กล้องวงจรปิดโครงข่ายแบบ MESH

MESH WIFI SURVEILLANCE CAM

โดย

นางสาวศิริลักษณ์ พึ่งชาติ

นางสาวอัสรียา หะยีหวัง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2564

กล้องวงจรปิดโครงข่ายแบบ MESH

MESH WIFI SURVEILLANCE CAM

โดย

นางสาวศิริลักษณ์ พึ่งชาติ 61011021

นางสาวอัสรียา หะยีหวัง 61011228

อาจารย์ที่ปรึกษา

ดร.สมปอง วิเศษพานิชกิจ

อาจารย์ที่ปรึกษาร่วม

ผศ.ดร.ธเนศ พัฒธาดาพงษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2564

ปริญญานิพนธ์ปีการศึกษา 2564

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง กล้องวงจรปิดโครงข่ายแบบ MESH

MESH WIFI SURVEILLANCE CAM

ผู้จัดทำ

1. นางสาวศิริลักษณ์ พึ่งชาติ 61011021
2. นางสาวอัสรียา หะยีหวัง 61011228

..... **อาจารย์ที่ปรึกษา**

(ดร.สมปอง วิเศษพานิชกิจ)

..... **อาจารย์ที่ปรึกษาร่วม**

(ผศ.ดร.ธเนศ พัฒนธาดาพงษ์)

กิตติกรรมประกาศ

ปริญญานิพนธ์เรื่อง กล้องวงจรปิดโครงข่ายแบบ MESH สำเร็จลงได้ด้วยดีจากความกรุณาช่วยเหลือและแนะนำอย่างดียิ่งจาก ดร.สมปอง วิเศษพานิชกิจ อาจารย์ที่ปรึกษา ผศ.ดร.ธเนศ พัฒนธาดาพงษ์ อาจารย์ที่ปรึกษาร่วม และ ผศ.ดร.นภัทร สระเอี่ยม ที่ได้ให้คำปรึกษาแนะนำ ข้อมูลและข้อคิดเห็น ตลอดจนแก้ไขข้อบกพร่องต่างๆ มาโดยตลอดการดำเนินงานให้มีความสมบูรณ์มากยิ่งขึ้น ทางผู้จัดทำขอขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณคณาจารย์ และเจ้าหน้าที่ประจำภาควิชาโทรคมนาคมเป็นอย่างยิ่งที่ได้อบรมสั่งสอน ให้ความรู้และประสบการณ์ ตลอดจนคำแนะนำต่างๆ จนทำให้ผู้จัดทำสามารถนำความรู้ทั้งหมดที่ได้มาจัดทำปริญญานิพนธ์เล่มนี้ได้สำเร็จลุล่วง

ขอกราบขอบพระคุณบิดา มารดา และครอบครัว ที่มอบความรัก ความเมตตาและเป็นกำลังใจให้แก่ผู้จัดทำเสมอมา จนทำให้ผู้จัดทำมีโอกาสได้รับการศึกษาที่ดี และสามารถจัดทำปริญญานิพนธ์เล่มนี้ได้สำเร็จ

ขอขอบคุณเพื่อนนักศึกษาในภาควิชาโทรคมนาคม รวมถึงบุคคลที่เกี่ยวข้อง ที่ได้ให้คำแนะนำและคอยเป็นกำลังใจจนทำให้ปริญญานิพนธ์สำเร็จลุล่วงไปได้ด้วยดี

นางสาวศิริลักษณ์ พึ่งชาติ
นางสาวอัสรียา หะยีหวัง
ผู้จัดทำ

กล้องวงจรปิดโครงข่ายแบบ MESH
MESH WIFI SURVEILLANCE CAM

โดย นางสาวศิริลักษณ์ พึ่งชาติ 61011021
นางสาวอัสรียา หะยีหวัง 61011228

อาจารย์ที่ปรึกษา ดร.สมปอง วิเศษพานิชกิจ
อาจารย์ที่ปรึกษาร่วม ผศ.ดร.ธเนศ พัฒนธาดาพงษ์

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ออกแบบกล้องวงจรปิดโครงข่ายแบบ Mesh ซึ่งประกอบด้วยอุปกรณ์ ESP32 CAM และESP32 ร่วมกับจอแสดงผล OLED โดยกล้องวงจรปิดทั้งสามเชื่อมต่อกันผ่านโครงข่ายแบบ MESH เพื่อรับ-ส่งข้อมูลระหว่างอุปกรณ์กล้องวงจรปิด โดยมีฟังก์ชันในการทำงาน คือ ผู้ใช้งานสามารถดูวิดีโอได้จากกล้องวงจรปิดแบบเรียลไทม์ผ่านลิงก์เซิร์ฟเวอร์, ตรวจจับการเคลื่อนไหวของวัตถุที่เคลื่อนไหวผ่านหน้ากล้องวงจรปิดได้ และจับภาพเพื่อบันทึกภาพนิ่งขณะตรวจจับการเคลื่อนไหวได้ลงบน MicroSD card สำหรับดูภาพที่บันทึกย้อนหลัง รวมทั้งเพื่อส่งภาพที่บันทึกได้แจ้งเตือนไปยังผู้ใช้งานขณะมีการตรวจจับเคลื่อนไหวผ่าน Line Notify และกล้องวงจรปิดแต่ละตัวที่เชื่อมต่อกันผ่านโครงข่ายแบบ Mesh จะส่งข้อความแจ้งเตือนไปยังผู้ใช้งานให้ทราบว่าขณะนั้นมีกล้องตัวใดที่เชื่อมต่อกันอยู่บนโครงข่ายผ่าน Line Notify เช่นเดียวกัน นอกจากนี้บนจอแสดงผล OLED ที่เชื่อมต่อกับ ESP32 CAM มีการสร้างภาพเคลื่อนไหวให้มีความน่าใช้งานมากยิ่งขึ้น

ABSTRACT

This thesis designs surveillance cameras using Mesh network includes an ESP32 CAM, ESP32 and OLED display. The three surveillance cameras are connected through a mesh network to send and receive data between the devices. Its functionality is that the user can watching the video from the surveillance camera in real-time through the link server, detect the motion and capture images while detecting motion record on the MicroSD card for watching the recorded images later. Including to send recorded images to alert users when motion is detected through Line Notify. Additionally in each camera that connected on Mesh network will alert to user to know nodes that connect or disconnect in this Mesh network on Line Notify, and on the OLED display that connected to the ESP32 CAM, animations are motion when detected object motion in front of the surveillance camera to make the surveillance camera more usable.

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูป

หน้า

รูปที่

สารบัญรูป (ต่อ)

หน้า

รูปที่

สารบัญรูป (ต่อ)

หน้า

รูปที่

สารบัญตาราง

หน้า

ตารางที่

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันเทคโนโลยี Internet of Things (IoT) มีการใช้งานเพิ่มมากขึ้นเพื่ออำนวยความสะดวกในชีวิตประจำวันของมนุษย์ โดย IoT ที่เป็นที่นิยมอย่างหนึ่ง คือ อุปกรณ์ Smart home หรืออุปกรณ์อำนวยความสะดวกภายในบ้าน ไม่ว่าจะเป็น อุปกรณ์ไฟฟ้าภายในบ้าน ประตูที่สามารถเปิด - ปิดตามคำสั่ง หรือแม้กระทั่งกล้องวงจรปิดภายในบ้านที่สามารถเชื่อมต่ออินเทอร์เน็ตเพื่อการใช้งานที่สะดวกขึ้น ซึ่งอุปกรณ์เหล่านี้ล้วนเป็นที่นิยมในการนำมาใช้งานมากขึ้นเรื่อย ๆ กล้องวงจรปิดเป็นอุปกรณ์ที่นับว่ามีความสำคัญอย่างหนึ่งภายในบ้าน เพื่อรักษาความปลอดภัยให้แก่ผู้ใช้งาน เนื่องจากการดำเนินชีวิตต่าง ๆ และการใช้ชีวิตในปัจจุบันที่ผู้คนส่วนมากมีการเดินทางท่องเที่ยวหรือออกไปทำงานภายนอกบ้าน ซึ่งกล้องวงจรปิดได้มีการพัฒนาหลากหลายรูปแบบเพื่อตอบสนองต่อความต้องการของผู้ใช้งาน ไม่ว่าจะเป็นรูปลักษณ์และฟังก์ชันที่ใช้งานสะดวกมากขึ้น

การเชื่อมต่อโครงข่ายแบบ Mesh หรือโครงข่ายแบบตาข่าย เป็นที่นิยมในการใช้งาน เนื่องจากมีการคำนวณหาเส้นทางเพื่อไม่ให้ข้อมูลชนกันและมีการส่งข้อมูลที่รวดเร็ว ทำให้มีประสิทธิภาพในการทำงานสูง โดยการเชื่อมต่อโครงข่ายแบบ Mesh จะนิยมสร้างบ้านเครือข่ายแบบไร้สาย

ดังนั้นในปริญญานิพนธ์นี้ทางคณะผู้จัดทำจึงได้มีการประยุกต์ใช้ความรู้ด้านเทคโนโลยีด้าน IoT และการเชื่อมต่อโครงข่ายแบบ Mesh เข้าด้วยกันเพื่อสร้างกล้องวงจรปิดที่มีความน่าใช้งานทั้งด้านรูปลักษณ์และฟังก์ชัน โดยใช้อุปกรณ์ ESP32 CAM ร่วมกับ ESP32 และ OLED เป็นอุปกรณ์หลักในการสร้างกล้องวงจรปิด และคาดว่าจะเป็นประโยชน์ในการนำไปต่อยอดและประยุกต์เพิ่มเติมในอนาคตได้

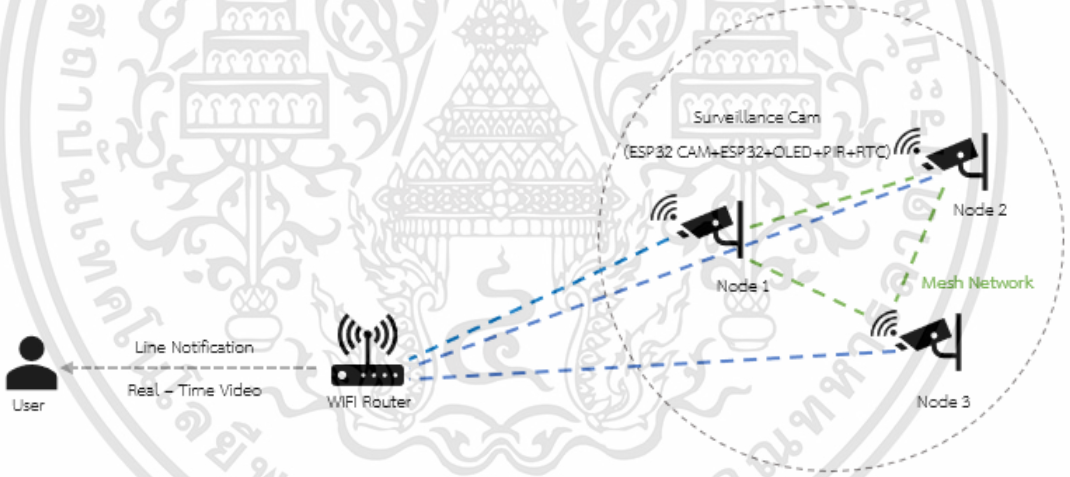
1.2 วัตถุประสงค์

1) เพื่อศึกษาการทำงานของ ESP32 CAM และ ESP32 ร่วมกับเซนเซอร์ต่าง ๆ
2) เพื่อศึกษาการทำงานของโครงข่ายแบบ Mesh
3) เพื่อสอดส่องความปลอดภัยภายในบ้าน และหากตรวจจับการเคลื่อนไหวได้จะทำการแจ้งเตือนไปยังผู้ใช้งาน

1.3 ขอบเขตของปริญญานิพนธ์

1) ศึกษาการทำงานของ ESP32 CAM ร่วมกับ ESP32 และ OLED
2) ออกแบบกล้องวงจรปิดโดยใช้ ESP32 CAM ร่วมกับ ESP32 ซึ่งกล้องวงจรปิดแต่ละตัวสามารถรับ - ส่งข้อมูล ผ่านการเชื่อมต่อโครงข่ายแบบ Mesh
3) ออกแบบให้ผู้ใช้งานสามารถดูวิดีโอจากกล้องวงจรปิดได้แบบ Real-Time
4) ออกแบบให้กล้องวงจรปิดสามารถตรวจจับการเคลื่อนไหวและบันทึกภาพนิ่งขณะมีการตรวจจับการเคลื่อนไหวได้ลงบน MicroSD card และส่งแจ้งเตือนไปยังผู้ใช้งานผ่าน Line Notify
5) พัฒนาและปรับปรุงให้ภาพบนจอแสดงผล OLED สามารถเคลื่อนไหวเมื่อมีการตรวจจับการเคลื่อนไหวได้

โดยมีการแสดงขอบเขตของปริญญานิพนธ์เป็นรูปภาพ แสดงดังรูปที่ 1.1 เพื่อให้สามารถเข้าใจกระบวนการทำงานโดยรวมได้ง่ายขึ้น



รูปที่ 1.1 ระบบกล้องวงจรปิดโครงข่ายแบบ Mesh

บทที่ 2

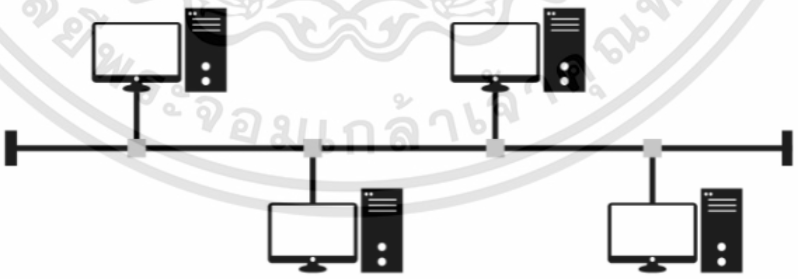
ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 โครงสร้างเครือข่ายคอมพิวเตอร์ (Network Topology)

โทโพโลจี (Topology) คือลักษณะทางกายภาพของระบบเครือข่าย หมายถึง ลักษณะของการเชื่อมโยงสายสื่อสารเข้ากับอุปกรณ์อิเล็กทรอนิกส์และเครื่องคอมพิวเตอร์ภายในเครือข่ายด้วยกัน โทโพโลจีของเครือข่ายแต่ละแบบมีความเหมาะสมในการใช้งานแตกต่างกันออกไป การนำไปใช้จึงมีความจำเป็นที่จะต้องทำการศึกษาลักษณะ คุณสมบัติ ข้อดีและข้อเสียของโทโพโลจีแต่ละแบบ เพื่อนำไปใช้ในการออกแบบพิจารณาเครือข่ายให้เหมาะสมกับการใช้งาน โดยโทโพโลจีมีรูปแบบต่าง ๆ ดังนี้

2.1.1 โทโพโลจีแบบบัส (Bus Topology)

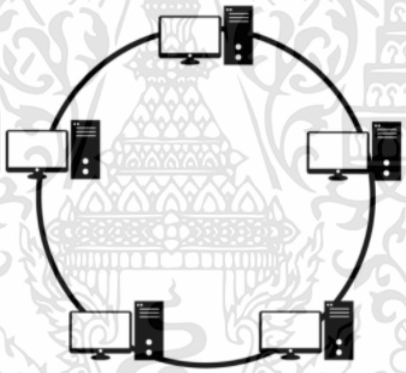
เครือข่ายแบบบัส เป็นรูปแบบที่มีผู้นิยมใช้มากแบบหนึ่งเพราะมีโครงสร้างไม่ยุ่งยากและไม่ต้องใช้อุปกรณ์สลับสาย การเชื่อมต่อมีลักษณะเป็นแบบหลายจุด สถานีทุกสถานีรวมทั้งอุปกรณ์ทุกชิ้นในเครือข่ายจะเชื่อมต่อเข้ากับสายสื่อสารหลักเพียงสายเดียว เรียกว่า "แบ็คโบน" (Back Bone) การจัดส่งข้อมูลลงบนบัสจึงสามารถทำให้การส่งข้อมูลไปถึงทุกสถานีได้ผ่านสายแบ็คโบน แสดงได้ดังรูปที่ 2.1 การจัดส่งวิธีนี้ต้องกำหนดวิธีการที่จะไม่ให้ทุกสถานีส่งข้อมูลพร้อมกัน เพราะจะทำให้ข้อมูลชนกัน โดยวิธีการที่ใช้อาจเป็นการแบ่งช่วงเวลา หรือให้แต่ละสถานีใช้ความถี่สัญญาณที่แตกต่างกัน [1]



รูปที่ 2.1 การเชื่อมต่อเครือข่ายแบบ Bus [1]

2.1.2 โทโพโลจีแบบวงแหวน (Ring Topology)

เครือข่ายแบบวงแหวน เป็นลักษณะการเชื่อมต่ออุปกรณ์ต่าง ๆ เข้ากันเป็นวงกลม โดยสถานีแต่ละสถานีจะต่อกับสถานีที่อยู่ติดทั้งสองข้างของตนเอง โดยจะมีการเชื่อมโยงเครื่องขยายสัญญาณของแต่ละสถานีด้วยกันเป็นวงแหวน สัญญาณข้อมูลจะส่งอยู่ในวงแหวนแบบจุดต่อจุดไปในทิศทางเดียวกันจนถึงผู้รับภายในเวลาที่กำหนด โดยเครื่องขยายสัญญาณเหล่านี้จะมีหน้าที่ในการรับข้อมูลจากเครื่องคอมพิวเตอร์ของตัวเองหรือจากเครื่องขยายสัญญาณตัวก่อนหน้า และส่งข้อมูลต่อไปยังเครื่องขยายสัญญาณตัวถัดไปเรื่อย ๆ เป็นวง แสดงได้ดังรูปที่ 2.2 หากข้อมูลที่ส่งเป็นของสถานีใดเครื่องขยายสัญญาณของสถานีนั้นก็รับและส่งให้กับสถานีนั้น จึงต้องมีการตรวจสอบข้อมูลที่ได้รับว่าเป็นของตนหรือไม่ ถ้าใช่ก็รับไว้ ถ้าไม่ใช่ก็ส่งต่อไป อีกทั้งสามารถตรวจสอบความผิดพลาดในการส่งด้วย ในกรณีที่เครื่องรับปลายทางไม่ได้รับสัญญาณข้อมูลในเวลาที่กำหนด จะมีการแจ้งว่าหากเกิดความผิดพลาดในเครือข่ายได้ [1]

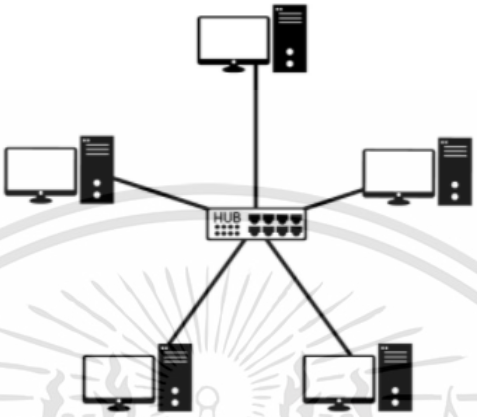


รูปที่ 2.2 การเชื่อมต่อเครือข่ายแบบ Ring [1]

2.1.3 โทโพโลจีแบบดาว (Star Topology)

เป็นการเชื่อมโยงการติดต่อสื่อสารโดยมีสถานีกลาง หรือฮับ (Hub) เป็นจุดผ่านการติดต่อกันระหว่างทุกโนดในเครือข่าย สถานีกลางจึงมีหน้าที่เป็นศูนย์ควบคุมเส้นทางการสื่อสารทั้งหมด นอกจากนี้สถานีกลางยังทำหน้าที่เป็นศูนย์กลางคอยจัดส่งข้อมูลให้กับโนดปลายทาง แสดงได้ดังรูปที่ 2.3 การสื่อสารภายในเครือข่ายแบบดาว จะเป็นแบบ 2 ทิศทางโดยจะอนุญาตให้มีเพียงโนดเดียวเท่านั้นที่สามารถส่งข้อมูลเข้าสู่เครือข่ายได้ จึงไม่มีโอกาสที่หลาย ๆ โนดจะส่งข้อมูล

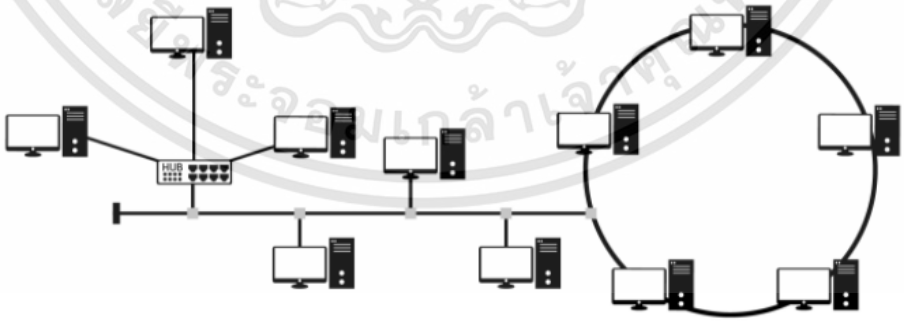
เข้าสู่เครือข่ายในเวลาเดียวกัน เพื่อป้องกันการชนกันของสัญญาณข้อมูล เครือข่ายแบบดาวเป็นโทโพโลจีอีกแบบหนึ่งที่เป็นที่นิยมใช้กันในปัจจุบัน [1]



รูปที่ 2.3 การเชื่อมต่อเครือข่ายแบบ Star [1]

2.1.4 โทโพโลจีแบบผสม (Hybrid Topology)

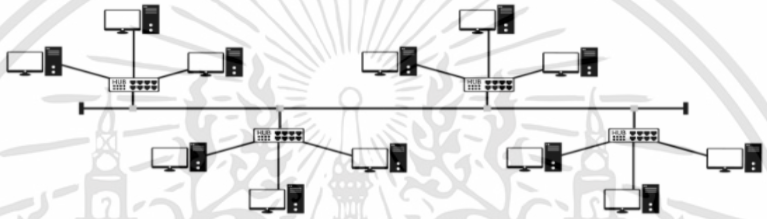
เป็นการเชื่อมต่อที่ผสมผสานเครือข่ายย่อย ๆ หลายส่วนมารวมเข้าด้วยกัน เช่น นำเอาเครือข่ายแบบบัส , เครือข่ายแบบวงแหวน และเครือข่ายแบบดาว มาเชื่อมต่อเข้าด้วยกัน เหมาะสำหรับบางหน่วยงานที่มีเครือข่ายเก่าและใหม่ให้สามารถทำงานร่วมกันได้ เครือข่ายแบบผสมนี้จะมีโครงสร้างแบบลำดับชั้น (Hierarchical) หรือ แบบต้นไม้ (Tree) ที่มีลำดับชั้นในการทำงาน แสดงได้ดังรูปที่ 2.4 [1]



รูปที่ 2.4 การเชื่อมต่อเครือข่ายแบบผสม [1]

2.1.5 โทโพโลจีแบบต้นไม้ (Tree Topology)

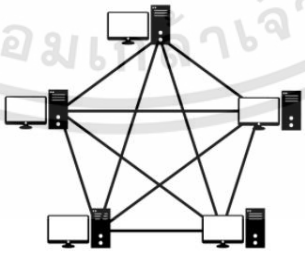
มีลักษณะเชื่อมโยงคล้ายกับโครงสร้างเครือข่ายแบบดาวกับเครือข่ายแบบบัสผสมกัน โดยมีสายนำสัญญาณแยกออกไปเป็นแบบกิ่งไม่เป็นวงรอบ โครงสร้างแบบนี้จะเหมาะกับการประมวลผลแบบกลุ่มจะประกอบด้วยเครื่อง คอมพิวเตอร์ระดับต่าง ๆ กันอยู่หลายเครื่องแล้วต่อกันเป็นชั้น ๆ แต่ละกลุ่มจะมีโนดแม่และโนดลูกในกลุ่มนั้นที่มีการสัมพันธ์กัน การสื่อสารข้อมูลจะผ่านตัวกลางไปยังสถานีอื่น ๆ ได้ทั้งหมด แสดงได้ดังรูปที่ 2.5 เพราะทุกสถานีจะอยู่บนทางเชื่อมและรับส่งข้อมูลเดียวกัน ดังนั้นในแต่ละกลุ่มจะส่งข้อมูลได้ทีละสถานีโดยไม่ส่งพร้อมกัน [1]



รูปที่ 2.5 การเชื่อมต่อเครือข่ายแบบ Tree [1]

2.1.6 โทโพโลจีแบบ Mesh หรือแบบตาข่าย (Mesh Topology)

รูปแบบเครือข่ายแบบนี้ ปกติใช้ในระบบเครือข่ายบริเวณกว้าง (Wide Area-Network) ลักษณะการสื่อสารจะมีการต่อสายหรือการเดินของข้อมูลระหว่างคอมพิวเตอร์หรือโนดไปยังโนดอื่น ๆ ทุก ๆ ตัว แสดงได้ดังรูปที่ 2.6 ทำให้มีทางเดินข้อมูลหลายเส้นและปลอดภัยจากเหตุการณ์ที่จะเกิดจากการล้มเหลวของระบบ แต่ระบบนี้จะมีค่าใช้จ่ายมากกว่าระบบอื่น ๆ เพราะต้องใช้สายสื่อสารเป็นจำนวนมากในกรณีที่มีการต่อแบบใช้สาย [1]



รูปที่ 2.6 การเชื่อมต่อเครือข่ายแบบ Mesh [1]

2.2 การเชื่อมต่อแบบ MESH WIFI

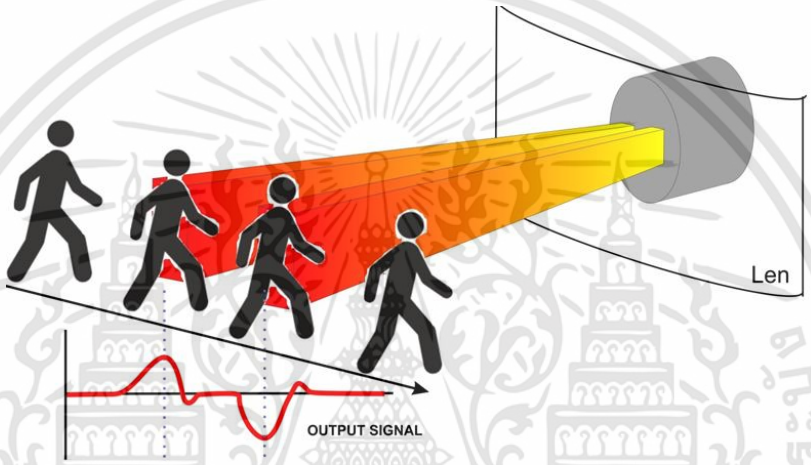
ในปัจจุบันได้มีการพัฒนาจากโทโพโลจีจากหัวข้อที่ 2.1 เกิดเป็นการติดต่อสื่อสารแบบไร้สาย เพื่อความสะดวกสบาย ความทันสมัยเข้ากับการใช้งานในโลกไร้สายมากยิ่งขึ้น และยังสามารถแก้ปัญหาบางอย่างได้ เช่น การนำการเชื่อมต่อเครือข่ายแบบ Mesh มาแก้ปัญหา Wi-Fi ไปเชื่อมต่อไม่ถึงในจุดอับสัญญาณ เป็นต้น

MESH WIFI คือ การเชื่อมต่อไร้สายแบบ Mesh หรือแบบตาข่ายใยแมงมุมที่ทุกเครื่องเชื่อมต่อถึงกันทั้งหมด การเชื่อมต่อเครือข่ายแบบ Mesh จะเชื่อมต่อได้ทั้งแบบต่อตรงและไม่มีลำดับขั้นไปยังจุดโนดต่าง ๆ สามารถเพิ่มจุดกระจายสัญญาณได้ง่ายและได้มากเท่าที่ระบบจะทำได้ สัญญาณที่ปล่อยออกมามีเสถียรภาพและมีรักษาความเร็วอินเทอร์เน็ตได้ดีกว่าอุปกรณ์ทวนสัญญาณ (Repeater) ทั่วไป โดยจะใช้ชื่อ Wi-Fi (SSID) เดียวกันได้ ไม่ต้องเชื่อมต่อใหม่เมื่อเดินจากห้องหนึ่งไปยังอีกห้องหนึ่ง แสดงได้ดังรูปที่ 2.7 โดยระบบจะเลือกการเชื่อมต่อแต่ละจุดให้อัตโนมัติตามความเหมาะสม ซึ่งถ้ามีเครื่องใดเครื่องหนึ่งหลุดออกจากระบบ เครื่องจะเชื่อมต่อเส้นทางใหม่ให้โดยอัตโนมัติ ทำให้ใช้งานได้อย่างต่อเนื่อง [2]

รูปที่ 2.7 การเชื่อมต่อไร้สายแบบ Mesh โดยใช้ชื่อ Wi-Fi (SSID) เดียวกัน [2]

2.3 หลักการทำงานของเซนเซอร์ตรวจจับคลื่นรังสีอินฟราเรด

HC-SR501 เป็นโมดูลเซนเซอร์ตรวจจับความเคลื่อนไหวที่ทำงานแบบ Passive โดยใช้หลักการไพโรอิเล็กทริก (Pyroelectric) จะทำการตรวจจับรังสีอินฟราเรด ซึ่งรังสีอินฟราเรดนี้จะเกิดจากการแผ่ความร้อนจากตัวของสิ่งมีชีวิต เมื่อมีสิ่งมีชีวิตผ่านหน้าเซนเซอร์ตัวนี้จะถูกจับรังสีอินฟราเรดได้ และถูกส่งไปยังวงจรขยายสัญญาณให้มีความเข้มพอที่จะส่งออกไป [3] แสดงได้ดังรูปที่ 2.8



รูปที่ 2.8 การทำงานของเซนเซอร์ตรวจจับคลื่นรังสีอินฟราเรด [4]

2.3.1 คุณสมบัติของโมดูลเซ็นเซอร์ HC-SR501

1) Input 5-20V (แนะนำที่ 5V)

2) Output 3.3V TTL

3) ปรับการหน่วงเวลา Output ได้ 3 วินาที ถึง 5 วินาที

4) ปรับช่วงระยะตรวจจับได้ 3 - 7 เมตร

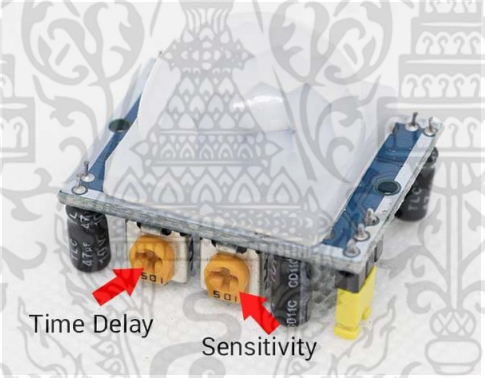
5) เลือก Output ได้ทั้ง Single Trigger และ Repeat Trigger

6) องศาการตรวจจับกว้าง 110 องศา

2.3.2 การปรับแต่งโมดูล

1) การปรับระยะเวลาการหน่วงของสัญญาณเอาต์พุตหรือ Time Delay สามารถปรับตั้งได้ตั้งแต่ 3 ถึง 5 นาที โดยสามารถปรับได้โดยการปรับตัวต้านทานทางด้านซ้าย หากหมุนทวนเข็มนาฬิกาจะเป็นการลดค่าหน่วงเวลาและหากหมุนตามเข็มนาฬิกาจะเป็นการเพิ่มค่าหน่วงเวลา

2) การปรับระยะตรวจจับหรือ Sensitivity เป็นการปรับระยะสูงสุดในการตรวจจับของเซนเซอร์ ซึ่งสามารถปรับค่าระยะการตรวจจับได้ตั้งแต่ 3 ถึง 7 เมตร โดยปรับได้ด้วยการปรับตัวต้านทานทางด้านขวา หากหมุนทวนเข็มนาฬิกาจะเป็นการลดระยะการตรวจจับ และหากหมุนตามเข็มนาฬิกาจะเป็นการเพิ่มระยะการตรวจจับ โดยตำแหน่งในการปรับค่าของโมดูลนี้ แสดงได้ดังรูปที่ 2.9



รูปที่ 2.9 ตำแหน่งการปรับค่าของโมดูล HC-SR501 [3]

3) การปรับโหมดของเอาต์พุต หรือ Trigger Mode โดยแบบ Single นั้นจะเป็นการส่งสัญญาณเพียงแค่ครั้งเดียวตามการปรับระยะเวลาการหน่วงของสัญญาณกำหนดหลังจากตรวจจับการเคลื่อนไหวได้ใน ซึ่งหากปรับเป็น Repeat สัญญาณเอาต์พุตจะยืดเวลาออกไปอีกตามระยะเวลาการหน่วงของสัญญาณที่ได้กำหนดจาก ในการปรับ Trigger Mode สามารถปรับได้จาก Jumper Pin บนโมดูล หากต่อขากลางกับขาที่อยู่ด้านเดียวกับตัวต้านทานจะเป็นการปรับ Trigger Mode ให้เป็น Single Trigger แต่ถ้าหากต่อขากลางกับขาอีกด้านจะเป็นปรับ Trigger Mode ให้เป็นแบบ Repeat Trigger

2.4 หลักการทำงานของเซนเซอร์ Real Time Clock

Real Time Clock (RTC) เป็นอุปกรณ์ที่ให้ค่าเวลาตามจริง ทำงานโดยการจับสัญญานนาฬิกาที่ได้มาจาก Crystal มีถ่านสำรองมาให้เพื่อให้สามารถบันทึกเวลาได้ ทำให้ไม่ต้องตั้งเวลาใหม่ทุกครั้ง โมดูล RTC นี้จำเป็นอย่างยิ่งกับการใช้งานที่ต้องมีการบันทึกเวลา (Time Stamp) หรือมีการทำงานที่เกี่ยวพันกับเวลาจริง เช่น การตั้งเวลาเปิดปิด Relay Module เป็นต้น

ในความจริง Adafruit Feather M0 มีตัวจับเวลาอยู่แล้ว เช่น การใช้ฟังก์ชัน millis() เป็นต้น แต่การประมวลผลคำสั่งจะทำงานแบบอนุกรม (Serial) คือ ทำทีละบรรทัด ทำให้การทำงานของคำสั่งจับเวลาจะถูกรบกวนจากการประมวลผลคำสั่งอื่น ๆ ไปด้วย เวลาที่ได้จากการใช้คำสั่งนี้อาจไม่สามารถนำมาเป็นเวลาตามจริงที่ต้องการบันทึกไปพร้อมกับค่าอื่น ๆ ที่ต้องการวัดได้ ดังนั้น ในการประยุกต์ใช้กับงานที่ต้องการเวลาที่แม่นยำ และสามารถบอก วันที่ เดือน ปี ชั่วโมง นาที วินาที จึงต้องใช้อุปกรณ์ที่ทำหน้าที่จับเวลาแยก

DS3231 เป็น RTC Module ที่มีข้อดีคือ มีการชดเชยการเปลี่ยนแปลงของสัญญาณนาฬิกา เนื่องจากการเปลี่ยนแปลงของอุณหภูมิแวดล้อมด้วย หรือก็คือ เวลาที่อุณหภูมิเปลี่ยนสัญญาณนาฬิกาจาก Crystal ก็เปลี่ยน ทำให้เวลาก็เพี้ยนไปด้วย แต่โมดูลนี้ได้ทำการวัดค่าอุณหภูมิพร้อมทั้งชดเชยความเปลี่ยนแปลงนี้ไปด้วย ทำให้เวลาที่ได้มีความแม่นยำสูง อีกทั้งยังใช้การเชื่อมต่อแบบบัส I2C ทำให้ง่ายต่อการใช้งาน [5] แสดงได้ดังรูปที่ 2.10

รูปที่ 2.10 โมดูลนาฬิกา DS3231 Real Time Clock [6]

2.5 หลักการทำงานของจอแสดงผล OLED

จอแสดงผล OLED (Organic Light-Emitting Diode display) เป็นจอแสดงผลที่สร้างจากวัสดุสารกึ่งตัวนำอินทรีย์ (Organic Semiconductor) ที่มีลักษณะเป็นชั้นสารกึ่งตัวนำบาง ๆ อยู่ระหว่างขั้วบวก (Anode) และขั้วลบ (Cathode) และสามารถเปล่งแสงได้เมื่อมีกระแสไฟฟ้าไหลผ่าน การเปล่งแสงนี้เรียกว่ากระบวนการ อิเล็คโทรลูมิเนสเซนส์ (Electroluminescence) จอแสดงผล OLED มีข้อดีซึ่งแตกต่างจากจอแสดงผล LCD (Liquid Crystal Display) ทั่วไปคือ จอ OLED ไม่มีวงจรที่สร้างแสง Backlight จึงทำให้จอมีความหนาน้อยกว่าและเบากว่า ใช้กำลังไฟฟ้าต่ำ นอกจากนั้นจะไม่มีการเปล่งแสงในบริเวณที่ต้องการให้เป็นสีดำ มีการเชื่อมต่อขาของโมดูล OLED แบบ I2C ง่ายต่อการใช้งาน ในปัจจุบันพบในอุปกรณ์หลายอย่างเช่น โทรทัศน์, สมาร์ทโฟน (Smartphones) และ แท็บเล็ต (Tablets) ได้เริ่มเปลี่ยนไปใช้จอแสดงผล OLED กันมากขึ้น [7] แสดงได้ดังรูปที่ 2.11

รูปที่ 2.11 จอแสดงผล OLED [8]

2.6 การสื่อสารอนุกรม (Serial Communication)

ในปัจจุบันมีการนำไมโครคอนโทรลเลอร์ไปเชื่อมต่อกับอุปกรณ์ต่าง ๆ โดยการเชื่อมต่อนั้นก็มีอยู่ด้วยกันหลายรูปแบบ โดยการเชื่อมต่อแบบหนึ่งที่ได้รับความนิยมก็คือการเชื่อมต่อแบบสื่อสารอนุกรม โดยการสื่อสารอนุกรมจะมีได้หลายแบบดังนี้

2.6.1 การสื่อสารอนุกรมแบบ UART

UART (Universal Asynchronous Receiver-Transmitter) หมายถึงอุปกรณ์ที่ทำหน้าที่รับและส่งข้อมูลแบบอะซิงโครนัส (asynchronous) ซึ่งมักใช้กับการสื่อสารแบบอนุกรม โดยการสื่อสารแบบอนุกรมอาจจะแบ่งได้เป็น 2 แบบ คือ

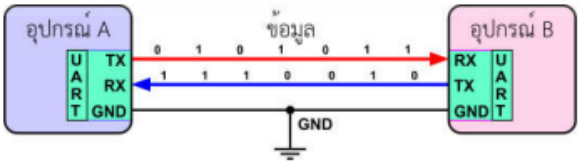
1) ซิงโครนัส (Synchronous)

เป็นการส่งข้อมูลเป็นบล็อก ครั้งละหลายไบต์ โดยจะมีสัญญาณนาฬิกา (Clock) ที่ช่วยในการทำงานของตัวส่งและตัวรับสอดคล้องกัน โดยสัญญาณที่ส่งอาจจะถูกเข้ารหัสร่วมกันอยู่ในชุดข้อมูลนั้นหรือแยกอิสระออกเป็นสายต่างหากก็ได้ ลักษณะแนวคิดพื้นฐานในการส่งข้อมูลแบบซิงโครนัส แสดงได้ดังรูปที่ 2.12

รูปที่ 2.12 การสื่อสารแบบซิงโครนัส

2) อะซิงโครนัส (Asynchronous)

การส่งข้อมูลแบบอะซิงโครนัสเป็นการสื่อสารแบบที่ใช้มากในไมโครคอนโทรลเลอร์ รูปแบบการสื่อสารจะเป็นการรับส่งข้อมูลครั้งละ 1 ไบต์ [9] แสดงได้ดังรูปที่ 2.13



รูปที่ 2.13 การสื่อสารแบบอะซิงโครนัส

2.6.2 การสื่อสารอนุกรมแบบ I2C (Inter-Integrated Circuit)

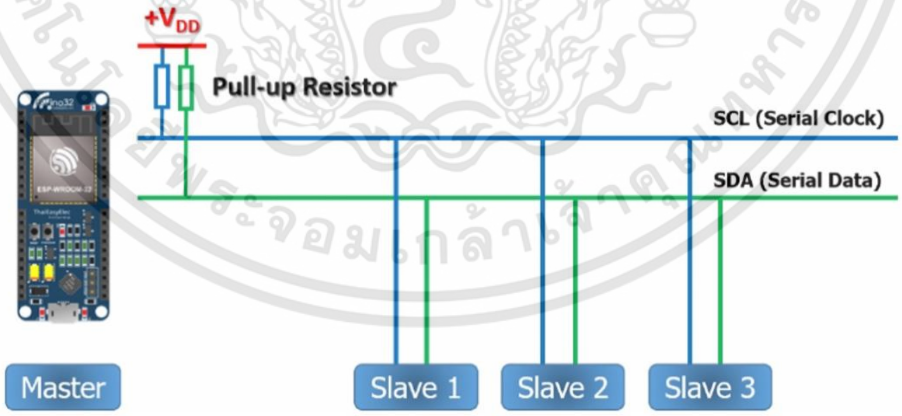
การสื่อสารอนุกรมแบบ I2C เป็นรูปแบบการสื่อสารข้อมูลอย่างหนึ่งที่สร้างขึ้นมาเพื่อสื่อสารข้อมูลความเร็วต่ำ นิยมใช้กับอุปกรณ์จำพวกไมโครโปรเซสเซอร์ ไมโครคอนโทรลเลอร์และอุปกรณ์ต่าง ๆ ที่เกี่ยวข้อง

การสื่อสารอนุกรมแบบ I2C เป็นการใช้งานบอร์ดไมโครคอนโทรลเลอร์เพื่อรับส่งข้อมูลระหว่างบอร์ดไมโครคอนโทรลเลอร์กับอุปกรณ์ภายนอกอื่น ๆ ที่เกี่ยวข้อง โดยการสื่อสารแบบ I2C จะเป็นส่วนหนึ่งของการรับส่งข้อมูลแบบอนุกรมในแบบซิงโครนัส (Synchronous) ตัวอย่างที่ใช้งานการสื่อสารอนุกรมแบบ I2C อาทิ เช่น การสื่อสารระหว่างบอร์ด ESPino32 กับจอภาพแสดงผล, การสื่อสารระหว่างบอร์ด ESPino32 กับอุปกรณ์ขยายจำนวนพอร์ท GPIO และการสื่อสารระหว่างบอร์ด ESPino32 กับอุปกรณ์อ่านค่า ADC/DAC เป็นต้น

ข้อดีของการสื่อสารอนุกรมแบบ I2C คือ สามารถรับและส่งข้อมูลได้หลายอุปกรณ์ในบัสเดียวกัน แสดงได้ดังรูปที่ 2.14 โดยการเชื่อมต่อระบบด้วยการสื่อสารอนุกรมแบบ I2C และใช้สายสัญญาณเพียง 2 เส้นในการรับส่งข้อมูล ทำให้สามารถลดสายสัญญาณที่ใช้ในการเชื่อมต่ออุปกรณ์ลงมาก โดยสายสัญญาณทั้ง 2 เส้นแบ่งเป็น

1) SDA (Serial Data) คือ สายสัญญาณสำหรับรับและส่งข้อมูล

2) SCL (Serial Clock) คือ สายสัญญาณนาฬิกาใช้เป็นสำหรับควบคุมการรับส่งข้อมูล



รูปที่ 2.14 ตัวอย่างการรับส่งข้อมูลของการสื่อสารอนุกรมแบบ I2C [10]

จากรูปที่ 2.14 แสดงตัวอย่างการรับส่งข้อมูลของการสื่อสารอนุกรมแบบ I2C ประกอบด้วย 2 ส่วน ได้แก่ อุปกรณ์หลัก (Master Device) และอุปกรณ์ย่อย (Slave Device) ในระบบการสื่อสารแบบอนุกรม I2C สามารถต่ออุปกรณ์ I2C ได้หลายอุปกรณ์ ขณะสื่อสารกันในระบบอุปกรณ์หลักจะอ้างอิง Address ของอุปกรณ์ย่อยเพื่อระบุว่าต้องการสื่อสารกับอุปกรณ์ย่อยตัวใด [10]

2.6.3 การสื่อสารอนุกรมแบบ SPI

SPI (Serial Peripheral Interface) คือรูปของแบบการสื่อสารข้อมูลแบบอนุกรมแบบซิงโครนัส (Synchronous) รูปแบบหนึ่ง ถูกพัฒนาขึ้นมาเพื่อใช้ในการการสื่อสารระยะใกล้ สื่อสารอนุกรมแบบ SPI จะอาศัยสัญญาณนาฬิกา (Clock) เป็นตัวกำหนดจังหวะการรับส่งข้อมูล สามารถส่งข้อมูลไปยังปลายทางและรับข้อมูลจากปลายทางกลับมาในครั้งเดียวกัน (Full Duplex) การสื่อสารอนุกรมแบบ SPI จะแบ่งอุปกรณ์ออกเป็น 2 ส่วน คือ อุปกรณ์หลัก ซึ่งเป็นตัวควบคุมการรับส่งข้อมูล และอุปกรณ์ย่อย ซึ่งเป็นอุปกรณ์ที่รอรับคำสั่งจากอุปกรณ์หลักในบัสการสื่อสารแบบอนุกรมแบบ SPI สามารถมีอุปกรณ์ย่อยได้มากกว่า 1 ตัว ตัวอย่างที่ใช้งานการสื่อสารอนุกรมแบบ SPI อาทิ เช่น การสื่อสารระหว่างบอร์ด ESPino32 กับอุปกรณ์อ่านค่า ADC/DAC, การสื่อสารระหว่างบอร์ด ESPino32 กับจอภาพแสดงผล, การสื่อสารระหว่างบอร์ด ESPino32 กับ SD Module และการสื่อสารระหว่างบอร์ด ESPino32 กับ RFID เป็นต้น [9]

การสื่อสารอนุกรมแบบ SPI ใช้สายสัญญาณทั้งหมดจำนวน 4 เส้น มีรายละเอียดดังนี้

1) SCK ใช้สำหรับส่งสัญญาณนาฬิกาจากอุปกรณ์หลักไปยังอุปกรณ์ย่อย

2) MISO ใช้สำหรับรับข้อมูลจากอุปกรณ์ย่อย

3) MOSI ใช้สำหรับส่งข้อมูลจากอุปกรณ์หลักไปยังอุปกรณ์ย่อย

4) SS/CS ใช้สำหรับเลือกอุปกรณ์ย่อยที่ต้องการใช้งาน

2.6.4 อัตราการส่งข้อมูล

อัตราการส่งข้อมูลหรือ บอดเรท (Baud Rate) คือความเร็วของการรับส่งข้อมูล เป็นจำนวนบิตต่อวินาที (bit per second, bps) เช่น 300, 1200, 4800, 9600, 19200, 38400, 57600, 74880, 115200 bps เป็นต้น การเลือกอัตราการส่งข้อมูลขึ้นอยู่กับชนิดของสายสัญญาณ, ระยะทาง และปริมาณสัญญาณรบกวน โดยการส่งข้อมูลระหว่างไมโครคอนโทรลเลอร์และอุปกรณ์

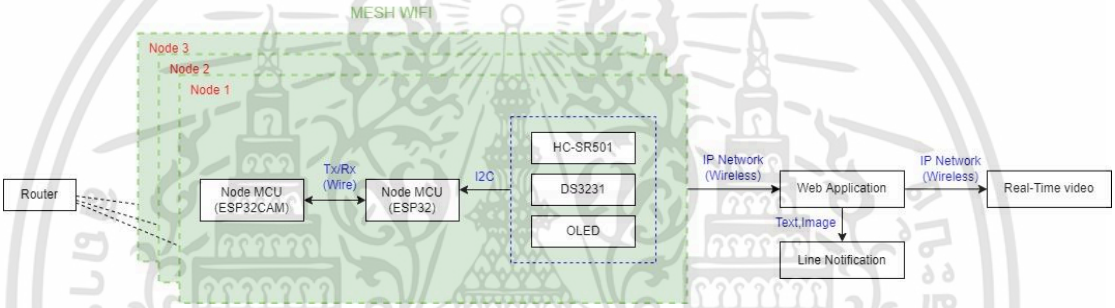
อื่น ๆ นั้น จะทำได้ก็ต่อเมื่อเรากำหนดให้อัตราการส่งข้อมูลมีค่าตรงกันเท่านั้น โดย Baud Rate มีค่าที่ใช้บ่อยคือ 9600 และ 115200 [9]

บทที่ 3

การออกแบบและการจัดทำปริญญานิพนธ์

3.1 การออกแบบ

ปริญญานิพนธ์นี้ในส่วนของการออกแบบได้ทำการออกแบบและจัดทำชุดอุปกรณ์กล้องวงจรปิดโครงข่ายแบบ Mesh ซึ่งประกอบไปด้วยกล้องวงจรปิด 3 ตัว มีการใช้โมดูล ESP32-CAM, ESP32, DS3231, HC-SR501 และจอแสดงผล OLED เป็นอุปกรณ์หลัก โดยกล้องวงจรปิดทั้ง 3 ตัว มีการเชื่อมต่อกันผ่านโครงข่ายแบบ Mesh แสดงได้ดังรูปที่ 3.1



รูปที่ 3.1 บล็อกไดอะแกรมของปริญญานิพนธ์

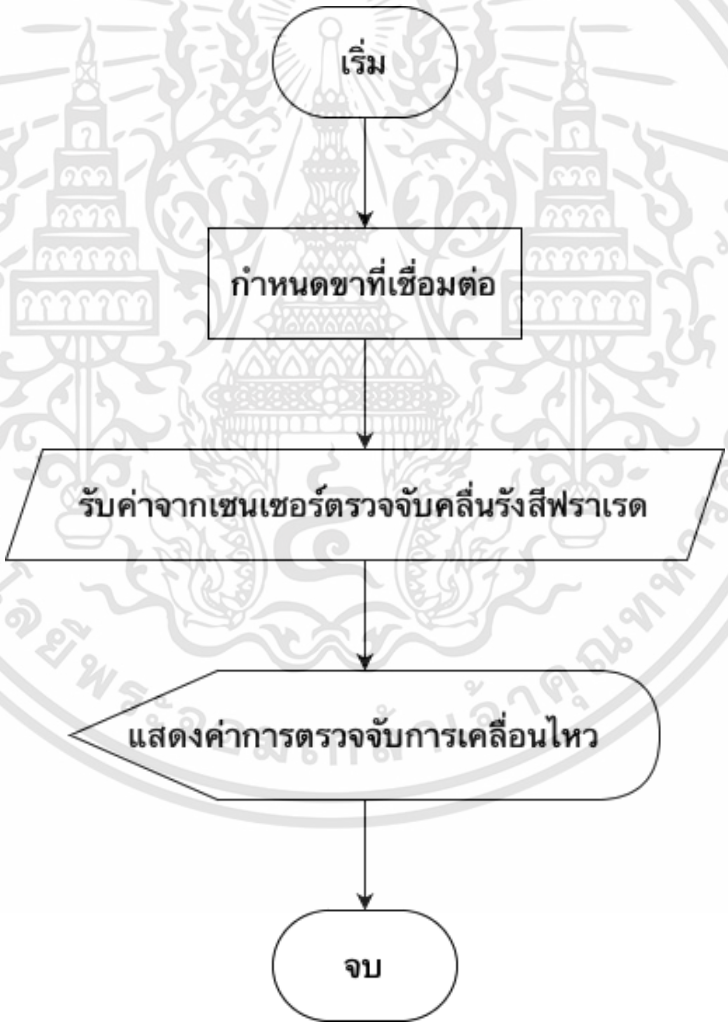
3.1.1 การออกแบบการเชื่อมต่ออุปกรณ์

3.1.1.1 การเชื่อมต่อเซนเซอร์ตรวจจับรังสีอินฟราเรดและอุปกรณ์ ESP32

อุปกรณ์ตรวจจับรังสีอินฟราเรดที่เลือกใช้ คือ โมดูล HC-SR501 เป็นโมดูลเซนเซอร์ตรวจจับความเคลื่อนไหวที่ทำงานแบบ Passive โดยใช้หลักการไพโรอิเล็กทริก (Pyroelectric) จะทำการตรวจจับรังสีอินฟราเรด ซึ่งรังสีอินฟราเรดนี้จะเกิดจากการแผ่ความร้อนจากตัวของสิ่งมีชีวิต โดยจะทำการเชื่อมต่อเซนเซอร์ตรวจจับรังสีอินฟราเรดเข้ากับโมดูล ESP32 แสดงได้ดังตารางที่ 3.1 และแผนภาพการทำงานของเซนเซอร์ตรวจจับรังสีอินฟราเรด แสดงได้ดังในรูปที่ 3.2

ตารางที่ 3.1 การเชื่อมต่อ HC-SR501 เข้ากับโมดูล ESP32

HC-SR501	ESP32
3V3	VCC
GND	GND
OUT	D2



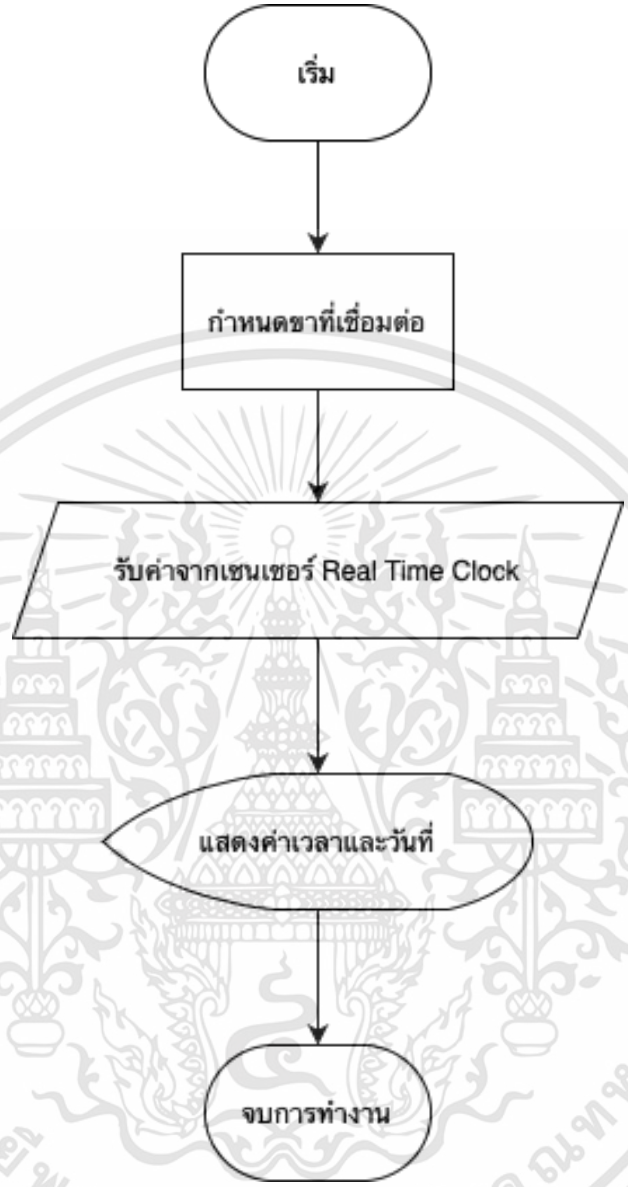
รูปที่ 3.2 แผนภาพการทำงานของเซนเซอร์ตรวจจับรังสีอินฟราเรด

3.1.1.2 การเชื่อมต่อเซนเซอร์ Real Time Clock และอุปกรณ์ ESP32

อุปกรณ์ที่ให้ค่าเวลาตามจริงที่เลือกใช้คือ DS3231 ทำงานโดยการจับสัญญานนาฬิกาที่ได้มาจาก Crystal มีถ่านสำรองมาให้เพื่อให้สามารถบันทึกเวลาได้ ทำให้ไม่ต้องตั้งเวลาใหม่ทุกครั้ง โมดูลนี้ได้ทำการวัดค่าอุณหภูมิพร้อมทั้งชดเชยความเปลี่ยนแปลงนี้ไปด้วย ทำให้เวลาที่ได้มีความแม่นยำสูง อีกทั้งยังใช้การเชื่อมต่อแบบบัส I2C ทำให้ง่ายต่อการใช้งาน โดยจะทำการเชื่อมต่อเซนเซอร์ Real Time Clock เข้ากับโมดูล ESP32 แสดงได้ดังตารางที่ 3.2 โดยแผนภาพการทำงานของเซนเซอร์ Real Time Clock แสดงได้ดังในรูปที่ 3.3

ตารางที่ 3.2 การเชื่อมต่อ DS3231 เข้ากับโมดูล ESP32

DS3231	ESP32
GND	GND
VCC	3V3
SDA	D21
SCL	D22



รูปที่ 3.3 แผนภาพการทำงานของเซนเซอร์ Real Time Clock

3.1.1.3 การเชื่อมต่อจอแสดงผล OLED และอุปกรณ์ ESP32

การแสดงผลผ่าน OLED เมื่อมีการตรวจจับการเคลื่อนไหวผ่าน HC-SR501 จะแสดงผลเป็นการแสดงรูปดวงตากระพริบตา 1 ครั้ง เมื่อมีการตรวจจับการเคลื่อนไหว โดยจะทำการเชื่อมต่อ OLED เข้ากับโมดูล ESP32 แบบ I2C แสดงได้ดังตารางที่ 3.3

ตารางที่ 3.3 การเชื่อมต่อ OLED เข้ากับโมดูล ESP32

OLED	ESP32
GND	GND
VCC	Vin
SDA	D21
SCL	D22

3.1.1.4 การเชื่อมต่ออุปกรณ์ ESP32 CAM และ FT232RL

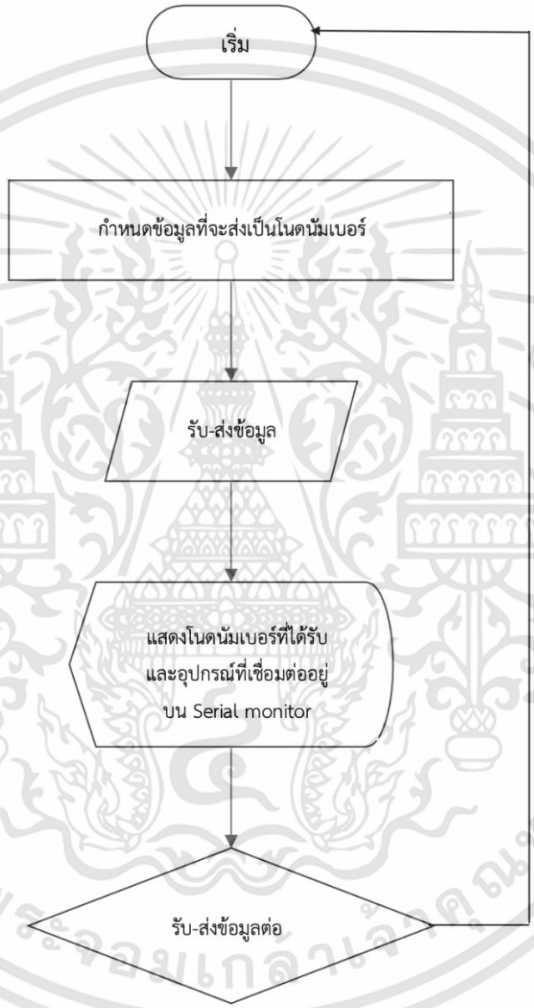
ในการออกแบบการทำงานส่วนนี้ได้ทำการเชื่อมต่อ ESP32 CAM และ FT232RL เนื่องจากบอร์ด ESP32 CAM ไม่มีส่วนของ USB TTL สำหรับการอัปโหลดโปรแกรมลงบอร์ด จึงต้องต่อสายไฟกับโมดูล USB TTL เพิ่ม ซึ่งบอร์ด FT232RL เป็นบอร์ดสำหรับเชื่อมต่อกับบอร์ดที่ไม่มีช่องสำหรับเชื่อมต่อ USB เพื่อให้อัปโหลดโปรแกรมได้ โดยมีการเชื่อมต่อขาระหว่างสองบอร์ดนี้ ดังแสดงในตารางที่ 3.4

ตารางที่ 3.4 การเชื่อมต่อขาระหว่าง ESP32 CAM และ FT232RL

ESP32 CAM	FT232RL
3.3 V	VCC
GND	GND
UOR	Tx
UOT	Rx
IO0 – GND	

3.1.1.5 การออกแบบการเชื่อมต่ออุปกรณ์ ESP32 ผ่านโครงข่าย Mesh

ทำการออกแบบการเชื่อมต่ออุปกรณ์ ESP32 ผ่านโครงข่ายแบบ Mesh โดยแผนภาพการทำงานของโปรแกรมการเชื่อมต่ออุปกรณ์ ESP32 ผ่านโครงข่ายแบบ Mesh แสดงได้ดังในรูปที่ 3.4



รูปที่ 3.4 แผนภาพการทำงานของโปรแกรมการเชื่อมต่ออุปกรณ์ ESP32 ผ่านโครงข่ายแบบ Mesh

จากรูปที่ 3.2 แสดงแผนผังการทำงานของโปรแกรมการเชื่อมต่อระหว่างอุปกรณ์ ESP32 CAM ผ่านโครงข่ายแบบเมชโดยได้มีการกำหนดข้อมูลที่ใช้ในการส่งโดยจะการกำหนด MESH_PREFIX ซึ่งทำหน้าที่คล้ายกับ SSID เป็นการกำหนดชื่อโครงข่ายแบบเมช MESH_PASSWORD เป็นการกำหนดรหัสของโครงข่ายแบบเมชภายใต้ PREFIX เดียวกัน และ

กำหนด MESH_PORT โดยจะกำหนด 555 ในการใช้โครงข่ายแบบเมช ส่วนตัวข้อความที่จะใช้ส่งจะให้ ESP32 CAM แต่ละตัวส่งโนดไอดีของตัวเอง เพื่อที่จะทำให้ทราบว่าในโครงข่ายเมชเดียวกันมีอุปกรณ์ ESP32 CAM โนดไอดีเลขอะไรเชื่อมต่อกันอยู่

3.1.1.6 การออกแบบการเชื่อมต่ออุปกรณ์แต่ละโนด

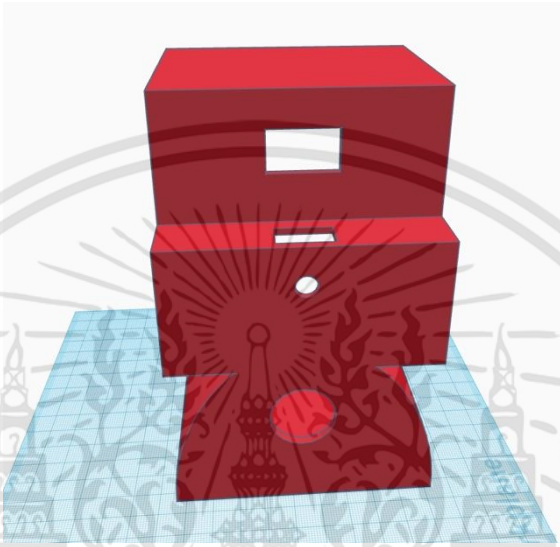
ในการออกแบบแต่ละโนดได้นำอุปกรณ์ทั้งหมดมาเชื่อมต่อกันเพื่อให้สามารถทำงานร่วมกันได้ตามการออกแบบการทำงาน โดยการเชื่อมต่อระหว่างอุปกรณ์ทั้งหมดแสดงดังตารางที่ 3.5

ตารางที่ 3.5 การเชื่อมต่อระหว่างอุปกรณ์ในแต่ละโนด

OLED	DS3231	HC-SR501	ESP32	ESP32 CAM	FT232RL
			Rx2	OUT	Rx
			Tx2	OUR	Tx
		OUT	D2		
SDA	SDA		D21		
SCL	SCL		D22		
GND	GND	GND	GND	GND	GND
VCC	VCC	VCC	Vin 3V3	5V	VCC

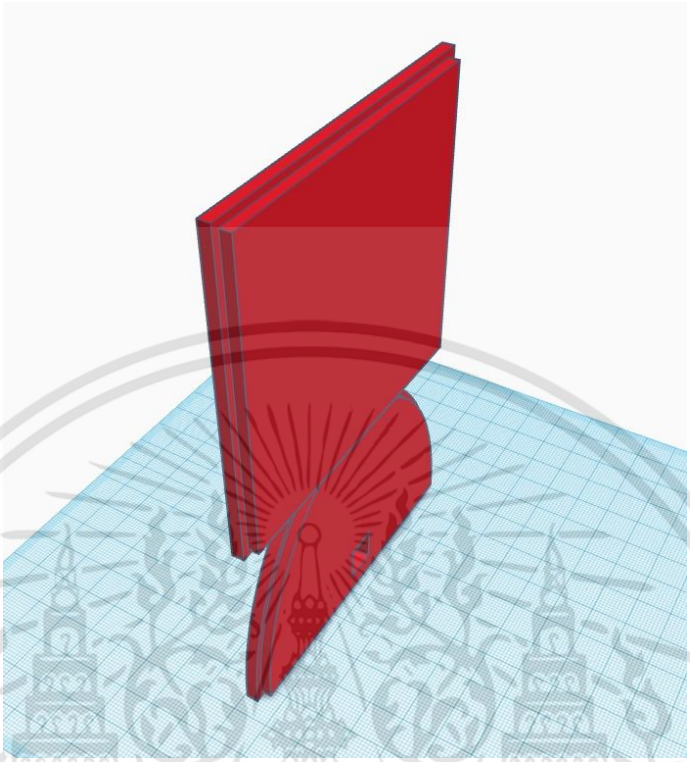
3.1.1.6 การออกแบบกล่องบรรจุอุปกรณ์

กล่องบรรจุวงจรแต่ละโนดจะมีขนาดกว้าง x ยาว x สูง เท่ากับ 96 x 97 x 146.5 มิลลิเมตร แสดงได้ดังรูปที่ 3.5 ถึง รูปที่ 3.8



รูปที่ 3.5 ด้านหน้าของการออกแบบกล่องบรรจุอุปกรณ์

รูปที่ 3.6 ด้านหลังของกล่องบรรจุอุปกรณ์



รูปที่ 3.7 ฝาปิดของกล่องอุปกรณ์

รูปที่ 3.8 ฝาปิดช่อง SD card ของกล่องใส่อุปกรณ์

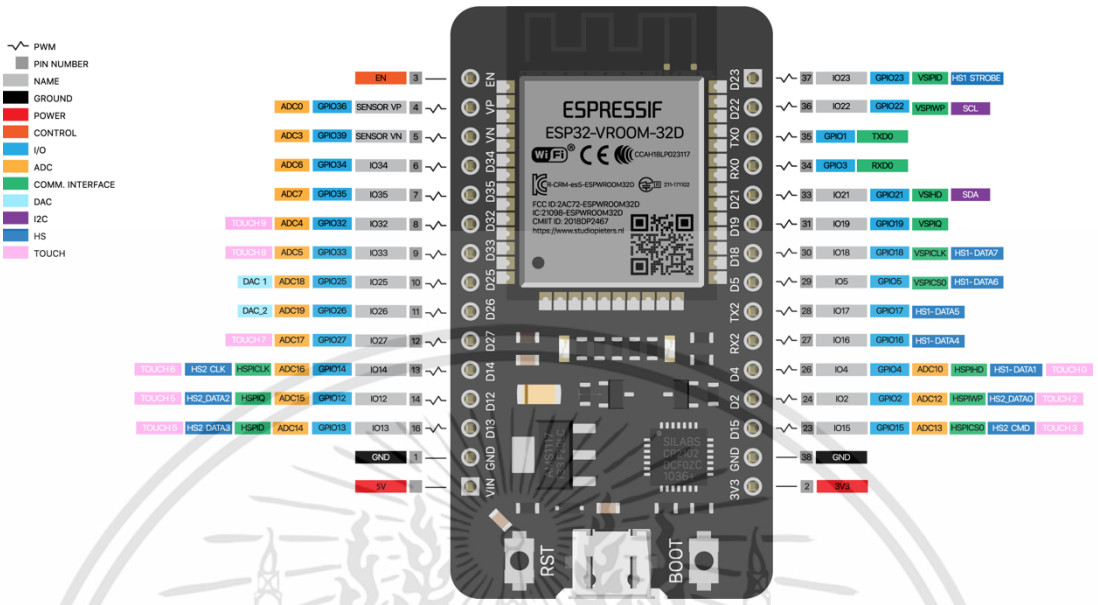
3.2 เครื่องมือที่ใช้ในการทดลอง

3.2.1 ESP32

ตัวบอร์ด ESP32 ที่นำมาใช้งานเป็นตัวบอร์ดที่ประกอบด้วยชิป ESP-WROOM-32 มีวงจร Regulator รับไฟได้ที่ 3.7 ถึง 12 โวลต์ สามารถปรับแรงดันให้คงที่ 3.3 โวลต์ เพื่อจ่ายไฟให้กับชิป ESP32 มีส่วนของวงจร USB TTL ใช้ชิป CP2102 สำหรับเชื่อมต่อกับคอมพิวเตอร์ขณะอัปโหลดโค้ดผ่านสาย Micro USB แสดงได้ดังรูปที่ 3.9

รูปที่ 3.9 ส่วนประกอบภายในบอร์ด ESP32 [11]

จากรูปที่ 3.9 แสดงส่วนประกอบของบอร์ด ESP32 ในส่วนของชิป ESP-WROOM-32, CP2102 และ Voltage regulator โดยการทำงานของบอร์ดเป็นการทำงานแบบ Dual Core มีโปรเซสเซอร์สองตัวทำงานได้พร้อมกัน ทำงานแบบ 32 บิต และ มีขาทั้งหมด 30 ขา ข้างละ 15 ขา แสดงได้ดังรูปที่ 3.10



รูปที่ 3.10 ขาของบอร์ด ESP32 [11]

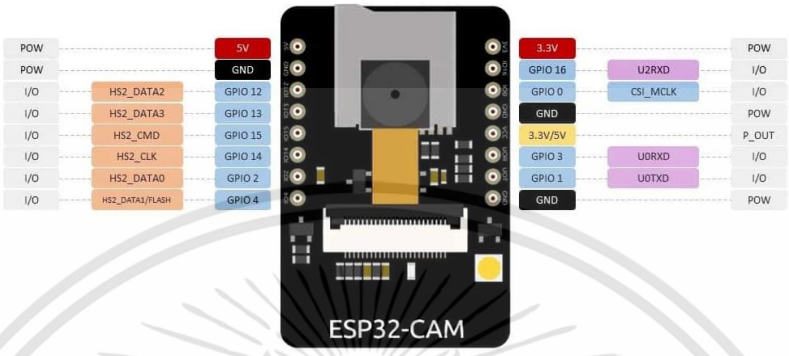
จากรูปที่ 3.10 แสดงขาบอร์ด ESP32 ซึ่งมีทั้งหมด 30 ขา ข้างละ 15 ขา โดยบอร์ด ESP32 สามารถอัปโหลดโปรแกรมโดยใช้ Arduino IDE โดยมีรูปแบบการใช้งานเหมือน Arduino ทั่วไป นอกจากนี้ยังสามารถใช้งานร่วมกับโปรแกรมอื่น ๆ ได้อีก เช่น Epressif IDE, Micropython, JavaScript และ LUA เป็นต้น

3.2.2 ESP32 CAM

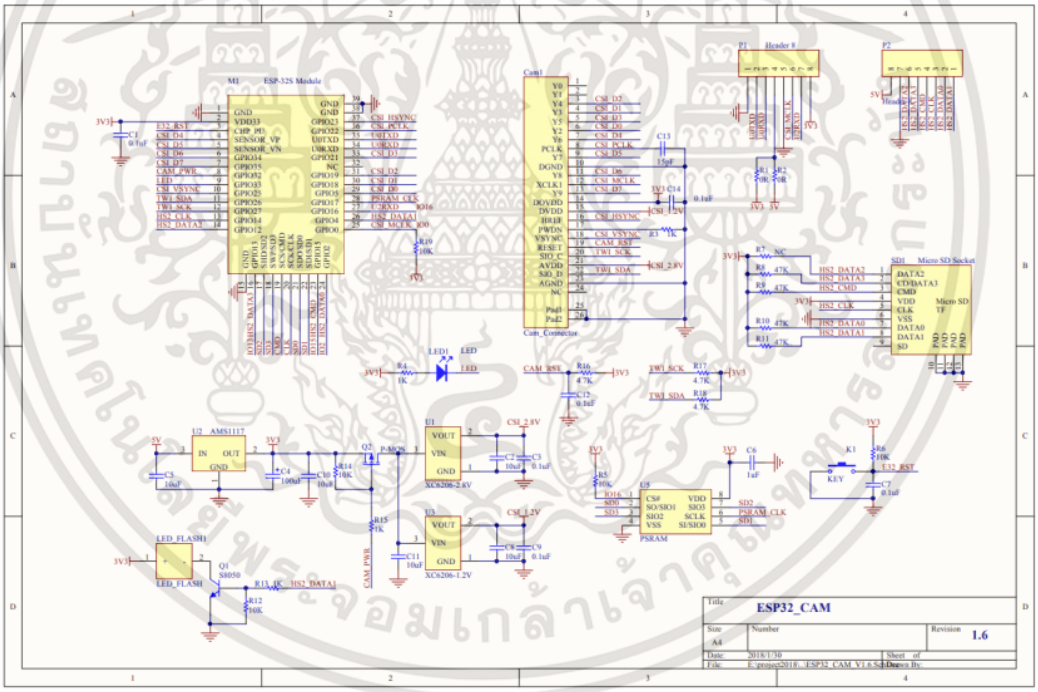
รูปที่ 3.11 ESP32 CAM [12]

โมดูล ESP32 CAM ซึ่งเป็นโมดูลที่สามารถใช้งานร่วมกับกับบอร์ดต่าง ๆ รวมทั้ง Arduino โดยภายในโมดูลของ ESP32 CAM เป็นชิฟ ESP32-S และประกอบด้วยไมโครคอนโทรลเลอร์ภายในตัว แสดงได้ดังรูปที่ 3.11 สามารถเชื่อมต่อไวไฟและบลูทูธ (Bluetooth) ได้ รวมทั้งมีกล้องวิดีโอซึ่งเป็นกล้อง OV1640 และช่องเสียบ MicroSD Card สำหรับ

จัดเก็บข้อมูลภายในบอร์ด โดยมีขา (Pin) บอร์ด แสดงได้ดังในรูปที่ 3.12 และแผนผังภายใน แสดงได้ดังรูปที่ 3.113



รูปที่ 3.12 ขาของ ESP32 CAM [13]



รูปที่ 3.13 แผนผังภายในโมดูล ESP32 CAM [14]

จากรูปที่ 3.12 และรูปที่ 3.13 แสดงขาและแผนผังภายในโมดูล ESP32 CAM ตามลำดับ โดย ESP32 CAM ประกอบด้วยขา 16 ขา ซึ่งรายละเอียดขาต่าง ๆ ที่สำคัญมีดังนี้

- ขา Power

ESP32 CAM ประกอบด้วยขา GND 3 ขา, ขา Power 2 ขา คือ 3.3 โวลต์ และ 5 โวลต์ โดยสามารถจ่ายไฟให้กับ ESP32 CAM ผ่าน Pin 3.3 โวลต์ หรือ 5 โวลต์ โดยในรูปที่ 3.12 ระบุขา Power เป็นสีแดง

- ขา Output

จากรูปที่ 3.12 แสดงขา Output ด้วยสีเหลือง โดยบนโมดูลจะระบุเป็น VCC สามารถจ่ายเอาต์พุต ได้ 3.3 โวลต์ หรือ 5 โวลต์

- ขา Serial

จากรูปที่ 3.13 ที่ขา GPIO 1 และ GPIO 3 เป็น Serial Pin โดยบนโมดูลจะระบุเป็นขา TX และ RX ตามลำดับเนื่องจาก ESP32 CAM ไม่มีโปรแกรมภายในตัว จึงจำเป็นต้องใช้ขา Serial ในการสื่อสารกับบอร์ดและอัปโหลดโค้ด

- ขา GPIO 0

จากรูปที่ 3.12 ขา GPIO 0 บนโมดูลจะระบุเป็นขา IO0 ซึ่งมีหน้าที่ในการระบุว่า ESP32 CAM อยู่ในโหมด Flashing หรือไม่ เมื่อขา GPIO 0 เชื่อมต่อกับขา GND แล้ว ESP32 CAM จะเข้าสู่โหมด Flashing และทำให้สามารถอัปโหลดโค้ดไปยังบอร์ดได้ โดยการเชื่อมต่อของ MicroSD Card กับโมดูล ESP32 CAM แสดงได้ดังตารางที่ 3.6

ตารางที่ 3.6 การเชื่อมต่อขาระหว่าง MicroSD Card กับโมดูล ESP32 CAM

MicroSD card	ESP32 CAM
CLK	GPIO 14
CMD	GPIO 15
DATA0	GPIO 2
DATA1 / flashlight	GPIO 4
DATA2	GPIO 12
DATA3	GPIO 13

จากตารางที่ 3.6 แสดงขาที่ใช้เชื่อมต่อระหว่าง MicroSD Card และ ESP32 CAM ซึ่งมีทั้งหมด 16 ขา โดยมีรายละเอียด ดังนี้

- ไฟแฟลช

บนโมดูล ESP32 CAM มีไฟ LED ภายในตัวซึ่งสามารถใช้เป็นแฟลชได้เมื่อถ่ายภาพ โดย LED นั้นเชื่อมต่อภายในกับขา GPIO 4 แสดงได้ดังรูปที่ 3.13

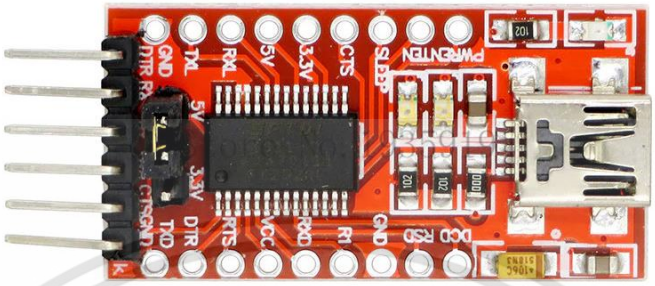
- การเชื่อมต่อระหว่างกล้องกับโมดูล ESP32 CAM

กล้องวิดีโอบนโมดูล ESP32 CAM เป็นกล้อง OV1640 โดยการเชื่อมต่อของกล้องกับโมดูล ESP32 CAM แสดงได้ดังตารางที่ 3.7

ตารางที่ 3.7 การเชื่อมต่อขาระหว่างกล้องวิดีโอและโมดูล ESP32 CAM

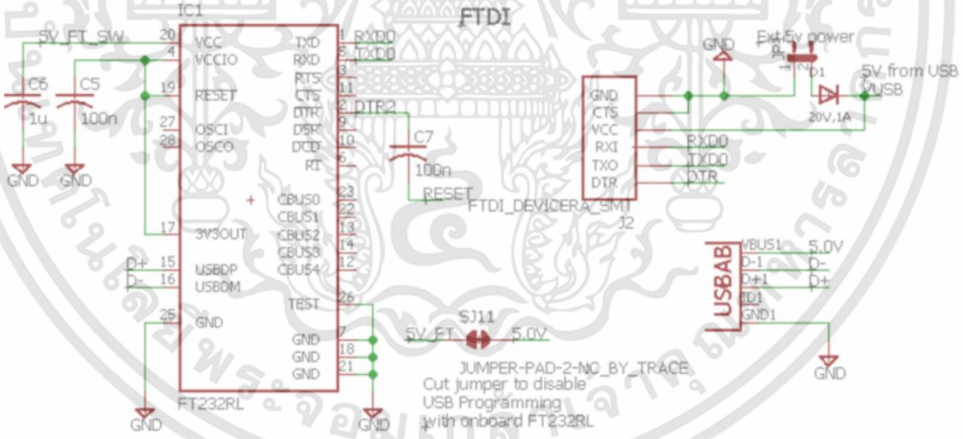
OV2640 CAMERA	ESP32	Variable name in code
D0	GPIO 5	Y2_GPIO_NUM
D1	GPIO 18	Y3_GPIO_NUM
D2	GPIO 19	Y4_GPIO_NUM
D3	GPIO 21	Y5_GPIO_NUM
D4	GPIO 36	Y6_GPIO_NUM
D5	GPIO 39	Y7_GPIO_NUM
D6	GPIO 34	Y8_GPIO_NUM
D7	GPIO 35	Y9_GPIO_NUM
XCLK	GPIO 0	XCLK_GPIO_NUM
PCLK	GPIO 22	PCLK_GPIO_NUM
VSYNC	GPIO 25	VSYNC_GPIO_NUM
HREF	GPIO 23	HREF_GPIO_NUM
SDA	GPIO 26	SIOD_GPIO_NUM
SCL	GPIO 27	SIOC_GPIO_NUM
POWER PIN	GPIO 32	PWDN_GPIO_NUM

3.2.3 FT232RL



รูปที่ 3.14 โมดูล FT232RL FTDI [15]

จากรูปที่ 3.14 แสดงโมดูล FT232RL FTDI ซึ่งเป็นโมดูลสำหรับแปลง USB เป็น TTL โดยใช้ไอซีเบอร์ FT232RL ของ FTDI ซึ่งมีขา DTR ใช้สำหรับการอัปโหลดโค้ดและสามารถเลือกแรงดันได้ทั้ง 3.3 โวลต์ และ 5 โวลต์ และประกอบด้วยขา ต่าง ๆ ได้แก่ RXD , TXD , CTR , DCD , DSR , RTS , DTR , RI และ GND โดยแผนผังภายในของโมดูล FT232RL FTDI แสดงได้ดังรูปที่ 3.15



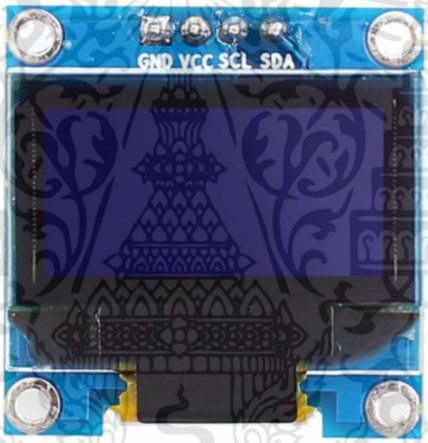
รูปที่ 3.15 แผนผังภายในโมดูล FT232RL FTDI [16]

3.2.4 จอแสดงผล OLED (Organic Light-Emitting Diode)

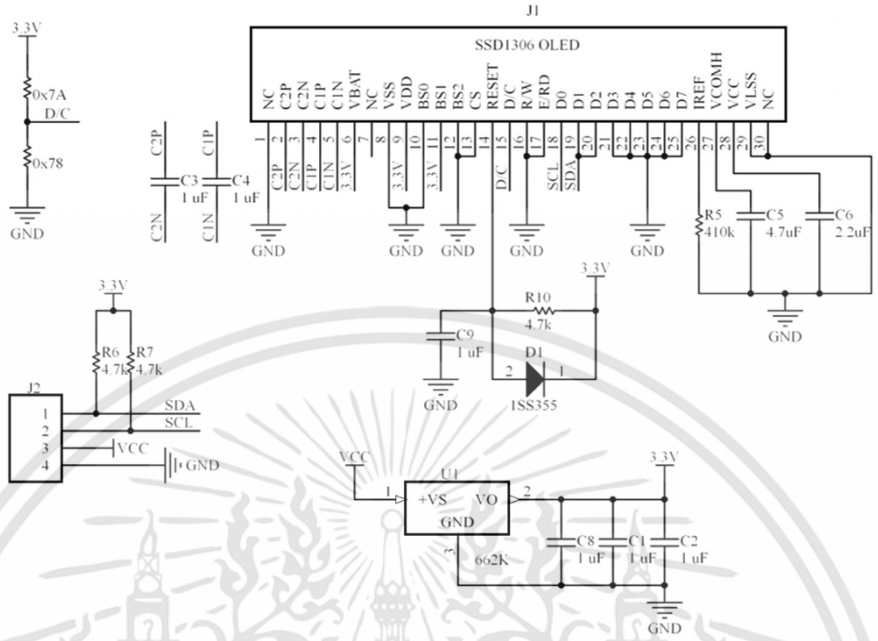
จอแสดงผล OLED เป็นจอแสดงผลที่สร้างจากวัสดุสารกึ่งตัวนำอินทรี (Organic Semiconductor) ที่มีลักษณะเป็นชั้นสารกึ่งตัวนำบาง ๆ อยู่ระหว่างขั้วบวก (Anode) และขั้วลบ (Cathode) และสามารถเปล่งแสงได้เมื่อมีกระแสไฟฟ้าไหลผ่าน การเปล่งแสงนี้ เรียกว่า กระบวน

การอิเล็คโทรลูมิเนสเซนส์ (Electroluminescence) จอแสดงผล OLED มีข้อแตกต่างจากจอแสดงผล LCD (Liquid Crystal Display) คือ จอแสดงผล OLED ไม่มีวงจรที่สร้างแสง Backlight จึงทำให้จอมีความหนาน้อยกว่าและเบากว่า, ใช้กำลังไฟฟ้าต่ำ และไม่มีการเปล่งแสงในบริเวณที่ต้องการให้เป็นสีดำ

ในปริญญานิพนธ์นี้ได้ใช้งานโมดูลจอแสดงผล OLED ขนาดเล็ก ซึ่งมีขนาด 128x64 พิกเซล โดยมีการเชื่อมต่อแบบ I^2C โดยโมดูล OLED ที่ได้นำมาใช้งานมีขนาด 0.9 นิ้ว สามารถแสดงผลได้สองสี คือ สีเหลืองและสีฟ้า โดยมีชิปภายใน คือ SSD1306 และสามารถเลือกจ่ายไฟได้ 3.3 โวลต์ หรือ 5 โวลต์ จอแสดงผล OLED แสดงดังรูปที่ 3.16 และแผนผังภายในจอแสดงผล OLED แสดงได้ดังรูปที่ 3.17



รูปที่ 3.16 จอแสดงผล OLED [17]



รูปที่ 3.17 แผนผังภายในของจอแสดงผล OLED [18]

จากรูปที่ 3.16 และรูปที่ 3.17 แสดงภาพของจอแสดงผล OLED และแผนผังภายในของจอแสดงผล OLED ตามลำดับ โดยบนโมดูลประกอบด้วยทั้งหมด 6 ขา ดังนี้

GND : กราวด์

VCC : แหล่งจ่ายไฟ 3.3 V หรือ 5 V

SCL : นาฬิกา CLK

SDA : ข้อมูล MOSI

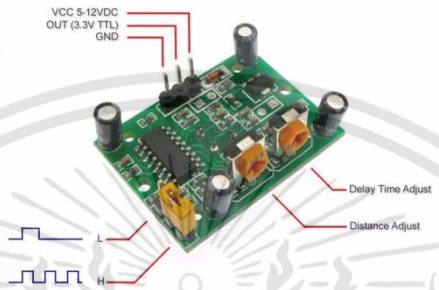
RST : รีเซ็ต

DC : ข้อมูล/คำสั่ง

3.2.5 เซนเซอร์ PIR

เซนเซอร์ PIR อุปกรณ์เซ็นเซอร์ชนิดหนึ่งที่ใช้ตรวจจับคลื่นรังสีอินฟราเรดที่แพร่จากมนุษย์หรือสัตว์ ที่มีการเคลื่อนไหว ทำให้ผู้ใช้งานนิยมนำเซ็นเซอร์นี้มาประยุกต์ใช้งานกันเป็นอย่างมากในตรวจจับการเคลื่อนไหวของสิ่งมีชีวิตหรือตรวจจับการบุกรุกในงานรักษาความปลอดภัย [19] สามารถปรับความหน่วงของเวลาในการตรวจจับครั้งต่อไปได้และปรับระยะทางการตรวจจับได้ 3 ถึง 7 เมตร ขา OUT จะให้เอาต์พุตออกมาในรูปของดิจิตอล สามารถปรับลักษณะของสัญญาณขา

ออกได้จากการเปลี่ยนจั๊มเปอร์ มีรูปแบบสัญญาณเอาต์พุต 2 แบบ คือ สัญญาณแบบคลื่นพัลส์ต่อเนื่อง และสัญญาณลอจิก 1 ค้างไว้จนกว่าจะไม่สามารถจับความเคลื่อนไหวได้จึงจะกลับมาเป็นลอจิก 0 โดยส่วนประกอบของเซนเซอร์ PIR แสดงได้ดังรูปที่ 3.18



รูปที่ 3.18 ส่วนประกอบของเซนเซอร์ PIR [20]

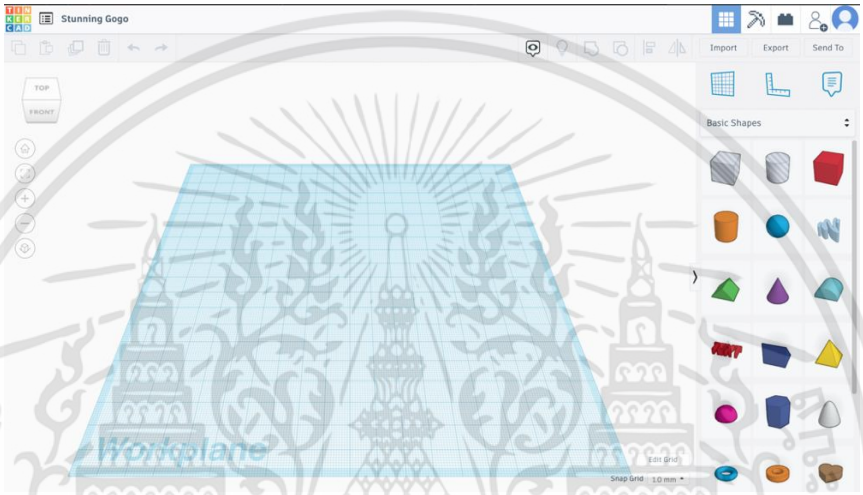
3.2.6 เซนเซอร์ RTC

เซนเซอร์ RTC ที่ใช้เป็นโมดูล DS3231 ซึ่งเป็นโมดูลที่มีความถูกต้องแม่นยำสูง เนื่องจากภายในมีวงจรวัดอุณหภูมิเพื่อนำอุณหภูมิจากสภาพแวดล้อมมาคำนวณชดเชยความถี่ของ Crystal ที่ถูกรบกวนจากอุณหภูมิภายนอก สามารถตั้งค่า วัน เวลา ได้และสามารถตั้งค่าการแสดงผลเวลาแบบ 24 ชั่วโมงหรือแบบ 12 ชั่วโมงได้ [21] แสดงได้ดังรูปที่ 3.19

รูปที่ 3.19 เซนเซอร์ PIR โมดูล DS3231 [22]

3.2.7 TINKERCAD

โปรแกรม Tinkercad เป็นโปรแกรมออกแบบที่ทำงานบนเว็บเบราว์เซอร์ ซึ่งสามารถใช้งานได้ง่าย และสามารถบันทึกไฟล์เพื่อนำไปพิมพ์กับเครื่องพิมพ์สามมิติได้ โดยโปรแกรม Tinkercad เป็นโปรแกรมที่พัฒนาโดยบริษัท Autodesk [23] โดยตัวอย่างหน้าโปรแกรม Tinkercad แสดงได้ดังรูปที่ 3.20



รูปที่ 3.20 ตัวอย่างหน้าโปรแกรม Tinkercad [24]

3.2.8 LINE Notify

LINE Notify เป็นบริการรับการแจ้งเตือนจากบัญชีทางการในรูปแบบ API สำหรับสร้างการแจ้งเตือนแบบข้อความไปยังกลุ่มหรือบัญชีส่วนตัวได้โดยไม่เสียค่าใช้จ่าย ยกเว้นกรณีที่เชื่อมต่อกับเว็บเซอร์วิสอื่น ๆ ซึ่งอาจมีบางบริการที่ใช้ได้เฉพาะบัญชีแบบเสียค่าบริการเท่านั้น [25]

3.2.9 ภาษา C++

C++ คือ ภาษา C programming language รุ่นใหม่ เป็นภาษาในการเขียนโปรแกรม ถูกพัฒนาโดย Dr.Bjarne Stroustup ภาษา C++ เกิดจากแนวคิดในการเพิ่มประสิทธิภาพภาษา CC โดยได้นำความสามารถของภาษา C มาพัฒนาให้เป็นโปรแกรมภาษาที่มีความเป็น Object Oriented Programming (โปรแกรมเชิงวัตถุ)

ภาษา C++ ถูกออกแบบมาสำหรับการทำงานภายใต้ระบบปฏิบัติการ UNIX ด้วยภาษา C++ ผู้เขียนโปรแกรมสามารถเขียนโปรแกรมได้อย่างมีประสิทธิภาพมากขึ้น นอกจากนี้การเขียนโปรแกรมเพื่อให้สามารถนำกลับมาใช้ได้ใหม่ (reusability) ก็สามารถทำได้ง่ายขึ้น [26]

3.2.10 Arduino IDE

Arduino IDE ส่วน IDE ย่อมาจก (Integrated Development Environment) คือ ส่วนเสริมของระบบการพัฒนา หรือตัวช่วยต่าง ๆ ที่จะคอยช่วยเหลือ Developer หรือช่วยเหลือคนที่พัฒนา Application เพื่อให้เกิดความรวดเร็ว ถูกต้อง แม่นยำ ตรวจสอบระบบที่จัดทำได้ ทำให้การพัฒนางานต่าง ๆ เร็วมากขึ้น [27]

Arduino IDE เป็นโปรแกรมที่สามารถใช้งานได้ในลักษณะ Open source ซึ่ง Arduino IDE จะทำหน้าที่ติดต่อระหว่างคอมพิวเตอร์ ทั้งระบบ Windows , Mac OS X หรือ Linux กับบอร์ด Arduino ซึ่งโปรแกรมนี้ออกแบบให้ง่ายต่อการเขียนโค้ดและอัปโหลดโปรแกรมที่อัปโหลดเข้าสู่บอร์ด Arduino

3.2.11 ภาษา HTML

HTML ย่อมาจากคำว่า Hypertext Markup Language เป็นภาษาหลักที่ใช้ในการสร้างไฟล์เว็บเพจ โดยมีแนวคิดจากการสร้างเอกสารไฮเปอร์เท็กซ์ (Hypertext Document) ซึ่งพัฒนาขึ้นมาจากภาษา SGML (Standard Generalized Markup Language) โดย Tim Berners-Lee เป็นภาษามาตรฐานที่ใช้พัฒนาเอกสารในรูปแบบของเว็บเพจเผยแพร่บนระบบเครือข่ายอินเทอร์เน็ต มีโครงสร้างการเขียนที่อาศัยตัวกำกับ เรียกว่า แท็ก (Tag) ควบคุมการแสดงผลของข้อความ , รูปภาพ หรือวัตถุอื่น ๆ สามารถเรียกใช้เอกสารเหล่านี้โดยการใช้โปรแกรมเว็บบราวเซอร์ (Web Browser) เช่น Mozilla Firefox , Opera , Nescape navigator และ Internet Explorer เป็นต้น

ในปัจจุบัน HTML เป็นมาตรฐานหนึ่งของ ISO ซึ่งจัดการโดย World Wide Web Consortium (W3C) ในปัจจุบัน ทาง W3C ผลักดันรูปแบบของ HTML แบบใหม่ที่เรียกว่า XHTML ซึ่งเป็นลักษณะของโครงสร้าง XML แบบหนึ่ง ที่มีหลักเกณฑ์ในการกำหนดโครงสร้างของโปรแกรมที่มีรูปแบบที่มาตรฐานกว่ามาทดแทนใช้ HTML รุ่น 4.01 ที่ใช้ในปัจจุบัน [28]

3.2.12 Visual Studio Code

VS Code หรือ Visual Studio Code จากบริษัทไมโครซอฟต์ เป็นโปรแกรมประเภท Editor ใช้ในการแก้ไขโค้ดที่มีขนาดเล็กแต่มีประสิทธิภาพสูง เป็น OpenSource โปรแกรมจึงสามารถนำมาใช้งานได้โดยไม่มีค่าใช้จ่าย เหมาะสำหรับนักพัฒนาโปรแกรมที่ต้องการใช้งานหลายแพลตฟอร์ม ซึ่งรองรับการใช้งานบน Windows, macOS และ Linux รองรับหลายภาษาทั้ง

JavaScript , TypeScript และ Node.js สามารถนำมาใช้งานได้ง่ายไม่ซับซ้อน มีเครื่องมือและส่วนขยายต่าง ๆ ในการใช้งานหลากหลาย รองรับการเปิดใช้งานภาษาอื่น ๆ ทั้งภาษา C++ , C# , Java , Python , PHP หรือ Go และสามารถปรับเปลี่ยน Themes ได้ มีส่วน Debugger และ Commands เป็นต้น [29]

3.3 การจัดเก็บผลการทดลอง

3.3.1 ทดสอบการทำงานของอุปกรณ์

การทดสอบการทำงานของอุปกรณ์เพื่อตรวจสอบการทำงานของอุปกรณ์ และกำหนดการตั้งค่าต่าง ๆ ให้เข้ากับสภาพแวดล้อม และสามารถเชื่อมต่อกับอุปกรณ์อื่น ๆ เพื่อทำงานร่วมกันได้ โดยมีการทดสอบทั้งหมดดังนี้

3.3.1.1 ทดสอบการทำงานของเซนเซอร์ตรวจจับคลื่นรังสีอินฟราเรด

3.3.1.2 ทดสอบการทำงานของเซนเซอร์ Real Time Clock

3.3.1.3 ทดสอบการทำงานของจอแสดงผล OLED

3.3.1.4 ทดสอบการทำงานของ ESP32 CAM ผ่านการสตรีมวิดีโอ

3.3.1.5 ทดสอบการบันทึกภาพนิ่งจากอุปกรณ์ ESP32 CAM

3.3.1.6 ทดสอบการเชื่อมต่อระหว่างอุปกรณ์ ESP32 ผ่านโครงข่ายแบบ Mesh

3.3.1.7 ทดสอบการแจ้งเตือนผ่าน Line Notify

3.3.1.8 ออกแบบส่วนแสดงผล Web Server

3.3.1.9 การสร้างกล่องใส่อุปกรณ์สำหรับสร้างเป็นกล้องวงจรปิด

3.3.1.10 ทดสอบการทำงานร่วมกันของอุปกรณ์ทั้งหมดและประกอบลงบนกล่องอุปกรณ์ที่ออกแบบ

3.3.1.11 ทดสอบระยะทางสูงสุดที่อุปกรณ์สามารถเชื่อมต่อกันได้ผ่านโครงข่ายแบบ Mesh และระยะทางสูงสุดที่อุปกรณ์แต่ละตัวสามารถเชื่อมต่อ Wi-Fi ได้

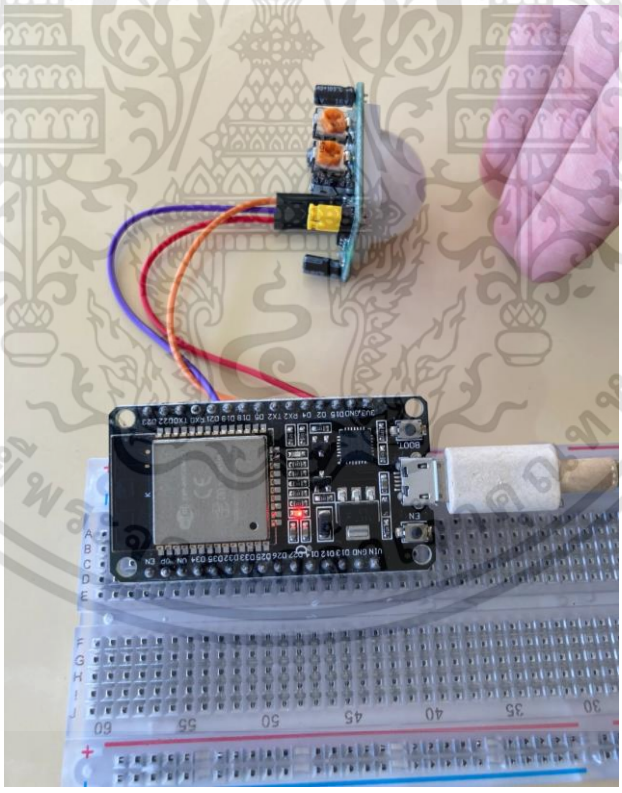
บทที่ 4

ผลการทดลอง

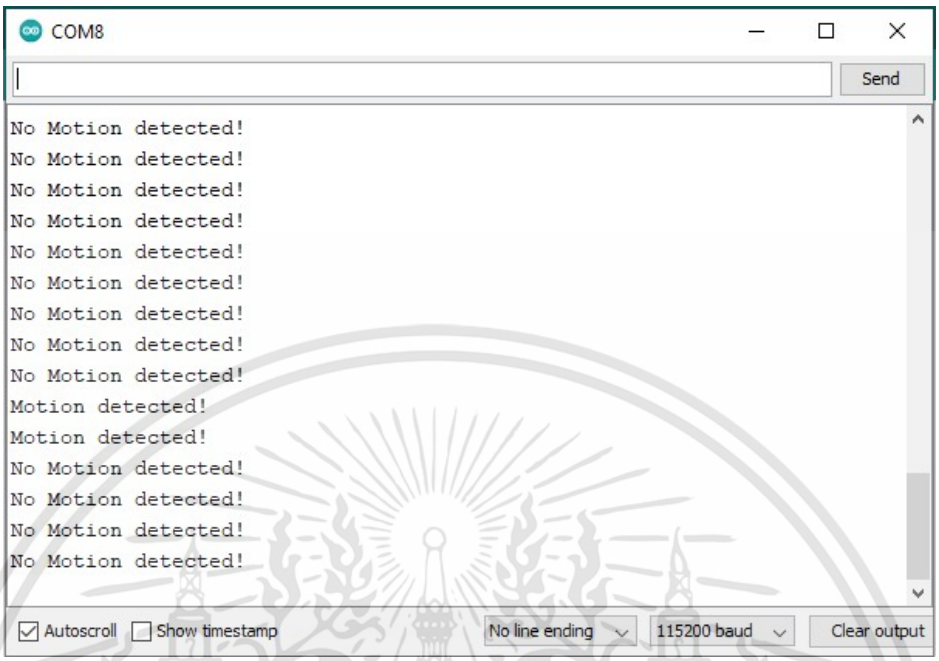
จากขั้นตอนในการออกแบบและการจัดทำปริญญานิพนธ์ในบทที่ 3 ได้มีการจัดเก็บผลการทำงานของระบบออกเป็นส่วน ๆ ดังต่อไปนี้

4.1 ผลทดสอบการทำงานของเซนเซอร์ตรวจจับคลื่นรังสีอินฟราเรด

ทำการทดสอบใช้เซนเซอร์ตรวจจับคลื่นรังสีอินฟราเรดโมดูล HC-SR501 เชื่อมต่อกับ ESP32 โดยการเชื่อมต่อขาระหว่างอุปกรณ์ทั้งสอง แสดงได้ดังตารางที่ 3.1 เพื่อทดสอบการเพื่อทดสอบการตรวจจับการเคลื่อนไหว โดยการเชื่อมต่อระหว่างเซนเซอร์ตรวจจับคลื่นรังสีอินฟราเรดโมดูล HC-SR501 และ ESP32 แสดงได้ดังรูปที่ 4.1 เมื่อทำการส่งข้อมูลยังอุปกรณ์ ESP32 และนำค่ามาแสดงบน Serial Monitor สามารถแสดงค่าได้ แสดงได้ดังรูปที่ 4.2



รูปที่ 4.1 การเชื่อมต่อ HC-SR501 เข้ากับ ESP32

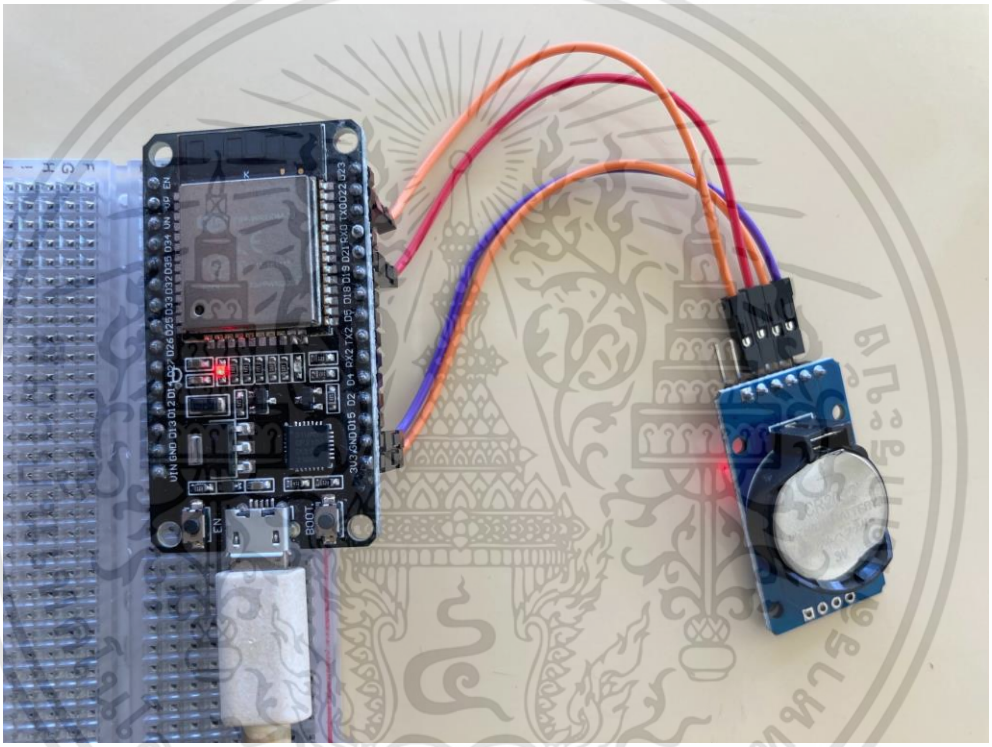


รูปที่ 4.2 Serial Monitor ของ ESP32 แสดงค่าที่รับจาก HC-SR501

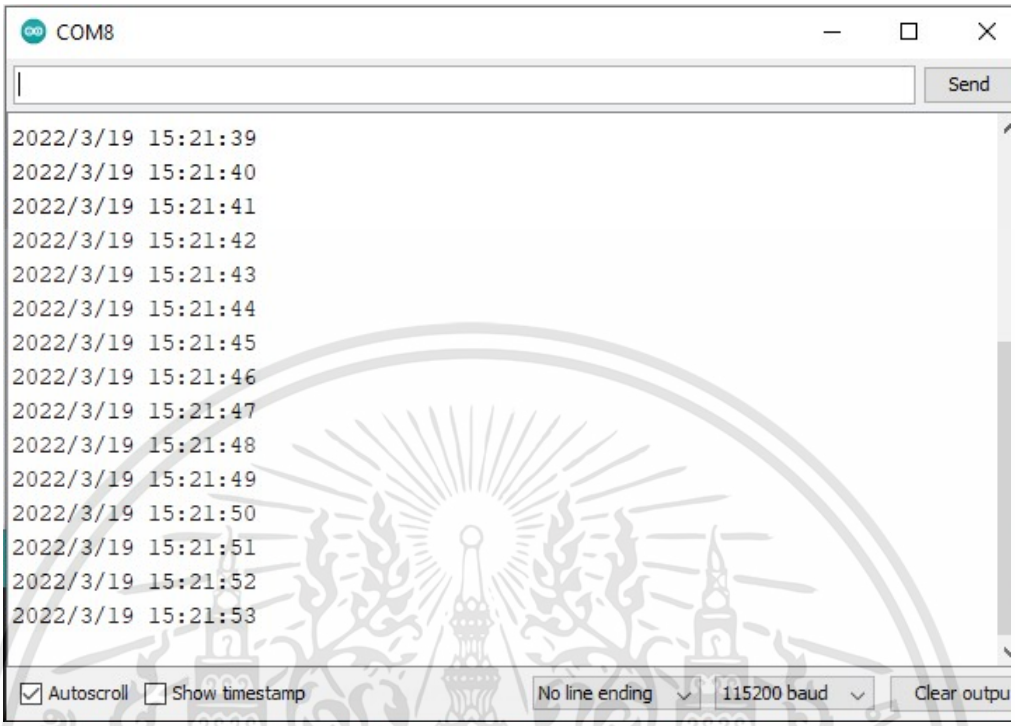
จากรูปที่ 4.1 แสดงการเชื่อมต่อ HC-SR501 เข้ากับ ESP32 โดยมีมือที่เคลื่อนไหวอยู่ข้างหน้าโมดูล HC-SR501 เพื่อทดสอบการตรวจจับการเคลื่อนไหว โดยรูปที่ 4.2 แสดง Serial Monitor ของ ESP32 แสดงค่าที่รับจาก HC-SR501 พบว่าเมื่อ โมดูล HC-SR501 มีการตรวจจับการเคลื่อนไหว จะส่งข้อมูลให้ Serial Monitor ของ ESP32 แสดงข้อความ Motion detected! และเมื่อโมดูล HC-SR501 ไม่มีการตรวจจับการเคลื่อนไหว จะส่งข้อมูลให้ Serial Monitor ของ ESP32 แสดงข้อความ No Motion detected!

4.2 ผลทดสอบการทำงานของเซนเซอร์ Real Time Clock

ทำการทดสอบใช้เซนเซอร์ Real Time Clock โมดูล DS3231 เชื่อมต่อกับ ESP32 โดยการเชื่อมต่อขาระหว่างอุปกรณ์ทั้งสอง แสดงได้ดังตารางที่ 3.2 เพื่อทดสอบการดึงค่าเวลาปัจจุบัน โดยการเชื่อมต่อระหว่างเซนเซอร์ Real Time Clock โมดูล DS3231 และ ESP32 แสดงได้ดังรูปที่ 4.3 เมื่อทำการส่งข้อมูลยังอุปกรณ์ ESP32 และนำค่ามาแสดงบน Serial Monitor สามารถแสดงค่าได้ แสดงได้ดังรูปที่ 4.4



รูปที่ 4.3 การเชื่อมต่อ DS3231 เข้ากับ ESP32

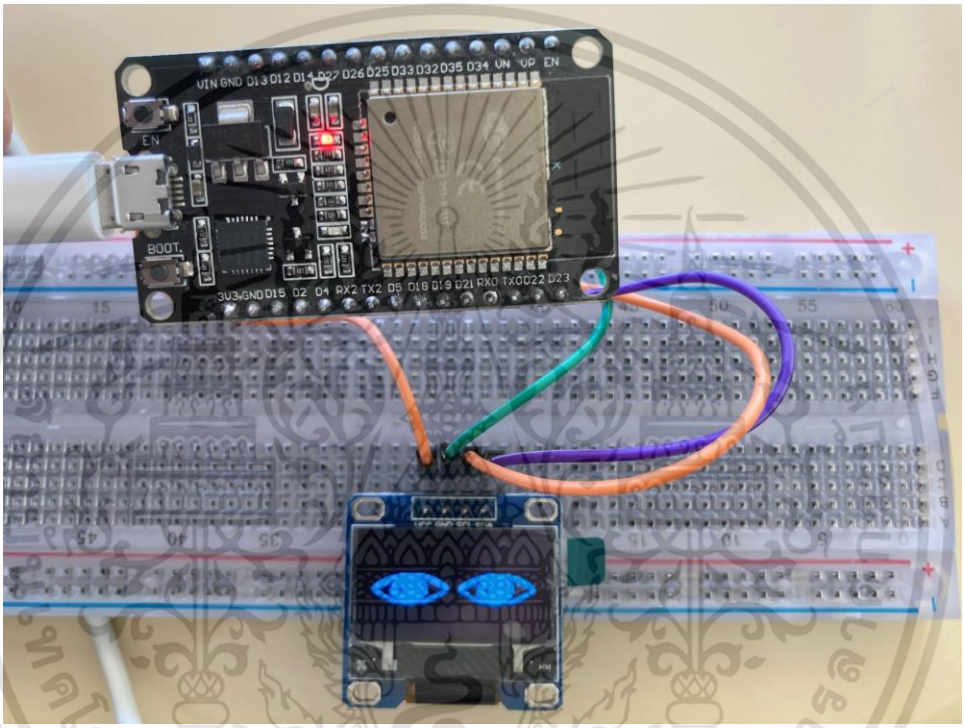


รูปที่ 4.4 Serial Monitor ของ ESP32 แสดงค่าที่รับจาก DS3231

จากรูปที่ 4.3 แสดงการเชื่อมต่อ DS3231 เข้ากับ ESP32 เพื่อทดสอบการดึงค่าเวลาปัจจุบัน โดยรูปที่ 4.4 แสดง Serial Monitor ของ ESP32 แสดงค่าที่รับจาก DS3231 พบว่าเมื่อโมดูล DS32311 มีการทำงาน จะส่งข้อมูลให้ Serial Monitor ของ ESP32 แสดงข้อความเป็นค่าวันที่และเวลาปัจจุบัน

4.3 ทดสอบการทำงานของจอแสดงผล OLED

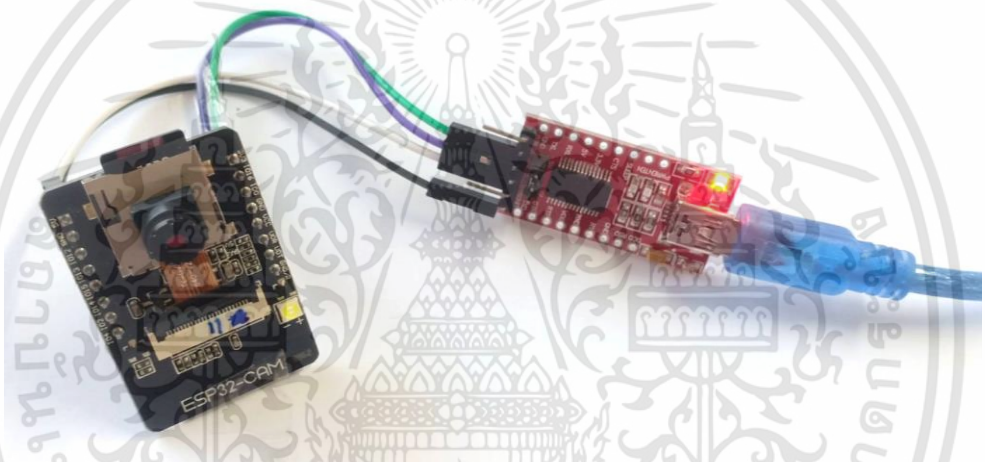
ทำการทดสอบใช้จอแสดงผล OLED เชื่อมต่อกับ ESP32 โดยการเชื่อมต่อขาระหว่างอุปกรณ์ทั้งสอง แสดงได้ดังตารางที่ 3.3 เพื่อทดสอบการแสดงภาพบนหน้าจอ OLED โดยการเชื่อมต่อระหว่างจอแสดงผล OLED และ ESP32 และภาพที่แสดงบนหน้าจอ OLED แสดงได้ดังรูปที่ 4.5



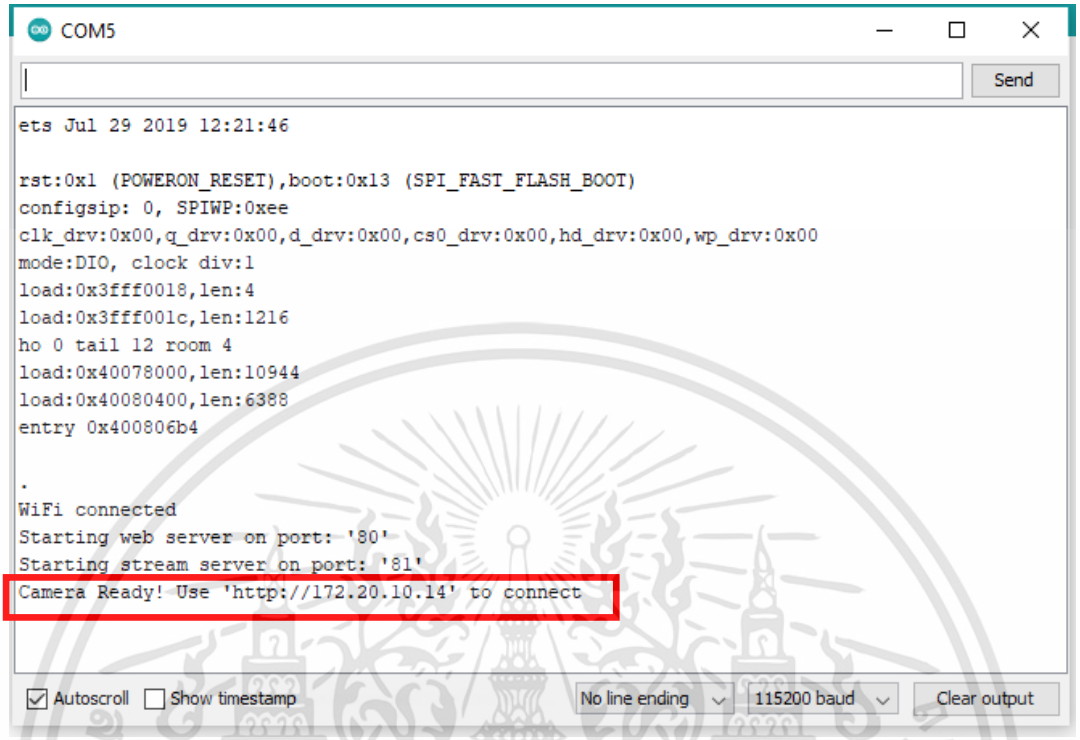
รูปที่ 4.5 การเชื่อมต่อจอแสดงผล OLED เข้ากับ ESP32

4.4 ผลทดสอบการทำงานของ ESP32 CAM ผ่านการสตรีมวิดีโอ

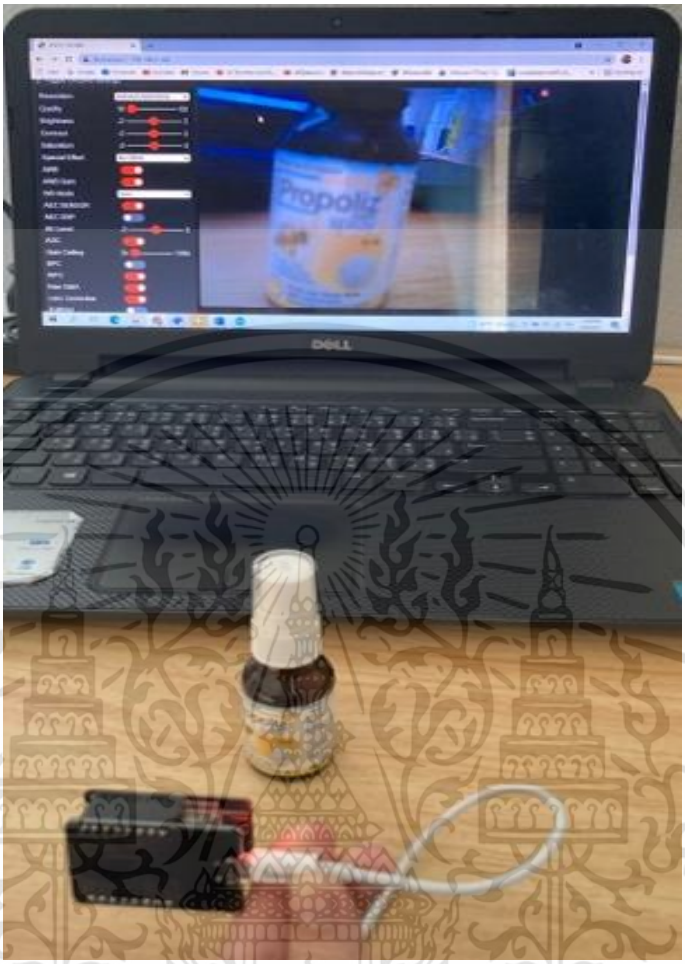
ทำการทดสอบโดยใช้ ESP32 CAM เชื่อมต่อกับ FT232RL โดยการเชื่อมต่อขาระหว่างอุปกรณ์ทั้งสอง แสดงได้ดังตารางที่ 3.4 เพื่ออัปโหลดโปรแกรมเว็บเซิร์ฟเวอร์การสตรีมวิดีโอสำหรับทดสอบการสตรีมวิดีโอของ ESP32 CAM โดยการเชื่อมต่อระหว่าง ESP32 CAM และ FT232RL แสดงได้ดังรูปที่ 4.6 หลังจากอัปโหลดโปรแกรมแล้วอุปกรณ์จะส่งลิงก์เว็บเซิร์ฟเวอร์ผ่าน Serial Monitor แสดงได้ดังรูปที่ 4.7 จากนั้นนำลิงก์ที่ได้ไปเปิดในเว็บเบราว์เซอร์เพื่อดูสตรีมวิดีโอ แสดงได้ดังรูปที่ 4.8



รูปที่ 4.6 การเชื่อมต่อระหว่าง ESP32 CAM และ FT232RL



รูปที่ 4.7 Serial Monitor ของ ESP32 CAM แสดงลิงก์สำหรับเปิดดูสตรีมวิดีโอ

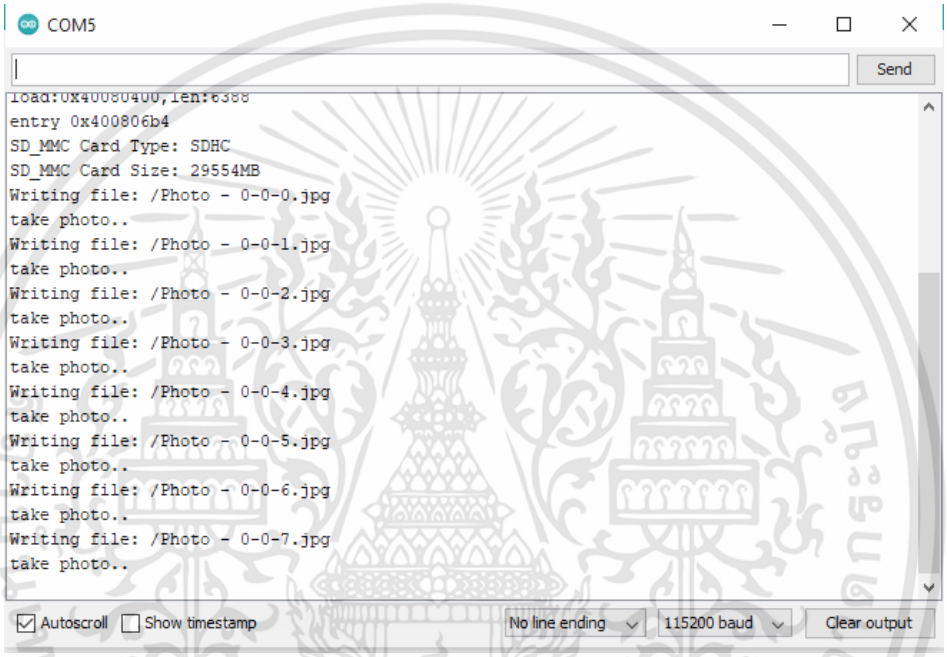


รูปที่ 4.8 นำลิงก์จาก Serial Monitor เปิดบนเว็บเบราว์เซอร์

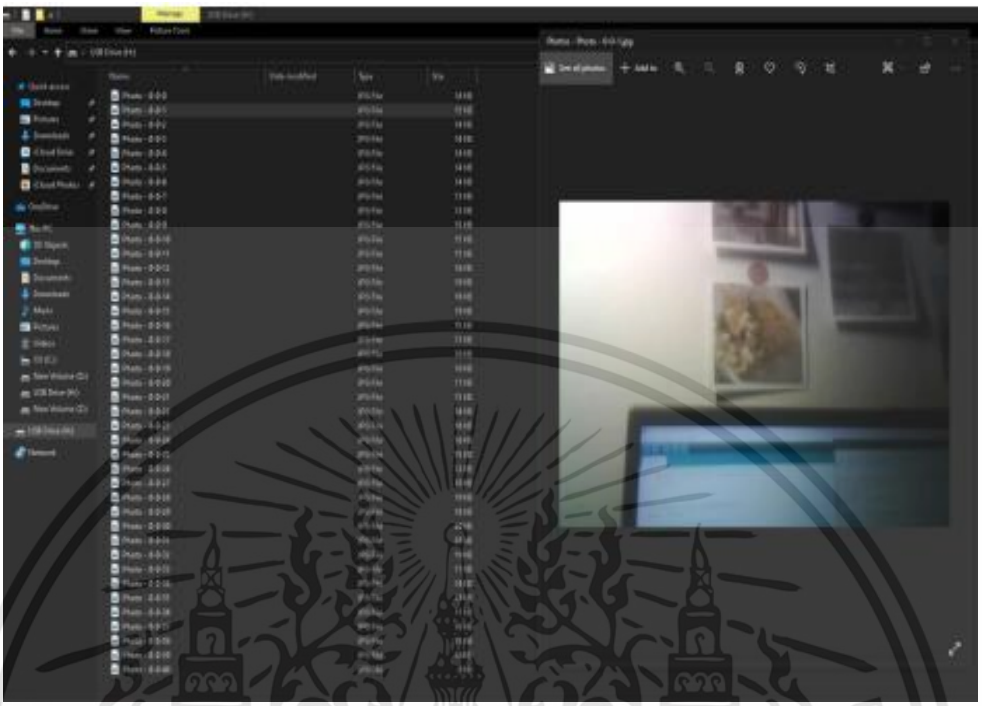
จากรูปที่ 4.6 แสดงการเชื่อมต่อระหว่าง ESP32 CAM และ FT232RL เพื่ออัปโหลดโปรแกรมเว็บเซิร์ฟเวอร์การสตรีมวิดีโอลงบน ESP32 CAM หลังจากทำการอัปโหลดโปรแกรมเสร็จสิ้นแล้ว เมื่อ ESP32 CAM ทำงาน จากรูปที่ 4.7 พบว่าบน Serial monitor ของ ESP32 CAM แสดงลิงก์สำหรับเปิดดูสตรีมวิดีโอจากกล้องของ ESP32 CAM และจากรูปที่ 4.8 แสดงหน้าเว็บหลังจากนำลิงก์ที่ได้เปิดดูการสตรีมวิดีโอของ ESP32 CAM

4.5 ผลทดสอบการบันทึกภาพนิ่งจากอุปกรณ์ ESP32 CAM

ทำการทดสอบอัปโหลดโปรแกรมลงใน ESP32 CAM โดยเชื่อมต่อ ESP32 CAM กับ FT232RL แสดงได้ดังรูปที่ 4.6 เพื่อให้ ESP32 CAM บันทึกภาพนิ่งลงบน SD card ของ ESP32 CAM หลังจากมีการบันทึกภาพนิ่งได้เปิดไฟล์ภาพจาก SD card พบว่า ESP32 CAM สามารถบันทึกภาพนิ่งลงบน SD card ได้ แสดงได้ดังรูปที่ ที่ 4.9 และรูปที่ 4.10



รูปที่ 4.9 Serial Monitor แสดงชื่อไฟล์ภาพนิ่งที่บันทึกจาก ESP32 CAM

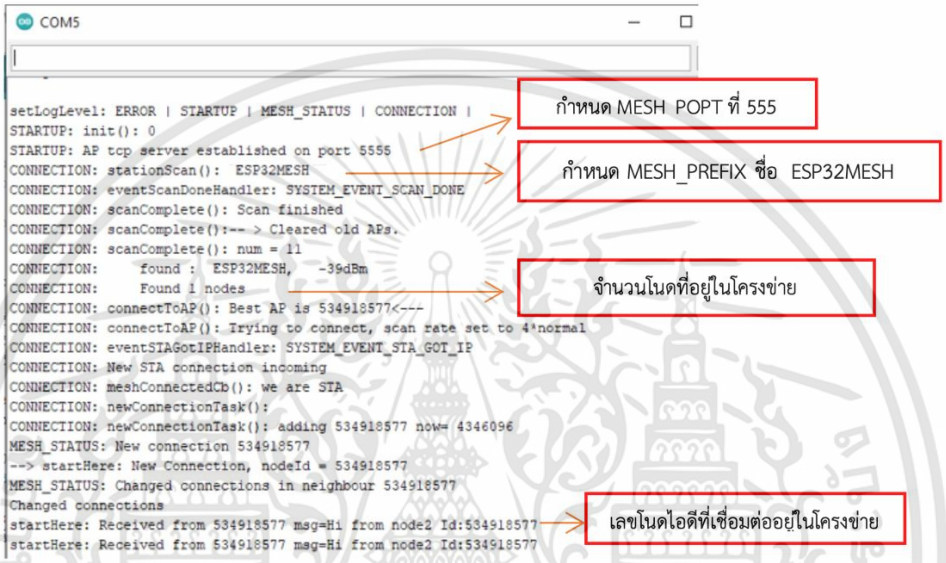


รูปที่ 4.10 ไฟล์ภาพนิ่งที่บันทึกจาก ESP32 CAM บน SD card

จากรูปที่ 4.9 แสดง Serial Monitor ของ ESP32 CAM ซึ่งแสดงชื่อไฟล์ภาพนิ่งที่บันทึกได้ และในรูปที่ 4.10 แสดงไฟล์ภาพนิ่งที่บันทึกจาก ESP32 CAM บน SD card

4.6 ผลทดสอบการเชื่อมต่อระหว่างอุปกรณ์ ESP32 ผ่านโครงข่ายแบบ Mesh

จากการทดลองอัปโหลดโปรแกรมเพื่อเชื่อมต่อระหว่างอุปกรณ์ ESP32 ผ่านโครงข่ายแบบ Mesh โดยใช้ไลบรารี่ Painless mesh ในการใช้งาน โดยให้ ESP32 ส่งข้อความแบบบรอดแคสหากันได้ทั้งหมดในการรับและส่งข้อความ แสดงได้ดังรูปที่ 4.11

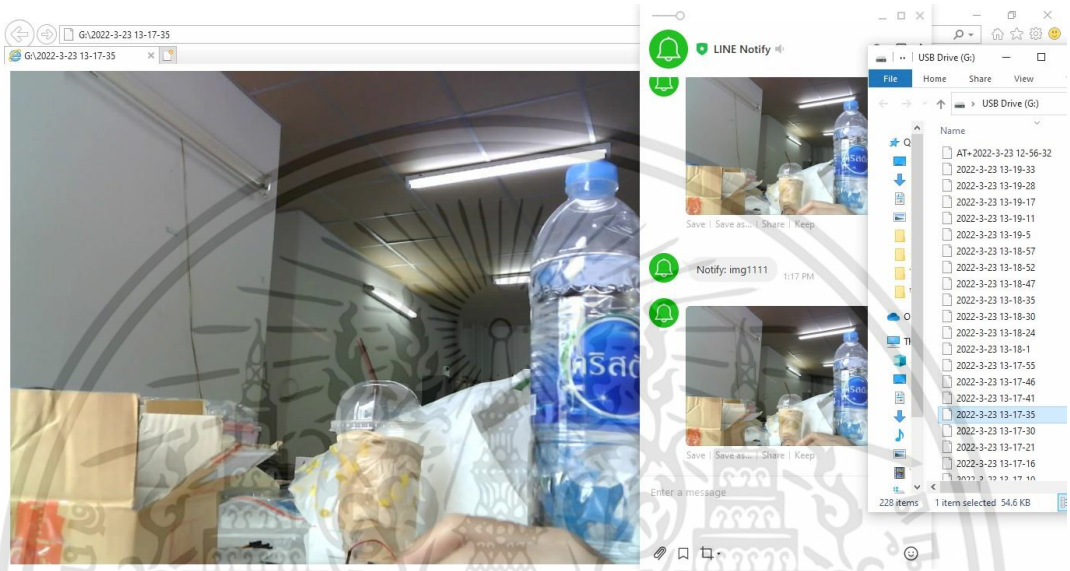


รูปที่ 4.11 Serial Monitor แสดงการเชื่อมต่อระหว่างอุปกรณ์ ESP32 ผ่านโครงข่ายแบบ Mesh

จากรูปที่ 4.11 แสดง Serial Monitor ของ ESP32 ที่เชื่อมต่อกันผ่านโครงข่ายแบบ Mesh โดยแสดงการเชื่อมต่อในโครงข่ายแบบ Mesh ชื่อ ESP32MESH มีการกำหนดไว้ใน MESH_PREFIX การเชื่อมต่อใน port ที่ 5555 ซึ่งจะมีการกำหนดไว้ใน MESH_PORT และแสดงสถานะการเชื่อมต่อหากมีอุปกรณ์เข้ามาเชื่อมต่อใหม่ หรือมีอุปกรณ์ออกจากการเชื่อมต่อภายในโครงข่ายแบบ Mesh เดียวกัน พร้อมกับแสดงจำนวนโนดที่มีการเชื่อมต่ออยู่ในโครงข่าย นอกจากนี้มีการรับส่งข้อมูลเป็นข้อความซึ่งจะแสดงข้อมูลว่ามีการรับข้อความมาจากโนดไอดีหมายเลขใด

4.7 ผลทดสอบการแจ้งเตือนผ่าน Line Notify

จากการทดสอบการแจ้งเตือนผ่าน Line Notify โดยใช้ ESP32 CAM ในการส่งข้อมูลเข้าใน Line Notify แสดงได้ดังรูปที่ 4.12

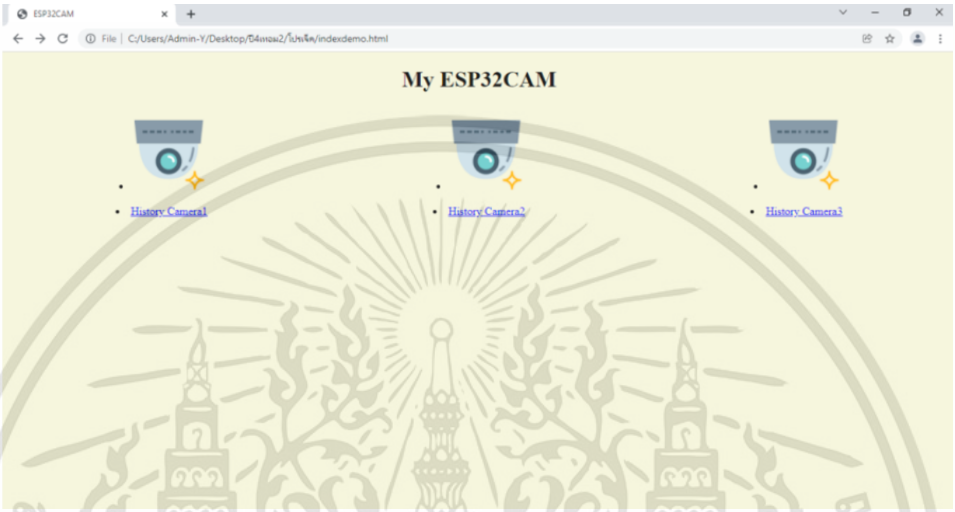


รูปที่ 4.12 การส่งข้อมูลที่ ESP32 CAM บันทึกได้เข้า Line Notify

จากรูปที่ 4.12 แสดงการส่งข้อความที่ ESP32 CAM บันทึกได้เข้า Line Notify โดยส่งเป็นข้อความที่กำหนดและรูปภาพที่บันทึกลงบน SD card ของ ESP32 CAM เข้าใน Line Notify

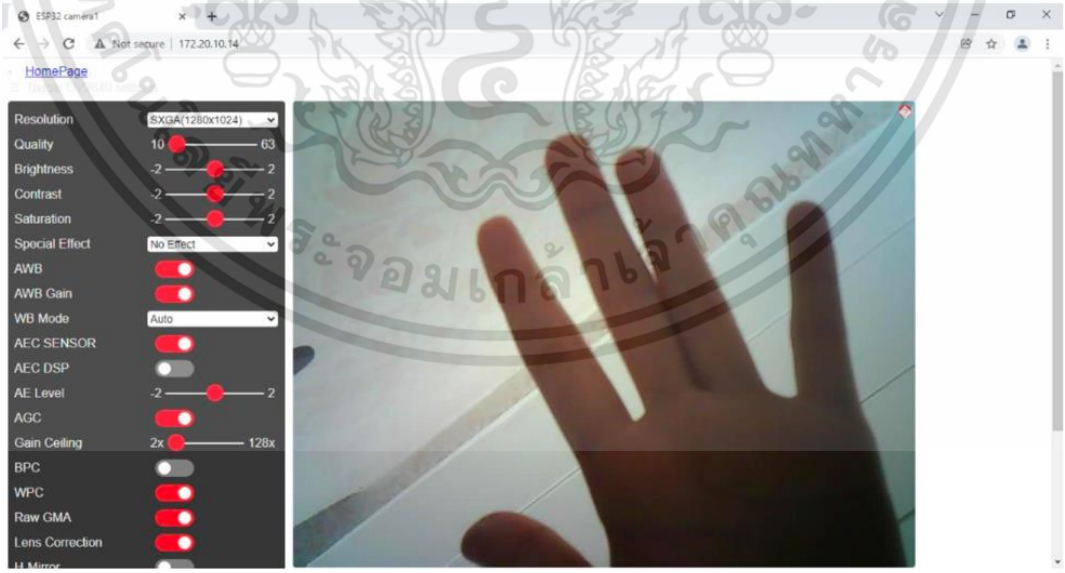
4.8 ผลการออกแบบส่วนแสดงผล Web Server

การออกแบบส่วนแสดงผลเป็นส่วนที่จะรวม ESP32 CAM ทั้ง 3 ตัวเพื่อให้ผู้ใช้งานสามารถใช้ดูวิดีโอแบบเรียลไทม์ได้สะดวกมายิ่งขึ้น แสดงได้ดังรูปที่ 4.13



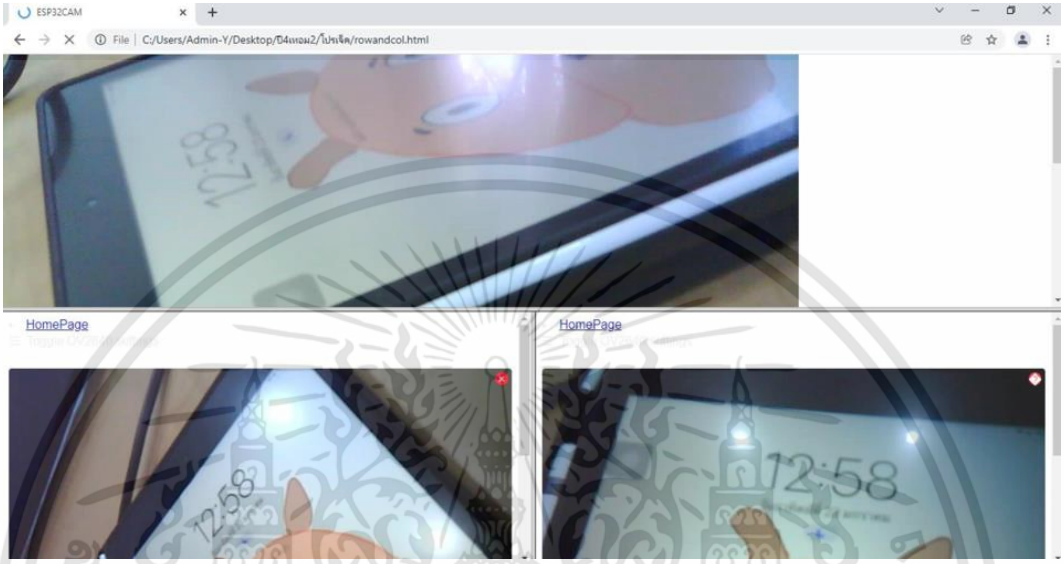
รูปที่ 4.13 Homepage หน้าหลักของ Web Server

จากรูปที่ 4.13 แสดง Homepage หน้าหลักของ Web Server เมื่อเข้ามาในหน้า Homepage จะสามารถเลือกดูภาพวิดีโอเรียลไทม์จาก ESP32 CAM ทั้ง 3 ตัวได้ หากกดรูปกล้องตัวที่ 1 Web Application หน้าต่างของกล้องตัวที่ 1 แสดงได้ดังรูปที่ 4.14



รูปที่ 4.14 วิดีโอเรียลไทม์จากกล้องตัวที่ 1

จากรูปที่ 4.14 แสดงวิดีโอเรียลไทม์จากกล้องตัวที่ 1 ที่กดดูจาก Homepage หน้าหลักของ Web Server หลังจากที่สามารถเปิดวิดีโอเรียลไทม์จากกล้องตัวที่ 1 ได้แล้ว จึงได้ทำการทดลองเปิดวิดีโอเรียลไทม์ พร้อมกัน 3 ตัว แสดงได้ดังรูปที่ 4.15



รูปที่ 4.15 วิดีโอเรียลไทม์พร้อมกัน 3 ตัว โดยถ่ายไปที่วัตถุเดียวกัน

4.9 ผลการสร้างกล่องใส่อุปกรณ์สำหรับสร้างเป็นกล้องวงจรปิด

จากหัวข้อ 3.1.1.6 การออกแบบกล่องใส่อุปกรณ์สำหรับสร้างเป็นกล้องวงจรปิดโดยใช้โปรแกรม Tinkercad ซึ่งหลังจากออกแบบเสร็จสิ้นแล้วได้นำการออกแบบที่ได้ไปพิมพ์ในเครื่องพิมพ์สามมิติ และได้อุปกรณ์ตามที่ได้ออกแบบไว้ซึ่งประกอบไปด้วยชิ้นส่วน 3 ชิ้น แสดงได้ดังรูปที่ 4.16 ถึง รูปที่ 4.18

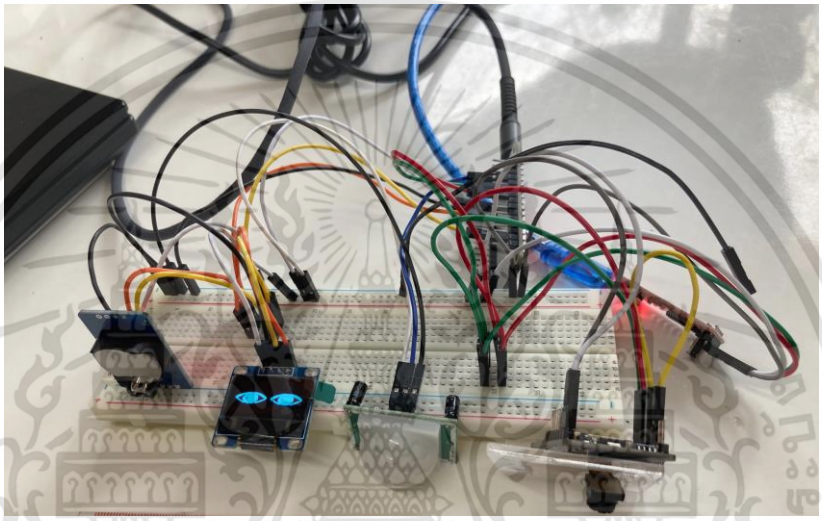
รูปที่ 4.16 ชิ้นส่วนอุปกรณ์ชิ้นที่ 1

รูปที่ 4.17 ชิ้นส่วนอุปกรณ์ชิ้นที่ 2

รูปที่ 4.18 ชิ้นส่วนอุปกรณ์ชิ้นที่ 3

4.10 ผลทดสอบการทำงานร่วมกันของอุปกรณ์ทั้งหมดและประกอบลงบนกล่องอุปกรณ์ที่ออกแบบ

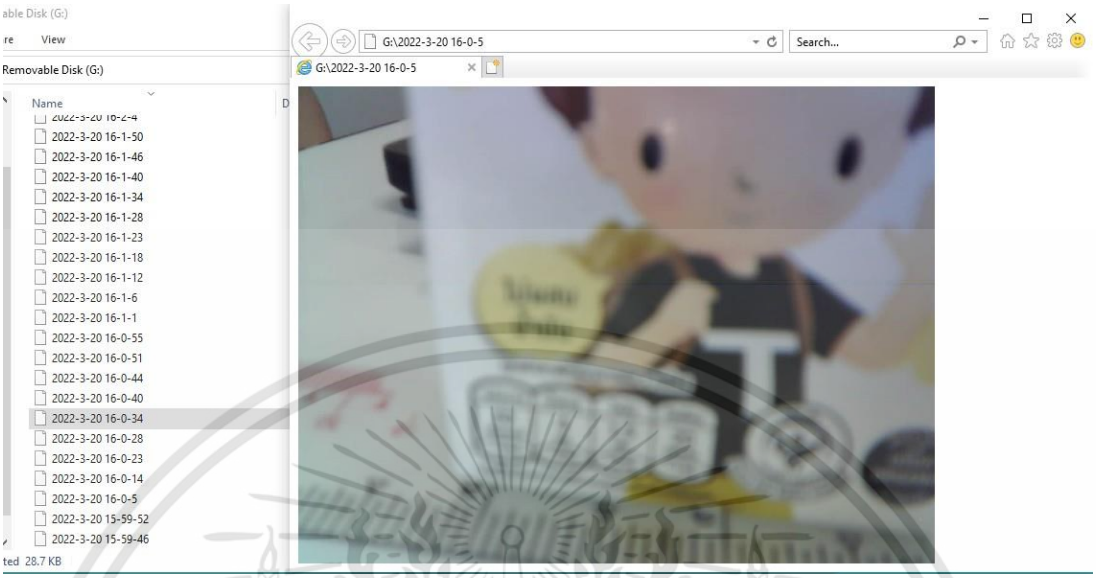
เมื่อทำการทดสอบอุปกรณ์ในแต่ละส่วนแล้ว ได้นำการทำงานของแต่ละส่วนมาประยุกต์ใช้รวมกันโดยเชื่อมต่ออุปกรณ์ทุกส่วนเข้าด้วยกัน ซึ่งการเชื่อมต่อระหว่างอุปกรณ์ แสดงได้ในตารางที่ 3.5 และในรูปที่ 4.19



รูปที่ 4.19 การเชื่อมต่อระหว่างอุปกรณ์ทั้งหมด

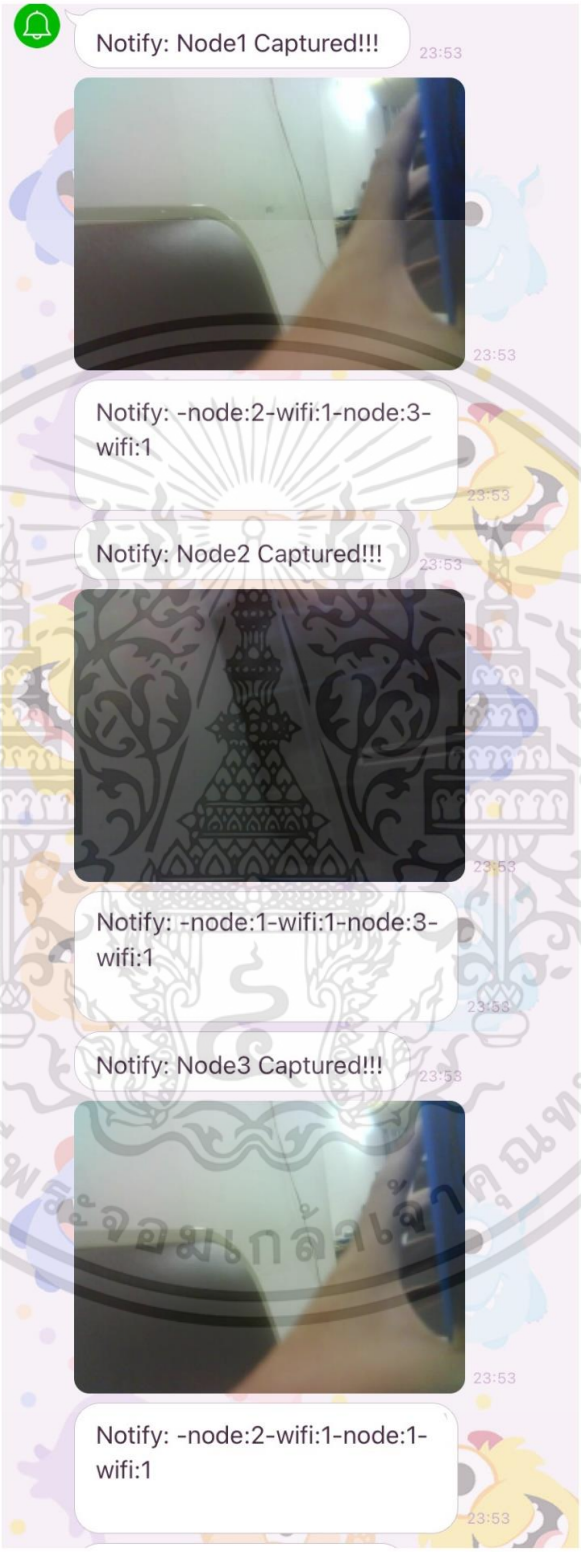
จากรูปที่ 4.19 เป็นการเชื่อมต่ออุปกรณ์ทั้งหมดเข้าด้วยกัน ซึ่งประกอบด้วย ESP32 CAM , ESP32 , FT232RL , จอแสดงผล OLED , เซนเซอร์ตรวจจับคลื่นรังสีอินฟราเรด และเซนเซอร์ Real Time Clock นำมาประกอบเป็นกล้องวงจรปิดโครงข่ายแบบ Mesh ซึ่งมีอุปกรณ์ทั้งหมด 3 ชุด หรือ 3 โนด ในการส่งข้อมูลผ่านโครงข่ายแบบ Mesh

โดยมีหลักการทำงานในแต่ละโนด คือ เซนเซอร์ Real Time Clock จะส่งค่าวัน เวลาไปให้ ESP32 CAM ผ่าน ESP32 ที่เซนเซอร์ได้เชื่อมต่ออยู่ และเมื่อเซนเซอร์ตรวจจับคลื่นรังสีอินฟราเรดสามารถตรวจจับการเคลื่อนไหวได้จะส่งข้อมูลผ่าน ESP32 ไปยัง ESP32 CAM เพื่อให้ ESP32 CAM บันทึกภาพนิ่งลงบน SD card โดยชื่อไฟล์ภาพที่บันทึก ตั้งชื่อตามวันที่และเวลาที่ได้รับมาจากเซนเซอร์ Real Time Clock แสดงได้ดังรูปที่ 4.20



รูปที่ 4.20 ชื่อไฟล์ภาพนิ่งที่ ESP32 CAM บันทึกลงบน SD card

หลังจาก ESP32 CAM บันทึกภาพลงบน SD card จะส่งภาพที่บันทึกได้เข้าไปใน Line Notify และข้อความระบุว่าเป็นภาพมาจากอุปกรณ์โนดใด พร้อมกับข้อความตรวจสอบการเชื่อมต่อของโนดอื่น ๆ ในโครงข่ายแบบ Mesh เดียวกันรวมทั้งระบุการเชื่อมต่อ Wi-Fi ของแต่ละโนด แสดงได้ดังรูปที่ 4.21 ถึงรูปที่ 4.25



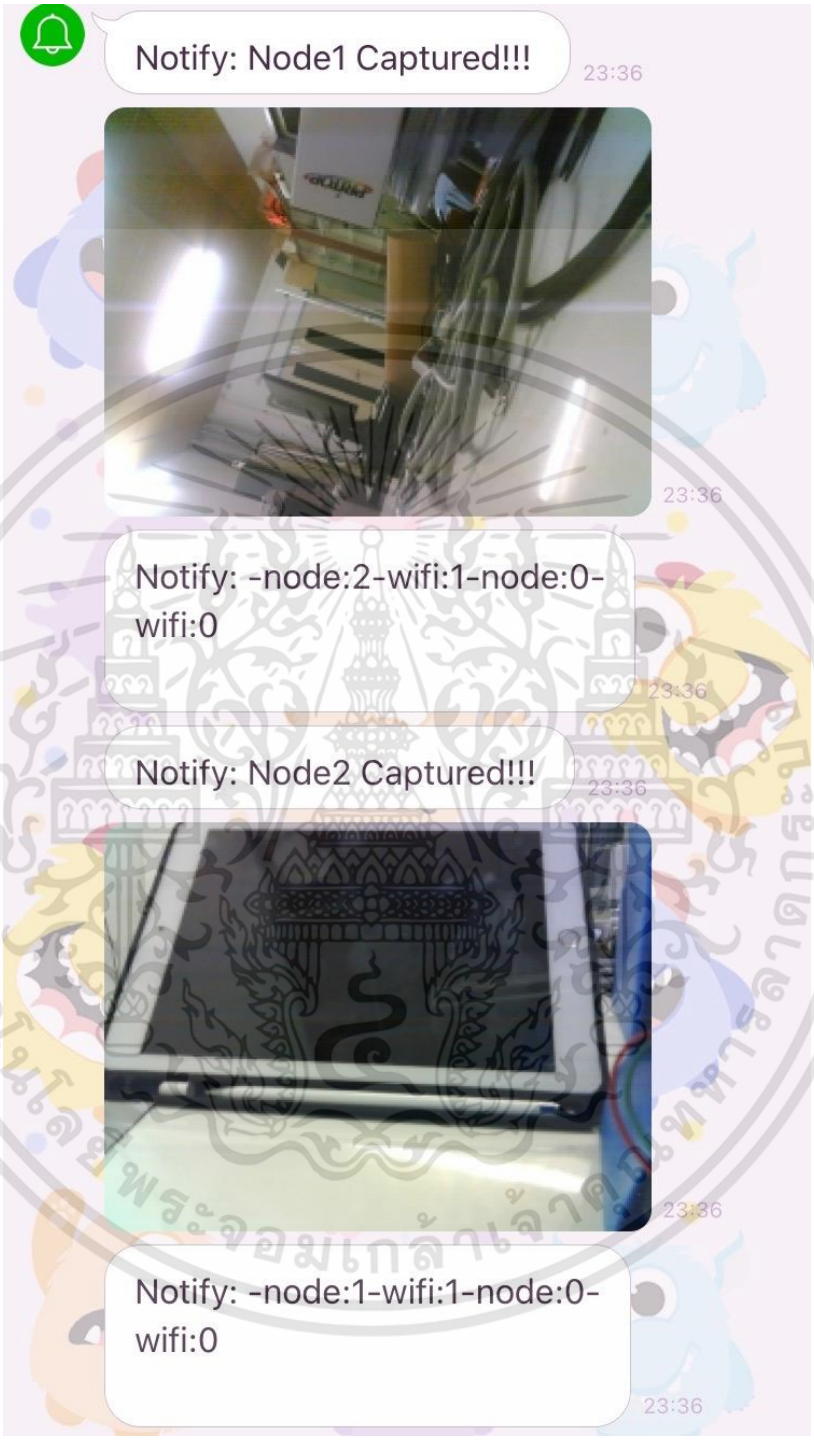
รูปที่ 4.21 ข้อความและรูปภาพที่ทุกโนดส่งเข้า Line Notify

จากรูปที่ 4.21 แสดงข้อความและรูปภาพที่ทุกโนดส่งเข้า Line Notify โดยข้อความ Notify: Node1 Captured!!! , Notify: Node2 Captured!!! และ Notify: Node3 Captured!!! คือ ข้อความที่แจ้งเตือนว่ามีการบันทึกภาพจากโนด 1 , โนด 2 และโนด 3 ตามลำดับ

หลังจากข้อความแจ้งเตือนการบันทึกภาพของแต่ละโนดและภาพที่บันทึกจากแต่ละโนดส่งมา จะส่งมาพร้อมกับอีกข้อความ คือ ข้อความ Notify: -node:2-wifi:1-node:3-wifi:1 ซึ่งหมายถึงโนด 1 มีการเชื่อมต่อกับโนด 2 กับโนด 3 โดยข้อความ wifi:1 ที่อยู่ข้างหลังชื่อโนดที่มีการเชื่อมต่อ แสดงให้เห็นว่าโนด 2 และโนด 3 มีการเชื่อมต่อ Wi-Fi อยู่ด้วย ทำให้เมื่อโนด 2 และโนด 3 เมื่อมีการบันทึกภาพจะสามารถส่งภาพที่บันทึกมาที่ Line Notify ได้

ข้อความ Notify: -node:1-wifi:1-node:3-wifi:1 ซึ่งหมายถึงโนด 2 มีการเชื่อมต่อกับโนด 1 กับโนด 3 โดยข้อความ wifi:1 ที่อยู่ข้างหลังชื่อโนดที่มีการเชื่อมต่อ แสดงให้เห็นว่าโนด 1 และโนด 3 มีการเชื่อมต่อ Wi-Fi อยู่ด้วย ทำให้เมื่อโนด 1 และโนด 3 เมื่อมีการบันทึกภาพจะสามารถส่งภาพที่บันทึกมาที่ Line Notify ได้

ข้อความ Notify: -node:2-wifi:1-node:1-wifi:1 ซึ่งหมายถึงโนด 3 มีการเชื่อมต่อกับโนด 2 กับโนด 1 โดยข้อความ wifi:1 ที่อยู่ข้างหลังชื่อโนดที่มีการเชื่อมต่อ แสดงให้เห็นว่าโนด 2 และโนด 1 มีการเชื่อมต่อ Wi-Fi อยู่ด้วย ทำให้เมื่อโนด 2 และโนด 1 เมื่อมีการบันทึกภาพจะสามารถส่งภาพที่บันทึกมาที่ Line Notify ได้

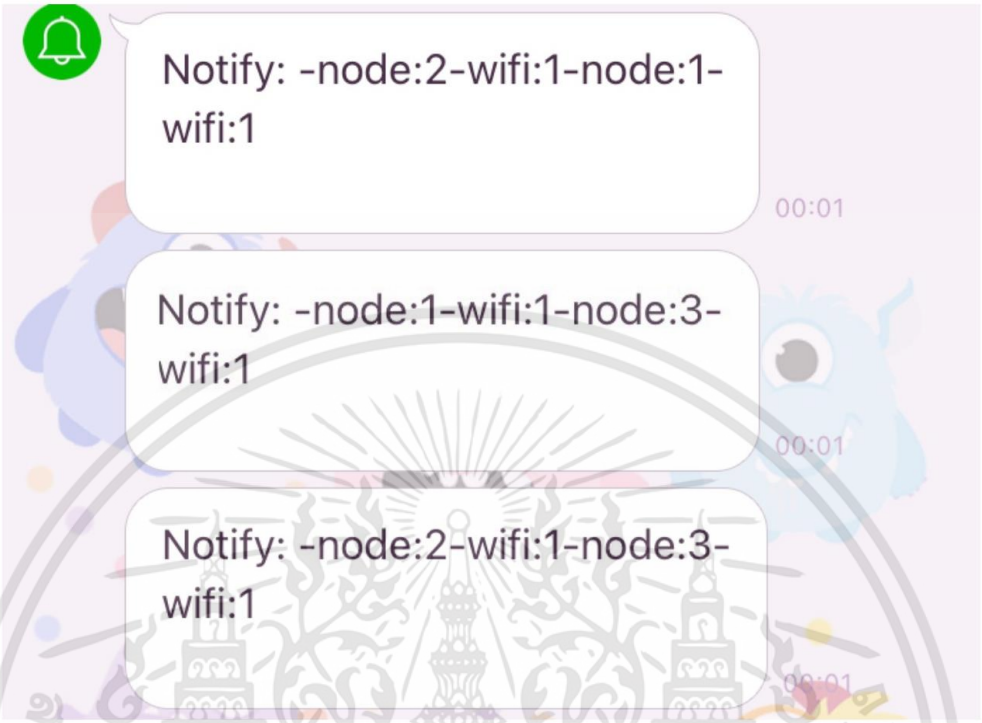


รูปที่ 4.22 ข้อความและรูปภาพที่โนด 1 และโนด 2 ส่งเข้า Line Notify

จากรูปที่ 4.22 ข้อความและรูปภาพที่โนด 1 และโนด 2 ส่งเข้า Line Notify โดยข้อความ Notify: Node1 Captured!!! , Notify: Node2 Captured!!! คือข้อความที่แจ้งเตือนว่ามีการบันทึกภาพจากโนด 1 และโนด 2 ตามลำดับ

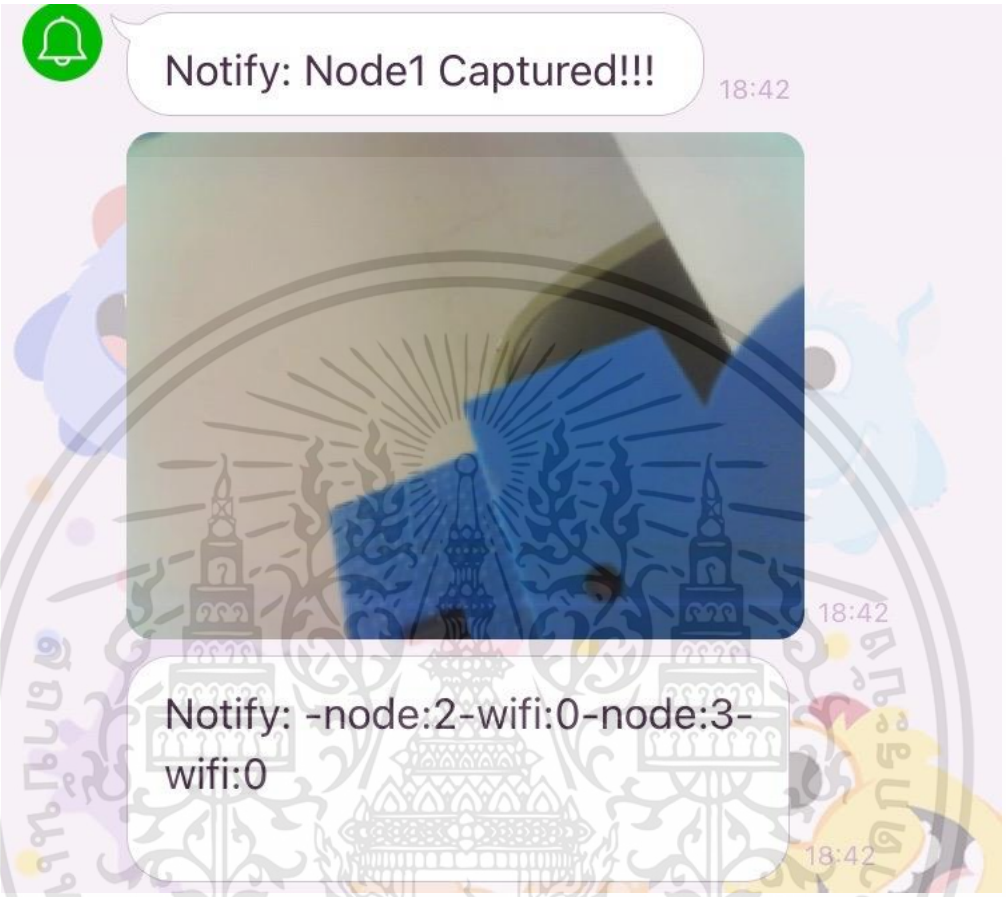
หลังจากข้อความแจ้งเตือนการบันทึกภาพของทั้งสองโนดและภาพที่บันทึกจากแต่ละโนดส่งมา จะส่งมาพร้อมกับอีกข้อความ คือ ข้อความ Notify: -node:2-wifi:1-node:0-wifi:0 ซึ่งหมายถึงโนด 1 มีการเชื่อมต่อกับโนด 2 โดยข้อความ wifi:1 ที่อยู่ข้างหลังชื่อโนดที่มีการเชื่อมต่อแสดงให้เห็นว่าโนด 2 มีการเชื่อมต่อ Wi-Fi อยู่ด้วย ทำให้เมื่อโนด 2 เมื่อมีการบันทึกภาพจะสามารถส่งภาพที่บันทึกมาที่ Line Notify ได้ และข้อความ node:0-wifi:0 แสดงถึงโนด 1 ไม่มีการเชื่อมต่อกับโนด 3 และไม่สามารถตรวจสอบได้ว่าโนด 3 เชื่อมต่อ Wi-Fi อยู่หรือไม่

ข้อความ Notify: -node:1-wifi:1-node:0-wifi:0 ซึ่งหมายถึงโนด 2 มีการเชื่อมต่อกับโนด 1 โดยข้อความ wifi:1 ที่อยู่ข้างหลังชื่อโนดที่มีการเชื่อมต่อ แสดงให้เห็นว่าโนด 1 มีการเชื่อมต่อ Wi-Fi อยู่ด้วย ทำให้เมื่อโนด 1 เมื่อมีการบันทึกภาพจะสามารถส่งภาพที่บันทึกมาที่ Line Notify ได้ และข้อความ node:0-wifi:0 แสดงถึงโนด 2 ไม่มีการเชื่อมต่อกับโนด 3 และไม่สามารถตรวจสอบได้ว่าโนด 3 เชื่อมต่อ Wi-Fi อยู่หรือไม่



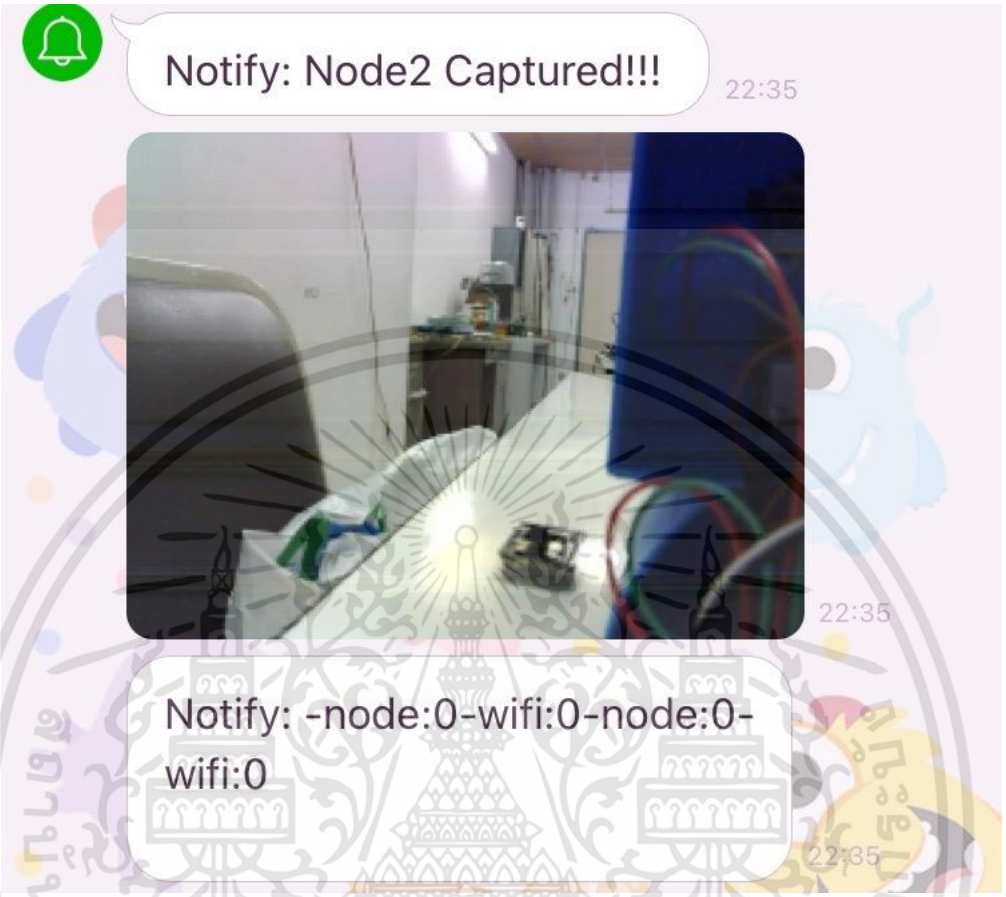
รูปที่ 4.23 ข้อความที่ทุกโนดส่งเข้า Line Notify

จากรูปที่ 4.23 แสดงข้อความและรูปภาพที่โนด 1 โนด2 และโนด 3 ส่งเข้า Line Notify โดยมีข้อความ Notify: -node:2-wifi:1-node:1-wifi:1 , Notify: -node:1-wifi:1-node:3-wifi:1 และ Notify: -node:2-wifi:1-node:3-wifi:1 แสดงให้เห็นว่าโนด 3 มีการเชื่อต่อกับโนด 2 และโนด 1 , โนด 2 มีการเชื่อต่อกับโนด 1 และโนด 3 และโนด 1 การเชื่อต่อกับโนด 2 และโนด 3 ตามลำดับ



รูปที่ 4.24 ข้อความที่โนด 1 ส่งเข้า Line Notify

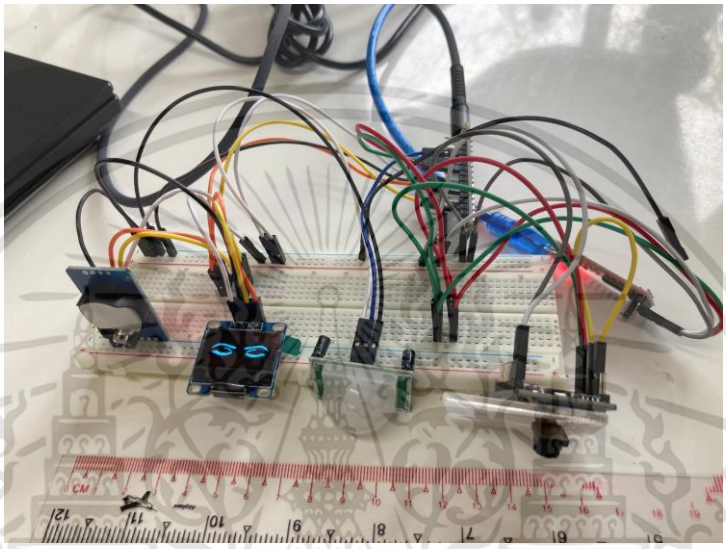
จากรูปที่ 4.24 แสดงข้อความและรูปภาพที่โนด 1 ส่งเข้า Line Notify โดยมีข้อความ Notify: -node:2-wifi:0-node:3-wifi:0 แสดงให้เห็นว่าโนด 1 มีการเชื่อมต่อผ่านโครงข่ายแบบ Mesh กับโนด 2 และโนด 3 และข้อความ wifi:0 แสดงถึงโนด 2 และโนด 3 ไม่มีการเชื่อมต่อ Wi-Fi ทำให้เมื่อโนด 2 และโนด 3 มีการบันทึกภาพจะไม่สามารถส่งภาพเข้ามายัง Line Notify ได้



รูปที่ 4.25 ข้อความที่โนด 2 ส่งเข้า Line Notify

จากรูปที่ 4.25 แสดงข้อความและรูปภาพที่โนด 2 ส่งเข้า Line Notify โดยมีข้อความ Notify: -node:0-wifi:0-node:0-wifi:0 แสดงให้เห็นว่าโนด 2 ไม่มีการเชื่อมต่อผ่านโครงข่ายแบบ Mesh กับโนดใดเลย

นอกจากนี้การเมื่อมีการตรวจจับการเคลื่อนไหวได้ จอแสดงผล OLED ที่สร้างภาพไว้บนจอจะมีการเปลี่ยนแปลงรูปภาพให้คล้ายกับภาพดวงตากระพริบ แสดงได้ดังรูปที่ 4.26 และเมื่ออุปกรณ์ทั้ง 3 โนด สามารถทำงานร่วมกันได้จึงนำชุดอุปกรณ์ในแต่ละโนดมาประกอบลงในกล่องอุปกรณ์ เพื่อให้มีความน่าใช้งาน แสดงได้ดังรูปที่ 4.27



รูปที่ 4.26 ภาพที่แสดงบนจอ OLED เมื่อมีการตรวจจับการเคลื่อนไหวได้

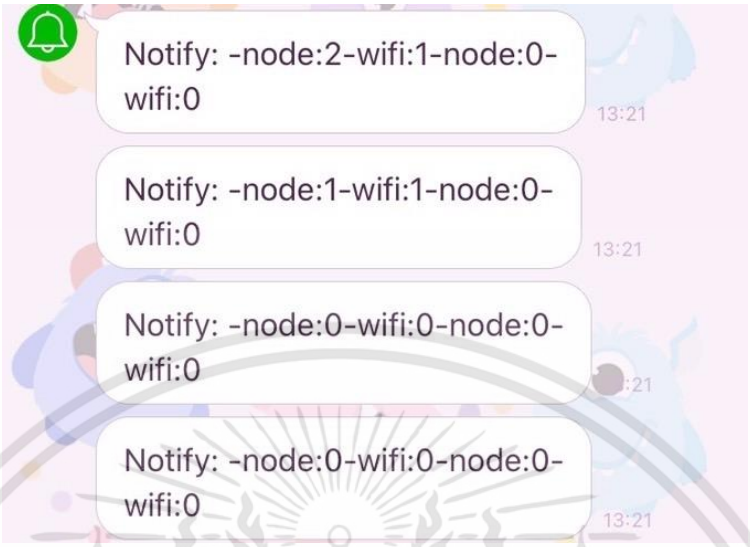
รูปที่ 4.27 ชุดอุปกรณ์หลังจากนำมาประกอบลงในกล่องใส่อุปกรณ์

4.11 ผลทดสอบระยะทางสูงสุดที่อุปกรณ์สามารถเชื่อมต่อกันได้ผ่านโครงข่ายแบบ Mesh และระยะทางสูงสุดที่อุปกรณ์แต่ละตัวสามารถเชื่อมต่อ Wi-Fi ได้

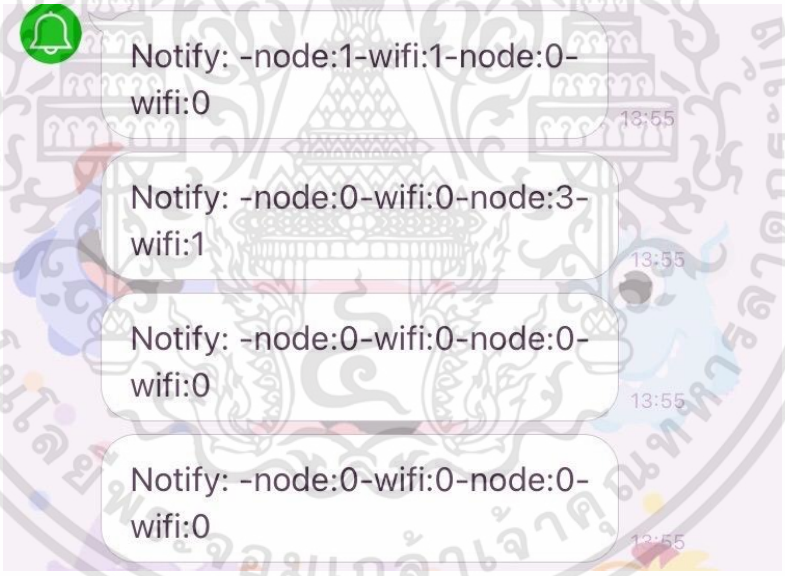
ในการใช้งานจริงของกล้องวงจรปิดทั้ง 3 ตัว เป็นการใช้งานผ่านโครงข่ายแบบ Mesh และใช้ Wi-Fi ในการส่งแจ้งเตือนไปยังผู้ใช้งานร่วมด้วย ดังนั้นในการทดสอบจึงจำเป็นต้องทดสอบระยะทางสูงสุดที่อุปกรณ์หรือกล้องวงจรปิดแต่ละตัวสามารถเชื่อมต่อกันได้ผ่านโครงข่ายแบบ Mesh และระยะทางสูงสุดที่อุปกรณ์หรือกล้องวงจรปิดแต่ละตัวเชื่อมต่อ Wi-Fi ได้ เพื่อนำผลการทดสอบนี้ไปออกแบบและวางแผนการจัดวางตำแหน่งของกล้องวงจรปิดแต่ละตัวว่าควรจัดวางระยะห่างระหว่างกล้องวงจรปิดแต่ละตัวเท่าไร และห่างจากจุดกระจายสัญญาณ Wi-Fi หรือเราต์เตอร์เท่าไร จึงจะทำให้กล้องวงจรปิดแต่ละตัวสามารถเชื่อมต่อกันผ่านโครงข่ายแบบ Mesh ได้ครบทุกตัว และแต่ละตัวสามารถส่งแจ้งเตือนไปยังผู้ใช้งานผ่าน Line Notify โดยใช้ Wi-Fi ได้ จึงได้แบ่งหัวข้อผลการทดสอบออกเป็น 2 ส่วน ดังนี้

4.11.1 ผลการทดสอบระยะทางสูงสุดที่อุปกรณ์แต่ละตัวสามารถเชื่อมต่อกันได้ผ่านโครงข่ายแบบ Mesh

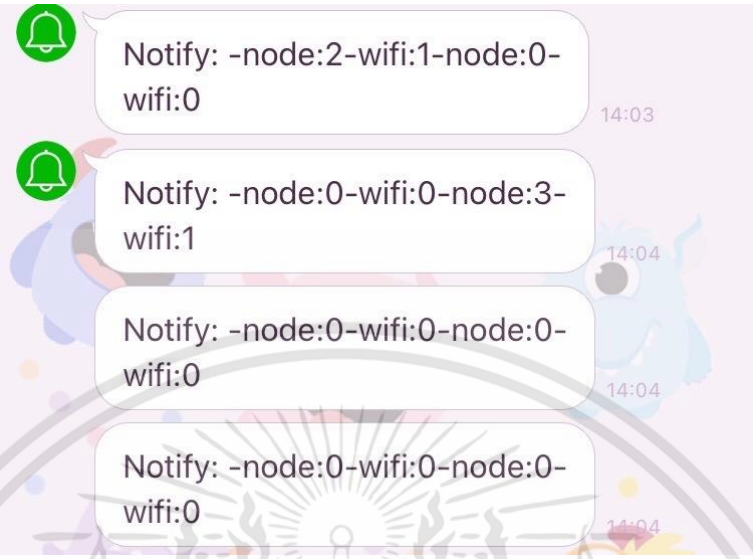
ในการทดสอบเพื่อหาระยะทางสูงสุดที่อุปกรณ์แต่ละตัวสามารถเชื่อมต่อกันผ่านโครงข่ายแบบ Mesh ได้ ทำการทดสอบโดยทดสอบการเชื่อมต่อกันผ่านโครงข่ายแบบ Mesh ระหว่างอุปกรณ์ครั้งละ 2 อุปกรณ์ โดยอุปกรณ์ตัวแรกจะตั้งไว้ที่ตำแหน่งเดิม และทำการเลื่อนตำแหน่งอุปกรณ์อีกตัวให้ห่างจากอุปกรณ์ตัวแรกเพิ่มขึ้น ซึ่งได้กระจายสัญญาณ Wi-Fi ให้อุปกรณ์กรแต่ละเชื่อมต่อตลอดเวลา เพื่อตรวจสอบการเชื่อมต่อกันผ่านโครงข่ายแบบ Mesh ขณะห่างอุปกรณ์แต่ละตัว โดยสามารถตรวจสอบการเชื่อมต่ออุปกรณ์แต่ละตัวผ่านโครงข่ายแบบ Mesh ได้จากข้อความที่ส่งเข้ามาใน Line Notify แสดงได้ดังในรูปที่ 4.28 ถึง รูปที่ 4.30



รูปที่ 4.28 ข้อความที่ส่งเข้าใน Line Notify เพื่อตรวจสอบการเชื่อมต่อผ่านโครงข่ายแบบ Mesh ระหว่างอุปกรณ์ตัวที่ 1 และอุปกรณ์ตัวที่ 2



รูปที่ 4.29 ข้อความที่ส่งเข้าใน Line Notify เพื่อตรวจสอบการเชื่อมต่อผ่านโครงข่ายแบบ Mesh ระหว่างอุปกรณ์ตัวที่ 1 และอุปกรณ์ตัวที่ 3



รูปที่ 4.30 ข้อความที่ส่งเข้าใน Line Notify เพื่อตรวจสอบการเชื่อมต่อผ่านโครงข่ายแบบ Mesh ระหว่างอุปกรณ์ตัวที่ 2 และอุปกรณ์ตัวที่ 3

จากรูปที่ 4.28 ถึง รูปที่ 4.30 แสดงข้อความที่อุปกรณ์แต่ละตัวส่งเข้ามาใน Line Notify โดยจากรายละเอียดที่ได้อธิบายจากรูปที่ 4.21 ถึงรูปที่ 4.25 สามารถนำมาประกอบใช้อธิบายรูปที่ 4.28 ถึง รูปที่ 4.30 ได้ โดยถ้าหากอุปกรณ์ที่เชื่อมต่อกันผ่านโครงข่ายแบบ Mesh ยังอยู่ในระยะที่สามารถเชื่อมต่อกันได้จะมีการส่งข้อความเข้ามาใน Line Notify เพื่อระบุว่ามีอุปกรณ์ที่เชื่อมต่ออยู่ รวมทั้งสามารถตรวจสอบสถานการณ์เชื่อมต่อสัญญาณ Wi-Fi ของอุปกรณ์ที่เชื่อมต่ออยู่ด้วยได้ และถ้าหากอุปกรณ์ขาดการเชื่อมต่อกันผ่านโครงข่ายแบบ Mesh จะมีข้อความส่งมาใน Line Notify เพื่อระบุว่าไม่มีอุปกรณ์ตัวอื่นที่เชื่อมต่อกับอุปกรณ์ตัวนั้นผ่านโครงข่ายแบบ Mesh และไม่สามารถตรวจสอบสถานการณ์เชื่อมต่อสัญญาณ Wi-Fi ของอุปกรณ์ตัวอื่นได้ โดยระยะห่างระหว่างอุปกรณ์แต่ละตัวที่เชื่อมต่อกันผ่านโครงข่ายแบบ Mesh มีระยะห่างที่แตกต่างกัน แสดงได้ดังในตารางที่ 4.1 ถึงตารางที่ 4.3

ตารางที่ 4.1 ตำแหน่งและระยะห่างสูงสุดระหว่างอุปกรณ์ตัวที่ 1 และอุปกรณ์ตัวที่ 2 ที่เชื่อมต่อกันผ่านโครงข่ายแบบ Mesh

	ตำแหน่งละติจูด	ตำแหน่งลองจิจูด	ระยะห่าง
อุปกรณ์ตัวที่ 1	13°43’37.7”N	100°46’38.7”E	450 เมตร
อุปกรณ์ตัวที่ 2	13°43’38.3”N	100°46’25.3”E	

จากตารางที่ 4.1 แสดงตำแหน่งที่ตั้งของอุปกรณ์ตัวที่ 1 และอุปกรณ์ตัวที่ 2 โดยระบุตำแหน่งในรูปแบบละติจูดและลองจิจูด ซึ่งสามารถวัดระยะห่างระหว่างอุปกรณ์ตัวที่ 1 และอุปกรณ์ตัวที่ 2 ได้ประมาณ 450 เมตร ซึ่งเป็นระยะทางสูงสุดที่อุปกรณ์ตัวที่ 1 และอุปกรณ์ตัวที่ 2 สามารถเชื่อมต่อกันผ่านโครงข่ายแบบ Mesh ได้ และเมื่อทำการเลื่อนตำแหน่งของอุปกรณ์ตัวที่ 2 ให้มีระยะห่างจากอุปกรณ์ตัวที่ 1 มากกว่า 450 เมตร พบว่าอุปกรณ์ตัวที่ 1 และอุปกรณ์ตัวที่ 2 จะขาดการเชื่อมต่อจากกันผ่านโครงข่ายแบบ Mesh

ตารางที่ 4.2 ตำแหน่งและระยะห่างสูงสุดระหว่างอุปกรณ์ตัวที่ 1 และอุปกรณ์ตัวที่ 3 ที่เชื่อมต่อกันผ่านโครงข่ายแบบ Mesh

	ตำแหน่งละติจูด	ตำแหน่งลองจิจูด	ระยะห่าง
อุปกรณ์ตัวที่ 1	13°43’37.7”N	100°46’38.7”E	400 เมตร
อุปกรณ์ตัวที่ 3	13°43’37.6”N	100°46’24.7”E	

จากตารางที่ 4.2 แสดงตำแหน่งที่ตั้งของอุปกรณ์ตัวที่ 1 และอุปกรณ์ตัวที่ 3 โดยระบุตำแหน่งในรูปแบบละติจูดและลองจิจูด ซึ่งสามารถวัดระยะห่างระหว่างอุปกรณ์ตัวที่ 1 และอุปกรณ์ตัวที่ 3 ได้ประมาณ 400 เมตร ซึ่งเป็นระยะทางสูงสุดที่อุปกรณ์ตัวที่ 1 และอุปกรณ์ตัวที่ 3 สามารถเชื่อมต่อกันผ่านโครงข่ายแบบ Mesh ได้ และเมื่อทำการเลื่อนตำแหน่งของอุปกรณ์ตัวที่ 3 ให้มีระยะห่างจากอุปกรณ์ตัวที่ 1 มากกว่า 400 เมตร พบว่าอุปกรณ์ตัวที่ 1 และอุปกรณ์ตัวที่ 3 จะขาดการเชื่อมต่อจากกันผ่านโครงข่ายแบบ Mesh

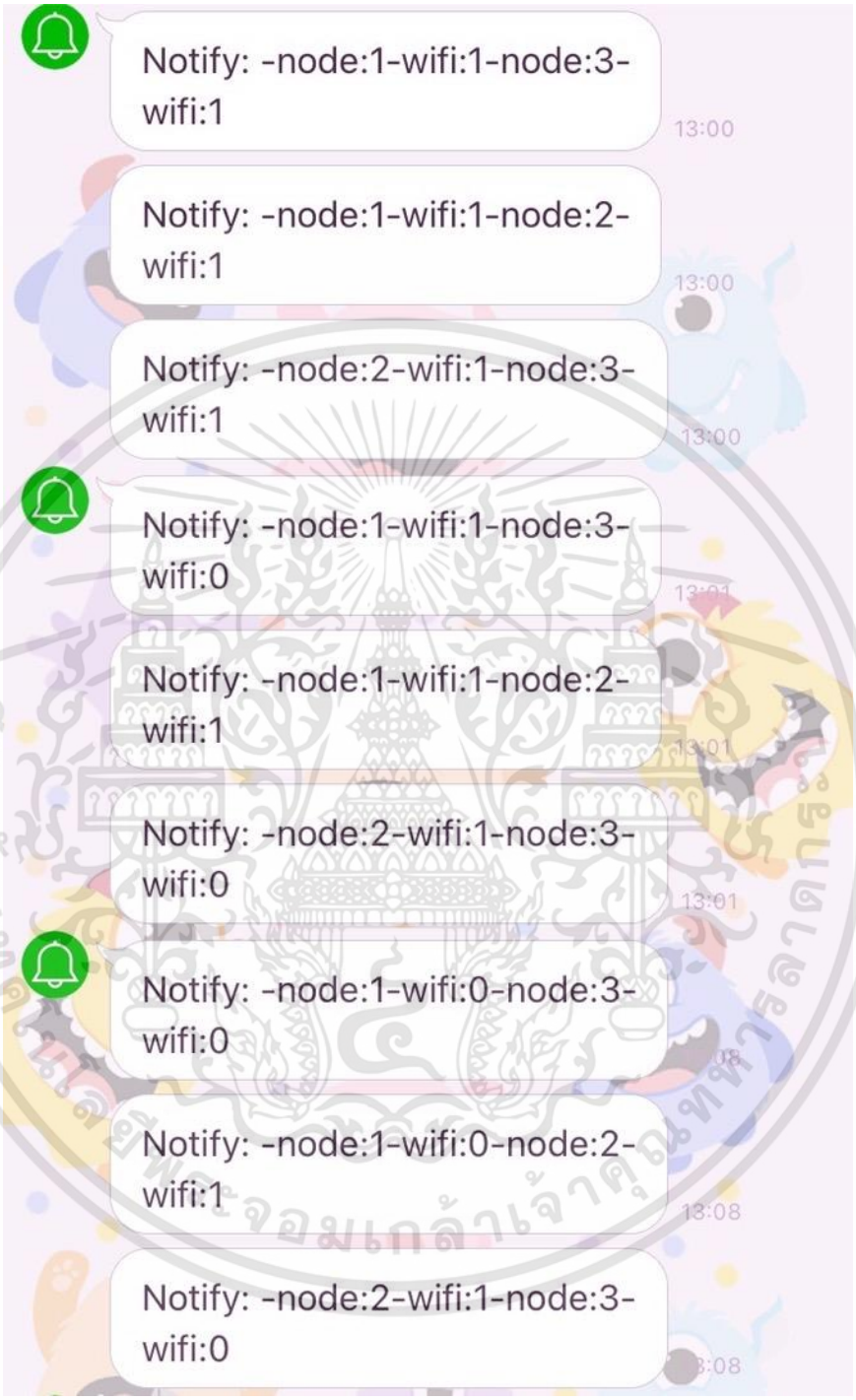
ตารางที่ 4.3 ตำแหน่งและระยะห่างสูงสุดระหว่างอุปกรณ์ตัวที่ 2 และอุปกรณ์ตัวที่ 3 ที่เชื่อมต่อกันผ่านโครงข่ายแบบ Mesh

	ตำแหน่งละติจูด	ตำแหน่งลองจิจูด	ระยะห่าง
อุปกรณ์ตัวที่ 2	13°43’37.7”N	100°46’38.7”E	400 เมตร
อุปกรณ์ตัวที่ 3	13°43’37.7”N	100°46’23.9”E	

จากตารางที่ 4.3 แสดงตำแหน่งที่ตั้งของอุปกรณ์ตัวที่ 2 และอุปกรณ์ตัวที่ 3 โดยระบุตำแหน่งในรูปแบบละติจูดและลองจิจูด ซึ่งสามารถวัดระยะห่างระหว่างอุปกรณ์ตัวที่ 2 และอุปกรณ์ตัวที่ 3 ได้ประมาณ 400 เมตร ซึ่งเป็นระยะทางสูงสุดที่อุปกรณ์ตัวที่ 2 และอุปกรณ์ตัวที่ 3 สามารถเชื่อมต่อกันผ่านโครงข่ายแบบ Mesh ได้ และเมื่อทำการเลื่อนตำแหน่งของอุปกรณ์ตัวที่ 3 ให้มีระยะห่างจากอุปกรณ์ตัวที่ 2 มากกว่า 400 เมตร พบว่าอุปกรณ์ตัวที่ 2 และอุปกรณ์ตัวที่ 3 จะขาดการเชื่อมต่อจากกันผ่านโครงข่ายแบบ Mesh

4.11.2 ผลการทดสอบระยะทางสูงสุดที่อุปกรณ์แต่ละตัวสามารถเชื่อมต่อกันได้ผ่าน Wi-Fi

ในการทดสอบเพื่อหาระยะทางสูงสุดที่อุปกรณ์แต่ละตัวสามารถเชื่อมต่อ Wi-Fi ได้ทำการทดสอบโดยนำอุปกรณ์ทั้งสามวางไว้ที่ตำแหน่งเดียวกันและเชื่อมต่อกันผ่านโครงข่ายแบบ Mesh ทั้งสามตัว และมีอุปกรณ์กระจายสัญญาณ Wi-Fi 1 จุด สำหรับกระจายสัญญาณ Wi-Fi ให้อุปกรณ์ทั้งสาม โดยทำการเลื่อนตำแหน่งของอุปกรณ์กระจายสัญญาณ Wi-Fi ห่างจากอุปกรณ์ทั้งสามออกไปเรื่อย ๆ เพื่อตรวจสอบการเชื่อมต่อ Wi-Fi ของอุปกรณ์แต่ละตัว โดยสามารถตรวจสอบการเชื่อมต่ออุปกรณ์แต่ละตัวผ่านข้อความที่ส่งเข้ามาใน Line Notify แสดงได้ดังในรูปที่ 4.31



รูปที่ 4.31 ข้อความที่อุปกรณ์แต่ละตัวส่งเข้ามาใน Line Notify เพื่อตรวจสอบสถานการณ์เชื่อมต่อ Wi-Fi ของอุปกรณ์แต่ละตัว

จากรูปที่ 4.31 แสดงข้อความที่อุปกรณ์แต่ละตัวส่งเข้ามาใน Line Notify โดยจากรายละเอียดที่ได้อธิบายจากรูปที่ 4.21 ถึงรูปที่ 4.25 สามารถนำมาประกอบใช้อธิบายรูปที่ 4.31 ได้ โดยจากรูปที่ 4.31 พบว่าอุปกรณ์ตัวที่ 3 หรือที่กำหนดให้เป็นโนดที่ 3 ขาดการเชื่อมต่อสัญญาณ Wi-Fi ก่อนอุปกรณ์อีกสองตัว และอุปกรณ์ตัวที่ 1 จะขาดการเชื่อมต่อจากสัญญาณ Wi-Fi เป็นลำดับที่สอง และอุปกรณ์ที่ขาดการเชื่อมต่อจากสัญญาณ Wi-Fi ลำดับสุดท้าย คืออุปกรณ์ตัวที่ 2 ซึ่งแสดงว่าระยะห่างระหว่างอุปกรณ์แต่ละตัวกับจุดกระจายสัญญาณ Wi-Fi จะมีระยะห่างไม่เท่ากัน แสดงได้ดังในตารางที่ 4.4 ถึงตารางที่ 4.6

ตารางที่ 4.4 ตำแหน่งและระยะห่างสูงสุดระหว่างอุปกรณ์ตัวที่ 1 และจุดกระจายสัญญาณ Wi-Fi

	ตำแหน่งละติจูด	ตำแหน่งลองจิจูด	ระยะห่าง
อุปกรณ์ตัวที่ 1	13°43’37.7”N	100°46’38.7”E	70 เมตร
จุดกระจายสัญญาณ Wi-Fi	13°43’36.1”N	100°46’36.4”E	

จากตารางที่ 4.4 แสดงตำแหน่งที่ตั้งของอุปกรณ์ตัวที่ 1 และจุดกระจายสัญญาณ Wi-Fi โดยระบุตำแหน่งในรูปแบบละติจูดและลองจิจูด ซึ่งสามารถวัดระยะห่างระหว่างอุปกรณ์ตัวที่ 1 และจุดกระจายสัญญาณ Wi-Fi ได้ประมาณ 70 เมตร ซึ่งเป็นระยะทางสูงสุดที่อุปกรณ์ตัวที่ 1 สามารถเชื่อมต่อสัญญาณ Wi-Fi ได้ และเมื่อทำการเลื่อนตำแหน่งของจุดกระจายสัญญาณ Wi-Fi ให้มีระยะห่างจากอุปกรณ์ตัวที่ 1 มากกว่า 70 เมตร พบว่าอุปกรณ์ตัวที่ 1 จะขาดการเชื่อมต่อจากสัญญาณ Wi-Fi

ตารางที่ 4.5 ตำแหน่งและระยะห่างสูงสุดระหว่างอุปกรณ์ตัวที่ 2 และจุดกระจายสัญญาณ Wi-Fi

	ตำแหน่งละติจูด	ตำแหน่งลองจิจูด	ระยะห่าง
อุปกรณ์ตัวที่ 2	13°43’37.7”N	100°46’38.7”E	90 เมตร
จุดกระจายสัญญาณ Wi-Fi	13°43’35.4”N	100°46’35.0”E	

จากตารางที่ 4.5 แสดงตำแหน่งที่ตั้งของอุปกรณ์ตัวที่ 2 และจุดกระจายสัญญาณ Wi-Fi โดยระบุตำแหน่งในรูปแบบละติจูดและลองจิจูด ซึ่งสามารถวัดระยะห่างระหว่างอุปกรณ์ตัวที่

2 และจุดกระจายสัญญาณ Wi-Fi ได้ประมาณ 90 เมตร ซึ่งเป็นระยะทางสูงสุดที่อุปกรณ์ตัวที่ 2 สามารถเชื่อมต่อสัญญาณ Wi-Fi ได้ และเมื่อทำการเลื่อนตำแหน่งของจุดกระจายสัญญาณ Wi-Fi ให้มีระยะห่างจากอุปกรณ์ตัวที่ 2 มากกว่า 90 เมตร พบว่าอุปกรณ์ตัวที่ 2 จะขาดการเชื่อมต่อจากสัญญาณ Wi-Fi

ตารางที่ 4.6 ตำแหน่งและระยะห่างสูงสุดระหว่างอุปกรณ์ตัวที่ 3 และจุดกระจายสัญญาณ Wi-Fi

	ตำแหน่งละติจูด	ตำแหน่งลองจิจูด	ระยะห่าง
อุปกรณ์ตัวที่ 3	13°43’37.7”N	100°46’38.7”E	60 เมตร
จุดกระจายสัญญาณ Wi-Fi	13°43’36.7”N	100°46’36.7”E	

จากตารางที่ 4.6 แสดงตำแหน่งที่ตั้งของอุปกรณ์ตัวที่ 3 และจุดกระจายสัญญาณ Wi-Fi โดยระบุตำแหน่งในรูปแบบละติจูดและลองจิจูด ซึ่งสามารถวัดระยะห่างระหว่างอุปกรณ์ตัวที่ 3 และจุดกระจายสัญญาณ Wi-Fi ได้ประมาณ 60 เมตร ซึ่งเป็นระยะทางสูงสุดที่อุปกรณ์ตัวที่ 3 สามารถเชื่อมต่อสัญญาณ Wi-Fi ได้ และเมื่อทำการเลื่อนตำแหน่งของจุดกระจายสัญญาณ Wi-Fi ให้มีระยะห่างจากอุปกรณ์ตัวที่ 3 มากกว่า 60 เมตร พบว่าอุปกรณ์ตัวที่ 3 จะขาดการเชื่อมต่อจากสัญญาณ Wi-Fi

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

ปริญญานิพนธ์ฉบับนี้เสนอการออกแบบกล้องวงจรปิดที่มีการสตรีมวิดีโอในเว็บเซิร์ฟเวอร์ และเชื่อมต่อระหว่างโนดผ่านโครงข่ายแบบ Mesh ซึ่งในแต่ละโนดประกอบด้วยอุปกรณ์ ESP32 CAM , ESP32 , FT232RL , จอแสดงผล OLED , เซนเซอร์ตรวจจับคลื่นรังสีอินฟราเรด และเซนเซอร์ Real Time Clock โดยจะมีการตรวจจับการเคลื่อนไหวจากเซนเซอร์ตรวจจับคลื่นรังสีอินฟราเรด และเมื่อมีการตรวจจับการเคลื่อนไหวได้ ESP32 CAM จะทำการบันทึกภาพนิ่งลงบน SD card โดยชื่อไฟล์ที่บันทึกตั้งชื่อตามวัน และเวลา ซึ่งเป็นค่าที่เซนเซอร์ Real Time Clock ส่งผ่าน ESP32 มายัง ESP32 CAM และส่งภาพที่บันทึกได้ไปยัง Line Notify เพื่อแจ้งเตือนผู้ใช้งาน พร้อมข้อความระบุว่าเป็นภาพที่บันทึกจากโนดหรือกล้องตัวใด รวมทั้งเมื่อมีการตรวจจับการเคลื่อนไหวได้ภาพบนจอแสดงผล OLED จะมีการเปลี่ยนแปลงหรือเคลื่อนไหวเพื่อให้ดูน่าใช้งาน นอกจากนี้ในแต่ละโนดที่มีการเชื่อมต่อกันแบบ Mesh จะมีการส่งข้อความแจ้งเตือนไปยัง Line Notify เพื่อแจ้งให้ทราบว่าขณะนั้นมีโนดใดที่เชื่อมต่ออยู่หรือขาดการเชื่อมต่อจากโครงข่าย

จากผลการทดสอบหัวข้อที่ 4.10.1 และหัวข้อที่ 4.10.2 สามารถนำข้อมูลระยะห่างสูงสุดระหว่างอุปกรณ์แต่ละตัวที่สามารถเชื่อมต่อกันได้ผ่านโครงข่ายแบบ Mesh และข้อมูลระยะห่างสูงสุดที่อุปกรณ์แต่ละตัวสามารถเชื่อมต่อสัญญาณ Wi-Fi ได้ มาออกแบบหรือกำหนดตำแหน่งของอุปกรณ์แต่ละตัวเมื่อนำมาใช้งานเป็นกล้องวงจรปิด เพื่อกำหนดระยะห่างระหว่างกล้องวงจรปิดแต่ละตัวให้สามารถเชื่อมต่อกันได้ผ่านโครงข่ายแบบ Mesh และกำหนดระยะห่างระหว่างกล้องวงจรปิดแต่ละตัวกับจุดกระจายสัญญาณ Wi-Fi หรือเราต์เตอร์ได้

โดยจากตารางที่ 4.1 ถึงตารางที่ 4.3 สามารถสรุปได้ว่า ควรจัดวางกล้องวงจรปิดทั้งสามให้มีระยะห่างกันไม่ควรเกิน 400 เมตร และจากตารางที่ 4.4 ถึงตารางที่ 4.5 ระยะห่างระหว่างกล้องแต่ละตัวกับจุดกระจายสัญญาณ Wi-Fi หรือเราต์เตอร์ไม่ควรเกิน 60 เมตร โดยค่าระยะที่นำมากำหนดเพื่อวางแผนการจัดวางตำแหน่งของกล้องวงจรปิดและจุดกระจายสัญญาณ Wi-Fi นี้เป็นค่าที่น้อยที่สุดจากตารางที่ 4.1 ถึงตารางที่ 4.6 เนื่องจากหากใช้เป็นค่าที่มากที่สุดหรือค่าเฉลี่ย

จะทำให้อุปกรณ์บางตัวไม่สามารถเชื่อมต่อกันผ่านโครงข่ายแบบ Mesh หรือเชื่อมต่อกับจุดกระจายสัญญาณ Wi-Fi ได้

5.2 ข้อเสนอแนะ

เนื่องจากบอร์ด ESP32 CAM มีข้อจำกัดด้านความจุทำให้ไม่สามารถทำงานได้เต็มที่เมื่อรวมโปรแกรมคำสั่งหลาย ๆ คำสั่งเข้าด้วยกัน รวมทั้งขาของบอร์ด ESP32 CAM ได้ใช้ในการเชื่อมต่อกับตัวกล้องและ SD card ทำให้มีขาไม่เพียงพอในการเชื่อมต่อกับเซนเซอร์หรือจอแสดงผล OLED ที่ต้องการนำมาใช้ จึงได้เพิ่มบอร์ด ESP32 เข้ามาเชื่อมต่อเพื่อเป็นการเพิ่มขาในการเชื่อมต่อเซนเซอร์และจอแสดงผล OLED และเพิ่มความจุในการเพิ่มโปรแกรมคำสั่ง รวมทั้งหากมีการนำไปพัฒนาปรับปรุงเพิ่มเติม ผู้ที่นำไปปรับปรุงควรศึกษาข้อมูลของอุปกรณ์ที่นำมาใช้เพื่อให้เกิดความเหมาะสมและเพื่อให้อุปกรณ์มีความเสถียรในการใช้งานมากยิ่งขึ้น

ระยะห่างระหว่างอุปกรณ์และจุดกระจายสัญญาณ Wi-Fi อาจจะเกิดความคลาดเคลื่อนหรือสามารถเปลี่ยนแปลงได้ เนื่องจากความแรงของสัญญาณและสิ่งกีดขวางสัญญาณต่าง ๆ ดังนั้นในการกำหนดตำแหน่งการจัดวางอุปกรณ์ในการใช้งานจริงควรจัดวางที่ระยะห่างที่ในช่วงระยะห่างที่วัดได้ หรือน้อยกว่า เพื่อป้องกันการขาดการเชื่อมต่อระหว่างอุปกรณ์ผ่านโครงข่ายแบบ Mesh และสัญญาณ Wi-Fi

บรรณานุกรม

[1] P.CCTV NETWORK ENGINEERING. “โครงสร้างเครือข่ายคอมพิวเตอร์ (Network Topology).” https://www.pi-tech.biz/17234173/-network-topology.

[2] Gimme. “Mesh WiFi คืออะไร | ขยายสัญญาณไวไฟให้ครอบคลุมทั่วบ้าน แก้ปัญหาแพ็กเกจแรง แต่เน็ตไม่วิ่ง.” https://droidsans.com/whatis-mesh-wifi/.

[3] Fitrox Electronics. “การใช้งาน HC-SR501 Pyroelectric IR Motion Sensor.” http://fitrox.lnwshop.com/article/39/tutorial-%E0%B8%81%E0%B8%B2%E0%B8%A3%E0%B9%83%E0%B8%8A%E0%B9%89%E0%B8%87%E0%B8%B2%E0%B8%99-hc-sr501-pyroelectric-ir-motion-sensor.

[4] SUPPORT THAIEASYELEC. “PIR Motion Sensor Getting Started.” https://blog.thaieasyelec.com/getting-started-pir-motion-sensor/.

[5] Choonewza. “CAT LoRa Starter Kit ตอนที่ 7 เพิ่ม Real Time Clock Module.” https://choonewza.medium.com/cat-lora-starter-kit-%E0%B8%95%E0%B8%AD%E0%B8%99%E0%B8%97%E0%B8%B5%E0%B9%88-7-%E0%B9%80%E0%B8%9E%E0%B8%B4%E0%B9%88%E0%B8%A1-real-time-clock-module-6dfc110ae994.

[6] CyberTice. “โมดูลนาฬิกา DS3231 module กับ Arduino.” https://www.cybertice.com/article/40/%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%B9%82%E0%B8%A1%E0%B8%94%E0%B8%B9%E0%B8%A5%E0%B8%99%E0%B8%B2%E0%B8%AC%E0%B8%B4%E0%B8%81%E0%B8%B2-ds3231-module-%E0%B8%81%E0%B8%B1%E0%B8%9A-arduino.

[7] suwitkiravittaya. “การแสดงผลด้วยจอแอลอีดีขนาด 128×64 พิกเซล.” http://suwitkiravittaya.eng.chula.ac.th/B2i2019BookWeb/oleddisplay.html.

[8] Banggood. “I2C OLED แสดงผล โมดูล.” https://www.banggood.com/th/5Pcs-0_96-Inch-Blue-Yellow-IIC-I2C-OLED-Display-Module-p-971294.html.

[9] suwitkiravittaya. “การสื่อสารผ่านพอร์ตอนุกรม.” http://suwitkiravittaya.eng.chula.ac.th/B2i2019BookWeb/pdf/serial.pdf.

[10] SUPPORT THAIEASYELEC. “บทความ ESPino32 ตอนที่ 8 การสื่อสารอนุกรมแบบ I2C.” https://blog.thaieasyelec.com/espino32-ch8-how-to-use-i2c/.

[11] CyberTice. “ESP-32S NodeMCU ESP-WROOM-32S Wi-Fi and Bluetooth Module Dual Core Consumption CP2102.” https://www.cybertice.com/product/4375/esp-32s-nodemcu-esp-wroom-32s-wi-fi-and-bluetooth-module-dual-core-consumption-cp2102.

[12] saixiii. “HTTP คืออะไร และ HTTPS คืออะไร และต่างกันอย่างไร.” https://saixiii.com/http-https/.

[13] zixzax. “HTTP คืออะไร.” https://zhort.link/qUi.

[14] IT Bolt. “TCP/IP หมายถึงอะไร.” https://itbolt.blogspot.com/2017/08/tcpip.html.

[15] Best IDC. “IP Address (Internet Protocol Address) คืออะไร.” https://www.bestinternet.co.th/single_blog.php?id=55&IP%20Address%20(Internet%20Protocol%20Address)%20%E0%B8%84%E0%B8%B7%E0%B8%AD%E0%B8%AD%E0%B8%B0%E0%B9%84%E0%B8%A3.

[16] RobotSiam. “การใช้งาน ESP32-CAM กล้อง OV2640 เป็น Video Stream ภาพเคลื่อนไหว.” https://www.robotsiam.com/article/72/%E0%B8%81%E0%B8%B2%E0%B8%A3%E0%B9%83%E0%B8%8A%E0%B9%89%E0%B8%87%E0%B8%B2%E0%B8%99-esp32-cam-%E0%B8%81%E0%B8%A5%E0%B9%89%E0%B8%AD%E0%B8%87-ov2640-%E0%B9%80%E0%B8%9B%E0%B9%87%E0%B8%99-video-stream-%E0%B8%A0%E0%B8%B2%E0%B8%9E%E0%B9%80%E0%B8%84%E0%B8%A5%E0%B8%B7%E0%B9%88%E0%B8%AD%E0%B8%99%E0%B9%84%E0%B8%AB%E0%B8%A7.

[17] Sara Santos. “ESP32-CAM AI-Thinker Pinout Guide: GPIOs Usage Explained.” https://randomnerdtutorials.com/esp32-cam-ai-thinker-pinout/.

[18] Sara Santos. “ESP32-CAM AI-Thinker Pinout Guide: GPIOs Usage Explained.” https://randomnerdtutorials.com/esp32-cam-ai-thinker-pinout/.

[19] SUPPORT THAIEASYELEC. “PIR Motion Sensor Getting Started.” https://blog.thaieasyelec.com/getting-started-pir-motion-sensor/.

[20] CyberTice. “PIR เซ็นเซอร์ตรวจจับความเคลื่อนไหว Motion Sensor Detector Module HC-SR501.” https://www.cybertice.com/product/28/pir-%E0%B9%80%E0%B8%8B%E0%B9%87%E0%B8%99%E0%B9%80%E0%B8%8B%E0%B8%AD%E0%B8%A3%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%E0%B8%A1%E0%B9%80%E0%B8%84%E0%B8%A5%E0%B8%B7%E0%B9%88%E0%B8%AD%E0%B8%99%E0%B9%84%E0%B8%AB%E0%B8%A7-motion-sensor-detector-module-hc-sr501-2.

[21] arduino2robot. “real time clock, โมดูลนาฬิกา DS3231 module, ความแม่นยำสูง RTC DS3231 AT24C32 IIC Module, (ไม่มีถ่าน).” https://www.arduino2robot.com/product/138/real-time-clock-%E0%B9%82%E0%B8%A1%E0%B8%94%E0%B8%B9%E0%B8%A5%E0%B8%99%E0%B8%B2%E0%B8%AC%E0%B8%B4%E0%B8%81%E0%B8%B2-ds3231-module-%E0%B8%84%E0%B8%A7%E0%B8%B2%E0%B8%A1%E0%B9%81%E0%B8%A1%E0%B9%88%E0%B8%99%E0%B8%A2%E0%B8%B3%E0%B8%AA%E0%B8%B9%E0%B8%87-rtc-ds3231-at24c32-iic-module-%E0%B9%84%E0%B8%A1%E0%B9%88%E0%B8%A1%E0%B8%B5%E0%B8%96%E0%B9%88%E0%B8%B2%E0%B8%99.

[22] ArduinoNa. “โมดูลนาฬิกา RTC DS3231 และเซนเซอร์วัดอุณหภูมิ พร้อมถ่าน.” https://www.modulemore.com/product/197/%E0%B9%82%E0%B8%A1%E0%B8%94%E0%B8%B9%E0%B8%A5%E0%B8%99%E0%B8%B2%E0%B8%AC%E0%B8%B4%E0%B8%81%E0%B8%B2-rtc-ds3231-%E0%B9%81%E0%B8%A5%E0%B8%B0%E0%B9%80%E0%B8%8B%E0%B8%99%E0%B9%80%E0%B8%8B%E0%B8%AD%E0%B8%A3%E0%B9%8C%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%9E%E0%B8%A3%E0%B9%89%E0%B8%AD%E0%B8%A1%E0%B8%96%E0%B9%88%E0%B8%B2%E0%B8%99.

[23] admin. “TINKERCAD คือโปรแกรมอะไร.” https://ra-software.com/tinkercad-%E0%B8%84%E0%B8%B7%E0%B8%AD%E0%B9%82%E0%B8%9B%E0%B8%A3%E0%B9%81%E0%B8%81%E0%B8%A3%E0%B8%A1%E0%B8%AD%E0%B8%B0%E0%B9%84%E0%B8%A3/.

[24] tinkercad. “tinkercad.” https://www.tinkercad.com/.

[25] บาร์เทอร์อินเตอร์เนชั่นแนล. “LINE Notify คืออะไร.” https://www.lineofficialaccount.com/what_is_line_notify.php.

[26] mindphp. “C++ คืออะไร.” https://www.mindphp.com/%E0%B8%84%E0%B8%B9%E0%B9%88%E0%B8%A1%E0%B8%B7%E0%B8%AD/73-%E0%B8%84%E0%B8%B7%E0%B8%AD%E0%B8%AD%E0%B8%B0%E0%B9%84%E0%B8%A3/2183-c%20%20-%E0%B8%84%E0%B8%B7%E0%B8%AD%E0%B8%AD%E0%B8%B0%E0%B9%84%E0%B8%A3.html.

[27] The Invention. “Arduino IDE.” https://www.ai-corporation.net/2021/11/18/what-is-arduino-ide/.

[28] krupiyadanai. “รู้จักภาษา HTML.” https://krupiyadanai.wordpress.com/%E0%B8%9A%E0%B8%97%E0%B9%80%E0%B8%A3%E0%B8%B5%E0%B8%A2%E0%B8%99-html/%E0%B8%A3%E0%B8%B9%E0%B9%89%E0%B8%88%E0%B8%B1%E0%B8%81%E0%B8%A0%E0%B8%B2%E0%B8%A9%E0%B8%B2-html/.

[29] อาจารย์ ดร.ณัฐพล แสนคำ. “Visual Studio Code.” http://cs.bru.ac.th/%E0%B8%AA%E0%B8%AD%E0%B8%99%E0%B8%A7%E0%B8%B4%E0%B8%98%E0%B8%B5%E0%B8%81%E0%B8%B2%E0%B8%A3%E0%B9%83%E0%B8%8A%E0%B9%89-visual-studio-code-2/.

ภาคผนวก ก

โปรแกรมคำสั่งของ Node 1

1. ชุดคำสั่งในส่วนของ ESP32 CAM

```
#include "esp_camera.h"
#include <WiFi.h>
#include <TridentTD_LineNotify.h>
#include "esp_timer.h"
#include "img_converters.h"
#include "Arduino.h"
#include "fb_gfx.h"
#include "FS.h"
#include "SD_MMC.h"
#include "soc/soc.h" //disable brownout problems
#include "soc/rtc_cntl_reg.h"  //disable brownout problems
#include "driver/rtc_io.h"
//#include "dl_lib.h"
#include "esp_http_server.h"
#include <EEPROM.h>
#define EEPROM_SIZE 1
int pictureNumber = 0;

#define LINE_TOKEN  "**********************" // SET LINE TOKEN
String MESSAGE_LINE = “Node1 Captured!!!”;
//Replace with your network credentials
const char* ssid = "**********"; // แก้ชื่อ wifi
const char* password = "********"; // แก้รหัสผ่าน

#define PART_BOUNDARY "123456789000000000000987654321"

// This project was tested with the AI Thinker Model, M5STACK PSRAM Model and
M5STACK WITHOUT PSRAM
#define CAMERA_MODEL_AI_THINKER
```

```
#if defined(CAMERA_MODEL_WROVER_KIT)
#define PWDN_GPIO_NUM    -1
#define RESET_GPIO_NUM   -1
#define XCLK_GPIO_NUM    21
#define SIOD_GPIO_NUM    26
#define SIOC_GPIO_NUM    27

#define Y9_GPIO_NUM      35
#define Y8_GPIO_NUM      34
#define Y7_GPIO_NUM      39
#define Y6_GPIO_NUM      36
#define Y5_GPIO_NUM      19
#define Y4_GPIO_NUM      18
#define Y3_GPIO_NUM       5
#define Y2_GPIO_NUM       4
#define VSYNC_GPIO_NUM   25
#define HREF_GPIO_NUM    23
#define PCLK_GPIO_NUM    22

#elif defined(CAMERA_MODEL_M5STACK_PSRAM)
#define PWDN_GPIO_NUM     -1
#define RESET_GPIO_NUM    15
#define XCLK_GPIO_NUM     27
#define SIOD_GPIO_NUM     25
#define SIOC_GPIO_NUM     23

#define Y9_GPIO_NUM       19
#define Y8_GPIO_NUM       36
#define Y7_GPIO_NUM       18
#define Y6_GPIO_NUM       39
```

```
#define Y5_GPIO_NUM        5
#define Y4_GPIO_NUM       34
#define Y3_GPIO_NUM       35
#define Y2_GPIO_NUM       32
#define VSYNC_GPIO_NUM    22
#define HREF_GPIO_NUM     26
#define PCLK_GPIO_NUM     21

#elif defined(CAMERA_MODEL_M5STACK_WITHOUT_PSRAM)
#define PWDN_GPIO_NUM     -1
#define RESET_GPIO_NUM    15
#define XCLK_GPIO_NUM     27
#define SIOD_GPIO_NUM     25
#define SIOC_GPIO_NUM     23

#define Y9_GPIO_NUM       19
#define Y8_GPIO_NUM       36
#define Y7_GPIO_NUM       18
#define Y6_GPIO_NUM       39
#define Y5_GPIO_NUM        5
#define Y4_GPIO_NUM       34
#define Y3_GPIO_NUM       35
#define Y2_GPIO_NUM       17
#define VSYNC_GPIO_NUM    22
#define HREF_GPIO_NUM     26
#define PCLK_GPIO_NUM     21

#elif defined(CAMERA_MODEL_AI_THINKER)
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
```

```
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27

#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM       5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22
#else
#error "Camera model not selected"
#endif

static const char* _STREAM_CONTENT_TYPE = "multipart/x-mixed-replace;boundary="
PART_BOUNDARY;
static const char* _STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY "\r\n";
static const char* _STREAM_PART = "Content-Type: image/jpeg\r\nContent-Length:
%u\r\n\r\n";

httpd_handle_t stream_httpd = NULL;

static esp_err_t stream_handler(httpd_req_t *req) {
  camera_fb_t * fb = NULL;
  esp_err_t res = ESP_OK;
  size_t _jpg_buf_len = 0;
  uint8_t * _jpg_buf = NULL;
```

```
char * part_buf[64];

res = httpd_resp_set_type(req, _STREAM_CONTENT_TYPE);
if (res != ESP_OK) {
  return res;
}

while (true) {
  fb = esp_camera_fb_get();
  if (!fb) {
    Serial.println("Camera capture failed");
    res = ESP_FAIL;
  } else {
    if (fb->width > 400) {
      if (fb->format != PIXFORMAT_JPEG) {
        bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
        esp_camera_fb_return(fb);
        fb = NULL;
        if (!jpeg_converted) {
          Serial.println("JPEG compression failed");
          res = ESP_FAIL;
        }
      } else {
        _jpg_buf_len = fb->len;
        _jpg_buf = fb->buf;
      }
    }
  }
  if (res == ESP_OK) {
    size_t hlen = snprintf((char *)part_buf, 64, _STREAM_PART, _jpg_buf_len);
    res = httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
```

```
    }
    if (res == ESP_OK) {
      res = httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len);
    }
    if (res == ESP_OK) {
      res = httpd_resp_send_chunk(req, _STREAM_BOUNDARY,
strlen(_STREAM_BOUNDARY));
    }
    if (fb) {
      esp_camera_fb_return(fb);
      fb = NULL;
      _jpg_buf = NULL;
    } else if (_jpg_buf) {
      free(_jpg_buf);
      _jpg_buf = NULL;
    }
    if (res != ESP_OK) {
      break;
    }
    //Serial.printf("MJPG: %uB\n",(uint32_t)(_jpg_buf_len));
  }
  return res;
}

void checkwifi(){
  bool wifi = false;
  if(WiFi.status() != WL_CONNECTED){
    wifi = false;
    Serial.print("WiFi");Serial.println(wifi);
  }
  else if(WiFi.status() == WL_CONNECTED){
```

```
    wifi = true;
    Serial.print("WiFi");Serial.println(wifi);
  }
}

void startCameraServer() {
  httpd_config_t config = HTTPD_DEFAULT_CONFIG();
  config.server_port = 80;

  httpd_uri_t index_uri = {
    .uri       = "/",
    .method    = HTTP_GET,
    .handler   = stream_handler,
    .user_ctx  = NULL
  };

  if (httpd_start(&stream_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(stream_httpd, &index_uri);
  }
}

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); //disable brownout detector

  Serial.begin(115200);
  Serial.setDebugOutput(false);

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
```

```
config.pin_d1 = Y3_GPIO_NUM;
config.pin_d2 = Y4_GPIO_NUM;
config.pin_d3 = Y5_GPIO_NUM;
config.pin_d4 = Y6_GPIO_NUM;
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sscb_sda = SIOD_GPIO_NUM;
config.pin_sscb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.pixel_format = PIXFORMAT_JPEG;

if (psramFound()) {
  config.frame_size = FRAMESIZE_XGA;
  config.jpeg_quality = 10;
  config.fb_count = 2;
} else {
  config.frame_size = FRAMESIZE_SVGA;
  config.jpeg_quality = 12;
  config.fb_count = 1;
}

// Camera init
esp_err_t err = esp_camera_init(&config);
if (err != ESP_OK) {
```

```
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }
  // Wi-Fi connection
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  Serial.print("Camera Stream Ready! Go to: http://");
  Serial.print(WiFi.localIP());

  if (!SD_MMC.begin()) {
    Serial.println("SD Card Mount Failed");
    return;
  }
  uint8_t cardType = SD_MMC.cardType();
  if (cardType == CARD_NONE) {
    Serial.println("No SD Card attached");
    return;
  }
  // Start streaming web server
  EEPROM.begin(EEPROM_SIZE);
  startCameraServer();
  LINE.setToken(LINE_TOKEN);
}
String str = "";
void loop() {
```

```
while (Serial.available()) {
  char c = Serial.read();
  String readmsg = Serial.readString();
  Serial.println(readmsg);
  checkwifi();
  if (c == '\n') {
    if (str.indexOf("AT+") != -1) {
      Serial.print("str length = ");Serial.println(str.length());
      String filename = str.substring(3, str.length());
      camera_fb_t * fb = NULL;
      fb = esp_camera_fb_get();
      if (!fb) {
        Serial.println("Camera capture failed");
        return;
      }
      pictureNumber = EEPROM.read(0) + 1;
      String path = "/" + String(filename) + ".jpg";
      fs::FS &fs = SD_MMC;
      Serial.printf("FILENAME:%s\n", path.c_str());
      LINE.notifyPicture( MESSAGE_LINE, fb->buf, fb->len);
      LINE.notify(str.substring(31, str.length()));
      File file = fs.open(path.c_str(), FILE_WRITE);
      if (!file) {
        Serial.println("Failed to open file in writing mode");
      }
      else {
        file.write(fb->buf, fb->len); // payload (image), payload length
        //          Serial.print("START#");
        delay(2000);
      }
      esp_camera_fb_return(fb);
```

```
        file.close();
      }


      str = "";
    } else {
      str += c;
    }

  }
}
```

2. ชุดคำสั่งของ ESP32 Node ที่ 1

```
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Wire.h>
#include "painlessMesh.h"
#include <Arduino_JSON.h>
#include <WiFi.h>
#include "DS3231.h"
#include "images1.h"
#include "images2.h"
RTClib RTC;

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

#define RXD2 16
#define TXD2 17
#define PIR_PIN 2
bool _sensor = false;
bool old_sensor = false;

#define bitmap_height   128
#define bitmap_width    64

#define   MESH_PREFIX     "RNTMESH" //name for your MESH
#define   MESH_PASSWORD   "MESHpassword" //password for your MESH
#define   MESH_PORT       5555 //default port

String readings;
```

```
const int nodeNumber = 1;
unsigned long postTime = millis();
int years;
int months;
int days;
int hours;
int minutes;
int seconds;
bool getwifi;
int node1, node2;
bool wifi1,wifi2;

Scheduler userScheduler; // to control your personal task
painlessMesh  mesh;

void sendMessage1() ;
String getReadings1(); // Prototype for sending sensor readings

//Create tasks: to send messages and get readings;
Task taskSendMessage1(TASK_SECOND * 5 , TASK_FOREVER, &sendMessage1);

String getReadings1 () {
  JSONVar jsonReadings;

  readings += String(years);
  readings += String(months);
  readings += String(days);
  readings += String(getwifi);
  //readings += String(hours);
  //readings += String(minutes);
  //readings += String(seconds);
```

```
  jsonReadings["node"] = nodeNumber;
  jsonReadings["years"] = years;
  jsonReadings["months"] = months;
  jsonReadings["days"] = days;
  jsonReadings["wifi"] = getwifi;

  readings = JSON.stringify(jsonReadings);
  return readings;
}

void sendMessage1 () {
  String msg1 = getReadings1();
  mesh.sendBroadcast(msg1);
}

void receivedCallback1( uint32_t from, String &msg2 ) {
  //Serial.printf("Received from %u msg2=%s\n", from, msg2.c_str());
  JSONVar myObject = JSON.parse(msg2.c_str());
  int node = myObject["node"];
  bool wifi = myObject["wifi"];
  if (node == 2) {
    node1 = node;
    wifi1 = wifi;
    Serial.print("Node : ");
    Serial.println(node1);
    Serial.print("WiFi : ");
    Serial.println(wifi1);
  }
}
void receivedCallback2( uint32_t from, String &msg3 ) {
  //Serial.printf("Received from %u msg3=%s\n", from, msg3.c_str());
```

```
  JSONVar myObject = JSON.parse(msg3.c_str());
  int node = myObject["node"];
  bool wifi = myObject["wifi"];
  if (node == 3) {
    node2 = node;
    wifi2 = wifi;
    Serial.print("Node : ");
    Serial.println(node2);
    Serial.print("WiFi : ");
    Serial.println(wifi2);
  }
}

void setup() {
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3D for 128x64
    Serial.println(F("SSD1306 allocation failed"));
    for (;;);
  }
  delay(2000);
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 10);
  display.println("Hello, world!");
  display.display();

  Serial.begin(115200);
  Serial2.begin(115200, SERIAL_8N1, RXD2, TXD2);
  pinMode(PIR_PIN, INPUT);
  _sensor = digitalRead(PIR_PIN);
  old_sensor = _sensor;
```

```
  Wire.begin();

  display.clearDisplay();
  display.drawBitmap(0, 0, bitmap1, bitmap_height, bitmap_width, WHITE);
  display.display();
  mesh.setDebugMsgTypes( ERROR | STARTUP );  // set before init() so that you can
see startup messages

  mesh.init( MESH_PREFIX, MESH_PASSWORD, &userScheduler, MESH_PORT );

  mesh.onReceive(&receivedCallback1);
  mesh.onReceive(&receivedCallback2);
  userScheduler.addTask(taskSendMessage1);
  taskSendMessage1.enable();
}

int counts = 0;
String img = "";
unsigned long postSerail = millis();

void loop() {
  mesh.update();
  if (millis() - postTime >= 1000) {
    DateTime now = RTC.now();
    years = now.year();
    months = now.month();
    days = now.day();
    hours = now.hour();
    minutes = now.minute();
    seconds = now.second();
    Serial.print(years);
```

```
  Serial.print('/');
  Serial.print(months);
  Serial.print('/');
  Serial.print(days);
  Serial.print(' ');
  Serial.print(hours);
  Serial.print(':');
  Serial.print(minutes);
  Serial.print(':');
  Serial.print(seconds);
  Serial.println();
  postTime = millis();
}
_sensor = digitalRead(PIR_PIN);
if (!_sensor && old_sensor) {
  String strsend = "AT+";
  strsend += String(years) + "-";
  strsend += String(months) + "-";
  strsend += String(days) + " ";
  strsend += String(hours) + "-";
  strsend += String(minutes) + "-";
  strsend += String(seconds)  + "-";
  strsend += "node:" + String(node1) + "-";
  strsend += "wifi:" + String(wifi1) + "-";
  strsend += "node:" + String(node2) + "-";
  strsend += "wifi:" + String(wifi2);
  Serial.println(strsend);
  Serial2.println(strsend);
  display.clearDisplay();
  display.drawBitmap(0, 0, bitmap2, bitmap_height, bitmap_width, WHITE);
  display.display();
```

```
  delay(200);
  display.clearDisplay();
  display.drawBitmap(0, 0, bitmap1, bitmap_height, bitmap_width, WHITE);
  display.display();
  delay(200);
  display.clearDisplay();
  display.drawBitmap(0, 0, bitmap2, bitmap_height, bitmap_width, WHITE);
  display.display();
  delay(200);
  display.clearDisplay();
  display.drawBitmap(0, 0, bitmap1, bitmap_height, bitmap_width, WHITE);
  display.display();
  delay(200);
  display.clearDisplay();
  display.drawBitmap(0, 0, bitmap2, bitmap_height, bitmap_width, WHITE);
  display.display();
  delay(200);
  display.clearDisplay();
  display.drawBitmap(0, 0, bitmap1, bitmap_height, bitmap_width, WHITE);
  display.display();
  delay(200);
}
old_sensor = _sensor;

if (Serial2.available()) {
  char c = Serial2.read();
  uint8_t c1 = c;
  img += c;
  String msg = Serial2.readString();
  Serial.println(msg);
  if (msg == "WiFi1") {
```

```
      getwifi = true;
    }
    else if (msg == "WiFi0") {
      getwifi = false;
    }
    postSerail = millis();
  }

  if (millis() - postSerail >= 2000) {
    int len = img.length();
    if (len > 0) {
      Serial.print("img.length() = ");
      Serial.println(len);
      // img
      img = "";
    }
  }
}
```

ภาคผนวก ข

โปรแกรมคำสั่งของ Node 2

1. ชุดคำสั่งในส่วนของ ESP32 CAM

```
#include "esp_camera.h"
#include <WiFi.h>
#include <TridentTD_LineNotify.h>
#include "esp_timer.h"
#include "img_converters.h"
#include "Arduino.h"
#include "fb_gfx.h"
#include "FS.h"
#include "SD_MMC.h"
#include "soc/soc.h" //disable brownout problems
#include "soc/rtc_cntl_reg.h"  //disable brownout problems
#include "driver/rtc_io.h"
//#include "dl_lib.h"
#include "esp_http_server.h"
#include <EEPROM.h>
#define EEPROM_SIZE 1
int pictureNumber = 0;

#define LINE_TOKEN  "**********************" // SET LINE TOKEN
String MESSAGE_LINE = “Node2 Captured!!!”;
//Replace with your network credentials
const char* ssid = "**********"; // แก้ชื่อ wifi
const char* password = "********"; // แก้รหัสผ่าน

#define PART_BOUNDARY "123456789000000000000987654321"

// This project was tested with the AI Thinker Model, M5STACK PSRAM Model and
M5STACK WITHOUT PSRAM
#define CAMERA_MODEL_AI_THINKER
```

```
#if defined(CAMERA_MODEL_WROVER_KIT)
#define PWDN_GPIO_NUM    -1
#define RESET_GPIO_NUM   -1
#define XCLK_GPIO_NUM    21
#define SIOD_GPIO_NUM    26
#define SIOC_GPIO_NUM    27

#define Y9_GPIO_NUM      35
#define Y8_GPIO_NUM      34
#define Y7_GPIO_NUM      39
#define Y6_GPIO_NUM      36
#define Y5_GPIO_NUM      19
#define Y4_GPIO_NUM      18
#define Y3_GPIO_NUM       5
#define Y2_GPIO_NUM       4
#define VSYNC_GPIO_NUM   25
#define HREF_GPIO_NUM    23
#define PCLK_GPIO_NUM    22

#elif defined(CAMERA_MODEL_M5STACK_PSRAM)
#define PWDN_GPIO_NUM     -1
#define RESET_GPIO_NUM    15
#define XCLK_GPIO_NUM     27
#define SIOD_GPIO_NUM     25
#define SIOC_GPIO_NUM     23

#define Y9_GPIO_NUM       19
#define Y8_GPIO_NUM       36
#define Y7_GPIO_NUM       18
#define Y6_GPIO_NUM       39
```

```
#define Y5_GPIO_NUM        5
#define Y4_GPIO_NUM       34
#define Y3_GPIO_NUM       35
#define Y2_GPIO_NUM       32
#define VSYNC_GPIO_NUM    22
#define HREF_GPIO_NUM     26
#define PCLK_GPIO_NUM     21

#elif defined(CAMERA_MODEL_M5STACK_WITHOUT_PSRAM)
#define PWDN_GPIO_NUM     -1
#define RESET_GPIO_NUM    15
#define XCLK_GPIO_NUM     27
#define SIOD_GPIO_NUM     25
#define SIOC_GPIO_NUM     23

#define Y9_GPIO_NUM       19
#define Y8_GPIO_NUM       36
#define Y7_GPIO_NUM       18
#define Y6_GPIO_NUM       39
#define Y5_GPIO_NUM        5
#define Y4_GPIO_NUM       34
#define Y3_GPIO_NUM       35
#define Y2_GPIO_NUM       17
#define VSYNC_GPIO_NUM    22
#define HREF_GPIO_NUM     26
#define PCLK_GPIO_NUM     21

#elif defined(CAMERA_MODEL_AI_THINKER)
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
```

```
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27

#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM       5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22
#else
#error "Camera model not selected"
#endif

static const char* _STREAM_CONTENT_TYPE = "multipart/x-mixed-replace;boundary="
PART_BOUNDARY;
static const char* _STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY "\r\n";
static const char* _STREAM_PART = "Content-Type: image/jpeg\r\nContent-Length:
%u\r\n\r\n";

httpd_handle_t stream_httpd = NULL;

static esp_err_t stream_handler(httpd_req_t *req) {
  camera_fb_t * fb = NULL;
  esp_err_t res = ESP_OK;
  size_t _jpg_buf_len = 0;
  uint8_t * _jpg_buf = NULL;
```

```
char * part_buf[64];

res = httpd_resp_set_type(req, _STREAM_CONTENT_TYPE);
if (res != ESP_OK) {
  return res;
}

while (true) {
  fb = esp_camera_fb_get();
  if (!fb) {
    Serial.println("Camera capture failed");
    res = ESP_FAIL;
  } else {
    if (fb->width > 400) {
      if (fb->format != PIXFORMAT_JPEG) {
        bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
        esp_camera_fb_return(fb);
        fb = NULL;
        if (!jpeg_converted) {
          Serial.println("JPEG compression failed");
          res = ESP_FAIL;
        }
      } else {
        _jpg_buf_len = fb->len;
        _jpg_buf = fb->buf;
      }
    }
  }
  if (res == ESP_OK) {
    size_t hlen = snprintf((char *)part_buf, 64, _STREAM_PART, _jpg_buf_len);
    res = httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
```

```
    }
    if (res == ESP_OK) {
      res = httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len);
    }
    if (res == ESP_OK) {
      res = httpd_resp_send_chunk(req, _STREAM_BOUNDARY,
strlen(_STREAM_BOUNDARY));
    }
    if (fb) {
      esp_camera_fb_return(fb);
      fb = NULL;
      _jpg_buf = NULL;
    } else if (_jpg_buf) {
      free(_jpg_buf);
      _jpg_buf = NULL;
    }
    if (res != ESP_OK) {
      break;
    }
    //Serial.printf("MJPG: %uB\n",(uint32_t)(_jpg_buf_len));
  }
  return res;
}

void checkwifi(){
  bool wifi = false;
  if(WiFi.status() != WL_CONNECTED){
    wifi = false;
    Serial.print("WiFi");Serial.println(wifi);
  }
  else if(WiFi.status() == WL_CONNECTED){
```

```
    wifi = true;
    Serial.print("WiFi");Serial.println(wifi);
  }
}

void startCameraServer() {
  httpd_config_t config = HTTPD_DEFAULT_CONFIG();
  config.server_port = 80;

  httpd_uri_t index_uri = {
    .uri       = "/",
    .method    = HTTP_GET,
    .handler   = stream_handler,
    .user_ctx  = NULL
  };

  if (httpd_start(&stream_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(stream_httpd, &index_uri);
  }
}

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); //disable brownout detector

  Serial.begin(115200);
  Serial.setDebugOutput(false);

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
```

```
config.pin_d1 = Y3_GPIO_NUM;
config.pin_d2 = Y4_GPIO_NUM;
config.pin_d3 = Y5_GPIO_NUM;
config.pin_d4 = Y6_GPIO_NUM;
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sscb_sda = SIOD_GPIO_NUM;
config.pin_sscb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.pixel_format = PIXFORMAT_JPEG;

if (psramFound()) {
  config.frame_size = FRAMESIZE_XGA;
  config.jpeg_quality = 10;
  config.fb_count = 2;
} else {
  config.frame_size = FRAMESIZE_SVGA;
  config.jpeg_quality = 12;
  config.fb_count = 1;
}

// Camera init
esp_err_t err = esp_camera_init(&config);
if (err != ESP_OK) {
```

```
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }
  // Wi-Fi connection
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  Serial.print("Camera Stream Ready! Go to: http://");
  Serial.print(WiFi.localIP());

  if (!SD_MMC.begin()) {
    Serial.println("SD Card Mount Failed");
    return;
  }
  uint8_t cardType = SD_MMC.cardType();
  if (cardType == CARD_NONE) {
    Serial.println("No SD Card attached");
    return;
  }
  // Start streaming web server
  EEPROM.begin(EEPROM_SIZE);
  startCameraServer();
  LINE.setToken(LINE_TOKEN);
}
String str = "";
void loop() {
```

```
while (Serial.available()) {
  char c = Serial.read();
  String readmsg = Serial.readString();
  Serial.println(readmsg);
  checkwifi();
  if (c == '\n') {
    if (str.indexOf("AT+") != -1) {
      Serial.print("str length = ");Serial.println(str.length());
      String filename = str.substring(3, str.length());
      camera_fb_t * fb = NULL;
      fb = esp_camera_fb_get();
      if (!fb) {
        Serial.println("Camera capture failed");
        return;
      }
      pictureNumber = EEPROM.read(0) + 1;
      String path = "/" + String(filename) + ".jpg";
      fs::FS &fs = SD_MMC;
      Serial.printf("FILENAME:%s\n", path.c_str());
      LINE.notifyPicture( MESSAGE_LINE, fb->buf, fb->len);
      LINE.notify(str.substring(31, str.length()));
      File file = fs.open(path.c_str(), FILE_WRITE);
      if (!file) {
        Serial.println("Failed to open file in writing mode");
      }
      else {
        file.write(fb->buf, fb->len); // payload (image), payload length
        //          Serial.print("START#");
        delay(2000);
      }
      esp_camera_fb_return(fb);
```

```
        file.close();
      }


      str = "";
    } else {
      str += c;
    }

  }
}
```

2. ชุดคำสั่งของ ESP32 Node ที่ 2

```
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Wire.h>
#include "painlessMesh.h"
#include <Arduino_JSON.h>
#include <WiFi.h>
#include "DS3231.h"
#include "images1.h"
#include "images2.h"
RTClib RTC;

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

#define RXD2 16
#define TXD2 17
#define PIR_PIN 2
bool _sensor = false;
bool old_sensor = false;

#define bitmap_height   128
#define bitmap_width    64

#define   MESH_PREFIX     "RNTMESH" //name for your MESH
#define   MESH_PASSWORD   "MESHpassword" //password for your MESH
#define   MESH_PORT       5555 //default port

String readings;
```

```
const int nodeNumber = 2;
unsigned long postTime = millis();
int years;
int months;
int days;
int hours;
int minutes;
int seconds;
bool getwifi;
int node1, node2;
bool wifi1,wifi2;

Scheduler userScheduler; // to control your personal task
painlessMesh  mesh;

void sendMessage2() ;
String getReadings2(); // Prototype for sending sensor readings

//Create tasks: to send messages and get readings;
Task taskSendMessage2(TASK_SECOND * 10 , TASK_FOREVER, &sendMessage2);

String getReadings2 () {
  JSONVar jsonReadings;

  readings += String(years);
  readings += String(months);
  readings += String(days);
  readings += String(getwifi);
  //readings += String(hours);
  //readings += String(minutes);
  //readings += String(seconds);
```

```
  jsonReadings["node"] = nodeNumber;
  jsonReadings["years"] = years;
  jsonReadings["months"] = months;
  jsonReadings["days"] = days;
  jsonReadings["wifi"] = getwifi;

  readings = JSON.stringify(jsonReadings);
  return readings;
}

void sendMessage2 () {
  String msg2 = getReadings2();
  mesh.sendBroadcast(msg2);
}

void receivedCallback1( uint32_t from, String &msg1 ) {
  //Serial.printf("Received from %u msg1=%s\n", from, msg1.c_str());
  JSONVar myObject = JSON.parse(msg1.c_str());
  int node = myObject["node"];
  bool wifi = myObject["wifi"];
  if (node == 1) {
    node1 = node;
    wifi1 = wifi;
    Serial.print("Node : ");
    Serial.println(node1);
    Serial.print("WiFi : ");
    Serial.println(wifi1);
  }
}
void receivedCallback2( uint32_t from, String &msg3 ) {
  //Serial.printf("Received from %u msg3=%s\n", from, msg3.c_str());
```

```
  JSONVar myObject = JSON.parse(msg3.c_str());
  int node = myObject["node"];
  bool wifi = myObject["wifi"];
  if (node == 3) {
    node2 = node;
    wifi2 = wifi;
    Serial.print("Node : ");
    Serial.println(node2);
    Serial.print("WiFi : ");
    Serial.println(wifi2);
  }
}

void setup() {
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3D for 128x64
    Serial.println(F("SSD1306 allocation failed"));
    for (;;);
  }
  delay(2000);
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 10);
  display.println("Hello, world!");
  display.display();

  Serial.begin(115200);
  Serial2.begin(115200, SERIAL_8N1, RXD2, TXD2);
  pinMode(PIR_PIN, INPUT);
  _sensor = digitalRead(PIR_PIN);
  old_sensor = _sensor;
```

```
  Wire.begin();

  display.clearDisplay();
  display.drawBitmap(0, 0, bitmap1, bitmap_height, bitmap_width, WHITE);
  display.display();
  mesh.setDebugMsgTypes( ERROR | STARTUP );  // set before init() so that you can
see startup messages

  mesh.init( MESH_PREFIX, MESH_PASSWORD, &userScheduler, MESH_PORT );

  mesh.onReceive(&receivedCallback1);
  mesh.onReceive(&receivedCallback2);
  userScheduler.addTask(taskSendMessage2);
  taskSendMessage2.enable();
}

int counts = 0;
String img = "";
unsigned long postSerail = millis();

void loop() {
  mesh.update();
  if (millis() - postTime >= 1000) {
    DateTime now = RTC.now();
    years = now.year();
    months = now.month();
    days = now.day();
    hours = now.hour();
    minutes = now.minute();
    seconds = now.second();
    Serial.print(years);
```

```
  Serial.print('/');
  Serial.print(months);
  Serial.print('/');
  Serial.print(days);
  Serial.print(' ');
  Serial.print(hours);
  Serial.print(':');
  Serial.print(minutes);
  Serial.print(':');
  Serial.print(seconds);
  Serial.println();
  postTime = millis();
}
_sensor = digitalRead(PIR_PIN);
if (!_sensor && old_sensor) {
  String strsend = "AT+";
  strsend += String(years) + "-";
  strsend += String(months) + "-";
  strsend += String(days) + " ";
  strsend += String(hours) + "-";
  strsend += String(minutes) + "-";
  strsend += String(seconds)  + "-";
  strsend += "node:" + String(node1) + "-";
  strsend += "wifi:" + String(wifi1) + "-";
  strsend += "node:" + String(node2) + "-";
  strsend += "wifi:" + String(wifi2);
  Serial.println(strsend);
  Serial2.println(strsend);
  display.clearDisplay();
  display.drawBitmap(0, 0, bitmap2, bitmap_height, bitmap_width, WHITE);
  display.display();
```

```
  delay(200);
  display.clearDisplay();
  display.drawBitmap(0, 0, bitmap1, bitmap_height, bitmap_width, WHITE);
  display.display();
  delay(200);
  display.clearDisplay();
  display.drawBitmap(0, 0, bitmap2, bitmap_height, bitmap_width, WHITE);
  display.display();
  delay(200);
  display.clearDisplay();
  display.drawBitmap(0, 0, bitmap1, bitmap_height, bitmap_width, WHITE);
  display.display();
  delay(200);
  display.clearDisplay();
  display.drawBitmap(0, 0, bitmap2, bitmap_height, bitmap_width, WHITE);
  display.display();
  delay(200);
  display.clearDisplay();
  display.drawBitmap(0, 0, bitmap1, bitmap_height, bitmap_width, WHITE);
  display.display();
  delay(200);
}
old_sensor = _sensor;

if (Serial2.available()) {
  char c = Serial2.read();
  uint8_t c1 = c;
  img += c;
  String msg = Serial2.readString();
  Serial.println(msg);
  if (msg == "WiFi1") {
```

```
      getwifi = true;
    }
    else if (msg == "WiFi0") {
      getwifi = false;
    }
    postSerail = millis();
  }

  if (millis() - postSerail >= 2000) {
    int len = img.length();
    if (len > 0) {
      Serial.print("img.length() = ");
      Serial.println(len);
      // img
      img = "";
    }
  }
}
```

ภาคผนวก ค

โปรแกรมคำสั่งของ Node 3

1. ชุดคำสั่งในส่วนของ ESP32 CAM

```
#include "esp_camera.h"
#include <WiFi.h>
#include <TridentTD_LineNotify.h>
#include "esp_timer.h"
#include "img_converters.h"
#include "Arduino.h"
#include "fb_gfx.h"
#include "FS.h"
#include "SD_MMC.h"
#include "soc/soc.h" //disable brownout problems
#include "soc/rtc_cntl_reg.h"  //disable brownout problems
#include "driver/rtc_io.h"
//#include "dl_lib.h"
#include "esp_http_server.h"
#include <EEPROM.h>
#define EEPROM_SIZE 1
int pictureNumber = 0;

#define LINE_TOKEN  "**********************" // SET LINE TOKEN
String MESSAGE_LINE = “Node1 Captured!!!”;
//Replace with your network credentials
const char* ssid = "**********"; // แก้ชื่อ wifi
const char* password = "********"; // แก้รหัสผ่าน

#define PART_BOUNDARY "123456789000000000000987654321"

// This project was tested with the AI Thinker Model, M5STACK PSRAM Model and
M5STACK WITHOUT PSRAM
#define CAMERA_MODEL_AI_THINKER
```

```
#if defined(CAMERA_MODEL_WROVER_KIT)
#define PWDN_GPIO_NUM    -1
#define RESET_GPIO_NUM   -1
#define XCLK_GPIO_NUM    21
#define SIOD_GPIO_NUM    26
#define SIOC_GPIO_NUM    27

#define Y9_GPIO_NUM      35
#define Y8_GPIO_NUM      34
#define Y7_GPIO_NUM      39
#define Y6_GPIO_NUM      36
#define Y5_GPIO_NUM      19
#define Y4_GPIO_NUM      18
#define Y3_GPIO_NUM       5
#define Y2_GPIO_NUM       4
#define VSYNC_GPIO_NUM   25
#define HREF_GPIO_NUM    23
#define PCLK_GPIO_NUM    22

#elif defined(CAMERA_MODEL_M5STACK_PSRAM)
#define PWDN_GPIO_NUM     -1
#define RESET_GPIO_NUM    15
#define XCLK_GPIO_NUM     27
#define SIOD_GPIO_NUM     25
#define SIOC_GPIO_NUM     23

#define Y9_GPIO_NUM       19
#define Y8_GPIO_NUM       36
#define Y7_GPIO_NUM       18
#define Y6_GPIO_NUM       39
```

```
#define Y5_GPIO_NUM         5
#define Y4_GPIO_NUM        34
#define Y3_GPIO_NUM        35
#define Y2_GPIO_NUM        32
#define VSYNC_GPIO_NUM     22
#define HREF_GPIO_NUM      26
#define PCLK_GPIO_NUM      21

#elif defined(CAMERA_MODEL_M5STACK_WITHOUT_PSRAM)
#define PWDN_GPIO_NUM      -1
#define RESET_GPIO_NUM     15
#define XCLK_GPIO_NUM      27
#define SIOD_GPIO_NUM      25
#define SIOC_GPIO_NUM      23

#define Y9_GPIO_NUM        19
#define Y8_GPIO_NUM        36
#define Y7_GPIO_NUM        18
#define Y6_GPIO_NUM        39
#define Y5_GPIO_NUM         5
#define Y4_GPIO_NUM        34
#define Y3_GPIO_NUM        35
#define Y2_GPIO_NUM        17
#define VSYNC_GPIO_NUM     22
#define HREF_GPIO_NUM      26
#define PCLK_GPIO_NUM      21

#elif defined(CAMERA_MODEL_AI_THINKER)
#define PWDN_GPIO_NUM      32
#define RESET_GPIO_NUM     -1
#define XCLK_GPIO_NUM       0
```

```
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27

#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM       5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22
#else
#error "Camera model not selected"
#endif

static const char* _STREAM_CONTENT_TYPE = "multipart/x-mixed-replace;boundary="
PART_BOUNDARY;
static const char* _STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY "\r\n";
static const char* _STREAM_PART = "Content-Type: image/jpeg\r\nContent-Length:
%u\r\n\r\n";

httpd_handle_t stream_httpd = NULL;

static esp_err_t stream_handler(httpd_req_t *req) {
  camera_fb_t * fb = NULL;
  esp_err_t res = ESP_OK;
  size_t _jpg_buf_len = 0;
  uint8_t * _jpg_buf = NULL;
```

```
char * part_buf[64];

res = httpd_resp_set_type(req, _STREAM_CONTENT_TYPE);
if (res != ESP_OK) {
  return res;
}

while (true) {
  fb = esp_camera_fb_get();
  if (!fb) {
    Serial.println("Camera capture failed");
    res = ESP_FAIL;
  } else {
    if (fb->width > 400) {
      if (fb->format != PIXFORMAT_JPEG) {
        bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
        esp_camera_fb_return(fb);
        fb = NULL;
        if (!jpeg_converted) {
          Serial.println("JPEG compression failed");
          res = ESP_FAIL;
        }
      } else {
        _jpg_buf_len = fb->len;
        _jpg_buf = fb->buf;
      }
    }
  }
  if (res == ESP_OK) {
    size_t hlen = snprintf((char *)part_buf, 64, _STREAM_PART, _jpg_buf_len);
    res = httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
```

```
    }
    if (res == ESP_OK) {
      res = httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len);
    }
    if (res == ESP_OK) {
      res = httpd_resp_send_chunk(req, _STREAM_BOUNDARY,
strlen(_STREAM_BOUNDARY));
    }
    if (fb) {
      esp_camera_fb_return(fb);
      fb = NULL;
      _jpg_buf = NULL;
    } else if (_jpg_buf) {
      free(_jpg_buf);
      _jpg_buf = NULL;
    }
    if (res != ESP_OK) {
      break;
    }
    //Serial.printf("MJPG: %uB\n",(uint32_t)(_jpg_buf_len));
  }
  return res;
}

void checkwifi(){
  bool wifi = false;
  if(WiFi.status() != WL_CONNECTED){
    wifi = false;
    Serial.print("WiFi");Serial.println(wifi);
  }
  else if(WiFi.status() == WL_CONNECTED){
```

```
    wifi = true;
    Serial.print("WiFi");Serial.println(wifi);
  }
}

void startCameraServer() {
  httpd_config_t config = HTTPD_DEFAULT_CONFIG();
  config.server_port = 80;

  httpd_uri_t index_uri = {
    .uri       = "/",
    .method    = HTTP_GET,
    .handler   = stream_handler,
    .user_ctx  = NULL
  };

  if (httpd_start(&stream_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(stream_httpd, &index_uri);
  }
}

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); //disable brownout detector

  Serial.begin(115200);
  Serial.setDebugOutput(false);

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
```

```
config.pin_d1 = Y3_GPIO_NUM;
config.pin_d2 = Y4_GPIO_NUM;
config.pin_d3 = Y5_GPIO_NUM;
config.pin_d4 = Y6_GPIO_NUM;
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sscb_sda = SIOD_GPIO_NUM;
config.pin_sscb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.pixel_format = PIXFORMAT_JPEG;

if (psramFound()) {
  config.frame_size = FRAMESIZE_XGA;
  config.jpeg_quality = 10;
  config.fb_count = 2;
} else {
  config.frame_size = FRAMESIZE_SVGA;
  config.jpeg_quality = 12;
  config.fb_count = 1;
}

// Camera init
esp_err_t err = esp_camera_init(&config);
if (err != ESP_OK) {
```

```
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }
  // Wi-Fi connection
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  Serial.print("Camera Stream Ready! Go to: http://");
  Serial.print(WiFi.localIP());

  if (!SD_MMC.begin()) {
    Serial.println("SD Card Mount Failed");
    return;
  }
  uint8_t cardType = SD_MMC.cardType();
  if (cardType == CARD_NONE) {
    Serial.println("No SD Card attached");
    return;
  }
  // Start streaming web server
  EEPROM.begin(EEPROM_SIZE);
  startCameraServer();
  LINE.setToken(LINE_TOKEN);
}
String str = "";
void loop() {
```

```
while (Serial.available()) {
  char c = Serial.read();
  String readmsg = Serial.readString();
  Serial.println(readmsg);
  checkwifi();
  if (c == '\n') {
    if (str.indexOf("AT+") != -1) {
      Serial.print("str length = ");Serial.println(str.length());
      String filename = str.substring(3, str.length());
      camera_fb_t * fb = NULL;
      fb = esp_camera_fb_get();
      if (!fb) {
        Serial.println("Camera capture failed");
        return;
      }
      pictureNumber = EEPROM.read(0) + 1;
      String path = "/" + String(filename) + ".jpg";
      fs::FS &fs = SD_MMC;
      Serial.printf("FILENAME:%s\n", path.c_str());
      LINE.notifyPicture( MESSAGE_LINE, fb->buf, fb->len);
      LINE.notify(str.substring(31, str.length()));
      File file = fs.open(path.c_str(), FILE_WRITE);
      if (!file) {
        Serial.println("Failed to open file in writing mode");
      }
      else {
        file.write(fb->buf, fb->len); // payload (image), payload length
        //          Serial.print("START#");
        delay(2000);
      }
      esp_camera_fb_return(fb);
```

```
        file.close();
      }


      str = "";
    } else {
      str += c;
    }

  }
}
```

2. ชุดคำสั่งของ ESP32 Node ที่ 3

```
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Wire.h>
#include "painlessMesh.h"
#include <Arduino_JSON.h>
#include <WiFi.h>
#include "DS3231.h"
#include "images1.h"
#include "images2.h"
RTClib RTC;

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

#define RXD2 16
#define TXD2 17
#define PIR_PIN 2
bool _sensor = false;
bool old_sensor = false;

#define bitmap_height   128
#define bitmap_width    64

#define   MESH_PREFIX     "RNTMESH" //name for your MESH
#define   MESH_PASSWORD   "MESHpassword" //password for your MESH
#define   MESH_PORT       5555 //default port

String readings;
```

```
const int nodeNumber = 3;
unsigned long postTime = millis();
int years;
int months;
int days;
int hours;
int minutes;
int seconds;
bool getwifi;
int node1, node2;
bool wifi1,wifi2;

Scheduler userScheduler; // to control your personal task
painlessMesh  mesh;

void sendMessage3() ;
String getReadings3(); // Prototype for sending sensor readings

//Create tasks: to send messages and get readings;
Task taskSendMessage3(TASK_SECOND * 15 , TASK_FOREVER, &sendMessage3);

String getReadings3 () {
  JSONVar jsonReadings;

  readings += String(years);
  readings += String(months);
  readings += String(days);
  readings += String(getwifi);
  //readings += String(hours);
  //readings += String(minutes);
  //readings += String(seconds);
```

```
  jsonReadings["node"] = nodeNumber;
  jsonReadings["years"] = years;
  jsonReadings["months"] = months;
  jsonReadings["days"] = days;
  jsonReadings["wifi"] = getwifi;

  readings = JSON.stringify(jsonReadings);
  return readings;
}

void sendMessage3 () {
  String msg3 = getReadings3();
  mesh.sendBroadcast(msg3);
}

void receivedCallback1( uint32_t from, String &msg2 ) {
  Serial.printf("Received from %u msg2=%s\n", from, msg2.c_str());
  JSONVar myObject = JSON.parse(msg2.c_str());
  int node = myObject["node"];
  bool wifi = myObject["wifi"];
  if (node == 2) {
    node1 = node;
    wifi1 = wifi;
    Serial.print("Node : ");
    Serial.println(node1);
    Serial.print("WiFi : ");
    Serial.println(wifi1);
  }
}
void receivedCallback2( uint32_t from, String &msg1 ) {
  Serial.printf("Received from %u msg1=%s\n", from, msg1.c_str());
```

```
  JSONVar myObject = JSON.parse(msg1.c_str());
  int node = myObject["node"];
  bool wifi = myObject["wifi"];
  if (node == 1) {
    node2 = node;
    wifi2 = wifi;
    Serial.print("Node : ");
    Serial.println(node2);
    Serial.print("WiFi : ");
    Serial.println(wifi2);
  }
}

void setup() {
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3D for 128x64
    Serial.println(F("SSD1306 allocation failed"));
    for (;;);
  }
  delay(2000);
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 10);
  display.println("Hello, world!");
  display.display();

  Serial.begin(115200);
  Serial2.begin(115200, SERIAL_8N1, RXD2, TXD2);
  pinMode(PIR_PIN, INPUT);
  _sensor = digitalRead(PIR_PIN);
  old_sensor = _sensor;
```

```
  Wire.begin();

  display.clearDisplay();
  display.drawBitmap(0, 0, bitmap1, bitmap_height, bitmap_width, WHITE);
  display.display();
  mesh.setDebugMsgTypes( ERROR | STARTUP );  // set before init() so that you can
see startup messages

  mesh.init( MESH_PREFIX, MESH_PASSWORD, &userScheduler, MESH_PORT );

  mesh.onReceive(&receivedCallback1);
  mesh.onReceive(&receivedCallback2);
  userScheduler.addTask(taskSendMessage3);
  taskSendMessage3.enable();
}

int counts = 0;
String img = "";
unsigned long postSerail = millis();

void loop() {
  mesh.update();
  if (millis() - postTime >= 1000) {
    DateTime now = RTC.now();
    years = now.year();
    months = now.month();
    days = now.day();
    hours = now.hour();
    minutes = now.minute();
    seconds = now.second();
    Serial.print(years);
```

```
  Serial.print('/');
  Serial.print(months);
  Serial.print('/');
  Serial.print(days);
  Serial.print(' ');
  Serial.print(hours);
  Serial.print(':');
  Serial.print(minutes);
  Serial.print(':');
  Serial.print(seconds);
  Serial.println();
  postTime = millis();
}
_sensor = digitalRead(PIR_PIN);
if (!_sensor && old_sensor) {
  String strsend = "AT+";
  strsend += String(years) + "-";
  strsend += String(months) + "-";
  strsend += String(days) + " ";
  strsend += String(hours) + "-";
  strsend += String(minutes) + "-";
  strsend += String(seconds)  + "-";
  strsend += "node:" + String(node1) + "-";
  strsend += "wifi:" + String(wifi1) + "-";
  strsend += "node:" + String(node2) + "-";
  strsend += "wifi:" + String(wifi2);
  Serial.println(strsend);
  Serial2.println(strsend);
  display.clearDisplay();
  display.drawBitmap(0, 0, bitmap2, bitmap_height, bitmap_width, WHITE);
  display.display();
```

```
  delay(200);
  display.clearDisplay();
  display.drawBitmap(0, 0, bitmap1, bitmap_height, bitmap_width, WHITE);
  display.display();
  delay(200);
  display.clearDisplay();
  display.drawBitmap(0, 0, bitmap2, bitmap_height, bitmap_width, WHITE);
  display.display();
  delay(200);
  display.clearDisplay();
  display.drawBitmap(0, 0, bitmap1, bitmap_height, bitmap_width, WHITE);
  display.display();
  delay(200);
  display.clearDisplay();
  display.drawBitmap(0, 0, bitmap2, bitmap_height, bitmap_width, WHITE);
  display.display();
  delay(200);
  display.clearDisplay();
  display.drawBitmap(0, 0, bitmap1, bitmap_height, bitmap_width, WHITE);
  display.display();
  delay(200);
}
old_sensor = _sensor;

if (Serial2.available()) {
  char c = Serial2.read();
  uint8_t c1 = c;
  img += c;
  String msg = Serial2.readString();
  Serial.println(msg);
  if (msg == "WiFi1") {
```

```
      getwifi = true;
    }
    else if (msg == "WiFi0") {
      getwifi = false;
    }
    postSerail = millis();
  }

  if (millis() - postSerail >= 2000) {
    int len = img.length();
    if (len > 0) {
      Serial.print("img.length() = ");
      Serial.println(len);
      // img
      img = "";
    }
  }
}
```