing โดยมีการเชื่อมต่อระหว่างบอร์ด ESP32, โมดูล RTC ZS-042 และเซ็นเซอร์ DHT บนบอร์ดทดลอง (Breadboard) การเชื่อมต่อหลักประกอบด้วยสายไฟหลายสี โดยมีการต่อ I2C ระหว่าง ESP32 กับโมดูล RTC (สาย SDA สีเขียว และ SCL สีฟ้า) และการเชื่อมต่อไฟ (VCC สีแดง และ GND สีดำ) ระหว่าง อุปกรณ์ทั้งหมด

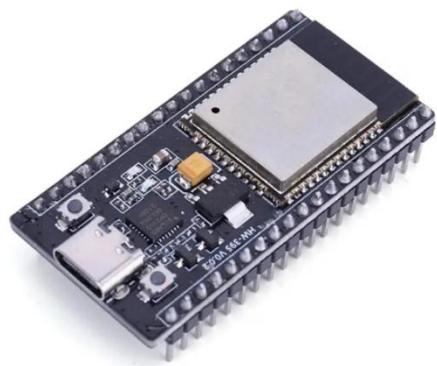
นอกจากนี้ยังมีเซ็นเซอร์ DHT ที่เชื่อมต่อกับ ESP32 ผ่านสายสัญญาณสีส้ม และมี LED พร้อมปุ่มบนบอร์ดทดลอง วงจรนี้ น่าจะเป็นระบบสำหรับการวัดอุณหภูมิ พร้อมการแสดงเวลาและบันทึกข้อมูล รวมถึงสามารถกดปุ่มเมื่อต้องการหยุดจับเวลา

บทที่ 5

อุปกรณ์และเทคโนโลยีที่เกี่ยวข้อง

5.1 อุปกรณ์ที่เกี่ยวข้อง

5.1.1 บอร์ด ESP32



ESP32 คือไมโครคอนโทรลเลอร์ที่พัฒนาโดยบริษัท Espressif Systems ซึ่งมีคุณสมบัติที่หลากหลายและเหมาะสมสำหรับการพัฒนาโครงการที่ต้องการการเชื่อมต่อเครือข่าย เช่น Wi-Fi และ Bluetooth ในตัว

คุณสมบัติหลักของ ESP32

1. Dual-core processor - ใช้หน่วยประมวลผลแบบ dual-core ที่ทำให้สามารถประมวลผลได้รวดเร็วและมีประสิทธิภาพ
2. Wi-Fi และ Bluetooth - รองรับการเชื่อมต่อ Wi-Fi และ Bluetooth ทั้งแบบ classic และ BLE (Bluetooth Low Energy)
3. GPIO pins - มีขา GPIO ที่สามารถใช้งานได้หลายรูปแบบ เช่น การอ่านค่า Analog, PWM, I2C, SPI
4. ประสิทธิภาพสูง - สามารถทำงานได้ที่ความถี่สูงถึง 240 MHz
5. ความสามารถในการประยัดพลังงาน - มีเหมดการประยัดพลังงานที่ช่วยให้การใช้งานในระยะยาวเป็นไปได้อย่างมีประสิทธิภาพ

5.1.2 โมดูล RTC (Real-Time Clock)



RTC (Real-Time Clock) คือวงจรอิเล็กทรอนิกส์ที่ใช้ในการติดตามเวลาและวันที่ในระบบคอมพิวเตอร์หรืออุปกรณ์อิเล็กทรอนิกส์ต่าง ๆ โดย RTC จะทำหน้าที่ในการเก็บข้อมูลเวลาอย่างต่อเนื่อง แม้เมื่ออุปกรณ์ถูกปิดหรือไม่มีการจ่ายไฟ

คุณสมบัติหลักของ RTC

1. การเก็บข้อมูลเวลา: RTC จะเก็บข้อมูลเวลา เช่น ชั่วโมง, นาที, วินาที, วัน, เดือน, ปี เป็นต้น
2. ความแม่นยำสูง: RTC มีความแม่นยำสูงในการติดตามเวลา
3. ใช้แบตเตอรี่: RTC มากจะใช้แบตเตอรี่สำรอง (เช่น CR2032) เพื่อให้มีการทำงานได้แม้เมื่ออุปกรณ์ไม่ได้เปิดหรือไม่มีแหล่งจ่ายไฟ
4. การใช้งานทั่วไป: RTC ถูกใช้ในระบบต่าง ๆ เช่น คอมพิวเตอร์, ไมโครคอนโทรลเลอร์ (เช่น ESP32), อุปกรณ์ IoT, นาฬิกาดิจิตอล, และการตั้งเวลาต่าง ๆ ในระบบอิเล็กทรอนิกส์

5.1.3 LED

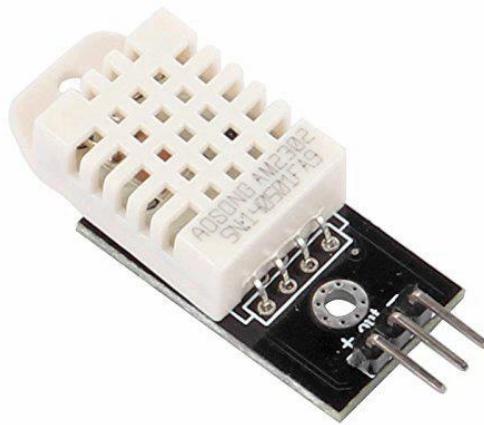


LED (Light Emitting Diode) คือ ไดโอดที่สามารถปล่อยแสงออกมามาได้เมื่อมีการจ่ายกระแสไฟฟ้า โดยที่ไม่ต้องใช้หลอดไฟหรือสารเรืองแสงอื่น ๆ ซึ่งทำให้ LED เป็นเทคโนโลยีที่มีประสิทธิภาพสูงและประหยัดพลังงาน

คุณสมบัติหลักของ LED

1. ประหยัดพลังงาน: LED ใช้พลังงานน้อยกว่าหลอดไฟแบบเก่า เช่น หลอดไส้หรือหลอดฟลูออเรสเซนต์
2. อายุการใช้งานยาวนาน: LED มีอายุการใช้งานที่ยาวนานกว่าหลอดไฟชนิดอื่น โดยสามารถใช้งานได้หลายหมื่นชั่วโมง
3. ความทนทาน: LED ทนทานต่อแรงกระแทกและการสั่นสะเทือน ทำให้เหมาะสมกับการใช้งานในสภาพแวดล้อมที่รุนแรง
4. การควบคุมแสง: สามารถควบคุมความสว่างของแสงได้ง่าย เช่น การปรับความสว่าง (dimming) โดยไม่ต้องใช้วิธีการที่ซับซ้อน
5. อุณหภูมิการทำงานต่ำ: LED จะไม่ร้อนเหมือนหลอดไฟทั่วไป ทำให้สามารถใช้งานในพื้นที่ที่ไม่สามารถใช้หลอดไฟที่มีอุณหภูมิสูงได้

5.1.4 DHT22



DHT22 คือเซ็นเซอร์วัดอุณหภูมิและความชื้นที่ใช้ในโปรเจกต์ IoT หรือไมโครคอนโทรลเลอร์ต่างๆ โดยเฉพาะกับอุปกรณ์อย่างเช่น ESP32 หรือ Arduino. เซ็นเซอร์ DHT22 มีความสามารถในการวัดอุณหภูมิและความชื้นในอากาศได้พร้อมกัน

คุณสมบัติหลักของ DHT22

1. วัดอุณหภูมิและความชื้น: DHT22 สามารถวัดทั้งอุณหภูมิ (จาก -40°C ถึง +80°C) และความชื้น (จาก 0% ถึง 100% RH) โดยให้ข้อมูลออกมาในรูปแบบดิจิตอล

2. ความแม่นยำ:

- อุณหภูมิ: $\pm 0.5^{\circ}\text{C}$

- ความชื้น: $\pm 2\text{-}5\% \text{ RH}$

3. การใช้งานง่าย: ใช้งานง่ายกับไมโครคอนโทรลเลอร์ เช่น Arduino หรือ ESP32 ผ่านการเชื่อมต่อที่ใช้ขาเดียว (Single-Wire Communication)

4. การใช้พลังงานต่ำ: ใช้พลังงานต่ำ ทำให้สามารถใช้งานในโครงการที่ต้องการประหยัดพลังงาน

5. ราคาไม่แพง: DHT22 เป็นเซ็นเซอร์ที่มีราคาค่อนข้างถูกและเหมาะสมกับการทดลองหรือโครงการ DIY

5.1.5 Button



ปุ่มกดแบบ Tactile Switch

เป็นปุ่มสวิตซ์ขนาดเล็ก ที่เมื่อคุณกดลงไป มันจะเข้าม่วงไฟฟ้า (ON) และเมื่อปล่อยมือ มันจะตัว回去 (OFF) มากใช้บนบอร์ดวงจรอิเล็กทรอนิกส์ เช่น Arduino, Raspberry Pi, หรืองาน prototyping อื่น ๆ

โครงสร้าง:

1. มีขาสำหรับต่อ กับแผ่นวงจร (PCB) จำนวน 4 ขา (บางแบบอาจใช้แค่ 2 ขา)
2. ด้านบนมีปุ่มยางหรือพลาสติกสำหรับกด
3. เมื่อกด ข้างในจะมีการสัมผัสกันของหน้าคอนแทค (contact) ที่อยู่ภายใน

5.2 เทคโนโลยีที่เกี่ยวข้อง

5.2.1 แอปพลิเคชัน Blynk

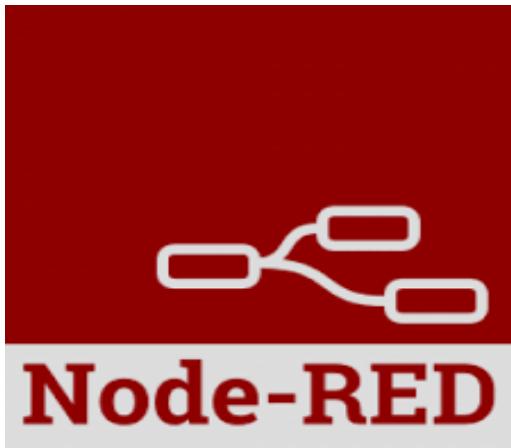


Blynk คือแพลตฟอร์มที่ช่วยให้ผู้ใช้งานสามารถสร้างแอปพลิเคชัน IoT (Internet of Things) ได้ง่ายและรวดเร็ว โดยไม่ต้องเขียนโค้ดซับซ้อน ใช้การเชื่อมต่ออุปกรณ์ต่างๆ ผ่านอินเทอร์เน็ตเพื่อควบคุมหรือเก็บข้อมูลจากเซ็นเซอร์ต่างๆ จากระยะไกล

คุณสมบัติหลักของ Blynk

1. ใช้งานง่าย: Blynk มาพร้อมกับแอปพลิเคชันมือถือที่ใช้งานง่าย สามารถควบคุมอุปกรณ์ IoT ได้จากスマาร์ทโฟนหรือแท็บเล็ต
2. รองรับหลายอุปกรณ์: Blynk สามารถทำงานร่วมกับไมโครคอนโทรลเลอร์ต่างๆ เช่น Arduino, ESP32, Raspberry Pi และอื่นๆ
3. อินเตอร์เฟซแบบลากและวาง (Drag-and-Drop): ผู้ใช้งานสามารถสร้างแอปพลิเคชัน IoT ด้วยการลากและวางวิดเจ็ตต่างๆ (เช่น ปุ่ม, ตัวเลื่อน, กราฟ) บนแอป Blynk เพื่อเชื่อมต่อและควบคุมอุปกรณ์
4. การเชื่อมต่อผ่าน Wi-Fi, Bluetooth, หรือ Ethernet: รองรับการเชื่อมต่อที่หลากหลายทั้ง Wi-Fi, Bluetooth, และ Ethernet เพื่อให้เหมาะสมกับโปรเจกต์ต่างๆ
5. การแจ้งเตือนและการควบคุมแบบเรียลไทม์: Blynk รองรับการส่งการแจ้งเตือน (Push Notifications) และการควบคุมจากระยะไกลแบบเรียลไทม์
6. สามารถเก็บข้อมูล: Blynk สามารถเก็บข้อมูลจากเซ็นเซอร์และแสดงผลในรูปแบบกราฟหรือแผนภูมิบันรอบ
7. รองรับการใช้โค้ด: เมื่อจะเป็นแพลตฟอร์มที่เน้นความง่ายในการใช้งาน แต่ Blynk ก็ยังรองรับการเขียนโค้ดเพิ่มเติมในโปรเจกต์ของคุณได้ด้วย

5.2.2 แอปพลิเคชัน Node-RED



Node-RED คือเครื่องมือพัฒนาแบบโอเพนซอร์สที่ใช้ในการสร้างแอปพลิเคชัน IoT (Internet of Things) ด้วยการลากและวางโนด (Nodes) ที่แทนฟังก์ชันต่างๆ บนเว็บเบราว์เซอร์ ช่วยให้ผู้ใช้งานสามารถพัฒนาแอปพลิเคชันได้ง่ายโดยไม่ต้องเขียนโค้ดจำนวนมาก

คุณสมบัติหลักของ Node-RED

1. **อินเทอร์เฟซใช้งานง่าย:** Node-RED มีระบบลากและวาง (Drag-and-Drop) ในรูปแบบของ Flow-based Programming ทำให้สามารถสร้างระบบที่ซับซ้อนได้โดยไม่ต้องเขียนโค้ดยาก
2. **รองรับหลากหลายอุปกรณ์:** สามารถเชื่อมต่อกับอุปกรณ์ IoT หลากหลาย เช่น ESP32, Raspberry Pi, Arduino และอื่นๆ
3. **รองรับหลายโปรโตคอล:** รองรับการเชื่อมต่อผ่านโปรโตคอลต่างๆ เช่น MQTT, HTTP, TCP, UDP และ WebSocket
4. **สามารถประมวลผลข้อมูล:** สามารถแปลง วิเคราะห์ หรือจัดการข้อมูลจากเซ็นเซอร์ได้ เช่น การฟิลเตอร์ข้อมูล การคำนวน หรือการจัดเก็บลงฐานข้อมูล
5. **แสดงข้อมูลผ่าน Dashboard:** Node-RED มีปลั๊กอิน Dashboard ที่ช่วยแสดงผลข้อมูลจากอุปกรณ์ในรูปแบบกราฟ, เกจ, ปุ่มควบคุม ฯลฯ
6. **ใช้งานบนหลายแพลตฟอร์ม:** Node-RED สามารถติดตั้งได้บนหลายระบบปฏิบัติการ เช่น Windows, Linux, macOS รวมถึง Raspberry Pi
7. **ขยายความสามารถได้ง่าย:** ผู้ใช้งานสามารถเพิ่มโนดใหม่จาก Node-RED Library ที่มีชุมชนพัฒนาอย่างต่อเนื่อง
8. **เน้นภัยภัยการทดลองและพัฒนา:** เหมาะสำหรับนักพัฒนา นักเรียน และผู้เริ่มต้นที่ต้องการทดสอบระบบ IoT แบบรวดเร็ว

5.2.3 ไมโครคอนโทรลเลอร์ Arduino



Arduino คือแพลตฟอร์มโอเพนซอร์สที่ใช้สำหรับการพัฒนาอุปกรณ์อิเล็กทรอนิกส์แบบฝังตัว (Embedded Systems) โดยมีทั้งฮาร์ดแวร์และซอฟต์แวร์ที่ออกแบบมาให้เรียนรู้และใช้งานได้ง่าย เหมาะสำหรับผู้เริ่มต้นไปจนถึงผู้เชี่ยวชาญในการสร้างโครงการ IoT และหุ่นยนต์ต่างๆ

คุณสมบัติหลักของ Arduino

- ใช้งานง่ายสำหรับผู้เริ่มต้น: มีตัวอย่างโค้ดและไลบรารีมากมายที่ช่วยให้เริ่มต้นเขียนโปรแกรมได้ทันที
- โอเพนซอร์สและต้นทุนต่ำ: มีบอร์ดให้เลือกใช้งานหลากหลายรุ่นในราคาย่อมเยา เช่น Arduino Uno, Nano, Mega
- พัฒนาโดยใช้ภาษา C/C++: ใช้ Arduino IDE ที่รองรับการเขียนโปรแกรมแบบง่าย พร้อมการอัปโหลดโค้ดลงบอร์ดโดยตรง
- รองรับเซ็นเซอร์และไมค์จำนวนมาก: สามารถเชื่อมต่อกับอุปกรณ์ต่างๆ ได้ เช่น เซ็นเซอร์วัดอุณหภูมิ แสง ความชื้น โมเตอร์ ฯลฯ
- เชื่อมต่อกับอุปกรณ์ IoT ได้: สามารถใช้งานร่วมกับโมดูล Wi-Fi (เช่น ESP8266) หรือ Bluetooth เพื่อส่งข้อมูลไปยังแพลตฟอร์มต่างๆ เช่น Blynk หรือ Node-RED
- มีชุมชนผู้ใช้งานทั่วโลก: มีแหล่งความรู้ บทเรียน และตัวอย่างโปรเจกต์มากมายที่ช่วยในการเรียนรู้และพัฒนา
- เหมาะสมสำหรับการทดลองและต้นแบบ (Prototyping): นิยมใช้ในการทำโปรเจกต์ต้นแบบหรือการทดลองในห้องเรียน และงานวิจัย

บทที่ 6

ผลลัพธ์ของโครงการ

จากการศึกษาและพัฒนาระบบตรวจสอบสุขภาพผู้ป่วยที่บ้าน โดยใช้บอร์ด ESP32 เป็นแกนหลักในการรับข้อมูลจากเซ็นเซอร์ DHT22 สำหรับวัดอุณหภูมิและความชื้น และโมดูล RTC (Real-Time Clock) สำหรับจับเวลาการเกิดไข้ พบร่วมระบบสามารถทำงานร่วมกับอุปกรณ์ต่าง ๆ ได้อย่างมีประสิทธิภาพ โดยเมื่อค่าที่วัดได้มีอุณหภูมิเกิน 39 องศาเซลเซียส ระบบจะเริ่มจับเวลาและแสดงสถานะผ่านไฟ LED และเมื่อมีการกดปุ่มจากผู้ป่วยหรือผู้ดูแล ระบบจะหยุดจับเวลาและบันทึกระยะเวลาที่มีใช้ไว้เพื่อใช้วิเคราะห์ในภายหลัง

ระบบใช้การสื่อสารแบบ MQTT เพื่อส่งข้อมูลระหว่าง ESP32 และ Node-RED โดยใน Node-RED จะมีการประมวลผลข้อมูล และส่งค่าไปยังแอปพลิเคชัน Blynk ผ่าน Virtual Pin (V1-V3) เพื่อแสดงผลค่าต่าง ๆ ได้แก่ อุณหภูมิ, ความชื้น และระยะเวลาที่อุณหภูมิสูง ซึ่งสามารถดูผ่านแอป Blynk ได้จากทุกที่ที่มีอินเทอร์เน็ต ทำให้ญาติหรือผู้ดูแลสามารถเฝ้าระวังอาการของผู้ป่วยได้แบบเรียลไทม์แม้อยู่ห่างไกล

จากการทดลอง พบร่วมระบบสามารถวัดค่าได้อย่างแม่นยำ และส่งข้อมูลผ่าน MQTT ได้เสถียร การแสดงผลบน Blynk มีความชัดเจนและตอบสนองได้ดีตามสถานการณ์จริง นอกจากนี้ยังสามารถขยายระบบเพิ่มเติมในอนาคต เช่น การแจ้งเตือนอัตโนมัติเมื่อผู้ป่วยมีไข้ หรือการบันทึกประวัติสุขภาพในระยะยาวได้อีกด้วย ซึ่งแสดงให้เห็นว่าระบบนี้มีศักยภาพในการนำไปใช้งานจริงและพัฒนาในเชิงสาธารณะต่อไปได้

6.1 การต่อบอร์ดจริง

ในการต่อบอร์ด ESP32 กับอุปกรณ์ต่าง ๆ สำหรับโครงงานระบบตรวจสอบสุขภาพผู้ป่วยที่บ้าน คณะกรรมการได้เขียนรายละเอียดและส่วนควบคุมเข้ากับบอร์ด ESP32 ดังนี้

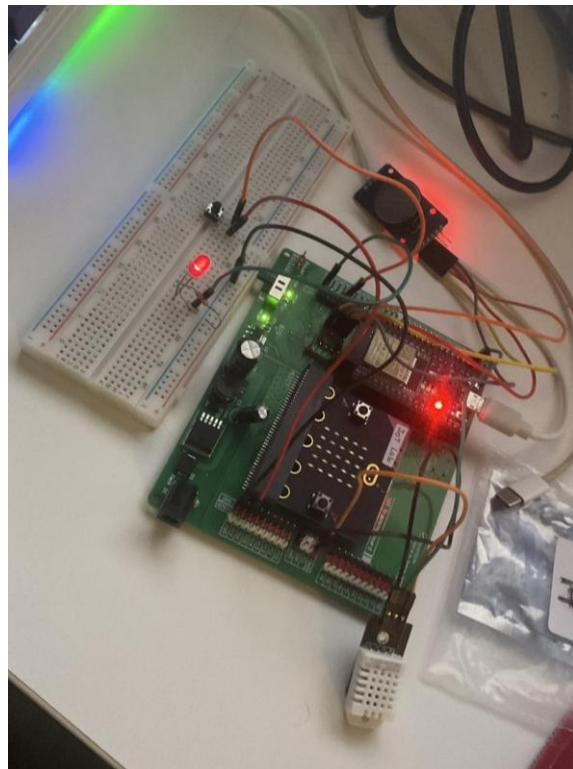
- เซ็นเซอร์ DHT22 สำหรับวัดอุณหภูมิและความชื้น
- RTC สำหรับบันทึกเวลาเมื่ออุณหภูมิเกินค่าที่กำหนด
- LED สีแดงใช้แสดงสถานะเมื่อเกิดอุณหภูมิสูง
- Push Button สำหรับหยุดการนับเวลาเมื่อผู้ป่วยกดตوبสนอง

การเขียนโปรแกรมทั้งหมดใช้การต่อสายเข้ากับ GPIO บนบอร์ด ESP32 ได้แก่:

- DHT22 → GPIO 27
- RTC (ผ่าน I2C) → SDA และ SCL ของ ESP32
- LED → GPIO 23 (ผ่านตัวต้านทาน)

- Button → GPIO 18 (แบบ pull-up)

วงจรทั้งหมดถูกประกอบบนเบรดบอร์ดและต่อ กับบอร์ด ESP32 ผ่านชุดทดลอง โดยมีการจัดเรียงสายไฟอย่างเป็นระเบียบ เพื่อให้สามารถตรวจสอบการทำงานได้อย่างง่ายดาย ทั้งนี้ การติดตั้งวงจรถูกออกแบบให้สามารถทดสอบข้ามได้สะดวก และเหมาะสมต่อการนำไปใช้งานจริงกับผู้ป่วยในบ้านดังภาพ



6.2 Code สำหรับอัปโหลดโปรแกรมไปยังบอร์ด ESP32

โปรแกรมนี้เขียนขึ้นเพื่อควบคุมการทำงานของระบบผ่านบอร์ด ESP32 โดยใช้เซ็นเซอร์ DHT22 เพื่อเก็บข้อมูลสภาพแวดล้อมของผู้บ่วยแบบเรียลไทม์ หากอุณหภูมิสูงกว่า 39°C ระบบจะสั่งเปิดไฟ LED เพื่อแสดงสถานะอันตราย และเริ่มจับเวลาโดยใช้โมดูล RTC DS3231 เพื่อคำนวณระยะเวลาที่อุณหภูมิสูงเกินกำหนด

นอกจากนี้ยังมีปุ่มกดสำหรับหยุดเวลาเมื่อมีการตอบสนองจากผู้บ่วย ซึ่งช่วยให้สามารถประเมินระดับความเสี่ยงหรือความช่วยเหลือที่จำเป็นได้อย่างเหมาะสม

ระบบยังรองรับการเชื่อมต่อกับเครือข่ายอินเทอร์เน็ตผ่าน Wi-Fi เพื่อส่งข้อมูลไปยัง Node-RED และแสดงผลผ่านแพลตฟอร์ม Blynk โดยใช้โปรโตคอล MQTT ทำให้สามารถตรวจสอบสถานะของผู้บ่วยจากระยะไกลได้ ทั้งในรูปแบบของตัวเลข (เช่น อุณหภูมิ, ความชื้น, ระยะเวลาที่อุณหภูมิสูง) และการแจ้งเตือนผ่าน Event Notification ของ Blynk

Code
#define DHTPIN 27 #define DHTTYPE DHT22 #define LED_PIN 23 #define BUTTON_PIN 18 #include <DHT.h> #include <Wire.h> #include "RTClib.h" #include <WiFi.h> #include <PubSubClient.h> DHT dht(DHTPIN, DHTTYPE); RTC_DS3231 rtc; // WiFi + MQTT // const char* ssid = "misochoyu_2.4G"; // const char* password = "misochoyu"; // const char* mqtt_server = "192.168.1.112"; // IP เครื่อง Node-RED/MQTT Broker const char* ssid = "Next";

```

const char* password = "0972854443";
const char* mqtt_server = "172.20.10.4"; //ใช้เน็ตมือถือ

WiFiClient espClient;
PubSubClient client(espClient);

bool overheatStarted = false;
bool buttonPressed = false;
unsigned long overheatStartMillis = 0;
unsigned long overheatDuration = 0;

void setup_wifi() {
    delay(10);
    Serial.println("Connecting to WiFi...");
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("\nWiFi connected");
}

void reconnect() {
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        if (client.connect("ESP32Client")) {
            Serial.println("connected");
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            delay(2000);
        }
    }
}

```

```

void setup() {
    Serial.begin(115200);
    pinMode(LED_PIN, OUTPUT);
    pinMode(BUTTON_PIN, INPUT_PULLUP);
    dht.begin();
    rtc.begin();

    setup_wifi();
    client.setServer(mqtt_server, 1883);
}

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

    float temp = dht.readTemperature();
    float hum = dht.readHumidity();
    DateTime now = rtc.now();

    if (isnan(temp) || isnan(hum)) {
        Serial.println("อ่านค่า DHT22 ไม่ได้");
        delay(2000);
        return;
    }

    Serial.print("⌚ ");
    Serial.print(now.timestamp());
    Serial.print("🌡️ Temp: ");
    Serial.print(temp);
    Serial.print("°C 💧 Humidity: ");
    Serial.print(hum);
    Serial.println("%");
}

```

```

// เริ่มจับเวลาเมื่อเกิน 39
if (temp > 39.0 && !overheatStarted) { // ใส่เป็น 25.0 ได้เพื่อเอาไว้ test
    Serial.println(" 🔥 อุณหภูมิเกิน 39 → เริ่มจับเวลา");
    overheatStarted = true;
    buttonPressed = false;
    overheatStartMillis = millis();
    digitalWrite(LED_PIN, HIGH);
}

// กดปุ่ม = หยุดจับเวลา
if (overheatStarted && digitalRead(BUTTON_PIN) == LOW && !buttonPressed) {
    buttonPressed = true;
    overheatDuration = millis() - overheatStartMillis;
    digitalWrite(LED_PIN, LOW);

    Serial.print(" ⚡ หยุดจับเวลา: ");
    Serial.print(overheatDuration / 1000);
    Serial.println(" วินาที");

    // รีเซ็ตให้พร้อมรอบต่อไป
    overheatStarted = false;
}

// ส่งข้อมูลแบบ JSON ไป Node-RED
String payload = "{";
payload += "\"temp\":" + String(temp, 1) + ",";
payload += "\"hum\":" + String(hum, 1) + ",";
payload += "\"duration\":" + String(overheatDuration / 1000),
payload += "}";

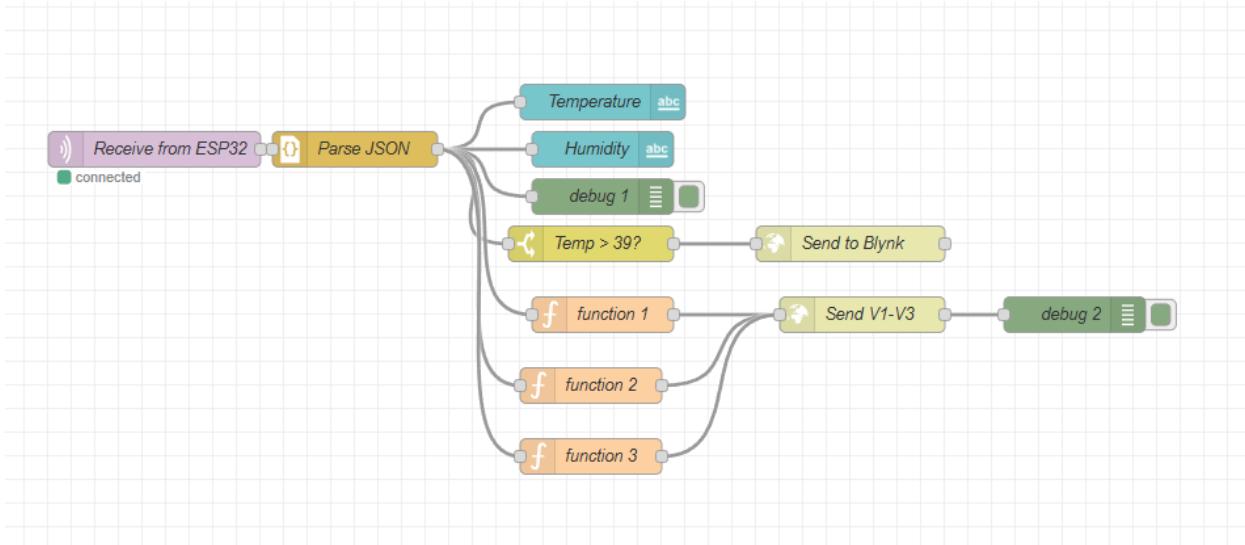
client.publish("healthmonitor/data", payload.c_str());

delay(2000);

```

}

6.3 การทำงานของ Flow Node-RED



Flow.json

```
[  
  {  
    "id": "093da678bba0d2b7",  
    "type": "mqtt in",  
    "z": "8b7022256ef337ca",  
    "name": "Receive from ESP32",  
    "topic": "healthmonitor/data",  
    "qos": "0",  
    "datatype": "auto",  
    "broker": "mqtt_broker",  
    "nl": false,  
    "rap": false,  
    "inputs": 0,  
    "x": 650,  
    "y": 400,  
    "wires": [  
      {  
        "x": 350, "y": 150, "x2": 450, "y2": 150},  
        {  
          "x": 350, "y": 150, "x2": 450, "y2": 200},  
          {  
            "x": 350, "y": 150, "x2": 450, "y2": 250},  
            {  
              "x": 350, "y": 200, "x2": 450, "y2": 200},  
              {  
                "x": 350, "y": 200, "x2": 450, "y2": 250},  
                {  
                  "x": 350, "y": 250, "x2": 450, "y2": 250},  
                  {  
                    "x": 350, "y": 250, "x2": 450, "y2": 300},  
                    {  
                      "x": 350, "y": 300, "x2": 450, "y2": 300},  
                      {  
                        "x": 350, "y": 300, "x2": 450, "y2": 350},  
                        {  
                          "x": 350, "y": 350, "x2": 450, "y2": 350},  
                          {  
                            "x": 350, "y": 350, "x2": 450, "y2": 400},  
                            {  
                              "x": 350, "y": 400, "x2": 450, "y2": 400},  
                              {  
                                "x": 350, "y": 400, "x2": 450, "y2": 450},  
                                {  
                                  "x": 350, "y": 450, "x2": 450, "y2": 450},  
                                  {  
                                    "x": 350, "y": 450, "x2": 450, "y2": 500},  
                                    {  
                                      "x": 350, "y": 500, "x2": 450, "y2": 500},  
                                      {  
                                        "x": 350, "y": 500, "x2": 450, "y2": 550},  
                                        {  
                                          "x": 350, "y": 550, "x2": 450, "y2": 550},  
                                          {  
                                            "x": 350, "y": 550, "x2": 450, "y2": 600},  
                                            {  
                                              "x": 350, "y": 600, "x2": 450, "y2": 600},  
                                              {  
                                                "x": 350, "y": 600, "x2": 450, "y2": 650},  
                                                {  
                                                  "x": 350, "y": 650, "x2": 450, "y2": 650},  
                                                  {  
                                                    "x": 350, "y": 650, "x2": 450, "y2": 700},  
                                                    {  
                                                      "x": 350, "y": 700, "x2": 450, "y2": 700},  
                                                      {  
                                                        "x": 350, "y": 700, "x2": 450, "y2": 750},  
                                                        {  
                                                          "x": 350, "y": 750, "x2": 450, "y2": 750},  
                                                          {  
                                                            "x": 350, "y": 750, "x2": 450, "y2": 800},  
                                                            {  
                                                              "x": 350, "y": 800, "x2": 450, "y2": 800},  
                                                              {  
                                                                "x": 350, "y": 800, "x2": 450, "y2": 850},  
                                                                {  
                                                                  "x": 350, "y": 850, "x2": 450, "y2": 850},  
                                                                  {  
                                                                    "x": 350, "y": 850, "x2": 450, "y2": 900},  
                                                                    {  
                                                                      "x": 350, "y": 900, "x2": 450, "y2": 900},  
                                                                      {  
                                                                        "x": 350, "y": 900, "x2": 450, "y2": 950},  
                                                                        {  
                                                                          "x": 350, "y": 950, "x2": 450, "y2": 950},  
                                                                          {  
                                                                            "x": 350, "y": 950, "x2": 450, "y2": 1000},  
                                                                            {  
                                                                              "x": 350, "y": 1000, "x2": 450, "y2": 1000},  
                                                                              {  
                                                                                "x": 350, "y": 1000, "x2": 450, "y2": 1050},  
                                                                                {  
                                                                                  "x": 350, "y": 1050, "x2": 450, "y2": 1050},  
                                                                                  {  
                                                                                    "x": 350, "y": 1050, "x2": 450, "y2": 1100},  
                                                                                    {  
                                                                                      "x": 350, "y": 1100, "x2": 450, "y2": 1100},  
                                                                                      {  
                        
```

```
[  
    "036b32e49f9d3878"  
]  
]  
},  
{  
    "id": "036b32e49f9d3878",  
    "type": "json",  
    "z": "8b7022256ef337ca",  
    "name": "Parse JSON",  
    "property": "payload",  
    "action": "",  
    "pretty": false,  
    "x": 820,  
    "y": 400,  
    "wires": [  
        [  
            "3c671d15e226e296",  
            "514c1893bbcd23f8",  
            "04d8345a4f2d0111",  
            "d996303e4247a21b",  
            "50e02a5b9b8812bb",  
            "e6acd123fd0bfeb7",  
            "6d6ec4b28b084c8e"  
        ]  
    ]  
},  
{  
    "id": "3c671d15e226e296",  
    "type": "ui_text",  
    "z": "8b7022256ef337ca",  
    "group": "ui_group",  
    "order": 1,  
    "width": 0,  
    "height": 0,
```

```
"name": "Temperature",
"label": "Temperature",
"format": "{{msg.payload.temp}} °C",
"layout": "row-spread",
"className": "",
"style": false,
"font": "",
"fontSize": "",
"color": "#000000",
"x": 1030,
"y": 360,
"wires": []
},
{
  "id": "514c1893bbcd23f8",
  "type": "ui_text",
  "z": "8b7022256ef337ca",
  "group": "ui_group",
  "order": 2,
  "width": 0,
  "height": 0,
  "name": "Humidity",
  "label": "Humidity",
  "format": "{{msg.payload.hum}} %",
  "layout": "row-spread",
  "className": "",
  "style": false,
  "font": "",
  "fontSize": "",
  "color": "#000000",
  "x": 1030,
  "y": 400,
  "wires": []
},
{
```

```
"id": "04d8345a4f2d0111",
"type": "debug",
"z": "8b7022256ef337ca",
"name": "debug 1",
"active": true,
"tosidebar": true,
"console": false,
"tostatus": false,
"complete": "payload",
"targetType": "msg",
"statusVal": "",
"statusType": "auto",
"x": 1030,
"y": 440,
"wires": []
},
{
"id": "d996303e4247a21b",
"type": "switch",
"z": "8b7022256ef337ca",
"name": "Temp > 39?",
"property": "payload.temp",
"propertyType": "msg",
"rules": [
{
"t": "gt",
"v": "39",
"vt": "num"
}
],
"checkall": "true",
"repair": false,
"outputs": 1,
"x": 1020,
"y": 480,
```

```

"wires": [
  [
    "4e06a900ea0515dd"
  ]
],
},
{
  "id": "4e06a900ea0515dd",
  "type": "http request",
  "z": "8b7022256ef337ca",
  "name": "Send to Blynk",
  "method": "POST",
  "ret": "txt",
  "paytoqs": "ignore",
  "url": "https://blynk.cloud/external/api/logEvent?token=q5Pavrg7r3TZ40opDo9BSlzvxfqk-X-O&event=high_temp_alert",
  "tls": "",
  "persist": false,
  "proxy": "",
  "insecureHTTPParser": false,
  "authType": "",
  "senderr": false,
  "headers": [],
  "x": 1240,
  "y": 480,
  "wires": [
    []
  ]
},
{
  "id": "50e02a5b9b8812bb",
  "type": "function",
  "z": "8b7022256ef337ca",
  "name": "function 1",
}

```

```
"func": "const token = \"q5Pavrg7r3TZ40opDo9BSlzvxfqk-X-O\";\\nconst temp =  
msg.payload?.temp ?? 0;\\n\\nmsg.method = \"GET\";\\nmsg.payload = null;\\nmsg.url =  
`https://blynk.cloud/external/api/update?token=${token}&V1=${temp}`;\\n\\nreturn msg;\\n",  
    "outputs": 1,  
    "timeout": "",  
    "noerr": 0,  
    "initialize": "",  
    "finalize": "",  
    "libs": [],  
    "x": 1030,  
    "y": 540,  
    "wires": [  
        [  
            "62c3134ae9087ddb"  
        ]  
    ]  
},  
{  
    "id": "62c3134ae9087ddb",  
    "type": "http request",  
    "z": "8b7022256ef337ca",  
    "name": "Send V1-V3",  
    "method": "use",  
    "ret": "txt",  
    "paytoqs": "ignore",  
    "url": "",  
    "tls": "",  
    "persist": false,  
    "proxy": "",  
    "insecureHTTPParser": false,  
    "authType": "",  
    "senderr": false,  
    "headers": [],  
    "x": 1250,  
    "y": 540,
```

```

    "wires": [
      [
        "ac193d36f780b924"
      ]
    ],
  },
  {
    "id": "ac193d36f780b924",
    "type": "debug",
    "z": "8b7022256ef337ca",
    "name": "debug 2",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "payload",
    "targetType": "msg",
    "statusVal": "",
    "statusType": "auto",
    "x": 1430,
    "y": 540,
    "wires": []
  },
  {
    "id": "e6acd123fd0bfeb7",
    "type": "function",
    "z": "8b7022256ef337ca",
    "name": "function 2",
    "func": "const token = \"q5Pavrg7r3TZ40opDo9BSlzxvfqk-X-O\";\nconst hum =\nmsg.payload?.hum ?? 0;\n\nmsg.method = \"GET\";\nmsg.payload = null;\nmsg.url =\n`https://blynk.cloud/external/api/update?token=${token}&V2=${hum}`;\n\nreturn msg;\n",
    "outputs": 1,
    "timeout": 0,
    "noerr": 0,
    "initialize": ""
  }
]

```

```

    "finalize": "",

    "libs": [],

    "x": 1020,
    "y": 600,
    "wires": [
        [
            "62c3134ae9087ddb"
        ]
    ],
    "id": "6d6ec4b28b084c8e",
    "type": "function",
    "z": "8b7022256ef337ca",
    "name": "function 3",
    "func": "const token = \"q5Pavrg7r3TZ40opDo9BSlzvxfqk-X-O\";\nconst duration =\nmsg.payload?.duration ?? 0;\n\nmsg.method = \"GET\";\nmsg.payload = null;\nmsg.url =\n'https://blynk.cloud/external/api/update?token=${token}&V3=${duration}';\n\nreturn msg;\n",
    "outputs": 1,
    "timeout": 0,
    "noerr": 0,
    "initialize": "",
    "finalize": "",
    "libs": [],
    "x": 1020,
    "y": 660,
    "wires": [
        [
            "62c3134ae9087ddb"
        ]
    ],
    "id": "mqtt_broker",
    "type": "mqtt-broker",

```

```
"name": "Local MQTT",
"broker": "172.20.10.4",
"port": "1883",
"clientId": "",
"autoConnect": true,
"useTLS": false,
"protocolVersion": "4",
"keepalive": "60",
"cleansession": true,
"autoUnsubscribe": true,
"birthTopic": "",
"birthQos": "0",
"birthPayload": "",
"birthMsg": {},
"closeTopic": "",
"closePayload": "",
"closeMsg": {},
"willTopic": "",
"willQos": "0",
"willPayload": "",
"willMsg": {},
"userProps": "",
"sessionExpiry": ""

},
{

"id": "ui_group",
"type": "ui_group",
"name": "Patient Monitor",
"tab": "ui_tab",
"order": 1,
"disp": true,
"width": "6",
"collapse": false,
"className": ""

},
```

```
{  
  "id": "ui_tab",  
  "type": "ui_tab",  
  "name": "Health",  
  "icon": "dashboard",  
  "order": 1,  
  "disabled": false,  
  "hidden": false  
}  
]
```

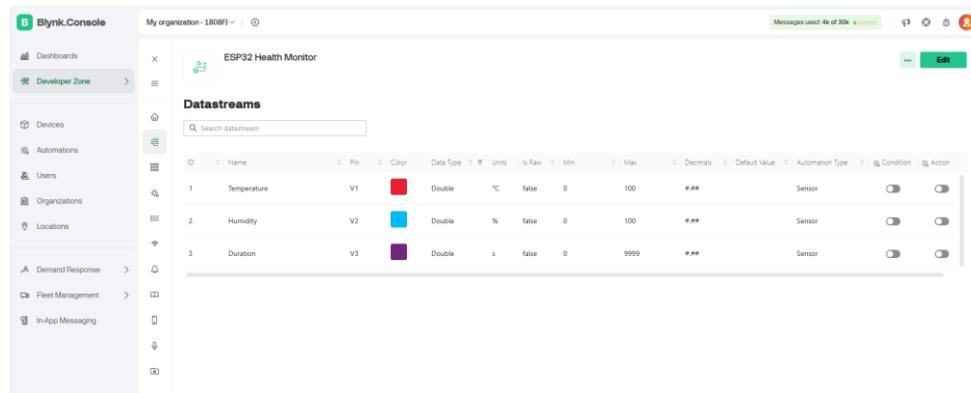
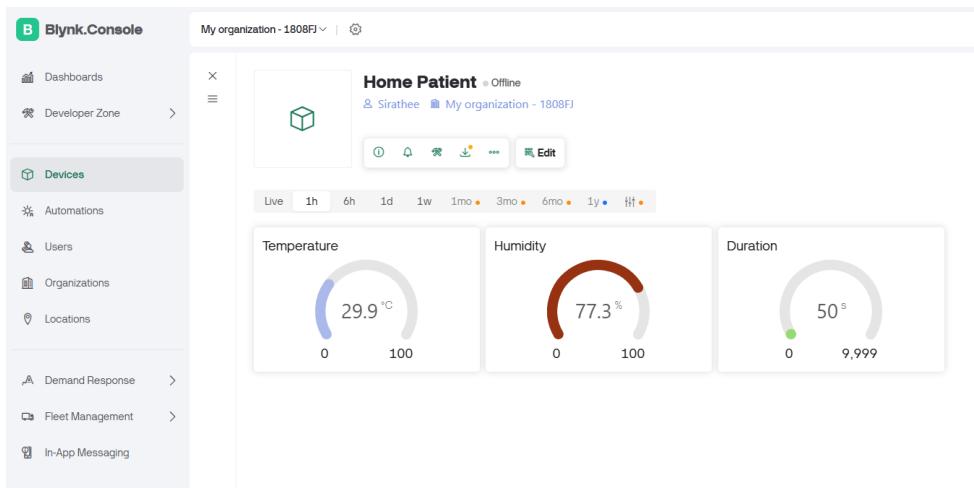
6.4 การทำงานของ Blynk

ในโครงงานนี้ ระบบ Blynk ทำหน้าที่เป็นแพลตฟอร์มแสดงผลข้อมูลจาก ESP32 ที่ถูกส่งผ่าน Node-RED โดยแสดงค่าที่เกี่ยวข้องกับสุขภาพหรือสภาพแวดล้อม เช่น อุณหภูมิ ความชื้น และระยะเวลาที่อุณหภูมิสูงเกินกำหนด ผ่านหน้าจอ Dashboard ของ Blynk ทั้งบนเบราว์เซอร์และมือถือ

ข้อมูลจะถูกส่งไปยัง Blynk โดยใช้ Virtual Pin (VPin) ดังนี้:

- V1: แสดงค่าอุณหภูมิ ($^{\circ}\text{C}$) ที่วัดจากเซ็นเซอร์ DHT22 ในรูปแบบเรียลไทม์
- V2: แสดงค่าความชื้นสัมพัทธ์ (%) ที่ได้จาก DHT22
- V3: แสดงค่าระยะเวลา (s) ที่อุณหภูมิสูงกว่าเกณฑ์ 39°C โดยค่าจะถูกเรียกเมื่อผู้ใช้งานปุ่ม

ข้อมูลทั้งหมดถูกส่งจาก ESP32 ไปยัง Node-RED ผ่าน MQTT จากนั้น Node-RED จะประมวลผลและใช้ node http request ส่งค่าไปยัง Blynk ผ่าน URL API เช่น <https://blynk.cloud/external/api/update?token=...&V1=...&V2=...&V3=...> เพื่ออัปเดตข้อมูลบน Dashboard แบบเรียลไทม์



6.5 สรุปผลการศึกษา

จากการศึกษาและพัฒนาระบบเฝ้าระวังสุขภาพผู้ป่วยที่บ้านโดยใช้บอร์ด ESP32 ร่วมกับเซ็นเซอร์ DHT22, RTC, บุ่มกด และไฟ LED พบว่าสามารถออกแบบระบบที่ตรวจจับอุณหภูมิและความชื้นแบบเรียลไทม์ได้อย่างแม่นยำ และเมื่อตรวจพบอุณหภูมิเกิน 39°C ระบบสามารถเริ่มจับเวลาโดยใช้ RTC และหยุดจับเวลาเมื่อผู้ป่วยกดปุ่ม เพื่อใช้งานแพทช์ได้อย่างมีประสิทธิภาพ

ระบบนี้ยังเชื่อมต่อกับ Node-RED เพื่อจัดการข้อมูล และส่งออกไปยังแพลตฟอร์ม Blynk ผ่าน API โดยใช้protoคอล MQTT ทำให้สามารถแสดงค่าต่าง ๆ ได้แก่ อุณหภูมิ (V1), ความชื้น (V2) และระยะเวลาที่อุณหภูมิสูงเกินค่ากำหนด (V3) ได้แบบเรียลไทม์ผ่านอินเทอร์เน็ต

ผลลัพธ์ที่ได้แสดงให้เห็นว่าระบบทำงานได้ถูกต้องและสามารถนำไปใช้งานจริงในการติดตามผู้ป่วยสูงอายุหรือผู้ป่วยที่ต้องการการดูแลอย่างต่อเนื่องจากระยะไกล ช่วยให้ญาติหรือแพทย์สามารถติดตามอาการเบื้องต้นได้อย่างสะดวก และยังสามารถนำไปต่อยอดเพิ่มฟังก์ชันอื่น ๆ เช่น ระบบแจ้งเตือนผ่าน LINE หรือ SMS ได้ในอนาคต

อ้างอิง

Berryb-Face Recognition Infrared Thermometer อุปกรณ์ตรวจจับความร้อน วัดไข้ วัดอุณหภูมิ

<https://www.berryb.co.th/product/berryb-face-recognition-infrared-thermometer-%E0%B8%AD%E0%B8%B8%E0%B8%9B%E0%B8%81%E0%B8%A3%E0%B8%93%E0%B9%8C%E0%B8%95%E0%B8%A3%E0%B8%A7%E0%B8%88%E0%B8%88%E0%B8%B1%E0%B8%9A%E0%B8%84%E0%B8%A7%E0%B8%B2/>

สอนใช้งานบอร์ด Arduino กับ GY-906 ตรวจวัดอุณหภูมิร่างกาย หากมีไข้ไม่สามารถเข้าได้ให้นอนนอกบ้าน

<https://www.analogread.com/article/111/%E0%B8%AA%E0%B8%AD%E0%B8%99%E0%B9%83%E0%B8%8A%E0%B9%89%E0%B8%87%E0%B8%B2%E0%B8%99%E0%B8%9A%E0%B8%AD%E0%B8%A3%E0%B9%8C%E0%B8%94-arduino-%E0%B8%81%E0%B8%B1%E0%B8%9A-gy-906-%E0%B8%95%E0%B8%A3%E0%B8%A7%E0%B8%88%E0%B8%A7%E0%B8%B1%E0%B8%94%E0%B8%AD%E0%B8%B8%E0%B8%93%E0%B8%AB%E0%B8%A0%E0%B8%B9%E0%B8%A1%E0%B8%B4%E0%B8%A3%E0%B9%88%E0%B8%B2%E0%B8%87%E0%B8%81%E0%B8%B2%E0%B8%A2-%E0%B8%AB%E0%B8%B2%E0%B8%81%E0%B8%A1%E0%B8%6B5%E0%B9%84%E0%B8%82%E0%B9%89%E0%B9%84%E0%B8%A1%E0%B9%88%E0%B8%AA%E0%B8%B2%E0%B8%A1%E0%B8%B2%E0%B8%A3%E0%B8%96%E0%B9%80%E0%B8%82%E0%B9%89%E0%B8%B2%E0%B9%84%E0%B8%94%E0%B9%89%E0%B9%83-%E0%B8%AB%E0%B9%89%E0%B8%99%E0%B8%AD%E0%B8%99%E0%B8%99%E0%B8%AD%E0%B8%81%E0%B8%9A%E0%B9%89%E0%B8%B2%E0%B8%99>